UCF SENIOR DESIGN 2

Smart Surveillance Hub

Department of Electrical Engineering and Computer Science
University of Central Florida
Dr. Lei Wei
Final Project Document and Group Identification
Divide and Conquer

Group 15

Matthew Weinert	Computer Engineer
Korey Lombardi	Computer Engineer
Kenneth Ancrum	Electrical Engineer
Joshua Sherrill	Electrical Engineer

Table of Contents

Pages

1.0 Executive Summary

Privacy and security are a standard in today's society, and the ability to keep yourself and your family safe should be accessible and affordable. According to the FBI’s Uniform Crime Reporting (UCR) Program, the number of burglaries has decreased 48.5 percent from 2010 to 2019. We believe the decrease in burglaries is directly related to the increase in IoT devices. With such an increase in IoT devices, there’s been a vast growth in the number of cost-effective ways to secure your home. A visible camera displayed in the open will most likely scare off a burglar who realizes they are being recorded.

Security companies tend to fully integrate homes with alarms and sensors, building a very complex system. Also, these types of systems usually won’t alert you until an intruder has broken into your home and set off the alarm. By the time an intruder has entered the home, it might be too late to respond accordingly. If you were alerted before the burglar entered your home, you might have time to call the police, hide somewhere, possibly scare the intruder off, or even get ready to defend yourself accordingly.

It used to be nearly impossible to have your home secured without spending a massive amount of money on unwanted features. Nowadays, having a security company install a fully-integrated system comes with an extremely high cost, and it usually includes a contract that consists of installation fees and continuous subscriptions. For example, ADT has a similar product with more features, but it also has an extensive contract and installation fees. The system they provide doesn't allow you to use your own personal IP camera, so you have to spend extra money on their marked-up cameras. If you aren’t interested in extra features that don't fit your needs or you don’t want to be locked into a contract, you really just need an IP camera to solve your problem.

We believe people of all incomes should be able to afford an alarm system without being obligated to a contract. It is also important for people with big homes or lots of property to be able to add as many cameras as necessary to secure their property. With our product, it is possible for people to afford to install multiple cameras, providing their homes with excellent security.

However, it is not typical for the average individual being able to install their own IP cameras and viewing the camera’s footage from inside their home. Our system is designed in a way that the user of our application can simply input their camera's configuration details and begin viewing its footage. Since configuring the product is such a simple task, our product is accessible to a broad range of audience.

Our project's main task is to provide safety for individuals or families when they are located inside their homes and provide them with enough time to respond accordingly. Whether the user wants to have a bird's eye view of their front door simply or if they're going to use it to identify potential intruders, it's a simple way to be alerted without being pestered. Although our product will secure the homes of users while they are located inside their homes, it won't provide them with footage when they are not located inside their homes. This is a feature that could be added to our project in the future to provide a higher level of security.

For users with big homes, our product can save people time from walking all the way across their house to see who is at the door. It could also help parents see who their kid's friends are that come over without physically going to the door to greet them. Parents will also be able to tell if their kids leave the house or show up at the home when they're not supposed to. The elderly could also benefit from the product because they might not hear or see as well as a younger person, and the hub will provide a better view of those that arrive at their house compared to a peephole attached to a front door. Anyone in a wheelchair now has the potential to see who is at their front door in a simple manner. There is a wide range of people that could benefit from using Security Smart Hub to make their home secure in a customizable and straightforward fashion.

2.0 Project Description

The problem with products in today's market is they include too many features, and the prices are way too high. Our product will target the customer who is looking for a human detection product, who is looking to have a single camera or multiple cameras installed at a low cost. Since most security companies generally require you to use their cameras that are usually marked up in price and they typically charge an installation fee, we believe in having a solid market advantage. Our main goal is to build a product for the average person who understands technology and wants to buy an affordable product. We made it possible for users to avoid enrolling in subscriptions that most security companies include with their products. Our product will be inexpensive because it won't require a technician to provide setup and installation. The Security Smart Hub is meant to provide safety and comfort for users while they are located inside their homes. Although the first design of our project is intended to provide security for the user while they are home, in the future, we hope to make the system accessible from a remote location. This is one of our stretch goals and most likely won't be implemented in the first version of our product due to time constraints. The design will be modular and straightforward, making it easy to add and remove cameras from its current configuration.

2.1 Project Motivation and Goals

Security Smart Hub's primary focus is to provide a high-quality, affordable, and flexible product.It will be connected through wifi to a hub that streams real-time video that you'll have access to through a clear unpixelated LCD display. Whenever the IP camera detects any human presence on your property, you will be alerted through the LCD display that's attached to the hub. The hub and display will be located inside your home, and the IP camera will send the video to the central hub to be processed. We used use facial recognition to determine if somebody is on or around your property. The different types of cameras that can be configured for use will be incredibly flexible. It will be configurable to run on the smart hub as long as it's an IP camera. This is extremely useful if you want to replace your active camera with another one due to malfunction or you simply want to upgrade to a higher quality IP camera.

Many IP cameras today only come with an app and no onsite video viewing method, meaning that if someone is not staring directly at their phone as a burglar attempts to break into their home then they will have no knowledge. With the smart hub the home owner will be notified the moment anyone comes into view of their camera via a led flashing as well as a loud noise coming from the hub. The owner can then view the hub and understand the situation. Another feature of this device is onsite storage as well, another issue with the IP cameras such as Ring, is they do not offer continuous recording storage. They offer cloud storage but it comes at a monthly subscription cost. With the Smart Hub they will be able to have their data stored onsite and be able to cycle through a month of recording to make sure no matter what happens they know their home will be protected.

By utilizing low-cost yet effective IoT devices, we provided an affordable way of keeping a close eye on your property. Since fully integrated security systems are so expensive, we plan to save you money, whether you're a lower-class, middle-class, or upper-class individual. Surveillance Smart Hub is perfect for you if you'd like to have live footage of your property sent to you in real-time. Our product will be affordable, flexible, and simple yet very effective.

2.2 Objective

Our overall objective is to give the user all the power and not the system. Security is essential because it gives people the ability to prevent dangerous situations and be safe. Many current systems are not priced friendly for everyone, which is an issue. Safety and security should not be only for those who can afford them. Our design aims to give the user everything they need to use whatever camera they want and be safe while observing the camera feeds or being away from them. Our specifications go into a bit more detail on the desired deliverables from the system while ultimately being affordable and customizable to everyone. By building our design with efficient and low-cost components, it is possible to provide security for users of all types of incomes. It will also benefit business owners who want to secure their business more realistically than by subscribing to a well-known security company that will charge them more than necessary. We feel that everyone from all walks of life and ranges of income should feel safe and confident in their home without paying large amounts of money.

2.3 Requirements Specifications

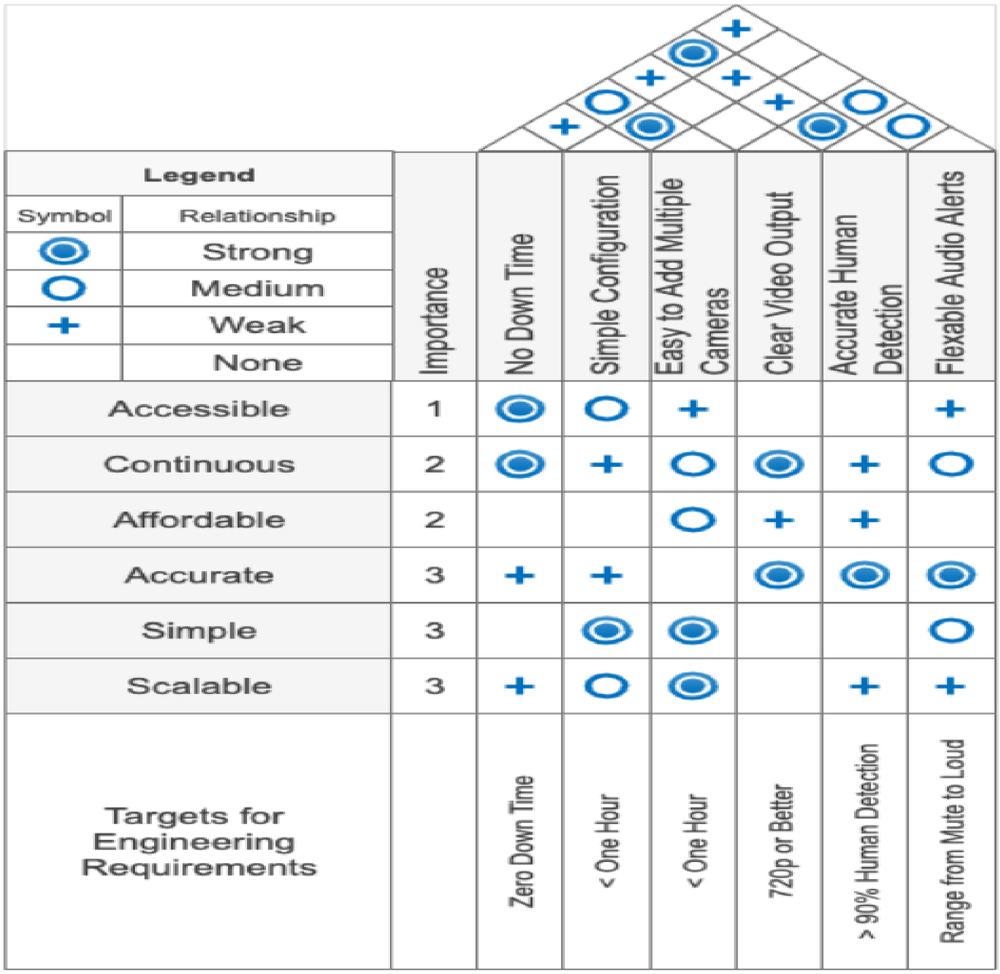
Specification Number	**Specification Description**
1	Users will have access to an UI application to interact and view the data sent by the IP camera.
2	Smart hub's desktop application will be able to interface with any generic IP camera within the same network. With a connection time of under 5 seconds.
3	To produce a sound queue with a sensitivity rating of 80dB or more
4	A system that is able to run the application that will act as a "hub". It will utilize A.I. to detect humans in up to 5 seconds

5	Parent desktop application will be able to display the live streamed video feed on a LCD screen that will connect via DSI. This application will also save recorded video feed onto a storage device.
6	A PCB that will receive data from the hub and notify the user with audio on the object the IP camera detects.
7	LCD that will be configured to the 3d printed case that will be able to run the desktop application as an alternative to using a monitor. With a weight of less than 5 pounds.
8	Lighting that will utilize the results of the A.I. detection to act as a security threat ‘mood light’. Depending on the level of security threat, the light will switch colors. (Red = threat or Blue = no threat)

Table 1: Specifications

2.4 House of Quality

The house of quality is a product-building matrix. Its goal is to ultimately show the strengths and weaknesses in different aspects of the system and product in terms of accessibility, continuity, affordability, accuracy, simplicity, and scalability. These things can be overlooked and easy to forget in the development of the product, but here we can think about and discuss these strengths and weaknesses through the house of quality chart. The most critical parts of our design consist of it having no downtime, cameras can easily be added to the system, and it is producing clear video output. Other important aspects of our design are represented in the table below. Using a house of quality makes it easier for our team to identify the essential elements of our overall systems design. We built our project to accommodate the house of quality diagram created below. When our project is finished being built, it will give us a strong indication of the system's quality by comparing its functionality with the house of quality we produced.



Legend	
Symbol	Relationship
◉	Strong
○	Medium
+	Weak
	None

	Importance	No Down Time	Simple Configuration	Easy to Add Multiple Cameras	Clear Video Output	Accurate Human Detection	Flexable Audio Alerts
Accessible	1	◉	○	+			+
Continuous	2	◉	+	○	◉	+	○
Affordable	2			○	+	+	
Accurate	3	+	+		◉	◉	◉
Simple	3		◉	◉			○
Scalable	3	+	○	◉		+	+
Targets for Engineering Requirements		Zero Down Time	< One Hour	< One Hour	720p or Better	> 90% Human Detection	Range from Mute to Loud

Figure 1:House of Quality

2.5 Main Goals,Advanced Goals,Stretch Goals

We split or goals into three categories of Main, Advanced and Stretch. Our main golas were goals the were essential to our project and absolutely needed to function in order for our project to meet its end goal. Advanced goals were Other function that increased the functionality of or project and what we considered to be must have for a good user experience, these goals were more challenging overall. Stretch goals are a group of added features and concepts that we would have wished to include but do to complexity and time constraint were not included in the final build.

2.5.1 Main Goals

- **Funtionig AI detection system**

To having a function trained AI model capable of detecting Humans even unlikely and un ideal scenarios as stated in our specification the model should being bale to identify and send audible alerts within 5 seconds.

- **Function user application**

To have a functional user application that can be easily navigated on and touch screen LCD display. The application is responsive and has very low latency making it highly user friendly. The objects and on screen are very noticeable and easy on the eyes for user expirence.

- **Functioning Notifcation system**

To have a have a noticeable and alertive notification system for users to engage with. When a human is detected a human should be alarmed and notify with urgency. The Sound notification should be biuld to off a Sound sensitivity rating of more than 80dB capable of filling a house space.

2.5.2 Achieved Advanced Goals

- **Video Control Features**

Providing user with playback features when viewing past records and alerts. These buttons will give users control of their past streamed feed allowing for more access and recording usage. Buttons include:

  - Fast Forward
  - Rewind
  - Play/Pause

- **Multi Camera Capability**

To give users the ability to access multiple RTSP streams through their hub allowing for additional cameras. This would in crease the level of security as users could have multiple viewing angle of their property or living space and recieve ques for ever time someone was detected.

- **Different Tones**

To give user the customization to choose what audio file they would like to be alerted to. Some sound could be to harsh or obnoxious so giving users a variety of set audio sample would increase the user expierence.

2.5.3 Stretch Goals

- **Cloud storage capability**

Storing data that was recorded from the cameras in the hub and transferring all the recordings to the cloud would be a great addition to the system and allow for a greater product. Time is the main constraint when dealing with this task. There could be a variety of options for the user for the amount of storage available and how long they would have access to the recordings stored on the cloud.

- **Mobile application**

Developing a mobile application that will be able to stream current camera footage. Also being able to see the recordings on the cloud storage. This would be able to access outside of the local network of the IP camera and hub. If the mobile application is built with a framework similar to React Native, the mobile application will be available on both Android and Apple devices.

- **Backup battery**

A backup battery in the hub would allow the hub to be more mobile for the customer to carry around and still be able to have access from a non-stationary location. It would also save customers money on their home's energy bill if they decided to charge the batteries elsewhere. The main constraint when designing for batteries would be the effect it has on the systems overall schematic.

- **Developing our own IP camera**

Creating our own IP camera with off the shelf devices is the ultimate stretch goal. It would allow the product to become a complete system with recording and storing as well as smart notification services. Time and complexity of this task is what makes this a stretch goal. If the hub is completed in a timely manner then this will be the first thing we try to develop. Since we only have three months to build the system this is not a realistic implementation for the project's first sprint. However, once the project is presented, this is something we could add to make our project a marketable product on the market.

- **Adding increased functionality to control other types of IP Cameras**

Cameras like PTZ cameras come with extra features to pan tilt and control zoom. Allowing access to these features through our hub would give the user more control over how they would like to use their device. This is also a design goal that would make our project more marketable and increase its functionality. The problem we run into when trying to implement this is we run into a time constraint. Although this feature would make our project more unique and provide more functionality, we don't believe it can be done in a three month time period.

- **Ability to cast received data**

As a higher-level feature, we would also like to include the ability to cast data received from the camera to give users the option to see the video without a hardwired cable connected to their display. To implement this feature, we would need advanced software that provides the system with the necessary drivers and protocols to cast the data to an external device that is not connected to the LCD display that is attached to the hub.

- **Adding a microphone to communicate with the person at the door**

Our project's design is set up to send users alerts when someone is at their front door or elsewhere on their property. It would be a useful additional feature if the person located inside could communicate with the person at the door by clicking a button on the hub. The user could click on a button through the hub that activates the microphone, giving the person from within the house the ability to tell the person they need to leave or communicate any other message they may want the person at the door to hear.

- **Screenshot capability**

A great feature that would take the project to another level is the ability for the system to screenshot when a person is detected. It would then save the photo to a SD and allow the user to view the pictures of the people detected on or near their property. This is a stretch goal because of the time constraint. If the system is completed and prototyping works as expected this will be one of the first features we add to the system. This feature alone will set us apart from a lot of other systems. Most systems record clips of motion then the user has to find the motion and take a snapshot of the person. With this feature the system will do it for them and simply allow them to have the information faster and with less hassle to the user.

- **Ability to speak over device for added control**

Another feature that would be a great addition in the future would be the added capability to speak over your system. Such as when you have been alerted to someone in your area you can not only view the alert through the app but speak over it as well to either converse with authorities or ward off any intruders. This feature is most commonly referred to as two way audio and has been a well accepted feature in other flagship devices such as Ring and Vivint. And has been extended to not just cameras, but doorbells and other outside peripherals as well.

2.6 Project Hardware and Software Design Details

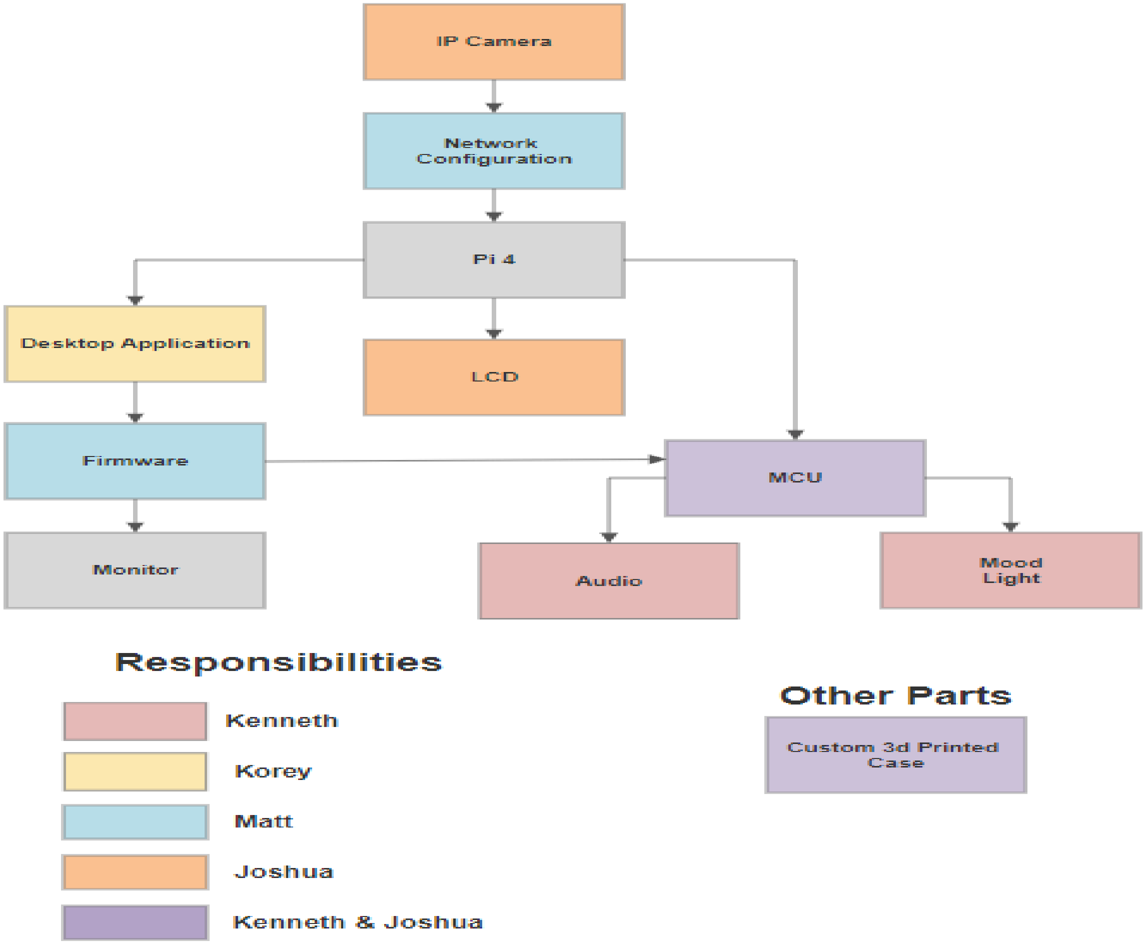


Figure 2: Hardware Block Diagram

This diagram shows the highest level interaction between the hardware and software along with which group member is responsible for what tasks. If there are different layers of views, this would be a 10,000 foot view from the sky making this the highest level view of how things interacted. Starting from the top, the ip camera is going to be the first item that does anything in this scheme. The camera is the first point of data entry where as it records the camera's data and then encodes it with an H.264 video codec. This is done to compress the data so that the throughput on the network isn't affected and that it can go to the destination quickly and smoothly.

Upon video compression from the codec, the desired network configuration is now going to handle the data and send it over the network. The network configuration will require everything to be within the same network in order to operate. We usined localhost and websockets in order to communicate with different devices on the network and to display the data from one point of origin to the other.

The network will be the interface between the raspberry pi, which will be our smart hub's main part, and the ip camera, which is the independent off-the-shelf camera that will be configurable to the smart hub as long as it comes with the ability to obtain an RTSP stream which is later discussed. Most, if not all, IP cameras have RTSP streams because that is the industry standard for cameras to operate across multiple different, or even the same network. That is why we decided to hone in on this stream and utilize it within our product.

Raspberry pi is essentially a micro-computer that comes in different models and specifications. For our use, we have the raspberry pi 4 model B, which like a computer has its own ram, processor and storage capability along with some other computer-like features. Just like a computer, we need a display to use in congruent with it in order to use it's I/O capability. The raspberry pi as shown in the diagram will have a LCD touch screen attached to it, whereas the user can also use a regular monitor using a keyboard and mouse instead.

Also there are two other branches coming off of the raspberry pi. The first on the left is a branch that is the desktop application. This application is going to be within the pi itself and will ultimately be an executable file. These types of files essentially cause a computer to perform tasks or encoded instructions instead of having to compile and run a program itself. A typical execution of a program takes high level programming languages and encodes it to machine language. This process is called compiling which is common to computer programmers and easy to do if you know how but for users they just want to run the application. That is why we want to do this executable because it is pre-compiled and runs the program just like a script.

Within the pre-compiled code within the executable, it will execute and be responsible for the software code that interacts and uses the firmware to take care of the tasks responsible for the smart-hub. These tasks consist of things like human detection using machine learning libraries and interacting with the GPIO pins on the raspberry pi to communicate with our other hardware components. These signals will be sent through the GPIO but ultimately will be configured and coded and executed within the executable, which is why we can safely say that the application is responsible for this. Also a user can connect a monitor to display this or can let the system run itself. What that means is that the user can display the application on a monitor or let the smart notifier PCB work itself and alert the user of a threat which can be similar to products in the market like amazon's Alexa where it notifies users with sound alerts and color cues.

From the firmware it is connected directly to the MCU which is responsible for the smart notification system. This will be a DAC that will digitally convert analog to audio for the user to hear the audio that is sent from the raspberry pi. Also the firmware will send signals through the GPIO to the led's to display when there is either a threat detected or when there is no threat and that will idle as a certain color. We used the color red for threats as it is commonly known and used as a standard for danger and the color blue for an idled safe state because that is commonly known as safe and calm within the ADA industry and standards.

3.0 Research Related to Project Definition

The research section of this document will discuss the reasoning behind the design choices we choose for implementing our project. We'll look at other similar products to analyze their pros and cons and compare their features with the features we plan on implementing. With the research attained, we'll be able to determine the necessary functionality required to make our project a more marketable product. An in-depth analysis of different product features lets us gauge what their projects are missing that we can implement to make ours a more competitive product on the market.

We also analyzed the features that have already been adopted so we can use similar concepts but with improved functionality. This section will also include extensive research on the possible hardware components, selection of parts, and relevant technologies that can be used for building our project. The primary goal of the research is to understand the technologies currently being used for building products similar to ours. There will be some components and tools we research that we might never use. By having a better understanding of a variety of possible solutions, we benefited when building our design. It will also prevent us from getting stuck at a certain point of the project's build. For example, if one of our design implementations doesn't work properly or if it doesn't meet our standards, we have backup design plans due to the extensive research we provided in this chapter.

3.1 Existing Similar Products

Security companies have been on the market for many decades, but we believe there are no products similar to ours in terms of simplicity and flexibility. In this section, we will look at multiple products similar to ours to better understand the market. The products in this section all have something in common, they are large companies that have a large financial backing. However, we do not have a financial backing like the companies in this section, so we expect to use many of the same features but it may be hard to meet the level of performance compared to one of these companies. We're also a four person group compared to these companies that are a large team of engineers. Another constraint we have that these companies don't face is we have to complete the project in a matter of six months, compared to these companies that have more time to build their product.

Our research conducted on different products will focus on the product's physical appearance, usability, flexibility, and overall cost. Each security company that is mentioned in this subchapter has similar qualities to what we are offering. These companies have been doing similar things for the past 10-15 years but often lack the flexibility we offer. Throughout this section we will also do a deep analysis on other product's features such as video-streaming quality, wifi connectivity, installation techniques, and other essential features that may cause the user to like or dislike a product. By comparing other product's pros and cons, we'll be able to gauge which features the current products lack and determine which features we can add to our project to make ours more powerful and unique.

3.1.1 Vivint

Figure 3: Vivint Smart Hub

Vivint is a well established security company that provides many useful security features. They have a highly reputable product that has a broad range of services ranging from front door cameras to a car theft prevention product called Car Guard. Their cameras are what we are most interested in. They offer doorbell cameras, indoor cameras, and outdoor cameras. The doorbell camera pro and outdoor camera pro are the products that relate the most to our project since their main focus is surveilling the property of a user's home. Both of these cameras are installed using wires run through the housing complex. These cameras are wired for power but not for connection which makes them an interesting company to talk about. Their cameras are very flexible, they offer cloud recording at an industry leading quality along. They also use WiFi, have night vision recording, motion detector sensors, two-way audio, and they're weatherproof. The camera footage is stored in the cloud that can be accessed anytime from the user's smartphone app or from the user's hub located inside their home. All of their products are highly reviewed. However, to get a straightforward answer on how much the product costs, a sales representative has to come to your home and do an inspection.

An important comparison that we looked at is their smart hub. The smart hub is a central monitoring control that is hung on the wall. This hub is wired for power and has an array of features. This hub that Vivint creates is basically the brains of their system. Roughly the size of an ipad it is a stationary hub that lacks a few things, it is unable to move and unable to be unplugged or the system will set off an alarm. The Hub also allows for video surveillance but only from one location in the home. The hub they offer has been improved constantly every 2 years and is software updated around every 3 months. The hub is very user friendly and allows for virtually anyone to use but requires a passcode to login and view cameras. The thing that elevates this hub is it has two 4G LTE chips in the hub which allows it to maintain a steady connection even when the wifi shuts off. The hub does have a few issues with it, it requires a great internet speed and uses a lot of bandwidth, which causes people with a lower internet to suffer greatly on video surveillance. If the connection on the hub is not strong the video quality often lags a lot and skips frames. They state that they need a minimum 2 mbps upload speed when having seen it first hand it requires a bit more. The bandwidth on this system typically leaves customers with a significantly lower quality than their stated quality of 4K.

Finally the price of the Smart Hub, while Vivint does not offer the purchase of the cameras and the Hub separately from the system there remain a few things to point out. If someone wanted to purchase a Hub the price is $1699 that price includes two sensors and a motion detector. To include cameras the customer will also have to purchase each Outdoor Camera Pro from Vivint at a price of $400 per camera. Without financing that is an extremely hefty price to pay for someone who just wants to view their home. While Vivint is marketed as a smart home they do not allow for much versatility in their product. The only plans their customers can choose are 3.5-5 year contracts at 0-9.99% financing based on their credit. An average system at vivint that includes 3 cameras would be a monthly price of around $95/month for 5 years which prices out to nearly $6,000 with installation. For someone who simply would like to monitor their home without all the bells and whistles that is not an affordable product.

In comparison to Vivint our hub will be a lot simpler and cheaper. Our product will allow for people to connect a variety of different cameras and have a lot more flexibility. Without the pressure to buy a full $6000 security system our product will offer a great alternative to the traditional full blown security system.

3.1.2 Ring

Ring is a company that was built by Jamie Siminoff in 2012. He decided to come up with the product to create a safer environment for people while they're in their homes. The company's mission statement is “Help make neighborhoods safer”, which is very similar to ours. Ring offers two products similar to our project's design, their Stick Up Cam and their Video Doorbell Pro. Both of the products are simple and give people the confidence they're looking for within their homes.

The Stick Up Cam is a wireless camera with human detection capabilities that can be installed anywhere outside the premise of one's house. The Stick Up Cam records live, 1080p HD, and has colored night vision. It uses rechargeable batteries that get inserted into the bottom of the camera. Stick Up Cam starts at $99.99 per camera, and the user easily installs it themselves. The Video Doorbell Pro is a 1080p HD, night vision camera that's connected using a wire. It acts as a doorbell, so it must be placed at the door of the user's home. The Video Doorbell Pro has two-way talk allowing the user to communicate through a speaker to the person outside of the door. It offers motion detection sensors to detect when a person is at the door. Video Doorbell Pro starts at $159.99 per camera, and the user easily installs it themselves. Both of Ring’s products are connected to the user’s network via wifi.

Neither type of camera comes with the technology for human detection or the ability to save recorded videos to the cloud. To get these features, there is a subscription that costs three dollars per month. These features are accessed and controlled through their mobile app, where the user can view alerts and live footage of the camera's recordings. The camera’s recordings are accessible through the cloud for sixty days. More expensive subscriptions allow the user to store footage for longer than sixty days in the cloud. The more expensive subscription costs twenty dollars per month, and it also offers emergency assistance if somebody breaks into the home and the police need to be called to the house.

Figure 4: Ring Doorbell Camera

While Ring offers a great alternative to the traditional security system the system also lacks in a few areas. Ring does not have a central monitoring hub that allows the user to look at. The only way to view the cameras is on the phone with a significant delay. Ring does not have a great alert system, every camera has issues that relate to sensitivity and notifications. Many reviews on the product state that the cameras are too sensitive and often receive tons of texts that someone is outside their home when it was simply the wind blowing too hard.

One of the features this product has that intrigues our group is the ability to communicate back and forth with the person at the door. Due to the time constraint we are facing for our project’s design, we will not be able to implement the same feature into our system. However, suppose we can get the Ring camera to configure with our system. In that case, users will still benefit from this feature, being able to communicate with the person at the door without physically opening the door for communication.

Our goal is to not replace the ring but simply be compatible with their system. With the Surveillance Smart hub the goal is to be able to purchase a ring and be able to view the Ring at the convenience of the hub. If the user is not staring at their phone or is occupied, when someone approaches, our Hub will change colors and simply alert the owner that someone is approaching. Ring will be a great product to work with and offer the user a central hub to view cameras on as well as an alert system. This is also beneficial to users compared to using other major vendors because the subscription fee to use Ring starts at two dollars per month compared to companies like ADT or Vivint that cost much more.

