SecureScape

Smart Deployable Security System

Group 15 Authors:

Colin Kirby	Philip Murano	Dylan Myers	Jaxon Topel
Computer Engineering	Electrical Engineering	Electrical Engineering	Computer Engineering

Review Committee/Mentors:

- Dr. Chinwendu Enyioha – UCF ECE Assistant Professor
- Dr. Mike Borowczak – UCF ECE Associate Professor

Table of Contents

List of Tables

List of Figures

Chapter 1 - Executive Summary

Our Senior Design project "SecureScape" a smart deployable security system, aims to address the rising needs for increased security in rural areas with little to no access to the internet. To achieve our goal of developing a full-scale security system, we have designed 2 main hardware components and have split our software components into three main sections: machine learning/software architecture, embedded code, and front-end development. In short, our demo will consist of 3 nodes (our first hardware component) that contain 2 ESP32-Cam MCUs, an active IR sensor, and a passive IR sensor. The passive IR sensor will constantly monitor the environment if motion is detected. Once motion is detected the ESP32 on the respective node will capture an image and send it over HTTP to our Gadget (our second hardware component). The Gadget is a device that must be within the area and is used to interact with our system, receive alerts for any potential threats in the area, and allows additional control over our system to adjust features for the customer's needs. Once the image is sent to the gadget, image processing is done locally on the ESP32-S3 board that lives within the gadget. The algorithm that processes the image will simply determine if there is a threat within the image by deciding whether a person is inside of the image using an object detection algorithm. If our algorithm determines there is a threat in the area, our system will then alert the user via an alarm that will live inside each node and notifications will be sent to the iOS application and the gadget.

Chapter 2 - Project Description

This chapter is meant to provide the reader with a basic understanding of what the project is, and answer both why and how we are doing it. In detail we shall present our planning process, with an emphasis on how every part of our system will interact with each other.

2.1 Project Motivation and Background:

We would like to first present some background knowledge on our group and our project before sharing the motivation behind what we are going to accomplish. For starters we are a group of 2 Computer Engineers and 2 Electrical Engineers. We formed our group in the Spring of 2024 and had originally decided to pursue a drone project. After further discussion, we concluded that we will not be able to gain experience in all the areas we wished to with that project. Tying this into the main motivation behind our current project, we also wanted to be able to get behind an idea that we were all excited about. The areas we were interested in working with for Jaxon Topel were image processing and successfully collecting sensor data to make software computations in a real time system. Colin Kirby wanted some experience with Embedded Systems integration along with front end development. Dylan Myers and Phillip Murano both wanted experience with hardware integration and the development and design of a PCB that will facilitate functionality to multiple peripherals. Our project requires a second PCB with different functionalities, allowing both students to work adjacently with their own board. With all these areas in mind to gain experience, we recognized the need for a portable security system that you could take with you to areas that you wish to feel more secure in. For example, our deployable security system will have various applications for hikers that can set up around their tent. Hunters can use our product to gain more situational awareness even in the dead of night. We hope to provide reliable security for anyone that wishes to remain safe in an unknown territory where you can be sure that you are secure even without phone service.

2.2 Goals and Objectives:

2.2.1 Basic Goals:

Our basic objective would be to successfully develop a deployable security system that will ensure no one can enter or walk around your area without having our alarm system triggered if it is a potential threat. The design requirements for our project consist of developing and integrating multiple hardware and software applications. We will develop 3 different "nodes" that will be retractable sticks in the ground with an IR sensor halfway up the pole, and two ESP32-CAM microcontrollers atop each node that will have a camera built into each of the boards for a total of two cameras per node. Each node will live inside a computer network that will interact with our handmade device we are calling gadget. Our gadget will be fully designed on the hardware side of things by Phillip and Dylan and will serve as the central point of communication in our network, receiving and sending information about our system to/from the nodes. All our devices will be powered from batteries that can be easily replaced or recharged. Requirements for hardware will consist of ensuring all our hardware is compatible together, for further details see below. Our software applications will consist of programming the computer network to send information between nodes and our gadget. An image processing algorithm that will be tripped by the IR sensor that detects movement and recognizes whether the movement outside our secure area is a threat. When the IR sensor detects movement or the image processing algorithm classifies the movement as a threat, communication signals will be sent from the node to the gadget and the node that detected the threat will enter an alarmed state along with the gadget alerting the owner of the threat. The node(s) in an alarmed state will have the capability to sound an alarm, flash an LED, and have the cameras operate in an excited state.

2.2.2 Advanced Goals:

- Incorporate onboard memory for local data storage.
- Implement multiple alarm modes, allowing users to select between quiet and loud alerts.
- Installing a switch to the gadget that will allow the user to manually toggle a specific function within the node(s).
- Configure software to trigger an alarm if any security node is tampered with.
- Capture image button on app that will populate current images from the nodes to the app.

2.2.3 Stretch goals:

- Develop a fully integrated and comprehensive mobile application.
- Enable secure cloud storage for data backup and access.
- Introduce rechargeable peripherals for enhanced convenience.
- Design a control hub that also functions as a charging station for peripherals.
- Create software capable of distinguishing between human and animal threats, automatically adjusting the alarm's threat level accordingly.
- Develop a threat recognition system that can identify weapons, such as knives, and notify users of the corresponding threat level.
- Implement real-time updates of imagery on a dedicated website or app.
- Integrate battery level monitoring to ensure system reliability.

2.3 Project Functionalities:

Our project focuses on developing an advanced portable security system that seamlessly integrates motion detection with real-time alerts. The system is designed to provide continuous monitoring of secure areas, ensuring that users are always informed of any potential security breaches. Key features include a user-friendly interface for easy data integration, along with a versatile control gadget that allows comprehensive management of the entire security system. The features are detailed below:

2.3.1 Motion Detection:

Our system will utilize IR sensors to detect motion within the range and FOV, the detection will process in real time, sharing images to our processing algorithms, and potentially alerting users of threats. Users will receive instant notifications through our mobile app and gadget whenever a threat is detected and will also hear the alarm from the node that detected the threat, informing the use of the location of the threat, giving an advantage to the user in the chance they will have to protect themselves.

2.3.2 Perimeter Security:

We will secure our system's perimeter by placing active IR sensors on each pole that will be set up to detect motion only along the perimeter of our secure area. This will be the 'Achilles Heel' of our system as the nodes will have to be set up in certain locations and lengths away from each other to ensure correct functionality. Users will receive notifications via our mobile app and our gadget we are building, along with all alarms being tripped sounding the highest level of threat.

2.3.3 Access and Control:

Users will be able to access the security system via mobile app or gadget. This will allow users to monitor the property and manage the system from anywhere. We will have to use secure cloud-based communication protocols and authentication methods to ensure safe remote access.

2.3.4 Response Integration:

This system can automatically alert pre-defined contacts in case of a specified level of threat detection. This will allow for a rapid response from the authorities and will enhance the security of our system.

2.3.5 Threat level alert system:

Multiple threat levels cases will be made to categorize threats into pre-defined areas. This feature will allow users to immediately and clearly identify what scenario they are in, enabling the user to take appropriate actions.

2.3.6 Feedback footage of detected motion/threats:

Available through the mobile app, users will be able to view in real time, recorded footage of detected motion/threats. This feature will allow users to respond appropriately to threats from distance or from a secure area.

In developing these features, we referenced input from customer reviews and conducted a marketing analysis of comparable products, such as the DIY home security system offered by Ooma [1], which emphasizes user-friendly installation and real-time threat alerts. Additionally, we considered insights from the best travel safety products, as highlighted by Travel and Leisure [2], to ensure our system offers portability, versatility, and effective perimeter protection.

https://www.ooma.com/home-security/diy-security-system/ [1]

https://www.travelandleisure.com/best-travel-safety-products-6833148 [2]

2.4 Existing Products:

Existing products are a solar outdoor motion sensor alarm. The key differences between this product and ours will be: We will create a network of sensor alarm nodes providing a higher degree of security, and we will also have more advanced image processing algorithms providing a higher level of detail of security. Another key difference is that their product has a full 360-degree field of view while ours will have a 260-degree field of view from each node, this is because we are monitoring the area around the perimeter instead of monitoring from one point from within.

Figure 2.4: Image sourced by HULPPRE [3]

2.5 Specifications:

Specifications	Minimum	Maximum
Detection Accuracy	75%	100%
Detection ranges	5 feet	15-20 feet
Deployment time	5 minutes	10 minutes
Response time	> 5 sec	10 sec
Scalability	50-60%	90- 95%
Installation time	Program Run Time	Program Run Time
Encryption level	128 bits	256 bits
Cloud latency	10-50 ms	50-100 ms

Table 2.5: Specifications for Project metrics.

2.6 Hardware Planner:

Work Items	Name	To be Acquired	Acquired	Investigating	Designing	Prototyping	Completed
Control Hub PCB	Phillip	P			P		
Node PCB	Dylan	D			D		
Control Hub power	Phillip	P		P			
Control Hub housing	Dylan	D			D		
Router	Dylan	D			D		
Esp int.	Phillip		P		P		
Esp cam int	Dylan		D		D		
IR sensor (active)	Dylan		D		D		
IR sensor (passive)	Dylan		D		D		
LED/Alarm	Phillip		P	P			
Node Housing	Phillip	P			P		
Node Power	Phillip	P		P			
System Integration	Dylan	D		D			
Memory Integration	Dylan	D		D			
Gadget Switches	Phillip	P		P			

Table 2.6: Hardware planner distributing tasks amongst members.

2.7 Hardware Diagram:

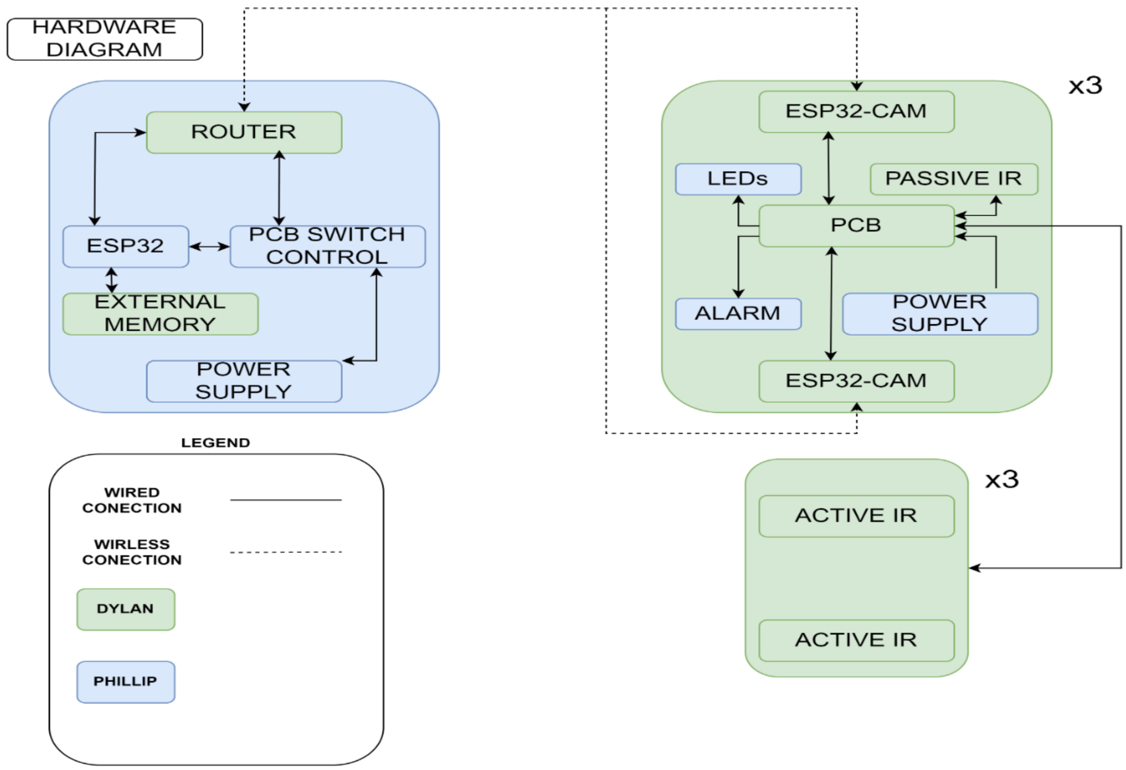


Figure 2.7: Hardware Diagram.

2.7.1 Hardware Overview:

For the hardware of this assignment, we will be designing two unique pieces of hardware. A control hub that is referred to as “gadget”, and then “nodes” that survey the area for movement. We will call the collective functionality of every piece our “system”. The gadget will be in control of housing the resources capable of creating a wireless LAN, the control unit in charge of powering on and off the system, and additional memory to hold imagery data. The main method of communication between nodes and the gadget will be done through a wireless LAN. We have advanced and stretch goals mentioned above specifically calling for more capability of the gadget like having a quite mode or manually turning on the alarm mode. For both Electrical Engineers, we will need to design a PCB board for the gadget that will regulate power and house the switches for system power as well as alarm modes. There will be three nodes built to give an active perimeter. With three nodes we can have the capability to have complete coverage without any blind spots. The nodes will contain all the sensors and alarm hardware. Each node will have its own PCB built and will control the signals to and from the passive and active IRs respectively, power regulation to the node, and power on and off the alarm hardware. Our focus for selecting hardware is to save as much power as possible without neglecting the user's security. Our nodes will all have a low power mode for the peripherals that aren't always on. The devices that are always on are our passive and active IR sensors which will need to have a low power usage. The cameras and other alarm devices will be active once the system triggers into an alarmed state. This endeavor will be difficult to achieve with the project's time limit, but with efficient time management we should achieve this goal.

2.8 Software Diagram:

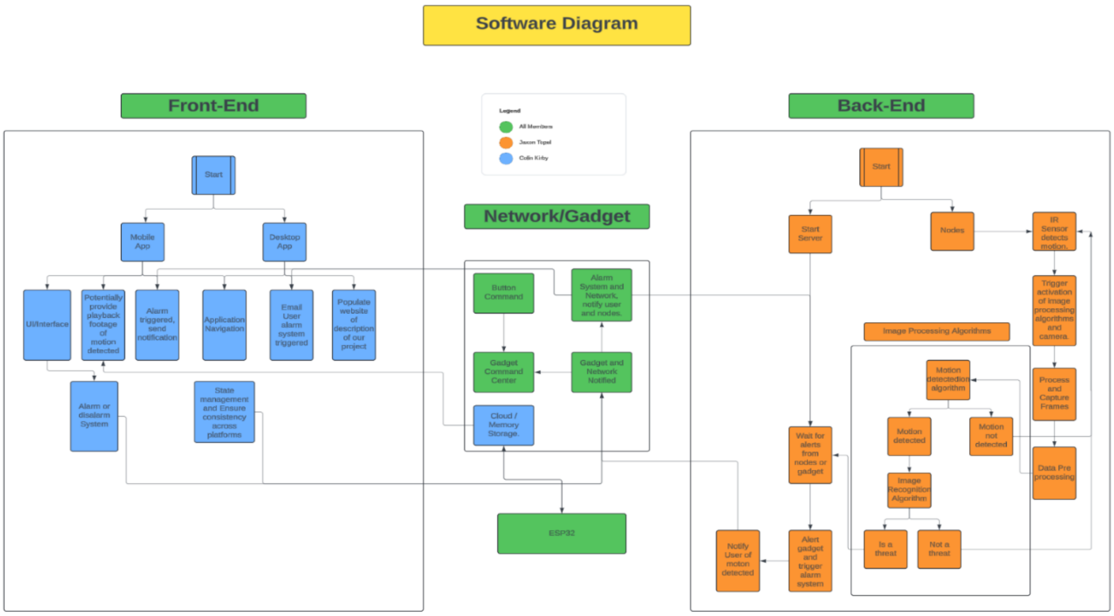


Figure 2.8: Software diagram created using Lucid Flowchart.

2.8.1 Original Software Components Overview:

Our project involves creating a sophisticated software environment. Jaxon will take the lead on back-end development, focusing on image processing, software architecture, and developing the computer network to send information between nodes and gadgets. Kirby will oversee the development of both mobile and desktop applications for the front-end, focusing on enhancing the control our mobile applications will have on the functionality of our system. Both team members will collaborate closely as part of the software team to build a robust computer network system, ensuring seamless data communication between nodes and users.

Below is an overview of our initial software development plans. Please note that file names, concepts, and purposes are subject to change as the project evolves.

Main.py:

From the very start of our software (main.py) we are importing from many of the files that our database will contain. We will create the network/server that our nodes and our gadget are going to use to communicate with each other. After this we will instantiate the 3 nodes for our system, instantiate the gadget inside our network, and then add all 3 nodes to our network as well. After this, we are going to have the software on each esp32 node running at all times, therefore, we have a continuous loop that will monitor the network, look for alerts from the gadget (details discussed below), and then will monitor the environment of each node (also discussed in more detail below). From this central point of the software, we will accomplish the goals of a secure, abstract, and scalable system that will allow us to develop easily throughout the next two semesters.

Gadget.py:

This file will represent the gadget inside our security system and provide all functionalities desired from it. Inside we will have the constructor that will assign the correct network to what our gadget will live in. Another function will be for the gadget to check for alerts. Inside we will call a function from network that will receive alerts inside of an array. Then we will go through the alerts and trigger the alarm for any valid alerts. Lastly, we will have the alarm trigger function. Ideas are still being discussed as to how we wish to trigger the alarm, but as of right now we plan to have some sort of loud alarm that comes from the node triggered that will allow the user to easily identify the location of the threat while also potentially fending off the threat.

Network.py:

This is the file where all nodes will communicate from. We will have an initial constructor that will set the nodes and alerts as an array of values, initially empty. From there we will have an add gadget function (Still debating if this is going to be necessary), an *'add_node'* function, a function that will send alerts, and lastly a function that will receive alerts.

Image_processing.py

This file will be where we use image processing algorithms to determine if the motion detected is a threat. Called from node.py, we will be using OpenCV and potentially many other libraries to implement many algorithms for image processing such as image recognition, threat level detection, and potentially more advanced algorithms that will give more detail of the threat with more accuracy. This is where the pride and joy of our software will live and most of the work done here will be new for both Jaxon and Kirby. Research is already being done for methods of interacting with our esp32 and obtaining the images and getting them into a consistent form for feature extraction and pre-processing our images. The general framework we have already consists of our initialization function which will recognize our esp32. A function that will get the images from the esp32, it has not yet been finalized as to how we wish to obtain the images, either directly from the hardware or from a web server that our esp32 will live stream to as the esp32 supports that feature. After this we will have a '*process_image'* function (the function called from node.py), here we will load the image, apply some image pre-processing techniques, and then determine if the image contains a threat or not for the image recognition algorithm. One method discovered by our software team is the potential use of a powerful library and tool called yoloV5. YoloV5 is a deep learning-based object detection model that is widely used due to its speed and accuracy for detecting images in real time systems. Further research is needed, however, I have provided this, so you have a general idea of how we wish to implement this model.

Node.py:

This file serves as the representation of our nodes in our security system. Instantiated 3 times in main.py, our constructor when called will assign the node id, location, network, and then instantiate an image processor inside each node. Eventually once we get access to the hardware, we will have to assign an esp32 to each node. After the constructor, we have a function that will monitor the environment. Inside of here we will use our IR sensor to monitor for movement outside the secure area. When movement is detected, we would call the image processor algorithm to identify if the movement detected was a threat. If the movement detected was a threat, then we will send an alert to the network. Lastly, the next function is the one used above that sends an alert to the network.

