



**University of Central Florida**  
Department of Electrical Engineering and Computer Science

**Smart Package Lockbox**  
**Senior Design 2**  
**Group 5**

Dominic Amalfitano | Computer Engineering | [Amalfid@knights.ucf.edu](mailto:Amalfid@knights.ucf.edu)  
Daniel Corredor | Computer Engineering | [Dnlcorredor@gmail.com](mailto:Dnlcorredor@gmail.com)  
Brandon Dziewior | Electrical Engineering | [Bsdziewior@gmail.com](mailto:Bsdziewior@gmail.com)  
Michael Light | Electrical Engineering | [Mlight@knights.ucf.edu](mailto:Mlight@knights.ucf.edu)

## Table of Contents

1.0 Executive Summary .....	1
2.0 Project Description.....	2
2.1 Motivation.....	3
2.2 Goals and Objectives .....	4
2.3 Requirements & Specifications.....	4
2.3.1 Marketing Requirements.....	4
2.3.2 Engineering Requirement Specifications.....	5
2.3.2.1 Enclosure.....	5
2.3.2.2 Power .....	6
2.3.2.3 System Security .....	7
2.3.2.4 Wireless Communication.....	8
2.3.2.5 Cost .....	8
2.3.2.6 Software .....	9
2.4 House of Quality Analysis .....	9
2.5 Initial Project Flow .....	10
3.0 Related Standards and Design Constraints .....	12
3.1 Standards.....	12
3.1.1 Standards Impact on Design .....	12
3.2 Design Constraints .....	13
3.2.1 Economic Constraints .....	13
3.2.2 Time Constraints.....	13
3.2.3 Environmental Constraints.....	14
3.2.4 Social Constraints .....	14
3.2.5 Ethical Constraints.....	14
3.2.6 Health & Safety.....	15
3.2.7 Manufacturability.....	15
3.2.8 Sustainability.....	15
4.0 Research .....	16
4.1 Existing Products .....	16
4.2 Hardware.....	18
4.2.1 Lid Sensors.....	18

4.2.1.1 Force Sensor.....	18
4.2.1.2 Motion Sensor.....	20
4.2.1.3 Magnet Sensor .....	22
4.2.2 System Movement Detection.....	23
4.2.2.1 Tilt Switch.....	23
4.2.2.2 Sonar Rangefinder .....	24
4.2.2.3 Inertial Measurement Unit .....	25
4.2.3 Alarm .....	26
4.2.4.1 Wi-Fi.....	27
4.2.5 Keying Mechanism .....	28
4.2.6 Locking Mechanism.....	29
4.2.7 Power .....	30
4.2.7.1 AC/DC Conversion.....	30
4.2.7.2 Battery.....	30
4.2.8 Enclosure.....	31
4.2.9 Microcontroller .....	32
4.2.9.1 MSP430.....	33
4.2.9.2 ATmega2560.....	34
4.2.9.3 ATmega328p.....	35
4.3 Software .....	37
4.3.1 Microcontroller .....	38
4.3.1.1 AVRDUDE Command Line Program .....	38
4.3.1.2 Arduino IDE.....	39
4.3.2 Web Communications & Hosting.....	39
4.3.2.1 Web Communication .....	39
4.3.2.2 Hosting Options .....	40
4.3.3 Mobile Device Application.....	42
4.3.3.1 Choosing a Development Platform.....	42
4.3.3.2 Android OS Platform & IDE .....	43
4.3.3.3 Apple iOS Platform & IDE.....	44
4.3.3.4 Cross-Platform Development & IDE.....	46
5.0 Part Selection and Acquisition Summary .....	48

5.2 Part Selection .....	48
5.2.1 Lid Sensor Selection .....	48
5.2.2 Inertial Measurement Unit Selection .....	49
5.2.3 Alarm Selection .....	49
5.2.4 Wi-Fi Module Selection.....	50
5.2.5 Keying Mechanism Part Selection.....	51
5.2.6 Locking Mechanism Part Selection .....	52
5.2.7 Power Related Component Selection.....	52
5.2.7.1 AC/DC Adapter .....	53
5.2.7.2 Battery.....	54
5.2.8 Enclosure Selection.....	54
5.2.9 Microcontroller Selection .....	55
5.3 Part Acquisition .....	56
5.3.1 Amazon .....	56
5.3.2 Digi-Key .....	56
5.3.3 Adafruit .....	56
6.0 Hardware Design .....	56
6.1 Initial Hardware Design.....	57
6.1.1 Power .....	58
6.1.2 Microcontroller .....	62
6.1.3 Sensors .....	63
6.1.3.1 Lid Sensor .....	64
6.1.3.2 Inertial Measurement Unit .....	64
6.1.3.3 Power Loss Detection .....	65
6.1.4 Alarm .....	65
6.1.5 Locking Mechanism.....	66
6.1.6 Wi-Fi Module.....	67
6.1.7 Keying Mechanism .....	67
6.1.8 Enclosure.....	69
6.1.9 Initial PCB Design .....	69
6.2 Hardware Design Review Process .....	70
6.3 Hardware Design Revisions.....	70

6.3.1 Power .....	70
6.3.2 Primary Power Loss Detection .....	71
6.3.3 Lid Sensor .....	72
6.3.4 Alarm Pull-Down Resistor.....	73
6.3.5 Locking Mechanism.....	73
6.3.6 Microcontroller Clock.....	74
6.3.7 Revised PCB Design.....	75
7.0 Software Design.....	76
7.0.1 Use Case.....	76
7.0.2 Database Structure .....	77
7.0.3 High Level System Flow .....	78
7.0.4 Low Level Operational .....	79
7.1 High Level Software System Architecture .....	80
7.2 Software Development Life Cycle.....	81
7.2.1 Planning & Analysis .....	81
7.2.2 Design .....	81
7.2.3 Implementation .....	82
7.2.4 Testing & Integration.....	82
7.2.5 Maintenance .....	82
7.3 Microcontroller Software.....	82
7.3.1 Keypad Entries.....	82
7.3.2 Lock Status.....	83
7.3.3 Lid Sensor .....	83
7.3.4 Alarm .....	84
7.3.5 IMU .....	84
7.3.1 Modes of Operation .....	85
7.4 Web Communications & Hosting.....	87
7.5 Web Development Revised.....	89
7.6 Mobile Device Application.....	91
7.7 Mobile App Development (SD2).....	94
8.0 Prototype Construction .....	98
8.1 PCB Vender & Assembly.....	98

8.2 Schematics .....	98
8.3 Coding Plan.....	101
9.0 Prototyping.....	103
9.1 Breadboard Prototype .....	103
9.1.1 Keypad .....	104
9.1.2 Inertial Measurement Unit .....	105
9.1.3 Alarm & Lid Sensor.....	105
9.1.4 Locking Mechanism.....	106
9.2 PCB Prototype .....	107
10.0 Testing & Verification .....	108
10.1 Table of Tests & Outcomes .....	108
11.0 Administrative Content.....	114
11.1 Role Assignment.....	114
11.2 Milestones .....	116
11.3 Initial Budget & Bill of Materials .....	117
11.3 User Manual.....	121
12.0 Project Summary and Conclusions .....	127
Appendices.....	129
Appendix A - Copyright Permissions.....	129
Appendix B - Datasheets .....	129
Appendix C – Requirements Specification Changelog .....	130
Appendix D – References .....	130

## List of Figures

Figure 2.4-1 House of Quality .....	10
Figure 2.5-1 Initial Project Flow.....	11
Figure 4.1-1 BoxLock.....	16
Figure 4.1-2 Amazon Key Application.....	17
Figure 4.2.1.1-1 Round Force-Sensitive Resistor .....	20
Figure 4.2.1.2-1 Dual Element Sensor.....	22
Figure 4.2.1.3-1 Magnetic Switch.....	23
Figure 4.2.2.2-1 Ultrasonic Rangefinder .....	24
Figure 4.2.2.3-1 Inertial Measurement Unit MPU-6050 .....	25
Figure 4.2.3-1 Audio Transducer.....	27
Figure 4.2.4.1-1 Wi-Fi Module ESP8266.....	27

Figure 4.2.5-1 Keypad Switch .....	29
Figure 4.2.7-1 uxcell Power Supply DC 3.7V 2500mAh.....	31
Figure 4.2.8-1 Keter 55 Gallon Outdoor Storage Cube.....	32
Figure 4.2.9.3-1 Arduino UNO.....	35
Figure 4.2.9.3-2 ATmega328P Pin Mapping (Courtesy of Wikimedia).....	37
Figure 4.3.1.1-1 AVRDUDE Command Example (CC BY-SA 4.0 License).....	38
Figure 4.3.3.4-1 React Native Bridge.....	47
Figure 5.2.1-1 Gikfun MC-38 Wired Door Sensor Magnetic Switch.....	49
Figure 5.2.3-1 5V Active Alarm Buzzer.....	50
Figure 5.2.4-1 ESP8266 Surface Mount.....	51
Figure 5.2.5-1 1824 Keypad Switch.....	51
Figure 5.2.6-1 Adafruit Lock-style Solenoid 12VDC Lock.....	52
Figure 5.2.7.1-1 SainSmart 1A Power Adapter 12V.....	53
Figure 5.2.7.2-1 Floureon 7.4V 5200mAh 2S 30C Lipo Battery.....	54
Figure 5.2.8-1 Suncast SSW1200.....	55
Figure 5.2.9-1 ATmega328P Surface Mount.....	56
Figure 6.1-1 SMD Part Size Comparison (Courtesy of Wikimedia).....	57
Figure 6.1.1-1 System Power Input.....	59
Figure 6.1.1-2 5V Regulator Circuitry.....	60
Figure 6.1.1-3 3.3V Regulator Circuitry.....	60
Figure 6.1-4 Power Source Loss Detection and Switch.....	61
Figure 6.1.1-5 Battery Charging Circuit.....	62
Figure 6.1.1-6 Overall System Power Specific Circuitry.....	62
Figure 6.1.2-1 Microcontroller Specific Circuitry.....	63
Figure 6.1.3.1-1 Lid Sensor Specific Circuitry.....	64
Figure 6.1.3.2 IMU Specific Supporting Circuitry.....	65
Figure 6.1.4-1 Alarm Specific Circuitry.....	66
Figure 6.1.5-1 Locking Mechanism Circuit Connections.....	66
Figure 6.1.6-1 Wi-Fi Module Specific Supporting Circuitry.....	67
Figure 6.1.7-1 Keypad input Specific Circuitry.....	68
Figure 6.1.9-1 PCB Eagle Design.....	69
Figure 6.3.1-1 Revised Power Circuitry.....	71
Figure 6.3.2-1 Power Loss Detection Circuit.....	72
Figure 6.3.2-1 Pull Up Resistor Placement.....	72
Figure 6.3.2-1 Revised Lid Sensor Circuitry.....	73
Figure 6.3.5-1 Revised Locking Mechanism Circuitry.....	74
Figure 6.3.6-1 16Hz Crystal Clock Circuitry.....	74
Figure 6.3.7-1 Revised PCB.....	75
Figure 6.3.7-2 Revised PCB without Ground Planes.....	76
Figure 7.0-1 Use Case Diagram.....	77
Figure 7.0.2-1 Database High Level Structure.....	78
Figure 7.0.3 High Level System Flow Chart.....	79
Figure 7.0.4 Low Level Operation Flow Chart.....	80

Figure 7.5-1 Website Home Page .....	90
Figure 7.5-2 Website Login Page .....	90
Figure 7.6-1 Mobile App Login UI.....	92
Figure 7.6-2 Mobile App Flow Diagram .....	93
Figure 7.7-1 React-Native Install with npm .....	95
Figure 7.7-2 Mobile App Authentication.....	96
Figure 7.7-3 Mobile App User Profile.....	97
Figure 8.2-1 Initial System Schematic.....	99
Figure 8.2-2 Revised System Schematic .....	100
Figure 8.3-1 Coding Plan Workflow .....	102
Figure 9.1-1 Breadboard Prototype.....	103
Figure 9.1.1-1 Keypad Connections and Circuit .....	104
Figure 9.1.2-1 IMU Connections .....	105
Figure 9.1.3-1 Lid Sensor and Alarm Circuitry .....	106
Figure 9.1.4-1 Locking Mechanism Connections.....	107
Figure 9.2-1 PCB .....	108
Figure 11.3-1 Register Account.....	121
Figure 11.3-2 Login to Account.....	122
Figure 11.3-3 Login Successful .....	122
Figure 11.3-4 Website Navigation Bar .....	123
Figure 11.3-5 Profile Dropdown Menu .....	123
Figure 11.3-6 Add Lockbox Button.....	124
Figure 11.3-7 My Lockbox Page .....	124
Figure 11.3-2 Edit Profile Page .....	126
Figure 11.3-2 Edit Profile Page Cont.....	127

## List of Tables

Table 2.3.2.1-1 Enclosure Requirements.....	5
Table 2.3.2.2-1 Power Requirements.....	6
Table 2.3.2.3-1 System Security Requirements.....	7
Table 2.3.2.4-1 Wireless Communication Requirements .....	8
Table 2.3.2.5-1 System Cost Requirements.....	8
Table 2.3.2.6-1 System Software Requirements.....	9
Table 4.2.1.2-1 Comparison of Force Sensors.....	19
Table 4.2.1.2-1 Comparison of Motion Sensors .....	21
Table 4.2.1.3-1 Comparison of Magnet Sensors.....	22
Table 4.2.3-1 Comparison of Audible Alarms.....	26
Table 4.2.5-1 Comparison of Keypad Switches .....	28
Table 4.2.6-1 Comparison of Servo Motors .....	29
Table 4.2.9.1-1 MSP430G2553 Specifications.....	33
Table 4.2.9.2-1 ATmega2560 Specifications .....	34
Table 4.2.9.3-1 ATmega328P Specifications .....	36



Table 4.3.2.3-1 Wireless Connection Comparison .....	42
Table 4.3.3.2-1 Android Studio System Requirements .....	44
Table 4.3.3.3-1 Xcode 9.3 (iOS SDK) System Requirements.....	45
Table 4.3.3.4-1 React Native IDE Comparison .....	47
Table 6.1.1-1 System Component Voltage and Power Needs .....	58
Table 7.3.1-1 Keypad Entries Function Structure .....	83
Table 7.3.3-1 Lid Sensor Function Structure.....	83
Table 7.3.4-1 Alarm Function Structure .....	84
Table 7.3.5-1 IMU Function Structure .....	85
Table 7.4-1 Database Software .....	88
Table 10.1-1 Tests Conducted .....	109
Table 10.1-2 Tests Conducted (Part 2) .....	110
Table 10.1-3 Tests Conducted (Part 3) .....	111
Table 10.1-4 Tests Conducted (Part 4) .....	112
Table 10.1-5 Tests Conducted (Part 5) .....	113
Table 10.1-6 Tests Conducted (Part 6) .....	114
Table 11.1-1 Role Assignment .....	115
Table 11.1-2 Final Work Distribution .....	115
Table 11.2-1 Senior Design I Milestones .....	116
Table 11.2-2 Senior Design II Milestones .....	117
Table 11.3-1 Initial Budget Breakdown.....	118
Table 11.3-2 Final Cost Per Unit .....	119
Table 11.3-3 Final Development Cost.....	120
Table 11.3-3 Final Development Cost Cont. ....	120

## 1.0 Executive Summary

Homeowners who receive packages on a daily to weekly basis usually receive them in an unsecure setting, their front porch. If the homeowner isn't home at the time of a package drop-off there is a possible chance of the package being stolen or damaged. Our team looked at possible alternatives to solutions that can counter package thieves in any type of environment. We first took a practical approach to the problem and listed what a consumer may be comfortable with when it comes to security for their deliveries. The key focus of our thought process for the design was to emphasize on security functionality, while also emphasizing a non-invasive approach to the consumers' daily lives. Many products already exist for this current market with more being created every day, however, our design wishes to alleviate some of the possible issues others may face with previous products. This design aims to create a barrier between invasion of privacy, as well as create a cheaper product with exceptional security features.

The simple solution to the issues with thieves stealing packages off porches is out on the market today where people sell custom mechanical boxes with a keypad for a large cost, or cheap alternatives that is a lock with a custom scanner on it, so they can fit on any box they deem chosen. There is also a more in-depth option where people can drop off packages directly into your home, using a service that authorizes the user and then takes pictures for security purposes. However, the issues with these designs that we aimed to focus on with our project was cost, security, and invasiveness, and deemed these approaches unacceptable for our overall outlook on fixing porch thievery. We aimed to create a product that counters the price points of lockboxes like the expensive ones available, the form factor like locks that can be put on any box, and the non-invasiveness of people entering your home.

The lockbox being designed will offer multiple security features to detect if tampering is occurring, and the ability to transmit this data to the end user, the consumer. The design focuses on the form factor of small but decently sized boxes to then convert into a smart lockbox where they will then be a more secure alternative to other products on the market, and for a fraction of the price. The size of the lockbox aims to be a size easy enough for the consumer to move to any designated location they wish on their front porch. Other than the form factor of the lockbox itself, this lockbox has the capability to send out alerts to the end user when power has been lost, battery has run out of charge, the box is being tampered with, the box has been unlocked and package received. Another security measure added was adding a loud alarm to scare away the potential thieves if tampering does occur.

This project is being collaborated with in accordance to two electrical engineers and two computer engineers. The electrical engineers are focusing on the wiring of the lockbox and all of its Arduino components, as well as soldering or adding any necessary additions to the project. The computer engineers are focusing on the computers website, applications, and connection from user to lockbox as well as some coding for the microcontroller. Both groups of engineers will collaborate on the most effective ways to approach each stage in the design and how to move forward when issues arise.

The final product of the lockbox will achieve the main purpose of security features with complete end user functionality. The box will be placed on any porch and will surface a locking mechanism to open the box when the delivery person utilizes the digital keypad to enter the correct password sequence. If someone tries to tamper with the box, the alarm goes off due to a motion detection component detecting unauthorized movement and sends out an alert to the user. In some situations, the box can be bolted down to ensure even more security for the customer if they so choose. Overall, our design aims to be similar to the approach of past products with more added functionality and security decisions in mind first. The cost for this product should be exponentially cheaper than the alternatives and offer features they also don't have to offer.

## 2.0 Project Description

The following sections contain an overview of the senior design project of group 5, including a short description of the project's planned functionality, the motivation for the project, and the project design specifications. For senior design, group 5 was attempting to develop a smart package lockbox system, a package protection system design to prevent package theft. The system was designed as a box with an electronic locking system with keying mechanisms that enable approved individuals, such as delivery service workers, to easily open the smart lockbox system in a specified timeframe, such as the expected timeframe for a package delivery. The smart lockbox system was also to be designed to be capable of detecting tampering or potential attempted theft as well as include capabilities for both alerting the owner via online communication and activating an included alarm as a deterrent.

The system is Wi-Fi capable and alerts the user through text or an app when the lockbox has been opened, the primary power source has been disconnected, or there is a security issue. Keying mechanisms for the box includes a keypad and the ability to open the device with either an app or text with a cell phone. The user can modify settings for the lockbox through an app or website, including the ability to set a time limited one-time keypad passwords or provide time limited access to specific phone numbers. Additional details of the Wi-Fi module use and software flow for the project are provided below in Section 2.5 in Figure 2.5-1.

The smart package lockbox was designed to run on 120V and 60Hz AC electricity from a standard US power outlet. In the case the power is lost, it can operate for a minimum of 24 hours on secondary power from a rechargeable battery that will be charged during normal operation of the device from the primary power source. Any external wireless modules utilized, such as a wireless keypad, will run on a rechargeable battery and operate for a minimum of 1 month on a single charge.

Aside from the primary power supply and access tools, such as a keypad, all components related to security, including the electronic locking mechanism, shall be contained within the box to prevent tampering. An inertial Measurement Unit (IMU) is used to determine if

theft of the entire device is being attempted and to aid in determining if someone is attempting to break into the box in the case that theft of the entire device is not being attempted. A lid sensor also is used in determining if someone is attempting to break into the device. In the case that the device determines that theft or tampering is taking place, an alarm of at least 90 dB activates as a deterrent and the Wi-Fi module will notify the user. Additional details of the hardware structure of the device are provided below in both the initial project flow in section 2.5 (figure 2.5-1) as well as in section 5.0 of this document.

## 2.1 Motivation

In recent years, online order services, such as eBay and Amazon, have become increasingly prevalent in the retail industry. These services provide users a convenient and often more affordable alternative to the traditional brick and mortar stores. Recently, some retail services have even managed to minimize two of the major drawbacks to online retail services, shipping cost and delay. For example, for a yearly fee, users of the Amazon Prime service are provided with free two and one day shipping options.

Unfortunately, another potential drawback of online retail services is theft of the products before they are received by the consumer. Although online package tracking options are available, and an estimated time of delivery is usually provided, the exact delivery window may be wide, and it may be difficult for some consumers to be available to receive their packages. This often leads to packages being left at the front door where they are vulnerable to potential theft. Especially during the holiday seasons, when the volume of potentially expensive items being shipped is at its highest, there have been many recorded cases of package theft. In order to prevent potentially expensive packages from being left in places where they are vulnerable to theft when the recipient is unavailable, a smart package lockbox is to be developed.

Current package theft solutions have various drawbacks. Amazon, for example, sells a door lock and camera combination system that allows the Amazon delivery service to open the owner's front door in the case they are not home. Although this includes a camera that would monitor the delivery worker, it is obvious that a solution that requires providing strangers access to a consumer's home is far from ideal.

Unfortunately, other currently available solutions to this issue either do not offer adequate security or are only available at high price points and inconvenient to use. Cheaper solutions provide minimal actual security for the delivered packages and can often be circumvented with bolt cutters. However, products marked as solutions to this issue that do provide sufficient security are costly and require the owner to secure or bolt the system to a surface. The purpose of this group's project is to create a practical package protection solution that balances cost, security and ease of use. The smart package lockbox would provide easy to use, affordable, and secure storage for packages until the recipient is available to receive them.

## 2.2 Goals and Objectives

To design a secure smart package lockbox that does not require the user to secure or bolt it to a surface, the device will need to include security features that can prevent the theft of not only the packages, but also the lockbox itself. For this purpose, the device will need to be able to detect if someone is attempting to take the entire lockbox. To prevent tampering, the security features should also all be contained within the box and not be accessible when the device is locked. However, the security of the product needs to be balanced with the final product cost to make the device affordable.

In order to get delivery services to utilize the lockbox, the device needs to also be easy for the delivery service to use. In addition, the device needs to be able to detect and alert the user when a package is delivered. In the case of attempted theft, the smart package lockbox should be capable of deterring the theft with an alarm system and alerting the user. There could potentially be a variety of different sizes and models to suit the user, however, the size of the lockbox for this proof of concept should be made to balance cost and assumed average package size for most potential users.

## 2.3 Requirements & Specifications

The marketing requirements as well as the engineering specifications for the Smart Package Lockbox are contained herein. Verification methods for the project engineering requirements are detailed in section 8.0.

### 2.3.1 Marketing Requirements

Section 2.1 and Section 2.2 detailed the motivation, goals and objectives for the system to be designed. To summarize, research on the current existing products as well as products in development marketed as solutions to package theft either require the consumer to allow the delivery service access to the consumers home or do not adequately balance provide security and cost and/or ease of installation. Many of the available products either offer low cost, but inadequate security or come with a prohibitively high cost. A detailed analysis of both currently available and in development related products marketed as solutions to package theft is provided in Section 4.1. The primary motivation of the Smart Package Lockbox is to develop a solution to the package theft issue that both balances cost with provided security and lacks the major drawbacks of requiring the consumer to either grant a delivery service the ability to open the front door of a home or bolt a lockbox to a surface. The following are the marketing requirements for the project that were determined after taking the project goals and motivations into consideration:

1. The system should be easy to install.
2. The system should be easy to use and maintain.
3. The system should have low cost.
4. The system should be secure against theft of the system or its contents.

5. The system should be configurable and unlockable through the internet and a phone application.
6. The system should be large enough to accommodate most typical package sizes.

## 2.3.2 Engineering Requirement Specifications

This section documents the engineering requirement specifications for the spring 2018 senior design project for group 5. At the time of writing, the engineering requirements contained herein are meant to serve as a starting point and may not form a complete list. Requirements may be added, removed, or altered at a later date with both approval from the senior design professor and unanimous agreement from each member of group 5. Any changes made to the engineering or marketing requirements as well as the reason for the change in requirements will be documented in the requirements specification changelog in Appendix C.

### 2.3.2.1 Enclosure

The engineering requirements for the enclosure of the Smart Package Lockbox system are detailed below in Table 2.3.2.1-1. The following two requirements are set to ensure the final design has adequate storage space for delivered packages and to increase ease of installation.

**Table 2.3.2.1-1 Enclosure Requirements**

ID	Marketing Requirements	Engineering Requirements	Justification
E1	1, 2, 4, 6	The enclosure interior should contain 4 to 12 cubic feet of storage space.	Common outdoor storage chest sizes typically fall in this range. It was determined that this range could accommodate most typical package sizes.
E2	1	The enclosure should not require the user to bolt or secure the system enclosure to a surface.	Requiring that the user secure the system to a surface is undesirable. It both increases the difficulty of installation and makes the project an unviable option for some consumers.

## 2.3.2.2 Power

The engineering requirements relating to the power sources utilized by the system are detailed below in Table 2.3.2.2-1. The following three requirements were written for the purpose of improving system reliability and security as well as reducing the required maintenance. It should be noted that the below required battery life values in the power requirements are only the minimum acceptable durations for the project to still be considered successful. Attempts will be made to increase battery lifetime durations, but this will need to be balanced with project cost and available time.

**Table 2.3.2.2-1 Power Requirements**

ID	Marketing Requirements	Engineering Requirements	Justification
P1	1, 3, 4	The system must be able to run on 120V and 60Hz AC electricity from a standard US power outlet.	The specified primary power source has higher reliability and lower cost than other explored options, such as solar power. Because the project is intended for use in front of the consumer's home, this power source is typically easily accessible.
P2	2, 4	The system will be able to operate for a minimum of 24 hours on secondary power from a rechargeable battery in the case that primary power is lost.	A secondary power supply is needed to prevent a potential thief from just cutting power to the lockbox. The system ease of use is also improved because potential issues due to temporary loss of primary power are prevented.
P3	2	Any battery powered external modules (such as a wireless keypad) must be able to run for a minimum of 2 weeks on a single battery charge.	This is achievable with the lower power draw a wireless keypad would have. A long battery life on external modules will reduce the maintenance required by the user.

### 2.3.2.3 System Security

The system security engineering requirements are detailed below in Table 2.3.2.3-1. The requirements below outline the initial plan for achieving a reasonable level of system security with the design and cost constraints imposed by the requirements in Section 2.3.2.1 and Section 2.3.2.5. Additional details on the overall project design constraints are provided in Section 3.0.

**Table 2.3.2.3-1 System Security Requirements**

ID	Marketing Requirements	Engineering Requirements	Justification
SE1	4	All security related components of the system, including the locking mechanism and alarm, must be contained within the lockbox and not accessible to tampering while the box is locked.	Containing all security related components within the lockbox prevents potential tampering and damage due to wear and tear. System security and reliability are improved as a result.
SE2	4	The system shall include an alarm with a minimum Sound Pressure Level (SPL) of 85 dB or higher as measured from outside the lockbox while locked.	In the case of attempted theft, the alarm is meant to act as a deterrent. It was determined that the set SPL level was achievable and sufficient to draw attention to the attempted theft of the lockbox.
SE3	4	The attempted theft or unauthorized moving of the system as a whole shall be detectable with a minimum success rate of 95%.	Due to requirement E2, a countermeasure for the potential theft of the system as a whole is required. The minimum success rate specified was determined to be within reasonable limits.
SE4	4	The system shall include a lid sensor that will correctly detect and report the lid open/closed position status.	The system needs a method to detect forced entry into the lockbox.
SE5	2, 4	The system should be able to detect and report the loss of the primary power source.	The system needs to provide notification to the consumer in the case of primary power loss to prevent eventual total loss of power to the system.



## 2.3.2.4 Wireless Communication

The engineering requirements relating to the wireless communication modules are detailed below in Table 2.3.2.4-1.

**Table 2.3.2.4-1 Wireless Communication Requirements**

<b>ID</b>	<b>Marketing Requirements</b>	<b>Engineering Requirements</b>	<b>Justification</b>
W1	1, 2, 4, 5	The system shall include a Wi-Fi module with a minimum range of 100 meters from within the lockbox while closed.	The system need to be within the range of the consumer's Wi-Fi signal. This range is achievable with many of the currently available Wi-Fi modules.
W2	2	External wireless modules (such as a wireless keypad) shall have a minimum range of 50' from within the lockbox while closed.	It was determined that 50' would be a sufficient and achievable range.

## 2.3.2.5 Cost

The engineering requirements are detailed below in Table 2.3.2.5-1. The economic constraints of the project are explained in Section 3.2.1. The Initial project budget and bill of materials can be found in Section 9.3.

**Table 2.3.2.5-1 System Cost Requirements**

<b>ID</b>	<b>Marketing Requirements</b>	<b>Engineering Requirements</b>	<b>Justification</b>
C1	3	The costs for developing the system should not exceed \$500.	This is based on the available funds of group 5.
C2	3	The total material cost of the system should not exceed \$250 per unit.	This is based upon market analysis of existing products.

## 2.3.2.6 Software

The software engineering requirements are listed below in Table 2.3.2.6-1. See Section 4.3 and Section 6.0 for additional details on the research and design of the project software.

**Table 2.3.2.6-1 System Software Requirements**

ID	Marketing Requirements	Engineering Requirements	Justification
S1	2, 4	System should be capable of sending notifications to the user via email or a phone application.	The system will need to be able to notify the user in the case of attempted theft, a received delivery, and loss of the primary power supply.
S2	5	The system should allow remote access via a computer and/or phone application in order to view the status of the system, lock/unlock the lockbox, and change security and/or access settings.	The specified functions will enable the user to monitor the system as well as configure the system based on the user's needs.

## 2.4 House of Quality Analysis

This section details the house of quality for the smart package lockbox project. The house of quality seen below in figure 2.4-1 is a diagram that shows the correlation between marketing and engineering requirements as well as the correlation between the differing engineering requirements of the project. The information presented by the house of quality can aide with determining effective tradeoffs between marketing and engineering requirements as well as between the different engineering requirements during product design. The selected marketing requirements for the project were the cost, lifetime of batteries used, product security, and the wireless module range. The engineering requirements selected were the product material cost, lifetime of the batteries, alarm strength, and the wireless module range. Aside from the product security and ease of use, most of the requirements share a weak negative correlation.

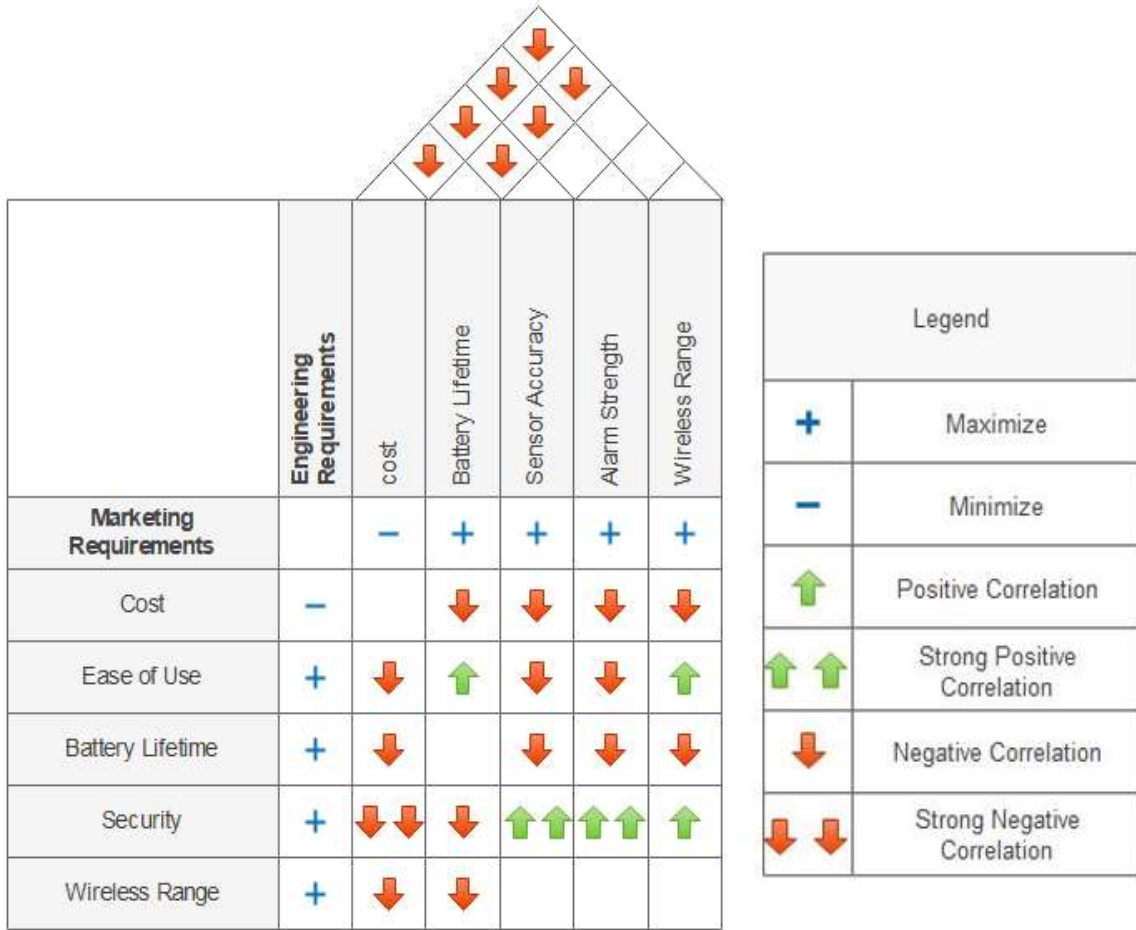
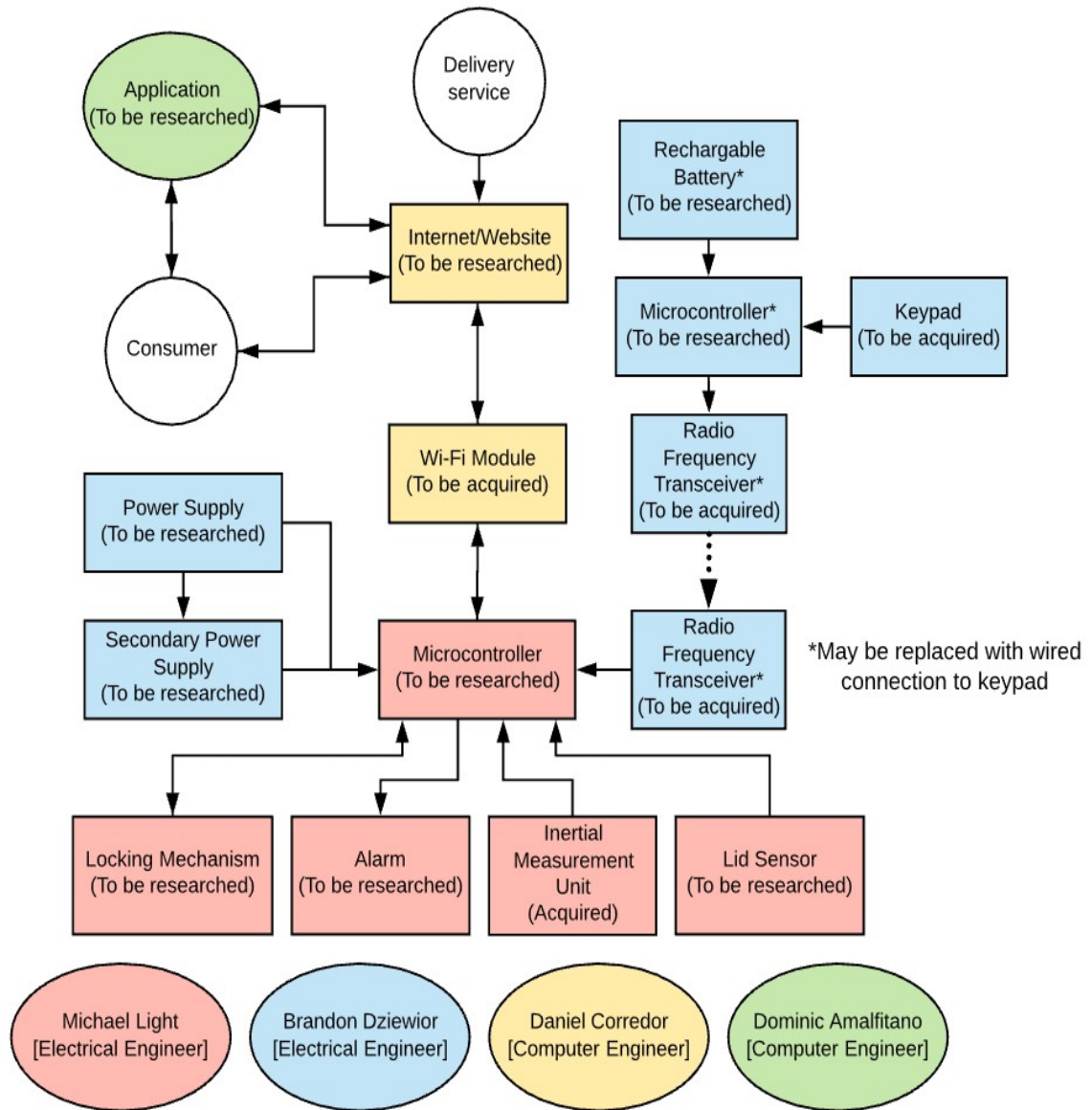


Figure 2.4-1 House of Quality

## 2.5 Initial Project Flow

The initial project flow for the smart package lockbox is displayed below in figure 2.5-1. This project flowchart depicts both the project flow and the members primarily responsible for each section of the project. It should be noted that a member being assigned primary responsibility to a project section does not necessarily mean that the assigned member will be the only member working on that section. Secondary members will be assigned to each section in addition to the primary member. When required, members not assigned either primary or secondary responsibility to a project section may also provide assistance for that part of the project. Further details of project responsibility assignment can be found in section 9.1.