3.1.3 ADT

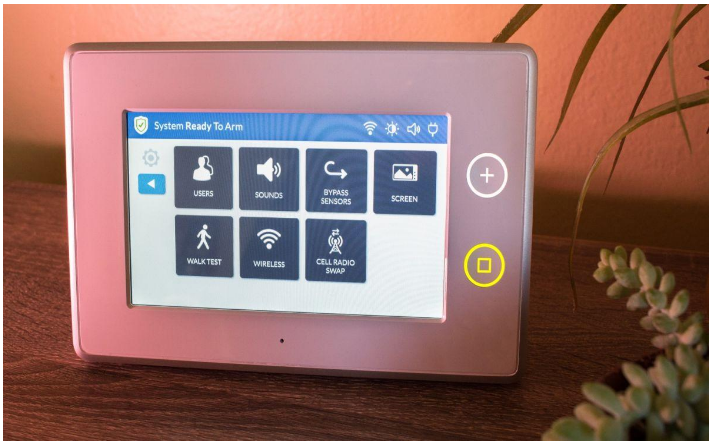


Figure 5: ADT’s Security Management Hub

ADT offers a wide variety of security products ranging from home security systems, home security cameras, life safety alarms, and smart home automation. We are interested in their home security cameras and their central monitoring hub because their features are related to our project. ADT offers a Google Nest Doorbell that allows users to view their front door from any location using their smartphone. It uses human detection to alert the user when someone is at their front door. It has the capability of detecting faces that it's previously encountered but will still let the user know they’re at their home. Their other product that interests us is their outdoor security camera. Their outdoor cameras provide the same features as the Google Nest Doorbell, but the locations of the cameras have different pros and cons. The benefit of the outdoor camera is it has a broader range of video since it has an open area to record compared to the front door that is generally narrow in design. The benefit of the Google Nest Doorbell is it's easier to communicate with the user since they are waiting at the door for a response. Both cameras offer two-way communication, a feature controlled using the smartphone app allowing the user to communicate with the person at the door even if the user is not home. Both cameras also record the footage using 1080p HD, with night vision capabilities, allowing users to view live footage from their smartphone. ADT won't provide a price estimate until you consult with one of their sales representatives, who must first come to inspect your home.

In regards to ADT's hub, they use a similar hub to Vivint. The ADT hub lacks the ability to view and monitor the cameras. Their hub simply acts as a way to arm and disarm their system. While ADTs prices are slightly less expensive than Vivint they still do not offer the ability to monitor cameras from the hub. A basic ADT system starts at around $65 and if third party cameras are added the system can range up to $110 a month with a 3 year contractual obligation.

3.1.4 Existing Similar Products Summary

As mentioned in the previous sections of this subchapter there are plenty of security companies that provide the services necessary for securing one's home. There is a variety of full fledged security systems that fully secure the premise of a home. There also exist less complex systems such as doorbell cameras that give the user a simple way to interact with the person at the front door without opening the door. Although there are lots of methods for securing ones home, we believe no system exists in which a user can easily configure their own IP cameras to interact with a home security application. There are certain features that exist in each of the products mentioned in this subchapter that we find useful. However, we don't feel it's necessary to implement all of the features as some of them may never be used by some of their customers.

As mentioned in the introduction to this subchapter, the majority of the most commonly used security companies require the user to engage in a subscription contract that can be very expensive. Also the major security companies often offer more features than the user may want or need. By extracting some of the useful techniques and features that well known security companies implement in their design and implementing them in our application's design, we feel we can provide a product that is just as useful at a fraction of the cost. Our product will also keep customers from being locked into a particular camera vendor since we allow any type of IP camera opposed to popular security companies that require you to use the cameras that they provide.

3.2 Existing Similar Projects

In this section, we will analyze other projects that engineering students at other major universities designed. Since IoT has become much more popular within the past decade, projects similar to ours have become increasingly more relevant. It is common for computer and electrical engineering students who are getting ready to graduate to use single-board computers and microcontrollers to build a project that showcases their skills while completing their engineering degrees. It is also common for engineering students, especially computer engineers, to be interested in security since computer engineers constantly face threats that must be addressed to make systems safe. Although our project is not directly related to cyber security, we believe it has the same fundamental concept, secure something from intruders with bad intentions.

When analyzing the different projects, we considered the various hardware components and software stacks used for their implementations. We also conducted research on the fundamental behavior that these applications produce. After analyzing each project, we summarized what features and components we could mimic and how we implemented a design that enriches these features.

3.2.1 The Home Alarm System

The Home Alarm System is a project that is similar to our project idea. It was conducted as a senior design project at Mississippi State University in 2016. The key idea of the project is to keep homes safe from trespassers and intruders. It provides the ability for the user to observe the status of their home through a user interface on their cell phone. The application provides the ability for the user to determine when someone is at their home and to determine whether that person is a threat or not. If the system detects a threat, the user can easily notify the police by simply clicking a button on the application's interface.

Their project was implemented by integrating software and hardware onto a single-board computer. The single-board computer they use is a Raspberry Pi B+ that contains 512 MB of memory. Since the project was implemented in 2016, the Raspberry Pi they used did not contain as much memory as the ones commonly used today in 2022. Although their project works fine with 512 MB of memory, we feel our project will have a smoother workflow since we have access to more powerful single-board computers with memory sizes in the GigaBytes range.

Their project has the same fundamental purpose as ours, but our project has different ways of addressing the problems that potential users may face. For example, their application is controlled by using a mobile phone application, while ours will be controlled through a main hub located inside the user's home. We plan to build a mobile phone app as one of our stretch goals because we believe it's powerful for the user to be able to view their home's status from a remote location when they are not currently home. We believe that using a hub located inside the home is the best method for alerting the user when someone has entered their property because it cannot be mistaken for a phone call or text message. If the alert is outputted through someone's phone, we believe it could be accidentally ignored. But if there is a hub inside their home, the hub will have one purpose, to alert the user that someone is on their property. Therefore, since the hub only has one purpose, the alerts cannot be mistaken for another type of notification.

3.2.2 IoT Based Security System

The IoT Based Secured System is a research project conducted by the College of Science and Engineering at Central Michigan University. The purpose of their design is to provide banks with an inexpensive security product that will help secure the premise of their building. Since banks need to be an extremely secure piece of property, it is important for the product to work effectively providing the bank owner and police with alerts as quickly as possible whenever the system detects activity.

By placing the product in areas of the bank that people are not authorized to be in, the product acts as a silent alarm. PIR motion sensors are used for detecting any human presence. Once a human is detected, the security system begins recording and taking photos of that person. Once the footage of the intruder is captured, the footage will be sent to the cloud, which will then be sent to the bank owner and to the police. To alert the police and the bank owner in a timely manner, there is a process for alerting them. First, a text message is sent to the bank owner and the police to alert them somebody has been detected in an unauthorized location. Then the footage is sent to both parties via email, and since they have already received a text alerting them about the situation, they will know to check their emails to get access to the footage.

To make the product affordable while maintaining quality, it uses inexpensive IoT products. It uses a Raspberry Pi Model B as the fundamental brains of the product. The Raspberry Pi is connected to additional hardware that helps make the detections of humans possible. Adafruit PIR motion sensors are used for capturing motion whenever someone enters an unauthorized room located inside the bank. Once the PIR motion sensors detect motion, the Raspberry Pi begins recording and taking photos of that person’s actions using a Raspberry Pi Model B USB Camera.

Since an alert is sent to the police and the bank owner, the system must use a certain networking protocol to transmit the data. To send data via email and text using a Raspberry Pi Model B, a protocol called SMTP, for simple mail transfer protocol is used. Before the text messages can get sent to the bank owner and the police, a Twilio account must be setup. Twilio is an online messaging service that is simple and quick to setup an account. To set up and account there are a few commands that need to be run on the Raspberry Pi. Then the phone numbers that will be receiving the text messages get configured by hardcoding them into a Python application located on the Raspberry Pi.

3.2.3 Home Security System Using Raspberry Pi

The Home Security System Using Raspberry Pi was a design project conducted at Cornell University presented by the school of Electrical and Computer Engineering. The system's primary purpose is to provide the same functionality as an existing product called DLINK security system, but without requiring access to the cloud to use the product. The reason for building a product that doesn't use the cloud for storing the user's data is due to security concerns. By excluding the cloud for storing the user's data, it prevents people from gaining access to sensitive data that could maliciously make decisions with bad intentions.

The system works by constantly polling the home's conditions by using various sensors. It uses a Raspberry Pi 3 that takes the data polled from the sensors and compares it to a threshold limit set by the users. If the data ever reaches the threshold set by the user, the user will be sent an email providing them will all of the information regarding the conditions of their home. Two sensors are used for polling the data, a PIR motion sensor for detecting motion and a temperature and humidity sensor for detecting the temperature and humidity of the user's home.

The PIR motion sensor is used for detecting any motion going on inside the user's home. PIR stands for passive infrared, which detects a slight amount of infrared radiation to be triggered. The product's design was done so that the parts used could be purchased at a low price. The PIR motion sensor used for this product is an HC-SR501 PIR motion sensor. The sensor is relatively inexpensive and integrates easily with the Raspberry Pi.

The temperature and humidity sensor used in this product is used to detect the humidity and temperature within the user's home. Relative to the PIR motion sensors, the temperature and humidity sensor also needs to be affordable, so they use ones that can be purchased at a low cost. The temperature and humidity sensor they used is a DHT-11 Temperature and Humidity sensor. This type of sensor has a humidity sensor that detects the humidity, and it uses a thermistor to sense the temperature. One of the main reasons they chose this sensor is because it only requires one wire for transmitting its data to and from the Raspberry Pi.

Once the system detects data that reaches the threshold that the user set, the system sends an email to the user alerting them of their home conditions. The emailing service is implemented using Python's standard library, and the emailing is implemented in Python using an object-oriented approach. Once an email is sent, a boolean value is returned to the program to ensure the email has been property sent to the user.

3.2.4 Existing Similar Projects Summary

Out of the three projects discussed in this subchapter, there are plenty of helpful design choices we could use and tailor to make our product more reliable. Just like our project, the projects in this subchapter were designed by computer and electrical students as a milestone project to showcase the skills they have acquired as engineering students to obtain their college degrees. The projects all used a single-board computer, various sensors for collecting data, and a significant PCB design.

Not only did all of the projects use a single-board computer to act as the central logical unit, but they all also used Raspberry Pi models for this component of the system. We believe that using Raspberry Pis to build systems such as ours is so popular because they are relatively inexpensive and provide lots of computational power. They also have the potential to connect various types of sensors in order to collect the data necessary to make powerful applications. We also believe these groups may have chosen the Raspberry Pi because it offers lots of detailed documentation and online resources.

All of the projects provide the user with a form of security that will help the user feel more comfortable about the well-being of their homes. Although all of the projects offer protection in different ways, we still haven't found a project like ours that allows users to integrate their own IP cameras to make a customizable security application. For these projects, they use cameras that Raspberry Pi provides. The Raspberry Pi cameras may not be what the user wants to use for their implementation due to their resolution and physical size or appearance. The Raspberry Pi camera must be attached to the Raspberry Pi itself.

In contrast, our application's cameras are not connected to any external hardware giving the user more freedom to where they would like to install their cameras. Allowing the user the ability to place the camera where ever they'd want gives them the ability to hide the camera more efficiently. If the user tries hiding one of the cameras provided by the projects in this subchapter, they will most likely struggle since the camera is attached to the Raspberry Pi's chip.

3.3 Relevant Technologies

Within this section we will discuss things that are indirect technologies that will be used that are secondary or tertiary technologies that are important to the project but are not dependent on the project. Things such as a way to display to an LCD screen or the capability of how to power the device. These features are pre-built within the device that we did not invent, choose or create but have relevant and major impact on how we use and operate the device.

3.3.1 Display Serial Interface

Figure 6: Raspberry Pi Display Serial Interface

Since the Raspberry pi 4 module has the ability to accept a Display serial interface we considered this a display option for lowering the cost of our overall build. The DSI model was a specification introduced by the Mobile Industry Processor Interface (MIPI) Alliance,This is a global tech alliance that specializes in developing specifications for the mobile ecosystem. Using DSI is our best option for low power consumption, since the MIPI interface uses low voltage signaling which has the benefit of low power operation. But even at such a low power it can transmit at a wide range data transmission allowing for less interference of other peripheral devices while allowing for less pins and offering better performance than other interfaces.

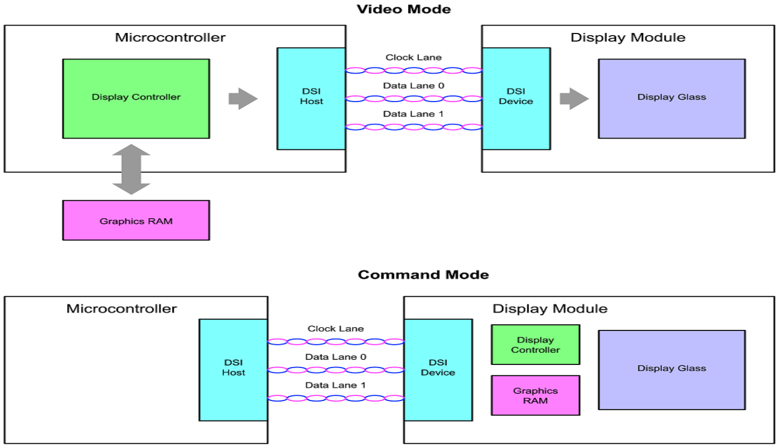


Figure 7: Hardware Communication Diagram

The interface operates with a differential signaling point-to-point serial bus on the physical layer with the latest with the latest lanes for D-PHY operating at speeds of 4.5 Gbps/lane. The Bus consists of one high speed clock lane and one or more data lanes where each lane is carried on a two wire configuration. All lanes will bridge between the DSI host and device where the lanes will transmit in parallel, meaning if four lanes are being used then all 4 bits will be transmitted simultaneously. There are two modes that can be used: the video mode and the command mode.

The video mode will be the standard operation for most systems. In Video mode RGB pixel data and horizontal and vertical sync signals provided by the display controller are encoded into the serial stream by the DSI Host and decoded by the Device. Alternatively, the display controller and graphics RAM can be integrated into the display, this can take the load off the DSI host as its only responsibility would be to transmit the pixel data. These functions make the MIPI DSI interface a lot more complex than the classic parallel RGB plus clock and sync signals, but offers High performance, Low power, and Low EMI.

3.3.2 High-Definition Multimedia Interface

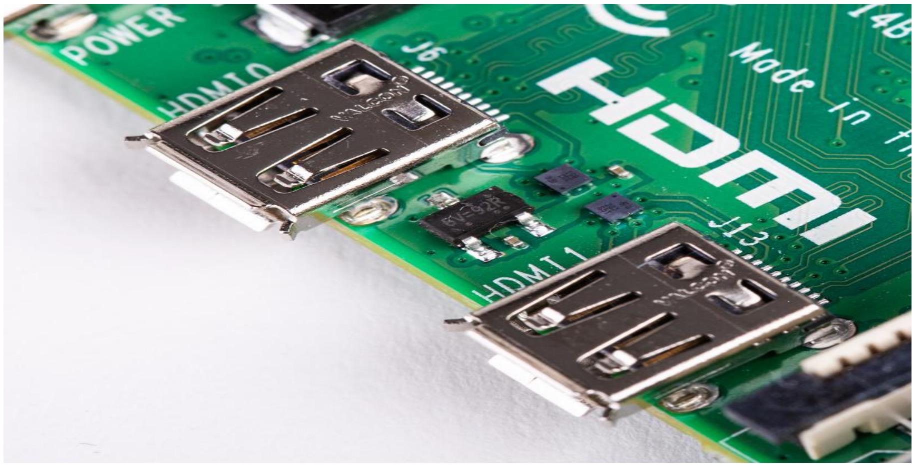


Figure 8 : HDMI interface on a Raspberry Pi

The high-definition multimedia interface is currently an industry standard for video and audio transfer on LCD displays, and our microcontroller comes equipped with micro hdmi ports capable of transmitting a 4k image. The HDMI interface has replaced the old standard of analog Video standards for uncompressed/compressed audio and video transmission.The HDMI standard implements EIA/CEA-861, which defines video formats and waveforms, transport of LPCM audio both compressed and uncompressed, auxiliary data, and implementations of the Video Electronics Standards Association Extended Display Identification Data. although there are many different version of HDMI that basic output is designed with 19 pins that are sectioned off for Transition-minimized differential signaling (TDMS) data, Consumer Electronics Control (CEC), HDMI Ethernet Channel (HEC), and Audio Return Channel (ARC) , as well as power pins and I2C data lines. To be more specific, the HDMI has three physically separate communication channels, which are the DDC, TMDS and the optional CEC with HDMI 1.4 adding ARC and HEC.

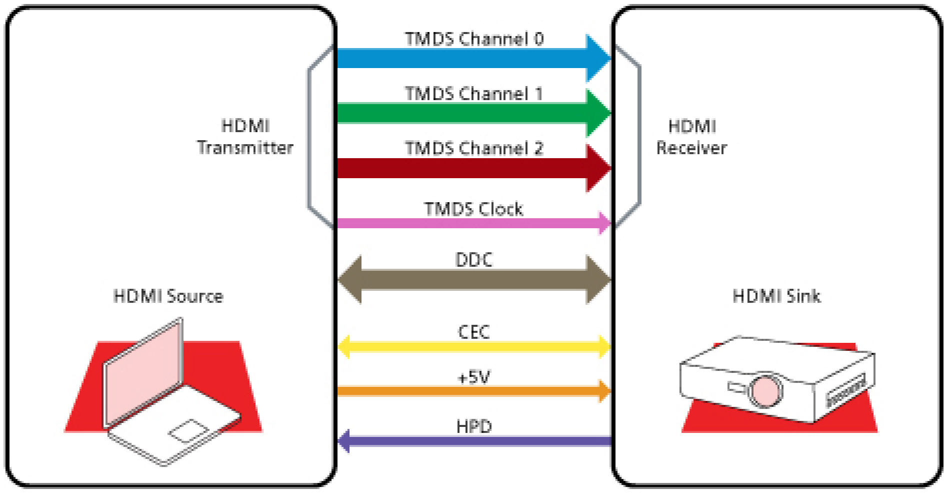


Figure 9: HDMI pin communication diagram

TMDS on HDMI combines video, audio and auxiliary data using three different packet types or channels referred to as the video data period, the data island period, and the control period. When in the video period active pixels are being transmitted, the channel is sending 10-bit characters that are encoded using 8b/10b encoding during the video period. This differs from the island period where audio and auxiliary data are transmitted into packets with each packet containing four subpackets, and each subpacket is 64 bits in size, including 8 bits of parity data used for error correction and describing the contents of the packet. This allows for each packet to carry up to 224 bits of audio data.

DDC is a communication channel based on the I2C bus specification. Specifically the Enhanced Display Data Channel would be implemented to read the E-EDID to discern between what audio/video formats the HDMI source device can take. This function is also necessary for being used for High-bandwidth Digital Content Protection (HDCP).CEC is an HDMI feature designed that allows the user to command and control enabled devices. Based on the CENELEC standard protocol, it is a one-wire bidirectional serial bus that conducts remote control activities. Additionally the use of HDMI's HEC and ARC channels are two extra features. HDMI 1.4 added the HDMI Ethernet and Audio Return Channel (HEAC), which offers a high-speed bidirectional data communication link (HEC) as well as the ability to send audio data upstream to the source device (ARC).

ARC is an audio link that could be used to replace other cables between the TV and the A/V receiver or speaker system, with the TV generating or receiving the visual stream rather than the other equipment. Without ARC, the TV's audio output must be routed through a separate wire to the speaker system.HEC combines video, audio, and data streams into a single HDMI connection, as well as allowing IP-based applications and bidirectional Ethernet connectivity.

3.4 Strategic Components and Part Selections

This section is research on some of the requirements to meet the specification requirements, some different parts that can be used to achieve the requirements and go into a bit more detail on what the options are. There are various hardware parts that were required in order to achieve the goals and efficiency of this project. The main components involve the LCD display, the DSI interface, LEDs, DAC, and a Power supply. Each subsection will go into detail regarding what they are and they are needed for this project.

Each component needs to be carefully selected in order to work well with the software we selected. A major concern involving the hardware is going to relate to the providing power to each part of the Hub from the computer all the way to the speakers. The basic design will involve a LCD display that is able to display and control the camera, if a person is detected using the AI face recognition software, a signal will be sent from the raspberry pi to the DAC. The DAC will then play an audio through the speakers and trigger the LEDs. Overall the hardware design seems pretty simple from an overview but there are a lot of parts that are required in order for the system to work smoothly and coherently throughout.

Regarding the hardware design there are a few options besides the ones mentioned that could have been added in order to make the project easier to complete. The more complexity added to the design of the hardware adds cost to the project. When the goal of the project is to keep the cost as low as possible.

3.4.1 LCD Display

The display interface is one of the most fundamental pieces to this project, the interface between the user and the cameras is done through the display. There are multiple options when it comes to display. These kinds of options include just a regular screen with no I/O or touch screen interaction or even a screen that has touch screen implementation that can take a user's input instead of using external I/O devices, like a “budget” smart tablet. Since our goal is creating an all in one smart-hub then this is a potential option that will most likely be explored for the purposes of our project.

Figure 10: Capacitive Touch Raspberry Pi LCD

In the Smart Surveillance Hub there will need to be a large liquid-crystal display (LCD) that is able to be easily controlled by the user. A LCD consists of 6 main components

1. A polarizing filter that polarizes light as it enters.
2. Glass substrate that consists of ITO electrodes.
3. Twisted nematic liquid crystal.
4. A glass substrate with ITO electrode film.
5. Polarizing filter film.
6. A reflective surface

We have the option to choose between two forms of LCDs: Display Serial Interface and High-Definition Multimedia Interface. No matter which interface we choose between or display it will need to have capacitive touch capabilities in order to meet our design criteria for ease of use.

For the selection of a LCD there are many options available but the goal is to find the LCD display that offers a large surface area for the user to see from a good distance away as well as being cost efficient. LCD can get expensive especially when trying to find one in order to the criteria of this project. The display has to meet the minimum requirements of the cameras we are testing in the project, the display needs to output a minimum resolution of 1080p. In order to take an invention like this to market, a LCD would probably need a slightly better resolution, but for the scope of this project and the budget limitations we went with the minimum of our standard requirements.

3.4.1.1 Thin-Film-Transistor Liquid-Crystal Display

In 1957, Thin film MOSFETs were patented, over the next 20 years the technology progressively grew as transistors got smaller and smaller. In 1974 the first flat active matrix display was developed by T. Peter Brody, which allowed for better resolution and higher-quality visual displays.

This technology is a variant of the LCD display that uses TFT technology that allows for better image quality and contrast. Similar to the LCD the TFT LCD uses an matrix LCD but the main difference is in the TFT the matrix is an active matrix as opposed to a passive one. A passive matrix LCD uses a simple grid to control pixels, has a slow response time, imprecise voltage control, and is used in low-resolution displays. An active matrix will have larger range and more vibrant colors, it is a faster response as opposed to the matrix.

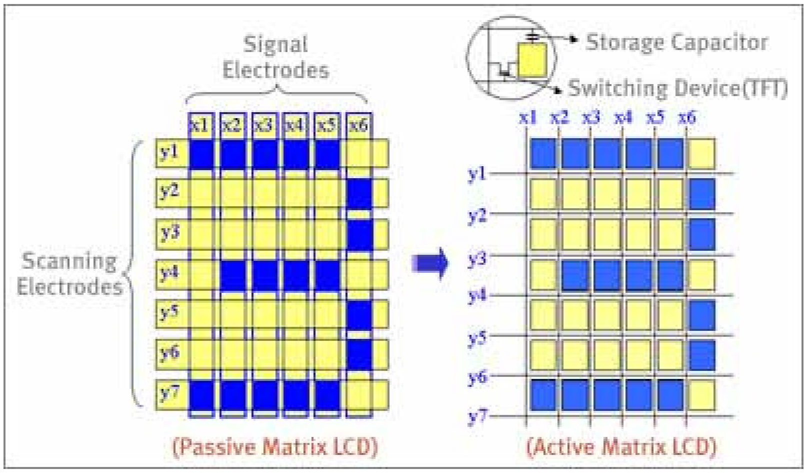


Figure 11: Passive Matrix vs Active Matrix

The development of the active matrix is what progressed the technology and allowed for LCD screens to be more adaptive and perform at a higher efficiency.

3.4.1.2 Capacitive Touch Screen

A separate part from the LCD screen that needed to be considered is making the display touch screen. The capacitive touch screen is a transparent piece of tempered glass. Capacitive touch screens rely on finger pressure to create an electrical inductor. This inductor affects the electrostatic field of the touch screen. There are 4 main types of capacitive touch screens: Surface Capacitance, Projected Capacitive Touch (PCT), PCT Mutual Capacitance (PCT), PCT Self Capacitance. Surface Capacitance is coated on one side and consists of small voltage conductive layers with limited resolution. Projected Capacitive Touch (PCT) consists of conductive layers with electrode grid patterns and includes a very robust architecture. PCT Mutual Capacitance mainly used for multi touch consists of capacitors at the intersection of each grid. Finally PCT Self Capacitance similar to Mutual Capacitance but supports a stronger signal and is not optimal with more than one finger.

Type	Optical	Sensitivity	Thickness	Cost
	Moderate	Low	High	High
Projected Capacitive Touch	Moderate	Moderate	High	Moderate
PCT Self Inductance	High	High	Moderate	Low
PCT Mutual Inductance	High	High	Moderate	Low

Table 1: Touchscreen Technology Comparison

3.4.2 Audio Alarm Components

A vital part to this project is the alarm system. There are a few ways of going about creating the alarm system. The first part needed is an audio subsystem. This subsystem there are a few different hardware designs that can produce a sound needed to be used as an alarm. The first design receives a signal from the microcontroller, travels through a digital to audio converter, the signal is then amplified in an audio amplifier, then finally reaches the speaker system. The second design option to produce an alarm sound is a tone generator, this simply works by receiving a voltage and depending on the voltage a specific frequency is generated and sent to the speaker system.

This design still requires an audio amplifier. The last option to generate a sound is a piezoelectric sound buzzer. This is the simplest of designs and works by receiving a voltage and once the voltage is applied to the buzzer a buzzing sound is produced through the speaker. Each of these options will be explored and their respective components will be discussed as well. The goal of this section is to analyze each component and compare each system to see which fits the best for the scope of this project.

3.4.2.1 Digital to Analog Converter

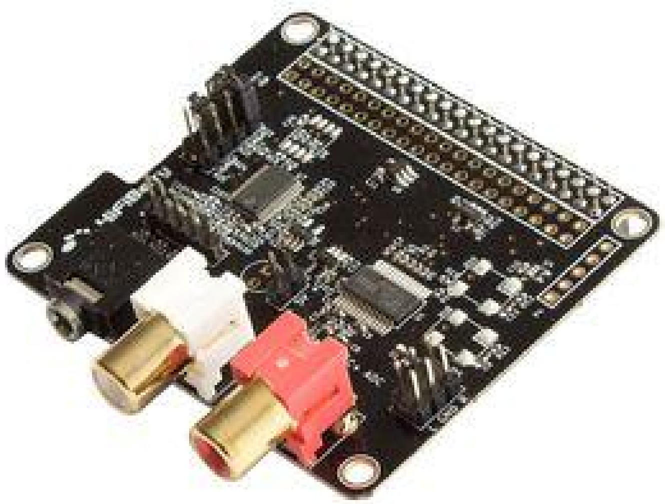


FIgure 12: DAC+ ADC Digital to Analogue Converter for Raspberry Pi

A digital to analog converter takes a digital signal and converts it into an analog audio signal. Any system that involves the playing of an audible sound sent from a computer requires a DAC. Without the use of a DAC the sound is merely just a collection of bits, a collection of 1’s and 0’s. The DAC is the middle man in regards to translating that code and turning it into a sound that is able to be heard. In regards to Smart Surveillance Hub the DAC is going to be the piece that receives a file from the Raspberry Pi and turns the signal into an actual alarm sound. The alarm sound will then be sent to the speakers from the DAC and the alarm will play. The DAC will also control the LED lights as well. The LEDs will always remain on and once the audio file is received from the Raspberry Pi it will trigger the LEDs causing them to change to a different color.

The specific needed DAC we need has to be able to be controlled by the Raspberry Pi using NodeJs which is the software selected for the project. NodeJs will communicate with the GPIO pins on the DAC and that is how the audio file will be sent over and able to be displayed on the speakers. The majority of current audio transmissions are recorded in digital form, such as MP3s and CDs, and must be transformed into an analog signal in order to be heard. Through the usage of Random access memory DACs, DACs are also employed in video sampling (RAMDACs). Due to the nonlinear response of cathode rays and the way the naked eye operates, video sampling had a distinct scale of work before the standard of LED TVs. The usage of RAMDACs in the design of a "gamma curve," which was supposed to produce the impression of equally distributed brightness increments over the display's full dynamic range, rendering hardcoding for each channel in a DAC obsolete.

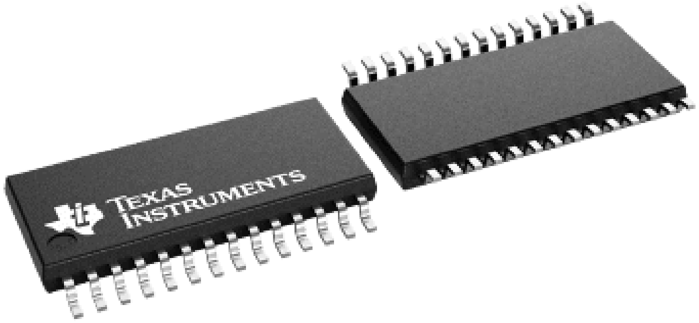


Figure 14: Texas Instruments PCM5122

Currently our design for our DAC includes the DAC microprocessor PCM5122. It provides for CM Interface and Fixed Audio Processing With a two channel configuration,and a 32 bit resolution bus. This will meet our criteria for expanding our audio capabilities from our microcontroller. Some drivers need to be installed onto the Raspberry Pi to get the DAC working correctly with the Raspberry Pi. It is possible to use another type of DAC for our project, but we would have to build the necessary drivers from scratch for it to work correctly. Since we are limited on time, creating our own drivers would not be an intelligent design choice. Hence, the DAC we use should have drivers compatible with the Raspberry Pi for simple implementation.

3.4.2.1.1 MP3 Audio Files

If we decide to use a DAC for producing the necessary sound required for pushing an alert through the Security Smart Hub, we need to use a specific type of audio file. MP3 audio files are one of the types being considered for this project. The MP3 audio file format is derived from MPEG-1 and MPEG-2 video standards. The benefit of using MP3 type audio files is that they are compressed to a reasonably small size. Using an audio file small in size is an advantage because the audio file will be stored on the microcontroller's RAM. Microcontrollers have limited capacity, so when designing for audio, the size of the file plays a significant role.

The next critical factor that goes into choosing the correct type of audio file is quality. MP3 has been around since the 1990s and is still relevant today. The reason MP3 files have maintained credibility for nearly thirty years has a lot to do with their high quality in sound.

3.4.2.1.2 WAV Audio Files

The next type of audio file we explored for producing sound to the hub is a WAV audio file. This type of file has also been around since the 1990s, proving it to be high in quality for various reasons. Compared to MP3 audio files, WAV audio files are much larger, sometimes ten to eleven times larger in size. WAV files are so much larger because MP3 files are compressed, and WAV files are not compressed. It is possible for WAV files to be compressed, but it is not common practice.

The tradeoff for larger audio files is higher sound quality. WAV audio formatting is used for the audio of CDs. The format for WAV files is linear pulse code modulation allowing the bits to be sampled at a rate of sixteen bits per sample. Professionals who work in the audio industry tend to use WAV files when they are looking to produce the highest quality sound.

3.4.2.1.3 DAC Audio File Type Selection

As mentioned in the two subsections of the last subchapter, the audio file type will be necessary to produce a high-quality alert. The microcontroller is limited in space, so choosing a small audio file would benefit the system. Between the two audio files researched, we need to do some testing before deciding.

As of now, we believe the best approach for choosing the correct type of audio file would be to use an MP3 file initially. The first step is to use an MP3 file and test the quality of the audio sample. If the audio sounds professional, it would be best to keep this as our format due to its size. As long as the sound quality is good, MP3 would be the file type that produces the smallest overhead for the microcontroller.

If we tested the MP3 file's quality and it didn't not meet our standards, we explored using a WAV file. Since the WAV file size is larger, we don't worry about the quality of the sound. We could run into the issue when using a WAV file because it could potentially slow down our microcontroller if it is too large. Since a microcontroller has a limited amount of memory, we needed to test this to make sure it didn't slow down our application. We believe this shouldn't be an issue since our audio file will have a short duration of time.