Ir_sensor.py:

This file will be used to detect movement of the IR sensor. No code architecture has been laid out for this file; however, I anticipate that we will need at least 2 motion detection algorithms. One for monitoring the environment outside of the secure system, and another for ensuring nothing goes in or out of the perimeter of the secure area without us knowing.

Note:

Remember that more files may be created, and the code's architecture may change as needed to ensure our project's success.

2.8.2 Revised Software Components Overview:

Since software development began, many changes have happened all across the board and a complete restructuring of our software architecture had to take place as further research was done to ensure that we could meet the requirements of our project. Section 2.8.1 was kept in this document to serve as a showcase of progress that has been made since the original D&C submission.

Our new software architecture is simpler, consisting of backend code split into 2 main sections, ESP32-S3 Code (Gadget code) and ESP32-CAM Code (Node code). See below for respective sections. Keep in mind that as development of software continues in order to satisfy the objectives of the project, code restructuring may happen and will be noted in further documentation.

2.8.2.1 ESP32-S3 Code:

The code that will live on the gadget will be split into two main files. Main.ino and ObjectDetection.ino. Main.ino serves many different functionalities such as connecting the gadget to the router, monitoring the server for any images sent by a node, and interacting with the hardware that lives on the gadget, and the respective desired functionality for the hardware to perform. Inside of a function that is called once an image has been sent from a node to the server, the gadget receives the image, stores it respectively in PSRAM, and then calls a function inside ObjectDetection.ino. The function called ObjectDetection.ino simply returns a Boolean that will indicate whether a person has been detected inside of the image sent from a node. Inside ObjectDetection.ino, we will be using TensorFlow lite, which is described in section 3.2.3.2. ObjectDetection.ino interacts with many libraries to perform the image processing libraries and is the part of software that requires the most computational power.

2.8.2.2 ESP32-CAM Code:

Inside the ESP32-CAM, we will have 1 main.ino file running constantly. This code's main purpose is to interact with the hardware on the nodes we have in our system. Main.ino monitors the environment constantly in a low power mode state, implemented by a “sleep” function that turns off hardware that consumes the most power, ensuring that we can keep our system on as long as possible. Inside of this code, we connect each node to the router via http, monitor the environment by reading inside our loop function for a voltage change with our passiveIR sensor. If the passive IR sensor detects motion, we call a function that turns the camera on, captures an image, and sends it over the server that will immediately be taken care of by main.ino on the gadget. Other functions we have in this code interact with the active IR sensor, constantly monitoring this as well, and in the case that the sensor has detected motion, we will send an alert over the network and our gadget will sound an alarm.

2.9 Prototype Illustration:

The following illustration was created by us to show how the process of our system works. Our design idea consists of a system that lives inside of a computer network, we also will have infrared sensors as an additional measure of security, lastly along with a central gadget that will serve as the governing point of operations for the user.

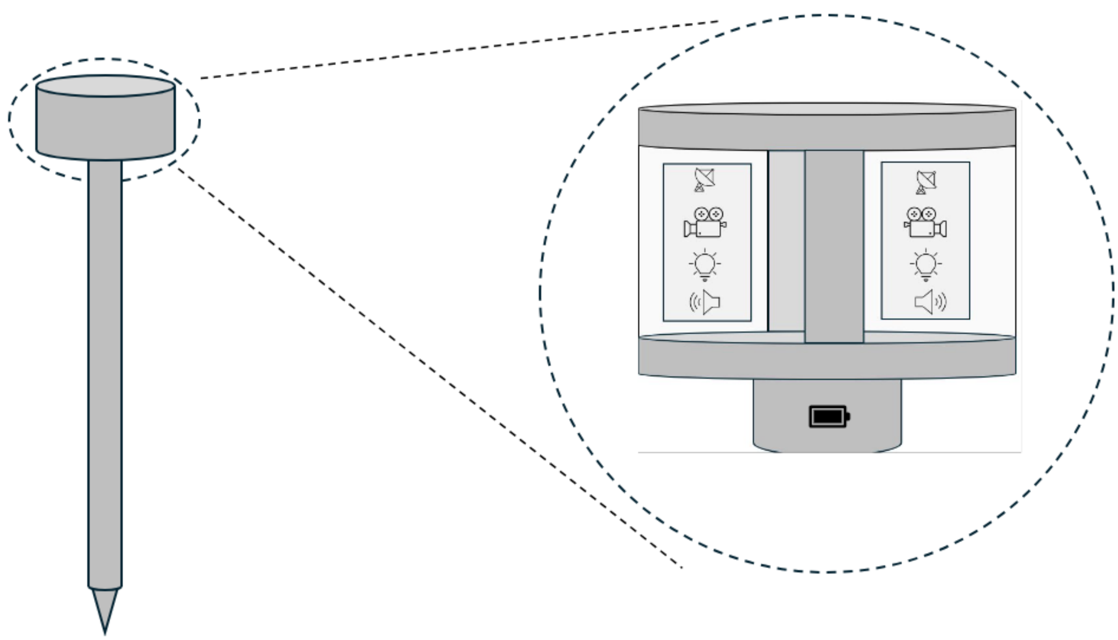


Figure 2.9.1: Prototype illustration of a single node.

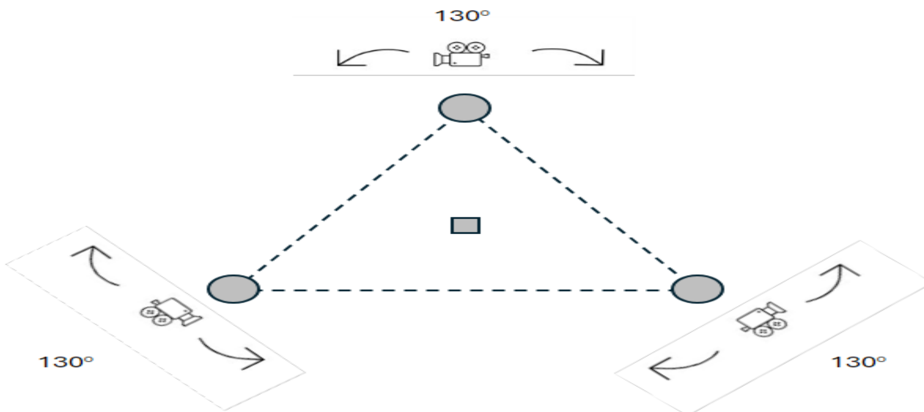


Figure 2.9.2: Overall system prototype including 3 nodes and a center gadget.

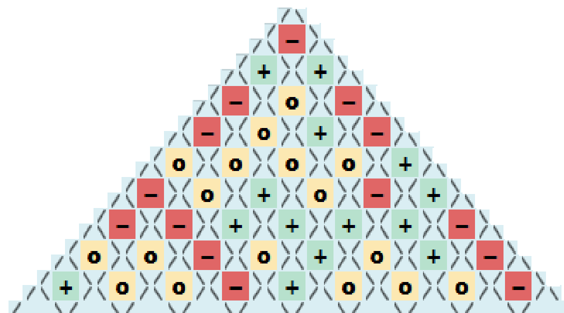
2.10 House of Quality:

The House of Quality shows the correlations and relationships of desirable features of our project. Following the legend will guide you to a better understanding. This figure shall guide our project to make a more educated decision when comparing tradeoffs.

Correlations	
Positive	+
Negative	–
No Correlation	o

Relationships	
Strong Positive	↑↑
Weak Positive	↑
No Relation	⊗
Weak Negative	↓
Strong Negative	↓↓

Direction of Improvement	
Maximize	▲
Minimize	▼



Column #		1	2	3	4	5	6	7	8	9	10
Functional Requirements / Customer Requirements (Explicit and Implicit)	Direction of Desirability	Size	Weight	Response Time	Communication Range	Battery Efficiency	Fault Tolerance	Signal Integrity	Weather Resistance	Latency	Cost
Direction of Desirability		▼	▼	▼	▲	▲	▲	▲	▲	▼	▼
Affordability	▲	↓↓	↓↓	↓	↓	↓↓	↓	↓	↓↓	↓↓	↑↑
Accuracy	▲	⊗	⊗	↑↑	⊗	↑↑	↑↑	↑↑	⊗	↑↑	↓↓
Battery Life	▲	↑	↑	⊗	⊗	↑↑	⊗	↑	↑	⊗	↓↓
Environmental Adaptability	▲	↑	↑	⊗	↑↑	↑↑	↑↑	↑↑	↑↑	⊗	↓
Portability	▲	↓↓	↓↓	↓	↑↑	↑↑	⊗	↑↑	↑↑	↑	↓
User Experience	▲	↓	↓	↑↑	↑↑	↑↑	↑	↑	↑↑	↓↓	↓↓
Range	▲	⊗	⊗	↓↓	↑↑	⊗	↑	↑↑	↑↑	↑	↓↓
Set Up Difficulty	▼	↓	↓	↑↑	↑	↑	↑↑	↑↑	⊗	↑	↓↓
Reliability	▲	↑	↑	↑↑	↑↑	↑↑	↑	↑	↑↑	↑↑	↓↓
Connection Time	▼	⊗	⊗	↑↑	↑↑	⊗	↑	↑↑	⊗	↑↑	↑↑
Product Targets		~ 6 in^3	~5lbs.	> 1 s	< 20 ft	1 - 2 Months Per Charge	≤ 5% fail	≥ 95%	IP65	≤ 100 ms	~$300

Figure 2.10: House of quality.

Chapter 3 Research and Investigation

3.1 Research Overview:

Research was performed for our project in 3 main areas: Hardware selection and comparison, Machine Learning Algorithms and software architecture, and then front-end software. Each member oversaw their own respective sections (Dylan and Phillip covered hardware, Jaxon covered ML and software architecture, and Kirby covered Front-end integration). Each member contributed roughly 15 pages for this chapter and performed extensive research into their area, ensuring that all technology and parts were able to meet the restrictions of the assignment while meeting our main goals.

3.2 Technology Comparison and Selection:

3.2.1 Introduction:

When selecting software technologies and parts there are countless options available on the internet that allow developers to not so easily find exactly what they need to fit their needs. Selecting software that will run and be compatible with all hardware restrictions requires us to look for solutions that will maximize performance, minimize power consumption, and allow for successful integration with all hardware. Deciding which technologies to use inside of our software system design consisted of choosing between different programming languages, image processing libraries, communication protocols, Development environments, and data storage solutions.

3.2.2 Programming Languages:

Technology	Performance	Control	Complexity	Memory usage
C/C++	High efficiency. Ideal for low level hw interactions.	Provides almost complete control over resources,	Highest complexity out of all languages seen.	Requires manual memory management
Python	Slower execution compared to C++/C	Less control over hardware compared to C++	Simple syntax, easy to learn language.	Higher memory consumption.
Arduino C++	Slightly less efficient than C/C++.	More control over your program than python, less than C/C++.	Simplified C++ syntax, easier for beginners to pick up.	Less memory consumption than python, more than C++.
JavaScript	Slower execution compared to others.	Good for error handling, less control than C++, no hw interaction.	High level language, easy to debug.	Higher memory consumption.

Table 3.2.2: Programming language comparison table.

3.2.2.1 C++:

[5] "C++ was designed with Systems programming and embedded, resource-constrained software and large system in mind, with performance, efficiency, and flexibility of use as its design highlights."

Given this quote above from Wikipedia, you can now understand why using the language is so desirable for a project such as ours that requires our software to be very particular with memory management and the architecture of how our code will run. C++ gives us that advantage that other languages don't offer. Another huge reason we will be using C++ is due to the direct interaction of our software to GPIO pins that will be used to monitor our Active IR, Passive IR sensors, and more.

3.2.2.2 Python:

[6] "Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library."

Python is the easiest programming language any developer can use. From doing simple algebraic equations all the way to writing deep neural networks, python provides an extensive choice of libraries that make developers lives insanely easy. Although we will not be throwing python code onto the ESP32 for many reasons, we will be using python and its extensive resources for testing out the code we have. For example, once I began development with the ESP32-CAM board that goes onto our node, I wrote a simple python script that would capture the image and display it to my computer to ensure that the hardware was working.

3.2.2.3 Arduino C++:

[7] "The microcontrollers can be programmed using the C and C++ programming languages (Embedded C), using a standard API which is also known as the **Arduino Programming Language**, inspired by the Processing language and used with a modified version of the Processing IDE."

Arduino C++ is almost the same as C++. The primary difference lies in the abstractions and libraries provided by Arduino. Such Abstractions might simplify common tasks such as what our goal of object detection is. Libraries can provide functionality through providing easy to interact with functions that allow developers to avoid manipulating hardware. There are thousands of little, tiny advantages that Arduino C++ software provides and can make the lives of embedded system developers easier.

3.2.2.4 JavaScript:

[8] "JavaScript is a high-level, often just-in-time compiled language the has dynamic typing, prototype-based object-orientation, and first-class functions". Expanding off the quote from Wikipedia, according to the same source [5], 99% of websites use JavaScript on the client side and webpage behavior.

JavaScript is an easy-to-use language that allows front-end developers to execute their client code and display as they wish. JavaScript's most popular use is web development, specifically creating dynamic and interactive front-end interfaces, as most interfaces you interact with today are built on JavaScript. While JavaScript is primarily used for the front-end side of development, there are still major applications for JavaScript with backend applications through environments such as with Node.js, which allows developers to use JavaScript for server-side programming as well. It allows us to use JavaScript throughout the stack, which reduces the complexity of development by reducing context switching between languages. However, because JavaScript's backend applications are not as suitable for our project's current backend applications with embedded programming it may not be ideal for embedded programming.

3.2.2.5 Overview:

Out of these programming languages, requirements for our selection consisted of a programming language that requires an easy-to-use platform, while also maximizing performance, and minimizing memory usage. We will be using both python and Arduino C++. Python was chosen for its substantial amount of image processing tools available online, allowing for easier development with our image processing algorithms on our main gadget board where power consumption and memory usage is less of a worry due to the hardware selections made. We will not be putting any python code on the board; however, we will be using python for testing, especially in our early development stages as we have already been used this language to ensure our camera functionality is working by displaying images easily to the computer screen using built in functions. We will also be using python to implement our image processing ideas before we spend the tedious time working in C++ code to ensure our functionality will work as planned.

We will be using Arduino C++ for our embedded code selection due to the simplicity of the language combined with the control and efficiency it provides. C++ has many open libraries available for use with our project and will allow for the best implementation on both our Gadget and Node ESP32 boards. We will be using this language mainly due to its highly efficient coding capabilities, offering developers direct access to our board, specifically registers and cameras, allowing us to interact directly with the hardware. Another advantage of using this language is that we will have complete control of the memory management inside of our system, allowing us to free memory for flash or ram as deemed necessary. We will also be taking advantage of the object-oriented programming features this language offers, ensuring we will be abstracting as much as possible to implement easily readable code that will make debugging much easier. Overall working with C++ is going to give us a severe advantage of the control we will have over our system, allowing us to save memory, power, and time while our system is running.

3.2.3 Image Processing Libraries:

Technology	Performance	Power consumption	Memory usage	Programming Language Integration
OpenCV	Good for image processing, not optimized for deep learning.	Moderate to high.	High memory usage.	Python, C++
TensorFlow Lite	Optimized for edge devices.	Low to moderate.	Moderate, can be optimized for low memory usage.	Python, C++
YoloV5	High performance for real time object detection	Moderate to high.	High memory usage.	Python. C++
ESP-WHO	Optimized for ESP32, good for face detection, needs optimization for person detection.	Designed for low power applications.	Low to moderate memory usage, optimized for esp32's memory.	C/C++

Table 3.2.3 Image processing libraries comparison table.

3.2.3.1 OpenCV:

[9] "OpenCV (Open-Source Computer Vision Library) is a library of programming functions mainly for real-time computer vision."

OpenCV was originally created by intel in 1999, and this was my first idea of an approach to our object detection goal. Implementation had begun and then was paused as more research was being done, hence we are now using TensorFlow Lite. The main reason we did not pursue an implementation with OpenCV was because we originally started our code architecture using python but faced difficulties finding a way to upload python code with the way we wanted our system to work. OpenCV is one of, if not the most powerful modern computer vision libraries today. It offers help for smaller machine learning applications such as linear regression, all the way to novel areas at the forefront of technology like deep neural networks and reinforcement learning.

3.2.3.2 TensorFlow Lite:

[10] "TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks."

TensorFlow Lite is the machine learning framework that we have chosen to implement our object detection algorithm with. There are many advantages for us using this library to accomplish our goals, however the main one is due to TensorFlow Lites open-source ESP-32 implementations that will allow us to easily develop algorithms to complete our project. The other libraries mentioned, although they can be developed on the ESP-32, would either require too much computational power or not accomplish our desired task.

3.2.3.3 YoloV5:

YoloV5 is a product made by Ultralytics, a company founded by graduates at Washington University. This product is one of many releases that fall under the common name Yolo (you only live once). YoloV5 was compared specifically, not because it is the best out of all the models or the most accurate, but because of the extensive open-source implementations using this version of the Yolo model. In the end, this product was not selected for implementation because it would require too much power and memory for our ESP-32 board that is quite restricted in both of those areas.

3.2.3.4 ESP-WHO:

ESP-WHO is an Internet of Things (IOT) application framework, that offers extensive API's available to developers for the area of Face recognition. It is possible to develop on the ESP-32 board, however, research was cut short for this product soon after realizing that its main applications would not meet the requirements of our project goal.

3.2.3.5 Overview:

Selecting an image processing library is key to the success of our project. With the above options there are many tradeoffs that factored into our decision such as performance, Dev board compatibility, and the main metric considered was power consumption. Our ESP32 boards are not the best for high performing computations so something like YOLOv5 that would make my life a lot easier is not possible as we just do not have the power and even memory to store what we would need to. Options such as ESP-WHO and OpenCV were also considered given their features such as board compatibility, but did still not meet all the requirements we would need to perform object detection. Specifically, ESP-WHO while being made for our board, is not great for object detection and mainly used for image processing for tasks like face detection. That being said, we decided to pursue development as of now with TensorFlow lite and potentially using features of OpenCV, specifically CV2 for its image pre-processing features. This approach will allow us to meet memory requirements of our board and uses low computational power, all while allowing us to meet the main objective of the image processing algorithms which is to perform object detection identifying people and potentially more. Overall, our implementation of using both TensorFlow lite and OpenCV will allow us to accomplish our goals for the IP algorithm while meeting hardware constraints from our ESP32 board.

3.2.4 Communication Protocols:

Technology	Performance	Power Consumption	Memory usage	Board integration
HTTP	Higher latency	Moderate, short-lived connections.	Low-Moderate, data dependent	Easy-Moderate, many available libraries
WebSocket	Low latency	Higher than http	Higher than http	Harder than http

Table 3.2.4: Communication Protocols comparison table.

3.2.4.1 HTTP:

[11] "HTTP (Hypertext Transfer Protocol) is an application layer protocol in the Internet protocol suite model for distributed, collaborative, hypermedia information systems."

HTTP is used in our everyday lives when searching through the internet. There is a more secure option known as https which allows user to be more confident in the website they will be visiting. For our implementation, we simply are using this because it is easily done in C++ on the ESP-32.

3.2.4.2 WebSocket:

[12] "WebSocket is a computer communications protocol, providing a simultaneous two-way communication channel over a single Transmission Control Protocol (TCP) connection."