**Figure 2.5-1 Initial Project Flow**

To summarize, the following are the expected higher-level final inputs and outputs of the system to be designed:

**Inputs:**

- Phone application and website that can send keying information (or user defined settings changes) to system via Wi-Fi
- Wireless or wired 12-button keypad that can transmit keying information to the system

- Lid sensor to determine and record lid position data
- Inertial Measurement unit to determine and record data on system orientation and movement
- Locking Mechanism sensor to determine and record whether or not the systems locking mechanism has successfully engaged
- 120V and 60Hz AC electricity from a standard US power outlet

### **Outputs:**

- Phone application and website that can receive (via Wi-Fi module) and display information on system status and alert the user in the case of authorized or unauthorized access of system
- Electronic locking mechanism that can engage or disengage as directed by the system's primary microcontroller
- Alarm with a sound pressure level (SPL) of at least 88dB (as measured from outside the box while closed) that can engage or disengage as directed by the system's primary microcontroller

## **3.0 Related Standards and Design Constraints**

The standards relating to the project as well as their impact on the project are contained herein in addition to the various project design constraints and their impact.

### **3.1 Standards**

- Power standards
  - Adapting the US power standard of 120 V and 60 Hz for electric power supply. The lockbox will operate mainly under a wired power connection to an outlet. Therefore, we will abide by this standard.
- Wi-Fi standards
  - 802.11 is the standard set by IEEE for wireless communication in the US. Our device will use a wireless adaptor chip therefore we will abide by these standards and they are relevant to our project. Under this standard communication on the 2.4 GHz wireless band is allowed on up to three non-overlapping channels. The Wi-Fi band will be used to sync and communicate with the lockbox.

#### **3.1.1 Standards Impact on Design**

##### **Power**

Our power adaptor to the lockbox will need to abide by the US power standard of 120 V and 60 Hz electrical power supply. This is necessary to receive power from the outlet in

most homes. Without this our device may not operate correctly or be hazardous to users. Following this constraint will allow our device to be successful and safe for users.

### **Wi-Fi**

Following the Wi-Fi standards will allow our device to communicate effectively and correctly with Wi-Fi enabled devices. This is essential due to the need of other devices to be used in unison with our own. This standard will allow our device to be integrated into the Wi-Fi network so long as everything is implemented correctly.

## **3.2 Design Constraints**

There were multiple design constraints that had to be looked at when coming up with the design of the Smart Package Lockbox. This chapter analyzes the economic, time, environmental, social & political, ethical, health & safety, manufacturability, and sustainability constraints that had to be examined.

### **3.2.1 Economic Constraints**

The bigger constraint as a team we must face is the financial costs that this project will require for it to successfully succeed. Due to the non-backed project we undertook we must financially secure the components and hardware ourselves to accomplish the final product. This will be best accomplished by receiving a financial sum from each student so that together the expected financial cost will be around \$400. With each student attributing \$100 as agreed upon this will alleviate the economic constraint of paying for the entire project by ourselves.

If further down the line the project ends up exceeding the estimated cost it should be around only a \$100 or \$200 which was also agreed upon by the group members that we would all pay equally to contribute to the final amount. A major concern with the financial cost is the good chance of components going bad, or products being damaged. If this occurs, we have agreed upon as previously stated to equally share the cost so that product is successfully completed.

### **3.2.2 Time Constraints**

The design and construction of the Smart Package Lockbox will span from January, 2018 to July, 2018, approximately around 30 weeks of research, design, and development. During this interval, breaks and absences from school will occur, but the focus on the project must remain to effectively contribute enough to finish the final product in a timely manner.

Within this period of meetings and lectures the group will need to sort out the appropriate contributions to make sure the project requirements are met and assigned to the most capable person for each task. With two Electrical Engineers and two Computer Engineers the correct tasks should be split up to the appropriate expertise so as to effectively utilize

the little time we have the most efficiently and to not interfere with others work unless assistance is requested.

### **3.2.3 Environmental Constraints**

The nature of our Smart Package Lockbox and its components will require a closed in or shaded environment that is elevated above the ground. This is due to the container we are utilizing will have many electronic components attached inside and maybe outside the box and the best course of action for the hardware to not get damaged would be to isolate the product in a more environmentally friendly area. The container itself will be made of material that should be able to resist wear and tear from environmental conditions, but to be safe our main constraint would be isolating the container as to make sure an issue doesn't arise due to rain seeping into the bottom of the container or rain hitting the side and damaging components.

If the design phase decides the best course of action is to implement all the components within the box itself besides the locking mechanism, then the environmental constraint may be lifted or more lenient on the placement of the final product. Otherwise, the constraint should be monitored and based on our current standings with the design should be a primary focus on how our box is constructed and where it should be placed if publicly made available.

### **3.2.4 Social Constraints**

The social constraints our design was facing with in construction was the idea of the regular consumer disregarding products that would be intrusive into their life or environment and privacy near homes. One of the main issues we faced with in the idea phase of our project was the security of our system over other products out on the market. People tend to dislike intrusive measures that would affect their daily lives and so we had to take into that account and disallow methods of product construction that would require bolting to properties or delivery systems through the front door. Also, a key factor to note in relation to social constraints would be porch privacy. A lot of consumers dislike people on their doorsteps and some even may feel uncomfortable, we also had to work around this when figuring out a proper delivery method for the packages, while also being secure.

### **3.2.5 Ethical Constraints**

The ethical constraints our design is approached with respond to ethical concerns in our society regarding home-built appliances that handle sensitive information like personal packages. The design is built with the process in mind of keeping the confines of a personal's belonging securely safe within the users own home boundaries, while still maintaining the functionality that is mandated by our design process. One of the more prominent ethical constraints our design will face is the personal space of the end user having the design on their property with the full responsibility of the packages inside the box. The design of this lockbox must consider the liability of the delivery driver, and our

storage unit in the process of transporting packages from one source to the user. This is approached using security measures to notify the user if the box is being tampered with and multiple fail safe's such as if the power is cut the lockbox will not open for anyone until power is restored by the user.

### **3.2.6 Health & Safety**

It is crucial that our device is securely grounded and properly weatherproof. This ensure there will be no damage to the internal components that may cause harm or damage the lockbox integrity. Additionally, proper grounds along with following electrical standards will ensure that the device is safe for users to manage. Other than the power adaptor there is very little area for a user to be harmed. Good insulation and padding will help to make sure circuitry and PCB board are kept secure and away from contact.

### **3.2.7 Manufacturability**

The manufacturability constraints our design faces are the overall affordability and ease of construction. The main task with construction of these smart lockboxes will be the use of manual labor to perform the ideal placement and attachment of their corresponding correct components. All components utilized in the design will be purchasable from outside sources and are readily available for delivery in a decent timespan. For example, the Arduino board that will be used is a very popular unit that is sold in bulk to many distributors that can be purchased at any moment. The one component that will cause constraint issues with other container shapes will be the Inertial Measurement Unit (IMU). The IMU uses personally coded data to understand the placement and orientation of each individual container and will need to be coded to their respective lockbox that is constructed. With the purpose of this design being to demonstrate the ability to create a cheap, affordable, and secure lockbox we will be using one container type for our manufacturing purposes. The IMU though will be flashed to be coded to work with any box, the configurations just need to be placed with respect to the differing dimensions and placement. The design for this project will utilize numerous components that have many alternatives in the situation where one device does not work with our specific design it can be replaced without much time delay due to availability of differing components. Another issue with manufacturability will be the battery types and temperature ranges. In general batteries are fitted for ranges of temperature that the lockbox is only suitable for the correct environment that will not cause any performance loss on the battery. The design overall needs to be structurally sound so that there are no loopholes in the security measures our design has to offer, the containers used in manufacturing will need to be tightly sealed as well as have strong hinges to alleviate possible tampering with the lid of the box.

### **3.2.8 Sustainability**

The sustainability constraints this design will face will be the longevity of the box in multiple environments while maintaining the same security and operating features that our project hopes to offer to the end user. The main constraint will be the battery life of the



project and why one of the many alternatives is to have a plugged-in power source with a battery backup in case of power loss due to home power outage. This backup battery source will help alleviate sustainability concerns where energy loss or damage to the power source will still lead to an operable lockbox to counter measures held in place. The project overall construction should be sustainable in long term use as long as measures are put into place that will help alleviate long term negative effects on the battery and PCB units.

## 4.0 Research

Section 4.0, Research, is primarily focused on finding the correct components and products that will work best with our design phase of the project. Through looking up different component types and prices the design will be best accommodated to works best with the currently available components and capabilities. This section goes over, lid sensors, system movement devices, alarms, power supplies, microcontrollers, wifi modules, wireless communications, radio transmitters, keying mechanisms, locking mechanisms, and enclosures for the packages to reside in.

### 4.1 Existing Products

When looking at similar products currently out on the market revolving around ideas pertaining to personal lockboxes for package deliveries, a few major options stuck out during the idea phase of this project. One of the products that had a good selling point was the BoxLock product developed by BoxLock, Inc. This design was focused primarily on an external locking device that has an embedded scanner to open the lock when a package that is expected arrives on a particular day that is predetermined by the delivery service. This was the primary product that was looked at due to it being a product that was just introduced commercially and the ease of use that it gives the customer due to it being similar to a regular lock that you put on a bicycle for safekeeping. A picture of the device can be shown below in Figure 4.1-1.



**Figure 4.1-1 BoxLock**

The issues that may appear with this type of product is the nature of thieves of delivery packages would adapt to this new set of theft protection developed by BoxLock, Inc. Within the “FAQs” section of their site they mention their shackle on the locks themselves are made of the same materials as regular padlocks that a regular pair of bolt-cutters can easily destroy with enough force. There also wasn’t information on whether or not there was an alarm system hardwired into the device, but that would be another issue that could possibly come from this product. The total cost for the BoxLock is approximately \$129 currently.

Another major product that was looked at during the idea phase of the project was the new product that Amazon is offering called Amazon Key. Amazon Key is an interesting concept where the driver that drops off the packages has a code given to them that opens the door so they can walk into the home and drop off the package. Amazon admits the product’s package of the Amazon Cloud Cam, and Amazon Key-compatible smart lock also allows their product to be used for other situations such as house maids, guests, etc. The product of the Amazon Key Application is shown below in Figure 4.1-2.



**Figure 4.1-2 Amazon Key Application**

The concept of this product is interesting in that it allows the user to be away from their home, but still have the packages delivered to a safe location for retrieval. The issues that may arise from this product is the security concerns that customers may have and may decide not to purchase the Amazon Key. The total cost for the Amazon Key is \$249 currently.

Lastly, looking at general lockboxes for outside use most available storage containers that are purchased through other retailers to store packages in are usually covered in metal and other expensive materials that drives the price up to high levels around \$300 or higher which on average also includes a keypad or lock of some sort for security purposes. Other

storage containers have simple outside latches that can be locked or unlocked for low costs around \$100 or less. None of these options however seem to have an alarm system that alerts the user when their container is being stolen or broken into or an inside locking mechanism on the cheaper models so that thieves can't break into the containers and steal the contents and leave.

## **4.2 Hardware**

There are many different hardware components that are necessary for our design to successfully meet the requirements we set out for the final product to achieve. Due to the main features of our design revolving around security and convenience to the consumer many components are needed for the final product to give a result that is comparable to products that are already on the market that exhibit additional features or similar to lower costs. Lid sensors, detection technologies, and alarms are all necessary components to help alert when the device is being tampered with in either allowed circumstances or at times when someone other than the package handler is trying to confiscate the packages within the container. The hardware features of this design also need to be able send out signals through means of wireless communications to transmit data to the consumer when their package has been dropped off or the container's lid has been opened. The design also of course needs a proper locking mechanism and/or keying mechanism for the container to be opened and closed with security and proper authentication. The main component for this design will be the power unit and the most cost effective, but long-lasting solution, to maintaining a constantly working security lockbox.

### **4.2.1 Lid Sensors**

Through research, there are a few different sensor types that may help identify when the lid is lifted/removed by the user and to transmit this information to the microcontroller. The goal of the lid sensor is detrimental to the security of the overall design portion of the project and to ensure that the user is notified if a package is being dropped off. The main requirement of this component is tasked with being able to detect a change in the lid of the container being used in any manner, which ever bests fits the purpose of the design and helps alleviate costs for the overall project.

#### **4.2.1.1 Force Sensor**

A force sensitive resistor, or FSR, is a resistor that detects a change in force being applied to it via weight, pressure, and actual squeezing of the component. Force sensitive resistors are highly manufactured products and are easy to use as well as being low cost products. They are also generally small components where most are the size of a quarter or smaller, so the utilization in our designs could entertain the idea of using a force sensitive resistor. The components design uses a round circular top portion that is used to detect the change in force connected to a long frame houses the wires for the pins of any compatible programming board. FSR's are three-layer components, one being the flexible substrate

with the electrodes, one being the flexible substrate with the semi-conductor, and the final layer being the spacer adhesive that separates the two main layers.

Force sensitive resistors basic functionality is to change the resistive value inside the component the more pressure is applied to the surface. Due to the low cost and ease of use in application the component in general isn't accurate as other alternatives, but if the sole purpose is detecting a change in resistive values it will work just as effectively as other components that fit the same role as sensing products. In the general sense for our specific design purposes the product would work wonderfully due to it being a low power draw component that is also cheap to buy. The following stats are common across all FSR's:

- Size: ½" (12.5mm) diameter active area by 0.02" thick
- Resistance range: Infinite/open circuit (No pressure) otherwise 100kohms (light pressure) to 200kohms (max pressure)
- Force range: 0 to 20lb (0 to 100 Newtons) applied evenly over the 0.125 sq in surface area
- Power supply: Uses less than 1mA (Depends on any pullup/down resistors used and supply voltage)

A few different types of force sensors could be an option for our experiment due to the pressure of the lid against the frame of the container. Some sensor types such as resistive force sensors can be used to detect when the frame of the container is pressed against the lid. If the sensor recognizes when the lid is resting, or not resting, against the frame this can be transmitted to the microcontroller as resistive data to let the user know the lid has been opened or closed. There are a few different options for force sensors, with each having different shape types to accommodate different designs. Below is Table 4.2.1.2-1 that shows the comparisons between possible options for a force sensor.

**Table 4.2.1.2-1 Comparison of Force Sensors**

<b>Digi-Key Part Number:</b>	<b>Description:</b>	<b>Voltage Supply:</b>	<b>Current:</b>	<b>Pins:</b>	<b>Price:</b>
1528-2335-ND	Round Force-Sensitive Resistor	~4.9V	1mA/cm <sup>2</sup> of applied force	2	\$7.00
30056-ND	Force-Sensing Resistor	2.7V~5.6V	1mA/cm <sup>2</sup> of applied force	2	\$24.99
FSS020WNSR-ND	Sensor Surface Mount	3V~6V	1.5mA	2	\$36.94

The outcome of the force sensor would work in our experiment due to the effective transmission of constant data letting the microcontroller know when the lid is opened or closed. Many different types of force sensors for different circumstances are available as listed above in Table 4.2.1.2. Each of these sensors exhibit the same sensitivity range that will work for the purpose of the recognition of the lid touching the frame of the container. Due to the nature of resistive force sensors requiring a constant current running through them to give an updated consistent measurement to the microcontroller the power supply would have to be constantly feeding power to the force sensor. Overall, the force sensor seems like a feasible option for the design of our project, but alternatives may appear that give the same results with less requirements or for a smaller cost. If one of the above parts were to be chosen the best product will be that cause the least amount of current draw from the power supply, but also a huge factor is cost and any given container allowing the sensor to be attached appropriately. The best product for the cost would have to be the first option in Table 4.2.1.2, 1528-2335-ND (shown below in Figure 4.2.1.2-1), due to the low current draw of the sensor and lower price point then most other options. The only issue with the first option may be the setup procedure for the sensor to fit accordingly into the container with ease, but due to the small size of the sensor in width it will fit nicely along the frame of the container if chosen as a final component piece for the design phase of this project.



**Figure 4.2.1.1-1 Round Force-Sensitive Resistor**

## **4.2.1.2 Motion Sensor**

A motion sensor component is a device that is incorporated into various devices to be utilized as infrared detection. The infrared functionality determines a difference in distance if the component is correctly coded. The main design purpose is to determine the distance between an object and a person, to send that signal to the receiver and use that data for a meaningful application. However, motion sensors also come in other forms were the device

can be used to determine the distance between inanimate objects such as a ball on the floor or in our personal case a lid being moved away from the top of the container.

A different type of sensor that can be used to detect movement for the lid opening and closing would be the motion sensor. Motion sensors detect the slightest change in movement to output a signal to the microcontroller with data on when the environment has changed to a different specification. This could be an easier alternative to the force sensor due to the nature of the sensor requiring less precise positioning against the frame. A few different options are available for motion sensors shown below in Table 4.2.1.2-1.

**Table 4.2.1.2-1 Comparison of Motion Sensors**

<b>Digi-Key Part Number:</b>	<b>Description:</b>	<b>Voltage Supply:</b>	<b>Field of View (Vertical):</b>	<b>Pins:</b>	<b>Price:</b>
269-4928-ND	ZMotion Pyroelectric	2V~3.6V	82°	1	\$4.24
PYD5790TR-ND	SMD Digipyroelectric	2.7V~3.6V	76°	1	\$3.40
LHI878-ND	Pyroelectric Dual Motion	2V~12V	87°	1	\$6.12

Overall, the decision of choosing a motion sensor could be an effective one due to the nature of it requiring just a sense in any sort of motion instead of precise resistive data like the force sensors. The only issue that may lie with the motion sensors over force sensors or other type of sensors is the complexity of the device itself as well as voltage supply levels. The motion sensor price ranges are much more reasonable or varying then force sensors as well, but another sensor type still may be best suited for this specific design we have in mind. If a sensor from the motion sensor section was to be chosen the best suited would be a sensor that requires the least amount of voltage and gives a decent vertical field of view so that sensor can be attached to the container in a non-invasive manner. The closest option from the looked at sensors would be the PYD5790TR-ND motion sensor (shown below in Figure 4.2.1.2-1) from SMD. The sensor is cheaper than the alternatives and the field of view doesn't have to drastically be larger then the other choices for it to work as effectively.



**Figure 4.2.1.2-1 Dual Element Sensor**

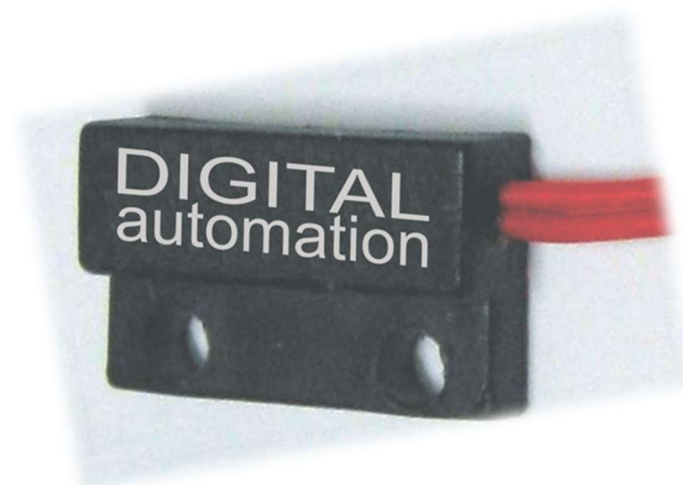
### 4.2.1.3 Magnet Sensor

One other sensor type that could be of use for this design is the magnet sensor. The magnet sensor has two solid magnets connected to different materials and when they meet one another the device registers a magnet lock and the switch turns on. This would work well for this design where one magnet is attached to the bottom of the lid and one magnet attached to the top of the frame, primarily along the corner of the container. This would give the “resistive” data of the force sensor where the magnet sensor detects a change in motion due to the two pieces being separated from one another and giving an immediate switch state to recognize this. A few different magnet sensors are shown below with voltage, pin requirements as well as prices in Table 4.2.1.3-1.

**Table 4.2.1.3-1 Comparison of Magnet Sensors**

<b>Digi-Key Part Number:</b>	<b>Description:</b>	<b>Voltage Supply:</b>	<b>Current:</b>	<b>Pins:</b>	<b>Price:</b>
TCS40DPRLFCT-ND	Magnetic Switch	2.3V~5V	1.6mA	1	\$0.40
620-1232-1-ND	Magnetic Switch	2.1V~5V	2mA	1	\$1.09
480-3327-1-ND	Magnetic Switch	2.2V~5.5V	640uA	1	\$1.51

Magnet sensors seem like one of the better options for this design due to the low cost, voltage, and pin requirements. The magnet sensor completes the tasks of the previous sensors types looked at as well, but for a cheaper overall cost. The simplicity of the switch state also helps alleviate current load on the power supply unit by not needing to be always run at a constant high voltage for the sensor to operate at full capacity. This would help alleviate strain on the power supply and increase longevity of the container's battery life time for less frequent recharge periods or battery replacements. The final part that seems the most appropriate for this project would be the 480-3327-1-ND (shown below in Figure 4.2.1.3-1), due to the low current supply required for the overall component. The lower the current draw can be for a component such as the lid sensor would be the ideal choice due to it constantly needing to sense whether the lid is closed or unclosed. Even though the price point is the most expensive out of the three looked at and numerous others, the lower current will make up for it since the design that was chosen will have many components drawing power from the power supply regularly. If one component was to be chosen for the lid sensor it would be preferred to be the magnet switch.



**Figure 4.2.1.3-1 Magnetic Switch**

## **4.2.2 System Movement Detection**

Per requirement SE3 detailed in Section 2.3.2.3, the system must be able to detect the attempted theft or unauthorized moving of the Smart Package Lockbox. The below sections describe the research of potential methods for filling this requirement.

### **4.2.2.1 Tilt Switch**

Tilt switches can mechanically detect basic orientation and motion. A tilt switch originally were designed as a hermetically sealed chamber containing two electrical contacts at the bottom of the chamber and a bead of mercury. A simple tilt switch can detect basic orientation or motion because when the device changes orientation the mercury bead inside



the hermetically sealed chamber will touch the contacts at the bottom of the chamber and complete a circuit. Due to safety and environmental hazards presented by the use of mercury, most modern tilt switches now use metallic balls instead. Some tilt switches also include contacts on each side of the chamber to determine the tilt directions and increase the resolution of the data.

One potential solution that was looked into was the use of a tilt switch to detect changes in orientation with the reasoning that it would be unlikely that a thief would hold the lockbox perfectly level while attempting to steal it. As a switch, the power consumption would be negligible. The cost of tilt switches varies depending on user needs, however, most tilt switches are relatively low cost and can be obtained for under \$8. Although tilt switches offer a low cost solution with a negligible power draw, it was determined that they were less effective than other available solutions for this project's application.

#### 4.2.2.2 Sonar Rangefinder

Another potential solution researched was the use of a rangefinder to determine the distance between the ground and the sensor in order to detect changes in that distance if the Smart Package Lockbox was either lifted or changed orientation. Two types of rangefinder were looked into, laser rangefinders and ultrasonic rangefinders. For the purpose of this project, laser rangefinders were found to be unviable primarily because of their prohibitively high cost ranging from \$150 to over \$300 from suppliers such as DigiKey.com. Ultrasonic rangefinders, however, had significantly lower prices reaching as low as \$4 from the same suppliers. An example of one of the cheaper ultrasonic range finding modules available, the HC-SR04, can be seen below in Figure 4.2.2.2-1.



**Figure 4.2.2.2-1 Ultrasonic Rangefinder**

Ultrasonic rangefinders work by sending short bursts of sound pulses in narrow beams that are reflected off objects and returned to the device. The time delay between the ultrasonic rangefinder sending the signal and receiving the signal is then measured and used to determine the distance between the rangefinder and an object in front of it by multiplying

the speed of sound by the time delay divided by 2. Heat and humidity can change the speed of sound and may require heat and humidity sensors to compensate for some applications. For the purpose of this project, however, those changes would be negligible because the device would be used to detect sudden changes in recorded distance rather than triggering if the distance value fell outside of a specified range. The downsides to this approach are the power draw and a decrease in security. The ultrasonic rangefinder module shown above, the HC-SR04, requires 15mA while working. The primary issue with this method, however, is that per requirement E3 detailed in Section 2.3.2.1, all security related components need to be contained within the lockbox. Even if the device were within the lockbox and holes were made to allow it to transmit and receive sound pulses, the holes would pose a risk to both the system security and reliability, which is undesirable.

### 4.2.2.3 Inertial Measurement Unit

An Inertial Measurement Unit (IMU) is a device capable of measuring linear and angular motion. Basic inertial measurement units are essentially just self-contained systems containing both a gyroscope and accelerometer. Gyroscopes are devices that utilize gravity in order to determine orientation. The device's angular motion can be measured by monitoring the change in that orientation. Accelerometers are devices that measure dynamic acceleration. An example of one of the cheaper Inertial Measurement Unit modules available, the MPU-6050, is shown below in figure 4.2.2.3-1.



**Figure 4.2.2.3-1 Inertial Measurement Unit MPU-6050**

Unlike the other potential solutions, the use of an Inertial Measurement Unit could provide detailed data on the movement and orientation of the system without requiring that either the module be placed outside the enclosure or that an opening be made. The average power draw for the module shown above is approximately 3.5mA. Although the power draw would be higher than a tilt switch, it is still relatively low compared to the 15mA a rangefinder would require and can provide much more detailed data and greater control

over detection sensitivity than the other options explored. For these reasons it was decided that we would utilize an inertial measurement unit for this project.

### 4.2.3 Alarm

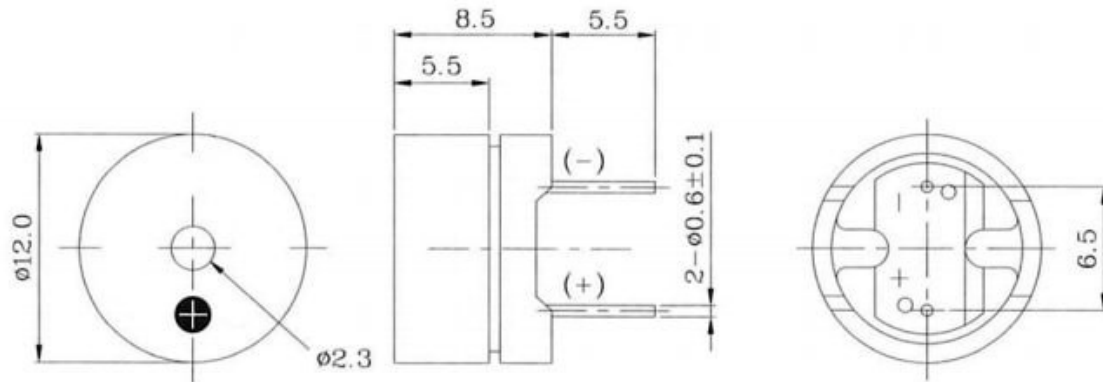
A major part of this design is the security aspect over other product choices out on the market currently. Most products don't have a system to let the user know in some manner if the container or device is being tampered with and using an alarm would be one key way of dealing with an at home alarm system feature for this product. There are a few different types of these components that can be chosen for this design, but they ultimately should serve the same purpose of making an audible noise to let the consumer know when at home that the box is being tampered with. This is only one additional solution to determine if theft is occurring behind the IMU sensor from the system movement detection Technologies in section 4.2.2. This component also would act as a deterrent to suspecting package thief's that try to steal from the box, but when hearing an alarm may stop and retreat from dealing with the container. A few different alarm components can be shown below in Table 4.2.3-1 and they are distinguished by sound level (in dB), voltage requirements, frequency, and price.

**Table 4.2.3-1 Comparison of Audible Alarms**

<b>Digi-Key Part Number:</b>	<b>Description:</b>	<b>Voltage Rated:</b>	<b>Current Supply:</b>	<b>Sound Level:</b>	<b>Frequency:</b>	<b>Pins:</b>	<b>Price:</b>
433-1055-ND	Audio Transducer	3V	1mA	70dB @ 3V	4kHz	1	\$0.75
668-1204-ND	Audio Indicator	3V	9mA	100dB @ 3V	3.5kHz	1	\$3.55
668-1503-ND	Audio Transducer	3V	3mA	103dB @ 1V	1kHz	1	\$4.70

The alarm components looked at from our research indicate a few variety of options for choosing an alarm. The alarm could be loud, but the price and current draw would be higher. The alarm could be low, and therefore the price and the decibel rating would be lower, so the alarm wouldn't be nearly as loud. One of better middle ground options for the alarms would for the device to emit a frequency, and decibel rating loud enough for the assailant and consumer to hear the product being tampered with but current supply required to be low enough that it wouldn't require too much draw from the power supply. The third option in Table 4.2.3-1, 668-1503-ND might be the best option due to the low voltage, current requirements and the high level of sound it will still emit from the device. All the

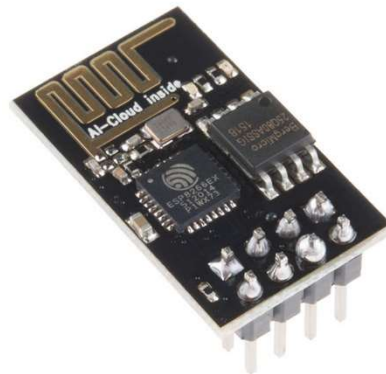
options require the same number of pins at one pin which is an acceptable amount for this component. Below in Figure 4.2.3-1 illustrates a typical alarm/buzzer dimensions and design that was used in the project.



**Figure 4.2.3-1 Audio Transducer**

### 4.2.4.1 Wi-Fi

A Wi-Fi module is a system on a chip (SOC) that is integrated with TCP/IP protocol stacks that can allow a cohesive system between a microcontroller and the network that it is sought to be integrated with. The module offloads networking functions and incorporates a use of feedback between the Wi-Fi network and the device. The Wi-Fi module being chosen for this specific project needs to be able to meet the designated design of the overall project without too many extra features to make the cost too large. This component is necessary for the security features that we plan to offer in our design and needs to be carefully researched to find the best alternative and most widely used option for Arduino boards with reliable recommendations. When looking for possible Wi-Fi modules the only real line of Wi-Fi modules that seem a good fit for this project are the ESP modules and primarily the ESP8266 model. When looking through multiple sites about these modules they are the most accessible and user friendly when it comes to implementing into Arduino builds as well as reliable. The picture below demonstrates what the ESP8266 module looks like in particular, but also in general for all types of different Wi-Fi modules.



**Figure 4.2.4.1-1 Wi-Fi Module ESP8266**

## 4.2.5 Keying Mechanism

The keying mechanism is the use of a keypad to transmit the user's inputted combination of numbers into a code the microcontroller can process for security release of the locking mechanism. This device complements the accessibility of keyboard functionality, but in the form of digits that can be transmitted as a sequence such as a password to be memorized. This allows the main design to have the security option of those only with the correct password access to the contents of the box. Through research of the different keypad switches a few different options become available for this design where matrix size, pin size, and price were the primary key factors when looking at components. Below in Table 4.2.5-1 we take a look at the different options of keypads we can select and their appropriate specifications that would affect the design of our project.

**Table 4.2.5-1 Comparison of Keypad Switches**

<b>Digi-Key Part Number:</b>	<b>Description:</b>	<b>Matrix Size:</b>	<b>Rating At 12 Vdc:</b>	<b>Pins:</b>	<b>Price:</b>
96AC2-102-F-ND	Switch Keypad	3x4	5 mA for 0.5s	7	\$14.23
1528-2161-ND	Switch Keypad	3x4	5mA for 0.5s	7	\$7.50
96BB2-056-R	Switch Keypad	4x4	5mA for 0.5s	8	\$19.90

The overall selection process for the differing keypad switches result shown from above in Table 4.2.5-1 shows the voltage rating for each of the different options and they all use the same 5mA for 0.5s of contact on the keypad which should not be an issue for our specific design, but due to the large pin count on each of these keypad's we will need to use a multiplexer device to convert the large amount of pins into a feasible quantity that can be transmitted to the microcontroller. The figure below, Figure 4.2.5-1, shows the general design of the keypad's looked at from above, but not the exact replica.



**Figure 4.2.5-1 Keypad Switch**

## 4.2.6 Locking Mechanism

The locking mechanism options revolve around the use of a typical mechanical latch being pulled by a servo motor connected to the power supply and Arduino board for programming. The easiest option for our purpose and only real solution to our design constraints was to take this approach of requiring a non-intrusive electronic configuration that utilizes a mechanical component without having to use any mechanical engineering. There were a few servo options that were looked at in the research phase of this project. The overall design focuses on a low power draw servo that can be compatible with the design idea we have in mind of using a sweeping motion to pull the latch out. The latch will also need to in a strung state using strong rubber to pull the latch back to its previously locked state. The other alternative is to have the servo motor directly be utilized as a latch that clips into the lid of the box when locked and unclip when unlocked so the lid can be lifted. Here are the following servo motors looked at for this design of this project in Table 4.2.6-1:

**Table 4.2.6-1 Comparison of Servo Motors**

<b>Digi-Key Part Number:</b>	<b>Description:</b>	<b>Voltage Required:</b>	<b>Current Draw:</b>	<b>Torque:</b>	<b>Price:</b>
1738-1300-ND	Micro-Servo	4.8VDC~6VDC	100mA~200mA	1.2~1.6Kg x cm	\$4.12
900-00005-ND	Micro-Servo	6VDC	140 +/- 50mA	2.74Kg x cm	\$13.95
1568-1320-ND	Micro-Servo	6VDC	100mA~200mA	1.19~1.50Kg x cm	\$8.95

When looking at the alternative options for the micro-servos that can be utilized in the above table we can see cost isn't too expensive for any reasonable option we make going forward as well as current draw or voltage. The design we have in mind and the power supply we plan on using will be able to account for the servo's draw. The primary focus on this component is to get the cheapest alternative without losing too much torque for the product. The option 900-00005-ND offers a large torque threshold but at a greater cost. The final decision on the which servo to be used will have to be carefully looked at with correspondence to the locking mechanism we choose to use.