3.4.2.2 Audio amplifier

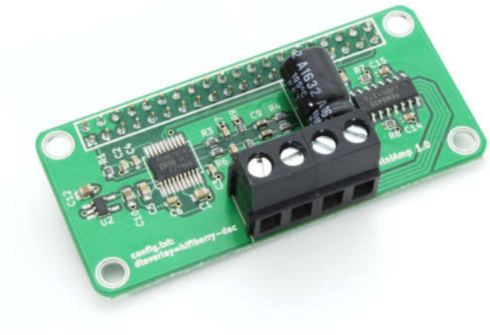


Figure 15: Audio Amplifier PCB example

An audio amplifier is an electronic amplifier that amplifies low-power electronic audio signals to a level that is high enough for driving loud speakers or headphones. Audio power amplifiers are found in all manners of sound systems and are typically the final stage of the audio playback chain before the signal is sent to the loudspeakers. We need an audio amplifier since we used a DAC for emitting a sound notification. The audio amplifier must produce a sound loud enough that the user of our system can hear it from a distance or if any background noise exists.

The first audio amplifier was invented in 1906 by Lee de Forest,the triode vacuum amplifier made the function of amplifying electrical components possible. The triode is a three terminal device with a control grid that can modulate the flow of electrons from the filament to the plate, and was also used to create the first AM radio. Ever since its invention, the technology has become more sophisticated providing users with higher sound quality and louder amplifications.

With the invention of transistors and their inexpensive availability, most modern day audio amplifiers have implemented solid-state transistors, such as the bipolar junction transistor (BJT) and the metal–oxide–semiconductor field-effect transistor (MOSFET). This made transistor based amplifiers lighter in weight, more reliable and require less maintenance than tube amplifiers. If we would have decided to use a tone generator we would most likely not need to use an audio amplifier. But most likely we would use a DAC and need to provide some amplification for the clearest and loudest possible sound.

3.4.2.3 MOSFET voltage regulator

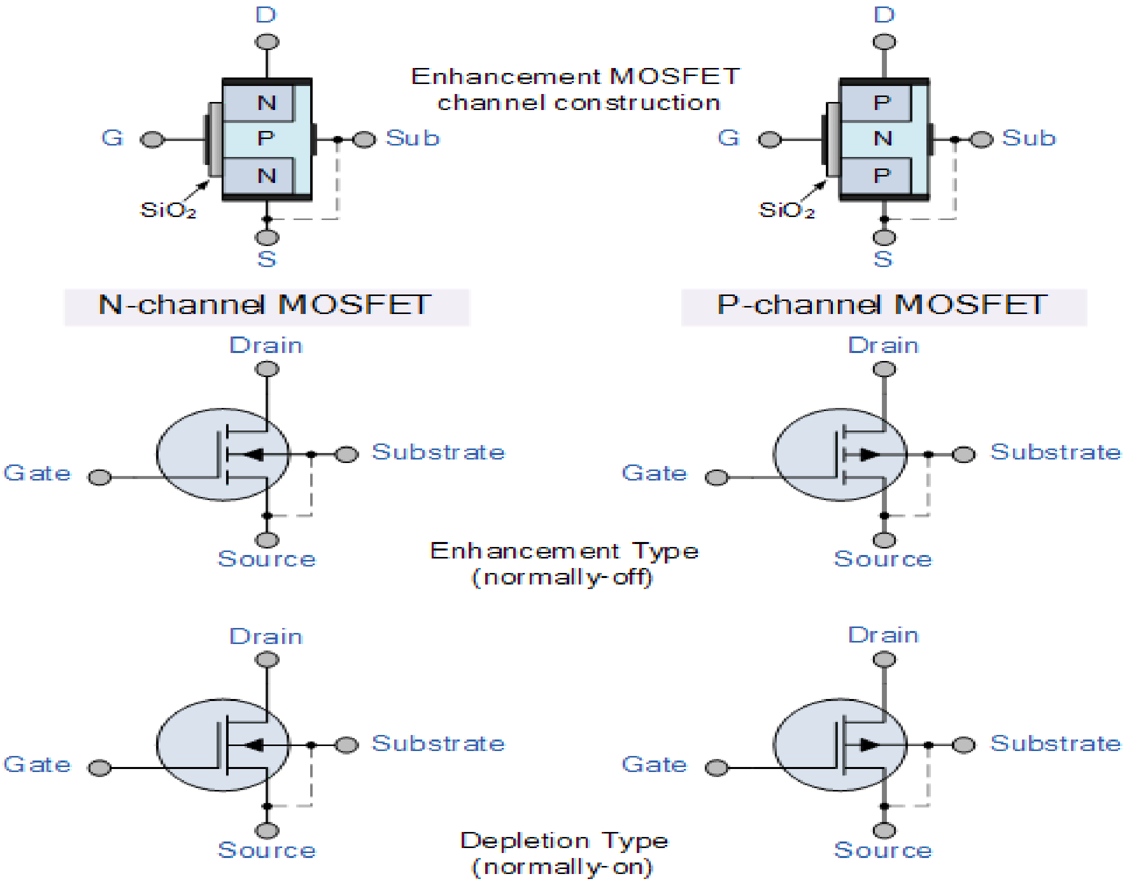


Figure 15: MOSFET Regulators

In our design, the use of a voltage regulator will be implemented in our Digital-to-Analog Converter (DAC). We are stepping down from 5 to 3.3 Volts using an ADP150 component. Inorder to lower the voltage noise in our system and provide each component with there specification needs a voltage regulator is necessary. Since our main goal is to retrieve a clear audio signal from our raspberry pi we want our voltage specification to stay within their means as much as possible without too much oscillation. This means that we would need to use metal–oxide–semiconductor field-effect transistor (MOSFET). A MOSFET device has four terminals which are source, gate, drain and body with the body being tied to the gate. The flow of charge carriers enters through the source terminal and exits through the drain terminal, the width of the channel depends on the voltage applied at the gate terminal.

There are usually four types of MOSFETs that are used: The Depletion-type N-channel, Depletion-type P-channel, Enhancement-type N-channel, and the Enhancement-type P-channel. The Depletion-type MOSFETs are at maximum efficiency when the voltage on the gate is zero. When negative voltage is on the gate the conductivity will decrease and the voltage will increase, meaning the channel is always open. Inversely, The enhancement-type MOSFETs wont operate at maximum efficiency until voltage is applied. Meaning that the voltage gate conductance will not increase until a voltage is applied, so the normal state of the channel is always closed.

MOSFETs are also classified by P-channel and N-channel based on what the substrate is made up of. P-channel MOSFETs are usually heavily doped in the P region for source and drain terminals. When a negative voltage is applied to the gate the electrons from the metal-oxide layer are leaked into the substrate attracting electron-holes to the channel widening the gap. N-channel MOSFETs have heavily doped N regions at the source and drain, where the body is made up of p type substrate. When a positive voltage is brought to the gate electrons from the substrate become attracted to the metal-oxide layer and repel holes widening the channel gap.

3.4.2.4 Passive/Active Speaker System

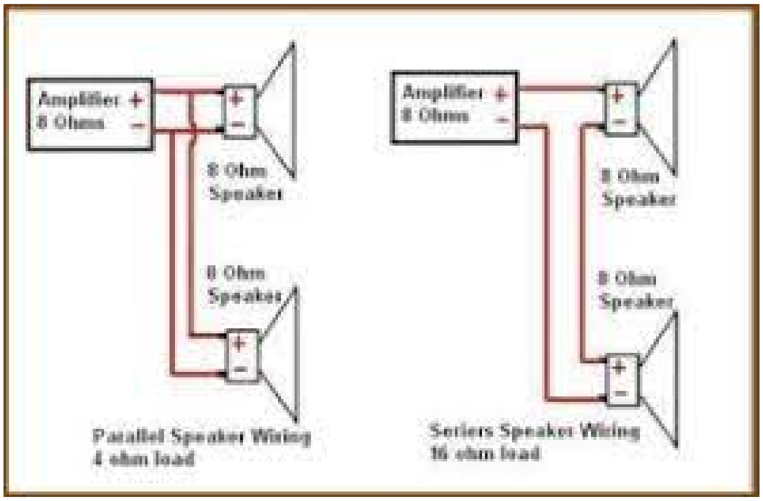


Figure 16: Passive vs Active Speakers

In our system we use smaller mobile passive speakers to drive our audio and system alarm. In the typical speaker configuration there are active and passive speakers. Passive speakers have no internal amplifier which leaves them to be more mobile and modular and less expensive. The tradeoff with passive speakers is they are capable of producing as loud of an alert opposed to active speakers.

Comparatively active speakers have built in amplifier systems, because of this they can carry a higher voltage, thus giving a higher power output for a greater sound. In a passive configuration we can design for a parallel or series connection. Within a parallel wiring system we can cut the ohm rating by half allowing for an easier connection and a better overall quality. But series connections increase the ohmic rating while giving off a bad or odd sound when playing at a certain level. Although using active speakers may provide the system with higher quality of sound, it will also be more intensive in terms of power usage.

	Brand	Ohmic rating	Power rating	Sound sensitivity
	Adafruit -Stereo Enclosed Speaker Set - 3W 4 Ohm	4 Ohms	3 watts	85dB
	Adafruit-Speaker - 3" Diameter - 4 Ohm 3 Watt	4 Ohms	3 watts	80dB
	CQRobot Speaker 3 Watt 4 Ohm Compatible	4 ohms	3 watts	85dB
	MakerHawk 2PCS 4 Ohm 3 Watt Speaker	4 ohms	3 watts	86dB
	Adafruit Speaker - 3" Diameter - 8 Ohm 1 Watt [ADA1313]	8 ohms	1 watt	80dB

Table 2: Speaker Comparison

3.4.2.5 MOSFET driver

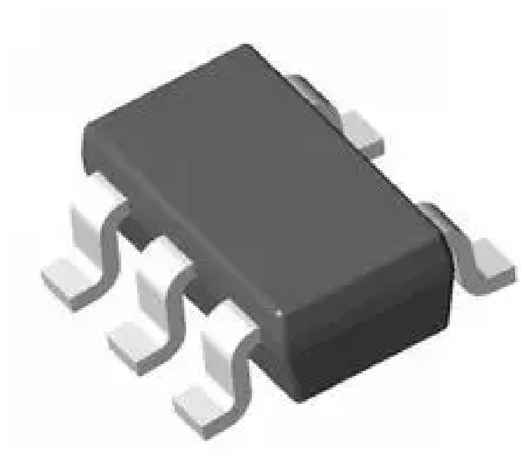


Figure 16: MOSFET driver

Because we are designing around a changing LED function we chose to implement a basic MOSFET driver. This component will be responsible for operating the mood LEDs. The MOSFET driver, also referred to as the MOSFET gate driver, is a specialized circuit. This circuit is used to drive the gate of power.

MOSFETs are for voltage switching applications meaning these components can be logically controlled using external means, such as GPIO pins of our microcontroller unit, specifically the pulse width modulations (PWM). Another feature of a MOSFET is lower the switching time this means the MOSFET will spend less time in the saturation region rather than the ohmic region, this will lower power dissipation in the transistor.

3.4.2.6 Tone Generator

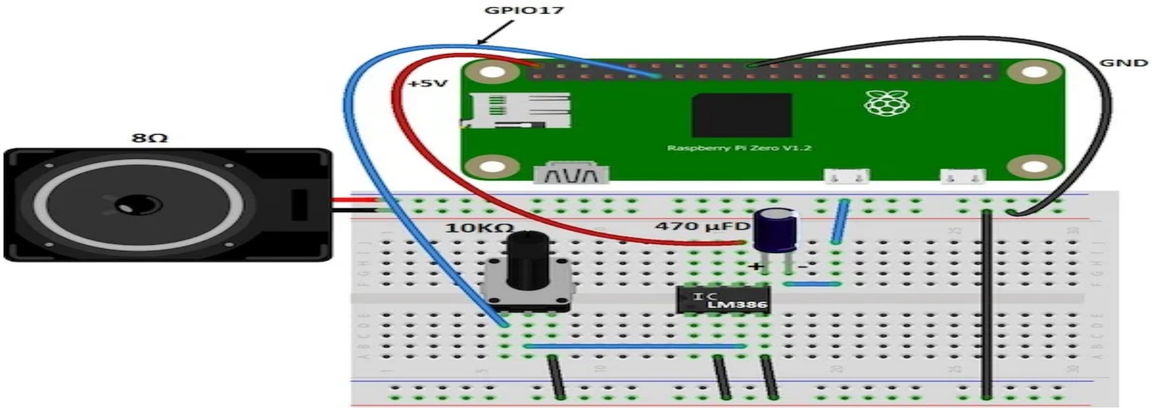


Figure 17: Tone Generator using raspberry pi

For our project to work correctly, the system must be able to generate an alert that is loud enough that the user hears it but not so loud that it becomes annoying to the user. The alert will be triggered to notify the user of our system that the system has detected somebody is present at the front door or elsewhere on their property. Although there are a few ways to generate the sound necessary for the alert notification, we explored using a tone generator to produce the required audio.

A tone generator is a speaker that outputs an audio signal depending on the input frequency that it receives. Since our design will need at least two different types of audio, we can use different frequencies for the various desired outputs necessary for our project. The most common format for a tone generator consists of a potentiometer that is responsible for controlling the frequency that gets imputed to the tone generator. For instance, we can build the tone generator to output a specific audio output for a given input. The input will be controlled by adjusting the potentiometer to a particular value. Then for the tone generator to output the other type of audio output, the potentiometer will change its value to the required signal needed to generate that particular audio output.

For our project, we would find a speaker that can be connected to the circuit, making it possible to generate the necessary output. The speaker used must be loud enough for the application to work as intended but not so loud that the application becomes belligerent. There are many speakers out there that have the potential of fulfilling our project's requirements that can be purchased at an affordable cost.

3.4.2.7 Piezoelectric Buzzer

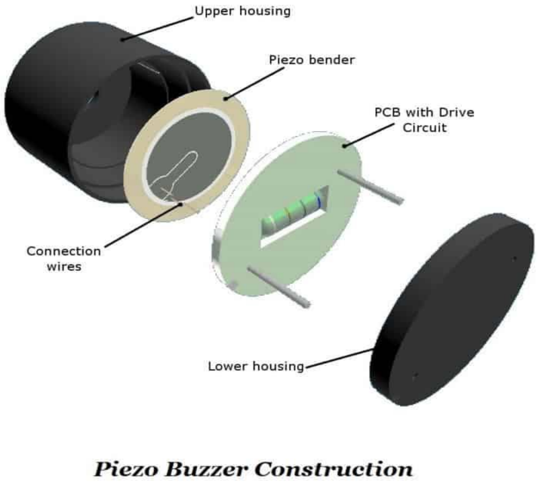


Figure 18: Piezo Buzzer component breakdown

This simple device is used to generate different basic tones. This device utilizes a piezo crystal which changes shape when a voltage is applied to it. The design shown above consists of a piezoelectric element on both sides of a metal plate with a silver electrode which allows for electrical contact. The benefit of this system is it can be used with or without a drive circuit. The piezo buzzers work by utilizing the reverse piezoelectric effect. Once a signal is applied to the metal layer it vibrates quickly causing pressure waves to form which in turn is able to be heard by the human ear. This option is used in a variety of alarms already from home devices to car alarms. The circuit design for a piezoelectric buzzer is the simplest of the options. The only concern with using a piezo buzzer is how loud the speaker tone generated will actually be and the cost of the size that we need. The benefit of using a piezoelectric buzzer is that it does not require files to be sent from the mcu containing the tones we want to generate but rather generates the sound from within.

Technology	**Tone Generator**	**Piezoelectric Buzzer**	**DAC-Amplifier-Speaker**
Part Consideration	Circuit using 555,741 IC	MSE14LTT3	PCM5122
Operational Voltage or Power	4.5-15 V	7 to 14 VDC	3.3V to 5v
Frequency	670-680 Hz	3000-4700 Hz	100-20KHz
Typical Current Consumption	20mA	50mA MAX	150-200ma
dB range	45-70 dB	85 to 95 dB	70 to 90 dB
Input Type	DC	DC	DC
Software Required	No	No	Yes
Cost	~$3	$22.77	~$12

Table 3: Audio Technology Comparison

The reason we chose this specific method is because it gives us more flexibility down the line. This allows us to generate any type of alert we want and can push it out. Using the other methods limits the kind of sound we are able to produce. With the other methods the hub would be unable to generate sound from the video if we chose. The other methods also do not allow for different types of sounds to be produced. They only allow for simple tones, this in turn limits the capability of the project. We felt that if we wanted to have a more practical design we needed to use something that allows us to push our own recordings to the speakers, rather than only having a range of tones that can be generated.

3.4.3 Microcontroller

The Microcontroller is a small programmable single integrated circuit on a single metal-oxide-semiconductor. The MCU consists of one or more CPUs, memory, programmable input and programmable output. The memory is in the form of ferroelectric RAM, EEPROM, NOR flash or OTP ROM with a small amount of RAM. For the scope of this project RAM will need to be utilized so identifying a MCU with a good amount of RAM is important.

The most important function of the MCU is going to be receiving signals from the Raspberry Pi and pushing out the audio packages that are stored in the RAM of the MCU to the speaker system. The MCU will also be sending a signal to the RGB controller and causing the lights to change colors depending on the signal received from the Raspberry Pi. The Microcontroller will act as the middle component between the other hardware components and the SBC. There are multiple MCUs we can use to achieve this but it is important for the MCU to be low cost, and function with a low power footprint since the power will be coming from the Raspberry Pi.

3.4.3.1 ATmega32A

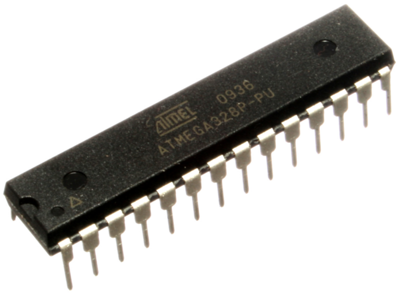


Figure 19: ATmega32A chip

The Atmega32A is a low power 8 bit microcontroller with a high performance CPU. It contains 32 KB of ISP Flash memory with the capability of reading and writing. It uses relatively low power drawing anywhere from 1.8 to 5.5 volts. This component also contains 2 KB of SRAM, 32 general purpose registers and 1KB EEPROM. Communication between devices can be done using its SPI serial port or by using its serial programmable ASART interface.

Each clock cycle the chip performs extremely well in terms of instructions executed. This microchip is capable of performing a throughput of one million instructions per MHZ. For our application to work properly we'll need to use a clock system for executing certain tasks. The ATmega32A has three timers that include compare modes, and internal and external interrupts. Like many other microchips, it includes a programmable watchdog timer with an internal oscillator.

3.4.3.2 RP2040

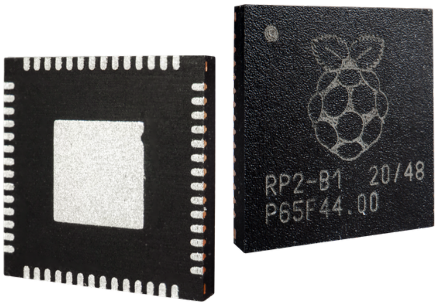


Figure 20: RP2040 Chip

The RP2040 is a strong microcontroller chip. RP stands for raspberry pi, it is within the same family as our raspberry pi 4, but it is not the same as the chip on our actual raspberry pi 4. This being because the raspberry pi 4 is essentially a full fledged computer that is going to be handling extensive functions such as video compression and deal with large amounts of data. A chip like the RP2040 is strong but not strong or capable enough to handle what our raspberry pi 4 is going to handle. The RP2040 is a dual ARM cortex M0 chip, these chips have low overhead and are very quick for simple tasks. Typically you can see these on the raspberry pico boards, which we used to use for breadboard testing.

In regards to the RP2040, the main test will be configuring software to communicate from the raspberry pi 4 to a smaller PCB with the chip on it. There are a few different ways to communicate and interface the main raspberry pi 4 to the RP2040, one of the methods being a UART interface. This interface can be sent over USB and sends serial data that is to be converted to UART. The conversion works both ways so we can send from the RP2040 or receive the data. An alternative data signaling method would be I2C. This protocol would use a master/slave confiruation where the raspberry pi would be acting as a master device sending signals to the RP2040 which would be the slave. This would use a clock whereas UART doesn't need a clock because we can asynchronously signal with UART whereas the I2C needs a synchronous pipeline for this configuration.

Features	**RP2040**	**ATmega32A**	**esp32-s3**
Clock Frequency	133MHz	16 MHz	240Mhz
I2C Buses	2	1	2
SPI Buses	2	1	4
UART Channels	2	1	3
PWM Chanel	16	6	36
GPIO's	28	32	48
Architecture	32-bit ARM COrtex-M0+	AVR RISC 8 Bit	Dual high performance Xtensa® 32-bit LX7 CPU cores
SRAM	264KB	2KB	520KB
Flash Memory	16MB	32KB	16MB
Cost	$1	$2.55	$2.00

Table 4: MCU comparison

3.4.4 LED RGB Controller

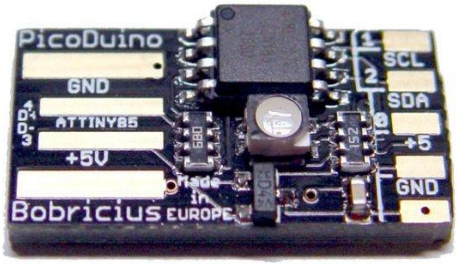


Figure 21: RGB Controller

Another feature of the project is to have a visual representation of detection, to be more specific, we wanted to provide the user with a visual signal when someone was being detected. A simple way to do this would be LED alerts. The understanding of this idea was that when the device is idle power would still be applied to drive enough voltage for still green or blue LED light notification. Once our system detects someone on the premises this light would change to yellow or red. After some research I saw that this could be done using a programmable LED RGB controller or configuring the GPIO pins to create a simple ciuti for running an LED strip.

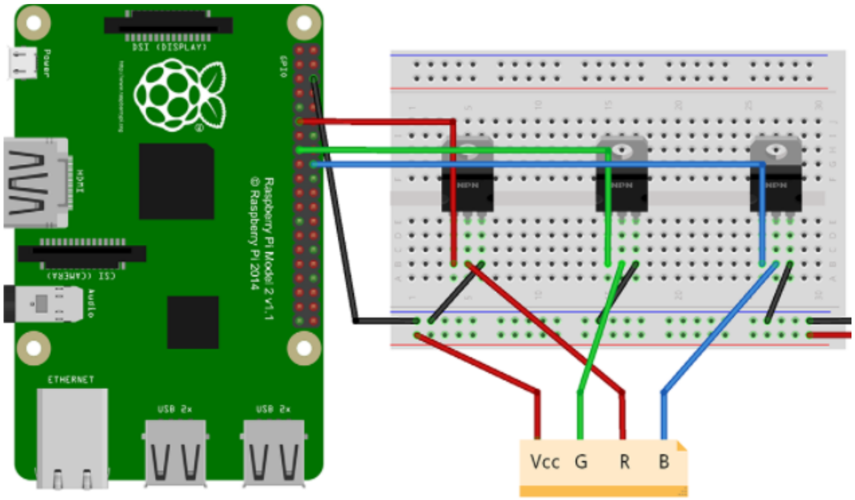


Figure 22: RGB Controller Raspberry Pi configuration

The idea of a simple RGB controller could consist of a LED light or strip, mosfet, load resistor (to keep the excess current from burning out the LED), and a power source. By configuring the LED with the Mosfet gates we can control what colors come on when we program or control the GPIO pins of the microcontroller. The connection for mosfet to LED should be as follows: Gate to GPIO, Source to LED, and Drain to ground.

3.4.4.1 LED Strips

Figure 23: LED light strip

One of the most popular inventions in today's age is the LED strip, made extremely popular in 2017. LEDs strips can be seen in almost anyone's room ages 17-25. The versatility and options with these strips are endless. LED strips consist of a series of red, blue, and green light emitting diodes that in series with each other can be controlled to create color. When current flows through the diode a light is produced, when a different amount of light is produced in each of the 3 diodes different colors are generated. LEDs have many advantages over different light sources due to the low power consumption and the life of the diodes. LEDs require a direct current (DC) because a pulsating AC wave does not allow the diode to emit enough light consistently.

LED strips were utilized in conjunction with a RGB controller in order to change the color of our Hub when an foreign person is recognized. These strips are vital to the Hub because if they are unable to provide a bright light the user may not notice the trigger of this alarm.

There are three options when it comes to selecting a lighting method, we only truly need two colors for the scope of the project we need red and another varying color blue or green. The first option is a RBGW strip that is 5M in adjustable length, a RGB strip that is 5M in adjustable length, and another RGB strip with 12V rating and 5M in adjustable length.

3.4.4.2 LED Comparison

Description	**LED MOD RGBW 5M PER REEL**	**LED MOD RGB 5M PER REEL**	**12V SUPER BRIGHT RGB 5M REEL**

Voltage Forward	12V	12V	12V
Height	10.00mm	10.00mm	2.29mm
Configuration	Linear Light Strip, Flexible	Linear Light Strip, Flexible	Linear Light Strip, Flexible
Type	LED Module	LED Module	LED Engine
Luminous Efficiency (lm/w)	90	90	-
Red	25	25	
Blue	21	2	
Voltage	12V	12V	12V
Price	$1.91000	$2.06000	$2.29000

Table 5: LED light strip comparison and selection

3.4.6 Single-Board Computers

Single-board computers (SBC's) are essentially microprocessors. They contain computer processors that perform the arithmetic and logic required to perform complex computing operations. We are most interested in Single-board computers that are low in cost but high in computing power. The majority of desktop computers run using Windows or macOS operating systems. However, most SBC's that are low in cost and heavily adopted in the scientific and engineering community generally operate using an open-source operating system. The SBC's we are most interested in are limited to a wide variety of Linux operating system distributions and aren't capable of running with Windows or macOS operating systems. Although Linux-based operating systems are not the most popular operating systems on the market, they are known to be potentially more powerful than Windows and macOS due to their open-source architecture. The IoT industry is well known for implementing and taking advantage of these low-cost microprocessors capable of providing the computation power required to carry out complex computing tasks.

We've decided it'd be best to use a single-board computer since our project will process video and analyze it with deep learning in real-time. Since deep learning is a computation-intensive task, using a SBC gives us the required processing power to perform the many computer instructions necessary to operate human detection software. It also provides the critical performance needed to power the software responsible for running our desktop application to display the video camera's live footage. The SBC we use will act as our project's main component that communicates and controls the behavior of its external hardware. Most microprocessors are particularly good at running multiple processes simultaneously, especially ones with multicore central processing units. Since our project will be performing various tasks concurrently, we believe using a SBC with a multicore microprocessor is the best design choice for maximizing the performance of the application.

3.4.6.1 Raspberry Pi 4 Model B

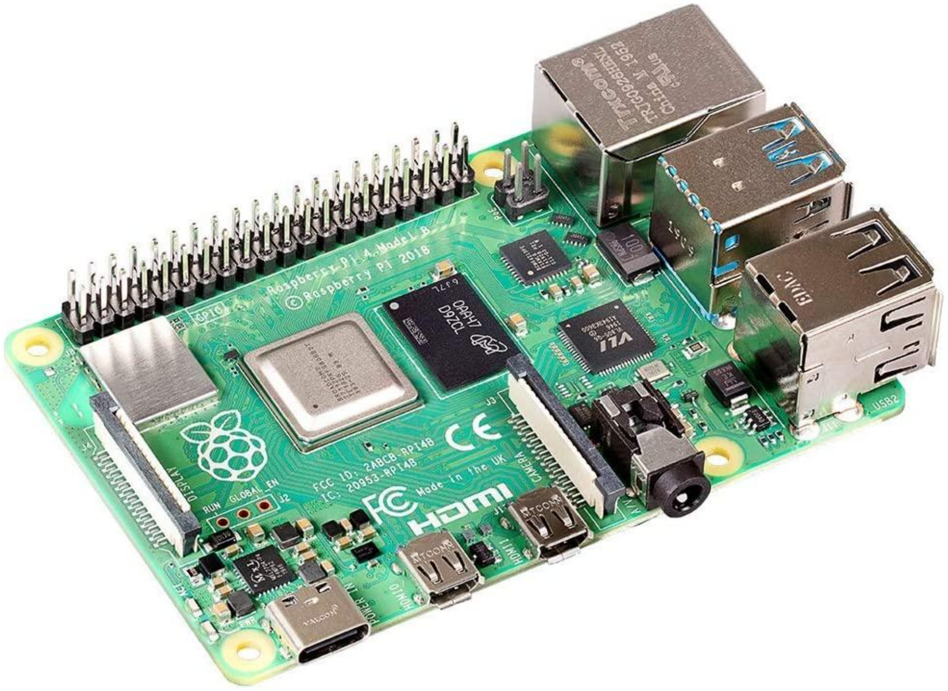


Figure 23: raspberry pi 4

Since 2012, Raspberry Pi's official release date, it has been one of the most commonly adopted SBC's used for building IoT applications. It's also widely used as a fully-functional desktop computer capable of running a Linux-based operating system. Raspberry Pi as a company offers a variety of different devices, all of them providing different specification tradeoffs. Due to its cost, physical size, and performance, we conducted further research on the Raspberry Pi 4 Model B to act as the central hub of our project.

The Raspberry Pi 4 Model B uses a 64-bit architecture that utilizes a 1.5 GHz quad-core Cortex-A72 (ARM v8) processor. The SBC's dynamic memory is available in 1GB, 2GB, 4GB, and 8GB models. Since our project will require a large amount of data processing due to its human detection capabilities, we used the 8GB model to ensure our project runs as fast as possible.

The Model B comes with a standard 40-pin GPIO (General-Purpose Input/Output) header that acts as an interface for the Raspberry Pi to control and communicate with external electronic circuits. The GPIO header can handle inputs and outputs, giving the Raspberry Pi the flexibility to add a wide variety of sensors and switches. The board also comes with plenty of available ports to connect external devices. These ports include but are not limited to two micro HDMI ports, a display port, and a camera port. The board uses the H.264 and H.265 protocols for processing videos. The H.264 protocol is capable of encoding and decoding video, while the H.265 protocol is only capable of decoding video. H.264 provides an efficient method for compressing the video and audio (encode) and sending the compressed data over some type of network. It can also accept the compressed video over a network and decompress it back to its original state (decode).

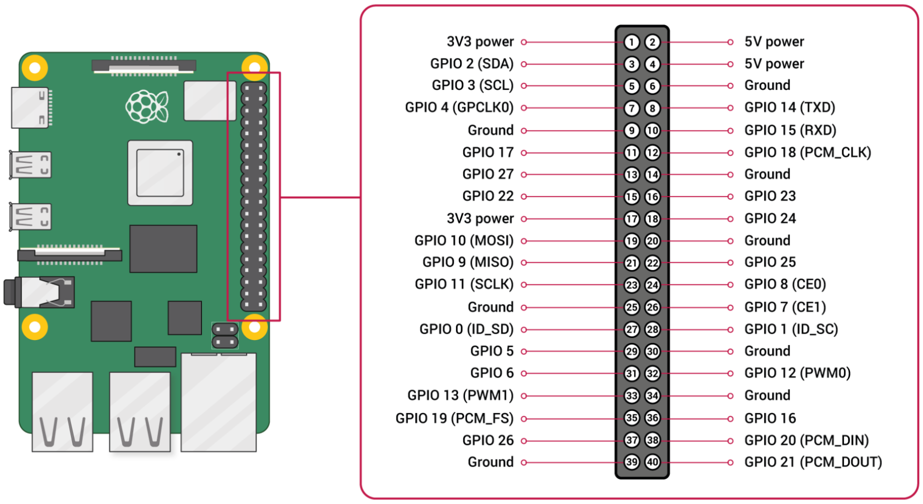


Figure 23: GPIO header

The Model B can run various operating systems. We're choosing to use Raspberry Pi OS, the default Raspberry Pi operating system formerly known as Raspbian. Raspberry Pi OS is Debian-based; specifically, the version we're using is based on Debian version 11 (Bullseye). Since the operating system is designed specifically for the Raspberry Pi, we believe it's the best design choice for maximizing performance and reliability. There is both a 32-bit version and a 64-bit version of Raspberry Pi OS. Raspberry Pi OS offers two 64-bit versions, Raspberry Pi OS Lite (doesn't include a desktop version) and Raspberry Pi OS (consists of a desktop version). We used Raspberry Pi OS (desktop version) because our project will consist of a GUI that controls the surveillance hub and displays the camera's views.