In comparison to HTTP, WebSocket communication is also able to be implemented on the ESP-32 and would provide lower latency and more efficient communication. This implementation was not selected as it is harder to implement than HTTP with less APIs for the ESP-32, and as a group we decided our time was spent better elsewhere since HTTP could do the job needed.

3.2.4.3 Overview:

When choosing a good communication method for our system, the two options seen above, HTTP, and WebSocket, were compared. Inside of our ESP32 system, we will be taking advantage of library functions using HTTP that allow for fast and efficient communication between our nodes and the Gadget. This implementation will use little memory and is the best option for our system in terms of power consumption as well. We will be connecting all nodes and our gadget to the router through Wi-Fi allowing easy setup. Overall, taking advantage of the functionality of the board will allow us to easily develop any means of communication needed for our system through the use of HTTP Wi-Fi server offered to developers.

3.2.5 Development Environments:

Technology	Ease of use	ESP32 Compatibility	Debugging capabilities
Arduino IDE	User-friendly, easy setup	Good, many libraries available	Serial print debugging
Visual Studio Code	Moderate, requires Platform Io setup	Good, official ESP extensions like platform IO	Advanced debugging, hardware debugging and breakpoints.
Eclipse IDE	More complex setup	Good, official ESP plugins	Advanced

Table 3.2.5: Development Environments comparison table.

3.2.5.1 Overview:

Between the development environments mentioned in the table above, choosing which one to use was fairly easy. In the early stages of development, we took the initial approach of using Platform IO with Visual Studio Code to work with. Quickly after we realized that using Arduino IDE would be much easier for all our developers to work with, so this is what we have chosen to use to upload code to our board. Visual studio code is being used to upload code to GitHub easily if you are working out of the same directory in Arduino IDE as the GitHub repository is set up with. Overall, we are using both Visual studio code for version control and sharing code and we are using Arduino IDE for code development with our board which allows for easy integration and testing of the software.

3.2.6 GitHub:

For our project we are using GitHub. GitHub is a developer git-based platform that allows for any developers/engineers to efficiently collaborate together and share, store, and manage their code. Our group will mainly be taking advantage of the ability to share code efficiently and ensure our engineers are working on the most up to date version possible. See our GitHub link: GitHub Link - SecureScape

3.2.7 Mobile App Frameworks:

Technology	Performance	Ease of Use	Cross-Platform Compatibility	UI/UX Flexibility	Memory Usage
Flutter	High, near-native performance	Moderate	Excellent (iOS, Android)	High, customizable UI	Moderate
React Native	High, but slower than Flutter	High, JavaScript-based	Excellent (iOS, Android)	Moderate	Moderate
Ionic	Moderate, uses web tech (HTML, CSS, JS)	High, web-based	Excellent (iOS, Android, PWA)	Moderate	High
Xamarin	High, near native	Moderate, requires C#	Excellent (iOS, Android, Windows)	High, but complex to implement	Moderate

Table 3.2.7: Mobile App frameworks comparison table.

3.2.7.1 Flutter:

[13] “Flutter's control of its rendering pipeline simplifies multi-platform support as identical UI code can be used for all target platforms.”

Flutter is one of the most popular mobile development frameworks on the market right now, specifically known for its near-native performance across all platforms. It has control over its rendering pipeline which simplifies the multi-platform support by allowing us to use the same UI code for multiple platforms. As our desired goal is a website and mobile app working together it stands out to us considering its high efficiency and high-quality UIs. As noted, Flutter’s ability to streamline the development process by eliminating the need for context switching between separate codebases for different platforms significantly reduces both time and complexity. [13]

3.2.7.2 React Native:

[14] “Reusable components are a key advantage of React Native. In addition to cost efficiency, reusing large amounts of code across platforms allows for shorter development time and a simplified development process overall.”

React Native allows users to reuse components of code across platforms which are similar to Flutter and helps improve development speed and efficiency. Furthermore, the framework uses JavaScript which makes it much more approachable for most UI developers considering JavaScript’s popularity. Although it does not perform at the same level as Flutter in some areas, the flexibility and ease of use makes it a preferred option for a faster-paced cross-platform development process. [14]

3.2.7.3 Ionic:

[15] “It provides a set of pre-designed UI components and tools for building high-quality, interactive applications.”

Ionic is a versatile framework designed to facilitate the development of cross platform apps using the most popular web technologies like HTML, CSS, and JavaScript. A major advantage of Ionic is the set of pre-designed UI components and tools, which allows us to create interactive applications quickly. However, because it is not a native framework its performance is limited, yet it remains a solid option for teams focused on rapid development with wide compatibility. [15]

3.2.7.4 Xamarin:

[16] “With a C#-shared codebase, developers can use Xamarin tools to write native Android, iOS, and Windows apps with native user interfaces and share code across multiple platforms, including Windows, macOS, and Linux.”

Xamarin provides native-like experience for Android, iOS, and Windows applications by leveraging a shared codebase. Considering it was created and fully backend by Microsoft it is most likely going to improve much more and adheres to the needs of the developers. Being able to share code across these platforms is a key component while still not sacrificing the native functionality. While Xamarin can produce high-performance apps, its complexity and reliance on C# may require more development effort on our side because of lack of experience in comparison to React Native or Flutter.

3.2.7.5 Overview:

In evaluating these frameworks, Flutter was selected for its ability to deliver high performance, near native capabilities, and support for complex UI designs. Out of the frameworks mentioned above, it provides the best balance of speed, flexibility and ease of use for our project, which involves real-time image processing. React Native was the original idea solely because of its ease of use, but following this research, Flutter’s extensive support and robust performance makes it the optimal choice for our app.

3.2.8 Mobile App Hosting Comparison:

Technology	Performance	Cost	Scalability	Ease of Use
AWS Lambda	High, serverless and event-driven	Pay-per-use (cost-efficient for small workloads)	Automatic, scales with demand	Moderate, requires knowledge of serverless architecture
Firebase	Moderate, real-time database optimized	Free for limited usage, pay-as-you-go	High, auto-scaling	High, easy integration with mobile apps and real-time databases
Heroku	Moderate, simplified deployment	Free tier, paid plans for higher usage	Good, but limited compared to serverless platforms	High, very easy to set up and deploy
DigitalOcean	High, requires manual server setup	Low-cost, starts at $5/month	High, manual scaling options	Moderate, more complex than Heroku and Firebase

Table 3.2.8: Mobile App hosting comparison table.

3.2.8.1 AWS Lambda:

[17] "AWS lambda are server-less compute functions are fully managed by the AWS where developers can run their code without worrying about servers. AWS lambda functions will allow you to run the code without provisioning or managing servers."

AWS Lambda is a serverless service that would allow us to run code without needing to manage any servers. This serverless architecture is very beneficial for mainly event-driven applications, such as those that scale on demand. Flexibility and cost-effectiveness make it ideal for projects with fluctuating workloads.

3.2.8.2 Firebase:

[18] "It helps developers to build their apps faster and in a more secure way. No programming is required on the firebase side, which makes it easy to use its features more efficiently. It provides services to android, iOS, web, and unity. It provides cloud storage. It uses NoSQL for the database for the storage of data."

Firebase provides a real-time database along with other backend services which streamline the development process for mobile applications. With seamless integration across Android, iOS, and web platforms, and its NoSQL database, it has very high scalability and makes it adaptable for various applications. Considering its quick set up and scalability, it has become a very strong option for our application.

3.2.8.3 Heroku:

[19] “Heroku is a cloud platform as a service supporting several programming languages where a user can deploy, manage and scale their applications. It is widely used to deploy server-based web applications, APIs, discord bots, and more.”

Heroku is a Platform-as-a-Service solution, this may not be as applicable for us considering we want more flexibility and scalability with our project. However, it would allow us to simplify the app development process by pushing code changes without worrying about infrastructure management. Furthermore, its ease of set up makes it suitable for smaller projects or prototype testing.

3.2.8.4 DigitalOcean:

[20] “DigitalOcean provides developers, startups, and SMBs with cloud infrastructure-as-a-service platforms.”

Digital Ocean provides developers with cloud infrastructure services, this allows for more control over server set up in comparison to AWS Lambda and Firebase. It does have a much more manual configuration, so ease of use is limited, yet its low cost and flexibility is very appealing for smaller business applications.

3.2.8.5 Overview:

After evaluating the hosting options, AWS Lambda was selected for its serverless scalable architecture. This aligns with our projects' needs to handle varying workloads efficiently. Considering its a pay-per-use model too, we can save money while reducing the potential burden fo server management, allowing us to focus on app development rather than backend infrastructure.

3.2.9 Frontend Frameworks for User Interactions:

Technology	Ease of Use	Performance	Learning Curve	Compatibility with HTML / CSS
jQuery	Easy	Moderate	Low	High
Alpine.js	Easy	High	Low	High
Stimulus.js	Easy	High	Low	High

Table 3.2.9: Frontend Frameworks for user interactions comparison table.

3.2.9.1 Overview:

Prioritizing performance and ease of use, Alpine.js stood out, considering its lightweight nature, high performance, and our members' previous experience, we believe it is the best choice for us. Its integration with HTML and CSS, and a low learning curve it offers the balance for creative interactive UIs for our project.

3.2.10 UI Component Libraries:

Technology	Ease of Use	Customizability	Performance	Library Size
Material UI	High	High	High	Large
Ant Design	Moderate	Moderate	High	Large
Tailwind CSS	Moderate	High	High	Small/Lightweight
Bootstrap	Easy	Moderate	Moderate	Large

Table 3.2.10 UI Component Libraries comparison table.

3.2.10.1 Overview:

For UI Component Libraries, multiple of these will most likely be used for specific UI components that they may provide. However, Tailwind CSS will be the primary use. Tailwind CSS was selected specifically for its high customizability, it provides a powerful set of tools for designing modern interfaces which is great for satisfying our application need for a fast, flexible UI. Furthermore, Tailwind's ability to integrate efficiently with existing frameworks ensures we can design interfaces that are aesthetically pleasing and performant.

3.2.11 Database for App Backend:

Technology	Ease of Use	Performance	Scalability	Security Features
Firebase Firestore	High	High	High	Built-in auth features
MongoDB Atlas	Moderate	High	High	Needs external auth setup
PostgreSQL	Moderate	High	High	Secure but complex to set up
Supabase	High	High	High	Built-in user authentication

Table 3.2.11 Database for app backend comparison table.

3.2.11.1 Firebase Firestore:

[21] "Firestore is a NoSQL document database built for automatic scaling, high performance, and ease of application development. While the Firestore interface has many of the same features as traditional databases, as a NoSQL database it differs from them in the way it describes relationships between data objects."

Firestore's automatic scaling makes it a very strong choice for our application considering its real-time data synchronization. Its ability to scale on demand, paired with its ease of use, makes it an attractive option for app developers who desire quick and reliable database solutions.

3.2.11.2 MongoDB Atlas:

[22] “MongoDB is a source-available, cross-platform, document-oriented database program. Classified as a NoSQL database product, MongoDB utilizes JSON-like documents with optional schemas.”

MongoDB Atlas is well-suited for apps that require flexible data storage. It is a document-oriented structure that gives developers the ability to store and query data in a much more customizable format. For projects that need to handle complex data relationships, it is a very strong candidate.

3.2.11.3 PostgreSQL:

[23] “PostgreSQL is an advanced relational database system. PostgreSQL supports both relational (SQL) and non-relational (JSON) queries. PostgreSQL is free and open-source.”

Considering PostgreSQL’s popularity alone makes it very appealing to developers. Furthermore, it is known for being feature-rich, as it has support for both SQL and JSON queries allowing developers to work with structured and semi-structured data in a single system. For applications that need relational and non-relational data models, it is an excellent choice.

3.2.11.4 Supabase:

[24] “Every Supabase project comes with a full Postgres database, a free and open source database which is considered one of the world's most stable and advanced databases... Supabase Auth makes it easy to implement authentication and authorization in your app. We provide client SDKs and API endpoints to help you create and manage users.

Supabase provides a managed PostgreSQL backend with built-in features such as user authentication and real-time functionality which is very advantageous for a project such as ours. Its ease of integration with modern development workflows, along with the robustness of PostgreSQL makes it an ideal solution for developers who need a full-featured backend that is secure.

3.2.11.5 Overview:

For our deployable security system, we selected Firebase Firestore due to its real-time data synchronization capabilities, ease of use, and automatic scaling. Firestore’s NoSQL document model allows for flexible data handling, which is essential for storing and processing real-time sensor data, alerts, logs from multiple security nodes. The cloud storage and Firebase Authentication streamline will significantly accelerate the development process as it enables us to focus on building the system without worrying about the backend infrastructure as much. Furthermore, considering we want our devices to be able to handle scaling, such as adding more nodes to a network, Firestore’s ability to scale on demand matches perfectly. Overall, Firestore’s integration with developer applications and its real-time functionality perfectly fits our project’s needs.

3.3 Parts Comparison and Selection:

3.3.1 Camera:

In our project, we will need four to six cameras. They will need to be able to stream video and have a high enough resolution to perform image processing. These two requirements conflict with our other two needs, which are low cost and low power drain. Picking the correct technology and camera will be a balancing act of the two sides of our needs. The camera has a lot of constraints on it. The one we choose needs to be able to produce video, have a low power mode, be integrated with the system, and be low cost enough for multiple cameras to fit into the budget. We are looking at three technologies, the micro CCV cameras, basic off-the-shelf webcams, and MCU level cameras.

3.3.1.1 Technology Comparison Table:

Technology	CCV	Webcam	MCU Level
Cost: > 20$	down	2 down	up
Resolution Optimal: 1080p	up	2 up	down
Power Consumption	down	down	up
Integration Ease	Down	2 down	2 up
Video	up	up	up

Table 3.3.1.1 Camera Technology comparison table.

3.3.1.2 Micro CCV:

Part	Cuifati	BCOOSS	EYBEAR
Cost	9.99	17.99	26.99
Resolution Optimal: 1080p	720p	1080p	720p
Power Consumption	-	↓	-
Integration Ease	↓	↓	↓
Video	yes	yes	yes
link	ccv	ccv 2	ccv 3

Table 3.3.1.2 Micro CCV comparison table.

The Micro CCV doesn't look to be a good solution. While a few of them have some good features like night vision and onboard memory, the core functionality is subpar. The average cost is high considering that we will be buying 6 of them. Along with that our PBC board would need multiple Video ports, along with receiving and managing 2 video feeds at once.

3.3.1.3 Webcam:

Part	STINYTECH	PEGATISAN	YUKIJIANG
Cost	49.99	40.59	33.70
Resolution Optimal:1080p	1080P	1080P	720P
Power Consumption		-	**240MA/3.7V**
Integration Ease			
Video	yes	yes	yes
link	WEB1	WEB2	WEB3

Table 3.3.1.3 Webcam comparison table.

Webcam is a better solution technically wise. With extra functions like Wi-Fi connectivity and onboard memory storage the webcam would make other parts of the project easier. The downsides are they are larger than the other two technologies and the fact that they are a fully developed product they all have software and applications already installed. This would mean either removing the software and OS settings or integrating them. This could be impossible depending on what product we get.

3.3.1.4 MCU level camera:

Part	ESP32 - CAM	Raspberry Pi 4 B 3 B+ Camera Module	Hd Free Driver USB Camera Module 3.6 mm Lens USB Camera
Cost	7.66	~83	43.84
Resolution Optimal: 1080p	480p	1080p	1080p
Power Consumption	260 mA/h	2.85 watts	
Integration Ease	Up arrow	Up arrow	
Video	yes	yes	yes
link	link	link	link

Table 3.3..4 MCU level camera comparison table.

The MCU level is clearly the best choice for our application. Any cost that is incurred is outweighed by the ease of integration. Each choice would not only be the camera but would also be the backbone for the rest of the product. Each of these choices can natively be integrated with other applications and parts. The only downside is video quality, but this point will be negatable for our system.

3.3.2 Router:

WiFi routers are devices that allow multiple devices to connect to the internet wirelessly. Normally they work by receiving data from a modem, then broadcasting it as a wireless signal. But we will be using one as the backbone of our Wireless LAN. They connect devices like phones, laptops, and smart gadgets to this signal and access the internet. Most modern routers support multiple frequencies, allowing better performance and range depending on the environment. But due to the ESP32 only having the ability for 2.4 Ghz band, we will be staying on this band.

The 2.4 GHz band is a widely used frequency for Wi-Fi connections. It has a longer range than higher frequency bands, like 5 GHz, but tends to be slower. This is both good and bad for us. One hand the area covered will give us flexibility where we deploy our nodes but trying to push video through this band could be a problem. 2.4 Ghz can pass through walls and other obstacles more easily. Making it good for larger areas. However, many devices such as Bluetooth and microwaves also operate on 2.4 GHz, this could be problematic in high population areas but in the woods, it should be negatable problem. Despite these limitations, it remains a reliable option for basic internet needs and larger coverage areas.

Travel routers are smaller and more portable than full-sized routers. They are convenient for people who need to connect multiple devices to the internet while traveling. They offer easy setup in hotel rooms/public spaces. However, they often have fewer features, like lower signal strength and fewer Ethernet ports. Also, might not perform as well as full-sized routers for heavy internet use. The Travel type should be fine for a localized wireless LAN.

3.3.2.1 Technology Comparison Table:

Part	GL.iNet GL-SFT1200	TP-Link AC750	GL.iNet GL-AR300M16-Ext Portable
Cost	34.90	33.99	31.90
CPU	SF19A28, Dual-Core @1GHz	IPQ4018, Quad Core@717Mhz	QCA9531, @650MHz SoC
Product Dimensions	4.65"L x 3.35"W x 1.18"H	2.91"L x 0.87"W x 2.64"H	58*58*25mm
EAP Support	No	yes	Yes
Memory	DDR3 128MB	DDR3L 128MB	DDR2 128MB
Storage	SPI NAND Flash 128MB		FLASH 16MB
Speed	Max. 300Mbps	Max. 400Mbps	300Mbps
Link	link	link	link

Table 3.3.2.1 Router Technology comparison table.

3.3.3 Memory:

With us needing to be offline we will need to hold data and processes locally. The users of the Wireless LAN will need to access the app and website offline so it will need to be hosted locally. Then on top of that the image processing will need to be done locally too. With all these functions needing to be hosted locally we need a lot of memory.

With these constraints there are a few solutions. First there are exterior SSD this will give us a huge amount of data storage but is quite large and needs a large amount of power to operate. Then there are SD cards, these are very small and low powered. The third being a flash drive, this one is the cheapest per GB and removable but is easily corrupted.

3.3.3.1 Technology Comparison Table:

Technology	External SSD	Micro SD	Flash Drive
Cost	down	up	up
Capacity	up	up	up
Power Consumption	down	up	up
Retrieval	2 up	-	-
Product Dimensions	down	2 up	up
Weight	down	2 up	up
Longevity			
Encoding	up	down	-

Table 3.3.3.1 Memory Technology comparison table.

3.3.3.2 SSD:

SSDs (Solid State Drives) are storage devices that use flash memory rather than spinning disks like traditional hard drives. This design eliminates moving parts, which means SSDs are faster, more durable, and consume less power. They have quick boot times and silent operation. Though historically more expensive than HDDs, the price of SSDs has been dropping. For our project these were a good starting point to get a feel for the type of memory we needed.