## **4.2.7 Power**

Per requirements P1 and P2 detailed in Section 2.3.2.2, the system must be able to run on 120V and 60 Hz AC power from a standard US wall outlet and continue operation for a minimum of 24 hours on a backup rechargeable secondary power source in the case of primary power loss. The below sections describe the research of performed related to meeting these requirements.

### **4.2.7.1 AC/DC Conversion**

The most important portion for our project will be AC/DC conversion, or the transfer of power to the device through any means. When looking at differing design choices the most optimal solution was to directly connect the design of our system to the wall jack on the outside of a house. The overall layout and construction of our product requires a steady stream of power at a high rate, and the only solution that seems plausible for the project to maintain this current draw for all its components is through the use of an AC/DC adaptor.

### **4.2.7.2 Battery**

The backup battery is an important aspect to our design in case the project itself loses power from its main source. In this case, a backup battery is needed for use with the design to keep the design stable long enough for the user to be verified that the power source has been unplugged or damaged. The main focus points our project is looking for when deciding a backup battery choice are as follow:

- Low cost
- Decent/High capacity
- Form factor (Small enough to be utilized efficiently)
- Low discharge loss
- Long battery life

When looking for possible backup batteries the two main battery types that were looked at were rechargeable batteries called Lithium-Ion and Lithium-Polymer. Both battery types offer similar capabilities our design wants to be able to rely on like good capacity, reliable sustainability for long term use, and compliance with normal temperature ranges. When looking at both battery types it became apparent the option our specific design would work

with better would be the Lithium-Ion battery. This is in part due to the similar capabilities as the Lithium-Polymer batteries, but with a better energy density than the latter. Also, due to the nature of Lithium-Polymer batteries the form factor that makes the battery type so beneficial isn't necessary for our design purposes with respect to the increase in cost to capacity ratio that we need to achieve for our design. One such picture of a possible Lithium-Ion battery we are deciding to choose is the uxcell Power Supply DC 3.7V 2500mAh shown below in Figure 4.2.7.2-1.



**Figure 4.2.7-1 uxcell Power Supply DC 3.7V 2500mAh**

## 4.2.8 Enclosure

The main purpose of the enclosure section is to make sure to get a container that will fit the most delivered package sizes, but also have enough security so that it can't be broken into easily with various tools. With a range of 4 to 12 cubic feet for any container it should be appropriately sized to fit most common packages. The container needs to be looked at with cost and size in mind as well as accessibility for the user. When looking at possible containers that fit the requirements of most packages being able to be stored the following container was noticed to be the most appropriate for the requirements of our design below in Figure 4.2.8-1, known as the Keter 55 Gallon Outdoor Storage Cube.





**Figure 4.2.8-1 Keter 55 Gallon Outdoor Storage Cube**

This product works nicely with what is needed for our design due to the small nature of the container to not take up too much space on a consumer's door, but still fit the average package firmly within the confines of the container. The design of this product also allows a seamless lid to frame connection, so the lid of the container is less prone to removal by means of tools such as a crowbar. The grooves within the lid and sides of the container also will allow easy placement of components where needed. The image above in Figure 5.9 also demonstrates the hinges on the left and right showing they are non-intrusive to possible component place as well as the density of the box with the review photo showing a generator of a decent size placed into the box with plenty of space remaining. There are alternatives that may be chosen in the part selection phase depending on process of elimination due to cost and size such as another popular and cheap container such as the Suncast SSW1200.

## 4.2.9 Microcontroller

The microcontroller is the core of the Smart Package Lockbox system design. As can be seen in Figure 2.5-1, all system input and output passes through the microcontroller. It will read data from the sensors and keying mechanism, control the alarm and locking mechanism, and communicate with the web server through the Wi-Fi module. The following considerations were made while researching potential microcontrollers for the Smart Package Lockbox system:

- Cost
- Power consumption
- Number of general purpose input/output pins
- Operating clock frequency

- Operating voltage
- Flash Memory
- Ease of coding

Per requirement C2 detailed in Section 2.3.2.5, the system needs to be low cost. The power consumption also needs to be kept low in order to increase the secondary power source battery life per requirement P2 detailed in section 2.3.2.2. The selected microcontroller needs to have enough general-purpose input/output pins to accommodate all of the other subsystems and have enough processing speed to process the system input/output data. The operating voltage should be close the voltage used by the various system subassemblies to avoid the need for additional voltage regulators which would lower system power efficiency and the secondary power source battery life. Due to time constraints, it is also desirable that the selected microcontroller be user friendly and have a specialized integrated development environment as well as community support which would increase ease of coding. Development board availability and cost are also taken into account for prototyping and testability purposes. The following subsections detail the microcontrollers evaluated based on these criteria.

### 4.2.9.1 MSP430

Texas Instruments offers many microcontroller lines, including the popular MSP430 lines. The TI MSP430 microcontrollers have a considerable amount of supporting documentation available. The MSP430 are also the microcontrollers used for academic purposes in the embedded systems course at the University of Central Florida that is required for electrical and computer engineering students, meaning that all the members of group 5 have experience with utilizing this microcontroller for various tasks. An integrated development environment that can simplify coding the MSP430, Energia, is also available. The specifications for the MSP430G2553 from the ultra-low-power series value line of TI microcontrollers are shown below in Table 4.2.9.1-1.

**Table 4.2.9.1-1 MSP430G2553 Specifications**

Specifications	
Clock Speed	16 MHz
Flash Memory	16 KB
RAM Memory	512 B
Operating Voltage	1.8 – 3.6V
Power Draw	420 uA at 1 MHz
Number of GPIO Pins	24
Cost	\$2.46
Development Board	MSP-EXP430G2
Development Board Cost	\$10.35

As can be seen in the table above, this microcontroller and the associated development board are both low cost. The number of general purpose input/output pins available are sufficient for the purposes of this project. There are multiple ultra-low-power modes available and an average power draw of 420 uA at 1MHz, however, the microcontroller can be configured to run at up to 16MHz if needed. Although the flash memory is sufficient, the RAM memory is low and may require code optimization.

### 4.2.9.2 ATmega2560

Even more so than the MSP430 lines, microcontrollers used in Arduino boards are particularly popular, especially for use in do it yourself (DIY) hobbyist projects. This is likely in-part due to the fact that Arduino is an open source platform with considerable community support and resources available for microcontrollers utilizing this platform. This includes an integrated development environment that is known for greatly simplifying the coding process. This is a highly desirable inclusion in the case of this project because of time constraints. Due to this high availability of community support and resources, the microcontrollers used in Arduino boards were researched as a potential option. The specifications for a mid-range microcontroller using the Arduino platform, the ATmega2560, can be seen below in Figure 4.2.9.2-1.

**Table 4.2.9.2-1 ATmega2560 Specifications**

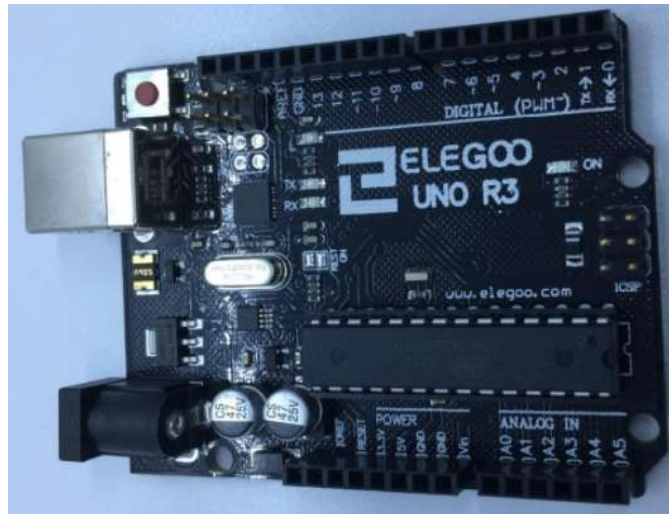
Specifications	
Clock Speed	16 MHz
Flash Memory	256 KB
RAM Memory	8 KB
Operating Voltage	4.5 – 5.5V
Power Draw	500 uA at 1 MHz
Number of GPIO Pins	86
Cost	\$12.35
Development Board	AVR ARDUINO MEGA2560
Development Board Cost	\$38.50

As can be seen in the table above, the specifications are mostly similar or superior to the previously explored MSP430 line option. The number of general purpose input/output pins available are not only sufficient for the purposes of this project, but also provide ample

room for expansion. There are multiple sleep modes available and an average power draw of 500 uA at 1MHz. In a similar manner as the MSP430G2553, however, this microcontroller can also be configured to run at up to 16MHz if needed. Both the flash memory and RAM memory are more than sufficient for the purposes of this project and it would be unlikely that code optimization would be required. However, the increased cost of the microcontroller and development board required for prototyping and testing purposes compared to the previous option is undesirable. That being said, the increased costs are not to the point of being prohibitively high and the ATmega2560 is still a potential option if increased general purpose input/output pin availability or more memory is required as the initial project planning progresses.

### 4.2.9.3 ATmega328p

The ATmega328P can be viewed as a smaller and less expensive version of the previously explored ATmega2560. In addition, it is the microcontroller used in some of the most popular Arduino boards, including the Arduino UNO; a board that is commonly cited as one of the best beginner boards available. An example of the Arduino UNO is shown below in Figure 4.2.9.3-1.



**Figure 4.2.9.3-1 Arduino UNO**

The immense popularity of this particular Arduino board means that there is even more community support and resources available for it than other Arduino platform boards. In addition to the availability of many code examples and projects for it on the internet, the majority of tutorials posted online are for boards using this particular microcontroller. The fact that Arduino is open source and the availability of resources also open up the option of using one of the Arduino boards that utilizes an ATmega328P microcontroller, such as the Arduino UNO, as a baseline starting point when developing the PCB for this project. The specifications for this microcontroller are shown in Table 4.2.9.3-1.

**Table 4.2.9.3-1 ATmega328P Specifications**

Specifications	
Clock Speed	16 MHz
Flash Memory	32 KB
RAM Memory	2 KB
Operating Voltage	1.8 – 5.5V
Power Draw	200uA at 1 MHz
Number of GPIO Pins	23
Cost	\$2.17
Development Board	Arduino UNO R3
Development Board Cost	\$11.86

As can be seen in the table above, the specifications, although not as high as the ATmega2560, are still sufficient for the purposes of this project and come with a lower cost. The lower cost and increased availability of resources make the ATmega328P an enticing option for this project. The number of general purpose input/output pins available are sufficient and the average power draw of 200uA at 1MHz is better than with any of the previously explored options. This microcontroller can be configured to run at up to 20 MHz if needed, although 16 MHz is considered typical for error free operation and would still be sufficient for the purposes of this project. Both the flash memory and RAM memory are also sufficient and it is unlikely that code optimization would be required. The low cost of both the ATmega328P and the Arduino UNO also allows for ordering multiples of each for testing and prototyping purposes with minimal effect on the project budget. Figure 4.2.9.2-1 below shows the pin mapping for the DIP version of the ATmega328P.

PCINT14/RESET) PC6	1	28	PC5 (ADC5/SCL/PCINT13)
(PCINT16/RXD) PD0	2	27	PC4 (ADC4/SDA/PCINT12)
(PCINT17/TXD) PD1	3	26	PC3 (ADC3/PCINT11)
(PCINT18/INT0) PD2	4	25	PC2 (ADC2/PCINT10)
(PCINT19/OC2B/INT1) PD3	5	24	PC1 (ADC1/PCINT9)
(PCINT20/XCK/T0) PD4	6	23	PC0 (ADC0/PCINT8)
Vcc	7	ATmega22	GND
GND	8	28PDIP	21
(PCINT6/XTAL1/TOSC1) PB6	9	20	AVCC
(PCINT7/XTAL2/TOSC2) PB7	10	19	PB5 (SCK/PCINT5)
(PCINT21/OC0B/T1) PD5	11	18	PB4 (MISO/PCINT4)
(PCINT22/OC0A/AIN0) PD6	12	17	PB3 (MOSI/OC2A/PCINT3)
(PCINT23/AIN1) PD7	13	16	PB2 (SS/OC1B/PCINT2)
(PCINT0/CLKO/ICP1) PB0	14	15	PB1 (OC1A/PCINT1)

**Figure 4.2.9.3-2 ATmega328P Pin Mapping (Courtesy of Wikimedia)**

## 4.3 Software

Section 4.3, Software, will contain research subsections that are relevant to our particular design with compatibility with our specific microcontroller and network. The subsections will consist of looking into microcontroller compatibility, web communications & hosting, and mobile device application. The primary focus of this subsection with the Research section is to look at the individual applications we can instill into our final product and what will work best meeting our requirements and fundamental purpose for the end result design to have.

### UART

UART or Universal Asynchronous Receiver-Transmitter serves to transmit and receive serial data. It is a peer-to-peer asynchronous protocol which connects two devices via two signal lines. UART serves as an intermediate stage used to connect two devices. This is done by shifting out a bit sequence from a register and transmitting it bit by bit across the signal lines. The bit rate can be adjusted and must be within a safe limit for both transmitter and receiver or the bits may become corrupted. The receiver will then shift the data bit by bit into its own registers. Unlike in an I2C setup one device can act as both a receiver and a transmitter in the same system and does not have a set role. UART is a popular form of communication protocol in DIY electronics, GPS modules, Bluetooth modules, and RFID modules.

## 4.3.1 Microcontroller

When it comes to the software programming of the ATmega328P microcontroller, there seem to really only be a couple of options for the programmer. A more technical programming tool called AVRDUDE, and the Arduino IDE.

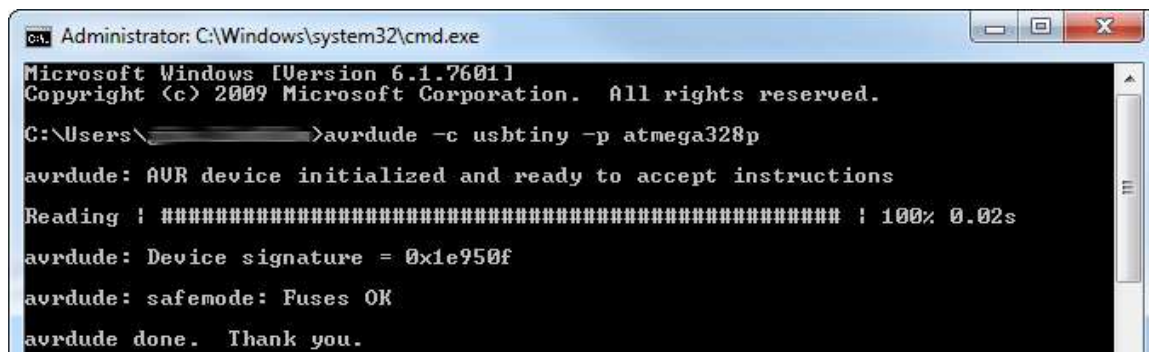
### 4.3.1.1 AVRDUDE Command Line Program

AVRDUDE was developed by Atmel at the commencement of 1996 to use in its family of AVR microcontrollers. At the time, these microcontrollers were pretty special because they were some of the first to use on-chip flash memory to store program code, as opposed to all other microcontrollers at the time, that used ROM, EPROM, or EEPROM. This was an enormous advantage when compared to ROM that could only be written to once.

AVRDUDE is a command-line program and is really targeted at programmers that are a bit more technical. Some programmers really enjoy using this software because it gives them more control over the microcontroller. It requires the use of the command prompt on Windows, or Terminal on Mac and Linux. An example of a command that could be used to test whether AVRDUDE is working and communicating with the ATmega328p microcontroller would look something like this:

```
avrdude -c usbtiny -p atmega328p
```

By typing this in correctly, and if the microcontroller is properly communicating with AVRDUDE, the command prompt should return something that looks like what is shown on Figure 4.3.1.1-1.



```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\>avrdude -c usbtiny -p atmega328p
avrdude: AVR device initialized and ready to accept instructions
Reading : ##### : 100% 0.02s
avrdude: Device signature = 0x1e950f
avrdude: safemode: Fuses OK
avrdude done. Thank you.
```

**Figure 4.3.1.1-1 AVRDUDE Command Example (CC BY-SA 4.0 License)**

### 4.3.1.2 Arduino IDE

The Arduino Software IDE is an open-source environment that is written in Java and based on Processing and other open-source software. It runs on all three major operating systems Mac, Linux, and Windows. It has many features including, syntax highlighting, brace matching, and automatic indenting to name a few. It supports both the C, and C++ programming languages, but with both languages having to use a special set of rules for code structuring. The Arduino IDE actually uses the previously talked about program AVRDUDE in the background to convert executable code into a text file that contains hexadecimal encoding which is then loaded onto the Arduino board by the board's firmware using a loader program. Just the fact that Arduino IDE makes use of the AVRDUDE command-line program lets us know that Arduino IDE was made in order to help simplify a lot of the technicalities that are required to use the AVRDUDE software, hence the lack of a need to use the command line prompt.

### 4.3.2 Web Communications & Hosting

This section discusses the web communication and hosting details for the project. There are a few options to consider such as which stack to use and where to host the project. With price, availability, and ease of use to consider as factors.

#### 4.3.2.1 Web Communication

**JSON** - Most databases only recognize text. Due to this pure JavaScript and the objects created cannot be sent directly as they are to the database. In order to communicate one of the common methods is using JSON (JavaScript Object Notation). JSON converts the data contained in the forms of the website or app into text in a format the database will recognize and vice versa.

**XML** - Extensible Markup Language (XML) is another alternative to JSON. Overall, they both do the same thing only XML is a bit more robust and less compact than JSON. Due to the fact XML is larger and uses a specialized and more strict format often it requires the use of an XML parser.

**XML vs JSON** - Due to the fact that JSON is often shorter and does not require the use of an XML parser often times JSON is the go to. Because of these reasons and with prior experience with JSON, JSON will be our method of communication between the database and user.

**AJAX** - Asynchronous JavaScript XML (AJAX) is used in order to update web pages asynchronously as opposed to a synchronous option. This allows for a webpage to be updated on the fly without the page needing to be reloaded by the user. Traditionally on a form when you hit submit JavaScript would make a request to the server wait for a response



and direct you to a new page with the returned information. With AJAX upon hitting submit the information would be transmitted to the server, interpret the response, followed by updating the current page for the user. This allows for the page to not have to be refreshed or reloaded.

### 4.3.2.2 Hosting Options

This section looks at the possible hosting options available and considered for the project. Among them several free and paid options are considered. The advantages and disadvantages of each are discussed as well as how they may impact the project as a whole. The budget as well as the potential gains are the main concern.

**Apache** - Apache is a free, open source hosting server with no licensing fees.

Advantages

- Flexibility in choosing various modules when setting up hosting
- Enhanced security options
- Strong user-community support
- Resources to aid users
- Runs on UNIX, Windows, Linux, Mac OS

Disadvantages

- Process based server, this requires each connection to have a thread which leads to more overhead

Popular sites using Apache server

- Wikipedia
- PayPal
- Apple
- Huffington Post

**Microsoft IIS** - Microsoft Internet Information Services is a server option that is included with Windows. Costs are included with licensing and certain options.

Advantages

- Continual support from Microsoft
- Provides support for .NET framework and ASPX scripts
- Easy integration with other Microsoft services (MS SQL Server, ASP, ect.)

Disadvantages

- Not as customizable or versatile
- Cost with licensing

Popular sites using Microsoft IIS

- Live.com
- Bing.com
- Microsoft
- MSN

#### 4.3.2.3 Connection Options:

**Bluetooth** – Bluetooth was first introduced in 1994 so therefore it is a relatively mature technology. Bluetooth is used in many mobile devices such as phones, mice, keyboards, and portable speakers. Standard Bluetooth operates at 2.4 GHz and uses fairly low power consumption when compared to other methods of wireless communication. In order to two devices to communicate using Bluetooth both must have a Bluetooth adaptor connected.

**Wi-Fi** – Wi-Fi was first introduced in 1991 and is also a mature technology. It has undergone many revisions and is currently operating under the IEEE 802.11 standard. Wi-Fi serves to set up a wireless local area network (WLAN) in order to allow communication between devices. These devices may communicate in either the 2.4, 3.6, 5, or 60 GHz frequency bands. Wi-Fi serves not only to communicate between devices as Bluetooth does but also connect them. Wi-Fi acts as a wireless ethernet cable in the sense that it allows devices to access the internet. Wi-Fi is a common feature on many mobile devices such as cell phones and laptops. It is also used in less mobile devices especially home devices. With advancements in wireless technology the number of wireless devices has radically increased. Wi-Fi is used in desktops, smart TVs, wireless cameras, and other smart home devices. In order for all these devices to communicate and be connected Wi-Fi takes a bit more setup compared to Bluetooth. However, the signal strength and range is quite higher. With this boost in power comes a larger power consumption in most cases.

**Zigbee** – Zigbee was developed by the Zigbee Alliance in 2004. Zigbee is an IEEE 802.15.4 based specification for high-level communication protocols. It was designed to be simpler and less expensive than other wireless personal area networks (WPANs) such as Bluetooth and Wi-Fi. Zigbee devices operate in the 868 MHz, 915 MHz, and 2.4 GHz frequency bands. The maximum data transfer rate is 250 Kbits per second. This makes Zigbee ideal for more basic devices such as smart home devices. Due to the low data speed and with support for a sleep mode on most devices Zigbee is integrated with the power consumption is extremely low. Zigbee was also designed with ease-of-use in mind and most devices can operate without the need to connect them directly to a router. This is done by each device containing built in wireless capability and usually communicates to a hub device. This hub device in turn is connected to the wireless network and from there communication can be made to all the devices. This allows users to install multiple lightbulbs and only configure 1 hub on their home network rather than connecting each lightbulb to the network individually.

A comparison of all three connection types can be seen in the table 4.3.2.3-1 below.

**Table 4.3.2.3-1 Wireless Connection Comparison**

<b>Method of Communication</b>	<b>Wi-Fi</b>	<b>Bluetooth</b>	<b>Zigbee</b>
Power Consumption	High	Low	Low
Signal Range	300 Feet or More	~30 Feet	Up to 300 Feet
Number of Connected Devices	Many	~ 7 on Most Devices	Many Using a Hub Device
Data Speeds	Up to 600 Mb/s	2.1 Mb/s	250 kb/s
Ease of Use	Moderate	Easy	Easy
Requires Adaptor	Yes	Yes	Yes
Use	Mobile and Home	Mobile and Home	Home

### 4.3.3 Mobile Device Application

A mobile device application will be developed for use by the consumers of the Smart Package Lockbox. From the app, we expect to achieve most of the functionality available in the web version. However, there may be a few differences and restrictions while using the app.

#### 4.3.3.1 Choosing a Development Platform

When it comes to developing a mobile device application the first design choice is made between making a web application versus a native application.

Web App Pros:

- Common code base
- Easier to maintain
- Multiple mobile platforms
- Adaptable to older devices
- No app download required
- No submission to any app store for approval

Web App Cons:

- Limited scope when accessing mobile device features
- Multiple mobile browser support can be expensive
- Difficult to maintain record of user usage patterns
- Not usually listed in app stores
- No quality control system
- Users not always guaranteed safety and security
- Slower performance

#### Native App Pros:

- Support for devices built in features
- Faster performance
- Full support from app store or market place
- App store approval brings assured safety and security to users
- SDK and other tools provided for developers
- Easy to develop

#### Native App Cons:

- More expensive for developers
- Cost of app maintenance is higher
- Sometimes difficult to support more than one mobile platform
- App store approval process is sometimes long and unsuccessful
- Providing support can be difficult for users using different versions of app

The second design choice is based on whether the team chose to make a native application. Usually when programmers decide to develop a native application for a mobile device, there are realistically only two ecosystems to aim for when they have to make a choice. The Android ecosystem or the Apple ecosystem. There is an endless debate on whether one is superior to the other, but in total honesty, both ecosystems are exceptional in their own respects and by no means the wrong choice.

### **4.3.3.2 Android OS Platform & IDE**

Choosing the Android ecosystem for the development of our mobile application would seem like the most obvious choice when taking into account that the global market share for the Android mobile operating system is sitting at 74.24% as of March 2018 according to GlobalStats' statcounter website. This would technically mean that by developing the mobile application on the Android ecosystem we would reach the highest number of customers or users of our Smart Package Lockbox. Taking only the global market share into account when making a decision would be a mistake though, since the Smart Package Lockbox is initially being developed and prototyped in the United States. When looking at the United States market share of Android, it is only sitting at 43.24%. If we were to select the Android ecosystem we would be cutting off more than half of potential users.

When developing applications for the Android ecosystem, the primary integrated development environment (IDE) used to be Eclipse in conjunction with Android Development Tools (ADT) provided by Google for native Android application development. In December 2014, Google released the first stable build of what was going to be the replacement of Eclipse as the primary IDE for Android development. Ever since, the primary and official IDE for Android application development is Android Studio. The system requirements to run Android Studio are described in Table 4.3.3.2-1.

**Table 4.3.3.2-1 Android Studio System Requirements**

Recommended Specs	
<b>OS Version</b>	Microsoft® Windows® 7/8/10 (32-bit or 64-bit) Mac® OS X® 10.10 (Yosemite) or higher, up to 10.13 (macOS High Sierra) GNOME or KDE desktop Linux (64 bit capable of running 32-bit applications) (GNU C Library (glibc) 2.19+)
<b>RAM</b>	3 GB RAM minimum, 8 GB RAM recommended; plus 1 GB for the Android Emulator
<b>Disk Space</b>	2 GB of available disk space minimum, 4 GB recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
<b>Java Version</b>	Java Development Kit (JDK) 8
<b>Screen Resolution</b>	1280×800 minimum screen resolution

Android Studio IDE is compatible with all three major operating systems, Windows, macOS, and Linux. It is built using JetBrains' (Czech company) IntelliJ IDEA software, so by extension, it also supports the same programming languages that IntelliJ supports. Even though it supports multiple programming languages, for the purposes of building and creating Android applications, Java is the primary programming language that is used.

### 4.3.3.3 Apple iOS Platform & IDE

After analyzing the Android ecosystem and taking into account the IDE, programming language used for development, the system requirements, and the market share of the Android operating system in the United States as well as globally, the Apple ecosystem paints a completely different picture. The market share for mobile operating systems is essentially split between Apple's iOS platform and Google's Android OS. The remaining mobile operating systems combined don't even make up 4% in the global market share, and in the U.S., their market share is even less impressive at just under half of 1%

combined. Therefore, it's safe to say that these other mobile OS platforms are negligible in regards to this project. In the global front, Apple's iOS platform market share is just above 20%, coming in at 20.83%, which seems alarmingly small compared to Google's Android OS' 74.24% of dominance. While the iOS platform's global market share is relatively slim, its U.S. market share is significantly larger at 56.26% compared to Android OS' 43.24%. Even though Apple's ecosystem takes the prize in U.S. market share, which is technically our initial relevant market, 56% is still leaving nearly half of all potential users without access to the mobile application.

Like Google's Android Studio IDE for Android OS development, Apple's iOS Platform also has its own dedicated IDE for application development called Xcode. Apple first released Xcode IDE back in 2003, but obviously mobile devices like we know them today didn't show up until the late 2000's. Xcode supports an extensively large number of programming languages to code in and is not limited only to the iOS platform. During the Apple World Wide Developers Conference in June of 2014, Apple released the 6<sup>th</sup> iteration of Xcode which included included a new programming language called Swift. Prior to Swift's introduction in Xcode 6 most if not all iOS development was done using Objective-C programming language. At this point, Objective-C can still be used to develop for iOS devices, but as of now, almost the entirety of iOS platform development is done using the latest iteration of the Swift programming language, Swift 4.

One of the downsides of developing for the iOS platform is the fact that it lacks support for multiple operating systems. As of now, the only operating system supported is macOS, therefore a Mac is required to develop any sort of mobile application for an iOS device. The recommended system requirements to run Xcode are detailed on Table 4.3.3.3-1 below.

**Table 4.3.3.3-1 Xcode 9.3 (iOS SDK) System Requirements**

<b>Recommended Specs</b>	
<b>OS Version</b>	macOS® High Sierra 10.3.2
<b>RAM</b>	4 GB RAM minimum, 8 GB RAM recommended
<b>Disk Space</b>	25 GB of available disk space minimum 128 GB of available disk space recommended
<b>Programming Languages</b>	Objective-C, Swift 4
<b>CPU</b>	Intel 6 <sup>th</sup> Generation i5 or i7 recommended

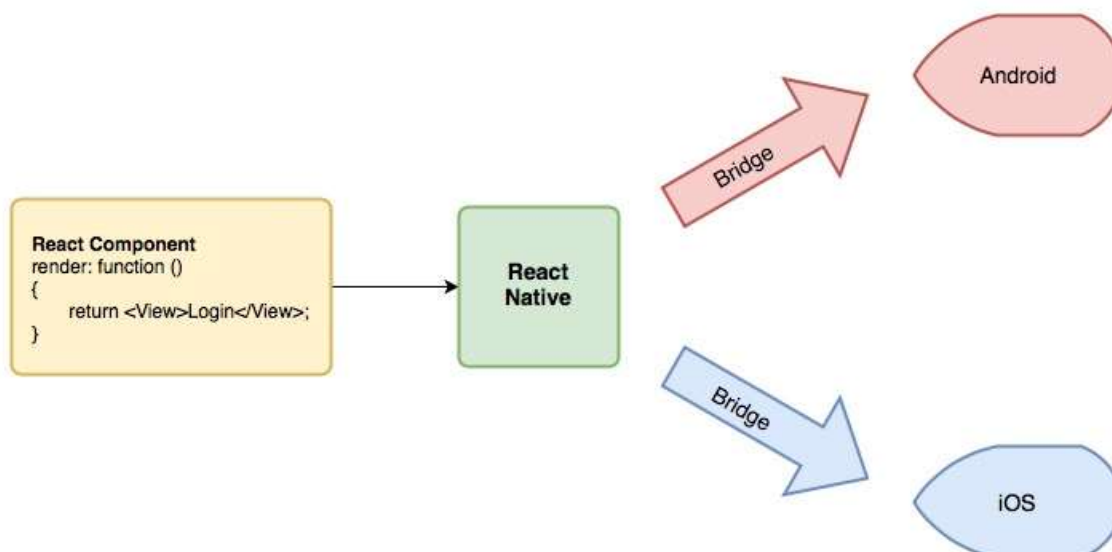
### 4.3.3.4 Cross-Platform Development & IDE

As it turns out, there is a third option that doesn't require choosing between any of the ecosystems that were mentioned in the previous two sections. When researching for options to see if it was possible to develop for both the Android and Apple ecosystems at the same time, we stumbled upon a JavaScript library for building user interfaces that is referred to as React, or sometimes ReactJS or React.js. This JavaScript library is maintained primarily by Facebook and Instagram with a community of individual developers and other corporations that help contribute to its development.

During Facebook's React.js Conference in February of 2015, they announced React Native, which is a part of the React framework that enables native iOS, Android, and Universal Windows Platform development. This enables cross-platform development by applying the React architecture to native Android, iOS, and UWP applications. React Native offers several advantages to the developer experience, because it's basically programming in JavaScript, CSS, and HTML like developing a standard web application, but without the cons that web app development brings to mobile devices. React Native gives the developer access to native performance and, animations, and behaviors on your mobile device without having to write any Objective-C, Swift, or Java. Another advantage is that the application will be mostly written in JavaScript, and like in web apps, this allows developers to instantly "refresh" the applications in order to see any code changes. This is an important advantage because it can potentially save a lot of time for the developers of the application. Instead of having to rebuild your mobile application on Xcode or Android Studio and send it to the device every time you change something in the code, you would just hit refresh.

Even though it is possible to write an application with React Native and not use any platform-specific code, there's still the possibility that if you want to take advantage of a certain platform-specific API, you will indeed need to write some platform-specific code. React Native makes this rather easy as well by allowing the developer to specify platform-specific versions of each component, which can then be integrated into the React Native application. One of the few limitations of React Native is that Apple will force you to use an Apple computer if you want to make a React Native app that also works on iOS.

React Native's core differentiator from other cross-platforms is that React has its own components describe their own appearance, and then React handles the rendering of those components for you. These two functions are separated by an abstraction layer. Just how React uses standard HTML tags to render components for the web, this same abstraction layer or "bridge" allows React Native to invoke the actual rendering API's on iOS and Android. The following Figure 4.3.3.4-1 describes the flow of how this happens.



**Figure 4.3.3.4-1 React Native Bridge**

The system requirements for React Native depends entirely on the IDE that the programmer or developer decides to use. Most of the IDE's that are compatible with React Native support all three major operating systems, Windows, Linux, and macOS. For our project's purposes, the atom.io IDE seems to be the best fit. Table 4.3.3.4-1 shows some of the features and requirements of the atom.io IDE compared to several others.

**Table 4.3.3.4-1 React Native IDE Comparison**

IDE	Supported Platforms	License	Features
<b>Atom.io</b>	Windows, Linux, Mac	Open-Source	<ul style="list-style-type: none"> <li>• Cross-Platform editing</li> <li>• Built-in package manager</li> <li>• Smart auto-completion</li> <li>• File System Browser</li> <li>• Multiple Panes</li> </ul>
<b>Nuclide</b>	Windows, Linux, Mac	Open-Source	<ul style="list-style-type: none"> <li>• Built-in Debugging</li> <li>• Remote Development</li> <li>• Developing Hack</li> <li>• Mercurial Support</li> <li>• Working Sets</li> </ul>
<b>Deco IDE</b>	Mac (only for iOS)	Open-Source	<ul style="list-style-type: none"> <li>• Component Search and Insert</li> <li>• Real-Time Tweaking</li> <li>• New File Scaffolds</li> </ul>



Atom.io IDE has some very interesting packages that can be added on as plugins to the IDE. Since Atom is used widely for an extensive variety of all major technologies, it has a huge community that helps by contributing enhancements and plugins for almost anything. One of these add-ons or plug-ins that atom uses is called atom-xcode, and hence the name, it helps bridge the gap between Xcode and Atom so that once the plug-in is installed, Xcode's iOS simulator can be controlled from within the Atom application itself.

## 5.0 Part Selection and Acquisition Summary

The following sections serve as a summary of the decisions related to the initial selection of system components and suppliers used. Most parts are narrowed down to one or several viable options to be considered for the project.

### 5.2 Part Selection

The reasoning behind the part selection for the major components used in the Smart Package Lockbox system are detailed herein. It should be noted, however, that this list is not meant to be all-inclusive. Some of the design decisions for minor component selection are stated as needed in the hardware design section. A complete list of specific components purchased can be found in the bill of materials found in Section 10. Throughout the selection process of the components in this section we established certain previous selections were not best suited for our design and went back and gathered a different part to replace them.

#### 5.2.1 Lid Sensor Selection

A magnetic switch version of the SL353HT product manufactured by Honeywell Sensing and Productivity Solutions was the lid sensor detection device that was initially selected for this project. The SL353HT met all project requirements for this experiment as well as giving the most optimal current requirement out of the other opposing magnetic switch options. The SL353HT is frequently used in many Arduino projects as well as the magnetic switch option in general for detection purposes.