3.4.6.2 NVIDIA Jetson Nano

Figure 24: NVIDIA Jetson Nano

The NVIDIA Jetson Nano’s official release date was back in March 2019. It has ever since increased in popularity and has been widely adopted by the scientific and engineering community. It is specifically designed for embedded systems applications, IoT-based applications, and training artificial intelligence models. The Jetson Nano is listed at an affordable price of $99 in the United States.

This powerful single-board computer uses a 1.42 GHz quad-core ARM Cortex-A57 processor that runs on a 64-bit architecture. It also contains an NVIDIA Maxwell architecture graphics processing unit (GPU) with 128 NVIDIA CUDA cores. It offers two different models that include dynamic memory composed of 2 GB or 4 GB of LPDDR4 that operates at 1.6 GHz and 25.6 GB/s.

The NVIDIA Jetson Nano contains the NVIDIA Jetpack SDK that supports plenty of artificial intelligence frameworks such as Tensorflow, Pytorch, Caffe, and Mxnet.

The board uses a microSD card that most commonly operates using Linux4Tegra, a Linux-based operating system based on Ubuntu 18.04. Although Linux4Tegra is the most widely used operating system, it is capable of running on a wide range of other Linux-based operating systems.

3.4.6.3 ASUS Tinker Board S

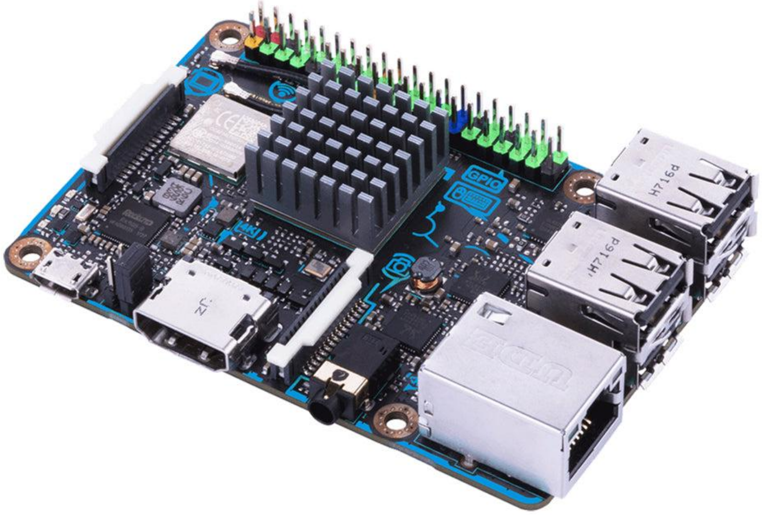


Figure 25: ASUS Tinker Board S

Launched in 2017 the single-board computer is designed to be a fully functional computer that offers class leading performance without leveraging hardware compatibility. The board is made for IoT.

The modern quad-core ARM-based processor improves performance and provides flexibility to a variety of projects. The SD 3.0 allows significantly faster read and write speeds. The GPU allows for a high quality image processing and HD & UHD playback. In relation to the project this would allow the video playback to be smooth and quick on the hubs end.

The HD audio quality also differentiates itself from a lot of other SBC boards. The Tinker board features a HD codec that supports up to 24-bit/194kHz audio. The interface uses audio jacks which support audio output and audio input, without additional hardware.

This SBC includes popular applications such as IDLE/Python & Squeal/ Scratch. This board supports Android Operating System which allows media playback.

Rockchip RK3399
2X Cortex-A72 @ 2.0 Ghz
4X Cortex-A53 @1.5 Ghz
Mali T860 MP4 GPU @ 800 MHz

LPDDR4 2GB-4GB
16GB eMMC - Micro SD Slot
802.11ac Wifi & BT 5.0
12~19 V
Android 10 - Debian

Table 6: Tinker Board Features

3.4.6.4 Single-Board Computer Selection

After researching all three of the single-board computers we chose the Raspberry Pi 4 Model B. The NVIDIA Jetson Nano and the ASUS Tinker Board S can provide the necessary computing power to accomplish the behavior required for our project to excel. The main reason we're choosing the Raspberry Pi is due to its low price and because it has the most extensive memory consisting of 8 GB.

Since this is a product designed for users to adopt, we want to make the system as low cost as possible without affecting the application's performance. Although the other two boards are good options for implementing our project's design, the Raspberry Pi will lower the cost of the overall system while maximizing its performance. Even at 99$, this is an affordable component for building our project. As of a couple of years, due to coronavirus, the price of boards like these has increased due to chip manufacturing issues. So it's possible that in the future, the price could possibly drop below 99$ once manufacturing becomes more stable.

It is crucial that our application must have extremely quick response times. To guarantee our application is fast, our board must have a large dynamic memory. The Raspberry Pi has 8GB of memory to help us achieve fast application response times due to caching.

There are some features for our project that all three boards could provide. For example, we need to have a relatively small single-board computer. But since they're all relatively small in size and contain 40 GPIO pin headers, these specifications were not the deciding factors for which board we chose.

Device	Raspberry Pi 4 Model B	NVIDIA Jetson Nano	ASUS Tinker Board S

Price	$75.00 (8GB model)	$99	$150
Size	3.4 x 2.2 inches	2.76 x 1.77 inches	3.37 x 2.125 inches
CPU	Quad-core Broadcom BCM2711 Cortex-A72 (ARM v8) SoC	Quad-core ARM Cortex-A57 MPCore processor	Quad-core Rockchip RK3288 Cortex-A17 SoC
GPU	VideoCore IV	NVIDIA Maxwell architecture with 128 NVIDIA CUDA	ARM Mali-T760 MP4
Memory	Available in 1GB, 2GB, 4GB, and 8GB of LPDDR4 SDRAM Models	4 GB 64-bit LPDDR4	2GB Dual-CH LPDDR3
USB Ports	2x USB 2.0 ports, 2x USB 3.0 ports	1x USB 2.0 ports, 4x USB 3.0 ports	4x USB 2.0 ports
Display Ports	2x HDMI display output up to 4K 60fps	1x HDMI 2.0 and eDP 1.4	1x HDMI with CEC
Number of Pin Headers	40	40	40
Video Decoding	H.264 hardware decode (up to 1080p 60fps), H.265 (HEVC) hardware decode (up to 4K 60fps)	H.264 hardware decode (up to 1080p 240fps), H.265 (HEVC) hardware decode (up to 4K 60fps)	H.264 hardware decode (up to 4K 60fps)

Table 7: SBC comparison and selection

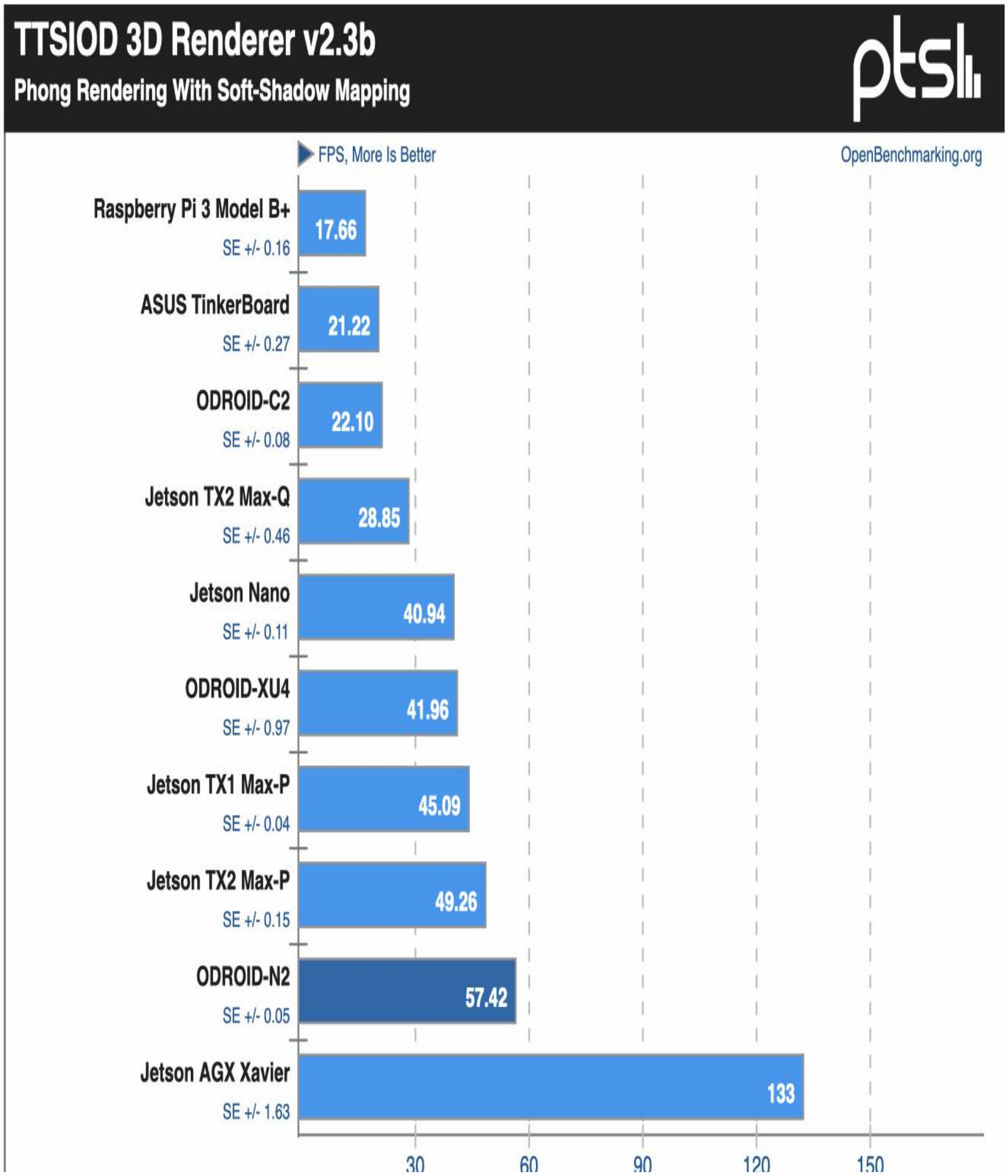


Table 8: Graphics comparison

3.5 Architecture and Related Diagrams

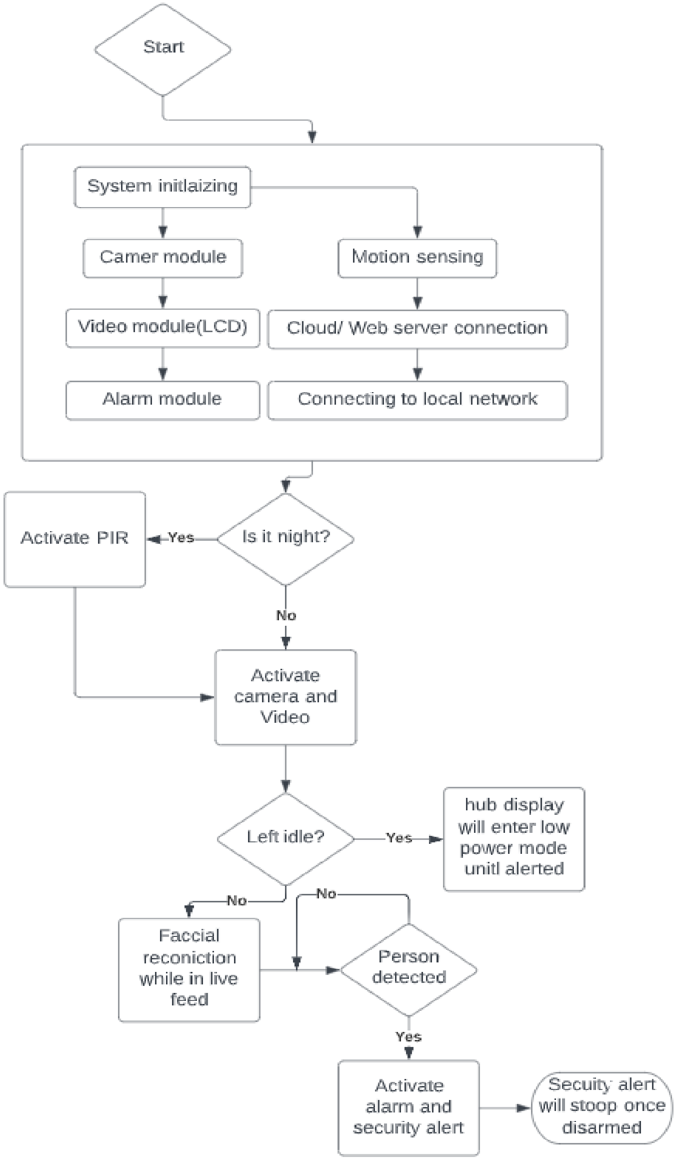

Start
System initlaizing
Camer module
Motion sensing
Video module(LCD)
Cloud/ Web server connection
Alarm module
Connecting to local network
Activate PIR
Yes
Is it night?
No
Activate camera and Video
Left idle?
Yes
hub display will enter low power mode unitl alerted
No
No
Faccial reconiction while in live feed
Person detected
Yes
Activate alarm and security alert
Secuity alert will stoop once disarmed

Figure 26: Architecture flow chart

This hardware diagram illustrates the connections between subsystems that form the single product. Every line directs the flow of data from the front to the endpoints of the product. There will be two main components after all said and done, the IP camera and the hub. The camera can be any off the shelf IP camera that streams an industry standard RTSP stream, while the hub will be the smart DVR that will be plug and play with the IP camera and will be housed in a singular unit that will be designed and 3d printed. Since the product is receiving information from the IP camera, this will be the start of where the data is taken.

The camera will be receiving information, this information being the camera stream. This streamed data will be sent over the network, which in our case and most cases will be over the wifi network, although it can be sent over a hardlined ethernet cable if the user desires. The IP camera and Smart Hub must be on the same network. If they are not on the same network, then the hub will not be able to connect. This is because the hub uses the camera's RTSP stream which will only display on the same network. The network is the median between the camera and the hub in this architecture.

The hub consists of the main component, the Raspberry Pi 4. The Pi 4 is acting as the main computer for the hub, which will take and receive all the computational functionality from the camera and it will decide what to do with it. The Pi initially will receive the RTSP stream from the IP camera, and this will allow the user to be able to see the stream. This will also foremost ultimately be used in coherence with another software to determine human detection. The user will have an application that they can see the RTSP stream from the camera which will be a live stream of what the camera sees. This application can be seen on either a monitor or the hub's built-in LCD screen, whichever the user chooses.

The firmware will analyze and dissect the stream using the A.I. software to determine if an object outside is a human or not; since this is a security system, we are only concerned about strangers and humans that are where they should not be. Things like cars or animals can be implemented but on this initial version of the project, we only care about humans. When the firmware of the application analyzes the data from the IP camera, it can then send this data to the MCU that will be the PCB notification feature. This MCU will be sent data from the Pi, through the firmware's functionality to alert the user if there is a threat. There will be two main components of this notification PCB. The audio cue that alerts the user, and an idling mood light that will be certain colors depending on the 'threat level'. Colors more aggressive like red will mean that there is a threat that is currently happening, or extremely recently just happened. Colors more calming such as blue or purple will mean that there is no threat and things have been, or are currently calm. All of this can be displayed below in Figure 2.

The software diagram illustrates the flow of execution between the data from the start to the endpoint. As previously mentioned, the start point is the moment that the IP camera records and streams the video feed, if this camera is not on then there is no flow of data or start point for the Smart Hub to function. In the software diagram, there are two components, the hardware and software.

The hardware is only there to fill in the gaps of this diagram, whereas we go into greater detail in this diagram on how and what software is truly needed to execute the overall operational goal. The camera will get the data and encode it to send over the network. There will most likely be some sort of SSH network protocol used to communicate between the Smart Hub and the camera. We need the video data encoded for fast and effective streaming capability. Without it being encoded, the data will be too big to send and it will have latency or resolution issues because the bandwidth or throughput will be suffering tremendously.

The industry standard of video encoding is using the H.264 codec which is typically a separate electronic device on a camera that handles it instead of the main processor, in order to alleviate the stress of a processor. Since we are encoding the data to send over the network, we decoded it on the hub's endpoint after it is transferred over the network. This data will immediately be sent to the application, which will have the capability to use A.I. for human detection, to display the stream and to ping the notification PCB over the Raspberry Pi 4's pin headers. These are the core functionalities that will be on the first version of the product, with additional stretch goals desired depending on the time constraints.

3.6 Communication Interfaces

There are four common communication protocols commonly used for electrical communication between different devices and integrated circuits. It is also important to note that these protocols are considered low speed communication therefore there is no need to worry about impedance control or transmission line behavior as the data is not sent at a high enough rate to need to factor these things in. Here we discuss what each of the communication interfaces are and decide what is appropriate to use with our MCUs and peripheral devices.

3.6.1 Universal Asynchronous Receiver-Transmitter

UART is one of the most commonly used device-to-device communication protocols. It is a communication protocol that uses only two wires to enable one device to communicate with another device. The two lines are usually labeled Tx and Rx, which stands for the Transfer and Receiver wires. UART uses bidirectional, asynchronous, and serial data transmission meaning that data can be transmitted as well as received by either device. Each wire is going to only send data in one direction, namely from the Transfer pin of one device to the Receiver pin of the cooperating device. Therefore since each of the 2 wires involved in the UART communication protocol only send data in one direction, the wires themselves are unidirectional.

That is, UART can function as a full-duplex communication protocol since data may be transferred in both directions at the same time . It is vital to remember that the hardware must be full-duplex in order for the UART to work. have a dedicated transfer and receiver buffer if the hardware employs a The system would be considered if there was a common buffer for transmission and receiver data. The term "serial data transmission" refers to the transfer of data one bit at a time. In contrast to parallel data transmission, data is transmitted from a single pin on the transmitting device. that delivers a large amount of data all at once.

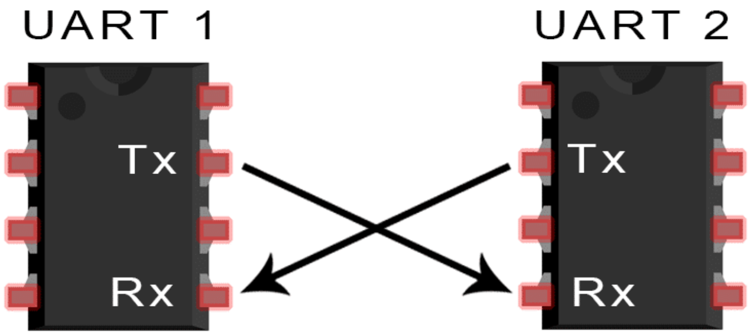


Figure 27: UART protocol

3.6.2 Inter-Integrated Circuit (I2C)

I2c is a protocol that benefits from being synchronous and operating on the same clock speeds but its main use is that it only requires two wires to operate. This means although there can be many devices on the I2C bus, an engineer or designer only needs to use two total wires to connect them all together. The Two wires are usually noted as “SCL”, short for serial clock, and “SDA”, short for serial data. Both of the signal wires on the I2C bus are bidirectional, meaning data can be sent in either direction. Because data cannot be sent in both directions simultaneously, I2C is half duplex.

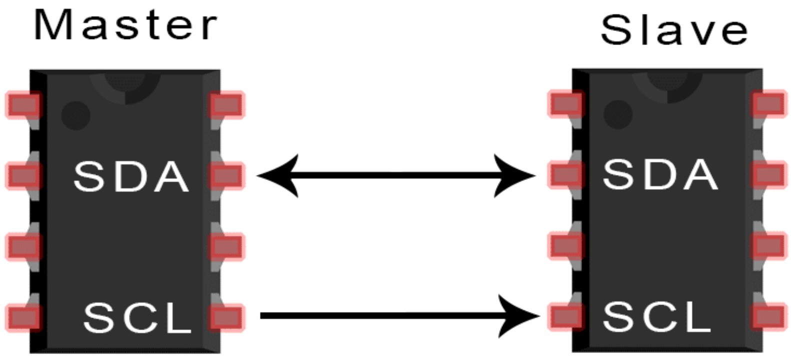


Figure 28: I2C protocol

3.6.2 Serial Peripheral Interface (SPI)

SPI is a synchronous, serial, full-duplex protocol that is simple to implement in hardware. SPI is a single master protocol that connects master and slave through four primary lines. SCLK, MOSI, MISO, and SS are the wires that stand for Serial Clock, Master-Out-Slave-In, Master-In Slave-Out, and Slave Select, respectively. Each slave that is added to this setup requires just the addition of another SS, slave select, line to the interface. Because the clock signal, SCLK, is delivered from the bus master to all slaves on the bus, this interface is also synchronous.

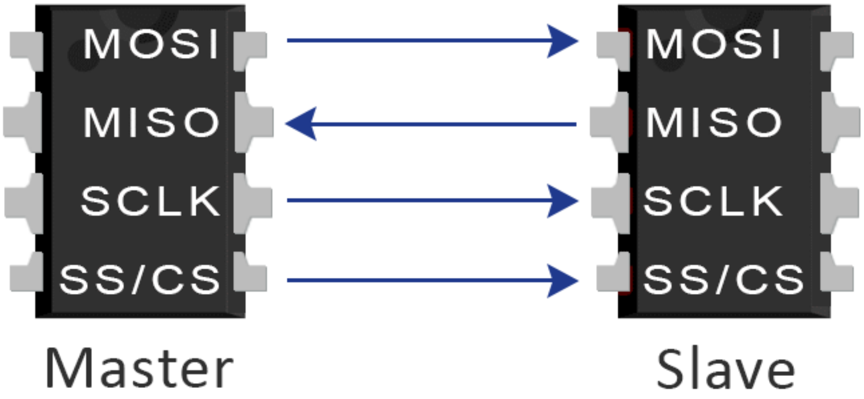


Figure 29: SPI protocol

3.7 Part Selection Summary

In order to reflect our overall goal of having a cost efficient security system, the overall goal is to make the price point of the product affordable to all. This reflects our mission statement of making security possible for everyone, especially coupled with the ability to pick and choose your own camera's to append to the system. So with cost efficiency in mind, the parts chosen will be effective at their job giving people ease of mind and will also be affordable. As we get closer to a more ironed product we discuss more about the specifics on what was chosen and why.

4.0 Related Standards and Realistic Design Constraints

Throughout this subsection a variety of the constraints mentioned above will be further investigated in greater detail. Constraints limit how much of the product can be developed by us and how much of the project is going to be third party completed parts such as a LCD screen or LED lights. With more resources and less constraints these are things that could be developed by us and therefore delivering a more original product. Over this section the constraints that will be further explained will be time, budget, manufacturing, power, size, availability, environmental, political, social, and ethical.

4.1 Standards

This section will discuss a few standards in relation to the Smart Surveillance Hub. The standards will cover LEDs, software, power, PCB and soldering. Each standard affects the design of the hub. Standards are important in engineering and we must discuss them because they promote safety, reliability, efficiency and overall the biggest thing is that they help with productivity for any and all industries that rely on engineering components along with engineering equipment. It was important for us to realize, know, and understand the standards that we used because without them we are sacrificing all the previously stated strengths simply due to lack of effort or acknowledgement.

4.1.1 Hardware & Electrical Standards

The board design we are using communicates mainly through the GPIO ports to communicate with the Raspberry Pi device. The Raspberry Pi utilizes the use of a 2 by 20 design on the side of the board and connecting these pins to the pins of the DAC. We soldered the GPIO pins of the DAC to the RGB controller then to an RGB strip. To ensure all the connections are secure we followed the IPC soldering standards IPC-J-STD-001 which explains the standard for soldering PCBs. This standard states the requirements that the pin be fully wetted Concave Meniscus Pad well covered and the full pad covered. This standard also states that there should not include one side of the pad not fully covered, the solder not touching the pad, the solder sitting on the top of the pin, too little solder, pin not wetted, or too much solder. By following these guidelines for the soldering it will ensure a secure connection. Also the standard states that we should use lead free solder to reduce toxicity. To ensure the best method is used, each pin header should be connected to its port then a drop of solder is applied to the top of the connection. Every pin should be heated prior to soldering to ensure the solder covers every connection thoroughly.

The next standard that will be utilized is IEEE Draft Standard for System, Software and Hardware Verification and Validation - Corrigendum 1. To ensure the Smart Hub works as intended we used the verification and validation (V&V)

processes to determine that our product does what is intended and meets its stated use and user needs. The V&V processes include systems, software, hardware and their interfaces. The standard would need to be utilized in order to take the product to market. The full scope of this process includes the firmware, microcode, each of the terms system, software and hardware. The full process will include analysis, evaluation, review, inspection, assessment, and testing of the product. This standard basically lays out everything that will be discussed throughout this paper, but in order to take to market will need to be verified.

4.1.2 Software Standards

This project has a lot of moving parts that will require a lot of different standards. As previously discussed we require a few things for this project to work, including WiFi integration along with other networking standards in order for things to simply communicate to each other and integrate the different parts. Things that we have standards for can be how certain components within the system work and their standards. This can be things like the WiFi, the encoder or even the interfaces used to display on the LCD. Also software standards that are important but not nearly as relevant is the feel and flow of the application. The goal is to create software that has a similar feel to consumer products that people are already familiar with and not to create something so new that no one knows how to operate it or feels comfortable using it. That will deter users from these actions and ultimately be less user friendly.

The most important overall standard is the OSI stack. Essentially the purpose of this stack is to guide technology with how the current standards are for how technology communicates with each other and handles data. Without the simplest understanding of this stack or a layer not working, the data cannot go from one the start point to the end point; but that is contingent on the start and destination of the data.

A typical flow of data from a person's computer to a web server utilizes the entire stack and is very important. For our purposes we used the entire stack including all the layers from the start point of the data transmission to the desired endpoint. The start point in our model is the camera, it receives data and the end point being the computer where the application is on. The entire communicative process is handled by the OSI stack and that is why it is the main standard. As seen below in the image we will go over the individual layers and what purposes they serve for our software.

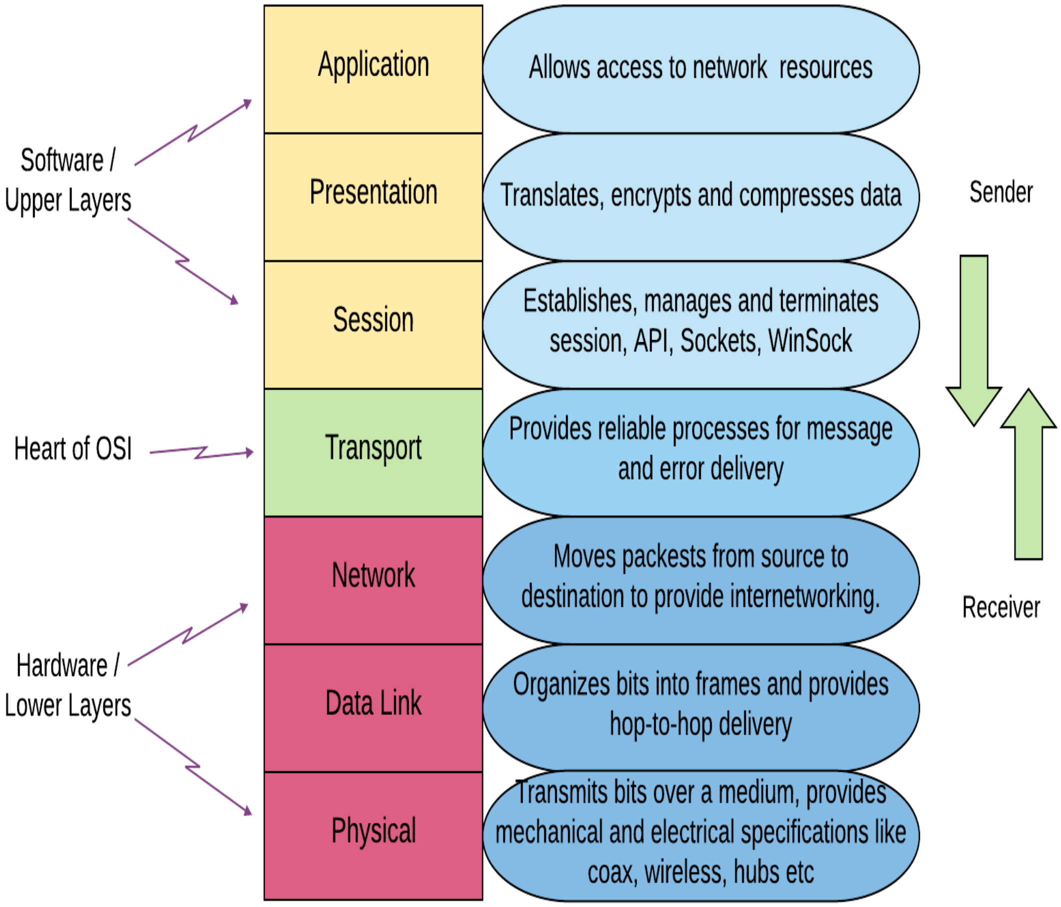

Software /
Upper Layers
Heart of OSI
Hardware /
Lower Layers
Application
Allows access to network resources
Presentation
Translates, encrypts and compresses data
Session
Establishes, manages and terminates session, API, Sockets, WinSock
Transport
Provides reliable processes for message and error delivery
Network
Moves packests from source to destination to provide internetworking.
Data Link
Organizes bits into frames and provides hop-to-hop delivery
Physical
Transmits bits over a medium, provides mechanical and electrical specifications like coax, wireless, hubs etc
Sender
Receiver

Figure 30: OSI layering

Referring to the image above, that is the OSI stack, we will break down some of what the layers are and why all 7 layers are very important and crucial for our project along with some of the current standards within the industry and how things are properly used.

Starting with the application layer, this is the closest the user can get to control of the software. For most users, this can be a desktop application or a web application. For developers and on the more technical side of things this can be software that is interfaced with a coding layer to an API. After all, coding is simply a way to communicate with the computer to tell it what to do but more precisely and with more control given to the user. An application with a user interface that gives the user options and buttons is simply a way for a user to interact with code but with a simple visual interface instead of coding; clicking a button on a web page is the easiest way for a user to call a method or function that responds to code. For our case, the application will be a desktop application that the user will be interacting with and is a simple user interface that is created using code that we engineer and gives the user a way to discreetly operate code with the use of simple forms that will hold and send data across the stack. For a lot of users this is the most used layer that they will see and interact with, along with the network which we will discuss in a few layers from now.

The next layer is the presentation layer, which is very important on the client side of things as seen in the image, described as an upper layer. The presentation layer is essentially as it sounds, it presents the data in a visual format otherwise known as the user interface. There is not much to this other than the HTML, CSS and Javascript which operates as the front end user interface is formatted in this layer to show the actual interface and layers between the user and the core functionality that the application gives the user. So essentially this layer presents the data that the application yields for the user.

The socket layer is important in terms of how and where the actual application is going to be hosted. The internet protocol address has 16 bit port addresses which means there are over 65,000 ports on a computer. The ports along with the network internet protocol address creates a websocket. Putting an application like the client or the server on this websocket is crucial for our use and will be the last user based layer before the data is transported across the network. For our purposes, there will be a client with its own individual socket and the server with its own individual socket. They must be separated because you cannot give one port multiple services which will be discussed later in the REST API software section in more detail.