Part	Vulcan Z	Crucial BX500	SanDisk SSD PLUS
Cost	17.99	62.80	29.99
Capacity	240 GB	1 TB	240GB
Retrieval (read speed)	520 Megabytes Per Second	540 Megabytes Per Second	1 Megabytes Per Second
Product Dimensions	3.94 x 2.76 x 0.28 inches	3.95" x 2.75" x 0.27" inches	3.96"L x 2.75"W x 0.28"Th
Weight	45g	2.8 g	31.75
Encoding(write speed)	450	500	530
Link	link	link	

Table 3.3.3.2 SSD comparison table.

3.3.3.3 Micro SD:

A microSD card is a small, removable storage device used in mobile phones, cameras, and portable gadgets. It's a smaller version of the SD card allowing it to fit into tight spaces while providing decent storage options. With no moving parts it's durable and energy efficient. Speeds vary by class and type. With higher-end models offering faster data transfers for tasks like 4K video. MicroSD cards come in capacities ranging from a few gigabytes to 1TB. Widely used, but reliant on compatible slots or adapters for broader device use. These seem like the optimal use for our product.

Part	SanDisk 128GB Ultra	SanDisk 256GB Extreme	TEAMGROUP GO Card
Cost	14.20	25.54	16.99
Capacity	128 GB	256 GB	256 GB
Retrieval	140 Megabytes Per Second	190 Megabytes Per Second	100 Megabytes Per Second
Product Dimensions	0.04"D x 0.59"W x 0.43"H	0.4"L x 0.5"W	0.04"D x 0.59"W x 0.43"H
Encoding	500	130	50
Link	link	link	link

Table 3.3.3.3 MicroSD comparison table.

3.3.3.4 Flash drive:

Flash drives (USBs) or thumb drive are a small, portable storage device that uses flash memory. They connect via a USB port and requires no external power source. These drives are popular due to their ease of use, durability, and ability to store large amounts of data. They are good for quickly moving files between computers or backing up important documents. they are slower than the other options but their ability to be easily removable could add functionality in testing and operations.

Part	SanDisk 128GB Ultra USB 3.0 Flash Drive	Samsung Type-C™ USB Flash Drive, 256GB	PNY 512GB Turbo Attaché 3 USB 3.2 Flash Drive
Cost	12.49	26.99	34.99
capacity	128 GB	256 GB	512 GB
Retrieval	130 Megabytes Per Second	400 Megabytes Per Second	100 Megabytes Per Second
Product Dimensions	0.32"D x 1.33"W x 0.63"H	0.32"D x 1.33"W x 0.63"H	0.32"D x 1.33"W x 0.63"H
Encoding	40	300	100
Link	link	link	link

Table 3.3.3.4 Flash drive comparison table.

3.3.4 DC to DC voltage regulators:

The nodes and gadget will be powered through batteries. Each device will have peripherals that must be powered by a specific voltage to function. We will be considering many different voltage levels, which will be or add up to above the desired voltage. We will introduce a voltage regulating element to solve the issue of having varying voltage inputs to efficiently convert to the required voltage.

3.3.4.1 Buck Converters:

Buck converters are step-down voltage regulators for when the output voltage is lower than the input voltage. They operate by modulating the on-off time of the MOSFET and supplying power to the inductor. A buck converter typically consists of an inductor, a switching FET (Field-Effect Transistor) or diode, a capacitor, and an error amplifier with a switching control circuit. The switching waveform is either generated by pulse-width modulation (PWM) or pulse-frequency modulation (PFM). It is then filtered through capacitors or inductors outside the chip to create the output voltage. Noise is usually a concern due to the inductors.

3.3.4.2 LDO:

Low dropout voltage regulators (LDO) are linear regulators that turn excess power into heat. They generate a stable output voltage when the difference between input and output voltages remains small. When the difference is at its lowest value is when it is called the dropout voltage. An LDO consists of a voltage reference, error amplifier, feedback voltage divider, series pass element, and a bipolar or CMOS transistor. The designs of LDOs are simple and produce no noise because there is no switching taking place. The loss of power through heat generation makes LDOs perform the worst in efficiency. The increase of the difference between input and output voltages only exacerbates the efficiency problem. Heat generation within our node or gadget must be carefully controlled due to the proximity of other electrical devices.

3.3.4.3 Power Management IC's:

Power Management ICs (PMIC) are a compact system-in-a-package solution to voltage conversion, voltage regulation, and battery management. A PMIC can manage multiple power sources and supply multiple loads. Most can mitigate overcurrent and undervoltage situations. Configuration is possible, allowing the user to control various aspects like output voltage, sleep and operating modes, and switching frequency.

3.3.4.4 Technology Comparison Table:

Technology	PMIC	Buck Converter	LDO
Cost	↓	↑	↑
Ease of implementation	↑↑	↓	↑
Power efficiency[1]	-	↑	↓
Heat generation[1]	-	↑	↓
Noise[1]	-	↓	↑
Size	↓	↑	↑
Versatility	↑	↓	↓

Table 3.3.4.4 DC to DC Voltage regulators comparison table.

1: Power efficiency, heat generation, and noise are dependent on the PMICs components

PMICs will be used because of their versatility and ease of implementation. There are variants to solve many different functions and can be easily scaled up and down to fit design requirements. Because we want to step down our input voltage to multiple levels, it is easier to achieve this with PMICs. Specifically, we will be looking at PMICs with multiple step-down buck converters because the LDOs are not as efficient and produce heat as a byproduct.

3.3.4.5 PMIC Parts:

Part	TPS65090	TPS65023	TPS65270	TPS65265
Voltage In Range (V)	6 -17	2.5 - 6	4.5 - 16	4.5 - 17
Voltage Out Range (V)	1 - 5	0.6 - 6	0.8 - 15	0.6 - 15
Max Current Out (A)	5	1.7	3	5
Quiescent Current (µA)	30 - 110	85	10	11.5
Step Down converters	3	3	2	3
LDOs	2	3	0	0
Cost	$10.85 / 1	$3.29 / 1	$1.96 / 1	$2.45 / 1
link	TPS65090	TPS65023	TPS65270	TPS65265

Table 3.3.4.5 PMIC Parts comparison table.

The TPS65265 best fits our nodes because it meets the requirements of powering a microprocessor, and many other peripherals at their rated voltage. This PMIC only has buck converters because we want to limit the heat generated in enclosed spaces.

The gadget will need 5 and 3.3 volts to power its router and microprocessor. Stepping down from 7.4 volts is best done through the TPS 65270. It offers two buck converters, which is exactly what the gadget needs, and other features such as overcurrent protection and low-power pulse skipping mode for better light load efficiency. Like the nodes, we will power other peripherals like the LEDs by stepping down using resistors.

3.3.5 Battery:

Our project's power management is one of the most important aspects of the entire project. There are seemingly endless amounts of paths we can take for powering our system, so the research in this section will go very in depth. Our main goal is to reach a max output voltage of at least 5V to reach the minimum voltage required by our parts so far. The most direct path to fill this requirement is to find a specific 5V battery that directly connects to our PCB that will direct the power elsewhere. Other types of batteries will need to be regulated by external factors to obtain a desired value.

Our system has multiple endpoints with a central control gadget, therefore, there will be two types of batteries chosen. Our node will need to be able to discharge a considerable amount of milliamperes at a moment's notice. The gadget on the other hand, will be consistently draining power from its active peripherals. Both devices will be powered by rechargeable batteries, so the chemistry composition will likely be similar.

Adding more batteries in parallel is an easy solution to powering our devices for longer. The electric potential will remain unchanged while adding more amp-hour capacity. The tradeoff for increasing the capacity will be an increase in charging time. The current delivered in a parallel system is the sum of each individual battery which decreases the energy used per cell that is added.

Battery voltages are determined by their chemical reactions and the concentration of the components. Nominal battery voltage is found at the equilibrium condition of a battery. Our system will be using rechargeable batteries which will limit the choice of materials. Common types include Lithium-ion, Lithium-ion polymer, Lithium-ion phosphate, and Nickel-metal hydride.

3.3.5.1 Nickel-metal Hydride:

The latter includes multiple drawbacks when in comparison with the other lithium-based batteries. The most important being their performance degradation from overcharging, and high amount of self-discharge. There is an LSD NiMH version which features a “Low Self-Discharge" at the expense of a small amount of capacity. The constant current draw is handled well by LSD batteries, but they may struggle when a spike in current occurs from activating the node.

3.3.5.2 Lithium-ion Phosphate:

Lithium-ion phosphate is similar to the other lithium-based batteries, but also suffers from high amounts of self-discharge. The cathode material is a non-flammable phosphate, so the lifespan is generally longer and safer. Their rugged nature is well suited to the harsh outdoors in both the heat and cold. This type of battery is also best for the environment due to the materials it is composed of.

3.3.5.3 Lithium-ion Polymer:

Lithium-ion polymer batteries use a form of highly conductive polymer in dry gel to store more energy in a smaller size. They are mainly used in laptops and smart phones because of their ability to conform to specific dimensions. A solid electrolyte in a pouch cell is safer because it is stronger and less prone to leakage. Although they aren’t completely safe because they are flammable if not used properly.

3.3.5.4 Lithium-ion:

Standard Lithium-ion batteries are the most common high-capacity rechargeable batteries in many devices including laptops, cell phones, and cameras. The cathode is made from lithium oxide, and the anode is made from carbon. Using this type of battery requires a way to regulate the energy flow to prevent overheating or even combustion. These batteries are also flammable which make them more sensitive in extreme conditions and perform poorly in low temperatures.

3.3.5.5 Technology Comparison Table:

Technology	Lithium-ion	Lithium-ion Polymer	Lithium-ion Phosphate	Nickel-Metal Hydride
Cost	↑↑	↑↑	↑	↓
Max Voltage	↑↑	↑	↑↑	↓
Energy Capacity	↑	↑	↑	↓
Self-Discharge[1]	↑	↑	↑	↓
Charge rate	↑↑	↓	↑	↓
Lifespan	↑	↓	↑↑	↓
Safety	↓	↓	↑↑	↑↑
Environmental	↑	↓	↑↑	↑

Table 3.3.5.5 Battery comparison table.

1: The self-discharge for NiMH is usually high, but it becomes low in its LSD version.

Our node will face the brunt of any weather conditions that occur. Our batteries must be ruggedized enough to withstand more than just temperate climates. The node has a passive and active mode, so the current must be able to spike at a moment's notice. The battery that best addresses the previously mentioned concerns would be the Lithium Phosphate batteries for the nodes. However, our self-implemented budget constraints render us to choose the next best option which is lithium-ion batteries.

Weather conditions will also affect the performance of the gadget batteries, but to a lesser degree because the gadget is meant to be within the safe zone created by nodes along with the user. There is a larger current draw at a more consistent pace in the gadget than in the node. Lithium-ion batteries are the best fit for the gadget because of their battery capacity to fit the needs of its high current drawing peripherals.

3.3.5.6 Lithium-ion Parts:

Voltage	1.5V	3.7V	9V
Cost / Amount	$31.49 / 4	$18.98 / 4	$34 / 8
mAh per battery	10000	9900	1300
Batteries required[1]	4	2	8
Cycle Life	1500	1000	1200
Dimensions per item	32*60 mm	67 * 18 * 18 mm	7.64 x 4.06 x 0.75 inches
link	1.5V Li	3.7V	9V Li

Table 3.3.5.6 Lithium-ion comparison table.

1: To meet 5 volts and about 10,000 mAh

3.3.5.7 Lithium-ion Polymer Parts:

Voltage	3.7V	7.4V	11.1V
Cost / Amount	$18.49 / 1	$30 / 1	$55 / 1
mAh per battery	10000	10000	10500
Batteries required[1]	2	1	1
Cycle Life	500	300-500	300-500
Dimensions per item	5.16 x 3.27 x 1.26 inches	3.86 x 1.26 x 3.46 inches	6.1 x 2 x 1.7 inches
Link	3.7 LiPo	7.4V LiPo	11.1V LiPo

Table 3.3.5.7 Lithium-ion Polymer comparison table.

1: To meet 5 volts and about 10,000 mAh

3.3.5.8 Lithium-ion Phosphate Parts:

Voltage	3.2V	6V	12.8V
Cost / Amount	$39 / 4	$35 / 1	$32
mAh per battery	12800	12000	10000
Batteries required[1]	2	1	1
Cycle Life	2000	2000	2000
Dimensions per item	32 * 70 mm	100 * 29.8 * 89.5 mm	5.94 x 2.56 x 3.7 inches
Link	3.2V LiFePO4	6V LiFePO4	12V LiFePO4

Table 3.3.5.8 Lithium-ion Phosphate comparison table.

1: To meet 5 volts and about 10,000 mAh

3.3.5.9 Nickel-metal Hydride Parts:

Voltage	1.2V	3.4V	6V
Cost / Amount	$23.33 / 4	$20 / 2	$30 / 2
mAh per battery	10000	3600	2500
Batteries required[1]	4	6	4
Cycle Life	1000	300	500
Dimensions per item	4.5 x 3.2 x 3 inches	7.13 x 3.07 x 2.32 inches	3.3 x 2.04 x .67 inches
Link	1.2V NiMH	3.4V NiMH	6V NiMH

Table 3.3.5.9: Nickel-metal Hydride comparison table.

1: To meet 5 volts and about 10,000 mAh

Our gadget will use the 3.7V lithium-ion batteries shown in table 3.3.5.6. The main considerations were the high capacitance and low cost. Lithium-ion batteries are very abundant making them much more accessible and cheaper for the average consumer. Its capacitance rivals the best in the market, while having the lowest costs.

The best batteries to weather storms and inclement weather are the lithium phosphate options. They are better for the environment and have the best lifespan and durability, however the costs of powering three separate nodes make them less than ideal. The next best option that will be used is the same batteries in the gadget.

3.3.6 MCU:

The MCU in the gadget must have a high data throughput to process the imagery coming from six cameras. To achieve this goal there needs to be a powerful processor and enough RAM to perform the intensive threat detection code. Another benefit to having more processing power is the data transfer rate to our storage device. The microprocessor must be able to quickly free its ram to move on to the task.

3.3.6.1 ESP32-S3:

The ESP32-S3 is a powerful microcontroller with enhanced AI and machine learning capabilities. On top of that it is relatively cheap for the functionality it provides. It features a dual-core Xtensa LX7 processor that can run at up to 240 MHz. The chip has 512 KB of internal SRAM and supports external flash and PSRAM for more memory. The on-board Rams gives us the capability to do most of the processing in each node locally. It also comes with 2.4 GHz Wi-Fi and Bluetooth 5.0, allowing for wireless communication in various IoT applications, perfect for our wireless LAN network. The ESP32-S3 supports hardware acceleration for vector instructions, making it ideal for tasks like neural network inference. It has plenty of GPIO pins, ADCs. And a variety of peripheral interfaces like SPI, I2C, and UART. Additionally, it includes secure boot and flash encryption for improved security. All this combined with its low power footprint and ability to go into low power modes make the ESP32-S one of our top picks.

3.3.6.2 STM32W:

The STM32W series microcontrollers offer a well-balanced combination of performance and power efficiency, particularly suited for wireless applications. With an Arm Cortex-M33 core running at 100 MHz, it supports multiple wireless protocols. It has a particularly low CoreMark score of 407, but it’s lower processing power also lowers its energy consumption, boasting a minimum power consumption of just 1.25 µA in standby mode while retaining 64 KB of RAM active. There is 1 MB of integrated flash and 128 KB of RAM, and up to 35 GPIO pins. While lacking in a few areas, this MCU provides a better low-power profile making it ideal for battery-operated devices.

3.3.6.3 PIC32MZXW1:

The PIC32MZXW1 has an M-Class core running at 200 MHz, making it more suitable for applications that require speed and reliability. The memory is broken down into 2 MB of embedded flash, 640 KB of RAM, which includes 512 KB allocated for data and 128 KB for Wi-Fi buffering. Its CoreMark score of 710 is great for complex calculations like our image processing. It includes advanced wireless features like a codeless WLAN development framework and has its power-saving modes consume only 2.7 mA in wireless sleep mode (WSM). With up to 37 GPIO pins, the PIC32MZXW1 is a versatile choice capable of handling our projects many peripherals.

3.3.6.4 CC3220:

The CC3220 from Texas Instruments is an MCU designed with wireless connectivity at its core, perfect for our applications which prioritize fast and reliable Wi-Fi communication. It features a dual core Arm Cortex-M4 running at 80 MHz which is slower than every other option by a significant margin. However, it is tailored specifically for Wi-Fi applications which are enhanced through SimpleLink technology, providing autonomous quick connections. The throughput ranges from 13 Mbps to 72 Mbps over TCP and UDP. Memory of the CC3220 includes 256 KB of RAM and optional 1 MB of executable flash. which overall makes this MCU optimized for efficient execution of wireless applications. While this MCU meets the minimum requirements for our project, it is extremely specialized and sacrifices too much processing power for wireless connectivity.

3.3.6.5 Technology Comparison Table:

Technology	ESP32-S3XX	STM32WXX	PIC32MZXW1	CC3220X
Cost	↓	↑↑	↑↑	↑
Processor	↑↑	↓	↑	↓↓
Memory	↑↑	↓	↑	↓
Power Consumption	↑	↑↑	↓	↑
Architecture Familiarity	↑↑	↓	↓	↓
Pin Count	↑	↑↑	↑↑	↓

Table 3.3.6.5: MCU comparison table.

Our gadget requires a very high throughput of data because of our image processing capabilities. Therefore, our processing power and memory are very highly valued. The product that would best fit this role is the ESP32-S3 series.

3.3.6.6 ESP32-S3 Parts:

Part	S3R8	S3FN8	S3R16V
Flash (MB)	0	8	0
PSRAM (MB)	8	0	16
Supply Voltage Min (V)	3	3	1.8
Cost	$2.28	$2.28	$2.49
link	S3R8	S3FN8	S3R16V

Table 3.3.6.6: ESP32-S3 comparison table.

There is minimal variance when selecting a specific version of the ESP32-S3 SoC. When the nomenclature ends in V, there is only a 1.8V external SPI flash. The letter F and R denote the usage of flash and PSRAM respectively. Pin mapping is also slightly different due to Quad versus Octal SPI modes. The deciding factor came down to the memory, and the best choice available for our gadget is the S3R16V.

3.3.7 LED:

Light Emitting Diodes (LED) are nonlinear semiconductor-based components that can't be connected directly to a power source. LEDs are like other diodes in that the current will flow in one direction. Because of this, polarity and the components orientation must be carefully considered for the unit to work. LEDs compared to traditional light sources have better power efficiency where less power is lost as heat generation. Another big difference is the response time of an LED is instant, while older lightbulbs tend to fade on and off. Color variance is a lot more common in an LED because they produce one wavelength while a normal lightbulb would need a color filter like dyed glass. Our project will need a small luminating device, and LEDs are our only realistic option. Miniature bulbs are quite expensive and fragile, which won't be useful for our ruggedized system.

Different colors require different wavelengths to be produced. The wavelength of an LED is determined by the specific semiconductor materials used. These wavelengths have their own typical forward voltage. It is expected that ultraviolet light is more voltage intensive than infrared light emission because the device must commit more energy to increase the frequency which shortens the wavelength. The same reasoning is applied to the visible spectrum where white and blue lights which have a greater frequency will require more voltage than a red light. There is an Ingress Protection (IP) rating system that applies to many LED light fixtures. It is recommended for outdoor LEDs to use a minimum IP65 rating for complete protection against dust and water jets from any direction. In this rating system it is important to know that the last digit represents the degree of water protection and that higher numbers equate to a greater degree of protection.

