Anti-Package Theft Locking System

A Technical Report submitted to the Department of Electrical and Computer Engineering

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Jamison Stevens

Spring, 2022.

Technical Project Team Members

John Chrosniak

AJ Given

Derek Martin

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Dr. Harry C. Powell Jr., Department of Electrical and Computer Engineering

You Underestimate Our Powell Bouncer Locking System

John Chrosniak, AJ Given, Derek Martin, and Jamison Stevens

Date of Submission

Capstone Design ECE 4440 / ECE4991

Signatures

John Chrosniak

AJ Given

Derek Martin

Jamison Stevens

Statement of work:

John Chrosniak

This semester I focused on developing embedded software to control the operation of our system after a password was entered. Specifically, this included verifying if a password was valid, appropriately flagging a password as used, triggering the electric strike to unlock, starting video recording, and lastly pushing the recording to the cloud. I designed the software drivers to interface the Raspberry Pi with the camera sensor and AWS and configured the processes for controlling the above operations. In addition, I designed background processes to periodically update the password registry on the Raspberry Pi from the AWS server and upload any video footage that was recorded during a time period when the Raspberry Pi lost internet connection. Other side responsibilities included assisting with package box assembly, setting up AWS cloud storage, and helping with printed circuit board (PCB) layout.

AJ Given

My primary responsibilities included writing device layer code to handle passcodes and writing application layer code to generate and hash new passcodes. This included writing general purpose input/output code to read user input from the keypad, implementing a SHA256 hash algorithm in C++ to compare codes to the hashes retrieved from the AWS database. The hashes stored in the AWS database were generated using application layer code that I wrote in python. In addition, I wrote device layer code to operate the solenoid driver that controls the state of the electric strike.

My secondary responsibilities included helping Jamison with the layout and testing of the printed circuit board PCB. Once we finished the layout and had the board populated by WWW Electronics (3W), I worked on testing for correct operation of the solenoid driver, ideal diode controller, and both power supplies.

Derek Martin

My primary responsibility for this project was to design the web application to accompany the hardware. I selected the Python Flask framework to create the app due to ease of integration with AWS as well as flexibility to expand functionality. I fully designed the user interface with separate tabs for managing passwords and viewing video footage to reduce clutter and keep the web page user friendly. I also designed the database schema so that the data stored in AWS can be understood by the Raspberry Pi while also staying secure. To eliminate security risks involved with transferring the passcodes generated by the root user to guest users, I wrote a function taking advantage of the Flask_Mail package to send the passcode to the user directly over secure email.

Jamison Stevens

Throughout this project, my primary responsibilities were developing, integrating, and testing the hardware. The development of this circuit board started by laying out block diagrams of subcircuits and connections needed to produce the desired behavior of the locking system. Using these block diagrams, I specified the individual electrical components required for these subsystems and hardware integration: power regulation, keypad, solenoid driver, and Raspberry Pi. For the power regulation subcircuit, the team wanted to use a house's main power line as the primary source of power and a back-up battery power source. To do this, I chose a barrel jack for a 15V power supply, a 12V pack of rechargeable batteries, and an ideal diode controller to switch the system from primary to back-up power supply in case of a power outage. I chose the 15V power supply based upon the electric strike lock mechanism's 12 V supply requirement and the projected current draw of the system. The decision for the 12V battery pack is so that the batteries can recharge with the 15V supply, while being able to power the circuit for an adequate duration so that the system stays locked through a power outage. The 12V battery pack supplies 12.92V at full charge, so the rechargeable batteries allow for normal system operation for approximately 10 hours. This calculation is included in the appendix. For the keypad circuitry, I decided to add TVS diodes to all the keypad input-output lines to protect the circuit against high-voltage transients.

After selecting the components and creating schematics for the electrical hardware, I developed computer-aided design (CAD) files for the printed circuit board (PCB). This layout included the connectors for the Raspberry Pi, barrel jack power supply, rechargeable batteries power supply, and electric strike. Additionally, I created screw holes for mounting the board and smooth integration with a package box use case. After creating these PCB CAD files, I helped to submit these files and a Bill of Materials to WWW Electronics Inc. for PCB manufacturing. I picked up the manufactured PCB from WWW Electronics Inc. after completion. After which, I assisted with hardware testing to ensure the PCB behaved as intended. I also picked up package box materials from Lowe's and constructed the package box for demonstration of the locking system developed in this Capstone project. Finally, I assisted in the integration of the locking system hardware with the package box. The primary intention of my work was to supply power to the Raspberry Pi, use the output signals generated by the Raspberry Pi to control the locking mechanism, deliver keypad user inputs to the Raspberry Pi, and develop a package box testbed to demonstrate the behavior of the locking system in a potential system use case.

Table of Contents

Contents

Table of Figures

Abstract

To address recent increases in package theft, we propose an improved home security system that uses a digitally-controlled locking system offering keyless entry through a master code or limited-use password, as well as video recording for added security. The system will provide features for preventing package theft and operating short-term rentals, such as an Airbnb [1]. The owner will use the master code to unlock the system and set limited-use passwords for guests that should only have temporary access. The limited-use passwords unlock the system for either a single use or a predetermined duration of time so that package deliverers can store packages in a secure place until the owner returns. Consumers can use the device on a standard door, a pet door-equivalent for packages, or a package box, depending on the user's security preferences. This project involves the use of a microcontroller with wireless capabilities, keypad to enter and set passcodes, and an interface to control the door-locking hardware.

Background

Since the start of the Coronavirus pandemic, more individuals are opting for online shopping experiences to stay healthy, while potentially saving more time and money [2]. However, as society is starting to return to more pre-pandemic lifestyles, this has led to the unintended consequence of increased opportunities for package pirates to raid people's front porches and package rooms [3]. Additionally, with the advent of Airbnb, more people are choosing these more affordable short-term rentals with at-home amenities. This project was seen as an opportunity to prevent both package thefts and help Airbnb hosts run their rental business.

Some existing products aim to address the issue of porch package theft by deterring thieves such of the ring doorbell camera. This product works by detecting motion in the vicinity of the owners door and activates a camera and sends a notification to the owner to alert them of activity [4]. Our project is similar in that it utilizes a camera to monitor activity, but is unique in that it provides a layer of physical protection around packages and adds further security than just deterrence. The camera is different in that it activates upon code entry, rather than on motion activation. Our camera is also programmed to not record on routine entry by the owner of the house, saving resources over devices like ring that record no matter who enters.

A more similar product that exists is the Amazon Key. This product allows Amazon delivery drivers to scan packages, have the system verify that the package belongs to the address and that the driver is near the door, and the door unlocks without the driver even receiving a code [5]. This differs furthest from our project when it comes to accessibility. Amazon's product is only available to Prime members who pay a subscription fee, and the services available with the product are only available for deliveries from Amazon and approved affiliates. Amazon's product also allows you to send codes to individuals as you choose, but this requires each person entering to have the app which makes it less accessible than our product, and does not allow for dynamic updates as our product does, meaning the owner has to manually deactivate each code rather than setting it to expire on a certain date or giving access for only a limited number or single use.

Working through this project called upon skills learned in many classes throughout the electrical and computer engineering curriculum. Designing and implementing the web application draws directly on material learned in Advanced Software Development Methods. Designing the circuitry as well as performing the board layout called upon skills learned in the FUN series. Developing the firmware and programming the embedded code called directly on techniques and skills in the Embedded Computing and Robotics courses. Storing data in AWS as well as sending and receiving data from the cloud storage took advantage of skills learned from the Cloud Computing course. Finally, encrypting the passcodes and designing the entire system with security in mind leveraged material from both Introduction to Cybersecurity as well as Defense Against the Dark Arts.

Constraints

Design Constraints

The main components needed for our design included a camera for video surveillance, a keypad for entering passcodes, an electric strike to control access, a power supply connecting to a standard outlet, and a backup power supply consisting of rechargeable batteries to keep the system running in the event of a power outage. With respect to hardware, our system will require a device capable of wireless connectivity for uploading video footage to the cloud and communicating with the web application for validating passwords [6]. The device will also need to store videos locally in the event the system cannot upload a video due to internet outages. With these design requirements come a multitude of constraints that we have considered throughout development to ensure the success of our final product.

Manufacturability

The final product requires professional fabrication of a single PCB that interfaces the electric strike, keypad, and power supplies with the core device. The PCB design primarily consists of through-hole components for ease of assembly and debugging. However, a design intended for product deployment would use strictly surface-mount components to decrease the surface area of the board. The PCB layout design was constrained to IPC standards [7].

Part Availability

The current semiconductor shortage limits the selection of microcontrollers (MCUs) and microprocessors (MPUs) available for use in the security system. Furthermore, the automotive industry is facing a significant shortage in part availability [8]. Given that many MCUs and MPUs that natively support wireless connectivity are used in vehicles, we had to carefully consider the feasibility of finding a device that fits our project's needs [9]. To mitigate this concern, we have decided to use a Raspberry Pi 3B v1.2 because of its availability in the classroom from previous semesters and ability to support wireless communication and local video storage. The electric strike used as our locking mechanism is also considered an automotive part, but we were able to obtain a functioning electric strike from previous years as well. Various electronic components experienced shortages throughout the semester, limiting our choices for various integrated circuits (ICs) and components rated for higher currents.

Consequently, many of the components selected for fabrication were picked based upon availability.

Economic and Cost Constraints

The project budget was set at $500 per team to cover all expenses. This helped cover the cost of the keypad, PCB manufacturing and assembly, and package box construction materials, as all other components present in the system were taken out of the Fundamentals of Electrical Engineering series lab kit or reused from previous Capstone projects.

Due to the $500 budget constraint, there were many attempts to reduce system prototype costs. For instance, the Raspberry Pi is the property of John Chrosniak, one of the Capstone team members. This led to a cost savings of roughly $35 [6]. This reduced the number of GPIO pins on the system compared to a Texas Instruments MSP430 or similar product, which has 64 GPIO pins. This prevented us from using a single-pull keypad, which would require significantly more GPIO pin to function but would be more accurate than a matrix-encoded keypad. In selecting many of the components and buying materials, the cheapest viable option was chosen. As a result, the package box was built by the team, rather than purchasing a pre-built box, smaller battery holsters were used instead of one large 10-AA battery housing, and a cheap keypad was used. Buying a pre-built package box would have made the final system prototype look more professional and would have led to more seamless hardware integration. A more expensive keypad would have allowed the system to receive user inputs more accurately and have allowed for a more professional electrical connection.

Given a larger budget, the team would have considered upgrading various components in this system to create a better and more professional-looking system prototype.

External Standards

The following external standards apply to this project.

1. IEEE 802.11 - These standards provide the basis for wireless network protocols using Wi-Fi. This standard is followed by most home and office networks to allow laptops, printers, and other wireless devices to communicate and access the Internet without a wired connection [10]. Since the security system utilizes wireless communications between the Raspberry Pi and the cloud storage, this standard must be followed to ensure the system can run on any consumer's Wi-Fi network.
2. IPC standards for PCB design - IPC sets standards for file formats and design software when designing PCBs [11]. IPC also sets standards for soldering and manufacturing. These standards are important to follow since this project required sending board designs to third party companies to manufacture and solder components. In order to make sure the product received is the same as the desired product, all files must be produced in the same way the manufacturer expects. It is also important to select a company that follows the soldering standards to perform the soldering to make sure all components are properly connected and safely attached.
3. NEMA standards for Mechanical Casings – NEMA Type 3 standards apply to enclosures for indoor or outdoor use to provide a degree of protection against rain, sleet, and wind-blown dust [12]. This is extremely important to follow as the keypad and locking mechanism are slightly exposed and need to be protected from the

elements. Failure to do so could result in the system malfunctioning or components getting damaged. In a commercial application this would need to be followed, but including this in the demonstration was not within the scope of the project and did not enhance the demonstration of the hardware or software in any way.

4. SMD (surface mount devices) such as chip resistors and chip diodes are manufactured in standard sizes defined by the SMT (surface mount technology) industry standards. JEDEC is the leading [13] standardization group, which simplified the PCB design since footprints are already available for these devices. The 0805 and 0201 footprints were used in this design.
5. I2C for Camera Serial Interface - I2C provides an interface to transmit data bidirectionally between devises [14]. In order to transport the video files from the camera to the Raspberry Pi, these standards must be followed. This will allow the Pi to have control over the camera and determine when and what data is being sent to avoid any corrupted data packets.
6. HTTP Standards for the web application - HTTP standards define the responses and data that should be loaded in a web browser when a request is made [15]. These standards influenced how the web application was written and how it should handle various user inputs included different unexpected requests and errors.
7. NEMA Standards for Electrical Connections - These standards define which types of connections are suitable to connect to AC power outlets including those commonly found in houses [12]. This standard influenced which component could be selected to power the device and connect the circuit board to the AC line of the house of the user.
8. Barr C Coding Standard - The Barr Coding standard outlines standards to reduce the number of defects in embedded software [16]. These standards helped ensure the code was safe and prevented us from including avoidable bugs. Some of these standards that were most closely followed included using proper variable and file naming convention as well as thoroughly documenting and commenting code.

Tools Employed

KiCAD [17] was used to design and test the circuits as well as route and lay out the PCB. Since only one group member had used this tool before, this required a significant amount of learning and improving upon throughout the project. NI Virtual Bench [18] was used to test the PCB and hardware. This involved measuring resistance values, measuring voltages across various test points, and measuring the current draw throughout our circuit. Since this tool was used through the FUN series this tool did not need to be learned or improved upon. AWS S3 was used as a cloud storage solution for video files and passcode information [19]. This tool was already learned in the Cloud Computing course and skill in using it was improved slightly in working through the project. VS Code was used as the text editor for this project. This text editor has been used extensively prior to this project and did not need to be learned [20].

Ethical, Social, and Economic Concerns

Internet of Things (IoT) devices have received scrutiny in recent years due to notable security issues that put users at risk [21]. Since these devices connect to a user's personal network, devices not designed with security in mind leave the potential for cyberattacks that compromise personal data. The severity of these issues will continue to grow as more consumers and industries begin to adopt IoT devices, such as healthcare, banking, and home security [22]. Thus, the final product was designed considering the following sociotechnical implications.

Security Concerns

The final product was designed with security at the forefront of every design decision that could potentially compromise the system. To prevent the possibility of leaked passwords, a cryptographically secure hash function was used to convert user-generated passwords into byte strings for information transfer between the device and web application through cloud storage [23]. The same hash function was used to convert passwords entered through the keypad into byte strings to evaluate if the entered password was valid. Additionally, a backup power supply powered by rechargeable batteries was added to keep the system online in the event of a power outage. Assuming the batteries were fully charged before the outage, this would power the system for approximately ten hours before requiring new batteries. Finally, in order to defend against SQL injection attacks and other potentially malicious inputs, all database queries were written to avoid using the input directly [24].

Even with these added features, there are still two potential sources of security concerns within the final design. First, the database that feeds the web application stores passwords to present this information to the user, which could be accessed by unwanted parties if not protected properly. Second, all passwords are sent to guests via email, which inherently poses the risk of unwanted parties receiving access to valid passwords.

Privacy Concerns

Since the system will be recording footage of the user's private residence, the storage and protection of this information heavily influenced the design process of the system. Amazon Web Services (AWS) S3 storage service was used for keeping track of a user's video footage and valid passwords, with encryption services enabled for added security [25, p. 3]. All requests to access or modify information in the cloud are sent through the official AWS SDK, which uses HTTPS requests for information transfer. Using the Flask framework [26] also provides tools for configuring a secure web application. To mitigate any privacy concerns regarding personal data, the application interface grants users full control over all their personal data, providing them with the ability to delete videos and passwords, as well as modify passwords if they feel their personal password has been compromised.

Environmental Impact

The batteries that power the system pose the greatest potential for having a negative environmental impact. The toxic chemicals within batteries can, if disposed of improperly, decay and leak into the surrounding environment causing both water and air pollution [27]. To mitigate this potentially harmful impact, we have decided to design the system to receive power primarily from the electrical grid of the residence and only use battery power as a backup. This design decision will greatly increase the life of the batteries and thus reduce the environmental impact of battery waste. Rechargeable batteries were used to further decrease the environmental impacts of batteries over time, as the batteries will continuously charge if the system is connected to the main power supply. Another environmental impact of the system stems from the process of manufacturing circuit boards, but this impact cannot be avoided since the system depends upon having a PCB to interface with the keypad and regulate power [28].

Sustainability

To make the security system more sustainable, the system was designed such that it minimizes power consumption whenever not in use [29]. All the system's peripheral devices operate in low power mode whenever the door remains locked and return to lower power mode after the door has been unlocked for more than twenty seconds. One flaw of the system relates to the electric strike requiring constant current to remain locked. This will consistently consume power as the system remains in its idle state, increasing the amount of power required and decreasing the lifespan of the backup battery supply. An electric strike with reversed logic would be used for this system if deployed, but the lack of available parts prevented the inclusion of this type of strike in this iteration. Given more time to develop the project, a microcontroller that consumes less power could be used in place of the Raspberry Pi to control the functionality of the system as well.

Economic Concerns

The largest anticipated expenses for this project are the electric strike, camera, and Raspberry Pi. If the system were to go into production, it may also require the user purchase a secure lockbox or pay for replacing their current door lock with an electric locking mechanism. To limit costs, a simpler microcontroller with wireless connectivity capabilities could be used in place of the Raspberry Pi [6]. Furthermore, the cost of cameras can greatly vary depending on resolution, frame rate, and other performance factors. We have decided to use a cheaper camera model to make our system more affordable. Since the homeowner should already know who is entering the house based on the passwords entered, our system does not need a camera with the highest levels of resolution to identify guests.

Some economic concerns within the project include the use of cloud storage, as this will require recurring payments that will increase the overall cost of using the system [30]. In addition, the overall cost of the product could be improved in deployment by transitioning from the Raspberry Pi to a cheaper microcontroller.

Intellectual Property Issues

"Video Recording Triggered by a Smart Lock Device" is a U.S. patent that covers a method of recording videos based upon an intelligent locking system [31]. The patent's first independent claim discusses the use of a motion detector: "At least one processor configured to: in response to the motion detector detecting motion of a person near the door at the building, instruct the camera to exit lower-power mode." The Capstone project presented toggles between low- and high-power modes based upon a user input of a valid one-time use password. Considering the main independent claim of [31], this Capstone project remains patentable. Additional independent and dependent claims in this patent further specify the components of this system including: the sensors, processor, and camera. However, the use of a motion detector for triggering video recording does not limit the patentability of the presented Capstone project.

"Security System" is a U.S. patent that covers a method for implementing access control with a wireless communication device [32]. The patent's first independent claim discusses: "Enabling or activating a protected function, said method comprising: storing an authorization code in a wireless communication device; transmitting an access request from said wireless communication device to an access control device; receiving an authentication challenge from said access control device at said wireless communication device in response to said access request; computing an authentication response based on said authentication challenge and said authorization code; and transmitting said authentication response from said wireless communication device to said access control device." This claim covers the work done in the presented Capstone project. Although most of the following independent and dependent claims also cover this Capstone project, in independent claim 60, this patent further specifies the access control device by stating it has a wireless transceiver and processor. In the case of the Capstone project presented, the access control device is the electric strike and the protected function is the lock status of the system. Since the electric strike used in this project has neither a wireless transceiver nor wireless processor, this patent does not limit the patentability of this Capstone project.

"Networked and Camera Enabled Locking Devices" is a U.S. patent that covers a networked lock box system with an integrated camera device [33]. The patent's first independent claim discusses returning the system to a locked state: "Automatically lock the electronic locking mechanism after unlocking the locking mechanism in response to the lid not being opened after a period of time greater than a threshold period of time." While the Capstone project presented automatically locks after a specific time threshold, this Capstone project also returns to a lock state at a user input, which is not mentioned in the claims of this patent. Additionally, this patent claims that the system's processing circuit returns to a low power sleep mode without the presence of user interaction and periodically wakes up to communication with a server for receiving up dates. For the system developed for this Capstone project, the processing circuit does not enter a low power mode and continuously receives updates from a server. Due to these differences, this patent does not limit the patentability of this Capstone project.

Detailed Technical Description of Project

This project aims to deliver an effective solution to the problem of porch package theft and a technology that facilitates short-term rentals. To accomplish this, a Raspberry Pi was used to seamlessly communicate with the cloud, a web application user interface, a keypad for passcode inputs, and an electric strike for access control. A system-level block diagram is shown below.

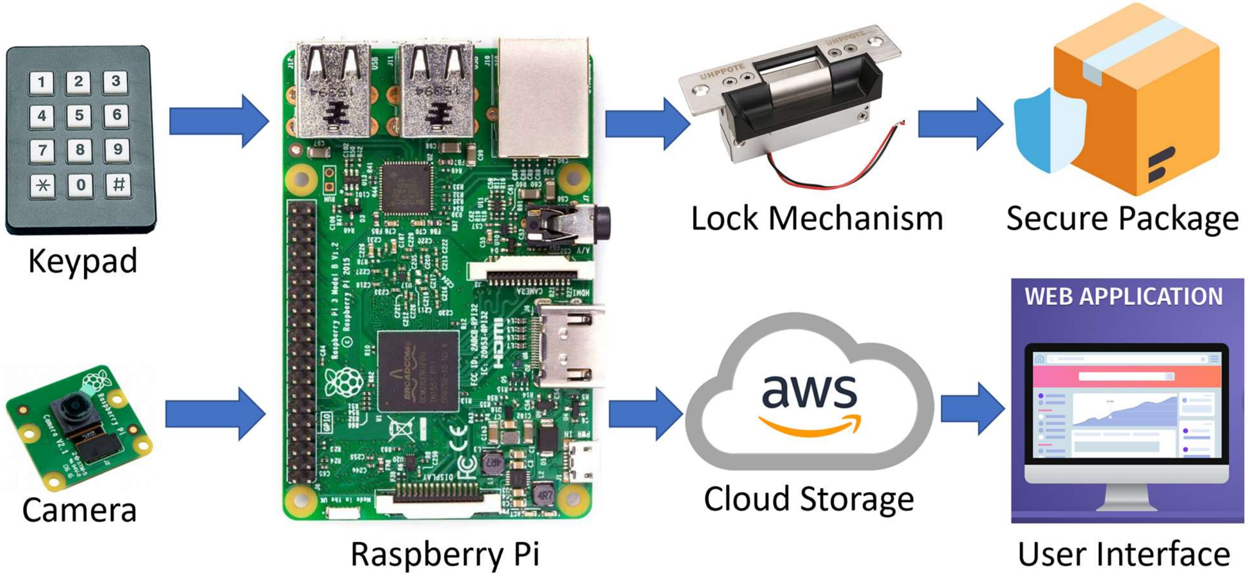


Figure 1 System Block Diagram

Each of the parts shown in the System Block Diagram in Figure 1 will be described in the Detailed Technical Description of Project section. The technical description of this project has been organized with the following sections:

1. Hardware
   a. Block Diagram
   b. Power Supply Circuitry
   c. Voltage Regulator Circuitry
   d. Solenoid Driver Circuitry
   e. Connectors Circuitry
   f. Subcircuit Schematics
   g. PCB Layout
   h. Problems and Design Modifications
2. Software
   a. Device Layer
      i. Keypad Driver
      ii. SHA256 Hash Function
      iii. Password Manager
      iv. Electric Strike Driver
      v. Camera Driver
      vi. Video Manager
      vii. Preempt RT Kernel Patch
   b. Cloud Storage Layer
      i. Device Communication
      ii. Application Communication
   c. Application Layer

Hardware

At the heart of this project is a Raspberry Pi 3 Model B Version 1.2 [6]. For the hardware portion of this project, a PCB was designed to produce the desired system behavior by allowing the Raspberry Pi to interface with the other requisite electrical components: a wall-plug power supply, a rechargeable battery power supply, a keypad, and an electric strike locking mechanism.

Block Diagram

After determining the problem that this project aimed to solve, the desired behavior of the system was determined. Given this intended behavior a block diagram was created to accomplish these behavioral tasks. A hardware block diagram is shown below.