The transport layer handles the transfer of data from the end systems. It controls things like how much data, the speed, throughput and data rate along with other analytics. There are two main protocols to transport data, the transmission control protocol (TCP) and user datagram protocol (UDP). TCP is a safer protocol that handles data transfer at a slower rate, but it is more optimal in making sure the data being transferred reaches the endpoint without any sort of issues. UDP however is known as a ‘hopeful’ protocol, where it sends data but there is no backup to provide the data if it doesn’t reach the end destination it is being sent to. Although TCP is more secure, for our case we do have a large amount of video data being sent over the wire and to make it as close to real time streaming as possible then UDP will be utilized as it is a live streaming standard and has less overhead in order to send the data over the network. If we happen to make our stretch goals then TCP will be used for smaller packets like password encryption but this is a stretch goal as it stands and we might not make it to that.

Next layer down the pipe is the network layer. This is where the internet protocol factors in, with router to router endpoint communication. Network layer is very important for packet forwarding, which is when there are pit stops for the data across the internet, like a mesh network, where routers are in congruent with other routers to send data from a large geographic distance between them. For our purposes, there will only be 1 router involved with no packet forwarding since everything for the base deliverables requires us to only use the one single local network and not have to connect to external networks.

The data link layer is out of our control and we do not really have anything to do with this layer, besides benefit from its existence. Overall this layer provides node to node data transfer and handles error correction on the layer below this which will be the physical layer. Most importantly this layer handles the LAN switches which also can filter and forward traffic on its predefined ports.

The most important layer is the physical layer. This layer is essentially the physical median that handles how and what the data is being transferred over, along with the electrical aspect of how this is actually going to transfer data. This can include things such as radio transmissions, wireless or even just through cables like ethernet or fiber optics. This requires things such as proper voltages and pins. For our transmission we used a wireless interface protocol otherwise known as Wi-Fi which will be the ultimate median between the upper layers to the lower layers. As for the camera's, the ones we have require ethernet usage which will connect and transport data directly from the router to the camera and from the camera. Connecting both the hub and the camera on the same network is a must for our case.

In the following sections we will discuss what we are using to fill in the layer for its respective place on the OSI stack and what their own standards are along with what they are in general and their purposes within the software development side of things.

4.1.2.1 WiFi Standards

WiFi has been around for a while and is one of the more commonly used network communication methods opposed to more traditional methods such as ethernet cabled. IEEE 802.11, otherwise known as WiFi, is a set of local area network (LAN) standards which uses both a MAC address (media access control) and physical layer protocols in order to implement the WiFi network. WiFi uses multiple frequencies, like the traditional 2.4 GHz network that is most commonly used within phones, to 5 GHz which is newer to phones but older for computers, to even upwards to 60 GHz. WiFi was initially released for consumers back in 1997 but wasn't commercially sold until August of 1999. For a while it was not as popular but eventually it became the golden standard used today virtually everywhere.

Within the project, the smart-hub will utilize a WiFi connection to interface with the network. The advantages of WiFi over the traditional standard ethernet cable is the portability. Since WiFi requires no cables and everything is over radio frequencies, a portable smart-hub is best because it can be used anywhere within the distance of the radio frequency and not be forced to be near a router, which gives the user more freedom within how they want their security system to operate.

4.1.2.2 H.264 Video Encoding Standards

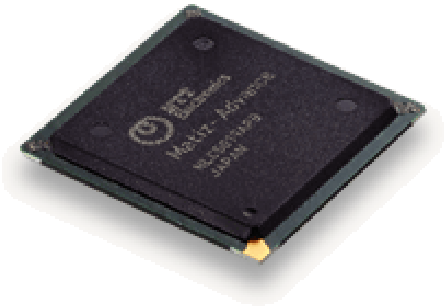


Figure 31: Video Encoder

Founded and created back in 1998, the first video codec was created which is the H26L. The original goal was to simply double the coding efficiency and was done by the Video Coding Experts Group otherwise known as VCEG. Generations and years later after the H26L, the state of the art codec widely used throughout the industry is the H264 codec. It is a crucial part in our project and discretely utilized within all off the shelf IP camera products due to the constraints that RAW video data has on a network.

In order for the video data to be sent over the network without bogging down the network, the best practice is to encode the data which is similar to zipping a file. It converts the RAW video files to digital files which are multiple magnitudes less in size and ultimately that takes stress off of the bandwidth while giving better throughput within the network. It is the most commonly used for recording and compressing video feed by a vast 91% majority of developers as of 2019. It can support resolutions from 720p all the way to 8k UHD.

Original intent of the H264 was to provide good quality with lower bitrate while being cheap to implement on the network. Typically the H264 architecture is used for lossy compression but can be used for lossless regions within lossy pictures but that is extra work for the framework to initially do.

4.1.2.3 User Interface - *ADA*

User interface is very crucial when developing software that has any interface at all. There are things to consider such as simplicity, scalability and useability which are some of the main things to think about when designing something that is interactable with the user. When developing frontend user interfaces, you must account for all types of users from across the spectrum of people. These people can be seen as profiles with things that can make up for them, such as people who are seeing impaired or simply not familiar with computers and software. This is why having some sort of standards in terms of user interface designing and engineering is important and something to consider when making a design.

In terms of who is going to use this product, it is marketed and used for anyone. Since the product will have audio and visual cues then anyone who is either hearing impaired or has impaired sight can benefit from one or the other. Also another handicapped group of users that need to be accounted for is people with colorblind issues. There is the Americans with Disabilities Act (ADA) that has set standards for what should and should not be used in terms of color combinations for people who are visually impaired in terms of color blindness. This will help us in determining what should and should not be used within our user interface. Colors like a dark gray background with a light orange text would contrast and allow the most amount of users to see without any challenges. Accounting for these people is a large market and color combinations mean very little to us therefore it is a great consideration and strong standard to utilize upon designing the user interface.

When designing a user interface, there is no guide upon what people use when putting certain features or aspects within certain areas of the applications. Over the years when it comes to websites, there are similarities that make the usability easier like navigation bars at the top of a page to move around a website. These have been adopted and became standards for commercial use websites and such. Our goal is to make a user interface that has all the features we need while making it easy and simple to use. We took inspiration from other applications when designing this and mold it into our own user interface. Designing something that people are already familiar with instead of breaking the mold and creating something entirely different from familiar standards is the ultimate goal. We do not want users to question how to use the application but rather feel comfortable and free when using it.

4.1.2.4 User Datagram Protocol (UDP)

This protocol was initially designed in 1980 and heavily used within the industry. This is best for fast communication within a network. This is because the alternative which is transmission control protocol (TCP) requires a response from the server every time you want to send a message whereas UDP does not require a response.

A standard of network communication between our RTSP stream and the application will be user datagram protocol. There are two main communication protocols, which are UDP and TCP. The core difference is how they obtain and maintain the connection. For our case, we want the fastest communication protocol since we are sending a lot of data over the wire. With this in mind that makes UDP perfect for our use. This standard is also beneficial when it comes to time-sensitive communications, which is because with UDP you only have to send the data and we do not need any sort of response from the server before sending the data. We simply send the data to the desired endpoint and hope that it ends up there with no additional overhead.

4.2 Realistic Design Constraints

For now, the main constraint for this project is going to be price. In order to develop a smart hub that is cheap and affordable, it would be more beneficial to design and manufacture our own touch screen led screen. Since time is an issue and there are many other aspects of the project we are designing, we decided to buy a touch screen that currently works. If this project were to be marketed, a cheaper touch screen led would have to be found in order to reduce the price as much as possible and make it as marketable as possible. Although time and money is one of our constraints, we are still making it one of our top priorities to design a product that is affordable for users. Implementing our project using a Raspberry Pi makes this possible since that is the brain of our project, and they are relatively inexpensive. Since COVID-19 the price of a Raspberry Pi has increased, but it’s still low enough in cost to build a product affordable for users to purchase. As previously discussed, the LCD display will be the most costly part of our project's components.

Another constraint is the need to figure out how every single camera works and creating a universal hub that works for every camera. Each camera has its own design and might not be universal to what we are creating. This is especially true since our target micro controller we used is an 8GB Raspberry Pi4 B model. More software may need to be developed in order to become a universal hub for any camera. But for the most part, our project will be capable of integrating with any IP camera that does not have any type of vendor lock in configuration details.

Time can also come into play when developing this project, the hardware design and prototyping is going to take a bit of time, and that is going to be needed to test the software. Many parts of this project need to be built in different pieces, tested, and combined after all have been tested. A Better illustration of the work needed for the project can be seen in the specification table (Table 1) and in the Project budget (Table 2). We had enough time to build the product that is intended.

4.2.1 Time

Time was the biggest limiting factor when it came to this project. In roughly six months, the product had to be researched, developed, tested and thoroughly documented. The research takes up a majority of the time and really limited the ability to thoroughly test the project. With more time, different opportunities could be explored, such as developing different parts that originally are being purchased, such as the LCD screen or the LEDs. Developing more of the project would allow for more money to be saved by not only us but by the consumer as well. In reality, if there was more time, multiple prototypes could be developed and investigated, which would result in a greater retail product. We are also short a month since we are completing this project during the Summer semester opposed to other senior design groups that generally complete this over Fall and Spring, giving them an additional four to five weeks. The six month time constraint is the reason we might not be able to implement as many of the stretch goals as we'd like.

4.2.2 Budget

Limiting the budget to around $600 causes the project to be as simple as possible. This budget only allows us to test one method of design. With a larger budget, more designs could be explored, and ultimately a better product could be developed. This budget allows us to buy a LCD screen as well as a Raspberry Pi and leaves a little room for a few other important parts. With this budget, we are aiming to create the cheapest product for our consumers. Certain parts are chosen because they are easily implemented into the project when in reality, we could have gone with a cheaper option which would have made the project more complex to create. It's a simple trade off of time vs money, and we chose saving time by spending more money. Since we have four team members, it shouldn't cost more than $200 each after the project is completed and ready for its demo.

4.2.3 Manufacturing

Access to large scale manufacturing is not available. Many of these companies have the ability to manufacture their own product. Since this product is a prototype, it is not much of an issue with manufacture since we are purchasing each part individually. If this product was taken to market, there would be serious issues relating to the purchase of each part. This also causes our retail price to be more expensive, with access to our own manufacturing method it would significantly lower the price and allow us to deliver a more market ready product.

Also, due to COVID-19 it might be hard to obtain some of the necessary hardware components since so many products are on back orders. To make sure we don't get stuck with this problem, we ordered our project's components months ahead of time. We also need to make sure that we order enough components so that if one of them breaks or fails, we have back ups. This is extremely important because, as previously stated, there are lots of components on backorder that could potentially be devastating to our project's design if we have to wait weeks or months for hardware components that fail during testing.

4.2.4 Size

The casing of the prototype is 3D printed, so the size is limited to the dimensions of the 3D printer used. This poses a couple of issues because we are designing the project before we know the dimensions of the 3D printer. If the project we have designed turns out to be too large for the 3D printer, we found other options for creating the housing. On another hand, if we design something that is large and heavy, it will be a turn off for consumers because the industry has already stated a standard of size for similar products. Since the hardware components we are using are relatively small, we believe we can build a product that is fairly small in size. To make the product a desirable size, we stacked as many components as possible while taking heat into account. The only part of our application that we might want to make large for a better customer experience is the LCD display.

4.2.5 Availability

The availability of each team member has proven to be difficult. Everyone has different responsibilities and time availability. Multiple team members have jobs and full class schedules, so even without designing a full project, their time is limited. This has limited us on the amount of time we are able to meet and design the project. If this was a product we were going to bring to market, we would be spending a lot more time together.

Also, due to COVID-19, it is hard to get everyone together in a safe environment. This constraint causes the most difficulty when designing the hardware since the hardware design deals with the physical aspects of the project's build. The software implementation is not as difficult to build remotely compared to the hardware since there are version control systems like GitHub. Availability is also a concern for presenting the project since our professors need to be safe at the time of our presentation. To make sure all team members can meet up and discuss the project's design, we did most of our meetings using Zoom and Discord.

4.2.6 Environmental

The smart surveillance hub has only one environmental constraint, and it relates to the power usage. The hub should properly manage power in an effective way to not draw more power than needed. If the power is monitored efficiently, it will reduce the environmental impact of the hub as a whole. The hub will not cause any damage to the outside or inside of the home. Most of the time, the Hub will be plugged into the wall in the central location of the home. The casing should allow for the hub to be protected and sealed from any outside tampering, such as bugs or water. This will allow the hub to maintain its integrity and work for an extended period of time.

As one of our stretch goals, we could possibly implement rechargeable batteries as the hub's power source. Using rechargeable batteries instead of disposable batteries would be the most environmentally friendly approach since disposable batteries can be more harmful to the environment. Using renewable batteries uses twenty-three times fewer non-renewable r

4.2.7 Social

The smart surveillance hub is a social product, just like most security systems on the market. If the system does not work as advertised, there will be terrible rep of the product and probably will not do as well. Cameras are the first defense of a person's home and family, if the hub does not alert the user, it could result in harm to the home and lawsuits against the company. Every system that is built needs to be thoroughly tested to ensure that the clientele is able to trust this product with their life because, in some cases they are.

Security poses a constraint as well, if the product is not secure from network intruders, then the system can do more harm than good. Ring recently went through this issue where their network was easily broken into, and their cameras were used by a third party. This resulted in a social blowback and the media pushing back on their product tremendously. The clients need to know that when something goes wrong, they are protected in their most vulnerable state, their home.

4.2.8 Political

The political constraints of this product relate to the network security of the hub. As long as the security of the hub is able to maintain itself on a private network and can protect itself in some way from foreign network intruders, there should be no political constraints that limit our product.

We cannot and are not responsible for any additional security protocols for the network and the smart hub other than the ones that are pre-created. The RTSP stream contains both the camera’s username and password, which is used to log into the camera’s feed and connect to the feed, assuming you are on the same network as the camera. If you are not on the same network, then you cannot connect to the camera or take its feed. With that being said, going back to the pre-created security protocols, routers typically have their own security from strangers entering their networks, and that is the typical security protocol for this project. If a user finds themselves in a situation where the network is compromised, then that is not our responsibility because we do not handle or have any relation to their network besides the ability that the user connects the device to the network, and that is it. The application will not have any means or capability to harm the network or expose them from additional attacks.

4.2.9 Ethical

The ethics of designing a security hub system should be a main priority. The system and its integrity rely on the fact that it is reliable to protect the home. The system needs to be protected from cyber attacks and have measures in place in order to protect the privacy of the home. The only way someone could be hurt through the hub is if the hub was subject to a cyber attack and the intruder was able to view the cameras. Protection of the hub's network is crucial to avoid any issues.

Not only are network cyber attacks a big concern for the hub, but there are also other ethics that we prioritize. The system's initial conception was thought upon for us to create a security solution for people who can't afford high end ones and want to be able to pay for what they get. The system we are creating is a single camera system, but a stretch goal is giving the capability to create a multi-camera system that can secure a wide area of places for a user without being priced out of the market. Since the hub is the only product, then the IP cameras are completely up to the user, allowing them to pick very cheap ones or very expensive ones. This reduces the cost tremendously compared to similar solutions in the market, and that is the overall goal of the project while having the user's safety and considering it as our highest priority

5.0 Full Hardware System Overview and Schematics

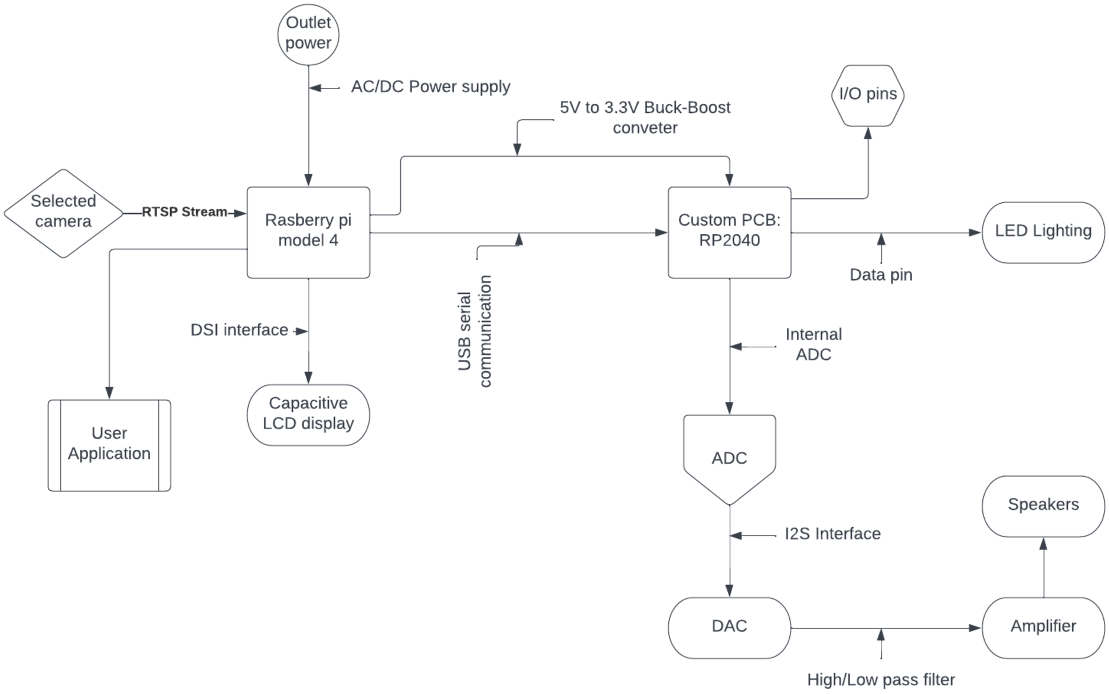


Figure 32: Full Schematic Design

This our overall schematic idea currently we are looking to control our sound functions using the RP2040 MCU to operate and DAC and to then send those signals through a power amp to for expanded sound our overall goal is to also control the lighting function with our MCU as well preferably though an LED strip header.

5.1 Hardware Design

In this section, it contains subsections on the hardware design. This is going to contain things that include information regarding a few subsystems such as the audio system or the power diagram. This is important because the subsystems are microcomponents within the larger scale of the project and making sure we know how they work on an electronic level will make it easier in regards to configure when creating the entire PCB itself.

5.1.1 Overall Schematic

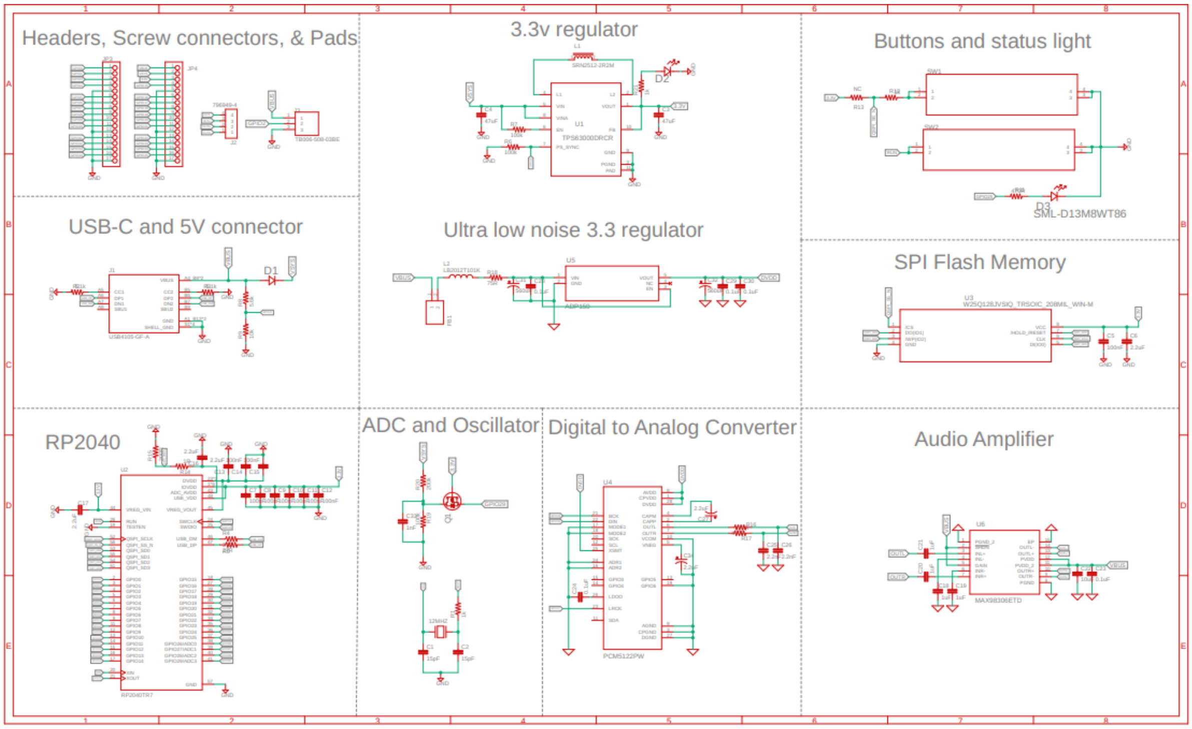
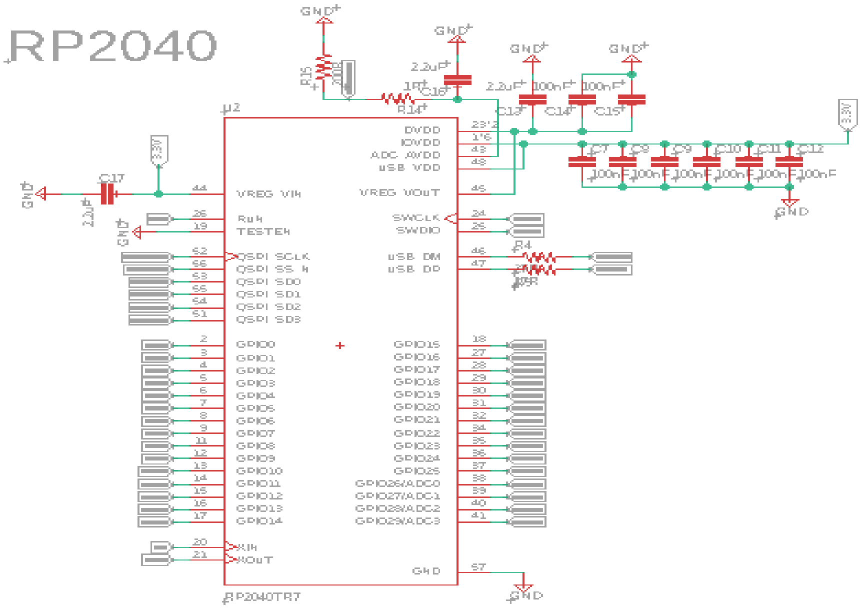


Figure 33: Overall Schematic

This schematic labels every connection in our final design and working project. Each circuit plays an important role in the completion of this project.

5.1.2 Microcontroller Design



Our needed specifications for our microcontroller included the need for expansion GPIO pins and a clock reading high enough to to support all incoming data. The RP240 is equipped with a Dual-core Arm Cortex-M0+ processor, flexible clock running up to 133 MHz wich is more than enough for what we needed. There is also support for popular IDEs, such as Visual studio, Arduino, and Thonny and a Large community support on the Raspberry Pi platform. The RP2040 also has needed low power modes for our systems extended use and. Additionally it has the option for 16MB of added flash RAM and programmable IO state machines for peripheral emulation.

Some alternative included the ATmeag32A and ESP32-S3, with the ESP32-S3 having bluetooth and being overall more powerful the out of all three, and the Atmega32A having huge online support. But the RP2040 was a great middle ground and gave us plenty of power for or funtiosn and system. This comparison can be seen in table 4

5.1.3 Audio Amplifier

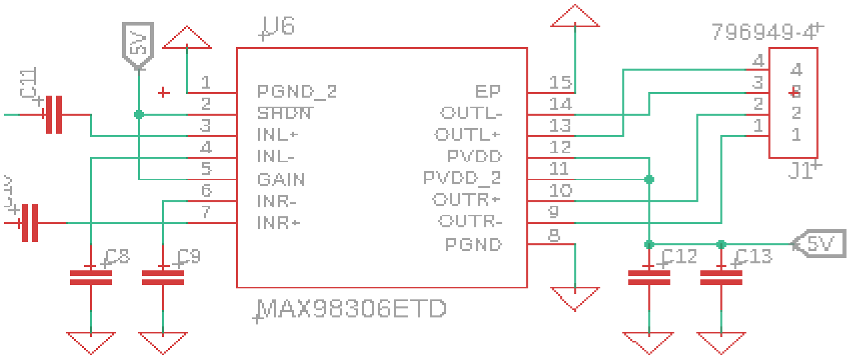


Figure 34: MAX98306 schematic design

This diagram represents the audio amplifier circuit with a MAX98306 chip. The MAX98306 is designed for low voltage consumption and is typically used for small electronic sound systems, such as phones and accessory speakers. The MAX98306 consists of 15 pins consisting of stereo left and right as well as a mute/shutdown pin. This audio amplifier also supports five selectable gain settings (6dB, 9dB, 12dB, 15dB, and 18dB). Low power consumption is ideal for this system since it will be on for a large period of time.

Although the MAX98306 is easy to implement and ideal for our system we we had other alternative for consideration. Therefore we also considered the MAX98303 that has a maximum power input of 3.1wats when running along 3 watt 4 ohm speakers. We settled with the the MAX98306 for its higher power rating and its easier to solder package type.

MAX98306ETD+T	MAX98303EWE+T
Output Power 3.7W at 3Ω with 5V Supply	Output Power 3.1W at 4Ω with 5V Supply
Click-and-Pop Suppression	Click-and-Pop Suppression
Class D Speaker Amplifier	Class D Speaker Amplifier
Gain selection	BGA package type
Unit price - $2.33	Unit price - $2.15

Table 8: Amplifier comparison

5.1.4 Voltage regulators

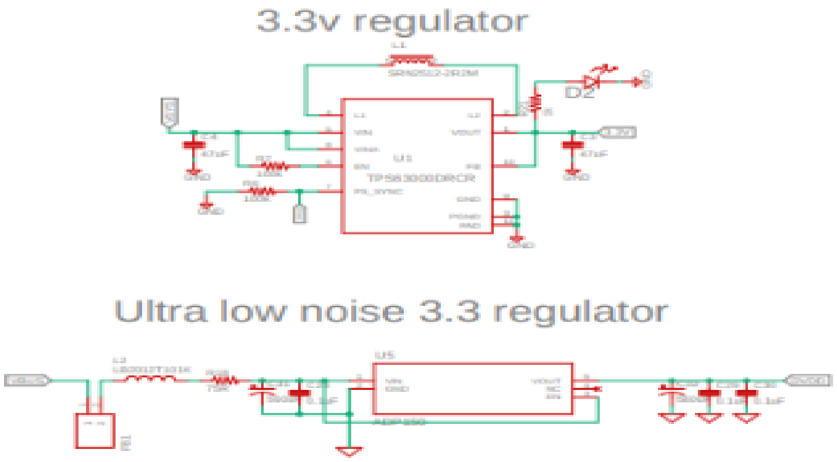


Figure 35: Buck Converter Schematic

Buck converters are required to convert the power source DC output voltage to the DC voltage level required for the various components in our system. A buck converter/ or voltage regulator is a DC-to-DC power converter that lowers down voltage from its input supply to its output while regulating changes in current.We considered three rugalors for our components: the TPS63000, TPS63001, and ADP150. We chose the ADP150 and TPS63001 for our DAC and MCU, as the RP2040 needed a 3.3V 600ma supply for overhead current and the PCM5122 needed a 3.3V 150-200ma supply. the ADP150 is a super low noise converter perfect for keeping unwanted noise and EMI out of our I2S signals, as the TPS63001 offers 3.3V for up to 1200ma wich is more then the needed reguiremnt for our microcontroller unit.

TPS63000	**TPS63001**	**ADP150**
Output currents can go as high as 1200 mA	Output currents can go as high as 1200 mA	Input voltage range: 2.2 V to 5.5 V output 3.3V
Voltage supply range 1.8V-5.5V output max 5.5v	Voltage supply range 1.8V-5.5V output max 3.3v	Maximum output current: 150 mA
Power-Save Mode for Improved Efficiency	Power-Save Mode for Improved Efficiency	Ultralow noise: 9 μV rms
Unit price-$1.73	Unit price-$1.73	Unit price-$1.36

Table 8: Voltage regulator comparison

5.1.5 Digital to Analog Converter

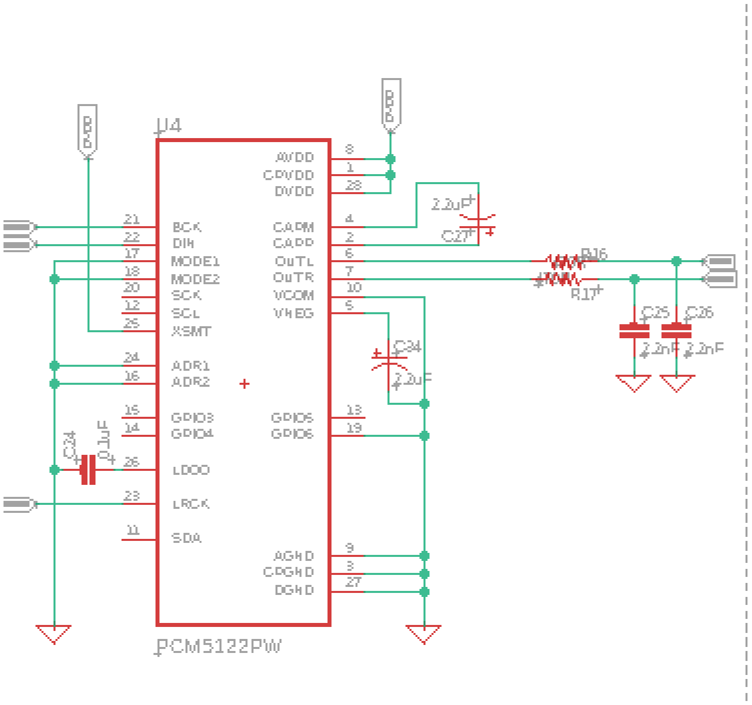


Figure 36: PCM5122 schematic

The PCM5122 DAC by texas instruments operates on the I2S bus mainly and relies on the SPI or I2C bus when controlled for different modes. This dac offers sound clarity and corrects for noise in specified voltage levels. The DAC is a 28 pin header device that comes equipped with PLL when configured in certain modes. Because sinking the clock to the cpu might cause jitter on the I2S bus we decided to down sample our used audio to prevent loss of sound and less distortion. Some other choices included the PCM5102A and the UDA1334ATS Although these alternatives could fit in our system we chose the PCM5122 because of its support for I2C and SPI for future development options, and its high sound sensitivity rating.

PCM5122	PCM5102	UDA1334ATS
Software and hardware configurable for I2C,I2S,SPI	Only hardware configurable	Only hardware configurable
Integrated PLL to drive system clock	Integrated PLL	Integrated PLL

Accepts 16-, 20-, 24-, and 32-Bit Audio Data	Accepts 16-, 20-, 24-, and 32-Bit Audio Data	Accepts 16-, 20-, 24-
dB rating 112	dB rating 112	dB 100
Unit price-$3.10	Unit price-$2.59	Unit price- $2.84

Table 8: Voltage regulator comparison

5.2 Software Design

The software design of this project will include a stack of technologies in congruent with each other. In the field this is called a software stack where there are different technologies communicating on different levels to handle different tasks. The diagram below describes the communication of data and the interactions between every node within the system. This is important because this visualizes the endpoint from the camera to the smart hub and the smart hubs features, which are the screen, the lights and the storage. The node that has the encoded video data is the H264 along with the node that is the decoded video data, as the H264 is responsible for not only encoding the stream being sent from the camera but also responsible for decoding it after it is received on the smart-hub's raspberry pi. This is in effect not only because we need it to be encoded for data restriction issues, but mainly because it is an industry standard and the video encoder comes installed on both unit's hardware. Without being able to decode the video data, the hub would not be able to interface with the IP camera's off the shelf and that would prevent the hub to be as easily interchangeable with standard IP cameras.