3.3.7.1 Through Hole LEDs:

Through-hole (THT) LEDs function by connecting physical leads on the opposite side of a PCB by passing through a premade hole. The LED consists of a chip protected in an epoxy resin case and two long leads. Holes in the PCB are needed, which could constrain the board layout. They require soldering at the end of the diode after the pins go through holes of the PCB. This creates a secure bond capable of handling more stressful situations. Replacing components is much easier, allowing for adjustments in the design process. Main applications of THT LEDs are for when durability and reliability is paramount.

3.3.7.2 Surface Mount LEDs:

Surface mounted (SMD) diodes are soldered directly to the surface of a PCB which decreases the size of the component. Because of their size they are generally more difficult to solder in place. Atop the solder which connects to the PCB board is the SMD frame. Inside the frame is a chip with gold wiring encapsulated by a phosphor substance. The frame does not allow light to pass through and acts like a cup around the chip which limits the light emitting angle. They are the most suitable for environments with limited space. They boast low power consumption, and thus have less heat generation and a longer lifespan than other LEDs.

3.3.7.3 Chip-on-Board LEDs:

Chip on Board (COB) LEDs have multiple LED chips directly mounted on a single substrate or board. The composition includes the chip on the PCB board encapsulated by a soft substance capable of withstanding deformations. There isn’t any frame to prevent light like in the SMD LEDs, which makes the light emitting angle much greater. The many chips on a small area produce a powerful and bright light. The best use case is for consistent and high luminosity and light uniformity. The close arrangement of LED chips creates an even distribution of light. COB LEDs are commonly found to be sold as long strips with very few other uses. Considering our projects gadget and nodes, there isn’t a strong necessity to have a uniform and bright light. For both types of devices in our project we will use LEDs primarily to denote on/off statuses. The only use case I can assume this type of LED providing for in our project is to light the surroundings of the nodes. The strips of COB LEDs are designed to bend to multiple surfaces allowing for versatile installation options. Something to consider is the typical voltage requirement is much higher than the other options. They are mostly found to be at 5 volts at the minimum, and commonly found to be higher too.

3.3.7.4 Technology Comparison Table:

Technology	THT	SMD	COB
Cost	↑	↓	↑↑
Size	↑	↑↑	↓
Ease of Usage	↑↑	↓	↑
Energy Efficiency	↓	↑	↑↑
Brightness	↓	↑	↑↑
Lifespan	↓	↑	↑↑
Durability	↑↑	↓	↑

Table 3.3.7.4: LED comparison table.

Lighting in our gadget will indicate whether the device and its peripherals are being powered. THT LEDs are preferred due to their ease of implementation and replacement. While still being mounted on a PCB, they extend further to enhance light distribution. It is also easier to design the housing of the devices around an LED that sticks out more.

To illuminate the surroundings of our nodes, COB LEDs are the optimal choice due to their superior brightness. They can be easily implemented as peripherals without direct attachment to the PCB. THT LEDs within the node will be used for the same purposes as and reasons mentioned above.

3.3.7.5 Chip-on-Board LED Parts:

Part	KXZM	Samsion	CHNMALITAI
Voltage (V)	5	5	5
Wattage per meter (W/m)	5	6	5
Length (m)	2	1	1
LEDs	640	320	320
Cost / Per	$11	$7	$10
IP rating (waterproof)	65	67	65
CRI index	93	85	90
link	KXZM	Samsion	CHNMALTITAI

Table 3.3.7.5: Chip on board LED comparison table.

Our nodes will perform best with Samsion brand COB LEDs. The key consideration was their waterproof rating. Samsion offers the most cost-effective option without compromising performance. Although the CRI index is the lowest, this does not affect quality since they will be used outdoors where it is less critical.

3.3.7.6 Through Hole LED Parts:

Part	CHANZON	Novelty Place	WOTOFULIN
Cost	$5	$6	$7.29
Quantity	100	100	100
Voltage (V)	2 – 2.2	2 – 2.2	2 – 2.4
Current (mA)	20	20	20
Brightness (mcd)	600-800	3000-4000	8000-12000
Light Beam Angle (Deg)	20	30	20-30
link	Chanzon	Novelty Place	WOTOFULIN

Table 3.3.7.6: Through Hole LED comparison table.

Chanzon LEDs are the best choice for minimal outdoor lighting. They offer the lowest costs and brightness, making them ideal for nighttime use due to reduced light pollution.

3.3.8 Alarm:

Audio hardware is an extremely dense topic, and we will investigate a few parts that will be beneficial to implement. Potentiometers are very useful in audio equipment. The benefits include frequency attenuation, and volume control. The taper in a potentiometer refers to how the resistance varies as the analog controlling device is rotated or as the wiper slides. A linear taper varies the resistance in an even manner, while the logarithmic taper slowly varies resistance on the low end and rapidly toward the high end. Audio tapers are logarithmic because our ears are sensitive to changes at lower volumes.

3.3.8.1 Speaker:

Speakers are the most common device used in audio, along with a supporting amplifier to increase the volume and improve sound quality. They are generally louder and have more capability to produce a variety of sounds. Sound is created by sending current through coils of wire to create an electromagnetic field. Audio signals travel through the voice coil. The coils repel and attract with a permanent magnet, which displaces the air from its alternating motion and that produces the sound that we hear.

3.3.8.2 Piezo Buzzer:

Named after the piezoelectric materials that comprise them, piezo buzzers provide auditory cues to the end user. Their materials offer a unique ability to convert energy between mechanical and electrical forces. The piezoceramic disk converts electric current into sound producing mechanical movement through vibration.

3.3.8.3 Magnetic Buzzer

Magnetic Buzzers differ from piezo buzzers in their components and process of converting current to sound. At the center is a ferromagnetic disk which is a flexible magnetic disk. The flexibility allows for vibration to produce sound. The current is run through a wire coil which creates an electromagnetic field. When the field is activated, the disk bends towards the field, and returns to a normal state when there is no current. The buzzer creates sound when rapidly alternating the current between on and off.

3.3.8.4 Technology Comparison Table:

Technology	Speaker	Piezo Buzzer	Magnetic Buzzer
Cost	↓	↑	↑↑
Operating Voltage	↓	↑	↑↑
Power Consumption	↓	↑↑	↑
Sound Pressure Level (dB)	↑↑	↑	↓
Frequency Response	↑↑	↑	↓
Size	↓	↑	↑↑

Table 3.3.8.4: Alarm comparison table.

For our needs we do not need all the functions of a speaker, but we still want to produce a loud sound. Piezo buzzers are the second loudest option while staying simple in design. We won't need to add an amplifier for either of the buzzers, and the power consumption is the lowest for the Piezo buzzer.

3.3.8.5 Speaker:

Technology	WEICHUANG	BNYZWOT	Jameco
Cost / Per	$5.49 / 3	$8.39 / 10	$7.90 / 2
Range of Voltage (V)	3 - 24	3 - 24	4 - 16
Rated Voltage (V)	12	12	12
Rated Current (mA)	15	12	25
Sound Output (dB)	100	85	95
Frequency Response (Hz)	3300 ± 500	3300 ± 500	2900
link	WEICHUANG	BNYZWOT	Jameco

Table 3.3.8.5: Speaker comparison table.

The WEICHUANG Piezo buzzer is the best option due to its highest decibel output, which is crucial for our needs. It is also the most cost-effective option and comes in the exact quantity required for our project.

Chapter 4 Standards and Design Constraints

4.1 Introduction:

In the development of our project, adhering to established standards and properly identifying design constraints is essential to ensure the reliability and functionality of any system. This chapter outlines the key standards and design constraints that influenced the development of our deployable security system. These standards not only guide the technical aspects of our project but ensure they operate effectively within the physical, environmental, and ethical boundaries of its intended use.

Considering our project's use as a security system designed for use in outdoor environments, it requires a more careful consideration of both resilience and usability. This helps us prioritize certain goals for our project such as providing efficiency and reliability while being able to withstand the elements. Throughout our design process this far we have been mindful of the standards related to resilience, user-centric design, and environmental sustainability.

This first chapter will explore relevant standards our project must meet, such as resilience against tampering and maintaining performance in varying weather conditions. Then moving onto design constraints which explain the obstacles that have impacted the system's development, such as material selection, power efficiency, memory management, and environmental factors. Analysis of these standards and constraints have informed our decisions on how to optimize the system for long-term functionality and ease of use. We aim to deliver a product that not only meets the needs of our users but also aligns with the best practices in security, design, and environmental responsibility.

4.2 Relevant Standards:

Some standards that our project is going to have to meet are resilience towards attacks on our system and ensure we have recovery mechanisms in such a case. Another standard that our system wishes to meet is having a user-centric design. Through meeting these standards, we wish to provide a reliable system that our customers will feel confident in, while also providing ease of use and abstracting any intellectual requirement from operating the system on the user side.

When designing our system, standards were not the first thing that popped into our minds, so it was something that we really had to put some thought into as far as how we wished to accomplish our goals. Reliability is one of, if not the most important aspects of a security system. If you are not confident in your security system to accomplish its sole purpose, then it is useless. We plan to gain trust of our customers by providing reliable and trustworthy product through our security methods such as anti-tampering, providing durable physical product to our users, and through thoroughly testing our product to identify any potential security breaches our system may have. Anti-tampering plays a pivotal role in our system's resilience. We plan to accomplish this feature through both hardware and software efforts. For example, if per say somehow someone walks up to one of our nodes without the passive IR sensor and ESP32-CAM detecting a person and alerting our system. Their next goal will more than likely be either to destroy one of our nodes or get inside of our secured area. Measures we have against getting inside of the area are a different topic, but for a short countermeasure we will be having an active IR sensor that will continuously monitor the perimeter. Measures we have against node tampering consist of software running inside of our network that will monitor the “Heartbeat” of each node inside of our network. We will be setting up alerts and notifications that will be sent out to the gadget and our mobile application, that when the system detects any unauthorized changes or if a component goes offline unexpectedly, our alarm will be triggered classifying this as a security breach, hence alerting the user through notifications and our physical alarm on each node. The software that will perform our heartbeat functionality will live inside of main.ino on each node. The goal will be to monitor, just like how we monitor the passive IR sensor continuously, and if the heartbeat is ever not detected then we will consider that node tampered with. We then will send a message over HTTP to the gadget. Once the gadget has received this message it will then Alarm the other nodes and the user via our alarm system. The alarm system will be set up as follows, whenever a threat has been detected, by either identifying a person from our image processing algorithms, the active IR sensor being tripped, or by a node being tampered with. The Gadget will send messages to all “alive” nodes and from there the nodes will sound an alarm contained in the housing of the nodes.

Another feature of our system resilience is what our physical hardware will be made of. This is a feature that does not play a role in the efficiency of detecting people or motion outside of our secure area, however it will allow us peace of mind with the fact that if our system is put in place in the real world, it will not crumble under pressure if there is a little wind or rain and completely fail. We plan for the physical pole that will hold up our “Nodes” to be roughly 5-8ft above the ground and they will be made from pvc pipes. Pvc pipes are resistant to corrosion and rust, which is perfect for outdoor usage in the rain. Their hollow interior increases spatial scalability because they will be insulated and protect the electronics we may want to implement inside. LEDs and other peripherals can have a ingress protection rating (IP) which will allow us to meet a variety of reliability standards such as being waterproof, durable, and able to withstand wind as we plan to implement ground spikes or to carve the bottom of the pvc pipe to ensure our pole does not get lifted out of the ground unintentionally unless we have a hurricane rolling into town. As far as our ESP32 housing goes, we plan to make that water resistant as well. We will be building custom housing for our ESP32-CAM nodes that will fit perfectly to our system requirements. Once we have fully integrated and developed all aspects of our security system, to ensure we can guarantee our customers' reliability under circumstances, we plan to thoroughly test our product with multiple test plans. For example, one of our tests will consist of a waterproof test where we will provide a rainlike environment to ensure the system is capable of enduring moderate weather as this product is geared towards the outdoors and should not have to be picked up every time it starts sprinkling outside. We will go over more testing in chapter 9, however rest assured that our system will be thoroughly tested to meet reliability under circumstances that our product could encounter once deployed.

Our second standard is based on our product having a user-centric design. For any product sold to consumers, it is common practice for there to be little to no hard effort to deploy a system that you have bought, hence why we are making this a standard for our project. We plan to accomplish this in a couple of ways. First, we wish to be able to set up and deploy our system in the field relatively quickly and easily. Simply, turn on the gadget, connect everything to the router by flipping a switch on the gadget, and then place your nodes in the correct formation to where the active sensors are at least pointing at each respective pole. Another feature we will be adding to our system design that ensures we keep this a user-friendly product is we will be developing an interactive mobile app. This new feature will allow users to not be physically present at the secured perimeter and will set up a communication network between the router and your phone. Through this app you will receive notifications if your secured area alarm has been triggered, and even can request for your cameras to take a snapshot of the perimeter and you will see 6 different captured images of the perimeter for the area on your phone. Another feature we have integrated into the design of our system that allows for a user-centric design is through the functionality of the gadgets. We will have multiple switches on the gadget that will allow users to interact with the system such as turning the power on, a switch to flip that will capture and send a message to the nodes and an image will be captured and sent to the mobile app, and then a switch that will turn the alarms on if a user has noticed any activity outside the area that hasn't already tripped by the alarms, and the last switch on the gadget will turn the alarms off if a threat has been handled and the user no longer wishes for the alarm system to be triggered. Keeping the user in mind while designing and developing a product is key to success with anything you create, and we believe that with these features we are developing we have met the requirements to meet the standard for our product by having a user-centric design.

[44] One of the key institutions we have benchmarked ourselves against was IEEE. The Institute of Electrical and Electronics Engineers (IEEE) is a global professional organization dedicated to advancing technology across multiple fields. Founded in 1963, it has grown to become one of the largest organizations of its kind. Its mission focuses on fostering innovation and education, including the development of technical standards. These standards ensure that technology is safe, reliable, and compatible across different platforms and industries worldwide.

IEEE is renowned for its role in creating standards that are followed across various sectors, such as telecommunications, information technology, and power systems. These standards are crucial because they provide guidelines for manufacturers and engineers, enabling the design of products that are interoperable with other systems. Additionally, they ensure the safety and efficiency of electrical systems, helping prevent accidents and equipment failure.

One focus of IEEE’s standards is low-voltage electrical systems, which typically operate under 1000 volts and are commonly used in residential, commercial, and industrial settings. The IEEE standards for these systems promote safe, energy-efficient designs while ensuring compliance with regulations. By providing installation guidelines for equipment like wiring, circuit breakers, and protective devices, they help mitigate risks such as electrical fires or shocks.

The IEEE 3189 standard is one example of their contributions, concentrating on energy-efficient solutions for computing devices. It encourages the design of electronics that minimize energy consumption, a growing concern as digital technology expands. This is particularly significant given the rising global demand for energy as more devices enter daily use.

Another prominent standard is the IEEE 802 series, which governs local area networks (LAN) and metropolitan area networks (MAN), including technologies like Wi-Fi and Ethernet. IEEE 802 serves as the foundation for many modern networking systems, ensuring that devices from different manufacturers can communicate seamlessly. This enhances compatibility and performance across networks, benefiting consumers and businesses alike.

The IEEE 1012 standard addresses software verification and validation. It offers a structured approach to ensuring that software is developed and maintained according to best practices. This standard outlines processes for planning, executing, and documenting software tests, which are critical in industries such as aerospace, medical devices, and automotive systems, where software reliability directly impacts safety and performance.

[44] Ingress Protection (IP) standards are another critical area of classification, used to assess the level of protection a device offers against solids and liquids. These ratings are vital for ensuring that electrical devices are safe in different environments. The ratings are represented by two numbers, such as IP96, where the first number indicates protection against solids like dust, and the second number shows resistance to liquids like water. The higher the number, the stronger the protection.

The IP standards emerged from the need for a consistent way to measure and communicate device durability under various conditions. Before the IP system, manufacturers often used their own protection ratings, leading to confusion and difficulty in product comparison. In 1976, the International Electrotechnical Commission (IEC) introduced the IP standard through the IEC 60529 classification system to provide clear and universal guidelines for manufacturers to follow.

The purpose of the IP system is to help both consumers and professionals understand how well a device withstands dust, moisture, and other environmental factors. For instance, a device rated IP68 offers full protection from dust (indicated by the “6”) and can be submerged in water for prolonged periods (indicated by the “8”). These standards ensure that devices intended for harsh environments, like outdoor cameras or industrial machinery, continue to function safely and reliably.

IP standards are crucial in many industries, from consumer electronics to construction and healthcare. They enable manufacturers to design products with appropriate protection for their intended environments. For example, smartphones often carry ratings like IP67 or IP68, signifying they are dust-tight and resistant to water immersion. In industrial applications, IP ratings help ensure that equipment can endure exposure to dust and water, promoting worker safety and reducing the likelihood of equipment failure.

4.3 Design Constraints:

Designing an outdoor electronic device comes with multiple challenges. One of the biggest constraints is weatherproofing. Since the device will be outside and needs to be protected from rain and humidity. This means using materials that are resistant to water. Such as plastics and silicone-based seals. Plastic cases with rubber/silicone gaskets will help keep water out and prevent damage to the internal electronics. Also, the materials should be able to withstand extreme temperatures, without warping or breaking down.

Preventing water damage isn’t just about protecting the outside. The inside of the device must also be carefully designed. Any opening of wires or ventilation can be a potential point for water to get in. So, these areas need extra protection. We have considered adding coatings or sealants to internal components to provide more water resistance but that is outside the scope of a prototype. And with moisture comes corrosion. This is another problem, so choosing corrosion-resistant materials is important. To avoid internal moisture, we are also looking at desiccants and other moisture reducing materials. Some parts in the design have specific waterproof rating called ingress protection or IP for short. It is a protection measurement rating system against solids like dust and dirt as well as liquids. Other peripherals also have IP ratings like buzzers; however, they aren't necessary with a proper housing design. It is more beneficial to invest in LEDs with an IP rating because they are more subject to the outside environment. They need to be more present and illuminate outside the safety of our housing units.

The interior of our housing units for the nodes will be accounting for stormy weather by allowing the rain to runoff away from our electrical parts. The goal is to never allow water inside of the housing and we will achieve this by creating an elevated center and the edges will extend further out. The runoff will fall off the edges without entering the device’s confines. The gadget’s housing will also be designed to work in the rain by sealing the exterior. Along the precipice of the exterior will be through-hole LEDs that are protected by a plastic casing, and the connections to the housing will be filled with a waterproof sealant. The charging ports on both the nodes and gadget will be protected as well. The housing design will include access points to charge the devices and will close off the access points for when in an operational mode.

Safety is another big factor in outdoor designs. The product needs to be electrically safe, especially if it’s exposed to moisture. It must be properly grounded to prevent electrical shocks or short circuits. Considerations have been made to add grounding strips to the spike. But with low voltage and amperage this is a marginal concern. Additionally, the device must be secured so it doesn’t pose a hazard if it falls. The product should also meet local electrical safety standards, and care must be taken with wiring.

The ESP32 is a powerful microcontroller with a lot of functionality. But its Wi-Fi capabilities are limited to the 2.4 GHz band. This creates certain design constraints. One key issue is interference because many other devices, like Bluetooth and appliances that operate on the 2.4 GHz frequency. This can cause signal congestion, leading to slower data transmission and unreliable connections, especially in environments with many competing devices.

