SentryEye

Vehicle Security Device

March 23, 2021

Derek Byrne
Ryan LaMarche
Michael Osei

Prof. Joshua Cuneo
Prof. Stephen Bitar

Table of Contents

Table of Tables

Table of Figures

Abstract

In an increasingly interconnected and complex world, security remains at the forefront of people's minds concerning physical safety as well as the safety of one's belongings. There are a variety of devices designed to integrate one's messages, email, financials, weather, and even home thermostat data with a seamless user experience accessible from anywhere with an Internet connection. This project demonstrates that one's approach to vehicle security should be no different with SentryEye. The cloud enabled SentryEye device captures pictures and videos of any suspected burglar near a vehicle, with built-in deterrent mechanisms and notifications to provide a seamless user experience.

1 Executive Summary

In an increasingly interconnected and complex world, security remains at the forefront of people's minds concerning physical safety as well as the safety of one's belongings. There are a variety of devices designed to integrate one's messages, email, financials, weather, and even home thermostat data with a seamless user experience accessible from anywhere with an Internet connection. The purpose of this report's product proposal is to demonstrate that one's approach to vehicle security should be no different with SentryEye. The cloud-enabled SentryEye device captures pictures and videos of any suspected burglar near a vehicle, with built-in light-enabled and auditory alarms to notify the vehicle owner via a seamless user dashboard.

1.1 The Problem

Our initial investigation involved the desire to ensure WPI's university campus safety and security, which provided us with significant amounts of data on prominent crimes that take place on campus. Many sources appearing in the search presented information on burglary and theft of motor vehicles [1]. According to a study conducted in 2014, 57% of more than 50,000 of on-campus related crimes were property crimes, which include burglary, motor vehicle theft, and robbery [8]. We also identified that properties such as motor vehicles are an important point of concern. One study found that 46.8 percent of students among the 214 National Universities that reported such data to the US brought cars to their campus for the 2016-2017 academic years, representing a significant portion of targeted property crimes [5].

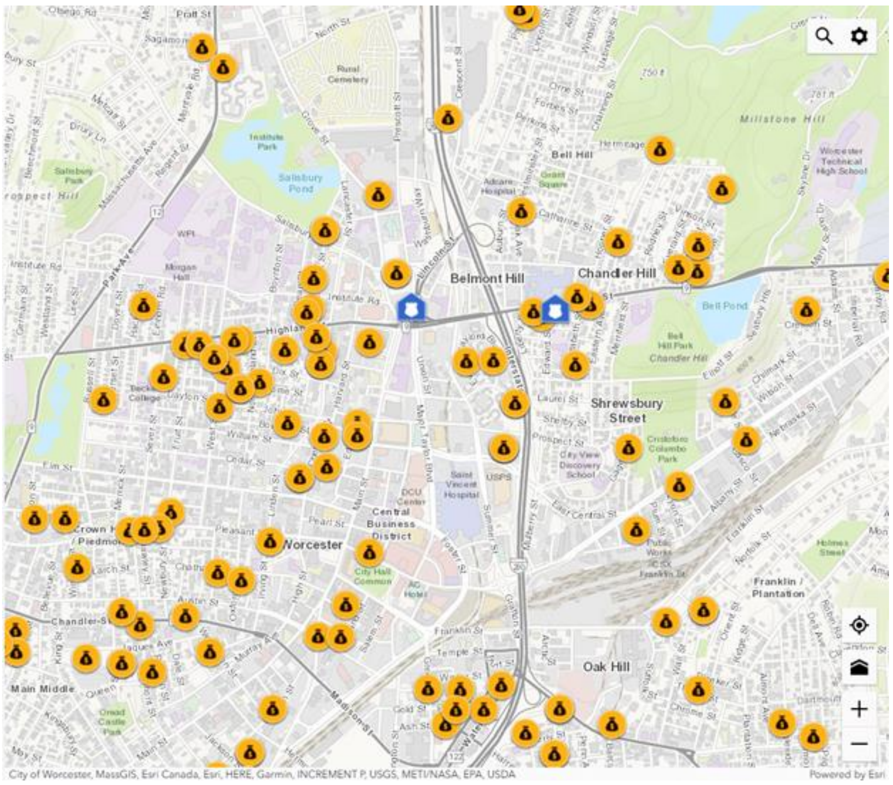


Figure 1: Reported Theft of/from a Motor Vehicle in Worcester Area Surrounding WPI Campus from February 1, 2021 to March 22, 2021 [6]

Theft of/from vehicles remains a constant threat, especially on university campuses. In Figure 1 shown above, a map of reported incidents of theft of/from a vehicle is displayed in the area surrounding WPI's campus where many students park their vehicles. Each marker indicates a theft in the immediate area surrounding WPI campus from February 1, 2021 to March 22, 2021 alone [6]. This data demonstrates that even at WPI, the concern for the safety and security of personal property remains an issue. These preliminary results prompted the investigation of existing systems that either combat or address such crimes. This initial investigation and market research have led us to conclude that there is a high frequency of burglary and theft of personal property of students attending college or university, particularly from vehicles. Students with cars on streets near the campus are nearly always at risk of burglary, especially if their property is not placed in a secure location under constant supervision. Students may only use their vehicles when buying groceries or visiting home a few times a week. The gaps between each car visit present a myriad of opportunities for vehicle theft to go unnoticed. As such, there is a need for a more versatile and robust security system.

Designed and built with everyday vehicle owners in mind, SentryEye provides an affordable solution within a highly functional security package for any vehicle. The device is modular such that multiple devices can be added and utilized via a single SentryEye user-friendly dashboard. Using any web browser or mobile device, the user can receive and view live images and videos within their vehicle whenever a break-in attempt is detected. In addition, SentryEye provides location data to detail where the break-in attempt took place along with pictures of the perpetrator's face.

1.2 Market Research

Some devices already address vehicle safety, and as such pose a potential competitive threat to SentryEye's place in the market. In terms of functionality, OwlCam is the closest competitor to SentryEye. As an all-in-one security approach, OwlCam provides functionality such as unlimited phone notifications, voice control, video history of the area surrounding the interior of the vehicle as well as the area in front of the windshield. However, OwlCam has a rather expensive entry price at $349.00 along with a $215 annual fee for the LTE connectivity service provided. OwlCam also provides dashcam functionality for the area in front of the vehicle in the event of an accident. SentryEye's design focuses on security alone in a small, modular fashion rather than the bulky, "all-in-one" dashcam approach taken by the OwlCam, all at a much lower price point of $50-$60. Two other competitors studied include the VAVA and Crosstour dashcams with vastly different entry price points as standards of comparison.

The VAVA dashcam records the area in front of the vehicle with a swivel mount that allows the user to turn the camera toward areas of interest. Though primarily designed to record accidents, VAVA provides security features in which the device will record when the parked car is shaken by an intruder. The device costs $120, which is still double the price of the SentryEye. Unlike SentryEye, the VAVA dashcam is also incapable of sending photos or videos to the user remotely, instead requiring the user to eject the internal microSD card to view the photos and videos.

The last competitor is the Crosstour dashcam, which is a small, inexpensive dashcam with an initial price point of only $30. The camera includes a G-force sensor that begins to record in the case of an accident or jostling of the vehicle. It also features a wide-angle lens and motion detection for security purposes, thus posing competitive risk to SentryEye. However, the device does not function when the car is off and it cannot detect intruders inside the car (only outside). Photos cannot be sent remotely and must be physically removed from the device just like the VAVA dashcam. SentryEye can send images and videos remotely wherever a cellular connection is available, is solely focused on detecting intruders inside the vehicle, and can perform extensively even when the vehicle is no longer running.

An extensive competitor value analysis was performed concerning factors such as device picture quality, hands-off maintenance, affordability, and external communication features. The results of this analysis indicate that SentryEye fits within a significant market gap demanding a balance of features and cost when compared to competitors' products.

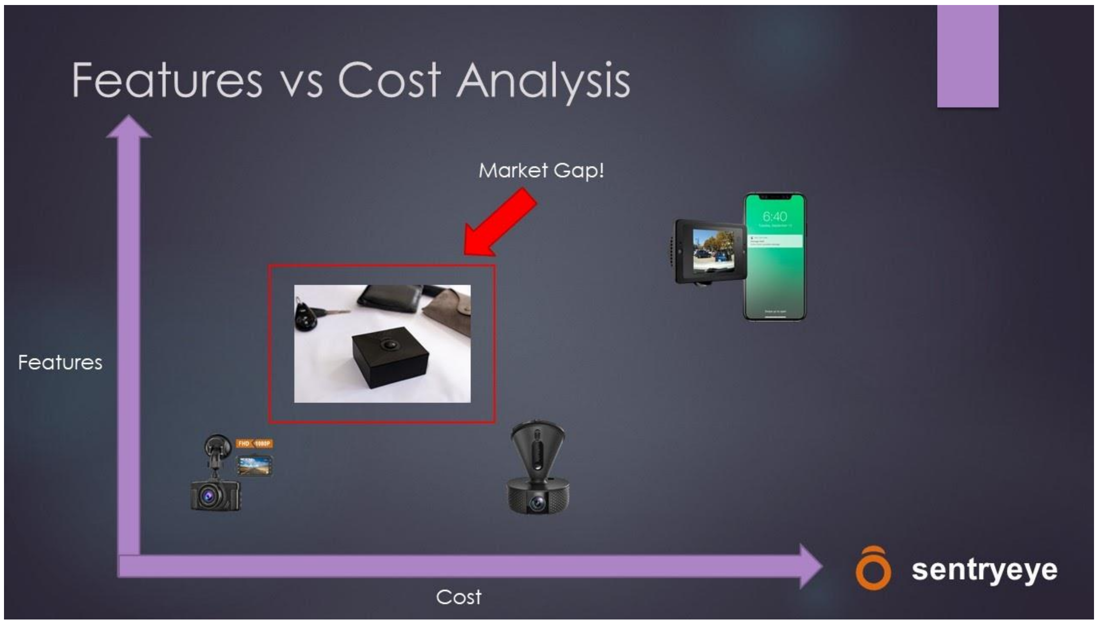


Figure 2: Cost vs. Features Chart for SentryEye

1.3 Design Approach

SentryEye was designed such that it captures and sends images without requiring additional user action. There should be little to no maintenance needed from the user. As such, SentryEye should be a small, compact device that is unobtrusive to the operation of the vehicle and can be fastened to the surface on which it is mounted. In the context of power and operation, the device should operate using low power and the device should be able to self-charge for convenience. The shape of the SentryEye, in addition to being small, should be aesthetically convenient and blend into the mountable locations such as the car's dashboard. All of the hardware requirements to meet these needs are outlined in the figure below.

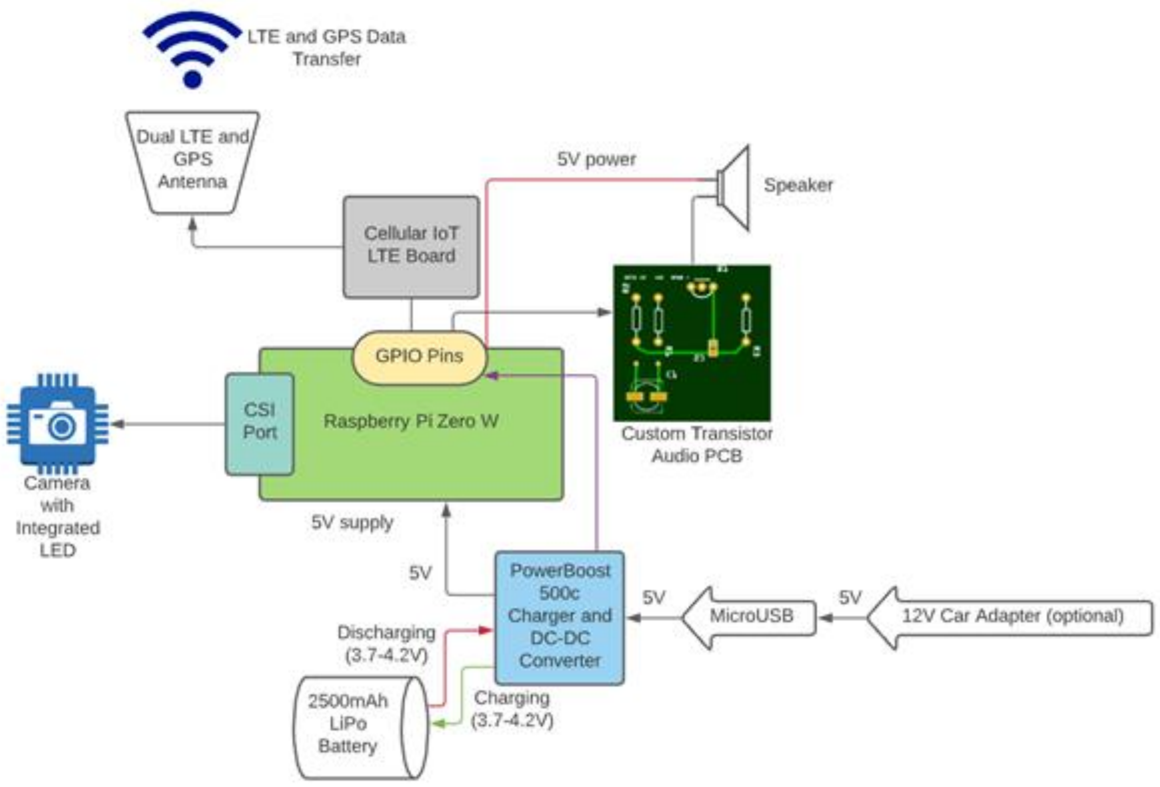


Figure 3: Top-Level Block Diagram of SentryEye System Components

As shown above, the SentryEye device accepts any 5V USB or 12V car adapter as a power input. The PowerBoost 500c module acts as both the charging circuit and the DC-DC voltage booster to power the Raspberry Pi Zero W. A 2500 mAh lithium-ion polymer battery provides a voltage of 3.7-4.2V that is boosted to 5V by the PowerBoost. This battery acts as a backup battery in case the user or perpetrator unplugs the device at any time. The Raspberry Pi Zero W was picked as the central single board computer responsible for all the image processing and distribution of the data collected by the camera. Its integrated Camera Serial Interface (CSI) port allows it to directly interface with the wide-angle camera to capture movement. The Pi's GPIO pins also allow a serial interface with the Sixfab Cellular IoT LTE board, which enables cellular and GPS connectivity. A custom printed single transistor audio circuit board provides audio output to a small speaker in order to function as an alarm to deter intruders. Each component underwent substantial value analyses to determine the advantages and disadvantages of each part selected.

1.3.1 Software Design

Complimentary mobile or web applications can be a hard find when dealing with IoT devices in the vehicle security industry. With SentryEye, we developed a companion web app which enables users to visualize media captured by their security devices, manage device settings, and track device location at any moment in time. Whether our Sentry users need to view historical surveillance or adjust device settings (capture mode, video duration, resolution, etc.) all this can be done remotely through SentryEye's web application.

SentryEye's technology stack was heavily influenced by the quality of the developers' experience, as well as effectiveness as it pertains to the needs of a security IoT device. The server-side logic which enables users to communicate with their devices remotely was written in JavaScript, specifically Node.js. We believed by improving the overall developer UX, we would directly impact the quality of code written to support the SentryEye infrastructure. One way we achieved this was by choosing Node.js. This runtime environment allowed us to rely on Javascript on both the client side and server side. In turn, this infrastructure design decision made our development process a lot more efficient.

SentryEye's backend functionality is broken down into cohesive pieces of software that can function independently of one another. This microservice approach was crucial for SentryEye, since we wanted to ensure the overall system would be robust and scalable. This means that if one piece of software fails, it will not have a negative impact on the other software components since they each function independently by design. SentryEye's web infrastructure also includes MongoDB as a database that is shared among the various SentryEye web services, as well as an integration with Stripe to handle customer payments and billing for SentryEye monthly subscriptions.
While maintaining a technology stack that would run our SentryEye companion application, it was also necessary that we implemented software that could be executed on the actual security device itself. The majority of the on-device software was written with Python. These scripts handle all executables or tasks that must be done by the device itself like performing motion detection, capturing images and video, connecting to Wi-Fi, uploading media, as well as recording GPS coordinates.

1.4 Product Functionality

Several different design aspects have been changed since SentryEye's first inception. The 3D-printed case has gone through several iterations beginning with snap connectors for the case's walls and ending with a final sealed unibody case. An optional solar panel was also conceived in the first phases of the design but was ultimately not implemented as an essential piece of the product but rather an expansion due to the need for more extensive testing. Several internal components were replaced in order to reduce the size of the device with fewer points of failure. In this case, an old TP4056 charge controller board connected to another DC-DC PowerBoost converter was replaced with a single PowerBoost 500c that contained an integrated charging circuit for the sake of convenience and efficiency.

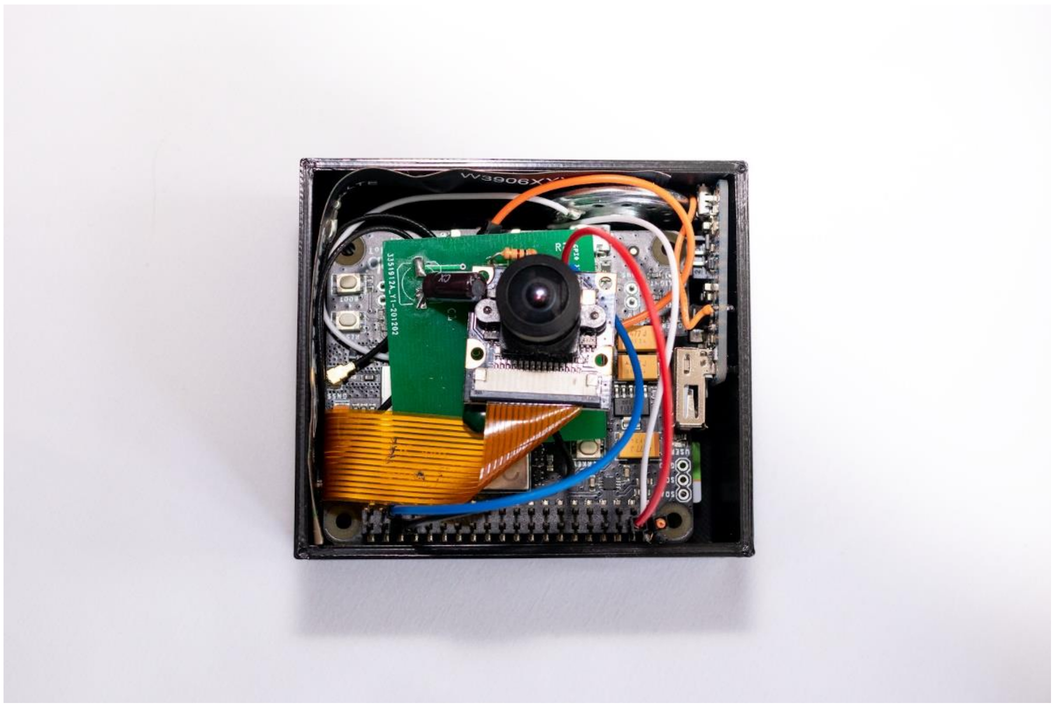


Figure 4: Internal View of SentryEye Device

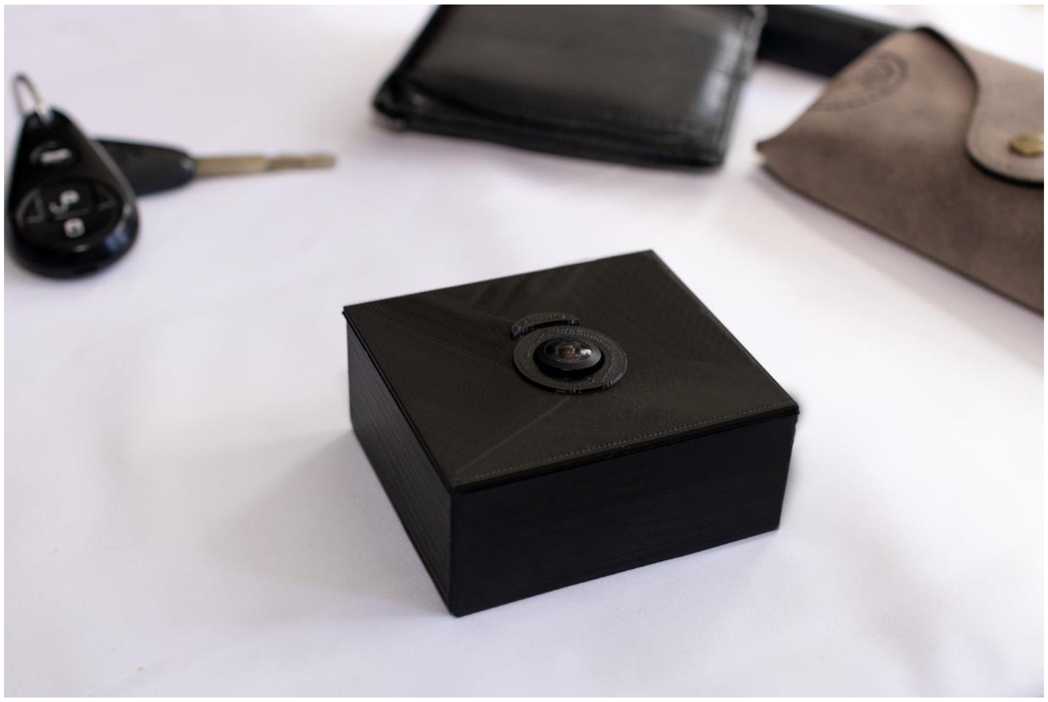


Figure 5: Final View of SentryEye Product

1.5 SentryEye Product Results

To test the capabilities of the SentryEye device, the system was evaluated in several different environments. First, the device was tested within the confines of an apartment to test its Wi-Fi capabilities. The device was tested as if the product had been purchased and opened by a consumer for the first time. The device was paired with the customer's account over the LTE network. The customer then added the Wi-Fi network to the device so that images and video taken by the device would be uploaded over the wireless network. The device was set up inside the apartment and armed for motion capture. Running on battery power alone, the device captured and uploaded 23 pictures to the SentryEye servers over the course of an hour. For the next test, the device was taken outside to test the connectivity under an open sky. The device was set up in a manner such that images were captured via movement detected in front of the lens from different angles as we walked by the mounted camera. This was done over the course of an hour with several intentional delays of movement between each capture. The location data from the GNSS/GPS module in the Sixfab board was analyzed for each capture. The GPS data was determined to be sufficiently accurate to within 10 meters of the device's location each time. These tests indicate that the SentryEye prototype has the complete desired functionality of the product. Additional hardware and software-specific future tests are outlined below.

