Spring 2022 Senior Design 1
Live Bolt Smart Lock

Department of Electrical and Computer Engineering
University of Central Florida
100 Page Draft
Group 22
March 25, 2022

Linda Anyosa - Electrical Engineering
Mohamed Faizel - Computer Engineering
Andrew O'Reilly - Electrical Engineering
Joel Shaw - Computer Engineering

100 Page Draft
Group 22

100 Page Draft
Group 22

1. Executive Summary

It’s every homeowner’s dream to discover an all-in-one security system with affordability to the consumer’s access. With a growing and competitive market of security systems, there is a lack of cost flexibility of how much security would be provided to the consumer. What makes this project different than other security systems is it incorporates home monitoring and entry available through a phone application or at door entrance. This will be accomplished by designing an Android/IOS application to pair with the module and grant access to home monitoring and real-time automated lock to the homeowner deadbolt. Hence the introduction of the project, Live Bolt.

One of our main objectives we’d like to accomplish in creating Live Bolt would be to ideally have a functioning system to present in our senior design showcase. Ensure that there’s no bugs and it able to operate and provide home monitoring to future users. To create interest in our project, we want to create an application aesthetically presentable with user friendly compatibility. As a team, we know that installation of any home system requires handy tools, so we are designing both the security and home monitoring box to have installation ease with an enclosed placement to harness the electronics. Further objectives to building Live Bolt will have its own section and be explained into more details.

Building the mechanism and harnessing the compatibility to a phone application would require having components in mind. To share the data between the hardware and software of the system, a Raspberry pi and an MCU will satisfy these requires. Although these components don’t consist of the whole system, it would bring together the data exchanged to control the open/close lock and push notifications for home monitoring. Other hardware and software design choice will be elaborated further given the research done to execute our goal.

After developing the ideal design, we’ll overcome realistic design constraints which are factors that’ll alter the systems performance. This is including wireless range access for entrance, weather suitable, aesthetic presentation of the system, and battery life implementation. Ethical challenges from the construction of the system would also have to align with the path of preventing security breach. External data will be for citational and referenced purposes only to keep the consumers privacy secured.

As a group we provide variety of experience ranging of two computer and two electrical engineers. We want to challenge ourselves in different ranges of developing an application, creating a PCB, learning about CAD, and implementing IoT to our understanding. Developing Live Bolt will give us that gateway to design and explore engineering concepts we want to learn.

2. Project Description

2.1 Project Motivation and Goals

The main motivation for creating this project was from our individual experience in package theft. In an instance that a package has been labeled delivered but, not within the front door or close radius of a home can be bothersome to a customer. A package is ideally something of a need or want from a consumer. Why not monitor/ report any front door activity through a security system? In this question, we came with the idea of building a security system to monitor and allow front door access to delivery workers with a one-pin code to reliably deliver the package within a home. Although this idea grants entrance, there must be an automated lock to open and close the door for a home. Therefore, we decided to expand upon this project to include variety of access control for the consumer. It'll logically make sense to not only have a locking mechanism for delivery worker but to include forms of entrances for the consumers own usage. We want to introduce a software application to harness the numeral access control to a consumer at their own usage of a cell phone. Additionally, this motivation of creating an application would challenge and utilize the team's software skills in building a reliable security system that could potentially be used in industry.

Our goal is to create a reliable low-cost security system with multiple methods of home/ package safety for the consumers of tomorrow. We want to challenge the current security market by creating a new set up with an app and hardware systems interface. In doing so, we would keep the system as user-friendly as possible while keeping it as secure as possible. We will achieve these goals by creating a lightweight security system that can be placed in desired monitoring areas (e.g., home, shed, garage), distributing power to electronic hardware, and establishing a connection between the lock and phone application. It is also a goal for us to keep costs of hardware low while maintaining quality to make this product accessible to low-income families.

2.2 Objectives

Our main objective for the project is to design an innovative security and integrated lock system that combines the features of available systems in the market into one unified package. To have our project stand out, we plan to offer the most features in access control and home monitoring for the best price while ensuring optimal quality, functionality, and aesthetics to our customers home and style. How we plan to target these objectives is by researching existing home security and locking systems in the market. In access control we will also scope upon versatility about how a homeowner would like to open their front door. In security measures, we'd like to research forms of home monitoring and reaction to keeping a property secured. While these two would create most of the

hardware for the project, we need to make it available through a mobile application for the customers convince. Objectives to execute that action would be to adhere to commonly held design standards to usability, have an always-online internet connection, and include tools like VoiceOver or TalkBack to incorporate accessibility. Additionally, we'd like to make this system small and applicable to every home door. Creating a compact/portable system will be our priority when engineering the smart lock design through choice of buying electrical/mechanical parts. From the team's knowledge and capabilities, we will ensure thorough testing that we will deliver a fully functioning product that our customers will love.

2.3 Requirements Specifications

For the design of our Live Bolt product, we'd like to incorporate a mixture of all the following research we've conducted on these home security devices. The list below in Table 1**,** Table 2**,** Table 3**,** and Table 4 will include requirement specifications on hardware and software execution of our senior design project.

Table 1: Requirement specifications for Security System Features

SECTION	REQUIREMENT	SOLUTION	VERIFICATION
Security System (SS)			
SS.1	Notifying motion sense to consume	Controlled through a MCU (IR motion detector)	Integrate sensors to application
SS.2	Home monitoring	5 Megapixel, 2592 x 1944 image resolution, 1080p video resolution	Viewable video stream sent from camera
SS.3	Time stamps unlock attempts	Sensing	Send message to user when attempts are made
SS.4	Single use PIN numbers for guests/ delivery workers	Application interface for one-use code	Send a text message to guest with one-use code
SS.5	Alarm system	Integrate LEDs and buzzers trigger when multiple attempts are failed in succession	Send message to app when alarm system is triggered

Table 2: Requirement specifications for Access Control Features

SECTION	REQUIREMENT	SOLUTION	VERIFICATION
Access Control (AC)			
AC.1	Accessibility to consumer with ease	Android/iOS application	Publish to App Store/Google Play Store
AC.2	Application unlocking system	4-to-6-digit PIN number, unlock remotely via mobile application, face detection through application, NFC	Notification sent when door is locked/unlocked
AC.3	Hardware unlocking system	Voice recognition, PIN code access	Door can be opened without interference

Table 3: Requirement specifications for Application Features

SECTION	REQUIREMENT	SOLUTION	VERIFICATION
Application (A)			
A.1	Simple and user-friendly UI/design	Adhere to commonly held design standards for usability	Use a variety of methods to test code, including end-to-end tests
A.2	Live Bolt must be always-online to mobile application	Use a React API to monitor internet connection	Display message to user if connection is lost
A.3	Accessibility	Tools like VoiceOver or TalkBack	Integrate TTS in user interface

Table 4: Requirement specifications for Design Features

SECTION	REQUIREMENT	SOLUTION	VERIFICATION
Design (D)			
D.1	Locking door system	High torque servo to turn lock, 10 kg*cm with about 170 degrees range of motion. 3D Printed cover for deadbolt turn knob	Servo is successfully able to turn the knob cover enough to lock/unlock door every time
D.2	Harness electronic parts in casing	Select a durable printing filament	Research different filaments PLA, ABS, PETG
D.3	Rechargeable Batteries	Design a reliable battery system to change AA batteries and/or recharge a battery pack	Understand power requirements for electronic parts

2.4 Quality of House Analysis

Before constructing Live Bolt, we created a house of quality for our system to ensure the highest quality of our product between engineering requirements and marketing requirements. The House of quality matrix, shown in **Table 5**, allows us to identify the level of importance and devise a strategy on how to achieve our requirements. Before beginning to identify the importance for each requirement, creating a legend for the correlation, relationship, and improvement of direction for our system is indicated below the table. In our table we highlighted what wad stated in our goals such as portability, security protection, sensors, battery life, etc., in engineering requirements. Now this must align with customer requirements within the quality or product and user preferences. This would market to customers with the understanding that our team puts their usability/ flexibility first. With the weight column ranging from 1-10, we identify cost and user interface as top priority to make it accessible and user-friendly as possible to be purchased by a larger audience. Engineering and customer requirements go together when finding the direction of improvement needed to be done in our system. As engineers, the house of quality outlines ways we can improve our design why considering a consumer desire. With Live Bolt, we intent to maximize improvement although intuitively we could overcome engineering constraints.

Table 5: House of Quality for Live Bolt

	Column #		1	2	3	4	5	6	7
	Direction of Improvement		▼	▲	▲	▼	◇	◇	▲
Category	Weight	Customer Requirements (Explicit and Implicit) / Engineering Requirements		Size	Power Usage	Security Protection	Cost	Sensors	Microcontroller	Battery Life
				-	-	+	-	+	+	+
Quality of Product	6	Good User Experience	+	○	▽	●	○	●	▽	○
	5	Robustness	+	○	○	●	○	○	▽	●
	5	Multiple Options	+	●	●	●	●	●	▽	○
User Preferences	9	Cost	-	○	●	○	●	●	●	○
	6	Smart Phone Application	+	▽	▽	▽	▽	○	●	○
	8	User Interface	-	▽	▽	▽	▽	▽	▽	○
	7	Intallation Ease	+	●	▽	▽	▽	○	○	●

Correlations	
Positive	+
Negative	−
No Correlation	

Relationships	
Strong	●
Moderate	○
Weak	▽

Direction of Improvement	
Maximize	▲
Target	◇
Minimize	▼

3. Research Related to Project Definition

3.1 Existing Similar Projects and Products

Home security and access control integrated in homes today are at an abundant. With this growing market, we would like to formally research about existing products and seek improvement in their systems. This research will inspire the design for Live Bolt.

3.1.1 Ring Doorbell

Ring video doorbell provides variety in the sense that there are different features of home monitoring installation: wired, battery powered, and bundles leading to solar power or security included. Ring doorbell is customized to reach as many consumers as possible no matter the circumstance of the front door, hence the

variety of options. Additionally, with Ring's growing reputation, they provide flexibility to consumers with different option in monitoring their home. From buying one of their products, it includes motion detection; application integration of notifying the consumer of front door activity; and see, hear, speaking que to visitors in real time.

There are some downsides to including one of the Ring video doorbell to one's home. One main reason being that there are so many options to choosing a reliable doorbell into one's home. With that comes variety of options for how to determine one's monitoring. Many options include variety but could be overwhelming to a customer. Additionally, to use one of their products it includes a monthly subscription plan from additionally purchasing the device. Another problem with ring doorbell is that it requires a strong Wi-Fi connection that essentially must be provided by the consumer. Information exchange between the device and application will be hindered if not provided with a strong network. This research is great notes on how to make our Live Bolt system enhanced and more available to the public. We would like to obtain features that the Ring doorbell incorporates like motion sensing and application availability while subtraction the subscription plan. Live Bolt differs since there'll also be access control of entering a home through the application. We also take reference of the Ring application to apply towards our application user interface.

3.1.2 Google Nest Doorbell

Another product similar to Live Bolt is the Google Nest Doorbell, Google's attempt at a home security monitoring system. It has many features such as a camera, doorbell, sound detection and familiar face detection. Moreover, the setup is wired to configure with an existing doorbell system or can run wirelessly. Google has made it possible to integrate the doorbell with the rest of the Google Nest environment. Features the Nest Doorbell that Live Bolt will include are the camera monitoring system, face detection, and app integration. However, unlike the Nest Doorbell, Live Bolt allows for multiple unlock methods and gives the user an unlock log report through the application interface. It additionally differs in that it is a door locking security system compared to Nest Doorbell which only offers video and audio monitoring. Through this comparison between the two systems, we can see clearly that Live Bolt is the more cost-effective option than the Nest Doorbell. The Nest Doorbell only offers Doorbell monitoring of your existing system while Live Bolt steps up the security of your system.

3.1.3 Yale Lock Integration

Google Nest Doorbell has the capability to interface with the Yale electronic door lock. The Yale lock is a keyless smart lock, similar to Live Bolt, that incorporates a digital keypad. Yale allows for a passcode you can give to people you trust that you can disable at any time. It also offers self-locking features, and the ability to

lock/unlock through a phone application. With Live Bolt, we will combine features from both the Nest Doorbell and Yale Lock to enhance safety and security in a single product.

3.1.4 August Smart Lock

The August Lock is a Smart Lock system like many of the others we researched which integrates directly onto an existing deadbolt system. It does require some unscrewing to integrate however August does include videos to guide users through the process. One benefit is that the August lock doesn't prevent you from using your traditional key so that option is still available to use. It includes an Auto Lock and Auto Unlock feature which allows for geofencing to be used to determine if the user is leaving or heading toward the home. It also allows for the ability to use a smartwatch to lock and unlock the door without having to bring out your phone. The lock has a mobile app which allows many features including controlling multiple locks, inviting other people to manage the locks, temporary guest keys, an activity timeline, and additional security with the option of mobile bio authentication through the smartphone. These are useful features and many of which we are including in our Live Bolt system to strengthen its security and usability. Additionally, the August Lock does have multiple models to choose allowing customers to choose what locking system works best for them. The pricing of their most costly locking system, the Wi-Fi Smart Lock, comes to 229 USD which is comparable to the wired version of the Google Nest Doorbell. Additionally, if customers end up getting a cheaper model and want to add the August Wi-Fi Bridge to get all the features the total cost is the same as or less than the cost of the Wi-Fi Smart Lock.

There are downsides to August's Smart Locks. For one the newest generation released by August, although smaller and less bulky than the previous generation, has a smaller battery life. The 3rd generation had a battery life ranging from 6 to 12 months where the current 4th generation has a battery life from 3 to 6 months. Additionally, the batteries for the 4th generation are not the standard AAA or AA batteries and are instead CR123A batteries. These batteries are both more difficult to find and expensive to purchase than normal replaceable batteries. A drawback from August lock comes to auto unlocking connectivity. Auto lock works well for when the lock is able to connect to Bluetooth easily. However, it doesn't work as well when it fails to connect before the User reaches the door making them wait. Another issue can happen when the door unlocks but the user takes too long to open it. If the user lives in an area where they must climb stairs or use an elevator the door can unlock and then relock before the user reaches. This also becomes a convenience issue for the user as it could force them to use their key or have to unlock the lock again from their phone. Another issue with the August lock and most other smart locks is the lack of RFID and Near-Field Communication (NFC). The August Lock already supports smartwatch integration and allowing for NFC unlocking would be a useful feature. It would eliminate the need to take out your phone or open the app allowing for the user's device to be held close to the lock to open. With Live Bolt we plan on

incorporating both RFID and NFC to give users more options to unlock the system and ease of access.

3.1.5 Nuki Smart Lock

The Nuki Smart Lock is another product that caught our attention as we were researching for our project. It is a smart lock system that directly integrates onto an existing locking system. There is no need for screwdrivers or other tools to install the system as it uses a physical key to lock and unlock the door as if a person was turning the lock themselves. This way of integrating the system allows for it to be compatible with many door locks. Nuki has recently revamped their product line with the 3.0 series offering a normal and pro model. Both share many of the same features aside from the pro having a Wi-Fi module in addition to its Bluetooth module, and a rechargeable battery where the normal uses 4 AA batteries. In our Live Bolt system, we will be providing only one product so users will not have to worry about product tradeoffs like with Nuki's products. Nuki also provides a smartwatch integration in addition to their mobile app allowing for you to unlock your door from your smartwatch. They also allow the ability to create and assign digital keys to family and friends which is similar to a feature we will include in the Live Bolt system.

We did see downsides to the Nuki Smart Lock 3.0 series which we have taken note of in designing the Live Bolt system. The first is that not all locks work with Nuki's system so customers will need to check their compatibility guide before purchasing a lock. Another is that for the normal lock model any Wi-Fi actions the user wants to do will need the Nuki Bridge, one of Nuki's many accessories. The Nuki Bridge has a Wi-Fi module allowing for the base model lock to connect to Wi-Fi. The Pro model comes with this feature in the box. Without the Wi-Fi module in the Pro or Nuki Bridge with the base model users lose out on features like Smart assistants, and Remote management. This significantly limits the users experience and these are features we plan to implement into Live Bolt without the need for accessories. Another downside of the Nuki is cost. The base model is 149 € which is 169 USD, and the Pro is 249 € which is 305 USD. This is already a high cost for the system, and it doesn't take into account the price of the many accessories Nuki offers to enhance the Smart Lock experience. From a Keypad to open your door with a code to the Door Sensor which enhances the sensing of the system, Nuki offers many accessories which can feel vital to a better experience. Lastly, the Nuki system does not include a camera for both models. Lacking this feature makes seeing who is at your door impossible however Nuki does include an activity report in the app for monitoring. This is limited however to Nuki products that can connect to Wi-Fi. The Live Bolt system will have a camera for viewing activity outside the door allowing for the user to know what is always happening.

3.2 Computation and Control Device

To execute our objective for Live Bolt, there'll need to be control and wireless communication between the servo and/or sensors to the application. Choosing the right control device will play a large role to our design; therefore, we will dive into the various options to what control devices are on the market and their characteristics for delivering performance.

3.2.1 Arduino Line

The Arduino line of microcontrollers operate from an ATmega chip. The specific model of Arduino varies depending on the application. The two that we have narrowed down to are the Arduino Uno and Arduino Mega. Both operate at 5V, are programmed via USB A/B; however, they have a lot more differences than similarities that will be outlines below. **Figure 1** shows the Uno and Mega boards side by side while Table 6 outlines the main differences between each board.

3.2.1.1 Arduino Uno

The Arduino Uno is a smaller microcontroller, with a size of 68.58 millimeters by 53.34 millimeters. Since it is physically smaller, it has fewer pins, with 14 digital pins and 6 analog pins. The Arduino Uno also has a 16MHz Atmega328p processor and 32kB of flash memory. These microcontrollers can be purchased in the range of $20.

3.2.1.2 Arduino Mega

The Arduino Mega is physically much larger than the Arduino Uno, at 101.6 millimeters by 53.34 millimeters. Since it is longer it has more pins, 54 digital pins and 16 analog pins, 50 more pins than the Uno. It has a 16MHz ATmega2560 processor and 256kB of flash memory. These models are usually around $42.

Table 6: Arduino Uno vs Arduino Mega Comparisons

Characteristic	Uno	Mega
Cost	$20	$42
Size	68.58 mm x 53.34 mm	101.6 mm x 53.34 mm
Processor	Atmega328p	ATmega2560
Clock Speed	16MHz	16MHz
Flash Memory	32kB	256kB
Pin Count	20	70

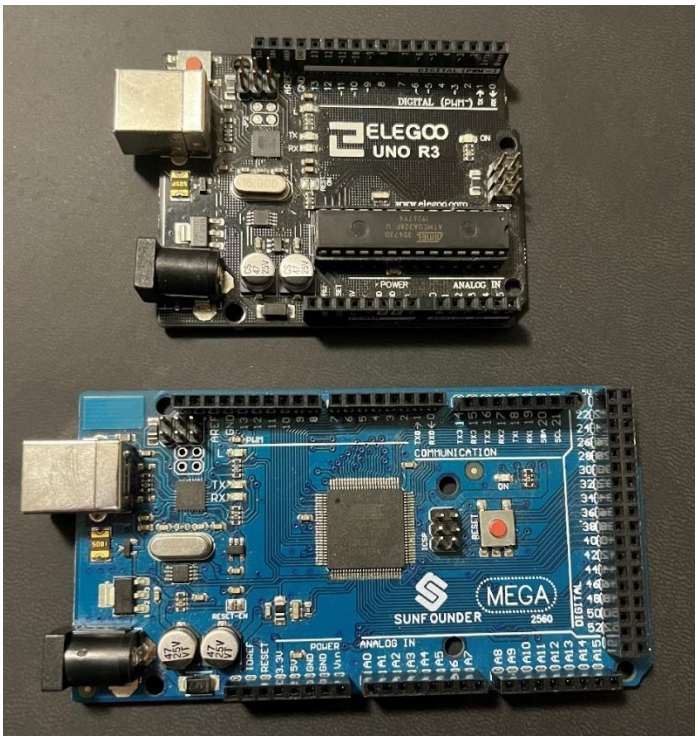

Figure 1: Reproduction Arduino Uno (Top) and Arduino Mega (Bottom) running ATmega architecture

3.2.2 Raspberry Pi Line

The Live Bolt System will need a more powerful processor to host the application and process voice recognition. The Raspberry Line of computation devices would easily serve this need. A Raspberry Pi is essentially a small computer and can run an operating system and programs. The two Raspberry Pi boards we will look at are the Model 3B+ and the newer Model 4. Both are 85.09 millimeters by 55.88 millimeters, house 40 GPIO pins, and operate off 5V. They are also capable of Wi-Fi natively. **Figure 2** shows each board side by side while Table 7 outlines the main differences between each board.

3.2.2.1 Raspberry Pi Model 3B+

The Model 3B+ is an older model, but not obsolete by any means. With a 1.5GHz Quad-Core Broadcom BCM2837B0 processor and 1GB DDR2 RAM, the Model 3B+ can handle multiple processes at once. It has one HDMI port, four USB 2.0 ports, and ethernet. It is also powered through either a micro-USB or a GPIO pin header to supply power. These models usually cost around $35.

3.2.2.2 Raspberry Pi Model 4

The Model 4 is the successor to the Model 3B+. It has improvements to the processor, now a 1.5 GHz Broadcom BCM2711. The RAM has also been updated to DDR4, with options for 1GB, 2GB, and 4GB. The power is now supplied through a GPIO pin or USB-C and there are two micro-HDMI ports rather than one full HDMI port. Two of the four USB 2.0 ports were also replaced with two USB 3.0 ports. The 2GB model costs about $45.

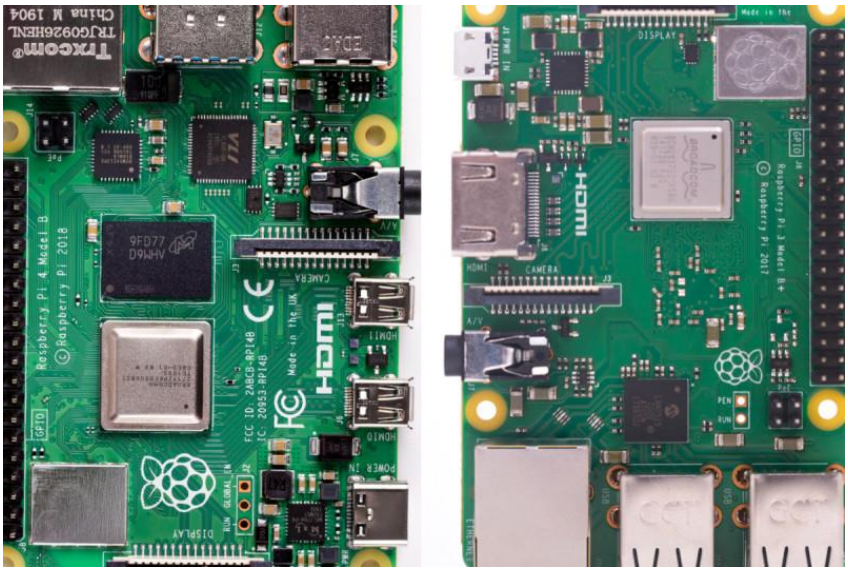

Figure 2: Raspberry Pi 4 (Left) and Raspberry Pi 3B+ (Right) [1]

Table 7: Raspberry Pi 3B+ vs Raspberry Pi 4 comparisons

Characteristic	Model 3B+	Model 4
Cost	$35	$45
Size	85.09 mm x 55.88 mm	85.09 mm x 55.88 mm
Processor	Broadcom BCM2837B0	Broadcom BCM2711
Clock Speed	1.5GHz	1.5GHz
Flash Memory	1GB DDR2 RAM	1GB, 2GB, 4GB DDR4
Pin Count	40	40
Input/Output	1 HDMI, 4 USB 2.0, Ethernet	USB-C,2 micro-HDMI, 1 HDMI, 2 USB 2.0, 2 USB 3.0

3.2.3 Texas Instruments MSP430 Line

Texas Instrument microcontrollers offers one of the best analog in the industry with affordability as its best option. The MSP430 line is 16-bit MCU including flexible, integrated analog components, such as ADCs, DACs, and Op-amps;

Integrated LCD controllers for displaying information; and devices with USB2.0 for communication with PC or application updates. As undergraduate engineering students at UCF, our classes have introduced two known microcontrollers from TI: MSP430FR6989 and MSP430G2553. Figure 3 shows each board side by side while Table 8 outlines the main differences between each board.

3.2.3.1 MSP430FR6989 Launchpad

The MSP430FR6989 provides ultra-low-power FRAM platform uniquely embedded and a holistic ultra-low-power system architecture to increase performance at lowered energy budgets. Additionally, the MSP430FR6989 includes features of a 16-Bit RISC architecture up to 16-MHz Clock, integrated LCD driver with 320 segments, and multifunction input/output ports. The board is 76.2 millimeters by 50.8 millimeters. A few parameters from this microcontroller (MC) include a 128 kB non-volatile memory, 2 KB of RAM, 12-bit SAR ADC, No USB port, 2 I2Cs and 4 SPI. Due to high demand, these are currently out of stock in the TI website, but alternative websites offer this microcontroller from $24 - $50.

3.2.3.2 MSP430G2553 Launchpad

The MSP430G2553 is like the FR6989 MC from the fact that it is also an ultra-low-power missed signal MC with a built in 16-bit timer. It has low supply-voltage range of 1.8 V to 3.6 V. The G2553 is a 16-bit RISC architecture and a basic clock module configuration with a 24 capacitive touch enabled I/O pins. This board is 69 millimeters by 51 millimeters. There is no LCD driver or USB port in this MC. Note that an online purchase will equate to the price of just a chip but, the whole controller would summarize to $40.

Table 8: MSP430FR6989 Launchpad vs MSP430G2553 Launchpad comparisons

Characteristic	MSP430FR6989	MSP430G2553
Cost	$24-$60	$40
Size	76.2 mm x 50.8 mm	69 mm x 51 mm
Processor	MSP430FR6989	MSP430G2553
Clock Speed	16 MHz	16MHz
Flash Memory	16KB	128KB
Pin Count	83	24
Input/Output	LCD, I2C, SPI, Micro USB	Micro USB

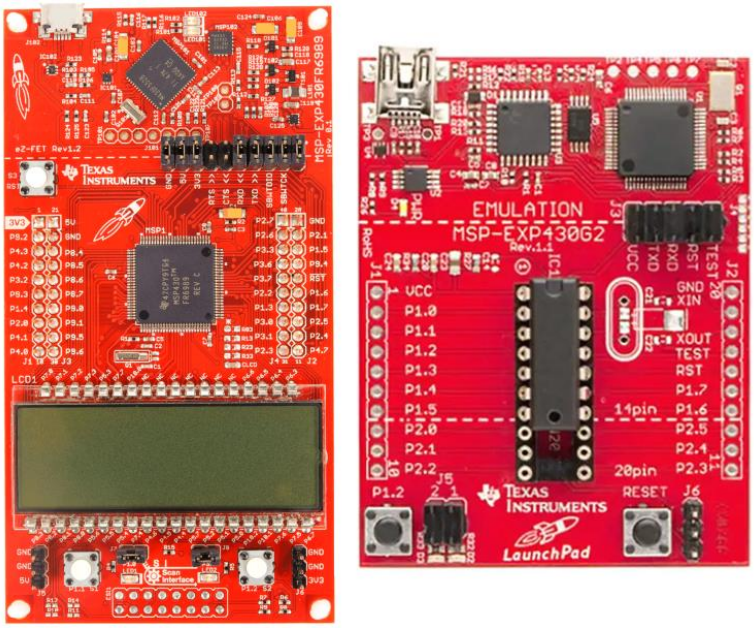

Figure 3: MSP430FR6989 Launchpad (Left) vs MSP430G2553 Launchpad (Right)

3.2.4 ESP Wi-Fi Module Line

The ESP line of microcontrollers are low-cost controllers with wireless capabilities that are great for internet-of-things and home automation projects, like the Live Bolt. If we choose to include Wi-Fi into the microcontrollers of the Live Bolt system, it might be better to opt for an ESP device over another microcontroller since Arduino and MSP430 do not have Wi-Fi natively. Both modules come with Wi-Fi and are programmed via the Arduino integrated development environment (IDE). Figure 4Figure 3 shows each board side by side while Table 9 outlines the main differences between each board.

3.2.4.1 ESP32

The ESP32 is a very powerful development board with a 160MHz clock frequency, 34 GPIO pins, 16 PWM channels, SRAM, and Flash RAM. These models also include a touch sensor, temperature sensor, and hall effect sensor. It operates on the HT40 40Mhz spectrum of a 2.4GHz Wi-Fi band. One major difference between the ESP32 and ESP8266 is that the ESP32 includes Bluetooth 4.2 and Bluetooth 5.0 with Bluetooth Low Energy (BLE). It is 54.6 millimeters by 27.94 millimeters by 13 millimeters and costs anywhere from $6 to $12.

3.2.4.2 ESP8266

The ESP8266 is less powerful than the ESP32 but is still very useful for the application of this project. There is a breakout board for development that is called the ESP8266 NodeMCU, that we will be considering. It has an 80MHz clock frequency, 17 GPIO pins, and 8 PWM channels. It lacks any Bluetooth capabilities and operates on the HL20 20MHz spectrum of the 2.4GHz Wi-Fi band. It also lacks the touch sensor, temperature sensor, and hall effect sensor that is present on the ESP32. Since it lacks some features, it is smaller and cheaper, coming in at 48 millimeters by 26 millimeters by 13 millimeters and $3 to $6.

Table 9: ESP32 vs ESP8266 comparisons

Characteristic	ESP32	ESP8266
Cost	$6 - $12	$3 - $6
Size	54.6 mm x 27.94 mm x 13 mm	48 mm x 26 mm x 13 mm
Clock Speed	160MHz	80MHz
Pin Count	50	25
Input/Output	Touch, temperature, hall effect sensors, micro-USB, Bluetooth 5.0, 40MHz 2.4GHz band.	Micro-USB, 20MHz 2.4Ghz band

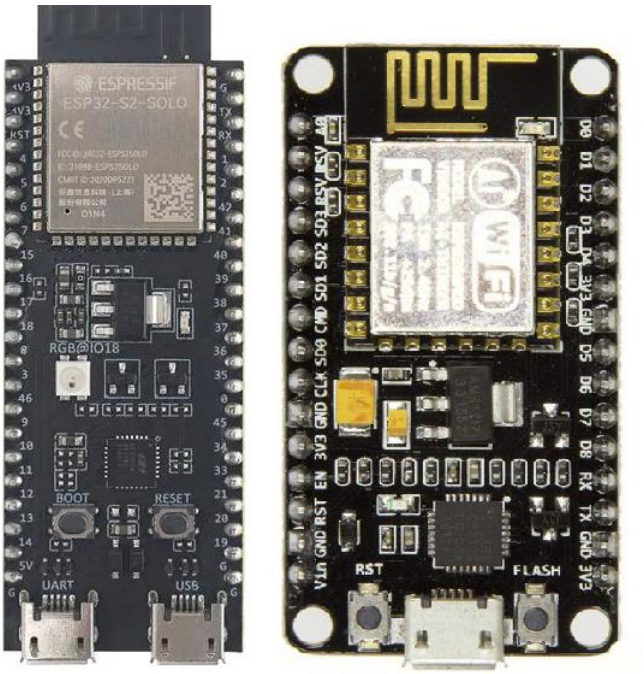

Figure 4: ESP32 (Left) vs ESP8266 NodeMCU (Right)

3.3 Sensors

The application for including sensors within our system is to include variety in access control for unlocking a door and provide home monitoring. We will detail the options for choices that could be provided in Live Bolt.

3.3.1 Voice/Phrase Recognition

Since the Raspberry Pi is processing the voice recognition software, a microphone is needed that is compatible with the Raspberry Pi. The microphone must also produce clean voice captures to be used in Artificial Intelligence applications. The voice input will be processed, and the program will determine whatever was said. If the word is a registered trigger word, the system will unlock the door. It would be best to incorporate multiple microphones into an array to get the best voice capture.

3.3.2 Face Recognition

For increased security, we would like to include some form of biometric facial recognition on the Live Bolt system. This can be implemented in a variety of ways, which are outlined below. Our main goal is to have a system that is as accurate as possible at detecting a registered face and rejecting unregistered faces. Overall cost, size and computing time can be greatly impacted depending on the method that is decided.

Table 10 outlines the differences between explored facial recognition methods.

3.3.2.1 Computer Vision

The Live Bolt will have a feature that unlocks the door when a registered face is detected. There are a few ways to go about this. The first method considered was to use a simple webcam and incorporate computer vision image processing methods through machine learning. This method would require a very large dataset of every person using the face identification features. The datasets would need to consist of at least 1000 photos of the person to start reliably detecting. It would require even more photos at different angles and light levels to increase consistency. Once a dataset is created, the photo captured during the unlock attempt would be passed through a convolutional neural network and eventually compared to the dataset. The output of the network would be a prediction if the subject is the same one as in the training dataset. Ideally, an accuracy of 99.5% or greater would be given every time for a positive match, and as close to 0% as possible for a negative match. Due to the limitations of computing hardware and

convolutional neural network training times, this method isn't the most reliable, so it would have to be combined with another form of access like the keypad, to ensure safety.

3.3.2.2 Intel RealSense ID

The second possibility that was researched was 3D mapping technologies like Intel RealSense ID and Xbox Kinect. 3D mapping, or photogrammetry mapping, allows for something in real life to be measured and projected onto a screen in the form of a 3D map or model. The measurement consists of taking small bits of data from different points around whatever you are mapping to create a depth map. One key advantage to 3D mapping for facial recognition, besides for more reliable recognition, is that the 3D map creates depth of someone's face. This keeps someone from simply showing a photo or model of someone else to the camera and have that be accepted as the target face. The process of using a photo of someone to trick a facial recognition system is known as spoofing.
Intel has developed a technology called RealSense ID that utilizes 3D mapping techniques to rapidly authenticate someone's face. They have all-in-one products that are ready to be used in a variety of applications straight out of the box, such as their F455 product. They also provide their architecture for custom designs in their F450 product. The features of the F455 will be focused on for the sake of this project. The F455 is 32.5 millimeters x 62 millimeters x 11 millimeters and has a viewing angle of 56° × 78°. It needs a Window, Linux, or Android operating system to operate. These devices have a 1:1,000,000 false acceptance rate (0.0001%), a 99.76% true acceptance rate, and a spoof acceptance rate of less than 0.1%. This makes them extremely secure and fit for a security lock application. Both models come with a system on the chip (SOC) which is an integrated circuit for computing and processing. They are also equipped with two 1920x1080 resolution cameras, IR sensor, 3D dot projector, and a light for more accurate capture. The Intel RealSense ID F455 is available for about $105.

3.3.2.3 Xbox Kinect

Microsoft has developed a product called Kinect for their line of Xbox gaming consoles that also utilize 3D mapping technology, for a gaming application rather than a facial recognition application. Since it is designed for gaming, it is meant more for detecting the body's of players for motion-based video games. The camera is equipped with a 30 frames-per-second 640x480 pixel resolution color camera and a 320x240 pixel resolution depth camera. The cameras also have a 57° × 43° viewing angle. The lower resolution isn't best for catching finer details, like what would be on a person's face. The way it is able to track bodies is by creating a virtual skeleton of up to two players on the screen. These skeletons have 20 joints per player and can track when the player moves one of their joints, like their legs or arms. The Xbox 360 Kinect is 76.2 millimeters x 279.4 millimeters x 76.2 millimeters. The module costs about $20 used.

A new version of the Kinect has been released for the newer Xbox One console and has an updated 1920 x 1080-pixel resolution 30 frames-per-second color camera and 512 x 424 pixel resolution depth camera that are able to track up to six active players. The newer model also has 26 joints that it tracks, rather than 20. It is smaller than the original Kinect, with a size of 66.8 millimeters x 248.92 millimeters x 66 millimeters and it costs approximately $80.

3.3.2.4 Smart Phone Face ID

The most well-known method of Face Recognition is the methods used on modern cell phones to unlock them. This section will mainly focus on the Apple iPhone Face ID and Android Face Authentication HIDL. They work similarly to the 3D depth scanner of the Intel RealSense ID technology. Since most people have a phone with face identification capabilities, this would be a good way to save space on our module while still maintaining quality face recognition. 90% of smart phones are expected to include some form of biometric facial recognition by 2024 [2]. They are also advantageous due to being able to detect someone's face with minor differences form the training dataset, such as if they were wearing sunglasses or if they grew a beard. Both platforms also have open-source application programming interfaces (APIs) for app developers to program them. Since this method incorporates the user's phone, there would be little to no cost associated with it. This would in turn keep the overall cost of the product lower, while also keeping the product size smaller since it would not need to house larger cameras or processors for face detection.

Since the group has an Apple iPhone 13, that phones specifications will be considered. The iPhone 13 is equipped with Apple's TrueDepth camera technology. This includes a camera that is capable of up to 3840 x 2160 pixels (4K) photos or video recording at 60 frames-per-second. The camera relies on a dot projector, similar to the Xbox Kinect, to detect enrolled faces. Over 30,000 dots are projected onto a user's face and it is able to detect whether the user is enrolled or not based on their facial structure. The TrueDepth camera is designed to work indoors, outdoors, or even in total darkness.

Table 10: Comparisons Between Facial Recognition Methods

Characteristic	Computer Vision	Intel RealSense ID F455	Xbox One Kinect	Smart Phone (iPhone 13)
Depth Camera Resolution	Dependent on webcam	1920 x 1080 pixels	512 x 424 pixels	30,000 dots

Color Camera Resolution	Dependent on webcam	1920 x 1080 pixels	1920 x 1080 pixels	3840 x 2160 pixels
Frames per second	Dependent on webcam	90	30	60
Detection Method	2D Neural Network matching face to database	3D dot projection/facial mapping	3D Depth/joint tracking	3D dot projection/facial mapping
Physical Size	Dependent on webcam	32.5 mm x 62 mm x 11 mm	66.8 mm x 248.92 mm x 66 mm	147 mm x 72 mm x 7.65 mm
Viewing Angle	Dependent on webcam	56° × 78°	70.6° x 60°	120°
Cost	Dependent on webcam	$105	$80	$830

3.3.3 Near-Field Communication

Near-Field Communication (NFC) is a popular use in today technology of contact-less communication from using radio frequency. This technology is used to primarily to exchange data within two devices through a touch gesture. The NFC controller needs to be connected with antenna transmit and receive all NFC communication frames to the NFC Forum Device. Using the NFC forum device will be an integration of using software in our application for Live Bolt and making sure it can recognize the chip that'll be placed in our design.

Since NFC is very flexible in operating in different modes, the most ideal application of NFC to Live Bolt would be through Host Card Emulation. Due to our idea for making it easy for the user to enter their home, having an NFC controller forward contact-less commands to the Device Host.

3.3.4 Infrared Sensor

The infrared sensor is integral to the security system because it will help to identify is someone has entered in front of the Live Bolt lock. Any Infrared sensor that is compatible with Arduino or Raspberry Pi would be fine for our application. Ideally it would have 2-5 feet of detecting distance. Using an infrared sensor for motion detection rather than another method like computer vision or ultrasonic sensing is beneficial because it can detect changes in temperature that the other sensors cannot, and therefore can tell is a human or animal is being detected. An ultrasonic sensor would trigger whenever anything gets into range, such as leaves or other debris.

3.3.5 Security Camera

The Live Bolt system will include a security camera for the customer to view a live feed of the area in front of their door. The security camera has two main requirements: a high Field-of-View and good resolution. This way it can view the largest area with the most clarity. Since most cameras are compatible with Raspberry Pi models, we needed to compare USB cameras, like webcams, to integrated camera modules that connect to the Raspberry Pi via ribbon cables. A secondary requirement for the camera is how much space it will take up. It will need to be mounted to the outdoor module of the Live Bolt, so it would be better for it to be as small as possible while maintaining Field-of-View and resolution quality.

3.3.6 Keypad

The number keypad will be used by guests to unlock the door with a one-time code that is 4 to 6 digits. If there was a single code that always unlocked the door, those keys would begin to fade and make it easy for an intruder to guess the code. The main requirement for this keypad is for it to be weatherproof, mainly from rain since it will be on the outside module, so it must be made of waterproof plastic or metal. An alphanumeric pad could be useful for added security; however, a numeric pad would also be fine for the scope of this project. There are many types of keypads available that fit these requirements such as membrane keypads, dome-switch keypads, and capacitive keypads.

A membrane keypad is probably the most popular option for products that require labeled buttons. They work by having two layers that are separated by an air gap. When the top layer is pressed by a user, it completes the circuit and allows the computer to know which button was pressed. Dome-switch keypads are similar to membrane keypads, except they are more mechanical and provide a tactile click when pressed. The sound and touch feedback help usability. Dome-switch keypads also have incredible reliability, being able to be clicked between one and five million times. Capacitive keypads act similarly to the membrane keypad, except when you press, it charges a capacitor. The change in the capacitors charge is detected and the computer can determine which key was pressed.

3.4 Power Sources

All the computational devices we have identified operate off 5V. This means we will need a reliable power source that can output 5V for a very long time. To identify the main source of power to operate our system, we will look at a variety of options to execute its functionality.

3.4.1 Solar Panel

Powering our system from solar panels was an early goal that we wanted to research more. Since solar panels do not output their peak voltage 100% of the time, a battery pack would also need to be included to keep the system running. A panel rated for at least 5V would be required. A possible issue with using solar panels is that the lock system might be shaded from the sun, making any solar panel useless. Solar panels also use up more space, and we would like to keep the system as compact as possible.

3.4.2 AC Power

The most reliable method to power the system would be to tie it to the 120V, 60Hz, AC power used to supply power to the rest of the home. However, this would mean that when the house loses power for any reason, the lock would also lose power. Working with 120V is also potentially dangerous for the engineers designing the system, and the customers installing it. Ease of installation is something we are working hard towards and having to tie the system to a house power system may not be safe for someone who is unexperienced. This method would also not be plausible for someone living in an apartment complex.

3.4.3 Conventional Battery Pack

Including a battery pack seems like the best method to supply power to the system. Since both Arduino and the Raspberry Pi operate off 5V, a pack of at least 5V is required. Most batteries are more or less, but not exactly 5V. This means that the battery supply will most likely need to be stepped up or down to 5V to meet power consumption needs. A major issue with batteries is that they will inevitably run out of charge and the system will lose power. We would like to make the battery packs easily swappable and rechargeable so customers can always have power to their system. Rechargeable AA batteries typically output 1.25V, so including four in series would provide us with the 5V we need. Since we want to include rechargeable batteries, something like a Lithium Polymer battery would not be the best choice for the engineers or the customer These types of batteries are very easy to mishandle by someone without experience and can lead to fire, smoke, or explosions if over or undercharged.

3.5 Locking Methods

Included in our objective, we want to make sure that there is a clear sense of security to the homeowner, hence the reason for including a good locking method. Below are the options we've discussed as a team that would be liable for improving security when designing Live Bolt.

3.5.1 Electric Strike Locks

Arguably the most important feature of this product is how it will physically lock a door and keep unwanted visitors out. The main methods that were considered for this project were electric strike locks, magnetic locking mechanisms, and traditional dead-bolt locks.

Electric strike locks and magnetic locks are both typically incorporated into doors of most institutional and commercial buildings, like schools or shops. An electric strike lock operates very similarly to a dead-bolt lock, except that instead of a human turning the bolt, an electric impulse causes the bolt to strike into place. A benefit of the electric strike lock is that once it is locked, it no longer needs an electrical signal and can keep the door from opening. Figure 5 provides how an electric lock looks on a door.

Figure 5: Example of an Electric Strike Lock on a Door [3]

3.5.2 Magnetic Locks

Magnetic locks feature an electromagnet on the doorframe and a piece of metal attached to the door. When the door is closed, the magnet and metal will make contact. Then, if an electrical signal is sent through the magnet, it will activate and attract the metal piece, causing the door to remain locked closed. The electrical signal for both locks can be sent via multiple methods but are typically sent from a key card reader or keypad. The magnetic lock needs a constant source of power to remain locked, or else the door will be able to be opened. A magnetic lock is shown in Figure 6.

Figure 6: Magnetic Lock on Door [4]

3.5.3 Dead bolt Locks

A traditional dead bolt lock is what is seen in most homes. It operates by having the user manually turn a thumb knob located on the inside of the home. When this knob is turned, the bolt is struck into place, blocking the door from being opened. To automate this task for the purpose of the Live Bolt system, having a servo motor turn the bolt would be best because it allows the door to be locked or unlocked remotely without the need for a person to physically be at the door. Most commercial smart locks on the market opt for a method of having customers unscrew the doorknob off their door and attach a unit that houses the servo motor to the inner mechanisms of the lock. Another method requires the thumb knob to be fitted with a cover. A servo motor will then turn this cover, and therefore the knob, to unlock the door. A downside to this is that not every door thumb knob is the same shape and size, making the cover hard to replicate for different doors. A dead bolt is shown in Figure 7.

Figure 7: Traditional Dead Bolt Lock on Door [5]

3.6 Feedback System

In creating Live Bolt, we want to ensure that there's home monitoring within the vicinity of the home. From that outcome, it's only logical to include a feedback system to the homeowner about any undisclosed guests within their property. This will help enhance the responsiveness and security of our system. The following feedback systems will be considered to take place in our design.

3.6.1 LED/Buzzer on Physical Device

One tactical way to prevent intruders from entering a home is through a reactive system. When an alarm goes off it typically does resign well within the situation. In our design, we essentially want to incorporate a buzzer/ flashing LEDs onto the physical device when entering through force. That way the intruder is indicated that our system is aware that there is entry and will notify the homeowner. These reactive systems may result in the intruder not entering the home which would be in favor of the customer. Visual deterrents such as these can help prevent a potential break through immediate feedback which will prove useful in our system.

3.6.2 Application Notifications

An added feature we'd like to incorporate is to provide notification through an application of someone unlocking the door, movement within the door entrance, and whether there's been a force entrance. This will be done with the use of our sensors cooperating with the raspberry pi. This information will then be relayed to our mobile application for the user to evaluate. The user interface will have the option of being sent notification of the following home indication. With including mobile notifications, we hope to provide security to our consumers knowing that their homes will be monitored 24/7.

3.7 Parts Selection

After considering all the possible parts listed above, we have narrowed down what we think would best fit the design we have created. All the parts chosen are listed below. Each part and its technical specifications will be discussed further in Section 7.1 Hardware Selections. A breakdown of each components cost and where they were sourced is in Table 18: Bill of Materials.

3.7.1 Computation and Control Device

It was decided that for the computational device, that a mix of using a Raspberry Pi and ESP8266 would provide the best combination of processing power. The Raspberry Pi is responsible for interfacing with the microphone for phrase detection, the security camera for video streaming, and hosting the phone application. The ESP8266 will process the NFC scanner input, the number pad input, and the passive infrared sensor. The output for the ESP8266 will be the security system, which consists of an LED and buzzer.

3.7.2 ReSpeaker 4-Microphone Array

For the voice capture device, we opted for the Seeed Studio ReSpeaker 4-Microphone Array. This product is designed for all Raspberry Pi models. It has four microphones, one in each corner of the board. This allows for a higher quality sample to be collected and used by the Raspberry Pi for phrase detection. This board is based on the AC108 quad-channel analog-to-digital converter (ADC) for high quality voice capturing.

3.7.3 Smart Phone Face Identification

Due to cost, computing, and size constraints, it makes the most sense to utilize the Apple iPhone Face ID and Android Face Authentication HIDL that is already built into most user's cell phones. This will mean that most of work that goes into this section will be done through the smart phone application. Both iPhone and

Android have open-source libraries that can be used to include the face recognition features of their platforms on our application.

3.7.4 HiLetgo NFC Module

The HiLetgo PN532 NFC kit is made up of a key card, key ring, and the actual NFC module that supports NFC/RFID reading and writing. It operates at 5V for I2C and UART communication, and 3.3V for SPI communication. The module is very small and can easily be incorporated into the final product.

3.7.5 HC-SR501 Infrared Sensor

The Stemedu HC-SR501 PIR Motion Sensor was chosen for the infrared sensor because it is a passive infrared sensor. This means that it can detect changes in temperature in its viewing area to see if a human or animal is in view. Passive infrared sensors are the most-used type of motion detector for security applications so this will be easily integrated into the final design. When someone enters its viewing area, it will output a high voltage level. After they leave, it will go back to a low voltage output. This sensor features an option to adjust the delay time and block time. Delay time is how long after someone leaves its sensing area that it will remain outputting a high voltage before outputting a low voltage. Block time refers to the amount of time after going back to a low voltage before it can be triggered and set to a high voltage again. The delay time can range from 0.5 seconds to 200 seconds and the block time can range from 0 seconds to over 100 seconds. These delays are adjusted via potentiometers on the board itself, which eliminates the need for any programming with a microcontroller. This sensor offers a range of about 5 meters and a viewing angle of about 100 degrees or less.

3.7.6 ArduCam Security Camera

The biggest requirement for the camera is to have a small form factor while retaining enough quality to be a good choice for security applications. The ArduCam is a camera module that interfaces with any Raspberry Pi module via the 15-pin ribbon cable on the top of most Raspberry Pis. The ArduCam offers 1080-pixel resolution at 30 frames-per-second video quality and a viewing angle of 54 degrees by 41 degrees.

3.7.7 Keypad

The group opted to go with a simple matrix membrane keypad that can interface with most microcontrollers, including the chosen ESP8266. It is a 4 x 4 pad with buttons for numbers zero through nine, letters a through d, as well as special

characters for “*” and “#”. It also comes with an adhesive on the back to be easily fixed to our final design.

3.7.8 Rechargeable Battery Packs

We have opted to go with rechargeable battery packs to power each module of the design. We have decided to include rechargeable AA batteries in these packs. The technical specifications such as how many batteries in each pack and how much power they will produce will be discusses further in section 7.2 Power Distribution.

3.7.9 Traditional Dead-Bolt Lock

To keep the installation as simple as possible, it was decided to keep the traditional deadbolt design on the door and instead have a servo motor turn the thumb turn that sets/unsets the dead bolt in place. The servo motor that was chosen has a torque of 10 kg*cm and can rotate up to 170 degrees. Typically, a servo motor with at least 180 degrees turn radius is desirable, but since we only need to turn the lock 90 degrees to lock/unlock it, 170 degrees is more than enough.

3.7.10 Feedback System

The way the system provides feedback to the user will be in two ways. The first method will be by indicators on the physical system. If an attempt to unlock the door fails, a buzzer will beep, and an LED will flash to indicate that the attempt was unsuccessful. The second method will be via the phone application. When a attempt to unlock is made, it will send a notification to the application and indicate whether the attempt was successful or not.

Category	Part Chosen
Computational Device	Raspberry Pi 3B+ and ESP8266
Phrase Recognition	ReSpeaker Four-Microphone Array
Face Recognition	Smart Phone Face Identification (Apple iPhone Face ID and Android Face Authentication HIDL)
Near-Field Communication	HiLetgo Near-Field Communication Module
Infrared Motion Sensor	HC-SR501 Infrared Motion Sensor
Security Camera	ArduCam Security Camera for Raspberry Pi
Number Pad	4x4 matrix membrane keypad
Power Source	Rechargeable battery packs
Locking Method	High-torque servo controlled dead bolt
Feedback System	Phone application and physical LEDs/buzzers

4. Related Standards and Realistic Design Constraints

4.1 Standards [Maybe 20 standards?]

Within the area of Engineering Design and project creation, there are several standards to follow. These standards are important because they directly affect the way we design our Live Bolt system and the technologies we will use. Our Live Bolt system will utilize a few of these standards as it operates. We will need to adhere to these standards when including Wi-Fi, Bluetooth, RFID, and NFC technologies in the Live Bolt system.

4.1.1 Wi-Fi Standards (802.11)

Originally, the first standard for Wi-Fi was created in 1997 by the Institute of Electronics Engineers (IEEE). It was called IEEE 802.11 and had a network bandwidth of only 2 Mbps. This is very slow compared to the most recently accepted Wi-Fi 6 standard, also known as 802.11ax, which can support up to 10 Gbps. Since the creation of the 802.11 standard there have been many Wi-Fi standard iterations which each have their own advantages and disadvantages. As a result of this just because a standard has been approved by IEEE does not mean that it may be the best in every situation. As such the Wi-Fi standard we choose to use will be taken into careful consideration when we design the Live Bolt system.

As mentioned before the most recent Wi-Fi standard that was approved was Wi-Fi 6 back in 2019 by IEEE. However, this is not the Wi-Fi standard that will be used primarily in our system. We will be using a Raspberry Pi 4 in our system which uses the 802.11ac standard or Wi-Fi 5. This standard supports simultaneous connections on both 5 GHz and 2.4 GHz Wi-Fi devices. It also boasts a bandwidth of at most 1300 Mbps when using the 5 GHz connection and 450 on the 2.4 GHz. [14]

4.1.2 Bluetooth Standards (802.15.1)

Bluetooth or IEEE 802.15.1 was originally first standardized by IEEE like Wi-Fi. It was first introduced in 1998 and has since had its standard management transferred to the Bluetooth Special Interest Group (SIG). This group manages Bluetooth standards which manufacturers must follow if they want to include the technology in their devices.

Currently the only devices that will be utilizing Bluetooth in the Live Bolt Smart Lock system will be our Raspberry Pi 4 and a smart phone connected to our system through the app. The Raspberry Pi 4 uses Bluetooth 5 technology. Bluetooth 5 was released on December 6th, 2016 and introduced many improvements over the past generation. Bluetooth 5 provides support for Bluetooth Low Energy, or BLE, which is a subset of Bluetooth 4.0 technology that is aimed at supporting low power applications. Some of the new features of Bluetooth 5 includes options to double the speed while reducing the range and quadrupling the range at the cost of the data transmission. [17]

4.1.3 RFID Standards

RFID is a technology that uses electromagnetic fields to uniquely identify items. A typical RFID system involves the interaction between RFID tags and readers. We will be using an RFID in the Live Bolt system as a possible method to unlock the lock. There are many regulations and standards that are in place for RFID. These guidelines are in place to protect people and animals from harm that certain frequencies can cause.

There is not a central organization that manages RFID communication. Instead, there are several industries and organizations that have set their own standards for RFID communications. Additionally, each country can set its own rules for the frequency bands allowed for RFID communication. These bands are also known as the ISM, Industrial Scientific and Medical, bands on the electromagnetic spectrum. These are a portion of radio wave bands laid out on the spectrum reserved specifically for scientific, industrial, and medical purposes. We will carefully consider the various standards as we complete our design for the Live Bolt Smart Lock.

4.1.4 NFC Standards

NFC is a similar technology to RFID. It is a communication protocol set that allows two devices to communicate over a small distance, roughly 4 centimeters. NFC technologies are mainly used to facilitate contactless payment systems. However, in the Live Bolt system it will be used as an authentication technology for unlocking the lock.

The actual standards for NFC communication are based on similar standards to RFID. These standards include ISO/IEC 14443, ISO/IEC 18000-3, FeliCa, and those defined by the NFC Forum. These standards deal primarily with identification, proximity, and contact card regulations in use with NFC. ISO/IEC 14443, and ISO/IEC 18000-3 are important NFC standards as the first deals with the way identification cards can store information. The second deals with the way RFID communication can interface with NFC technology.

We will be using the HiLetgo PN532 Module kit for both our NFC and RFID communications. It supports reading and writing for both technologies. Several cards and standards work with the kit such as ISO/IEC 14443-4 cards, FeliCa cards, and Innovision Jewel cards. The kit uses I2C, SPI, and High Speed UART for communication. [16]

4.1.5 Measurement Standards

To keep the design process of the project consistent, we would like to note that our design will keep in the international System of Units (SI). This choice was determined because it used universally in technical and scientific research to avoid confusion when converting between units. This would help those unfamiliar with the imperial system since it’s less popularly used. Specifically, we would like to keep all set measurements in design to millimeters (mm) when obtaining dimensions.

4.1.6 Deadbolt Lock Standards

Every lock needs a locking mechanism, and the Live Bolt system is no different. The locking mechanism we will be using for our project will be a deadbolt lock. Deadlocks, like many other tools we will be using, have standards in place that will assist us in our design.

There are a few different ways deadbolt locks can be configured. These include the Single Cylinder Deadbolt, Double Cylinder Deadbolt, One-Sided Deadbolt with Outside Trim, and One-Sided Deadbolt Without Outside Trim. A single cylinder deadbolt is a two-sided lock where the outside side has a key slot to turn, and the inside part of the lock has a thumb turn. With the double cylinder deadbolt there is a key slot on both sides of the lock allowing a key to be used

from the inside and outside of the door. A one-sided deadbolt with outside trim involves a plate being placed on the outside of the lock with a thumb turn on the inside of the lock. This allows the lock to be opened from the inside only. A one-sided deadbolt without outside trim is a configuration where there is nothing on the outside of the lock but there is a thumb turn on the inside. Like the one-sided deadbolt with outside trim configuration the lock can only be opened from the inside with the thumb turn. [15]

ANSI Grading System was formed to have a standard procedure to test the production quality of a lock. It was developed by the Builders Hardware Manufacturers Association (BHMA). There are currently 3 grades ranging from Grade 1 to Grade 3. Grade 1 is the most secure grade while Grade 3 is the least. Grade 3 locks are the least secure locks and are the locks used in many homes and residences. They also consist of a 5/8-inch latch bolt. They are the least expensive of the 3 grades and should not be used in situations that require maximum security. These locks are typically easy to fiddle with and can be picked easily to unlock. These locks are not recommended if the customer has enough money or access to another Grade. Additionally grade 3 deadbolt locks are tested to maintain functionality for at least 800,000 cycles. They also are tested to resist 2 strikes from at least 75 pounds of force. Grade 2 locks are used on many residential houses and buildings. They are a bit more expensive than grade 3 locks but pay off with the increased security they offer. This grade is also built to withstand 800,000 cycles and is a 5/8-inch latch bolt. This grade can withstand 5 strikes from 75 at least pounds of force. Grade 1 locks are the strongest of all 3 grades, however they are the most expensive. As such they are used in many commercial applications and applications that require the best quality locks. They are tested and certified to withstand 1,000,000 cycles for usage. This grade can handle 10 blows from at least 75 pounds of force. Similarly, to the other 2 grades this grade also is a 5/8-inch bolt lock. [15]

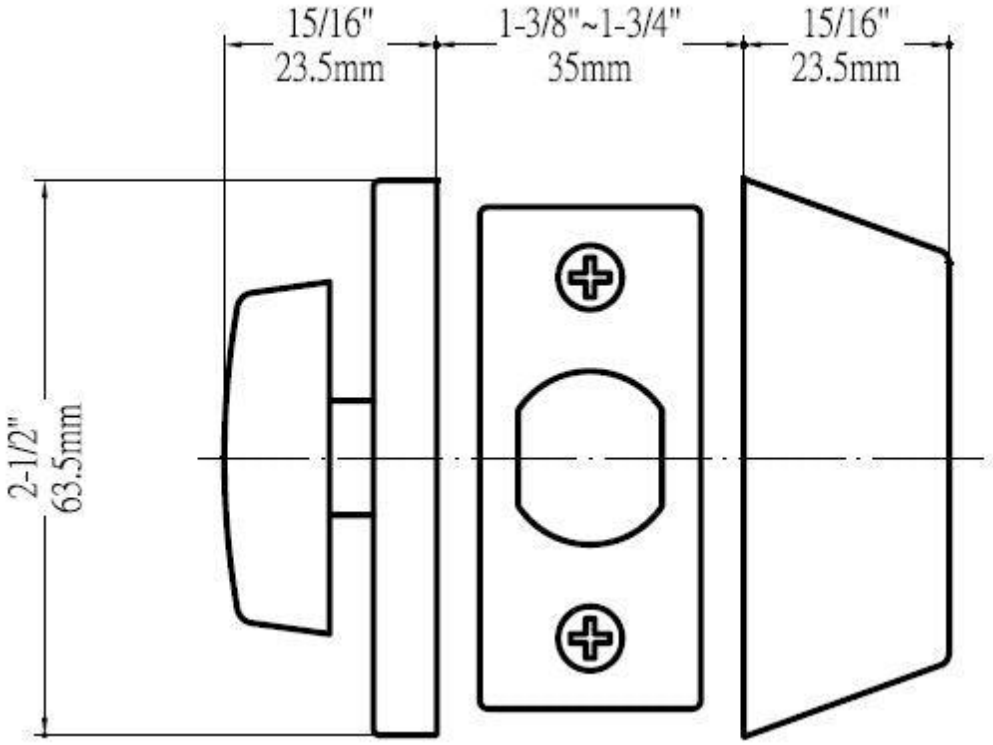

Figure 8. Deadbolt lock size standards

4.1.7 Battery Standards

When designing a system, it'll need a source of power to have all the electronics running and functioning. Our design choice for powering the components would be through AA batteries. AA Batteries is a single cell cylindrical dry battery. It falls in the line of General Batteries with the standard number International Electrotechnical Commission (IEC) 60086. They're commonly known for being in portable electronic devices that can be disposable or rechargeable. AA cells will typically have a 1.2 – 1.5 V terminal voltage unless specified by the manufacturer. The American National Standards Institution (ANSI) standardized AA battery size in 1947. AA cell measures 49.2 – 50.5 mm in length (with button terminal) and 13.5 – 14.5 mm in diameter. [6]

4.1.8 Face Recognition Standards

With Facial Recognition Technologies on the rise there are standards in place to regulate how they should be used. The Facial Identification Subcommittee creates standards and guidelines for facial recognition technology. The four main standards are ASTM E2916-19e1, ASTM E3115-17, ASTM E3148-18, and ASTM E3149-18. ASTM E2916-19e1 deals with the terms used for discussing digital and multimedia device testing. ASTM E3115-17 deals with guidelines for

the capture of facial images within facial recognition systems. ASTM E3148-18 deals with standards relating to the taking of facial images of deceased people. ASTM E3149-18 deals with standards for comparisons between different facial images. Since facial recognition technologies are new there are still many standards currently under development for facial recognition technology.

4.1.9 Programming Best Practices

Now for a view of some specific coding standards. The following are some of the best programming practices:
1. The first is regarding limiting the use of global variables. These include rules explaining which data types should and shouldn't be declared global variables.
2. The second is that standard headers should be used when working with different modules. This helps for easier understanding and maintenance of code. Some things that the header should contain include the name, date it was created, the modification history, and the author of the module.
3. Third are standards on naming conventions for variables, constants, and functions. They should be given meaningful and easy to understand names so that people unfamiliar with the project can understand their purpose. Global variables should use camelcase lettering with the first word having a lowercase letter and the following words being capitalized, for example localVariable. Global variables on the other hand should have the first letter of each word be capitalized, for example GlobalVariable. Constant variables should only use capital letters, for example CONSTANT.
4. Fourth is guidelines regarding indentation. Indentation is especially important due to the readability of the code.
5. Fifth is regarding handling error return values and exception handling. The general guideline is that functions encountering errors should return 0 or 1 to indicate whether an error has occurred or not.
6. Sixth is to avoid using a programming style that is hard for everyone working on the project to understand. This makes present and future development more difficult as the code is written. It also makes testing and debugging much harder later.
7. The seventh is that identifiers should not be used for multiple purposes. This means that each variable should be given its own unique name, making it distinguishable from other variables. By giving each variable its own unique name confusion can be avoided when referencing that variable.
8. Eighth is that any code written should be explained through good documentation. Comments should be provided and easily understandable to make code simple to follow.
9. Ninth is that any functions written should not be long. Long functions make code difficult to understand and follow so shorter functions are preferred.
10. Lastly avoiding the use of GOTO statements is preferred. GOTO statements ruin the structure of the code where they are placed. This makes reading the program more difficult as it is harder to follow along with where the statement jumps to. [7]

4.1.10 Apple App Store and Google Play Store Standards

We plan to release our mobile application, which will act as a Companion for our Live Bolt system, to both the Google Play and Apple App store. As such we will need to follow the standards and guidelines laid out by both corporations to get our application approved.

Google Play has several policies regarding what developers can and can't do when developing applications for their platform. The following are the main policies listed to follow for publishing an application to the Play Store:
1. The first of which is regarding restricted content. This includes restrictions regarding child endangerment, inappropriate content, financial services, real-money gambling, illegal activities, user generated content, and unapproved substances. This group of guidelines is included primarily to protect users from illegal, inappropriate, and deceptive content.
2. The next is impersonation which includes when a new application tries to fake being an existing application. This could be something as simple as containing the same image, using the full name or part of an existing app's name. It also ranges to the extreme to an app attempting to be another app in the store. This restriction is important as it seeks to protect the user from downloading a fraudulent app that is not what it claims to be.
3. Another guideline is restrictions against intellectual property. This protects existing developers with restrictions on Copyright, Trademarks, and counterfeit content. New and existing applications are not allowed to impose on the intellectual property of an existing application.
4. The next guideline deals with user privacy and device protection. This policy deals with protecting user data and specifies how an application is allowed to use the existing technology on a device. It seeks to prevent device, network, and permissions abuses as well as deceptive behavior within applications.
5. The next policy deals with Monetization and Ads. This is a broad category that deals with paid distributions, ad-based models, in-app payments, and subscriptions. This also includes the use of Google's billing system, free trials, introductory offers, and managing of subscriptions. It also places regulations on ads and how they can appear in applications. Ads can't be deceptive and can't be disruptive for the user experience. Without these restrictions users are placed at risk and Google's reputation for safe applications would sink.
6. The next policy relates to store listings and promotions. This affects how the app appears to users, such as on the applications store page, on the google play store. This is to prevent spammy store listings, poor quality promotions, or companies finding ways to falsely boost reviews for their product.
7. The next policy deals with spam and the minimum functionality an application needs to remain on the store. An application needs to function well and not include unfinished content to prevent the user from using it. Apps that crash and

degrade the user experience or spam the user with content are not appropriate for release on the store.
8. The next policy deals specifically with malware. Specifically, applications that contain any malicious content such as viruses, binaries, framework modifications, and potential harmful applications.
9. The last policy for google applications is regarding unwanted software. This deals with software that is maliciously placed to download alongside an application, send information against the user while using the application, or fails to be completely transparent with the user about applications intentions. [8]

Apple similarity to Google has several policies in place to ensure the safety of its app store. The guidelines include:
1. App Store Review Guidelines which cover policies related to design, safety, performance, business, and legal information.
2. App Store Identity Guidelines relating to guidelines involved in using marketing tools that lead back to the Apple Store product page. It also covers using Apple App Store badges in all marketing material to link back to the app page.
3. The next policy deals with App Store Promo Artwork Guidelines which only applies to applications that decide to create their own promotional artwork. These included guidelines on creating original artwork and submitting promotional artwork for review to Apple before they can be used.
4. The next policy deals with Apple Wallet Guidelines. This deals with using the Apple Wallet and Add to Apple Wallet buttons in apps, as well as other material related to the app.
5. Human Interface Guidelines are next. They deal with integrating UI resources to create a smooth and Universal experience across Apple’s many platforms.
6. The final set of policies are regarding Apple Pay Marketing Guidelines. This policy deals with the ways and methods developers can let customers know they can use Apple Pay. This can be both through marketing as well as the application itself. [9]

4.1.11 Design impact of relevant standards

When designing our system many of the standards we have discussed will limit our design in some capacity. As such we will need to consider these standards closely when designing our product.

One instance of this are the standards previously talked about in section 4.1.1 based on technology that must communicate wirelessly. As such the limitations of wireless technology compared to wired technology will impact our system. Wi-Fi may limit the time it takes for our Raspberry Pi with our mobile application. Although we are using a recent version of the Wi-Fi standard it is fallible. As such we will be limited to the normal constraints of Wi-Fi based on the connectivity of our devices, interference mediums, and congestion. These constraints could be reduced by using a wired connectivity medium such as ethernet, however this becomes more inconvenient for the user to use. Additionally, although Wi-Fi 5

will be used in our design we will not be losing much in terms of functionality when compared to Wi-Fi 6. Wi-Fi 6, although newer and faster than the previous generation, is still relatively new in its adoption. As such until Wi-Fi 6 reaches wider adoption there is no reason to upgrade over Wi-Fi 5.

As previously discussed, Bluetooth 5 will be used in our Live Bolt system. Bluetooth is a type of Personal Area Network, or PAN, which is a type of computer network designed to operate within the space around a person. As such Wi-Fi holds an advantage over Bluetooth in terms of connectivity. However, the introduction of Bluetooth 5 brought significant improvements to the range and connectivity over Bluetooth 4.0. This will help with connecting with mobile devices that use Bluetooth 5.

Both RFID and NFC work over a set distance like Bluetooth. NFC has a set range of 4 centimeters or less which is ideal for close communication. This is a bit different than the range of RFID tags and readers which can vary greatly. NFC unlocking of the lock will require the device to be in very close proximity to the Live Bolt system. The range of NFC is really it's only downside however this downside also provides a huge benefit. NFC is more secure than RFID because of its restricted range which prioritizes the secure connection made. For NFC if someone wanted to interfere with the connection, they would need to be very close to the devices that are communicating. With RFID data can be read from potentially many feet or yards away. This is already a problem with RFID readers which can detect information from credit and debit cards left out in the open. We want to avoid security issues like this as much as possible in our system.

4.2 Realistic Design Constraints

Seeing as we will be designing a Smart Lock system, we know that we will be up against several design constraints. These constraints will have an inevitable impact on the overall development of our product.

4.2.1 Knowledge constraints

One of the biggest constraints on our design will be our current knowledge. No one in our group has designed a Smart Lock system before so it will be a new learning experience for us all. Additionally, our team is made up of 2 computer and 2 electrical engineers with varied levels of experience. We also plan to integrate a wide variety of communication technologies as unlocking mechanisms for the lock. These technologies include Bluetooth, NFC, RFID, fingerprint scanning, and face unlock to name a few. Getting the connections to be established and function the way we would like will involve some research. We will be integrating the mobile application to act as a companion for the Smart Lock system. Not all of us have worked with a Raspberry Pi and interfacing it with an application. That is a whole other area that we will have to explore while we

plan our development. We will also be using an Arduino Uno which we all have varied levels of experience with. Another important aspect that we will need to research is testing our design once we have completed it. This is something that we will not need to worry about immediately but in the future when we must submit our design it will be particularly important. With that both the hardware and software elements of our design will need to be tested to ensure accuracy. We wouldn't want to justify software bugs or hardware breakdowns just because this is our first time designing this system. We do expect that our classes and previous work experiences have laid a solid foundation for us to build upon when tackling these issues.

4.2.2 Economic constraints

Of course, economic concerns are going to come into play when working on a project like this where funding will be provided by the team. We happen to also be an unsponsored team so the funding will come from our own pockets. As such the price of the project can't be very expensive. The more expensive our project is the more money we will have to pay out of pocket. This is a significant constraint as if we had a high budget, we would have more room for experimentation and a better-quality product overall.

As of right now we have been able to keep our budget down to 200 to 300 dollars overall. Our estimated total is 242 dollars which comes out to about 61 dollars per team member. This is a very good theoretical cost as it is low enough that each member of the team can afford it. It also leaves flexibility that if a bit more money is required, such as with parts that fluctuate slightly in price, it should not cause an issue. We are currently sourcing the parts now and will see how much the price fluctuates when we have everything. We also were able to exclude some items that we already have from our price list. These items include the Arduino Uno, a Pin Pad, Wires, LEDs, the Buzzer, and other minor electrical components. This helped reduce the price while helping the cost of the project to not become a major constraint.

Another economic constraint involves the Covid-19 pandemic. Although it has nearly been 2 years since the virus broke out and the quarantine began, we are still feeling the effects. Most notably is the chip shortage among different technological sectors. Likewise, the demand for many different commercial items that use silicon chips has still not been met like previously before the pandemic. This has driven up the prices of things that rely on these chips. Although the automobile and interactive entertainment industries have been notably affected, they are not the only ones. When searching for parts for our project we ran into a couple parts that had been affected. These products were the Raspberry Pi 4 and the Jetson Nano. Both product versions came out around the beginning of pandemic and were readily available when things shutdown. However now as one of the aftereffects of the pandemic it is difficult to get them both for several reasons. The most significant is the price increase for them both. For instance,

the Raspberry Pi 4 is set to retail at around 45 dollars for the 2 GB model. However, upon comparing online sellers like Amazon, eBay, and Newegg we were able to see listings for the same model anywhere from 80 dollars and up. For the Jetson Nano the situation is worse. The Jetson Nano originally cost 59 dollars for the 2 GB model and 99 dollars for the 4 GB model at launch. When looking at online retailers, now the prices have gone up from 200 to 500 for both models. Additionally, at many of these retailers' stock is sold out, making obtaining both harder. If this were before Covid-19 obtaining these would be more readily available. Ultimately, we went with the Pi due to the cheaper price and goals of our project. We hope as we continue to purchase the parts, we will be using in our Live Bolt system we will not be further affected by price increases from Covid-19.

4.2.3 Time constraints

Time will also be a big constraint as we attempt to design our Smart Lock. The first major constraint with time will be how long we have to complete our project. We only have the remainder of this spring semester and the following fall semester to finish our project. We have realized that this is not a lot of time, roughly 3 to 6 months. We plan to circumvent this by trying to gather our parts early, researching parts of our design, and designing some fundamental components before fall this year. The best that we can do is to get ahead with what needs to be complete for our project to combat the time crunch. The constraint of time may also cause us to need to cut out or modify features if we are unable implement them within the time frame given. We have already begun doing this while we are still planning and researching our design. We originally wanted to use facial detection using a camera we would include in our Smart lock system. However, this would require training a model for facial detection as well as testing that model for anticipated behavior. This would already be a decent chunk of time that we would have to work on our project. When talking about it again as a group we decided to just use the face unlock through the mobile device. This is much easier to implement since the feature is already built into the mobile device. We would just have to prompt the user to scan their face on their mobile device which would be a better time tradeoff. We plan to do the same with fingerprint scanning. We hope that these will help us save time while keeping the original features we intended to implement.

Another reason why time may become a constraint is due to our schedules. Our team has 4 different people all with varying lives. We all have certain class schedules, work schedules, personal obligations, clubs, and other activities going on in our lives. Additionally, our schedules will inevitably change when we continue into Senior Design 2 in the fall semester. As such we can't all always be available when we need each other. Working around each other's schedules will be something that we will need to keep in mind when we begin to build our project. We plan to get around this by trying to meet regularly in Senior Design 2. For now, we are meeting about once a week as we work on our paper. For now,

this is working for us planning, researching, and writing the paper. Unfortunately, we can't predict what will happen in the future, however we plan to stay ahead to prepare for any unforeseen circumstances.

A significant factor affecting our time is due to the pandemic. Previously in the economics section we discussed how Covid-19 has led to an increase in prices that can be seen especially with devices that use chips. Many of these products are sold out which can make it difficult to find places that still do have them in stock. Waiting for a product to come back in stock will take time. Many times, it is unknown when they will come back in stock and often, they tend to be popular enough to sell out again quickly. This leads to price increases like the ones that we discussed in the Economics section. As such we are starting to collect our components earlier so that we can try and get around these inconveniences. Another issue that goes along with this are shipping times. Shipping times can vary largely depending on the seller. Availability as we discussed earlier can also influence shipping times as they can cause them to become extended. These delays can be anywhere from a little inconvenience of a few weeks to as great as a few months. The amount of time that a part takes to come will delay us from being able to work with that component. At the worst case it can cause us to reconsider working with that component even to the extent of removing it from our design or considering another technology. Unfortunately shipping times are out of our control so we can only do our best to obtain the parts earlier so we can plan accordingly later.

Another time constraint we will need to be aware of is workload management and deadlines. There will be differing degrees of work and tasks as we begin constructing our design. There will be both hardware and software components that will need to be developed for us to be successful in our design. We plan on having the electrical engineers in our group work on the hardware while the computer engineers work on the software. Of course, there will be some overlap as we all would like to know about the various parts of the system. As for managing the workload we plan to use a Gantt chart for mapping out where we should be in our project over time. This will keep us on track and help us to know where we should be and what we should be working. We will be using a task board like Trello to help us keep track of tasks that need to be done and priorities for each task. By getting higher priority tasks which are crucial to our design done first we can add in the things we absolutely need for our project. This may even allow us room to add in stretch goals and other things we wanted to add if given a bit more time. Workflow management will be crucial, especially as deadlines begin to get closer. The better we are at managing the work we need to do to complete our task the easier it will be to meet the expectations for each deadline. For example, our Senior Design 1 paper will be made easier by dividing the number of pages we should have by our members and then us writing pages over the course of the entire semester.

4.2.4 Environmental constraints

Environmental constraints refer to a way which the product will interact with and change the environment around it. Since this lock will be fastened to a home's door, it will not take up lots of space. It will also be relatively quiet and won't make enough sound to contribute to sound pollution. The main source of negative environmental impact will come from the battery power source. Batteries need to be carefully disposed of, or they can decay and leak acids into landfills. This acid can eventually find its way to water sources and contaminate them. Solutions to this battery waste problem will be discussed in the next section, Energy Constraints.

4.2.5 Energy constraints

As previously mentioned, incorporating batteries into our design has the potential to create negative environmental effects. To solve this problem, we will include rechargeable batteries that can easily be swapped out with other batteries once they run out of power. The next energy constraint that was identified is extending the battery life as much as we can to maximize the time between needing to change out the battery pack. This can be done by incorporating low power mode functions into our microcontrollers. Low power mode is like putting the device to sleep, and waking it up with a certain trigger action, like pressing a button. Since the lock only needs to be used a few times a day, it can be in low power mode for the remainder of the time it is not in use. This should significantly increase the time each battery pack can be in use.

4.2.6 Social constraints

Social constraints refer to how our product will impact customers and society. Things to consider are cultural norms and customs in the areas we will be using our product. A specific social constraint for Live Bolt comes from its face recognition capabilities. Some people may be uncomfortable with the use of face recognition technologies due to their possible malicious uses around the world. The Live Bolt phone application will allow the ability to toggle face recognition on or off to remedy this situation. Live Bolt also come with the social constraint of trusting smart systems. With an evolving technology era, more homes are becoming smart but, some customers are traditional. Although this won't reach older audiences, we do have to recognize the coming generation prepared for smart homes. Live Bolt will be a representative for evolving smart homes through security and access control.

4.2.7 Political constraints

Political constraints refer to our product's impact on any governmental bodies. This product should have no impact, positive nor negative, on any government body, nation, or group, and therefore has no political constraints associated with it.

4.2.8 Ethical constraints

Maintaining proper ethical standards is just as important to being an engineer as technical knowledge is. A great standard to follow is the code of ethics developed by the Institute of Electrical and Electronics Engineers. The codes are as follow: We, the members of the IEEE, in recognition of the importance of our technologies in affecting the quality of life throughout the world, and in accepting a personal obligation to our profession, its members and the communities we serve, do hereby commit ourselves to the highest ethical and professional conduct and agree:

I. To uphold the highest standards of integrity, responsible behavior, and ethical conduct in professional activities.
1. to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others, and to disclose promptly factors that might endanger the public or the environment.
2. to improve the understanding by individuals and society of the capabilities and societal implications of conventional and emerging technologies, including intelligent systems;
3. to avoid real or perceived conflicts of interest whenever possible, and to disclose them to affected parties when they do exist;
4. to avoid unlawful conduct in professional activities, and to reject bribery in all its forms;
5. to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, to be honest and realistic in stating claims or estimates based on available data, and to credit properly the contributions of others;
6. to maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations;
II. To treat all persons fairly and with respect, to not engage in harassment or discrimination, and to avoid injuring others.
7. to treat all persons fairly and with respect, and to not engage in discrimination based on characteristics such as race, religion, gender, disability, age, national origin, sexual orientation, gender identity, or gender expression;
8. to not engage in harassment of any kind, including sexual harassment or bullying behavior;
9. to avoid injuring others, their property, reputation, or employment by false or malicious actions, rumors or any other verbal or physical abuses;

III. To strive to ensure this code is upheld by colleagues and co-workers.
10. to support colleagues and co-workers in following this code of ethics, to strive to ensure the code is upheld, and to not retaliate against individuals reporting a violation. [10]

The group will strive to follow these guidelines to the best of our ability and integrity. This code of ethics will remain in the back of our minds during each major stage of the project. Since the Live Bolt is a door lock, we need to ensure the safety of the customer and their belonging. Specific safety constraints will be discussed in the next section.

4.2.9 Health and Safety constraints

Health and Safety Constraints refer to the consideration of the health and safety of not only the customer but also the engineers developing the product. Constraints identified for customers are ensuring that only the customer and people they allow, such as friends or delivery workers, can unlock the lock. Under no other circumstances should the door be able to unlock. This is important because if unauthorized users can enter someone's home, the customers' belongings and well-being could be in danger. A safety constraint identified for both the engineers and the customer is related to the battery pack. If exposed to extreme climates, like heat or weather, the batteries and other electrical components may malfunction and cause unwanted issues like leakage or fires. The engineers will keep this in mind while designing and building the product, and test to make sure the product is safe for consumer use.

4.2.10 Sustainability constraints

Sustainability constraints refer to the longevity and robustness of the product. Since a portion of the product (the camera, IR sensor, microphone, and keypad) will be placed on the outside of the door, the product will need to be able to withstand most environmental factors. Since the engineers are based in Florida, there are specific requirements that need to be considered for the casing of the product. It will need to be waterproof to withstand rain and humidity in the air and have proper airflow to prevent overheating components. A solution to these constraints is to make the housing out of some waterproof material, such as plastic or corrosion resistant metal.

4.2.11 Manufacturability constraints

Manufacturability constraints refer to issues that may arise while building the product. Our design relies on a cover placed around the thumb knob to turn it. A major issue identified with this project is that most locks do not have the same shape of thumb knob. We will work to design a universal knob cover that is able

to be placed on any knob. If we are unable to achieve this, we can also include multiple covers for different lock shapes. This constraint plays into a bigger constraint we have identified, being that the product should be easy to install for the consumer. We chose the knob cover so that the customer will not have to unscrew and dismantle portions of their lock, which is seen in most commercial smart locks. Our team would also like to make the product as small and compact as possible. We do not want it to look too bulky and intrusive on consumers' doors. This will be achieved by designing a compact PCB and housing for all the sensors and other components.

Due to the design choice of not unscrewing the door at all during installation, it will be difficult to connect the inside module and outside modules since we cannot pass wires through the hole that the lock is normally set in. We will need to create a solution to this problem, most likely through the use of wireless communications like Wi-Fi or Bluetooth so that each module can communicate with each other and the home network.

Another manufacturability constraint is access to lab equipment to design and create the product. A portion of our design will likely be 3D printed, and this will cost both time and money. There are many services online and locally that provide access to 3D printers. These services will be expanded on in a later section.

5. Initial Design Architectures and Related Diagrams

Inspiration of creating the design for the Live Bolt lock is by having a 3D printed doorknob cover the shape of the lock. The covering will fit over the deadbolt turn knob and house the servo motor that’ll create the turning gesture. This eliminates the need to unscrew the doorknob and make modifications to it. Installation at this of the lock will be as simple as adding two-sided tape to the door. **Figure 9** below shows a preliminary concept of the design on a door. **Figure 10** shows a close-up view of the cover.

As of the initial design for the UI home page, we plan to have the initial mockup for **Figure 11** to inspire the drafting for the whole application. The Live Bolt application design will take influence from existing home access/ monitoring applications to formulate simplicity to the user.

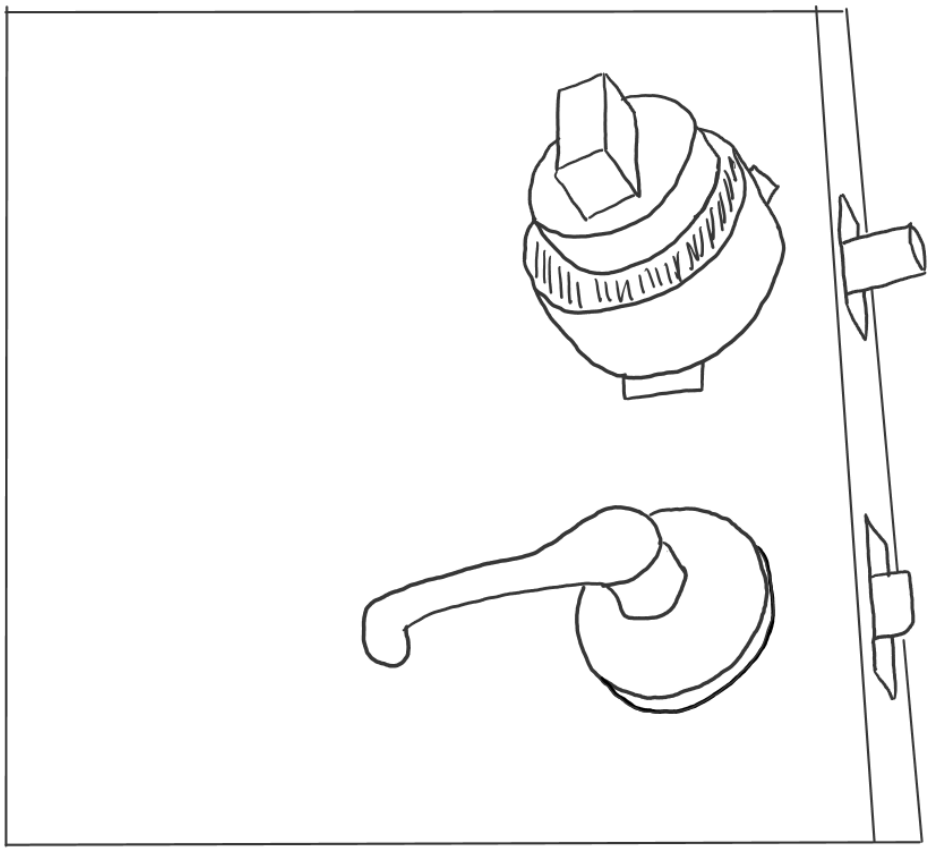

Figure 9. Deadbolt doorknob cover, fitted onto a door lock

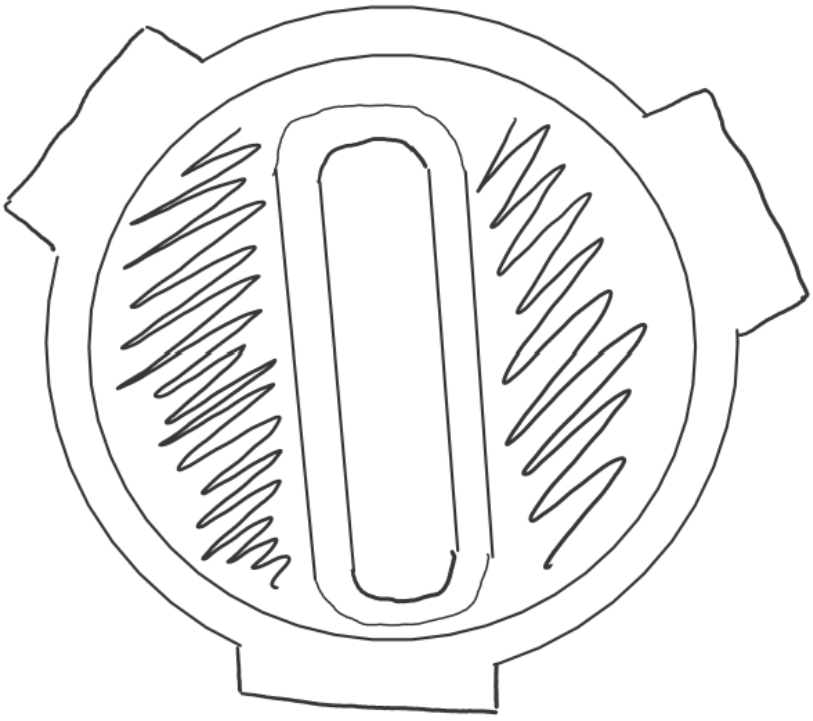

Figure 10. Frontal view of the Doorknob Cover

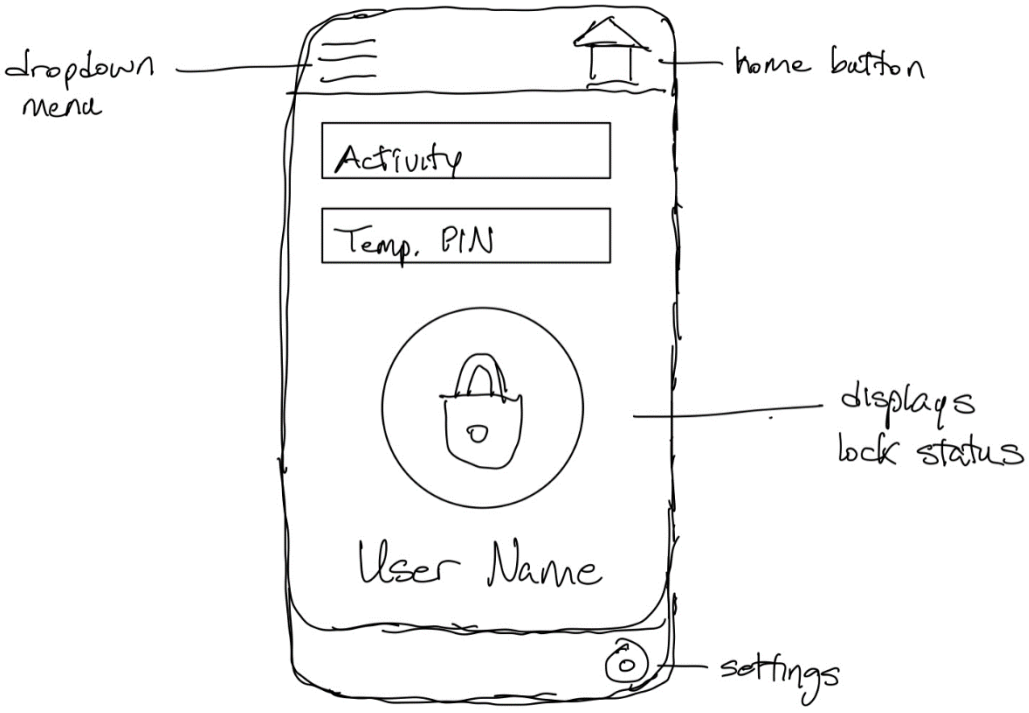

Figure 11. Initial Sketch of Application Main Interface

6 Software Design Details

6.1 Initial Design Architecture

The initial design for the logic of the mobile application is displayed below in Figure 12. When the user first opens the app, they are asked to either log in with their account credentials or create a new account. To create an account, a valid email address is required as well as providing their phone number for alerts. A confirmation email is sent to their account after creating the account, to ensure that the account information is secure. Once they successfully login, the user is sent to the home page and can use the app.

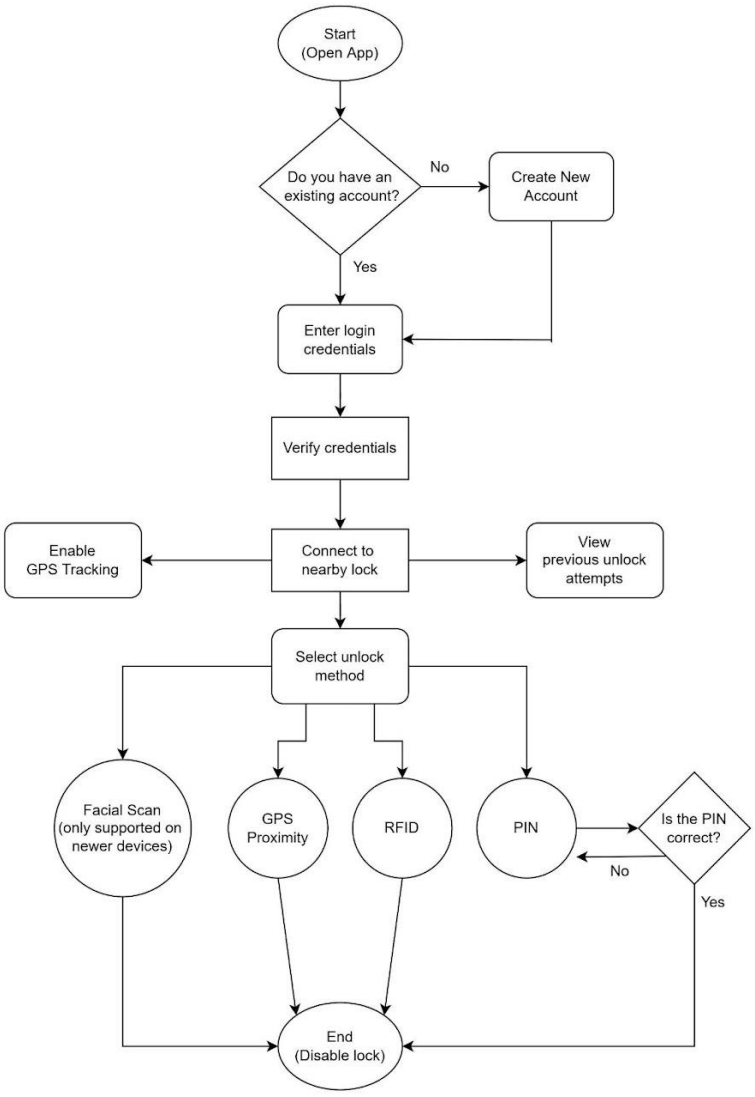

Figure 12: Lock/Unlock Flowchart

The primary function of this application is the ability to disable the lock via RFID or PIN number. Additionally, the user will be able to access a log of previous lock and unlock attempts by logging in to their account. The application will be available on both Android and iOS, with the iOS version utilizing the facial recognition hardware present in newer iPhones for easier access control. Another planned feature is the ability to enable GPS proximity unlocking which will track the location of the user's phone and disable the lock when it is determined that the user is outside the door. This feature is heavily inspired by a similar function of the previously mentioned August Smart Lock, which uses both Bluetooth and GPS technologies to track the user's device when leaving the house. To protect the user's privacy, our application will only use this feature once the user enables it.

6.2 Mobile Application Pros and Cons

While the decision to include a companion app to Live Bolt was a simple one, there are many kinds of mobile application frameworks to use, each with different combinations of various technologies. This section will give some information on some commonly used full stack applications, as well as any advantages and disadvantages that would come from using those technologies for our application.

6.2.1 LAMP Stack

A LAMP stack is a full stack application comprised of a Linux operating system, an Apache web server, a MySQL database, and can use either Python or PHP as its programming language. Some team members have had experience creating a LAMP stack, so the learning curve would not be as high as opposed to using a more complex stack type. Python as a programming language is very simple to use and has no shortage of support libraries to use. However, Python completely lacks any sort of native mobile framework to use for our application. If we were to use a LAMP stack, we would need to use a framework that has cross-platform functionality such as Kivy or BeeWare. Another problem is the difficulty that comes from using two programming languages when creating the application, Python/PHP for the server-side and JavaScript for the client-side. Switching between two languages can add some unexpected challenges for us on the software end that could be resolved by using a different type of stack.

6.2.2 MERN Stack

On the other hand, a MERN stack uses Node.js as its runtime environment and Express.js as its server-side framework. React is used as the client-side framework with a MongoDB database. Since MongoDB is a NoSQL database, it does not use schemas, but instead stores entries as binary JSON objects. This

gives us an additional measure of flexibility when creating our database of users. Using React as the JavaScript library has its benefits as well, with the React Native framework being well suited for mobile development. Since Node.js uses JavaScript for the backend, our team only needs to use a single programming language for our application. This will reduce the time needed to learn any new technologies since anyone with a solid grasp of JavaScript will be able to contribute to the development process. This also applies to Express since it is simply a web framework built off Node.js.

6.2.3 MEAN Stack

A MEAN stack is very similar to the previously mentioned MERN stack but uses AngularJS as its framework instead of React. AngularJS differs from React by being an actual MVC framework, opposed to React simply being a JavaScript library. Instead of using components like React, Angular uses directives to teach the browser new syntax. Angular is also continuously maintained by Google, ensuring that there are many resources at our disposal if we plan to use this framework. Angular also uses HTML in its development process, which some team members already have experience working with. However, this makes the learning curve of Angular much steeper when compared to React, which was designed to only use JavaScript.

6.3 UX Design

When designing our user experience, our team's philosophy was focused on readability and consistency. Our aim was to create an app that felt like a natural extension of the capabilities of the physical Live Bolt. We initially considered creating a web application instead of a published mobile app in order to save time on development. However, there are many issues to consider when creating a web app meant to be accessed by phones. Since many phones have unique screen sizes, the native web app would have to adjust to every screen size in order to provide consistency. This would be difficult to achieve without using a CSS library that automatically scales UI elements, such as Bootstrap. Even so, our team determined that using a native web app was not worth the tradeoff in customizability. There are many smart lock companion apps on both the Google Play Store and the App Store, so we desired to create an experience that would be immediately recognizable and unique for the consumer.

Even so, original ideas are exceedingly rare to come by. During the design process, our team studied apps of various functions, on both the App Store and Google Play Store. Most, if not all, apps use the same core principles and tools when it comes to the user experience. Above all, apps that are considered to have good UX design are created with the user first and foremost. The user must be able to navigate the app without being confused or overwhelmed by its layout and any interactive elements must be clearly defined such that the user

immediately knows that they can interact with them. Adhering to such accessibility principles ensures that users are not excluded from using our app for any reason.

As mentioned in Section 3.1, many smart locks nowadays come with their own companion apps. For example, products such as the August Smart Lock and Nuki Smart Lock have unique mobile apps. These apps usually allow the user to perform the same function of enabling and disabling the lock remotely. Additionally, most apps track the activity of the physical device as a viewable log. Since our team intends to include similar functionality in our mobile application, we have taken inspiration from these kinds of apps to create our own unique user experience.

6.4 Preliminary UI Design

At the beginning of development, our team used Figma to create rough designs for several of the pages for our app. Figma is a web-based graphical tool that can be used to create both app designs and prototypes. Using the free starter level functions, our team was able to come up with a variety of potential user interfaces. Additionally, Figma also has a companion app on iOS and Android that allows users to see how their designs look on phone screens. Some of the initial designs for the login and account creation pages are displayed below in Figure 13 to Figure 18.

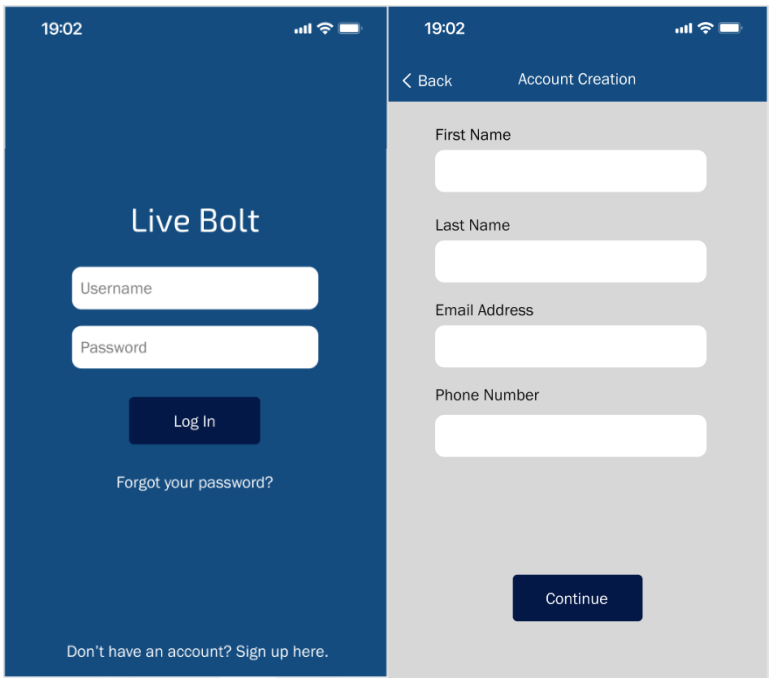

Figure 13: Sample Login/Register Page 1

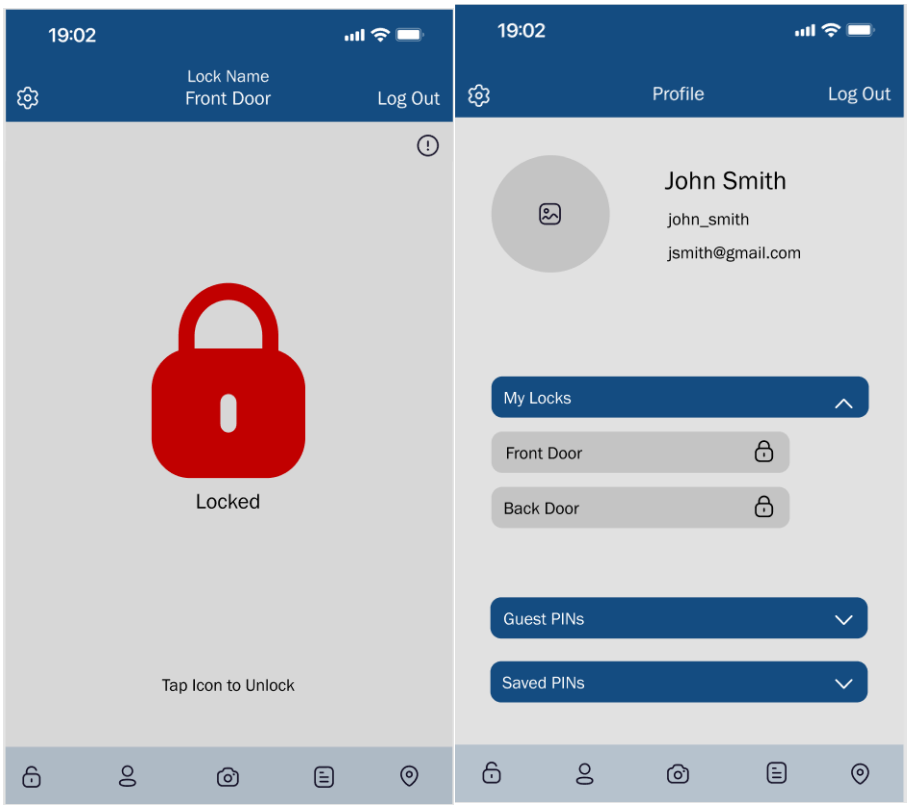

Figure 14: Sample Home/Profile Page 1

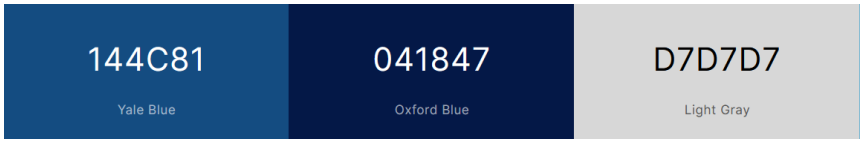

Figure 15: Color Palette 1

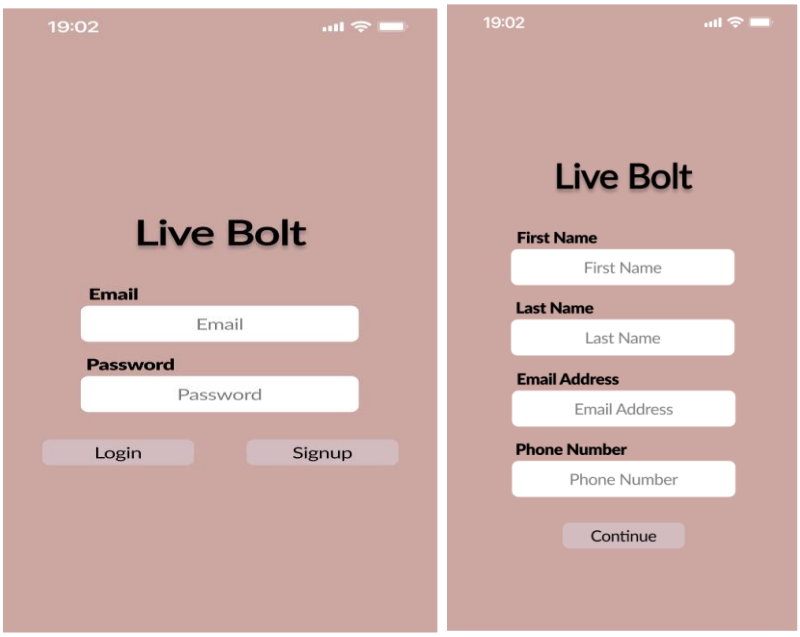

Figure 16: Sample Login/Register Page 2

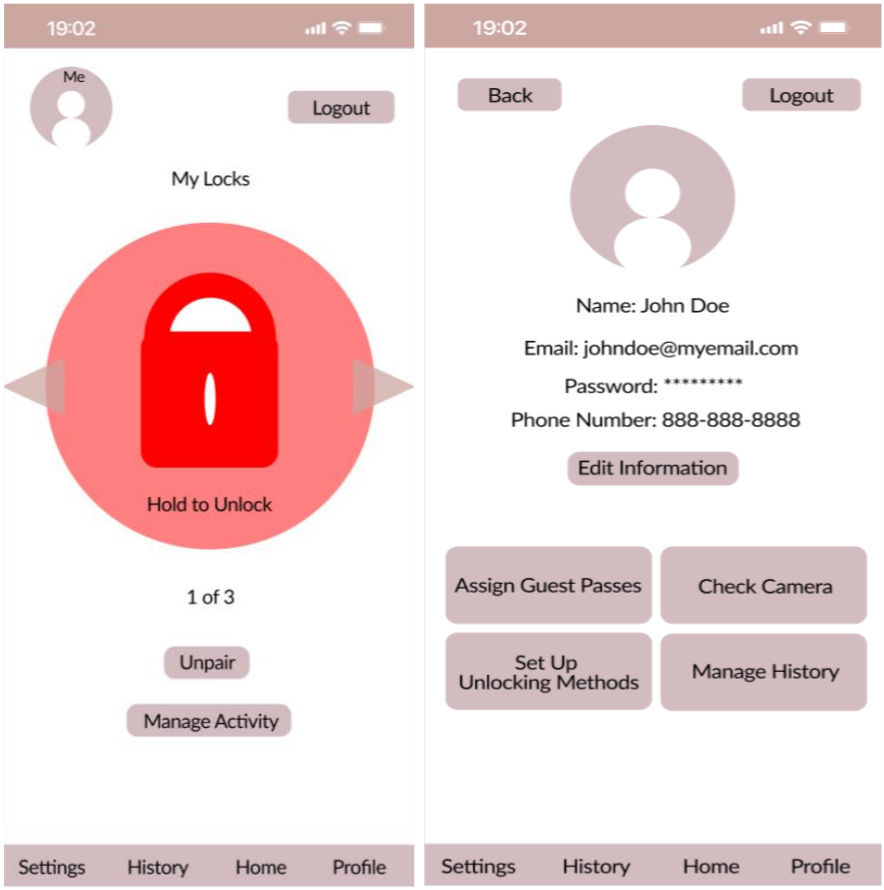

Figure 17: Sample Home/Profile Page 2

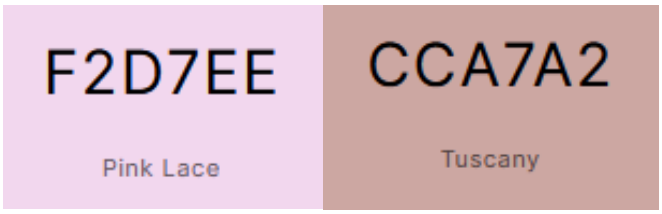

Figure 18: Color Palette 2

It is important to note that this version of the login page does not include the Live Bolt logo and instead has a placeholder, since the logo was not finalized at the time of creation. The color palette we decided to use for our app was meant to be visually pleasing, yet distinctive enough to make an impression on our users. Blue is a very non-aggressive color for many people and is well rendered on most mobile devices, even on older devices with limited color capabilities. For text rendering, gray works quite well due to its contrast with darker colors without being overly bright and unreadable. An alternative color scheme was also created before we decided on our main color scheme, which is also shown above.

The first screen a user is taken to depends on whether they are a new or existing user, also modeled in the use case diagram shown earlier. An existing user will be taken to the lock screen, where they can then access any nearby Live Bolt devices. The screen will then show the status of the lock, with a green icon designating a deactivated lock and a red icon designating an activated lock.

The profile screen is where the user is taken after login if they are on a freshly made account. The screen displays the user's contact information, along with an optional profile picture. Here, the user will be asked to enter any additional information that will be needed along with a short tutorial showing the functions of the Live Bolt app. As the user continues using the app, the profile will store more information, including used Live Bolt locks, saved PINs, and guest PINs currently being used. The version of the profile screen shown also displays the status of nearby locks for added convenience.

6.4.1 Application Navigation

When designing our application, we wanted to make our application as intuitive as possible. We took much inspiration from other Smart Lock systems currently available on the market as discussed in **Section 3.1**. We decided to create flowcharts to demonstrate the intended functionality of each page of our application to clarify what our goals were. The flowcharts were created using diagrams.net, a free tool for creating diagrams and charts. Figure 19 and Figure 20 show the separated functionality of the Login and Signup functions. Figure 21

shows the functionality when loading up the application to reaching the home screen.

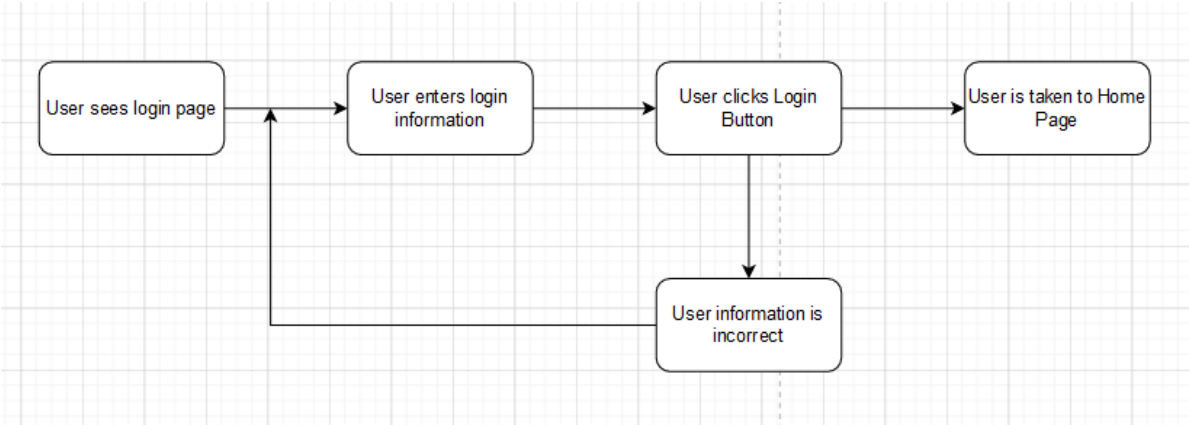

Figure 19: Login Navigation Flowchart

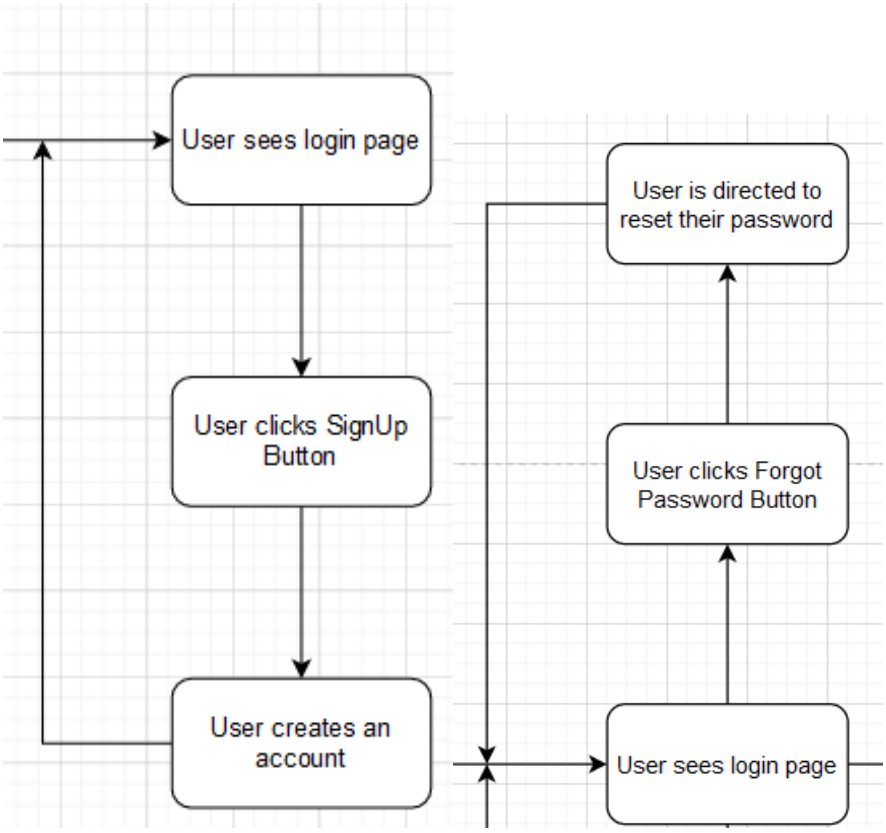

Figure 20: Signup Navigation Flowchart

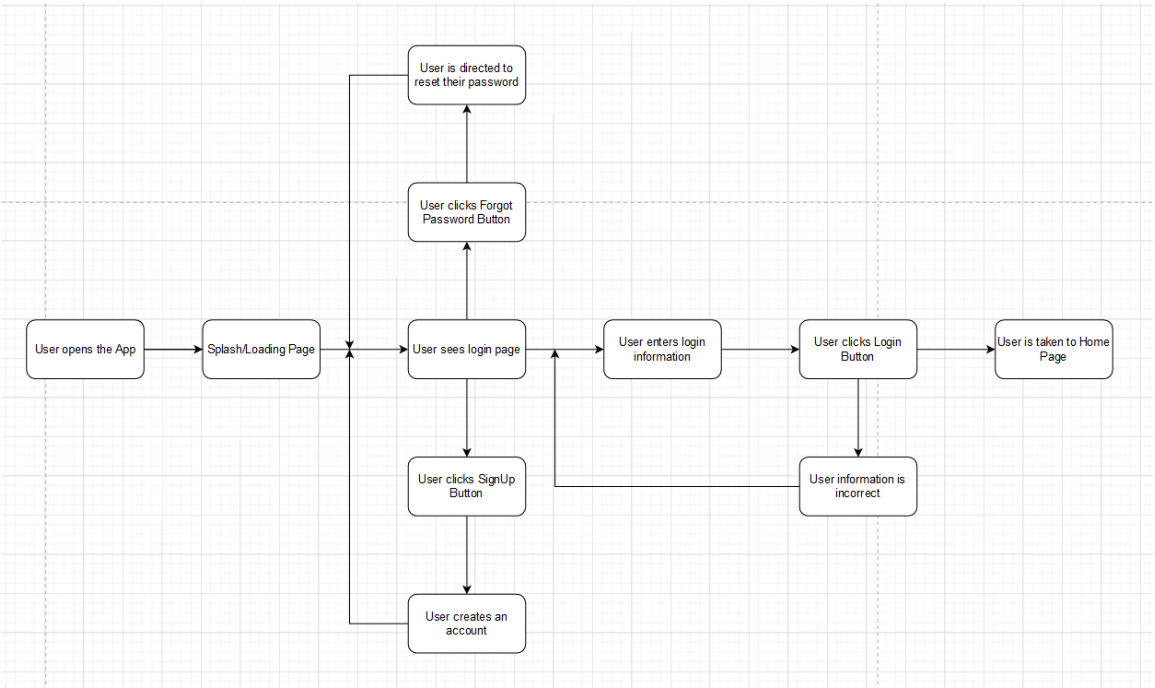

Figure 21: Login and Signup Navigation Flowchart

6.4.2 Use Case Diagram

A Use Case Diagram is used to describe and model the way actors interact with your system. These actors can be anything from users to administrators, really just anyone who uses the system. It is part of the Unified Modeling Language (UML). It is used frequently in software development to describe the way a system should function. It also helps to clarify specific situations that can occur as an actor works through the system. Another benefit is that it keeps the team visualize what the application should do. As an offshoot of this it clarifies how the system should interact with its various parts and what behaviors should occur when specific actions are taken. As an abstraction of our systems design this diagram gives the bigger picture behind what we intend to achieve with our Live Bolt system. Our Use Case Diagram was created using diagrams.net.

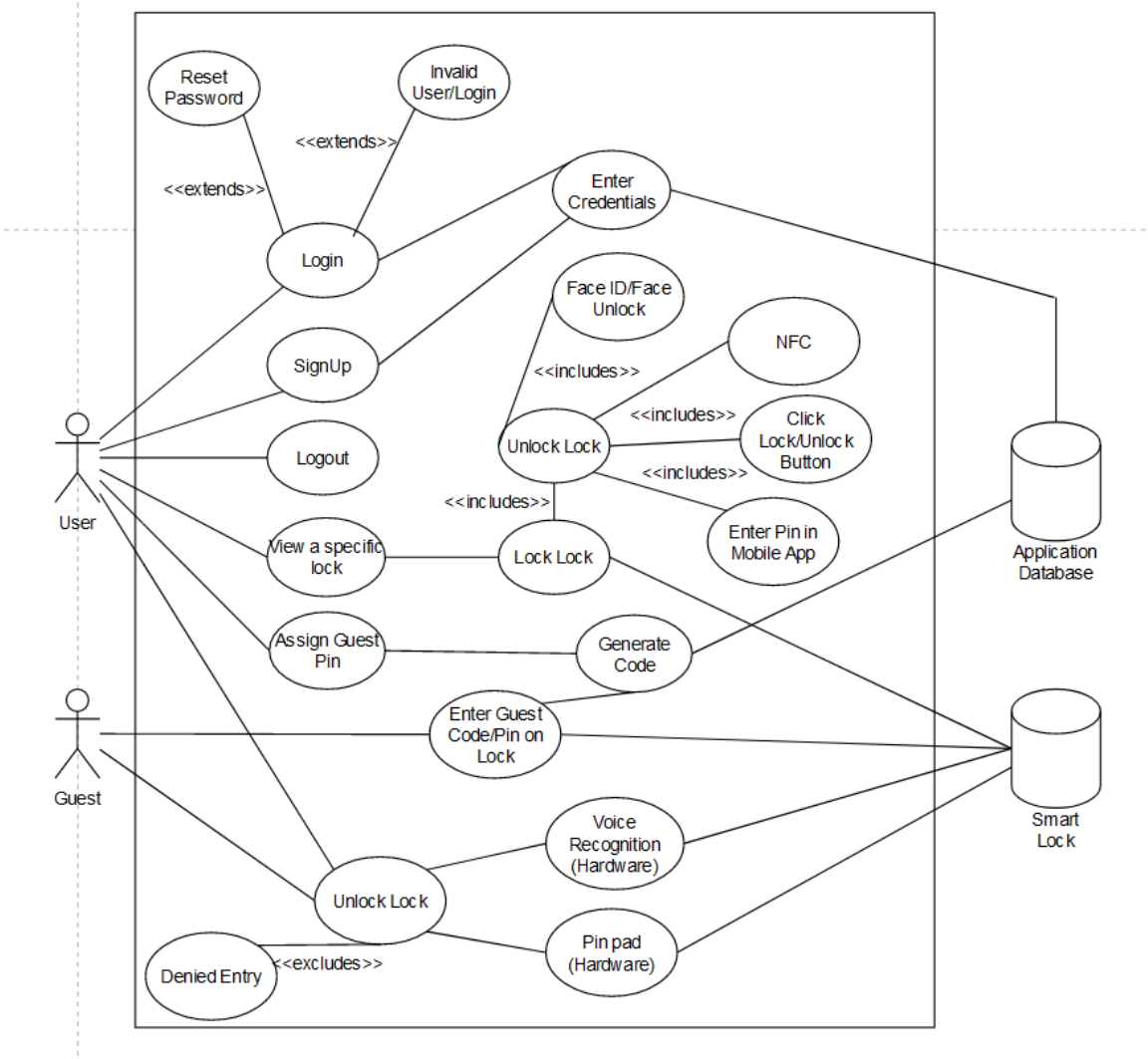

Figure 22: Use Case Diagram of Application including both Hardware/Software Unlocking Mechanisms

6.5 Logo Design

When creating a logo for any sort of product, the aim is to produce a unique design that reflects the brand or product that is being sold to consumers. Our team sought to create a logo that was simple and appealing in design, while accurately conveying the function of the Live Bolt Smart Lock to consumers. This section will contain several preliminary concepts that were created before our team decided on a final selection. Most of these logos were created using the Adobe Express tool with some slight adjustments made in GIMP.

Insert potential logos here

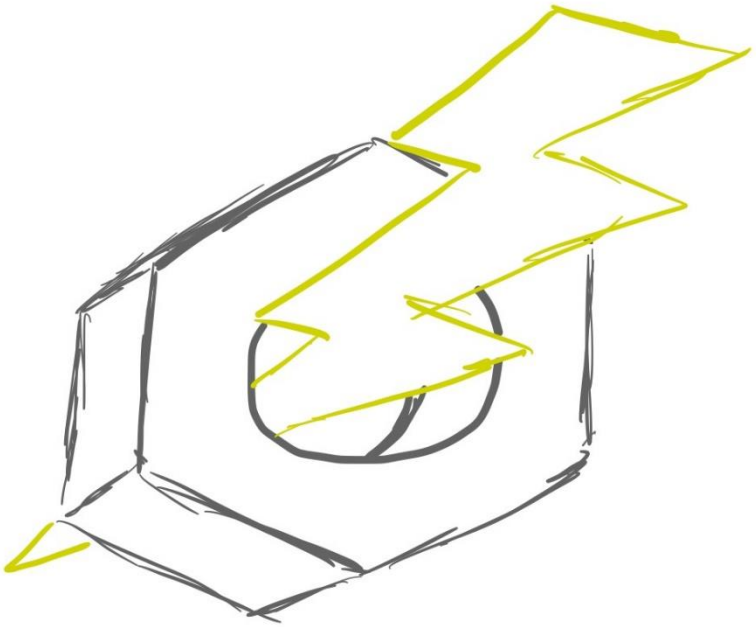

Figure 23: Sample Application Logo

6.6 Bluetooth Pairing

For a user to connect to a physical Live Bolt lock using the companion app, the app will need to be able to use Bluetooth pairing. According to the Android Developers website, there are three steps when a mobile application seeks to make a Bluetooth connection [2].

I. Find nearby Bluetooth devices.
II. Connect to a particular Bluetooth device.
III. Transfer data with the connected device.

While the website talks about Bluetooth functionality in an Android environment, these steps are general enough to also apply to web and iOS applications. Since data will be transferred between the physical lock and the user's phone, the connection must be secure to maintain the device's integrity and security. Additionally, the GPS proximity feature will use Bluetooth in tandem with the phone's physical location to automatically disable the lock when the user comes within a set distance. In order to perform these functions, our app will need to use an available plugin or API to get Bluetooth functionality. Since this app is meant to be cross-platform, there are several options that can be used. This section will

highlight APIs and plugins that provide Bluetooth functionality, as well as discussing any design choices that were made concerning this subject.

6.6.1 Android Bluetooth APIs

The Android development platform provides several APIs for developers to use when creating applications for the Google Play Store. The development process is further streamlined by having all Bluetooth APIs available in the *android.bluetooth* package. The Android Developers website includes all the key classes and interfaces needed to implement Bluetooth functionality in any mobile application [11].

6.6.2 react-native-bluetooth-serial Plugin

The React Native framework does not include any native libraries or components for Bluetooth connectivity. Instead, there are a wide variety of 3rd party components that are available to use for mobile developers. Most of the popular components are plugins that have been ported over to React Native, such as the *react-native-bluetooth-serial* plugin. This component allows React Native applications running on any version past 0.25 to connect to other Bluetooth devices and transfer data. This plugin works for both iOS and Android, however as of the time of writing this paper, services are still not configurable for iOS devices. The GitHub page provides the installation procedure for both iOS and Android, as well as all the methods available for use [12].

6.6.2 react-native-ble-plx Plugin

Another popular React Native component to use for Bluetooth communication is the *react-native-ble-plx* plugin. The main difference between this component and the previously mentioned one is compatibility with Bluetooth Low Energy (BLE) devices. The Github page details the functionality of this plugin in addition to any events that can be used [6]. Since this is the plugin that our team decided to use for our companion app, we also created a UML class diagram to easily communicate the various classes and methods of the plugin.

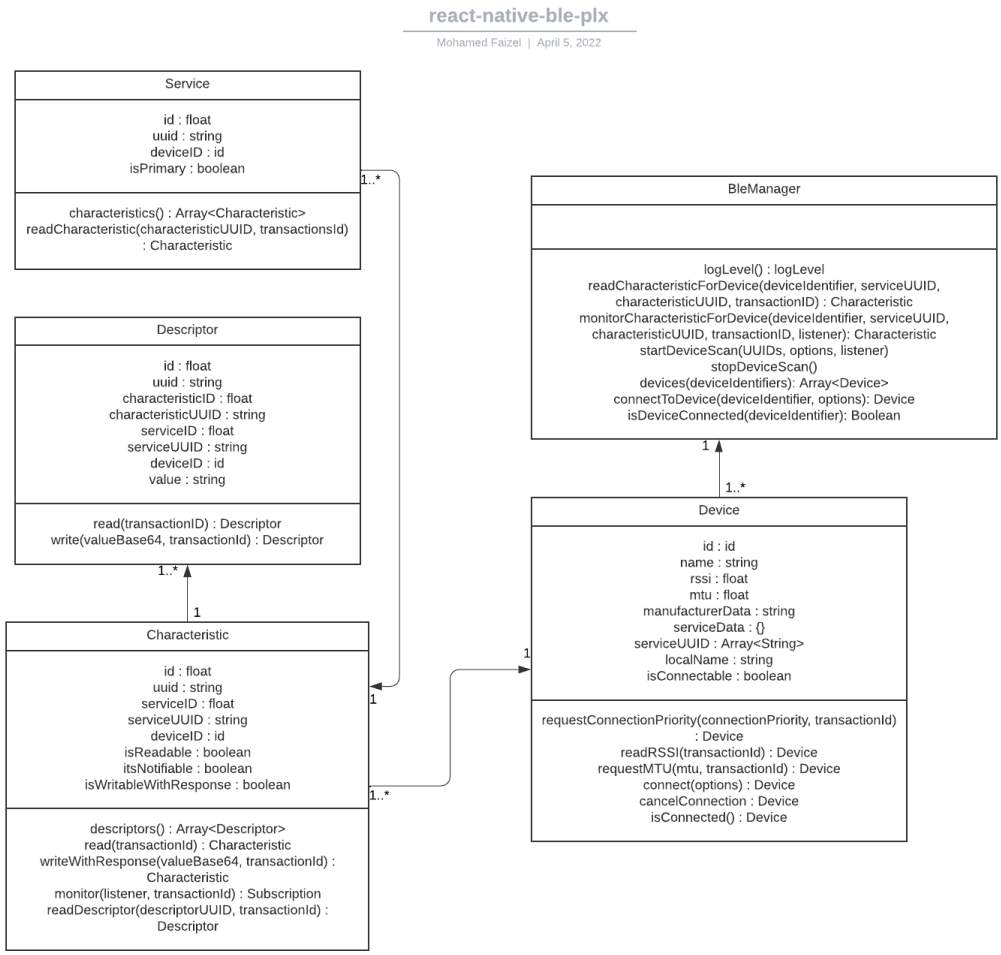

Figure 24: API for react-native-ble-plx

Figure 24 displays the five major classes of the API: Services, Descriptors, Characteristics, Devices, and the BLE Manager. The first section of each class contains all the relevant attributes it contains, while the second section displays the methods belonging to that class. Since BLE Manager is always the first class that is instantiated whenever a device needs to be connected, it contains no attributes of its own. A Device class contains several Characteristic classes, which are comprised of Descriptor and Service classes. It is also important to note that the BLE Manager or Device class is usually the one to call methods and that most methods listed in the other classes are only partial versions of methods available in Device or BLE Manager classes. However, these methods are still included in this UML class diagram to streamline the structure of the diagram.

6.6.3 Classic or Low Power Mode?

As mentioned earlier in Section 4.1.2, there are two standards of Bluetooth currently in use: Classic and Low Power. Since the physical lock will use the Raspberry Pi 4 to communicate with the user's phone, the API or component used for Bluetooth functionality needs to be usable with the Pi 4. Fortunately, the Pi 4 is compatible with both Classic and BLE devices.

6.7 Development Environment

When creating a program of any sort, a development environment is needed for programmers to edit and debug their source code in an easy-to-use space. While it is possible to edit code straight from the command line or using a simple text editor, using a development environment allows for us to fix syntax errors and track variables. Nowadays, many integrated development environments (IDEs) also include compliers to debug code.

Due to its simplicity and abundance of native tools for React development, we decided to use the Expo CLI development environment and the Virtual Studio Code IDE for editing source code. Expo CLI is a command line application that can be installed on any terminal of choice and serves as an interface between the developer and the React app. On the other hand, Visual Studio Code (VScode) is a text editor created and supported by Microsoft that is available on Windows, MacOS, and Linux. VScode includes support for both JavaScript and Node.js, making it a very desirable choice for our team. The installation instructions and documentation of both the Expo CLI environment and the VScode IDE are provided in the references section below [13].

6.8 App Store Publishing Process

As with any online platform or marketplace, the Apple App Store has its own unique guidelines and recommendations for developers who seek to publish their apps. Most of these standards are described above in Section 4.1.8 and will be referenced throughout this section. In addition to adhering to proper programming standards, our app must also adhere to the various design, compatibility, and legal standards detailed in the App Store Review Guidelines [6]. After we have determined that our app is fit to be reviewed, we will submit our app along with any metadata via App Store Connect. According to the Apple Developer website, 50% of submitted apps are reviewed in under 24 hours and over 90% are reviewed after 48 hours.

To avoid potential issues with submitting our app for review, we will continuously test and debug our app and ensure that it runs well on all available platforms. Additionally, we will adhere to the App Store's privacy guidelines by always asking our users for permission to use their data in a clear and understandable

manner. To avoid a rushed schedule, we will submit our app well ahead of our planned release date to account for any unexpected issues that may arise during the review process.

6.9 Google Play Store Publishing Process

The publishing process for the Google Play Store differs from the App Store process in several ways. The platform used for publishing to the Play Store is Google Play Console, which is used for development, review, and publishing. In order to start the review process, we must first publish a draft version of the application. To proceed to the review stage, the app must not have any detected errors. Just like the App Store process, our app will be rigorously tested to ensure that our users will not encounter bugs or crashes.

Additionally, there are two types of publishing our app can undergo: standard publishing and managed publishing. Standard publishing allows any updates or changes to all existing apps, with a waiting period of up to 7 days for review. Managed publishing lets the developer control where any changes are published after their app has been successfully cleared for release, allowing for greater version control over the standard method. The Google Play Console Help page also details every possible status for an app or update [7]. Since we are planning for a cross-platform release, we would like our app to be reviewed and given the status for production in order to ensure that both releases occur in the same timeframe.

6.10 Mobile Stack Technologies

After researching and analyzing various stack types, those of which were mentioned in Section 6.2, our team eventually decided what stack application our mobile app would be. This section details the reasons behind our decision, as well as any technologies being used that have not been mentioned so far.

6.10.1 MERN Stack Reasoning

Ultimately, we chose to create our application using a MERN stack for a variety of reasons, many of which are described above in Section 5.6.2. As mentioned earlier, a MERN stack comprises of a MongoDB database, a Node web server, an Express web framework, and a React client-side. Since most of our team has a measure of familiarity with developing a MERN stack in previous classes, namely in Processes of Object-Oriented Development, there will not be as much learning required to create our app using a MERN stack.

6.10.2 JavaScript

JavaScript, along with HTML and CSS, are the main building blocks for any kind of web application. It is a high-level language, along with languages such as C#, Ruby, Python, and Java. It is important to maintain that Java and JavaScript are separate languages, though they share similarities in syntax. JavaScript is also incredibly popular, with over 97% of websites using the language client-side for web page behavior [8]. This code usually executes client-side to load new pages, generate user results, and control web page animations. The amount of JavaScript libraries is incredibly large and several of them control large parts of the web development process. Many of them will be used in our app which are further explained below.

6.10.3 Ajax

While Ajax is not a library, it is an important technology that is often used in web development. Short for Asynchronous JavaScript and XML, it is the method the XMLHttpRequest object uses to communicate with the web server. Most important about Ajax is its ability to operate asynchronously, meaning that it can exchange data and update a web page without needing to reload the page. Additionally, the data that is sent and received can be in several formats, including JSON, XML, and even text files [9]. The Mozilla Developers website contains the documentation of Ajax, along with an introductory article through which most of this data was taken from.

6.10.4 jQuery

jQuery is a cross-platform JavaScript library that simplifies Document Object Model (DOM) traversal and manipulation. A DOM is an interface that converts an XML or HTML document into a tree structure, allowing a web page to be dynamically generated via JavaScript code. jQuery also boasts Ajax functionality which is used by the XML object to effectively communicate with the web server. It is important to note that there is a version of jQuery made to work with mobile devices, called jQuery Mobile. Unlike the standard jQuery API, jQuery Mobile is a complete UI framework, which means it needs to be built upon as a starting point. This conflicts with our preference towards using React Native as a framework, so this prevents us from fully using the jQuery Mobile API. For this reason, we will only be using version 3.6.0 jQuery API for our mobile application.

6.10.5 JSON

JavaScript Object Notation (JSON) is a data-interchange format based on the conventions of the JavaScript programming language. While it does use much of the syntax used in JavaScript, JSON is completely language-independent and

can be used with any programming language, though it commonly sees uses in high-level object-oriented languages. JSON usually is comprised of two kinds of data structures: a collection of name/value pairs and an ordered list of values. In most programming languages, these are both realized as arrays or lists. Due to its simple and effective structure, JSON is used for all sorts of data transfer operations in web development.

6.10.6 JSX

JSX is simply a syntax extension of JavaScript that is commonly used with React. Though it contains elements of HTML syntax, JSX still has all the benefits of JavaScript without sacrificing the benefits of a template language such as HTML or CSS. The main function of JSX is to produce React elements, which are the main building blocks of any React application. JSX uses rendering in order to implement React elements into browser DOMs, which has several advantages over the traditional method of DOM traversal and manipulation. Unlike browser DOM objects, React elements are plain objects and much cheaper to make. These developer-created elements work together to form React components, which can be described as the React equivalent of JavaScript functions. Like a function, these components accept certain properties (or props) as parameters and generate a React element to be used in the user interface. These components can be used in other React applications in the same way a JavaScript function can be used in a different program.

6.10.5 JSON Web Token

With our mobile application having several technologies communicating with each other, it is important to also maintain data integrity during the development process. JSON Web Token (JWT) is an open standard that defines a compact way for securely transferring information between entities as a JSON object [reference needed]. There are many reasons to use JWT as a method of securing data, such as verifying the identity of a sender or recipient of data and authorizing specific users to use an application.

The structure of a JWT consists of three parts: a header, payload, and signature. The header contains the token type and algorithm, which in most cases would be JWT and a standard signing algorithm such as HMAC SHA256 respectively. The payload usually contains information concerning the user and the type of claim being made. Lastly, the signature ensures that none of the contents of the token were altered in any way. The result after encoding is three URL strings separated by dots, a format that is easily passed through in HTML and HTTP settings.

6.11 Database Structure

The database our application will use will need to contain several types of user data including logins, passwords, user activity, emails. It must be structured in a manner that our app will be able to efficiently search for any information it needs at a moment's notice. A well-designed database typically has the following traits [18]:

- Efficient access to data
- Supports long-term maintenance of data integrity
- Clean and consistent structure
- Avoids redundant storage
- Accurately reflects the structure of the problem

To create a database with these principles in mind, our team first created a preliminary entity relationship diagram of the database. An entity relationship diagram (ERD) is used to map out the structure and relationships present in the database. Outlining an ERD before designing the actual database comes with many benefits to the overall creation process. For example, making changes with an ERD is far easier than having to modify the database's data directly, especially since adjusting data can be very risky as the scale of the database increases. Additionally, the creation of a visual aid helps with keeping track of all the needed data in one easy-to-read diagram, which makes it useful for gathering requirements and specifications. The ERD modeling of our database is shown below, complete with variable types and entity relationships.

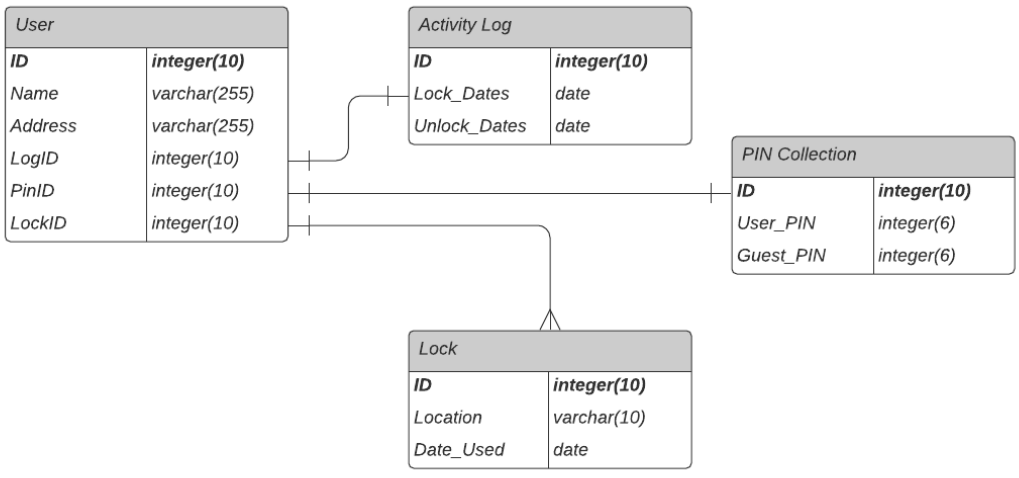

Figure 25: Entity Relationship Diagram

As shown above, the database for our companion app consists of four entities: users, activity logs, PIN collections, and locks. A user entity contains relevant information such as their name and address, as well as a unique ID. The user entity also has three foreign keys which reference an activity log, a PIN collection, and a Lock. It is important to note that all attributes in bold are primary keys, all of which are IDs in the case of this database. The data type is also listed next to its corresponding attribute, mostly comprising of integers and character strings. Of course, this diagram was made early in the project so changes may need to be made as development progresses.

6.11.1 MongoDB Creation and Syntax

Before creating a MongoDB database, a cluster must first be established and granted access to through the MongoDB shell. Once that is done by the user, the database can store new objects by inserting them through the Command Line Interface (CLI). For those who wish to use a more accessible interface, the MongoDB Compass is a GUI which can perform the same functions as the CLI. While our team was comfortable with using the CLI as our main interface, we decided to use Compass in order to more intuitively manage our database, as it is more complex than any other our team has worked on.

As mentioned before in previous sections, MongoDB is not an SQL language, meaning that we can store objects without restrictions as fields will be dynamically created to store any objects with new or unique fields. While in MySQL, an object with a new field had to wait for the user to manually create the field before being stored. Doing otherwise in this scenario would generate an error for the user. For example, the command to insert a user with the attributes included in Figure 25 would look something like this:

```
db.user.insert({id: 123456, name: "Joe Smith", address: “123 Fake Street”})
```

In the case of this command, ‘db’ is the name of the database and ‘user’ is the name of the table. Of course, this only includes the ID, name, and address of the user but can be modified to add the other required fields using the same format displayed.

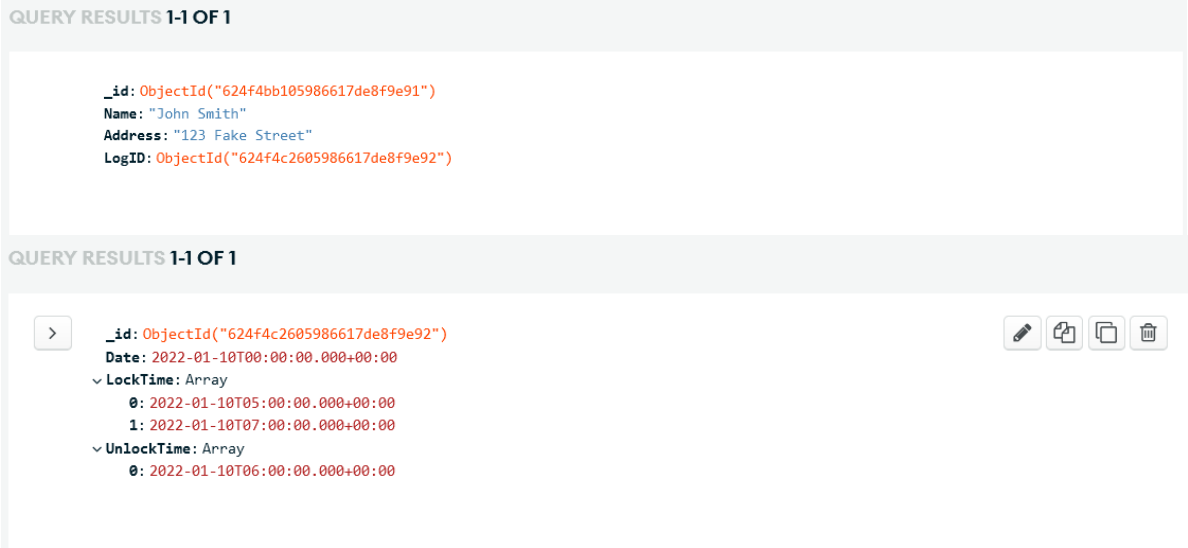

Figure 26: User and ActivityLog Objects

Figure 26 displays two BSON objects, a User and an ActivityLog, in the MongoDB Compass interface. Both have an ObjectID field which serves as both an index and a foreign key for other objects. The LogID field in the User object is the ObjectID of the ActivityLog object, relating both objects together for quick and reliable access. This method of foreign key relation is also used for the PinID and LockID objects.

6.12 Near Field Communication (NFC)

NFC, as mentioned in section 4.1.4, will be used as an unlock method. NFC connections are made from 4cm or less between two NFC capable devices. This connection allows for small amounts of data to transfer.

The implementation of NFC with Android devices differs from that of iOS devices. Android devices can facilitate an NFC connection between android devices or an NFC Tag. Android supports the following 3 methods of using NFC:
1. Reader/Writer mode allows a NFC device to read and/or write to an NFC tag or sticker.
2. P2P mode, which is used by Android Beam, allows for an NFC device to send data to other devices.
3. Card Emulation mode is what it sounds like. It allows an NFC device to function like an NFC Card which can be used for things such as contactless mobile payment. [14]

For iOS devices the implementation is slightly different. Apple's Core NFC Framework allows for NFC tags of types 1 through 5 stored in the NFC Data Exchange Format (NDEF). It also allows NFC tags to be written to and read. As

in the name of the Framework implies Core NFC requires a device that supports NFC. [15]

7. Hardware Design Details

This section outlines the specific hardware we identified for the Live Bolt system and each part's technical specifications. The lock box and the process to 3D model and 3D print it will also be discussed, along with how it will house all components.

7.1 Hardware Selections

Previously selected in section 3, we chose all the hardware components that'll be placed into our design. Our following selection would be to understand how pins will need to be configured in the MCU.

7.1.1 Arduino Keypad

Since we're using a 4x4 Arduino keypad, collecting the reading of the number and/or letter is a simple connection based on the division of rows to columns. Better analogy of the keypad configuration is shown in **Figure** 27**.** When no buttons are pressed, the columns are read as HIGH, and rows are read in LOW. When a button is pressed then the column is pulled to LOW from the current. From this observation of the MCU, it'd know the column in place so the roles would switch then the rows on HIGH. From the rows on HIGH and columns on LOW, the MCU can find the location of the button when the column pins retrieve back to HIGH. In our design, we will use this to configure onto the MCU and have the reading acknowledge the numbers and letters. Source for understanding the pin configuration is represented from the is from the resource of circuit basics [16].

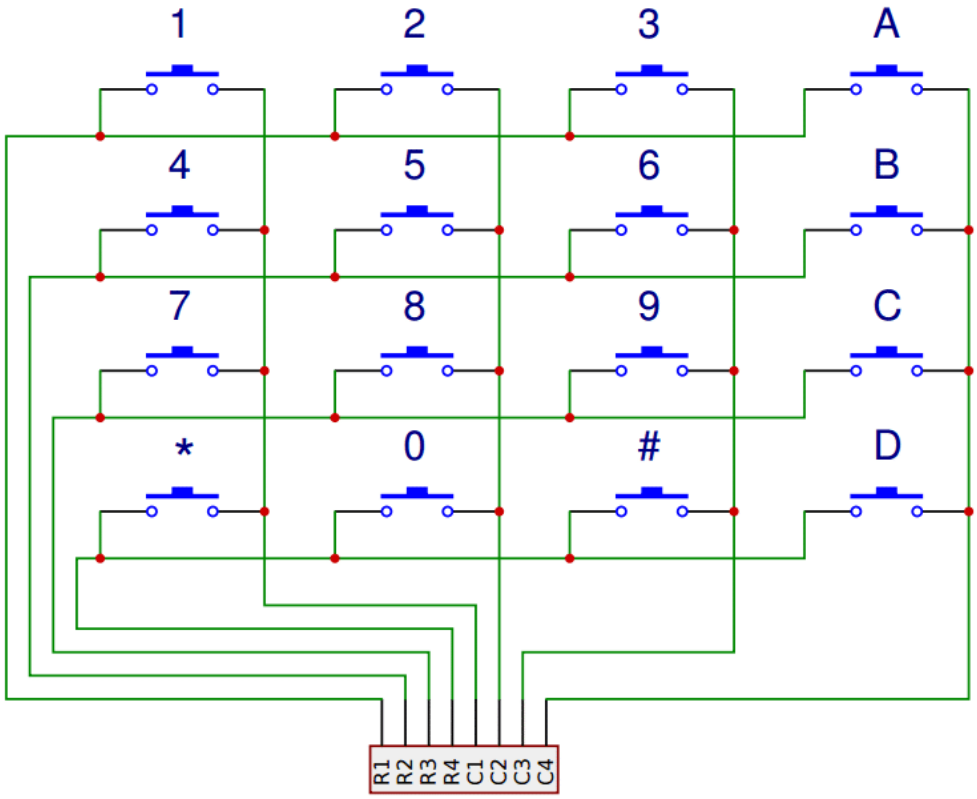

Figure 27. Arduino Keypad Configuration

7.1.2 Raspberry Pi

A Raspberry Pi incorporates numerous features including but not limited to multimedia performance, memory, and connectivity. Including this microcontroller to our project would create value in running the microphone and camera access while wirelessly transmitting data to the Live Bolt application. Configuration of how these would come together will be further elaborated in each respective part.

7.1.2.1 Raspberry Pi Microphone

A form of variety we want to incorporate into our product is to speak into the security box and have compatibly to home access. In doing so, our team will purchase a ReSpeaker 4-mic array to connect onto the raspberry pi. The main objective of using this is to have access to the microphone and add value to the use of the Raspberry Pi. The ReSpeaker includes a AC108 quad-channel ADC design for microphone arrays. One thing to note about this purchase is that it'll be objectively used for the microphone. A schematic of the ReSpeaker in **Figure** 28 shows the required 40-pin header Raspberry Pi connections along with the locations of the microphones. There is an addition to including LEDs, but we've decided to not use that portion of the component.

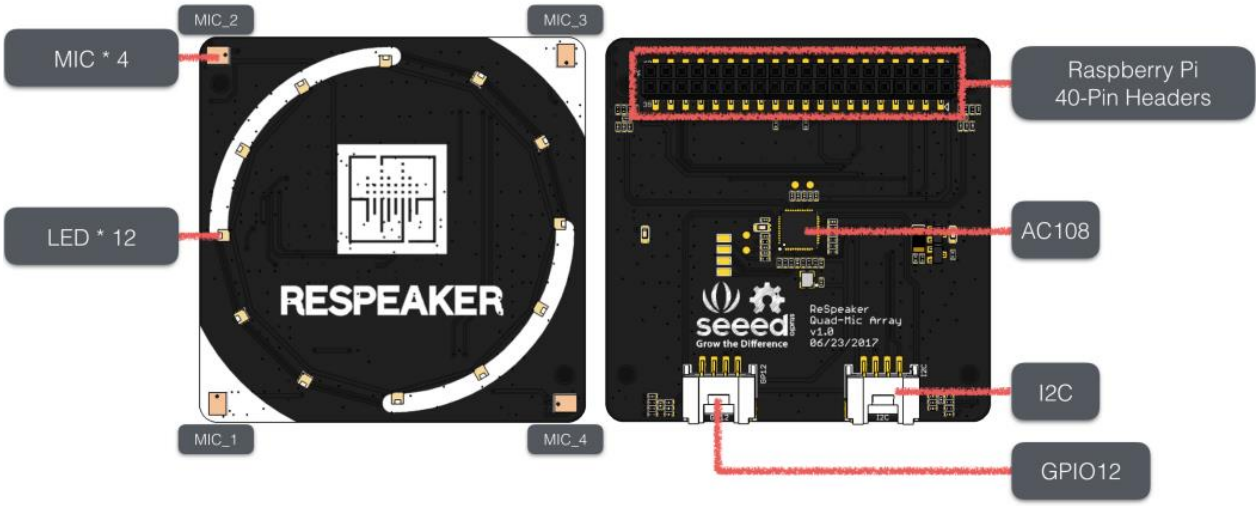

Figure 28. ReSpeaker 4-mic array

7.1.2.2 Arducam Camera Module

Another way we want to maximize the use of the Raspberry Pi is by including a Mini Camera Module. It's a low-cost add-on that is compatible with the models 4B, Pi 3B+, Pi 3B, Pi 2, and Model A/B/B+. It includes an angle view of 54 x 41 degrees with the capability of processing videos of 1080p30, 720p60, and 640x48-p60/90. The camera connects to the BCM2835/BCM2836 processor on the Pi via the CSI bus which carries a high bandwidth link of pixel data to the processor. An image of the connection is shown in **Figure** 29**.** We will enable the camera using once booting our raspberry pi.

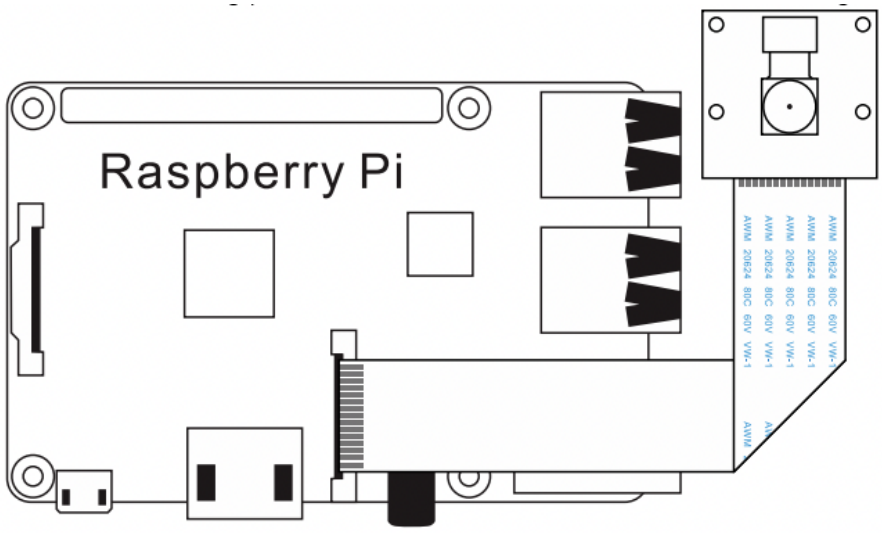

Figure 29. Arducam Connection to Pi

7.1.3 PIR Motion Sensor

The PIR motion sensor will be part of the Security portion of the design to detect font door activity. The sensor we chose is from Stemedu HC-SR501 since it is known for being energy efficient, high quality, and long service life. This choice even provides sensitivity adjustments of 3 – 7 meters and time delay ranging from 3 seconds to 5 minutes. The sensor cone even includes a 100 degree of coverage. This will be useful in our design for we plan on adjusting those parameters based our idea of designing the security box. The PIR motion sensor will be integrated to the MCU of the security box and relay activity to then turn on the camera. An image of the pin configuration for the sensor is shown in **Figure 30.**

Figure 30. PIR Motion Sensor Layout

7.1.4 High Torque – Metal Gear Servo

Based on a similar experiment of a smart lock we chose this High Torque Servo motor. Its features provide a rotation of 120 degrees (60 in both directions) with the flexibility of up to 170 degrees. It is light weight of 62.14 g with the size dimensions of 40.7x19.7x42.9 mm. We will use the servo to turn the knob in the dead bolt with the plastic horns provided from the kit chosen. How we plan to achieve this is through 3D printing different models that would turn the knob and attach to the plastic horns. The design of the 3D printed knob will vary through testing to see which would achieve a better effective outcome. Further elaboration of the knob design will be talked about hardware testing. Image of the servo and plastic horns that is determined to be purchased is shown in **Figure 31.**

Figure 31. Servo Motor and Plastic Horns

7.2 Hardware Configuration

The electrical schematic of connecting the hardware components together can be represented
[Images of port connections and such]

7.3 Power Distribution

Our system is complex in the sense that each component will require a voltage input to keep the system running. Since our design requires microcontrollers, sensors, Wi-Fi module and a servo, it's essential to make sure that each respective component will function properly and run given the power source provided. Before determining a power distribution system, collecting data of the operating voltage for each component would create value.

Table 11 below is a list of the components used and their voltage requirement. Based on the data collected, we will create a 5V battery pack to place onto the lock box and possibly including a 12V rechargeable battery onto the security box to compensate the total required power distribution.

Table 11: Component Power Requirements

Component Section	Power Requirement	Location of Hardware
Sensors		
Raspberry Pi Microphone	Source from Raspberry Pi	Security
Arducam	Source from Raspberry Pi	Security
Motion Sensor PIR Detector	4.5 – 20 V	Security
NFC Reader	2.7 – 5.5 V	Security
Microcontrollers		
MCU	7 – 12 V	Security
Raspberry Pi	5 V	Security
Microchip		
ESP8266	3.3 V	Lock
Servo Motor		
High Torque – Metal Gear	5 V	Lock

Collecting this data of all the power requirements for Live Bolt would allow us to move forward in choosing a reusable battery pack to charge power the system. The MCU (PCB Design) is an estimated power requirement and will be finalized within a future catalog of this report. Additional information about these individual components is that some come with voltage regulators within them. If they're not already provided, we will include a voltage regulator to step down the voltage and provide the required voltage output.

7.4 3D Printing

The deadbolt and security box can be created and built through different sets of materials. However, from our objective we want to ensure that Live Bolt is a lightweight portable product. Hence the choice of creating our prototype box will be from 3D printed filament.

Choosing the right filament to create a durable design and withstand temperature is our highest priority. We want to make sure that the customer is obtaining a long-lasting and eco-friendly product. Hence, we will narrow down the choice of using PLA filament for ease of printing quality, temperature withstands, and pricing.

Some qualities PLA filament includes are strength quality of 119 – 184 pounds; it's ability to stand with cold and hot temperatures without cracking; and its ability to be recyclable after usage. PLA filament is recommended to not be used outside where it is faced through UV rays or moisture; however, since this is a prototype and we're limited to manufacturing materials, it's best to showcase our design first from an enclosed box and later incorporate durable outdoor material. Choosing a PLA filament for the deadbolt design doesn't seem to be a problem since it won't be over-exposed to UV radiation or drastic temperature change.

7.5 Electronics Placement/ CAD

After researching and having the desired parts selected, it is essential to organize the electronic placements for our design. From our objective, we plan on having wireless intercommunication between the lock and security system to the app. Since we've decided to work with two aspects for our project, this design will have to be split into an indoor and outdoor design where they will respectively unlock the door and provide home monitoring. A free-handed sketch and basic CAD sketch, using solid works, of the electronics harnessed into a box will be shown.

7.5.1 Lock Box Prototype

Since this project is designed specifically to use a dead bolt lock, we need to integrate hardware to perform the locking/unlocking sequence at the command of the consumer. To meet our objective for the project, we will also keep the design of the lock as minimal and compact as possible.

There'll be three main components that'll be harnessed onto the deadbolt design: high-torque standard servo motor, 5V battery pack, and an ESP8266. The servo will oversee turning the knob in the dead bolt; the 5V battery pack will supply power to the servo motor and ESP8266; and the ESP8266 will be the wireless connection to command the servo to unlock/lock the door.

All dimensions of the electronics are taken into consideration when designing the length, width, and dead bolt diameter in the box. The customer will see a 120x80 mm box with curved corners to create an aesthetic design. More elaboration about the design of the box will be further discussed in the next section in CAD modeling. How we plan to organize the component in the box will be shown in the free sketch in **Figure** 32**.** This sketch is not to scale (NTS) and will be

modified to a more compact lock design. A better representation of the dimensions for the lock box is shown in the next section of the report.

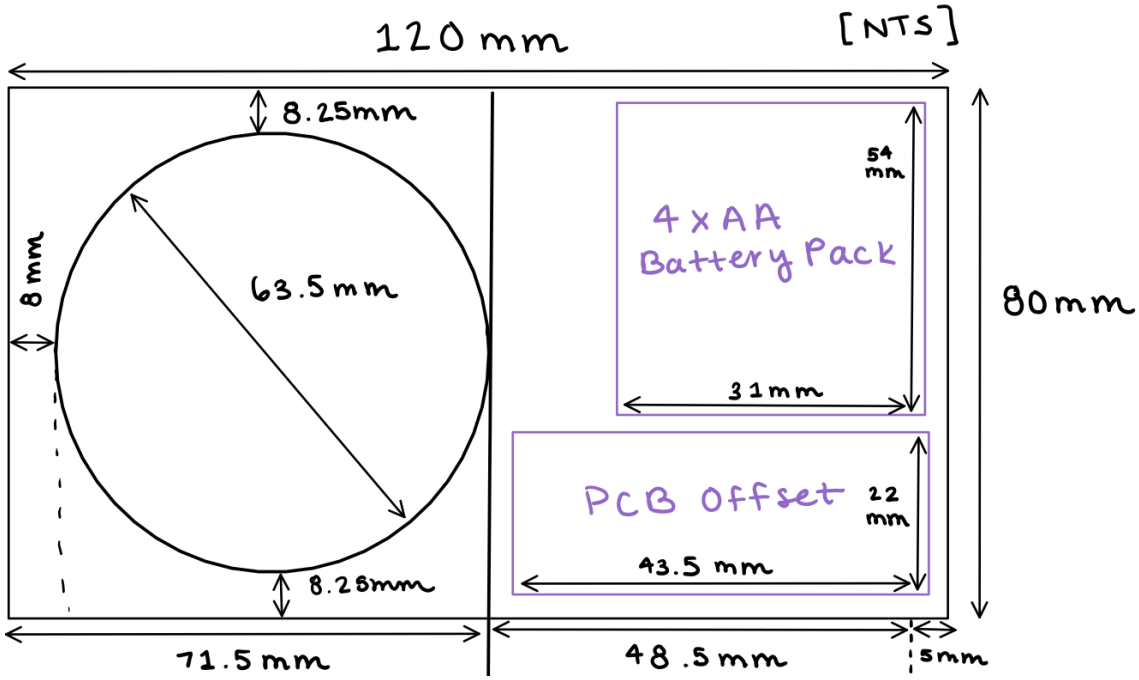

Figure 32. Lock Box with components

7.5.1.1 Lock Box CAD Model

Since we have the general layout of where the electronics will be place in our casing, the following step is to create a model of the box. We said previously that the dimensions will be 120x80 mm for the length and height. We did not mention that width of the box since it is being determined from the dead bolt extruded size along with the servo motor. Other than that, our overall dimension should be able to harness our components.

A unique portion for designing the casing is by having the consumer have easy access to the batteries. We mentioned in the design requirements that Live Bolt would have rechargeable batteries. To complete this requirement, the casing will have an ease of opening/closing the case with a railing system. It'll open the back portion of the case so the consumer can access the 4x AA battery pack and change when necessary. The CAD model of the idea is shown in **Figure** 33**.** Note that the casing will be 3D printed to also fit the objective of creating a lightweight box.

Figure 34 is an extension of the lock box case including the railing system to open and close the lock box. The image is a representation of the assembly minus the dimensions included.

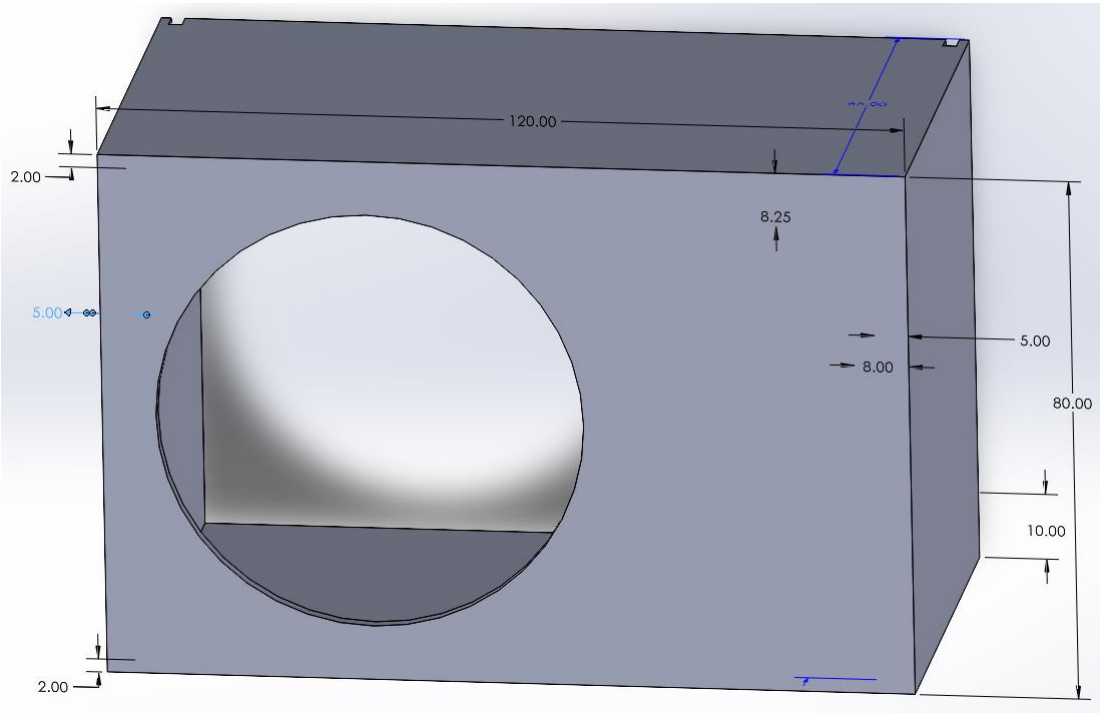

Figure 33. Lock Box CAD dimension model

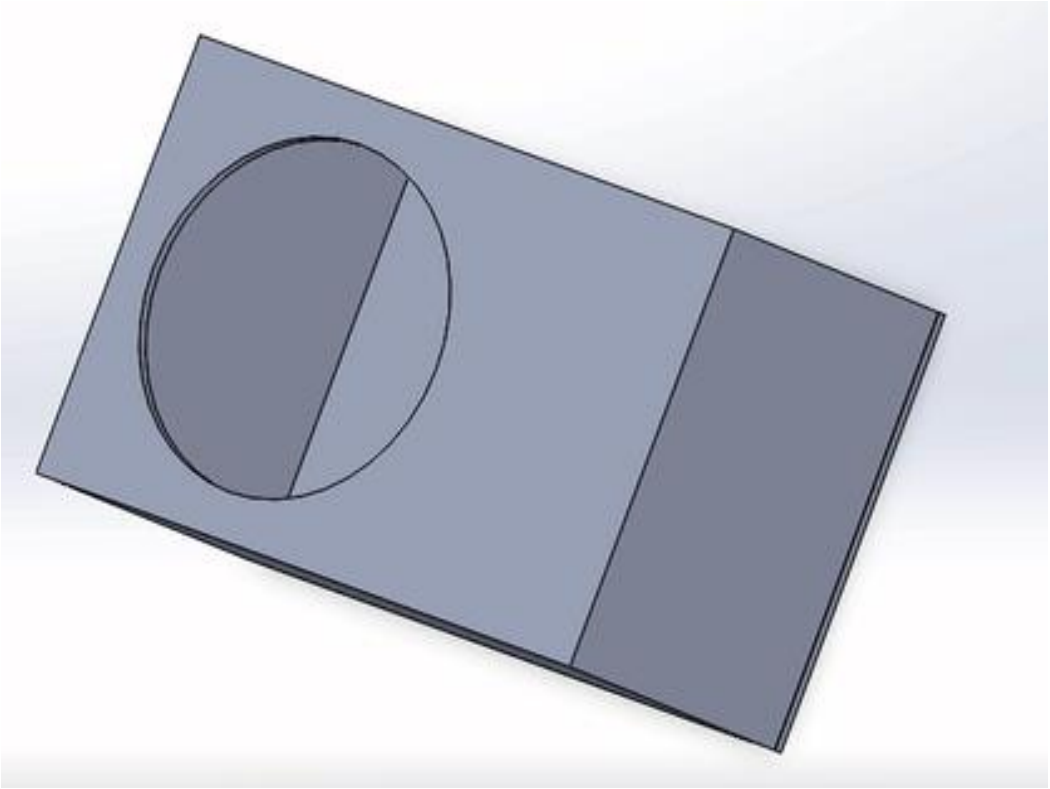

Figure 34. Lock Box assembled with removeable back

7.5.2 Security Box Prototype

The security box prototype design for Live Bolt will be built to accommodate the electronics to compensate with the requirements we laid out for home monitoring, home entrance, and wireless updates to the application. To complete the requirements, we narrowed down all the electronic parts to perform those actions, along with collecting their dimensions to comprise all the settings into a compact security system.

The outside perspective that'll be viewed by the customer will be a 160x82 mm box with cuts to represent the positions for the Raspberry Pi Arducam, PIR motion sensor, and Arduino keypad. The most interactive aspect outside of the box will be the keypad. A user can see and physically interact with the keypad for the home entrance Arducam and PIR motion sensor will be the security portion of the project. Their objective is to monitor and relay information about front door activity to the application. Including these components in our design will cover security object in our project.

The interactive portion of the box that is not viewed by the customer is the access to speech detection. Using a Raspberry Pi Microphone and relaying speech detection to the raspberry pi is way we wanted to incorporate access control to the customer. Since this microphone is flushed into the box, security breach from speech detection will be lowered since there is no outside acknowledgement of an entrance code.

Within the Live Bolt box, we will include a Raspberry Pi, MCU (PCB offset), NFC Reader, Raspberry Pi Microphone, and a rechargeable battery component. In the sketch for **Figure** 35**,** the box was designed to equate the size of the microcontrollers in parallel to compensate for the needs of the electronics, e.g., Arducam and PIR motion sensor. One thing to note that is missing from the sketch is including the design for the rechargeable battery. Since this portion of the project is still under review, it is still something we will consider when designing the box. The sketch also shown is not to scale (NTS) with representing dimensions, but the next section of the report will show an accurate scaling.

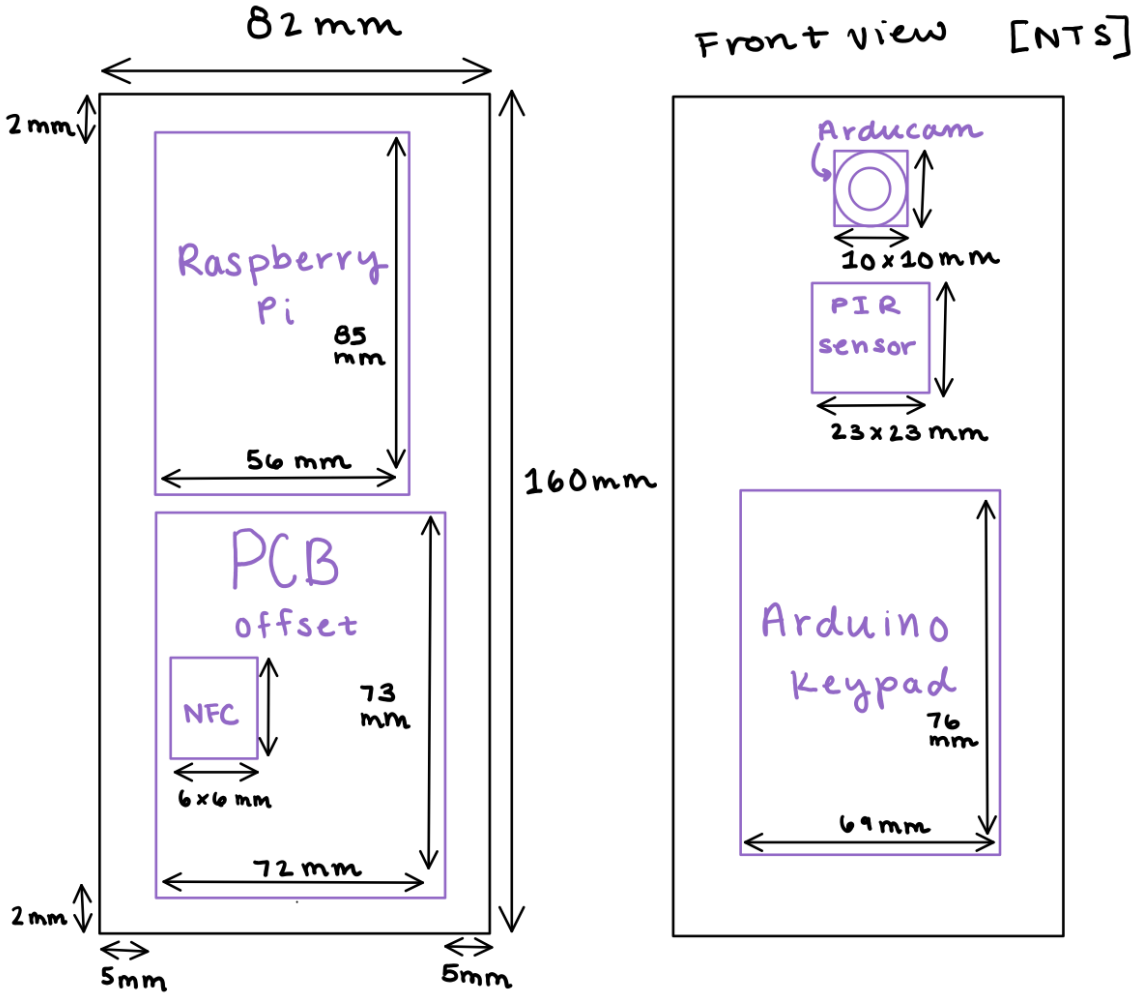

Figure 35. Security Box with components

7.5.2.1 Security Box CAD Model

After collecting the total dimensions of the components within the security box, the CAD model will have to cover mounting and connection. To achieve this there need to be 2 designs, the back portion of the box and the front view of the box. They will both vary in design based on what'll be required from each and come together in a railing system.

The back portion of the box will include 4 screwdriver holes and a railing system to mount onto the wall and then connect to the box. The box that harnesses the electronic components will have cuts for the Arducam, PIR motion sensor, and slit cut for the Arduino Keypad. It is scaled to provide the detection reading of the PIR 100-degree cone area and open space to include the keypad. This would amount to 160x82 mm in length and height. The width shown in **Figure** 36 is the offset amount we predict we'll need for the electronics. With this design, we hope

to make the security box have ease in mounting and access to the battery. In **Figure** 37, we have the outline for the box being assembled including the railing system for accessing the battery. There'll be a clipping method in this railing system that would lox the box in place and secure the mounting to the wall. Future edits that need to be added to the box design would be adding holes for the microphone.

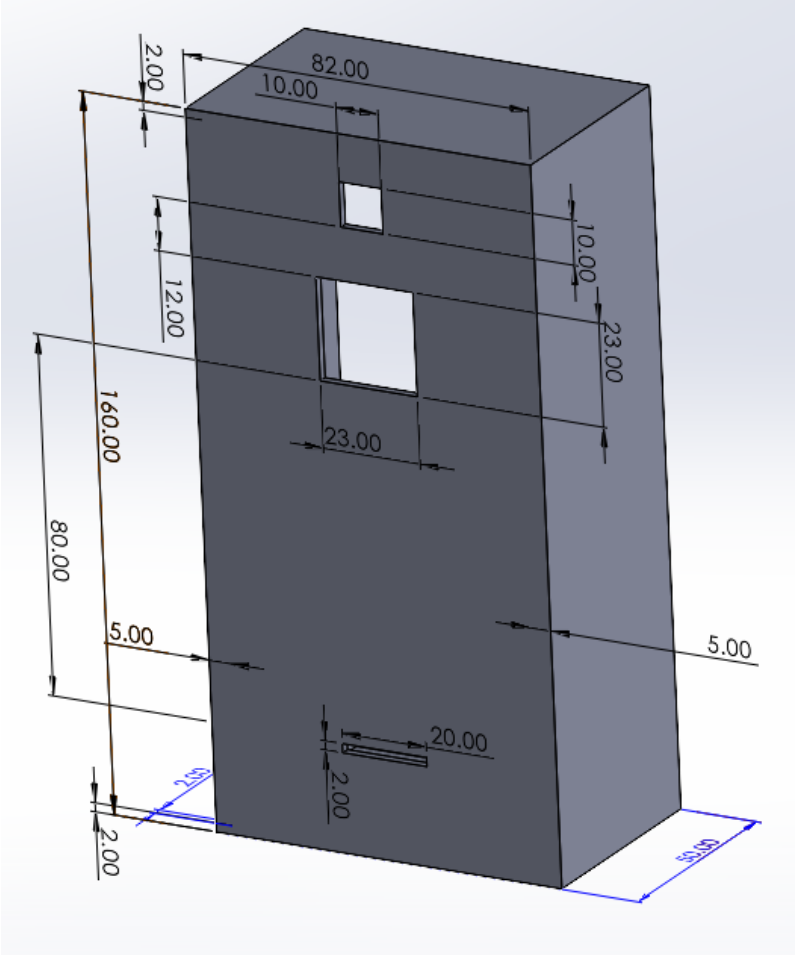

Figure 36. Security Box CAD dimension model

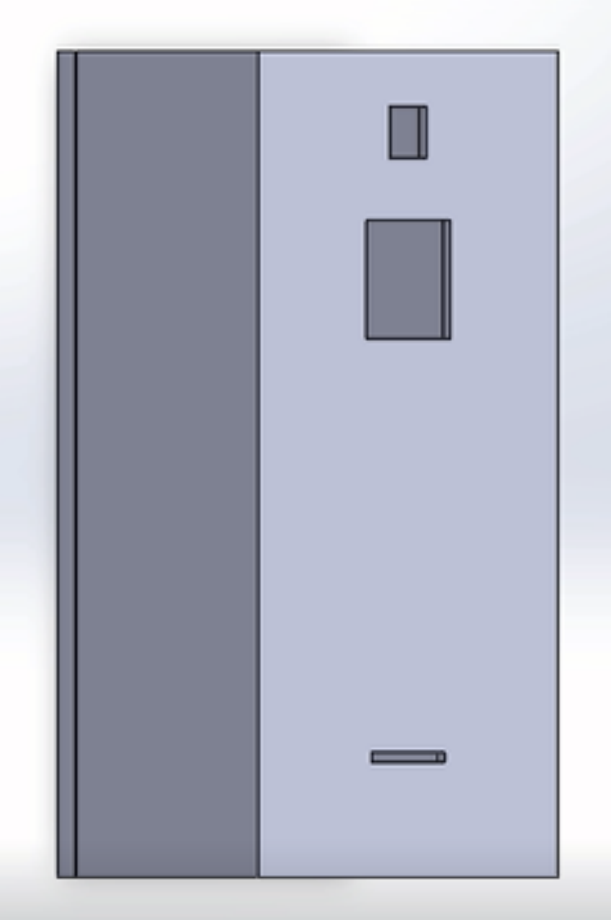

Figure 37. Secuirty Box assembled with removable back

7.6 Summary of Design

In hardware, we discuss about what components will be included but not the overall interaction of the system. Although Live Bolt is split into two aspects (lock and security), they're integrated together to perform the objective of home access and monitoring. In **Figure** 38 we how the microcontrollers, Wi-Fi module, and electrical hardware come together to provide the overall concept of what make Live Bolt a functioning home access and security system.

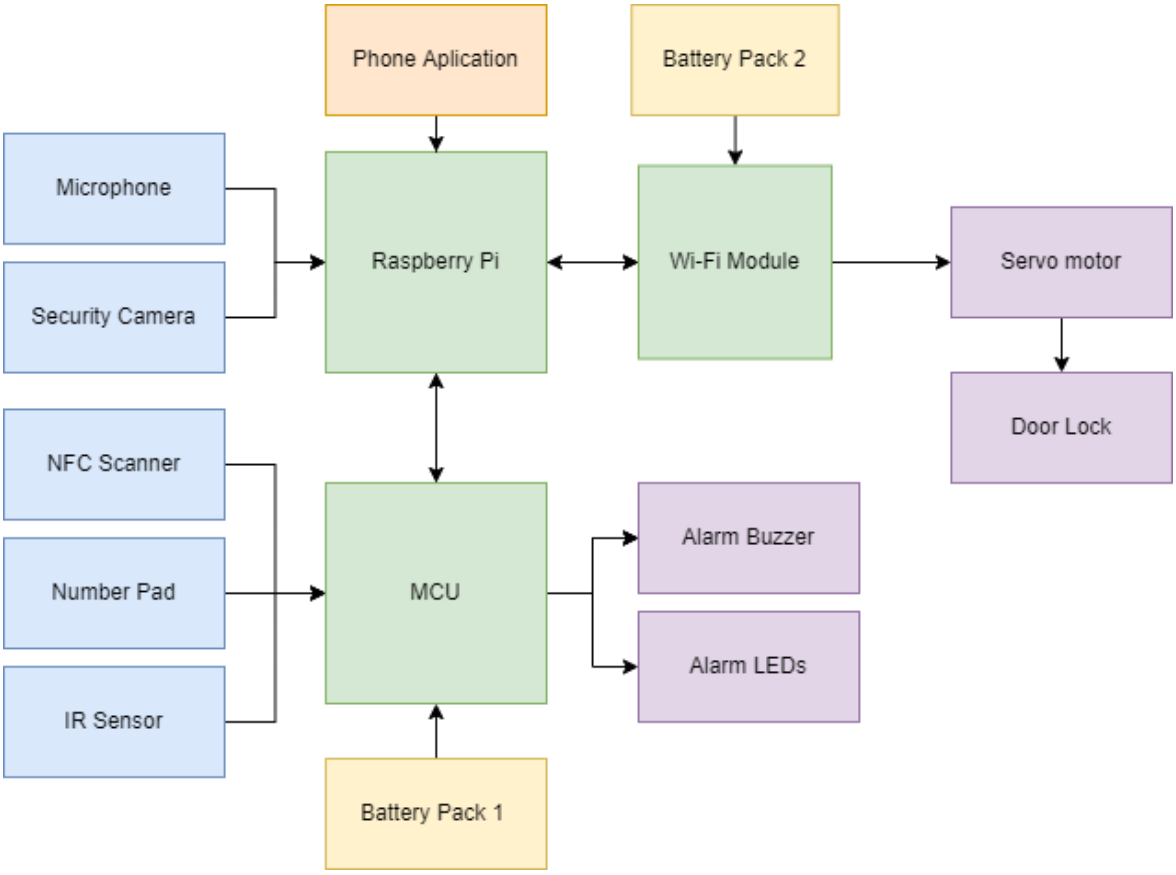

Figure 38. Hardware Design Block Diagram

8. Printed Circuit Board Integrated Schematics

To achieve a concise and compact final product, a printed circuit board will need to be designed after the breadboarding phase of the project is complete. This section will outline all aspects of the PCB design process as well as the ordering and assembly.

8.1 PCB Terminology

Table 12 provides some common terms and abbreviations regarding PCB terminology and what they mean.

Table 12: Common PCB Terms

Term	Explanation
CAD	Computer-aided Design. This refers to the software used to design a PCB.
Component	Components are what make up the circuit of the PCB. They get attached to the board after it is designed and printed.
Copper	Copper is a conductive metal that laminates most PCBs
DRC	Design Rule Checks are an automated process ran by the CAD software to ensure the PCB is ready to be printed. They find potential errors such as holes or traces being too close together, or errors in the circuitry.
Footprint	A footprint is a digital representation of a component that acts as a placeholder while designing the PCB. Usually, footprints have the component specifications like size to ensure everything fits properly on the board.
FR-4	The most common material PCBs are made of. It is a glass reinforced epoxy resin.
Gerber FIle	Gerber Files are files of the PCB that outline each layer of the design. They are sent to vendors who will print the board.
IC	Integrated Circuits are chips that house very small circuits that help to keep the overall PCB organized and neater. ICs have many uses and applications, from voltage regulators to timers and more.
Pad	A conductive spot for components to be soldered to the board.
Silk Screen	The silk screen is a nonconductive layer on the board, usually reserved for logos or words. It is useful to label specific spots in the silk screen to refer to them during assembly.
Solder Mask	Solder Mask is a nonconductive layer that covers most of the board. It insulates copper traces to negate shorting of the circuit.
SMT	Surface Mount Technology is a way to connect components to the board. Surface mount components are rested on pads and soldered directly to the board. Most consumer electronics rely on SMT.
Through Hole	This is the alternative to SMT and refers to the leads of electrical components being passed through holes in the PCB and soldered into it. This method uses up more space but is usually easier to do.
Trace	Copper traces are what conducts the electricity between each component. They are etched into the solder mask so that only the underlying copper remains.
Via	A via is a conductive hole in a PCB that can connect different layers of the board, such as components on the top of the board connecting to components on the bottom of the board.

8.2 PCB CAD Software

There are many computer-aided design (CAD) tools that allow for the design and testing of printed circuit boards. The two that we are considering for this project are KiCad and Autodesk EAGLE.

8.2.1 KiCad

KiCad is a free design suite for designing and simulating printed circuit boards that was developed in 1992. Since it is free of cost, there is a variety of tutorials and resources online. This makes the learning curve for KiCad much better for new designers. It offers Simulation Program with Integrated Circuit Emphasis (SPICE) simulation, open-source footprint libraries, and a 3D-view of the PCB once it is designed. Figure 39 shows a screenshot from the PCB designer/editor in KiCad.

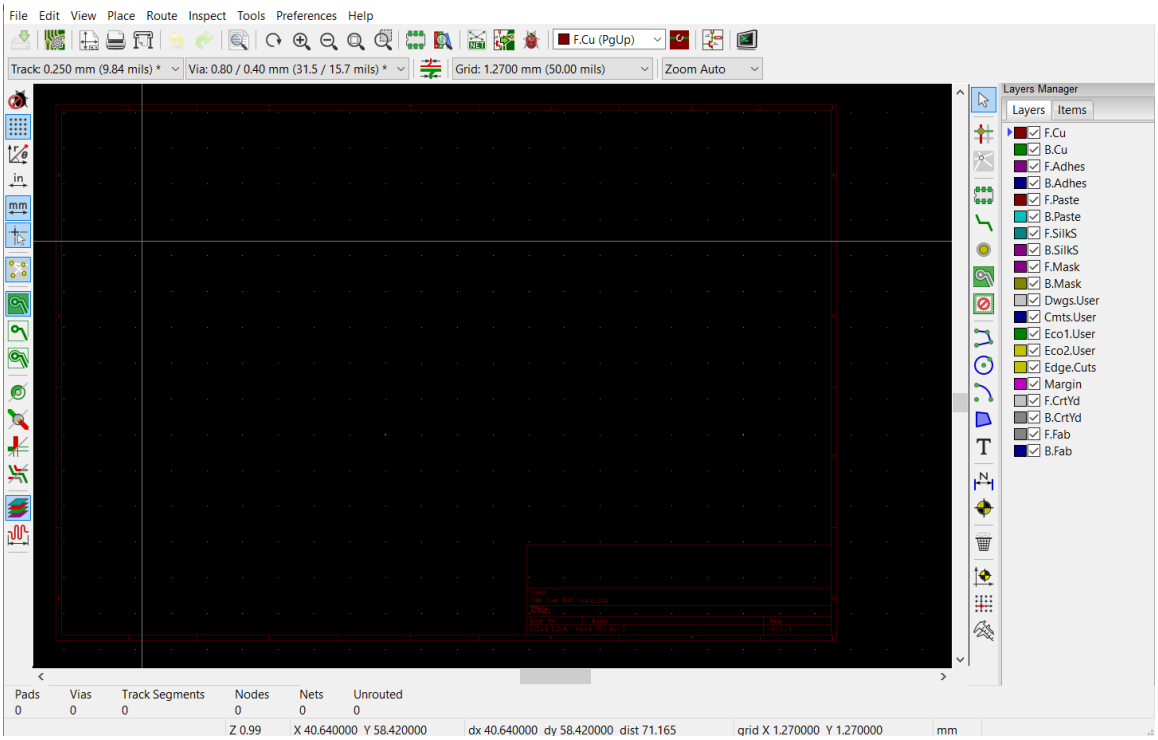

Figure 39: A screenshot of the PCB design window from KiCad

8.2.2 EAGLE

EAGLE, designed by Autodesk in 1988, is another CAD software for PCB design. EAGLE stands for Easily Applicable Graphical Layout Editor. Unlike KiCad, it is not free and instead offers a subscription-based model. However, University of

Central Florida students get Eagle for free with their student emails. Since EAGLE is closed source, there aren't as many online resources as possible and customization options, and it is a bit harder to start using when compared to KiCad. EAGLE was used as the CAD software of choice in the EEL 3926L Junior Design course at UCF, so students do have some experience with it.
EAGLE also offers the option to collaborate with others on a single design and dynamically update the design. Figure 40 shows a screenshot from the PCB editor in Eagle.

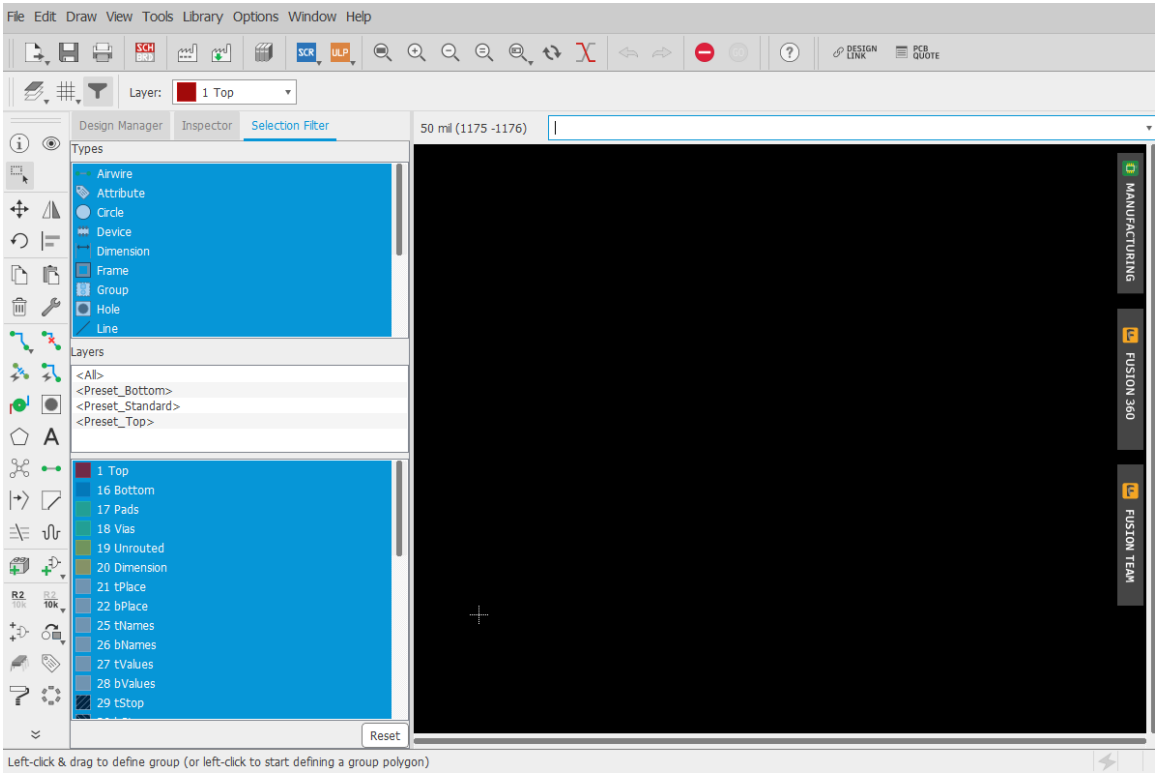

Figure 40: A screenshot of the PCB design window from Eagle

8.2.3 CAD Software Decision

Due to the advantages like, being free of charge, having a low skill level, and having open-source libraries and tutorials online, the team has decided to create our circuit and PCB design in KiCad.

8.3 PCB Vendor and Assembly

The process of getting a PCB printed is simple. Once the design is completed, the gerber files should be exported and sent to a vendor. Once they have the gerber files, the number of layers, size of the board, thickness, color, and

quantity are all defined. There are numerous vendor options online to get a PCB printed and mailed.

The option we are going to pursue will be JLCPCB due to having positive experiences with them in the past for printing PCBs. JLCPCB services over 20,000 orders daily and produce 6 million PCBs each year. They ship to over 170 countries and have an on-time delivery rate of 99.97%. Depending on the size and complexity of the board, they offer prices as low as $2 for 5 boards, excluding shipping. They also offer PCB surface mount assembly, since the process for surface mount soldering is more complex than through hole. For a standard PCB, meaning two layers and less than 400 millimeters by 500 millimeters, JLCPCB is able to start and finish production in one to two days. Once production is finished, the board is ready to be shipped. We do not expect our board to be larger than 150 millimeters by 150 millimeters, so we will use this size as a benchmark when calculating board cost and shipping. For a two-layer board of this size, the price of production equates to $13.40. This is made up of a $4.00 engineering fee and $9.40 for the actual board.

Table 13 gives the prices and shipping times for delivering to the United States of America for a two-layer 150 millimeter by 150 millimeters board. Since we are trying to minimize prices, we will most likely go with Standard Special Air Mail. We will have the summer semester to collect remaining parts before Senior Design 2 in the Fall, so a 12-25 business day delay is not an issue for us.

Table 13: JLCPCB Shipping Costs and Delivery Times

Shipping Method	Cost	Delivery Time	Restrictions
DHL Express Worldwide	$22.97	3-7 business days	≤10000kg
DHL Express Economy	$21.66	5-8 business days	≤11.5kg
UPS Express Saver	$22.82	6-9 business days	≤10000kg ≤120 cm x 70 cm
FedEx International Priority	$20.36	5-8 business days	≤10000kg ≤120 cm x 70 cm
Global Direct Line Saver	$11.69	6-16 business days	≤1.5kg ≤$150
Standard Special Air Mail	$11.46	12-25 business days	≤2kg ≤$150

9. Project Prototype Testing Plan

How we plan on testing Live Bolt will be split into hardware and software components. The testing plan are ideas/ goals we want to do to achieve a presentable project.

9.1 Hardware Test Environment

Designing Live Bolt comes with two components that is iterated throughout the document: a lock and security box. The lock box will only function with the use of a deadbolt lock while the security box just needs a brick, concrete, or stucco wall to mount onto. How we plan to initially test Live Bolt in an environment will be through our apartment homes. We plan on mounting the lock box to our doors and testing iteration between the wireless communication of locking/unlocking sequence.

As soon as we see a high success rate to the lock box, we will then transition to creating our own prototype door with a doorknob and deadbolt lock. The security box on the other hand would only be tested in an environment for when we're mounting the box. Mounting would require a sturdy wall and since we're in apartment complex, we'd like to avoid any consequence for drilling into a wall. Solution to presenting the mount and show in the senior design show case would be to create our own concrete wall. Meaning that we'd use a cement block to prove mounting. The reason to creating this prototype environment is to present the scenario for when it comes to demonstrating Live Bolt.

9.2 Hardware Specific Testing

In hardware we're working with electronic components, PCB design, and location choice that require a testing environment. We will highlight why we plan on testing

9.2.1 MCU Testing

Raspberry Pi & MCU (compatibility testing with app)

9.2.2 Deadbolt Knob Design Testing

In the earlier portion of the report, we mentioned that knob design would vary since all deadbolt knobs vary in sizing. From this understanding, our objective is to have the lock box be versatile with all deadbolt doors. We came up with two design solutions to complete that objective. One thing to note before creating the design is that knob case will be 3D printed & fit onto the servo motor's plastic horn. In **Figure** 41 the black oval will represent the deadbolt knob while the purple modification will represent the different designs to turn the knob.

Table 14 will the layout for outline for the pros and cons for each design. Although we will talk about what type of factors they bring, these are all assumptions we understand that could bring risk into the lock box. Testing these designs would provide better clarification on performance and durability.

Table 14: Pros and Cons for knob design

Knob testing	Pros	Cons
Design 1	Provides better flexibility to customize to knob sizing	Four pin design risks breakage
	Creates ease in movement of the servo	Requires extensive work for 3D printing
	Easy to install	Not easy to replace
Design 2	Durable	Causes a lot of friction making knob turn difficult (overworking the servo)
	Easy design	Fixed sizing therefore might not adjust to deadbolt knobs
		Requires more filament printing

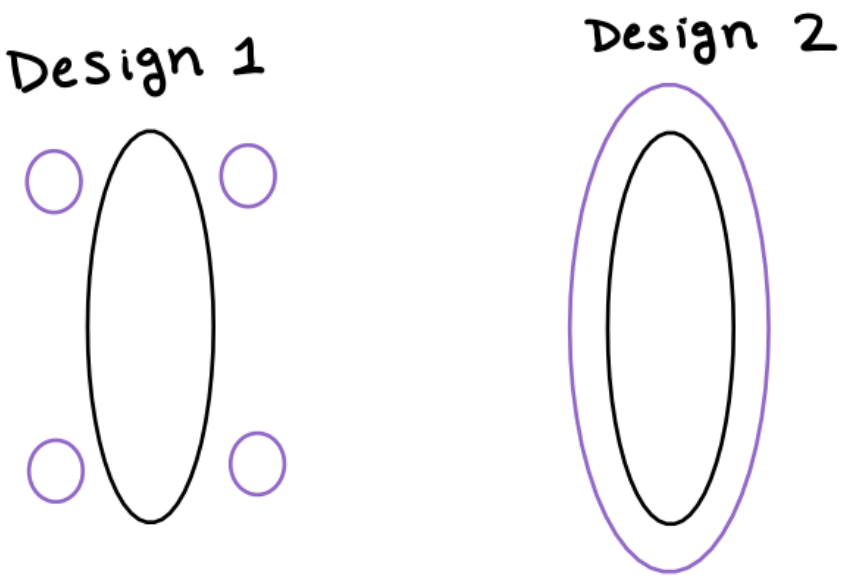

Figure 41. Deadbolt knob turning design

9.2.3 Electrical Components Testing

This section will go into detail how we will go about setting up and testing each component individually before integrating them together for the final design of the Live Bolt system.

9.2.3.1 ESP8266 Programming

Programming the ESP8266 NodeMCU is very similar to the way any Arduino board is programmed. There are a few things you must set up first, however.

1. First you must go to Preferences in your Arduino IDE and enter: http://arduino.esp8266.com/stable/package_esp8266com_index.json into the “Additional Boards Manager URLs” section since the ESP8266 isn’t running native Arduino architecture.
2. Then you must install the ESP8266 board package within the Board Manager menu. The option should be there if the URL was placed correctly in the previous step.
3. Plug in your ESP8266 into the computer via a micro-USB. If a low-quality cable is used, the computer may not be able to read that the board is plugged in.
4. Once plugged in and recognized, the computer should automatically download drivers. If it does not, you can manually install the CH340 drivers from the internet.
5. Once everything is correctly installed, you can select “NodeMCU 1.0 (ESP-12E Module)” from the board menu.
6. Select the new COM port in the Port menu.

With these steps completed, the ESP8266 should be ready to be programmed. There are some differences with programming an ESP8266 vs an Arduino. One major difference is that when defining pins, the Arduino allows you to use the number of the digital pin. Digital Pin 6 = D6 = 6. On the ESP8266 NodeMCU breakout board, the digital pin number does not correlate with the number you define in the code. This means you need to refer to a pinout sheet when coding. The number you define for each pin correlates to the respective GPIO pin. This means that D6 on the development board would be defined as “12”, since it is actually the GPIO12 pin.

There is also a way to program the ESP8266 without actually plugging it in to the computer and uploading the code through a serial connection. This method is known as Over the Air programming. You can set this up via the Arduino IDE. An example code in the IDE called “BasicOTA.ino” provides the framework for this process. With the ESP8266 plugged in to the computer, you have to configure your Wi-Fi parameters. After running the code and saving it to the board, unplug it. You can now power it with a portable power source such as a battery. When the board has power, you’ll see a new port in the tools that you can select. There are a few Over the Air functions that you must put into your original sketch code, and once that is done, you can upload the sketch and your board should have received it wirelessly.

The Over the Air method of uploading code to the ESP8266 will come in handy once we solder the chip to our designed PCB, since we will not have a micro-

USB port to upload sketches to it. This means we can make small changes to our code without having to desolder and resolder the microchip every time.

9.2.3.2 HC-SR501 IR Sensor

The IR sensor does not require any microcontroller to function. This mean that it can be hardcoded into a circuit. The main set up required for this component is configuring the sensitivity (range) and time delay of the sensor. There are two potentiometers on the board that can be turned to adjust these parameters. Since we would like to detect motion in front of the lock, the range will be set to 1-3 meters to get a decent coverage. The delay will also be set to 2-5 seconds so it doesn't trigger again for the same person passing by. There is also a pin jumper in the corner of the module to change between single trigger mode or repeat trigger mode. Single trigger mode will change the state of the sensor whenever motion is detected. Repeat trigger mode will turn the sensor on when motion is detected, and after a delay it will turn it back off until motion is detected again. Repeat trigger is more useful for our application because we will want to detect motion, send a notification to the application, and then wait for more motion afterwards. The final consideration for this sensor when setting up is that it needs up to a minute after receiving power to calibrate and get acclimated to the environment it is placed in. To test if the component works, an LED and 330-ohm resistor was connected to the sensors output. After calibration, when motion is detected, the output will go high and light up the LED. The test setup is shown in Figure 42.

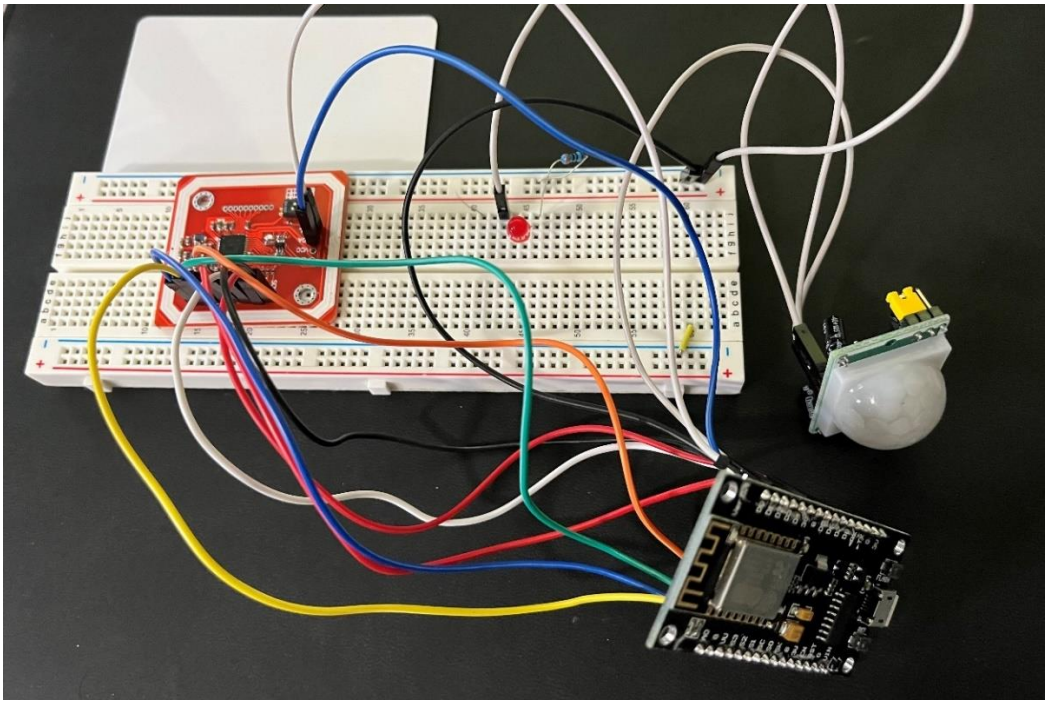

Figure 42: Hardware Development of NFC (Left) and PIR Sensor (Right)

9.2.3.3 Membrane Keypad

The keypad is simple to test with the ESP8266 and requires very little setup. In the Arduino IDE, a character array is created to mimic how the keypad looks. By using the Keypad.h library, the character can be printed when the corresponding button is pressed. For testing, the character is printed to the serial monitor in the Arduino IDE. For the actual product, the code inputted will be compared to a set pin code and will either grant or deny access to the user. The keypad test setup is shown below in Figure 43. Each of the eight pins on the keypad is connected to a digital pin on the ESP8266, D1 through D8.

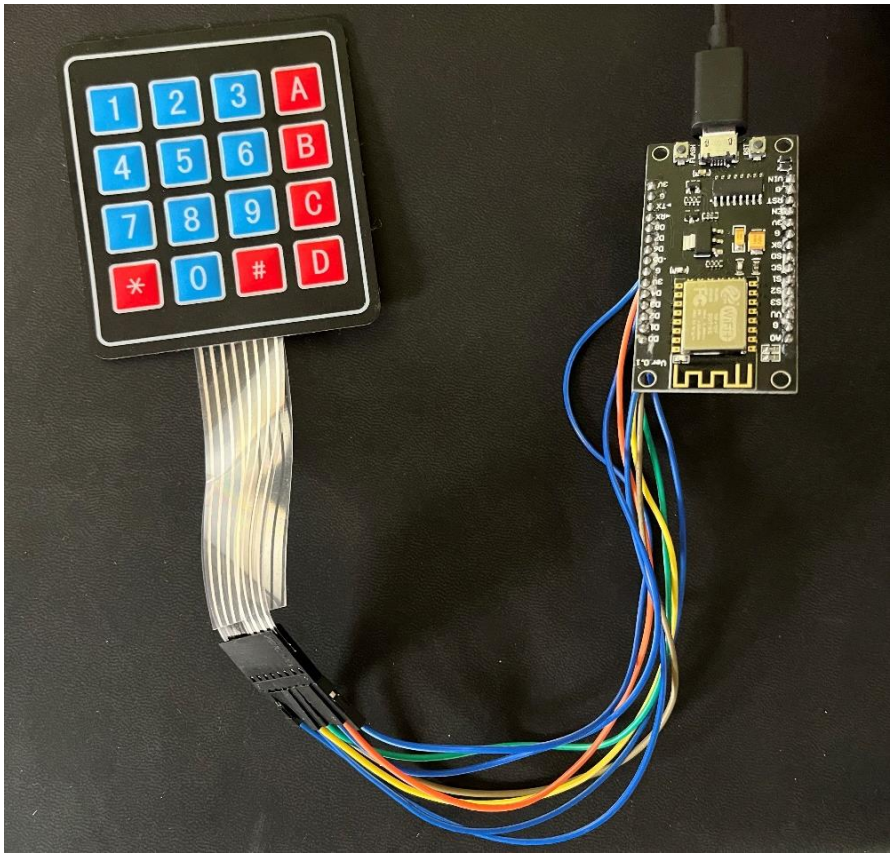

Figure 43: Hardware Development of Membrane Keypad

9.2.3.4 HiLetgo NFC Module

The HiLetgo PN532 module comes unsoldered, so header pins need to be soldered on to connect it to the final PCB design. For development, jumper cables were inserted into the header holes. There are two switches that act on binary logic to change the communication protocol of the board. These protocol options are shown in Table 15. I2C was chosen as the protocol for ESP8266

communication. We have a blank NFC card to test if the module is able to read it. Figure 42 shows the development of the NFC module.

Table 15: Communication protocols for NFC Reader

Protocol	Switch 1	Switch 2
HSU	0	0
I2C	1	0
SPI	0	1

9.2.3.5 High Torque Servo Motor
9.2.3.6 Arducam

The Arducam camera module can be seen interfaced with the Raspberry Pi in Figure 44.

9.2.3.7 ReSpeaker 4-Microphone Array

The Respeaker is an array of four microphones that work with the Raspberry Pi. To begin testing, the array needs to be mounted to the Raspberry Pi in the proper orientation, so every pin corresponds to the correct header. The microphone array can be seen mounted in Figure 44. Then the Respeaker Seed voice card must be installed to the OS on the Raspberry PI. In Audacity or another audio editing application, the input channels need to be configured to the AC108 4 channel, corresponding to the 4 AC108 microphones. Now the array should be configured to act as a microphone. There are also LEDs on the board, but since they will be placed inside the housing of the Live Bolt system, we do not need to program them.

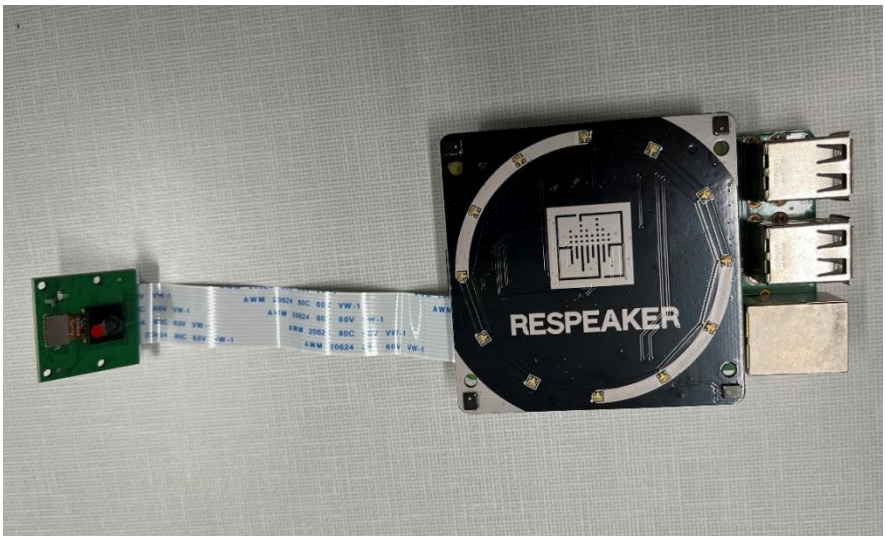

Figure 44: Microphone array and Arducam camera interfaced with Raspberry Pi

Now that the microphone is set up, software needs to be created to enable phrase detection. The programming language of choice for this is Python due to its ease of use and plethora of online materials to work with. There are a variety of packages for speech recognition in Python, such as:

- apiai
- assemblyai
- google-cloud-speech
- pocketsphinx
- SpeechRecognition
- watson-developer-cloud
- wit

Apiai and wit even offer natural language processing capabilities. Natural language processing allows for not only capture of words, but also emotion. Since we only need to recognize a spoken word to act as a passcode for the lock, recognizing emotion is irrelevant and we will not be considering natural language processing in the design. SpeechRecognition actually makes use of many of the APIs listed and allows for each one to be called and compared against each other. This makes SpeechRecognition the most versatile option and the best choice for the phrase detection software. The code works by using th Recognizer class for each specific API. The class won't work without an audio data type to pass through it, which is the voice captured by the microphone or from an audio file.

To capture voices in real-time, another library is required, PyAudio. PyAudio allows for the .Microphone() class to be used with SpeechRecognition. Now the .listen() method can be called to listen to voices live. From here, the system can detect what the voice said, and simple if logic statements can be called to do different things based on the phrase detected. One final thing to note is that sometimes there will be ambient noise, due to the lock being outside. To circumvent this, the adjust_for_ambient_noise() method of the Recognizer class can be called to equalize the audio. It does this by listening to the first bit of the recording and calibrates to that noise level to try to compensate for it. While not perfect, it can help for minor ambient noise like birds chirping or leaves blowing.

9.2.4 Battery Testing

Testing Live Bolt's batteries will be split into checking the locking and security system. We talked in power distribution about the require power for each device. The next step after configuring the power requirement would be to test if the rechargeable battery pack would suffice our desired output. Testing the battery pack in the lock box would be as simple as inserting the AA batteries into a 5V battery case and measuring the outcome with a digital multimeter. As for the security box batteries, it is still to be determined based on the PCB power

requirement. It could possibly be placed in a pack or be integrated to a portable charger. Regardless of the decision testing the voltage output will still a digital multimeter just to run verification.

9.3 Software Test Environment

Just as important it is to write clean code is to test that written code. This is more easily accomplished by a software testing environment. The key areas in a Software testing environment include:

- System and applications
- Test data
- Database server
- Front-end running environment
- Client operating system
- Browser
- Hardware includes Server Operating system
- Network
- Documentation

When first setting up the Test environment several factors should be taken into consideration. Some tests may not run locally on a computer so a server test environment may need to be created. Some tests may require a network connection to be established before tests can be run. Some testing frameworks can be run on different browsers or software's to be used in conjunction when testing. Additionally test data may need to be provided to ensure the functionality of the program. Many companies use a portion of their production data as test data which is something we will consider when we begin testing.

Some of the common challenges to a Test Environment include:

1. Planning how resources will be used.
2. Shared usage of the testing Environment by the team.
3. The time required to set up the test environment.
4. Variations in the way the test environment must be configured when dealing with different types of tests.

Some of the best practices to a Test Environment include:

1. Clarifying the Testing Requirements.
2. Checking connectivity of hardware and software components before testing.
3. Making sure that all the required hardware and software had been correctly installed.
4. Planning a routine usage of the testing environment.

5. Automating any tools and their dependencies. [17]

9.4 Software Specific Testing

9.4.1 Types of Testing

Within Software Testing there are different types of testing. Each one has a specific focus in mind as the reason for its use. Choosing the right testing type will also determines the type of testing frameworks we will use during our design. All have importance however for our project we will discuss the most common below.

Before we can discuss specific tests, we first must look at Automatic versus Manual testing. As both names suggest manual testing is done by a person running each test themselves using various tools. This can be quite expensive in both time and money. Additionally, humans are subject to error resulting in inaccurate results. Automatic testing on the other hand involves a machine running tests independently of a person. These scripts can be run by the computer a set interval of time and is typically much faster than manual testing.

1. Unit Tests involve testing individual methods and functions used by components of software. These test typically are cost effective and can be run quickly.
2. Integration Tests test how well different parts of an application work together. These are useful for testing two parts of an application that rely on each other such as the database and frontend.
3. Functional Tests focus on the business requirements of a piece of software. This deals primarily with the outputs of different functions without focusing on what preforms the action. These tests only verify what is produced by whatever is being tested.
4. End-to-end Tests verify a user's behavior through software within an application environment. By using End-to-end tests human behavior can be tested more rapidly and the application flow can be tested for accuracy.
5. Acceptance Tests, like functional tests, are tests that check business requirements. These tests require the entire application to be functioning before and test for user actions within the application.
6. Performance Tests put the system under stress to test the software when it is under a substantial load.
7. Smoke Tests are quick test designed to check basic functionality of a program. They are useful primarily after a new build has been released and is being tested to see if it requires further more expensive testing. [18]

9.4.2 Testing Frameworks

The 5 different testing frameworks we will be considering for our project include Jest, Mocha, Cypress, Jasmine, and Selenium. We will be discussing more about each one in each of the following sections. Each one has its own uses and implements at least 1 of the major models in automated testing.

1. The Linear Scripting Framework involves the creation and execution of tests in an incremental order. The tests are produced and run individually under this model. This model is based on the concept of record and playback.
2. The Modular Testing Framework is based on the concept of abstraction. Independent tests are developed, and an abstraction layer is used to hide the tests from the application.
3. The Data Driven Testing Framework uses tabular storage of the input and expected output results when testing.
4. The Keyword Driven Testing Framework involves using data tables and keywords to analyze actions that occur during tests. It is an extension of Data Driven Testing.
5. The Hybrid Testing Framework is a combination of a couple other frameworks including modular, data-driven and keyword testing. This framework is used frequently in end-to-end testing.
6. The Test-Driven Development Framework (TDD) includes automated unit testing that pushes forward software development. This framework type allows for rapid development of ode as well as confidence in the systems reliability.
7. Behavior Driven Development Framework (BDD) stems from Test-Driven development but differs in how tests focus more on system behavior. [19]

9.4.3 Jest

Jest is a popular JavaScript testing framework created by Meta. It is known most distinctly for its simplicity. Some of the key features it provides includes requiring limited configuration, snapshots, isolated tests, and well documented API. Just after installation is set to run without requiring much groundwork out of the box. Snapshots allow for easer tracking of object in real time as testing is being done. Tests are also isolated by separating the processes that run them. Additionally, Jest is well documented both by the creators as well as the large community that uses the Framework. Jest works with a variety of languages and tools including those being used in our MERN stack. Jest is simple to install and can be easily through entering one of the two following commands into the command line: “npm install --save-dev jest” or “yarn add --dev jest”. [20]

9.4.4 Mocha

Mocha is a JavaScript testing framework that runs on Node.js through the browser. Mocha can be used for both synchronous and asynchronous testing. Mocha also used frequently with Chai, a well know assertion library for Node.js. Mocha's default interface is Behavior-driven development (BDD) which helps developers create more predicable behavior within developed code. Behavior-driven development rose out of an offshoot from Test-Driven development (TDD). Mocha can be installed easily by using the command "npm install mocha" in the command line. [21] [22]

9.4.5 Cypress

Cypress is a frontend testing framework that allows for developers to create faster and more reliable tests. Cypress addresses the major areas of software testing including:

- Setting Up Tests
- Writing Tests
- Running Tests
- Debugging Tests

Cypress is similar to Selenium, discussed in section 9.4.7, however differs in its architecture and lack of the same restrictions that hold Selenium back from further use. With Cypress end-to-end, integration, and unit test can all be implemented. Cypress also includes the freely available cypress ecosystem which includes Test Runner for running test and a dashboard interface for keeping track of recorded tests. Cypress also includes the following features:

- Time Travel which includes screenshots capture as tests run. These screenshots can be viewed to see further information regarding any issues that happen.
- Debuggability is present in how errors are easy to read, availability of debug tools, and stack traces.
- Automatic Waiting allows for automatic waiting for commands before the rest of your code runs. As such commands to manually make a program wait are not needed when testing.
- Spies, Stubs, and Clocks involves managing how timers, server responses, and functions run.
- Network Traffic Control as it sounds allows for management of network traffic.
- Consistent Results are made possible by not relying on or integrating existing programs.
- Screenshots and Videos are made available as the test runs. They can be viewed through the Command Line Interface as the program runs.

- Cross Browser Testing allows for Cypress to run on multiple different browsers to ensure functionality.

All of this makes Cypress a compelling option for testing our development. [23]

9.4.6 Jasmine

Jasmine is a testing framework for JavaScript. Because it was developed for JavaScript it runs on any languages or tools that implement JavaScript, like React. Jasmine has easy to understand syntax and supports asynchronous testing. It also allows for using test doubles through the use of spies. Additionally, it can facilitate Frontend testing through the Jasmine-jQuery extension. Jasmine can either be added to Node.js or from a browser. Only 3 simple commands need to be run before Jasmine can be used for testing. As a result of this the set-up process for Jasmine is relatively easy. Additionally, Jasmine has much documentation making it quick to find references any issues are run into. There is also an active community for the framework making in a solid choice for testing our application. [24]

9.4.7 Selenium

Selenium is at its core a few tools for automating web browser testing. Selenium has an IDE can easily be downloaded as an extension for Google Chrome and Firefox. The extension is easy to use and has much documentation on their website. It also has a domain-specific language, Selenese, that allows for tests to be written in many popular languages. Selenium has a couple tools available as part of its suite. These tools include the Selenium IDE, client API, Remote Control, WebDriver, and Grid. The IDE as discussed before manages test scripts, automated testing, and code completion. The client API allows for tests to be written in different programming languages which are translated to Selenese through calls to the Selenium API. Selenium Remote Control is a server that allows for automated tests to be run in a browser. The Selenium WebDriver includes an interface that communicates instructions to make content work the same across browsers. The Grid allows for tests that would typically run-in browsers to run on remote machines. Selenium’s wide array of tools make it a compelling option for us as a Testing Framework. [25]

10. Administrative Content

Designing a smart door lock and security system has been a challenging yet fun and rewarding experience for the Live Bolt team. All the administrative tasks like sharing documents and creating meetings were done through a group chat platform known as Discord. Discord played a large role in allowing the group to always keep an open line of communication during the design process. It has the

capability for text communication as well as voice calling and screen sharing features.

In addition to constantly communicating online, the team also met up in person once a week. Since the members all had lots of outside obligations, such as work and class, Friday mornings at 9:30 am were decided to be the recurring meeting time. The team usually opted to meet in the engineering atrium or library study rooms. The purpose of these weekly meetings was to provide updates on each member's specified area. We broke down the project into two main areas: hardware design and software design. Software design mainly includes the phone application and the User Experience design. Hardware design includes the prototyping of the circuit and PCB design, and the design of the housing that will hold the components. Our team was made up of two computer engineering students who focused on the app, and two electrical engineering students who focused on the circuit, PCB, and 3D design of the housing. Separate meetings were held on occasion for each sub team to get more work done. By splitting the team this way, we were able to divide the work up and complete the design process much quicker. The division of labor can be seen in **Figure** 45.

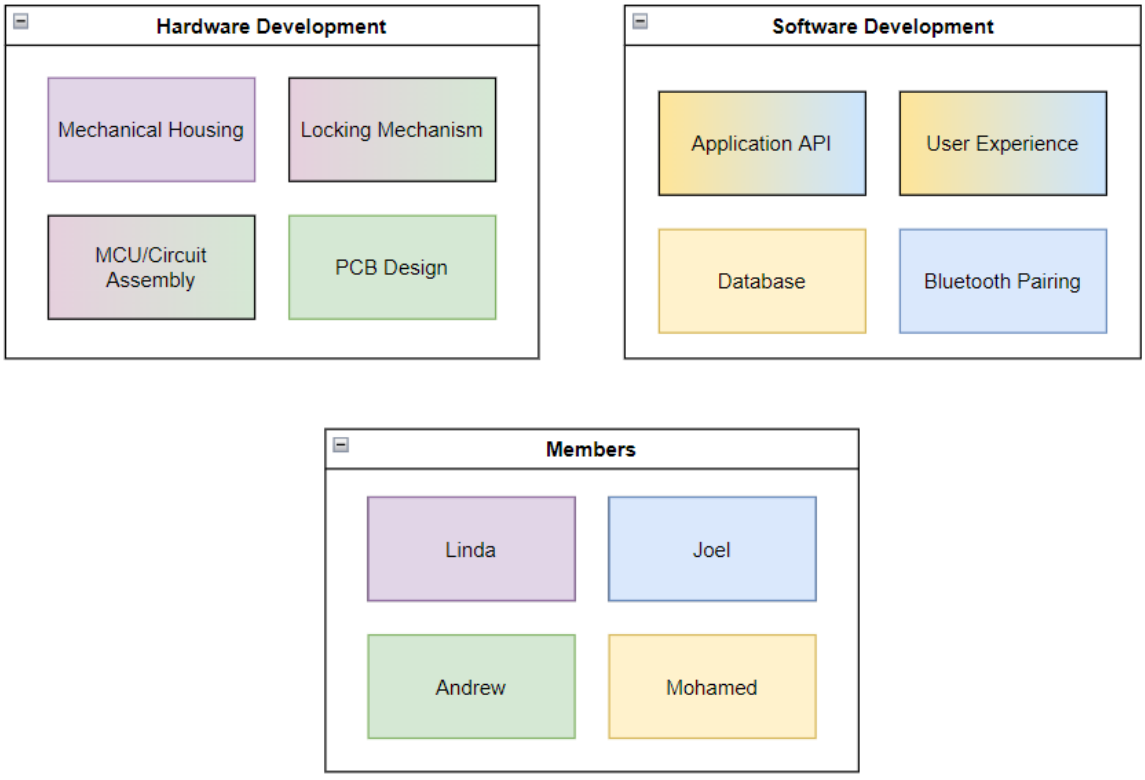

Figure 45: Division of Labor for Live Bolt

Going into Senior Design 2, the team will continue to carry out this division of duties when creating the application and physical product. Figure 47 provides

some important milestones for the hardware development process between Senior Design 1 and Senior Design 2.

10.1 Milestone Discussion

In **Table 16** and **Table 17,** shown below detail our plans of researching and completing our senior design project. Table 16 focuses on our milestones we will be working on in Senior Design I while Table 17 lists our milestones for Senior Design 2. The milestones column lists the tasks we are trying to accomplish during each semester. The duration column lists the number of weeks we will devote to each milestone. The dates column lists the deadlines we must complete the milestone by to keep on track. Some of the dates in Table 17 are tentative as we do not know the exact dates, we will need to complete milestones that far into the future. As we continue working on and researching our project, we plan to revisit these milestones frequently.

Table 16: Senior Design 1 Milestones

Milestone	**Duration**	**Dates**
Brainstorm	1 Week	January 14
Select Project	1 Week	January 21
Divide and Conquer	2 Weeks	February 4
Research	4 Weeks	March 4
Table of Contents	1 Week	March 11
60 Pages Due	2 Weeks	March 25
Edit Draft	3 Weeks	April 15
Finalize Report	1 Week	April 22
Submit Final Report	1 Week	April 29

Table 17: Senior Design 2 Milestones

Milestone	Duration	Dates
Source Remaining Parts	2 Weeks	September 2
Build Prototype	4 Weeks	September 30

Test and troubleshoot	3 Weeks	October 21
Finalize Prototype	3 Weeks	November 11 (Tentative)
Final Report	2 Weeks	November 25 (Tentative)
Final Presentation	1 Weeks	December 2 (Tentative)

10.2 Software Development Gantt Chart

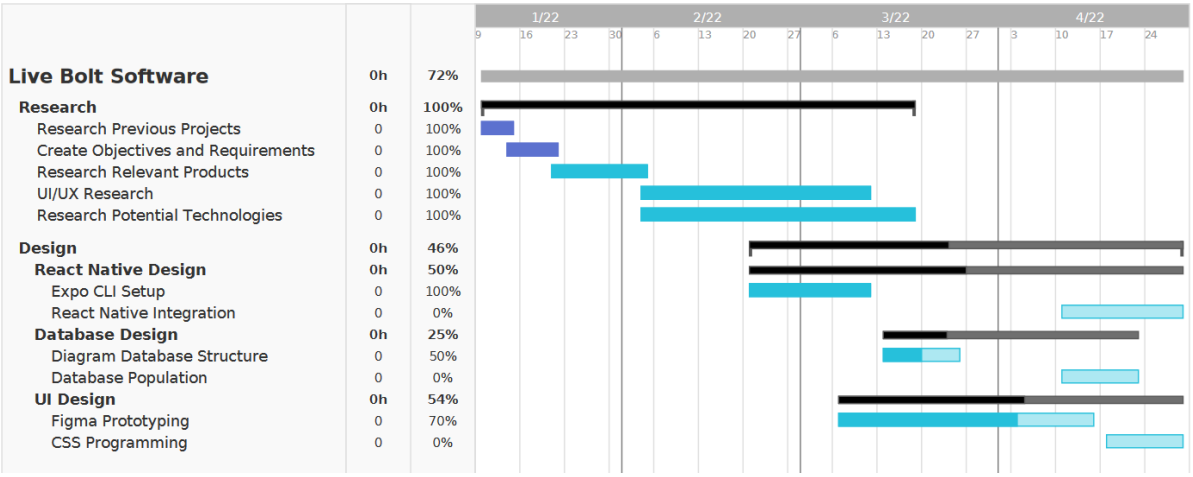

Figure 46: Software Development Gantt Chart

The Gantt chart shown in **Figure** 46 above was created using the TeamGantt website and covers all the major software planning and development tasks for the Spring 2022 semester. The timeline of this chart is from January 9th to April 29th, four months into the semester. The two categories, Research and Design, contain several smaller tasks with their completion percentages displayed as well. At the time of writing this section, all research tasks have been completed and almost half of the application's design has been done. The main tasks being worked on now are the design of the application's UI and population of test data to the application database. Most of the software planning and design were done by two members of the team, with much of the work divided evenly between them.

10.4 Hardware Development Gantt Chart

TeamGantt is the software used when developing the hardware gantt chart to outline the team's current and future progress. In **Figure** 47**,** the senior design 1 hardware development process is highlighted. The outline consists of crossing from February to May showing our preliminary research to finalizing the design for our senior design report. A couple of important mentions to the gantt chart is

that the dark blue represents finished, purple means in progress but behind, pink is present actions and light blue it future actions. This Gantt chart extends to senior design 2 to outline our goals and keep track of future progress. This gantt chart isn't a final product to represent all of hardware's necessary action. Instead, this is represented as a draft that can edited later.

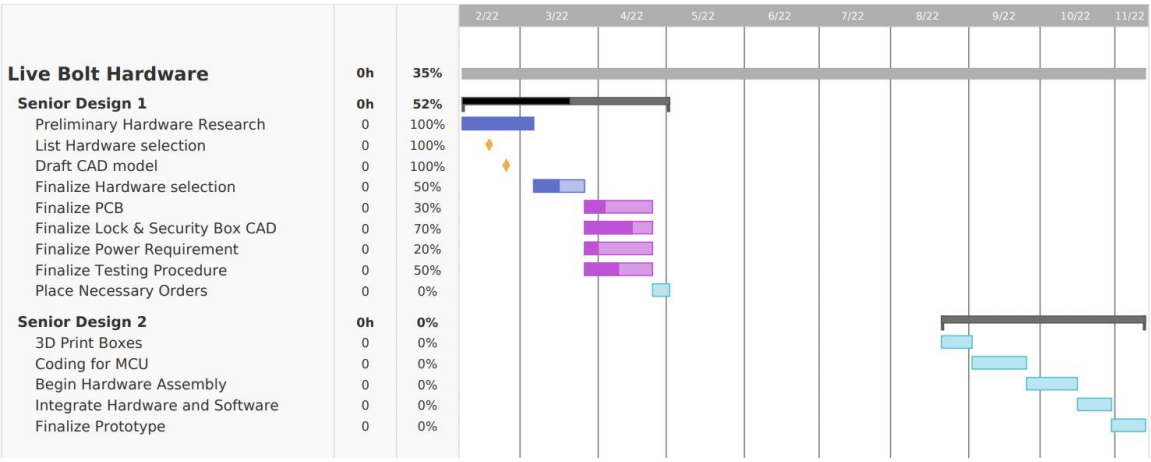

Figure 47: Hardware Development Gantt Chart

10.3 Budget and Finance Discussion

A goal is to make Live Bolt as cost-effective as possible while maintaining quality in the build and design. This is due to two main reasons: wanting to keep costs down for any hypothetical consumer who purchases the system, and lack of outside sponsorship for this project. All costs associated with Live Bolt's development are being directly funded by the four team members. .

Table 18 outlines the parts we have identified for the product and their costs. Most items are marked as 2 needed, so there will be backup components in case any malfunction. The price per single unit and total price are both listed. Items marked as $0 indicate that the team already has them, so there is no need to include those costs in the final price. The final row also provides the total cost of all components and the amount of money each member of the group will contribute.

Table 18: Bill of Materials

Part Description	Qty Needed	Price/Unit	Estimated Total Price
Raspberry Pi 4	2	$40	$80

Assorted Wires, LEDs, Buzzer, and other electrical components	-	$0	$0
Camera	2	$15	$30
Keypad	2	$0	$0
NFC Reader	2	$9	$18
IR Motion Sensor	2	$8	$16
Raspberry Pi Microphone Array	2	$29	$58
High Torque Servo	2	$20	$40
Lock Battery Holder	1	$2	$2
Security Battery holder	1	tbd	
ESP8266	2	$0	$0
PCB Printing	-	tbd	
Doorknob and lock for Testing	1	$0	$0
PLA 3D Printing Filament	1	$22	$22
Total Members	4	$66.50	$266

Development for the Live Bolt system will cost approximately $266 for two fully working products. This is within the team's original budget and is relatively cheap and cost effective. The cost would be higher to produce more however, since the team would need to purchase more components that they already had before. We would have to purchase more ESP8266s, keypads, LEDs, and buzzers, which would drive up the production cost.

Appendices
Appendix A – References

References

[1] [Online]. Available: https://magpi.raspberrypi.com/articles/raspberry-pi-4-vs-raspberry-pi-3b-plus.

[2] [Online]. Available: https://www.biometricupdate.com/202001/biometric-facial-recognition-hardware-present-in-90-of-smartphones-by-2024.

[3] [Online]. Available: https://www.wynnslocksmiths.com.au/electric-strike-magnetic-lock-difference/.

[4] [Online]. Available: https://www.hvrmagnet.com/blog/what-is-electromagnetic-lock-and-how-it-works/.

[5] [Online]. Available: https://www.frontrangelocksmith.com/blog/the-best-lock-for-your-home-is-your-lock-really-safe/.

[6] "AA battery," [Online]. Available: https://en.wikipedia.org/wiki/AA_battery.

[7] "Coding Standards and Guidelines," [Online]. Available: https://www.geeksforgeeks.org/coding-standards-and-guidelines/ .

[8] "Developer Content Policy," [Online]. Available: https://play.google.com/about/developer-content-policy/.

[9] "App Store Review Guidelines," [Online]. Available: https://developer.apple.com/app-store/review/guidelines/#performance.

[10] [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html.

[11] [Online]. Available: https://developer.android.com/guide/topics/connectivity/bluetooth.

[12] [Online]. Available: https://github.com/rusel1989/react-native-bluetooth-serial#events.

[13] [Online]. Available: https://code.visualstudio.com/docs.

[14] "Near field communication overview," [Online]. Available: https://developer.android.com/guide/topics/connectivity/nfc.

[15] "Core NFC," [Online]. Available: https://developer.apple.com/documentation/CoreNFC.

[16] K. Pattabiraman, "Circuit Basics," [Online]. Available: https://www.circuitbasics.com/how-to-set-up-a-keypad-on-an-arduino/. [Accessed 2022].

[17] "Test Environment for Software Testing," [Online]. Available: https://www.guru99.com/test-environment-software-testing.html#3.

[18] "The different types of software testing," [Online]. Available: https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing.

[19] T. B. A. Test Automation Frameworks – Why. [Online]. Available: https://www.testingxperts.com/blog/test-automation-frameworks.

[20] "Jest," [Online]. Available: https://jestjs.io/.

[21] "Mocha," [Online]. Available: https://mochajs.org/.

[22] "Mocha.js, the JavaScript test framework: A tutorial," [Online]. Available: https://blog.logrocket.com/a-quick-and-complete-guide-to-mocha-testing-d0e0ea09f09d/#whatismochajs.

[23] "Cypress," [Online]. Available: https://www.cypress.io/.

[24] "Jasmine," [Online]. Available: https://jasmine.github.io/.

[25] "Selenium," [Online]. Available: https://www.selenium.dev/.

[26] "Introduction to Expo," [Online]. Available: https://docs.expo.dev/.

[27] "Publish your app," [Online]. Available: https://support.google.com/googleplay/android-developer/answer/9859751?hl=en#zippy=%2Capp-status.

[28] "Usage statistics of JavaScript as client-side programming language on websites," [Online]. Available: https://w3techs.com/technologies/details/cp-javascript/.

[29] "Getting Started," [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX/Getting_Started.

[30] "Guidelines," [Online]. Available: https://developer.apple.com/app-store/guidelines/.

[31] "802.11 Standards Explained: 802.11ax, 802.11ac, 802.11b/g/n, 802.11a," [Online]. Available: https://www.lifewire.com/wireless-standards-802-11a-802-11b-g-n-and-802-11ac-816553.

[32] "How to Choose a Deadbolt - The Deadbolt Guide," [Online]. Available: https://www.doorware.com/specials/help-center/deadbolt-guide.cfm.

[33] "Near Field Communication Technology Standards," [Online]. Available: http://nearfieldcommunication.org/technology.html.

[34] "Bluetooth Standards and Technology," [Online]. Available: https://www.digi.com/solutions/by-technology/bluetooth-standards.

[35] "Referential Integrity and Relational Database Design," [Online]. Available: http://web.mit.edu/11.521/www/lectures/lecture10/lec_data_design.html.

[36] "Test Environment for Software Testing – A Detailed Guide," [Online]. Available: https://www.softwaretestingmaterial.com/test-environment/?utm_source=rss&utm_medium=rss&utm_campaign=test-environment.

[37] "Top 3 JavaScript Testing Frameworks with their Pros and Cons," [Online]. Available: https://www.rlogical.com/blog/top-3-javascript-testing-frameworks-with-their-pros-and-cons/.

[38] [Online]. Available: https://www.hackster.io/petewarden/recognizing-speech-with-a-raspberry-pi-50b0e6.
[39] [Online]. Available: https://stt.readthedocs.io/en/latest/index.html.

Appendix B - Datasheets (if needed)
Appendix C – Software (if needed)
Appendix D – Other