Another constraint in the 2.4 GHz is speed. The band has lower data transfer rates compared to higher frequencies like 5 GHz. This can be a problem if your network needs to handle a lot of data quickly, such as streaming video. With our product having video streaming capability, this will be a hurdle for us. But the 2.4 GHz band has better range and can penetrate walls more effectively.

Additionally, the ESP32's use of the 2.4 GHz band affects network scalability. Since it's a shared frequency, the more devices you add, the more likely the network performance will degrade. We have to carefully plan how many devices connect to the network to avoid bottlenecks. These design constraints require trade-offs between range, speed, and the number of devices to maintain a stable network. This is one of our largest constraints

One of the primary design constraints for our deployable security system is the efficient management of memory allocation for storing and processing images captured by the ESP32-CAM nodes. The ESP32-CAM has limited onboard memory, which presents a challenge when dealing with high-resolution images or video streams. Given that the system needs to store multiple images for analysis, including those from motion detection events, we must carefully manage how memory is allocated and used.

The limited onboard RAM of the ESP32-CAM restricts the size and number of images that can be stored locally at any given time. This constraint requires us to use memory-efficient image formats, such as JPEG compression, to reduce the file size while maintaining enough quality for threat detection. Additionally, the image processing tasks performed on the ESP32-S3, such as identifying potential threats using machine learning algorithms, require sufficient memory to run effectively. Balancing the storage of raw images and processing capabilities is crucial to avoid performance bottlenecks.

Another constraint arises from the need to store images long-term for review and analysis. Since the system is designed to operate in remote environments, we have to account for the limited storage available on the ESP32-CAM itself. One potential solution is to periodically offload images to external storage or cloud services, but this introduces additional power and network constraints, especially if operating in areas with limited or no internet connectivity. Therefore, efficient memory management strategies are essential to ensure the system can store, process, and transmit images without running out of memory or compromising system performance.

4.3.1 Time Constraints:

It is essential that our system reliably performs its duty to ensure the safety and security of the environment. In a real-time system like our own, a desirable response is degraded by the delay of its arrival. Depending on the magnitude of the delay, it could be just as hazardous as an incorrect response. The node works in a low-power mode when idle and must be capable of running at full capacity at a moment's notice.

Performance is an important non-functional requirement in our real-time system. In contrast with other real-time systems, response time is not necessarily important to protect the system but to inform the user of their surroundings. Another aspect of a performance requirement is the systems throughput. More specifically, it is crucial when our system is handling imagery from capturing at the node to processing on our gadget, and finally alerting the user of what was detected.

Threat detection in our image processing is our greatest timing concern. There are potentially six cameras sending imagery to be processed at the same time over the same network.

Factors outside of our system can also affect the response time. A major selling point of our system is that it can be deployed just about anywhere the environment allows. Both extreme hot and cold temperatures will affect the performance and by proxy the response time. Obstructions near the devices may cause interference when transmitting data over the Wi-Fi network.

Our solution to this constraint is combatted by our software architecture. The architecture has been designed to be able to wake up the entire MCU within a split second, capture an image, send that to the gadget, and process the image on the gadget quickly. We are accomplishing this through multiple ways. The first way we are doing this is by the selection of language for our software. We have chosen C++ for numerous reasons, however, the main being that with C++ we can have the most control of what our system does as possible. For example, with C++ we can take advantage of manipulating memory directly, giving us access to how much memory we use and ensure that no time is wasted in that process of allocating/deallocating too much or too little. Another way we will be saving time is in our image processing algorithm, which is the most time-consuming program our system runs. Most programs that are in python, although a lot easier to implement, take up quite a bit of resources and time that our system simply does not have. Through the use of C++ and simply the libraries such as TensorFlow Lite that we will be using, we will be using far less time and resources to accomplish person detection.

4.3.2 Part Constraints:

Our product must be rugged and mobile, requiring parts to be lightweight, low power, and durable enough for constant use and abuse. With the main use case being outside we needed to prepare for rain, dirt and other hazards to electrical systems. Given time and a proper R&D cycle we would aim for an IP67 rating to guarantee the safety of our equipment. But for this prototype we will not be aiming for any IP rating and will not be testing using water or dust.

A key parameter we considered was the use of self-contained systems. For example, we chose the ESP32-CAM as the camera system. It is low power, lightweight, and only requires physical integration for power. Software integration is handled internally or over our local wireless LAN to the gadget and the locally hosted website.

Power consumption is another priority. We aim for at least 24 hours of operation between charges, which requires over 10,000mAh for the nodes and 15,000mAh for the gadget. Lithium-ion packs are the best solution due to their flexibility in size and shape, fitting well into our design. They also operate within the voltage range of common phone chargers (6-12V), allowing us to use micro-USB for easy charging.

The main challenge will be ruggedizing the system. Ruggedizing the PCBs and ESP32-CAMs is beyond the scope of this project. For consumer-grade products, we would use higher-grade PCBs and water-resistant casings, but this is too costly for a prototype. To mitigate potential issues, we're using low-profile components on the PCB to reduce shake and moisture damage. The casing will be rigid to protect the battery pack from impacts.

4.3.3 Safety Constraints:

With any product a good Indepth safety review is a must. This allows designers to look at their products from a different angle and design safety features as well as catch problems before development is in effect. Saving time and money but more importantly making sure their product doesn't harm anyone.

Our product will have a few pieces of equipment that could be harmful. The main concerns are the Infrared sensors, the battery packs, and the alarm system. We will be using OSHA's Technical Manual for our safety standards.

Infrared emitters can be harmful to the eyes. This is compounded by the fact that infrared is invisible to the human eye. For our first idea for our active IR trip wire, we planned on using a class I laser as the emitter to gain distance. Chapter 6, section III of OSHA's regulations covers lasers and their hazards and states that "Class I: cannot emit laser radiation at known hazard levels (typically continuous wave: cw 0.4 μW at visible wavelengths). Users of Class I laser products are generally exempt from radiation hazard controls during operation and maintenance (but not necessarily during service)." While this is a good thing, we still need to be careful with power generation and heat generation. For our prototype we have opted for an IR LED instead of the class I laser due to its lower energy/heat cost as well as a wide beam width compared to the laser giving us more room to work with. In future iterations of our product a laser could be implemented to give us a significant range increase.

Our batteries onboard have a need for significant power compacity. This has pushed us to use lithium as our battery of choice. Lithium provides several safety concerns. Lithium Batteries are not impact resistant and when damaged can experience a thermal discharge resulting in fire and even detonation of the battery pack. There are also temperature concerns with Lithium; battery packs can be damaged with temperatures over 130 F and below 32 F. Both are addressed inside our casing. We have opted for a nonremovable battery pack to mitigate these problems. With the PVC pipe protecting the pack and light insulation protecting the temperature, the battery pack shouldn't experience any environmental dangers. On future iterations, upgrading the battery to a more stable solution will be to meet the demands of harsher climates.

For the alarm system we need to be able to scare off any perceived threat to the area, but we also need to stay within the correct limits. OSHA standard 1910.95 is the Occupational noise exposure. We used this as guidance for our noise level. As we designed for 80 dB max level this is lower than the lowest level, they state placing us in a very safe range as far as noise is concerned.

4.3.4 Environmental Constraints:

The outdoors introduces many new considerations when creating an electrical system. Our goal is to have full functionality of every part of our system in temperate climates. The housing unit for our node must be able to withstand a moderate amount of rain at the very least. Considering our target demographic, we must provide a product that can be easily transported and deployed in multiple terrain types. The PVC slant at the bottom of the nodes will not penetrate harder substances like rock or concrete. This prohibits practical usage in most urban and indoor environments. Part of our project's functionality requires sensors to properly line up, thus the nodes must be planted in the ground or held by external devices.

Factors such as temperature are very important to consider when working with electrical systems. Thorough research for every single part was completed to find that the optimal temperature range is between -10 and 50 degrees Celsius. The operating range is limited due to the batteries. When operating outside of the range, it is expected to have reduced charge capacity, and a general sense of degradation in the performance of the whole system. It is advised by REDARC Electronics to charge lithium-ion batteries in an ambient temperature over 0 degrees Celsius. Below freezing has adverse effects like reducing the lifespan and performance of the battery.

Environmental conservation is a major consideration in our project’s functionality. Our project will have minimal disruption to the wildlife when it is deployed. There will be a slight amount of displacement in the dirt when placing the nodes in the ground. Upon completion and dismantling of our system, that displacement can be immediately filled manually or over time naturally. According to the National Wildlife Federation, “The moment the ecosystem begins adjusting to one stress, another appears. [4]” Our product may cause stress amongst the wildlife which could affect the ecosystem of that area. This is an unavoidable issue because we must prioritize our user's safety. The deployment of our system should therefore be discouraged from being used for long periods of time. The disruptions caused by our system include a loud alarming noise, and flashing LEDs to alert the user and keep wildlife away. Our gadget was designed in a way to create no disturbance to the environment. Our system from its deployment to its application, and after tearing down, will leave no waste or pollutants in the area. When used appropriately, our system will create little to no disturbance to its surroundings.

4.3.5 Sustainability Constraints:

A major concern for our system is its ability to operate effectively over time, particularly in suboptimal outdoor conditions. Sustainability focuses on how well our system can function over long periods while minimizing resource consumption. Given that the primary usage of the deployable security system will be outdoors, our design process aimed to create a device that not only serves users efficiently but also offers a long operational lifespan.

Durability plays a crucial role in sustainability. A highly durable system reduces the need for frequent replacements and minimizes waste. The physical components of our system, such as the frame, will be made from ABS filament, known for its toughness and resistance to environmental stress. This material choice ensures that our system can withstand outdoor conditions such as rain, wind, and temperature fluctuations. Additionally, the ESP32-CAM nodes in our system are designed to operate with low power consumption, which extends battery life and significantly reduces the frequency of battery replacements.

In terms of energy consumption, our product is designed for continuous operation in outdoor environments. We selected lithium-ion batteries for their high energy density, which strikes a balance between power requirements and sustainability. These batteries are lightweight, portable, and offer longer charge cycles than other technologies, reducing waste and the environmental impact associated with battery disposal.

To address hardware issues and ensure easy maintenance, our system features a modular design. This allows non-technical users to perform basic repairs without the need for specialized tools or expertise. If one of the nodes malfunctions, the issue can be easily identified and resolved. This repair-friendly approach reduces the likelihood of users discarding the entire system due to a single component failure.

In summary, sustainability is a vital consideration in the design of our Deployable Security System. By prioritizing material durability, efficient power management, and modularity for easy repair, we ensure that our system remains functional, reliable, and sustainable over time.

4.3.6 Ethical Constraints:

Our product's main purpose is security and the safety of the user. Our product will give the user a higher situational awareness of the surrounding area and give them a heads-up on perceived threats. With this surveillance of the surrounding area, we must not cross the line into intruding on others' personal privacy. To keep in line with ethical concerns we will be using our state laws as our ethical guideline.

Our design is meant to be used in camping and outdoor activities. Many campgrounds are on either state or federal land and are considered public areas, even businesses are considered public areas in terms of privacy. Florida Statutes Title XLVI. Crimes § 810.145. Video voyeurism (5). states as follows: "(c) Video surveillance device that is installed in such a manner that the presence of the device is clearly and immediately obvious;". Our product is clearly marked and visible as it is made to be a deterrent. We will not be implementing any Stealth features to disuses our product.

Along with design we need to consider if the end user missuses our product. In the same statute: "(d) Dissemination, distribution, or transfer of images subject to this section by a provider of an electronic communication service as defined in 18 U.S.C. s. 2510(15), or a provider of a remote computing service as defined in 18 U.S.C. s. 2711(2). For purposes of this section, the exceptions to the definition of "electronic communication" set forth in 18 U.S.C. s. 2510(12)(a), (b), (c), and (d) do not apply, but are included within the definition of the term." This subsection defines us as the service provider and aren't held accountable for misuse of our product or services.

We also have operation states that let the end user determine the uses of our equipment. With the ability to use just the IR triggers with no Camera and the quiet mode that doesn't use the alarms, end users can be conscientious of the surrounding area and of others privacy and peace.

4.3.7 Software Constraints

Software constraints are a very common difficulty when developing any system, therefore, this is a crucial section to include. Some of the issues for software addressed above are memory management and time, but there are plenty more to be discussed for the scope of this project. One of the other major software constraints is a resource constraint. When running our image processing algorithms, a serious issue we are going to encounter is the amount of computational power required by our algorithm vs what our board is going to offer. Extensive research has been performed for this topic and we have concluded that TensorFlow Lite is going to be a library low-level enough for us to meet the objectives of our goal in a timely manner while not crashing our MCU. Another software constraint must be developed for real-time processing. Meaning all algorithms must be geared and architected carefully with a low latency design in mind. Plans to achieve such a task have been accounted for with research into the best data structures that will result in the lowest run times. Typically, with a real time system the data structures should aim to be as close to O(1) run time as possible.

Chapter 5 - Comparison of ChatGPT with other similar Platforms

Chat GPT, developed by OpenAI, is a free virtual assistant / chatbot that came online in November of 2022 and quickly gain popularity. There are many products on the market like ChatGPT like Claude and Google Gemini. But ChatGPT is the most popular, this is in part due to it being free and being easily accessible as well as having versatility across multiple disciplines. It also is very good at formatting information to be easily digestible as possible. With many added features like image generation capabilities and conversation memory features, users feel very connected to the platform. The only negative many users experience is occasional latency when accessing real-time information. But overall, the platform is highly competent at helping with a wide range of tasks.

Google's Bard excels particularly in current events analysis and general inquiries, leveraging its direct integration with Google Search and the world wide web. The platform offers seamless compatibility with Google Workspace applications, providing significant value for users invested in the Google ecosystem. However, its responses tend to be less sophisticated than ChatGPT's, and its programming capabilities are somewhat limited. Googles Bard seems to be focused on a wider user base rather than deep Indepth analysis.

Microsoft's Bing Chat, powered by GPT-4 technology, combines advanced language processing with Bing's search functionality. While it should bevmore functional than GPT alone, Bing chat feels limited in its capabilities. Its integration with Microsoft Office applications makes it particularly suitable for professional environments, though it typically provides more concise responses compared to its competitors.

Claude, Anthropic's contribution to the field, distinguishes itself through its emphasis on ethical considerations and exceptional clarity in communication. Its capacity to process extensive text inputs makes it particularly valuable for document analysis and summarization tasks. It shows a good level of nuanced understanding of complex topics. While it may not offer as comprehensive coverage of technical subjects such as coding or as many interactive features as its counterparts, it excels in its core functionalities.

Each platform offers distinct advantages that cater to specific use cases and user preferences, making the optimal choice dependent on individual requirements and objectives.

Feature	ChatGPT	Google Bard	Bing Chat	Claude
Real-Time Info	Limited (Browsing)	Yes	Yes	Limited
Code Proficiency	High	Moderate	Moderate	Moderate
Tool Integration	DALL-E, Code Tools	Google Services	Office Suite	None
Content Length	Medium to Long	Medium	Short	Long Context Support
Best Use Case	Multi-functional	Real-Time Insights	Office Integration	Document Analysis

We used Chat GPT and other models extensively throughout this project and report. The benefits to the more tedious and mundane parts of the report like consistent tone, consistent tense, wording, and grammar are a tremendous boon to our team. These functions were used throughout the report. Most of our processes with these tools were regenerative in nature. This means we have information, and we are using these tools as a standardizer and checker rather than using them as a generative tool that produces new information.

We also have used them to gain different perspectives than our own. Listed below were a few ideas we had and used chatGPT to give ideas and feedback that we might not have thought of.

We have also used ChatGPT for technical help too. We have used it to verify the thought processes of wiring between components and verify if there are any logical faults we have missed. It has also been used to help explain technical language in documentation, quickly explaining foreign nomenclature and verbiage. It has also been used to verify code and syntax of said code.

While we can't document every time, we used these tools as they were used within the development process as a regenerative tool, we still want to give them the credit they are due for being useful tools.

5.1 Case study 1:

Prompt: “What functionality should a deployable security system have, noting it will need to be on battery power.”

Outcome: A deployable, battery-powered security system needs to balance functionality with power efficiency. Here are key features it should have:

1. Power Management: The system should have low-power operation, optional solar charging, and battery monitoring for power efficiency.

2. Sensors for Detection: It should include motion detection (PIR or microwave radar), cameras (with night vision), and environmental sensors for tamper detection.

3. Communication & Alerting: Wireless connectivity via Wi-Fi, LoRa, or cellular for remote alerts, local alarms, and push notifications. It should offer cloud storage for footage.

4. Video Recording & Storage: Onboard storage (e.g., SD card) with triggered recording to save power, along with a low-power live video feed option.

5. User Interface: Mobile app control, web interface for remote access, and a physical keypad for local control.

6. Tamper Resistance: The system needs a rugged, weatherproof enclosure, tamper alarms, and backup communication to resist physical interference.

7. Expandability & Customization: It should allow additional sensors or cameras, and include geofencing for auto arming/disarming based on GPS.

8. Autonomy & Smart Features: AI integration for object detection, and a self-learning algorithm to reduce false alarms based on typical movement patterns.

By focusing on power-efficient components, modularity, and robust communication options, the system can balance effectiveness with long-term deployment on battery power.

Takeaway:

In this case Chat GPT gives us good information on our direction with our product. Most of the features it suggested have already either been implemented into the design or looked at and deemed outside the scope of the project. One of these features being Night vision. While useful the technology adds a lot of cost and complexity that is too large for a prototype like this.

5.2 Case study 2:

Prompt: "Please provide the software architecture and design to approach such a project like a deployable security system, please include both code written on the gadget (embedded and python) and code written on the node(s) both embedded and python. Thank you!"

Outcome: To create a deployable security system involving gadgets and nodes, we can break down the software architecture and design into two main components: **Node** and **Gadget**. Each component will interact with each other via a network, and both will have specific responsibilities.

System Overview:

- **Nodes**: These ESP32-CAM devices will act as security monitors, capturing images or video and sending them to the central gadget. The nodes will operate over a network using secure communication protocols.
- **Gadget**: This acts as the main hub that processes the data sent from the nodes. The gadget will run Python code to process images and trigger any necessary actions like alarms or alerts based on the motion detected.

1. Node Design (ESP32-CAM)

Node Overview:

The nodes are ESP32-CAM devices placed at various security points. They capture images or video and send data to the gadget when triggered.

Key Tasks:

- **Image/Video Capture**: Using the ESP32-CAM camera.
- **Motion Detection**: Using an infrared (IR) sensor or onboard algorithm to detect motion.
- **Secure Communication**: Sending images securely to the gadget.
- **Periodic Heartbeat**: Sending regular updates to ensure the node is functional.

Node Code (Embedded - C++):

- **Libraries**: WiFi, WebServer, esp32cam, SSL, IR sensor.
- **Key Functions**:
    - o **WiFi Setup**: Connect to the local network using WiFi.
    - o **Image Capture**: Capture images at a specified resolution and send them to the gadget.
    - o **Motion Detection**: Detect motion using the IR sensor and trigger image capture.
    - o **Secure HTTP**: Use SSL to ensure secure communication when sending the captured data to the gadget.
    - o **Health Checks**: Send regular pings or heartbeats to ensure the node is active.

2. Gadget Design (Raspberry Pi or PC)

Gadget Overview:

The gadget serves as the central hub, receiving and processing images or videos from nodes, analyzing the data (motion detection), and providing output actions (alerts, alarms, etc.).