However, the previous selection did not end up making the final selection phase due to ease of use and construction. The Gikfun MC-38 Wired Door Sensor Magnetic Switch ended up being our final design choice due to the already connected wire and both components of the sensor being delivered upon purchase. The construction and application of this switch versus the other one allowed a less cumbersome process of applying a switch to our design. The cost of this component was \$5.66. Figure 5.2.1-1 below shows the magnetic switch we decided to use in our project. The reasons below are why we selected this magnetic switch:

- Meets the project requirement of lid movement detection

- Low current strain on the system
- Reliability versus alternatives previously looked at
- Ease of construction
- All required components in one package



**Figure 5.2.1-1 Gikfun MC-38 Wired Door Sensor Magnetic Switch**

## 5.2.2 Inertial Measurement Unit Selection

A surface mount version of the MPU-6050, as seen in Figure 4.2.2.3-1, was selected for this project. The MPU-6050 met all of the project requirements and is commonly used in Arduino projects with the selected microprocessor for this project and has a large amount of community support, including written tutorials and example code. The module cost from Digi-Key was \$8.29, however, the full GY-521 MPU-6050 IMU module board from Amazon was \$4.68. The MPU-6050 was primarily selected for the following reasons:

- Meets the project needs for movement detection
- Low Cost
- High community support

## 5.2.3 Alarm Selection

An alarm component version of the AT-1810-T-LW50-R is an audio piezo transducer that was selected for this design. The AT-1810-T-LW50-R meets all of the product requirements and is commonly used in Arduino projects with basic designs on alarm applications and compatibility with multiple Arduino boards. The component cost on Digi-Key is \$4.70, which was one of the higher component cost selections.

However, the previous component listed above had an issue with producing a viable product in time for our project and we had to switch to a different option that also had faster shipping. We chose the 5V Active Alarm Buzzer from Amazon at \$8.99 for a pack of 12 alarms which was our best option while also getting the required decibel rating we needed to maintain the functionality of our project. At a decibel rating of 85dB we met the requirements of our projects scope as well as having extra alarms for testing and backup purposes. This particular alarm was primarily selected for the following reasons:

- Meets the project requirements for security notification
- Has a large decibel count for loud noise
- Uses minimal current in the overall project
- Easy application

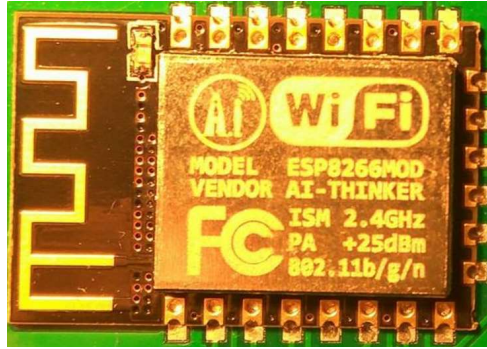


**Figure 5.2.3-1 5V Active Alarm Buzzer**

## 5.2.4 Wi-Fi Module Selection

A surface mount version of the popular ESP8266, as seen below in Figure 5.2.4-1, was selected to supply Wi-Fi communications for the Smart Package Lockbox system. The ESP8266 is commonly used in Arduino projects with the selected microcontroller for this project and has a large amount of community support, including written tutorials and example code. This module was available on Amazon in a pack of 2 for \$10. The ESP8266 was selected primarily for the following reasons:

- Meets the project requirements
- Low Cost
- High community support

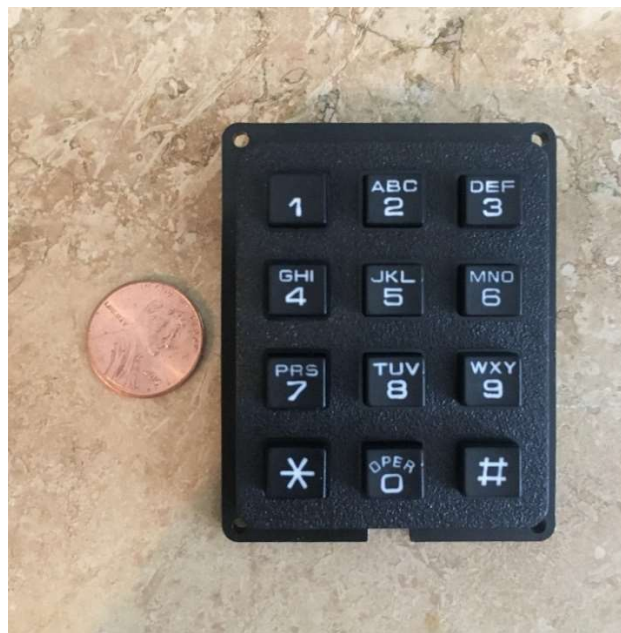


**Figure 5.2.4-1 ESP8266 Surface Mount**

## 5.2.5 Keying Mechanism Part Selection

A keypad component version of Adafruit Industries LCC's 1824 was selected for the design of this project. This keypad fit the requirements of what was needed for the security purposes of our project as well as compatibility with the overall design of the enclosure and microcontroller connection. The 1824 was \$7.50 and utilizes 12 numbers of key that was coded to work with our locking mechanism to open the box when the correct input has been put into it. We used a voltage divider to get the pin count to a serviceable amount that wouldn't cause any intrusion on the overall number of pins needed from other components. Below is Figure 5.2.5-1 which shows the exact image of the product we used for this project. This keypad, the 1824 Adafruit, was selected for the following reasons:

- Low cost
- Enough key selections
- Low current strain on overall design

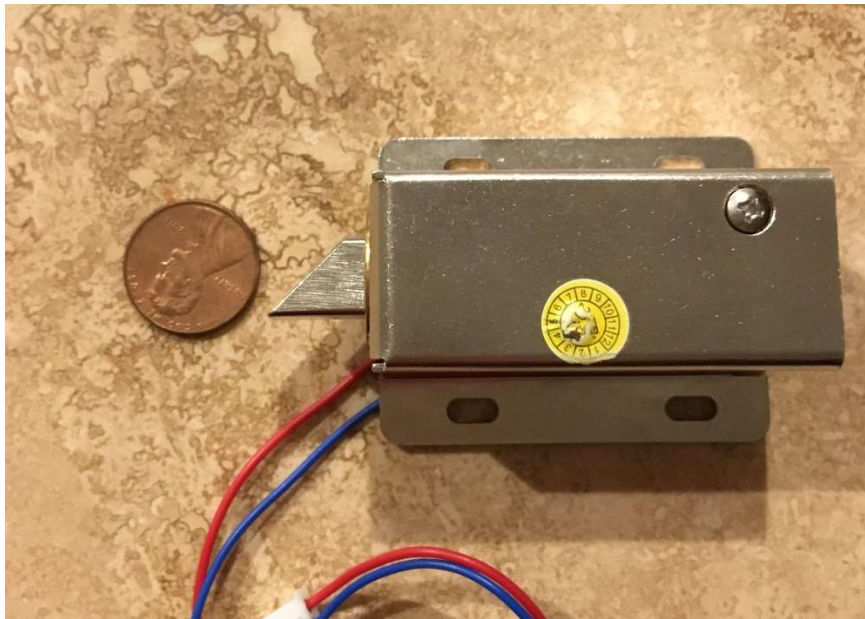


**Figure 5.2.5-1 1824 Keypad Switch**

## 5.2.6 Locking Mechanism Part Selection

The locking mechanism chosen with this project was the ROB-09065 SparkFun Electronics Micro-Servo Motor. This servo utilizes a sturdy construction with the ability to generate enough torque to perform the task of pulling back a strung latch but not strong enough to rip the rubber string from the confines of the mechanism.

In the second phase of the project we ended up deciding to switch to a different form of a locking mechanism by using an electronic lock instead of the mechanical alternative. The Adafruit Lock-style Solenoid 12VDC Lock was what we decided to use instead of a combination of mechanical components to get the result of locking and unlocking the box. This electronic lock took a binary signal to understand when to lock and unlock when the correct combination of keys was pressed on the keypad. This electronic lock cost \$18.12 on Amazon. In Figure 5.2.6-1 below shows the electronic lock we decided to use.



**Figure 5.2.6-1 Adafruit Lock-style Solenoid 12VDC Lock**

## 5.2.7 Power Related Component Selection

The Power Related Component Section focuses on the hardware components necessary for this design to remain operable. The AC/DC Adapter section, 5.2.7.1, is to convert an external plug-in source into power for the design to operate. The Battery section, 5.2.7.2, is in case of external power source loss the design can still operate off a backup power from internal battery devices hooked up to the Arduino board.

### 5.2.7.1 AC/DC Adapter

The AC/DC adapter chosen for this project was the SainSmart 1A Power Supply 12V Adapter shown below in Figure 5.2.7.1-1. This adapter in particular was aimed for compatibility with Arduino and offers the designated 12V we needed for the overall design and maintaining of the power of each component. The 1A it offers from its supplied voltage was also more than enough for the overall required current draw of the project where the electronic lock will be one of the more expensive in terms of current requirements, but only when the electronic lock is given permission to be activated or deactivated.

The following reasons are why this specific adapter was chosen for this project:

- Low cost
- High voltage
- High amperage
- Form factor
- Cable length



**Figure 5.2.7.1-1 SainSmart 1A Power Adapter 12V**

The adapter in general was one of the most important parts to our project due to the necessity for constant power draws from all the components. At the minimum of each of the products there was a low minimum usage for current, but when the components become utilized the draw became much larger which was why a decent voltage was needed for buffer in case of extra amperage use within the project. This adapter was also relatively cheap at \$7.99 and due to this was easily replaceable but was not needed.

## 5.2.7.2 Battery

The battery we initially selected was the uxcell Power Supply DC 3.7V 2500mAh unit. We planned to align two of these battery types in series to get the required 7.4V for our overall project's design. This was needed due to the electronic lock and other components requiring a constant high voltage to remain functional in situations where the main power supply has lost connection with the box.

The battery we decided on choosing for our backup power is the Floureon 7.4V 5200mAh 2S 30C Lipo Battery. This battery utilizes lithium ion technology discussed previously in the section titled 4.2.7.2 Battery, where the pros and cons of battery types were weighed on which would be the most cost effective, and long-term solution to our design choice for this project. The cost for this backup battery is \$29.99. This specific battery is shown below in 5.2.7.2-1. The following reasons are why this battery was chosen for this design:

- High capacity
- Rechargeable
- Fast recharge and discharge rate
- Moderately sized dimensions



**Figure 5.2.7.2-1 Floureon 7.4V 5200mAh 2S 30C Lipo Battery**

## 5.2.8 Enclosure Selection

The enclosure selection required to take a more realistic approach when dealing with possible containers that would fit the requirements of the project as well as become the most cost effective and non-intrusive container. The Suncast SSW1200 ended up being the selection choice for our enclosure with the cost of the container being approximately \$39.00. This container holds 22 gallons which was more than enough for testing purposes

of this project without being too small for components to not be able to fit inside the container when constructing the Smart Package LockBox. The Suncast SSW1200, shown below in Figure 5.2.8-1, ended up being our chosen product for the following reasons:

- Low cost
- Small container
- High quality
- Most secure option for specific design
- Lightweight



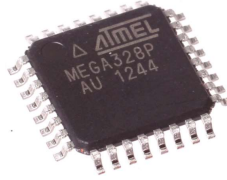
**Figure 5.2.8-1 Suncast SSW1200**

## 5.2.9 Microcontroller Selection

The ATmega328P microcontroller, seen below in Figure 5.2.9-1 was selected for reasons detailed in Section 4.2.9.3. To summarize, the ATmega328P was selected primarily for the following reasons:

- Low power consumption
- Low cost
- High community support
- Specifications are sufficient for project needs





**Figure 5.2.9-1 ATmega328P Surface Mount**

## **5.3 Part Acquisition**

With a project cost and time constraints, part quality needed to be balanced with cost and shipping time for this project. Listed below are the suppliers utilized for part acquisition for the Smart Package Lockbox System.

### **5.3.1 Amazon**

Although not primarily an electronics component supplier, many of the popular or commonly utilized modules or parts can be purchased from Amazon at low cost. However, because amazon is not primarily an electronics component supplier, efforts will be made to purchase components from the other suppliers, such as Digi-Key when time and cost constraints allow for us to do so. Amazon will primarily be used for the acquisition of development board prototype components for test purposes as well as for modules that can easily be tested before being soldered onto the PCB, such as the Wi-Fi module.

### **5.3.2 Digi-Key**

Digi-Key is a reputable electronic component distributor based in the US. The reliability of components from this supplier will likely be higher than those obtained from Amazon, however, this comes with the tradeoff of shipping costs and increased shipping time. This supplier will be used primarily for the majority of the system components.

### **5.3.3 Adafruit**

Adafruit is an open source hardware supplier based in the US. Like DigiKey, this is a reputable source for hardware. When a part or component is not available from DigiKey, efforts will be made to obtain them from Adafruit before utilizing Amazon.

## **6.0 Hardware Design**

The following sections detail the initial hardware design, review process, and hardware design revisions of the Smart Package Lockbox.

## 6.1 Initial Hardware Design

The free version of Eagle AutoCAD software will be utilized in the creation of the schematic and PCB design for this project. The Arduino UNO, shown in Figure 4.2.9.3-1, is an open-source electronic prototyping board that was used as a baseline starting point. Aside from the microcontroller used, the following aspects of the Arduino UNO board design were utilized for this project:

- Reset switch circuitry
- External clock circuitry
- Sections of the power circuitry that were utilized and/or repurposed for the project (Detailed in Section 6.1)

Components selection decisions will be discussed in this section as needed, however, hardware related research and major part selection are discussed in both Section 4.2 and Section 5. For most components, efforts will be made to keep the surface mount component sizes above metric code 1608 for ease of mounting. For reference, a part size comparison chart is provided below in Figure 6.1-1.

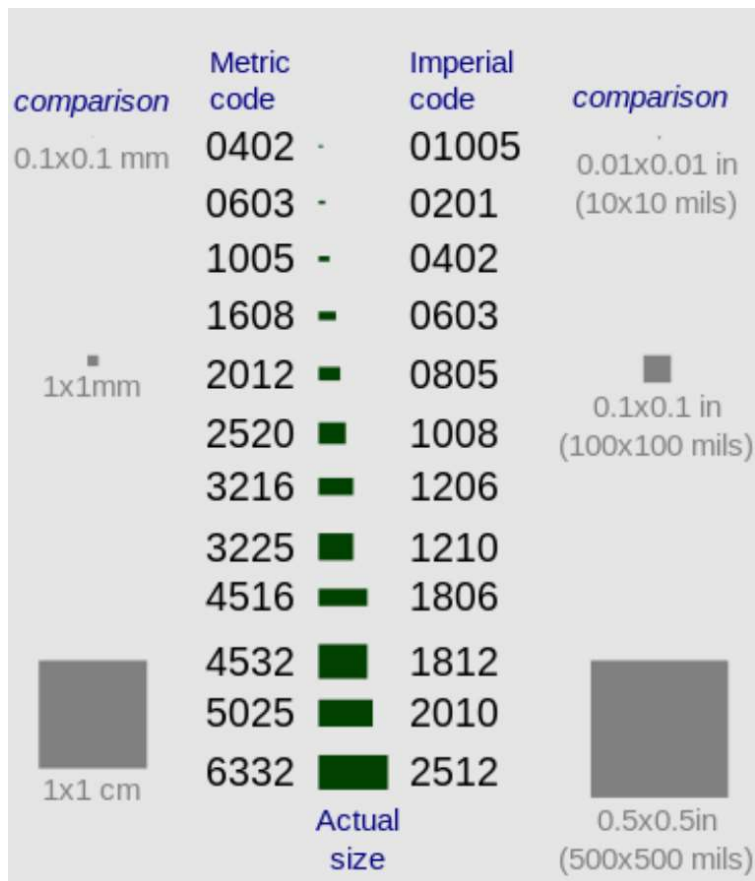


Figure 6.1-1 SMD Part Size Comparison (Courtesy of Wikimedia)

It should be noted that this design is meant to serve as an initial draft and will undergo the design review process detailed in Section 6.10.

## 6.1.1 Power

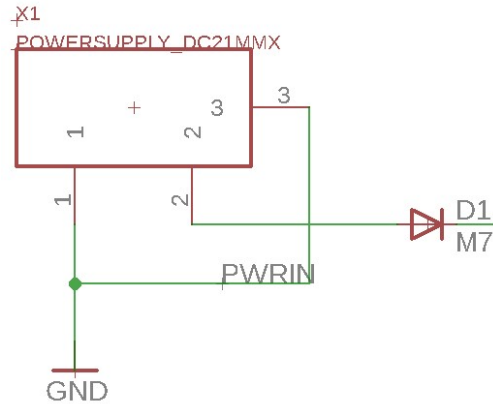
This section covers the power related circuitry for the system. However, before the power circuitry can be discussed, the main power and voltage requirements of each part of the design must be defined. Shown below in Table 6.1.1-1 is a summarization of the required voltages and estimated average current draw for the primary project subsystems.

**Table 6.1.1-1 System Component Voltage and Power Needs**

Part	Voltage Requirement	Current Draw
ATmega328P running at 8 MHz	3.3V	5 mA
ESP8266 (Wi-Fi Module)	3.0 - 3.6V	120 mA
MPU-6050 (IMU)	2.375 - 3.46 V	3.9 mA
Magnetic Switch (Lid Sensor)	2.2 - 5.5 V	7 mA when active
Keypad	N/A	Rated for 5mA for 0.5 seconds at 12V
Audio Transducer (Alarm)	3 V	3 mA
Servo (Locking Mechanism)	4 - 7 V	100 mA

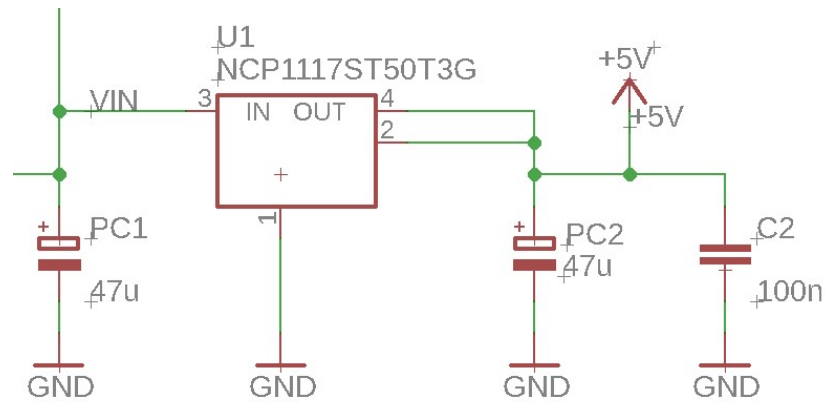
It should be noted that the above listed values are only meant to serve as a reference and are not appropriate for use in verification of requirement P2 detailed in section 2.3.2.2. This is because software and sleep modes will be utilized to bring the power consumption down on various parts. Also, the servo will not be running continuously and only will only be used when the system lock is engaged or disengaged.

As shown below in Figure 6.1.1-1, this system utilizes the same power input as Arduino UNO, a DC jack followed by the M7 diode for reverse polarity protection. This power input line then goes into the same 5 volt power regulator with supporting capacitors used by the Arduino UNO as well as the system battery charging module and a comparator (discussed later).

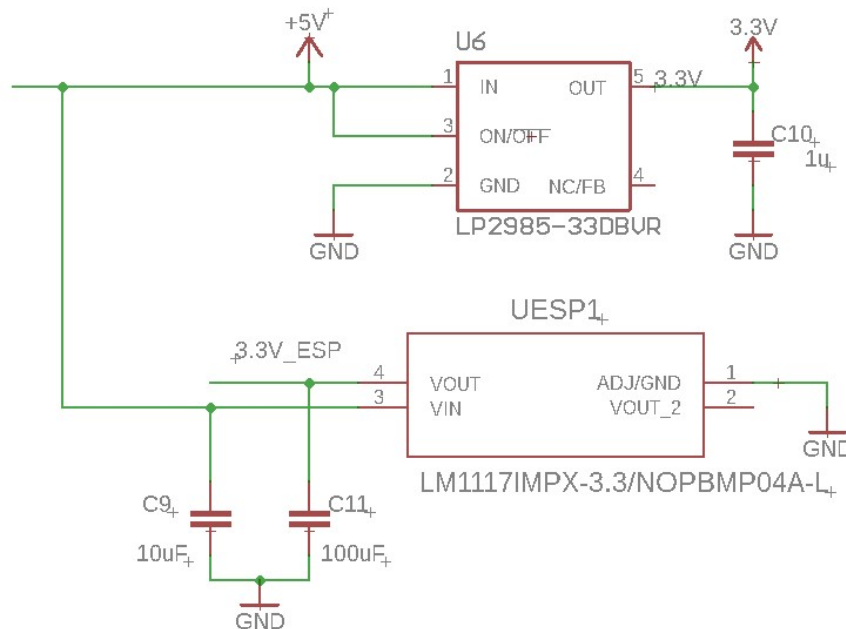


**Figure 6.1.1-1 System Power Input**

The 5 volt regulator power line is used to power the locking mechanism servo and also feeds into the two 3.3 volt regulators used in the system. The 5 volt power regulator and its supporting circuitry illustrated below in Figure 6.1.1-2 as well as 3.3 volt regulator U6 and its supporting circuitry illustrated below in Figure 6.1.1-3 are the same as the Arduino UNO. The 5 volt regulator has a maximum current draw of 1 ampere which is sufficient to supply power to the system. The LP2985 3.3 volt regulator just mentioned, however, has a maximum power current draw of only 150mA. This may be sufficient for most of the system 3.3 volt power, but there is a potential risk of overloading the regulator. It was found on multiple forum threads that the ESP8266 Wi-Fi module utilized in this project may occasionally have current spikes of up to 470 mA. Although software will be used to drastically lower the average power consumption of this Wi-Fi module, 470 mA is obviously well over the specifications of the LP2985. For this reason, a second 3.3 volt regulator with a higher maximum current draw, the LM1117 in Figure 6.1.1-3, was added to supply power to the ESP8266. The LM1117 was selected because it is a low cost 3.3V regulator that has a higher current limit of 800 mA. The LM1117 is only currently used for the ESP8266 with previous regulator supplying power to the rest of the 3.3 volt components. This was done to prevent potential current draw spikes on the ESP8266 from interfering with the operation of the microcontroller. If leaving the old 3.3 volt regulator for this reason is deemed unnecessary upon further review, it may be removed. On the LM1117, similar capacitors were used on the other regulators, but with the capacitance values recommended in the LM1117 data sheet.



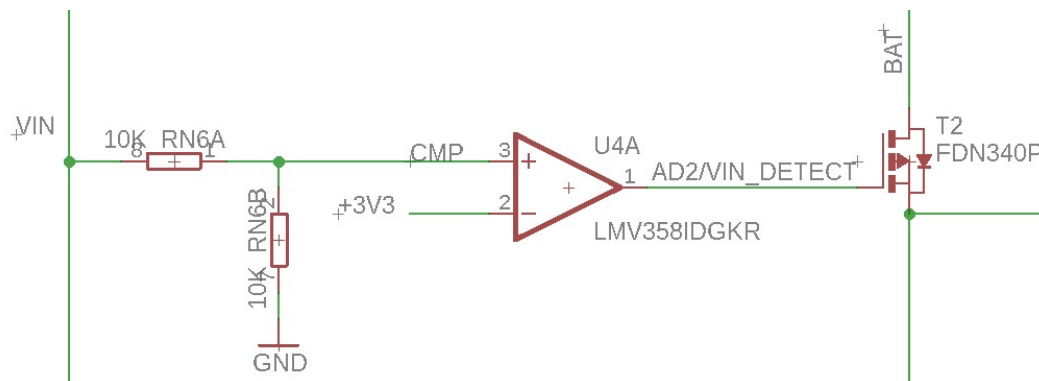
**Figure 6.1.1-2 5V Regulator Circuitry**



**Figure 6.1.1-3 3.3V Regulator Circuitry**

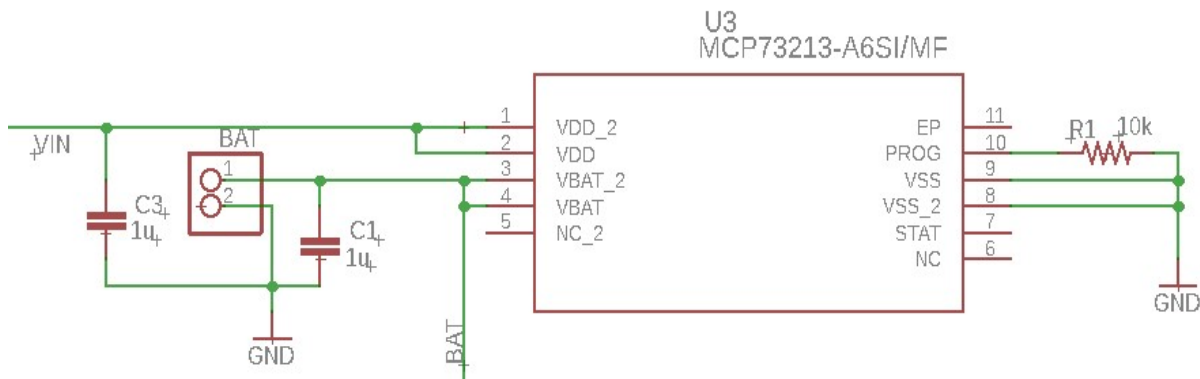
The Arduino UNO has both a DC jack input and a USB input for power. The circuitry utilized in the Arduino UNO to determine which of the power sources to utilize has been repurposed to both detect primary power loss to the circuit and to switch to the secondary battery power source in the case of primary power loss. This repurposed circuit can be seen below in Figure 6.1.1-4. The primary 12 volt power source from the Dc power jack goes through a voltage divider with 2 10k OHM resistors, resulting in a 6 volt input to the positive terminal of a LMV358 operational amplifier being used as a comparator in this application. The 3.3 volt power line not used for the ESP8266 is connected to the negative terminal of the operational amplifier being used as a comparator. The output of the comparator is connected to the gate of a P-channel MOSFET transistor. The result is that when half of the 12 volt dc input is higher than 3.3 volts, the comparator output will be

high and the transistor will remain switched off. On loss or severe reduction of the primary power source, however, the 3.3 volt line attached to the negative terminal of the comparator will become higher than the value on the positive terminal, resulting in the comparator output going low. This will switch the P-channel MOSFET transistor on, switching the system to battery power. This circuit also acts as a primary power loss detection circuit because the comparator output can be read by the ATmega328P microcontroller to determine whether power is being supplied from the 12 volt DC jack or the backup battery.



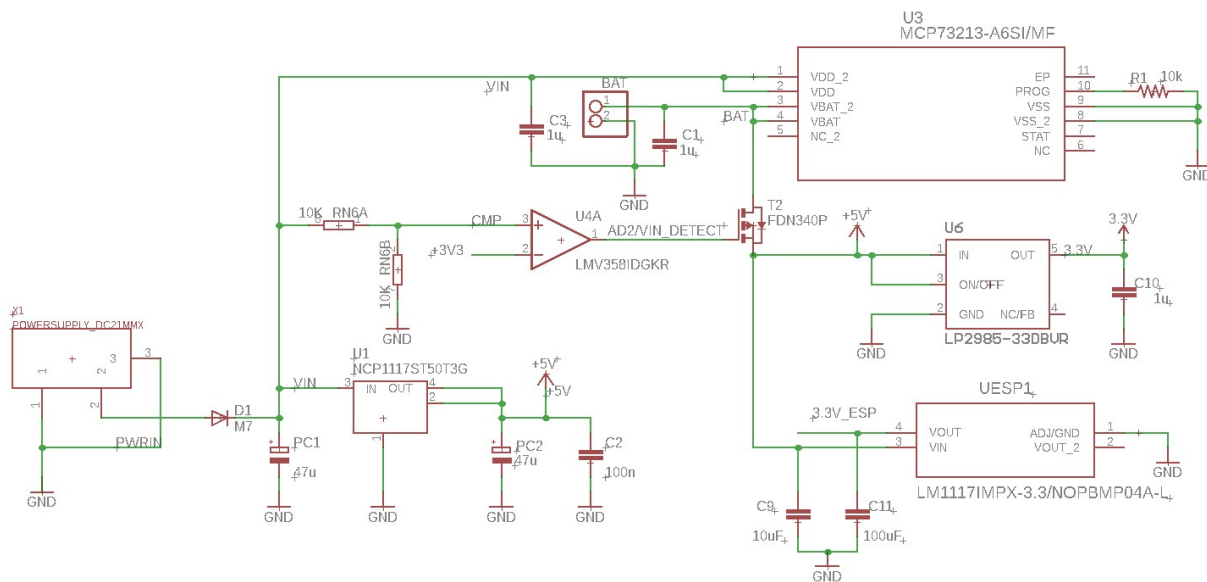
**Figure 6.1-4 Power Source Loss Detection and Switch**

In addition to being run into the 5 volt regulator and comparator sections of the circuit, the 12 volt DC input line also feeds the system battery charging circuit. The battery input as well as battery charging circuitry are illustrated below in Figure 6.1.1-5. For this projects application, the battery used and that needs to be charged by the circuit is a 2 cell lithium ion polymer battery. The battery charging IC, the MCP73213, was selected for use in this circuit because it is a 2 cell lithium ion polymer battery charge management controller targeted for use in low cost and low space applications. The capacitors connected between both the input and output of the MCP73213 and ground are there for circuit stability and are the recommended minimum capacitance for this application. Capacitors with voltages ratings of at least 12 volts for the input capacitor and 8.4 volts for the output capacitor will be used. This battery charge management IC has two input and two outputs, because this circuit will only use one 2 cell battery or two separate 1 cell batteries wired in series, the two inputs have been wired together and the two outputs have also been wired together. The PROG pin on the MCP73213 is for setting the constant charge current for charging the 2 cell lithium ion polymer battery, a 10k ohm resistor was selected for a constant charge current of 130 mA because charging speed is not critical for this application and a conservative value was considered desirable. Pending the review process detailed in Section 6.2, this resistor value and the constant charge current setting may be subject to change later on. This backup power charging setup makes it so that the battery is only used when the primary power source is down and so that the battery charging IC is off when the battery is in use.



**Figure 6.1.1-5 Battery Charging Circuit**

In this section, the power circuitry for the Smart lockbox System was displayed in parts for ease of reading due to the many different aspects of the power circuitry to discuss. However, the entire power specific section of the system is illustrated below to provide a complete picture of this circuit section in Figure 6.1.1-6.

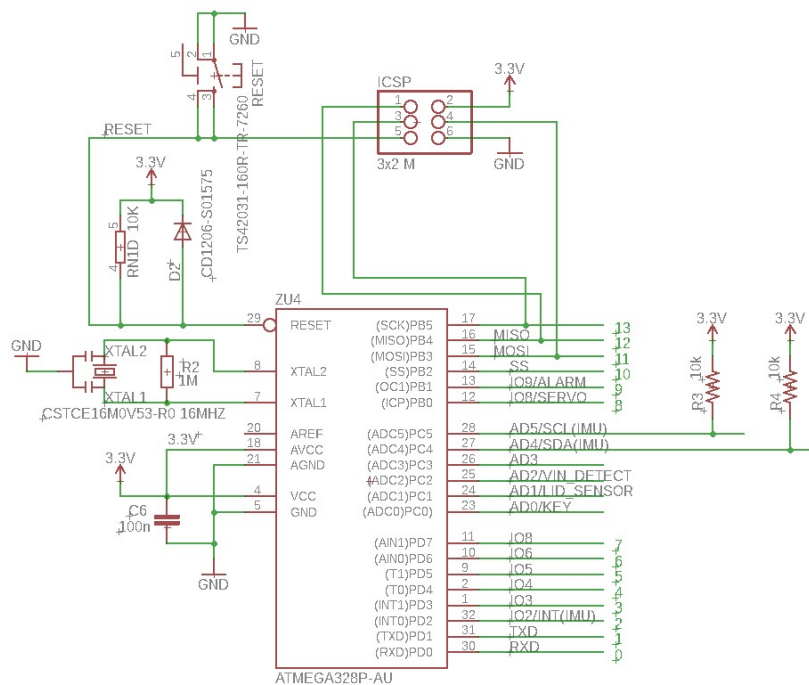


**Figure 6.1.1-6 Overall System Power Specific Circuitry**

## 6.1.2 Microcontroller

This section details the circuitry specific to the microcontroller illustrated below in Figure 6.1.2-1. The ATmega328P was selected for use in this project for reasons discussed in Section 4.2.9.3 and Section 5.2.9. As mentioned in Section 6.1, the external clock circuitry used in the Arduino UNO was also utilized for this project. This clock is not only sufficient for use in the current draft of the project, but allows for modification of the microcontroller clock rate through software if needed without circuitry changes so long as the new clock rate can be supported by the 3.3 volt power line. The AVCC pin has been connected to the

3.3 volt power line and both the GND and AGND pins have been connected to ground. The reset pin input circuitry from the Arduino UNO board as well as the In-Circuit Serial Programming (ICSP) header has also been utilized in this project. The In-Circuit Serial Programming header was used for manufacturability purposes, this header will be utilized to flash program the ATmega328P microcontroller. The header originally utilized the 5 volt power line on pin number 2 in the header below in Figure 6.1.2-1. This was changed to the 3.3 volt line because this is the line that now powers the microcontroller. It was determined that damage resulting from this change for other components on the 3.3 volt power line while flashing the microcontroller was unlikely. If this determined to be detrimental to other components on the 3.3 volt power line pending design review detailed in section 6.2, it will be changed later on. The communication lines leading to the inertial measurement unit, the MPU-6050, have pull-up resistors attached to avoid floating pin values and was done as best engineering practice. However, it was determined that the TXD and RXD communication pins leading to the ESP8266 Wi-Fi module did not require pull-up resistors. Also pending review, this may be changed later on.



**Figure 6.1.2-1 Microcontroller Specific Circuitry**

## 6.1.3 Sensors

This section details the design of the circuitry utilized for system sensors. A variety of sensors are used to ensure proper detection.



### 6.1.3.1 Lid Sensor

For the Smart Package Lockbox system lid sensor, was utilized for reasons detailed in Section 4.2.1.3 and Section 5.2.1. The specific magnetic switch used was the SL353HT. Although the component is surface mount, it is planned that this switch will be soldered to wires that will be soldered to the pads shown below in Figure 6.1.3.1-1. This is also pending the design review process covered in Section 6.2 and may be changed. The power pins on the magnetic switch are connected to pad 7 and pad 6. The signal pin is attached to pad 8 with a pull-down resistor to keep the signal wire low when not active high.