1.6 Conclusion

To address the issue of on-campus personal safety, our team designed and developed SentryEye to help monitor vehicles and offer car owners a modularized device that can capture pictures of potential threats of vehicular larceny, while also notifying the owner of the threat. Our team was able to create and test a prototype in the limited time and resources that were available. While the current design differs slightly from the original, it maintains most if not all the specified features determined from the customer requirements. In the process of putting this project together, we have learned a great deal on product development and product commercialization, we have encountered challenges and issues, most of which we were able to resolve, and we have identified many opportunities for future improvement and further development for SentryEye. While our prototype may not be quite ready for commercialization, we believe that we have constructed a sufficient prototype that could soon serve as a minimal viable product, with which further product marketing and validation can be conducted.

2 Introduction

To develop a product that ensures individual safety, security, and peace of mind for personal property whether on a university campus or at home, our team developed SentryEye. This device provides an affordable and highly functional layer of security for people's vehicles. As a modularized device for monitoring and capturing pictures, SentryEye acts as both a deterrent and evidence collector. This device has the following functionality:

- Integrated web functionality with a user dashboard accessible from any web browser or mobile device
- Detects the presence of a person in proximity for a sustained period of time
- Takes bursts of pictures of a person too close to the vehicle
- Tracks the GPS location of your vehicle over time
- Emits a bright LED flash periodically as a deterrent
- Small and modularized for versatility of usage

Initial research provided us with statistics regarding key points of concern in the context of safety and security of students and their property. Our market research provided us with the ability to plan out what place SentryEye would occupy within the market. Utilizing these results, we have developed our product as the final aggregate of several iterations and decisions rendered from research. The product of this work has yielded a device with low manufacturing cost while maintaining comparatively high efficiency.

2.1 Problem Statement

Our initial investigation involved the desire to ensure campus safety and security, which provided us with significant amounts of data on prominent crimes that take place on campus. Major crimes in terms of frequency include alcohol-related crimes, theft, assault, forcible sexual attacks or stalking, hazing, drug-related crimes, and bullying and harassment [1]. Many sources appearing in the search presented information on burglary and theft of motor vehicles. According to a study conducted in 2014, 57% of more than 50,000 of on-campus related crimes were property crimes, which include burglary, motor vehicle theft, and robbery [8]. Shown in Figure 6 below is a graph of on-campus crimes per 10,000 full-time-equivalent (FTE) students over the recent decades, from the National Center for Education Statistics [2].

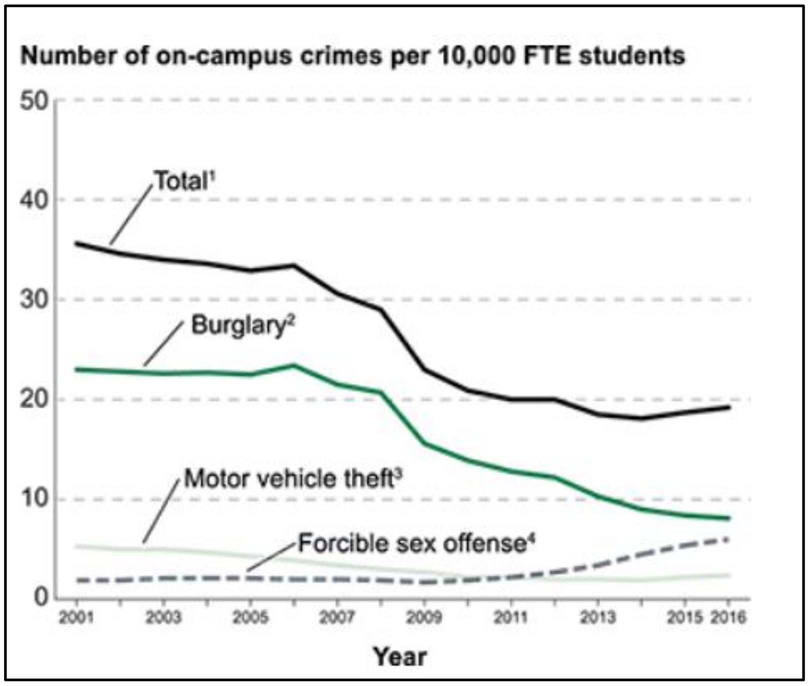


Figure 6: National Center for Education Statistics on on-campus crimes [2]

As indicated in the figure, there was a drop in overall burglary crimes towards the end of the previous decade, but the frequency is leveling off in the recent decade while the number of motor vehicle thefts is trending upwards. By the end of 2016, there was an estimated rate of 8.1 burglaries and 2.4 motor vehicle thefts per 10,000 FTE students.

Similarly, The University Department of Public Safety (DPS) reports that burglaries have increased from 17 to 27 and motor vehicle thefts have increased from five to 17 just between 2017 and 2018 [3]. On a local level, four of 12 WPI safety notifications were reported as breaking and entering and one was of larceny from a motor vehicle [4].

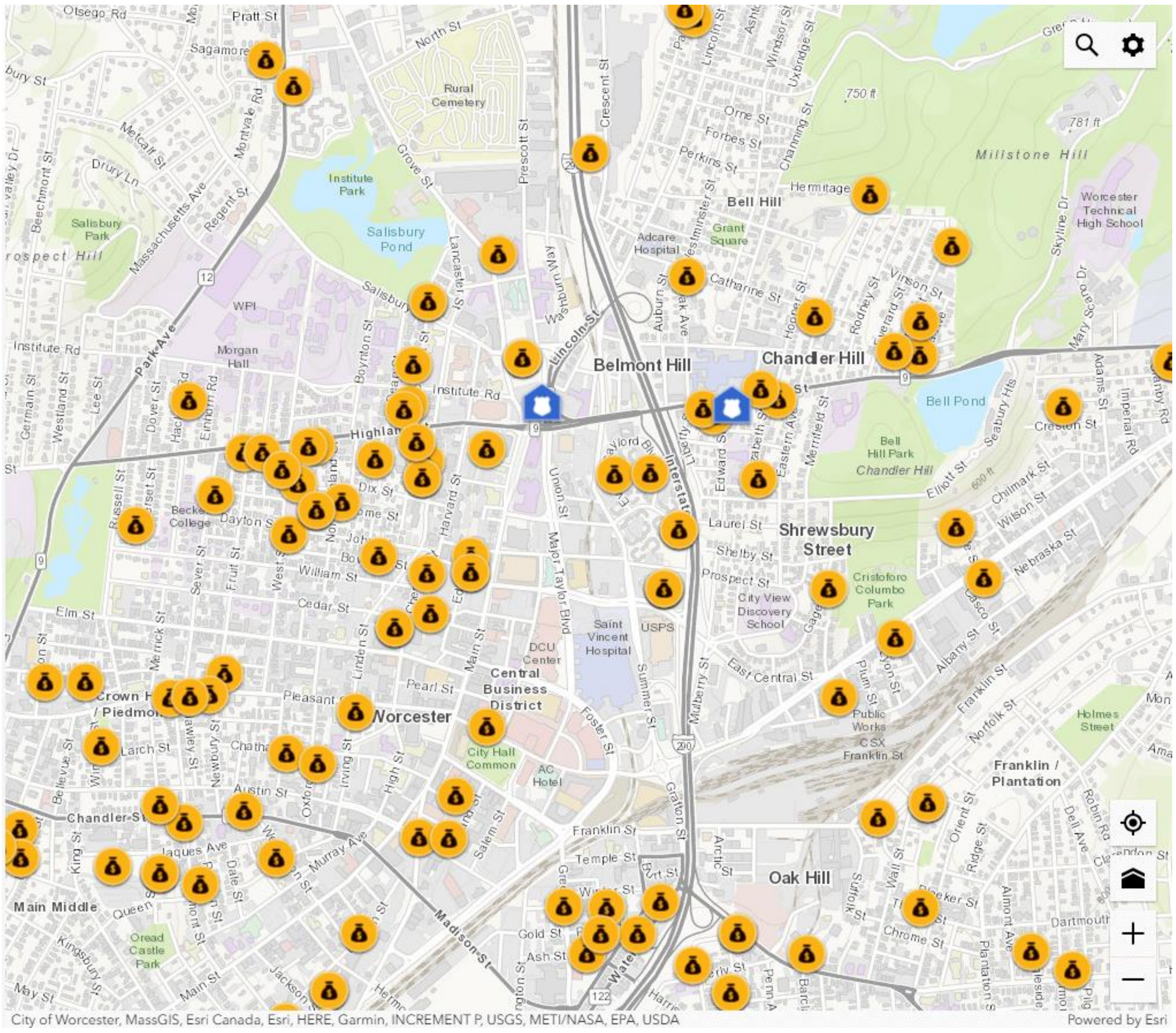


Figure 7: Reported Theft of/from a Motor Vehicle in Worcester Area Surrounding WPI Campus from February 1, 2021 to March 22, 2021 [6]

Over the recent decades, the number of burglaries and car-theft-related crimes has dropped significantly; however, there remains a substantial number of incidents of theft from the vehicles themselves. In Figure 7 shown above, a map of reported incidents of theft of/from a vehicle is displayed. Each marker indicates a theft in the immediate area surrounding WPI campus from February 1, 2021 to March 22, 2021 [6]. This data demonstrates that even at WPI, the concern for the safety and security of personal property remains an issue. These preliminary results prompted the investigation of existing systems that either combat or address such crimes.

We identified that properties such as motor vehicles are an important point of concern. A study found that 46.8 percent of students among the 214 National Universities that reported such data to the US brought cars to their campus for the 2016-2017 academic years [5]. While this data does not specifically pertain to WPI students, the general census provides the conclusion that a large population of students drives a vehicle while attending an institution of higher education.

In our investigation of current systems that protect or prevent the theft of motor vehicles, we have found that most conventional methods focus on prevention or provide only one layer of protection. This layer of protection might simply include an auditory car alarm if the car is jostled. For motor vehicle theft prevention, reputable car insurance sites such as GEICO suggest preventive measures such as to not leave the vehicle running and unattended, to not leave valuables displayed or visible to on-lookers, and to keep the vehicle locked at all times [7].

Our initial investigation and market research have led us to conclude that there is a high frequency of burglary and theft of personal property of students attending college or university, particularly from vehicles. Students with cars on campus are at risk of burglary, especially if their property is not placed in a secure location under constant supervision. Students may only use their vehicles when buying groceries or visiting home a few times a week. The gaps between each car visit present a myriad of opportunities for vehicle theft to go unnoticed. As such, there is a need for a more versatile and robust security system.

2.2 Market Research

Initial market research provided us with a very concrete idea of what technology is already present and which market vacancies are waiting to be filled. OwlCam, an all-in-one approach to vehicle security, was the closest competitor in terms of functionality. The device provides functionality such as unlimited phone notifications, anti-theft deterrence beacons, voice control, and video history within the vehicle, at a rather expensive entry price. OwlCam is priced at $349.00, a very high price considering that a paid $215 yearly subscription is required and not factored in the total above. The other two potential competitors studied were VAVA and Crosstour that for the most part function as common dashcams. SentryEye fills a role geared towards the benefit of students. Given the high cost of education in the modern age, SentryEye was designed to be much more affordable than devices such as OwlCam, with functionality focusing on vehicle security rather than crash incidents, which are the main functions of devices such as VAVA and Crosstour dashcams. Figure 8 shows a cost-versus-features chart, which identifies that there is in fact a gap in the market.

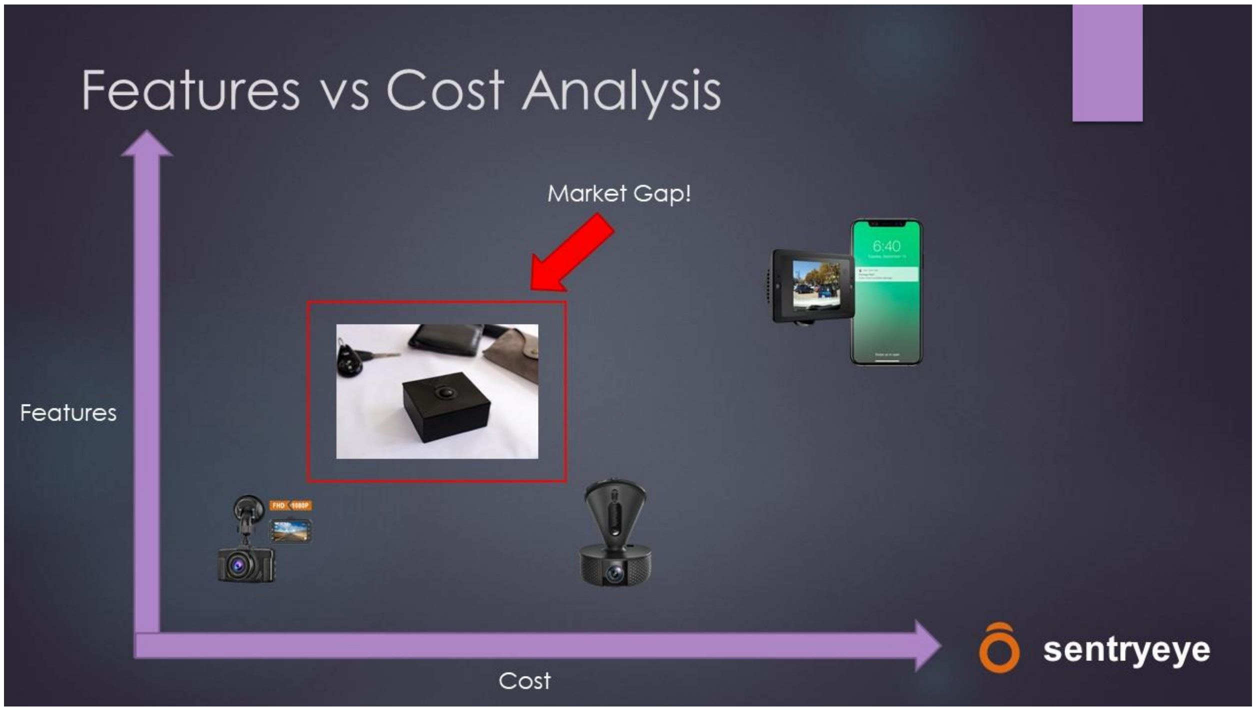


Figure 8: Cost versus Features Chart

3 Product Requirements

Using the information obtained from investigating campus safety and security and researching the relevant market, we identified a set of customer requirements for our product. From the customer's perspective, the product should do the following:

- Take bursts of pictures of suspicious personnel approaching vehicle
- Send pictures to the owner’s phone from anywhere within the cellular network range
- Flash a bright light to deter the suspicious person as “too close”
- Function during the day & at night using vehicle power and backup battery
- Provide an audible message or alarm to deter unwanted suspects from breaking into the vehicle.

Collectively, these explicit requirements should address the lack of a robust and active security system. The implicit customer requirements address the minute details for the product to be marketable as intended. The implicit customer requirements are as follows:

Affordability: The product itself should be cheap for college students. The customer may purchase any number of the product as they see fit, and they should be able to affordably replace any lost or stolen product without too much expense.

Small Size: The device should be compact, so it is easily managed and moved and does not obstruct any of the vehicle's functions.

Modular: The device should be self-contained, function independently, and be able to be mounted in a variety of different locations for effective placement. Multiple devices on one vehicle should not interfere with the functionality of each other. An integrated software dashboard must allow the user to sync data across all devices.

Wide Camera Angle: For effective picture captures, the camera used in the device should have a wide angle to capture a broad image that provides the user with visibility of hard-to-reach areas.

Reliable Battery/Charging: For the sake of simplicity and user convenience, the battery must be able to be charged via microUSB within the car, so the owner does not have to worry about constantly charging the device manually. An optional solar panel module can also be used to avoid using a vehicle battery.

Low Power Consumption/Back-Up Battery: The product must operate such that if it is disconnected from its vehicle power source, it will continue to function without pause. The device will warn the user of any low battery detection.

Day/Night Operation: The product must perform its functions during the day and night, as it should provide the owner with security in both environments when the property is unattended. The device will draw a low enough current such that it can be left plugged into the vehicle without exhausting its battery.

Integrated User Dashboard: The product must be easily managed and set up via an online user dashboard that allows the user to change the device settings remotely. The user must be able to add and manage multiple SentryEye devices with ease. Users must be able to manage their cellular subscription service and change their payment plan anytime.

Cellular Network Connectivity: The product must be able to send photos captured remotely over an LTE network regardless of whether a WiFi connection is available.

3.1 Product Specifications

The product base (which adheres to the user's desired surface) is a mere 77 x 68 mm, while the height extending outward from the attached surface will reach approximately 32 mm. The relative size of the module considers the customer need for a compact, modular device encasement that can be used to adhere to whatever vehicle surface desired. The flat base of the device encasement allows it to stand upright or on its side for enhanced flexibility in terms of where the device needs to be placed. The 3M adhesive allows placement anywhere within the car in several positions for long-term operation. An optional solar cell can be mounted anywhere within a car with dimensions of approximately 130 x 150 mm. In terms of functionality, we constructed a functional block diagram, shown in Figure 9 below, of our product.

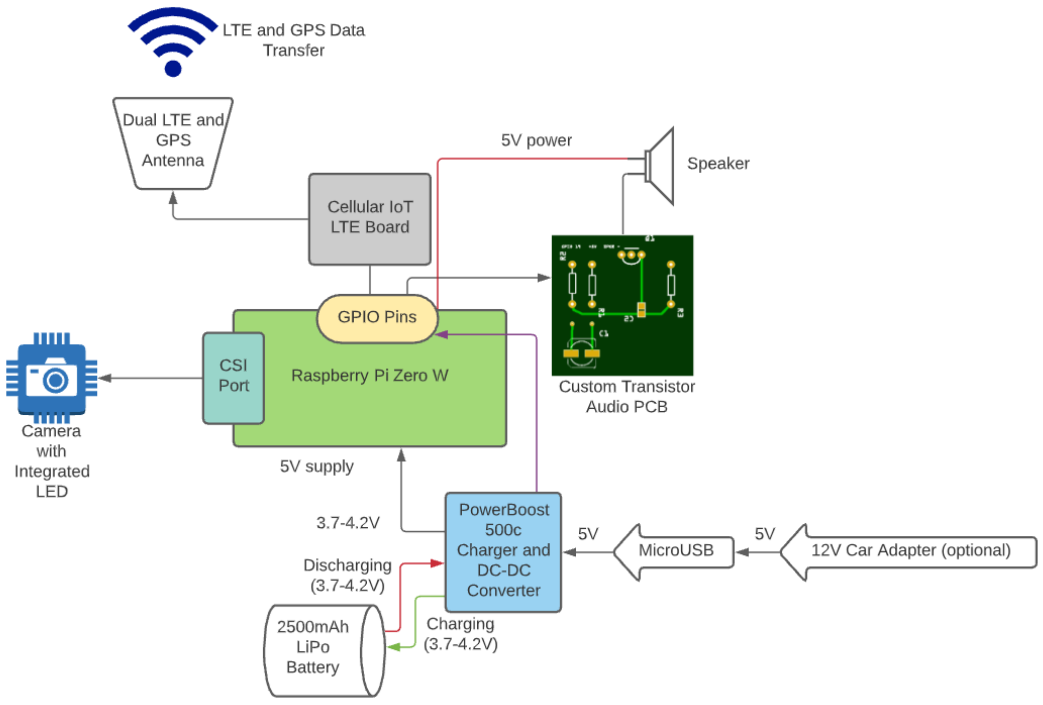


Figure 9: Functional Block Diagram of SentryEye

The Raspberry Pi Zero W will act as the single-board computer and microcontroller used to operate the motion-detecting portion of a wide-angle camera lens. This device is cellular LTE and Wi-Fi enabled, which allows the transfer of photos and videos to the SentryEye web server anywhere there is cellular service. This feature enables the user to remotely access their SentryEye device captures from any device. An optional 5V solar panel will be connected to a charge controller that will be needed to regulate the voltage that varies in conjunction with the operational output of the panel. The PowerBoost 500c also features an integrated charging circuit for the battery, with additional overcurrent protection of up to 3 Amps. The battery in question will encompass a 2500 mAh lithium ion-polymer battery type. This battery has a 3.7V output, which requires a DC-DC voltage booster module (PowerBoost 500c) to meet the 5V input via microUSB as required by the Pi Zero. A ribbon cable will utilize the Pi Zero's CSI port to provide power and functionality to a wide-camera lens that will detect movement and respond appropriately with bursts of photos.

The mechanical, electrical, and ergonomic components in the design are as follows:

- **Mechanical:** A small, compact device with a 3D-printed case that is unobtrusive to the operation of the vehicle will be utilized in an aesthetic manner by being designed to blend in with a car's dashboard. A 3M adhesive back will allow the device to stick and unstick to surfaces for flexible placement options.
- **Electrical:** The lithium ion-polymer battery trickle-charged using a charge controller with varying voltage according to the solar panel output will be used in conjunction with the wide-angle camera allowing for full functionality as well as longevity of the device. A custom printed circuit board will provide single transistor audio for the device speaker.
- **Ergonomic:** As shown in Figure 10, the device's footprint is small relative to a double-A battery and is thus quite compact for its variety of functionality.

Shown below in Figure 10, 11, 12, and 13 are the different view perspectives of the product, as indicated.

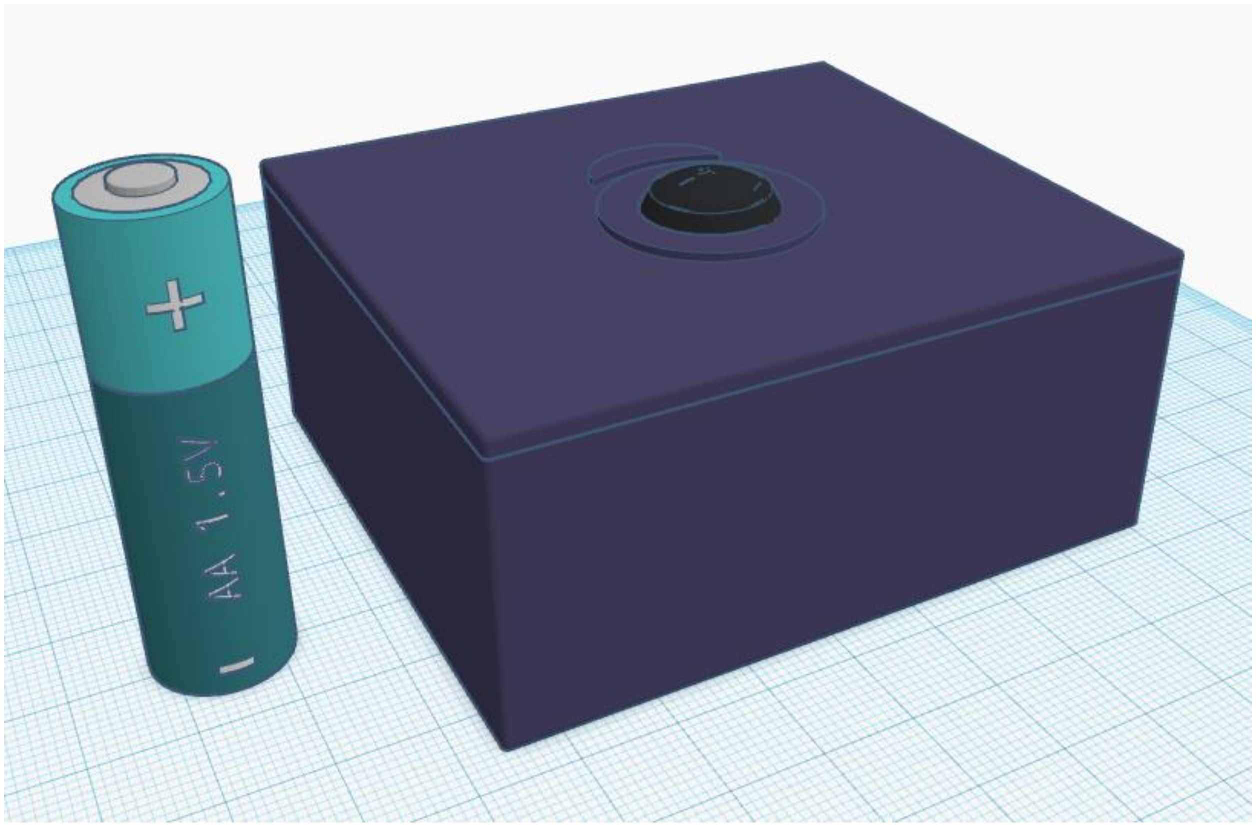


Figure 10: Side of the device when closed; AA battery for scale

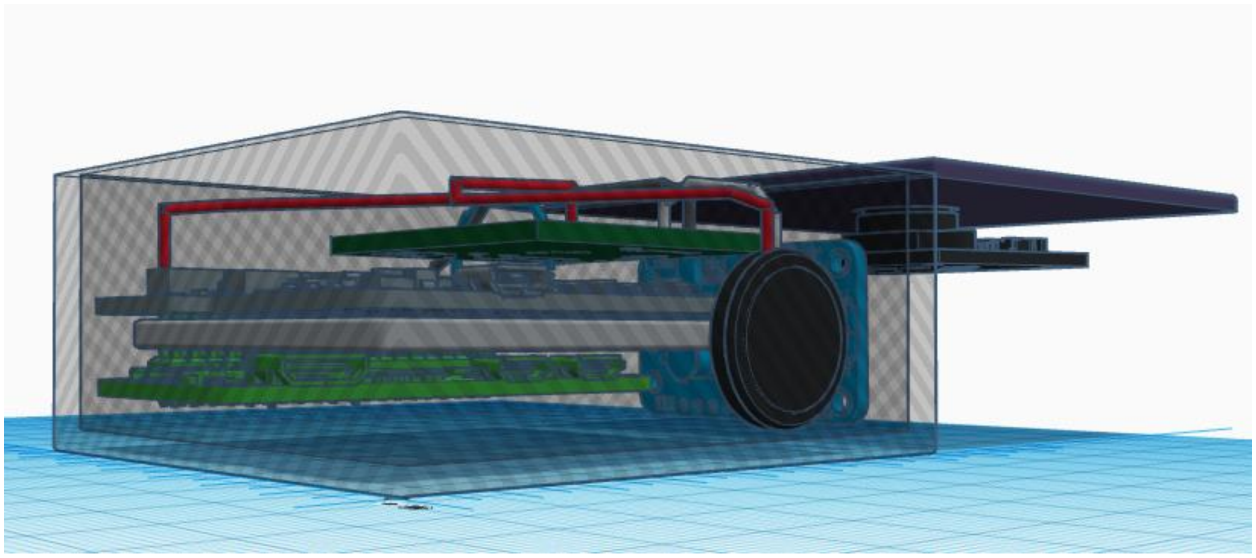


Figure 11: Internal Device View with Raspberry Pi Zero W in light green, lithium-ion battery in white, black cylindrical speaker, and PowerBoost 500c voltage booster component in light blue.

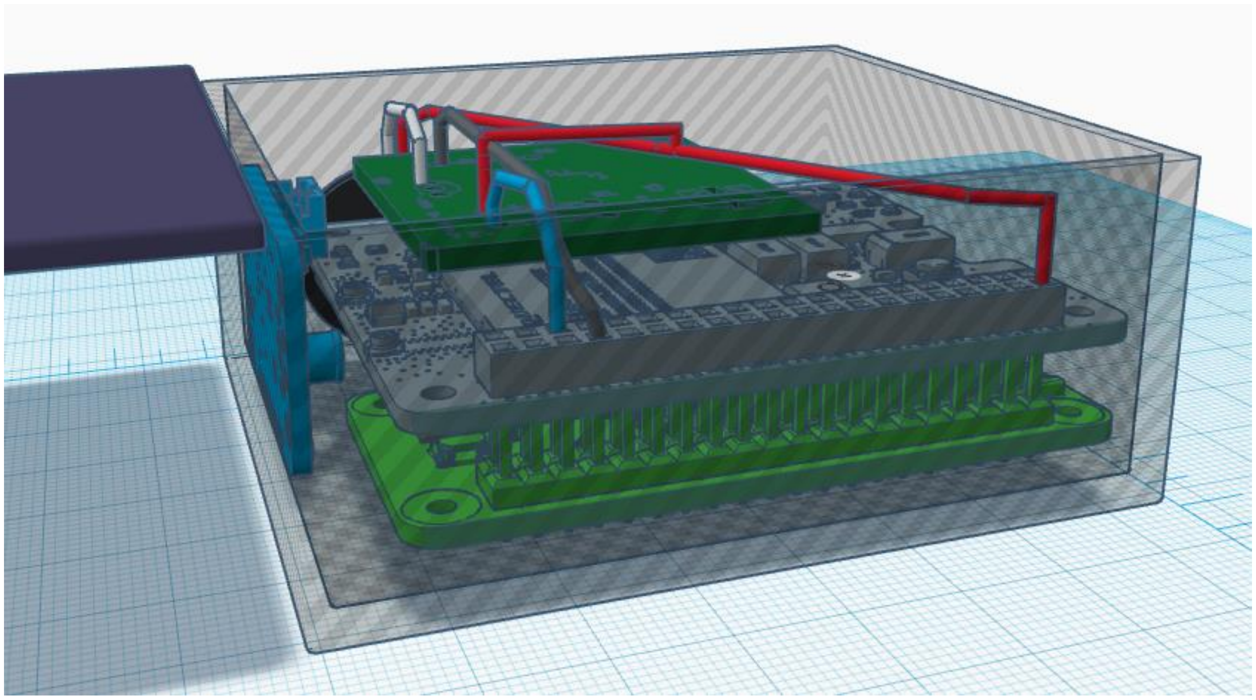


Figure 12: Other side of the device with light gray Sixfab LTE module in frame with Raspberry Pi Zero W beneath.

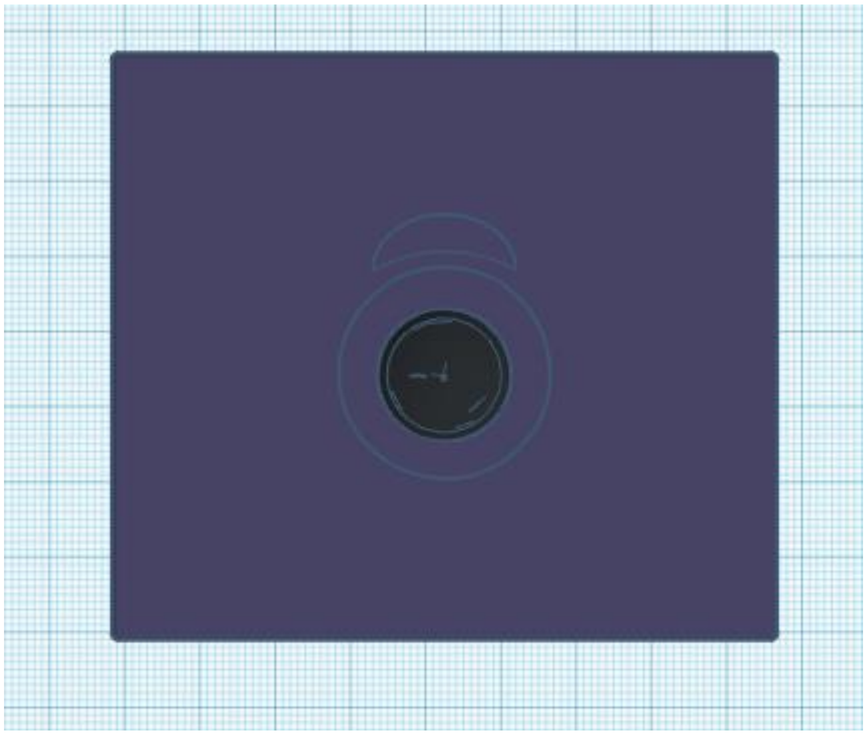


Figure 13: Top-down view with camera module attached.

3.2 Competitive Value Analysis

To better understand the competition that exists in the market for our product, we conducted a competitive value analysis. The purpose of this analysis is twofold: to identify the competition and to validate the product using the customer requirements we have outlined. To evaluate our product along with each of our competitors' products, we defined the following requirements from the customers' perspective regarding quality, convenience, and cost:

- **Picture quality:** The quality of the image provided will no doubt be on the forefront of customers' minds. A relatively clear image quality that provides adequate details of a suspect's face will be necessary for SentryEye to be an attractive budget vehicle security option for consumers.
- **Hands-off maintenance:** The device must remain active and autonomous for most of its lifespan. The customer will likely demand that the device does not need maintenance, as it is a security-enhancing add-on rather than part of the standard maintenance of a vehicle. If the user must constantly change batteries in the device, the extra maintenance for an aspect of their vehicle that is essentially a last line of defense is extremely inconvenient. For this reason, the SentryEye must be capable of hands-off maintenance for long periods of time.

- **Affordability:** Concerning the target audience of college students with cars on campus, it is essential that the device is affordable yet effective. The cost must not exceed approximately $50 when mass-produced, though prototypes with parts bought individually will likely cost more.
- **Communication with External Device:** As a security device, the SentryEye must be capable of communication via WiFi and 4G cellular to alert the customer of a potential break-in and send photos of the suspect immediately, as most consumers expect this IoT technology to be relatively standard on-board most devices.

Each of the requirements are assigned a relative weight of importance from the customer's perspective. Shown in Table 1 below is the requirements and their assigned weights on a scale of 1 to 10, were 10 demonstrates most importance (heaviest) and 1 demonstrates the least importance (lightest).

Table 1: Weighted Customer Requirements

Requirement	Customer Weight
Picture Quality	6
Hands-Off Maintenance	8
Affordability	10
Communication with External Device	7

The picture quality is important, but not essential to the desires of the customer in terms of the device's function as a security device. The target audience would likely regard affordability on a much higher scale than the image quality of a device designed to take pictures clear enough to recognize the person in its wide field of view. For the college student audience in mind, affordability is most definitely at the top of the list in terms of customer criteria. For this reason, affordability is rated at a customer weight of 10, whereas the picture quality is 6. Hands-off maintenance is likely quite high on the list of customer demands, as increased maintenance (such as replacing the battery) also negatively impacts affordability due to the price of repairs. Consumers with the desire to ease their worries about vehicular break-ins would be more likely to buy a device that will be capable of autonomously alerting them regardless of the situation or status of the device. Therefore, hands-off maintenance and communication capability with an external device are rated at customer weights of 8 and 7 respectively.

The OwlCam is an all-in-one car security system that performs as a two-way dashcam as well as a flashing theft deterrent that triggers an alarm when the sound of broken glass or an intruder (when the car is off) is detected. The system is $350 and requires a cellular subscription of $215 annually for the 4G LTE feature of sending videos of accidents and break-ins to the user. This device is a potential competitor due to the security features of detecting movement and sending videos of intruders. However, our device aims to be much less expensive and modular, while focusing on preventing thefts alone rather than as a dashcam recording accidents.

The VAVA dashcam is another potential competitor with a parking mode that will record when the parked car is shaken by an intruder or from a collision caused by another car. In this way, the dashcam records potential security threats and is thus competition for the SentryEye. However, the dashcam is on a swivel mount, and will only record in one direction due to the inability to stick to surfaces (the front of the car or the interior of the car). The device costs $120, which is still significantly more than the desired price range of the SentryEye (approximately $50-$60).

The last competitor is the Crosstour dashcam, which provides a G-force sensor that will record only in the case of an accident or the jostling of the vehicle. The camera features a wide-angle lens (170 degrees) as well as motion detection for security purposes, which makes it a potential competitor for the SentryEye, especially at a price point of only $30. However, the device does not function when the car is off and cannot detect intruders from inside the car or send the photos remotely to the user's phone. Shown in Table 2 is the result of our competitor value analysis based on the weighted customer requirements.

Table 2: Competitor Value Analysis

	Criteria					
Product	Picture Quality	Hands-Off Maintenance	Affordability	Communication with External Device		
Weight	6	8	10	7	Raw Score	Weighted Score
OwlCam	10	8	2	9	29	207
VAVA Dash Cam	8	7	4	7	26	193

Crosstour Dash Cam	5	4	10	1	20	169
Team #13 - SentryEye	5	8	9	7	29	233

As shown above, the raw score for the SentryEye tied the OwlCam concerning the four criteria considered here. However, when the weighted score calculated via value analysis was applied, the SentryEye was the clear winner when factoring in picture quality, hands-off maintenance, affordability, and communication with an external device.

Without sufficient testing, the camera's picture quality is solely based on the specification sheets provided online for each. For example, both the Crosstour dash cam and the SentryEye utilize 5 MP cameras, but the sensitivity of the aperture, as well as the performance in low light that might factor into the customer's interpretation of camera quality for a device of this nature cannot be accurately determined. Additionally, the SentryEye meets a particular budget security market rather than a dashcam service, and thus the criteria above were chosen with this fact in mind.

4 Design Approach

Our product's goal is to provide users with an affordable layer of security. SentryEye should capture and send images to the user as specified and should operate independently and in an unobtrusive manner. There should be little to no maintenance needed from the user. As such, SentryEye should be a small, compact device that is unobtrusive to the operation of the vehicle and can be fastened to the surface on which it is mounted. In the context of power and operation, the device should operate using low power and the device should be able to self-charge for convenience. The shape of the SentryEye, in addition to being small, should be aesthetically convenient and blend into the mountable locations, such as the car's dashboard.

In accordance with the requirements and operational context specified above, our team has approached designing in two aspects; one set of designs focuses on the functional system while the other focuses on the type of visual media to be reported back to the user.

First, our team identified two system-level design concepts for SentryEye: a modular device (piece-by-piece approach) and a security hub system (all-in-one approach). In the modular device design, a single device would count as one unit, useable for one field of view. This device would be self-contained and would provide image processing for a single wide-angle direction in the manner specified above. A security system design would utilize multiple devices potentially connecting to a centralized hub that communicates with this user. This system together would be one unit hardwired to monitor multiple fields of view that would require a longer and potentially more permanent installation process.

The modular device design allows for the user to choose the number of devices and therefore the level of security and the amount they want to spend for the security. However, if in the future the user decides to acquire more devices, there would be the slight inconvenience of setting up and mounting the additional devices. The security system design has a one-time initial set-up and provides a full range of security for the user. This design would cost the user more money up front, while limiting their desired level of security.

Regarding the type of visual media sent to the user, the potential outlets are pictures, video, and live video feed. Picture capturing and flash deterrent can be integrated together in which one does not obstruct the other, for the sake of operation during the day and at night. Pictures also generally occupy less storage space than videos, which is advantageous given the limited microSD storage space. However, the device pushes these pictures to a remote web server, which means they can be removed from the device itself without taking up much space. Videos are more useful compared to pictures when identifying captured subjects, but specific design considerations for integration would be more involved, as the camera would potentially need different modes for operation in daylight and in the evening. Similarly, a live video feed would allow the user to view live footage of suspicious activity, but battery usage and limited-light environments would prove to be a convoluted factor to address.

The best approach to suit the device's needs and desired functionality is to ensure a "hands-off" approach with little maintenance required by the user. This means the device must be powered continuously without interruption using a high-capacity battery trickle-charged via a solar panel input or vehicle battery with a sufficiently capable charge controller. As such, capturing pictures suits the requirements of low-powered operation. The camera must provide clear photos of a suspect's face, even when it is dark. The device must also alert the user on the phone and send the photos via LTE or Wi-Fi if it is available. Emphasis on low costs for the consumer as well as an unobtrusive, compact design means that the device must be portable enough to stick anywhere on the vehicle desired. A modularized approach would address the issue of affordability and compact design, providing the user with flexibility in level of security and cost. Additionally, the device must feature enough storage space for potentially thousands of photos stored in a local manner if a cellular network were unavailable for any period.

4.1 Hardware Components

A top-level block diagram of SentryEye can be seen in Appendix B. Each module is explained in depth as shown below:

4.1.1 Optional Solar Panel (5 V)

Since our device is intended to be placed within/on vehicles, it will at times be exposed to light. As a result, our group decided that power through solar charging would be the most efficient way of charging the device. A 130 by 150 mm solar panel with 3M adhesive will be able to connect to our product's charge controller. With a 17% energy efficiency rating, the solar panel will be an additional source of power when SentryEye is mounted on the vehicle dashboard or window. This module can serve as a viable add-on in regions with sufficient sunlight for most of the day. However, this module is optional as in areas with relatively little UV light exposure (in addition to the UV-blocking coat on the car windshield) it will prove less useful.

4.1.2 Lithium-Polymer Ion Battery

Our battery will be connected to the previously mentioned charge controller. Given a product constraint of high operational duration, our battery (selected for its capacity values) has a capacity of 2500 mAh. At worst operating conditions (in which the draw is about 160-180 mA), this battery will last approximately nine to eleven hours; a modestly acceptable amount when no other power source is available via the solar panel or vehicle battery.

4.1.3 Voltage Booster and Charge Circuit (PowerBoost 500c)

The Voltage Booster will be a module connected between the LiPo Battery and the Raspberry Pi Zero W. It will convert our lower input voltage of 3.7 V into 5.2 V in order to properly power the Pi, which requires an input voltage within the approximate range of 5 V in order to function. This module also replaces the previously used TP4056 charge controller board. This is due to its inclusion of its own charge controller circuit that can charge the LiPo battery while accepting the 5V input.

4.1.4 Raspberry Pi Zero W

In order to ensure there is WiFi functionality, the Raspberry Pi Zero W single board computer was selected. Connected between the PowerBoost, and Peripherals (LED/Camera/User), the Raspberry Pi Zero W was selected for its small size, low cost, and Camera Serial Interface (CSI) ports. It will act as our device's "hub" and will control the functionality and synchronization of our device.

4.1.5 Sixfab Cellular IoT LTE Module

In order to ensure that the device will be capable of sending bursts of photos of the burglar to the user from anywhere, this module includes GPS (GNSS) and LTE/NB-IoT compatibility using its BG96 integrated chip. The GNSS module allows the device to send location data that accompanies each photo it uploads to the user's online dashboard. These are displayed on map so that the user can determine where their device is located at any time. The BG96 chip allows data transfer over the LTE network via the Ting service provider. This module will utilize several GPIO pins on the Raspberry Pi Zero W for a direct interface with the microcontroller. This module allows cellular connectivity for transmitting alerts, images, or videos if desired.

4.1.6 LED Deterrent

The LED will be powered through the CSI camera connected to the microcontroller and was selected as part of an attempt to deter any attempted theft or prolonged interest around our device's protected vehicle. As the device originally only notified the user, the LED was chosen to act as an extra deterrent layer of defense.

4.1.7 CSI Wide-Angle Camera

Our Wide-Angle Camera contains a lens chosen for its wide field of view and picture quality. Powered by the Raspberry Pi's CSI port, it will transmit bursts of images through the device's integrated WiFi functionality. This will be our device's eye through which our integrated software will detect motion. The camera has a wide aperture to effectively capture photos even in low-light environments.

4.1.8 Custom Single Transistor Audio Printed Circuit Board / Speaker

Using the Fritzing software, a single transistor custom PCB was designed such that the Raspberry Pi Zero W could output adequate audio as an alert or verbal warning to intruders. The resulting Gerber design file was compiled and sent to JLCPCB, a printed circuit board service based in Hong Kong. Five prototypes were made and received a few weeks later. The small PCB is connected via the Raspberry Pi Zero W's GPIO pins. Only two pins were used, and they were modified such that they produced a pulse-width modulated signal so as to emulate native digital audio. This was used because the Raspberry Pi Zero W is incapable of native audio output via an analog audio jack. The speaker is a simple 8-ohm coin speaker directly wired to the PCB.

4.1.9 User (WiFi/LTE)

This module refers to our device's WiFi and LTE functionality. As the Wide-Angle Camera takes the images, the microcontroller will transmit these to the user through WiFi/LTE into the SentryEye web server.

4.2 Design Modules

To satisfy the product requirements discussed, we identified potential components to be utilized for each module in the design. In a successive process, we conducted value analyses for each set of potential components for assembling SentryEye. First, we identified microcontrollers that can be utilized with our design. Then, we looked at different kinds of cameras to address the monitoring needs of the product. Finally, we looked at different batteries that would be able to power the system and trickle charge over time. Utilizing value analyses, we identified the advantages and disadvantages of each component.

4.2.1 Microcontroller System

The microcontroller will serve as the brain of the entire system. It will control the behavior of the system and act as the point of communication through transmitting the visual media. There are a variety of different kinds of microcontrollers, but our product needs a reliable device with a strong connectivity factor so that the system can communicate and transmit relevant data. In our investigation, we identified three candidates, all of which come from the Raspberry Pi family. This is due to the fact these series of devices are reliable in performance and have strong documentation on uses and applications. In addition to the ubiquity of these devices, our team is also familiar with using these microcontrollers.

The Raspberry Pi Zero Standard is by far the cheapest of the three devices and has an operating voltage of 5 Volts. It is also the smallest microcontroller, relative to the other three devices. It provides a handful of different port connections, which include Mini HDMI, microUSB, CSI, and HAT 40 pins. These pins allow connections to external peripherals ("HATs" or external displays and cameras). This device does not support any communication over Bluetooth or WiFi.

With the median price, the Raspberry Pi Zero W operates using the same voltage and provides the same port connections as the Raspberry Pi Zero Standard. It is only slightly larger than the Standard, with an increase in length of 1.0 mm. The W provides three different options for connectivity which are WiFi, Bluetooth, and Bluetooth Low Energy (BLE).

Our final option is the Raspberry Pi 3, which is the most expensive of the microcontrollers in our list of candidates. Operating with the same voltage as the above devices, this microcontroller is significantly larger than the other two; the size is comparable with that of a credit card. The Pi 3 has a large variety of port connections, which include GPIO, USB, Full HDMI, Pole stereo, CSI, DSI, Micro SD, and Ethernet. It also has WiFi and BLE capabilities as well. Shown in Table 1 is a breakdown of all the specifications for each device mentioned above.

Table 3: Microcontroller Specifications Used in Value Analysis

Microcontroller Model	Cost	Operating Voltage	Form factor (Dimensions)	Connectivity (Bluetooth, WiFi)	Port Variety
Raspberry Pi Zero Standard	$5.00	5 V	65.0mm x 31.0mm x 5.0mm / 2.6" x 1.2" x 0.2"	None	Mini HDMI, micro USB, CSI, HAT 40 pin
Raspberry Pi Zero W	$10.00	5 V	66.0mm x 30.5mm x 5.0mm / 2.6" x 1.2" x 0.2"	WiFi, Bluetooth, BLE	Mini HDMI, micro USB, CSI, HAT 40 pin
Raspberry Pi 3	$35.00	5 V	86.9mm x 58.5mm x 19.1mm / 3.4" x 2.3" x 0.8"	WiFi, BLE	GPIO, USB, Full HDMI, Pole stereo, CSI, DSI, Micro SD, Ethernet

Using the values for each microcontroller in Table 3, we compared and advantages and disadvantages of each device, and determined the ideal device for the SentryEye system. We utilized a value analysis chart to better quantify the ratings provided for each microcontroller. Shown below in Table 4 is the result of our value analysis.

Table 4: Microcontroller Value Analysis

	Microcontroller Design Choice Criteria						
Microcontroller Model	Cost	Operating Voltage	Form factor (Dimensions)	Connectivity (Bluetooth, WiFi)	Port Variety		
Weight	8	8	10	9	5	Raw Score	Weighted Score
Raspberry Pi Zero Standard	7	7	8	3	3	28	234
Raspberry Pi Zero W	6	7	8	9	3	33	280
Raspberry Pi 3	5	7	5	8	7	32	253

The scoring for the criteria concerning each microcontroller was determined based on the objective performance or impact if included in the system. The range of the score is on a 1 to 10 scale, with a score of 1 indicating little to no impact or use, and a score of 10 indicating a significant impact or use case. The weight factor for the value analysis was determined based on how pertinent the criterion is to the system and the outlined requirements. The scale of weighing ranges from 1 to 10, with 1 serving as the least pertinent and 10 serving as the most pertinent.

The cost factor for the microcontroller has a heavier weight due to the constraints of the given budget, as well as for production; when in mass production, the values will scale downwards, but we are using a nominal value that serves as a point of reference. The microcontrollers have a large range, in terms of cost, which is why it has a higher weight. The operating voltage also has a higher weight, as this influences the design of the internal components of the system. While there is flexibility in the general case, all our candidates for microcontroller operate using the same voltage. The form factor has a weight of 10 because it directly influences the shape of the device, as well as how components fit together in the device. Our product is designed to be small and will thus require small components in order to satisfy the requirement. Connectivity has a high rating because it addresses the communication requirements of the system, which is very fundamental to the design. Finally, port variety has a median weight since it is not as pertinent and mostly provides convenience in terms of connection and design.

4.2.2 Camera

The camera provides the means with which SentryEye will capture pictures; it is the eye of the device. Having determined the family of microprocessors from which our team will select a device to use, we identified potential cameras made for Raspberry Pis that can be used in our system. Our choice to choose strictly cameras made for Raspberry Pi devices was to alleviate the team of any overhead or issues caused by other cameras since we already know that these cameras were designed to be compatible with the Raspberry Pi devices and connect using the CSI cable-port system.

The Standard CSI Camera Module is a standard camera that provides 5 MP resolution. It does not have an adjustable focus nor does it have any night time or enchanted performance in low-light settings. The camera has a 53.50 degrees horizontal and 41.41 degrees vertical field of view.

Similar to the Standard CSI Camera Module, the CSI Camera Module without IR filter provides a resolution of 5 MP and has no adjustable focus. It does not have a dedicated night vision feature but does allow for moderately improved low light performance due to the lack of IR filtration. This camera module also has a 53.50 degrees horizontal and 41.41 degrees vertical field of view.

The Wide-Angle Camera has a resolution of 5 MP and does not have an adjustable focus feature, but it is a wide-angle camera. This camera provides a field of view of 120 degrees and does not have infrared night vision for low light performance.

The Variable Angle Camera has a resolution of 5 MP and does provide variable or adjustable focus. Considering the variability of this camera, it has a total field of view of 160 degrees. While there is no dedicated night vision feature for this camera, it can adjust the angle for different types of compensations for some level of improvement. The effective field of range, however, is going to be less than 160 degrees at any given time. Shown in Table 5 are the camera specifications for each camera mentioned above.

Table 5: Camera Specifications Used in Value Analysis

Camera Model	Cost	Low light performance	Picture quality	Field of view

Standard CSI Camera Module	$17.76	Infrared Night Vision: No	Resolution: 5MP Adjustable-focus: No	53.50 degrees horizontal, 41.41 degrees vertical field of view
CSI Camera Module without IR filter	$17.76	No IR filter	Resolution: 5MP Adjustable-focus: No	53.50 degrees horizontal, 41.41 degrees vertical field of view
CSI Wide-Angle Camera	$20.53	Infrared Night Vision: No	Resolution: 5MP Adjustable-focus: No, Wide Angle	120 degrees field of view
CSI Variable Angle Camera (170°)	$20.53	Infrared Night Vision: No	Resolution: 5MP Adjustable-focus: Yes	170 degrees field of view

Using the specifications from Table 5, we conducted a value analysis to assess the camera modules and to help us determine which candidate is the most viable camera for SentryEye. The criteria for the analysis was based on the available specification information and their relevance to the design and the features of SentryEye. The results of the analysis are shown below in Table 6.

Table 6: Camera Value Analysis

	Camera Design Choice Criteria					
Camera Model	Cost	Low light performanc e	Picture quality	Field of view		
Weight	6	8	6	9	Raw Score	Weighted Score
Standard CSI Camera Module	5	5	5	5	25	145

CSI Camera Module without IR filter	5	8	3	5	21	157
Wide-Angle Camera	6	5	6	7	24	175
Variable Angle Camera (160 °)	6	6	6	8	32	192

The general method of scoring for each module based on each criterion was done in the same manner as the scoring for the microprocessors in the above section. The scale of both scoring and the weights are the exact same. Comparatively speaking, the weight for the cost is not as significant, since the price range for the cameras is small, though the cameras themselves are relatively expensive. The light performance has a higher weight since it is directly connected to performance throughout the day and nighttime. The picture quality weight was lower relative to the other criteria as well, due to the cameras having essentially the same effect across the board in terms of quality. Field of view was rated as the highest and was therefore given the highest value. A limiting factor of SentryEye's performance in terms of security is the field of view; the range specifies the upper bounds for which the system can monitor and capture effective pictures.

4.2.3 Battery

The battery is an essential component of our system; the battery must supply enough power to the system, last for a reasonable duration of time, be rechargeable, and have a lasting lifespan in all conditions if not plugged into a car's USB port or 12V port. Our team has decided to look at four different types of batteries for consideration. These types of batteries are ubiquitous and can typically be found in many devices and applications outside of SentryEye.

The first battery on the list is Nickel Metal Hydride (NiMH), which is a moderately rechargeable battery. It has a relatively shorter lifespan in terms of charge cycles, which range between 300 to 500 cycles. This battery has a nominal voltage value of 1.25 or ~1.2 volts, which is low for powering a 5 V Raspberry Pi device. The battery, however, is operable in temperatures as low as -20 °C, which would address environmental performance concerns for usage in regions that get extremely cold. The NiMH battery typically comes as cylindrical or prismatic cells, but not as widely available in the latter form. Prismatic cells are flat and easily fit between device components, while cylindrical batteries tend to require more room and structural support within the device's case.

The Nickel Cadmium (NiCd) is very similar to the NiMH battery; with a nominal voltage of 1.25 V, this battery also operates at low temperatures and exceeds that of the NiMH battery, with a lower threshold of -40 °C. This battery is typically in the cylindrical form. There is a tradeoff for low rechargeability for a higher lifespan of 1500 charge cycles.

The Lithium-Ion battery has high chargeability and a relatively high life span ranging from 500 - 1000 charge cycles. The nominal voltage is much higher at 3.6 V and operates between the temperatures of -20 to 60 °C, much like the NiMH battery. This battery comes as both cylindrical and prismatic cells, with the prismatic cells being more available than that of the NiMH battery.

Lastly is the Lithium Polymer (LiPo) battery. This battery is much like the lithium ion battery, with a nominal voltage of 3.6 V, but is slightly less chargeable. Additionally, it has a lifespan of 300-500 charge cycles like the NiMH battery, but does work well below temperatures of 0 °C, which is relatively high for a lower limit. The LiPo battery, however, is very much available in the prismatic cell form and has a lot of different sizes, which is relevant to the overall size concern for SentryEye. There are also cylindrical options, but those are not as available in comparison. Shown below in Table 7 are the batteries and their specifications as mentioned above.

Table 7: Battery Specifications Used in Value Analysis

Battery Model	Cost	Rechargeability	Lifespan (charge cycles)	Operating Nominal Voltage (V)	Operating Nominal Temperature Range	Form factor
NiMH	Medium	Medium	300-500	1.25	-20 to 60 °C	Cylindrical, Prismatic (limited)
NiCd	Low	Low	1500	1.25	-40 to 60 °C	Cylindrical
Lithium Ion	Hi	Hi	500-1000	3.6	-20 to 60 °C	Cylindrical, Prismatic

LiPo	HI	Medium	300-500	3.6	0 to 60 °C	Prismatic, Cylindrical (limited)

Using the battery specifications in Table 7, we conducted a value analysis to evaluate the different batteries. Shown below in Table 8 is the result from the analysis. Similar to the previous sections, the criteria were selected and weighted as relevant and applicable to the design and functionality of SentryEye.

Table 8: Battery Value Analysis

	Battery Design Choice Criteria							
Battery Model	Cost	Rechargeability	Lifespan (charge cycles)	Operating Nominal Voltage	Operating Nominal Temperature Range	Form factor		
Weight	6	8	7	9	7	10	Raw Score	Weighted Score
NiMH	6	7	5	3	8	5	34	260
NiCd	7	6	7	3	9	5	37	243
Lithium Ion	4	8	6	7	8	8	41	329
LiPo Ion	5	7	5	7	5	9	38	309

The most important criterion for the battery value analysis is the form factor, which encompasses the general shape and size of the battery itself. The battery in theory is one of the bulkier components of the device. As such, there is a lot of weight (in terms of score) on the shape and size such that SentryEye meets the specified design requirements. Cost is rated at a median weight value, due to the similar distribution in cost of each type of battery. Rechargeability is a heavier weight because it is directly tied with the specification of SentryEye, that it should self-charge to be independent and low-maintenance. Though not as important as chargeability, lifespan (the number of charge cycles) is important so the device can last a reasonable amount of time. The operating nominal voltage is rated second highest to the form factor since it essentially dictates the resulting design that will power the system. The higher the nominal voltage, the more convenient the design will be to ensure SentryEye is properly powered. The nominal temperature range is high in our weighting because it is a factor that directly affects whether the battery is working safely. The region in which we reside tends to have colder temperatures, which is a factor for consideration since SentryEye will be in vehicles not in operation in the cold for extended periods of time.

4.3 Software Design

When designing a product in the security industry, service reliability matters. Service reliability can be achieved through the use of a microservice architecture, which consists of splitting the software up into cohesive components that can act independently of one another. The device software should not stop working as a result of a failure in an unrelated server routine. Microservices also enable scalability of individual components as needed. If there is increased load in the customer payment processing software component, we can individually attribute more resources to that piece of software, or even create additional server instances to support the increased demand. For these reasons, this is the architecture used in SentryEye's software ecosystem.

At the top level, SentryEye's software is broken down into two separate packages. One package is for the software that runs on the device hardware, and the other package contains modules that run on SentryEye web servers. Each of these packages is maintained using GitHub repositories for version control. This allows us to develop features separately from the stable version of the code with the ability to merge those changes in once they are adequately tested and working as expected. Using git repositories also allows us to more easily collaborate and work remotely.

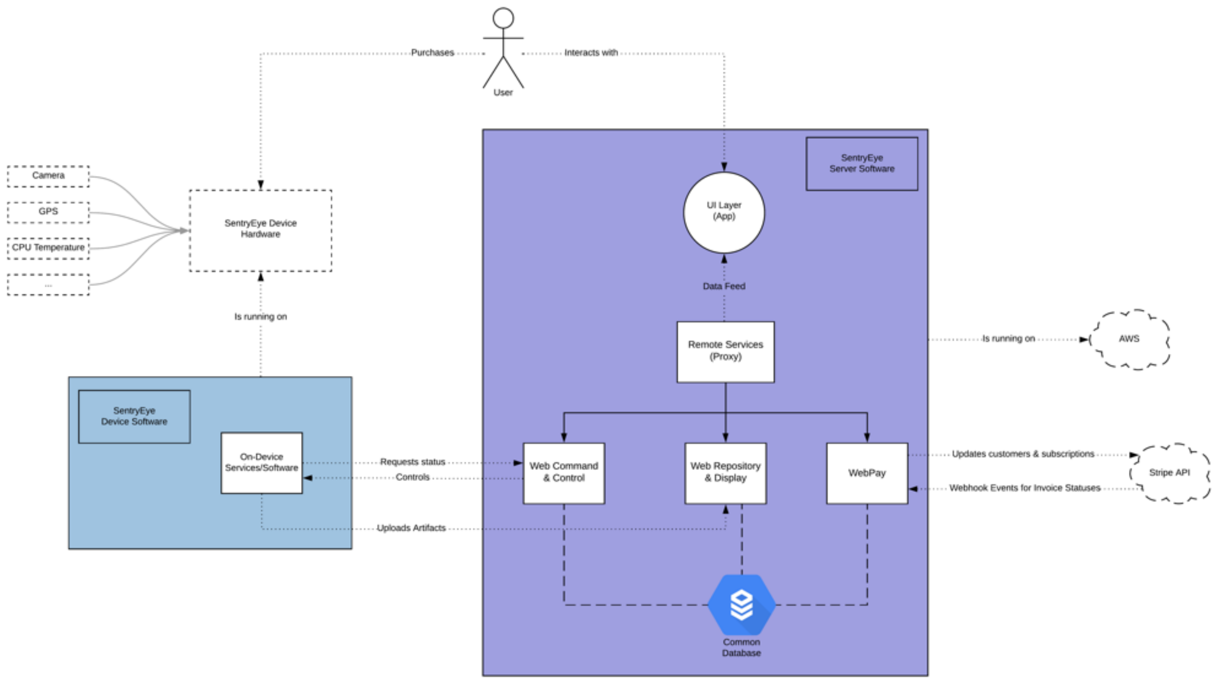


Figure 14: An overview of the organization of SentryEye Software components

4.3.1 Web Application User Interface

As a user, one of the biggest problems in this industry is finding the right complimentary mobile or web application that presents you with an effective means of keeping track of device information. In our case, there are many different data formats that we need to handle. For instance, we are in the business of handling videos, pictures, audio, or even different forms of sensor data such as GPS coordinates. With multimedia platforms in general, there are usually so many different ways of presenting such information. Most of the problems in developing these types of platforms are generally found when a developer is trying to effectively provide the user with what they need without overwhelming or underwhelming the user experience. One notion that the developer must keep in mind is that it is always important to account for the context in which the user is conducting an action. Generally, with SentryEye, there are two possible use cases where a user would be communicating with their Sentry device: the individual would be interacting with the SentryEye device either through a mobile device or from a computer.

In our current age, apps are most commonly mobile-driven due to the wide accessibility of portable devices in the world in comparison to computers. When we look at the scope of our industry, the mass availability of mobile phones has led to the decline in the quality of web applications that complement their parent mobile apps. When it comes to choosing an ecosystem to develop, there are both advantages and disadvantages to being web-driven vs. mobile-driven.

With SentryEye, we wanted to develop a platform that was not limited by web or mobile technologies. When building solely mobile applications, there are several features that cannot be implemented simply because of the limitations of the device. Generally speaking, mobile applications are usually a scaled-down version of a bigger ecosystem. With that being said, some applications may not require as many technical resources, which makes mobile the perfect platform to start with. In SentryEye's case we did not want to limit what features we could develop just because we chose to solely develop for a mobile environment.

Web applications also have their limitations. Given a bigger screen and more computing power in a lot of cases, there inherently is more complexity that can be involved in feature development. This can be a good thing, but it can also be a terrible thing for a user. Developing on the web can also bring about feature creep because of the computing power available. A basic web server can handle a much stronger load than the average mobile device. This could result in a blurring of the initial focus on the problem you were originally trying to solve. Being too feature rich can inadvertently cause negative effects on the user's experience. Furthermore, in the mobile community that we live in, screen time will most likely be greater on mobile than it is on anything else.

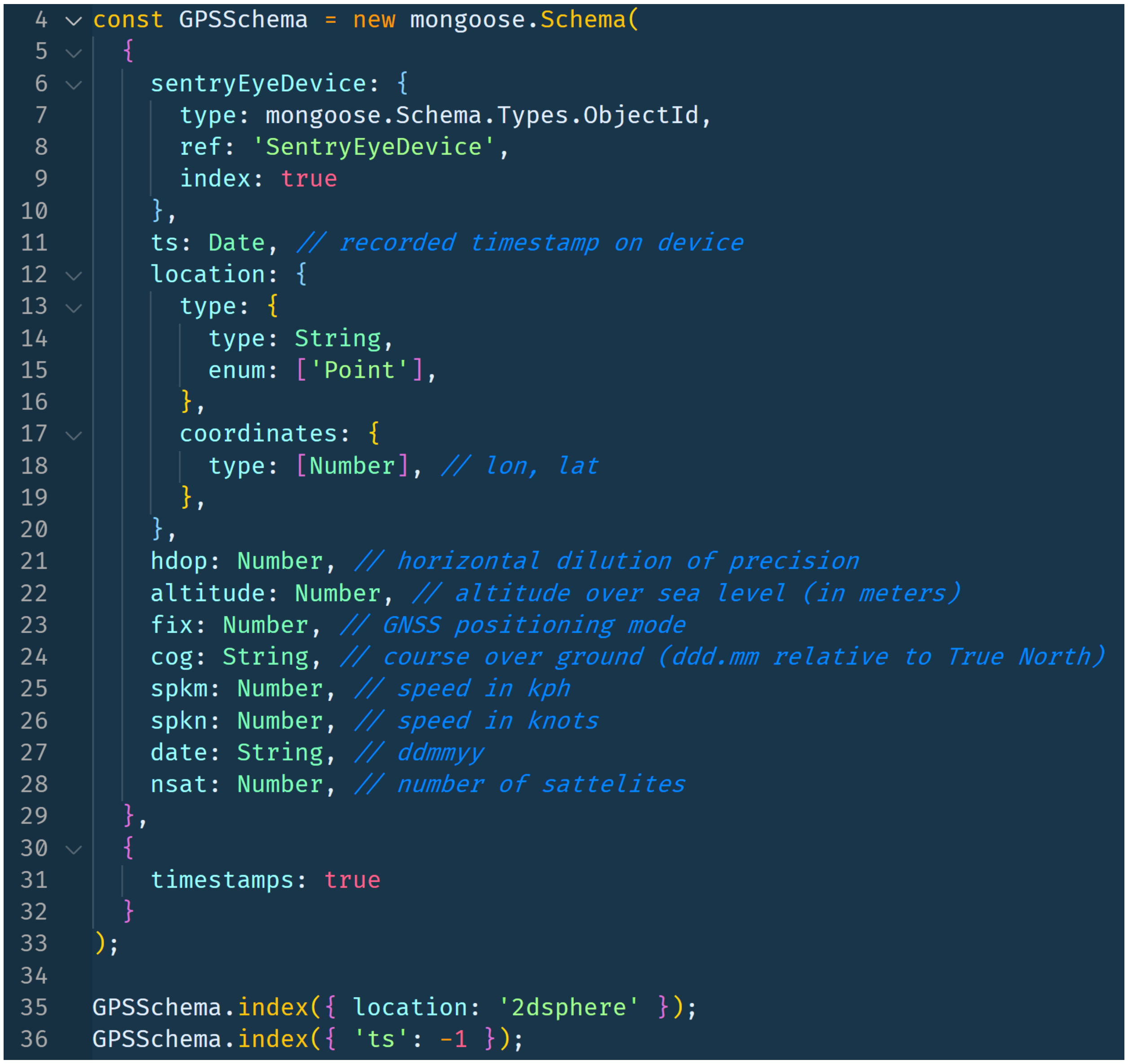
Both web and mobile have their qualities that make them good choices, but in the case of SentryEye we chose to be web driven. We do believe in supporting mobile as well; however, beginning our development with the web forces us to be more open to the user's needs. As web technology continues to advance, it is also becoming increasingly easier to design websites responsively so that they scale to any screen size, meaning users will be able to comfortably use the SentryEye app from a mobile browser. Interestingly enough, many of our competitors are mobile-first, and we believe that our flexibility within itself could serve as a separation between us and our competitors.

4.3.2 Web Services and Database Structure

In choosing a technology stack for designing backend services to support the SentryEye infrastructure, it is important to take into account both the effectiveness of using this technology in the context of SentryEye as well as the developer experience. Node.js is a JavaScript runtime environment built on Chrome's V8 JavaScript Engine, and it is the technology solution we chose for implementing server-side logic. The primary benefit to writing the backend application in Node.js is that it enables the developers to use a common language for both front and backend development with very subtle differences between them. This can speed up the development process because developers spend less time context switching and more time being productive.

4.3.2.1 Database Design

When selecting a database for the SentryEye web server, we wanted to ensure that it was flexible enough to allow us to rapidly iterate. NoSQL databases fit this need, and the development team has prior experience in using MongoDB (a NoSQL, document-based database system). We also wanted to be able to maintain some level of control over the schema of our database models, which can be challenging given the fact that NoSQL databases are by nature unstructured. By making use of a schema-based solution to this problem, otherwise referred to as an Object-Document Mapper (ODM), most of this validation, casting, and business logic can be handled out of the box. The ODM allows us to describe the fields, data types and sizes of our data for each entity in the database. For example, an ODM schema for the GPS readings from your SentryEye device might look something like this:



```
const GPSSchema = new mongoose.Schema(
  {
    sentryEyeDevice: {
      type: mongoose.Schema.Types.ObjectId,
      ref: 'SentryEyeDevice',
      index: true
    },
    ts: Date, // recorded timestamp on device
    location: {
      type: {
        type: String,
        enum: ['Point'],
      },
      coordinates: {
        type: [Number], // lon, lat
      },
    },
    hdop: Number, // horizontal dilution of precision
    altitude: Number, // altitude over sea level (in meters)
    fix: Number, // GNSS positioning mode
    cog: String, // course over ground (ddd.mm relative to True North)
    spkm: Number, // speed in kph
    spkn: Number, // speed in knots
    date: String, // ddmmyy
    nsat: Number, // number of sattelites
  },
  {
    timestamps: true
  }
);

GPSSchema.index({ location: '2dsphere' });
GPSSchema.index({ 'ts': -1 });
```

Figure 15: Sample ODM GPS Schema

Every database inherently has risks associated with it (particularly risks of injection). For this reason, it is important never to trust end-user data, and always to sanitize and validate every input to the system. Fortunately, this validation and injection protection is also performed by the ODM, so there is very little validation that we need to incorporate manually when it comes to trusting end-user data.

Our other primary concern when it comes to the database design for SentryEye is that the performance of the database is consistent even with a very large amount of data in it. This is handled by creating indexes on each of our database collections that support the most common database queries that we will make. For example, if we want to be able to sort the

GPS data by date and time, it makes sense to create an index on that field in the GPS collection. The perfect analogy for this concept is postal delivery. Imagine that the houses on your street were not numbered – the person delivering your mail would have a lot of difficulty trying to find your house to deliver your mail, and they would likely have to knock on every door until they find you. Databases function the same way: if it is searching for documents that meet certain conditions (the database query), it needs to check every document. Computers are fast, so this is not a big deal if there are only a few dozen documents, but when there are potentially millions of documents in the database this can lead to a significant bottleneck of the system. By indexing the fields that we commonly query, the database can do some very clever things when fetching documents. This significantly reduces latency and memory usage on the server and makes the system much more scalable.

By using an ODM and being very careful that our schema design and indexes meet the needs of our backend system, we are able to have a degree of trust that the database will be secure, robust, and scalable.

4.3.2.1.1 Image and Video Storage

MongoDB stores data in BSON format. BSON is short for Binary JSON, or Binary JavaScript Object Notation. The details of its implementation are not particularly important to understand for the purposes of SentryEye, but it is important to know that when reading and writing data, the database must parse the BSON documents to make sure they meet the specified format. Because they need to be parsed, BSON documents have a size limitation of 16 megabytes. This is a challenge for us, because we are storing images and video, which can easily exceed this 16-megabyte limitation.

Another reason MongoDB was the database of choice for the SentryEye's web services is that MongoDB's GridFS specification allows storage and retrieval of documents that exceed the BSON size limitation of 16 megabytes. GridFS provides the system with a storage format that is excellent at storing large binary data, such as the images and videos that are captured by SentryEye devices. It does so by splitting the image or video up into smaller data chunks and distributing these chunks across more than one BSON document. The direct alternative to this format is to store the images and videos directly on the file system, which is much less scalable. By storing them in the database, the data can be easily replicated to additional servers as needed to support scaling the SentryEye infrastructure (Ex: if we deployed SentryEye servers on the west coast of the US or in Europe, MongoDB has tools to make sure the database is synchronized across these different regions). Storing these artifacts in MongoDB GridFS also means that they are included in the automated backups of the database.

Another benefit to using GridFS is that the binary data is split up into chunks of 255 kilobytes before being stored, and the MongoDB driver enables us to retrieve portions of these large binary files. This is hugely beneficial when it comes to transmitting the video files over HTTP because it allows us to make use of chunked transfer encoding. This means that the video can be streamed to the SentryEye web app in segments, rather than transferring the entire video file all at once. Benefits to this method of data transfer include more efficient use of network resources (only use network bandwidth as needed), allowing the user to scrub through the video (skipping forwards/backwards similarly to a YouTube video), and reduced server load since the entire video file does not need to be loaded and sent in a response all at once.

We purchased the domain `sentry-eye.app` via Google Domains. Using their DNS configuration settings, we can create A records to point to the static IP addresses of any SentryEye web servers we have hosting the software. A records redirect traffic from a domain name, such as `sentry-eye.app`, to an IP address that can handle the HTTP requests. We also created a CNAME (which maps subdomains to a canonical domain name) for the subdomain `dev.sentry-eye.app` so that we could have a development instance of the SentryEye software set up for testing software deployments and iterating on the software as needed.

As mentioned at the beginning of this section, SentryEye's server architecture follows a microservice pattern, enabling us to build small, maintainable pieces of software that are very cohesive and loosely coupled with one another. However, this brings about the challenge of deploying those services and managing user requests via HTTP. The server-side software is deployed on an Amazon Web Services (AWS) Lightsail instance, which is a virtual private server (VPS). NGINX is used as the reverse proxy for each microservice, so that different types of requests can be properly routed to the correct piece of software to handle the request. It is important to note that this AWS & NGINX configuration is infinitely scalable with the introduction of a load balancer. Load balancing can be done in many different ways using AWS both at the application layer and network layer, including regional or zone-based DNS resolving so that users get a SentryEye server that is close to them. This reduces latency and packet loss and improves user experience when scaling software-based services nationally or globally. For the purposes of this project, however, SentryEye software is deployed on a single AWS compute instance with requests that are proxied using NGINX. These requests are passed to one of the following microservices to be handled:

- Web User Interface (UI)
- Web Command & Control (WebCC)
- Web Repository & Display (WebRD)

- Web Payment Handler (WebPay)

4.3.2.2 Web User Interface

Reusable component design was a big part of our design system. One major emphasis we made on the UI side was selecting a modular framework (ReactJS) that would fit these needs. React allowed us to quickly create the necessary components that would scale across various parts of our web platform.

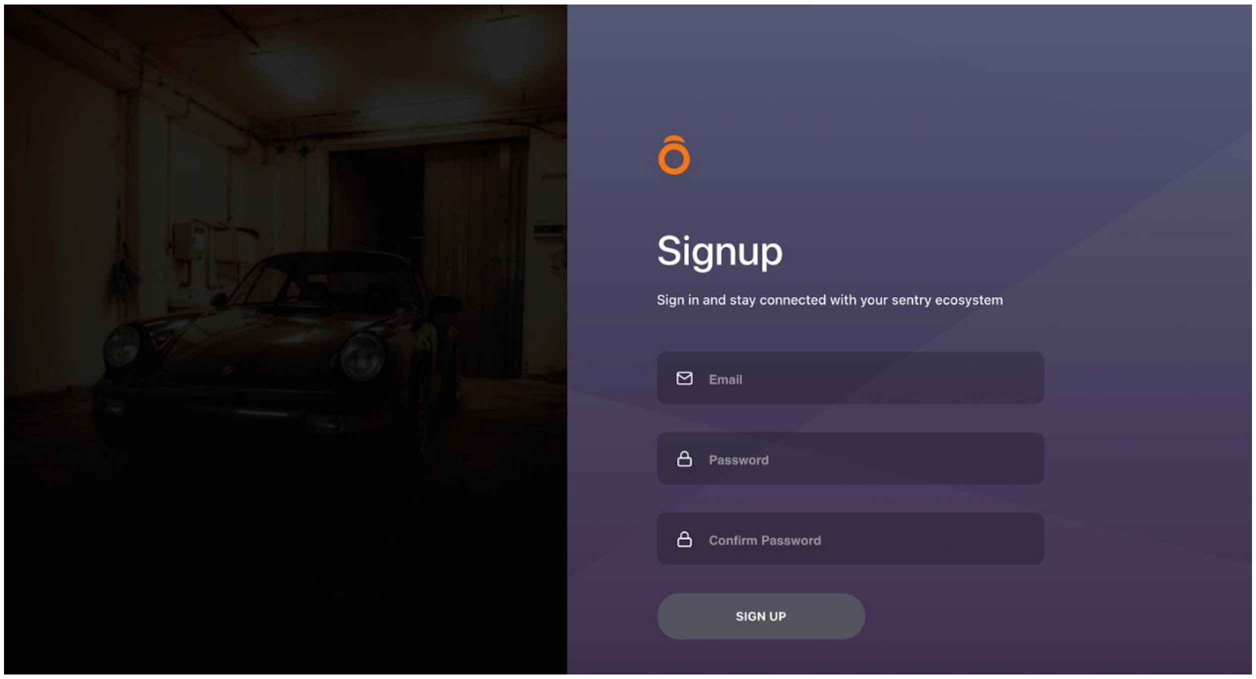


Figure 16: Registration Page for a New SentryEye User

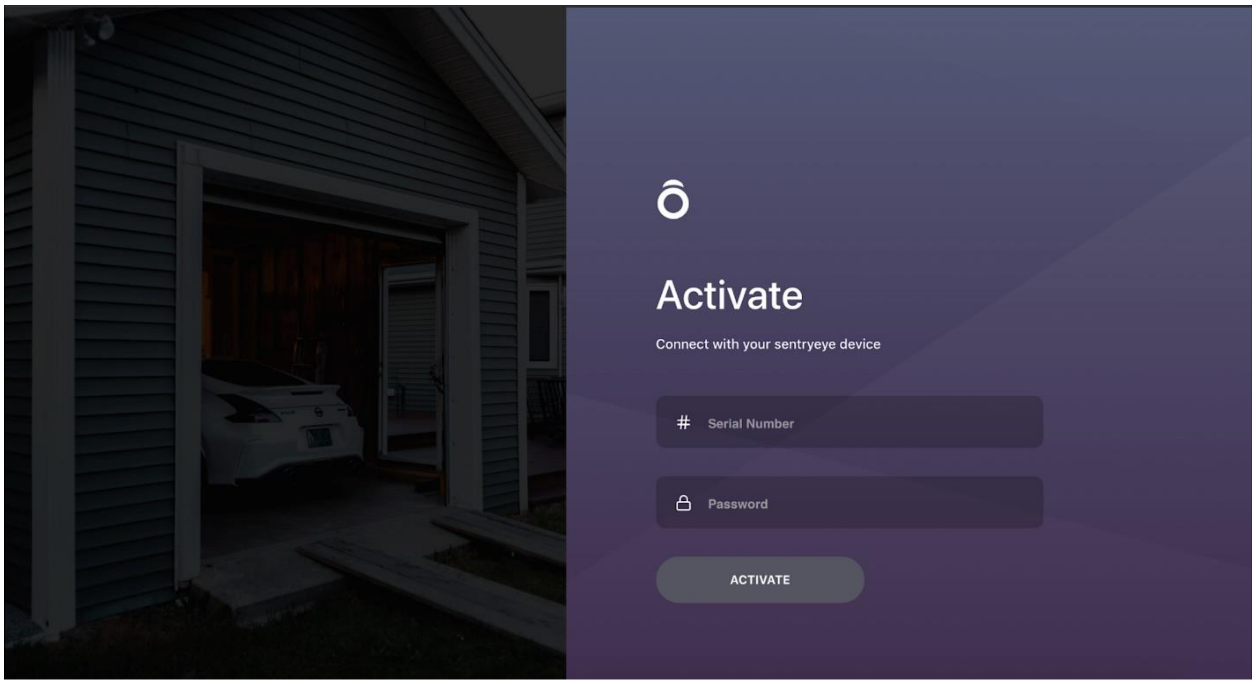
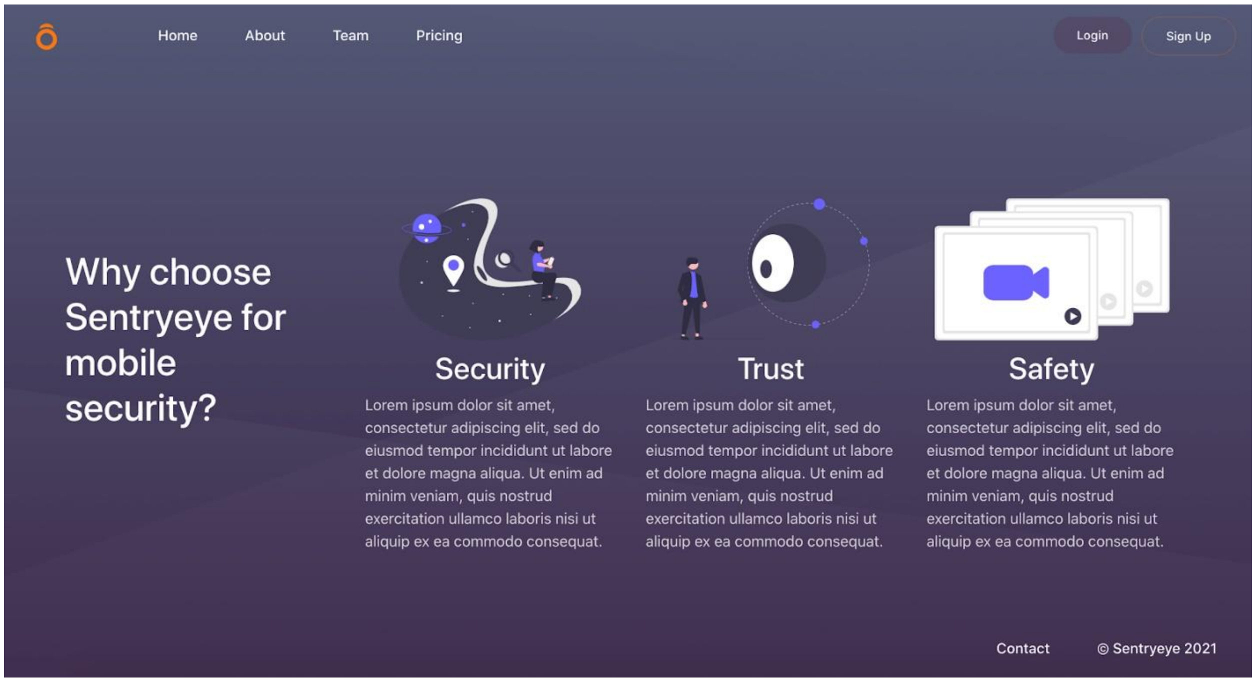


Figure 17: Activation Page for a SentryEye user

One example of this design approach can be found in our login page. If you look closely at the figure above you will notice that the layout is very similar to that of the layout in the activation page. Traditionally, using vanilla JavaScript one would have to manually duplicate code if they wanted to achieve what we have done above. Thankfully, React provides us with the ability to utilize Class level components that can be used elsewhere.

Due to the flexibility of React. We are able to do this with almost any component imaginable. Whether it be a button, a form, a combination of both, an input, or a whole page we can reuse anything that we write previously.



Why choose Sentryeye for mobile security?

Security

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Trust

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Safety

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Figure 18: About Page for the Web Platform

Another example of this can be found within our About view above. You'll notice that the security, trust, and safety mid-section can all be grouped as one component that takes in a title, description, and cover image. Instead of having to individually map all these out, using the React Framework we were able to reduce technical debt.

The role of the React microservice was that it allowed users to make an API request to the backend server. Whether it was a request in relation to WebCC, WebRD, or WebPay we had the infrastructure to handle all interactions with a single frontend framework. This is where Human Computer Interaction (HCI) came into play as well.

With all the work we did on the hardware and backend side of this platform, it was necessary for us to think about how the user would interact with the Sentry web application. If this was not a smooth interaction, then that would kill the overall user experience. Our frontend played a major role in making sure SentryEye was not simply a hardware product, but an ecosystem.

4.3.2.3 Web Command & Control (WebCC)

The Web Command & Control software component handles communication between users and their devices. When the user changes the settings of a device, such as changing it from photo mode to video mode, the request is handled by the WebCC. The SentryEye device will then fetch the changes made to the settings and make adjustments to the on-board software as necessary. WebCC also handles a variety of other tasks, such as user account creation, login, notifications, and SentryEye device activation.

4.3.2.4 Web Repository & Display (WebRD)

This software component is responsible for processing and storing any artifacts that are uploaded by SentryEye devices. When images, videos, and GPS data are uploaded, the SentryEye device also senses a secure, unique token to identify itself to the WebRD. This allows the server to trust that the data came from a valid SentryEye device, as well as identify the user that owns the device so that we can send the user a notification that their device has detected motion. The images are uploaded in JPG format are stored making use of MongoDB GridFS. The videos are captured and uploaded in a raw format, so they are converted into mp4 format by the WebRD using ffmpeg, which is an open-source tool for video conversion, before being stored by MongoDB GridFS.

The WebRD also handles HTTP requests for images and videos via the WebUI system, so that users can conveniently view and download artifacts captured by their SentryEye devices.

4.3.2.5 Web Payment Handler (WebPay)

The WebPay system handles customer payment for the SentryEye subscription services and includes an integration with Stripe. Stripe is a cloud software company that enables companies to accept payment easily, safely, and securely from customers and manage billing cycles for subscriptions. Customers can sign up for a SentryEye subscription plan using any payment card via the WebUI. This transaction is handled by Stripe, and an automated billing cycle is setup. Stripe then sends webhook events (which are essentially notifications) to the SentryEye WebPay server whenever the subscription is cancelled, renewed, upgraded, downgraded, or their payment card is rejected. These webhook events are handled by WebPay, and customer records in our database are updated accordingly. In summary, Stripe handles the billing cycles and payment processing, while SentryEye web servers handle the provisioning of the service (what the user gets access to with their active subscription).

4.3.3 On-Device Software and Services

On-Device Software and Services consists of everything that runs on the SentryEye device. The core of the on-device software is written as a set of various Python scripts that handle independent tasks that the device needs to perform. These tasks include verifying the device is activated and associated with a user account, performing motion detection, capturing images and video, connecting to Wi-Fi, uploading the images and video, capturing GPS coordinates. Each of these python scripts is launched from a unique bash script when the device is powered on. The bash scripts make it convenient for us to start the device software as a service making use of systemd unit files. This way, the operating system manages the services and tasks for us. This also has the added benefit that in the case of an unhandled error in any of our code, systemd will automatically try to restart the errored service, keeping the device functional.

5 Product Functionality / Results

As SentryEye is a continuously changing and adapting project, we were not able to, nor did we wish to, keep the design unchanged since the moment of its inception. For a summarized view of the iterations on SentryEye please refer to the Table 9.

Table 9: Design Module Modifications

Design Aspect	Has This Changed	Change Description
Case	Yes	• Previously lateral sections were free hanging. Attaching these with silicon or Velcro was considered but ultimately dismissed in favor of adding **snap connectors** to the case. • The next design for the case also has an **additional archway structure** in order to add support to the camera module that it might not cause stress to its CSI ribbon cable. • The last iteration of the device has a completely redesigned case to account for the cellular LTE module and audio PCB. These no longer need the snap connectors or archway structure as it is a unibody case design.
Solar Panel	No	• Neither the solar panel nor its role in the device changed.
Charge Controller	No	• The TP4056 charge controller's **position** was adjusted to a more accessible orientation within the case. • Long cables that have since been spliced were also a factor in the decision to shift the controller's position. • In the last SentryEye prototype, this charge controller was scrapped in favor of the PowerBoost 500c's integrated charging circuit. This created more room within the case and fewer soldered joints, which reduces the potential points of failure within the device.

LiPo Battery	No	• Neither the LiPo Battery nor its role in the device changed.
PowerBoost 500	No	• The PowerBoost 500 was upgraded to its 500c variant to include an integrated charging circuit and remove the need for the TP4056 charge controller.
Raspberry Pi Zero W	No	• Neither the Raspberry Pi Zero W nor its role in the device changed.
Camera Module	No	• The Arducam wide-angle lens chosen incorporates a fisheye lens capable of a 170-degree view and has remained unchanged from the original design.
LED Deterrent	Yes	• SentryEye's original design first called for the use of the GPIO pins to solder an **external LED onto the microcontroller**. With the Arducam camera module, an integrated LED provides this functionality.
Cellular Module	Yes	• In the latest iteration of SentryEye, the Sixfab Cellular LTE module was added for cellular and GPS location connectivity.
Audio Component	Yes	• A custom single transistor audio circuit was designed and printed for use with the Raspberry Pi Zero W's GPIO pins. The printed circuit also connects to a coin-sized 8-ohm speaker for the device's audio projection.

Figure 19: Fish-Eye Camera Module

Originally, SentryEye's design was comprised of a quarter arch case which served as both a hub for every component and protection from the external conditions. However, the orientation needed to be adjusted due to the length of the USB to microUSB connector. Additionally, the inclusion of both the top and bottom case of the Raspberry Pi Zero W (which came with the microcontroller) added more volume, which exacerbated to the issue. In order to better fit our preselected form factor, the positions of our device's modules were altered, and the case was redesigned to have a more uniform shape. The case was also designed to better accommodate the fisheye camera module, shown in Figure 19 above. The USB to microUSB cable was spliced and shortened in order to reduce the previously large space it held within the device. Shown in Figure 20 and Figure 22 are side views of the internal components. Figure 21 shows the 3D printed case without the components mounted.