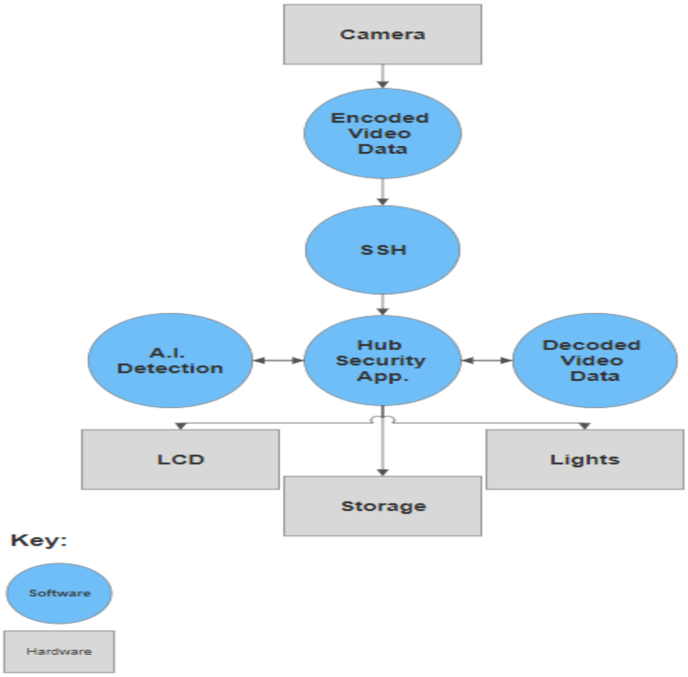


Figure 37 :Software-Hardware Interface Block diagram

5.2.1 Software Stack

As per the software research, this section and following subsections will include information on potential paths and operations that will consist of different options to achieve the same end goals and initial specifications. In the software world, there are plenty of software stacks that operate differently internally but overall can achieve the same goals. This makes software a bit more difficult, because it puts us engineers in a tricky situation, giving us more options but making our choices more complex.

Researching the software before picking the first choice is the best practice because if we jump onto the first stack that seems to be good we might run into future issues such as the stack not having the capability to do things that we later need as features. That is why researching before developing is the most optimal path because developing and having to start over due to lack of research on the capabilities on the stack will ultimately waste time and resources of everyone involved and can create blocks within the development cycle.

The software requirements for the desired model of our project to work is that there will be a desktop application that is independent from any sort of web application in the user's interface. This will hold all the executable code so the user doesn't need to run any code environments or have any pre-existing software requirements, because the executable code should compile and run all within its own application environment. The application will be obtaining the RTSP stream from the user. Ideally we would like to automate this so the camera's stream automatically connects to the application, but every IP camera has a different unique RTSP URL stream that we cannot account for, so the user will be required to get the RTSP URL themselves and simply plug it into the client. The user will input this and send it to the business logic end of the code, which will handle all the streaming generation and use A.I. to evaluate the stream. Lastly, the software needs to be able to send signals to the Raspberry Pi-4's pin headers to output signals to the smart notifier.

After some research, we found three realistic options. These options consist of NodeJS, Tkinter, JavaFX. They have all capabilities to fill the needs of the software requirements previously mentioned.

5.2.1.1 NodeJS

NodeJS is a very powerful server side, cross platform and open source runtime software that has pre-existing and community created libraries and modules that can supply functionality to users giving people the power to create ground up applications.

Traditionally NodeJS is used for web applications or websites, with the traditional Restful API with the client/server model. In this model, this allows users to see things on the front end and interact with things on a website and send requests to the server and can get responses which can be data from databases, such as a system to allow a user to log in or query and parse data from a database.

NodeJS as previously described has plenty of libraries, otherwise known as modules, that are used by the Node Package Manager (NPM) to allow users to use an extremely large variety of open-source software that can get the user capabilities and functionality to do things. We will look a bit in depth on what some of these modules are, what they can do and how they will offer us functionality to achieve our requirements and meet our deliverables.

NodeJS offers a bunch of modules, which act as libraries, that allow simplicity for using different libraries which are ported into these modules. These modules use javascript to utilize their functions, but give accessibility to things that other programming languages exclusively do. Modules are essentially a way to port other libraries from other languages.

One library we needed to communicate with the output pins on GPIO is called *onoff*. This module gives accessibility to all raspberry pi's, regardless of their generations. There are features within this module to essentially communicate with electronics to send on/off signals which will be used to toggle the MCU notifier to turn colors on the mood light or activate an audio cue.

There is a specification for A.I. for object detection, specifically human detection. An A.I. library needs to have the capability to have a model that can detect humans at the minimum, but for future stretch goals it'd be nice to be able to add and edit models to detect other objects in the IP stream too. A couple of options are either OpenCV or Tensorflow, as there are modules in Node that allow us to utilize their features within our NodeJS application.

Electron is a module that will allow us to port the application. Typically with NodeJS applications, the applications are localhosted within a web browser. Since Node acts as the software that can configure the backend server side management, ReactJS typically is the frontend software. Since React is the software stack's standard frontend framework, the default display is within a web browser. Electron makes it so that instead of an application existing only in a web browser, we can have a desktop application that runs your code as an executable file, but acts as a web application. Electron will be the way we wrap our application so it's more desktop oriented and software portable across different platforms.

The server we used to make to communicate with the client is an express back end. It's an open-source framework that will communicate with React using the Rest API, which is HTTP calls. The React client will send the URL needed for the RTSP stream and the server will do all the logic needed to run the stream from the camera. We put the server on a port and the websocket we create with the RTSP stream will be on its own port, which the client will use to display the stream on the front end. The server will also use the node-rtsp-stream module that allows us to stream using any RTSP stream within the same network as the ip camera.

5.2.1.2 PyQt5

One of the other alternative software stacks is using Python. There are a ton of built in libraries and interfaces that will handle what we need to make a commercial desktop application to display the IP camera RTSP stream using A.I. detection and that will interface with the smart notifier. We will go into a bit of detail in how to achieve all the deliverables with python and how each function will use each respective library within python.

PyQt is a built-in python interface within the python library that is used to create desktop applications. It is otherwise known as a GUI toolkit. With PyQt you can specify the size of the window and things that the application window can do. This will allow us to create an executable desktop application and is the alternative to the previously mentioned ElectronJS mentioned within the prior NodeJS section.

Within raspberry pi, the pi can run its micropython library that gives us accessibility to the GPIO pins and will allow us to toggle them for the smart notifier. In the micropython library, there is the machine library, which gives us access to the GPIO pins. This allows us to control and send signals to and from the GPIO header pins making it easy to interface with any PCB, or even a simple breadboard.

As far as the capability to stream an RTSP stream, python's OpenCV can take in an RTSP stream directly and also analyze it with A.I. to detect humans. Previously in NodeJS we needed two separate modules to stream and then to use an A.I. module to analyze it to detect humans. We can kill two birds with one stone using direct OpenCV to stream and detect objects.

5.2.1.3 JavaFX

The last software stack we will consider is JavaFX, an open-source framework built with Java. The framework was built to replace Swing, a library used to build desktop applications. However, it never became a staple of Java SE but was adopted by Oracle and published as part of the OpenJDK. Although JavaFX is used to develop desktop, mobile, and embedded applications, we would use it for building a desktop application. JavaFX can implement various open-source libraries like TensorFlow and Jetty consumed in the application as JAR files.

The application's architecture must have a client-server model to make a valid connection to an RTSP stream to display the IP camera's footage. To establish the connection to the RTSP stream, we would use Jetty, an open-source project that supports protocols such as HTTP and WebSocket. Using WebSocket is what we are interested in using for setting up a WebSocket allowing for the client to communicate with the server. The client will then input the streaming credentials needed to make a valid connection to the RTSP stream, and data will be sent to the server via WebSocket, where the connection will either be accepted or denied. If the stream is rejected, JavaFX will catch it in a try/catch block, alert the user that the connection could not be made, and redirect them to the page where they can input the credentials again.

TensorFlow has an available JAR file for implementing OpenCV that would be responsible for handling the deep learning model used for human detection. The model will be able to read the pixels from the RTSP stream display canvas. The model will be able to predict if a human is nearby in the background without the user seeing what's going on behind the scenes. Not until an actual human is detected will the alert be triggered.

A java library called Pi4J is explicitly built to manipulate the hardware on a Raspberry Pi and control its GPIO pins. Once the library is installed inside the application's directory, the JAR files that contain the logic we would be using are located inside the io.gpio package. The library will enable us to turn the GPIO pins on and off, monitor the state of each pin, and read the output produced by each pin.

JavaFX has relatively the same capability as the previously discussed frameworks and libraries. However, some of the libraries that would be used to connect to the IP camera, manipulate the Raspberry Pi pins, and make the deep learning predictions lack detailed documentation. It also doesn't have as popular of a following as the previously discussed frameworks making it hard to find solutions to commonly encountered bugs. Despite the lack of documentation and popularity, JavaFX is speedy and lightweight.

Underneath we can see the core differences on the table between all the previously mentioned software stack options, Java, NodeJS and Python.

	Java	**PyQt5**	**NodeJS**
Speed	Fastest	Fast	Faster
Performance	High	High	Low
Scalability	High	Medium	Highest
Simplicity	Simple	Very Simple	Medium
Community	Strong	Strong	Strong
Libraries	Good	Good	Excellent
Cross-functionality	High	High	High

Table 9 : Software Stack Differences

5.2.1.4 REST API

In order for successful application communications across the platform, we utilized something called RESTFUL API. In order to understand what it is, we first should understand what it does and to understand why it is useful and a necessity for this project and why we need it.

In the block chain diagram, we have a generic server and client communication flow. The use of the client is front end related utilization where the client handles showing the front end user interface and any reactive events can and will be handled there. For example, when a user clicks a button or types in data, this is all front end related actions and events that have no server or backend utilization up to that point. After the user wants to submit the data to do some sort of event, for our case the user would submit a RTSP url to a server, this would implicate a server based method, but behind the scenes we need to discuss how this actually works.

This is using the REST API, where API stands for application interface. This is the layer between the client and the server that allows the two to connect and communicate with each other. Let’s slightly go back to before the application is even running. For our purposes, as discussed we have the client and the server. In order to run both we need to dedicate ports to both applications.

Ports allow us to traffic data to and from the application to and from other ports. Luckily for us all the ports we need for core functionality will be on the local network, otherwise known as localhost. This makes it easy because we do not need to network outside of the local network to communicate and send data between the client and the server. In a typical network that consists of a client server architecture, we would see the client being local to the user and the server being not local, but hosted elsewhere and the two would communicate across different networks sending messages to the ports these services are being run off of. The ports are dedicated to one process at a time so there is no overlap and confusion on the network and would not allow for multiple servers or clients being run off of one port. For our services for the project, we dedicated two ports for use for the software. Both ports will be available on the network that is free from use from other software being used on the computer, because as discussed we can only use ports that are not already in use. The two ports will be for the client and the server. The client, as previously discussed, is front end services whereas the server will handle the logic side of things.

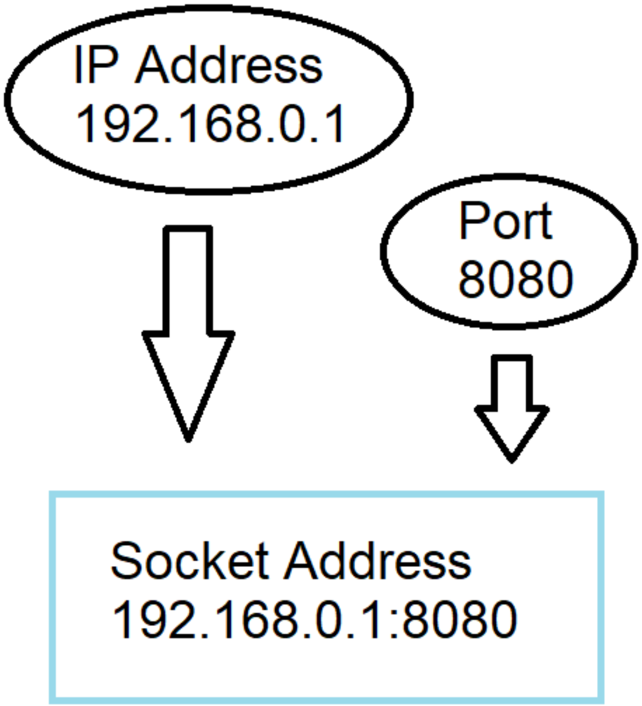


Figure 38: IP Socket Configuration

Since we have two ports being used for two different services within the same network, the localhost will be the network's IP address. The localhost’s IP address by default is 127.0.0.1, whereas the actual network’s overall ip address is 192.168.0.1. Going back to the basics, how it works is that we need to specify the network we are trying to connect to and then identify its port. As you can see in the figure above, there is the IP address and then the port that the service we want to be configured into a single address called the socket address. The client’s address for our purposes will be accessible on a browser within the network of the application on localhost:4200. As discussed earlier, localhost is interchangeable for 127.0.0.1 which is the device’s local ip mask, and 4200 is the port that we are targeting on the network that will host the client. When the client, otherwise known for our use as the electron application, is running, we will see it discreetly as localhost:4200.

The server on the other hand works with the client to handle other tasks that are not specific to frontend visual type of things. The server will handle the HTTP requests between the client and other servers if need be. The server for our purposes is accessible from port 8080 within the network and will be running in the background along with the client.

For the most basic version and core features of our software, we run a frontend that displays the video feed along with an input field for the RTSP stream. The frontend will have an HTTP request that will send data from the client to the server. This will use a post request. The typical options are a put, post, get or delete. Since we are sending data from the front end to the back end that is why we use post, because it posts data from client to server using a message format called json. HTTP requests are extremely common and an industry standard for networking applications and connecting these applications from the client to servers. Due to this being a standard, a lot of services and servers allow for requests to be handled to respond with data. In our case we don’t necessarily need the response but rather need to be able to send data within the request in which the server will use to run a websocket of the stream.

The HTTP requests have two main parts of the message, the headers and the body. The headers specify things like the message type, the kind of request it is, destination, message size and other metadata things that the server can use when receiving the request. Within that message is the slightly more important stuff (for our developers standpoint) which is the request body. This is the json message that can structure data to be sent and parsed after the endpoint. The structure of it is similar to key value pairs where we can parse the key for the value.

Our post request will have only 1 key value pair, which is the RTSP stream and the URL responding to that stream. The RTSP stream is used to run the websocket and post the stream to where the client will read, run and display to the client. So overall we have minimal requests and responses from the server to the client but it is necessary for us to use this as we need to be able to connect the client and server as previously discussed. Without that in place then the server cannot and will not be able to create the stream for the client to use.

In regards to our stretch goals, one of them being a server that can store footage from the camera's and such. This would also be implemented on the server side of things using REST API. Assuming we get to this software being implemented on the design, we packaged and saved the feed once a day and then we can send it off to a cloud storage service using API calls and put all the footage within the body of a HTTP request. This would be the simplest version of a backup storage service.

5.2.1.5 Audio Software

Audio software is and was important when it comes to alerting the user about a threat. Depending on the route of the hardware we choose in regards to what was used to alert the user, the option of the audio software can change. Going back to the hardware, we have a few different options. One being a tone generator; this option does not require audio software but we can still produce a sound. This is due to the fact that the tone is being generated based on the voltage supplied to the hardware. This creates a generic sound that we can only control the decibel levels of and not the type of sound. Another similar method that would be like using a tone generator is using an electronic buzzer, otherwise known as a piezo buzzer. This also generates a different type of sound than a tone generator by supplying a voltage.

Going upon the route of using a DAC, we have a couple of options. A software called Volumio is a mp3 audio playback that is open-source and based and plays audio from linux distributions. Volumio being open source is very important because that gives us as developers more freedom on how to use it. There are HTTP requests we can send the software on the system that we can play audio from any REST API application. A typical request architecture is that we are sending the requests to a server, which is still true, but for this instance the server is a local hosted server that is running and that we must specify within the URI the server being local so we know where the request messages are directed to. Also within this architecture, Volumio will contain all the mp3 files within its library and we are just telling it what to play through the requests. So we are not sending the actual mp3 file to the server but rather we are sending instructions for what the server to play and such. We would supply an audio mp3 file that would serve as a notification sound for when there is a threat and send the request to the server to play the sound.

Lastly another DAC related option is using a python library called AudioCore. This library is going to be also open-source and very simple, where we can open a mp3 file like a wave file and then specify within python to output the audio from the pcb to the DAC. Since this is more of an instruction or functional based coding method, everything will be internal and not require any HTTP requests or external piping to different softwares; meaning python will be a hub for all the instructions and external software would not be needed.

As previously mentioned within this section, there are a few options we are up for choosing. With that being said, the AudioCore is what we be used. The software is already included in the library we are using and it is a direct straightforward method for implementing the audio. The DAC will retrieve the audio signals and output the appropriate sounds to the speakers.

5.2.2 Deep Learning Libraries

One of our application's key responsibilities is to detect when a human is present on the premise of the user's property. Choosing a suitable deep learning library for our project will determine how accurate our project is at making detections. This part of the project's design is critical because if this part of the application fails, our application will behave unintended, and the user will not be satisfied with the product. If the model gives false detections, the user will be frustrated with the product, and even worse, if the model doesn't make a detection when it should, the user becomes vulnerable instead of feeling secure.

Luckily, many open-source libraries are already built that offer the type of model needed for our application to make the proper detections. The open-source libraries are very prevalent and get millions of downloads. The libraries researched were able to provide us with the model we need without modifying any of the training code. But since the code is open-source, if we wanted, we could modify the training code to fine-tune the model to behave in a more specific approach. We are interested in OpenCV, TensorFlow, and Scikit-Image, all of which provide models that were built by a group of intelligent people and have been extensively tested.

5.2.2.1 OpenCV

OpenCV is a library for programming functions that are targeted for real-time computer vision, hence the CV in the name. It was developed by intel with an initial release in 2000 and is a very easy plug-and-play library that can do powerful quick things right out of the box. Originally it supported python but there are ports for other programming languages out there like NodeJS javascript and even C++. Although it is a very powerful library the size of it is only roughly 200MB making it strong for smaller applications that need some sort of A.I. computer vision detection.

Immediately out of the box, OpenCV has pre-trained models for human detection which is great because that is our desired A.I. workload in terms of what model we need. Therefore there is less training of a model since it already comes with a pre-trained function to do what we need. It can also be trained for other things, such as if we desire our system to detect animals or cars we can inject models that will be able to detect such. The training side of OpenCV is not the best in the industry standard but the deployment is ultimately the strength of OpenCV due to its portability and a lot of applications just want human detection which, as previously mentioned, this already has within its core functionality. We looked into better libraries and see what their strengths and weaknesses are compared to OpenCV.

5.2.2.2 TensorFlow

Tensorflow is a very powerful library more so for research and advanced A.I. implementations. The main purpose of Tensorflow and the goal on its initial release is to build machine learning models. Opposed to OpenCV which was initially used for computer vision tasks and later deep learning implementation, Tensorflow was initially developed with deep learning in mind.

With this in mind, therefore tensorflow is better for injecting custom datasets for specific tasks. If we decide to detect more than just human detection, then OpenCV does not have that capability because it only has human model datasets within its framework but not other objects. Tensorflow would reign superior in this case because it’s easy to train when giving it new datasets. Another thing to consider between tensorflow and OpenCV is the performance. Some models can be an order of magnitude faster in OpenCV compared to tensorflow, no matter the language that the library supports. Lastly there is another library for image and computer vision tasks, scikit-image. We will discuss this last option in the next section.

5.2.2.3 Scikit-Image

Initially we talked about OpenCV which is widely used across commercial applications because of its simplicity to deploy and performance measures being very fast compared to some of its counterparts along with tensorflow being used for more customized models to be trained, now we can look into scikit which is a more middle-ground between tensorflow and OpenCV.

OpenCV is python’s easiest image processor and scikit is its main competitor. Both libraries are seen as image processing and management tools. Scikit offers I/O, filtering, morphology, transformations, measurements, annotations, color conversions, test data sets and more. Like OpenCV it was initially written in python with a lot of documentation for the library. It is a widely used open source with over 30,000 lines of code and hundreds of contributors adding to the library. Scikit and OpenCV are very similar whereas tensorflow is more of a different story. For our purpose though, scikit doesn’t have the out of the box features we would want compared to OpenCV although it is a very good alternative if we wanted to train our data models to detect humans through neural networks.

Below we can look at some of the core differences between all the previously mentioned deep learning libraries and get a better understanding on which one might fit our needs. Also it shows us a comparison on certain aspects on which functions are better than other libraries which is ultimately the best knowledge for when choosing a library.

	OpenCV	**Tensorflow**	**Scikit-image**
Level of API	Lowest-level API	High & Low level APIs	Lower-level API

Speed	Medium	High	High
Architecture	Simple & Concise	Not easy to use	Simple & Concise
Debugging	No need to debug	Difficult to debug	Good debugging capabilities
Dataset Compatibility	Slow & Small	Fast & Large	Fast & Large
Popularity Rank	1	2	3
Created By	Intel	Google	Open-source
Ease of Use	User-friendly	Incomprehensive API	Moderate

Table 10: Deep Learning Libraries

5.2.3 Desktop Application

The overall goal of the project is to create a desktop application for the users that is user friendly. Our definition of user friendly is not only easy to use but capable to use to all users, regardless of any health conditions that might prevent users from using it. This includes meeting ADA standards as previously mentioned in section 4.1.2.3. We can now go into more detail on the frontend user interface and how it works congruently with the backend to make the entire application.

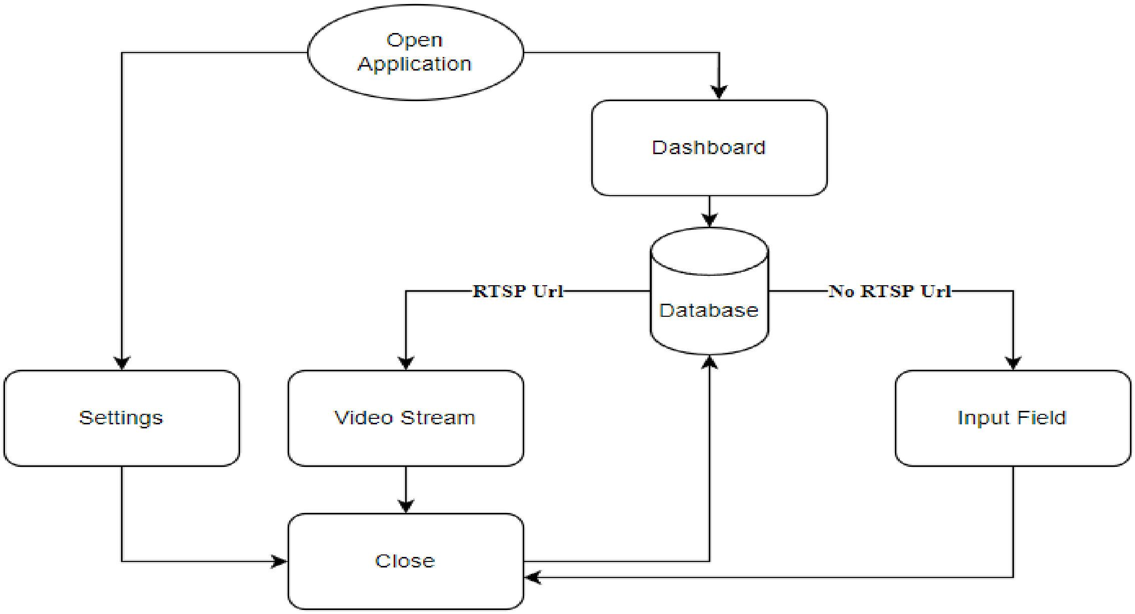


Figure 39: Frontend-Backend relation

5.2.3.1 Frontend

Our desktop application was very basic with all the features required to control the application but to ultimately use the application on a basic level. Upon initial load of the application for first time users we directed them to the dashboard. This dashboard will contain an input field for tasking the stream url. We need the user to input the url in order to view the video feed and also run the human detection on it. Without this then the application is useless along with the smart notifier.

They will also have a settings area to edit and change the url string. When revisiting users on their same device will have the same url that was previously used prior to closing the application. We save this url locally on the device so whenever it is edited, changed or deleted then they will have to make sure it works but this also makes it so the user doesn’t have to always re-enter their url into the field upon initial launch of the application. When the user has a url in, the input field for the url will be hidden and instead will be the video feed of the camera.

Overall for the frontend, there are the two main components, the dashboard and the settings. Both have minor functionality but serve a large purpose because without them the application simply would not run nor will the human detection.

5.2.3.2 Backend

The backend is going to be very simple due to what we need it to do and how it was implemented. The backend will save the url of the RTSP that is being used so the user doesn't have to always add the url string every time the user enters. We won’t be using a traditional backend server since that is overload, we can instead store the JSON message locally as our backend which is similar to MySQL lite.

In order to connect the frontend to the backend there was middleware. This was the server that interfaces the frontend and backend. It will do things like run a websocket stream for the frontend, handle the API for the GPIO pins, handle the deep learning to detect humans and handle the audio communication for the DAC to receive analog signals to convert to audio.

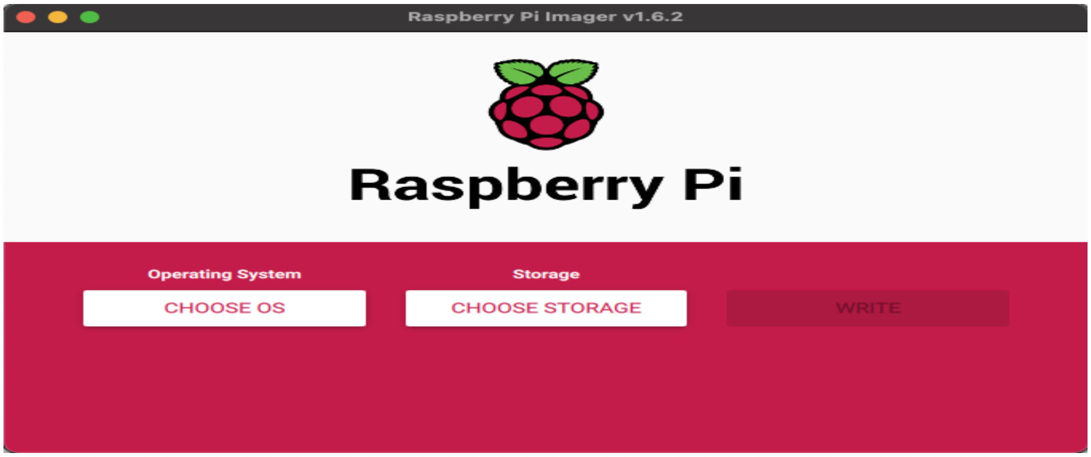
The middleware was a simple ExpressJS server through NodeJS that will listen to a port different from the frontend server. It will receive HTTP requests through Rest API in order to communicate with the frontend. Everything that isn’t being displayed is handled on the server side.

5.2.4 Operating System Research

Choosing the correct operating system is a critical design choice for designing a seamless application since the operating system is responsible for managing the computer hardware and software resources necessary for running applications. Many different operating systems are compatible with running on the Raspberry Pi 4 Model B. The Raspberry Pi we are using is a 64-bit machine capable of running 32-bit and 64-bit architectures. The operating systems that we analyzed were Linux distributions, and they all have significant tradeoffs for different aspects of applications. We conducted research on Raspberry Pi OS, Linux Ubuntu, and Kali Linux. Raspberry Pi OS offers a 32-bit and a 64-bit architecture, so we researched both architectures before determining which one is tailored best for our application.

The Raspberry does not come with an operating system pre-installed. However, they make it very simple to install an operating system using a microSD card. There are many options for installing the operating system onto the microSD card. I have found that the simplest method is to use the Raspberry Pi Imager implementation provided by the Raspberry Pi community. The Raspberry Pi Imager must be installed using a separate computer. For this step, we used macOS.

Since we used a Mac to install the operating system onto the microSD card, we needed a thunderbolt cable with a female port capable of connecting to a microSD card. Once the thunderbolt cable and the microSD are connected, the Raspberry Pi Imager has many different options for operating systems to download by clicking “CHOOSE OS.” Once the desired operating system is selected, “CHOOSE STORAGE” is the following icon to be configured. Once clicking on the button, it will show the microSD card that is inserted into the Mac. Once the microSD is selected, the button “WRITE” must be clicked, and the operating system will begin the installation. Once the operating system is installed, the microSD can be ejected, and the operating system is stored on the microSD card.



Once the operating system is written onto the microSD card, the microSD card needs to be inserted into the Raspberry Pi. Once the card is inserted into the Raspberry Pi and the Raspberry Pi is powered on, the operating system will begin installing on the device. A few steps are involved in configuring the operating system, such as setting up the username, password, timezone, and wifi configuration. Once the installation is complete, the device is ready to be programmed and configured in finer detail.

5.3.4.1 Raspberry Pi OS 32-bit

Raspberry Pi OS offers both a 32-bit and a 64-bit architecture. The operating system was formerly known as Raspbian OS when it was only provided as a 32-bit architecture. However, once they provided a 64-bit architecture, they changed the operating system's name to Raspberry Pi OS to accommodate both architectures. Both of the architectures are based on Debian Linux-based distributions, and Debian is a very popular Linux distribution that has been around since 1993. This section's research will focus on the 32-bit architecture.

The 32-bit architecture was established in 2013 and has flourished ever since. Due to the fact that the operating system has been around for nearly ten years, the operating system has worked out many of the bugs that generally occur in newly developed operating systems. It provides an ecosystem that offers tons of support since it's the recommended operating system for the Raspberry Pi, and the Raspberry Pi has become so popular in the IoT community. There are countless forums where users discuss bugs and solutions to problems that may occur when interacting with the operating system.

Raspberry Pi OS's 32-bit architecture is the safest architecture for running applications. However, it comes at the expense of not using the CPU to its maximum potential when running on a 64-bit processor. Since the 32-bit operating system is more predominantly worked on than the 64-bit version, the Raspberry Pi team recommends the 32-bit architecture for the smoothest workflow. Also 32-bit pointers are not capable of referencing addresses in memory systems larger than 4GB and since the Raspberry Pi model we are using for our project has 8GB of memory we find this to be wasteful for maximizing our projects maximum performance.

5.2.4.2 Raspberry Pi OS 64-bit

As mentioned in the previous section, Raspberry Pi OS offers a 32-bit architecture and a 64-bit architecture. This section of the research will include details concerning the 64-bit architecture and some of its trade offs compared to the 32-bit architecture. It has been a little over one year since the 64-bit version was released, and it has been released from beta and gone into production February 2022. The Raspberry Pi team decided to release a 64-bit version ever since they started using ARMv8-A architectures that are built based on the AArch64 architecture that uses the A64 instruction set. The implementation of the ARMv8-A architecture began with the Raspberry Pi 3, which was released in 2016. The primary reason Raspberry Pi has decided to make the 32-bit version the default operating system is to guarantee compatibility between existing devices and to avoid inexperienced users from running into compatibility issues.

The most significant benefit of using the 64-bit architecture is that it will maximize the CPU's performance since the operating system is compatible with the CPU's 64-bit architecture. As previously discussed, the single-board computer we used is 64-bit in architecture and has 8GB of memory. Since our application depends on maximizing performance, we chose to use an architecture with the largest memory possible to maximize caching. If the model we are using has 8GB of memory and we used a 32-bit architecture, we would not be using the system to its potential.

Raspberry Pi OS 64-bit architecture has not been on the market for long, and the product has only been out of beta for a few months now. The main concern with using the 64-bit operating system is that it may contain bugs that don't offer much online troubleshooting support. As discussed in the previous section, the 32-bit version is the default operating system and has been on the market for nearly ten years. Hence the amount of troubleshooting information online is much denser than in the 64-bit version.