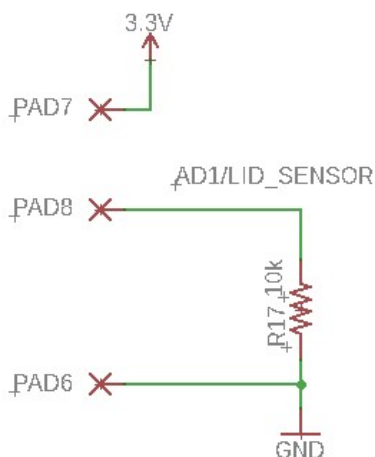
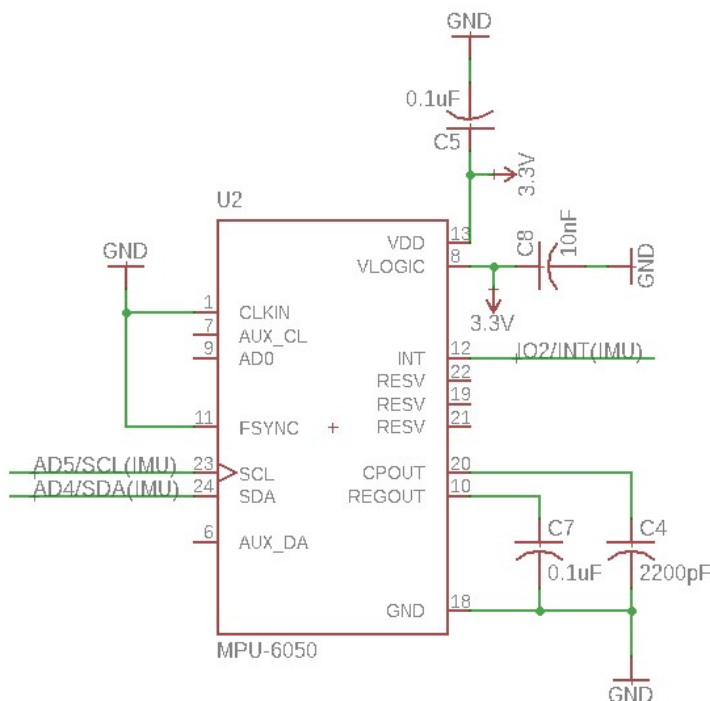


Figure 6.1.3.1-1 Lid Sensor Specific Circuitry

### 6.1.3.2 Inertial Measurement Unit

Per requirement SE3 listed in Section 2.3.2.3, the system shall be able to detect the attempted theft or unauthorized moving of the system. An inertial measurement unit, specifically the MPU-6050, was selected for use in order for the Smart Package Lockbox to fulfill this requirement for reasons detailed in Section 4.2.2.3 and Section 5.2.2. The inertial measurement unit and its supporting non-power related circuitry are illustrated below in Figure 6.1.3.2-1. The SCL, SDA, and INT pins are connected to the ATmega328P microcontroller to relay the measured system orientation changes and dynamic forces applied onto the system. The SCL and SDA pins are pulled high for reasons detailed in Section 6.1.2. However, it is essentially just done as best engineering practice. The FSYNC and CLKIN pins are connected to ground. FSYNC stands for frame synchronization digital input and it is connected to ground per the MPU-6050 datasheet because it is unused in this application. The CLKIN is for an external clock input and it is also connected to ground per the MPU-6050 datasheet because an external clock is not required for the inertial measurement unit for this project. The CPOUT and REGOUT pins are for charge pump capacitor connection and the regulator filter capacitor connection respectively. Specific capacitors have been connected to these pins per the MPU-6050 datasheet. A power line

with a voltage of 3.3 volts has been attached to the VDD pin and to supply power within the MPU-6050 voltage input specification range of 2.375 - 3.46 V with a filtering capacitor added per the datasheet and as best engineering practice.



**Figure 6.1.3.2 IMU Specific Supporting Circuitry**

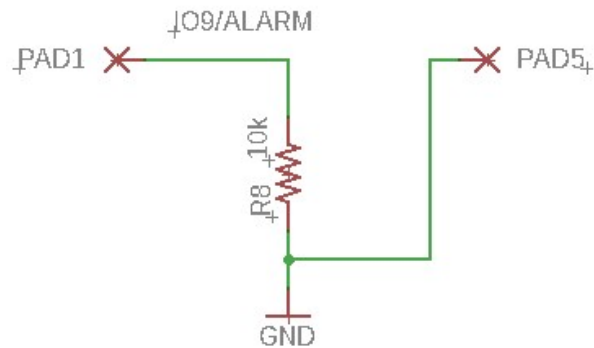
### 6.1.3.3 Power Loss Detection

Per requirement SE5 listed in Section 2.3.2.3, the system should be capable of detecting the loss of the primary power source. This was done with the use of a comparator circuit and is detailed in Section 6.1.1.

### 6.1.4 Alarm

Per requirement SE2 listed in Section 2.3.2.3, the system shall include an alarm with a minimum sound pressure level of 85 dB or higher as measured from outside the Smart Package Lockbox while the lockbox is closed and locked. This was done by utilizing an audio transducer from DigiKey, part number 668-1503-ND, for reasons listed in Section 4.2.3 and Section 5.2.3. Specifically, however, it was selected because it had a high output for low cost and low voltage. The supporting circuitry and connection points for this component are illustrated below in Figure 6.1.4-1. The positive wire of the audio transducer will be soldered to pad 1 and the negative end will be soldered to pad 5. Due to nature of the transducer, a pull-down resistor was attached to the alarm activation pin to keep it low

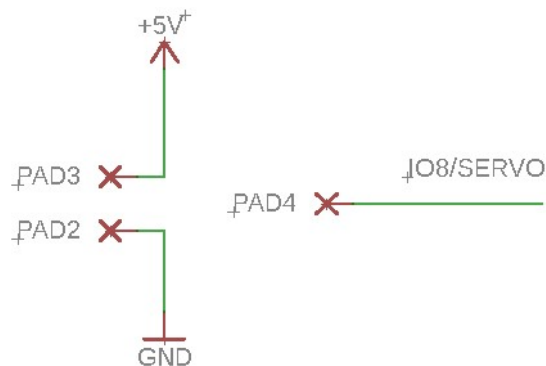
unless the microcontroller pulls it high. This was done in order to prevent possible unintentional activation of the transducer and as best engineering practice.



**Figure 6.1.4-1 Alarm Specific Circuitry**

## 6.1.5 Locking Mechanism

Per requirement SE1 listed in Section 2.3.2.3, all security related components, including the locking mechanism, must be contained inside the lockbox. For reasons detailed in Section 4.2.6 and Section 5.2.6, a servo that will engage and disengage a bolt lock has been selected to provide locking capability while not violating this requirement. The voltage and estimated active power draw for the selected servo are listed in Table 6.1.1-1. The supporting circuitry for the locking mechanism is illustrated below in Figure 6.1.5-1. The positive and negative wires for the locking mechanism servo are connected to pad 3 and pad 2 respectively. The control signal for the servo is connected to pad 4 which is wired to the IO8/SERVO pin on the ATmega328P microcontroller. This setup is also pending the design review process detailed in Section 6.2 and will be changed if needed.



**Figure 6.1.5-1 Locking Mechanism Circuit Connections.**

## 6.1.6 Wi-Fi Module

For reasons detailed in Section 5.2.4, the ESP8266 Wi-Fi module was utilized in this project for Wi-Fi communications. One of those reasons being a high level of community support. Per requirement W1 listed in Section 2.3.2.4, the system Wi-Fi communication range must be at least 100m while the box is closed. It has been determined by tests run in the supporting community that the range of this module exceeds 230m. Because of this, it has been assumed that the project Wi-Fi range requirement should be met with this module. Specifically, the ESP-12 surface mount version of the ESP8266 was used per the course requirement that surface mount technology be used where applicable.

The supporting circuitry for the ESP8266 Wi-Fi module in the Smart Package Lockbox system is illustrated below in Figure 6.1.6-1. The RXD and TXD pins are connected to the TXD and RXD pins on the ATmega328P microcontroller. The pin swap between RXT and TXD is intentional as the TXD pins are for transmitting and the RXD pins are for receiving communications. Thus, they are swapped so that the Wi-Fi module and ATmega328P microcontroller can talk to one another. The GPIO15 pin is pulled low with a 10k ohm pull-down resistor and the RESET and CH\_PD pins are both pulled high with 10k ohm pull-up resistors. The TXD and RXD do not require pull-up resistors like the inertial measurement unit communication lines because the ESP8266 has internal pull-up and pull-down resistors for this purpose.

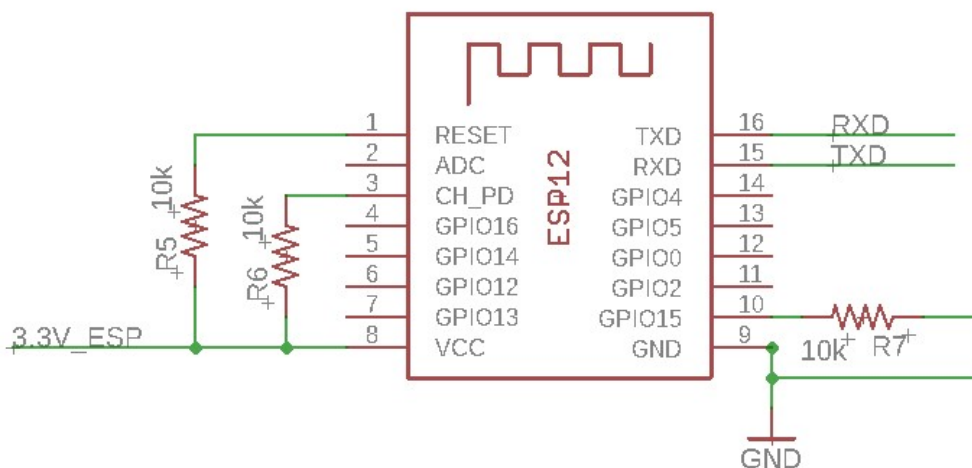
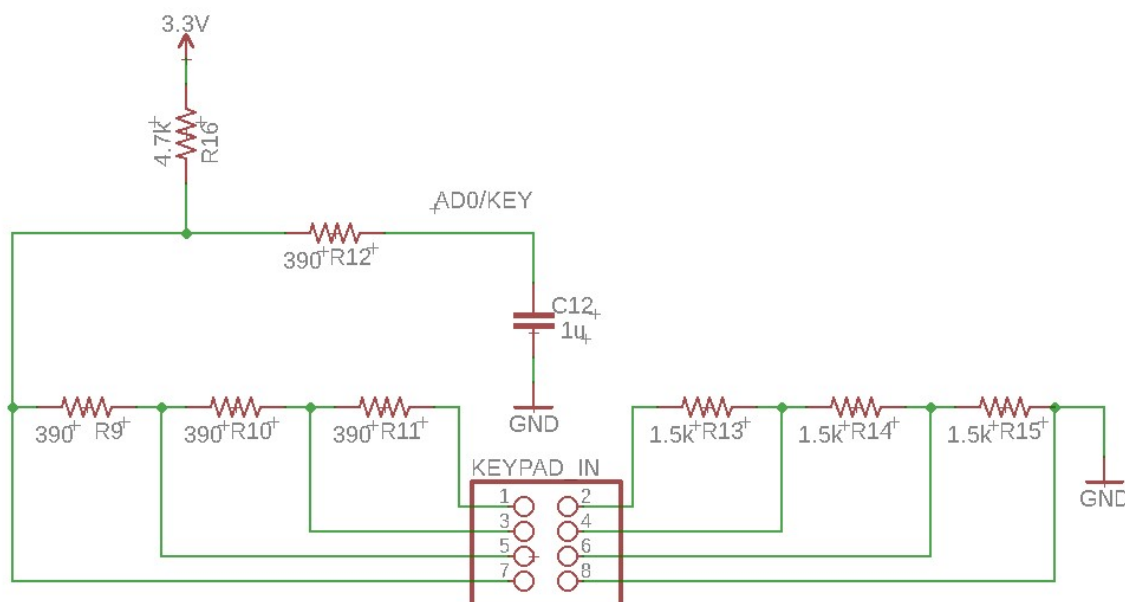


Figure 6.1.6-1 Wi-Fi Module Specific Supporting Circuitry

## 6.1.7 Keying Mechanism

The Smart Package Lockbox system will utilize Wi-Fi to allow the system to be unlocked remotely via Wi-Fi, however, a manual keypad keying mechanism is also included to enable authorized people without cell phones to unlock the system. This is also done so that the unlocking capability of the system is retained in the case loss of Wi-Fi

communications for any reason. The specific keypad utilized as well as the reasoning behind the selection are discussed in Section 4.2.5 and Section 5.2.5. The specific circuitry relating to the keypad input is illustrated below in Figure 6.1.7-1.



**Figure 6.1.7-1 Keypad input Specific Circuitry**

As can be seen above in Figure 6.1.7-1, the keypad input has 8 pins. This is because the keypad has a wire going down each row and column of the keypad that are not connected. The button presses detectable because the buttons push down switches that connect the row and column wire for that button, enabling the detection of many 16 buttons with only 1 output for each row and column. One method of determining the key pressed is by using software to figure out which key was pressed by looking at the row and column input signals. However, a tutorial was found that detailed a method for reading keypad input with only one pin. This is done by utilizing the ability of the analog input pins on the ATmega328P microcontroller to read varying voltage levels with the use of an internal analog to digital converter. This works by constructing a voltage divider circuit with the keypad-connected resistors providing a different resistance between the microcontroller pin and ground depending on what key is pressed. Essentially, when a key is pressed, one of the column wires (the pins on the left side of the header) and one of the row wires (the pins on the right side of the header) are connected and a unique combination of resistors in series are used as the bottom half of the constructed voltage divider. This results in a unique voltage value that can be read by the ATmega328P microcontroller analog input pin. The additional resistor between the voltage divider and the pin as well as between the pin and ground are added for signal smoothening purposes. This will require setting up a test unit that transmits the read voltage levels corresponding to each keypad button to a CPU so the read voltage levels for each button pressed can be recorded and coded in with tolerances as deemed appropriate.



## 6.2 Hardware Design Review Process

Is it desirable for there to be as few errors as possible when designing a system, however, human error is usually unavoidable and for this reason it is common for most designs to go through a review process. The current design is meant to serve as an initial draft which will undergo the review process as stated below:

1. Group 5 will meet, and each member will present/explain their design decisions (or revision design changes) in detail to the rest of the group
2. The group will then ask questions and discuss potential tradeoffs of those design decisions as well as potential alternatives
3. After discussion, potential concerns or areas for improvement will be recorded
4. Obvious or simple implementations for improvements or fixes are implemented
5. The design is then potentially presented to an expert in the field for critique or basic review. In this case questions will also be asked based on the notes from step 3
6. Design improvements are implemented based on the previous steps and a new design revision is made and documented
7. Steps 1 through 6 are repeated until time does not reasonable allow for another revision

It should be noted that a considerable amount of time will be put into the revision process in the 2-3 week following the conclusion of senior design 1 in order to identify design weaknesses and improve system functionality, performance, reliability, cost, and power efficiency.

## 6.3 Hardware Design Revisions

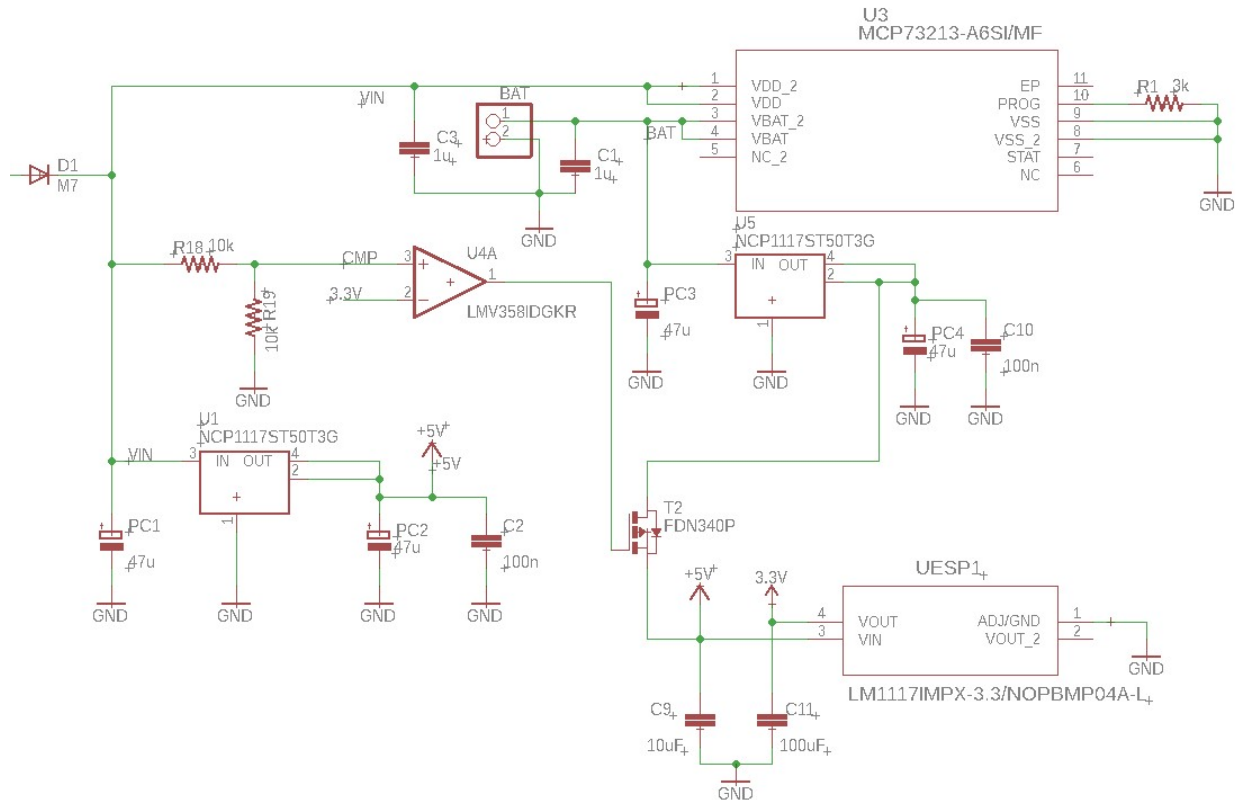
Documented herein are the hardware design revisions made to the Smart Package Lockbox design following the first semester of senior design.

### 6.3.1 Power

Following review, it was determined that using separate 3.3 volt regulators for the Wi-Fi module and the rest of the project peripherals was unnecessary. For this reason the LP2985 3.3 volt regulator was removed and all of the projects circuit sections requiring 3.3 volt power were wired to utilize the same 3.3 volt power line as the ESP8266 Wi-Fi module supplied by the LM1117 3.3 volt regulator. Although the LP2985 3.3 volt regulator was unable to handle the current draw of the ESP8266 Wi-Fi module, it was determined that the LP1117 3.3 volt regulator would be capable of handling all of the 3.3 volt power needs of the circuit.

It was also determined that there was a need to regulate the power coming from the battery before it passed through the PNP transistor used for switching to battery power when the primary battery source is lost. The same 5 volt regulator and supporting circuitry used for

regulating the 12 volt DC input was also used for the purpose of regulating the battery power. The revised project power circuitry can be seen below in Figure 6.3.1.

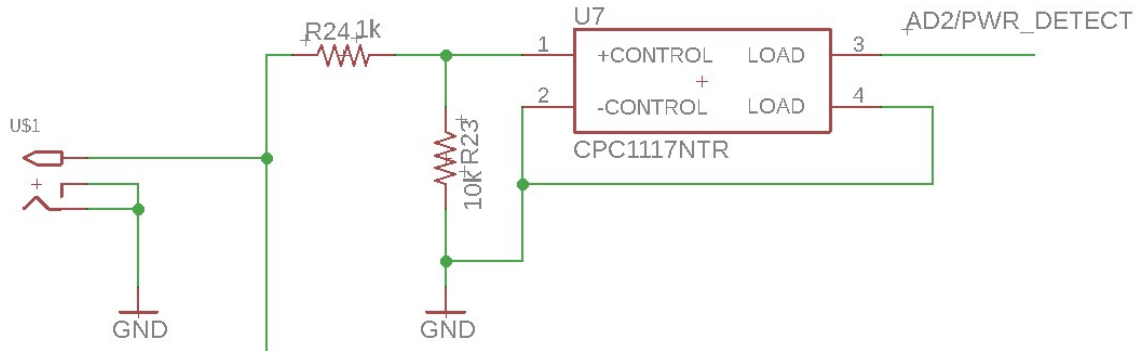


**Figure 6.3.1-1 Revised Power Circuitry**

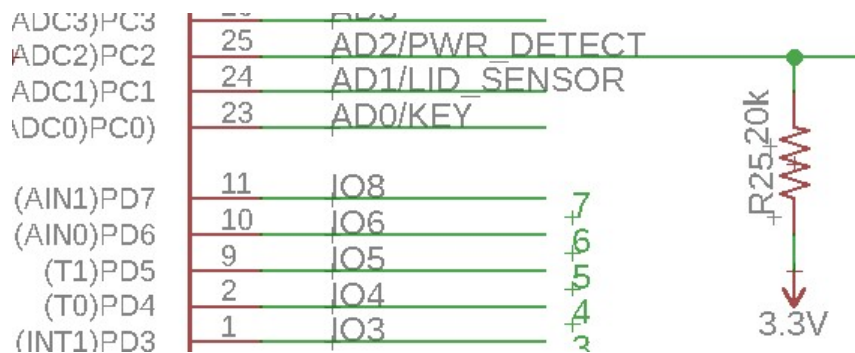
## 6.3.2 Primary Power Loss Detection

Due to concerns that connecting a pin of the ATmega328P chip directly to the output of the LMV3581 Op-Amp could potentially damage the microcontroller, another power loss detection circuit was created. The new power loss detection solution utilizes an optically coupled and normally closed MOSFET relay connected directly to the 12 volt DC power input. In order to limit the current going through the relay, a 1k Ohm resistor is placed in series with the relay and a 10k Ohm resistor is placed in parallel with the relay. The output of the relay is connected to the AD2/PWR\_DETECT pin which is pulled up to 3.3 volts by a 20k Ohm resistor. The result is that when 12 volt DC power is applied to the circuit, the normally closed relay opens and the AD2/PWR\_DETECT pin is pulled up to 3.3 volts. When the 12 volt DC primary power supply is lost, the relay will close and short this pin to ground. The microcontroller will then be able to detect that the pin has gone low and use that to determine that the primary power was lost. The Relay circuit and pull up resistor placement can be seen below in Figure 6.3.2-1 and Figure 6.3.2-2.





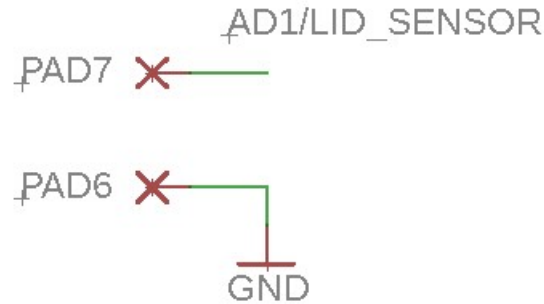
**Figure 6.3.2-1 Power Loss Detection Circuit**



**Figure 6.3.2-1 Pull Up Resistor Placement**

### 6.3.3 Lid Sensor

For reasons discussed in Section 5.2.1, the magnetic switch utilized for the system lid sensor was changed to the SL353HT, a simple two-wire magnetic switch. For the new magnetic switch, the supporting circuitry required is only 2 pads. One wired directly to ground and the other wired directly to the AD1/LID\_SENSOR pin on the ATmega328P microcontroller chip. Depending on the lid position, the magnetic switch will open or close and short the pin to ground. This is detected by the pin and utilized by the microcontroller to determine the lockbox lid position. The supporting circuitry for the new lid sensor is shown below in Figure 6.3.3-1.



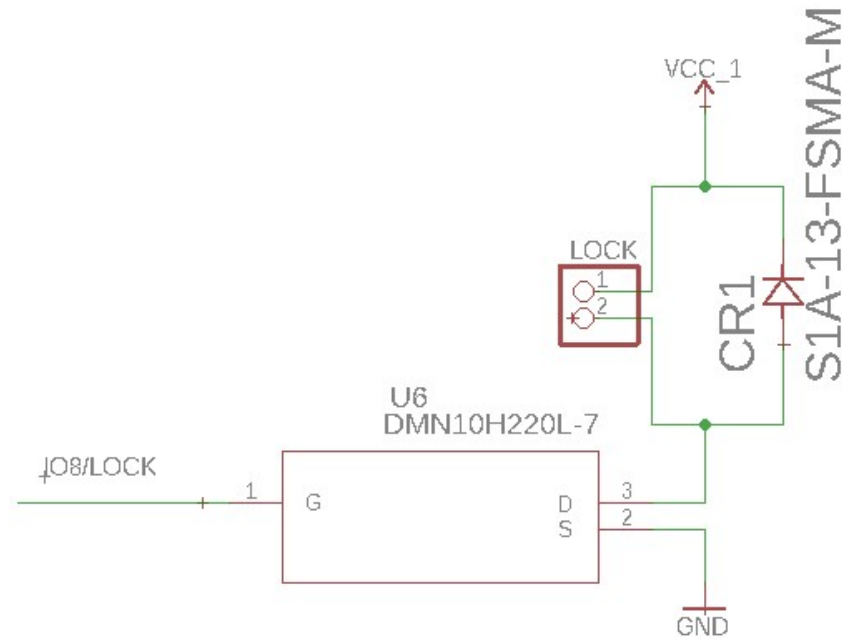
**Figure 6.3.2-1 Revised Lid Sensor Circuitry**

### 6.3.4 Alarm Pull-Down Resistor

It was determined during the prototyping phase that the 10k Ohm pull-down resistor shown in Figure 6.1.3.1-1 was unnecessary. For this reason, R17 on the PCB was not populated. The slot for R17, however, was left on the PCB because it does not have an impact on the circuit.

### 6.3.5 Locking Mechanism

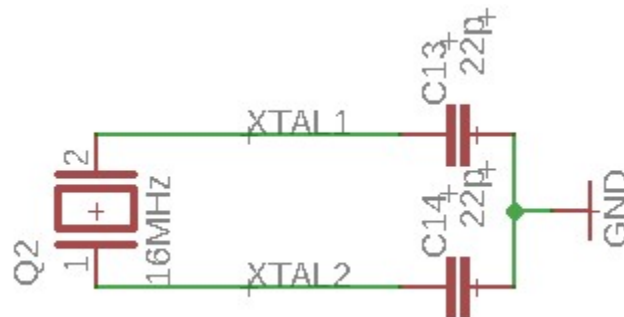
For reasons discussed in Section 5.2.6, it was determined that the system locking mechanism would be changed from a servo lock mechanism to one that utilized a solenoid style lock that requires 0.5 Amps of current to activate. The solenoid lock is connected directly to the 12 volt DC power source in series with a reverse voltage protection diode rated for up to 1Amp. To control the lock, an NPN transistor is placed between the lock and ground. The NPN transistor is rated to handle up to 1.4A of continuous drain current and has a drive voltage of 4.5 volts, meaning that it can be switched on by the IO8/LOCK pin on the microcontroller. The revised circuitry for the locking mechanism can be seen below in Figure 6.3.5-1.



**Figure 6.3.5-1 Revised Locking Mechanism Circuitry**

### 6.3.6 Microcontroller Clock

The initial design for the microcontroller supporting circuitry utilized a 16MHz surface mount oscillator for the system clock. The circuit was revised to also include the ability to use a through-hole 16MHz crystal oscillator for the clock. This was done to enable the group to use either component type for the system clock if needed for debugging purposes. This is because a through-hole clock could be removed and replaced without the need for a soldering service. It should be noted that only one of the two supported clock types will be used at any one time. The crystal oscillator and its supporting circuitry can be seen below in Figure 6.3.6-1.



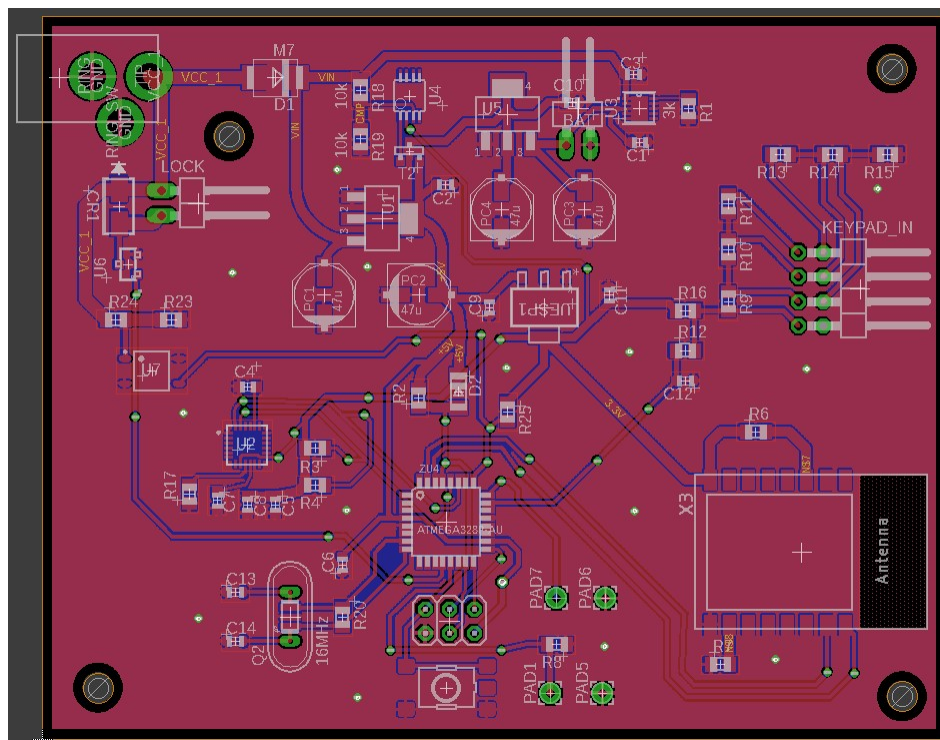
**Figure 6.3.6-1 16Hz Crystal Clock Circuitry**

### 6.3.7 Revised PCB Design

The PCB design shown in Section 6.1.9 was only meant to serve as an initial draft and thus had multiple issues. Following a review of the initial PCB design, a revision was made with the following changes:

- Components layout was done in a more logical manner that, when practical, followed the system schematic layout.
- A ground plane was added to both layers with vias connecting to two ground planes in various places on the PCB for increased stability.
- Traces were made thicker wherever it was practical to do so, with extra care being taken to ensure the power traces were as large as they could be practically made.
- Parallel traces on opposing sides of the board were avoided when possible.
- Communication traces were distanced from other communication traces and components that could potentially interfere with the communication signals when possible.
- Efforts were made to increase the distance between components, not including the components (such as stabilizing capacitors) that benefitted from close proximity.
- Sharp turns and angles in traces were avoided.
- Mounting holes were added.

The revised PCB is shown below in Figure 6.3.7-1. However, for increased visibility of the traces, an image with the ground planes removed is also shown below in Figure 6.3.7-2.



**Figure 6.3.7-1 Revised PCB**

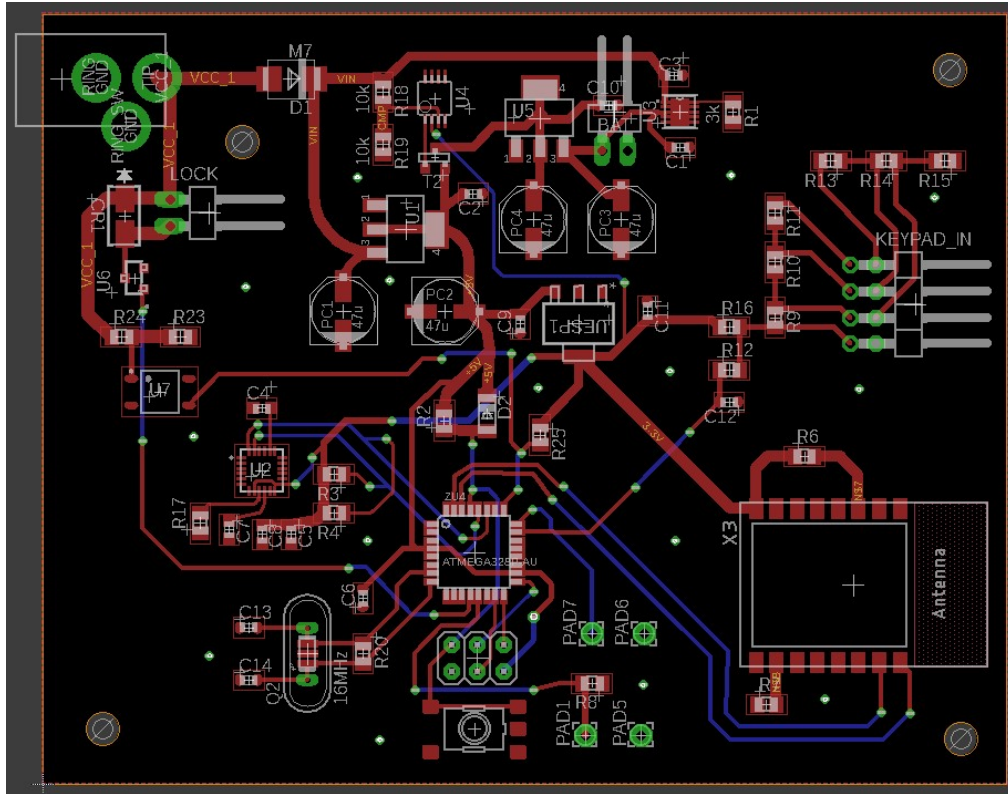


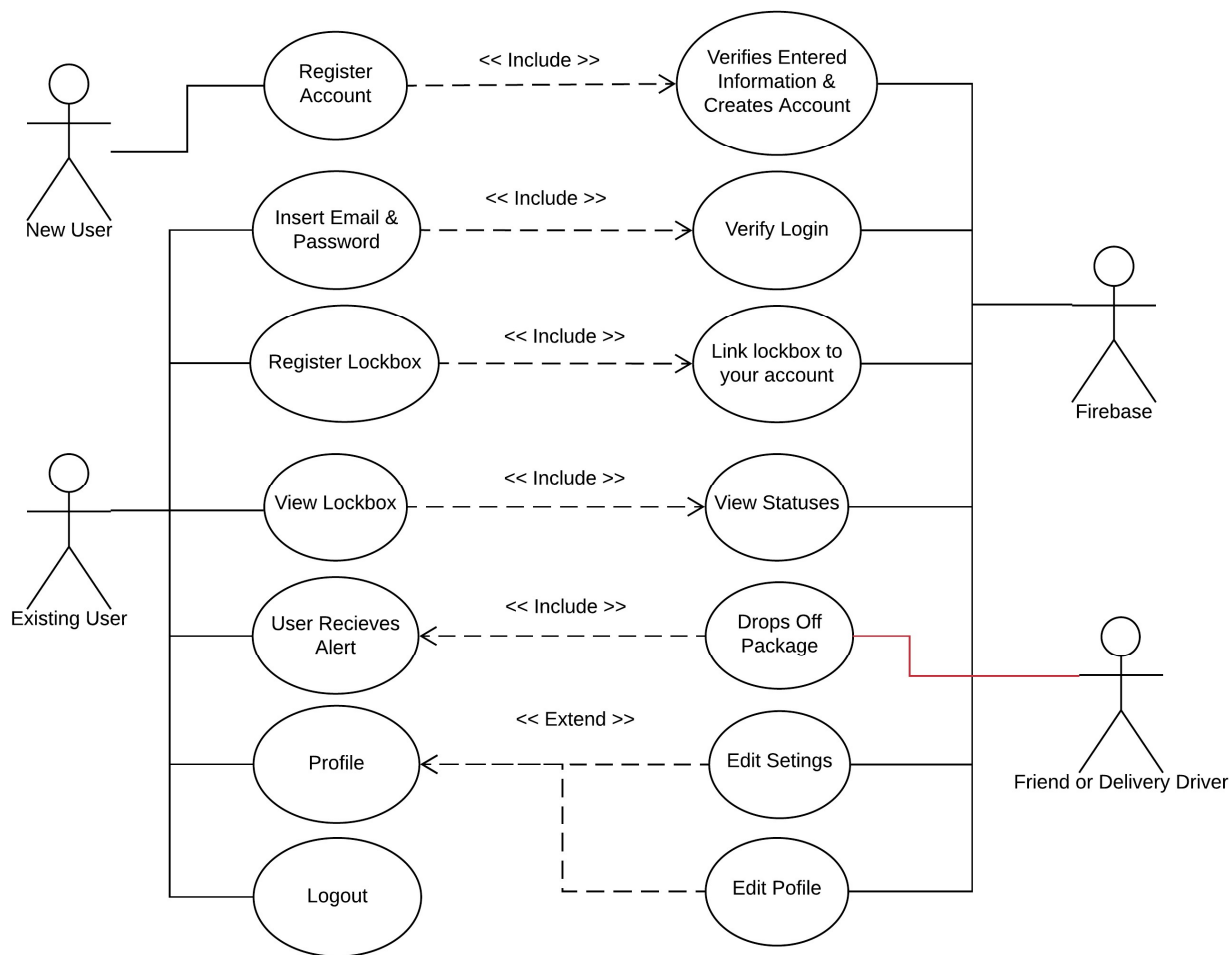
Figure 6.3.7-2 Revised PCB without Ground Planes

## 7.0 Software Design

Overall our product will consist of a web user interface and a mobile application. They will both be tied into a database which will contain their user information and all data relevant to their lockbox. The user will be able to register their individual lockbox and update their user settings. High level diagrams of the database and interaction are contained below.

### 7.0.1 Use Case

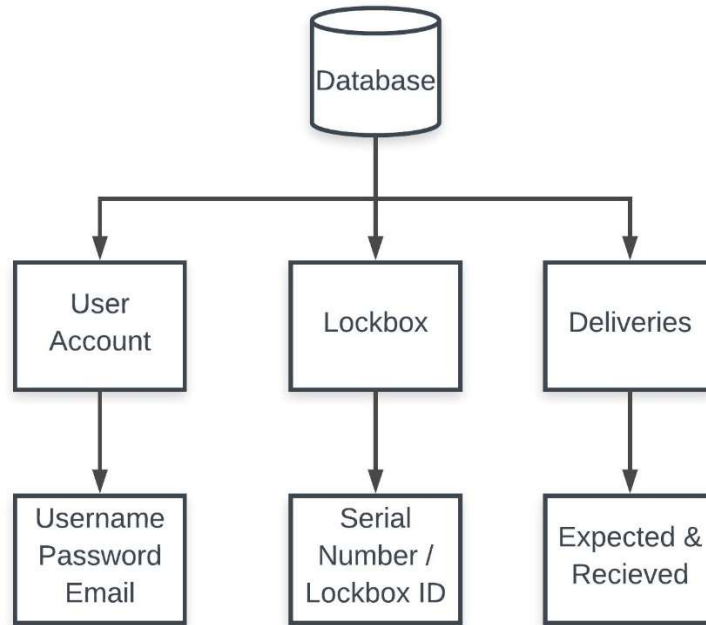
The use case diagram in figure 7.0-1 below details a high-level interaction between a user and the Smart Lockbox. The user will initially register if they are a new user or if they are a returning/existing user they will just log in. Upon logging in the database will verify all entries then bring you to your home page. From here you may view deliveries or edit your profile. Any changes will be reflected and stored in the database. A delivery driver will only have access to the delivery he/she is scheduled to drop off. The final action being taken is logging out.



**Figure 7.0-1 Use Case Diagram**

## 7.0.2 Database Structure

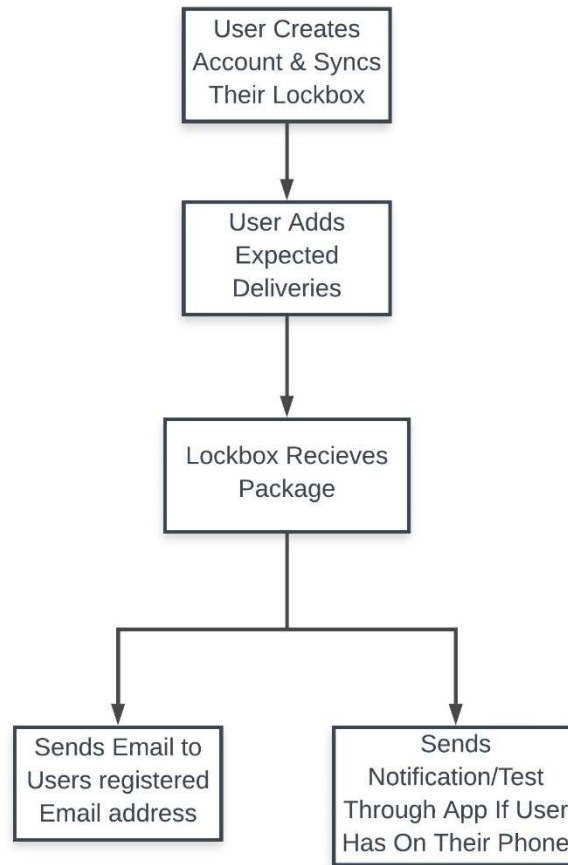
With the LAMP stack in mind our database of choice will be MySQL. The overall structure of the database we will be creating can be seen in figure 7.0.2-1 below. Overall the database will be simple and just store user information. The three primary pieces of information the database will store are user accounts, lockboxes, and deliveries. Each user account will be tied to a username, password, and email. Each lockbox will have a serial number or lockbox ID. The deliveries will be categorized into expected and received and will also vary by user. User A who is expecting a package from FedEx will have no knowledge of an expected deliver user B has from UPS.



**Figure 7.0.2-1 Database High Level Structure**

### 7.0.3 High Level System Flow

A high-level system view can be seen in figure 7.0.3 below. The initial step is a user creates an account if they don't already have one and syncs their lockbox. Upon successfully doing so the user will add expected deliveries which will then be stored to the database on the backend. After the lockbox receives a package the lockbox will wirelessly send a notification to the owner's phone (if they should have the app installed on their phone and enable the notifications). If they don't have the mobile version or don't have a phone, the lockbox also sends an email notification.

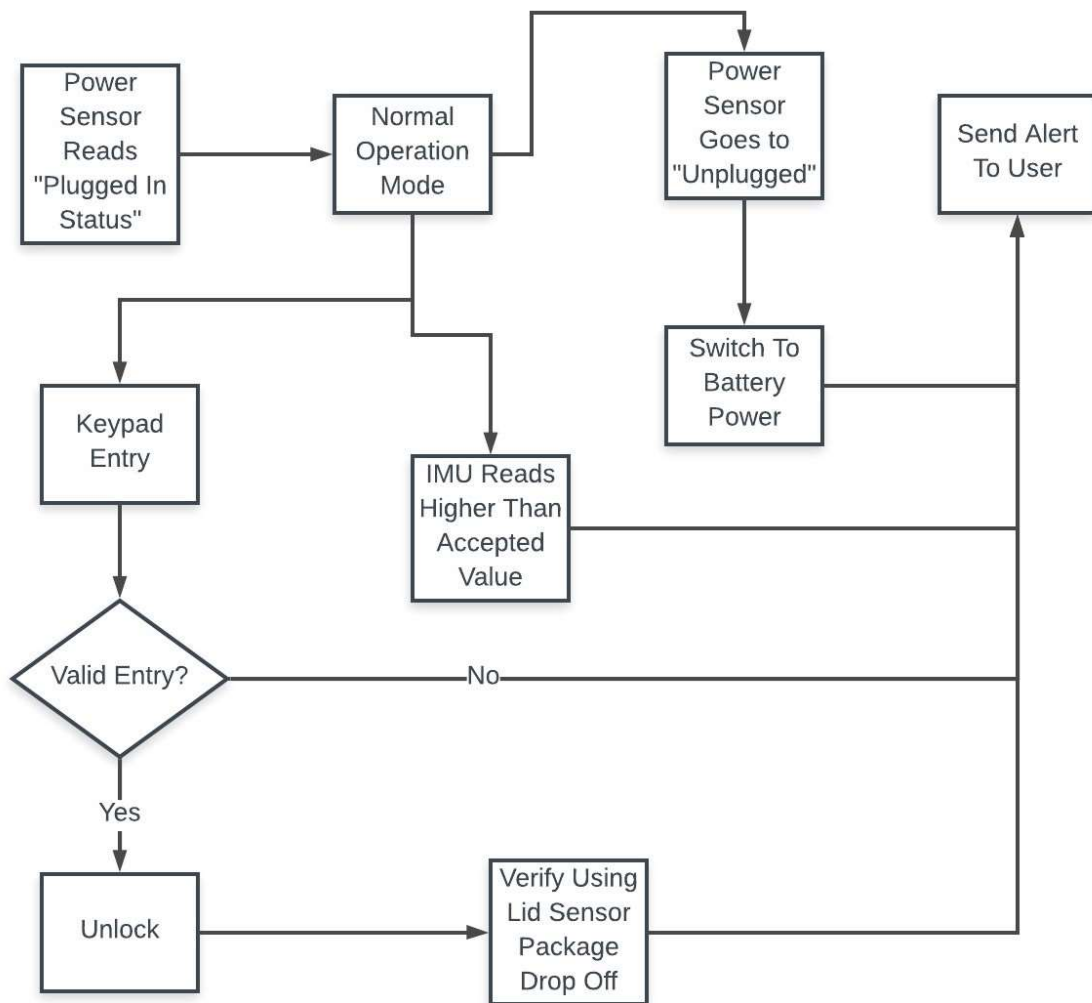


**Figure 7.0.3 High Level System Flow Chart**

## 7.0.4 Low Level Operational

From a peripheral and low-level standpoint, the lockbox operates as normal so long as a power sensor plugged in variable will read true. If this variable is switch to false then the lockbox switches to the battery secondary power and sends an alert to the user. Back to normal operation if a keypad entry is registered the lockbox verifies the entry was valid. If so the box unlocks if not the box remains locked and after a number of unsuccessful attempts notifies the user. After the box unlocks the lid sensor verifies the box was opened and closed signaling a delivery and successful drop-off. From here a notification is sent to the user notifying of a package delivery. Going back to normal operation if the IMU sensor receives a value above its set threshold, it will perceive that the lockbox is being tampered with and send an alert to the user along with starting an alarm. A visual representation of this can be found in figure 7.0.4-1 below.





**Figure 7.0.4 Low Level Operation Flow Chart**

## 7.1 High Level Software System Architecture

In order to create an architecture for our product there are a few pre-existing architectures that exist and are commonly used in web and application creation. The architecture consists of a stack which is a collection of software required for the development of web applications. Among the stacks out there the most common are LAMP, WISA, and MEAN. Each of these are acronyms and each letter stands for one component of the stack. Each

stack consists of an operating system, a programming language, database software, and a web server.

## **7.2 Software Development Life Cycle**

The software development life cycle of this project consists of six different phases or stages. The methodology that will be used in conjunction with this software development life cycle is the agile development model. Using the agile development methodology will allow the team of programmers to quickly adapt to any changes in the requirements of the project, or any constraints that were not previously present.

The agile methodology was elected over the waterfall method because of the implementation of weekly sprints. The team will meet at the beginning of every week in a sprint meeting and define a realistic weekly sprint that will need to be implemented to the best degree by the following sprint meeting. At that point, the team will analyze and evaluate the completed sprint and determine the tasks for the following sprint. Usually sprints should be around 2 – 4 weeks long in a normal development environment, but because of the shortened summer semester and only an effective 8 weeks to develop the software for the project, shorter and more selective weekly sprints are required.

### **7.2.1 Planning & Analysis**

The first and second stage of the software development life cycle consist of a preliminary analysis that will serve as the planning stage. During this phase of the development, the team will devise and propose solutions to the wide variety of software related objectives and problems that might come up during software development. Looking at how other similar or related products have come up with a solution to their issues could be another part of the planning process.

The analysis phase comes into play after the solutions that were proposed during the planning stage have been taken into account. At this point, by taking the proposed solutions, the team can define the project or sprint goals for their intended section of software development. This is a critical part when it comes to diagnosing problems from a previous sprint so the team can properly adapt and recommend improvements to be done in the next sprint of development.

### **7.2.2 Design**

The design phase of the software development life cycle consists of designing detailed descriptions of the required features and operations that are ultimately going to be implemented in the final product as well as pseudocode and screen layouts when dealing with application development for consumer use. When approaching this phase in agile development, the sprints are commonly used to get one or two of those goals that can realistically be implemented and tested by the end of each sprint.

## **7.2.3 Implementation**

The implementation phase of the software development life cycle is where the actual coding gets written.

## **7.2.4 Testing & Integration**

The testing and integration phase of the software development life cycle is a key section to get the project up and running. During this phase, all the code has to be tested, and the different sections of the software development have to be integrated. In this case, the microcontroller code has to properly integrate with the web communications & hosting section of the project in order for the dataflow to be able to reach the mobile device application. This will not be as critical during the early sprints of development since the integral sections of code that have to be integrated to work seamlessly will probably not have been produced yet. The later sprints of development become really critical during this phase of development in order to produce a fully working prototype at the end of the project.

## **7.2.5 Maintenance**

The maintenance phase of the software development life cycle involves continuous evaluation of the software in order to work out any bugs or improve on issues that can be improved on. Changes to the original software can also occur during this final stage of the development life cycle.

## **7.3 Microcontroller Software**

The microcontroller software will all be designed using Arduino IDE. The basic functions along with variables can be found in Table 7.3.1-1 and Table 7.3.3-1 and in the following sections.

### **7.3.1 Keypad Entries**

The keypad contains an array of keystrokes entered by the user. These keystrokes are then sent to a validate function in order to check that the code was correct. This validate function will return either true or false depending on if the code was in deed correct. If the code was correct the command to open the locking mechanism will be sent. Table 7.3.1-1 Keypad Entries Function Structure.

**Table 7.3.1-1 Keypad Entries Function Structure**

<b>Variable</b>	<b>Function</b>	<b>Value</b>
Keystrokes	Stores the users entered keystrokes on the keypad.	An array of integers between 0 and 9
Validate	Serves to check the entered keystrokes against the access code stored in database.	Boolean

### 7.3.2 Lock Status

The lock status will either be 0 or 1 (locked or unlocked respectfully). If the lock status is set to 0 by the user, the lock will engage. Inversely if the lock is set to 1 the lock will disengage allowing access. This scenario will likely be called by a correct keypad entry or override from the computer or mobile application. Either 0 or 1 (locked or unlocked).

### 7.3.3 Lid Sensor

Contains a status variable which will be binary 0 corresponding to an open state and 1 corresponding to a closed state. Upon being opened calls a function to send a notification to the user as well as updating the status to being opened. Upon being shut the status updates again reflecting the now closed status.

**Table 7.3.3-1 Lid Sensor Function Structure**

<b>Variable</b>	<b>Function</b>	<b>Value</b>
Status	Holds the status of the lid (open or closed).	0 or 1
sendLidNotification	Sends a notification to the user upon lid opening.	N/A

### 7.3.4 Alarm

If the IMU goes out of range the alarm will engage. Is also a binary variable with 0 representing an alarm off state and 1 representing an alarm on state. Additionally, has an internal timeout variable which will allow the alarm to deactivate after a set interval. This way if the alarm is tripped and the owner is not home it does not disturb those around for an uncomfortable amount of time. If the alarm is engaged a notification is sent to the user.

**Table 7.3.4-1 Alarm Function Structure**

Variable	Function	Value
Engage	Serves to sound alarm or remain silent.	0 or 1
sendAlarmNotification	Sends a notification to the user upon alarm sounding.	N/A
Timeout	Has the time in seconds before alarm disengages automatically.	Integer

### 7.3.5 IMU

- Contains an array of safe permitted values.
- Compares current values of lockbox to the permitted values.
- If the values exceed the permitted values sends alert to user and sounds alarm.

**Table 7.3.5-1 IMU Function Structure**

Variable	Function	Value
permittedValues	Serves to hold the range of accepted values that will not cause an alarm or alert.	Array of values
currentValues	Contains the current values the IMU is recording on the box.	Array of values
compareValues	Compares the current values to the permitted values. If the current values exceed the permitted the alarm is engaged and alert is sent out.	0 or 1 (0 being within permitted range, 1 being outside permitted range)
sendImuNotificaion	Sends alert too user that the IMU has been triggered	N/A
engageAlarm	Triggers the alarm	0 or 1

### 7.3.1 Modes of Operation

**Low Power Mode:** A low power mode state could be used in order to drastically reduce power consumption. Reduced power would benefit not only the user in terms of their expenses in electrical utilities but also if the device were to become disconnected from its primary power source (wall outlet) the backup battery would drain at a slower rate. There are a few ways to implement this on the Arduino Uno we have chosen. Lowering the voltage settings, we can adjust between 5V and 3.3V doing so drastically reduces the power consumption with just a few tweaks. In addition to this the clock speed can also be reduced. This will in turn reduce the amount of processes the microcontroller does per second and therefore reduce the amount of power needed. Each process it does requires a small amount of power. However, lowering the clock speed does have drawbacks in system responsiveness and in some user features. For example, if the clock speed is reduced too much the user may experience some “lag” or increased wait times for response in wireless communication to the lockbox. The two examples previously stated are hardware modifications, the last modification that can be made is a bit of software. By implementing a low power mode function which puts the microcontroller into a sleep mode. From here we can choose which peripherals and which parts of the microcontroller to disable while

sleeping. To wake from the sleep mode, we will send an interrupt which sets the system back to normal power mode and resumes all functions. The interrupt in our case will most likely be a keypad entry. As soon as a key entry is registered an interrupt will turn the system to normal power mode.

**Security:** Different security modes may be available to the user of the lockbox if they so choose. These modes will require a bit of setup on the user's part but will offer more versatility in control and maintaining their lockbox. Upon registering their lockbox, a user will be prompted to enter an administrative password for their lockbox. This will be their code to open the lockbox which they may change at a different date if necessary. From here their lockbox will be fully functional. However, there is the option to set up a secondary password which may be given to close friends or relatives which the user chooses to allow access. This gives permission to other people to open the box but the owner never has to give out their admin password. Additionally, the user may set up a 1-time use password. This password will be tied to a counter which upon activation the counter will decrement to zero and be deleted. This 1-time password will allow for delivery drivers or other drop-off scenarios to occur without the need to change your password every time. The 1-time password will be generated by a random number generator and will be different every time.

**Power:** Primary power mode will be active so long as the lockbox is drawing power from the power outlet. This will be monitored by a power sensor which will in turn have some software attached to it on the backend. If the lockbox ever becomes disconnected from the primary power source (the power outlet), the lockbox will switch to secondary power. This will be the two Uxcell 3.7V batteries connected in series which will be housed with the microcontroller. The microcontroller will use a plug-in adaptor in order to receive power from a standard wall outlet.

Overall, the methods of operation are meant to give users flexibility and enhance the quality of our product. The security modes help to ensure that the product is secure and that the users investments are protected and manageable. The power modes help the make our device more sustainable which in turn provides better user experience, longer battery life, and cheaper cost for the user. The different power source modes help ensure that our Smart Package Lockbox stays up and running even during loss of primary power. This means if something goes wrong with the outlet, the user needs to move the box away from the outlet and doesn't have a primary power source, or an emergency happens and the primary power source is lost, that our user can still operate their lockbox.

## 7.4 Web Communications & Hosting

### Operating System

The operating system (OS) serves as the lowest layer of the stack. It serves as the basis that the server will operate on.

### Web Server

The web server is the shell that contains all the files related the application. This could include HTML documents, CSS stylesheets, images, JavaScript files, ect. The server controls how users can access hosted files. At a minimum the web server should support HTTP requests. This means that it understands URLs (web addresses) and HTTP (the protocol browsers use to view web pages). Based on the web server and the developer it may have increased functionality and support a broader range of files.

### Programming Language

The programming language is the dynamic language used so that a user can navigate a web page. Web pages can be broken down into two types of pages: static and dynamic. A static page does not support any sort of changing elements such as those done in PHP, ASP, or JSP. This means that the site cannot pull data from a database and add it to the page itself. Advantages of a static website are that the costs are usually lower and it allows flexibility and individual design between pages. The disadvantages are increased cost over time through changes and scalability becomes a problem. If you want something updated it needs to be changed in the raw HTML. On the other hand, dynamic sites are usually more expensive or time consuming up front but they pay off in the long run and in scalability. They can be connected to a database and from there update themselves. This means if a user adds something and wants it displayed on the site it can be done dynamically and on the fly without having to be done by a developer on the backend.

### Database Software

The database software serves as a means to effectively store information. With the software you can tie specific information to user accounts or create different relationships between the data stored. For example, a user can be connected to the items he has for sale or a credit card that is saved on file.

The table below contains a breakdown of the WISA and LAMP stacks with their individual components.



**Table 7.4-1 Database Software**

<b>Stack</b>	<b>Operating System</b>	<b>Programming Language</b>	<b>Database Software</b>	<b>Web Server</b>
WISA	Windows	ASP.net	SQL Server	IIS (Internet Information Services)
LAMP	Linux	PHP	MySQL	Apache

The MEAN stack is not included in the above table because it follows a less rigid structure. MEAN is broken down below.

- **Angular** front-end web framework. This runs your JavaScript code within the browser and allows the application User Interface (UI) to be dynamic for the user.
- **Express** is a back-end web application framework. It runs on top of Node.js
- **Node.js** is a JavaScript runtime environment that lets you implement your application backend in JavaScript.
- **MongoDB** is the back-end database which stores your data as JSON (JavaScript Object Notation) documents. Saving in JSON allows the data to be sent between front-end and back-end applications easily.

### Choosing A Stack

In choosing between each stack it is important to consider the purpose of your application. If cost is not a big issue or if you plan on going commercial in the application then WISA is probably the best bet for you. Most of the components of the WISA stack are maintained and included in Windows by Microsoft. This being said, they are designed to fit together very well and work seamlessly with each other in development. However, this ease of use doesn't come free. The initial costs and licenses to create an application on the WISA stack can be costly. On the other hand, the LAMP stack in the spirit of Linux is all open source and free. The "no cost" factor is huge to first time developers or non-commercial developers.

The big advantage to the MEAN stack is that both client-side and server-side applications can be written in one language. However, of the three stacks it is the newest and some documentation and assistance can be scarce depending on the application you are developing.

Consider what your team strengths are in regard to the components of the stack as this can be a determining factor in which stack you go with. For example, if you are well versed in PHP you should lean towards the LAMP stack as PHP is not supported in the WISA stack.

It is important to keep in mind that the stacks themselves are not static. Certain pieces can be swapped out or changed depending on what you or your team are comfortable with. For example, if your team is primarily built up of proficient windows users but are well adapted to PHP you may go with a WAMP stack.

## 7.5 Web Development Revised

This section discusses the software tools that were implemented in the final revision of the Smart Package Lockbox website and database backend.

### Firestore

Firestore was used as an alternative to the stack architecture discussed before. This is due to a number of reasons and advantages that Firestore provides. First of which Firestore is a free realtime database. This means that any changes that are applied happen almost instantaneously which is great for ease of use and responsiveness. Given that we were sending real time updates to the user this was a great advantage which we took up on. Additionally, Firestore has built in libraries and functionality which make it easy to use and implement. This was a huge advantage over developing our own stack and took an enormous workload off of designing the backend and JSON communication as well as API.

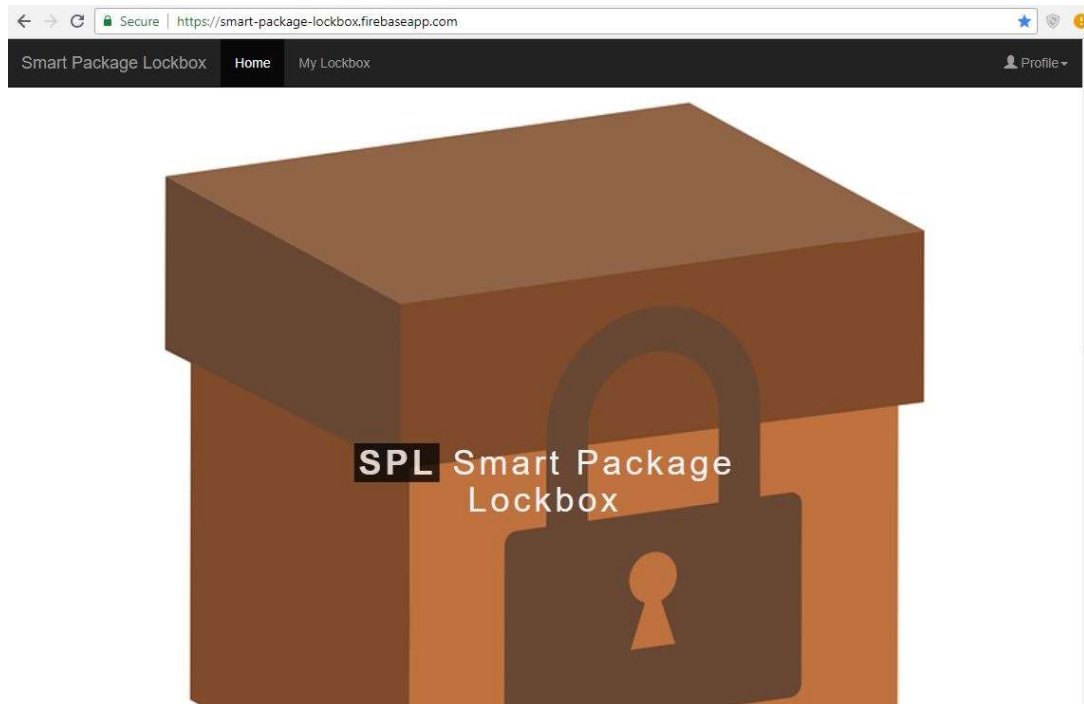
### Firestore & ESP8266

In addition to Firestore being used as the database it was also used in coding the ESP8266 to communicate between the database and the MCU. Through the use of the FirestoreArduino and ArduinoJSON libraries commands to get and set information from the database were made simple. As stated before in prior sections Firestore is also a realtime database. This means that the get and set methods performed in real time visibly on the back end and had rapid response times. This helped to allow users to be notified when specific actions were done on their box such as the lid being opened or the keypad becoming unlocked.

### Website Revisions

Originally, the plan was for at registration the user must provide an email, password, username, and phone number. It was decided as an ease of use route to eliminate the phone number and username requirement. The email was adopted to just act as the username field followed by their password. The phone number was not altogether gotten rid of. It still is available to add later as a feature which can be used to receive text notifications through

Trello if the user wishes. The final website login as well as the home page can be seen in the figures 7.5-1 and 7.5-2 below. Figure 7.5-1 shows the home page once a user successfully registers and logs in to our website. Figure 7.5-2 shows the login screen as well as registration which the user will be automatically redirected to if they are not currently logged in.



**Figure 7.5-1 Website Home Page**

A screenshot of the website's login and registration page. The browser address bar shows 'https://smart-package-lockbox.firebaseio.com/login.html'. The page has a dark header with 'Smart Package Lockbox'. Below the header, it says 'Welcome to Smart Package Lockbox!' followed by a smaller line of text: 'Returning user? Login below otherwise register account and begin the process to secure your deliveries.' The page is divided into two columns: 'Login' and 'Register'. The 'Login' column has fields for 'Email' and 'Password', both marked 'Required', and a 'Sign in' button. The 'Register' column has fields for 'Email', 'Password', and 'Confirm Password', all marked 'Required', and a 'Register' button.

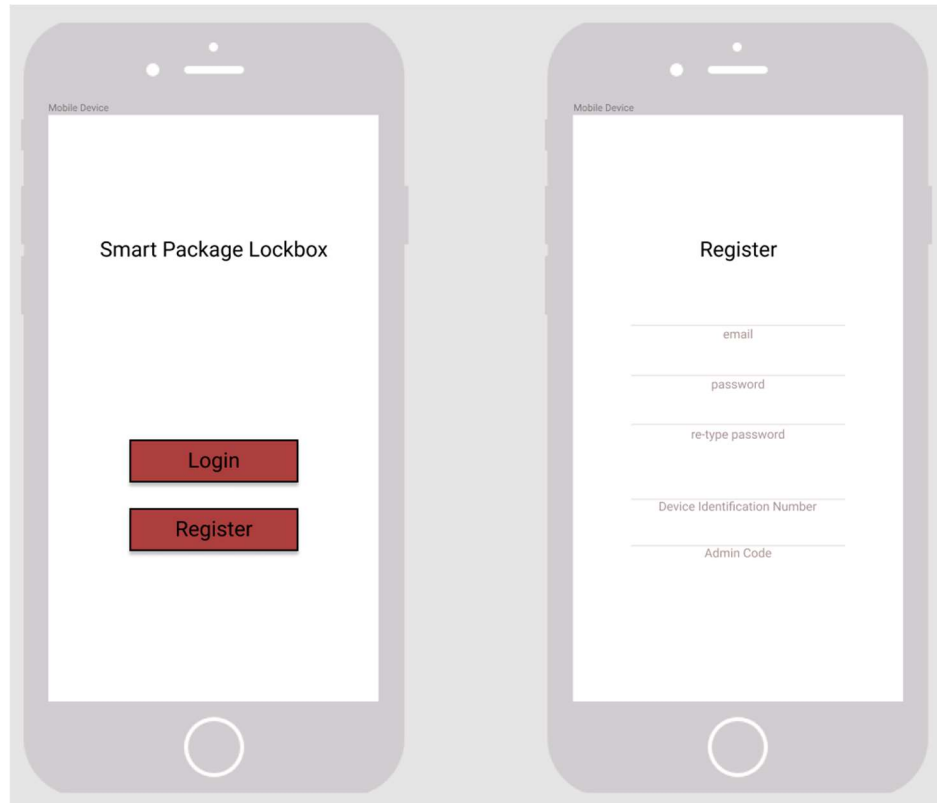
**Figure 7.5-2 Website Login Page**

Currently, the only requirements are that the password be 8 characters minimum and the email must be a valid email. Email verification as well as additional password security features may be implemented easily at a later date. However, for demo purposes as well as ease of use for the user these features were left out of revision 1 of Smart Package Lockbox.

## **7.6 Mobile Device Application**

As was discussed in the software research section of this document, the mobile device application for our project will be developed using React Native so we can develop an application from the start that is cross platform. This will allow us to take advantage of all users of the Apple ecosystem as well as all the users from Google's Android ecosystem and reach as many people as possible. It is important to note that all of the software design that will be talked about in this section is strictly preliminary and as the team moves forward into the second semester of senior design, some of the design choices will be subject to change at any given time. Attention to detail will be kept in order to try and maintain this document as up to date as possible.

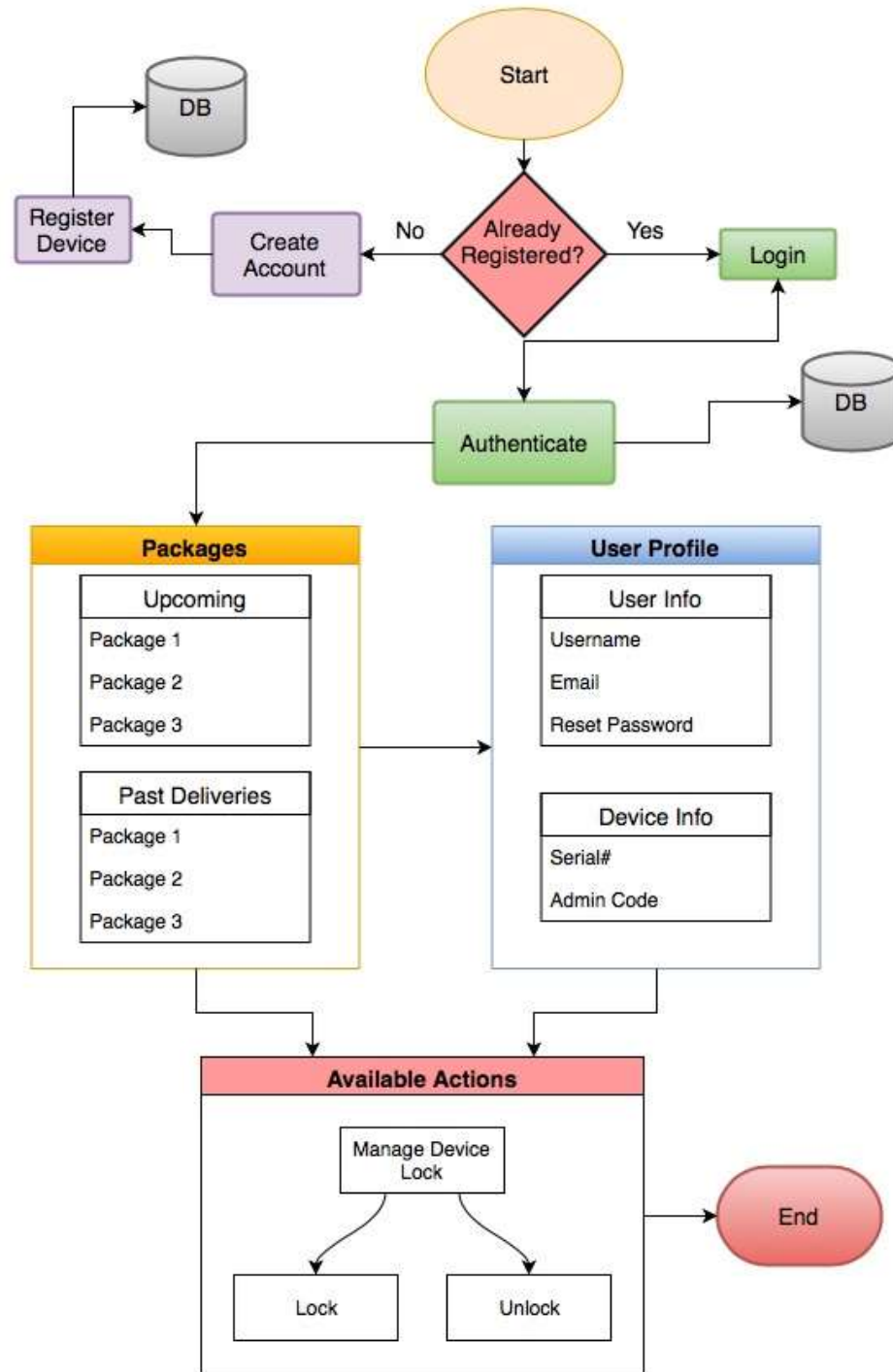
The application will be done in a somewhat simple design with the necessary features to create a satisfactory user experience. Upon starting the application, the user will be greeted by a login screen. In case the user is a new customer and has not yet registered, they will have the option to create an account. Creating an account will prompt the user for an email and password requiring him to retype the password in a second password field. Subsequently the application will attempt to gather the user's device identification number or serial number in order to attach that device to that user's profile and store all of that information in the database. An example of the user interface is shown on Figure 7.5-1.



**Figure 7.6-1 Mobile App Login UI**

Once logged in, the user will land on their main application page which is also the *Packages* page. This page should contain all of the information of that user's upcoming packages with a description of the package, its expected delivery date, and the code type that is assigned for that package to be able to open the Smart Package Lockbox. A section with past deliveries will also be present in the packages page with the available information pertaining to those packages as well. All of the information on the packages page should be able to be managed and edited from inside the mobile application. Code types can be assigned or reassigned to upcoming packages.

The software flow diagram for the mobile application is depicted in the following Figure 7.5-2.



**Figure 7.6-2 Mobile App Flow Diagram**

From the packages page of the application, the user has access to their lockbox's locking mechanism. There will be a button that lets the user unlock or lock the lockbox that should also reflect the current status of the lockbox, whether its locked or unlocked. From here,

the user can also access their profile page which will serve as the account page for the user to either unregister a device, register a new one, change his administrator code to open the lockbox, as well as edit his email information or password for the application. The profile page will also have access to the locking mechanism.