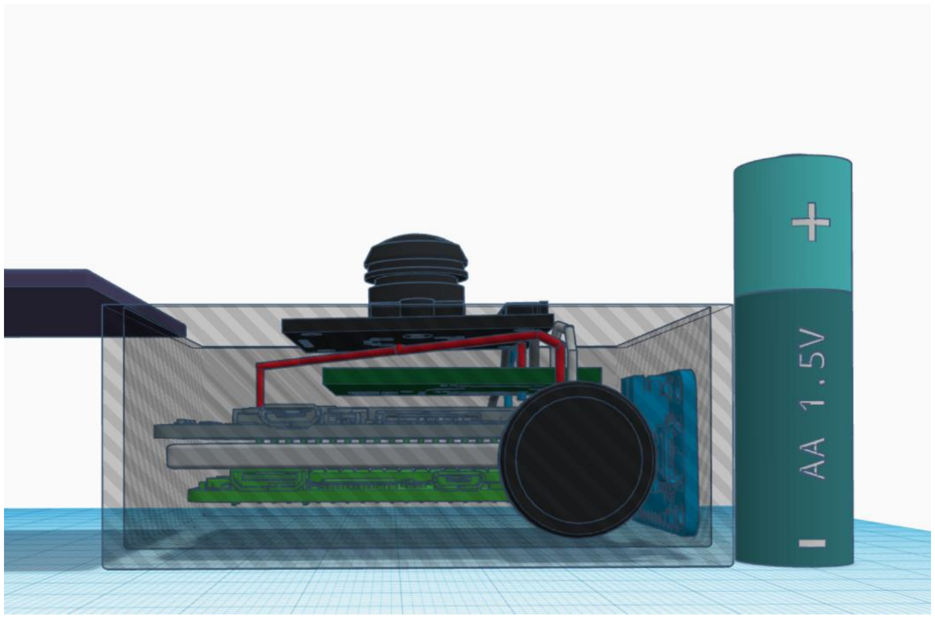


Figure 20: SentryEye Side View of Internal Components, AA battery for scale

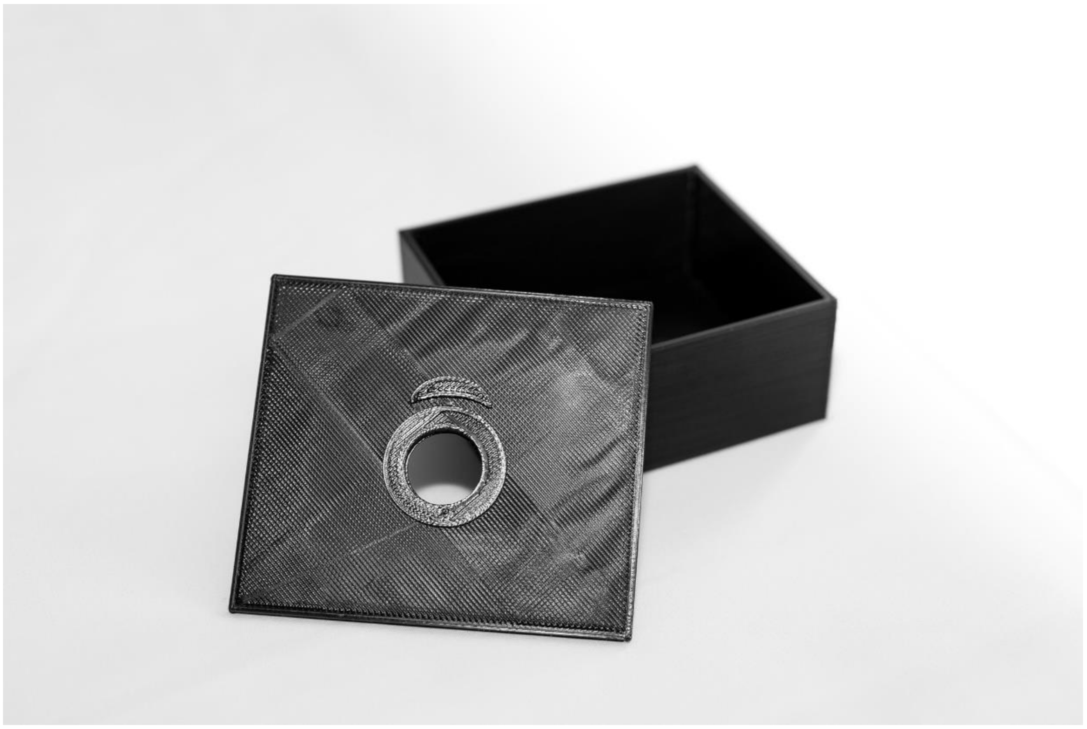


Figure 21: 3D printed prototype SentryEye case

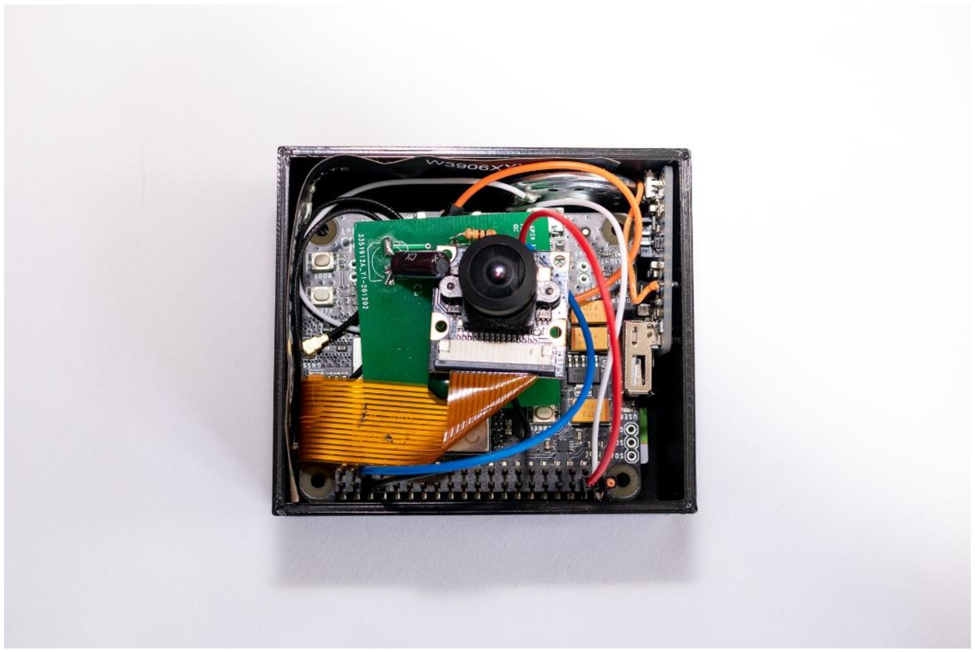


Figure 22: Prototype Top View of Internal Components

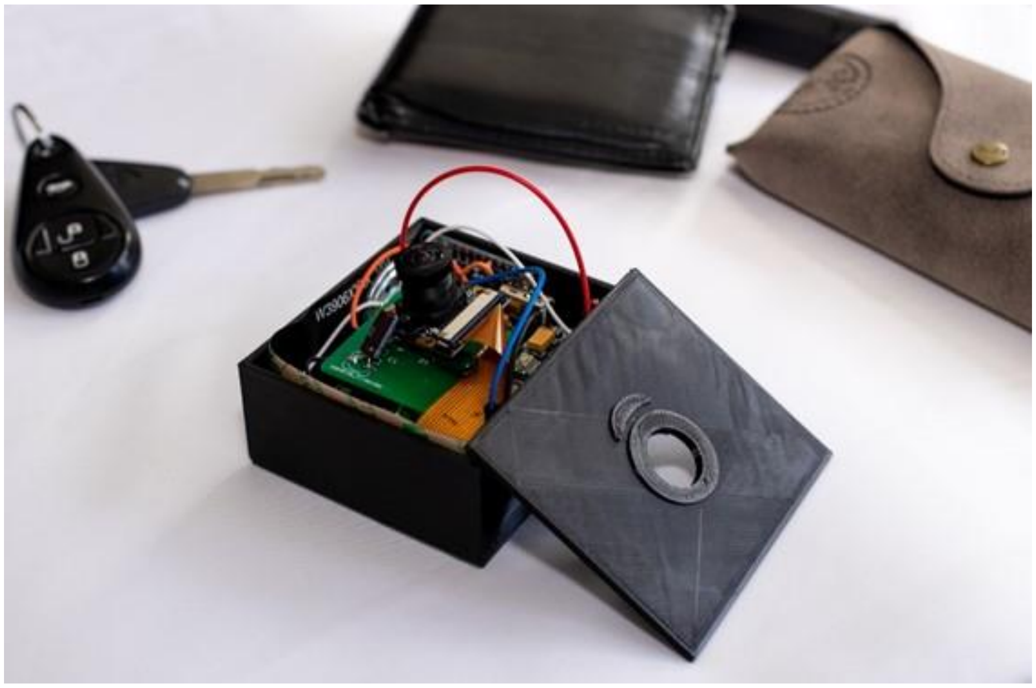


Figure 23: SentryEye prototype case with components inside, cover removed

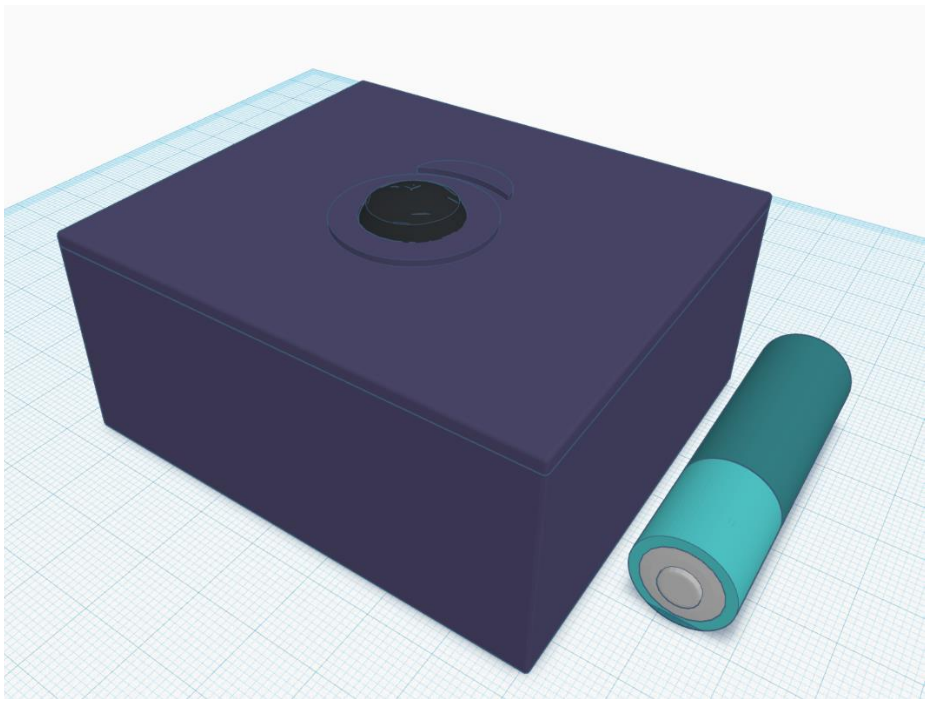


Figure 24: SentryEye 3D-Printed Case, AA battery for scale

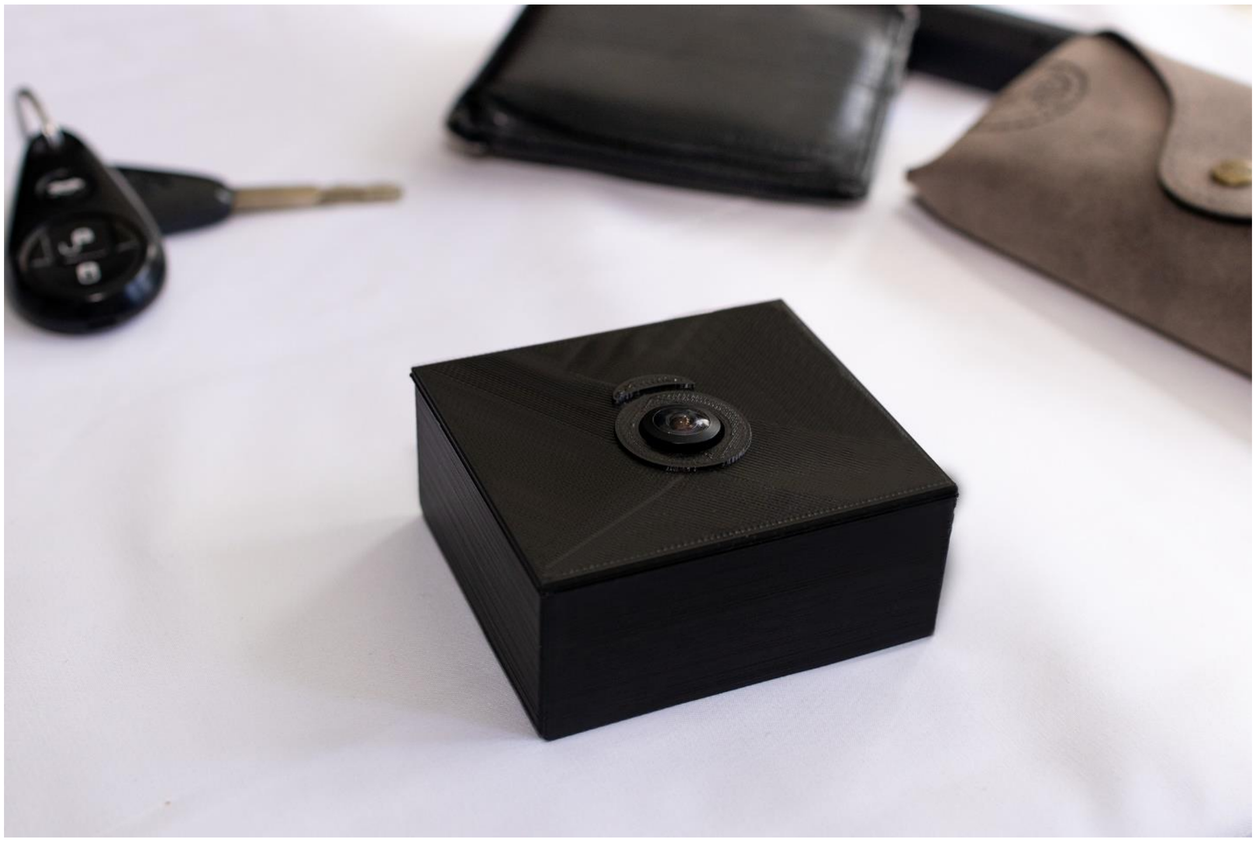


Figure 25: SentryEye prototype product photo

5.1 Revised Product Design

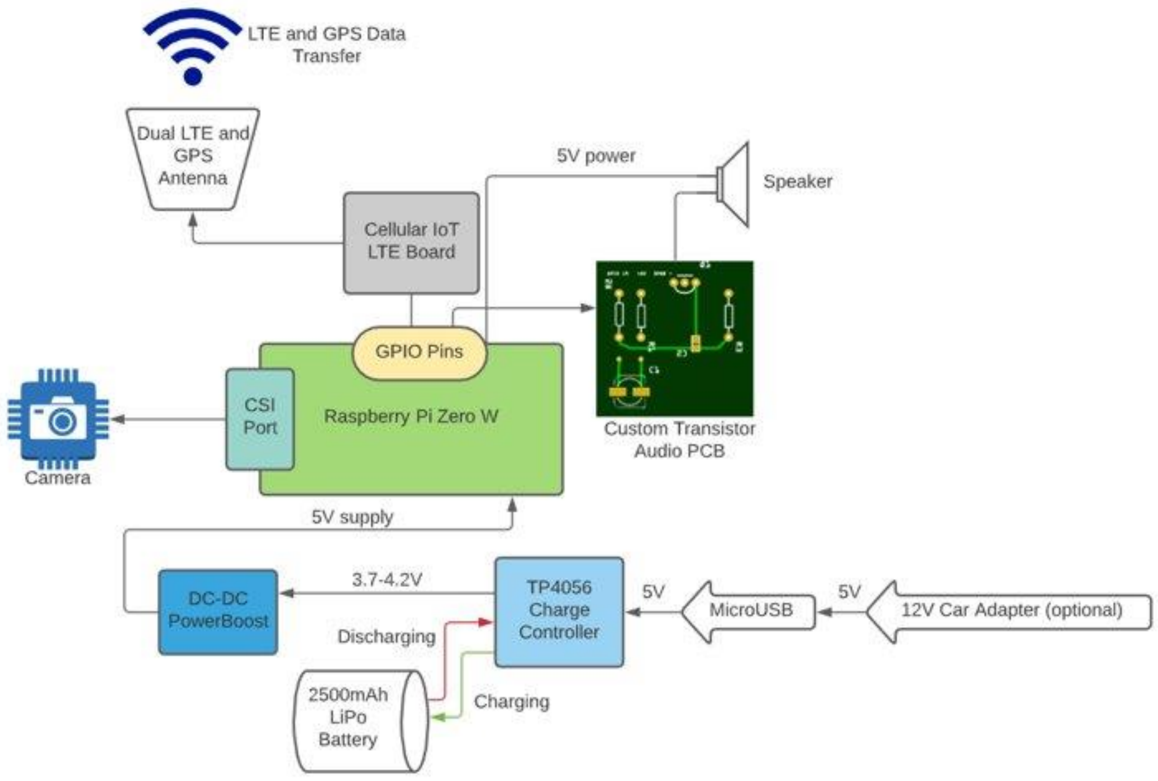


Figure 26: Hardware Block Diagram (before integrated charge controller)

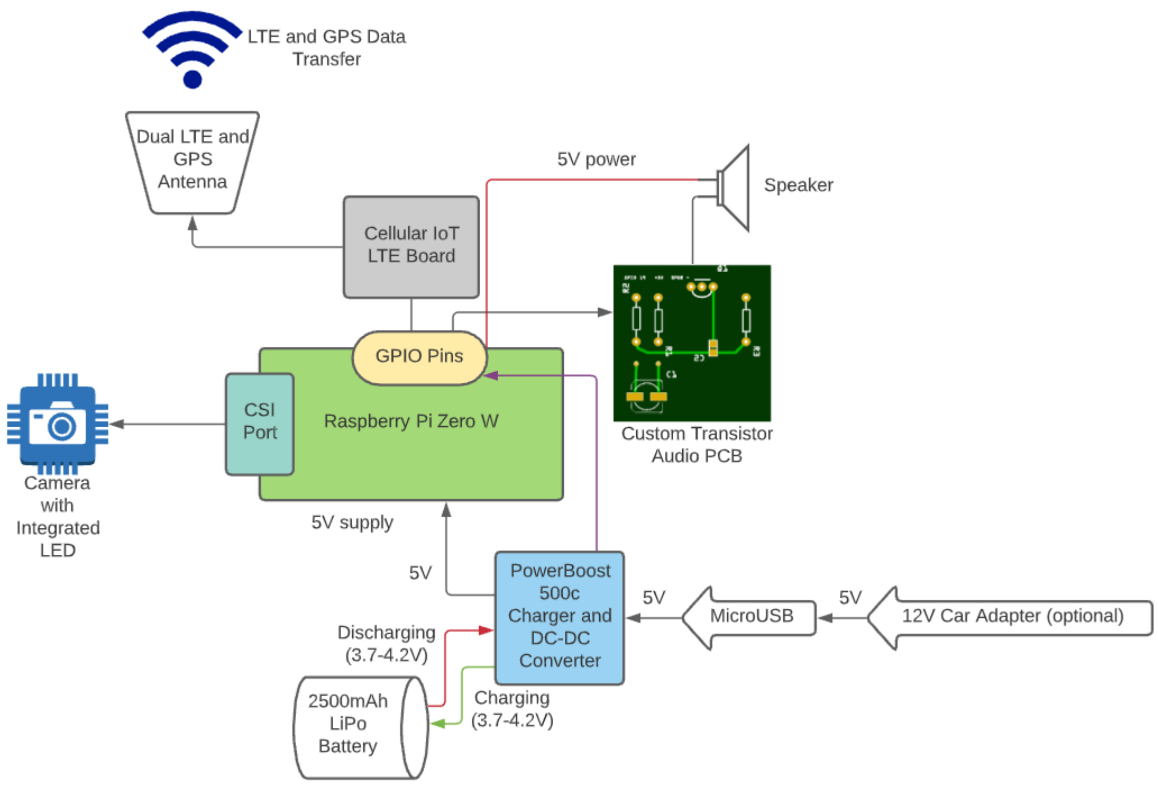


Figure 27: Hardware Block Diagram (after integrated charge controller inclusion)

The overall product design remained relatively unchanged concerning each respective module according to the block diagram shown above. We decided that instead of soldering wires directly onto the PowerBoost 500c board to provide the 5V power to the Raspberry Pi Zero W, we would first solder a USB connector on the available terminals of the board first. In this manner, we were then able to connect and disconnect the PowerBoost 500c from the microcontroller’s microUSB port with ease during the troubleshooting and testing phase of the prototype’s development. Additionally, all available microUSB cables proved too large and inflexible to fit properly in the SentryEye case, so instead, a flat microUSB cable was cut, stripped, and each individual cable within the wire casing was spliced together to form a shorter cable. Electrical tape was then utilized to coat any sensitive ends to ensure the splicing process as well as the applied solder held in a sufficiently strong manner.

Originally, the design included the use of a TP4056 charge controller board in addition to the PowerBoost 500 DC-DC converter. However, this design was upgraded by utilizing a new PowerBoost 500c with an integrated charging circuit. This addition saves space within the case and provides fewer points of failure within the organization of the device.

5.2 System Testing & Results

SentryEye is comprised of both hardware and software; described in this section are the plans and methodology for integration and testing process for both hardware and software. This section identifies the hardware and software tests performed as well as the testing criteria for future module-specific tests required for large scale production. Additionally, we will also describe our process and indicators for testing at the system level.

5.2.1 Hardware Module Integration & Testing

The hardware component of SentryEye is made up of a battery, which serves as the main power source of the device, a charge controller, which regulates the voltage and current when charging the battery, a DC/DC booster board, which steps up the voltage to power the microcontroller, and the microcontroller itself, which serves as the brain of the system. Additional hardware are the peripherals, which include the wide-angle camera for image capturing and the solar panel for trickle charging the battery. Many of these modules were designed to be compatible with other selected modules, which simplifies parts of the hardware integration and testing. For a detailed overview of the interconnection of the different components in terms of voltages and current, see Appendix B.

5.2.2 System Deployment Test / Candidate Build

To test the capabilities of the SentryEye device, the system was evaluated in several different environments. First, the device was tested within the confines of an apartment to test its Wi-Fi capabilities. The device was tested as if the product had been purchased and opened by a consumer for the first time. The device was paired with the customer's account over the LTE network. The customer then added the Wi-Fi network to the device so that images and video taken by the device would be uploaded over the wireless network. The device was set up inside the apartment and armed for motion capture. Running on battery power alone, the device captured and uploaded 23 pictures to the SentryEye servers over the course of an hour.

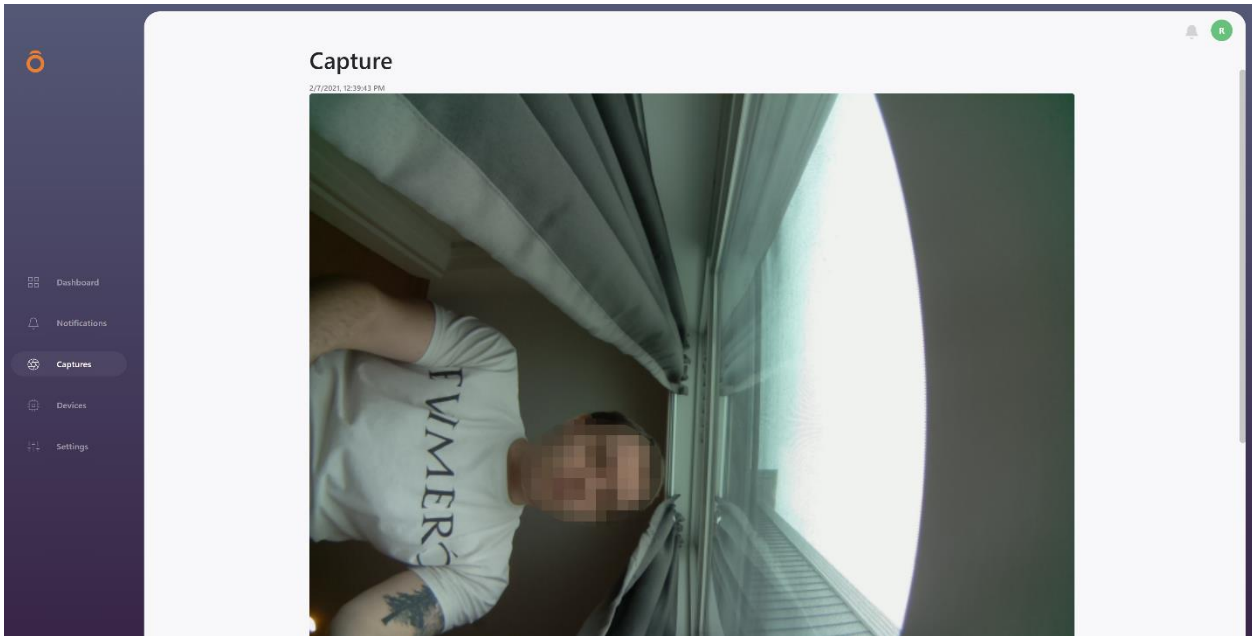


Figure 28: Sample capture from system testing inside an apartment; image blurred to protect privacy for inclusion in this paper

For the next test, the device was taken outside to test the connectivity under an open sky. The device was set up in a manner such that images were captured via movement detected in front of the lens from different angles as we walked by the mounted camera. This was done over the course of an hour with several intentional delays of movement between each capture. The location data from the GNSS/GPS module in the Sixfab board was analyzed for each capture. The GPS data was determined to be sufficiently accurate to within 10 meters of the device's location each time. These tests indicate that the SentryEye prototype has the complete desired functionality of the product. Additional hardware and software-specific future tests are outlined below.

5.2.2.1 Battery Connection and Boosting to Power Microcontroller

The LiPo battery selected to power this device was identified to have both desirable form factor and battery capacity. The battery outputs a 3.7-4.2 nominal voltage and has a capacity of 2500 mAh or 10 Wh. However, the Raspberry Pi Zero W microcontroller to be used in SentryEye requires a voltage of 5 V to enable the device, which the LiPo battery alone cannot sufficiently provide. As such, a DC/DC boost converter is needed to bridge the gap. We have selected the PowerBoost 500c to boost the 3.7 V from the batter to 5 V to power the microcontroller. To safely confirm these results, we will be measuring the battery and DC/DC boost connection to identify to real output value. Additionally, we will test and evaluate the performance of this integration in terms of power consumption. Since the system will not be operating at full capacity for 100% of the time, we will determine the different levels of power consumption of the different levels of operation that occur during a single cycle of operation. From these values, we can determine how long the device can truly last without being charged, and with trickle charging.

5.2.2.2 Solar Charging

Specified in the product requirement is the ability for SentryEye to have trickle-charging capability through using a solar panel. This is a feature of the system that will allow the system to restore battery charge over time in parallel with the system performing its surveillance, which will assist in preventing the device from prematurely running out of power. The solar panel used in the prototype has a maximum output of 5 V and 500 mA, or a total of 2.5 W in terms of power. To charge the battery, a minimum of 4.2 V is required. By connecting the charge controller part of the PowerBoost 500c between the solar panel and battery, we are effectively creating a protected system for the battery. The charge controller will step down the input voltage of 5 V from the solar panel to the desired 4.2 V while also allowing a maximum of 1000 mA current going into the battery to prevent overcurrent draw. The charge controller also has an overcharge protection feature, which stops the battery from continually changing once at full capacity. While these outcomes can be determined by extrapolating data from the specifications of each component, we will be testing and measuring the outputs of both the solar panel and the charge controller with the solar panel connected to observe their performance.

5.2.2.3 Camera Connection

In determining the ideal camera for our system, we have identified that many cameras used for similar experimental applications have been developed for integration and connectivity with microcontrollers in the Raspberry Pi family. As such, it was our intention to select a camera module that would be compatible with our system, which we have done. The Wide-Angle camera module has a CSI ribbon that connects the camera directly to the microcontroller. As such, there is likely little to no issue regarding compatibility and connectivity between the two devices. However, to ensure that we are receiving the desired results from the camera, we will conduct basic testing of the system to demonstrate how the camera functions and to view the contents captured by the camera.

5.2.2.4 Encasement and Mounting

In early design and conceptualization of SentryEye, a TinkerCAD model was developed using dimensions obtained from the specifications of each device. To ensure that all components fit inside the frame of the device, an initial 3D printed frame will be produced, with which we will assemble the device into the frame. Once we have verified that the components to fit into the casing, with minimal moving parts, we will test preliminary placement of the device. The device was designed to be mounted on different parts of a vehicle with weaknesses for breaking and entering. To properly fasten the device on to, for example, the inner frame around the window of a car door, and while avoiding introducing permanent damage to the frame itself, we will be using 3M adhesives (similar to the ones used for 3M Command Strips). This form of adhesion will be tested for its ability to hold the device still and in place for extended periods of time, both when the vehicle is in motion and parked, and will be evaluated on damage/cleanup when removed.

5.2.3 Hardware & Software Integration

For SentryEye to work as intended, the software and programming logic needs to be loaded onto the device. Once both hardware and software components have been tested, the software will be loaded onto the microcontroller. The Raspberry Pi Zero W has a slot on board for a memory card, with which it will store the program. We have decided to use an 8 GB micro-SD card out of convenience, which is sufficient space for the program. Our on-device software can be loaded onto the microprocessor by running an automated setup script on the microprocessor. In this phase, we will also conduct an iterative test for sensitivity of the system. This form of testing will allow us to identify how sensitive the program is for identifying prolonged occupation of the camera view, as well as being triggered to notify the user. The test will also work out issues related to false alarms, where a person walking by the camera of a large vehicle passing the camera should not cause the device to notify the user.

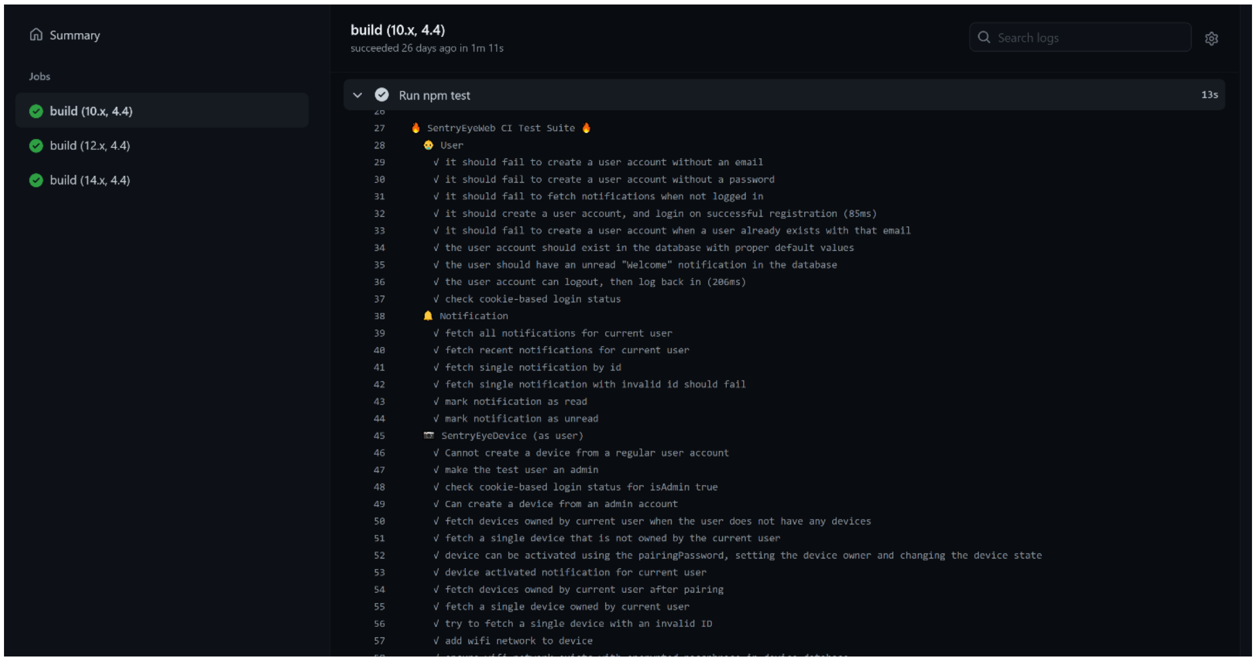
5.2.4 System Testing

Once all integration-level testing has been conducted and the system has been assembled, we will test the system in a mockup situation. To verify the functionality and performance of SentryEye, we will be placing the device into the vehicle of a consenting volunteer. The device will be mounted and tested in multiple areas susceptible to breaking and entering. For each iteration, we will verify if the device triggers on a prolonged occupation of a significant percentage of the camera's field of view and sending pictures to the user's phone, and if the device will not falsely trigger on people passing by the vehicle. From a product test perspective, this will help us determine optimal placement and identify any new weaknesses in the system. Due to the nature of this device and the time constraints of this project, we do not intend on performing evaluations and testing of the cycles, rather we will go off numerical calculations with nominal values for the conclusion in that regard.

5.2.4.1 Continuous Integration

GitHub has a built-in continuous integration (CI) platform, allowing us to run automated tests any time changes are made to our code base. We configured GitHub Actions to run a custom CI pipeline that installs Node.js, MongoDB, and other dependencies before building the database and testing all of our API endpoints. The CI run across multiple Node.js versions to ensure compatibility in different environments. The automated tests simulate actions that would be taken by the user such as creating an account and setting up a SentryEye device. There are also automated tests that simulate a device uploading pictures, videos, and GPS data. A sample output of SentryEye's CI build can be seen in Figure 29. All of this testing gives us good coverage of our codebase, and by setting CI up early in the development process, we were able to push changes with a high amount of confidence that there will be no negative downstream impact or bugs introduced.

When a test fails, we are notified of the exact commit and code changes that resulted in the failure. This gives us an excellent starting place to do some root cause analysis and discover precisely what caused the test to fail. From there, we have the option to revert those code changes completely, make modifications to the code so that the tests pass, or adjust the test cases if the functional requirements of the system have changed. This is fundamental to the reliability that we want SentryEye software to have.



```
🔥 SentryEyeWeb CI Test Suite 🔥
  😷 User
    √ it should fail to create a user account without an email
    √ it should fail to create a user account without a password
    √ it should fail to fetch notifications when not logged in
    √ it should create a user account, and login on successful registration (85ms)
    √ it should fail to create a user account when a user already exists with that email
    √ the user account should exist in the database with proper default values
    √ the user should have an unread "Welcome" notification in the database
    √ the user account can logout, then log back in (206ms)
    √ check cookie-based login status
  🔔 Notification
    √ fetch all notifications for current user
    √ fetch recent notifications for current user
    √ fetch single notification by id
    √ fetch single notification with invalid id should fail
    √ mark notification as read
    √ mark notification as unread
  📷 SentryEyeDevice (as user)
    √ Cannot create a device from a regular user account
    √ make the test user an admin
    √ check cookie-based login status for isAdmin true
    √ Can create a device from an admin account
    √ fetch devices owned by current user when the user does not have any devices
    √ fetch a single device that is not owned by the current user
    √ device can be activated using the pairingPassword, setting the device owner and changing the device state
    √ device activated notification for current user
    √ fetch devices owned by current user after pairing
    √ fetch a single device owned by current user
    √ try to fetch a single device with an invalid ID
    √ add wifi network to device
```

Figure 29: SentryEye software continuous integration for automated testing

6 Cost Analysis

To ensure the product feasibility from a financial perspective, we conducted a brief review and analysis of the cost of constructing our prototype of SentryEye, as well as an estimation of unit costs if we were to bring the product to commercialization. We also performed an analysis to determine profit margins at price points in the cost range we expect for SentryEye. Detailed in this section are the documentation of our prototype expenditure, the projected manufacturing expense, and a calculation of profit based on a brief analysis of anticipated costs.

6.1 Initial Investment

To create a prototype of SentryEye, we acquired parts from various vendors sold at retail price. As such, the total cost to create a single unit is not accurately reflected by the total spent to create the prototype. We realize that acquiring material in bulk directly from manufacturers would not only be economic but will also reduce the material cost to build one unit. The breakdown of the component costs is shown in Table 10.

Table 10: Component Costs (Original vs. Bulk Purchases)

	Original and Bulk Purchase Prices			
Components	Original Cost	Bulk Cost	Bulk	Amount Saved
Lipo Battery	$15/piece	$1.40/piece	100	$13.60
PowerBoost	$10/piece	50¢/piece	10	$9.50
Charge Controller	$1/piece	15¢/piece	100	85¢
Solar Panel	$9/piece	85¢/piece	40	$8.15
Fish-Eye Camera	$17/piece	$2.50/piece	1	$14.50
Raspberry Pi Zero W	$10/piece	$3.01/piece	20	$6.99
Sixfab LTE Cellular Board with BG96 Chip	$70/piece	$9.86/piece	1000	$60.14
Totals	$132.00	$18.27	------	$113.73

Our total for SentryEye is $131.15, which does not include the shipping fee. As can be seen from the table, the cost of constructing one unit using materials sourced from manufacturers is just $18.27, which would save us about $113.73 per unit.

6.2 Return on Investment (ROI)

Given the different price ranges and analysis of features from our competitors identified in our analysis of potential competitors, we expect our product to be sold at the price range of $50-$60, however further investigation and analysis was conducted to determine the results of this pricing range. It is our understanding that any accurate cost analysis should be more involved, requiring contact with experts and professional resources for each different contributing factor of overall cost and sales. However, for the purposes of our reporting to provide a general overview and brief estimate of return on investment (ROI), we will focus on significant factors whose values can be both relatively accurate and easily determined. In our analysis, we include the following specified to the best of our current ability and resources: bill of materials (BOM), labor, storage, marketing, and shipping costs.

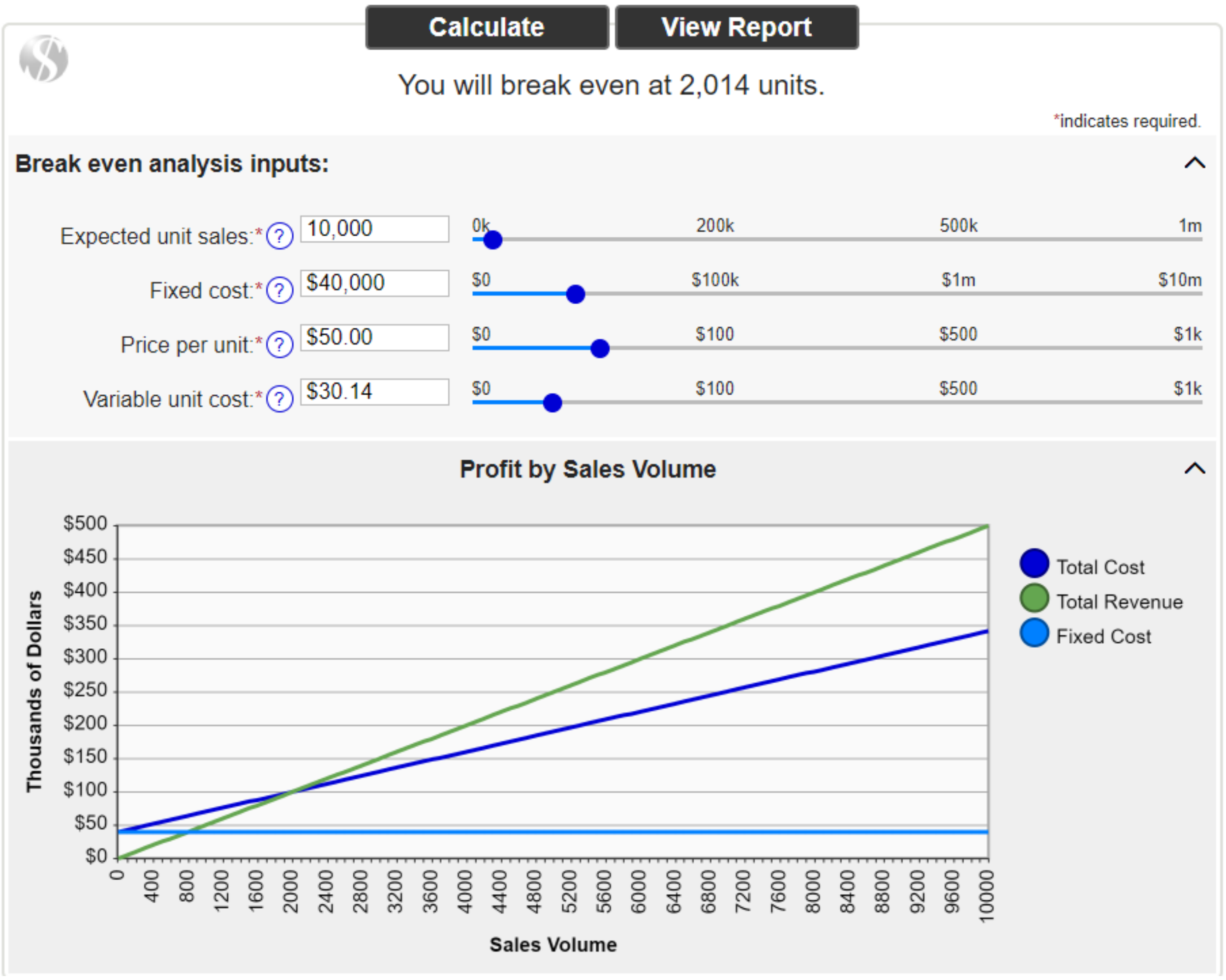
For our brief evaluation, we are assuming 10,000 units as the initial batch size. This size will be used for the subsequent calculations and to determine potential profits. In a well-defined analysis, a more appropriate value would be used, based on researched values for expected sales volume. However, we will use the nominal value of 10,000 due to time constraints, which also affects the overall dept of the analysis. For the difference cost factors, we looked at both fixed and variable costs. For fixed cost, we only looked at marketing expenses. All other contributing factors of fixed expenses required additional investigation and a well-defined business/commercialization plan, which we do not currently possess. For variable costs, we were able to find suitable values and averages to determine expenditure for labor, shipping, and storing.

Since the fixed cost factor is only comprised of marketing expenses, we only accommodated for an estimated marketing cost. One recommendation is to spend 7-8% of gross revenue on marketing [17]. Since our estimate is based on two price points of $50 and $60 for 10,000 units, the gross revenue for each price point would be $500,000 and $600,000, respectively. We consider 8% for marketing expenses, so the amount for marketing would be $40,000 and $48,000 for the respective price points.

Since the variable cost factor in our analysis includes labor, shipping, and storage costs, we will compute each cost and the variable cost will be the sum total of each individual part. We are not experts in determining accurate laboring costs, however, we estimate based on our personal experience that, if provided all the necessary equipment and materials, the time taken to assemble the produce would be at most 0.5 hours per unit. Using the minimum wage of $12 in Massachusetts as the rate for labor, we calculate the product of the labor rate per hour, as well as time to assemble each unit, which gives us $6 per unit for labor [18]. For storage cost, we used an estimate cost per square foot as well as assumed that for units can be stored in one square foot of area (already packaged). The average cost is given as $6.53, which means that the cost of storing one unit is $1.64 (ratio of cost for space and whole units per space) [19].

For shipping the product, we are using the average shipping rate for Amazon shipping, which is $3.70 per unit [20]. Using these determined values as well as the cost to produce one unit ($18.27), the total cost is $30.14 per unit.

Provided from a lecture in class is a free web tool that calculates and analyzes break-even points [21]. Shown in the following figures are the graphs generated based on the inputs of costs, number of units, and price per unit for the two price points. To summarize the results from the analysis, for the price of $50, the breakeven point would be 3200 units and if all 10,000 units sold, the profit would be $198,600. For a price of $60, approximately 1608 units would need to be sold to break even and a profit of $298,600 is made when all 10,000 units are sold.



You will break even at 2,014 units.

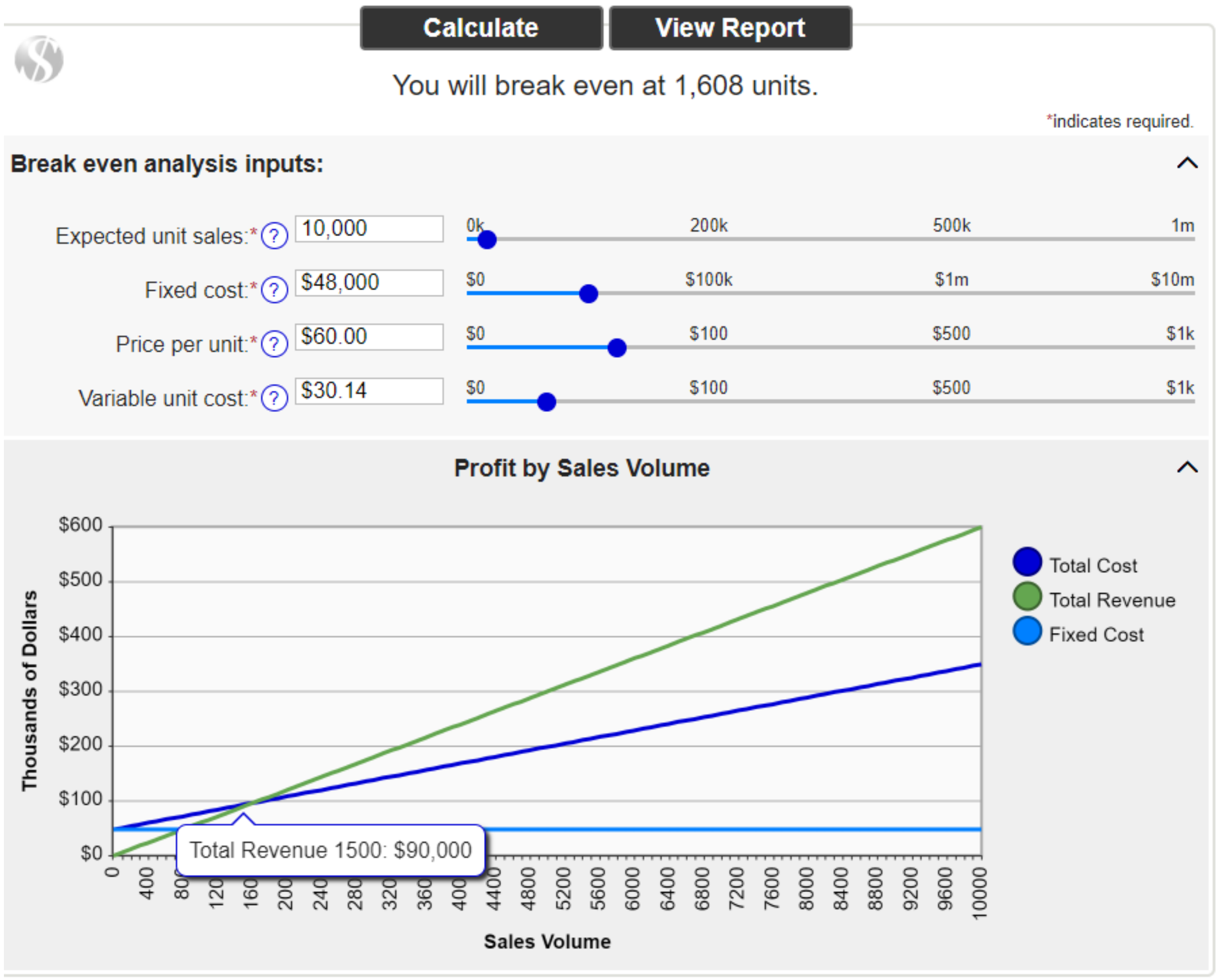
*indicates required.

Break even analysis inputs:

Expected unit sales:*	10,000	0k 200k 500k 1m
Fixed cost:*	$40,000	$0 $100k $1m $10m
Price per unit:*	$50.00	$0 $100 $500 $1k
Variable unit cost:*	$30.14	$0 $100 $500 $1k

Profit by Sales Volume

Figure 30: Analysis of Revenue by Sales Volume ($50 price point)



You will break even at 1,608 units.

*indicates required.

Break even analysis inputs:

Expected unit sales:*	10,000	0k – 200k – 500k – 1m
Fixed cost:*	$48,000	$0 – $100k – $1m – $10m
Price per unit:*	$60.00	$0 – $100 – $500 – $1k
Variable unit cost:*	$30.14	$0 – $100 – $500 – $1k

Figure 31: Analysis of Revenue by Sales Volume ($60 price point)

7 Future Work

Possible future renditions of the SentryEye would include a superior, seamless integration between the red flashing LED and the motion detection logic. Instead of periodic blinking to ward off potential thieves, the light could be introduced to the software so as to only blink when motion is detected. In addition to this feature, it is desirable to add possible LED flash capability for better low-light detection and photo capture capability. Additionally, it would be advantageous to provide Bluetooth tethering capability so that the device is not active when the user’s phone is within 35-50 feet of the vehicle. This addition to the existing design would not only save power but would also drastically reduce the number of false alarms possibly triggered by the owner of the vehicle. This potential feature was considered from an early point in the device’s development, as Bluetooth connectivity was one of the most significant factors when choosing a microcontroller. Because the Raspberry

Pi Zero W already has an integrated Bluetooth chip, the further development of proximity detection when a device is within range would not require any additional hardware to configure and would therefore not impact the cost of parts.

A more intuitive implementation of the solar panel option should be given to the user. As a result of the judges' recommendation, we adhered to the advice that perhaps a product that includes the solar panel separate from the main one would be more suitable for both the consumer and the manufacturer. Further testing of the viability of the solar panel should also be considered for the sake of long-term development. This solar panel would most likely take the form of a dock to which the device could adhere on the consumer's vehicle dashboard.

As it pertains to future work regarding software, there is a lot of space to grow in the SentryEye ecosystem. A feature that should be implemented is family sharing. One could imagine the problems that would arise if a user had more than one SentryEye device in the same environment or there was only one person with access to one sole Sentry device.

Furthermore, it is important for SentryEye to expand from a sole web platform to a mobile platform as well. Given the REST API that was built for the web, wrapping a mobile application around the given services would be fairly straightforward. The return on investment would be immense in regard to building a more robust user experience. Most SentryEye users have access to mobile phones, whether it be Android or iPhone, and those phones are most likely their primary devices. Having the ability to check on your car by opening a native app would build quality assurance with users. Having the ability to receive push notifications instead of emails from a third-party app like Gmail would also build more user trust. Having a mobile app would also create better user experiences with regards to onboarding and future family sharing. Bluetooth family pairing could be implemented given a mobile environment, as well as some form of Bluetooth onboarding. This implementation would cut down the steps it would take to experience SentryEye.

8 Conclusion

To address the issue of on-campus personal safety, our team designed and developed SentryEye to help monitor vehicles and offer car owners a modularized device that can capture pictures of potential threats of vehicular larceny, while also notifying the owner of the threat. Our team was able to create and test a prototype in the limited time and resources that were available. While the current design differs slightly from the original, it maintains most if not all the specified features determined from the customer requirements. In the process of putting this project together, we have learned a great deal on product development and product commercialization, we have encountered challenges and issues, most of which we were able to resolve, and we have identified many opportunities for future improvement and further development for SentryEye. While our prototype may not be quite ready for commercialization, we believe that we have constructed a sufficient prototype that could soon serve as a minimal viable product, with which further product marketing and validation can be conducted.

9 Appendix

9.1 Appendix A: Literature References

1. K. LaMance, "What Are Some Types of Crimes on College Campuses?," 18 April 2018. <https://www.legalmatch.com/law-library/article/crimes-on-college-campuses.html> (28 October 2019)
2. National Center for Education Statistics, "College crime," April 2019. <https://nces.ed.gov/fastfacts/display.asp?id=804> (28 October 2019)
3. B. Ball, "Burglary and motor vehicle theft increase, Public Safety reports," 30 September 2019. <http://www.dailyprincetonian.com/article/2019/10/burglary-and-motor-vehicle-theft-increase-public-safety-reports> (28 October 2019)
4. Worcester Polytechnic Institute, WPI Safety Notification, 2019. <https://www.wpi.edu/about/emergency-management/safety-notifications> (28 October 2019)
5. I. Kowarski, "11 Universities Where Students Rarely Bring Cars," 6 March 2018. <https://www.usnews.com/education/best-colleges/the-short-list-college/articles/2018-03-06/11-national-universities-where-students-rarely-bring-cars> (28 October 2019)
6. *CityProtect*. "Vehicle Theft Data from February 1, 2021 to March 22, 2021." <https://cityprotect.com/map/list/incidents?pageSize=2000&parentIncidentTypeIds=101%2C99&zoomLevel=15&latitude=42.26886471110317&longitude=-71.7971967290135&days=1%2C2%2C3%2C4%2C5%2C6%2C7&startHour=0&endHour=24&timezone=%2B00%3A00&relativeDate=custom&fromDate=2021-02-01T00%3A00%3A00.000Z&toDate=2021-03-22T23%3A59%3A59.999Z> (22 March 2021)
7. Geico, "How To Prevent Your Car From Being Stolen," 2019. <https://www.geico.com/information/safety/auto/preventing-auto-theft/> (28 October 2019)
8. Studentcaffee, "Preventing theft," March 2019. <http://studentcaffe.com/thrive/student-safety/preventing-theft> (28 October 2019)
9. Raspberry Pi Zero Specifications <https://www.raspberrypi.org/products/raspberry-pi-zero/>.
10. Raspberry Pi Zero W Specifications <https://www.raspberrypi.org/products/raspberry-pi-zero-w/>.
11. Raspberry Pi 3 Specifications <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>.
12. Camera Specifications (Standard, Without Infra-red Filter, Wide Angle, 160° Variable Focus)

<https://shop.pimoroni.com/products/raspberry-pi-zero-camera-module?variant=37751082058>.

13. Battery University for battery specifications <https://batteryuniversity.com/learn/archive/whats_the_best_battery>.

14. OwlCam <https://www.amazon.com/Owl-Car-Cam-I-Response/dp/B07J2TS6Q3>.

15. VAVA Dash Cam <https://www.amazon.com/VAVA-Feature-Rich-G-Sensor-Snapshot-Included/dp/B076FVPTLN>.

16. Crosstour Dash Cam <https://www.amazon.com/Crosstour-Dashboard-Recording-Detection-CR300/dp/B078B56BYJ>.

17. Nuphoriq Team, "How Much Should You Be Spending on Marketing?," 2019. <https://nuphoriq.com/create-a-marketing-budget/> (13 December 2019)

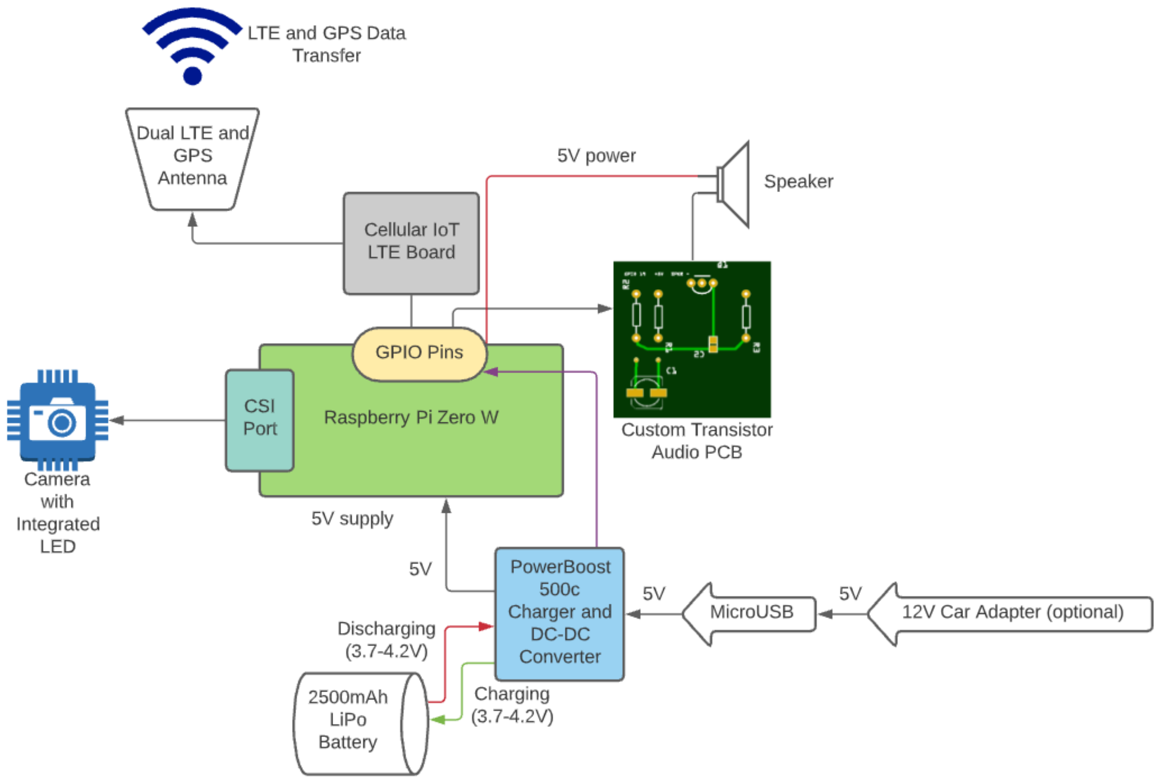
18. N. LaMarco, "How to Calculate Direct Manufacturing Labor Costs," 1 February 2019. <https://smallbusiness.chron.com/calculate-direct-manufacturing-labor-costs-19305.html> (13 December 2019)

19. "Warehousing and Fulfillment 2017 Warehouse Costs and Pricing Survey," 24 August 2017. <https://www.warehousingandfulfillment.com/warehousing-and-fulfillment-resources/warehousing-and-fulfillment-2017-warehouse-costs-and-pricing-survey/> (13 December 2019)

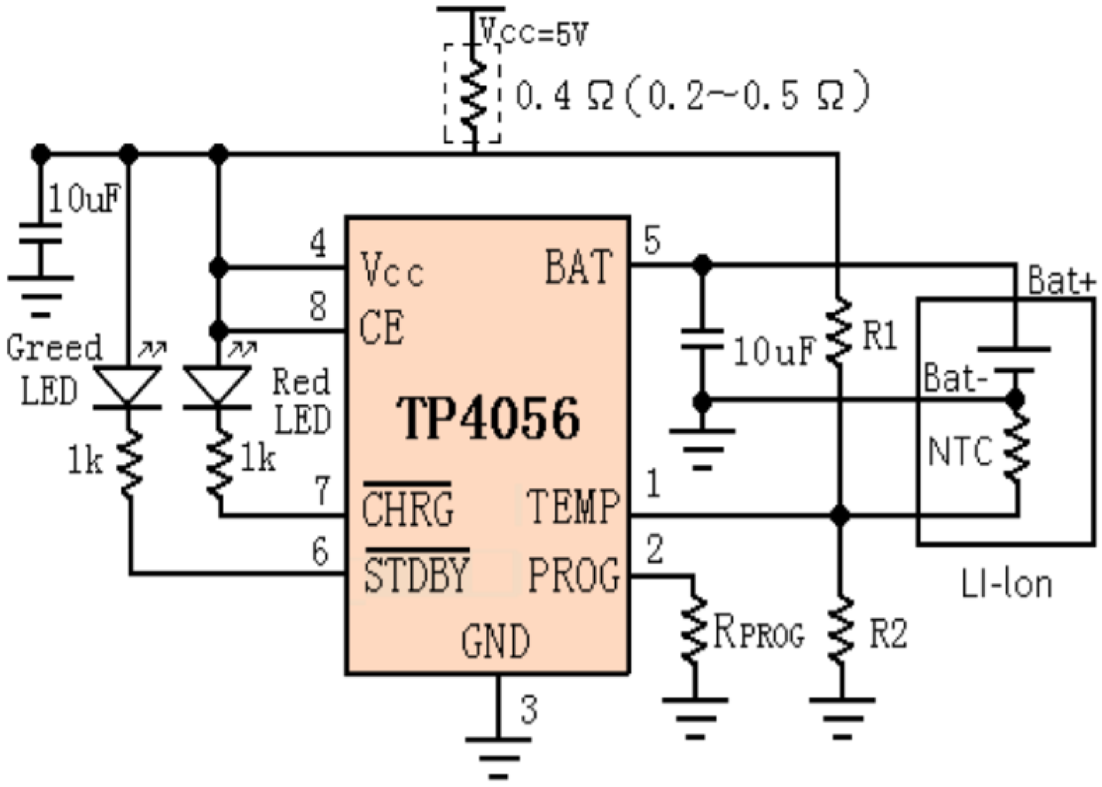
20. "An Amazon puzzle: How many parcels does it ship, how much does it cost, and who delivers what share?," 29 July 2019. <https://www.savethepostoffice.com/an-amazon-puzzle-how-many-parcels-does-it-ship-how-much-does-it-cost-and-who-delivers-what-share/> (13 December 2019)

21. "Breakeven Analysis Calculator," <https://www.nase.org/business-help/calculators/business/breakeven-analysis-calculator>(13 December 2019)

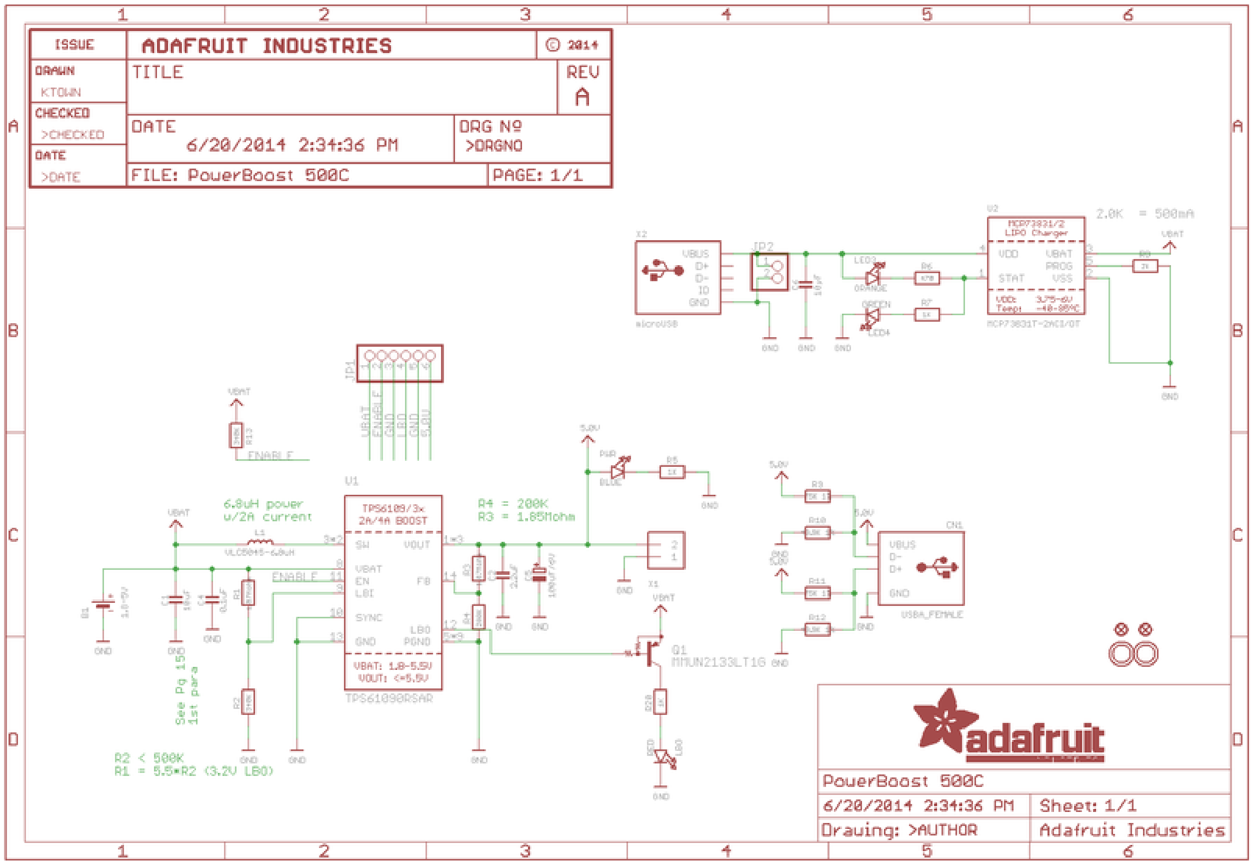
9.2 Appendix B: Top-Level Block Diagram



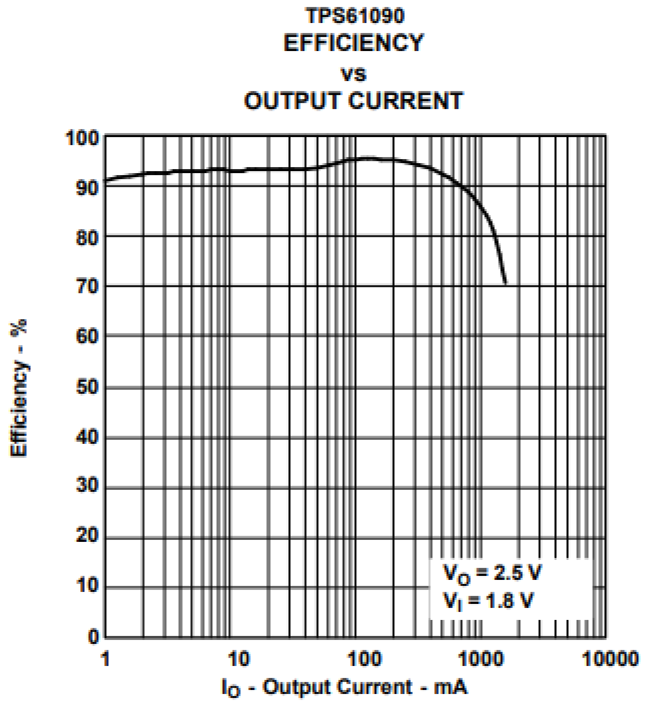
9.3 Appendix C: TP4056 Schematic



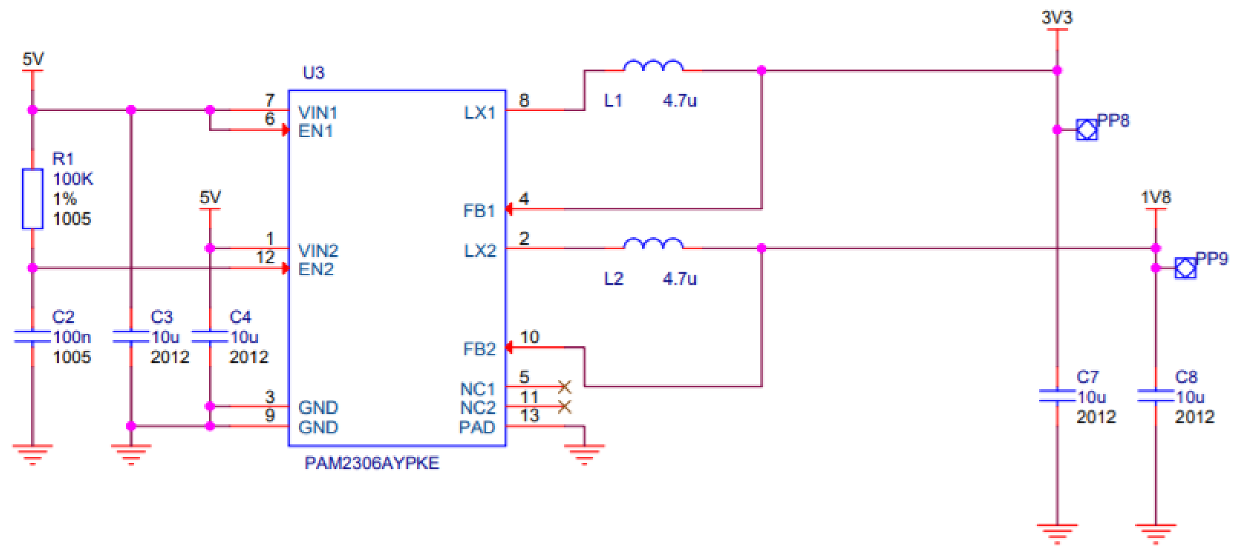
9.4 Appendix D: PowerBoost 500c Schematic



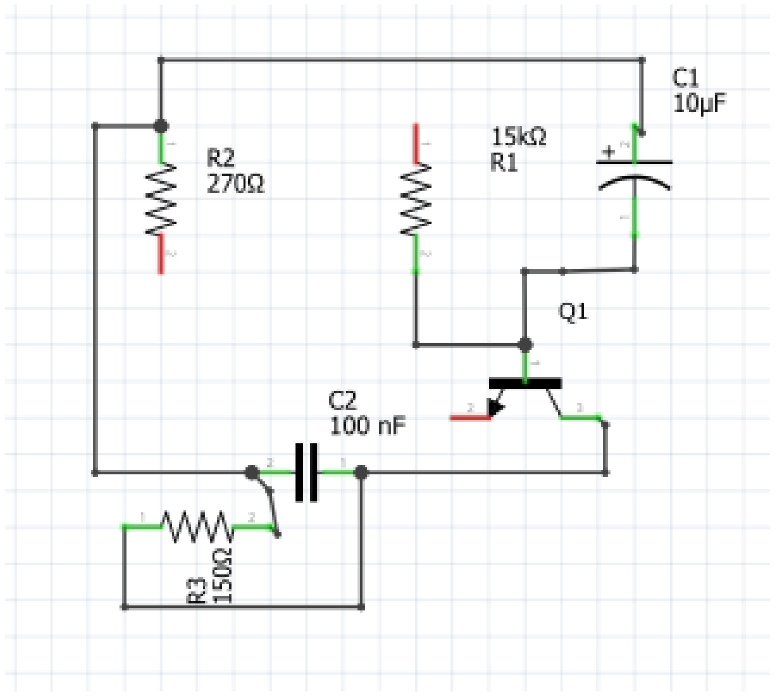
9.5 Appendix E: PowerBoost 500c Efficiency



9.6 Appendix F: Raspberry Pi Zero W Schematic



9.7 Appendix H: Custom Single Transistor Audio Printed Circuit Board Schematic



C1
10μF
R2
270Ω
15kΩ
R1
Q1
C2
100 nF
R3
150Ω