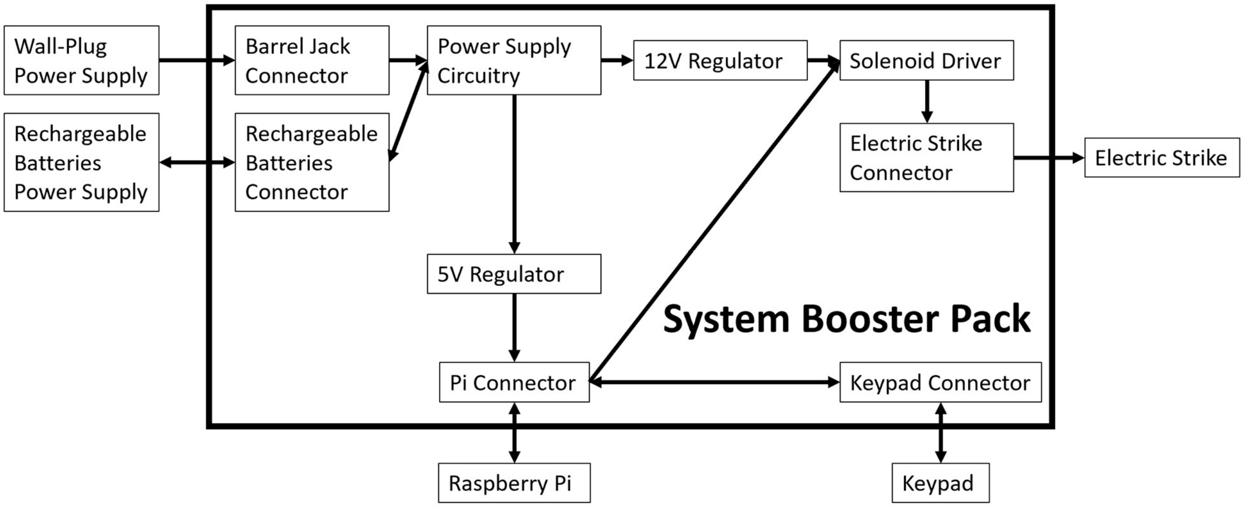


Figure 2 Hardware Block Diagram

Figure 2 displays a block diagram of the hardware portion of this project. Everything inside of the large box with a black border is in the system booster pack, which is the PCB that was designed and manufactured for this project. The arrows on the diagram show the direction in which signals will flow between blocks. This block diagram in Figure 2 was the starting point for hardware development for this project. Each of the following sections will elaborate on the design decisions for each block, describe the component selection process, and display the final schematics and PCB layout. Additionally, problems and design modifications will be discussed at the end of this section.

Power Supply Circuitry

There are main and backup power supplies for the Bouncer Locking System. It was deemed necessary to have both for the system's intended use cases: a door, package slot, or package box. Although a house's main power line is fairly reliable, in the case of a power outage the system should remain locked until the main line's power can be restored. As a result, the power supply circuitry must have the ability to support the power consumption needs of the system through both a main line power supply and a backup power supply.

Given the block diagrama in Figure 1 and Figure 2, the next step in designing the power supply circuitry was to develop a system power budget. It was found that the majority of current consumption comes from the Raspberry Pi and Electric Strike. During the design phase, the Raspberry Pi current consumption was conservatively estimated to be 260 mA when powered at 5V [34]. Additionally, the current draw of the Electric Strike was measured to be 156 mA when powered by 12VDC with the use of a National Instruments VirtualBench [18]. Based upon this information, the supplied voltage requirement of over 12V and supplied current requirement of at least 416 mA were found.

With these requirements in mind, battery backup power supplies were researched, and a basic design was found online [35]. The basic circuit backup power supply circuit design is shown in Figure 3. Using this general circuit layout, the necessary electrical components were determined: a wall-plug power supply, a rechargeable batteries power supply (BT1), an ideal diode controller (D2), a resistor, and a rectifier diode (D1).

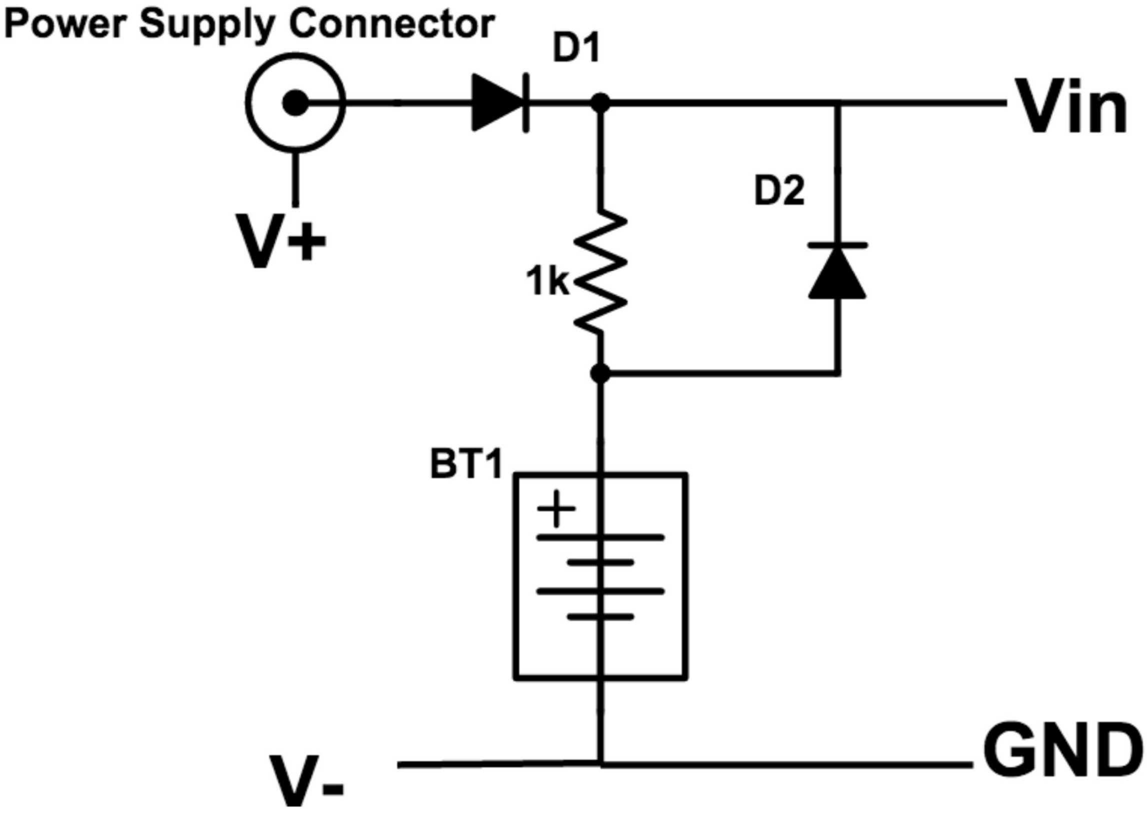


Figure 3 Basic Batter Backup Power Supply Circuit Architecture

The wall-plug power supply must deliver a higher voltage than the rechargeable batteries so that the circuit will recharge these batteries. A 15V wall-plug power supply that can supply up to 1.67 A was selected [36]. This power supply satisfies the voltage and current requirements of the locking system, while giving a 3V window for the rechargeable battery supply voltage (between 12V and 15V).

The initial circuit design incorporated 12, 1.2V rechargeable AA batteries, which would theoretically supply 14.4V and enough current to power the locking system. This assumption was used to determine the proper resistor value for the circuit architecture. The purpose of this resistor is to control the amount of current used to recharge the batteries, which determines the amount of time it takes the batteries to recharge. It is safe to recharge NiMH batteries at a rate of 10% of their capacity per hour, but it is safer to recharge these batteries at a rate of 0.3% of their capacity per hour to prevent damaging fully-charged batteries [35]. The chosen batteries had a capacity of 2,800 mAh [37]. The lifetime of the rechargeable batteries backup power supply is calculated to be roughly 10 hours; these calculations are shown in the appendix. Given the capacity of 2,800 mAh, the minimum resistance value was found.

$$I_{max,safe} = (Battery\ Capacity) * \left(\frac{1}{300} hour^{-1}\right) = \frac{2800\ mAh}{300\ h} = 9.333\ mA$$

$$V = I_{max,safe} R_{min,safe}$$

$$R_{min,safe} = \frac{V}{I_{max,safe}} = \frac{15V - 14.4V}{9.333 mA} = 64.286\Omega$$

When selecting a resistor for safe recharging, the minimum resistance value is 64.286Ω. A 110Ω resistor was chosen to have a buffer without slowing the recharge time by much. For the specific application of a locking system connected to a house's main supply line, the rechargeable batteries will be fully charged most of the time and proper operation is critical in the case of a power outage. Thus, this additional buffer seemed necessary.

The rectifying diode (D1 in Figure 3) was used to prevent current coming from the rechargeable batteries into the wall-plug power supply. A rectifying diode was selected that can handle the voltage and current requirements of the system.

The ideal diode controller (D2 in Figure 3) is used to switch from the wall-plug power supply to the rechargeable batteries power supply in the case of a power outage. An ideal diode controller was selected that can handle the voltage and current requirements of the system [38]. An ideal diode controller is a useful components for diode-OR setups, where there is power supply redundancy to handle loss of the main power supply [39], [40].

To incorporate the ideal diode controller into the power supply circuitry, the typical reverse input protection architecture listed in the datasheet was used [38]. The given architecture is shown in Figure 4. This includes placing a 100nF capacitor on the INTVcc pin and a MOSFET between the source, gate, and output pins. The MOSFET used was recommended in the datasheet, which can handle the voltage and current requirements of the locking system [41].

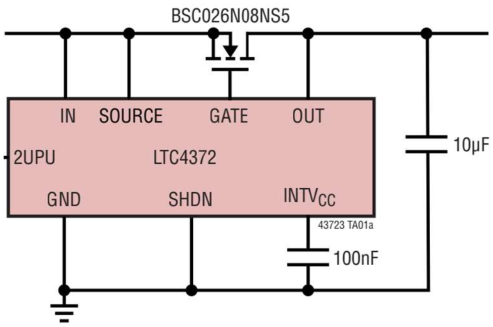


Figure 4 Ideal Diode Controller Reverse Input Protection Architecture

All other components in the power supply circuitry were taken from the Fundamentals of Electrical Engineering course series lab kit or the Capstone room, since there were many leftover components from previous Capstone projects.

Upon selecting all of the components for the power supply circuitry, a KiCAD schematic was developed with all the components and requisite connections to realize the power supply for this system [17]. A schematic of the power supply circuitry is shown below.

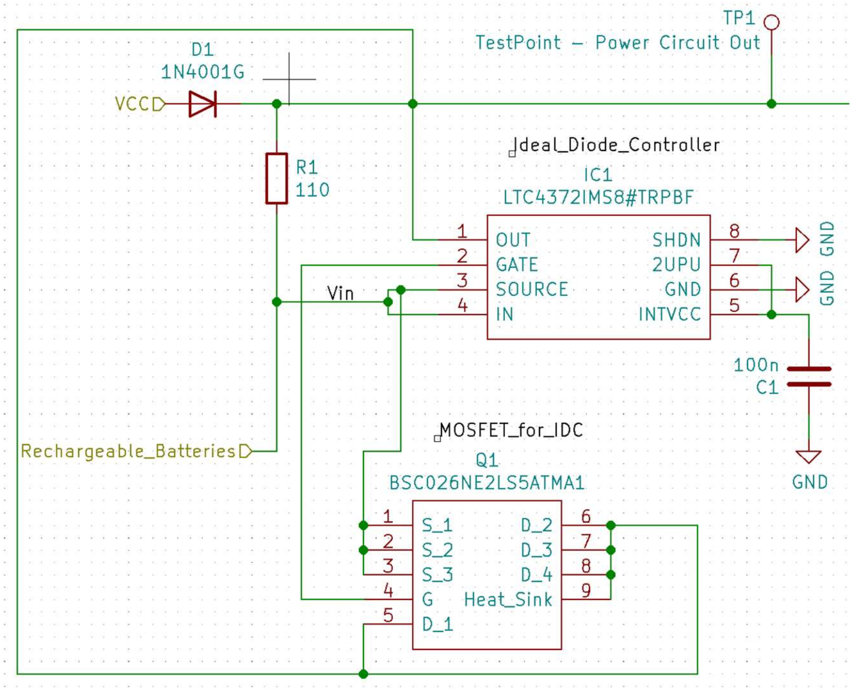


Figure 5 KiCAD Schematic of Power Supply Circuitry

Figure 5 shows the completed schematic of the power supply circuitry with connections between the selected components. In this schematic, VCC is the input from the main, wall-plug power supply, and Rechargeable_Batteries is the input from the rechargeable batteries backup power supply. TP1 is a testpoint to check the output voltage from the power supply circuitry. To design for test, testpoints were placed at the critical testing points in the circuit to check that the intended voltages are delivered to various parts of the circuit. Although it is not displayed on this schematic, testpoints were placed at VCC and Rechargeable_Batteries to check the voltages received by the barrel jack and rechargeable battery connectors, respectively. Additionally, ground testpoints were placed around the PCB to ensure ease of voltage measurements.

In this schematic, D1 corresponds to D1 in the Figure 3 power supply architecture, and IC1 corresponds to D2 in the Figure 3 power supply architecture. R1 corresponds to the resistor in the Figure 3 power supply architecture, and Q1 is the recommended MOSFET for use with the selected ideal diode controller [38], [41].

Voltage Regulator Circuitry

To accomplish the desired functionality of this system, the Raspberry Pi requires a 5V input power. Since the power supply circuitry delivers roughly 15V to the system under normal conditions and roughly 14.4V to the system when running off the backup battery supply, a voltage regulator is required to supply a steady 5V input to the Raspberry Pi. A voltage drop of 9.4V to 10V is high, so the selected voltage regulator must be able to handle large voltage drops along with the current requirement of the Raspberry Pi.

Due to the relatively large voltage drop required, linear and switching voltage regulators were investigated to determine which was better for this application. It was found that—compared to switching voltage regulators—linear voltage regulators are simpler but are more inefficient and cannot handle large input voltage to output voltage ratios [42], [43]. For these reasons, a switching voltage regulator was selected to supply the 5V required to power the Raspberry Pi [44]. The selected 5V switching regulator was chosen by looking for in-stock components that can handle the voltage drop requirement (9.4 V to 10V) and Raspberry Pi current requirement (at least 260 mA) [34].

To incorporate the chosen 5V switching regulator into the booster pack circuit design, the datasheet was referenced [44]. The datasheet gave architectures for the typical operating circuit and a 5V step-down circuit. These architectures are shown in Figure 6 and Figure 7, respectively.

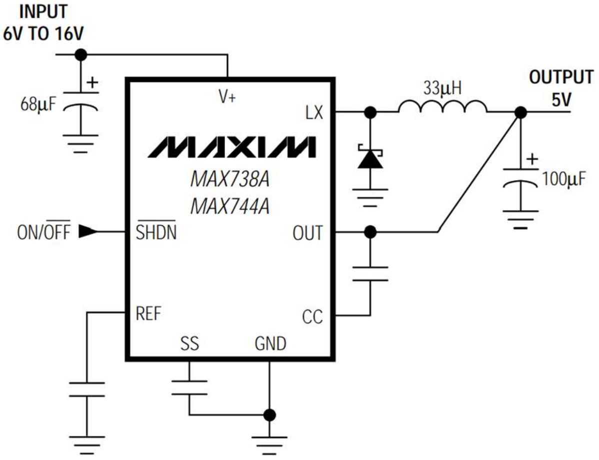


Figure 6 5V Switching Regulator Typical Operating Circuit

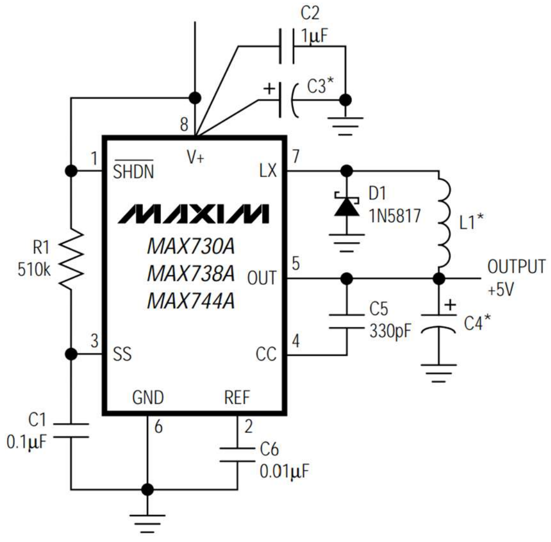


Figure 7 +5V Step-Down Switching Regulator Circuit

Figure 6 and Figure 7 give a general idea of how to implement this circuit in the PCB design. All component values given in Figure 7 were used in the incorporation of this 5V switching regulator into the PCB design. C3 in Figure 7 was given in the datasheet as 68µF, C4 in Figure 7 was given in circuit shown in Figure 6 to be 100 µF, and L1 in Figure 7 was given a range of 33µH to 100µH.

Unless a component was not in-stock, all component recommendations in the datasheet were used in the application of this 5V switching regulator. All component selections that were outside of the component recommendations were determined based upon the voltage requirements of the system. The chosen rectifier diode was a surface mount diode with similar characteristics to the suggested diode [45]. The chosen inductor had similar characteristics to the suggested inductors [46]. The chosen capacitor that corresponds to C2 in Figure 7 fulfills all of the voltage and current requirements of the circuit [47].

While using a switching regulator was deemed appropriate for supplying 5V, a linear regulator was chosen for the 12V supply to the electric strike [48]. Since available 12V linear regulators can easily handle the voltage drop from 15V or 14.4V to 12V, the complexity tradeoff between linear and switching regulators the main concern [42], [43]. Since it was not valuable to have a more efficient regulator for this relatively small voltage drop, the linear regulator was better suited for the 12V supply as it is simple to incorporate into a circuit relative to a switching regulator.

To implement the selected 12V Linear Regulator into the PCB design, the regulator's datasheet was referenced [48]. The datasheet contained a simple layout for a DC Characteristic circuit that supplies 12VDC, which corresponds to the voltage requirement of the electric strike. The layout of the DC Characteristic circuit is shown below.

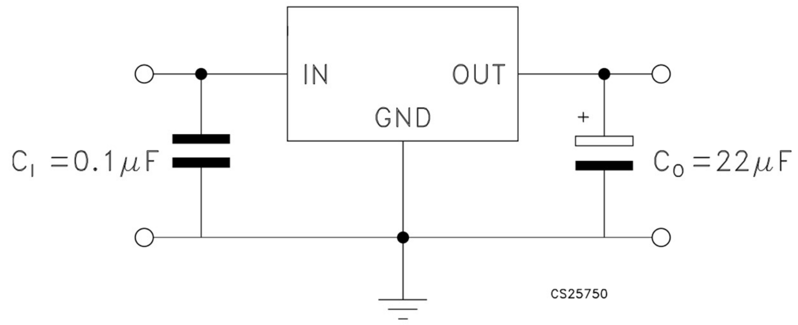


Figure 8 12V Linear Voltage Regulator DC Characteristic Circuit

Figure 8 displays the given DC Characteristic circuit layout from the selected 12V Linear Regulator's datasheet [48]. The selected capacitors matched the given values in Figure 8 and satisfy the voltage and current requirements of the system.

All other components in the voltage regulator circuitry were taken from the Fundamentals of Electrical Engineering course series lab kit or the Capstone room, since there were many leftover components from previous Capstone projects.

Upon selecting all of the components for the voltage regulator circuitry, a KiCAD schematic was developed with all the components and requisite connections to realize the desired supply voltages for this system: 5V for the Raspberry Pi and 12V for the electric strike [17]. A schematic of the voltage regulator circuitry is shown below.

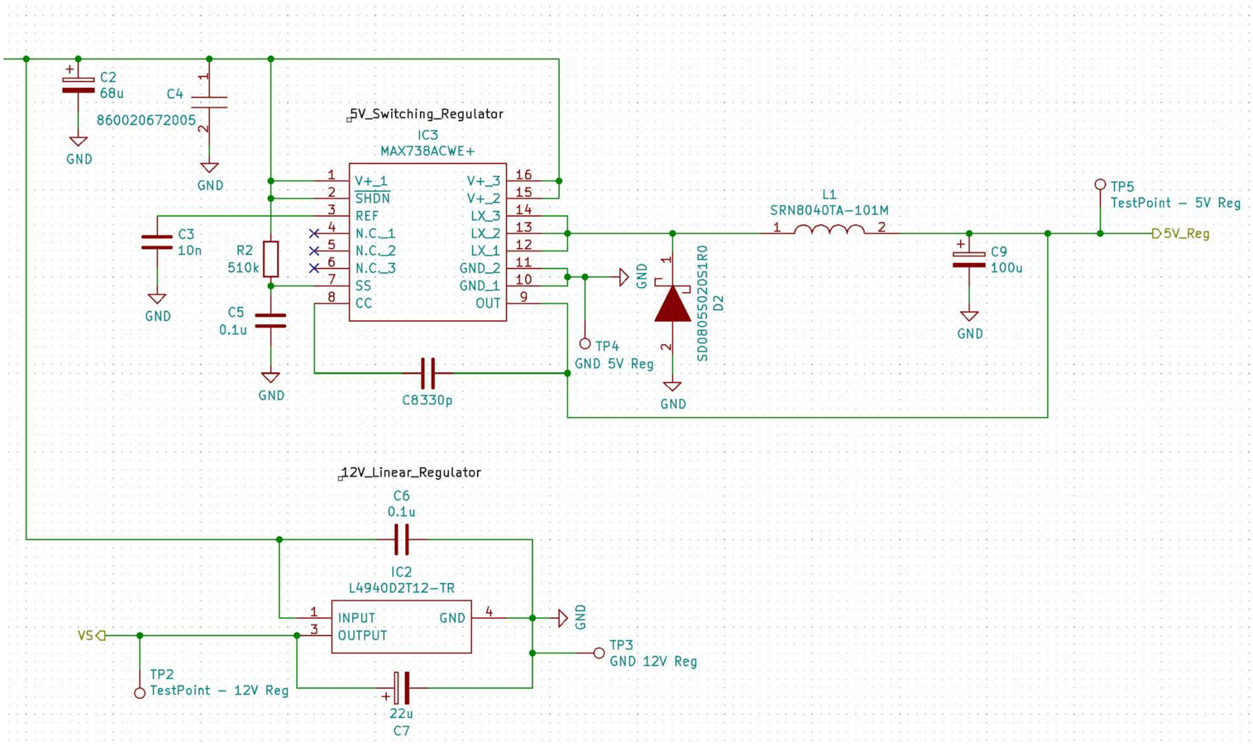


Figure 9 KiCAD Schematic of Voltage Regulator Circuitry

Figure 9 shows the completed schematic of the voltage regulator circuitry with connections between the selected components. In this schematic, the input to the voltage regulator circuitry is on the left side of the figure; it is electrically connected to 12V Linear Regulator INPUT terminal, positive lead of the C2 capacitor, and C4. This input corresponds to the output of the power supply circuitry. The 5V_Reg output connects to the Raspberry Pi Connector, which delivers the 5V supply to the Raspberry Pi. The VS output connects to the solenoid driver circuitry and electric strike connector, which controls the electric strike. The 5V switching regulator and 12V linear regulator are connected in parallel to power the Raspberry Pi and electric strike simultaneously. Additionally, testpoints were placed at the outputs of both voltage regulators to ensure ease of voltage measurements. These measurements are critical to determine that the expected voltages are applied to the Raspberry Pi and electric strike. If either the Raspberry Pi or electric strike do not receive the proper voltages, then the Bouncer Locking system will not perform its intended functions. Additionally, if too high of a voltage is applied, then there is a risk of damaging the Raspberry Pi or electric strike

For the 5V switching regulator in Figure 9, C2 corresponds C3 in Figure 7, C3 corresponds to C1 in Figure 7, C4 corresponds to C2 in Figure 7, R2 corresponds to R1 in Figure 7, C5 corresponds to C8 in Figure 7, C8 corresponds to C5 in Figure 7, D2 corresponds to D1 in Figure 7, L1 corresponds to L1 in Figure 7, and C9 corresponds to C4 in Figure 7.

For the 12V linear regulator in Figure 9, C6 corresponds to C1 in Figure 8, and C7 corresponds to C0 in Figure 8.

Solenoid Driver Circuitry

Since the Bouncer Locking System uses an solenoid-based electric strike as a means of access control, solenoid driver circuitry was investigated. It was found that the addition of a solenoid driver integrated circuit can reduce the system’s energy consumption while offering increased circuit protection [49], [50]. Given this information, it was determined that it is necessary to incorporate a solenoid driver into the PCB design for this system.

In searching for a potential solenoid driver for this application, the primary considerations were component availability, voltage limits, current limits, and circuit protections. After searching through available components that can handle the voltage and current requirements of the system, the Texas Instruments DRV103U/2K5 solenoid driver integrated circuit was selected [51]. This solenoid driver component offers current and thermal protections and allows for the incorporation of an LED that indicates proper operation. This LED is beneficial in design for test efforts as it offer a visual indication of proper operation.

After determining the solenoid driver integrated circuit to use in the PCB design, the solenoid driver’s datasheet was referenced to figure out how to properly integrate this component into the PCB design [51]. The datasheet offers several general implementations for the solenoid driver. The implementations that were referenced in the component selection process are the general layout, Time Delay Relay Driver, and Remotely Operated Solenoid Relay layouts. These layouts are shown below.

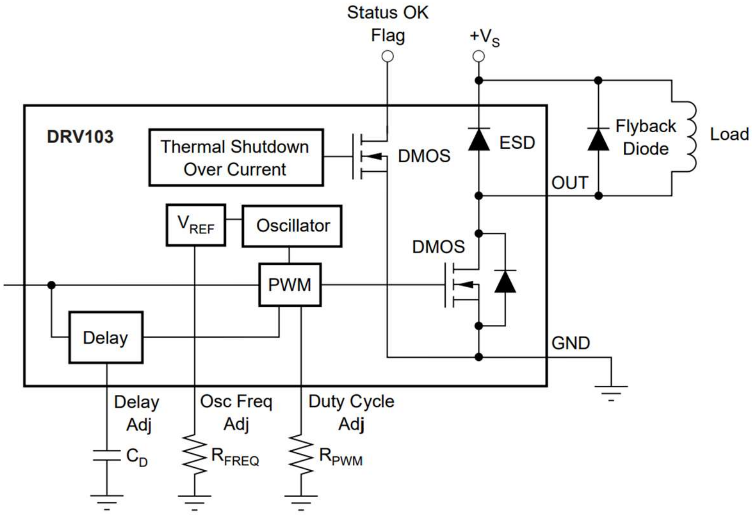
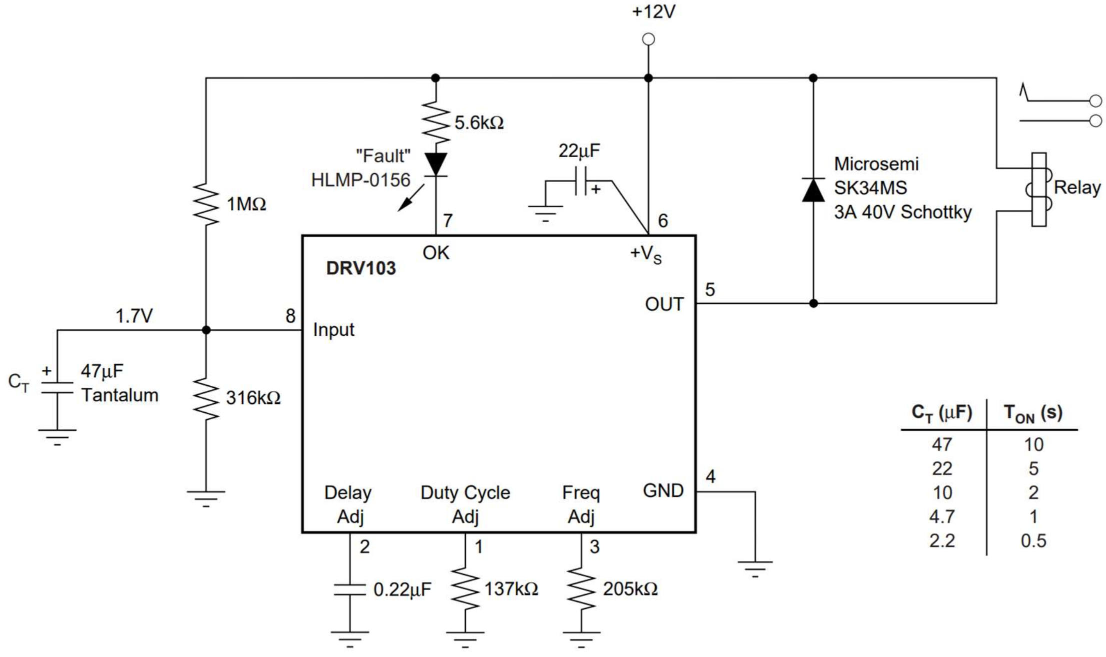


Figure 10 General Circuit Layout

Figure 10 shows a general layout of the circuitry inside and outside of the solenoid driver in a typical implementation. This information was drawn upon to better understand the workings and critical control components of the solenoid driver.



C_T (µF)	T_{ON} (s)
47	10
22	5
10	2
4.7	1
2.2	0.5

Figure 11 Time Delay Relay Circuit Layout

Figure 11 shows the layout for a time delay relay circuit. For the application of the Bouncer Locking System, there does not need to be a time delay, so most of the component values in this layout were not used in the design process. However, the resistor value of 5.6kΩ was used to select a resistor to go in series with the status indicator LED, as this was not included in the remotely operated solenoid relay circuit layout shown in Figure 12. This resistance value was also calculated to ensure intended behavior.

According to the solenoid driver's datasheet, approximately 2mA of current should flow through the LED branch and into the status pin on the component [51]. Given a supply voltage of 12V from the 12V linear regulator, the resistance value required to achieve a 2mA current can be calculated.

$$R_{status} = \frac{Supply\ Voltage}{Intended\ Current} = \frac{12V}{2mA} = 6k\Omega$$

These calculations neglect resistance from the LED, so a slightly lower resistance should be used to achieve a 2mA current. Since a 5.6kΩ resistor is a common value and was included in the University of Virginia Fundamentals of Electrical Engineering course series lab kit, this resistance value was chosen for the status indicator branch.

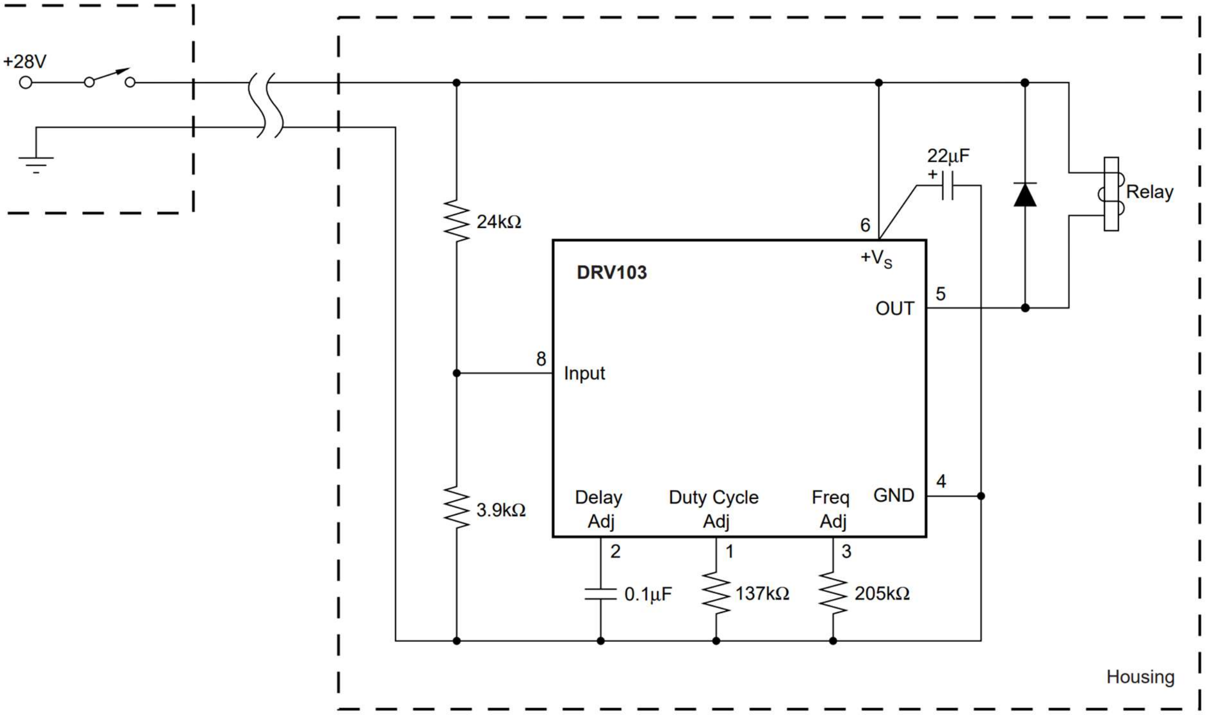


Figure 12 Remotely Operated Solenoid Relay Layout

Figure 12 shows the circuit layout for a remotely operated solenoid relay. All of the component values from this circuit layout were used in the selection of control components for the Bouncer Locking System's solenoid driver implementation, except for the 24kΩ and 3.9kΩ resistors that form a voltage divider. These two resistors may be neglected in the Bouncer Locking System application because the input signal is not a 28V switch supply as shown in Figure 12. These resistors are used to bring the maximum input signal down to a voltage of 3.914V, which is within the acceptable logical high input voltage range of 2.2V to 5.5V based upon the datasheet [51]. Since the input signal came from the Raspberry Pi GPIO in the Bouncer

Locking System application, the logical high output voltage is 3.3V, which is already in the acceptable input voltage range. As a result, these resistors are not needed.

All selected components were chosen based upon their adherence to the current and voltage requirements of the system. In this case, the current requirement was the electric strike current draw, which was measured at 156 mA with a National Instruments VitrualBench [18]. The selected flyback diode is the same as the one used for the switching regulator, since it fits with the system requirements and the datasheet-suggested diode was out-of-stock [45]. The 205kΩ frequency adjust resistor was selected because it was not found in the Fundamentals of Electrical Engineering Course Series lab kit or in the Capstone room. All other components for the solenoid driver circuitry were taken from the Fundamentals of Electrical Engineering Course Series lab kit or the Capstone room, which had an abundance of components leftover from previous Capstone projects.

Upon selecting all of the components for the solenoid driver circuitry, a KiCAD schematic was developed with all the components and requisite connections to realize the desired solenoid driver operation [17]. A schematic of the solenoid driver circuitry is shown below.

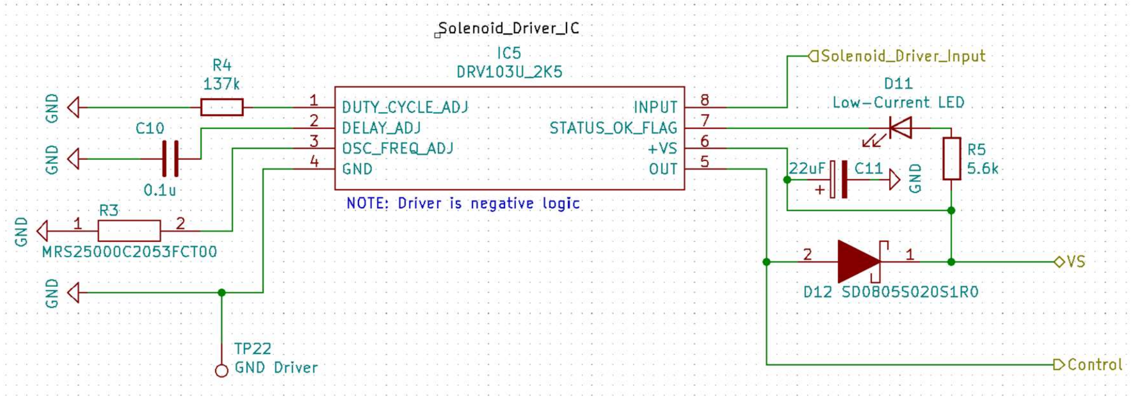


Figure 13 KiCAD Schematic of Solenoid Driver Circuitry

Figure 12 shows the completed schematic of the solenoid driver circuitry with connections between the selected components. In this schematic, Solenoid_Driver_Input is the signal from the Raspberry Pi GPIO to control lock status, VS is the supplied voltage from the 12V linear regulator, and Control is the output of the solenoid driver. The electric strike received VS into the positive terminal and Control into the negative terminal. When the Control signal is low, there is 12V across the electric strike, so it is locked. When the Control signal is high, the electric strike is unlocked. The voltage of the Control signal at logical high was measured to be approximately 7V with a National Instruments VirtualBench, which means roughly 5V is across the electric strike [18]. Since the electric strike requires 12VDC across for it to be locked, the electric strike is unlocked when the Control signal is driven high.

For the solenoid driver circuitry in Figure 13, R4 corresponds to the duty cycle adjust resistor, C10 corresponds to the delay adjust capacitor, R3 corresponds to the frequency adjust resistor, D11 corresponds to the status LED, R5 corresponds to the status resistor, D12 corresponds to the flyback diode, and C11 corresponds to the electrolytic bypass capacitor.

Connectors Circuitry

For the main power supply, an appropriate barrel jack was chosen based upon the jack size, voltage inputs, and current inputs from the wall-plug power supply. The selected barrel jack connector was from the same company as the wall-plug power supply to ensure compatibility [52].

For the rechargeable battery backup power supply, 5 battery holsters were acquired in the National Instruments Capstone Room, which were leftovers from previous Capstone projects. Each of these holsters holds 2-AA batteries and has two wires coming out of one end. The 5 holsters were soldered together to put all 10-AA rechargeable batteries in series, meaning that the connection to the board requires inputs for 2 wires: the positive and negative terminal of the equivalent battery, since they are all in series. In the power supply section, it was accurately stated that the design was based upon the assumption of 12-1.2V rechargeable batteries. However, after receiving the batteries and measuring the voltage output of the fully-charged batteries, it was determined that only 10 batteries were required to power the circuit in the case of power loss to the wall-plug power supply, such as in the event of a power outage. This will be further discussed in the Problems and Design Modifications subsection of the Hardware section. An appropriate connector for the rechargeable battery housings was selected based upon the current and voltage draws of the system. The selected connector has the necessary capability of fixing two loose wires to the connector [53].

The electric strike has two loose wires coming out of it. Since this case is identical to that of the rechargeable battery holsters, the same connection method can be used. The current and voltage requirements of this connector as lower than for the rechargeable batteries. As a result, the selected connector for the electric strike is the same as the connector for the rechargeable batteries [53].

The keypad selected for the Bouncer Locking System was based upon cost and availability. A 4-by-4 keypad was selected for this application (4 rows and 4 columns) [54]. According to the datasheet, the 4-by-4 keypad has 8 output pins, meaning that an 8-pin connector is necessary to connect the keypad to the PCB booster pack. An 8-pin connector was selected based upon dimensions of the output pins of the keypad [55], [56]. Once this 8-pin connector was received, it was discovered that the dimensions did not match up, so the header was removed on the keypad-facing side of the connector wire. Once the header was removed, the wires were connected to the keypad via soldering and 1-pin plastic connectors. This will be further discussed in the Problems and Design Modifications subsection of the Hardware section.

Once the keypad connector was determined, the connections between the keypad connector and the Raspberry Pi GPIO pins had to be investigated. After some research, it was decided that TVS diodes should be used to protect from high-voltage transients [57]. The appropriate TVS diodes were selected by searching through in-stock options that are rated for the system's current and voltage requirements, then selecting the lowest cost option [58].

To connect the Raspberry Pi to the PCB booster pack, a pin header was required. A Raspberry Pi pinout diagram was referenced to determine the appropriate size for this pin header. A 2-pin by 20-pin header was selected to connect the Raspberry Pi to the PCB booster pack [59].

Upon selecting all of the connector components, a KiCAD schematic was developed with all the components and requisite connections to realize the desired system behavior [17]. A schematic of the connector circuitry is shown below.

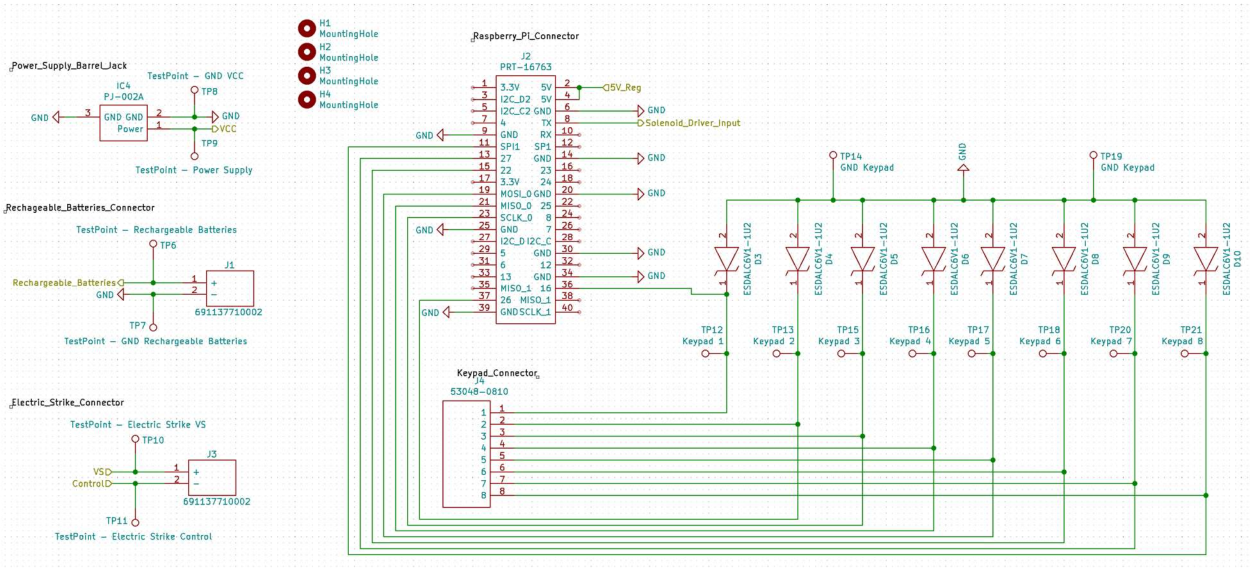


Figure 14 KiCAD Schematic of Connector Circuitry

Figure 14 shows the completed schematic of the connector circuitry. In the Power_Supply_Barrel_Jack part of the schematic, VCC is the input voltage from the wall-plug power supply. In the Rechargeable_Batteries_Connector part of the schematic, Rechargeable_Batteries is the input voltage from the rechargeable batteries. In the Electric_Strike_Connector part of the schematic, VS is the input voltage signal from the 12V linear voltage regulator and Control is the control signal that is output based upon the Solenoid_Driver_Input from the Raspberry Pi GPIO. In the Raspberry_Pi_Connector part of this circuit, 5V_Reg is the input voltage from the 5V Switching Regulator and Solenoid_Driver_Input is the signal the Raspberry Pi GPIO sends to the solenoid driver to for the electric strike to locked or unlocked status. The Raspberry Pi GPIO are also connected to the Keypad_Connector part of the schematic, which have TVS diodes in parallel.

Circuit Schematics

The schematics for the PCB booster pack design were broken up into three subcircuit pages: PowerPathing, SolenoidDriver, and Connectors. The SolenoidDriver schematic was shown in Figure 13, and the Connectors schematics was shown in Figure 14. Figure 5 and Figure 9 show two parts of the PowerPathing schematic. The full power pathing schematic is shown below.

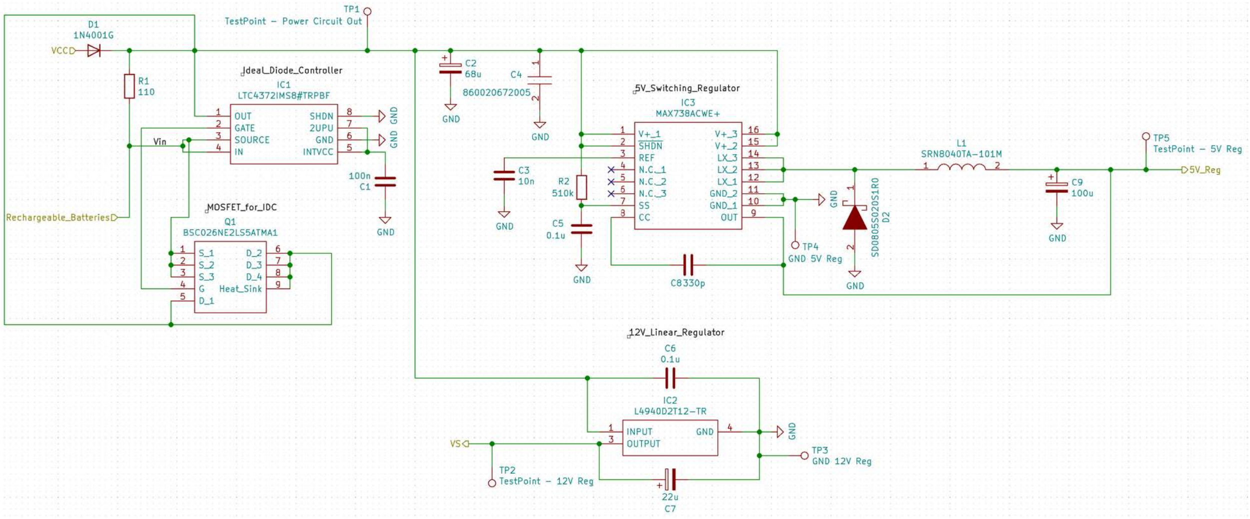


Figure 15 KiCAD Schematic of Power Pathing Circuitry

Figure 15 shows the completed schematic of the power pathing circuitry. The left half of the subcircuit (before TP1) is the power supply circuitry, and the right half of the subcircuit is the voltage regulator circuitry.

The PowerPathing, SolenoidDriver, and Connectors subcircuits are all connected in a higher level sheet. This was done for organization purposes when desiging the PCB booster pack. These connections are shown below.

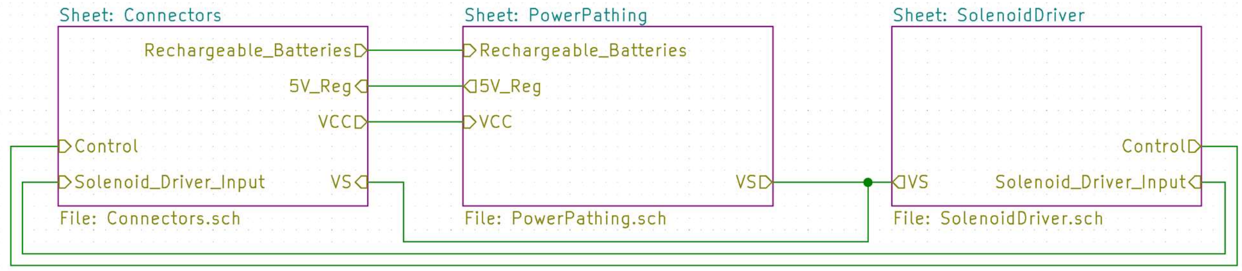


Figure 16 KiCAD Schematic of Subcircuit Connections

Figure 16 shows the connections between the subcircuits to create the intended behavior of the Bouncer Locking System.

PCB Layout

Following the completion of the KiCAD schematics, the next step in the hardware development process is preparing the PCB booster pack design for manufacturing by creating a PCB layout in KiCAD [17]. This layout is shown below.

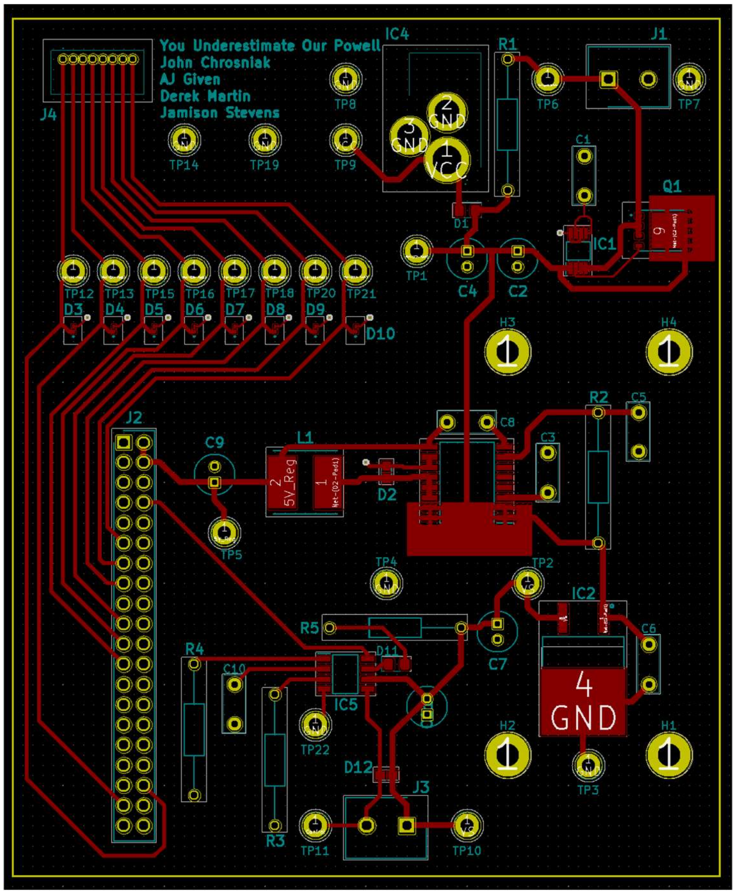


Figure 17 KiCAD PCB Layout

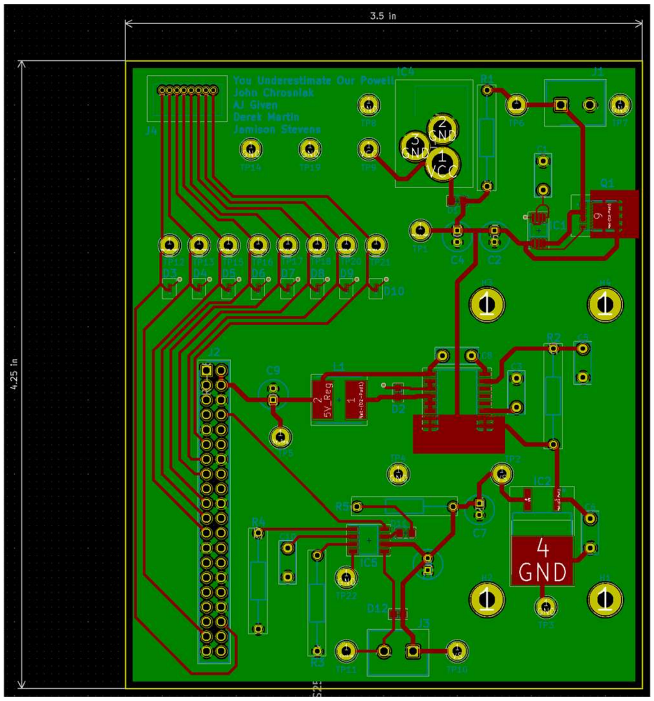


Figure 18 KiCAD PCB Layout with Ground Plane and Dimensions Showing

Figure 17 and Figure 18 show the PCB Layout based upon the circuitry designed for the Bouncer Locking System. The only differences between Figure 17 and Figure 18 are that the ground plane and board dimensions are showing in Figure 18. Figure 18 shows that the board dimensions are 4.25-inches by 3.5-inches.

To better show the relationship between the schematics and the PCB Layout, the figure below uses color-coded boxes to show the different subcircuits.

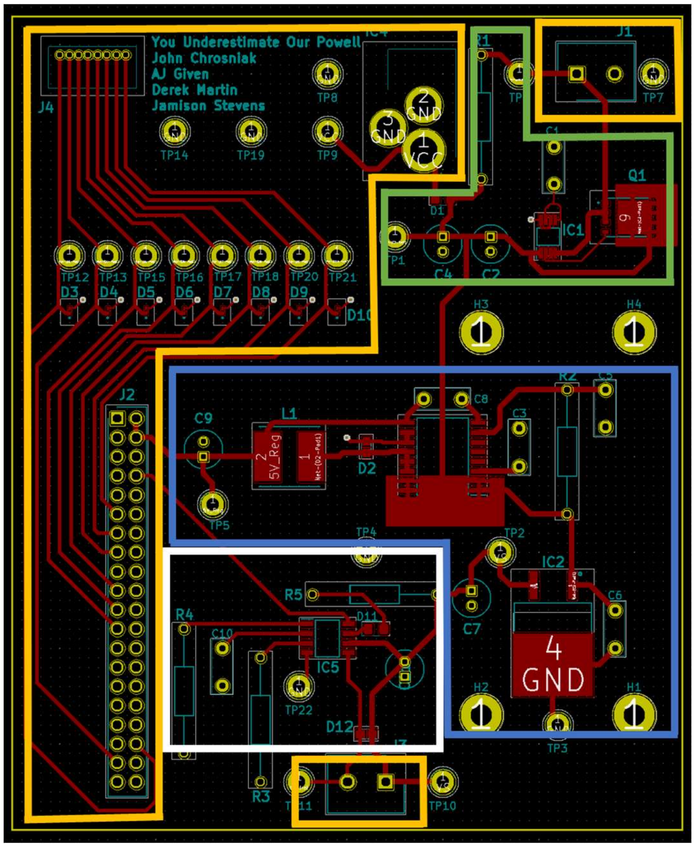


Figure 19 KiCAD PCB Layout with Color-Coded Boxes Around Subcircuits

Figure 19 shows the KiCAD PCB layout and has color-coded boxes to indicate where each subcircuit is on the board. The green box corresponds to the power supply circuitry, the blue box corresponds to the voltage regulator circuitry, the white box corresponds to the solenoid driver circuitry, and the orange boxes correspond to the connectors.

Problems and Design Modifications

There were not many problems or design modifications during the hardware section of this project. The main two problems faced were the realistic voltage readings of the fully-charged rechargeable batteries and the keypad connector.

When the rechargeable batteries were received, they were measured at 1.29V at full charge, instead of the expected 1.2V. As a result, using 12 batteries in series, the output voltage would be 15.48V, rather than the expected 14.4V. Since the wall-plug power supply supplied roughly 15V, the rechargeable batteries must supply less than 15V or the batteries will not recharge. To get around this problem, 10 batteries were used instead of 12, which reduced the voltage supplied to 12.9V. The intended system behavior was demonstrated experimentally with 10 rechargeable batteries, so this was a simple fix. The project's test plan and experimental verification well be elaborated upon in following sections.

The 8-pin keypad connector and associated 8-pin wire connector were selected based upon their datasheet information. It seemed that they wire connector's header was the same size as the 8-pin output from the keypad. However, after receiving these components, the sizes did not match up. To resolve this issue, the keypad connector on the board stayed the same, and the 8-pin header that would have connected to the keypad was removed. This ensured that there was a professional connection on the board. The keypad was connected to the 8 wires by soldering 1-pin connectors to each wire and plugging them in individually to the keypad.

Software

The software of the final design follows the standard three layer architecture of IoT systems, consisting of a perception (device), network (cloud storage), and application layer [60]. The device layer of the system includes the Raspberry Pi and all connected peripherals, such as the keypad and camera. The network layer consists of a single AWS S3 bucket that stores video surveillance footage and a JSON file containing information on generated passwords. The application layer is facilitated through a web application that allows the user to generate random passwords, view video surveillance footage, manage stored data, and send emails to guests containing their password information.

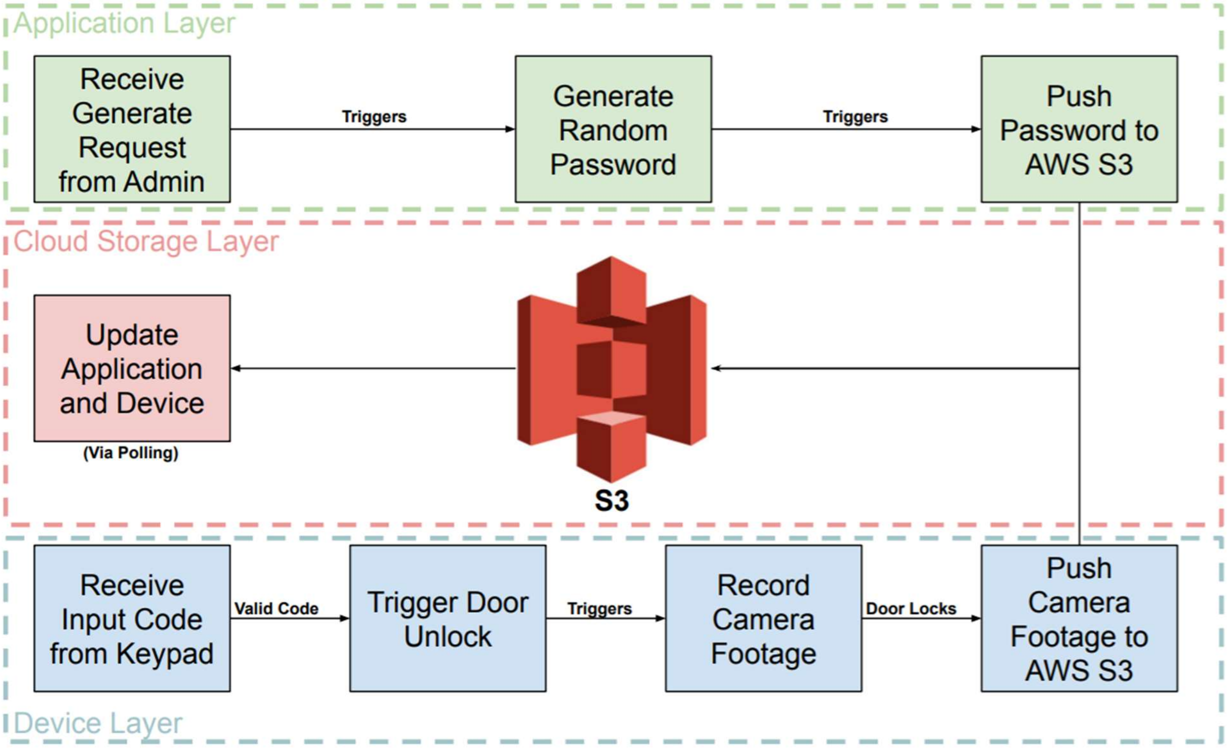


Figure 20 Software Flow Chart

The software flow chart system, shown in Figure 20, displays the basic functionality of each layer of the system, as well as how information is transferred between layers. The software for each layer will be discussed in detail in the following sections.

Device Layer

The device layer software consists of all drivers necessary for the attached peripherals (camera, keypad, and electric strike) and logic for hashing passwords to byte strings, validating entered passwords, marking passwords as expired, and uploading videos and password statuses to the cloud. The Unified Modeling Language (UML) diagram of the device layer software is shown in Figure 21 below, which was created using the Lucidchart diagramming application [61].

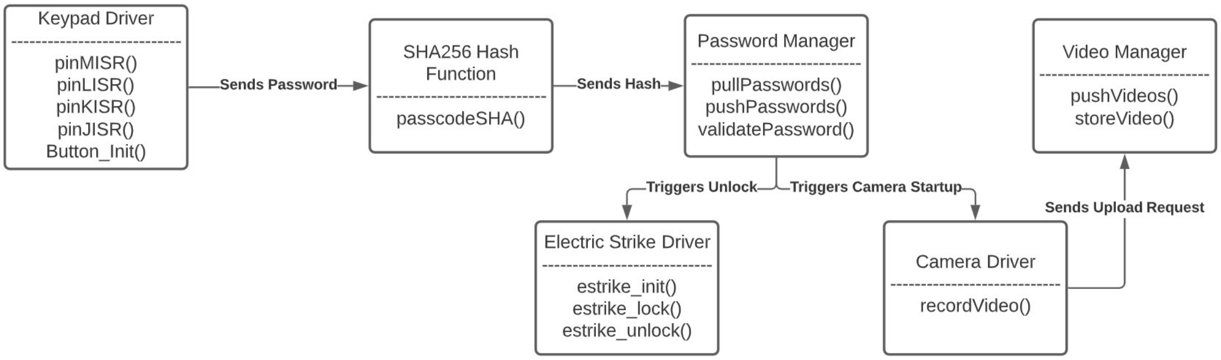


Figure 21 Device Layer UML Diagram

The UML diagram displays the general software flow of the device layer, as well as the modules along each stage of the pipeline and the functions within them. Each block present within the diagram is discussed in detail below.

Keypad Driver

The keypad Driver module includes the functions defined in ButtonISR.cpp as well as an infinite loop that runs in the main function of the project. The functions from ButtonISR.cpp are Button_Init(), pinMISR(int GPIO, int level, unsigned int tick), pinLISR(int GPIO, int level, pinKISR(int GPIO, int level, unsigned int tick)unsigned int tick), and pinJISR(int GPIO, int level, unsigned int tick).

The function Button_Init() initializes the general purpose input/output (gpio) library defined by pigpio.c, an opensource GPIO and thread library for the Raspberry Pi [62], sets the appropriate gpio levels for keypad input, associates the interrupt service routines (ISRs) with the correct pins, and calls the estrike_init() function.

The driver code from the main loop function of the Raspberry Pi functions in tandem with the ISR functions to determine which 2 of the 8 pins from the keypad are electrically connected (if any).

The ISR functions listed above are set to occur on rising edges by Button_Init(), however, since pigpio.c implements software interrupts instead of hardware triggered interrupts, false rising edges are possible. To mitigate this, the ISRs first check the level of their associated gpio pin to ensure that it is high. The main loop identifies which key has been pressed by finding which two pins have an electrical connection between them.[54] The datasheet defines the connection matrix based on which two pins are connected when a specific button is pressed. To determine which button was pressed, four of the pins marked as H, G, F, and E are set as outputs, while the pins marked M, L, K, and J are set as inputs and have associated rising edge interrupts. In order to determine which output pin the input is connected to, the output pins are periodically set to a logical high one at a time while the other outputs are set to a logical low. This process happens fast enough that any key pressed by a human should register. This has the added benefit that, should a malicious actor gain access to the wires connected to the keypad, the system will not allow them to execute a brute force computerized hack since it will not register inputs faster than a human can deliver them.

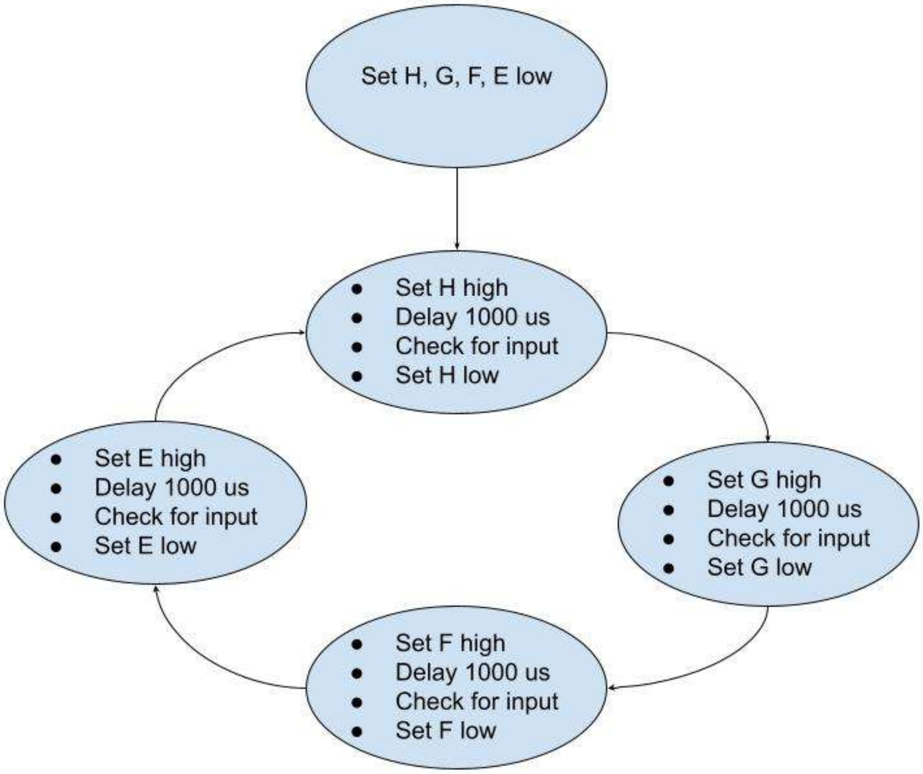


Figure 22 Button Recognition Algorithm

If, during the above algorithm, a button on the keypad is pressed, one of the 4 ISR functions will be called, which sets a variable indicating to the main loop which button was pressed. If this variable is set, the main loop will update a buffer that stores entered keystrokes in the following ways:

If the entered key is a number key (0-9) the program will check that 6 numbers have not already been entered. If less than 6 numbers have already been entered, the entered character is placed in a buffer and the iterator that keeps track of how many numbers have been entered is updated. If 6 numbers have already been entered, the buffer is not modified, but the iterator is updated so that a code cannot be checked until the clear button is pressed and a new code is entered. If the asterisk (*) key is pressed, the iterator is reset to 0, effectively clearing any previously entered input. If the pound (#) key is pressed the system checks to see that 6 numbers have been entered. If this is the case the data in the buffer is passed to the SHA hash function, and the output of that function is passed to the password manager. If the A, B, C, or D keys are pressed, nothing happens.

SHA256 Hash Function

The passcodeSHA(string code) function an implementation of the SHA256 algorithm first developed by the National Security Agency and published by the National Institute of Standards and Technology in 2001[23]. This implementation accepts a string input of up to 55 characters, but the system never calls the function with a string of any length other than 6. The extra length was added to allow more extensive testing of the implementation. The passcodeSHA function calls a number of helper functions including swapEndian(unsigned int & num), rightRotate(unsigned int n, unsigned int d), rightRotateNoEndian(unsigned int n, unsigned int d), rightShift(unsigned int n, unsigned int d), calculatew(unsigned int w16, unsigned int s0, unsigned int w7, unsigned int s1), and BinaryToHexString(uint8_t* inBinaryData, size_t inBinaryDataLength). The swapEndian function swaps the endianness of a 32 bit unsigned integer to assist in bit shifting, rotation, and addition since the SHA256 algorithm is defined using operations on a buffer of big endian unsigned integers and the RaspberryPi is a little endian machine. The rightRotate and rightShift functions rotate and shift their inputs d by n bits respectively. However, these functions use the swapEndian function to perform their respective operations as if the binary data passed to the function was a big endian representation of an unsigned integer. The function rightRotateNoEndian rotates the unsigned integer n by d bits without performing any endian swap operations. The function calculate performs a multistep calculation defined in the compression loop section of the SHA256 algorithm[23].

Password Manager

The password manager performs three primarily roles for the system: (1) verifying if a password is valid, (2) marking passwords as expired and uploading this information to the S3 bucket, and (3) periodically pulling the password directory from the S3 bucket to the device to fetch newly generated passwords. The first operation is performed within validatePassword() by receiving a hashed password from the SHA256 Hash Function module, which is then searched for in the JSON file containing passwords. The standard C++ library provides no implementation for interacting with JSON filetypes, so an open-source implementation provided by Niels Lohmann [63] was used. The manager will return a signal to unlock and start recording in the event a one-time password was entered, a signal to unlock and not record if a duration-based or super-user password entered, or a standby signal if the password was not found or is no longer valid. During this same process, the second operation of marking passwords as expired occurs if the entered password was a one-time password or a duration-based password that has passed its expiration time. This operation is performed by marking the status field in the JSON directory as used for the corresponding password hash. After this operation is completed, the pushPasswords() is called to upload the JSON directory to the S3 bucket to notify the web application a password was used. The AWS SDK for C++ [64] was used to perform this operation. The third operation is performed using a periodic interrupt scheduled to run every sixty seconds, in which the interrupt service routine pulls the most recent JSON directory from AWS S3. The open source pigpio library [62] was used to enable the timer interrupt and establish the interrupt service routine for it to execute.

Electric Strike Driver

The electric strike driver module contains the functions estrike_init(), estrike_lock(), and estrike_unlock(), which are all defined in the file estrike.c. The function estrike_init() sets the gpio pin that controls the solenoid driver to be an output with a logical value of 1. It should be noted that estrike_init() called on its own is not sufficient to initialize the gpio as the gpioInitialise() function from pigpio.c is not called from estrike_init(). For this reason, estrike_init() is called during Button_init(), which ensures that all gpios used are correctly initialized. The function estrike_lock() sets the GPIO, associated with the solenoid driver to a logical high, which causes the solenoid driver to provide power to the electric strike, locking it. If the gpio is already high, no changes are made to the system. The function estrike_unlock() sets the same gpio to a logical low, which causes the electric strike to lose power and unlock. As with estrike_lock() if the gpio is already set low, there are no changes made to the system.

Camera Driver

The camera driver was designed using the open source raspicam library [65] to interface with the Camera Serial Interface (CSI) Raspberry Pi camera [66]. The camera driver starts by toggling the device to high power mode for video recording and establishing a filename to write to. For each call to recordVideo(), the camera driver takes a snapshot from the camera and appends the output to the existing file using the VideoWriter tool provided by OpenCV, an open source library for computer vision applications [67]. The sequence of appended images results in a video sequence that compresses actual footage into a shorter timespan for easier viewing. The camera then switches back to low power mode once the recording loop terminates.

Video Manager

The video manager performs all operations related to pushing video surveillance to the S3 bucket and deleting footage that has been stored remotely. The storeVideo() function is called directly after a recording has been captured, which promptly sends a PUT request to store the specified video in S3. If the request returns a successful status code, the video manager deletes the footage from the device to free up storage and limit the amount of personal information stored on the device itself. However, if the PUT request returns an unsuccessful status code, which could happen in the event of an internet outage, the video footage will remain on the device. A separate process configured using timer interrupts with the pigpio library attempts to push all videos stored locally on the device to the cloud to account for the videos the device could not store remotely.

Preempt RT Kernel Patch

The Raspberry Pi microprocessor was not originally intended to operate as the core device of a real-time system. Thus, by default the Raspberry Pi does not provide deterministic responses to interrupts, which can pose problems for a system that requires event-triggered responses to execute within a defined time frame. Given that the system requires deterministic responses to read input from the keypad, a modified version of the standard Linux kernel was used to enable such responses. Specifically, the Preempt RT kernel patch [68] was chosen to enable real-time behavior on the Raspberry Pi.

Cloud Storage Layer

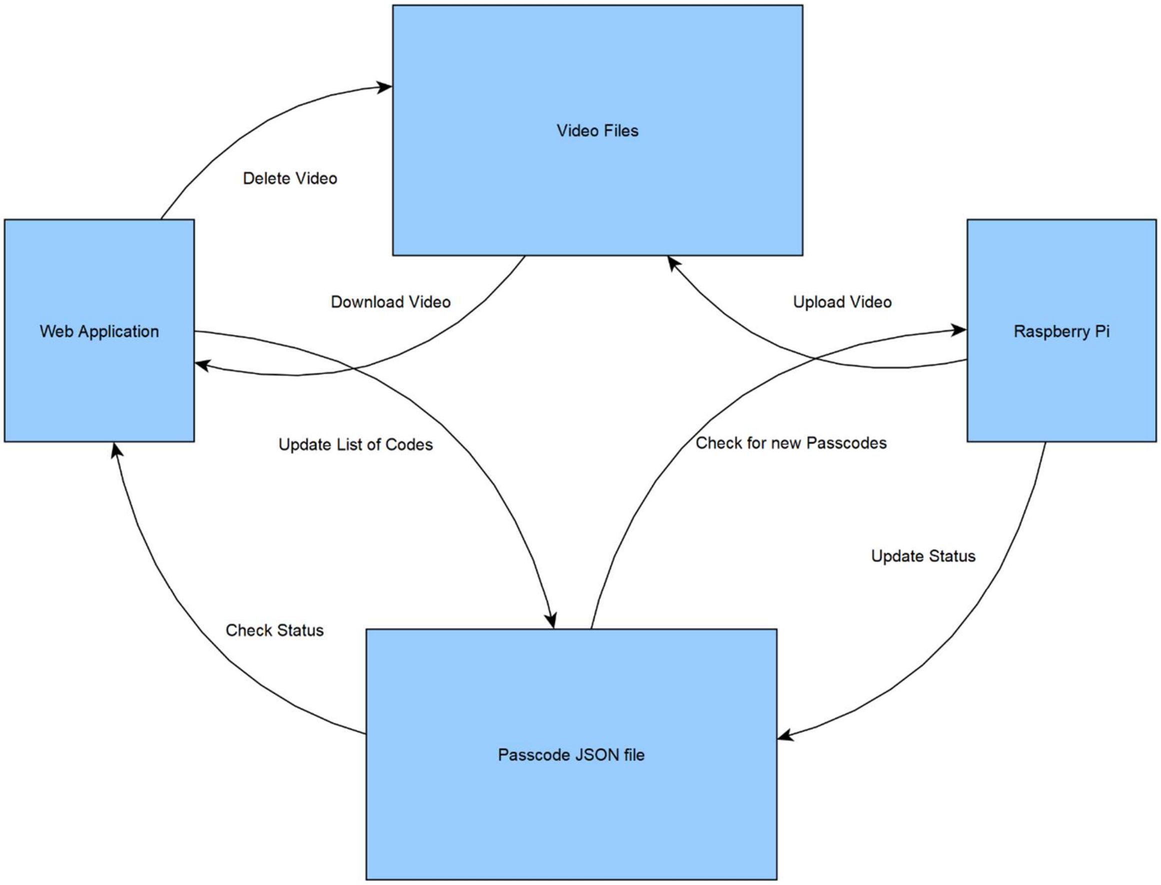


Figure 23 Cloud Storage Data Flow

Device Communication

The Raspberry Pi communicates with the AWS S3 bucket using PUT and GET requests of the REST API [69]. Software was designed using the official AWS SDK for interacting with the S3 bucket [64]. These interactions include uploading video footage and updating the password directory JSON file.

As previously mentioned in the Device Layer section, a request to upload video footage to the cloud is processed after each recording terminates. However, in the event that an internet outage prevents footage from successfully uploading, a separate process periodically checks for videos that have not been successfully pushed and reattempts to upload the footage.

The Raspberry Pi also periodically pulls the JSON file storing password information to ensure that the device recognizes any recently generated codes. In the event that either a one-time password or expired duration-based password is entered, the Raspberry Pi modifies the status of the password in the JSON file to invalid and reuploads the file to the S3 bucket to notify the web application.

Application Communication

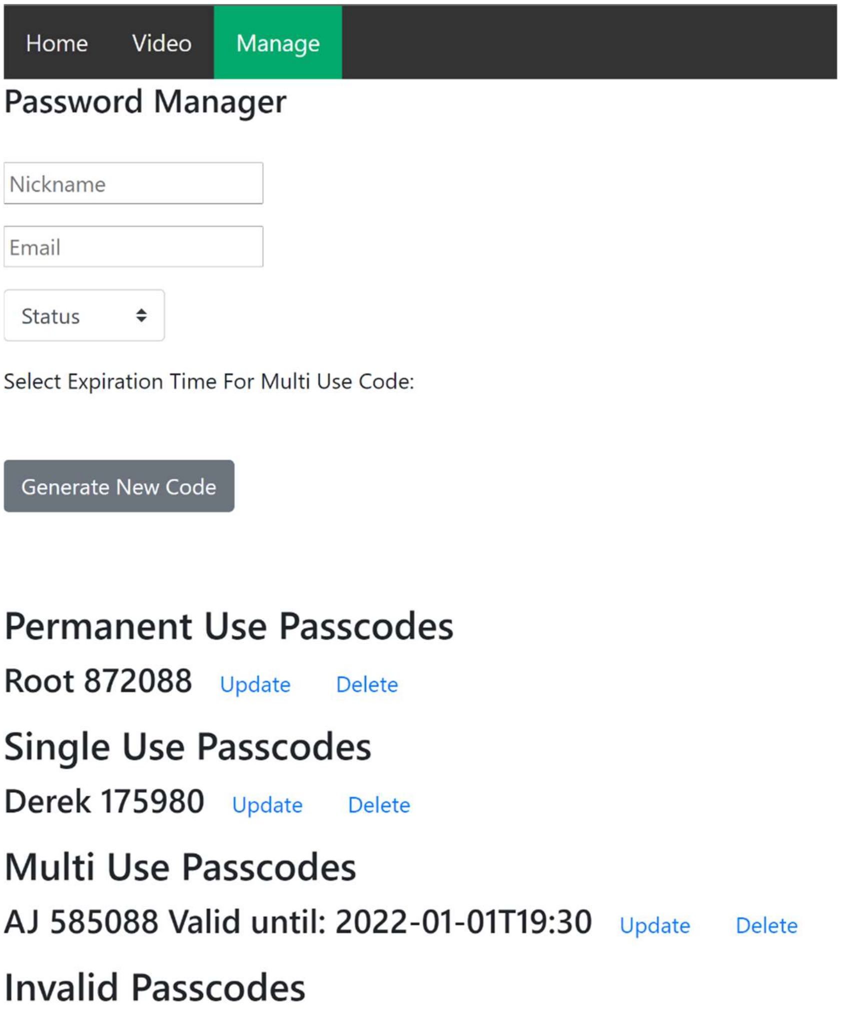
The web application communicates with the S3 bucket using boto3, an AWS Software Development Kit (SDK) for Python [70]. Interactions between the web application and the data stored in the cloud can be seen in the diagram in **Error! Reference source not found.** designed using [71]. These interactions can be split into those that target video files and those that target the JSON file containing passcode information.

Upon adding or deleting any passcodes, the contents of the database stored within the application are compiled into a JSON file with a hash of the passcode as the key and unique id, status, and datetime as additional fields. This JSON file is then sent to the cloud to overwrite the existing list. The web app will also pull a file from the same location to check if changes have been made by the Raspberry Pi before displaying passcode information to the user.

The web application will check to see which video files are saved in the cloud when the user navigates to the video viewer page and can download from or delete these video files if the user presses the corresponding buttons.

Application Layer

The application layer portion of the software consisted of a web application developed in Python using the Flask web framework. The web application was developed to provide an interface for the owner of the device to both manage passcodes and view video footage through a web browser. The application is made up of three main web pages which the user can navigate between using a navbar at the top of this screen. The three tabs are titled ‘Home’, ‘Video’, and ‘Manage’. A screenshot of the ‘Manage’ tab can be seen in Figure 24. The navbar was programmed to redirect to the correct url when clicked, and highlights green to indicate the current tab. The navbar will also turn from black to white when the user hovers over a specific tab that is not the current tab.



Password Manager

Nickname

Email

Status

Select Expiration Time For Multi Use Code:

Permanent Use Passcodes

Root 872088 Update Delete

Single Use Passcodes

Derek 175980 Update Delete

Multi Use Passcodes

AJ 585088 Valid until: 2022-01-01T19:30 Update Delete

Invalid Passcodes

Figure 24 Web Application User Schematic

The ‘Home’ tab is the landing page that the user will be directed to when initially accessing the web application. This tab acts as an introduction to the device and provides instructions for navigating and using the website.

The ‘Video’ tab displays information about all video files stored in the cloud. This information in rows and includes the nickname associated with the code that was entered for the video taken, a timestamp showing the date and time the video was recorded, and buttons to download the video and delete the video. If the delete button is pressed, the video file will be permanently deleted from the cloud. The initial design called for displaying a thumbnail of each video segment next to the nickname, but this made the web page too busy and required the page to load slowly as it would have to download each video file in advance. The updated design allows the page to load significantly quicker and makes the page more user friendly.

The ‘Manage’ tab provides all the functionality needed to manage passcodes for the device. The top half of the screen shows a form that can be filled out to generate a new passcode, while the bottom half of the screen lists all existing passwords separated and displayed in categories based on the associated status.

The form takes four fields that the user can fill out to provide information for each code generated. The first field, nickname, is a textbox that provides no functionality to the application and serves only to help identify video footage and passcodes as they are displayed. The second field, email, is also a textbox and allows the user to specify an email address for the passcode to be sent to when it is generated. The emails are sent using the Flask_Mail API [72] which sends messages using SMTP [73] to the specified address. The passcodes were initially going to be sent to the user via text message with phone number as a field instead of email address, but this was changed due to ease of use of API as well as accessibility in making a new email address to send all the codes from versus purchasing a phone number to send the codes from.

The next field allows the user to specify the type of passcode they would like to generate. These categories include single use, multi use, and permanent. Single use passcodes expire after being used once, but are valid for an unlimited amount of time before they are used, multi use passcodes are valid for an unlimited number of used, but expire after a specified amount of time, and permanent passcodes are valid for an unlimited amount of time and unlimited amount of uses. This field is implemented as a dropdown menu to allow the user to select from one of the three available options. If the user selects the multi use option, an additional datetime field will appear to allow the user to select the date, hour, and minute for the code to expire.

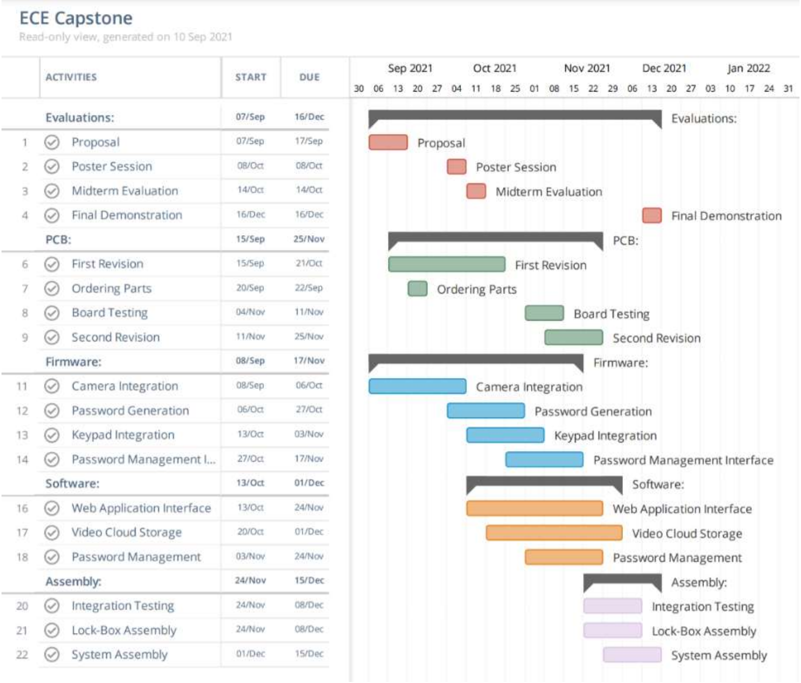
The database to store the passcodes and information associated with each one was created using SQLAlchemy [74]. The primary key for the database was an integer representing a unique id for that passcode that increments with each new code generated. Other fields in the database are username, code, status, and datetime with username and code fields being unique to differentiate codes and eliminate the possibility of the same code being generated for two different statuses or users. Status, code, and id are all stored as integers while datetime and username fields are strings.

The generate new code button compiles the information from the various fields and creates a new entry in the database. Once the database is updated, a function is called to update the information stored in the cloud to reflect this. This information in the cloud is stored as a JSON file which takes a hash of the passcode as the key, and the user id, status, and datetime as fields. The code is generated using the python secrets module which uses the most cryptographically secure source of entropy available from a given operating system to generate pseudorandom numbers[75]. The hash is computed at the application layer using the python function getHash(password), which uses the python hashlib module's implementation of the SHA256 hash[76] and returns the result as string of hex characters. This function also sends the email with the passcode and reloads the page.

Whenever the manage page is loaded or refreshed, the contents of the database are loaded in order to display for the user. Before displaying, the single use codes are compared by unique id to the codes saved in the cloud to see if they have been used. If this is the case, the status needs to be updated within the database to reflect this. The codes are then displayed by status under the headings shown in Figure 24. For each category the nickname and passcode are listed, and for multi use codes the expiration date is listed. The invalid codes header lists all single use codes that have already been used and are no long valid if entered in the keypad.

Next to each entry are two buttons, one to update the entry and one to delete the entry. If the update button is pressed, a separate page will load where the user can change the nickname in a textbox and press an update button which will update the value saved in the database and reload the 'Manage' page. If the delete button is pressed, the entry will be removed from the database and the JSON file saved in the cloud will be updated to reflect this removal.

Project Timeline



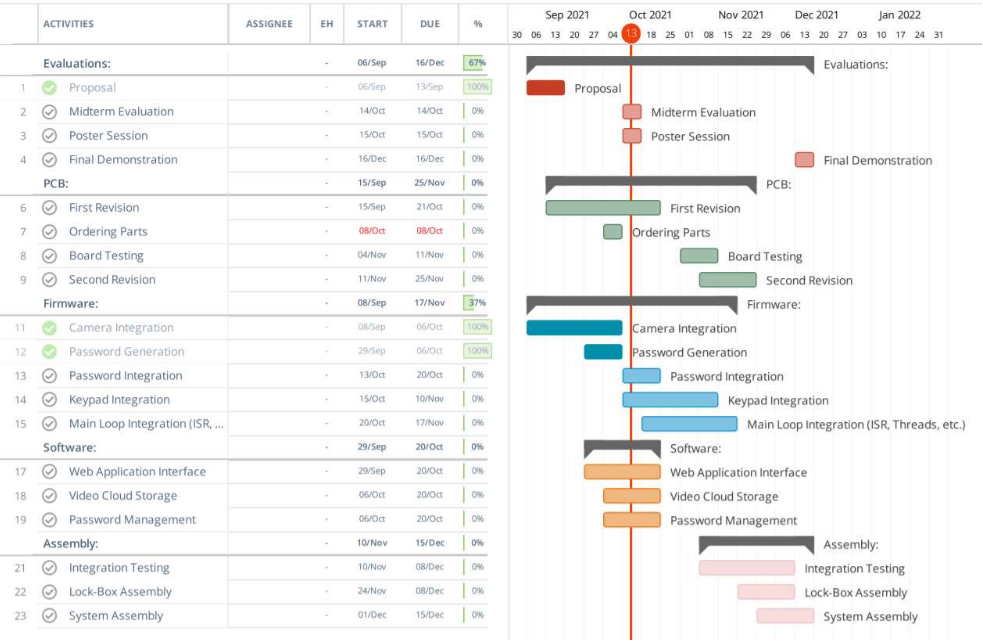
ECE Capstone

Read-only view, generated on 10 Sep 2021

	ACTIVITIES	START	DUE
	Evaluations:	07/Sep	16/Dec
1	✓ Proposal	07/Sep	17/Sep
2	✓ Poster Session	08/Oct	08/Oct
3	✓ Midterm Evaluation	14/Oct	14/Oct
4	✓ Final Demonstration	16/Dec	16/Dec
	PCB:	15/Sep	25/Nov
6	✓ First Revision	15/Sep	21/Oct
7	✓ Ordering Parts	20/Sep	22/Sep
8	✓ Board Testing	04/Nov	11/Nov
9	✓ Second Revision	11/Nov	25/Nov
	Firmware:	08/Sep	17/Nov
11	✓ Camera Integration	08/Sep	06/Oct
12	✓ Password Generation	06/Oct	27/Oct
13	✓ Keypad Integration	13/Oct	03/Nov
14	✓ Password Management I...	27/Oct	17/Nov
	Software:	13/Oct	01/Dec
16	✓ Web Application Interface	13/Oct	24/Nov
17	✓ Video Cloud Storage	20/Oct	01/Dec
18	✓ Password Management	03/Nov	24/Nov
	Assembly:	24/Nov	15/Dec
20	✓ Integration Testing	24/Nov	08/Dec
21	✓ Lock-Box Assembly	24/Nov	08/Dec
22	✓ System Assembly	01/Dec	15/Dec

Figure 25 Original Gantt Chart

The Gantt Chart shown in Figure 25 illustrates the original project timeline for accomplishing tasks throughout the semester. The project was divided into four main categories: PCB design and layout, embedded firmware design, user experience software development, and system integration. All hardware, firmware, and software design tasks can be accomplished in parallel, while system assembly will require that most of the earlier tasks have been completed.



	ACTIVITIES	ASSIGNEE	EH	START	DUE	%
	Evaluations:		-	06/Sep	16/Dec	67%
1	✓ Proposal		-	06/Sep	13/Sep	100%
2	✓ Midterm Evaluation		-	14/Oct	14/Oct	0%
3	✓ Poster Session		-	15/Oct	15/Oct	0%
4	✓ Final Demonstration		-	16/Dec	16/Dec	0%
	PCB:		-	15/Sep	25/Nov	0%
6	✓ First Revision		-	15/Sep	21/Oct	0%
7	✓ Ordering Parts		-	08/Oct	08/Oct	0%
8	✓ Board Testing		-	04/Nov	11/Nov	0%
9	✓ Second Revision		-	11/Nov	25/Nov	0%
	Firmware:		-	08/Sep	17/Nov	37%
11	✓ Camera Integration		-	08/Sep	06/Oct	100%
12	✓ Password Generation		-	29/Sep	06/Oct	100%
13	✓ Password Integration		-	13/Oct	20/Oct	0%
14	✓ Keypad Integration		-	15/Oct	10/Nov	0%
15	✓ Main Loop Integration (ISR, ...		-	20/Oct	17/Nov	0%
	Software:		-	29/Sep	20/Oct	0%
17	✓ Web Application Interface		-	29/Sep	20/Oct	0%
18	✓ Video Cloud Storage		-	06/Oct	20/Oct	0%
19	✓ Password Management		-	06/Oct	20/Oct	0%
	Assembly:		-	10/Nov	15/Dec	0%
21	✓ Integration Testing		-	10/Nov	08/Dec	0%
22	✓ Lock-Box Assembly		-	24/Nov	08/Dec	0%
23	✓ System Assembly		-	01/Dec	15/Dec	0%

Figure 26 Revised Gantt Chart

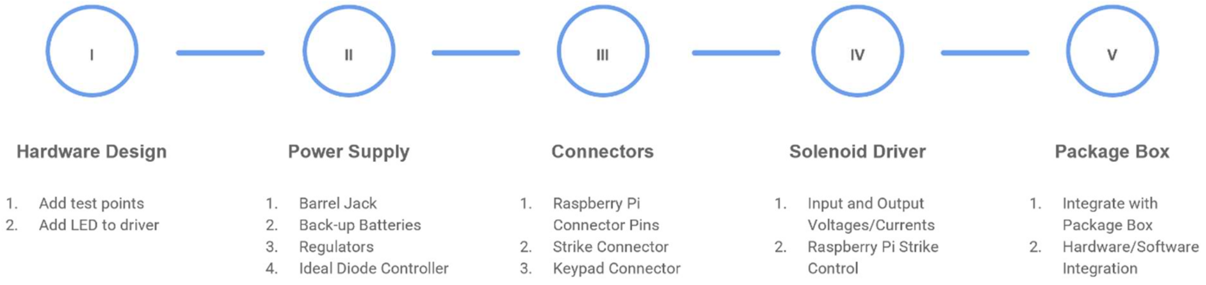
A revised Gannt Chart, displayed in Figure 26 Error! Reference source not found. **Revised Gantt Chart**

, shows the adjusted project timeline as of the midterm design review. All software and firmware tasks scheduled for completion by the design review had been successfully completed, with some tasks completed ahead of schedule. Due to time constraints, the first board revision was pushed back to the second submission date to allow for a more thorough review process before fabrication. The project timeline after the design review mostly followed the schedule of the revised Gantt chart, with the only exception being a second hardware revision was not needed.

Test Plan

Hardware

There was no mention of a test plan in the submitted project proposal. However, there was an outline of the test plan in the Midterm Design Review Presentation. This test plan outline is shown below.



Hardware Design

1. Add test points
2. Add LED to driver

Power Supply

1. Barrel Jack
2. Back-up Batteries
3. Regulators
4. Ideal Diode Controller

Connectors

1. Raspberry Pi Connector Pins
2. Strike Connector
3. Keypad Connector

Solenoid Driver

1. Input and Output Voltages/Currents
2. Raspberry Pi Strike Control

Package Box

1. Integrate with Package Box
2. Hardware/Software Integration

Figure 27 Test Plan Outline from Midterm Design Review Presentation

The test plan outline shown in Figure 27 was followed. As more parts of the project came together, this test plan became more specific in terms of what measurements were necessary.

In the design phase of the project, testpoints were added to increase ease of experimental hardware verification. These testpoints were added at the barrel jack connector, rechargeable batteries connector, power supply circuitry output, 12V linear voltage regulator output, 5V switching voltage regulator output, both terminals of the electric strike connector, and each of the keypad connections. Additionally, several grounded testpoints were added to increase ease of voltage measurements. A status LED was added to one terminal of the solenoid driver integrated circuit to serve as a visual indicator of proper solenoid driver operation.

To experimentally verify the system hardware, voltages were measured at each of the testpoints. Voltage measurements at each of the testpoints, except the keypad connection testpoints, are included in this section. These measurements were conducted with the use of a National Instruments VirtualBench [18].

When designing the circuitry for the PCB booster pack, there were testpoints placed at the inputs and output of each subcircuit: power supply circuitry, voltage regulator circuitry, solenoid driver circuitry, and connectors circuitry. These subcircuit divisions served to organize the schematic development process and divide the PCB booster pack into testable submodules.

The only testing that resulted in modifying the initial design was the measurement of the rechargeable battery voltages at full charge. The selected rechargeable batteries output 1.29V at full charge, rather than the 1.2V that the batteries advertise. As a result, the number of rechargeable batteries used for the backup power supply was reduced from 12 to 10 (reduced 15.48V output to 12.9V output) so that the voltage supplied by the backup batteries was lower than that of the wall-plug power supply. This was not a major change and did not result in any changes to the hardware design other than removing 2-AA batteries and an external battery holster, which holds 2-AA batteries. The remainder of the hardware operated as expected and produced the desired system behavior after integration.

The data collection process for the hardware testing phase is described below. The data collected from hardware testing is included below in the form of screenshots from a National Instruments VirtualBench [18].

To ensure the intended operation of the designed PCB booster pack, voltages at several points throughout the circuit were measured with the use of testpoints and a National Instruments VirtualBench [18]. There were three different scenarios under which voltages were measured: power from wall-plug and rechargeable batteries power supplies, power from only wall-plug power supply, and power from only rechargeable batteries power supply.

In the first scenario, both the wall-plug power supply and the rechargeable batteries power supply were connected to the PCB booster pack. This situation simulates normal power supply operation for the system, where power is received from both the wall-plug power supply and the rechargeable batteries backup power supply. The measurements from this scenario are shown in Figures 28 through 35.

In the second scenario, only the wall-plug power supply was connected to the PCB booster pack. This situation simulates powering the system with uncharged batteries, where power is only received from the wall-plug power supply. The measurements form this scenario are shown in Figures 36 through 41.

In the third scenario, only the rechargeable batteries backup power supply was connected to the PCB booster pack. This situation simulates loss of power to the house's main power line (ex. a power outage) or from the wall-plug power supply, where the system is powered by the rechargeable batteries backup power supply. The measurements form this scenario are shown in Figures 42 through 49.

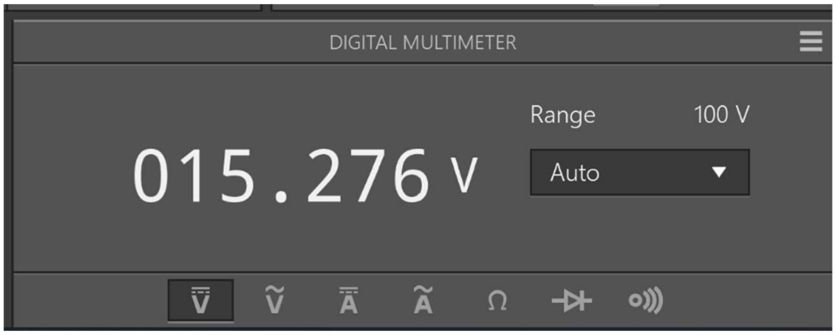
In each of the scenarios, the first figure shows the voltage at the wall-plug's barrel jack connector, the second figure shows the voltage at the rechargeable batteries' connector, and the third figure shows the voltage out of the power supply circuitry (at TP1 in Figure 5). The purpose of these measurements is to determine if the power supply circuitry is delivering enough power to the regulator circuitry. For this system, the power out of the power supply circuitry should be at least 12V, so that 5V can be supplied to the Raspberry Pi and 12V can be supplied to the electric strike.

For all scenarios, the fourth figure shows the voltage out of the 12V linear voltage regulator, and the fifth figure shows the voltage out of the 5V switching voltage regulator. The expected voltages out of the regulator circuitry is approximately 12V and 5V, respectively. There is a small margin of error for voltage regulator outputs in which the system will produce the desired behavior. The Raspberry Pi will operate as intended with a supply voltage between 5V and 5.1V [77]. The electric strike will operate as intend with a supply voltage approximately between 11V and 13V based upon measurements with a National Instruments VirtualBench [18].

In each scenario, the sixth figure shows the voltage into the positive terminal of the electric strike's connector. This terminal is electrically connected to the output of the 12V linear voltage regulator, so the reading should be identical to that shown in the fourth figure of each scenario.

In the first and third scenarios, the seventh figure shows the voltage into the negative terminal of the electric strike's connector when the electric strike is unlocked, and the eighth figure shows the voltage across the electric strike connector when the electric strike is unlocked. The voltage reading in the seventh figure should be roughly identical to the output of 12V linear voltage regulator shown in the fourth figure. The voltage reading in the eighth figure should be roughly equal to zero volts. These measurements were not necessary for the second scenario, since the voltage output of the power supply circuitry and voltage regulator circuitry in the first and second scenarios are roughly identical. This similarity in output voltages is because the ideal diode controller powers the circuit from the wall-plug power supply in both of these scenarios, due to the diode-OR setup of the power supply circuitry.

First Scenario (Power from both Wall-Plug and Rechargeable Batteries Power Supplies)

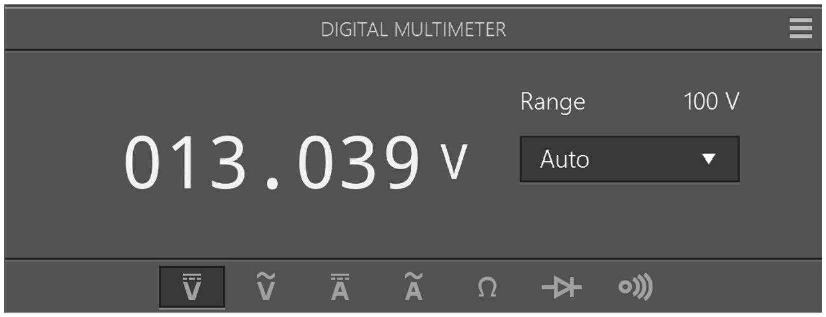


015.276 V

Range 100 V

Auto

Figure 28 Barrel Jack Power Supply Voltage

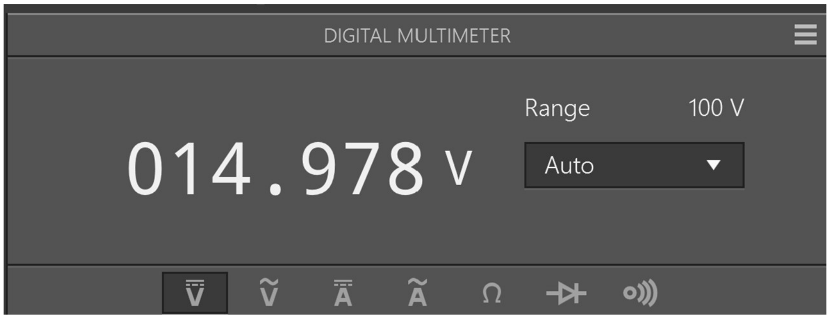


013.039 V

Range 100 V

Auto

Figure 29 Rechargeable Batteries Power Supply Voltage

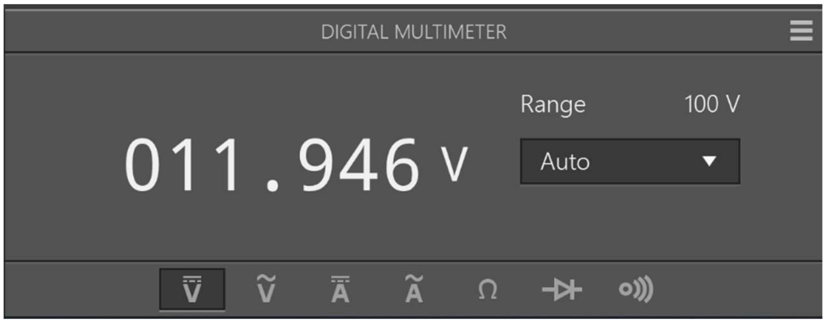


014.978 V

Range 100 V

Auto

Figure 30 Voltage out of Power Supply Circuitry

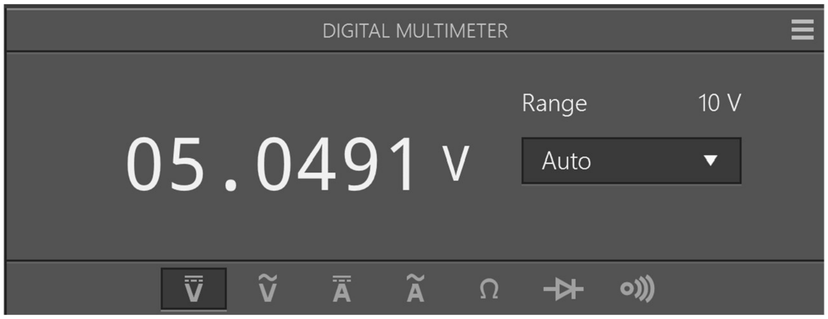


011.946 V

Range 100 V

Auto

Figure 31 Voltage out of 12V Linear Regulator

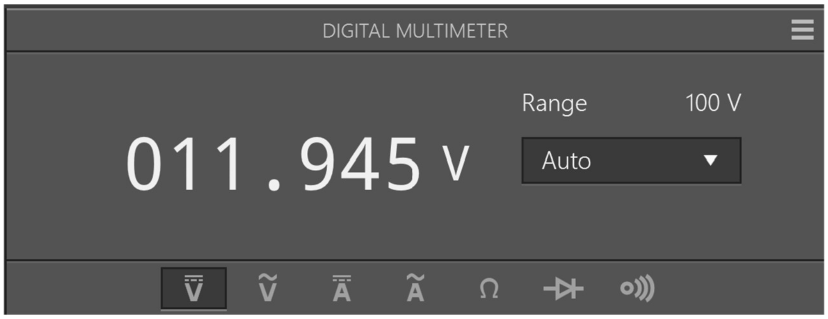


05.0491 V

Range 10 V

Auto

Figure 32 Voltage out of 5V Switching Regulator

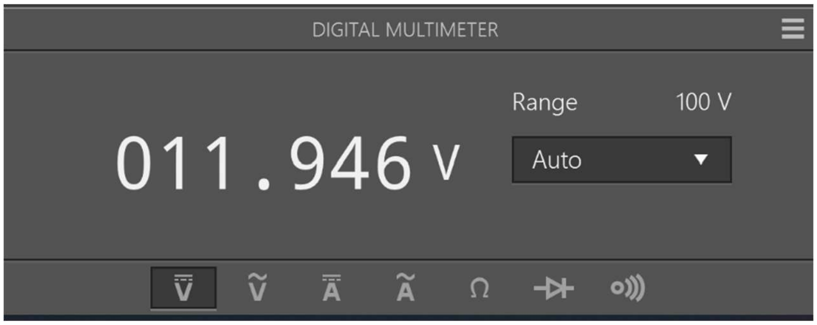


011.945 V

Range 100 V

Auto

Figure 33 Voltage into Positive Terminal of Electric Strike

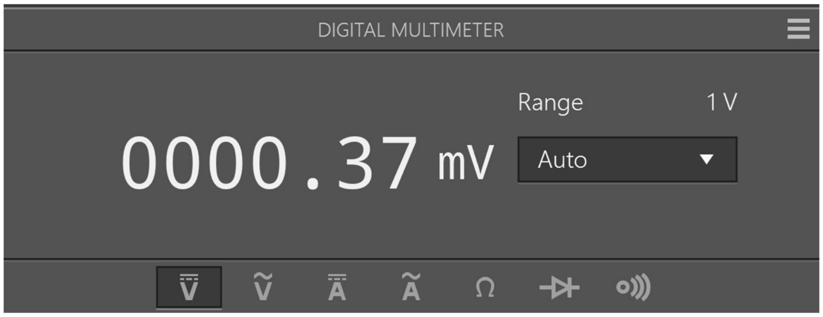


011.946 V

Range 100 V

Auto

Figure 34 Voltage into Negative Terminal of Electric Strike when Unlocked



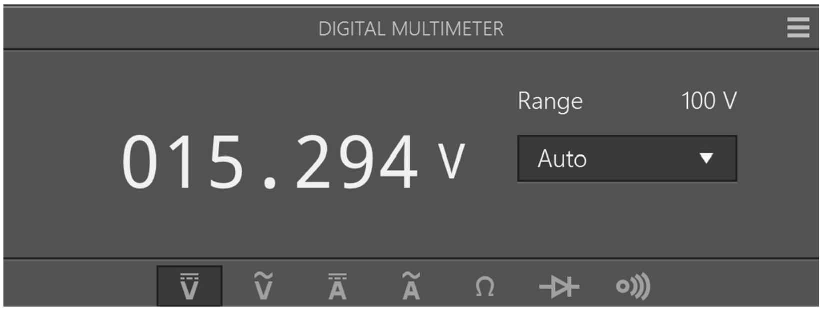
0000.37 mV

Range 1 V

Auto

Figure 35 Voltage Across Electric Strike when Unlocked

Second Scenario (Power from only Wall-Plug Power Supply)

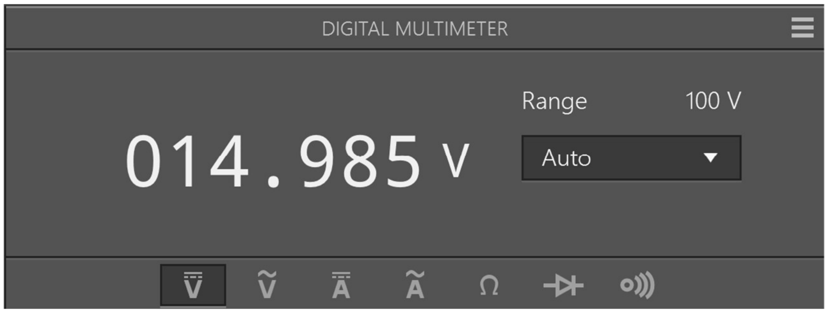


015.294 V

Range 100 V

Auto

Figure 36 Barrel Jack Power Supply Voltage without Rechargeable Batteries

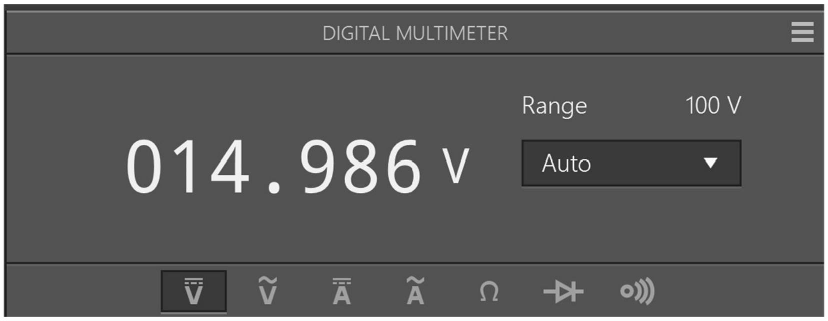


014.985 V

Range 100 V

Auto

Figure 37 Voltage at Rechargeable Batteries Connector without Rechargeable Batteries Connected

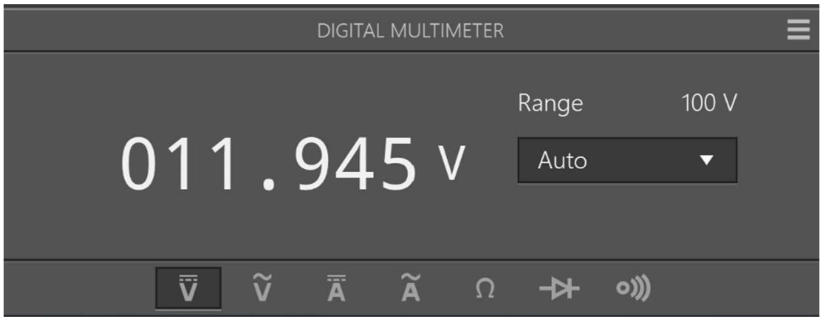


014.986 V

Range 100 V

Auto

Figure 38 Voltage out of Power Supply Circuitry without Rechargeable Batteries

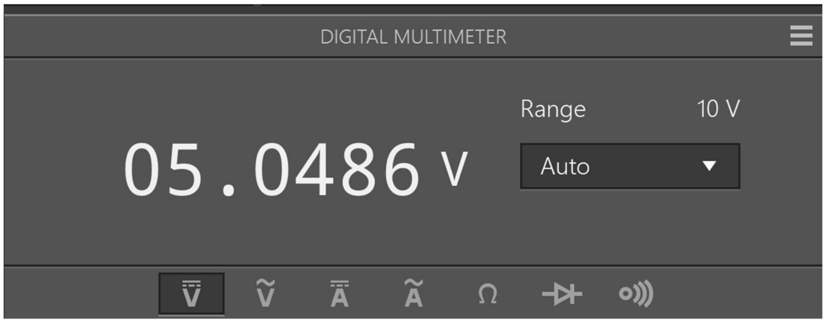


011.945 V

Range 100 V

Auto

Figure 39 Voltage out of 12V Linear Regulator without Rechargeable Batteries



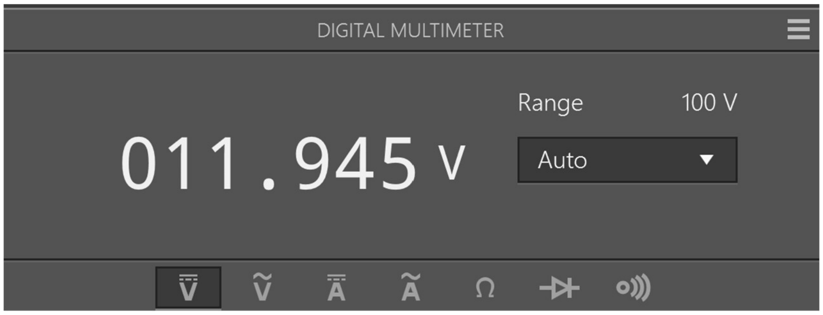
DIGITAL MULTIMETER

Range 10 V

05.0486 V

Auto

Figure 40 Voltage out of 5V Switching Regulator without Rechargeable Batteries



DIGITAL MULTIMETER

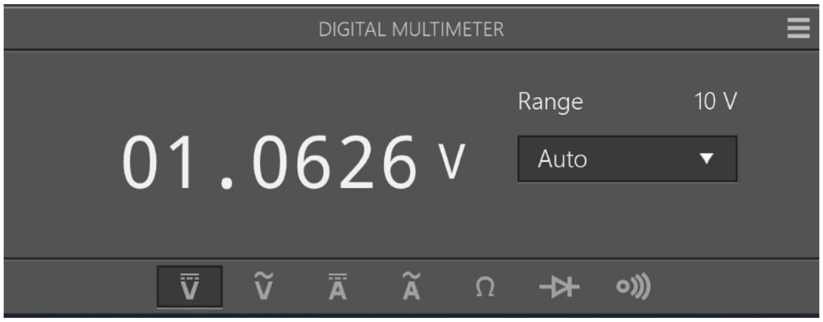
Range 100 V

011.945 V

Auto

Figure 41 Voltage into Positive Terminal of Electric Strike without Rechargeable Batteries

Third Scenario (Power from Only Rechargeable Batteries Backup Power Supply)



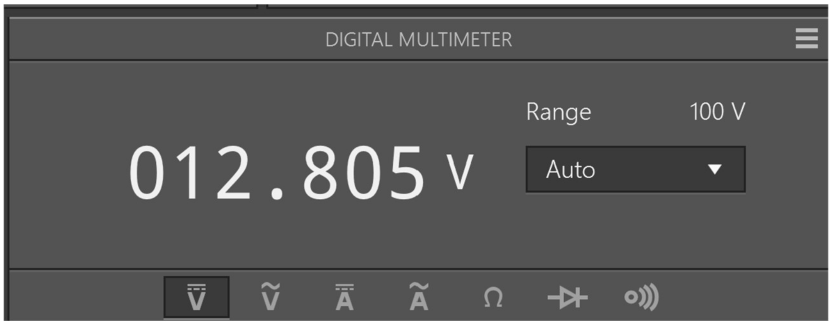
DIGITAL MULTIMETER

Range 10 V

01.0626 V

Auto

Figure 42 Voltage at Barrel Jack without Wall-Plug Power Supply



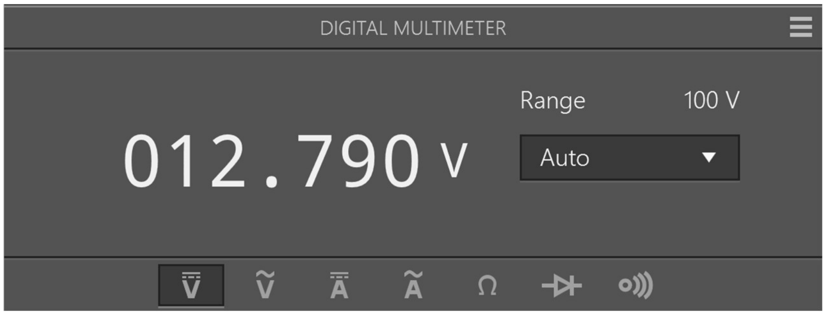
DIGITAL MULTIMETER

Range 100 V

012.805 V

Auto

Figure 43 Rechargeable Batteries Power Supply Voltage



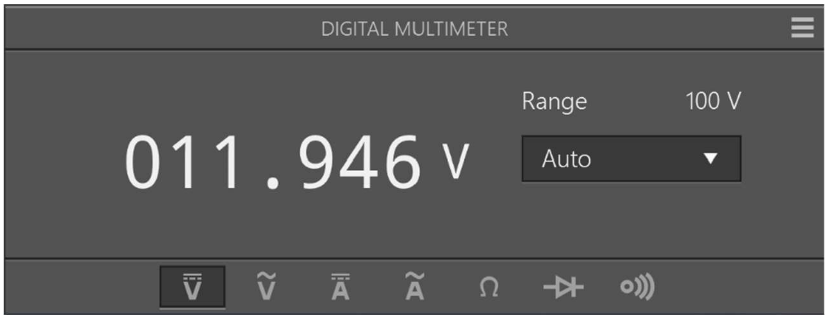
DIGITAL MULTIMETER

012.790 V

Range 100 V

Auto

Figure 44 Voltage out of Power Supply Circuitry without Wall-Plug Power Supply



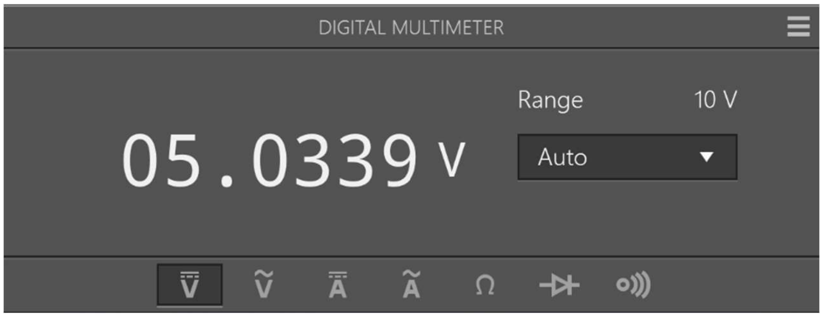
DIGITAL MULTIMETER

011.946 V

Range 100 V

Auto

Figure 45 Voltage out of 12V Linear Regulator without Wall-Plug Power Supply



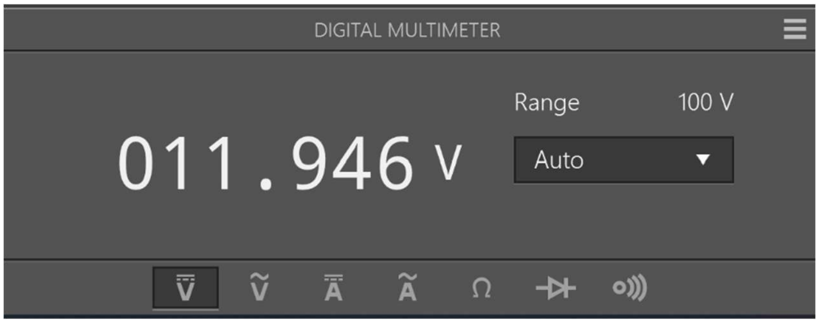
DIGITAL MULTIMETER

05.0339 V

Range 10 V

Auto

Figure 46 Voltage out of 5V Switching Regulator without Wall-Plug Power Supply



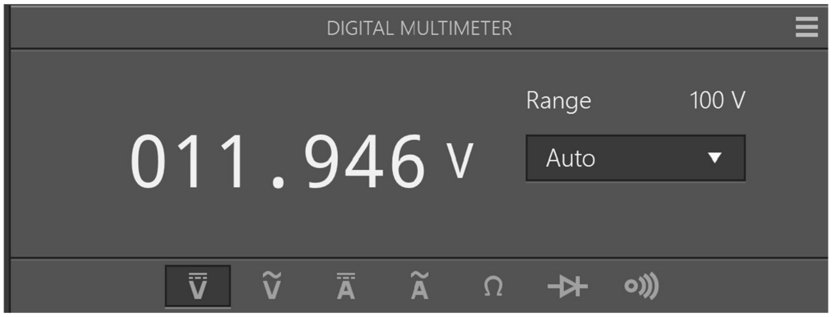
DIGITAL MULTIMETER

011.946 V

Range 100 V

Auto

Figure 47 Voltage into Positive Terminal of Electric Strike without Wall-Plug Power Supply

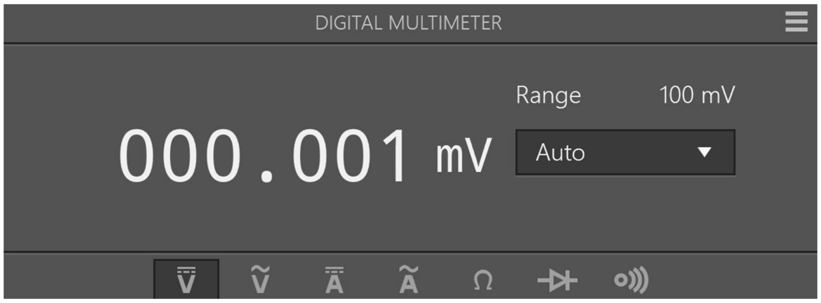

DIGITAL MULTIMETER

Range 100 V

011.946 V

Auto

Figure 48 Voltage into Negative Terminal of Electric Strike when Unlocked without Wall-Plug Power Supply


DIGITAL MULTIMETER

Range 100 mV

000.001 mV

Auto

Figure 49 Voltage across Electric Strike when Unlocked without Wall-Plug Power Supply

The measurements in each of the figures for all three scenarios match with the voltage expectations required for demonstrating intended system behavior. As a result, these tests experimentally verified the design of the PCB booster pack. The integration subsection will further discuss the demonstration of intended system behavior.

Software

To verify generating a passcode and adding it to the database, a nickname and email were entered and single use passcode was selected. After pressing generate, it was verified that the same nickname along with a random passcode were displayed under the single use passcodes header. After closing out of the browser and reloading the web page, this same information was once again verified. The program hosting the website was then stopped and restarted, and it was once again verified that the same nickname and passcode were displayed under the single use passcodes header.

To verify updating a user, the same passcode was left in the database. The update button to the right was then clicked and after the next web page loaded, a new nickname was entered and the update name button was clicked, it was verified that the nickname next to the same passcode as before was updated. After closing out of the browser and reloading the web page, this same information was once again verified. The program hosting the website was then stopped and restarted, and it was once again verified that the same nickname and passcode were displayed under the single use passcodes header.

To verify deleting a user, the same passcode was left in the database. The delete button to the right of the passcode was clicked and after the web page refreshed, it was verified that the passcode and corresponding nickname were no longer present on the screen. After closing out of the browser and reloading the web page, this was information was once again verified. The program hosting the website was then stopped and restated, and it was once again verified that the nickname and passcode were not present under any of the headers.

To verify cloud integration on the application side, several tests were performed. To verify that new passcodes generated in the web app appeared in the cloud storage, the contents of the JSON file present in the cloud were downloaded and inspected. A new single use passcode was then generated. After downloading the JSON file once again it was verified that an additional entry with status code 1, indicating a single use passcode, was present in the file.

To verify downloading a video from the cloud storage, a known video file was uploaded and placed within the Video folder in the S3 bucket. After verifying an entry appeared on the video tab with a timestamp equal to the time the video was uploaded, and ignoring the lack of nickname since this video was hardcoded in, the download button was pressed. After navigating to the downloads folder and playing the video, it was verified that the video was the same as the one uploaded.

To verify deleting a video from cloud storage from within the web app, the same video was kept. After pressing the delete video button from within the video viewer tab of the web application and allowing the page to reload, it was verified that the video and associated information were no longer displaying. Checking the video folder within the S3 bucket verified that the video previously uploaded was no longer present.

To verify that the C++ implementation of the SHA256 hash algorithm was correct, a github webpage published by github user emn178[78] that computes various hashes was used to hash strings and compare the known results to the output from the C++ function. Since, hash algorithm output change drastically from small changes to an input and the function is only being used for fixed length inputs, extensive testing was not needed to verify the hashing function.

Firmware

All firmware testing was performed based on deliverable demonstrations of each individual process. These processes include reading input from the keypad, hashing passwords to byte strings, recording video footage, uploading footage to the cloud, updating the JSON file containing password information, and reattempting to upload footage that was not correctly processed during the first attempt.

Keypad testing was performed using a simple program that printed out the character associated with each key press the keypad received. This program also allowed the user to clear input with a press to the asterisk key and enter input with a press to the pound key. Each button press was first tested individually, followed by presses to the clear and enter button to test the buffer storing characters. Further stress testing involved entering in too many or too few characters, in which the keypad operated as expected.

The SHA256 hashing function was tested using a program that accepted a string as input and printed out the hashed byte string. Each input was verified using the SHA256 online hash function tool [79], which was previously known to operated correctly.

Recording video footage was tested by manually setting recording to take place for a specified period. After the elapsed period, the video was manually inspected to ensure the quality of recording was viewable and of high enough quality.

Testing for uploading video footage consisted of a program that manually called PUT requests using the AWS SDK. The response for each request was printed to the terminal, and the S3 bucket was then manually inspected by signing into the AWS console.

Updating the JSON file containing password information was tested using a program that was manually given a byte string. This byte string was then compared with the strings present in the JSON file. A variety of strings were tested, including those of valid one-time passcodes, used one-time passcodes, valid duration-based passcodes, expired duration-based passcodes, and incorrect passcodes. The resulting JSON file was then printed to the terminal and uploaded to AWS for manual inspection.

Lastly, reattempting video uploads was tested by manually placing created videos within the appropriate directory. The process to reupload videos was then executed with statements that displayed all files found in the directory and the status code returned from the PUT request to AWS. The AWS console was also checked manually to ensure the footage had been uploaded successfully.

Integration

In order to verify that single use passcodes worked end to end, a single use passcode was generated with a known email address within the web app. It was verified that this code and the nicknamed entered appeared under the single use passcodes header. This code was then received by email on the specified account. After waiting for at least a minute to ensure the Raspberry Pi had updated, the code was entered on the keypad and it was verified that the box unlocked. After waiting for 30 seconds, it was verified that the box was locked. Checking the web app after showed that the video was present on the video tab with the nickname entered and the exact time that the code was entered displaying. On the password manager screen, it was verified that the code had been moved from single use passcodes to invalid passcodes. This same passcode was then entered into the keypad a second time and it was verified that the box did not unlock. It was also verified that no additional video footage was uploaded and displayed in the web app.

In order to verify that a valid multi use passcode functions as intended, a multi use passcode was generated with a known email address along with a datetime one month in the future within the web app. It was verified that this code, the associated nickname, and the entered datetime appeared under the multi use passcodes header. It was also verified that this code was received by email on the specified account. After waiting for at least a minute to ensure the Raspberry Pi had updated, the code was entered on the keypad and it was verified that the box unlocked. After waiting for 30 seconds, it was verified that the box was locked. Checking the web app after showed that the code and associated information was still present under the multi use passcode header. The process of entering the code, opening the box, verifying that it locked after 30 seconds, and verifying that the code was still listed under the same header was repeated two additional times.

In order to verify that an invalid multi use passcode does not unlock the system, a multi use passcode was generated with a known email address along with a datetime one month in the past within the web app. It was verified that this code, the associated nickname, and the entered datetime appeared under the multi use passcodes header. It was also verified that this code was received by email on the specified account. After waiting for at least a minute to ensure the Raspberry Pi had updated, the code was entered on the keypad and it was verified that the box did not unlock.

In order to verify that a deleted passcode does not unlock the system, a single use passcode was generated with a known email address within the web app. It was verified that the code and the nickname entered appeared under the single use passcodes header. It was also verified that this code was received by email on the specified account. This code was then deleted by pressing the delete button next to the code from within the passcode manager. After waiting for at least a minute to ensure the Raspberry Pi had updated, the code was entered on the keypad and it was verified that the system did not unlock.

Final Results

The group was able to show all of the functionality listed in the rubric below. While the system did not include some features that would result in a more complete product, none of these features were listed in the rubric as they were outside the scope a single semester project. The system does allow for multiple levels of user access. Single-use, time-limited, and superuser codes were implemented. These codes serve separate purposes and are distinguished both on the web application and the database. In addition, these codes can be managed from the web application, where they can be generated, modified (change nickname), and deleted entirely. The electric strike is attached to a package box for demonstration, which allows for maintaining the box's locked status until a valid passcode is entered. Once this code is entered, the electric strike will unlock, allowing the user access for 20 seconds, after which the electric strike will lock. After that, if the door is closed it will not open until a valid code is entered again. When a valid single use code is entered, the system will record the user's action and timestamp them in addition to opening the door and marking the code invalid. This video footage and timestamp are uploaded to the AWS database and the information is visible from the web application. This fulfills all the criteria that set in the rubric, accomplishing enough for full credit.

Letter Grade	Criteria
A+	• Camera data uploaded to cloud • Hinged top locks upon closing and unlocks with password input • Passwords generated and managed from phone via web application • Lock allows for multiple levels of access (master code, time-limited passcode, single-use passcode) • Timestamp on password input, password generation, and camera footage
A	• Four objectives completed
A-	• Three objectives completed
B	• Two objectives completed
C	• One objective completed
D	• No objectives completed

While the prototype functions and meets all of the standards that were set for the design, the team was unable to implement certain features in ways that would make the project a viable product. A raspberry pi was used in place of a traditional microcontroller such as an MSP432 in order to simplify the camera interface and network processing. This allowed the team to spend time on other aspects of design, rather than implementing multiple communication protocols for an MSP432. However, the raspberry pi is significantly more expensive than a simpler microcontroller, and a manufactured product would need to use the cheapest parts possible that can still perform the vital functionality in order to maintain a competitive cost.

The team also intended to implement the entire embedded system in a low power mode that would only require components to draw power when they were in use, but specific components such as the electric strike, which required power to stay locked and the keypad required the system to consume more power than initially expected. While this did not affect the functionality of the system, it does reduce the viability of the system as a commercial product since the increased power consumption costs the user more money to use than the power consumption of a low power system.

Costs

The calculations for the total cost of the system can be found in the Appendix. The predicted production cost for a single box system came out to $390.25, with actual spending on the projecting totaling $247.27 since many components were reused from previous projects. This left $252.73 remaining in the final budget. If the product were to be mass produced, the estimated cost for a single system would decrease due to decreases in costs from buying components in mass quantities. The calculations for components, seen in the Appendix, shows that the cost for components would decrease from $56.25 per board to $41.94 per board. Replacing the Raspberry Pi with a microcontroller with wireless connectivity would also further reduce costs by at least $20 per device. Lastly, manufacturing a package box at scale would cost significantly less than building a box from scratch using wood, screws, and hinges.

Future Work

If another group wanted to extend this project, it would be logical for them to implement many features that would make this project result in a more complete product. The team attempted to create a feature that would automatically check for video files that have not been uploaded to the cloud when the Raspberry Pi reboots. This would only occur when the system loses both of its sources of power (loss of utility power and drained batteries), so it was deemed to be a nonessential feature. The system was able to run the code upon boot, so the Raspberry Pi is able to recover from total power loss, but any videos that were not uploaded before the power loss will not be uploaded to the AWS database.

To make the keypad slightly more reliable for entering input as well as reducing the amount of power required to run the system, a single pole keypad could be used instead of a matrix encoded keypad [54]. This would require significantly more GPIO pins to operate, but it would allow the microcontroller to remain in a low-power mode for large portions of time where no keys are pressed. This would also simplify the key recognition algorithm by allowing an ISR to be associated with every key and use a rising edge interrupt without any other calculations to tell when a key is pressed.

The improvement that could reduce power cost the most is the usage of an active-low electric lock. The electric strike requires power to remain locked, so it actively draws power while locked. An active low lock would significantly reduce the power draw of the system, and it would allow the system to remain locked in the event that it loses all sources of power.

One additional improvement to help with scaling would be swapping the Raspberry Pi for a TI MSP432 microcontroller. Not only would this microcontroller reduce cost as discussed prior, but this would also take up less physical space and consume less power. However, in order to incorporate this, an additional network process would need to be included to handle the Wi-Fi connections. This would require additional drivers to be written to control this communication. Drivers would also need to be written to communicate with the camera as Camera Serial Interface is not by default compatible in the way it is with the Raspberry Pi. The project would also have to include a file storage system as the TI microcontrollers do not have the same storage capabilities and capacity as the Raspberry Pi.

A further improvement could be made regarding the camera and video recordings. Image recognition software could be written so that the system learns what to expect when a certain user attempts entry. If the owner enters the same code every time, the system will be able to recognize the owner's face and if the code is entered with a face that is not recognized by the system, it could be programmed to either refuse entry and require a manual override or send a notification to the owner to alert them to the action.

References

[1] "Airbnb: Vacation Rentals, Cabins, Beach Houses, Unique Homes & Experiences." https://www.airbnb.com/ (accessed Dec. 14, 2021).

[2] "Future of Online Shopping: Evolving E-Commerce Trends," *Maryville Online*, Apr. 09, 2019. https://online.maryville.edu/blog/future-of-online-shopping/ (accessed Dec. 14, 2021).

[3] "Protect Against Porch Piracy." https://upscapital.com/resources/protect-against-porch-piracy/&cb=1 (accessed Dec. 14, 2021).

[4] "Home Security Systems | Smart Home Automation," *Ring*. https://ring.com (accessed Dec. 14, 2021).

[5] "Amazon.com: Amazon Key | In-Garage Delivery." https://www.amazon.com/Amazon-Key-In-Garage-Delivery/b?ie=UTF8&node=21222091011 (accessed Dec. 14, 2021).

[6] R. P. Ltd, "Raspberry Pi 3 Model B," *Raspberry Pi*. https://www.raspberrypi.com/products/raspberry-pi-3-model-b/ (accessed Dec. 14, 2021).

[7] "IPC Standards | IPC International, Inc." https://www.ipc.org/ipc-standards (accessed Dec. 16, 2021).

[8] E. S. in H. on November 22, 2021, and 7:18 Am Pst, "Global chip shortage: Everything you need to know," *TechRepublic*. https://www.techrepublic.com/article/global-chip-shortage-cheat-sheet/ (accessed Dec. 16, 2021).

[9] E. Dyer, "Dyer: The Chip Shortage Is Totally out of Control," *Car and Driver*, Dec. 08, 2021. https://www.caranddriver.com/features/a38402687/dyer-chip-shortage/ (accessed Dec. 16, 2021).

[10] "IEEE 802.11-2016 - IEEE Standard for Information technology--Telecommunications and information exchange between systems Local and metropolitan area networks--Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications." https://standards.ieee.org/standard/802_11-2016.html (accessed Dec. 16, 2021).

[11] "IPC Standards," *IPC International, Inc.*, Aug. 07, 2020. https://www.ipc.org/ipc-standards (accessed Dec. 16, 2021).

[12] "Standards." https://www.nema.org/standards (accessed Dec. 16, 2021).

[13] "SMT / SMD components & packages, sizes, dimensions, details.," *Electronics Notes*. https://www.electronics-notes.com/articles/electronic_components/surface-mount-technology-smd-smt/packages.php

[14] "MIPI Camera Serial Interface 2 (MIPI CSI-2)." https://www.mipi.org/specifications/csi-2 (accessed Dec. 16, 2021).

[15] "HTTP Specifications and Drafts." https://www.w3.org/Protocols/Specs.html (accessed Dec. 16, 2021).

[16] “BarrCCodingStandard2018.pdf.” Accessed: Dec. 16, 2021. [Online]. Available: https://collab.its.virginia.edu/access/content/group/0b629dc5-7f19-41fc-bf8f-553c1ea9651d/BarrCCodingStandard2018.pdf

[17] “KiCad EDA - Schematic Capture & PCB Design Software.” https://www.kicad.org/ (accessed Dec. 15, 2021).

[18] “NI VirtualBench.” https://www.ni.com/en-us/shop/hardware/products/virtualbench-all-in-one-instrument.html (accessed Dec. 14, 2021).

[19] “Cloud Object Storage – Amazon S3 – Amazon Web Services,” *Amazon Web Services, Inc.* https://aws.amazon.com/s3/ (accessed Dec. 14, 2021).

[20] “Visual Studio Code - Code Editing. Redefined.” https://code.visualstudio.com/ (accessed Dec. 15, 2021).

[21] K. Zhao and L. Ge, “A Survey on the Internet of Things Security,” in *2013 Ninth International Conference on Computational Intelligence and Security*, Dec. 2013, pp. 663–667. doi: 10.1109/CIS.2013.145.

[22] J. Veijalainen, D. Kozlov, and Y. Ali, “Security and Privacy Threats in IoT Architectures,” presented at the 7th International Conference on Body Area Networks, Oslo, Norway, 2012. doi: 10.4108/icst.bodynets.2012.250550.

[23] “On the Secure Hash Algorithm family.pdf.”

[24] W. Du, *Computer & Internet Security*.

[25] “Amazon S3 Security Features - Amazon Web Services.” https://aws.amazon.com/s3/security/ (accessed Sep. 17, 2021).

[26] “Welcome to Flask — Flask Documentation (2.0.x).” https://flask.palletsprojects.com/en/2.0.x/ (accessed Dec. 14, 2021).

[27] “Battery Recycling is Important for Environmental Health,” *Gallegos Sanitation*, Jan. 20, 2020. https://gsiwaste.com/battery-recycling-is-important-for-environmental-health/ (accessed Dec. 16, 2021).

[28] “Guide for Recycling Printed Circuit Boards | PCB Recycling,” *mcl*, Jul. 21, 2018. https://www.mclpcb.com/blog/environmental-impact-semiconductor/ (accessed Dec. 16, 2021).

[29] O. US EPA, “Identifying Greener Electronics,” Nov. 14, 2014. https://www.epa.gov/greenerproducts/identifying-greener-electronics (accessed Dec. 16, 2021).

[30] “Amazon S3.” https://aws.amazon.com/pm/serv-s3/?trk=ps_a134p000004f2aOAAQ&trkCampaign=acq_paid_search_brand&sc_channel=PS&sc_campaign=acquisition_US&sc_publisher=Google&sc_category=Storage&sc_country=US&sc_geo=NAMER&sc_outcome=acq&sc_detail=aws%20s3&sc_content=S3_e&sc_matchtype=e&sc_segment=488982706719&sc_medium=ACQ-P|PS-GO|Brand|Desktop|SU|Storage|S3|US|EN|Text&s_kwcid=AL!4422!3!488982706719!e!!g!!aws%20s3&ef_id=Cj0KCQiAweaNBhDEARIsAJ5hwbduMU2ddROeBSBIeXka2pZrszZ2oicVync5-KxsbmrORPX6-tMXRrMaAikWEALw_wcB:G:s&s_kwcid=AL!4422!3!488982706719!e!!g!!aws%20s3 (accessed Dec. 15, 2021).

[31] J. Johnson, H. J. C. Letourneur, T. Unadkat, C. Dow, and C. Kim, “Vedeo Recording Triggered By A Smart Lock Device,” US 11,072,945 B2

[32] P. W. Dent and J. Skubic, “Security System,” US 7,114,178 B2

[33] G. Smith, M. Fitzpatrick, T. Celinski, M. Richards, and J. Bartucci, “Networked And Camera Enabled Locking Devices,” US 10,757,371 B2

[34] “Raspberry Pi 3 Power Requirements.” https://linuxhint.com/raspberry_pi_3_power_requirements/ (accessed Dec. 14, 2021).

[35] “Create Your Own Battery Backup Power Supplies - Projects.” https://www.allaboutcircuits.com/projects/battery-backup-power-supplies/ (accessed Dec. 14, 2021).

[36] “15V Wall-Plug Switching Adapter.” Accessed: Dec. 14, 2021. [Online]. Available: https://www.cui.com/product/resource/swi25-n.pdf

[37] “POWEROWL AA Rechargeable Batteries, 2800mAh High Capacity Batteries 1.2V NiMH Low Self Discharge, Pack of 16.” http://www.powerowl.net/product/product-87-20.html (accessed Dec. 14, 2021).

[38] “Ideal Diode Controller Datasheet.”

[39] “Primer on PowerPath Controllers, Ideal Diodes & Prioritizers | Analog Devices.” https://www.analog.com/en/technical-articles/primer-on-powerpath-controllers-ideal-diodes-prioritizers.html (accessed Dec. 14, 2021).

[40] “PowerPath, Ideal Diodes, and Load Switches | Analog Devices.” https://www.analog.com/en/products/monitor-control-protection/powerpath-ideal-diodes-load-switches.html (accessed Dec. 14, 2021).

[41] “MOSFET Datasheet (For Ideal Diode Controller).” Accessed: Dec. 14, 2021. [Online]. Available: https://componentsearchengine.com/Datasheets/2/BSC026NE2LS5ATMA1.pdf

[42] “A Comparison Between Dc Switching Regulators and Linear Regulators,” *CUI Inc*, Dec. 08, 2020. https://www.cui.com/blog/a-comparison-between-dc-switching-regulators-and-linear-regulators (accessed Dec. 15, 2021).

[43] “Linear vs. Switching Regulators | Renesas.” https://www.renesas.com/us/en/products/power-power-management/linear-vs-switching-regulators (accessed Dec. 15, 2021).

[44] “5V Switching Regulator Datasheet.” Accessed: Dec. 14, 2021. [Online]. Available: https://www.mouser.com/datasheet/2/256/MAX730A-MAX744A-258182.pdf

[45] “Schottky Rectifier Diode Datasheet (For Solenoid Driver and Switching Regulator).” Accessed: Dec. 14, 2021. [Online]. Available: https://componentsearchengine.com/Datasheets/1/SD0805S020S1R0.pdf

[46] “Inductor Datasheet (For Switching Regulator).” Accessed: Dec. 14, 2021. [Online]. Available: https://www.bourns.com/docs/Product-Datasheets/SRN8040TA.pdf

[47] “C4 Capacitor Datasheet (For 5V Switching Regulator).” Accessed: Dec. 14, 2021. [Online]. Available: https://componentsearchengine.com/Datasheets/2/860020672005.pdf

[48] “12V Linear Regulator Datasheet,” p. 25.

[49] P. Millett, “What’s the Best Way to Drive a Solenoid?,” *Electronic Design*, May 31, 2018. https://www.electronicdesign.com/industrial-automation/article/21806574/whats-the-best-way-to-drive-a-solenoid (accessed Dec. 15, 2021).

[50] “Solenoid Drivers | Overview | Motor Drivers | TI.com.” https://www.ti.com/motor-drivers/solenoid/overview.html (accessed Dec. 15, 2021).

[51] “Solenoid Driver Datasheet.” Accessed: Dec. 14, 2021. [Online]. Available: https://www.ti.com/lit/ds/symlink/drv103.pdf?ts=1639540477611

[52] “Barrel Jack Datasheet.” Accessed: Dec. 14, 2021. [Online]. Available: https://www.cuidevices.com/product/resource/pj-002a.pdf

[53] “Electric Strike/Rechargeable Batteries Connector Datasheet.” Accessed: Dec. 14, 2021. [Online]. Available: https://www.we-online.com/katalog/datasheet/691137710002.pdf

[54] “Keypad Datasheet.”

[55] “Keypad Connector Datasheet.” Accessed: Dec. 14, 2021. [Online]. Available: https://www.molex.com/pdm_docs/sd/530480810_sd.pdf

[56] “Molex Wire Datasheet (Keypad Wire).” Accessed: Dec. 16, 2021. [Online]. Available: http://www.literature.molex.com/SQLImages/kelmscott/Molex/PDF_Images/987651-5322.PDF

[57] “TVS Diodes | Surface Mount Diodes - Littelfuse.” https://www.littelfuse.com/products/tvs-diodes.aspx (accessed Dec. 15, 2021).

[58] “TVS Diodes Datasheet,” p. 11.

[59] “Raspberry Pi Connector Datasheet.” Accessed: Dec. 14, 2021. [Online]. Available: https://cdn.sparkfun.com/assets/0/b/8/5/2/DS-16763-2_X_20_Pin_Extended_GPIO_Header_-_Female_-_16mm_7.30mm.pdf

[60] P. Sethi and S. R. Sarangi, “Internet of Things: Architectures, Protocols, and Applications,” *J. Electr. Comput. Eng.*, vol. 2017, pp. 1–25, 2017, doi: 10.1155/2017/9324035.

[61] “Intelligent Diagramming,” *Lucidchart*. https://www.lucidchart.com (accessed Dec. 15, 2021).

[62] “pigpio library.” http://abyz.me.uk/rpi/pigpio/ (accessed Dec. 15, 2021).

[63] N. Lohmann, *JSON for Modern C++*. 2021. Accessed: Dec. 15, 2021. [Online]. Available: https://github.com/nlohmann

[64] *AWS SDK for C++*. Amazon Web Services, 2021. Accessed: Dec. 15, 2021. [Online]. Available: https://github.com/aws/aws-sdk-cpp

[65] C. Verstraeten, *RaspiCam: C++ API for using Raspberry camera (with OpenCV)*. 2021. Accessed: Dec. 15, 2021. [Online]. Available: https://github.com/cedricve/raspicam

[66] R. P. Ltd, “Buy a Raspberry Pi Camera Module 2,” *Raspberry Pi*. https://www.raspberrypi.com/products/camera-module-v2/ (accessed Dec. 15, 2021).

[67] *opencv/opencv*. OpenCV, 2021. Accessed: Dec. 15, 2021. [Online]. Available: https://github.com/opencv/opencv

[68] “realtime:start [Wiki].” https://wiki.linuxfoundation.org/realtime/start (accessed Dec. 15, 2021).

[69] “What is REST,” *REST API Tutorial*. https://restfulapi.net/ (accessed Dec. 15, 2021).

[70] “AWS SDK for Python,” *Amazon Web Services, Inc.* https://aws.amazon.com/sdk-for-python/ (accessed Dec. 15, 2021).

[71] yWorks company the diagramming, “yEd Graph Editor,” *yWorks, the diagramming experts*. https://www.yworks.com/products/yed (accessed Dec. 15, 2021).

[72] “flask-mail — Flask-Mail 0.9.1 documentation.” https://pythonhosted.org/Flask-Mail/ (accessed Dec. 15, 2021).

[73] “SMTP Email Relay Services - SMTP.com,” *SMTP.com*. https://www.smtp.com/ (accessed Dec. 15, 2021).

[74] “SQLAlchemy - The Database Toolkit for Python.” https://www.sqlalchemy.org/ (accessed Dec. 15, 2021).

[75] “Python Secrets Module,” *Python Documentation*, Dec. 15, 2021. https://docs.python.org/3/library/secrets.html

[76] "Python Hashlib Module," *Python Documentation*, Dec. 15, 2021. https://docs.python.org/3/library/hashlib.html

[77] "How do I power my Raspberry Pi?," *The Pi Hut*. https://thepihut.com/blogs/raspberry-pi-tutorials/how-do-i-power-my-raspberry-pi (accessed Dec. 15, 2021).

[78] "online-tools," *online-tools*. https://emn178.github.io/online-tools/sha256.html

[79] "SHA256 Online." https://emn178.github.io/online-tools/sha256.html (accessed Dec. 15, 2021).

Appendix

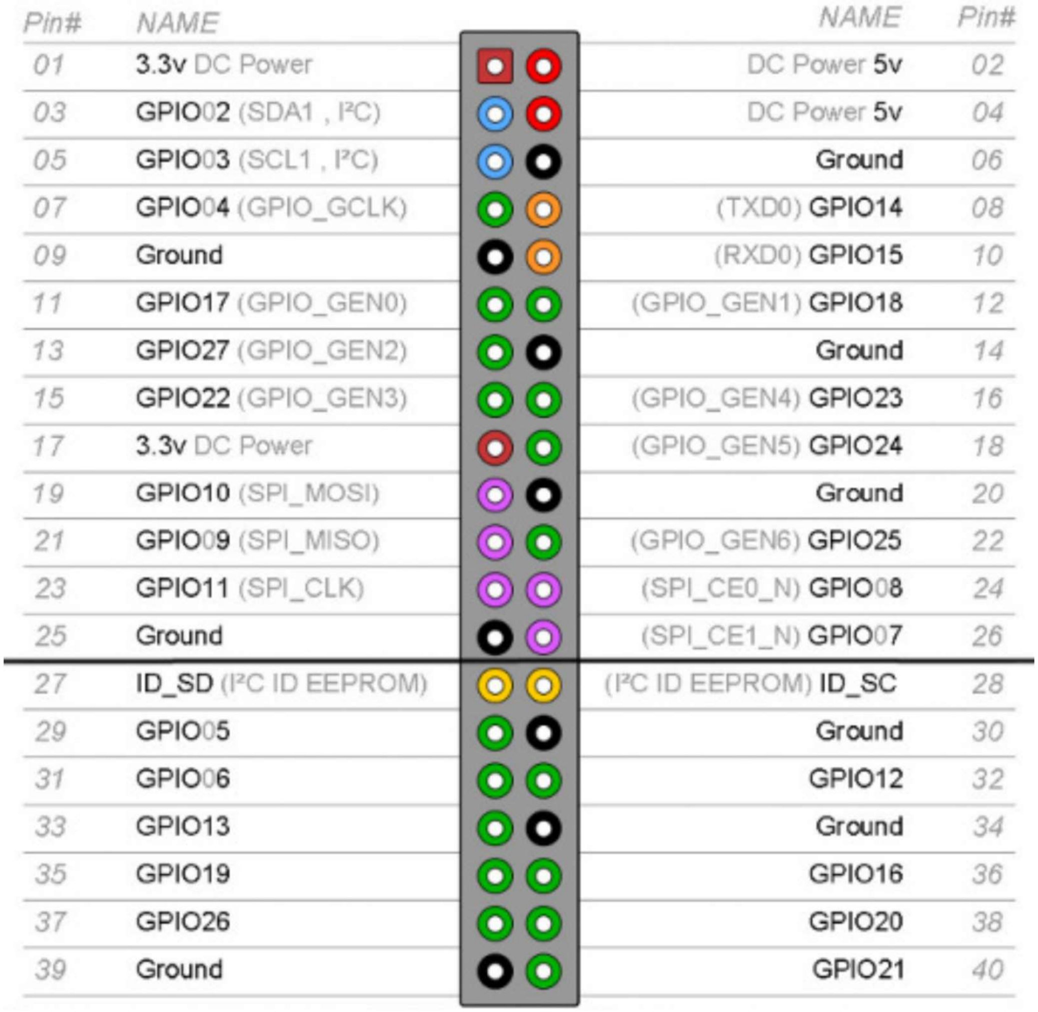
Battery Lifetime Calculations

The selected batteries are rated at 2800 mAh [37]. Based upon the current draw measurements of the circuit and this rating, the rechargeable battery power supply lifetime was calculated. Using a National Instruments VirtualBench, the Raspberry Pi's current draw was measured to be 125 mA, and the electric strike's current draw was measured to be 156 mA [18].

$$Power\ Supply\ Lifetime = \frac{2800\ mAh}{(125\ mA) + (156\ mA)} = 9.9644\ hrs$$

Based upon the batteries used for this project and the current draw from this system, the rechargeable batteries should power the system for roughly 10 hours, when power is lost from the wall-plug power supply.

Raspberry Pi 3 Model B V1.2 Pinout



Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I²C)	DC Power 5v	04
05	GPIO03 (SCL1 , I²C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)	(I²C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

Figure 50 Raspberry Pi 3 Model B V1.2 Pinout

Picture of Manufactured PCB

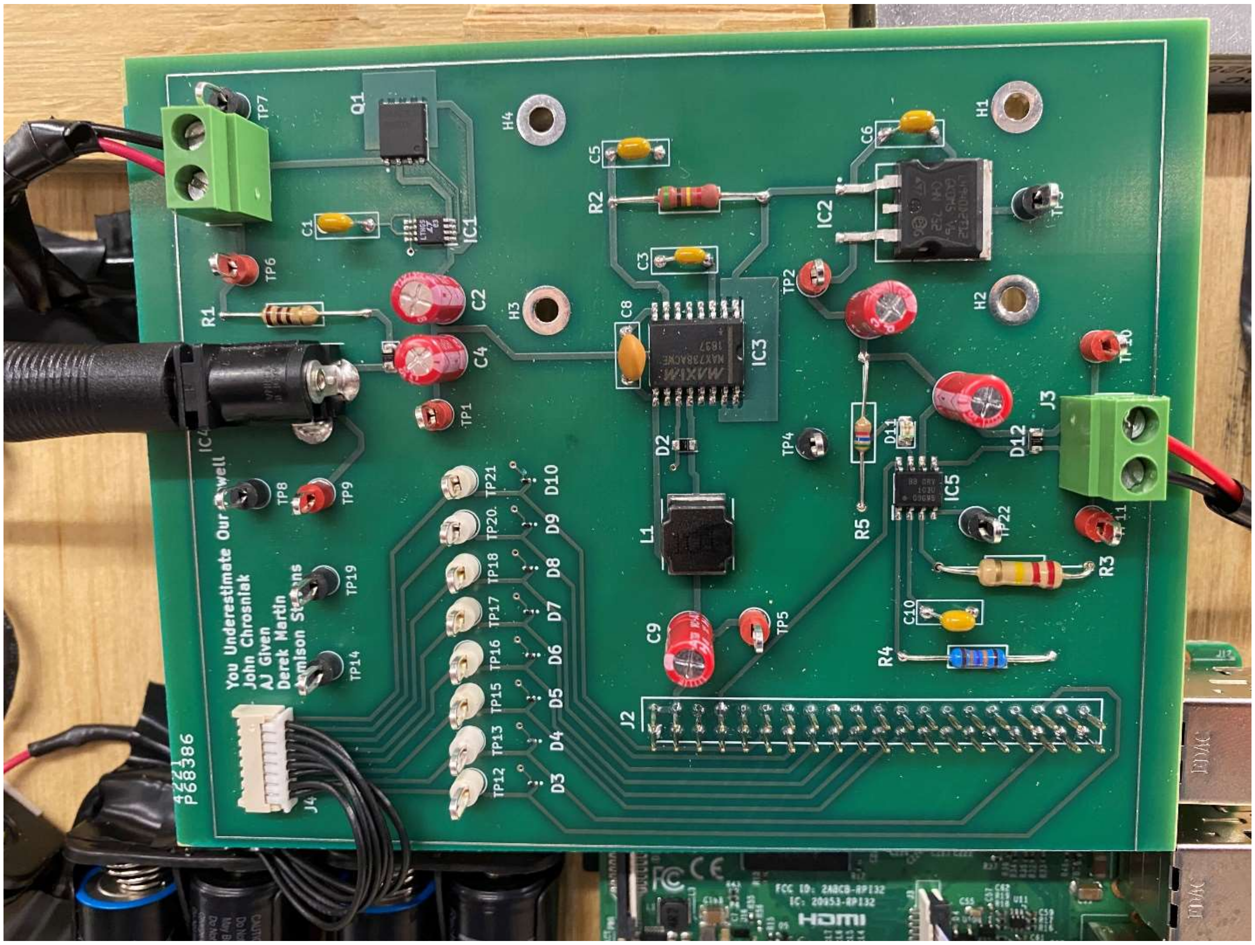


Figure 51 Manufactured PCB

Pictures of Package Box Lock Mechanism

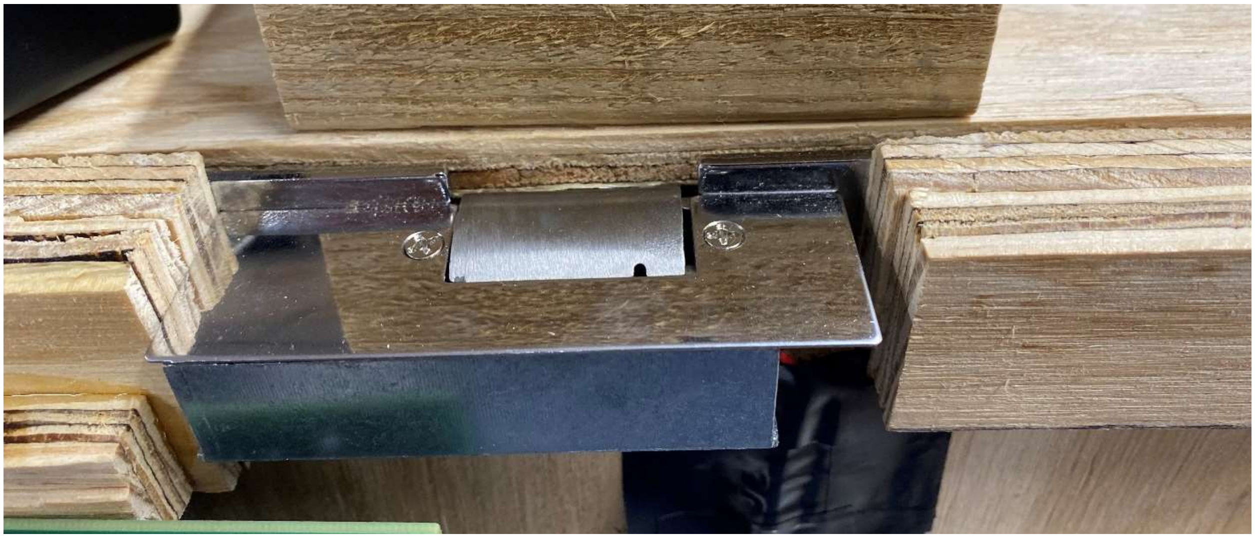


Figure 52 Front-View of Package Box Locking Mechanism

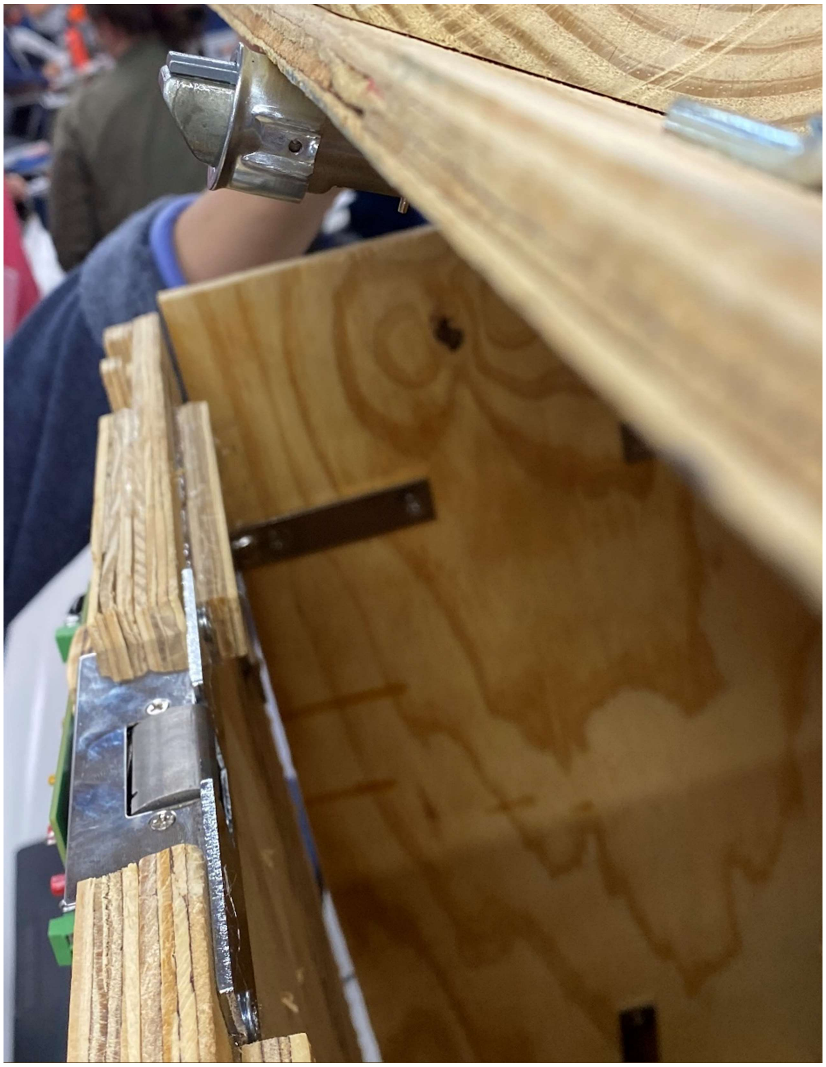


Figure 53 Side-View of Package Box Locking Mechanism

Picture of Integrated System

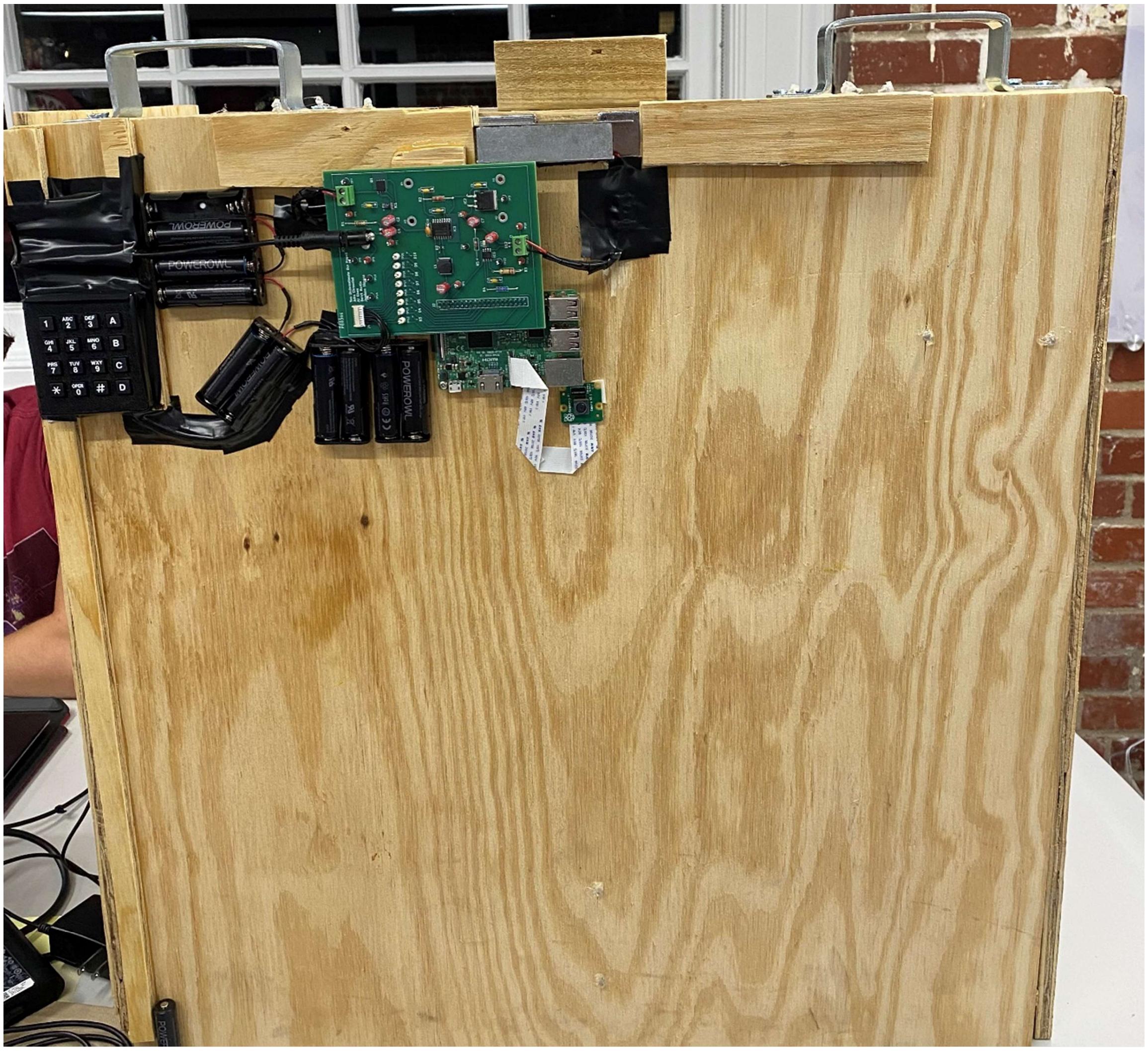


Figure 54 Bouncer Locking System with Lid Closed

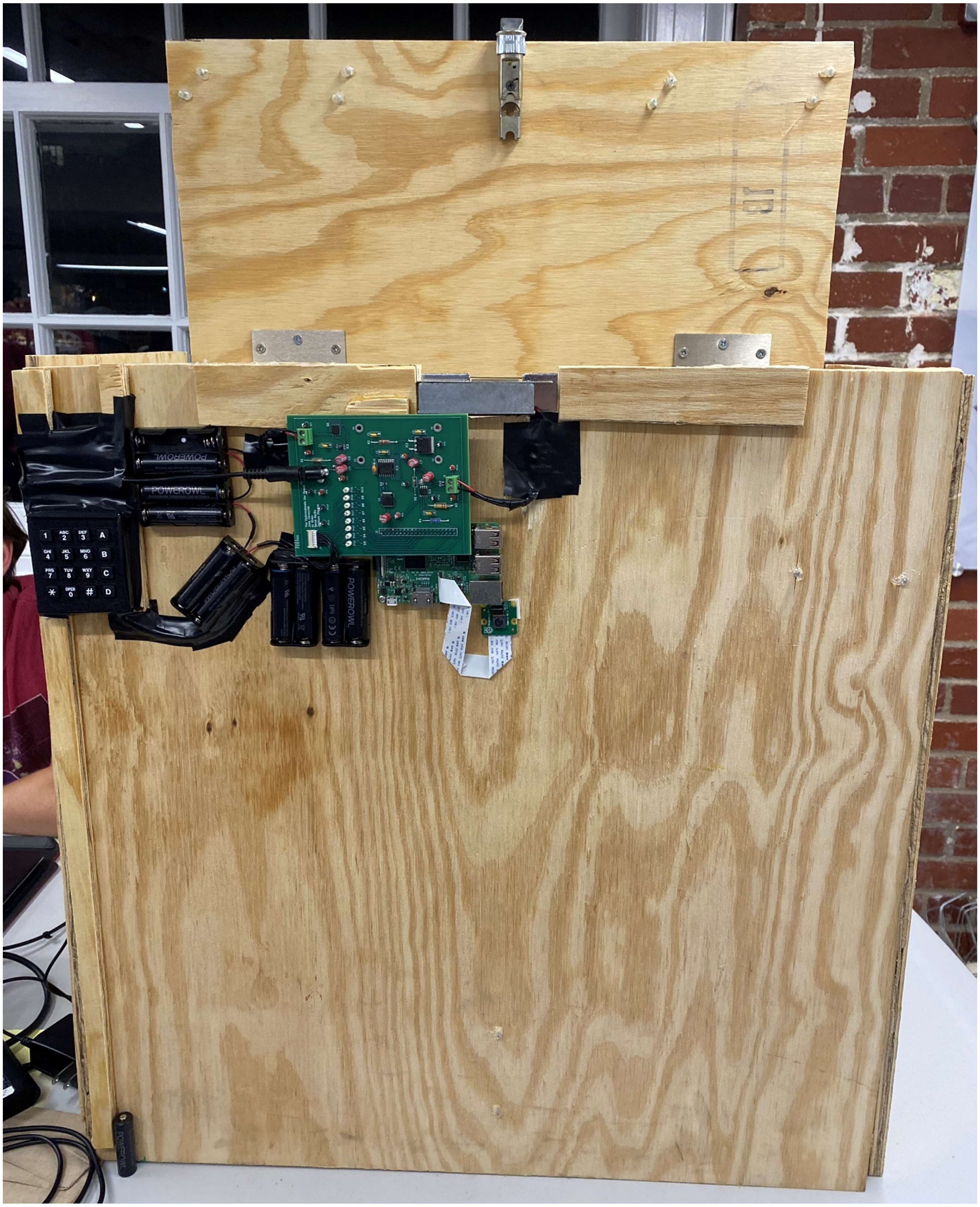
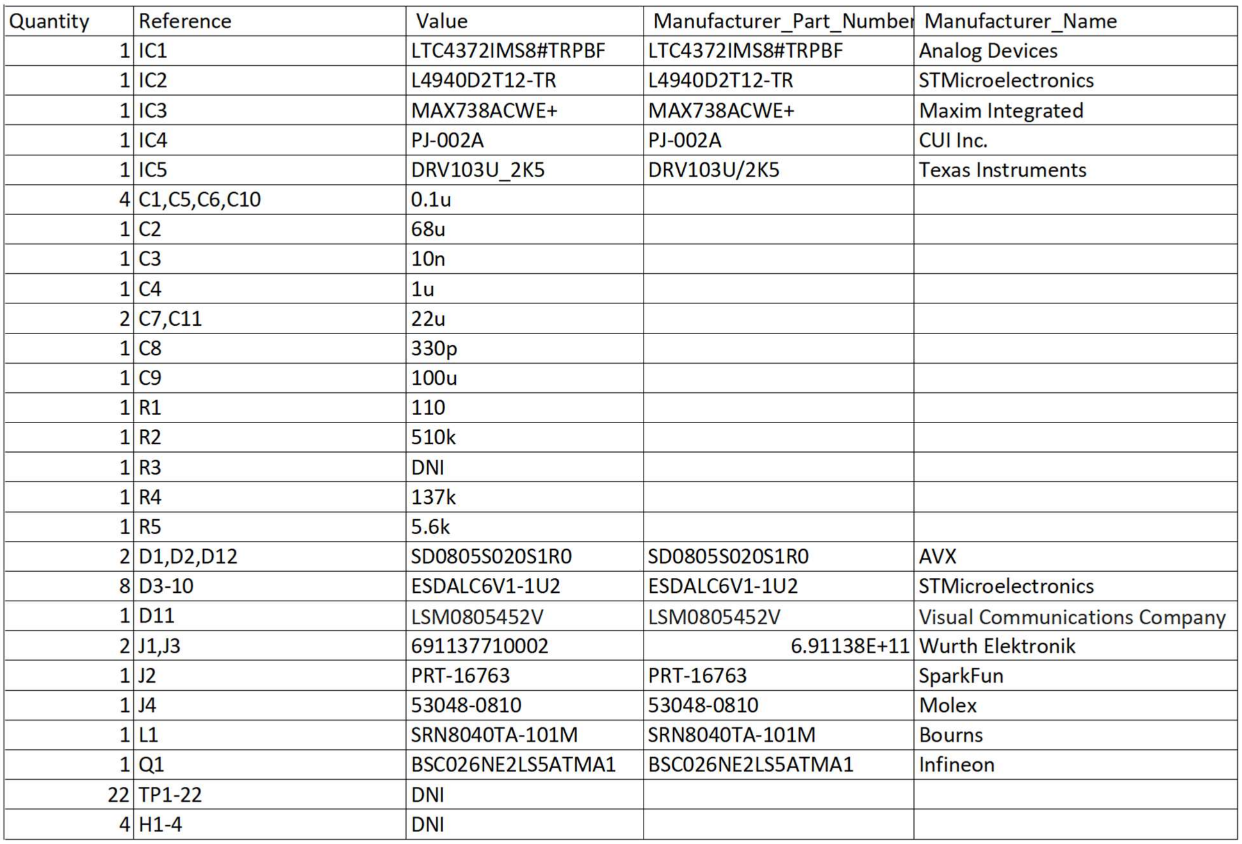


Figure 55 Bouncer Locking System with Lid Opened

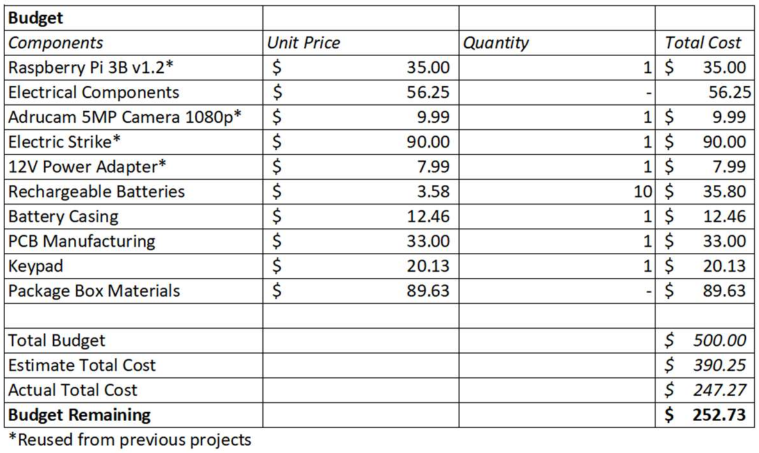
Bill of Materials (BOM)



Quantity	Reference	Value	Manufacturer_Part_Number	Manufacturer_Name
1	IC1	LTC4372IMS8#TRPBF	LTC4372IMS8#TRPBF	Analog Devices
1	IC2	L4940D2T12-TR	L4940D2T12-TR	STMicroelectronics
1	IC3	MAX738ACWE+	MAX738ACWE+	Maxim Integrated
1	IC4	PJ-002A	PJ-002A	CUI Inc.
1	IC5	DRV103U_2K5	DRV103U/2K5	Texas Instruments
4	C1,C5,C6,C10	0.1u		
1	C2	68u		
1	C3	10n		
1	C4	1u		
2	C7,C11	22u		
1	C8	330p		
1	C9	100u		
1	R1	110		
1	R2	510k		
1	R3	DNI		
1	R4	137k		
1	R5	5.6k		
2	D1,D2,D12	SD0805S020S1R0	SD0805S020S1R0	AVX
8	D3-10	ESDALC6V1-1U2	ESDALC6V1-1U2	STMicroelectronics
1	D11	LSM0805452V	LSM0805452V	Visual Communications Company
2	J1,J3	691137710002	6.91138E+11	Wurth Elektronik
1	J2	PRT-16763	PRT-16763	SparkFun
1	J4	53048-0810	53048-0810	Molex
1	L1	SRN8040TA-101M	SRN8040TA-101M	Bourns
1	Q1	BSC026NE2LS5ATMA1	BSC026NE2LS5ATMA1	Infineon
22	TP1-22	DNI		
4	H1-4	DNI		

Figure 56 Bill of Materials

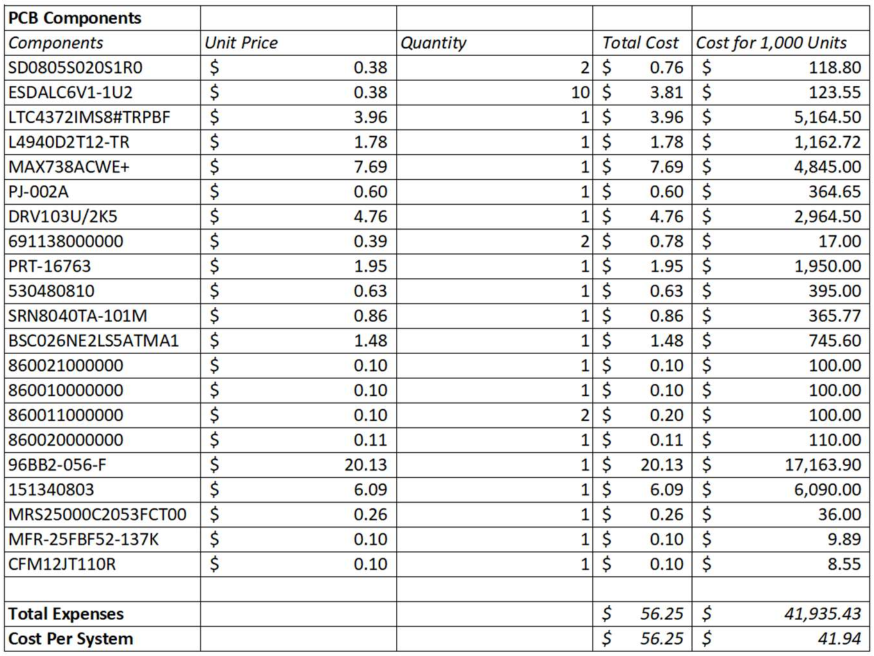
Budget Calculations



Budget			
Components	*Unit Price*	*Quantity*	*Total Cost*
Raspberry Pi 3B v1.2*	$ 35.00	1	$ 35.00
Electrical Components	$ 56.25	-	56.25
Adrucam 5MP Camera 1080p*	$ 9.99	1	$ 9.99
Electric Strike*	$ 90.00	1	$ 90.00
12V Power Adapter*	$ 7.99	1	$ 7.99
Rechargeable Batteries	$ 3.58	10	$ 35.80
Battery Casing	$ 12.46	1	$ 12.46
PCB Manufacturing	$ 33.00	1	$ 33.00
Keypad	$ 20.13	1	$ 20.13
Package Box Materials	$ 89.63	-	$ 89.63
Total Budget			*$ 500.00*
Estimate Total Cost			*$ 390.25*
Actual Total Cost			*$ 247.27*
Budget Remaining			**$ 252.73**

*Reused from previous projects

Figure 57 Final Budget Calculations



PCB Components				
Components	*Unit Price*	*Quantity*	*Total Cost*	*Cost for 1,000 Units*
SD0805S020S1R0	$ 0.38	2	$ 0.76	$ 118.80
ESDALC6V1-1U2	$ 0.38	10	$ 3.81	$ 123.55
LTC4372IMS8#TRPBF	$ 3.96	1	$ 3.96	$ 5,164.50
L4940D2T12-TR	$ 1.78	1	$ 1.78	$ 1,162.72
MAX738ACWE+	$ 7.69	1	$ 7.69	$ 4,845.00
PJ-002A	$ 0.60	1	$ 0.60	$ 364.65
DRV103U/2K5	$ 4.76	1	$ 4.76	$ 2,964.50
691138000000	$ 0.39	2	$ 0.78	$ 17.00
PRT-16763	$ 1.95	1	$ 1.95	$ 1,950.00
530480810	$ 0.63	1	$ 0.63	$ 395.00
SRN8040TA-101M	$ 0.86	1	$ 0.86	$ 365.77
BSC026NE2LS5ATMA1	$ 1.48	1	$ 1.48	$ 745.60
860021000000	$ 0.10	1	$ 0.10	$ 100.00
860010000000	$ 0.10	1	$ 0.10	$ 100.00
860011000000	$ 0.10	2	$ 0.20	$ 100.00
860020000000	$ 0.11	1	$ 0.11	$ 110.00
96BB2-056-F	$ 20.13	1	$ 20.13	$ 17,163.90
151340803	$ 6.09	1	$ 6.09	$ 6,090.00
MRS25000C2053FCT00	$ 0.26	1	$ 0.26	$ 36.00
MFR-25FBF52-137K	$ 0.10	1	$ 0.10	$ 9.89
CFM12JT110R	$ 0.10	1	$ 0.10	$ 8.55
Total Expenses			*$ 56.25*	*$ 41,935.43*
Cost Per System			*$ 56.25*	*$ 41.94*

Figure 58 Component Calculations