Key Tasks:

- **Image Reception**: Receiving images sent by the nodes.
- **Motion Detection**: Using a Python-based algorithm (OpenCV) to analyze images for any motion.
- **Alerts/Actions**: Trigger alarms or send notifications if motion is detected.
- **GUI**: A simple dashboard to view images and control the system.

Gadget Code (Python):

- **Libraries**: requests, OpenCV, Flask (optional for GUI), Socket, SSL.
- **Key Functions**:
    - o **Receive Image**: A Python web server that listens for incoming images from nodes.

- **Motion Detection**: Use OpenCV to process the received image.
- **Trigger Alarms**: Based on motion detection, trigger alarms or alerts.

3. Communication & Security

Data Flow:

1. **Node** captures an image using the ESP32-CAM when motion is detected.
2. The image is sent over HTTPS to the **Gadget**.
3. The **Gadget** receives the image, processes it for motion detection, and triggers alerts if needed.

Security:

- **SSL/TLS**: Secure the communication channel between the node and gadget.
- **Token Authentication**: Optionally, each node can use token-based authentication for additional security when sending data.

4. Image Processing Design

Algorithm:

- **Background Subtraction**: Compare the current image frame with a reference background to detect motion.
- **Thresholding**: Convert the image into binary for easy processing.
- **Contours Detection**: Use OpenCV to identify and highlight objects that have moved.

Summary

- **Nodes**: ESP32-CAM devices that capture images and send them securely to the gadget.
- **Gadget**: A central processor running Python, which processes images from the nodes and performs actions (motion detection, alerts).
- **Communication**: Secured via SSL, possibly with token-based authentication for added security.
- **Image Processing**: Uses OpenCV for detecting motion or analyzing images.

This approach balances hardware and software across both the embedded (Node) and application (Gadget) layers, ensuring secure and efficient operation.

Takeaway:

This response was very beneficial in the early stages of software development. It provided a very detailed guide of how to approach designing such a complex system and lead to validation of current ideas while also inspiring new ones. Validated ideas consisted of focusing on developing the image processing algorithms on the gadget and having the nodes send image captures over a secure communication message. New ideas inspired consisted of a heartbeat to ensure node functionality throughout the running program. This idea will let us be confident that our system is always working.

5.3 Case study 3:

Prompt: "How do I upload more than one file to an esp32 board when running code?"

Outcome: Using Arduino IDE

The Arduino IDE can also be used, but managing multiple files is a bit different.

Steps:

4. **Open Arduino IDE**: Make sure you have installed the ESP32 board package in the Arduino IDE.
5. **Create Tabs**:
   - To manage multiple files, create new tabs within the Arduino IDE.
   - Each tab will represent a file. You can create a .cpp file or a .h file by clicking the arrow button on the right side of the Arduino IDE and selecting **New Tab**.
6. **Add Your Files**:
   - Add your .cpp and .h files as needed.
   - The Arduino IDE will automatically compile all the files when you upload your project.
7. **Upload**:
   - Click the **Upload** button in the Arduino IDE to upload all the files to the ESP32.

Takeaway:

This response was very detailed and helpful as this was one of the main questions I had as to how we will implement software onto our board. From my experiences in Embedded Systems, I was knowledgeable in how to upload one code file to a board, however, I was not sure how to implement uploading multiple code files to a board, specifically to the ESP32 through our development platform Arduino IDE.

5.4 Case Study 4:

Prompt: "I'm creating a deployable security system that will be powered by lithium batteries. What other considerations should I have in terms of power?"

Outcome: Battery capacity and power requirements, power management circuits, solar integration, low power modes, Thermal considerations and battery longevity.

"When designing a deployable security system powered by lithium batteries, there are several critical power-related considerations you should take into account:

- **Battery Capacity & Power**: Calculate total power consumption and select batteries with enough capacity (mAh). Use energy-efficient components and optimize the ESP32-CAM's duty cycle.
- **Power Management**: Use buck/boost converters for stable voltage, battery protection circuits, and power monitoring to prevent overcharge/discharge and monitor usage.

- **Solar Integration**: Size the solar panel to match charging needs and use a solar charge controller to regulate input to the batteries.
- **Low Power Modes**: Utilize low-power or deep-sleep modes on components, and power down unused parts to extend battery life.
- **Thermal Considerations**: Ensure heat dissipation to avoid battery overheating during charging or heavy use.
- **Battery Longevity**: Plan for battery degradation over charge cycles and avoid charging in extreme temperatures.
- **Weatherproofing**: Use a weatherproof enclosure for outdoor deployments to protect the batteries and electronics.

Addressing these considerations will help you build a reliable and efficient power system for your deployable security system!" (OpenAI, 2024)

Takeaway:

This response is interesting because it remembers the past chat interactions about choosing to work with ESP32-CAMs without mentioning it in the prompt. ChatGPT correctly identifies many important aspects of powering our system. The most helpful feedback given was to consider how to dissipate heat to avoid damaging our electrical system. Because of this response, we will make efforts to create a ventilation system. Independent research within our group has addressed every other point made by ChatGPT.

5.5 Case Study 5:

Prompt: "I'm creating a deployable security system that uses cameras and sensors to detect threats, I want to integrate a mobile app for alerts and control. We are currently using an ESP32-CAM and ESP32-S3 for communication. What would be the best way to approach this?"

Outcome:

"To integrate a mobile app for alerts and control into your deployable security system using ESP32-CAM and ESP32-S3, here's a simple breakdown:

Backend Setup (Server for Alerts & Control):

Set up a lightweight backend (e.g., Node.js, Flask, or Firebase) to handle communication between the ESP32 devices and the mobile app.

The ESP32s will send data (like motion detection or camera images) to this server using HTTP requests or WebSockets.

The backend can trigger push notifications to the mobile app using services like Firebase Cloud Messaging (FCM).

ESP32 Firmware (Data Transmission):

Program the ESP32-CAM to capture images and detect motion. When a motion is detected or at regular intervals, send the data to the backend.

The ESP32-S3 can manage additional sensor data (e.g., temperature or IR sensors) and send updates to the server.

Use Wi-Fi for communication between the ESP32 devices and the server, sending data via HTTP POST requests or MQTT.

Mobile App (Alerts & Control):

Build a simple mobile app using Flutter or React Native for cross-platform compatibility.

Integrate the app with Firebase Cloud Messaging (FCM) or another notification service for real-time alerts when motion or threats are detected.

The app should also allow the user to control settings, view the camera feed (by pulling from the backend), and manage the system remotely (turning on/off sensors, adjusting settings, etc.).

Real-Time Video Streaming (Optional):

Set up real-time video streaming from the ESP32-CAM. This can be done by creating a web server on the ESP32-CAM that streams video, and then accessing the stream from the app through the backend.

This approach will allow for real-time alerts and system control via the mobile app, ensuring your deployable security system is functional and user-friendly."

, (OpenAI, 2024)

Takeaway:

The response provided a clear and actionable plan for integrating a mobile app with our deployable security system. It addressed essential components such as backend setup, data transmission, and mobile app development. The recommendation to use Flutter is particularly valuable, as extended research shows it simplifies the creation of clean, intuitive UI components. This guidance provides a solid foundation for establishing communication between our physical devices and the mobile app, enhancing the system's overall usability. We will incorporate these recommendations throughout our design and development process to ensure seamless integration and an improved user experience.

Chapter 6 – Hardware Design

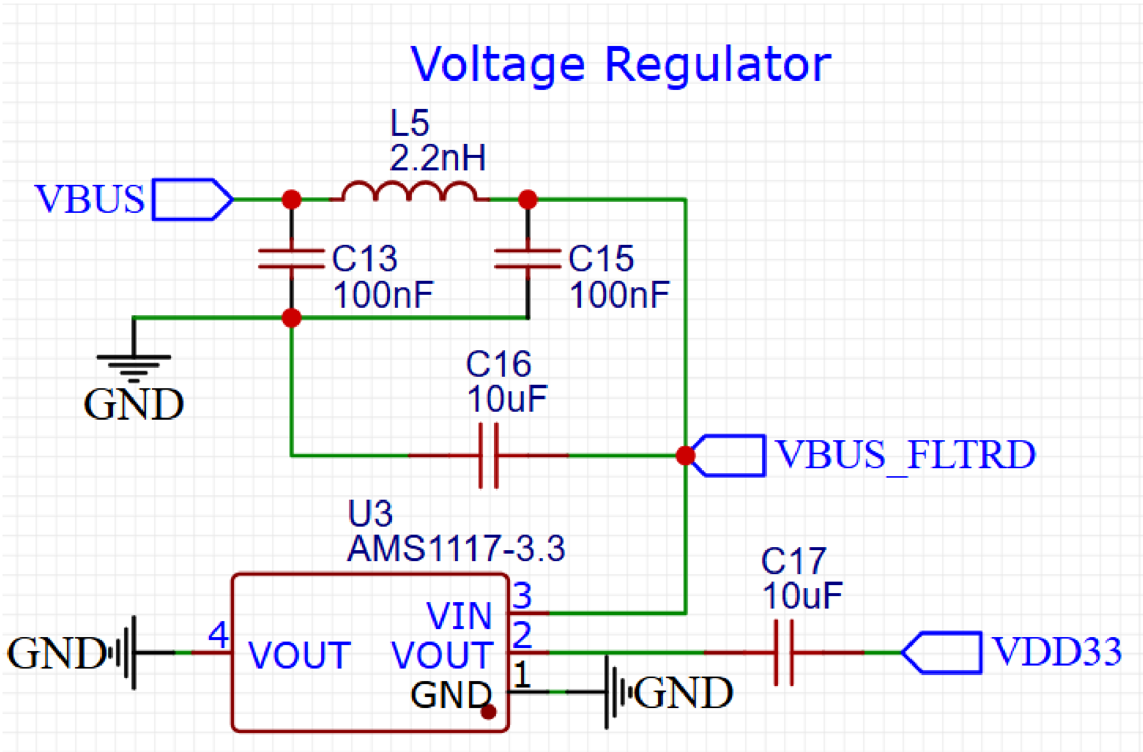
6.1 Hardware Design Introduction:

6.2 Gadget Design

6.2.1 Overview

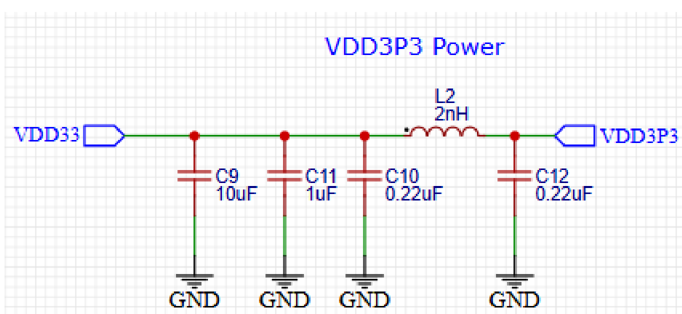
6.2.2 Power Distribution

6.2.2.1 Voltage Regulation



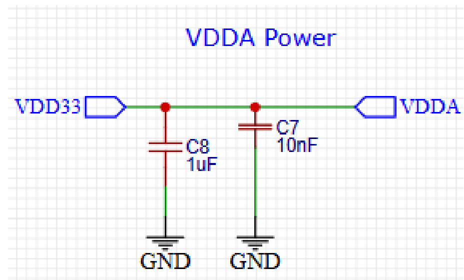
The AMS1117-3.3 ensures the ESP32-S3 receives a consistent 3.3V by stepping down the higher input voltage. Connected to ground by pin1 and pin 4 for more stable connection. Pin 2 is 3.3 voltage out. And pin three is unregulated voltage in from the USB-C.

6.2.2.2 VDD3P3 Power



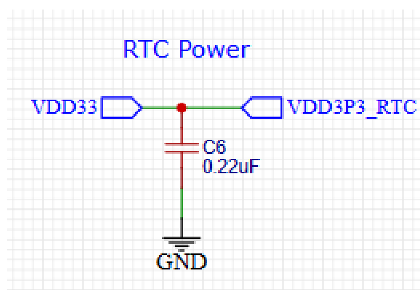
The VDD3P3 power is set up to capture any high frequency and unstable power. From the AMS1117-3.3 at VDD33. The capacitors, are used to capture and smooth power. These are commonly referred to as decoupling capacitors. While the indictor throttles amperage to also smooth power. Due to VDD3P3 powering many of the sensitive functions of the ESP32-S3 it is important to have smooth power.

6.2.2.3 VDDA Power



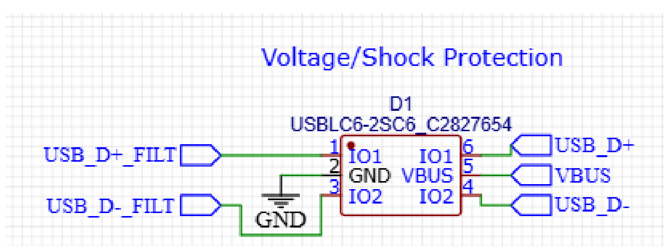
VDDA is the main power of the analog circuits of the ESP32-S3 at 3.3 v, given power from the AMS1117-3.3 at VDD33. This also has decoupling capacitors to ensure smooth power supply.

6.2.2.4 RTC Power



RTC (Real-Time Clock) power also is supplied by the AMS1117-3.3 at VDD33. RTC is important to a lot of the internal time keeping functions in the ESP32-S3. This also has a decoupling capacitor for smooth power supply.

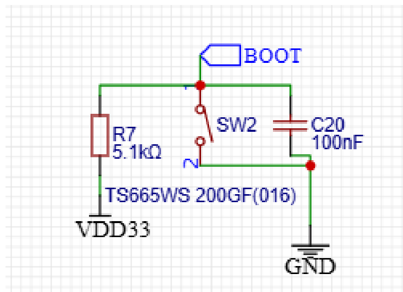
6.2.2.5 Voltage Shock Protection



The USBLC6-2SC6 is a transient voltage suppression (TVS) diode array to protect from voltage spikes and ESD. Here on the right side of the diagram pins 6 and 4 are unfiltered positive and negative data respectively. Pin 5 is a pin to guide any voltage spikes on the

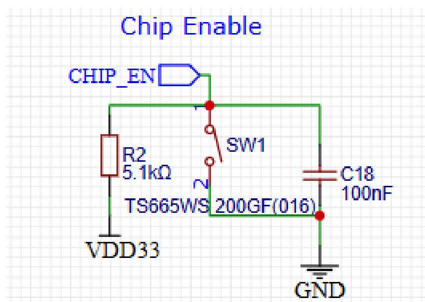
VBUS to ground, which is pin 2 for this part. Pins 1 and 3 are the filtered data from any voltage spikes.

6.2.3 Boot

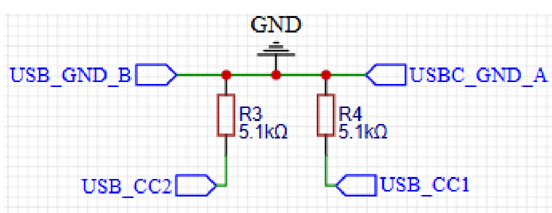


This set up allows us to put the ESP32-S3 into Boot mode or Flash mode. This lets us add new code onto the ESP32-S3. Added in resistor and capacitor to remove debounce and unwanted amperage.

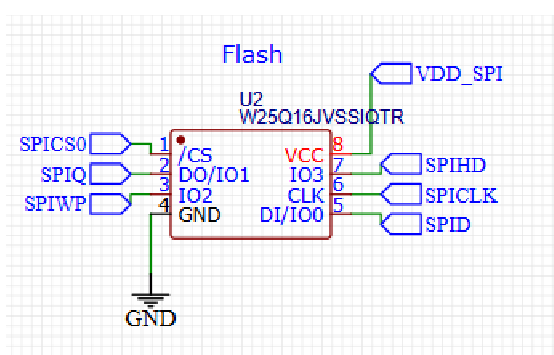
6.2.4 Chip Enable



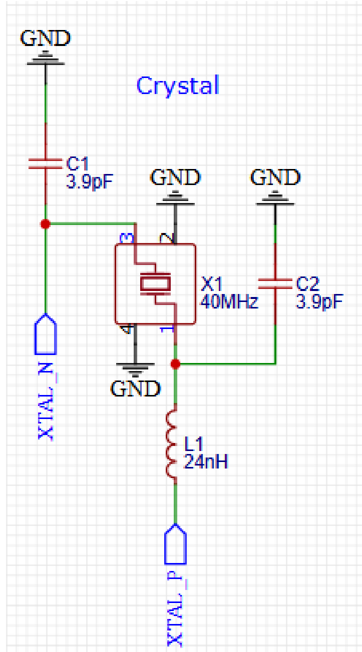
6.2.5 Common Ground



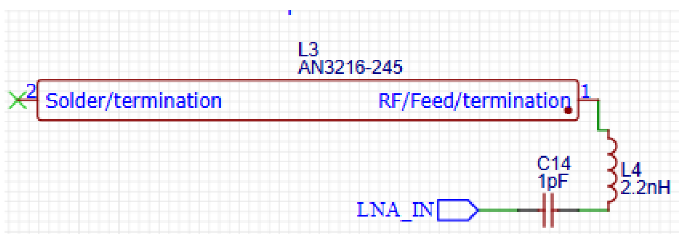
6.2.6 External Flash



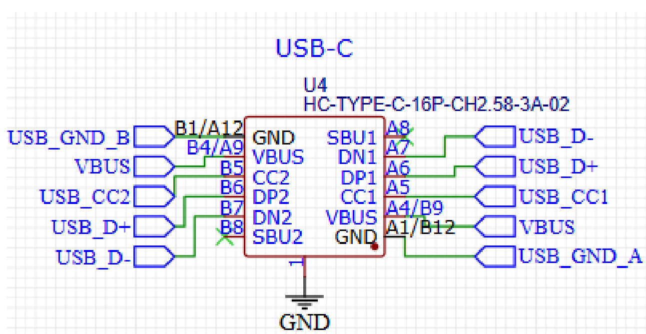
6.2.7 External Crystal



6.2.8 Chip Antenna



6.2.9 USB – C



6.2.10 Ethernet Port

6.2.11 External Memory

6.3 Node Design

Chapter 7 – Software Design

7.1 Software Design Introduction:

The software design chapter consists of detailed flowcharts, components/data structures, descriptions for our machine learning algorithm, class diagrams, and designs for the system and the communication protocols used. The primary function of our software inside the scope of our system is to provide effective interaction between our two main hardware components (Node & Gadget), interaction between our hardware and the camera/sensors, perform necessary data processing, and evaluate images captured from our ESP32-CAM (Node) using object detection algorithms through TensorFlow Lite. Shown in the subsections below, the system architecture is organized to manage several functionalities such as: Device and network initialization, data acquisition & storage, image processing using TensorFlow Lite, data transfer & communication, and interaction/processing sensor data through embedded code.

7.2 Software Design Flowchart:

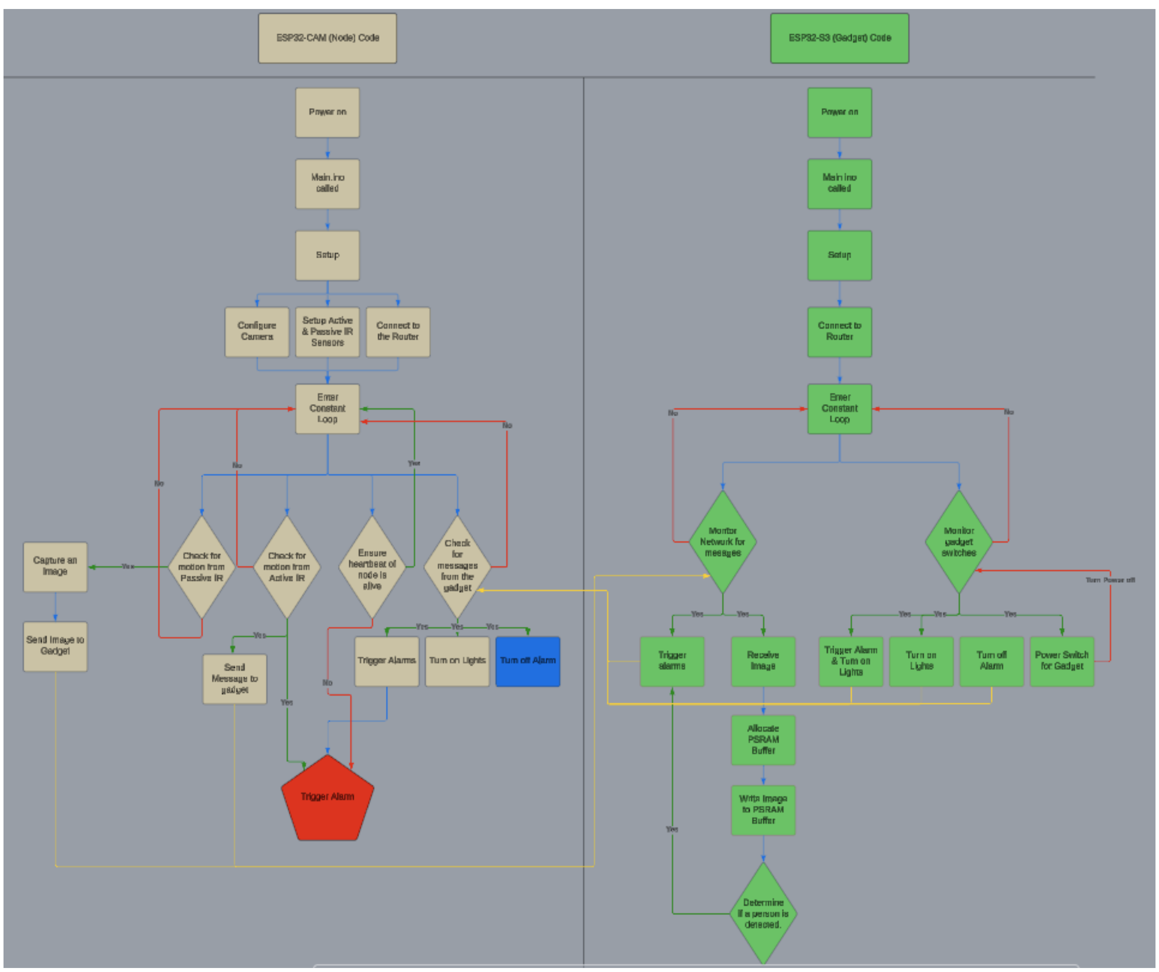


Figure 7.2 Software Design Flowchart created using Lucid Chart Software

7.2.1 ESP32-CAM Software Diagram Flowchart Explanation:

7.2.1 ESP32-S3 Software Diagram Flowchart Explanation:

7.3 Software Use Case Diagram:

7.4 Software State Diagram:

7.5 Software Structure Class Diagram:

7.6 Software Data Structures:

7.7 Software User Interface Design:

7.8 Software Data Transfer & Communication Protocol:

7.9 Software System Diagrams:

7.10 Machine Learning Algorithm Description:

Chapter 8 – System Fabrication/Prototype Construction

8.1 System Fabrication/Prototype Construction Introduction:

Chapter 9 – System Testing and Evaluation

9.1 System Testing and Evaluation Introduction:

Chapter 10 - Administrative Content

In this chapter we will go over our budgeting process and the pacing of our project. Our purchases will be included as well for our protype and the final system.

10.1 Budget/Cost Estimated:

The budget is self-imposed and was decided with every member's agreement. We will limit ourselves to $1,000 to incentivize our group to choose every part carefully. We want high performance only when needed, and not any costly unnecessary features.

Because the budget of our group is self-imposed, we also understand that our estimations may be off and that some parts may need to push our budgets boundaries. Our PCB budget calculation will include the cost to print the boards, and the parts on the PCB. Our system will have peripherals that cannot be attached to a PCB, thus having their own budget section.

Section	Budget
Gadget Peripherals	$ 100.00
Node Peripherals	$ 750.00
PCBs	$ 150.00
Total	$ 1,000.00

Table 10.1: Budget/part cost table.

10.2 Bill of materials:

Bill of Materials				
Part	**Suggested Part**	**Quantity**	**Cost / per**	**Cost Total ($)**
esp32-cam	TBD	2	22.5	45.00
Gadget batteries	TBD	24	~6	144.00
Node batteries	TBD	24	~6.25	150.00
pcb 1 build		~3	~30	90.00
PCB 2 build		~1	~30	35.00
passive IR	TBD	1	10	10.00
active IR	TBD	1	6	6.00
LEDs	TBD	1	10	10.00
speaker	TBD	1	10	10.00
Housing nodes		3	25	75.00
housing gadget		1	25	25.00
wire (22 awg)	TBD	1	15	15.00
SSD		1	60	60.00
Micro Router	TBD	1	45	45.00
Gadget Switch	TBD	TBD	.47	TBD
Total Cost				$ 720.00

Table 10.2: Bill of materials.

For this project, the development cost is steep regarding early prototype design. This analysis focuses on purely the material needed to do a first build. For this cost analysis, we have disregarded the tools and equipment needed for the development and any disposable items. This is to show the price breakdown for each part, and the total for a single build.

10.3 Project Milestones for SD1:

Task	Deadline	Status
Group Formation	Aug 22nd, 2024	Completed
Idea Brainstorming	Aug 22nd – 24th, 2024	Completed
Idea Selection	Aug 24th , 2024	Completed
Design Software Architecture	September 5th , 2024	Completed
Kickstart Meeting	September 6th , 2024	Completed
Divide and Conquer Document	September 7th , 2024	Completed
Divide and Conquer Meeting	September 10th , 2024	Completed
Begin Software development	September 15th,2024	Completed
Divide and Conquer Document Revision	September 27th, 2024	Completed
Software Hardware Compatibility Test #1	October 10th , 2024	Completed
Hardware Prototype #1	October 10th , 2024	Completed
Software Image Processing Algorithm development started	October 10th , 2024	Completed
Software Hardware Compatibility Test #2	October 24th, 2024	Completed
60-Page Draft Report	October 25th, 2024	Completed
60-Page Draft Report Revision	November 8th, 2024	Completed
Final Report	November 26th, 2024	Incomplete
Mini Demo Video	November 26th, 2024	Incomplete

Table 10.3: Project Milestones for SD1.

10.4 Project Milestones for SD2:

Task	Deadline	Status
Image processing Algorithms complete	February 1st, 2025	Incomplete
All Hardware Parts Ordered	February 1st , 2025	Incomplete
Software Development Complete	February 20th, 2025	Incomplete
Software integration with hardware	March 1st , 2025	Incomplete
Testing and Revision Phase 1	March 1st, 2025	Incomplete
Testing and Revision Phase 2	March 15th, 2025	Incomplete
Testing and Revision Phase 3	April 1st, 2025	Incomplete
System Integration complete	May 15th, 2025	Incomplete
Project completion	May 20th, 2025	Incomplete

Table 10.4 Project Milestones for SD2.

Chapter 11 – Conclusion

Citations, Research, and Resources:

Appendix A – References

[1] Wikipedia contributors. (2024, August 22). Digital image processing. In *Wikipedia, The Free Encyclopedia*. Retrieved 19:18, August 30, 2024, from https://en.wikipedia.org/w/index.php?title=Digital_image_processing&oldid=1241614779

[2] Wikipedia contributors. (2024, August 13). Signal processing. In *Wikipedia, The Free Encyclopedia*. Retrieved 19:19, August 30, 2024, from https://en.wikipedia.org/w/index.php?title=Signal_processing&oldid=1240021905

[3] Hulppre Solar Outdoor Motion Sensor Alarm-Dog Barking&Gunshot sound, support recording,130db loud noise maker,13 kinds of sound strobe light with USB,say goodbye to the intruders - amazon.com. (n.d.). https://www.amazon.com/HULPPRE-Outdoor-Alarm-Dog-Recording-Intruders/dp/B0C5HMS1PN

[4] *Understanding conservation*. National Wildlife Federation. (n.d.). https://www.nwf.org/Educational-Resources/Wildlife-Guide/Understanding-Conservation

[5] Wikipedia contributors. (2024, October 17). C++. In *Wikipedia, The Free Encyclopedia*. Retrieved 14:22, October 17, 2024, from https://en.wikipedia.org/w/index.php?title=C%2B%2B&oldid=1251670103

[6] Wikipedia contributors. (2024, October 16). Python (programming language). In *Wikipedia, The Free Encyclopedia*. Retrieved 14:26, October 17, 2024, from https://en.wikipedia.org/w/index.php?title=Python_(programming_language)&oldid=1251534510

[7] Wikipedia contributors. (2024, October 11). Arduino. In *Wikipedia, The Free Encyclopedia*. Retrieved 14:28, October 17, 2024, from https://en.wikipedia.org/w/index.php?title=Arduino&oldid=1250690945

[8] Wikipedia contributors. (2024, October 16). JavaScript. In *Wikipedia, The Free Encyclopedia*. Retrieved 21:01, October 18, 2024, from https://en.wikipedia.org/w/index.php?title=JavaScript&oldid=1251534737

[9] Wikipedia contributors. (2024, February 19). OpenCV. In *Wikipedia, The Free Encyclopedia*. Retrieved 20:57, October 18, 2024, from https://en.wikipedia.org/w/index.php?title=OpenCV&oldid=1208982530

[10] Wikipedia contributors. (2024, October 15). TensorFlow. In *Wikipedia, The Free Encyclopedia*. Retrieved 21:11, October 18, 2024, from https://en.wikipedia.org/w/index.php?title=TensorFlow&oldid=1251402385

[11] Wikipedia contributors. (2024, October 11). HTTP. In *Wikipedia, The Free Encyclopedia*. Retrieved 21:26, October 18, 2024, from https://en.wikipedia.org/w/index.php?title=HTTP&oldid=1250568137

[12] Wikipedia contributors. (2024, October 10). WebSocket. In *Wikipedia*, *The Free Encyclopedia*. Retrieved 21:32, October 18, 2024, from https://en.wikipedia.org/w/index.php?title=WebSocket&oldid=1250439164\

[13] Wikipedia contributors. (2024, September 29). Flutter (software). In Wikipedia, The Free Encyclopedia. Retrieved 22:30, October 21, 2024, from https://en.wikipedia.org/w/index.php?title=Flutter_(software)&oldid=12485147884

[14] *What is react native? Beginner's Guide + FAQ*. Coursera. (n.d.). https://www.coursera.org/articles/what-is-react-native?msockid=1a27620d21d963e41447771e204962fd

[15] Wikipedia contributors. (2024, July 9). Ionic (mobile app framework). In Wikipedia, The Free Encyclopedia. Retrieved 22:35, October 21, 2024, from https://en.wikipedia.org/w/index.php?title=Ionic_(mobile_app_framework)&oldid=1233499145

[16] Wikipedia contributors. (2024, September 27). Xamarin. In Wikipedia, The Free Encyclopedia. Retrieved 22:35, October 21, 2024, from https://en.wikipedia.org/w/index.php?title=Xamarin&oldid=1248078093

[17] GeeksforGeeks. (2024, July 3). *Aws Lambda*. https://www.geeksforgeeks.org/introduction-to-aws-lambda/#

[18] GeeksforGeeks. (2021, July 15). *Firebase - Introduction*. https://www.geeksforgeeks.org/firebase-introduction/

[19] GeeksforGeeks. (2022, July 15). *Getting started on Heroku with python*. https://www.geeksforgeeks.org/getting-started-on-heroku-with-python/

[20] Wikipedia contributors. (2024, July 26). DigitalOcean. In Wikipedia, The Free Encyclopedia. Retrieved 22:41, October 21, 2024, from https://en.wikipedia.org/w/index.php?title=DigitalOcean&oldid=1236707123

[21] Google. (n.d.). *Firestore documentation | google cloud*. Google. https://cloud.google.com/firestore/docs/

[22] Wikipedia contributors. (2024, August 5). MongoDB. In Wikipedia, The Free Encyclopedia. Retrieved 22:43, October 21, 2024, from https://en.wikipedia.org/w/index.php?title=MongoDB&oldid=1238821960

[23] *W3schools.com*. W3Schools Online Web Tutorials. (n.d.). https://www.w3schools.com/postgresql/index.php

[24] Supabase. (2024, October 21). *Database*. Supabase Docs. https://supabase.com/docs/guides/database/overview

[25] *Why you should not charge a lithium battery below 32 degrees*. REDARC. (2022, September 19). https://www.redarcelectronics.com/us/resources/chargers-isolators-faqs/do-not-charge-lithium-battery-below-32-degrees/

[26] *Taper*. Taper | Analog Devices. (n.d.). https://www.analog.com/en/resources/glossary/logarithmic_linear_taper.html#:~:text=When%20used%20in%20an%20amplifier,used%20for%20audio%20volume%20controls.

[27] *A comprehensive guide to PMIC*. FPT Semiconductor .,JSC. (2024, February 22). https://fpt-semiconductor.com/blogs/a-comprehensive-guide-to-power-management-integrated-circuit-pmic/

[28] *Understanding conservation*. National Wildlife Federation. (n.d.). https://www.nwf.org/Educational-Resources/Wildlife-Guide/Understanding-Conservation

[29] *Battery voltage*. PVEducation. (n.d.). https://www.pveducation.org/pvcdrom/battery-characteristics/battery-voltage

[30] *Different types of batteries for electronic products (Importer's Guide)*. Sofeast. (2021, January 6). https://www.sofeast.com/resources/different-types-of-batteries-for-electronic-products-guide/

[31] Battery University. (2021, December 3). *BU-107: Comparison table of secondary batteries*. https://batteryuniversity.com/article/bu-107-comparison-table-of-secondary-batteries

[32] How many months of a lithium polymer battery life. (n.d.). https://www.lipolbattery.com/lithium-polymer-battery-life.html

[33] *The fundamentals of LDO Design and Applications*. The Fundamentals of LDO Design and Applications | Analog Devices. (n.d.). https://www.analog.com/en/lp/001/fundamentals-of-ldo-design-and-applications.html#:~:text=A%20low%20dropout%20regulator%20(LDO,is%20recommended%20to%20ensure%20stability.

[34] *When should you choose LDO or Buck Converter?*. CircuitBread. (n.d.). https://www.circuitbread.com/ee-faq/when-should-you-choose-ldo-or-buck-converter#:~:text=A%20buck%20regulator%20is%20preferred,converter%20is%20the%20ideal%20choice.

[35] *The benefits of a compact power management IC and Power Loss Protection*. element14 Community. (n.d.). https://community.element14.com/technologies/power-

management/b/blog/posts/the-benefits-of-a-compact-power-management-ic-and-power-loss-protection

[36] Nqtronix, & Instructables. (2017, September 24). *Practical guide to leds 1 - pick your led!*. Instructables. https://www.instructables.com/Practical-Guide-to-LEDs-1-Pick-Your-LED/

[37] Ou, M. (2024, June 18). *The ultimate guide to cob LED STRIP*. MSHLED. https://www.stripsledlight.com/the-ultimate-guide-to-cob-led-strip/

[38] Ibe. (2024, September 14). *Guide to light-emitting diode (LED) : Types, application and manufacturers*. IBE Electronics. https://www.pcbaaa.com/light-emitting-diodes-led/

[39] Justin. (2024, June 12). *SMD vs. COB LED STRIP: A comprehensive comparison 2024 - led controller manufacturer, supplier, factory: Led Smart Light Supplier: Led Strip Lights Supplier*. LED Controller Manufacturer, Supplier, Factory| LED Smart Light Supplier | LED Strip Lights Supplier. https://onesmartlighting.com/blog/smd-vs-cob-led-strip/

[40] Ou, M. (2023, March 13). *IP rating: The definitive guide -MSHLED lighting*. MSHLED. https://www.stripsledlight.com/ip-rating-the-definitive-guide/

[41] Patel, S. (2023, November 20). *What is through Hole Technology*. Candor Industries. https://www.candorind.com/blog/what-is-through-hole-technology/

[42] *Piezo buzzers vs. Magnetic Buzzers: APC International*. americanpiezo. (2023, November 27). https://www.americanpiezo.com/blog/piezo-buzzers-vs-magnetic-buzzers/

[43] *Piezo vs. magnetic buzzers: Which to choose?:news*. FHD Electronics Corporation. (n.d.). https://fhdmfg.com/news/news_detail/magnetic-buzzer

[44] ChatGPT assisted text and formating; https://chatgpt.com/

[45] *How do speakers work: A guide on how speakers produce sound*. soundcore. (n.d.). https://www.soundcore.com/blogs/speaker/how-do-speakers-work-speakers-and-sound-production-explained

Appendix B – Copyright permission

- ChatGPT
  - ChatGPT was utilized throughout the document to review and refine written paragraphs, enhancing their clarity and flow.
- Lucid Flowchart
  - Lucid Flowchart was used to make the software diagram.

   - As stated in Permission for academic purposes . Since I have created the diagram I am free to use it however I like, mentioning the software used to make it is completely optional.
- Draw.io
   - Draw.io was used to make the hardware diagram.
- HULPPRE
   - Used an illustration to provide reference of our competitors

Figure 2.4 Requested:

YOUR CONVERSATION WITH
HULPPRE Solar

Sep 10, 2024 9:51 AM

Hello, I would like to ask for permission to use an image of the product:
https://www.amazon.com/HULPPRE-Outdoor-Alarm-Dog-Recording-Intruders/dp/B0C5HMS1PN?ref_=v_sp_product_dpx
as a reference in my senior design project.
I am an electrical engineer student and I'm designing a similar project and would like to include your product's image in my design document.
Please email [e-mail address removed] if you decide to give permission to my group. Your email and the product will be given proper citation in our design document. Please feel free to email me if you have any questions about our project.
Thank you for your time.

HW Diagram Requested:

Email confirmation to *

ph626126@ucf.edu

Summary *

Request for permission

Suggested articles

HOW DOES **REQUEST** AN APP INSIDE CONFLUENCE WORK?

As a user in Confluence you can **request** that an app is installed in the Confluence interface. This **request** goes to the administrator(s) of that instance.

CVE-2022-1575 FAQ FOR DRAW.IO FOR CONFLUENCE AND JIRA

A non-privileged user does not have the write **permissions** needed for the attack.

GENERATE A QUOTE FOR DRAW.IO FOR CONFLUENCE CLOUD (ANNUAL BILLING)

" or "**Request** an annual quote ", depending on which option you see.

Description

Normal text

Good afternoon,

My name is Phillip and I'm a student at the University of Central Florida. My group and I are working on our senior design project and used your product to create a diagram that will be included in our project's report. We would like to ask for permission to use your product in our design, and to include your acceptance email into our project with your permission.

If you have any questions or remarks please send me an email at ph626126@ucf.edu.

Thank you for your time.

HW Diagram Accepted:

N/A