Although Raspberry Pi OS 64-bit architecture hasn't been on the market for very long, it seems to work well for most users who understand the architecture. By implementing the 64-bit version, the CPU's performance can be maximized to its full potential, but possibly encountering a few bugs that the Raspberry Pi community hasn't fully diagnosed. After conducting research, we believe that the positive tradeoffs for adopting the 64-bit version of the operating system seem to outweigh the negative tradeoffs associated with it.

5.2.4.3 Linux Ubuntu

Ubuntu is another open-source Linux distribution based on Debian that is used for a variety of applications. Ubuntu is the most popular of all the operating systems we researched that we included in this section. Since Ubuntu is so heavily adopted, there is a broad spectrum of use cases. Ubuntu works exceptionally well for cloud computing, IoT, server configuration, and many other tasks. There is a considerable benefit when millions of users use a single piece of software like Ubuntu because of the high-quality support it provides when debugging an application.

Ubuntu has to offer three different editions: Desktop, Server, and Core. The operating system has a very structured way of releasing changes to the version. Its ecosystem is highly maintained, releasing a new version of Ubuntu every six months. To make the product more stable in the long term, we would adopt one of its long-term support releases. A long-term support version of the operating system is released every two years, equal to four releases. The most recent version of their long-term support version is 20.04. However, during the build phase of this project, there is a long-term support version 22.04 that is being released on April 21, 2022. If we decided to choose Ubuntu as our operating system, we would also have to choose which long-term support version to install.

Ubuntu has been around for a long time, going back to 2004. It offers excellent support, and it performs well at many things. Performing well at many tasks is good for the operating system's ecosystem, but this could be a problem for our project because we aren't necessarily looking for a highly complex architecture that's good at many tasks. Instead, we need an architecture designed specifically for the job we aim to achieve. That's not to say it doesn't perform well at IoT-specific tasks like ours. But instead, we feel it could be wasteful since there would be so many unused features.

5.2.4.4 Kali Linux

Kali is a Linux distribution based on Debian mainly used for digital forensics and penetrating tests. This type of operating system could be a good design choice for our application because this operating system performs well at interfacing with other devices. It’s also an appealing operating system because it provides the security necessary to lock down the system. Although our project won’t initially have all of its application’s security measures thoroughly designed, in the future, having the capability to have fine-grain control over the application’s security is a significant bonus. It’s not that the other operating systems we research lack security, and it’s actually that Kali Linux is specifically designed for that, so they’re going to do it the best.

The Kali operating system has been around for nine years and has gained a large following along the way. It has gained such popularity because, if appropriately implemented, it is known to perform exceptionally well at securing applications. It offers over 600 penetration testing programs used to find holes in the application’s security design.

Since the operating system is specifically designed for security, we feel that our application might not perform as well compared to the other operating systems regarding the whole state of the application. Yes, security is an essential aspect of our project’s design, but we believe it may not perform as well as we need it in other parts of our application.

5.2.4.5 Operating System Selection

Operating System	Raspberry Pi OS 32-bit	Raspberry Pi OS 64-bit	Linux Ubuntu	Kali Linux
Architecture Width	32-bit	64-bit	64-bit	32-bit and 64-bit

Initial Release Date	September 10, 2013	May 28, 2020	October 20, 2004	March 13, 2013
Latest Release	January 28, 2022	April 4, 2022	October 14, 2021	February 22, 2022
Distribution Derived from	Debian	Debian	Debian	Debian
Specialty	Applications specific to the Raspberry Pi running on a 32-bit architecture	Applications specific to the Raspberry Pi running on a 64-bit architecture	Cloud computing, personal computing, servers, IoT, and supercomputers	Digital forensics and penetration testing

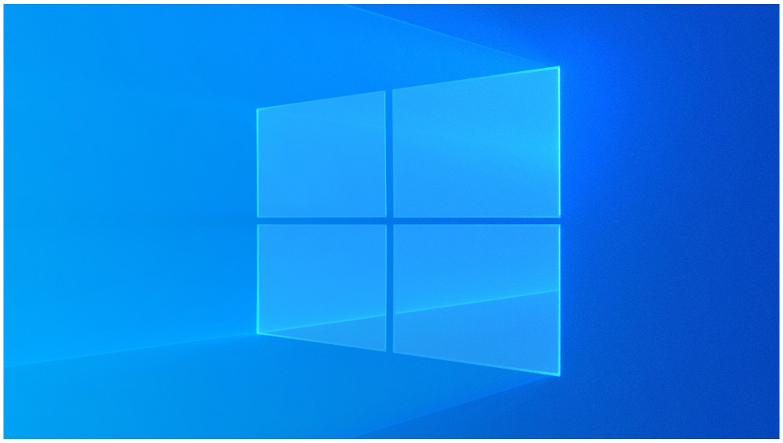
Table 11: Operating System comparison and selection

Making a design choice based on all of the operating systems we conducted research on is challenging because they all behave so similarly. Debian-based Linux distributions are known for performing well at configuring multiple resources to communicate together. All of the operating systems researched in this section are Linux-based distributions based on Debian. Since the operating systems share the fundamental design of the overall implementation, considering the tradeoffs for each different operating system can be tricky.

Ubuntu seems like a good option because it has such a popular ecosystem with lots of user support. Kali Linux would be a good choice since it has advanced security measures for locking down the application's state. Also, Kali Linux could be used in the future to add advanced security to the application's design. Although Ubuntu and Kali Linux have the potential of implementing our project's design, we ultimately chose one of the Raspberry Pi OS implementations since it was built specifically for the single-board computer we are using for the design.

Both of the Raspberry pi operating systems are also good options because they are designed specifically for the Raspberry Pi's hardware that we are using. But since we're using the Raspberry Pi model with the most extensive memory, containing a CPU with a 64-bit architecture, and the 64-bit architecture is designed to maximize the performance of our Raspberry Pi's CPU, we used this architecture for our design. One of the main reasons we were persuaded to choose the 64-bit architecture is because our Raspberry pi has a large memory consisting of 8GB. Since the 32-bit architecture can't fully use such a large memory module, the 64-bit architecture seemed to be the smartest choice for the application's performance.

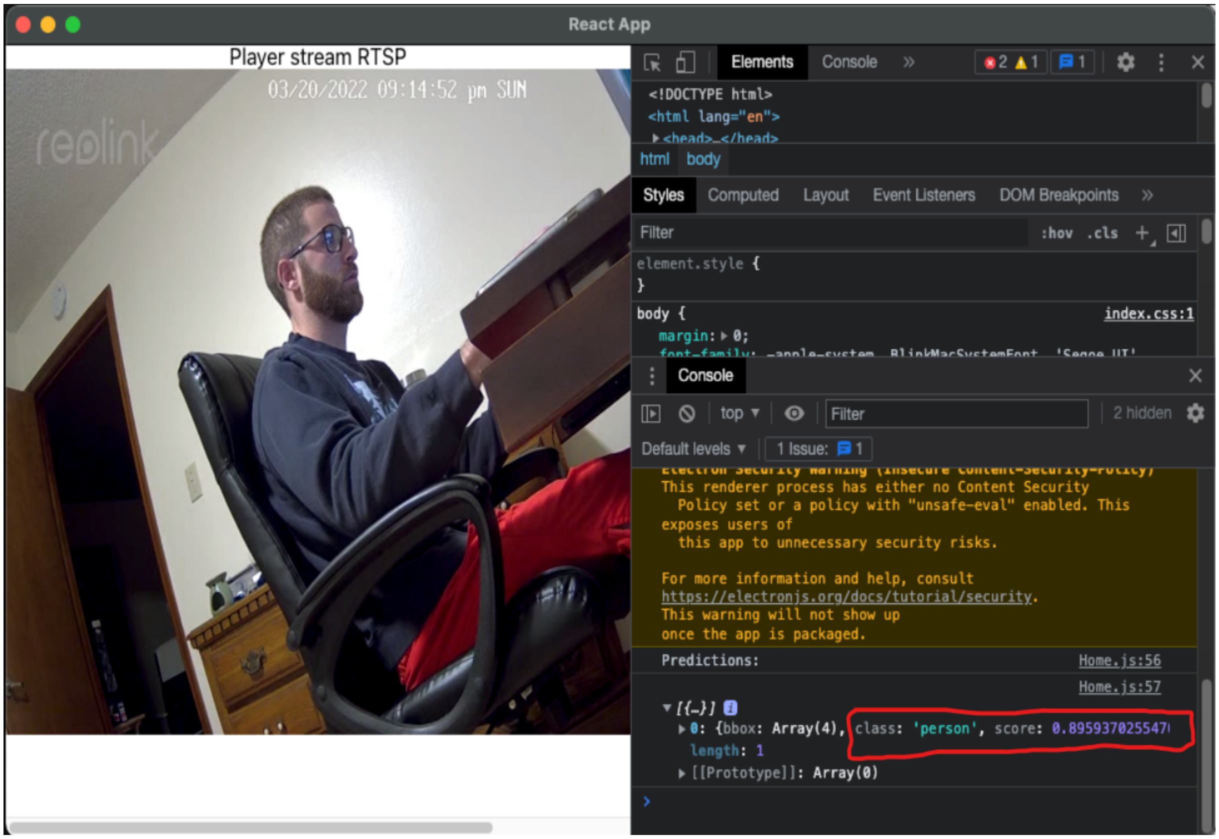
5.2.4.6 Windows



The chosen operating system for the pi is chosen because raspbian is the best option out of the box for the raspberry pi. It is an industry standard linux based operating system that can handle anything you throw at it from either a Unix or windows standpoint. With that in mind there are two routes we could take. We can download the appropriate software environments and compilers and such directly onto the raspberry pi itself and create the software we need to get the job done or we can develop and code the software from a windows device and port it onto the raspberry pi. The raspberry pi itself is a strong small computer but it still isn’t as good as an actual computer. Creating the software on a computer makes the development of it easier and quicker and we can just put the final product on the raspberry pi which will result in the same endpoint.

5.2.5 Initial Software Testing

As far as the testing goes for the software, on our first prototype in terms of software configuration with the stream and the machine learning, we can detect humans with a confidence score. This score allows us to filter out potential scores that are low that might not be humans that are false positives and we can set it to be more true positive. As seen on the right in the image, you can see the JSON message being sent to the console with the information that is being sent upon a conditional, if tensorflow detects a human it sends the message to the client. Within the code instead of sending a message, we have signals being sent to GPIO so this stepping stone is the first of many to come in terms of appending more features to this.



```
<!DOCTYPE html>
<html lang="en">
 ▸<head>…</head>

element.style {
}
body {                                                      index.css:1
   margin: ▸ 0;
   font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI',

Electron Security Warning (Insecure Content-Security-Policy)
This renderer process has either no Content Security
  Policy set or a policy with "unsafe-eval" enabled. This
exposes users of
  this app to unnecessary security risks.

For more information and help, consult
https://electronjs.org/docs/tutorial/security.
This warning will not show up
once the app is packaged.
Predictions:                                            Home.js:56
                                                        Home.js:57
 ▾[{…}] i
   ▸0: {bbox: Array(4), class: 'person', score: 0.895937025547
    length: 1
   ▸[[Prototype]]: Array(0)
>
```

Figure 40: Software Ai Test

5.3 Summary of Design

Ultimately for the design and the product, there were the two main components. These consist of the Raspberry Pi 4 and the smart notifier. In tangent these will create the smart-hub which was housed within the 3d printed case. The raspberry pi will also have LEDs connected to the GPIO which will serve as the mood light and all of the firmware and software application was run and held on the raspberry pi.

The software will consist of NodeJS which contains modules for ElectronJS and ExpressJS. Both of these modules are the server and application interfaces. The frontend used ReactJS as the frontend which is a component dependent module that consists a tribrid of javascript, html and css. All of these dependencies are the makeup of the software end of things.

The hardware was simple. It will consist of the raspberry pi as the main hub. This would be in congruence with the touch LCD screen and an optional port for a monitor. The monitor doesn't have anything to do with the product but the screen was attached to the 3d casing. Also the LED lights were visible from the outside of the 3d casing which was connected directly from the GPIO header pins. Also the DAC was connected to the GPIO header pines to send analog signals for digital audio conversion. The goal of the hardware is to create a moderately sized object, roughly the size of an regular ipad, that will be simple and easy to use. Alternatively, the project without the use of a DAC and another MCU would be the secondary option. The MCU can operate multiple functions, like having a tone generator instead of a DAC in charge of creating audio. The MCU could potentially handle other things too like the lighting and such of the hardware instead of the raspberry pi. These decisions were more ironed out as the project approached the end of senior design 2 and we had more things considered and explored all of our options that are aligned with what we must provide and engineer that met the ABET standards.

6.0 Project Prototype Construction and Coding

In this project, our goal as a group is developing a prototype as fast as possible so we have time to work out the kinks in the first version of it. Based on the build plan and previous section describing the workflow of the project, the prototype can be one of either two things, which we will discuss in a moment. First of all, the prototype needs to achieve a few things. These things consist of the pi having a desktop application that can stream and use object detection A.I. and this interfaces with some sort of PCB. Ideally, we would like the PCB to be the one we research, designed and created but depending on the workflow of the software side, we made our prototype using an off-the-shelf MCU that has most similar to our PCB, so we can get the firmware working from the hub to the PCB.

Another thing is the housing for the product. The end goal is to create a housing for the hub that makes sense for the user and holds all the components so the final product is clean. For purpose and level of necessity, we did not develop this 3d case until the very end; this is for a few reasons. For the first reason, the case is not something that will have any impact on the functionality, it is only meant to be done and used to have a better final product rather than some components grouped together within a random box. Another reason is that it is potentially not a valuable use of time, because the final product will probably change a few times on the hardware level of development and if we have a pre-constructed casing then we ultimately spent more time constructing another case to fit the new(er) versions of our product.

So with all of this being said and done, ultimately the prototype will consist of the core functionality to at least interact with the hub and a DAC. This was the biggest challenge and getting that working was a tall task to overcome, because there are a lot of requirements leading up to the prototype. On the second phase of the prototype assuming we have the DAC working, we got the lighting and other components on board with the software, but these are lower severity; although they are deliverables, they are just more simple to implement and will most likely have less things to overcome to make this work. If we want the lighting to be implemented in the first prototype, we can always just use a breadboard to test lighting capabilities from the GPIO but this is the easiest thing to do with an MCU so that’s why our time is initially going into the more difficult stuff.

The project has two of us CPE students working on the software side of things and two of the EE students working on the hardware side of things; therefore inherently there are two separate agile sprints going with different goals in mind. When developing the project, it’s goals, the standards and requirements, this was a key factor in order to have independent sprints that can proceed on their respective workload that will not necessarily be given a block due to waiting on the other side of development.

When it comes to the build plan, the software is being developed at a steady pace, getting things working for the application. This is because the software development in the early stages of just the application can be engineered without any software being interfaced to it. The first thing we do is research what software stack can be used to achieve our desired outcome along with how realistic it is that we can achieve it with the current software. Realistically, almost any software stack can achieve anything in this day and age, just because of how far we have come with software, but our goal is to create the smart-hub app and to not develop an entire software stack from the ground up; which is why extensive research is important. If we decide on a stack and find out later that it can't handle what we require of it, then that can run into major issues and potentially will require us to scrap things and start fresh or we might not make our deadlines.

As per software development, the following goal after we find the stack we are confident about and desire to use, we then can start making a simple desktop application. This application's requirement is to just simply be able to run on a linux system, since our hub will be utilizing Raspbian, which is derived from linux, then the app must be able to run on that. When we have a base app that can open and run on the smart-hub, then we can start adding the baseline features to it.

The core baseline feature is being able to stream an IP camera within the application. This will be achieved by using the camera's RTSP url, which is different for everyone. An RTSP url string is camera and network specific, so it's not a static RTSP string. The RTSP consists of multiple things that is passed through one string, starting with the camera's username which can be different per IP camera, the password to access that camera, the IP address of the network the camera is connected to, the port that the camera is accessing on that network and lastly there is typically a generic string of some sort that is going to be different per camera model. Below is a generic example of an RTSP url:

rtsp://{username}:{password}@{ip address}:{port}/{camera model's string}

There are multiple ways for the user to find out their RTSP string, but this is required upon the user because we cannot predict or dynamically know their string, this information they would know since it has sensitive information like the camera's password and such. In retrospect, if the camera had no password then anyone on the network can access that camera via the respective port and hack into it, therefore this protocol is best for the user's safety and for simplicity on our end.

Since the application requires the user to know and put in their own RTSP url, there will be some sort of form input within the application when the camera is not being streamed or displayed on the app. This is the next step upon the application development, where we generated a way to take in the user's input within the application for their RTSP url.

Upon receiving the RTSP from the user, we can work on the streaming process. Being able to stream the video is a must, because without it then the product is utterly useless. Also without being able to stream it, then we can't operate on the stream with A.I. to utilize the PCB notifier device.

Down the pipeline of software once we build and develop the streaming capability, we can now finally use some object detection to check if humans are detected. After all, this is a security system, so detecting humans where they shouldn't be or simply detecting humans to let us know someone is approaching, is ultimately the goal of this device. Up to this point, an application and stream is working and so the A.I. software will piggyback off the streamed data to then analyze it using A.I. modules. This development will happen at this stage and so there was conditional code, whereas there is a very positive match for a human detection, then we can send that data to the next phase of the software pipeline for the hub to act appropriately.

At this point, we have all the software on the hub working for the application side of things and now we can experience a potential block, where now the application requires us to interface it with the hardware that was developed.

Up to this point, we have a working application, a working stream and working A.I. to detect humans, the application is doing everything required of it on the hub's side of things. Now the firmware development would take place, where it will detect a human, show it on the hub's LCD or monitor attached to it and lastly send signals through the GPIO for the smart-notifier PCB board to do its thing. Within the application on the server side code, we made GPIO pin header calls to send signals to the hardware to act appropriately depending on the use case.

As previously mentioned, if there is no hardware to interface with due to the PCB workflow creation, we cannot test the software on the desired PCB, but this doesn't ultimately mean we can't stop engineering. We can get a similar PCB and test code on that, so when we get to the point of this stage and if we are waiting on the PCB development, it's not a big worry because we can test on similar products and so this will not halt but rather give a similar product that will configure easily to the final PCB board.

In regards to the other side of the development and engineering, there was aseparte research and creation process of the PCB. The first step is researching and making sure our options will fit our scope of what requires us to achieve our deliverables. The hardware needs to ultimately be able to communicate with the hub's Raspberry Pi 4's GPIO pins, or alternative pins for alternative methods. The PCB is going to be a DAC which will convert audio from the Pi to send the audio cue for the user to hear and know if there is a threat or someone detected.

This DAC will be discussed in greater detail in its respective section within the paper. Also there will be another component that the GPIO pins will communicate with, that will handle the LED mood lighting displayed on the housing component of the hub.

6.1 Integrated Schematics

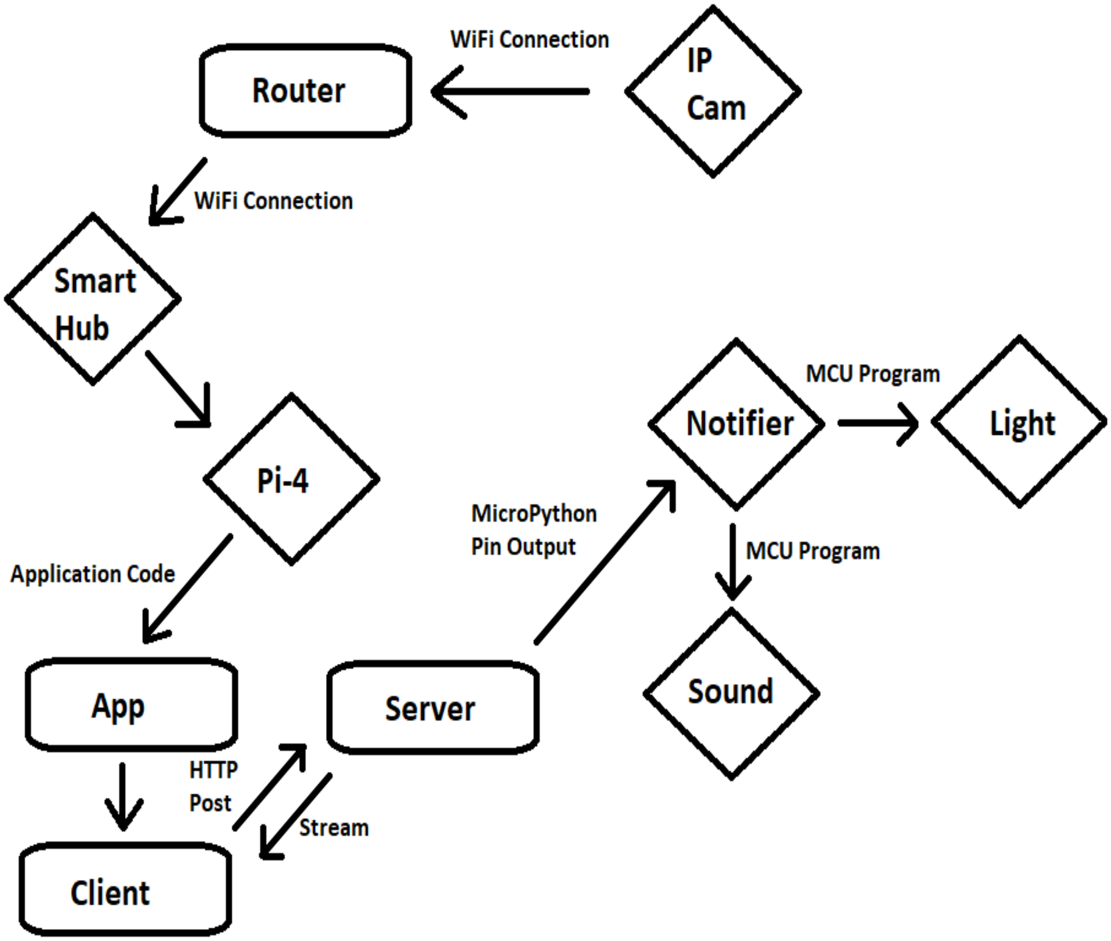


Figure 41:Hardware-Software Integration

This diagram explicitly details the relation and usage of the software used to communicate between all the hardware and software. The arrows direct the flow of data using their respective network or communication protocols. There is only one arrow missing and that is the relation between the Smart Hub and Pi-4, this is because the Pi-4 is a subsection of the Smart Hub and the block for the Smart Hub is only there for clarity and not an actual part, but the main hub of multiple components.

The router will be the main traffic controller that communicates the Hub with the IP camera as discussed previously. The Pi-4's application will hold not only the client but also the server. They run in tangent and have very unique tasks. The client is what the user will be using to interact with whereas the server will run the stream, using A.I. detection and communicating with the notifier MCU to do certain tasks, which are conditional on what the A.I. will find out. Also, the client and server will be communicating with HTTP Post requests to send the RTSP stream from the user to create the websocket stream on the server, that will be displayed on the application's web browser using Electron.js.

6.2 PCB Vendor

The printed circuit board or PCB is the board on which all of the wires and electrical connection lie. There are many PCB vendors that we can use and produce boards in a timely manner. Most vendors require Gerber and bill of material files in order to print the boards. Some of the most popular Vendors we have been looking at for printing our PCB JLC PCB,OSHPARK, and PCB Way. Each of these companies have the capability to do exactly what we need for the scope of this project. Once testing and design is complete, all that is needed is to send these vendors our Gerver and BOM files and they return quotes within 24 hours, we then buy them and within two weeks we received our PCB. We plan on ordering around 3 or 4 PCBs (depending on cost) so we are able to protect ourselves in case something occurs that damages the board.

6.2.1 Circuit Board Types

Printed circuit boards (PCB) have two functions, the first is to hold electrical components stationary and the second is to provide reliable electrical connections between the electrical components. The wires are run along the surface of the conductive layer, the layers are connected with holes called vias. There are a variety of different PCB styles that can be used depending on the task or requirements of the project.

- Single Layer PCBs
- Double Layer PCBs
- Multi-Layer PCBs
- Flexible PCBs
- Rigid-Flex PCBs
- High Frequency (HF) PCBs
- Aluminum Backed PCBs
- Heavy Copper PCBs

Each of these PCB designs are used for different types of projects. Single layer PCBs are mainly used for RF microwave applications such as consumer electronics (computers, radios, appliances, and cell phones.) Double layer PCBs are found in consumer electronics as well, and often used in power monitoring, amplifiers, and test equipment. Multilayer PCBs use a rigid-flex technology and can be folded into any shape. These are mainly used in high density interconnect PCBs which are used in industries such as Medical, military and aerospace. Flexible PCBs are made to be shaped and designed to bend and fit into specific projects. It utilizes a flexible material, usually thin, and contains multiple copper conductive layers. Rigid-Flex PCBs contain both hard and flexible materials and support up to 50 layers. High frequency PCBs are used in high speed communications and RF designs. Aluminum Backed PCBs are used in RF communications. Finally Heavy Copper is used for large current designs, any design that requires a significant amount of current traveling through its wire should utilize this design.

For this project our design is fairly simple and should support a one or two layer design. Keeping the layers simple will help our PCB printing process excel faster and reduce the cost significantly, the more layers that need to be supported the more expensive each PCB becomes. Since we ordered PCBs we need to reduce the cost of the project in any area that we can without compromising the performance of the project.

6.2.1.1 Surface Mount

Surface mount technology is the process of mounting any electrical engineering components onto the surface of a printed circuit board. This makes it easier to build and prevents the circuit from possibly overheating. By using surface mounts to place components onto the circuit board, it allows for more components to be fitted to the board. Surface mount technology makes production a more straightforward task. If our project becomes very popular and put into production, using surface mount technology will allow for a smoother automated build.

6.2.2 Assembly

For our project to go into production, we need to think about making the assembly process as simple as possible. Since we are using a relatively small amount of components and our printed circuit board is relatively simple, the assembly should be straightforward. The part of the assembly process that will take the longest and cost the most money is getting all of the printed circuit boards developed. Getting printed circuit boards built is simple, but it can take time, and the cost must be considered. Since it can take a while to manufacture the circuit boards, it is good practice to know how many boards need to be built in advance. So for assembling the product, the main concern is finding a manufacturing PCB manufacturing company that can produce the chips in a short amount of time at a low cost.

6.3 Final Coding Plan

As for the final coding plan, the application's intention is to be an executable file that can run on any operating system, but specifically it needs to run on raspbian, since that is the operating system used by our raspberry pi. Electron offers a way to port a localhost application by being able to compile and have it as an executable file. This is the ultimate goal because then this becomes a desktop application making it easy for a user to open and use.

When the application is local to the desktop that makes it easy to use without having to compile and run code for the user. The goal is user friendliness and the way to achieve this is by creating an experience that isn't foreign to the user, but rather something similar to what the industry standards are so they can navigate and operate the application without any real issue.

When the application can be opened as an executable file the final goal is to automate it so that the user can have the application boot up and run immediately when the device gets turned on. The smart hub's only goal is to display the application and so showing the desktop or any other user interfaces outside of the application to the user is pointless. We want to make the hub have its core functionality while only giving the user the option to use the application. This can be done using a script of some sort that runs after the user turns on the smart hub and it gets to the desktop. We deducted tests to see what can break that and see if there are any fixes or bugs we did not expect or account for. Also we can test other operating systems to see how compatible the application is, although the user will default to raspbian because that is the operating system raspberry pi uses, maybe they would want to port the software to another system. This can be relevant if they have another system that they want to utilize as a smart-hub which reigns back to the overall objective of the project which is to give users more control of their own security.

7.0 Project Prototype Testing Plan

Testing our application is a very critical step for delivering a product that the user's will be to be satisfied with. By conducting extensive testing, we minimized the amount of possible bugs that could be detrimental to our project's work flow. Our testing procedures will include software and hardware. The software was tested to ensure that each important aspect of our project performs and behaves as stated in the requirements. Our project's end-to-end testing was conducted by analyzing each of the project's required tasks and laying out detailed testing plans to verify it's behaving accordingly. The software will also be tested using the Jest testing framework that will carry out our project's unit testing and integration testing. Testing our project's functionality will give us the confidence that our project is ready for production and these steps are equally as important as any of the other steps that go into designing our project.

Unit testing does exactly as it sounds, they test individual pieces of the codebase, also known as units, to ensure each component is working properly. When conducting this type of testing, the unit being tested should not be connected to any other units of the application that could have a strange effect on the output. Handling test cases by using this technique makes it easier to diagnose bugs and correct them without affecting other parts of the system. The key advantage of running unit tests is they are very good at isolating bugs since the components are tested separately and the code being tested is relatively small.

Integration testing is generally conducted after all unit testing is successful and before conducting any end-to-end application tests. The main point of integration testing is to verify that all of the working modules and components behave as intended when they're interacting with the other modules and components within the system.

End-to-End testing was used for proving the functionality of our software meets the requirements stated throughout this document. This includes any major requirement that we promise to the user will work as intended. When conducting end-to-end application testing it is important to think of cases that aren't likely to occur often but could cause serious damage to the system if not handled correctly. Before conducting each test, there will be a desired result we are looking for and if the result differs from this, we know the code has a bug somewhere. If the tests pass this doesn't mean our code is bug-free, but it does mean our application works for the test cases we passed to it. That's why it's important to run many tests, and to think of certain cases that could potentially cause problems for the system.

Oftentimes projects that get rushed into production lack thorough testing. Due to a demand on shipping the product quickly this step gets skipped too often and because of this can end up with many bugs and unhappy users. Many times a project will go into production because the creators believe the product is in a working state because they only run a few tests that pass and figure their product is ready to deploy. There are many important tasks that our project is responsible for delivering. If any of these components fail due to lack of testing it can be detrimental to the product and any of the user's who are using it.

7.1 Hardware Test Environment

For most of the data collected, the hardware test environment consisted of the PCM5122 test board the raspberry pi pico development board, and the MAX98306 breakout board. And Raspberry pi 4. Software used to flash the code and trouble shoot inludde Thonny IDE, Circuitpython IDE and Rasberry pi OS. the purpose of the prototype testing was to insure that basic configuration worked and all power and data was supplied correctly. The configuration was setup on multiple bread board and even tried with alternative hardware to insure that the theoretical idea was feasible and achievable. Since the PCM5122 uses I2S we had to calculate the frequency for the world select, bit clock, and data pin to get an acceptable audio sample. This environment would also include the speakers and LEDs light as we have to confirm that all additional function of our system work correctly.

7.2 Hardware Specific Testing

Purpose:

The purpose of the hardware testing is to insure all components receive their needed voltage supply and retrieve data without interruption. Ideally when the raspberry pi receives power every component should receive there needed regulated voltage supply and boot up without fail. All components will also operate at their normal function when called upon and power modes will operate a specified. Data line were calculating by taking different audio simple sound and adjusted accorcingly to prevent distortion ans jitter.

Supplies:

- IP camera
- Raspberry Pi
- Speaker
- LED strip
- Prototype of or PCB & final PCB
- All other hardware that completes the system schematic & software
- Multimeter

Procedure voltage supply:

1. Create abread board prototype for flexible adjustments
2. Apply Voltage by hooking up to power source(rasberry pi or laptop)
3. Using volt meter measure the input voltage of all used nodes for components
4. Compare those results to whats specified in datasheets for accuracy
5. Record the results as either summary data or verbose raw plus summary data

Results: PCM accepts take a current of 32ma at 3.3V, the RP2040 takes a current of 24.4 ma at 3.3V, and the MAX98306 takes a current of 2.6ma at 5

Procedure I2S calculation:

1. Create abread board prototype for flexible adjustments
2. Make connection for I2S pins from pico to PCM5122
3. Play audio and take measurement also take note of the sample of the use audion file
4. Using equation BCLK= WS*WL*Word count
5. Record the results

Results: Using PCM5122 PLL is how we generate our word length of output frequency leaving us with a 16KHz max for our audio samples

7.3 Software Test Environment

For the software testing we can test it prior to the hardware integration. As for the test environment, there are a few different integrated development environments (IDE) that we can choose from and those would ultimately give us more options to choose from when it comes to testing the software. Code is converted from the application level to binary so any IDE that can compile the code we want can be used to test the software. For our purposes, we used the most utilized IDE across the software which is visual studio code. It is a free IDE that can compile any language and has an extensive library of free tools within its marketplace for whatever the user needs.

