Deployable Security System

Group 15 Authors:

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

Review Committee/Mentors:

- Dr. Chinwendu Enyioha – *UCF ECE Assistant Professor*
- Dr. Chung Yong Chan – *UCF ECE Senior Lecturer*
- Dr. Mike Borowczak – *UCF ECE Associate Professor*

Project Motivation and Background:

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

Goals / Objectives:

Basic Goals:

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

The node(s) in an alarmed state will have the capability to sound an alarm, flash an LED, and have the cameras operate in an excited state.

Advanced Goals:

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

Stretch goals:

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

Project Functionalities:

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

Real Time Motion Detection:

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

Perimeter Security:

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

Access and Control:

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

Emergency Response integration:

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

Threat level alert system:

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

Feedback footage of detected motion/threats:

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

➢ reference any input from customers or marketing analysis of comparable products/projects that has been used to identify project features

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

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

Existing Products:

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

Specifications:

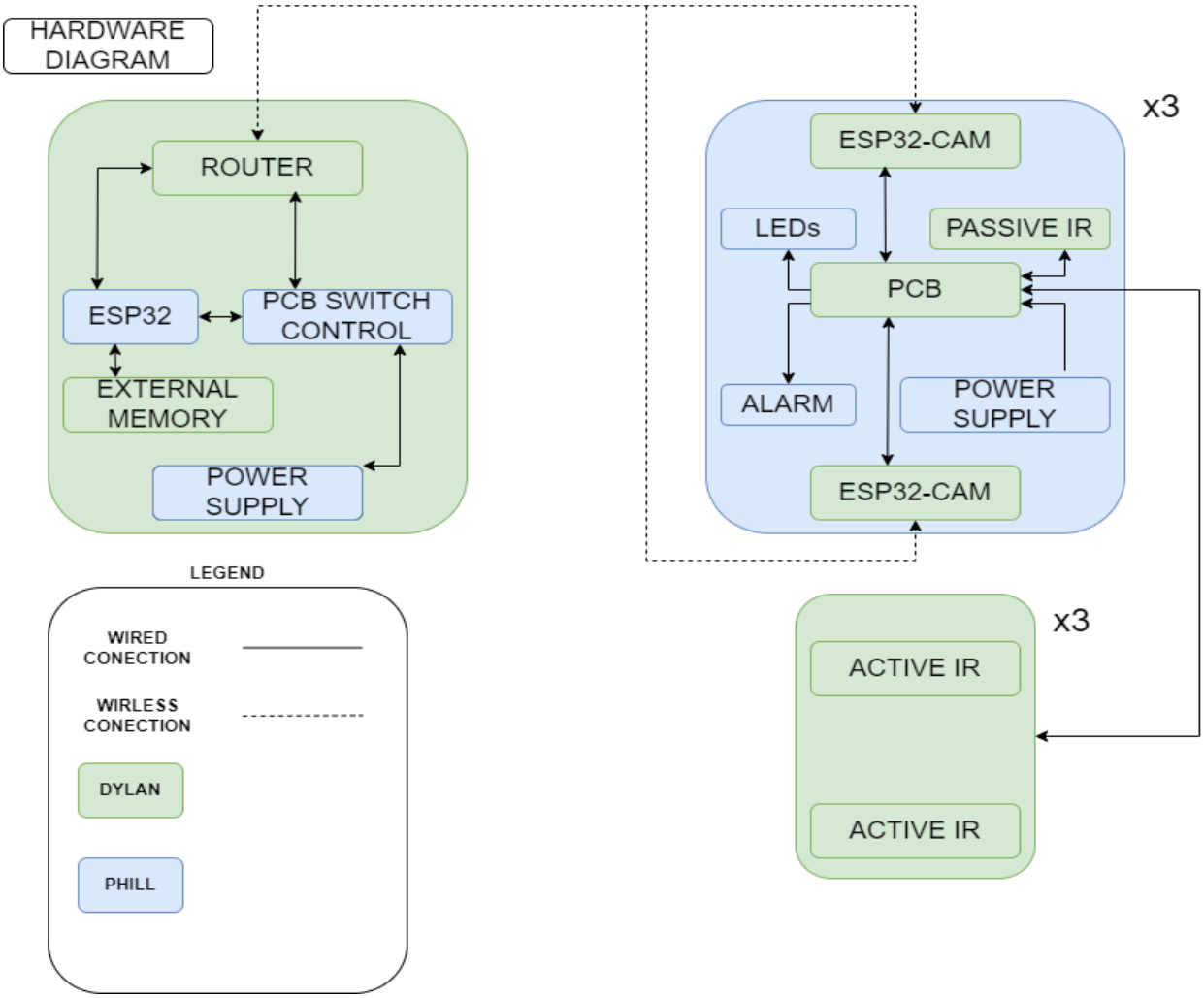
Specifications	Minimum	Maximum
Alert Accuracy (Time)	75%	100%
Power Consumption (node)	~77mA	~1200mA
Detection ranges	5 feet	15-20 feet
Response time	0.5 sec	10 sec
Scalability	50-60%	90- 95%
Installation time	Program Run Time	Program Run Time
Encryption level	128 bits	256 bits
Cloud latency	10-50 ms	50-100 ms

Hardware Planner:

Work Items	Name	To be Acquired	Acquired	Investigating	Designing	Prototyping	Completed
Control Hub PCB	Phillip	x			x		
Node PCB	Dylan	x			x		
Control Hub power	Phillip	x		x			
Control Hub housing	Dylan	x			x		
Router	Dylan	x		x			
Esp int.	Phillip		x		x		
Esp cam int	Dylan		x		x		
IR sensor (active)	Dylan	x		x			

IR sensor (passive)	Dylan	x		x			
LED/Alarm	Phillip	x		x			
Node Housing	Phillip	x			x		
Node Power	Phillip	x		x			
System Integration	Dylan	x		x			
Memory Integration	Dylan	x		x			

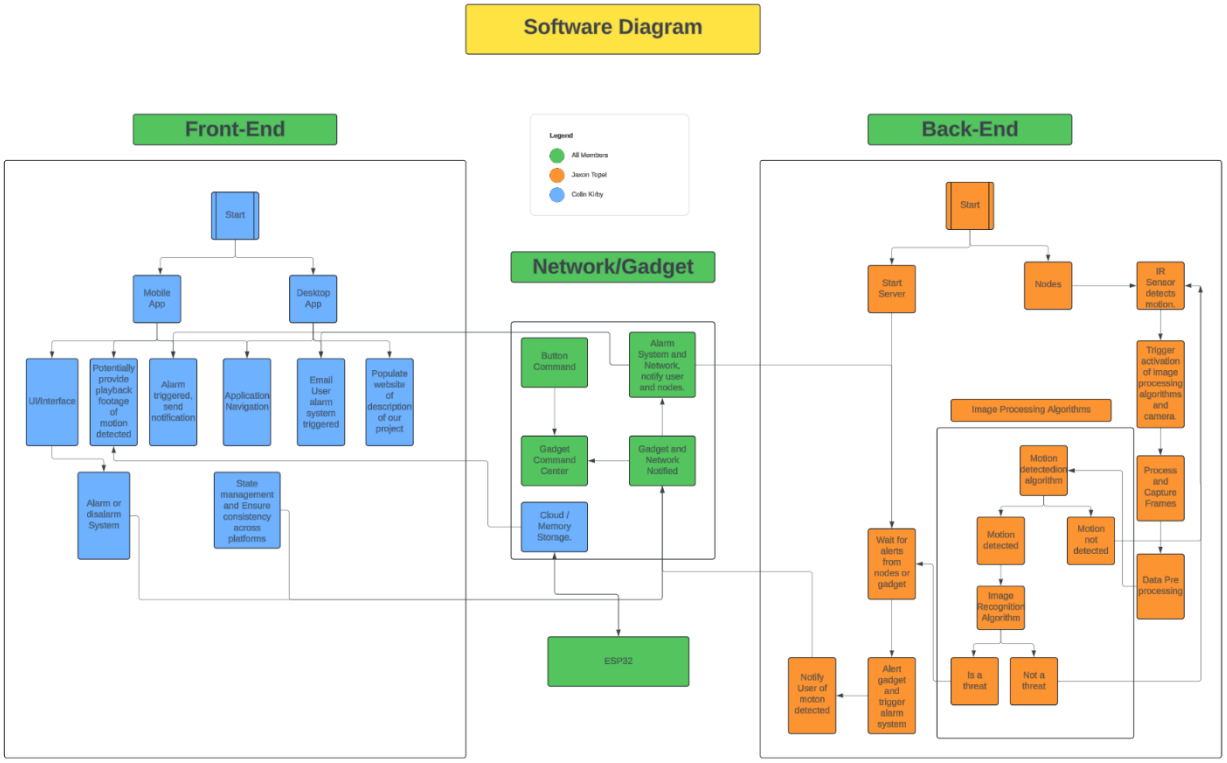
Hardware Diagram:



Hardware Overview:

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

Software Diagram:



Software Components Overview

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

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

Main.py:

From the very start of our software (main.py) we are importing from many of the files that our database will contain. We will create the network/server that our nodes and our gadget are going to use to communicate with each other. After this we will instantiate the 3 nodes for our system, instantiate the gadget inside our network, and then add all 3 nodes to our network as well. After this, we are going to have the software on each esp32 node running at all times, therefore, we have a continuous loop that will monitor the network, look for alerts from the gadget (details discussed

below), and then will monitor the environment of each node (also discussed in more detail below). From this central point of the software, we will accomplish the goals of a secure, abstract, and scalable system that will allow us to develop easily throughout the next two semesters.

Gadget.py:

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

Network.py:

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

Image_processing.py

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

Node.py:

This file serves as the representation of our nodes in our security system. Instantiated 3 times in main.py, our constructor when called will assign the node id, location, network, and then instantiate an image processor inside each node. Eventually once we get access to the hardware, we will have to assign an esp32 to each node. After the constructor, we have a function that will

monitor the environment. Inside of here we will use our IR sensor to monitor for movement outside the secure area. When movement is detected, we would call the image processor algorithm to identify if the movement detected was a threat. If the movement detected was a threat, then we will send an alert to the network. Lastly, the next function is the one used above that sends an alert to the network.

Ir_sensor.py:

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

Note:

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

Prototype Illustration:

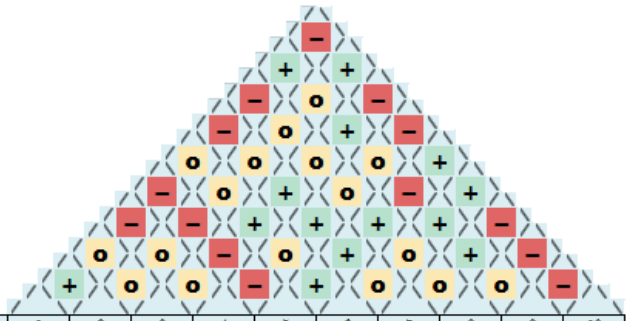
We do not have a physical prototype of our project yet; however, we have found a similar design for what we wish to accomplish. Differences between the product below and our design idea consists of a system that lives inside of a computer network, we also will have Infrared sensors as an additional measure of security, lastly along with a central gadget that will serve as the governing point of operations for the user.

House of Quality:

Correlations	
Positive	+
Negative	–
No Correlation	o

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

Direction of Improvement	
Maximize	▲
Minimize	▼



Column #		1	2	3	4	5	6	7	8	9	10
Customer Requirements (Explicit and Implicit) / Functional Requirements	Direction of Desirability	Size	Weight	Response Time	Communication Range	Battery Efficiency	Fault Tolerance	Signal Integrity	Weather Resistance	Latency	Cost
Direction of Desirability		▼	▼	▼	▲	▲	▲	▲	▲	▼	▼
Affordability	▲	↓↓	↓↓	↓	↓	↓↓	↓	↓	↓↓	↓↓	↑↑
Accuracy	▲	⊗	⊗	↑↑	⊗	↑↑	↑↑	↑↑	⊗	↑↑	↓↓
Battery Life	▲	↑	↑	⊗	⊗	↑↑	⊗	↑	↑	⊗	↓↓
Environmental Adaptability	▲	↑	↑	⊗	↑↑	↑↑	↑↑	↑↑	↑↑	⊗	↓
Portability	▲	↓↓	↓↓	↓	↑↑	↑↑	⊗	↑↑	↑↑	↑	↓
User Experience	▲	↓	↓	↑↑	↑↑	↑↑	↑	↑	↑↑	↓↓	↓↓
Range	▲	⊗	⊗	↓↓	↑↑	⊗	↑	↑↑	↑↑	↑	↓↓
Set Up Difficulty	▼	↓	↓	↑↑	↑	↑	↑↑	↑↑	⊗	↑	↓↓
Reliability	▲	↑	↑	↑↑	↑↑	↑↑	↑	↑	↑↑	↑↑	↓↓
Connection Time	▼	⊗	⊗	↑↑	↑↑	⊗	↑	↑↑	⊗	↑↑	↑↑
Product Targets		~6 in^3	~5lbs.	> 1 s	< 20 ft	1 - 2 Months Per Charge	≤ 0% fail	≥ 95%	1955	≤ 100 ms	~$300

Budget and Financing:

Budget/pricing					
Part	**order group (EA,box,ect.)**	**amount**	**cost / per**	**cost total**	**suggested part**
esp32-cam	group of 3	2	22.5	45	esp32-cam
rechargeable D batteries	EA	24	~6	144	rechargeable D batteries
rechargeable 9v batteries	EA	24	~6.25	150	rechargeable 9v batteries
pcb 1 build	EA	~3	~30	90	
PCB 2 build	EA	~1	~30	30	
passive IR	pack(4)	1	10	10	1.78" passive IR
active IR	pack(10)	1	6	6	active IR
LEDs	pack(200 assorted)	1	10	10	LEDs
speaker	pack(4)	1	10	10	speaker
Housing nodes	EA	3	25	75	
housing gadget	EA	1	25	25	
wire (22 awg)	pack	1	15	15	wire (22 awg)
SSD	EA	1	60	60	
micro router	EA	1	45	45	micro router
total cost				715	

For this project, the development cost is steep regarding early prototype design. This analysis focuses on purely the material needed to do a first build. For this cost analysis, we have

disregarded the tools and equipment needed for the development and any disposable items. This is to show the price breakdown for each part, and the total for a single build.

Project Milestones:

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

Citations, Research, and Resources:

Appendices:

- **Appendix A – reference**
  - Wikipedia contributors. (2024, August 22). Digital image processing. In *Wikipedia, The Free Encyclopedia*. Retrieved 19:18, August 30, 2024, from https://en.wikipedia.org/w/index.php?title=Digital_image_processing&oldid=1241614779
  - Wikipedia contributors. (2024, August 13). Signal processing. In *Wikipedia, The Free Encyclopedia*. Retrieved 19:19, August 30, 2024, from https://en.wikipedia.org/w/index.php?title=Signal_processing&oldid=1240021905
- **Appendix B – copyright permission**
  - ChatGPT
    - ChatGPT was utilized throughout the document to review and refine written paragraphs, enhancing their clarity and flow.
  - Lucid Flowchart
    - Lucid Flowchart was used to make the software diagram.
  - Draw.io
    - Draw.io was used to make the hardware diagram.
  - HULPPRE
    - Used an illustration to provide reference of our competitors HULPPRE Solar Outdoor Motion Sensor Alarm