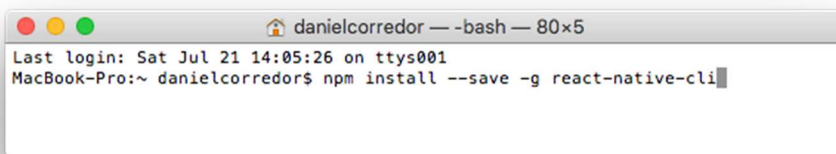
## 7.7 Mobile App Development (SD2)

After careful consideration of the alternatives for mobile app development with react-native with regards to the integrated development environment (IDE), a choice was made for Atom. Since this particular IDE was regarded online as one of the best, if not the best choice for react-native development because of its integration to react and the react-native framework, development went underway.

The beginning of development on a new platform, framework, and most importantly, a new programming language that the main developer for the mobile application in this project did not have any experience in will obviously bring many challenges and unforeseen delays when it comes to the expected time to completion and debugging mentioned for this portion of the project later on in this documentation in reference to the project milestones.

It should be mentioned that in preparation to make the mobile application for this project, the student in charge of this portion of the project decided to accelerate his knowledge in JavaScript, react-native, react, and redux by engaging in an extracurricular course online on those specific subjects in order to gain inroads into developing a mobile application with react-native, and facilitate the process. The online course touched on key subjects that could have been really tricky to get a grasp on by simply reading online forums or following video tutorials on each of the different subjects. By following a comprehensive course encompassing all of the aforementioned topics, the student was able to properly shine a light on the basics of the development process for a react-native mobile application.

The very first step in order to create a react-native application for mobile consists of actually installing react-native and all of its dependencies on the development machine. Usually, installing programs on a computer is a trivial process that most people go through every day, but the process of installing react-native on a machine is a bit different than that. Different packages have to all be installed individually through the command line by using a package manager program. In this case, npm. Npm is the default package manager for the JavaScript runtime environment Node.js. On Figure 7.6-1, an example of a command line argument is shown on how to install a package using the npm package manager.

A screenshot of a macOS terminal window. The title bar shows the user 'danielcorredor' and the shell '-bash' with a window size of '80x5'. The terminal text shows the last login time as 'Sat Jul 21 14:05:26 on ttys001' and the current command being executed: 'npm install --save -g react-native-cli'. The cursor is at the end of the command line.

```
danielcorredor -- -bash -- 80x5
Last login: Sat Jul 21 14:05:26 on ttys001
MacBook-Pro:~ danielcorredor$ npm install --save -g react-native-cli
```

**Figure 7.7-1 React-Native Install with npm**

After installing react-native and all the required packages for the projects mobile application, the Atom IDE had to be setup to be used with react-native. Actual setup was fairly straight forward by just adding the project folder onto the IDE and going from there. The key to make development easier for react-native came in installing some specific add-ons to the Atom IDE from the seemingly infinite library of add-ons found online that are specifically designed to augment the functionality of the IDE in order to specifically meet the needs of the developer.

The most important of these add-ons is called a linter tool. In this case, a specific linter tool called eslint. This tool is in charge of analyzing the developers source code to flag any unseen programming errors, bugs, stylistic errors, or suspicious constructs. Even these linter tools are extremely modifiable to the point where you can assign a set of rules for this linter tool to follow per project folder. This means that the linter tool will analyze the source code for errors in a specific way for this project. This proved to be an invaluable tool that would have probably gone unseen if not for the online course taken in preparation for this project.

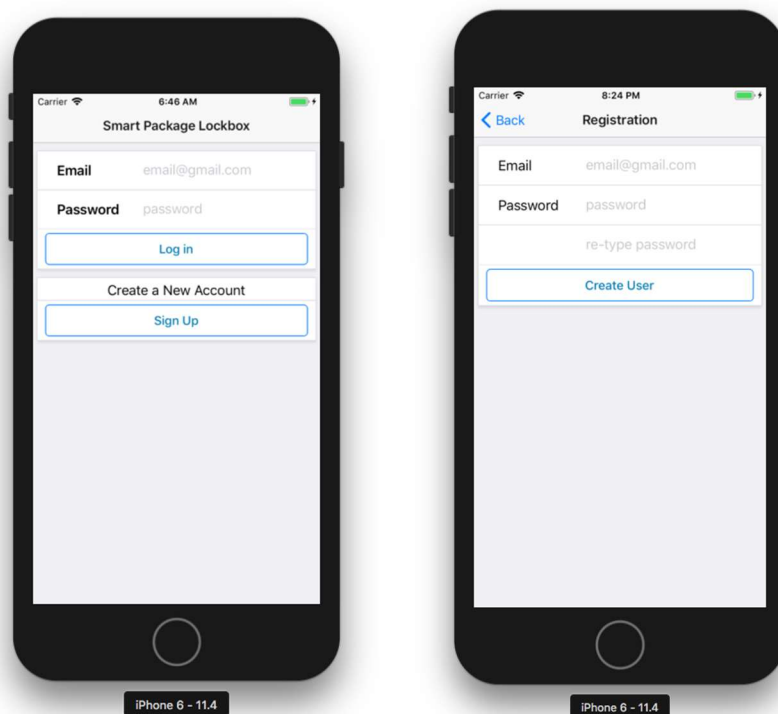
The online course taken by the student required the student to build 4 different applications from scratch using react-native. All these applications served different purposes in order to learn and understand the basics of programming in JavaScript and using React and react-native.

The first of these applications concentrated on being able to pull information from the web, and display it back on the screen. Whether it was an image, or simply text to display back to the user. The second application dealt with authenticating a user through a login screen. Putting the student through developing this second application was extremely beneficial when it came to this project's mobile application because it introduced us to Google's mobile and web application development platform called Firebase. Firebase can be integrated into any react-native app very easily, giving our application access to cloud messaging, authentication services, a database, online storage, as well as hosting. The third application helped understand how to display a larger amount of information without slowing down the phone by only loading what was currently visible on the screen. The last application created in the course implemented most of the concepts introduced with the



first three applications, but it also introduced saving and reading data from the firebase realtime database. This is something that we knew we were going to need for our smart package lockbox mobile application.

The actual development of the smart package lockbox mobile application began as soon as the online course was finished by the student developer. It proved to bring invaluable information and experience when actually beginning to develop the application. A login screen and sign up/register screen was properly developed with authentication. An example of the finished mobile app's authentication screens can be seen in Figure 7.6-2.

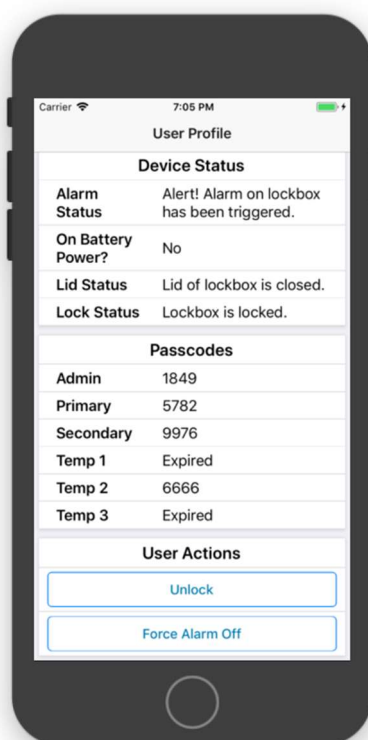


**Figure 7.7-2 Mobile App Authentication**

The authentication screens require a user to login by using their email and password, or to register a new account with the service by providing a new email account and password with the requirement to retype the password in order to avoid a mistake is made when typing in the password. Since authentication works through the authentication services provided by firebase, we are able to take any error messages that firebase spits back when a user is trying to log in or create a new account so that we can dynamically display those messages back to the user. This allows the user to identify what portion of his credentials, whether it's the email, or the password, that he is currently entering incorrectly.

Once the user has logged in, the user is able to see the status of his currently assigned smart package lockbox, as well as his current list of passcodes to successfully unlock the box.

This will be the user's profile screen. There are currently six different passcodes that are available for the user to see when he logs into his mobile application. The first is an administrator code that will unlock the box without the need for the WiFi module to be connected to the internet. This code will serve as a backup, in case the WiFi is down and the code currently cannot be changed. The second is the user's primary code for the box. The third code is the secondary code, and the last three codes are temporary one-time-use codes that will expire as soon as they are used for the first time. The latter five codes mentioned all have the ability to be changed from within the app itself to any four digit code the user desires. An example of what the mobile app's user profile screen can be seen in Figure 7.6-3.



**Figure 7.7-3 Mobile App User Profile**

As can be seen in Figure 7.6-3, the user will have access in realtime thanks to the firebase realtime database to a variety of different statuses regarding the conditions of his lockbox. The user's passcodes will be displayed to him, while also giving the user a set of actions that he can execute from the mobile application itself.

For now, the set of actions that the user will be able to execute are the ability to unlock his box remotely from literally anywhere that has a working internet connection, and also remotely turning off the box's alarm when it is going off.

Some of the functions that we wanted the mobile application to be able to accomplish included notifying the user via push notifications any alerts that the box might be experiencing. While we found out that actually implementing these notifications was not technically a difficult task, we were limited by Apple for not having a paid developer license with them. This means that our application would be denied the push notification service directly from Apple, rendering it useless for us to implement it for demonstration purposes.

## **8.0 Prototype Construction**

This section details the construction of the prototypes for the Smart Package Lockbox as well as presents the complete PCB schematic. For the pre-PCB prototype of the system, an Arduino UNO development board, shown in Figure 4.2.9.3-1, will be used as well as pin module versions of the ESP8266 and MPU-6050. However, the Arduino UNO will also likely be used to test the surface mount variants of these parts before they are attached to the prototype PCB.

### **8.1 PCB Vender & Assembly**

The primary PCB vendors explored as potential options for the system prototype PCB boards are PCBWay.com and JLCPCB.com. Both services offer low cost 2-layer prototype PCB boards. However, PCBWay.com ships the PCB boards from China and would have long shipping delays as a result. JLCPCB.com offers fast 2-day shipping, but comes at a much higher cost than the alternative. Efforts will be made to leave time so that PCBWay.com can be used, however, a more expensive service with a shorter shipping period will be utilized if needed. It should be noted, however, that shipping costs will not factor into the material costs for requirement C2 listed in Section 2.3.2.5.

For the mounting of the components onto the PCB board, Quality Manufacturing Services in Lake Mary Florida will be utilized. We have been informed that typical costs for this service are around \$2 per part, however, it was also stated that high discounts being provided for students is likely.

### **8.2 Schematics**

The complete initial and revised schematics for the Smart Package Lockbox system are displayed below In Figure 8.2-1 and Figure 8.2-2.

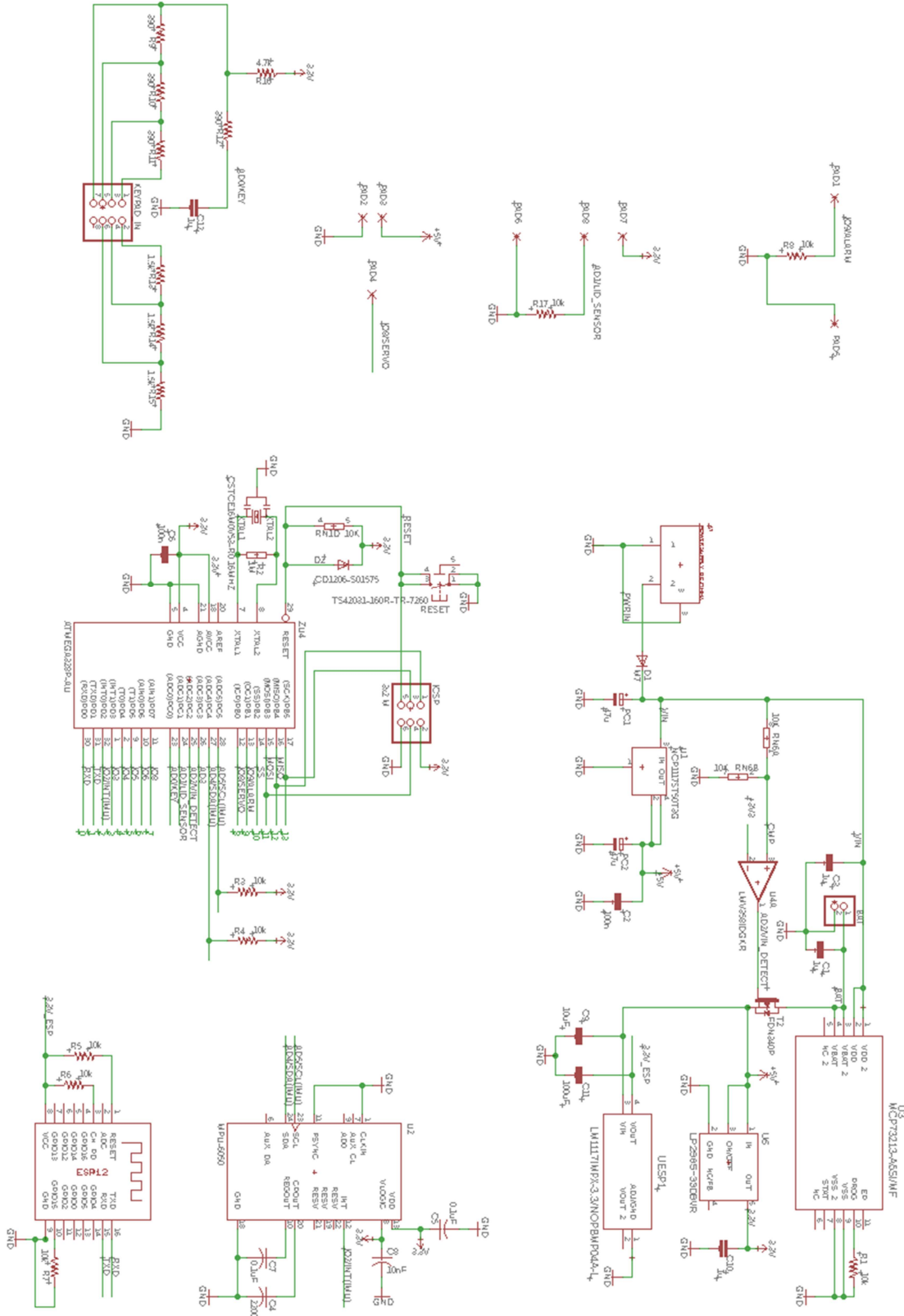


Figure 8.2-1 Initial System Schematic



## 8.3 Coding Plan

The coding plan for this senior design project will consist of three different coding environments that pertain to their subsection of the project. The development of code will be split between the mobile device application, the web communications and hosting services, as well as the lower level coding for the microcontroller(s).

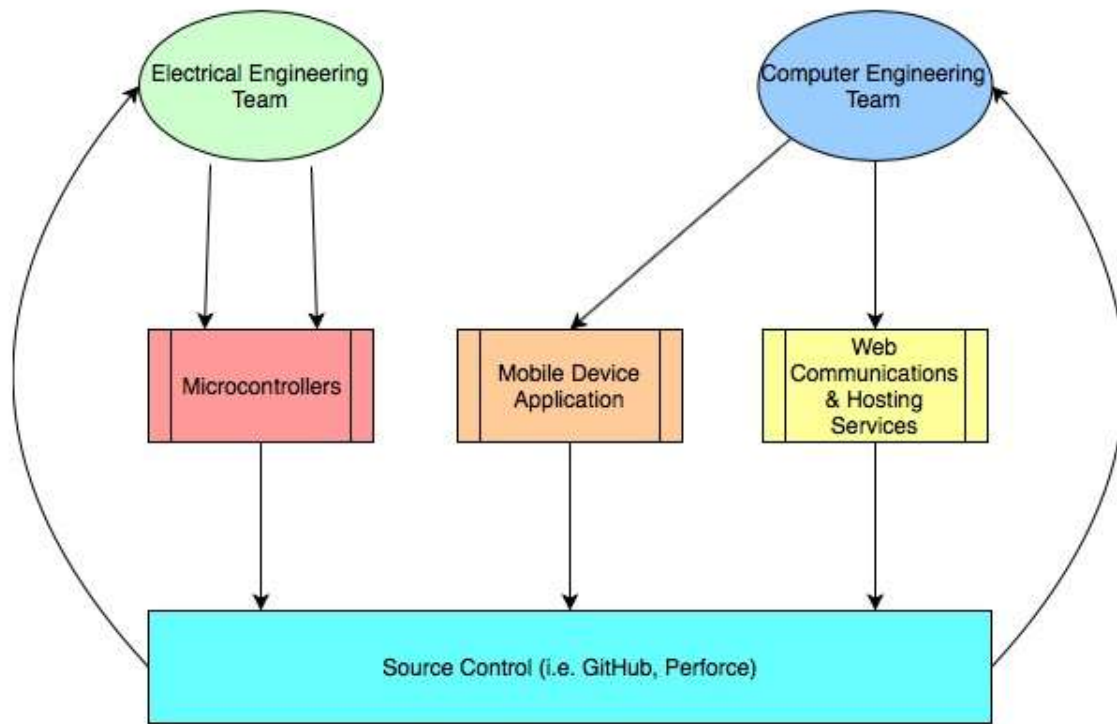
Since the four members of the project are divided between two computer engineering students and two electrical engineering students, they will henceforth be comprised of a computer engineering team and an electrical engineering team. All students' work towards the completion of the software development portion of the project will be split according to their knowledge and ability on the subject.

The electrical engineering team, having a background in the C programming language as well as microcontrollers, will dedicate their efforts on developing the code for the selected microcontroller.

The computer engineering team, having a more extensive background in several programming languages and programming platforms, will each work individually on the remaining portions of the project. One member will focus on developing the web communications and hosting services, while the other member will focus on developing the mobile device application.

With that being said, the separation of tasks regarding the software development of this senior design project does not signify that any of the team members are excluded from collaborating or helping another team member in another section of software development. It is important for all members of this project to be well informed of most aspects of the software development to avoid the delay of progress on the project in case of an emergency situation. This will help to ensure that no single team member is absolutely indispensable and development would be able to continue on the code relatively unhindered in the event of a team member's absence. The following Figure 8.3-1 shows the workflow of the coding plan.

## Software Coding Plan



**Figure 8.3-1 Coding Plan Workflow**

With all team members contributing code in one way or another to their respective or designated section of the project's software development, they should all still have access to each other's code at any given moment. In order to accomplish this, all team members will adopt using an online web repository like GitHub, or Perforce Software as a coding standard in order to give their team members access to the code for any contributions. This will also serve as source control for all of the code. A positive byproduct of giving access to everyone's code to all members is that team members will be able to keep each other in check and monitor progress of other team members during the development of code. This will facilitate meeting all the project milestones and deadlines.

Since the summer semester only allows for 12 weeks of development, most if not all of the software development will have to be completed by the eighth week. This includes requiring all pertaining sections of software development to have gone through an extensive set of testing and verification that will be defined in section 8 of this document. The remaining four weeks of development will be used for debugging purposes only.

## 9.0 Prototyping

The following sections document the creation and troubleshooting of the project prototypes. The testing and verification of the project prototypes are discussed in detail in Section 10.

### 9.1 Breadboard Prototype

For the breadboard prototyping we focused on the individual components then combined them into one coherent system that would take the comprised code and be able to interact with the database and app. The total circuit takes the keypads pin input, imu input, alarm and lid sensor to tell the system when the box is being tampered with or the box has been approved to continue with unlocking the locking mechanism to put the package into the box. This is all done seamlessly with the database and app through the wifi module which reads the possible inputs and accepts primary, secondary, and temporary codes. The app also notifies the user when the box has been opened or the alarm has gone off as well as the website itself. In Figure 9.1-1 below shows the overall breadboard prototype with all components combined.

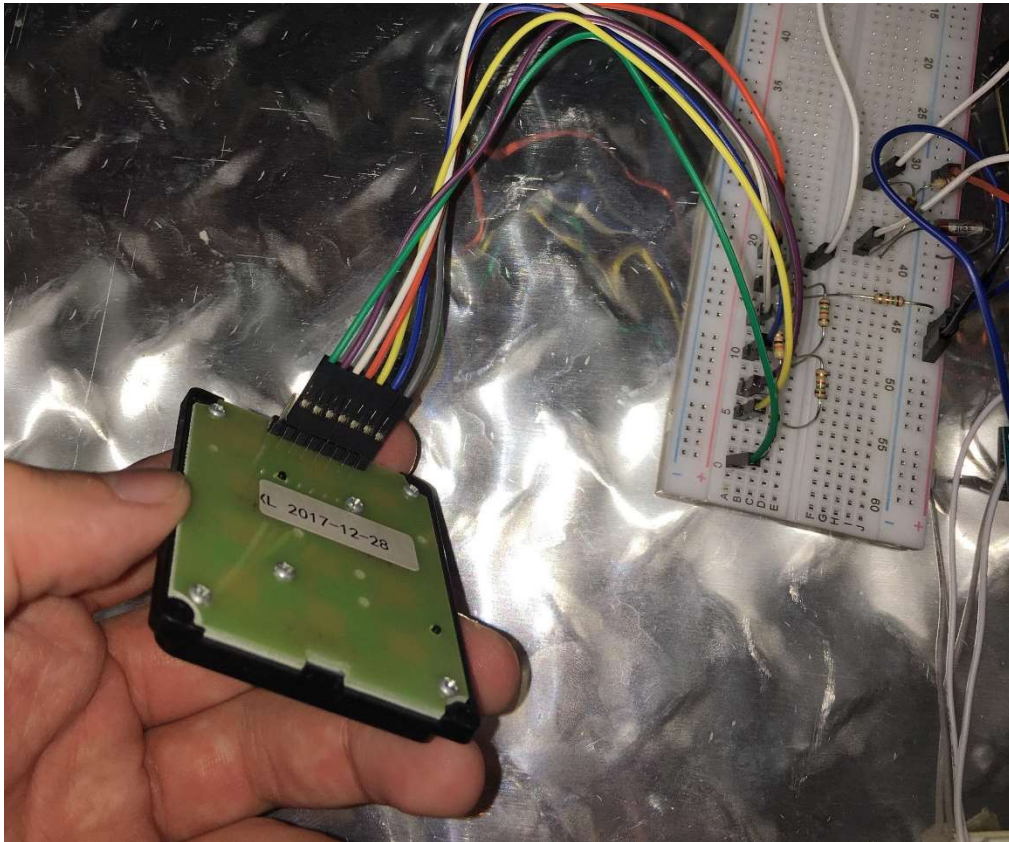


**Figure 9.1-1 Breadboard Prototype**



## 9.1.1 Keypad

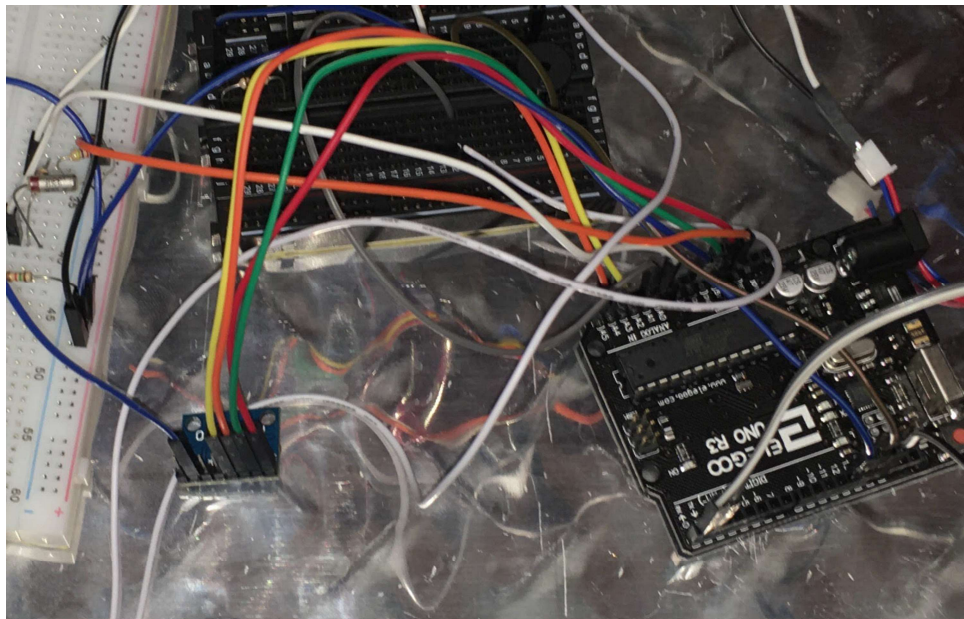
The keypad portion of the prototype took a lot of debugging and testing to get the correct values to be represented onto the serial monitor for real value representation. We first started by changing the keypads 8 pins into 1 pin through voltage division. This was achieved by connecting the appropriate pins to their corresponding assigned resistor values. The keypad needed to be made sure to include the correct pin orientation to not get the rows and columns confused with one another as well as attributing the correct pin to the right components. Once the circuit was complete and connected to the Arduino we had to read the resistance values being displayed on the serial monitor and assign them a number value when a button is pressed. The main issue we experienced in this section was the resistors in the breadboard prototype are not near perfect resistors like the PCB so the values fluctuate and we had to make sure the code represented that through the process of using if statements to look at numbers close by to the originally pressed value and have the same number assigned. We also had to utilize a debounce to counter the pressure sensitive nature of the keypad and eventually through all of this work we got the keypad to correctly read the right values and send the data to the database. In Figure 9.1.1-1 below shows the keypads pin and circuitry.



**Figure 9.1.1-1 Keypad Connections and Circuit**

## 9.1.2 Inertial Measurement Unit

The inertial measurement unit was less circuitry and more code analysis to get the right optimization for our box to read the data from the unit being moved around and send a warning to the user and set off the alarm if moving too much during an extended period of time. This was achieved by precise coordinate checks and statements commenting about acceleration to make sure the box is in a standstill mode as to not send an alert if the box is not moving. In Figure 9.1.2-1 below it shows the IMU and its connections to the Arduino.

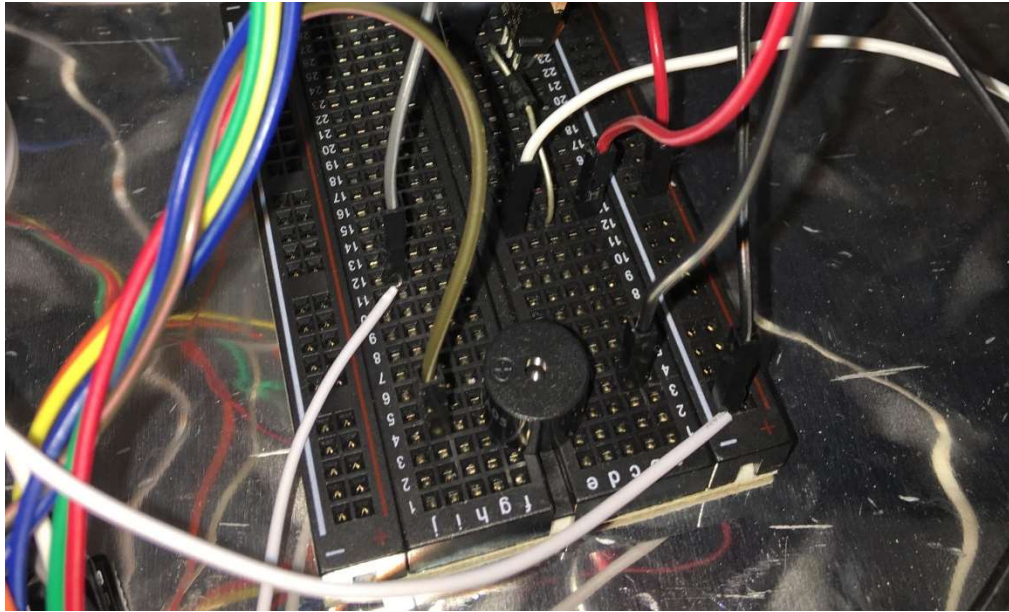


**Figure 9.1.2-1 IMU Connections**

## 9.1.3 Alarm & Lid Sensor

The alarm and lid sensor combined together to get the result of sending an alarm if the box is open for too long or if force opened without authorization from the keypad's input. The lid sensor only takes a distance of 1 inch to recognize if the lid has been separated achieved by placing the magnet underneath the lip of the lid right above the sensing unit. This data is transmitted into Arduino with correspondence to the alarm to issue a statement to set off the alarm when each case is presented where the alarm would need to be set off. These both work in tandem with the rest of the code while also independently checking for initial and simple thief prevention checks. In Figure 9.1.3-1 below shows the alarm and lid sensor portion of the breadboard and the circuitry between them. The white wires on the bottom

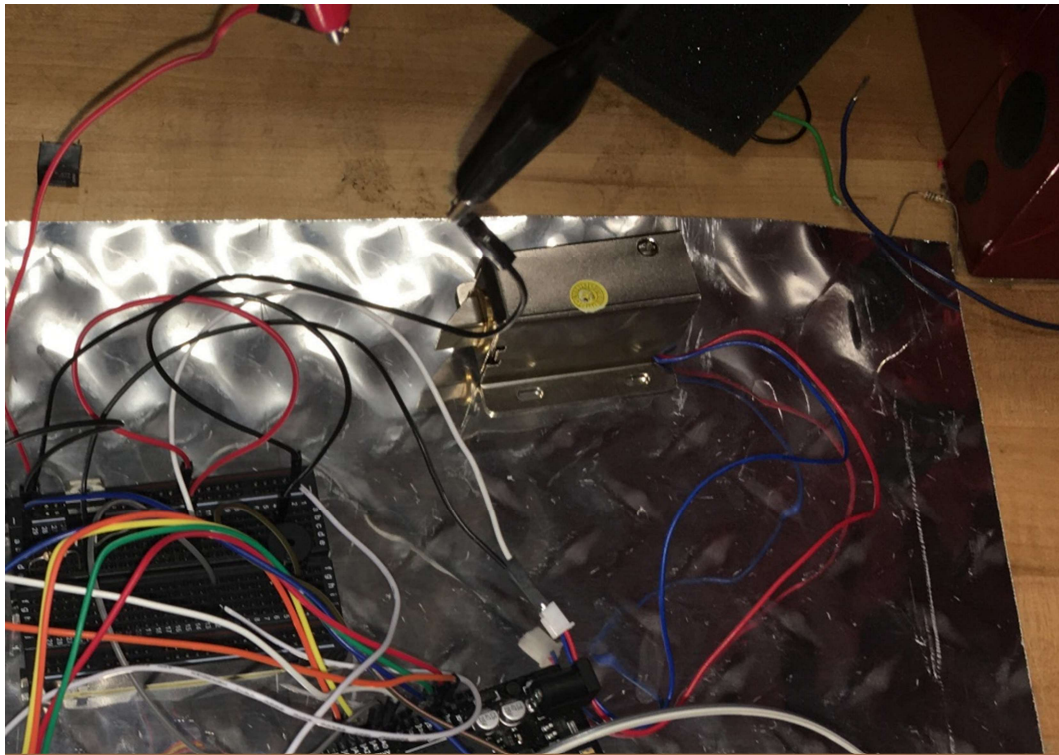
are the input and output of the lid sensor and the alarm is represented as the black circle on the breadboard.



**Figure 9.1.3-1 Lid Sensor and Alarm Circuitry**

## 9.1.4 Locking Mechanism

The locking mechanism was one of the bigger issue sections of the prototyping due to a miscommunication between the lock and the Arduino board. For a majority of the testing phase we were having issues with the lock sputtering upon reset or startup of the board and realized eventually that we were implementing our lock onto the incorrect pin which was pin 13. Pin 13 was issuing a flashing of voltage to the pin every time the code started up which had to be fixed by switching the electronic lock to a less volatile pin. Once that was achieved in worked perfectly with the rest of the circuit and stayed locked even in an off state which is what we wanted to achieve in our final project for security issues. In Figure 9.1.4-1 below shows our electronic lock along with its binary connection, red being connected to a 12V power source and connection to its corresponding Arduino pin.



**Figure 9.1.4-1 Locking Mechanism Connections**

## **9.2 PCB Prototype**

Our PCB prototype had a lot of issues present mostly with flashing the code to the MCU and it is receiving and transmitting the correct results when prerequisites have been met like in the breadboard prototype. Upon retrieval of the finalized board we noticed the power flowed through the circuit correctly and all the connections tested positive for connectivity. We went through desoldering components for troubleshooting purposes to get the PCB design functional and tested through flashing to a bootloader to get the designated code onto the PCB. In Figure 9.2-1 below shows the finalized PCB design without the Wifi Module present at the time.



**Figure 9.2-1 PCB**

## 10.0 Testing & Verification

The following sections will be used to describe and list all of the tests that the group members will run on the specified components of the project in order to verify all components are working as expected.

### 10.1 Table of Tests & Outcomes

In the table that follows a collection of all tests performed during debugging as well as product functionality can be found. Tests range from software to hardware and target both individual components as well as the product as a whole.

**Table 10.1-1 Tests Conducted**

<b>Test Name</b>	<b>Area of Focus</b>	<b>Equipment</b>	<b>Procedure</b>	<b>Outcome</b>
WiFi Connectivity Test	Hardware & Software	Esp8266 NodeMCU module.	The baseline WiFi module code was loaded onto the WiFi module itself. Followed by providing power and checking serial monitor that it successfully connected to either a home network or WiFi hotspot.	The WiFi module successfully connected to the hotspot or home network and displayed it's IP address to the serial monitor.
Lid Sensor & Alarm Test	Hardware & Software	Lid Sensor, Breadboard, Alarm, Arduino Uno with MCU.	The lid sensor as well as the alarm were connected to the proper pins on the MCU followed by flashing the software onto the MCU for the alarm and lid logic. The lid was then opened to test proper functioning of alarm.	Upon opening the lid without a valid authorization being set from the keypad, the alarm was triggered successfully.

**Table 10.1-2 Tests Conducted (Part 2)**

Keypad & Lock Test	Hardware & Software	Keypad, Arduino Uno with MCU, Solenoid Lock.	Several keypad attempts were made in order to attempt to unlock the box. These attempts included alterations between valid and invalid passcodes to test for errors or bugs. Additionally, the passcodes were cycled between primary and secondary passcodes.	Upon a valid keypad entry the box was unlocked for the user. Upon an invalid keypad entry the box was not unlocked.
Passcode Send & Receive Test	Hardware & Software	Esp8266 NodeMCU, Arduino Uno with MCU, keypad, Firebase database	Passcodes were set in the database followed by attempts to open the box via these passcodes using the keypad.	The ESP8266 successfully routed the passcodes from the MCU and compared them to the passcodes stored in Firebase.

**Table 10.1-3 Tests Conducted (Part 3)**

IMU & Alarm Test	Hardware & Software	Arduino Uno with MCU, IMU, breadboard.	The software logic for the IMU was loaded onto the MCU. Following this the IMU was moved constantly for 5 seconds. A separate test was done for motion on and off for 1 second with a 2 second interval pause.	Upon constant motion the IMU successfully tripped the alarm within the 5 second window. For the non-constant motion the IMU was not tripped due to imbedded coding to prevent sudden non constant motion from tripping the alarm.
Realtime Database Test	Hardware & Software	Complete breadboard, Arduino Uno with MCU, IMU, alarm, lid sensor, locking solenoid, Esp8266 NodeMCU, Firebase database	All components of the system were individually tested checking that the proper alerts and statuses were sent and updated in Firebase.	Upon testing each component individually the proper statuses in the proper fields were updated in the database.
Website Integration Test	Software	Website, Firebase database	The website was connected to firebase and tested to make sure login authentication as well as proper user information based on who is logged in was pulled.	Upon successfully fulfilling login parameters the user was logged in and their data was displayed to them from Firebase.



**Table 10.1-4 Tests Conducted (Part 4)**

Mobile App Integration Test	Software	Mobile App, Firebase database	Similar to the website portion the app was connected to firebase and tested for proper functionality with Firebase.	After successful login the user's data was stored and pulled in Firebase correctly.
Push Notifications Mobile App Test	Software	Mobile App, Firebase database	Upon alert statues being tripped such as the alarm or box unlocked, a notification is sent to the user.	Failure due to being a paid feature and must purchase an Apple Developer License for 100\$.
Website Email Test	Software	Website, Firebase database	Upon an alert being triggered an email notification is sent to the users email on file.	Email was successfully sent with status message for what alert was triggered.
Trello Notification Test	Software	Website, Trello, Firebase database	Upon an alert being triggered, a status is sent to the user's primary phone stored in the database with alert message.	The message was successfully sent via Trello text communication to the user.

**Table 10.1-5 Tests Conducted (Part 5)**

Force Unlock & Alarm Off from Web	Software & Hardware	Website, Firebase database, complete breadboard circuit (Esp8266 NodeMCU, Arduino Uno with MCU, lid sensor, alarm, solenoid lock, keypad).	Upon the override functions force unlock or force alarm off being selected from the website checked for proper functionality.	When alarm is triggered the force alarm off successfully kills the alarm so long as the lid sensor is closed. The force unlock does push the solenoid lock to open.
Force Unlock & Alarm Off from App	Software & Hardware	Mobile App, Firebase database, complete breadboard circuit (Esp8266 NodeMCU, Arduino Uno with MCU, lid sensor, alarm, solenoid lock, keypad).	Upon the override functions being selected from the mobile app, again proper functionality is tested.	When alarm is triggered the force alarm off function does terminate the alarm so long as the lid sensor is closed. Force unlock does set the solenoid lock to open.
Backup Power Test	Hardware	Complete breadboard circuit, backup battery	The entire circuit is disconnected from the primary power and adapts secondary power. Test to make sure box retains functionality.	All components of the box function properly. Only functionality that is lost is the solenoid lock remains in a locked status until the box regains primary power.

**Table 10.1-6 Tests Conducted (Part 6)**

WiFi Reconnection Test	Software	Esp8266 NodeMCU	Upon loss of connection checks to see that the module attempts to reconnect and no wrong packets are sent in the process.	Module immediately starts trying to regain connection to original source. Upon reconnection no wrong serial data is sent out.
WiFi Module & MCU Serial Communication Test	Software & Hardware	Esp8266 NodeMCU, Arduino Uno with MCU, breadboard.	The WiFi module and MCU are connected to each other's RX and TX pins. Outgoing serial communication is compared in serial monitor.	The data to and from both modules is sent correctly without any missed characters or wrong characters in the serial buffer.

## 11.0 Administrative Content

The following sections contain the administrative content relating to the smart lockbox project. The primary factors that will determine the success or failure of this senior design project are the budgeting and time management of the group. The following sections contain information regarding the team role assignment, project milestones, and budget.

### 11.1 Role Assignment

The table below details the roles assigned for each member of group five for the senior design 1.

**Table 11.1-1 Role Assignment**

<b>Section</b>	<b>Primary Developer</b>	<b>Secondary Developer</b>
Administrative Tasks	Michael	
PCB Design	Michael	
IMU and Wi-Fi Modules	Michael	Brandon
Other Peripheral Design	Brandon	Michael
Web Development	Dominic	Daniel
Database	Dominic	Daniel
App Development	Daniel	Dominic
Power Circuitry Design	Michael	Brandon

The final work distribution as decided and executed in Senior Design 2 for the fulfillment of the project is displayed in Table 11.1-2 below.

**Table 11.1-2 Final Work Distribution**

	<b>Brandon</b>	<b>Michael</b>	<b>Daniel</b>	<b>Dominic</b>
Power Design	S	P		
PCB Design / Debugging	S	P		
Inertial Measurement Unit	S	P		
Hardware Integration	S	P		
Keypad / Alarm / Lid Sensor / Locking Mechanism	P	S		
Wi-Fi Module		S	S	P
Online Database / Website			S	P
Application			P	S
MCU Software Integration	S	S	P	S

## 11.2 Milestones

Summarized below in tables 10.2-1 and 10.2-2 are the final 2018 spring semester senior design milestones for group 5, and the final 2018 summer semester senior design II milestones.

**Table 11.2-1 Senior Design I Milestones**

#	Milestone	Start Date	End Date	Status
1	Ideas	01/08/2018	01/12/2018	✓
2	Project Selection	01/16/2018	01/25/2018	✓
3	Role Assignment	01/25/2018	01/28/2018	✓
	Project Report			
4	Divide and Conquer	01/16/2018	01/28/2018	✓
5	Divide and Conquer Version 2	03/01/2018	03/11/2018	✓
6	60 Page Draft	03/22/2018	04/09/2018	✓
	Final Report	04/09/2018	04/27/2018	✓
8	Define List of Materials & Revisit Costs	01/25/2018	02/22/2018	✓
9	Define Lockbox Physical Specs	01/25/2018	03/29/2018	✓
10	Final Design Revisions	04/09/2018	04/19/2018	✓
	Break Between Semesters			
11	Hardware Design Review Process	04/30/2018	05/14/2018	

**Table 11.2-2 Senior Design II Milestones**

#	Milestone	Start Date	End Date	Status
1	Gather Materials	05/14/2018	05/28/2018	
2	Assemble Microcontroller Parts	05/14/2018	05/31/2018	
3	Build Website	05/14/2018	06/29/2018	
4	Build Mobile Application	05/14/2018	06/29/2018	
5	Build Database	05/14/2018	06/29/2018	
6	Test Hardware with Arduino	05/28/2018	06/18/2018	
7	Build PCB Schematic & Order PCB	05/14/2018	06/01/2018	
8	Build First Prototype	06/11/2018	06/20/2018	
9	Order PCB Revision 2	06/20/2018	07/05/2018	
10	Debug Code	06/25/2018	07/25/2018	
11	Build and Test Final Prototype	07/10/2018	07/19/2018	
12	Make Final Adjustments	07/15/2018	07/26/2018	
13	Final Project Presentation	07/27/2018	08/01/2018	

### 11.3 Initial Budget & Bill of Materials

This project is not funded, and the cost of development will be absorbed by the students. A breakdown of the initial project budget estimation is provided below in table 9.3-1. A total budget of \$600 is assumed at this time to account for unforeseen expenses or design changes. This cost will be split evenly among the 4 members of the group for an assumed \$150 contribution per student. Although \$600 is allocated for the overall project, the cost of the materials in the end design may not exceed \$250.

**Table 11.3-1 Initial Budget Breakdown**

Part	Quantity	Estimated Cost
Microcontroller PCB Boards	1 - 3	\$20 - \$200
Microcontroller Chip	1	\$1 - \$5
Microcontroller Test Board(s)	1 - 2	\$10 - \$20
Wi-Fi Module	1	\$5 - \$10
Wireless RF Transceivers*	2	\$5 - \$10
Lid sensor	1	\$5 - \$10
Inertial Measurement Unit (IMU)	1	\$10
Keypad	1	\$8 - \$10
Sound Alarm	1	\$1 - \$10
Locking Mechanism	1	\$20 - \$30
Rechargeable Battery	1	\$15 - \$30
Power Adapter	1	\$10 - \$15
Box	1	\$60
Miscellaneous materials	~	\$10 - \$40
Total cost	~	\$175 - \$460

\*May be replaced with wired connection

The final cost per unit after design completion in senior design 2 can be found in the table below 11.3-2. The cost per unit could be reduced even further through longer shipping times as well as buying in bulk. This is not a large scale final product to market price.

**Table 11.3-2 Final Cost Per Unit**

<b>Item</b>	<b>Qty</b>	<b>Cost</b>
Suncast Box	1	\$39.00
IMU	1	\$2.60
Wi-Fi Module	1	\$6.95
MCU	1	\$2.17
Locking Mechanism	1	\$14.95
Keypad	1	\$7.50
Alarm	1	\$2.98
Magnetic Switch	1	\$7.18
PCB	1	\$33.00
Backup Battery	1	\$29.99
Misc. PCB Components		\$28.40
Power Adapter	1	\$11.98
<b>Total:</b>		<b>\$186.70</b>

The finalized project budget is displayed in table 11.3-3 below. This final budget is below our requirement and includes extra supplies which were used for testing by different members of the team. Some of these were one-time costs and the cost of future development would decrease dramatically.



**Table 11.3-3 Final Development Cost**

<b>Item</b>	<b>Qty</b>	<b>Cost</b>
Suncast Box	1	\$39.00
IMU	2	\$2.60
Wi-Fi Module	3	\$6.95
MCU	2	\$2.17
Locking Mechanism	1	\$14.95
Keypad	1	\$7.50
Alarm	1	\$2.98
Magnetic Switch	1	\$7.18
PCB (Advanced Circuits)	3	\$33.00
PCB (PCBWay)	10	\$2.70
Backup Battery	1	\$29.99
Misc. Components		\$89.66
Power Adapter	1	\$11.98
Arduino Prototype Boards	4	\$11.86

**Table 11.3-3 Final Development Cost Cont.**

Demo Buzzer Pack	1	\$12.98
Alligator Clip Pack	1	\$6.99
Breadboard Kit	2	\$11.99
PCB Shipping (Advanced Circuits)		\$97.00
<b>Total:</b>		<b>\$548.02</b>

## 11.3 User Manual

Greetings from the Smart Package Lockbox Development team. First, we would like to thank you for your purchase of your Smart Package Lockbox and hope you will love our product. The following tutorial is designed to aid you with initial product setup.

### Step 1.)

Your Smart Package Lockbox comes pre-assembled and ready to go with all of the hardware components. The first step is to place the box in the desired location and within range of your home WiFi connection.

### Step 2.)

Connect the Smart Package Lockbox to your home or desired WiFi by inputting the domain name followed by the password. The Lockbox will automatically sync to our database and be ready to be linked to you in the following steps.

### Step 3.)

From a computer go to our website at: <https://smart-package-lockbox.firebaseio.com/login.html> and register an account with us if you are a new user. Figure 11.3-1 shows the register portion of our login page. As can be seen from the figure below to register you must input an email as well as a password. The password length must be 8 characters long.

**Register**

Email:	<input type="text" value="Required"/>
Password:	<input type="text" value="Required"/>
Confirm Password:	<input type="text" value="Required"/>
<input type="button" value="Register"/>	

**Figure 11.3-1 Register Account**

**Step 4.)**

Upon successfully registering with us you are now ready to login and begin setting up your account. Proceed to the login portion of our login page and input the credentials you made in step 3. A figure illustrating this step can be seen in Figure 11.3-2 below.



The image shows a login form titled "Login". It features two input fields: "Email:" with the value "test@test.com" and "Password:" with a masked password "\*\*\*\*\*". Below the password field is a "Sign in" button.

**Figure 11.3-2 Login to Account**

After inputting the correct credentials our website will verify your login and you should receive a popup stating that login was successful. A demonstration of the popup is shown in Figure 11.3-3 below.



**Figure 11.3-3 Login Successful**

**Step 5.)**

Congratulations you have successfully created an account and logged into our website! Along the top of our website is the navigation bar which is used to go between pages of our site. The navigation bar is shown in Figure 11.3-4 below.



### Figure 11.3-4 Website Navigation Bar

On the left of the navbar our Brand is displayed as well as there being the Home page and My Lockbox tabs. A breakdown of these two tabs is given below.

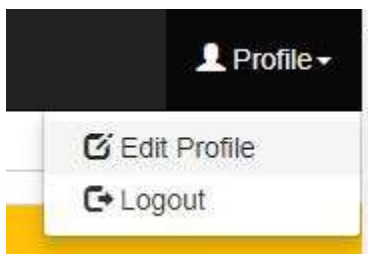
#### *Home Page*

The home page is the first page you are redirected to after login. It displays our logo as well as additional information about our product, creators, and a method of contact for support.

#### *My Lockbox*

The My Lockbox page contains information relative to your lockbox. This is where we will navigate to shortly in order to set up our Lockbox and view its statuses.

On the far left of the Navigation bar the Profile tab is visible. Upon clicking this tab there are two more options given to the user as shown in Figure 11.3-5 below. These options are Edit Profile and Logout which are discussed in more detail below.



### Figure 11.3-5 Profile Dropdown Menu

#### *Edit Profile*

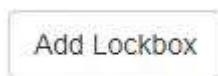
The edit profile page displays current profile information such as email and phone number if one is added (will be discussed later). The edit profile page's main function is to allow the user to update their information such as email, phone, and change their password.

#### *Logout*

Upon clicking the logout link the user will be redirected to the login page in step 4. Logout is used to provide the user with a way to secure their information when they are not at the computer.

**Step 6.)**

Proceed to the My Lockbox tab of our website. From here the page should initially be blank save an add lockbox button and an empty Current Lockboxes header. Select the add lockbox button shown in Figure 11.3-6 below and input your lockbox ID which came with your purchase. This will sync you to the lockbox in the database.



**Figure 11.3-6 Add Lockbox Button**






**Step 7.)**

Now that you have successfully added your lockbox, you will see the lockbox ID followed by the status of different sensors on the box as well as a table of passcodes. The passcodes will be discussed below. Under the passcode table two more buttons Unlock Box and Alarm off can be seen. These are the override buttons which can be called if there is an issue doing them manually from the box or you have someone stopping by and want to unlock it for them without providing a passcode. Figure 11.3-7 below shows a demo My Lockbox page with a user registered to Lockbox 654321 for clarity.

Add Lockbox

Current Lockboxes:

- 654321
  - Alarm Status: N/A
  - Lid Status: N/A
  - Lock Status: N/A
  - Box Unplugged: false

Passcode	Value	Edit
Admin	1849	*
Primary	5782	<input type="text"/> 
Secondary	9976	<input type="text"/> 
Temporary 1	5555	<input type="text"/> 
Temporary 2	6666	<input type="text"/> 
Temporary 3	7777	<input type="text"/> 

Unlock Box

Alarm Off

**Figure 11.3-7 My Lockbox Page**

## Passcodes

The following gives a brief summary of the passcodes as well as their functionality:

### *Admin*

The admin passcode is hard coded to the box. This is the failsafe passcode and should not be given to anyone. It is auto generated and cannot be edited. This passcode works regardless of the boxes connection to WiFi.

### *Primary*

This is the main passcode which provides full functionality and will always be active. Requires connection to WiFi to function properly.

### *Secondary*

This is similar to the primary passcode but can be given to a close friend or family member. Still provides full functionality to the box and requires WiFi.

### *Temporary Passcodes*

All three temporary passcodes have the same functionality. They provide full access one time to the box then are set to expire. These are used for one-time deliveries or drop offs to people you do not want to have access to your Lockbox whenever they please. They will have to be reupdated after use as they are set to expire. Require WiFi to function.

### *Edit Passcodes*

To edit your passcode type in the new passcode in the box and hit the blue edit pencil next to it to send it to the database. Upon success the value field to the left should update with the entered passcode. Keep in mind all passcodes must be numeric and exactly 4 digits long.

## **Step 8.)**

From here your Smart Package Lockbox is ready to go. Additional preferences as well as information can be edited from the edit profile tab under the profile dropdown on the far right of the navigation bar. Clicking this brings you to the page displayed below in Figure 11.3-8.

The screenshot displays a user profile editing interface. It is divided into two main sections: 'Current Profile' and 'Contact Settings'. Under 'Current Profile', there are two fields: 'Email' with the value 'test@test.com' and 'Phone Number' with the value '1234567890'. The 'Contact Settings' section contains two toggleable options: 'Recieve Email Notifications' and 'Recieve Text Notifications'. Both options have radio buttons for 'Yes' and 'No', with 'Yes' being the selected option for both.

Current Profile

Email: test@test.com

Phone Number: 1234567890

Contact Settings

Recieve Email Notifications

Yes

No

Recieve Text Notifications

Yes

No

**Figure 11.3-2 Edit Profile Page**

As can be seen from the figure above, the edit profile tab displays your current email and registered phone number (if one is added else displays N/A) in the database. Below this there are two options which can be toggled. Receive email notifications enables our system to send notifications to the email listed in your current profile with alerts to status changes in your box. The next option receive text notifications is similar and sends notifications to the phone number displayed in your current profile. The texts are sent using our Trello service.

Below these toggleable options is several fields which can be updated. These fields are shown below in Figure 11.3-9. Among these fields there is the option to change email, change password, and edit your phone number. Upon editing any of these to apply changes click the confirm button below the fields.

Make Changes

Change Email:

Confirm Email:

Change Password:

Confirm Password:

Edit Phone Number:

**Figure 11.3-2 Edit Profile Page Cont.**

### Step 9.)

Your initial setup of your Smart Package Lockbox is now complete. You may continue to edit settings or select the logout option in the profile dropdown to keep your lockbox safe. Thank you once more from the Smart Package Lockbox team and we hope you enjoy our product.

For any issues or support feel free to contact the development team at: [smartPackageLockbox@gmail.com](mailto:smartPackageLockbox@gmail.com)

## 12.0 Project Summary and Conclusions

We as the design team set out to create an effective solution to alternative products already on the market as well as focus on security features that market products currently do not have. The lockboxes on the market fill their purpose and in some manner are an effective use of concealing or safeguarding packages, however, due to the costs and convenience of these products caused our primary purpose in creating a worthwhile alternative to their products.

Starting from the beginning of Senior Design I we weren't entirely sure on which design would be the most effective, but we knew we wanted as many security features as possible to be implemented. We focused mainly on trying to make a product that would counter other designs as well as maintaining the purpose we set out with cost and security in mind. In order to accomplish our tasks, set out for us, the group was split up into individual parts where our focus was shifted to what our primary skillsets were. Since we previously have never worked with one another in a group it was hard to know what to expect from the each other, but the overall process has been a smooth transition with each of us being able to go to one another for help regarding the project. By communication frequently between one



another we gathered a constant stream of information on different topics that pertained to our project such as the actual design and software. Through the process of analyzing and establishing what other team members need help with we were able to successfully contribute a sizeable portion each of us towards the outcome of the design.

The most influential part of the documentation in reference to knowledge gained was the research section of the project. In this section we analyzed numerous components and alternatives to our preconceived plans to see what other options we had to explore. Through this we found one of the best components in the inertial measurement unit (IMU) which helps utilize one of our more robust security features the movement of the box itself. This security feature was vital to our design due to people possibly moving the box itself, or at least trying to tamper with it in anyway. The IMU would detect these changes in movement and immediately recognize the box hasn't been unlocked and send out an alert to the user and alarm with a high decibel rating to dissuade the thief from stealing the box. By the time the overall research section was complete we felt comfortable enough with the information gathered and the parts looked at to establish which we wanted to use for our final design and the correct approach we decided to go with.

After the research section we took on the much more demanding section within the design construction itself. We analyzed every component, it's power draw, length and width, where it would go in the box, how it would be coded, and so on, for each individual part. The container itself housed a perfectly sizable area for all the components and boards by utilizing the bottom left corner for safety and to implement an outside power supply hole for use when connected to the houses outside outlet. The overall final design ended up being a huge success, the power draw wasn't too excessive, and the cost of the final product is immensely cheaper than alternative on the market with the bonus of security features. By making this final product cheap we hope that it is sought out after as a decent product someone would decide to have on their front porch due to the many user-friendly features it has as well as security.

# Appendices

## Appendix A - Copyright Permissions



Brad Ruffkess <bar@getboxlock.com>  
 Fri 4/27, 7:24 AM  
 Brandon Dziewior; support@getboxlock.com

📧 📧 📧 Reply all | ▾

Hi Brandon, thanks for reaching out and yes, you have permission to use our images in your senior design project as long as it continues to be for non-commercial use and as long as you share your work with us.

We're excited to see it and appreciate you thinking of us.

--

Brad A. Ruffkess  
 Founder

e: [bar@getboxlock.com](mailto:bar@getboxlock.com)  
 c: 404-918-9999



<http://www.getboxlock.com>

...

[https://commons.wikimedia.org/wiki/File:ATmega328P\\_28-PDIP.svg](https://commons.wikimedia.org/wiki/File:ATmega328P_28-PDIP.svg)  
[https://commons.wikimedia.org/wiki/File:ICIC-TQ32-X-K328-01\\_\(16421989932\).jpg](https://commons.wikimedia.org/wiki/File:ICIC-TQ32-X-K328-01_(16421989932).jpg)  
[https://commons.wikimedia.org/wiki/File:AI-thinker\\_ESP-012\\_module.jpg](https://commons.wikimedia.org/wiki/File:AI-thinker_ESP-012_module.jpg)  
[https://commons.wikimedia.org/wiki/File:SMT\\_sizes\\_based\\_on\\_original\\_by\\_Zureks.svg](https://commons.wikimedia.org/wiki/File:SMT_sizes_based_on_original_by_Zureks.svg)  
<https://commons.wikimedia.org/w/index.php?curid=65560520>

“Creative Commons License Deed.” *CC BY-SA 4.0*, [creativecommons.org/licenses/by-sa/4.0/](https://creativecommons.org/licenses/by-sa/4.0/).

## Appendix B - Datasheets

### ATmega328P:

[http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P\\_Datasheet.pdf](http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf)

### MPU-6050:

[https://store.invensense.com/datasheets/invensense/MPU-6050\\_DataSheet\\_V3%204.pdf](https://store.invensense.com/datasheets/invensense/MPU-6050_DataSheet_V3%204.pdf)

### ESP8266:

[https://cdn-shop.adafruit.com/product-files/2471/0A-ESP8266\\_Datasheet\\_EN\\_v4.3.pdf](https://cdn-shop.adafruit.com/product-files/2471/0A-ESP8266_Datasheet_EN_v4.3.pdf)

### MCP73213:

<http://ww1.microchip.com/downloads/en/DeviceDoc/20002190D.pdf>

### AT-1810\_T-LW50-R:

<http://www.puiaudio.com/pdf/AT-1810-T-LW50-R.pdf>

**LMV358:**

<http://www.ti.com/lit/ds/slos263w/slos263w.pdf>

**LM1117:**

[https://cdn-shop.adafruit.com/product-files/2471/0A-ESP8266\\_Datasheet\\_EN\\_v4.3.pdf](https://cdn-shop.adafruit.com/product-files/2471/0A-ESP8266_Datasheet_EN_v4.3.pdf)

## Appendix C – Requirements Specification Changelog

Will be updated as needed with requirement specification changes.

## Appendix D – References

*Arduino - Software*, [www.arduino.cc/en/Main/Software](http://www.arduino.cc/en/Main/Software).

Basrai, Murtaza. “Top 10 Editors For React Native Mobile App Development.” *Icicle*, 9 Sept. 2016, [www.icicletech.com/blog/top-10-editors-for-react-native](http://www.icicletech.com/blog/top-10-editors-for-react-native).

“Introducing Trinket.” *Programming with AVRdude*, [learn.adafruit.com/introducing-trinket/programming-with-avrdude](http://learn.adafruit.com/introducing-trinket/programming-with-avrdude).

“Mobile Operating System Market Share Worldwide.” *StatCounter Global Stats*, [gs.statcounter.com/os-market-share/mobile/worldwide](http://gs.statcounter.com/os-market-share/mobile/worldwide).

*Pocket AVR Programmer Hookup Guide*, [learn.sparkfun.com/tutorials/pocket-avr-programmer-hookup-guide/using-avrdude](http://learn.sparkfun.com/tutorials/pocket-avr-programmer-hookup-guide/using-avrdude).

“Writing Cross-Platform Apps with React Native.” *InfoQ*, [www.infoq.com/articles/react-native-introduction](http://www.infoq.com/articles/react-native-introduction).

“React (JavaScript Library).” *Wikipedia*, Wikimedia Foundation, 26 Apr. 2018, [en.wikipedia.org/wiki/React\\_\(JavaScript\\_library\)](http://en.wikipedia.org/wiki/React_(JavaScript_library)).

*Arduino Mega 2560 Rev3*, [store.arduino.cc/usa/arduino-mega-2560-rev3](http://store.arduino.cc/usa/arduino-mega-2560-rev3).

*Arduino Uno Rev3*, [store.arduino.cc/usa/arduino-uno-rev3/](http://store.arduino.cc/usa/arduino-uno-rev3/).

“ATmega328P Datasheet.” *Microchip*, [ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P\\_Datasheet.pdf](http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf).

“ESP8266 Datasheet.” *Adafruit*, [cdn-shop.adafruit.com/product-files/2471/0A-ESP8266\\_Datasheet\\_EN\\_v4.3.pdf](https://cdn-shop.adafruit.com/product-files/2471/0A-ESP8266_Datasheet_EN_v4.3.pdf).

“How to Read a 4x4 Keypad Using Just One Arduino Pin!” *YouTube*, YouTube, 5 Oct. 2016, [www.youtube.com/watch?v=G14tREsVqz0](http://www.youtube.com/watch?v=G14tREsVqz0).

Inc., Microchip Technology. “MCP73213 Battery Charge Management Controller.” *Microchip*, [ww1.microchip.com/downloads/en/DeviceDoc/20002190D.pdf](http://ww1.microchip.com/downloads/en/DeviceDoc/20002190D.pdf).

“MPU 6050 Datasheet.” *Invensense*, [store.invensense.com/datasheets/invensense/MPU-6050\\_DataSheet\\_V3%204.pdf](http://store.invensense.com/datasheets/invensense/MPU-6050_DataSheet_V3%204.pdf).

“MSP430G2553 (ACTIVE).” *Mixed Signal Microcontroller | TI.com*, [www.ti.com/product/MSP430G2553](http://www.ti.com/product/MSP430G2553).

Tawil, Yahya. “Understanding Arduino UNO Hardware Design.” *All About Circuits*, 1 July 2016, [www.allaboutcircuits.com/technical-articles/understanding-arduino-uno-hardware-design/](http://www.allaboutcircuits.com/technical-articles/understanding-arduino-uno-hardware-design/).

“3x4 Phone-Style Matrix Keypad Datasheet.” *DigiKey*, [media.digikey.com/pdf/Data%20Sheets/Adafruit%20PDFs/1824\\_Web.pdf](http://media.digikey.com/pdf/Data%20Sheets/Adafruit%20PDFs/1824_Web.pdf).

“A1171EEWLT-P.” *Allegro MicroSystems, LLC | Sensors, Transducers | DigiKey*, [www.digikey.com/products/en?keywords=620-1232-1-ND](http://www.digikey.com/products/en?keywords=620-1232-1-ND).

“AI-3035-TWT-3V-R Datasheet.” *Puiaudio*, [www.puiaudio.com/pdf/AI-3035-TWT-3V-R.pdf](http://www.puiaudio.com/pdf/AI-3035-TWT-3V-R.pdf).

“Amazon Key.” *Amazon.com*, [www.amazon.com/b?ie=UTF8&node=17285120011](http://www.amazon.com/b?ie=UTF8&node=17285120011).

“AT-1810-T-LW50-R Datasheet.” *Puiaudio*, [www.puiaudio.com/pdf/AT-1810-T-LW50-R.pdf](http://www.puiaudio.com/pdf/AT-1810-T-LW50-R.pdf).

“Dual Element Detectors for Motion Sensing.” *Excellitas*, [www.excelitas.com/downloads/dts\\_lhi778\\_lhi878\\_pyd1388.pdf](http://www.excelitas.com/downloads/dts_lhi778_lhi878_pyd1388.pdf).

“Force Sensitive Resistor (FSR).” *Adafruit*, [cdn-learn.adafruit.com/downloads/pdf/force-sensitive-resistor-fsr.pdf](http://cdn-learn.adafruit.com/downloads/pdf/force-sensitive-resistor-fsr.pdf).

“Force Sensitive Resistor (FSR).” *Force Sensitive Resistor (FSR) | Adafruit Learning System*, [learn.adafruit.com/force-sensitive-resistor-fsr/overview](http://learn.adafruit.com/force-sensitive-resistor-fsr/overview).

“FSS Low Profile Sensor.” *Honeywell*, [sensing.honeywell.com/index.php?ci\\_id=50137+](http://sensing.honeywell.com/index.php?ci_id=50137+).

“Get BoxLock.” *Protect Your Deliveries - Get BoxLock*, [www.getboxlock.com/](http://www.getboxlock.com/).

Inc., Parallax. “FlexiForce Sensor Documentation.” *Parallax*, [www.parallax.com/sites/default/files/downloads/30056-FlexiForce-Sensor-Documentation-v2.0.pdf](http://www.parallax.com/sites/default/files/downloads/30056-FlexiForce-Sensor-Documentation-v2.0.pdf).

“Lithium Ion vs. Lithium Polymer - What's the Difference?” *Android Authority*, 27 Oct. 2013, [www.androidauthority.com/lithium-ion-vs-lithium-polymer-whats-the-difference-27608/](http://www.androidauthority.com/lithium-ion-vs-lithium-polymer-whats-the-difference-27608/).

“Micropower Omnipolar Digital Hall-Effect Sensor ICs.” *Honeywell*, [sensing.honeywell.com/index.php?ci\\_id=142130](http://sensing.honeywell.com/index.php?ci_id=142130).

“Motion Sensor Documentation.” *Panasonic*, [media.digikey.com/PDF/Data%20Sheets/Panasonic%20Electric%20Works%20PDFs/AMN%20Design%20Manual.pdf](http://media.digikey.com/PDF/Data%20Sheets/Panasonic%20Electric%20Works%20PDFs/AMN%20Design%20Manual.pdf).

“Parallax Standard Servo.” *Parallax.com*, [www.parallax.com/sites/default/files/downloads/900-00005-Standard-Servo-Product-Documentation-v2.2.pdf](http://www.parallax.com/sites/default/files/downloads/900-00005-Standard-Servo-Product-Documentation-v2.2.pdf).

*Piezo Buzzer Datasheet*. Soberton Inc., [cirlw3yrzg4afc1x47lv2k18.wpengine.netdna-cdn.com/wp-content/uploads/2016/11/PT-1307-data-sheet-2.pdf](http://cirlw3yrzg4afc1x47lv2k18.wpengine.netdna-cdn.com/wp-content/uploads/2016/11/PT-1307-data-sheet-2.pdf).

“ROB-09065 Datasheet.” *Sparkfun.com*, [cdn.sparkfun.com/datasheets/Robotics/Small%20Servo%20-%20ROB-09065.pdf](http://cdn.sparkfun.com/datasheets/Robotics/Small%20Servo%20-%20ROB-09065.pdf).

Rogers, R. Michael. “WiFi Module - ESP8266.” *WRL-13678 - SparkFun Electronics*, [www.sparkfun.com/products/13678](http://www.sparkfun.com/products/13678).

“SainSmart 1A Power Supply Adapter.” *Amazon.com*, [www.amazon.com/SainSmart-Adapter-5-5x2-1mm-100V-240V-Arduino/dp/B00WW26JWC/ref=sr\\_1\\_2?s=electronics&ie=UTF8&qid=1524796118&sr=1-2&keywords=arduino+12v+adapter](http://www.amazon.com/SainSmart-Adapter-5-5x2-1mm-100V-240V-Arduino/dp/B00WW26JWC/ref=sr_1_2?s=electronics&ie=UTF8&qid=1524796118&sr=1-2&keywords=arduino+12v+adapter).

“Sensors and Emitters.” *Excellitas*, [www.excelitas.com/Downloads/CAT\\_SensorsAndEmittersInfraredSensing.pdf](http://www.excelitas.com/Downloads/CAT_SensorsAndEmittersInfraredSensing.pdf).

“Series 96 Datasheet.” *Grayhill*, [www.grayhill.com/assets/1/7/Keypads\\_96.pdf](http://www.grayhill.com/assets/1/7/Keypads_96.pdf).

“Suncast SSW1200 Mocha Resin Wicker.” *Amazon.com*, [www.amazon.com/Suncast-SSW1200-Wicker-Gallon-Storage/dp/B0044V3UFS/ref=sr\\_1\\_4?s=lawn-garden&ie=UTF8&qid=1523315088&sr=1-4&keywords=outdoor+container](http://www.amazon.com/Suncast-SSW1200-Wicker-Gallon-Storage/dp/B0044V3UFS/ref=sr_1_4?s=lawn-garden&ie=UTF8&qid=1523315088&sr=1-4&keywords=outdoor+container).

“TCS40DPR,LF.” *Toshiba Semiconductor and Storage | Sensors, Transducers | DigiKey*, [www.digikey.com/products/en?keywords=TCS40DPRLFCT-ND](http://www.digikey.com/products/en?keywords=TCS40DPRLFCT-ND).

“TowerPro SG90C 360 Degree Micro Servo.” *DigiKey*, [media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/SER0043\\_Web.pdf](http://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/SER0043_Web.pdf).

“What Is Keypad? - Definition from Techopedia.” *Techopedia.com*, [www.techopedia.com/definition/7940/keypad](http://www.techopedia.com/definition/7940/keypad).

“ZSBG446671.” *Zilog | Sensors, Transducers | DigiKey*, [www.digikey.com/products/en?keywords=269-4928-ND](http://www.digikey.com/products/en?keywords=269-4928-ND).

*AJAX Introduction*, [www.w3schools.com/xml/ajax\\_intro.asp](http://www.w3schools.com/xml/ajax_intro.asp).

“Ajax (Programming).” *Wikipedia*, Wikimedia Foundation, 26 Apr. 2018, [en.wikipedia.org/wiki/Ajax\\_\(programming\)](http://en.wikipedia.org/wiki/Ajax_(programming)).

Ali, Mostafa. “The Differences Between Bluetooth, ZigBee and WiFi.” *LinkedIn SlideShare*, 19 Sept. 2015, [www.slideshare.net/MostafaAli39/understanding-differences-between-bluetooth-zigbee-and-wifi](http://www.slideshare.net/MostafaAli39/understanding-differences-between-bluetooth-zigbee-and-wifi).

“Basics of UART Communication.” *Circuit Basics*, 11 Apr. 2017, [www.circuitbasics.com/basics-uart-communication/](http://www.circuitbasics.com/basics-uart-communication/).

“Bluetooth vs Wi-Fi.” *Difference and Comparison* | Diffen, [www.diffen.com/difference/Bluetooth\\_vs\\_Wifi](http://www.diffen.com/difference/Bluetooth_vs_Wifi).

Gupta, Kitty. “LAMP Stack vs. WISA Stack Which Is Best for Your Startup.” *Web Dev, Design and Tips*, Web Dev, Design and Tips, 4 Aug. 2017, [www.freelancinggig.com/blog/2017/08/04/lamp-stack-vs-wisa-stack-best-startup/](http://www.freelancinggig.com/blog/2017/08/04/lamp-stack-vs-wisa-stack-best-startup/).

*JSON vs XML*, [www.w3schools.com/js/js\\_json\\_xml.asp](http://www.w3schools.com/js/js_json_xml.asp).

Raj, Jay. “An Introduction to the MEAN Stack.” *SitePoint*, SitePoint, 11 Apr. 2018, [www.sitepoint.com/introduction-mean-stack/](http://www.sitepoint.com/introduction-mean-stack/).

*Reducing Arduino Power Consumption*, [learn.sparkfun.com/tutorials/reducing-arduino-power-consumption](http://learn.sparkfun.com/tutorials/reducing-arduino-power-consumption).

“What Is a Web Server?” *MDN Web Docs*, [developer.mozilla.org/en-US/docs/Learn/Common\\_questions/What\\_is\\_a\\_web\\_server](http://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server).

Wyse, Josh. “Why JSON Is Better Than XML.” *Blog*, [blog.cloud-elements.com/json-better-xml](http://blog.cloud-elements.com/json-better-xml).

“Zigbee.” *Wikipedia*, Wikimedia Foundation, 23 Apr. 2018, [en.wikipedia.org/wiki/Zigbee](http://en.wikipedia.org/wiki/Zigbee).