Visual studio is simply a coding interface that can be used in tangent with a compiler to write and test any code. Depending on the software stack that is chosen and the software utilized from Java to Python or even Javascript, this all can be coded, run and tested within this IDE which is why it is ideal. For the purpose of testing, we can run the application within the IDE and get the websocket RTSP stream from the camera and then use conditional statements to see if the human detection gets triggered. If so, then the A.I. can be affirmed to work.

When that works on our development environment on the computer, we can then port the software to the raspberry pi to see if it also works there and conduct the same experiment on that. We can then set up the breadboard to an LED and use the GPIO library to interface the raspberry pi to send signals to an LED through the breadboard to see if the LED strips would work. The conditional is triggered when a human is detected from the user's IP camera of choice.

Assuming the LED can get lit up from the raspberry pi, we can append to the testing procedure by going to the next test. We can test our raspberry pi with a regular DAC to see if it can configure and work. If we can send signals to that DAC to play audio then that means the software is set up properly to now configure the PCB designed DAC that we created. This makes it more efficient to make sure everything works on the software side of things and that if anything doesn't work then it is most likely an electronic hardware issue.

The common theme is testing software in environments that we know will work prior to adapting our implementation, or the next layer of development, to them. This will make it easier to bug test and determine what potential issues are that we run into because we knew that our project works in certain circumstances but maybe not in others, therefore we can eliminate potential issues before we even have the issues. This can be seen as taking baby steps rather than simply throwing everything together and trying to debug an entire system rather than a smaller sub-system.

7.4 Software Specific Testing

In this section we will lay out a variety of tests in order to test the range of cameras the system will work with, the durability of the housing, and the response time of the system. Each one of these is critical in bringing a complete product to market. The range of cameras the system can work with is vital because the point of the hub is to be able to purchase any IP camera and hook it up to the system. The durability of the hub is important because the system is going to be running constantly so the electronic parts are going to heat up. Finally the response time is important, if the camera has a significant delay then the system does not work as intended.

7.4.1 Testing the Deep Learning Model

Purpose:
The purpose of testing this part of the software is to ensure that the system detects humans without any deficiencies whenever someone is approaching the camera. Ideally, the camera should detect a person from a range of about twenty

to thirty feet. However, it is critical that the system absolutely makes the correct detection when a person gets closer to a range of about five to ten feet. The model we are using for human detection makes predictions using an accuracy-based approach. We'll set up our application to trigger when the model's prediction accuracy rate is 80% or higher. Therefore the further away the human is from the camera, the less accurate the model's prediction may be. But when the user gets closer to the camera, the prediction will become more stable. So to ensure the system doesn't make any false predictions, the model's accuracy rate cannot be lower than 80% before making an assumption. This part of testing will also be set up to ensure it detects humans when the sun is out and when it is dark outside. It will also test for humans wearing different types of clothing, from hats to hooded sweatshirts, and humans with facial hair and humans without facial hair.

Supplies:

- IP camera
- Raspberry Pi
- Linux terminal
- Measuring tape
- Hat
- Hooded sweatshirt
- Four or more humans varying in hairstyles and facial hair

Preparation:

- Design the application to trigger an alert when it makes a prediction of 80% or greater.
- Gather each team member to participate in testing.
- Conduct testing when it's very bright outside and very dark outside.

Procedure:

1. Place the camera at the front door of one of the team member's house or apartment during a bright sunny day.
2. Boot the system and wait for it to initialize.
3. Have one of the team members walk up to the front door. Wait for the system to trigger it and detect a human.
4. Record the prediction's accuracy and the human's distance from the camera when the event was triggered.
5. Repeat steps 1-4 at night when the sun goes down.
6. Repeat steps 1-5 for each member of the group.

Results: We tested our application ten times during the day and at night. The human was detected 100% of the time within 3 seconds.

7.4.2 Testing the Audio and LED Alerts

Purpose:

It is very important that our deep learning model passes all of its tests and the model has been proven to be successful at detecting human presence. After the model has been proven, the LED and the audio component must be put through extensive testing. It is important that the user is alerted with both visual alerts and audio alerts. When a human appears on their property. Although testing the model is important, it is equally important that the sound hub triggers an audio alert and the LED's produce light to alert the user.

Supplies:

- IP camera
- Raspberry Pi
- Linux terminal
- Sound hub
- LED's
- Two humans

Preparation:

- Design the application to trigger audio alerts and glowing LEDs when a human is detected by the application.
- Design a testing program that generates simulated human detection instances replicating the deep learning model. The program will consist of a loop that produces fake deep learning predictions that are supposed to cause the applications alerts to be generated. The loop will run for one minute on and off increments. For the minute the loop is generating outputs, the alerts should be triggering and for the minute the loop is not active, the alerts should not be triggered since it's simulating no human presence.
- Position the camera at such an angle that it can pick up humans from a wide range of spectrum.

Procedure:

1. Power on the Raspberry Pi and the IP camera. Once the device is powered on, a script will automatically boot the application.
2. Have one person stand outside from a distance far enough away that the camera cannot make any detections. The other person that's sitting inside next to the hub will be on a phone call with the person outside to help instruct them on what movements they need to make to trigger the alarm.

3. Once the person inside verifies the application is in a ready state, they will instruct the person outside to start walking up the front door where the camera is positioned.
4. The person that is sitting inside the house will observe the LCD display to watch how close the person gets to the camera when the human gets detected and the alerts begin to fire off.
5. The person located inside is still connected to the person located on the outside through a phone call. Once the model has made the proper human detections, the person located inside will direct the person to slowly back away from the front door until the person is out of the camera's range. The person located inside will continue observing the LCD display to verify that once the person exits the premise, the application no longer triggers alerts and the user of the system will then know the person has exited the property.
6. After steps 1-5 are complete, it's time to pass the application some fake data that replicates the applications desired deep learning predictions.
7. Using the linux terminal, run the program that loops in one minute on and off iterations. One minute the loop will produce simulated human detections, and the other minute it will do nothing.
8. Observe the sound hub and LED's for the first minute verifying the alerts are triggered are they are supposed to. The other minute, observe that the sound hub and LED's aren't triggering any alerts.

Results: We tested our application ten times using humans and ten times using fake data. When using humans, the detection was made 100% of the time. When the application was given fake data, it did as expected and did not trigger any audio or visual alerts.

7.4.3 Durability Testing

Purpose:

Testing the durability of the system is critical, if the system is prone to overheating or there are power supply issues the system will not run for a long period of time. In order to test the efficiency of the system. The thermal output of the Raspberry Pi is crucial to understand the longevity of the hub.

Supplies:

- 1-3 IP cameras
- Raspberry Pi
- Prototype of our PCB

- Linux Terminal
- LCD Screen

Preparation:

- Turn on the system and assure the system is running the AI facial recognition software continuously.
- To start connect 1 of the IP cameras to the system and assure that a feed is present on the LCD screen.

Procedure:

1. Open the pi@raspberrypi terminal and run the command prompt “vcgencmd measure_temp” this will result in a temperature reading of the GPU.
2. Record the temperature of the GPU and record the time.
3. Next on the same prompt window run the command “cat / sys/ class / thermal / tehrmal1_zone0 / temp.”
4. Record the temperature of the CPU and divide the number by 1000 and the reading will result in a temperature in Celsius.
5. These numbers need to be recorded every hour for at least 24 hours in order to receive a complete set of data.
6. The data then needs to be compared to the acceptable range of the Pi.
7. Next add more cameras and record that data and compare again to the acceptable range.

Results: We tested our application ten times using the procedures described above. After testing, the mean temperature was 49.1 degree.

7.4.4 Response Time Testing

Purpose:

Response time is important in every application. Regarding security, the response time is critical. If the response time of the camera is significantly delayed it could cause issues for the consumer and ultimately rule the product useless. A great response time is very important because a person can appear in the view of the camera for a few frames and if it was too quick for the software to render then the software might not detect the human.

Supplies:

- 1 IP camera
- Raspberry Pi
- Prototype of our PCB

- Linux Terminal
- LCD Screen

Preparation:

- To start connect 1 of the IP cameras to the system and assure that a feed is present on the LCD screen.

Procedure:

1. Run the desktop application and open up the developer console.
2. While the application is running with the stream feed showing on the application, go to the IP camera.
3. Cover the camera lens, point it at a person or yourself and then for a few frames of the feed flash the camera of the person and then cover the lens again.
4. Go back to the desktop application and check the developer console.
5. Look for a console log that says the object detected and the accuracy score, no console log would mean that the software and camera was not quick enough to detect a human.

Results: We tested our application ten times using the procedures described above. During testing, the detection was made 90% of the time.

7.4.5 Testing for Pedestrians

Purpose:

Our application must not trigger any alerts if somebody is walking by but not approaching the user's home. For example, if somebody was walking on the sidewalk, the hub should not trigger any alerts, and this would cause the system to produce false alarms making the system untrustworthy. Our application users need to be confident when an alert is triggered that it's because someone is at their front door or approaching their property.

Supplies:

- 1 IP camera
- Raspberry Pi
- Prototype of our PCB
- Two humans
- Linux Terminal
- LCD Screen

Preparation:

- Connect 1 of the IP camera to the system and ensure that the system is running correctly. To run this test, the camera must be pointed in the direction of the sidewalk or street.

Procedure:

1. Start the hub and check that the system is working correctly by having a human walk up to the front door. The alert should be triggered as the human approaches the front door.
2. Once the system is verified to be working correctly, have a human start walking on the sidewalk out of view of the IP camera.
3. The person needs to walk across the sidewalk so that the camera catches a view of the person as they walk past. It is essential that the person does not step up to the camera but only horizontally past the camera. The person should walk slow enough that the camera catches it for at least a few seconds. While the person is walking past the camera, the other person will be inside next to the hub observing that the person walking by is on the camera and no alerts are triggered.

Results: We tested our application ten times using humans walking past on the sidewalk. Our application triggered the alerts 20% of the time giving false positives. In the future we will need to work on lowering this rate to single digits.

8.0 Administrative Content

This section will discuss all the content that would be handled by the lead of the company. The content in this section will include but not limited to budget and financing, marketing strategy, group management and milestone achievement, future direction.

In regards to how the project aspect has gone and continues to go in regards to administrative content, the project group hierarchy was pretty straightforward. Within the group structure there was no dedicated leader or project manager. Most projects, if not all projects in the real world, contain a program manager that manages the administrative side of the program and interfaces between the program and the outside fields. For our project we did not have this and we simply communicated as an effective group to determine what to do.

8.1 Project Personnel

This project consists of four members, each with unique backgrounds that contribute to the development of this product. Each member has a specialty that is utilized. In this section each member will be outlined and recognized for their skills, background and contributions to the overall design of this project.

8.1.1 Matthew Weinert

For me, I started out after high school not knowing what I wanted to do with my professional career. Coming from a smaller town with very little options, I had no clear direction because I also had no parental figures showing me any great paths for myself. After high school, I very slowly was taking 1 to 2 classes a semester at a local college and then as time went on and I got to my chemistry classes, my teacher inspired me to work within a chemistry related field. Eventually an internship opportunity opened up for me and I took it as an analytical chemist for Energizer. It was only late into my first degree, a bachelor's in Forensic Biochemistry, that I realized that was not my destined path as a student.

A roommate at the time was a mechanical engineering major and he would always tinker with electronics and computers in his free time. One day I decided to learn what I could in building a gaming computer because I also needed a better computer, so he helped me build my first computer. Building that first computer gave me a lot of pride and joy in what I created along with a true enjoyment of the process of learning something new. That is when I realized that computer engineering would be my true passion and the closest degree that would fulfill my thirst for all the technical questions I always wondered about how computers work and what software really is.

As I progressed into my computer engineering degree through UCF, I realized something in terms of how the degree is structured. The college splits the degree into two main concepts, the computer science core classes and the electrical engineering core classes. When understanding that, my initial path in this degree wa sto take the software classes first because software is easily engineered and tinkered with so I figured that learning it first would be beneficial because then I can make, practice and learn new software in my free time as I progressed through the second half of the degree which is the EE side of things. After taking computer science 1, I then realized even more how much I loved the software more than any hardware and that is when my portfolio started expanding rapidly.

Later into my software courses after I did a lot of research on the most optimal path of how to become a software engineer, I had a large portfolio of personal projects, a website that displayed information about me and a solid resume, I started looking for internships. Fast forward to today, I am a few months from graduating with three internships for software engineering, even one of them being a lead position, and when I get out of college I will have over a year and a half of professional software engineering experience within the federal contracting industry.

Approaching my graduation with a ton of experience and knowledge under my belt, I was looking for jobs local to me in Orlando. Currently I accepted a generous offer at 4c Strategies as a software engineer while also finishing the project and my last few classes. In relation to this project, my and Korey are the computer engineers that know how to integrate software into hardware whereas Kenneth and Joshua are responsible for the EE side of things. Korey and I have made some prototype software but since we require the hardware we are soft blocked, but nevertheless we are being as productive as we can be and continuously researching and creating software for the project for the time that everything came together we already had everything done that we needed.

8.1.2 Korey Lombardi

My journey to becoming a computer engineer began at the University of Central Florida in the Fall of 2017. Before my studies at UCF, I did not have any programming experience. Along the way, I initially picked up programming skills in the C and Verilog languages. Once I built a strong foundation in coding with the C programming language, I later learned Java and Python. Although I learned how to build electronics, I acquired a stronger passion for building software than building hardware and programming firmware.

For this project, we have two electrical engineers and two computer engineers. For the software side of the project, Matthew and I will collaborate on all aspects of the software. We both have similar strengths in designing software, so we feel it would be best if we pair programmed the majority of the project. From building previous projects, I have gained experience implementing facial recognition using libraries like Tensorflow and Keras. Since I have created these types of applications before, I will most likely be responsible for the deep learning aspect of the project. But I will still run all of my code and design implementations past Matthew as he has agreed to do the same. Although I have programmed firmware in previous applications, this is my weakest skill needed for building this project. By conducting extensive research and using trial and error, I am confident that Matthew and I programed the firmware professionally and efficiently.

8.1.3 Kenneth Ancrum

One of the two electrical engineering majors on the design team, Kenneth Ancrum is responsible for the digital to analog converter research, amplifier research, and overall PCB design and fabrication. Also the lead designer for the overall shematic and power solutions. Kenneth currently has stronger experience with circuit design and control solutions and have currently be prusing a role in the fileds or hardware, system, and control engineering. Kenneth started at UCF in the spring of 2018 and worked to key concepts in a short time. Kenneth collaborated with joshua on PCB design suggestions and the 3D shell design.

8.1.4 Joshua Sherrill

Joshua started at University of Central Florida in the Fall of 2017. After my first year he transferred from Biomedical Sciences to Electrical Engineering. Since switching majors he has worked in various jobs that offer him skills to help complete this project. I worked as an engineering intern with NAVAIR and worked on reverse engineering black boxes of T-45's. He has also gained experience in AutoCad and worked with designing a variety of solutions.

The second internship he had was working for a smart home security company. The idea of home security really came from my experience working in the industry and seeing the needs that were not being met by the current product on the market. This enabled him to experience every type of security system and made him do extensive research on every product that has been on the market for the past 10-15 years. I knew there was a need for something like this because where the market is now and how expensive and extensive the systems are now. He also got to work with the software side of the hardware and I got to see what made these products so special and so effective. This experience is what really helped progress his ideas regarding home security.

In this project Joshua was responsible for a lot of the hardware research, marketing strategy, standards compliance, as well as solutions to hardware issues that arise. Kenneth and Joshua collaborated on the design of the hardware components. Circuit design is not Joshua's strong suit but Kenneth had experience in it and working together they were able to develop a strategy to find solutions.

8.2 Marketing Strategy

The most important task after achieving a working prototype is marketing the product. The marketing of this product will be vital to the success. In the security industry there are hundreds of options for people to choose from so standing out is necessary. The Smart Hub Surveillance system needs to be marketed as a solution for people to have a control system for their cameras. People will also be able to customize a lot of things through the system. They will be able to use this hub to fully experiment with what their cameras can do for them. We are offering a tip of the iceberg product that allows the user to develop more on top of what we are offering them.

For tech connoisseurs this product will allow them to integrate different AI models into the system and allow them to fully expand upon what we are giving them. For the average consumer we are offering them a piece of mind. People get tired of staring at their cameras all day to see what's going on around them, with the smart surveillance hub we are doing all the hard work for them.

The first marketing related thing we would need to do is approach camera companies and try to do a partnership with them where they allow us to offer a package to sell the hub with their cameras. If someone went to the store and saw if they could purchase the smart surveillance hub and two discounted Ring cameras came with it, it would be a no brainer for a lot of people. Aligning ourselves with well established surveillance companies is the first goal.

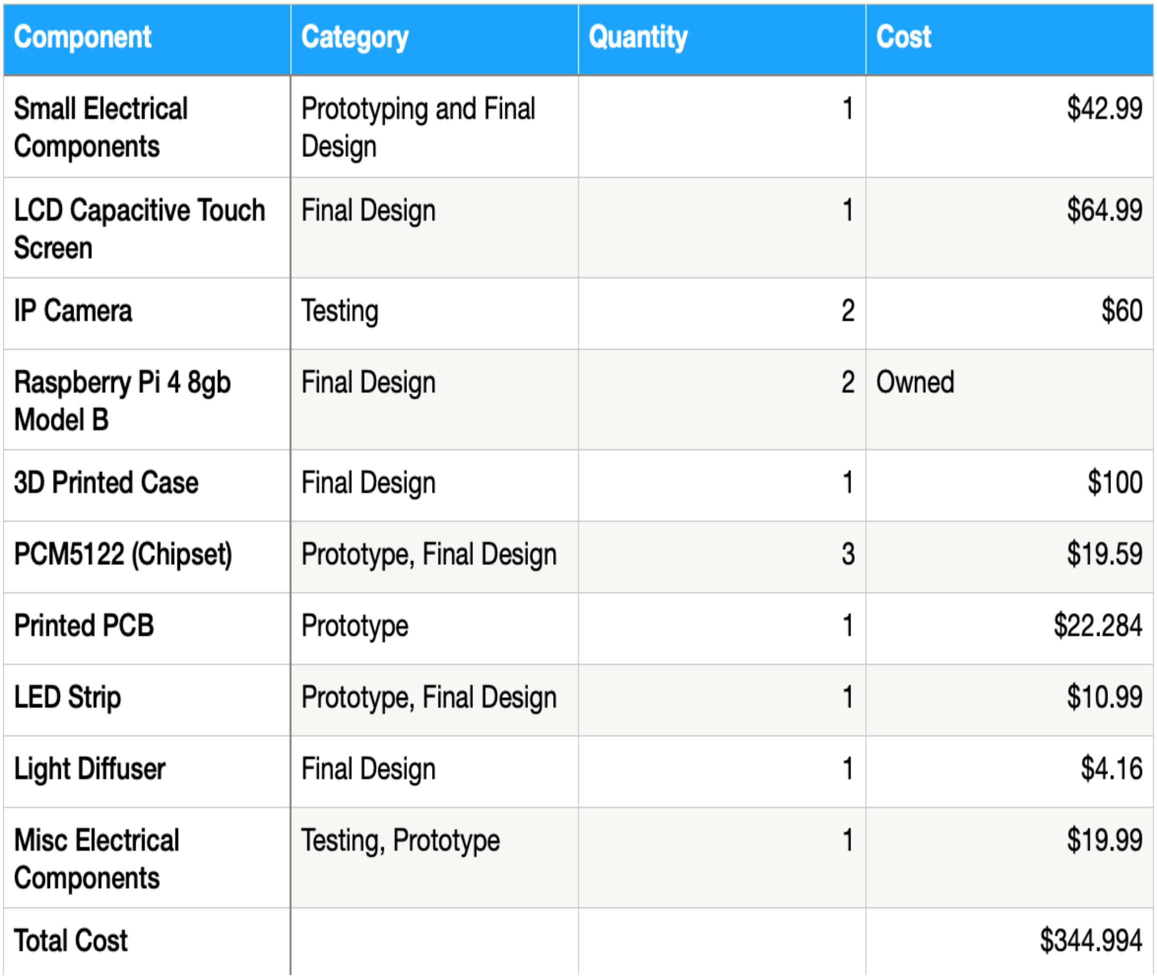
Furthermore, getting our product into a variety of stores would be crucial. Selling the Smart Surveillance Hub at Target and Best Buy would be ideal because those are where tech connoisseurs and our target customers will be. Implementing these strategies will result in a successful product roll out and help with the long term success of the product.

8.3 Budget and Finance Discussion

Our budget is estimated to be around $500, this gives us a little wiggle room when ordering parts. The project will most likely cost less than this but setting a value larger than the estimated cost is important because if things need to be replaced there is room for error. Budgeting out an idea before the design is 100% complete is a difficult task because when prototyping many things can go wrong.

These parts are items that *could* be used, but are subject to change. We are currently researching the most economically efficient parts for their respective capabilities in our block diagram (Figures 3 & 2). When viewing certain hardware we selected the cheapest most effective components that will lead to a cheaper overall build. The idea behind the project is to make a security solution that could be affordable for the average consumer in the market. With that being stated we viewed certain decisions, such as what kind of display interface to use, as a modular choice depending on the specifications and need.

In order to finance this project each group member has agreed to split the cost of the components, so each person will pay ¼ of the total price. As of right now the estimated price to be paid for each group member is around $86.25. This was extremely affordable for everyone so self financing was the best solution. Since the price is affordable for everyone we decided to not look for a sponsor for this project.



Component	Category	Quantity	Cost
Small Electrical Components	Prototyping and Final Design	1	$42.99
LCD Capacitive Touch Screen	Final Design	1	$64.99
IP Camera	Testing	2	$60
Raspberry Pi 4 8gb Model B	Final Design	2	Owned
3D Printed Case	Final Design	1	$100
PCM5122 (Chipset)	Prototype, Final Design	3	$19.59
Printed PCB	Prototype	1	$22.284
LED Strip	Prototype, Final Design	1	$10.99
Light Diffuser	Final Design	1	$4.16
Misc Electrical Components	Testing, Prototype	1	$19.99
Total Cost			$344.994

Table 12: Budget

8.4 Facilities and Equipment

The scope and goal of this project was to create a product that can simply replace current solutions and interface with ‘off shelf’ IP cameras. This entailed creating an at-home security system, or at least a security system anywhere that there is an internet network. As per the facilities, testing and development of this project were don done in the same type of environment that the goals entail.

As far as the development and end-goal environment for our product, we developed the project and hardware in the comfort of our individual abodes. This is good because ultimately this was the target environment for the end-product with all of our deliverables, will be that this system resides in a residential or commercial space. This requires internet access for both the camera and the smart-hub, because that’s the communication median between the two.

As previously mentioned, the project consists of creating a smart-hub that is an all-in-one system that interfaces with the camera. External equipment that is required, but not engineered or created by us is an off-the shelf IP camera. Since the goal is that our product can interface with one of these current production cameras, we want to get an ideal simple camera that offers all the capability we desire. Creating our own camera for this project would be useless and defeat the purpose of the product, which mission is to be a plug-and-play hardware interface with software to display and use any IP camera; therefore it’s ideal that we simply spend our time in researching, developing and engineering the hub rather than any sort of camera.

The last requirement is that there must be internet. This can be achievable by a simple router or wifi. Each IP camera is different, whereas ours that we bought utilizes an ethernet cable to connect to the network and not a wireless internet connection. Since our current camera we are using to produce requires an ethernet cable, then a router is required for development purposes to get the camera on the internet.

8.5 Milestone Discussion

The tasks listed in the table state the goal and the date we want to achieve each goal. Each group member is responsible for coming each week with their required work done and ready to be discussed among the group members. We had issues in regards to the PCB and what was required of us as a group from the professor. Multiple times we would communicate with the professor but we would be told misleading information and this would only be found out after it was too late. This delayed the engineering process of the schematic and pcb design rendering the milestones to not be where we strived to be at this point in the project.

To make up for lost time, we decided to start the software development prior to the hardware being set up. Using the NodeJS route we had the minimum amount of software working, but as we got further we realized that the software stack we were using did not have the desired capabilities we needed to interface with the hardware. In order to do this we need to use python, or at least raspberry pi’s version which is micropython. This doesn't stall anything since we still needed the hardware to be procured still but the time we spent coding was irrelevant. We ended up finishing the hardware after the software and interface both.

8.5.1 Senior Design 1

Senior design tasks include discussing various ideas. We started at creating a mesh radio network, moved to a drone, then to a threat deterrent camera, and finally landed at the smart surveillance hub. Each idea required a significant amount of research and discussion to determine if it was a viable/possible option for all of us to complete given our skills. Once we finally landed on the Smart Surveillance Hub we all had to do research regarding what our idea involved. We then separated the project into different tasks for each group member to complete each week. Every week we showed up and discussed research about what we had discovered and problems that arose. This helped us problem solve together and interface the hardware ideas with the software possibilities. Completing each task in a timely manner was vital to the project design. Moving forward without having a complete design, into Senior Design II, could cause many different problems and result in a not functioning prototype.

Number	Task	Start	End
1	Think tank - Ideas	01-17	01-24
2	Narrow down options, design parameters and select a project.	01-24	01-13
3	Initial Document - Device and Conquer	01-31	02-03
4	DC1	02-04	
5	Review and discuss our project with Lei Wei	02-07	
6	Investigate IP protocol networking	02-07	02-10
7	Obtain an off-shelf generic IP camera	02-10	02-14
8	60 page Senior Design draft	02-14	03-25
9	Explore options for software development	02-17	02-24
10	Determine and choose final parts for PCB	02-24	02-30
11	Assignment of standards	02-28	03-11
12	Test and configure software compatibility between the Pi4 and the IP camera.	03-01	03-10

13	Design the PCB and have it made	03-10	07-20
14	Acquire additional finalized parts	03-10	07-20
15	100 page Senior Design report	03-28	04-08

Table 13: SD1 Milestones

8.5.2 Senior Design 2

The Senior Design II tasks discuss a variety of prototyping steps and tests that need to be taken to ensure the completion of the project. Each task needed to be completed in a timely manner in order to ensure that the prototype was tested and working as designed. Many tasks were needed to be run before the final product was built. The most difficult task to complete before the assembly can be completed is the software and hardware configurations. Every step past that requires heavily on the software and hardware working together properly. Each step after that is just adding a little bit more complexity to the initial task. To complete this in a timely manner each group member needs to work diligently to handle their respective tasks. The Electrical Engineers needed to have the PCB designed and the hardware tested so that when it was passed to the Computer Engineers it was working properly and easily coded.

Number		**Start**	**END**
1	Test Software to hardware configuration	04-09	04-30
2	Develop software for the Smart Notifier	05-01	05-10
3	Develop software for the Hub application	05-11	05-30
4	Design a 3D printed case of AutoCad	06-01	06-10
5	Print 3D printed case	06-11	06-15
6	Stress test the finalized product before final assembly	06-16	06-20
7	Assemble the Smart Hub	06-20	07-21

Table 14: SD2 Milestones

9.0 Conclusion & Final Remarks

The senior design 1&2 project ultimately was a challenge, assembling a group that didn't know each other prior to the class, to come up with and agree upon a single idea that met the standards to the class and spend an entire semester researching and starting the creation of this idea was no easy feat. A large challenge for us as a group was agreeing on the scope of the project. Each project idea we came up with had to be thoroughly researched and discussed to see if we possess the skills necessary to complete the task. We ended up changing our idea around four times after doing a significant amount of research on each topic. Once landing on the current project we also had to deal with a variety of issues regarding the complexity of the design we were going for. We can all say we have learned a lot in regards to time management and communication. Whenever we felt like we could have done better it was because we were lacking those two major components. With that being said, in senior design 2 we were more conscious of our time and planed better while simply getting things done in a faster manner while also working together to maximize our potential and efficiency. Overall the team is really pleased with how the final project turned out and felt honored by the responses given to us by our committee.

Resources

[1] Federal Bureau of Investigation. (2019). 2019 Crime in the United States: Burglary. https://ucr.fbi.gov/crime-in-the-u.s/2019/crime-in-the-u.s.-2019/topic-pages/burglary

[2] ADT. (2022). Is a Wireless Security Camera Worth the Cost?. https://www.adt.com/resources/wireless-security-cameras-cost

[3] Raspberry Pi 4 Specification, Schematic, Diagrams https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

[4] Teske, Lucas. June, 2019. Reverse Engineering cheap chinese “VRCAM” protocol. Medium. https://lucasteske.medium.com/reverse-engineering-cheap-chinese-vrcam-protocol-515c37a9c954

[5] Vivint. 2022. https://www.vivint.com/ppc/brand?cq_src=google_ads&cq_cmp=146236557&cq_con=7629216237&cq_term=vivint&cq_med=&cq_plac=&cq_net=g&cq_plt=gp&utm_medium=cpc&utm_term=vivint&utm_source=google&c_ps=s.google_k.vivint_m.e_n.g_po._d.c&exid=117156&geo=9053023&gclid=Cj0KCQiA64GRBhCZARIsAHOLriIg_qJ0Kg_h_ZH3ORTHyE4MRUnbkoV8cjiA0kLyc_uWDAdT0i54SyUaAtDzEALw_wcB

[6] Ring. 2022. https://ring.com/collections/all-products

[7] ADT. 2022. https://www.adt.com/security-cameras

[8] Jest. 2022. https://jestjs.io/

[9] JavaFX. 2022. https://openjfx.io/

[10] Eclipse Foundation. 2022 The Eclipse Jetty Project. https://www.eclipse.org/jetty/

[11] S. Abdelgawad, K. Yelamarthi. 2019. IoT Based Security System: Pre-College Research Project. http://people.se.cmich.edu/yelam1k/asee/proceedings/2019/1/3.pdf

[12] J. Wang, H. Ping, M. Fang. 2019. Home Security System Using Raspberry Pi. https://people.ece.cornell.edu/land/courses/eceprojectsland/STUDENTPROJ/2018to2019/jw2527_hp394_mf734/project_142_report.pdf -better-choice/

[13] PCMag. 2022. https://www.pcmag.com/encyclopedia/term/mp3

[14] Texas Instruments. 2022. PCM5122. https://www.ti.com/product/PCM5122

[15]Texas Instruments. 2022. PCM5102A. https://www.ti.com/product/PCM5102A

[16]analog devices.2022. ADP150.
https://www.analog.com/en/products/adp150.html
[17][Texas Instruments. 2022. TPS63001
https://www.ti.com/product/TPS63001?utm_source=google&utm_medium=cpc&utm_campaign=app-null-null-GPN_EN-cpc-pf-google-wwe&utm_content=TPS63001&ds_k=TPS63001&DCM=yes&gclid=Cj0KCQjw2_OWBhDqARIsAAUNTTFk1onzqfqkLe8BCpHtk_ARoEiKlkzcJ0MYNdl-cb6WB2y-QfV5-hEaAgJGEALw_wcB&gclsrc=aw.ds
[18]Newark. 2022. MAX98306ETD+T.
https://www.newark.com/maxim-integrated-products/max98306etd-t/audio-amp-class-d-3-7w-tqfn-14/dp/73Y6584
[19]. Raspberry Pi Datasheets.
https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf.
[20]ATmega 32 Datasheet.
https://ww1.microchip.com/downloads/en/DeviceDoc/doc2503.pdf.
[21]ESP32S3 Series - Espressif.
https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf.
[22]Raspberry Pi RP2040 Datasheets.
https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf.