Group 34: Sentry Tracking Attack Turret

Code name: *S.T.A.T*

Nicholas Martucci	*Electrical Engineer*
Carlos Garcia	*Electrical Engineer*
Devak Sharma	*Computer Engineer*
Christian Solanet	*Computer Engineer*

Review Committee

Dr. Wei Sun	*ECE*	Professor
Dr. Qifeng Li	*ECE*	Associate Professor
Dr. Suboh Suboh	*ECE*	Associate Professor

Mentor

Sreeram Sundaresh

Table of Contents

2.0 Project description

Our project is not groundbreaking or something no one has thought of before. This project is to show the wide range and thoughts that engineers must consider. We all need each other to move forward, and one wrong step could mean the difference between keeping our country safe or tumbling down around us.

Our group decided to build a system that can track a target in a set coordinate system and be able to take it out using a turret. This is no easy task as multiple sensors need to be in communication along with needing to know what exactly to look out for. Should it look for a sudden movement or a certain color? Should it be if anything is sensed in this area, we should start tracking it?

How is this accomplished? Let us start with the subsystems we have. First is the power system/access box which will take a 120V/12V AC/DC converter from a wall plug in and then step the voltage down for further use. Voltage will then be distributed from there to the MCU, LEDs, buttons, and all other peripherals. We will have separate access panel which will house the power system along with status LEDs, buttons, kill switches, etc. for this project. All human inputs and status updates will be on this box. We will have LEDs for status such as friendly or enemy target as well as to know if our turret has locked on. We will have switches to turn on our devices as well as emergency kill switches in case something happens that is unexpected. Another subsystem we have is the Turret substation which will house the MCU due to the fact that the turret with its motors and servos will need most of the wiring connected to it. We plan on having this project be able to break down, which will be explained in a later section so having less wired connections externally will help with our requirements to stay portable and user friendly. Our last subsystem is the Sentry Camera set up along with our HuskyLens sensor on the turret itself. The Sentry Cameras will be used for early detection of an object and will allow our system plenty of time to read the target, determine if it's friend or foe, and give ample time to respond to this object. The Sentry Cams will send over the target's location to the HuskyLens which is attached to the turret. The turret will lock on to the objects position and will first turn on the laser diode to show we have locked on to said target then we will fire a dart at the target.

To help with the setting up of our system, allowing it to be repositioned, we have our systems connected by some external wires. As stated before, our access box will serve as power to the whole system. We will have wires that come out of this box that will connect and power the turret as well as the sentry cameras. We will want to position the turret and cameras however to minimize not only latency but also offset of the camera coordinates from the turret. This can be done by hardware, however, to avoid mistakes with our control, placement of the turret and camera should be closed and allow transition and translation of data to be easy.

2.1 Project background/motivation

The design of our project is set up to mimic how a battlefield would work in a real-life military situation. We have Sentry cameras to sense our target. This could be representative as a pilot giving coordinates to the naval ship which would have the artillery shoot the target in question a thousand miles away. For simplicity, our

information is communicated by wires. The pilot would give coordinates in which case the ship would have to adjust to those coordinates. Once the coordinates are locked on by the turret, the ship would be able to shoot, taking out the target. Just like our project, which requires data from the Sentry cameras to get the initial location. Then our turret must lock-on shown by the laser diode and be able to shoot the target as well.

2.2 Current technology/Past projects

Military technology – C-RAM

The C-Ram stands for Counter-Rocket, Artillery, and Motor. This technology will sense incoming projectiles and be able to counter them using predictive planning and missiles to stop them. This relates to our project in the sense that we will want to be able to see our target and stop them in their tracks by intercepting them. C-RAM and our STAT require the use of sensors to detect and guess how the target is moving to accurately suppress the threat.

Traffic Stop Watchdog

This project is taking slight inspiration from one of our own engineer's brothers and everyone's mutual interest in this type of technology. A past project called Traffic Stop Watchdog had a camera mounted on a movable base that would be able to scan a room and use image detection to determine if there was an officer present in the room based off the color of their shirt. The camera will then keep the officer in the center frame while they perform their duties. The scanning function of the Traffic Stop Watchdog served as inspiration for the two stationary sentry cameras that we will be using. The tracking functionality served as inspiration for the pan-tilt mount which will ultimately follow our target object. These cameras will work together to detect, locate, and follow an object of our choosing.

Nerf Dart Turret

Other projects have tried to build automated Nerf style turrets, but most stop short of full autonomy. One example is the Nerf Dart Turret posted on Printables. That design uses a computer and WiFi link to control the turret, letting the user aim and fire remotely. It's more about showing off the mechanical launcher than tracking a target, since the system doesn't detect or move to an object on its own.

Our design builds on this idea but adds several important layers. Instead of relying on manual control, the turret uses two stationary cameras to constantly watch the area. When motion is detected, the turret wakes up and its own onboard camera takes over for precise aiming. IMU provides orientation feedback to smooth turret motion and prevent overshooting. These additions make the system capable of detecting, tracking, and firing automatically, without operator steering.

2.3 Objectives and goals

The main goal is to be able to acknowledge and track a ground target using the 2D coordinate system set up by the sentry cameras on the housing station. We also want the turret to enter a fully "locked on" state. Other goals can be found below.

Main goals

- Build a stationary turret capable of detecting and tracking an object within a 270° arena.
- Turret locks on to target on ground with data from sentry cameras with laser within 1 second of receiving position data.
- Access box signals that an object has been detected.
- Turret hits target with 60%

Advanced goals

- Turret hits target with nerf dart with above 80% accuracy
- Be able to differentiate between friend and foe at the same time

Stretch goals

- Be able to track aerial targets moving in x, y and z axis.
- Be able to lock on to aerial targets based on data from sentry cameras.
- Be able to see laser from diode on aerial targets before firing with in 500ms of receiving position data.
- Be able to hit target in air with nerf dart 75%.

Objectives

Main Objectives

- **3D Printed Housing Station** – Provide a stable base for the turret and secure mounting for the sentry cameras at the required height. The housing will allow detachment of wiring and PCBs for safe storage, while ensuring stable support to prevent camera damage.
- **3D Printed Turret** – Custom design to integrate the shooting mechanism, HuskyLens sensor, and laser diode as a single moving unit. The printed structure allows for optimized motor startup, reliable sensor response, and the ability to tailor hardware/software performance beyond what off-the-shelf kits provide.
- **AC/DC PCB** – This PCB is needed for power. Battery would be sufficient, however, we are focusing on motion tracking and targeting, efficiency is a secondary goal. To allow us to work and worry about power at a later time we are using a wall plug in to power our system. This requires us to take a 120V/12V AC/DC converter wall plug into a PCB on the access board where we will then further regulate the voltage for other uses such as LEDs, buttons, MCU, and motors.

- **Sentry Cameras** – These cameras need to be able to detect that an object has entered its field of view, pinpoint where the object is on the screen, and relay this information to the MCU. Extensive research is needed to choose the correct camera for this operation. We need a camera that can take in the software and also give us back information. They need to be able to communicate with each other as well, if not the theoretical 2D coordinate system will not work as intended.
- **Huskylens sensor** – Motion tracking sensor that allows for full lock on attached to the turret. Using this sensor allows us to track the target after we are given the initial coordinates from the sentry cameras.
- **Laser Diode** – This is a visual verification that the turret has locked on to the target in case any failure happens when it comes to shooting the target.
- **Flywheel Motor Speed** – To obtain a consistent accuracy of 60% the speed for the flywheel brushless motors will be monitored and adjusted to keep a constant speed of 8,000–12,000 RPM

Advanced Objectives

- **Color-tracking system** – To be able to differentiate between friend and foe we would need to preprogram our sentry cameras to only detect an object of a certain size as well as a certain color. For example, if we have a green ballon in front of the camera and a red balloon in front of the camera the camera would only send position data of the red balloon since we would only want to detect enemy targets.
- **Accuracy –** To improve the accuracy of our turret, higher resolution cameras (e.g., Wyze Cam or Pi Camera) paired with a single board computer such as the Raspberry 4 can be used to run more advanced computer vision algorithms (OpenCV-based color/blob detection or lightweight object recognition). This setup would allow more precise detection and tracking while offloading the heavy image processing workload from the ESP32-S3, keeping it focused on real-time control of the mount and turret.

Stretch Objectives

- **(X, Y, Z) Location** - Use the apparent pixel size of the object in one or more camera feeds, combined with the camera's intrinsic parameters (focal length, sensor size, distortion coefficients), to estimate distance. Then calculate the objects X and Y coordinates by mapping the object's image centroid to real-world coordinates via camera calibration. This will need to be done within 20 – 50ms, with the maximum being 100ms.
- **Predictive Tracking -** Use a predictive tracking filter (Kalman / Extended Kalman / Constant-acceleration model) to estimate future target location at projectile impact time. This will need to be done within 10ms.
- **Faster lock on** – Achieving a faster lock on to the target is possible through a few different ways. We could have multiple ESP32 to split up the utilization of our system, allowing for faster processing of data. We could also prioritize lock on using FreeRTOS allowing this task to be at the forefront of the data processing. The last way is reducing the camera load and using a lower resolution on the Huskylens could allow for that process to end earlier, allowing the laser diode to act faster as well. The lock on should take no longer than 200ms.

- **Accuracy Improvement –** The system will use predictive tracking algorithms and confirm that the system is locked onto the target for more than 1s. It will then shoot the dart and accurately strike the target 95 times for every 100 attempts.

Software Specific Objectives

- The system must be able to generate a world coordinate system that each camera can transform its local coordinate system into to move the pan-tilt mount accurately.
- The two stationary cameras must be sampled one after the other and take precedence over whichever camera detects an object first.
- The two stationary cameras must determine an estimated distance based off the object being a known size
- The two stationary cameras must be able to ignore objects too large or too small to avoid hitting humans or objects too far away
- The system shall store the detected objects with a flag marking if they have been previously detected by the sentry cameras
- The software shall use the distance detected to determine what external indicators and peripherals should be activated or deactivated.

2.3.1 Hardware *Basic Goals*

The turret operates on a simple principle. Detect a target, track them, and fire safely at close range. Two stationary cameras monitor the area, waking up the turret when motion is detected. These cameras act as the first line of detection; they're always watching but don't drain power since they're not constantly processing. Once triggered, the turret's own camera takes over aiming, making sure it only fires when targets are within about five feet.

The mechanical system needs different motor types working together, a stepper motor handling horizontal rotation, giving smooth sweeping motion across the target area without the jerky movement you'd get from a regular DC motor. Vertical tilting uses a servo motor, which provides enough precision for elevation adjustments without getting too complex. The servo can't rotate continuously like the stepper, but it doesn't need to for simple up and down movement. The projectile system uses dual flywheels driven by a brushless motor, keeping dart speed consistent for better accuracy. The flywheels spin up to speed and stay there, so when a dart gets fed between them, it launches with the same velocity every time.

Power and safety controls sit in a separate access panel, positioned where users can reach it safely without getting in front of the turret. The system runs off a 12-volt wall adapter that is certified to take advantage of its safety benefits like overvoltage and overcurrent protection. Internal circuits then step-down voltages for the electronics, motors, and cameras since they all need different power levels. The control panel has the key safety features: an emergency stop button for instant shutdown, and LEDs showing if the system is powered, armed, ready to fire, or has a fault. A buzzer gives audio alerts too, which helps during demonstrations.

Safety drove most of the design decisions from the start. The launcher only fires soft foam darts like Nerf darts with motor speeds capped and measured, so they don't hit too hard and have higher accuracy. This keeps the demo safe for a classroom setting, which was always the main goal. The emergency stop provides backup protection if something goes wrong.

Splitting the design into two parts (control panel and turret) made everything safer and more reliable. It's easier to build this way since you can work on each part separately, and demonstrations can't accidentally become dangerous because all the safety controls are physically separated from the shooting mechanism. The modular approach also makes troubleshooting easier when something inevitably goes wrong during development.

2.3.2 Software *Basic Goals*

The software is built on the ESP-IDF/Arduino framework for ESP32-S3, which provides access to FreeRTOS multitasking, hardware timers, and communication devices. On the stationary (sentry) cameras, the built-in camera driver library is used for low-resolution frame capture, while lightweight OpenCV-style image routines (i.e. frame differencing or blob detection) run locally. Each node uses the ESP-NOW wireless library to send and receive data between devices with low latency and high throughput. The HuskyLens on the pan-tilt mount is accessed through its dedicated HuskyLens library (UART/I2C) to read bounding boxes, object IDs, and lock state. The ESP32-S3 microcontroller uses the FreeRTOS to separate the responsibilities of communication, motor control, safety monitoring, and ensure the turret remains responsive while processing vision data.

The laser and camera are coordinated in a hierarchical fashion. The stationary cameras on the left and right side of the housing station perform coarse detection of motion or color and transmit only target coordinates rather than full images either wirelessly or through UART/I2C. The ESP32-S3 moves the pan-tilt mount to point the main camera at the suspected target region. Once in view, the HuskyLens takes over for fine tracking and object/image recognition. A PID control loop continuously adjusts the turret until the target is centered. When the HuskyLens confirms a stable lock, the laser is enabled through GPIO controlled driver circuit. For safety measures, the software enforces interlocks and mechanical limits before activation.

A PID controller will be used to stabilize the mount’s horizontal movements (driven by stepper motor) and vertical movements (driven by the servo motor). This controller utilizes the IMU (gyroscope and accelerometer) to measure the angular velocity and acceleration, while the PID algorithm computes the proportional, integral, and derivative errors to adjust the motor outputs, ensuring stable and accurate orientation of the mount. A Complementary or Kalman Filter is applied to fuse the sensor data, so that the noise from the accelerometer and drift from the gyroscope is reduced, yielding an accurate and stable angle estimate.

The display design provides real-time feedback to the operator. To indicate different stages of operation, multiple LEDs will be used. When the system is powered, a green LED light up to indicate that the system has received power. A blue LED will light up to indicate that the stationary cameras are operating and are currently scanning objects. If an

object has been spotted by the stationary cameras, the orange LED will light up. Once the pan-tilt mount locates the object, if it detects an enemy, there are three possible LED outputs. If an enemy is detected but it is beyond the 10-foot range, the orange LED will stay on. If an enemy is detected and it is inside the 10-foot range, the red LED will turn on. If the enemy is within 5 feet of the pan-tilt mount, the red LED will start rapidly blinking.

The HuskyLens camera used for object detection comes equipped with a built-in 2-inch screen that displays the camera visuals in real time. This screen will be used as secondary verification that the camera is correctly detecting and identifying the objects in its FOV. The primary verification that the camera and mount are working properly will be the laser diode that pinpoints the target when identified.

2.3.2.1 Why didn't we use Arduino or Raspberry Pi?

The ESP32-S3 was specifically chosen because of its dual-core processor, which allows it to handle multiple functions simultaneously. One core can run vision and communication, while the other core handles peripherals and control loops. With 45 programmable GPIO pins, the ESP32 can support many types of interfaces such as displays, cameras, and sensors through UART, SPI, or I2C. The ESP32-S3 is the smallest and lightest microcontroller out of the three options, so the mount won't be weighed down by it, and provides leeway when choosing the cameras and other components. It costs around $10-$15, making it cost effective.

Table 1 Comparison of MCU selection

ESP32-S3	**Raspberry Pi Pico 2 (RP2350)**	**Arduino Uno R3 (ATMega328P)**
Dual Core 32-bit Microprocessor at 240MHz	Dual Core Microprocessor at 150MHz	Single Core 8-bit Microprocessor at 16MHz
512KB SRAM	520KB SRAM	2KB SRAM
45 programmable GPIOs	26 programmable GPIO pins	Supports audio
Low power management	Supports audio	1 USART, 1 SPI, 1 I2C interface options
Supports multiple interface options	Supports multiple interface options	Brown out detection

4MB-64MB of flash memory	4MB of flash memory	39 GPIO pins
Dimensions: 21.7 x 17.8mm, 1.2g	Dimensions: 51 x 21mm, 3g	Dimensions: 68.6 x 53.4mm, 25g
Input voltage of 3.3V	Input voltage of 3.3V	Input voltage of 5V
Ranges from $10-$20	Ranges from $5-$15	Ranges from $20-$30

2.4 Requirement Specifications

Below you will find our Specification requirements for demo purposes (highlighted in orange) along with other requirements we have set for ourselves to further our knowledge in real world engineering aspects during this project.

Table 2: Specification Requirements

Parameter	**Description**	**Target Value and Unit(s)**	**Market requirement corresponding #**
Sentry Range Detection	Sentry cameras must be able to detect an object that has entered its desired range and create a bounding box around said object**.**	Greater than eight feet	1,3
Success Rate of Detection	This would be the success rate of sentry camera detection of object along with the turret detection and laser diode lock on.	Greater than 90%	1,3,5
Turret Reaction Time	The time that it takes the turret to wake up, position itself towards the object, detect the object, and determine if it should lock onto it must be quick.	Less than or equal to 2 seconds	1,3
Success Rate of shooting dart	After the turret is locked on with the laser diode this would be the percentage of accuracy of being able to hit the target with a nerf dart.	Greater than 70%	1,3

Size	The dimensions of the housing station combined with the dimensions of the turret must be able to fit comfortably on a school desk.	Size of desk is about... Length less than equal to 30 inches Width less than equal to 24 inches	2,4,5
Weight	We want this to be portable so the weight must be manageable by the average person	Less than 25lbs	4,5
Connectivity	The delay between cameras taking a sample and sending it to the MCU which then sends the data to the HuskyLens attached to the turret	Less than or equal to one second	1,3,5
Setup/Tear down time	The device does not need to be set up quickly because it is not used in high-risk scenarios with a strict timing requirement. The device does not need to be taken down as it is designed to be installed and left alone.	Less than 5 minutes	2,4,5
Cost	This is a project design by college students so the cost should not be very high	Less than $1500	all

Table 3 – Marketing Requirements

Below can be found the marketing requirements our group has settled on for this project.

Marketing Requirements

1. Fast detection and fast lock on times
2. Ease of physical set up for user
3. Ease of use for software, only need to change what target to detect
4. We want to make sure this system is small enough to move around to reposition if needed (aka Portable)
5. Cost efficient

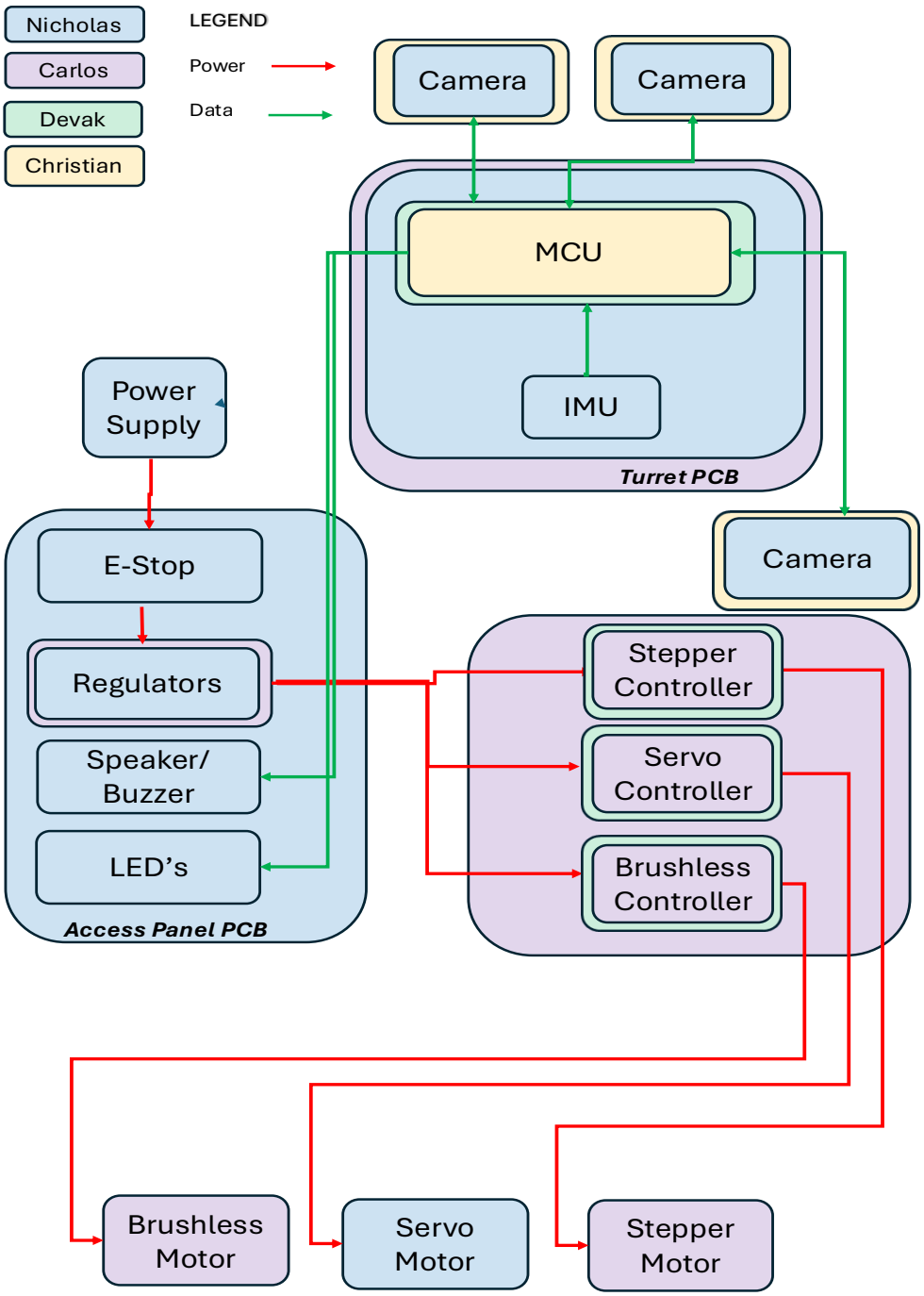
The table below is meant for showing more specifics on both the hardware side and the software side of quantitative values we expect during demonstration and test.

Table 4 - System Requirements

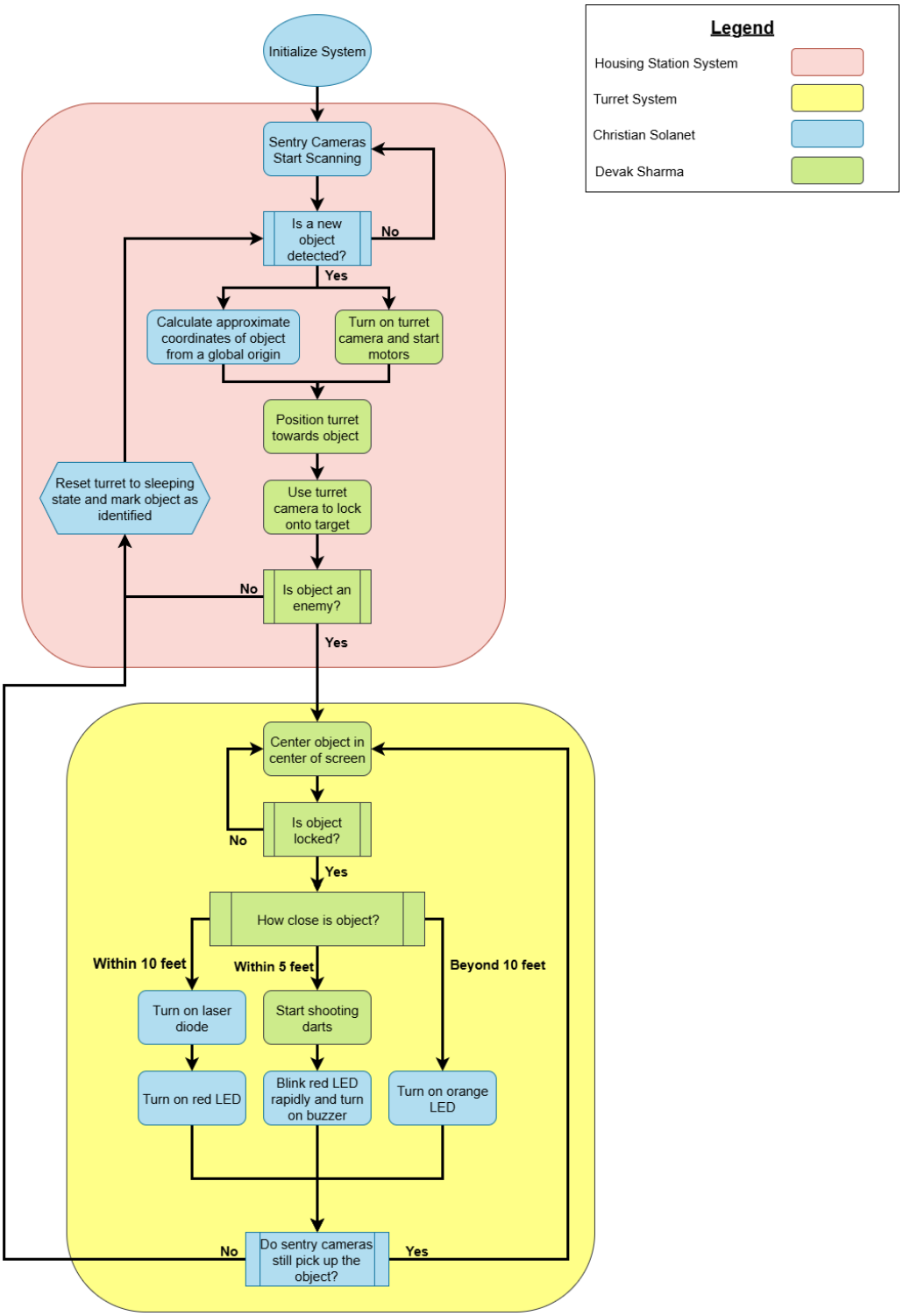
Component	Parameter	Subsystem and description	Target Value and Unit(s)
Cameras	Field of view	Software – The two stationary cameras need to provide an overall FOV that is sufficient to detect objects around the area.	Greater than 90 degrees
Cameras	Sentry Detection Time	Software – Two cameras need to be able to detect an object that has entered the desired range to then send a signal to wake the turret up	Detect target with at least 10 feet from cameras
Cameras	Turret Detection Time	Software – The turret camera needs to be able to detect an object and aim.	Detect/aim in < 2 seconds
Servo Motor / Stepper	Pan/Tilt angles	Hardware- The turret will be able to pan and tilt to the minimum angles that the fixed cameras are capable of detecting.	Pan- 270° Tilt- 90°
IMU (accelerometer / gyro)	Raw data processing	Hardware – Take in raw data to input into PID controller to stabilize signal to send over to the	Stabilize

		turret for further use.	
Access box	User input and status updates	This box will act as the user input for the system. We will have buttons to turn on the system and emergency stop as well. LEDs indicate status such as friendly object or enemy object when a target is detected.	LEDs and buttons turn on with in 500ms of expected turn on

Hardware diagram – Figure 1



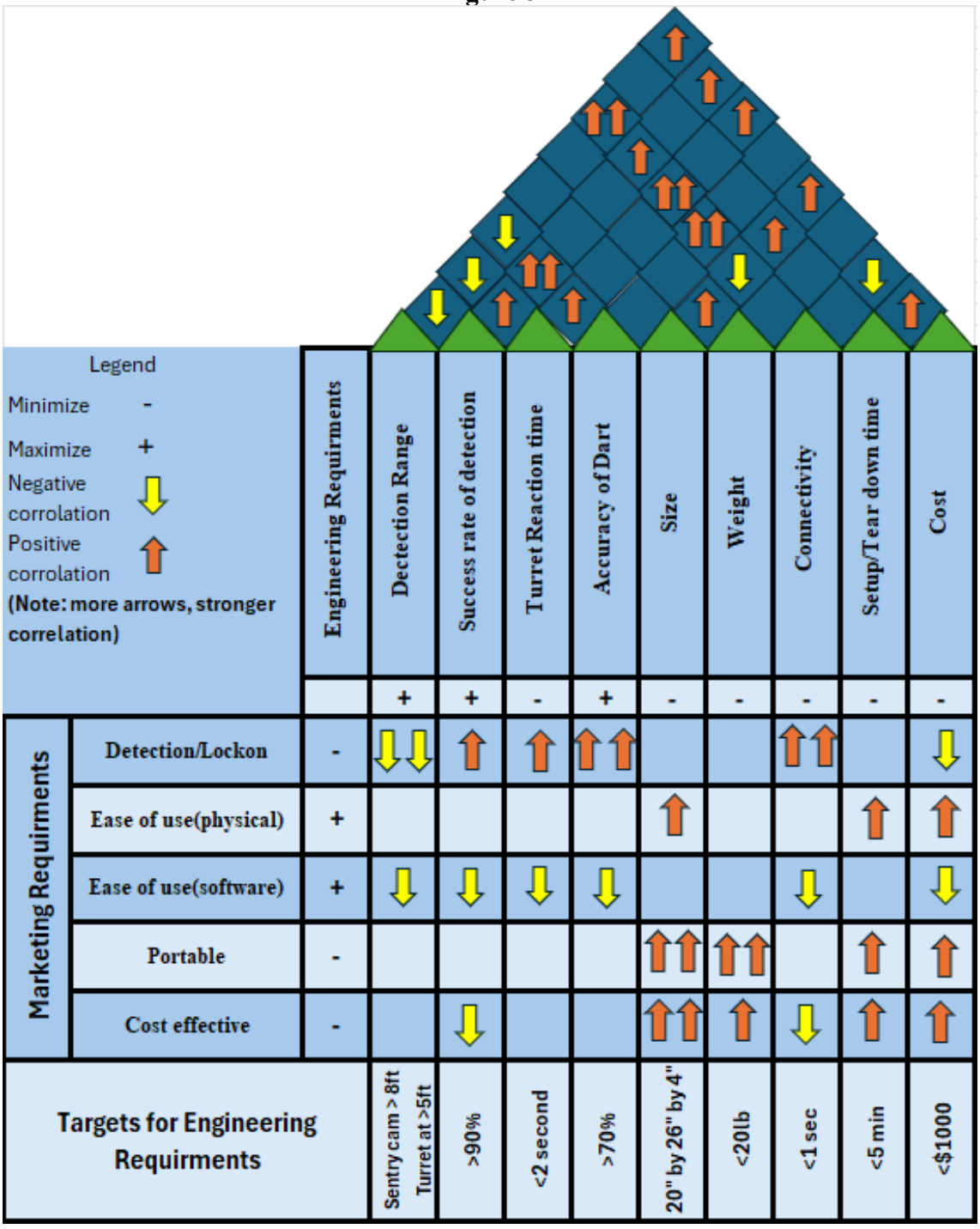
Software diagram – Figure 2



2.5 House of Quality

Below can be found our House of Quality for this project. As we can see there are plenty of interconnecting parts that all depend on each other such as connectivity and detection range of our object. However, cost as we see, can be saved a little from the few changes we have made in this project such as 3D printing when needed.

Figure 3



3. Chapter 3 - Research and Investigation

3.1 - Components

Now we will talk about the types of components we will be using. We will need to deep dive into the types of IMUs we can use, the types and ways to use LEDs, and some buzzer components to list a few. We will start with a short summary of each component, dive into the different types, create a table that compares pros and cons, then end the specific section deciding which part we will use.

3.1.1 - Types of LEDs

LEDs have various uses in many industries. In our case, we will be using LEDs as signals for ourselves based on what is happening in our environment. We will have LEDs for on status of our project, the status of the object in front of us, and lock on status just to name a few. We can use the simple amazon through hole design but in the spirit of this report let's look at a few others to see if there are any benefits to using something other than the most common type of LED. The LED we choose we want to be able to have the access box go around the LED with the LED or some kind of light popping out of the box. We also want to be able to remove the top of the box and fix the PCB inside in case anything breaks, and we want to make sure the LED stays in tack during this and make sure if we hit it at all it won't totally break it. If we do, we need to be able to replace this light easily. Knowing all of this let's look at our options

3.1.1.1 - Through-Hole LEDs with headers

This is the most straightforward. Design the PCB with through holes to solder on female headers to the VCC end and one on the GND end. Then we can put the LED in the correct way on these headers and just make sure when designing the 3D box to make it the correct height so that the LED can be near sticking out of where it is on the board. This is the easiest in case anything breaks as well. The LEDs would not be attached to the 3D printed box in anyway in this configuration so swapping what's needed is much easier and is minimally invasive

3.1.1.2 - Panel-Mount LEDs

These are LEDs that come as a module and are expected to be connected by wires rather than from headers. These have about the same difficulty connection wise to the through holes but can extend farther if we need to increase the height of the access box since these are connected via wires instead of headers. The through holes can be just like this if we solder wires onto those but again, we want a small access box, so we probably don't need these types of LEDs but it's something to hold on to incase the issue arises later

3.1.1.3 - Surface-Mount LEDs + Light Tunnels

This method is perhaps the most interesting of the three. This method can allow for any height of the access box again since the light can travel through these tunnels to reach the top. This is often used in industries today, so this would make our project more realistic. We would buy normal surface mount LEDs such as 150060SS75000, and we would also buy PLP2-350 which is a light tunnel that would take the surface mount LED and project

it through the entire rod. This would allow for us to solder on the LED to the PCB device and assuming we drill the whole directly where we need it, we would then stick this rod in the whole which would allow the light to travel through rod and outside the box enough for the user to see it. This is minimally invasive similar to our first option since this tube just goes over the light and it won't interfere with it at all. The only issue is the precision that will be needed for this step. Our PCB design can change at any point so need to re-drill a hole in our box will make it look unprofessional and messy.

Table 8: LED technology comparison table

LED type	**Mount Method**	**Connection**	**Height Flexibility**	**Ease of Replacement**	**Precision to Implement**	**Cost**
Through - Hole with headers	Female headers soldered to PCB	Pressed on by headers	Fixed height due to header height and LED connection height	Very easy since we just pull out and replace	Low since just need to keep proper distance of headers	$0.10 to $.30 per LED
Panel – Mount LEDs	Wire terminals through panel cutout	Solder wires to PCB pads	Very flexible since we can just leave wire length attached and cut as needed	Easy, just unsolder wires from pads and resolder new part	Medium due to having to drill panel holes which no one on the team has done	$1 to $3 depending on brand/type
Surface mount with light tunnels	SMD soldered to PCB and tunnel inserted in drilled hole on access box	Surface mount soldering	Very flexible since we can just change the tunnel height	Medium – need to know if tunnel or light is the issue first. Tunnel is quick replace, part would require unsoldering and	Harder than rest since hole on access box needs to be exactly over the light on the PCB	$0.50 to $1.50 per LED and tunnel depending on height

				resoldering new part		

3.1.1.4 - Decision on LED technology

We have decided to go with Through hole LEDs that can be attached by headers. This allows for a fixed height, so when designing the access box, we know what height to make it at. If the switches cause any issues with wires, they can be added if needed and won't cause many issues with wire management since they'll be in the access box. This is also the cheapest approach and easiest to change in case anything goes wrong when applying power. PCB layout is easier as well since we just need through holes and solder on headers to make this configuration work.

3.1.2.1 - LED part selection

Something that needs to be considered when selecting the LEDs is the forward voltage drop and the limiting current resistor. LEDs do not operate at a voltage rather can only their brightness is controlled by the rated current that comes in. For example, if an LED is rated at 20mA, and only 10mA comes through it the brightness of the LED will be less than expected. The reverse is true as well, the more current over the rated, the brighter it will be. We need to keep this within a reasonable range; we can't run the LED at a full amp if rated for 20mA rather we can run it at 25mA or 30mA to get a brighter LED if that's within the range of operation for the LED. To solve the resistor needed in front of the LED we can use a simple formula.

R = (Voperating – Vforward)/(Rated current)

So, if we are operating at 3.3V, with a forward voltage of 2V and rated current is 20mA we would end up with a current limiting resistor of 68ohms. Knowing this, let's investigate the types of LEDs we can use.

3.1.2.2 - Kingbright WP7113ID

This first choice of LED is your standard LED type. It runs at 2V and has a forward current of 20mA. It has a 5mm through hole design and costs about $0.20 per part. The part intensity is 200-400mcd which is slightly on the low end but is the most cost effective for our approach

3.1.2.3 - Lite-On LTL-4233

This LED has a forward voltage of 2.1V and has a similar current of 20mA. This is standard for the LEDs we have seen, so this does not affect our power too much when designing. This model is more expensive, around $0.40 per part. Its brightness is where it shines with a range of 2000 to 4000mcd. This is way over what we need but is not a bad idea to have since it still only runs at 20mA and we can adjust the brightness easier if needed.

3.1.2.4 - Vishay TLHG5400

This LED has a forward voltage of 2.2V and still has its forward current of 30mA. The higher current could become an issue when it comes to the power design of our project later on since the motors will take up so much current already. Its intensity is brighter than the Kingbright version of 800 to 1600 mcd. This is more within the range of luminosity we want. It has a through hole footprint of 3mm, and it has an average cost of $0.25 per part. This is a nice model but might be left as a backup if others are not as bright

3.1.2.5 - Cree C503B-RAN

This LED has a forward voltage of 2.1V but has a similar issue to the previous part where rated forward current is at 30mA. It does, however, have the ability to be much brighter than the rest of the models, being at 3000 – 6000mcd. It has a slightly bigger size, being a 5mm through hole. This part is more expensive, with the average cost being $0.60 per part. This part seems to be overkill, but a good premium option to take things up a notch if needed.

3.1.2.6 - Everlight 334-15/T1C1-4WYA

This part is closer to the knightbright in specs. Similar forward voltage and current of 2V and 20mA respectively. Its luminosity is only at 20-40mcd but its viewing angle is 120 degrees whereas the rest were 60 or 30 degrees. It has the bigger size through hole fo 5mm and its cost is around $0.20

Table 9: LED part comparison table

LED Part #	Diameter (mm)	Forward Voltage (V)	Forward Current (A)	Luminous Intensity (mcd)	Viewing Angle	Cost ($)
Kingbright WP7113ID	5	2.0	20	200-400	60	0.15 - 0.25
Lite-On LTL-4233	5	2.1	20	2000-4000	30	0.30 - 0.45
Vishay TLHG5400	3	2.2	30	800-1600	60	0.20 - 0.30

Cree C503B-RAN	5	2.1	30	3000-6000	20	0.50 - 0.70
Everlight 334-15/T1C1-4WYA	5	2.0	20	20-40	120	0.18 - 0.28

3.1.2.7 - LED part decision

We have decided to use the Kingbright WP7113ID due to its average use across all specs. It's a cost-effective part that does what we want at simple ratings that fit well into our design. We can even solder directly on to the PCB if need be but that is just a backup plan if needed. A back up part we want to have is the Vishay TLHG5400 due to its brightness difference between the Kingbright. It does have a higher rated current, so we will need to change our current limiting resistor as well.

3.1.3.1 - Header Parts

Below we will look at two different types of headers, circular or square. The reason we want to have one of each of these or at least look at the options is due to the contact shape of the header. Using a square header with the round LED contact will allow limited contact points compared to using a circular header. The rest of our project can rely on a square header, so selecting a square header for universal use and keeping the circular for anything that may not work covers all of our bases while still looking for a cheap option.

3.1.3.2 - Square header - PPTC021LFBN-RC

This is the first square header we are looking at. It can handle up to 3A at 250V and has an operating temperature of –40 degrees Celsius to 105 degrees Celsius. All of these specs are well within our range of use. It's cheap as well only being about $0.24 per part we get meaning if we use the Kingbright WP7113ID and this header, in total each part is only ~$0.40 which is again well within budget.

3.1.3.3 - Square header - SSQ-102-03-G-S

This is another square header made by Samtec Inc. Its more expensive compared to the previous option being around $0.75 per part meaning LED and header will be ~$1 per part which is a lot more expensive. Its ratings are 3A for current, 655VDC for voltage, and its operating temperature range is –55 degrees Celsius to 125 degrees Celsius. Again, these are well within range however the price is a huge driving factor to not choose this part,

3.1.3.4 - Circular header - SL-102-T-39

This circular header is from Samtec Inc. And has similar specs to the previous square header. Each contact can handle 3A and can handle up to 212VDC with the same temperature range of –55 degrees Celsius to 125 degrees Celsius. Its cost again is higher

than the square headers at closer to $0.90 per contact. Its contact finish is gold, which is part of the reason for the increased price.

3.1.3.5 - Circular Header - 310-13-132-41-001000

This circular header comes with 32 pins to a pack for $7.38. This would give us plenty of contacts for what we need, and we can even replace parts with spare ones we have if something breaks. This is still over what we really need but let's still look at the specs. This gold contact finish header can operate between –55 degrees Celsius to 125 degrees Celsius. It can handle 3A for each contact at a voltage of 150VDC. This would be a good back-up to have due to its wide range of use but 32 pins at one time is a lot to get, especially when the cost of this could go to other components.

Table 10: Header part comparison table

Part #	**Connection Type**	**Pitch(mm)**	**Current Rating(A)**	**Cost ($)**
PPTC021LFBN-RC	Female Pin header	2.54	3	~$0.24
SSQ-102-03-G-S	Female Pin header	2.54	3	$0.70 - $1
SL-102-T-39	Female Pin header(Round)	2.54	3	~$0.92
310-13-132-41-001000	Machine Pin Socket(Round)	2.54	3	~$0.23/per contact

3.1.3.6 - Decision on Header Parts

We have decided to go with the square header PPTC021LFBN-RC due to its cost. This header paired with the LED chosen gives a simple, clean, cheap approach to status indicators for our system. The lower cost allows for the budget to be shifted else wear such as extra PCB space or extra budget for the power bank. If there are not enough contact points for the square header we have decided to solder the LED into the header. This defeats the purpose of the “plug and play” approach we had when designing this approach, but it would still be easier to unsolder the LED from a header rather than from a board. The 310-13-132-41-001000 is going to be cheaper per pin but again this locks us into having to buy 32. If we order the PPTC021LFBN-RC we would only need to order what was needed which probably would not be more hen 10 to 15 which would only cost ~$4 instead of $7.38

3.1.4 - Buzzer components

For some extra peripherals, we have a buzzer component. We want this to trigger when the turret has locked onto an enemy target and is ready to fire. We will have some LEDs

that indicate similar status, but we wanted an extra component to help with this as well. There are a few different types of buzzers so let's look at each in some details

3.1.4.1 - PCB-Mounted Passive Buzzer

This buzzer component is your basic component with a few functions. It requires a PWM signal from our MCU to operate and has different levels of sounds when buzzing. The passive buzzer is soldered directly to the board and is small, allowing room to be freed up for more complex components and parts. It can generate different tones as well so this would allow us to use the buzzer more for different reasons. For example, we can have a loud sound for when we are locked on and a lower sound for when we are tracking. The issue with this is in general the sound isn't as high as an Active buzzer but also the use of a PWM signal. This could complicate our design both software and hardware.

3.1.4.2 - PCB-Mounted Active Buzzer

An Active Buzzer is simpler compared to a passive buzzer. This component will just need a signal from the MCU that triggers for high and low signals and uses a pull up resistor in parallel to help with current for when it's not in use. It does not have varying sounds like the Passive, but it has a single loud buzzer sound. This component can also be soldered directly on the board and will have a much easier configuration for design as well compared to the Passive.

3.1.4.3 - Panel-Mount Buzzer

These buzzers are an extended version of the Active buzzers. These have wires connected to them and instead of being directly soldered onto the board, they are instead mounted onto the outside of the box. This would be useful if we wanted to see the buzzer; however, this causes a problem with being able to replace parts. We would have to have this part with female headers so we can attach and detach this part from the box whenever we change the PCB if it's mounted onto the box. This could cause more chances of wiring issues arising and messing with traces, which is what we are trying to avoid.

Table 11: Buzzer technology comparison table

Buzzer Type	**Signal Required**	**Tone Flexibility**	**Volume**	**Mounting**	**PCB Complexity**	**Replacement**
PCB-Mounted Passive Buzzer	PWM from MCU	Multiple tones possible	moderate	Soldered directly to PCB	Higher difficulty due to PWM configuration needed	Medium since we need to desolder part from board
PCB – Mounted Active Buzzer	Simple High/Low signal	Fixed signal tone only	Loud	Soldered directly on to PCB	Simple since just GPIO from MCU and pull up resistor	Medium since we need to desolder part from board

	from MCU					
Panel – Mount Buzzer	Simple High/Low signal from MCU	Fixed signal tone only	Very Loud	Mounted to enclosure panel with wires	Medium - GPIO pin but extra difficulty with wire management	Can use headers to make process easier

3.1.4.4 - Decision on Buzzer Technology

We have decided to go with the PCB – Mounted active buzzer. This buzzer only has one tone, but we plan on using LEDs as the main status indicator and plan on using this buzzer for a specific indicator such as lock on to target. Also choosing this allows for minimal wiring issues as addressed in the Panel – Mount buzzer and the Passive buzzer as said has a harder signal to implement into the PCB where the Active is just a pull up resistor and a GPIO pin from an MCU.

3.1.5 - Buzzer Part Selection

For each buzzer we look at, something to consider is the recommended circuit in the datasheet. These components need to be readily available and easy to implement. If the circuit takes up too much space, doesn't use common parts, or in general it is too hard to implement, we can't move forward with this part.

3.1.5.1 - RS Pro 1710866

This through hole active buzzer has a rated voltage of 12V which is much higher than what we wanted to run this buzzer at. It does, however, have an operating voltage range of 3VDC to 24VDC. It has a rated current of 13mA and outputs 77dB. Its resonant Frequency is ~2200Hz. This part is priced at around $1.44 per part. Running at the 12V would not cause much of an issue since it's only 13mA and due to the small current, we could just run this buzzer through holes in the PCB with no extra circuit to help with the incorporation of this part into our system. We can just solder this to a connecting GPIO pin on the ESP32 and a pull up resistor just in case of any issues that arise.

3.1.5.2 - TDK SD1209T5-A1

This buzzer runs at a rated voltage we expected of 5V. It can draw up to 65mA which means we need to consider the circuit that we would need to implement this component. It has a nice sound of 80dB meaning the sound should be clear from a respectable distance. It has a frequency of 2.048kHz

3.1.5.3 - CEM-1206S

This buzzer runs at a cheaper price of around $0.85 per unit. This is good, but this will have additional cost due to it being a magnetic, active buzzer. We need to implement its recommended schematic due to the coil in the buzzer. Its current draw is about 45mA at

its rated voltage of 5V at a frequency of 2.4kHz. Its schematic only requires a MOSFET transistor on the back end of the buzzer, and a flywheel diode parallel to the terminals on the buzzer. All parts are easily accessible and have a CAD footprint for easy installation onto a PCB. It has a higher pitches sound at 85dB, allowing for the sound to travel further then both parts mentioned above.

3.1.5.4 - AI-1223-TWT-5V-R

This is similar to the CEM-1206S with a few differences. Its current draw is less at the same rated voltage being less than 30mA. This could help with designing the schematic meaning we may not have to buy extra parts to implement the buzzer. It does also have a similar output sound of around 85dB. This model is very widely used in embedded designs and hobbyist, so reviews are easy to come by. It still does have CAD footprints allowing for easy PCB work, but this is a very expensive product at $3.16 per unit. Even with the extra cost for extra parts in CEM-1206S, the total for that would still be cheaper than this option.

Table 12: Buzzer Part comparison table

Part #	Operating Voltage(V)	Current Draw(mA)	Sound(dB)
RS Pro 1710866	12V	~13	~77
TDK SD1209T5-A1	5V	~65	~80
CEM-1206S	5V	~45	~85
AI-1223-TWT-5V-R	5V	~30	~85

3.1.5.5 - Decision on Buzzer Part

The decision for this part was easy, and we will be going with the CEM-1206S. It does require extra work in the schematics but based on reviews and ease of use in systems we have gone with this model. It has the operating voltage we want of 5V, a small current draw and effective cost for our system. Its sound is also among the higher ones we have seen based off research which we want since we plan on using this buzzer for mission critical status such as lock on to target or enemy identification.

3.1.6 IMU Choices

The IMU will be used for stabilization of the turret movement in your board. Due to our turret needing to be able to move on the x and y axis constantly and wanting to have a consistent straight shot we will need to stabilize our system. The IMU allows for the turret to operate fast but stable to the signals that sentry cameras and the Huskeylens data that will be sent via UART, I2C, or SPI. We want to also have an IMU that can be used across multiple processors and communicate in various ways to allow flexibility when it comes to design changes that might need to happen down the line in cases something does not go our way. Having this component also be cheap can help push budget to other places such as better motors or servos or high-quality cameras.

3.1.6.1 - IMU with 6DoF

The main difference between IMUs is the degree of freedom. A degree of freedom (DoF) is an axis that the IMU is sensing. A 6 DoF IMU operates on an accelerometer and a gyroscope. The 3 – axis accelerometer is there to measure the linear acceleration of the turret, and the 3 – axis gyroscope is there to measure the angular velocity of the turret. Both will be needed to allow for targeting objects in front of it no matter the size of the object. These are going to have cheaper options compared to the rest but also not be as accurate and still have some noise in the system. It will consume less power, meaning less current will be needed in our system.

3.1.6.2 - IMU with 9DoF

If we want to make our station mobile the use of 9 DoF would need to be considered. This added 3 – axis magnetometer which acts like a compass for our system. This type of IMU is generally more used in robotics and drone projects due to having easier control sets of a set north is declared ahead of time. Since our project is stationary, we do not need an IMU with 9 DoF but if we were to expand on the system and make it mobile this would be something to consider

3.1.6.3 - Industrial grade IMU

These types of IMU can range in DoF and can have six or nine. Generally these will be more expensive but also consume less power. The increased accuracy is also a nice touch and will be needed if we want to reach our stretch goal of hitting aerial targets.

Table 13: IMU technology comparison table

Option	Features	Pros	Cons
6 DoF IMU	Accel and Gyro	Simpler to implement and cheap	No magnetometer
9 DoF IMU	Accel, Gyro and magnetometer	Better orientation accuracy when system is moving	Can drift with EMI and higher cost
Industrial IMU	Low noise and better stability	Extremely accurate	Expensive and power hungry

3.1.6.4 - Decision of IMU tech

Based on this table, we have decided to go with the 6 DoF IMU. We want to have an accurate stationary system, so we do not need a 9 DoF IMU and we do not want to spend $100 on the IMU alone. The 6 DoF has what we need to stabilize the movement of the turret during location of the target, and we will make sure to choose a cheap option that has good accuracy and lower power consumption to help with our system.

3.1.7 The basic choice – MPU6050

This IMU is as it is listed, basic but efficient. This IMU model operates at 3.3V - 5V and has a default communication of I2C but can also use SPI in some breakout cases. It has six DoF(Degrees of Freedom) with three for gyroscope and three for accelerometer. We can find this cheaply online on Amazon.com, so accessibility is not hard either. An issue that does pop up is due to this model being an older one; many in the industry say this is obsolete and should not be used.

3.1.7.2 Upgraded choice – ISM330DHCX

This IMU has a lot of what we need at a higher price. This IMU operates with 3.3V to 5V, has one of the smaller current ratings of any of the systems talked about above, has I2C and SPI communication, and has better degree ratings depending on the type of board used. This model is used much more in systems today due to the improved accuracy it allows in systems as well as having lower noise which could help with motor or servo issues we could run into later. The only issue as said above is that one of these boards could cost as much as three MPU6050. Taking care of this IMU will have to be high on the priority list to help with budget costs.

3.1.7.3 Expensive but worth it – LSM6DS3

This IMU is another great option for better power management of our system. It has similar features to the previous IMUs with 6 DoF and running in a voltage range of 1.71V - 3.6V. It has two communication protocols of I2C and SPI allowing us to communicate with multiple components at once. It does include motion detection features which correlates with our project very well. It has a lower current draw of around .65mA and even has low power mode. The issue with this type of IMU is it is not precise as other IMUs which in this project is a huge issue

Table 14: IMU part comparison table

IMU	Pros	Cons	Voltage range	Power usage	Interface	Cost
MPU – 6050	Cheap and available	Older and has higher drift then regular IMUs	2.3V-3.4V	Low	I2C, some SPI	About $6 per unit
ISM330DHCX	High stability, low noise	Expensive	1.71-3.6V	Very Low	I2C, SPI	$20 per unit

LSM6DS3	Power efficient, widely used	Not as precise as others	1.71-3.6V	Very Low	I2C, SPI	About $10 per unit

3.1.7.4 - Decision of IMU

We have decided to go with the ISM330DHCX due to its flexible voltage, low power usage, and most of all its higher accuracy. We will be paying the price for this from one unit being $20. However, we feel lots of the benefits will pay off. With the use of I2C communication, we can save this portion to communicate with the ESP-32 and allows for simpler wiring when making the PCB. The SPI communication can be saved for communication with higher difficulty systems such as the Huskeylens we have on the turret.

3.1.8 - Flywheel Motors

The flywheel pair sets response time and shot energy. They need to spin up fast, hold speed during the dart pinch, and behave predictably on a 12V wall supply. The performance of these motors directly affects projectile velocity, response time, and overall system reliability.

3.1.8.1 - Brushed DC Motors

Brushed DC motors wear quickly and create EMI noise at high rpm. The brushes physically rub against the commutator, which generates both mechanical wear and electrical arcing. At the speeds we need of 10,000+ rpm, this is becoming a serious problem. The arcing creates broadband electrical noise that can interfere with the microcontroller, sensors, and PWM signals. We'd end up needing more filtering and shielding, which adds complexity. The brushes also need periodic replacement, and brush dust can contaminate other components. While these motors are cheap and easy to control initially, the long-term maintenance and EMI issues make them less suitable for our application.

3.1.8.2 - Stepper Motors

Stepper motors are inefficient and lose torque at the 10k to 15k rpm range we need. Stepper motors are designed for precise positioning at low speeds, not sustained high speed rotation. As RPM increases, their torque drops rapidly due to inductance limiting how fast current can change in the windings. To get adequate torque at high RPM, you'd need a much larger stepper and a more sophisticated driver, which defeats the purpose. Steppers also have resonance issues at certain speeds where they vibrate badly or even skip steps. The continuous current draws even when holding still also creates unnecessary heat and power consumption.

3.1.8.3 - BLDC Outrunner Motors

BLDC outrunners are efficient, lightweight, and work with off the shelf fixed wing ESCs. The outrunner design puts permanent magnets on the outside rotor, which gives you more

torque for a given diameter compared to in-runners. The larger rotor diameter means higher magnetic flux and better cooling since the magnets are exposed to airflow. Fixed wing ESCs are specifically designed for this type of motor in propeller applications, which is mechanically similar to our flywheel use case.

Mechanically, the 2312 frame fits our mounts, accepts a stiff 4mm adapter, and has enough thermal mass for repeated ramps. The 2312 designation refers to a 23mm stator diameter and 12mm stator height, which is a common size in the hobby market. This means parts are readily available and well documented. The thermal mass matters because each firing cycle heats the motor, and with repeated demos we need the motor to dissipate that heat between shots without overheating.

Table 15: Flywheel Motor Technology Comparison Table

Motor Type	Maintenance	EMI Noise	Torque at High RPM	Control Complexity	Thermal Performance	Cost
Brushed DC	High, brush replacement needed	High, creates interference	Moderate	Low, simple DC control	Poor, brushes generate heat	Low
Stepper	Low maintenance	Low	Poor, drops rapidly above 1000 RPM	High, needs specialized driver	Moderate, continuous current	Medium
BLDC Outrunner	Very low, no brushes	Very low	Excellent at 10-15 krpm	Medium, needs ESC	Excellent, good cooling	Medium

3.1.8.4 - Decision on Flywheel Motor Technology

We have decided to go with BLDC outrunner motors in the 2312 frame size. Within that frame size, KV rating determines the rpm per amp behavior on 12V. We compared 980, 1250, and 1400 KV options. The 1250 KV hits the sweet spot: about 10,500 to 12,000 rpm under load on 12V with quick ramps and manageable current draw. KV is the motor's velocity constant, expressed as RPM per volt with no load. A 1250 KV motor spins at 15,000 rpm unloaded on 12V (1250 × 12), but under the load of spinning a flywheel and launching darts, it slows to around 10,500 to 12,000 rpm, which is right in our target range for adequate rim speed without excessive noise or current.

Table 16: KV Rating Comparison Table

KV Rating	No-Load RPM @12V	Expected Loaded RPM	Torque per Amp	Spin-Up Current	Noise @ Performance
980	11,760	8,200–9,400	High	Low	Low
1250	15,000	10,500–12,000	Medium	Medium	Medium
1400	16,800	11,760–13,440	Lower	Higher	Higher

Table 17: Flywheel Motor Part Selection

Type	Typical Mass	Expected Dart Speed	No-load Current	Continuous/Burst	Why Pick It
Mid-tier 2312–1250 KV	~52–55g		~1–1.5A	18–20A / 28–32A	Good bearings/laminations; 4mm shaft for adapters
Budget 2312–1250 KV	~50–52g		~0.8–1.2A	15–17A / 25–28A	Works if de-rated; more unit-to-unit spread
Premium 2312–1250 KV	~55–58g		~1–1.6A	20–22A / 32–35A	Highest consistency;

For each motor we look at, something to consider is the bearing quality and shaft runout specifications in the datasheet or product reviews. These components need to have minimal vibration and good bearing life since they'll be spinning at high speed during every demonstration. If the motor has excessive vibration or the bearings are known to fail quickly and can provide inconsistencies in the dart accuracy.

3.1.9 - Flywheel Diameter Selection

Diameter sets rim speed for a given rpm, the inertia the motors must spin, and the contact path during the dart pinch. It governs response time, noise, and shot consistency. The wheel diameter is one of the most important mechanical parameters that affects both performance and system behavior.

3.1.9.1 - 65mm Diameter Wheels

The 65mm wheels are the snappiest option but need higher rpm for adequate rim speed. With a smaller diameter, you need to spin faster to achieve the same rim speed, which is what actually matters for dart velocity. Higher RPM means more motor noise, more aerodynamic noise from the spinning wheels, and more sensitive control since small RPM changes translate to bigger velocity changes. The contact time between the dart and the wheels is also shorter with a smaller diameter, which can reduce shot consistency if the dart isn't perfectly centered. The main advantage is the fastest spin up time and lowest inertia, which reduces the load on the power supply during startup.

3.1.9.2 - 70mm Diameter Wheels

The 70mm wheels thread the needle between the 65mm and 80mm options. At 11,800 rpm (our target firing speed), 70mm wheels give us about 43 m/s rim speed, which translates to dart velocities in the safe range for foam projectiles. The inertia is manageable enough that spin up takes under a second, but high enough that the wheels don't bog down noticeably when launching a dart. Testing will tell us if we need to adjust, but 70mm gives us room to move in either direction if the performance isn't quite right. The balanced approach makes this the safest choice for initial testing.

3.1.9.3 - 80mm Diameter Wheels

The 80mm wheels run at lower rpm for the same rim speed and have steadier shots. The larger diameter gives you more contact area and time with the dart, which improves consistency. The flywheels act like energy storage, and larger diameter wheels store more rotational energy for a given RPM. This makes the velocity more stable shot to shot. However, the increased moment of inertia means the motors take longer to accelerate the wheels to speed, and the current draw during acceleration lasts longer. This creates broader pulses on the power supply, which our staged startup choreography has to accommodate.

Table 18: Flywheel Diameter Comparison Table

Diameter	Rim Speed @ 9,800 rpm	Rim Speed @ 11,800 rpm	Relative Inertia	Spin-up Time	Shot Consistency	PSU Transient
65mm	33.40 m/s	40.16 m/s	0.862×	Fastest	Fair (short contact)	Lowest
70mm	35.92 m/s	43.25 m/s	1.000×	Fast	Good	Low
80mm	41.05 m/s	49.43 m/s	1.306×	Slowest	Best	Higher

3.1.9.5 Decision on Flywheel Diameter

We have decided to go with 70mm diameter wheels. The 70mm option gives us adequate rim speed without requiring excessive RPM, keeps spin up time reasonable, and provides good shot consistency without the heavy current draw of 80mm wheels.

3.1.10 - ESC Controllers

Each ESC communicates its BLDC flywheel and controls the ramp, timing, and braking. This directly determines how kind the system is to the 12V wall bus. The ESC selection is critical for managing the power supply choreography and preventing voltage sag during motor startup.

3.1.10.1 - Multirotor ESCs

Multirotor ESCs are battery focused and often lack a gentle soft start. They're designed for quadcopters where you have a large battery that can source hundreds of amps without voltage sag. The control algorithms assume power sources and use aggressive throttle responses for acrobatic flights. Many multirotor ESCs also implement active braking to stop props quickly when throttle is cut, which dumps regenerative current back into the supply. That's fine with a battery that can absorb it, but a wall supply can't handle reverse current and may fault or oscillate. These ESCs would require significant configuration changes to work properly with our wall powered system.

3.1.10.2 - Car ESCs

Car ESCs add bulk and have wheel tuned punch curves that don't help us. They're designed for RC cars where you need high torque at low speeds for acceleration from a standstill, which is a completely different requirement than spinning a flywheel at constant high speed. The throttle curves are nonlinear to give good low speed control, which makes our application harder to tune. Car ESCs are also physically large because they have to handle the shock and vibration of an RC vehicle, and the extra size creates packaging problems in our turret housing.

3.1.10.3 - Fixed Wing ESCs

Fixed wing ESCs provide forward only operation, soft start, brake off capability, adjustable timing, and active freewheel to reduce ripple. Fixed wing aircraft don't need reverse thrust, so these ESCs are simpler and more power supply friendly. Soft start means the ESC gradually increases current during the first few hundred milliseconds of throttle application instead of hitting the motor with full current immediately. This is exactly what we need for staged startup.

The brake off setting prevents regenerative braking, so when throttle is reduced, the motor just coasts instead of trying to push current back. Active freewheel (sometimes called damped light) synchronously rectifies the motor's back EMF instead of letting it ring, which reduces electrical noise on the power rail. Two identical ESCs get programmed the same way, and the endpoint is calibrated once. Using identical units means we only have to figure out the optimal settings once and then copy them to the second ESC.

Table 19: ESC Technology Comparison Table

ESC Type	Soft Start	Brake Behavior	Throttle Curve	Size	Power Source Assumption
Multirotor	Usually no	Active braking (regenerative)	Aggressive, linear	Compact	Large battery
Car	Variable	Active braking	Nonlinear, low speed optimized	Large	Battery
Fixed Wing	Yes, standard feature	Configurable, can disable	Linear, forward only	Compact	Battery but adaptable

3.1.10.4 - Decision on ESC Technology

We have decided to go with Fixed Wing ESCs. These provide the soft start and brake off features we need for power supply friendly operation. Endpoint calibration teaches the ESC what PWM pulse widths correspond to minimum and maximum throttle, ensuring both motors respond identically to the same control signal. Without this calibration, manufacturing tolerances in the ESC's input circuitry could cause the motors to spin at slightly different speeds for the same command, which would make darts curve in flight.

Table 20: ESC Part Selection

ESC Model	Continuous / Burst	BEC	Notes
Hobbywing Skywalker V2 50A	50A / ~70A	5V SBEC	Selected: Soft-Start, Medium timing, active freewheel ON
ZTW Mantis G2 45A	45A / 55A	5.6/7.4V @ ~4A	32-bit, adjustable BEC; compact alternate
Castle Talon 60	60A cont.	8A cont / 20A peak	Premium tuning; larger/heavier (overkill for v1)

For each ESC we look at, something to consider is the programming options and whether they support brake off and soft start modes. These components need to have clear firmware configuration options documented in the manual.

3.1.11 - Gear Ratio & Transmission

The transmission sets slew speed, holding stiffness, and pointing resolution at the turret. The ratio trades speed for torque and finer control, so the turret snaps to target and then holds steady. The gear selection directly affects how quickly we can acquire targets and how accurately we can hold aim during firing.

3.1.11.1 - Direct Drive

Direct drive is simplest but offers no leverage. If you connect the servo output directly to the turret, you get one to one motion. The servo's torque and speed are what you get at the turret. For our application, the servos don't have enough torque to hold the launcher steady against firing impulses and wind up when moving, so direct drive isn't viable. The lack of mechanical advantage means the servo would have to work much harder, drawing more current and potentially overheating during extended operation.

3.1.11.2 - Timing Belts

Timing belts are quiet but add compliance and tension complexity. Belt drives can work well, but the belt itself stretches slightly under load, which creates backlash and reduces pointing accuracy. You also need tensioners and proper alignment of the pulleys, and belts can slip if tension isn't maintained. The belt teeth can also wear over time, increasing backlash. For a demonstration system where we want reliable, repeatable performance without constant adjustment, belts add more complexity than they're worth. The tension system would also add weight and take up space in the turret housing.

3.1.11.3 - Spur Gears

Spur gears are compact, stiff, and provide exact ratios using COTS acetal or steel gear sets. Spur gears mesh directly so there's minimal compliance, and they don't require tensioning. Commercial off the shelf gears are available in standard pressure angles and module sizes, making it easy to get replacement parts or change ratios. Acetal gears are self-lubricating and quiet, while steel gears handle higher loads.

For the tilt axis, we're using 30T:90T (3:1) gearing. The tilt only needs to cover about 90 degrees of travel, so we can sacrifice some of the servo range to gain torque and resolution. The 3:1 ratio triples the servo's torque, giving us enough holding force to resist gravity and firing impulses. It also divides the servo step resolution by 3, so small PWM changes translate to finer angular adjustments. This is particularly important for tilting since the launcher needs to hold steady against the constant downward pull of gravity. Without the gearing, the servo would buzz and hunt trying to maintain elevation angle.

For the pan axis, we're using 1:1 gearing (direct drive through gears for mechanical coupling). Pan needs to cover a wide horizontal sweep to track targets across the full field of view. The SF3218MG servo provides 270° of travel, and we want to use all of that range for panning. Since the pan axis doesn't fight gravity and the horizontal inertia is manageable, we don't need the torque multiplication that we need for the tilt. The 1:1 ratio preserves the servo's full speed and range while still providing a solid mechanical connection.

Table 21: Gear Transmission Technology Comparison Table

Transmission Type	Stiffness	Backlash	Tensioning Required	Maintenance	Ratio Options	Size
Direct Drive	Maximum	None (servo only)	None	None	1:1 fixed	Minimal
Timing Belt	Low (belt stretch)	Moderate	Yes, critical	Belt wear, tension checks	Flexible with pulley sizes	Medium
Spur Gears	High	Low (gear only)	None	Minimal	Discrete ratios	Compact

3.1.11.4 Decision on Gear Transmission Technology

We have decided to go with spur gears for both axes, but with different ratios. The tilt axis uses 30T:90T (3:1) to provide torque multiplication for holding against gravity. The pan axis uses 1:1 gearing to preserve the servo's full 270° range for maximum coverage. Going to 5:1 on tilt would give more torque and resolution but would make fine adjustments unnecessarily slow.

Table 22: Gear Ratio Part Selection

Gearset	Ratio	Output Stall estimated	No-load Output Speed estimated	Why It Fits
30T:60T	2:1	~4.4 N·m	~188°/s	Fastest slews; least stiffness gain; for very light heads
30T:90T	3:1	~6.6 N·m	~125°/s	Selected: balanced stiffness, finer resolution, moderate speed
24T:120T	5:1	~11.0 N·m	~75°/s	Steadiest aim; slower sweeps and higher inertia

3.1.12 - Pan Motor Selection

The pan motor determines how quickly we acquire a target, how still we hold heading while flywheels spin, and how quickly we settle between shots. The pan axis carries more load than tilt since it has to move the entire turret head assembly.

3.1.12.1 - Stepper Motors (NEMA-17)

Stepper motors with NEMA-17 drivers offer precise steps and high hold torque at zero speed, but torque drops with speed and drivers add complexity and resonance. Steppers are great when you need precise positioning and can move slowly, but they have issues at the speeds we need. The torque curve of a stepper drops significantly above a few hundred RPM, and we need faster motion than that for quick target acquisition.

Steppers also draw current continuously even when holding still, which wastes power and generates heat. The drivers can introduce mid band resonance where the motor vibrates badly at certain speeds, and you need tuning or dampers to fix it. More importantly, steppers are open loop, so if the motor skips steps due to a sudden load, you lose position and have no way to know it happened without adding an encoder.

3.1.12.2 - PWM Servos

PWM servos integrate motor, gearbox, and controller and accept simple PWM commands. A hobby servo has everything built in: DC motor, gear reduction, position sensor, and control electronics. You send it a PWM signal to set the desired position, and it handles the rest. The internal controller maintains position against external loads and provides damping to prevent oscillation.

This makes the system much simpler from a firmware and electronics perspective. You just need a PWM output from the microcontroller, no H-bridge, no encoder interface. Pairing with 1:1 gearing for pan gives us the full 270° range while the servo's internal gearbox provides adequate stiffness and torque for the horizontal axis. The servo itself has some backlash in its internal gearbox (typically under a degree), which is acceptable for our demonstration of accuracy requirements.

Table 23: Pan Motor Technology Comparison Table

Motor Type	Control Method	Position Feedback	Torque at Speed	Current Draw at Rest	Setup Complexity	Firmware Complexity	Backlash
Stepper	Open loop steps	None (unless encoder added)	Poor above 300 RPM	High (continuous)	Medium (driver needed)	Low to Medium	None intrinsically
PWM Hobby Servo	Integrated closed loop	Internal potentiometer	Good	Low (only when moving)	Very Low (just PWM)	Very Low (position command)	Low (internal gears)

3.1.12.3 - Decision on Pan Motor Technology

We have decided to go with PWM hobby servos paired with 3:1 spur gears. The gears also multiply the servo's holding torque, allowing it to resist external disturbances better

than it could alone. This combination gives us the simplicity of servo control with the performance we need for stable aiming.

Table 24: Pan Motor Part Selection

Model	Voltage	Speed	Stall Torque	Mass	Notes
SunFounder SF3218MG	4.8–7.2V	0.18→0.14 s/60°	20.5→22.8 kg·cm	~56g	Selected: metal gears, 270° range
Savox SC-1256TG	4.8–6.0V	~0.15 s/60° @ 6V	~20 kg·cm	~52–60g	Proven mid-tier if standardizing at 6V
JX PDI-6221MG	4.8–6.6V	~0.16 s/60° @ 6V	~20 kg·cm	~62–65g	Cost-effective alternative (shorter range)

For each servo we look at, something to consider is the gear material (metal vs plastic) and the feedback of potentiometer quality. These components need to have metal gears for durability under repeated operation and minimal dead band in the potentiometer. If the servo has plastic gears or excessive dead band, we can't move forward with that part.

3.1.13 - Tilt Motor Selection

The tilt motor sets an elevation angle. The tilt axis actually carries heavier loads than pan since it has to support the launcher assembly against gravity. Precision matters - we need crisp moves, quick settling for the vision loop, and steady holding against gravity and firing transients.

3.1.13.1 - Why Use the Same Servo

As with pan, we're avoiding the standstill current and resonance issues of steppers and the tuning plus backlash complexity of DC gearmotors. Servos keep electronics simple and hold position in closed loop. Using the same SF3218MG simplifies parts, wiring, and spares. Having identical motors for both axes means we only need to stock one type of spare, and if we damage a motor during testing or demos, we can swap it with the other axis temporarily to keep the system running.

The control code is also simpler when both motors have identical response characteristics. We don't have to write separate PWM timing or filtering for each axis. The tilt axis actually has heavier loads than pan since it has to support the weight of the launcher assembly and fight gravity constantly. The pan axis just rotates the turret horizontally without having to lift anything.

The tilt axis has to fight gravity, which creates a constant load that varies with elevation angle. When the launcher is pointed horizontally, there's maximum gravitational torque

trying to droop the assembly downward. The servo has to continuously provide holding torque to resist this, and if the servo isn't strong enough or the internal gearbox has too much backlash, the aim point will sag. This is why we use 3:1 gearing on tilt but only 1:1 on pan. Using the same SF3218MG servo for both ensures we have plenty of margin, and the 3:1 external gearing on tilt multiplies the servo's torque, providing approximately 60 to 68 kg·cm at the output, which is more than adequate for gravity compensation.

Table 25: Tilt Motor Part Selection

Model	Voltage	Speed	Stall Torque	Mass	Notes
SunFounder SF3218MG	4.8–7.2V	0.18→0.14 s/60°	20.5→22.8 kg·cm	~56g	Selected for tilt too; consistent pairing
Savox SC-1256TG	4.8–6.0V	~0.15 s/60° @ 6V	~20 kg·cm	~52–60g	Lighter mid-tier option
Hitec D955TW	4.8–7.4V	~0.15 s/60° @ 7.4V	~25–29 kg·cm	~66g	Premium margin; heavier/costlier

For each servo considered for tilt, the same criteria apply as for pan: metal gears for durability and good potentiometer quality for minimal dead band and position feedback accuracy.

3.1.14 - Filament Selection for 3D Printed Components

Printed parts include the flywheel cage, motor and servo mounts, camera brackets, covers, and cable guides. Material choice governs impact toughness, heat margin near motors and ESCs, and print risk like warping.

3.1.14.1 - PLA (Polylactic Acid)

PLA offers crisp prints but is brittle and softens around 55 to 60°C. PLA is the easiest filament to print with and gives an excellent surface finish and dimensional accuracy on the first try. It doesn't warp or require an enclosure, and it sticks well to most print surfaces. This makes it ideal for prototyping where you're iterating designs and want fast turnaround.

However, PLA's low glass transition temperature means parts will deform if they get warm. Near the ESCs or motors, ambient temperature can easily reach 50 to 60°C during operation, which is enough to soften PLA and cause brackets or mounts to sag. PLA is also brittle compared to other filaments. It can crack under impact or sustained stress, especially if the part has thin sections or sharp internal corners. This makes PLA unsuitable for structural components that experience loads or heat.

3.1.14.2 - ABS (Acrylonitrile Butadiene Styrene)

ABS has better heat resistance (95 to 105°C) but warps badly and needs an enclosure. ABS can handle the temperatures around motors and electronics without softening, and it's tougher than PLA under impact. However, ABS shrinks significantly as it cools, which causes warping during printing. Large flat parts will lift at the corners, and tall parts can crack as internal stresses build up.

To print ABS successfully, you need a heated enclosure to keep the part warm during printing, which we don't have. The fumes from ABS printing are also unpleasant and potentially hazardous, requiring ventilation. Without proper equipment, ABS prints fail more often than they succeed, which wastes time and material during development.

3.1.14.3 - PETG (Polyethylene Terephthalate Glycol)

PETG is tough with about 75 to 85°C usable heat tolerance and low warp. It's ideal for functional, accurate parts on open printers. PETG bridges the gap between PLA and ABS. It prints almost as easily as PLA on an open printer but has much better mechanical properties. The heat resistance is adequate for our application since we're not expecting sustained temperatures above 70°C anywhere in the system.

PETG has excellent layer adhesion, which makes parts stronger and less likely to delaminate under stress. It's also more ductile than PLA, so it bends before breaking, which is better for parts that might get bumped or dropped during handling. The main downside of PETG is stringing, where thin threads of plastic get dragged between features during printing. This can be minimized with proper retraction settings, and the strings are easy to clean off after printing.

3.1.14.4 - Nylon

Nylon is excellent but hygroscopic and not needed for this build. Nylon has outstanding wear resistance and flexibility, which makes it ideal for parts that slide against each other or need to flex repeatedly. However, nylon absorbs moisture from the air rapidly, and wet filament produces weak, porous parts with poor surface finish. You have to dry the filament in a heated chamber before printing and ideally keep it dry during printing with a dry box.

Nylon also requires higher print temperatures (around 250 to 270°C) and sticks poorly to standard print surfaces. For this project, we don't have any parts that really need nylon's properties like gears or bushings, so the extra hassle isn't justified.

Table 26: Filament Technology Comparison Table

Filament Type	Heat Resistance	Impact Toughness	Warping Risk	Print Difficulty	Layer Adhesion	Dimensional Accuracy	Cost

PLA	Poor (55-60°C)	Low, brittle	Very low	Very easy	Good	Excellent	Low
ABS	Good (95-105°C)	Good	High	Hard, needs enclosure	Moderate	Good	Low to Medium
PETG	Moderate (75-85°C)	Excellent, ductile	Low	Easy on open printer	Excellent	Good	Medium
Nylon	Excellent (90-100°C)	Excellent, flexible	Moderate	Hard, hygroscopic	Excellent	Moderate	High

3.1.14.5 - Decision on Filament Technology

We have decided to go with PETG as the primary filament for structural components. PLA will remain useful for prototyping and non load bearing cosmetic parts. ABS is off the table because the printing challenges aren't worth it for this project, especially since we don't have a reliable enclosure setup.

3.1.15 - Laser Aim Indicator

The laser is a visual alignment aid for labs and demos. It shows the aim point at about 10ft indoors, improving operator and audience understanding. It doesn't measure distance, just provides a visible dot where the turret is aimed.

3.1.15.1 - Red Laser Modules

Red lasers at 650nm wavelength are the cheapest and simplest option. They use basic laser diodes that are widely available and well understood. However, the human eye is much less sensitive to red light compared to green light at the same optical power. A 1mW red laser appears significantly dimmer than a 1mW green laser in typical room lighting. However, since our demonstrations will primarily occur in indoor lab environments with controlled lighting, visibility remains adequate at short distances. A small increase in drive current or beam focus adjustment easily compensates for this reduced visibility.

3.1.15.2 - Green Laser Modules (Class 2)

Green lasers at 520 to 532nm wavelength in Class 2 provide clearly visible dots while staying in the safest laser class. The human eye's peak sensitivity is around 555nm (yellow green), and the sensitivity curve drops off as you move toward red or blue wavelengths. A 532nm green laser at 1mW appears about 3 to 4 times brighter than a 650nm red laser at the same power.

This means we can use lower optical power to achieve the same visual brightness, which improves safety margins and reduces concerns during safety review. Class 2 is considered eye safe under normal circumstances because the blink reflex protects your eye. You'd have to deliberately stare into the beam for several seconds to risk eye damage. The drawback for this component is the cost which is much higher than a red laser diode.

3.1.15.3 - Green Laser Modules (Class 3R)

Green lasers at 520 to 532nm in Class 3R provide brighter dots for well lit spaces. Class 3R can cause eye damage with brief exposure, but it's still low enough that it's allowed without extensive safety controls. If we end up doing demos in brightly lit spaces or outdoors, we might need to step up to Class 3R to maintain visibility, but that would require additional safety briefing and possibly laser safety goggles available for operators.

Table 27: Laser Technology Comparison Table

Laser Type	**Wavelength**	**Class**	**Visibility in Room Light**	**Eye Safety**	**Cost**
Red, 5V module	650nm	<5mW	Poor	Moderate risk	Low
Green, Class 2 module	520-532nm	≤1mW	Excellent	Safest, blink reflex protects	Medium
Green, Class 3R module	520-532nm	1-5mW	Excellent	Requires brief safety training	Medium

3.1.15.4 Decision on Laser Technology

After comparing all available laser options, we decided to use a red, 650 nm, 5 V laser module. This choice provides the best balance of cost, simplicity, and safety for our capstone project. The red module is inexpensive, readily available, and requires no special handling procedures or safety reviews. While the red dot is less bright than a green one, it remains visible at the 10ft indoor range where our demonstrations take place. The reduced optical brightness also contributes to a safer system for presentations and classroom environments.

In short, the red laser meets all visibility and functionality requirements without adding unnecessary cost or complexity. It integrates directly into our existing 5 V rail with minimal current draw.

Table 28: Component listing

Component	**Supply Voltage**	**Current**	**Quantity**
Buzzer - CEM-1206S	5V	45mA	2
LED -	Vop = 3.3V or 5V, Vf = 2V	20mA	~7
IMU - **ISM330DHCX**	1.71V - 3.6V	1.2mA - 1.5mA	1
(Laser diode)	5V	1mA	1
Huskeylens	3.3V-5V	~230mA-320mA	1
Arducam Mega SPI Camera 3MP	3.3V	56mA-136mA	2
ESP-32	3.3V-5V	~50mA-100mA	2
TD-8120MG High Torque Digital Servo	4.8V to 7.4V	~2.3A +/- 10%	2
(Brushless DC Motor)	12V	18A start up 10A continuous	2
Total power 3.3V	3.3 * (20mA * 7 + 1.5mA + 136mA + 2*100mA =3.*4775mA = 1.4325W		
Total power 5V	5V * (1mA + 45mA*2+1mA + 230mA) = 5*352mA = 1.76W		

Total power 7V	7*2.3A = 16.1W
Total power 12V	Startup motors at same time 20A * 2 * 12V = 480W Stagger start motors 20A * 1 * 12V + 10A * 1 * 12V = 360W Worst case power draw is 480W
Total power in system	**480 + 1.432 + 1.76 = 483.192W**

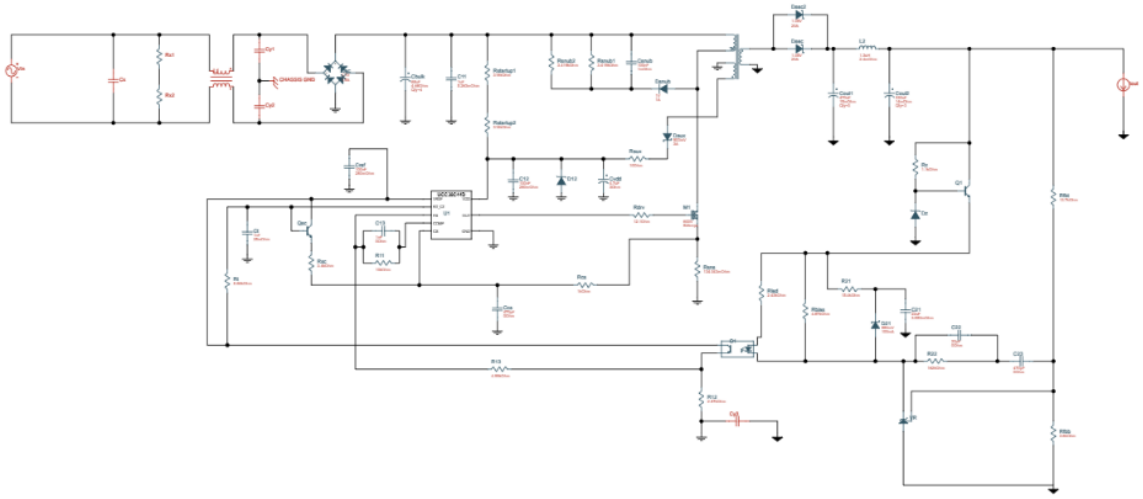
3.2 Power Supplies

For our project we wanted to imitate what a real system would look like in real world situations. Doing this allows us to bring up possible other ways to operate/power our system. Our choices lead us to these three options. Plug in power supply from wall that takes 120V/12V which will plug into access board PCB that then regulates voltage from 12V/5V/3.3V which then distribute to other peripherals, to build our own 120V/12V/5V/3.3V regulator PCB which will take in a plug from the wall into the board and then distribute power to our system, or lastly is use 9V batteries which will be on our access board and step up or down voltage for peripherals such as motors or cameras.

3.2.1 - Design of 120V/12V/5V/3.3V regulator

This design, like the rest of the regulators, will require us to total the current from each component to get the total current coming into our design needed. After adding all of these numbers up we ended up with around 7.5A needed for our system to operate. To account for spikes in current or anything that may cause issues, we decided to up the incoming current to 10A which also helps with the start of up motors or servos we have in our turret.

Figure 4. Power supply schematic for 120VAC/12VDC generated using Texas Instruments WEBENCH® Power Designer (accessed September 18, 2025).



The design has many faults that could occur, and complicated parts that would take multiple trial and error tests to make sure they work before plugging into 120VAC. This is only for taking 120VAC to 12VDC. We still need space to test the other 12V to 5V and 5V to 3.3V regulators. This goes against our requirements of trying to have a small and portable system. To allows us to focus on other specifics such as better coding and extra testing on easier designs with easier parts, we decided to not move forward with this design

3.2.1.2 - Build 12V/5V/3.3V regulators V1

Another option we are deciding on is the use of regulators that will step down the voltage incoming from our 120V/12V AC/DC wall plugin from 12V to 5V then to 3.3V. This would be accomplished by having two cascading regulators of 5V and 3.3V. We will have the output of the 5V design from WEBench feed into the input of our 3.3V which allows for easier design for the components that will need just 5V and the others that will just need 3.3V. With this design, a possible issue is the wires needed to come out of the access box to power the rest of the system. We would need a cord for the motors and servos which would be their own wires of at least two; we would need at least one 5V and 3.3V wire and the final one being a grounding wire on the turret PCB. Having so many wires could cause latency issues and open the design to issues if we connect wires incorrectly and do not check before applying power. We feel this is still better in the long run for possible frying components or shortages if everything were to be on one board. A lot of these designs can be found in common household tools and other electronic devices, so plenty of designs and schematics are available and easy to replicate. For this project we want to make sure we have plenty of time to test such designs and make sure they are cheap in case of any breaks or issues in initial designs; reordering would not cause us to go over budget.

3.2.1.3 - Build 12V/5V/3.3V regulators V2

Originally, the regulator mentioned in the previous paragraph would be made in cascade to help with managing current along with keeping the space between each regulator smaller. However, many problems come into play when we start to cascade regulators instead of running them in parallel. Running in cascade causes drops in voltage which at first doesn't sound like a problem. But when designing these regulators, we are putting an expected incoming voltage into the system. If we design the 12V/5V regulator and we have too much voltage drop due to the components that will need 12V, it could cause the incoming voltage to not be high enough to regulate. This could cause the output voltage to not be high enough either meaning that the components that need to be at 5V won't be able to operate and same for the 3.3V if the voltage is too low there as well. A fix is having the regulators separated from each other and running them in parallel with the incoming voltage. Splitting the voltage across multiple lines so the output can go to multiple devices would allow the components to take the current and voltage needed without the worry of dropping the voltage too low so that other regulators won't work. A drawback to this is that having more signal lines come out of the output of each regulator could allow more problems to happen such as messed up/not connected lines or wrong wire connections all together. Spacing could be expanded as well due to components needing to be spread out to help reduce noise between them as well, which goes against the requirements we have of keeping the access box and turret PCB small.

3.2.2 – Power Supply

The motors that power our turret can take up most of the current needed in this project. The current can be as high as 20A needed to start the motor and another 10A to keep one rotating at the speed we need to shoot. Meaning if we start the two motors at the same time, we need at least 40A coming into our system for the motors alone, not including other components. A possible way to get around this is having a power supply be at the start of our power cord which will then feed into one of the two versions of our regulator designs for the access box. We would need this power supply to allow for the motors to run at start up and at idle time while also supplying the other components of the system such as the cameras and the MCU. We would have a power cord that goes from the wall into our external power supply. This power supply would either then plug into the turret PCB or plug into the access box. Plugging into the turret PCB could help with wiring issues. Since most of the 12V components will be over by the turret, it would be easier to design circuits and PCBs so that everything is wired over there. From the turret PCB, we would then trace an output wire of 12V into the access box which would then go through one of our regulators to supply power to the other components. This approach would have minimal wire output coming out of the turret but could pose the risk of damaging the turret side of our project. The goal is to only have the access box be the main point of contact, and everything else is connected. Connecting the turret PCB leads to extra faults externally, such as someone tripping over the wire. This could massively affect our system due to how integrated the turret is. If the cord goes into the access box and someone trips on the cord, the access box can be repaired easily by replacing the board if a component gets damaged. If the turret were to get hurt in some way, we lose functionality of the project which would be catastrophic since the project revolves around needing to shoot a target.

3.2.3 - Using Batteries

Using batteries for our power system is also another possibility. These are cheap and readily available and have many parts that we can use in PCB designs. This would allow easier movability for our system as well and not always require the system to be near a wall in order to operate We have plenty of choices such as standard lithium-ion batteries, lithium polymer battery, etc. Many electronic devices use such batteries however we cannot use typical alkaline batteries such as Duracell AA batteries. These will not have enough current rating or proper resistance to handle the current needed to power our motors. If we used batteries, they would have to be recharged and come in packs that allow at least 12V and plenty of amperage to run our motors. Some options we found can be found below.

3.2.3.1 - LG MJ1 18650

These lithium-ion batteries would be meant to be run with at least 4 batteries in series to create an output of at least 12V for our motors and servos. This would allow us to keep a low current while keeping the necessary voltage still to keep the motors on even during start up. Even with doing these batteries in series, the current needed from the motors is still too high and requires a current regulator to step up the current needed for our system. After looking at WEBENCH and needing custom parts to accommodate this, it does not seem feasible to use this type of design even if it would be easier to use compared to a wall plug. The difficulty to implement this design far outweighs the complications of needing to be near a wall outlet

3.2.3.2 - Zippy Compact 4S 5000mAh 25C LiPo Pack

This lithium polymer battery pack is much more within the scope of what we need for this project. It can run automatically at higher voltages, usually around 11V to above or at 12V. It also has a much higher current draw ranging from 10A up to 40A+, which is the range needed for this system due to the motors and servos. These types of battery packs are used for other RC applications such as drones, RC cars and boats, and many other electronic toys/vehicles. This would not require the use of regulators for current or voltage that goes into the turret and can be directly plugged into our motors and servos, which could also help with noise and latency in our system. The issue with these batteries is the stability at long run times and usage. We would have to factor in possible extra safety measures if we implemented this design but also need to adjust the budget due to the unknown issues that could arise from using such batteries in this system

Table 28: Power Technology Options

	Supply Voltage Range	Current capacity	Build difficulty	Repair difficulty	Cost
Wall plug in and 120V to	12V to 3.3V	As high as needed	Extremely difficult	Extremely difficult	>$100

3.3V regulator					
Wall plug and build series 12V/5V and 5V/3.3V	12V to 3.3V	Limited to wall plug bought, usually around 20A-30A	Simple, designs come from WEBench	Simple, designs come from WEBench	>$60
Wall plug and parallel 12V/5V and 5V/3.3V	12V to 3.3V	Limited to wall plug bought, usually around 20A-30A	Simple, designs come from WEBench	Simple, designs come from WEBench	>$60
Use of Batteries	At 12V, use of regulators to step down	Rated off battery	Harder due to needing a battery management system	Extra systems needed to be kept an eye on	<$60
External Power supply with parallel regulators	12V use regulators to step down	Rated off Power Supply	Simple, plug from power supply into input of regulators	Higher than batteries and regulators due to higher current	>$100

Looking at this table, we decided to do go with an external power supply that will feed into parallel regulators to power our system. We need high current for our servos, which means we need freedom to choose a wall plugin with extra current. This can be done with an external power supply and then distributing power across the system will be the regulators in parallel. This helps with not having voltage drop between regulators and allows for easier design. The only issue with selecting all of this is the higher cost, but the difficulty is easier and so are repairs since things are more isolated.

3.2.4 - Power Architecture & Supply Selection

For the turret system we need a reliable power source that can handle the demands of spinning up two BLDC flywheel motors, moving servos, and running all the control electronics. The power supply is the only power source for the turret; no batteries involved. It has to maintain a stable 12V rail while two BLDC flywheel motors are ramping up, servos are moving around, and all the logic circuits are running. Wall power supplies aren't designed to absorb the kind of current surges you get when motors start, so we had to design the system to manage loads carefully.

The ESCs will have soft start enabled, we stagger the two flywheels by about 200 to 250 ms so they don't both hit the supply at once, and we avoid commanding big servo movements during the first few hundred milliseconds of flywheel spin up. There are also bulk capacitors placed at each ESC to buffer the inrush current and smooth out short load steps. Without these measures, the power supply voltage would dip and potentially cause resets or erratic behavior.

3.2.4.1 - Industrial Enclosed, UL Listed Supplies

These supplies have tight voltage regulation, documented overcurrent/ overvoltage/ overtemperature protections, finger safe covers, predictable thermal derating curves, and actual datasheets you can trust. The documentation tells you exactly what happens under fault conditions, which is critical when you're presenting.

These supplies also have proper EMI filtering built in, so they don't create noise on the AC line that could interfere with other lab equipment. The finger safe covers mean you can't accidentally touch live terminals even if someone opens the access panel during a demo, which is a requirement for public demonstrations on campus. With a soft start and the staggered flywheel ramp, we're expecting short peaks around 20 to 30A and steady state current somewhere around 4 to 8A.

3.2.4.2 - Repurposed Server PSUs

Server power supplies pack a lot of current into a small space, but they're loud, the fans run constantly, the connectors are nonstandard, and the documentation is inconsistent. Server PSUs are designed to run in a rack where noise doesn't matter and where they have forced airflow from the server's chassis.

In a standalone application, they're painfully loud, and the high-speed fans add vibration that could affect aiming stability. The connectors also require custom breakout boards or adapter cables, which adds cost and complexity. More importantly, most server PSUs don't have proper overload behavior documentation for standalone use.

3.2.4.3 - Low Cost LED Supply Bricks and Clones

These suppliers have attractive pricing, but their transient response is all over the place and the approval markings are questionable.

The voltage regulation on many of these units is adequate for LEDs, which don't care much about transients, but is inadequate for motor controllers that need stable voltage. Safety markings are also suspected of a lot of cheap supplies. You'll see CE or UL marks that may or may not be legitimate, and there's no reliable way to verify them without expensive testing.

Table 28: Power Supply Part Selection

Type	Typical 12 V Rating	Certifications	Terminals / Enclosure	EMI / Acoustic	Major Cons
Industrial Enclosed PSU	24–50 A (350–600 W class)	UL/CE/CB; documented protections	Covered screw terminals; closed chassis	Low EMI; some fan noise (UHP fanless)	Heavier/pricier than LED; higher-watt models larger
Low-Cost "LED" Supply	Labeled 30 A; practical 15–25 A	Often no UL/CE; sparse docs	Exposed terminals; thin case; open vents	Variable EMI; can buzz	Trips on motor surges; safety/doc concerns
Repurposed Server PSU	60–100 A (750–1200 W common)	UL/CE for server use	Edge connector; needs breakout & enclosure	Loud fans stock; strong internal EMI filtering	Needs enclosure & wiring work; noisy unless modded

3.2.4.4 - Decision on Power Supply Technology

We have decided to go with the Industrial Enclosed, UL Listed supply category. A 12V / 29A unit should cover the choreographed startup profile. If testing shows we want shorter stagger times or longer rapid-fire strings, we might bump up to a 12V / 42A unit with a remote sense to hold 12.0V right at the ESCs. This choice prioritizes safety approval, stable operation, and predictable behavior during demonstrations over cost savings.

Table 29: Power Supply Part Selection

Model	Output Rating	Notable Features	Why It Fits
Mean Well LRS-350-12	12V, 29A (348W)	Enclosed, protections	Selected baseline: covers our staged ramps with ESC bulk capacitors

Mean Well RSP-500-12	12V, 41.7A (500W)	Remote sense, PFC, remote on/off	Upgrade margin allows shorter staggers/longer strings
Mean Well UHP-350-12	12V, 29.2A (350W)	Slim, fanless	Quiet alternative; ensure airflow for full load

For each power supply we look at, something to consider is the thermal derating curve and protection behavior in the datasheet. These components need to be readily available and provide clear documentation on overcurrent and overvoltage shutdown behavior. If the supply doesn't have verified safety certifications or lacks proper documentation, we can't move forward with that part.

3.2.5 - Regulator Comparison

Below we will explore the different types of regulators we can use for our project. Most of our components can run at 3.3V or 5V with a few exceptions such as the brushless motors. We will need high efficiency and a cheap regulator when deciding which one to use since a lot of our costs are going into better accuracy such as the IMU. The other major cost will be PCB costs due to needing thicker traces and boards due to the high current of our motor. Let's see if we can find a cheap but reliable regulator

3.2.5.1 - TPS564242

This synchronous buck regulator can have an output current of up to 4A and has a voltage range of output between .6V and 7V. It's commonly used thanks to its compact design which is needed on space-constrained boards. This is not a major issue for us right now but later could be useful if we need to increase the copper on our boards. Having a smaller regulator and parts needed for it to operate might help with spacing of the board, bringing down the cost. It has an eco-mode and fast transient response however due to its compact size it does have thermal limits at high continuous current. Since these regulators won't be dealing with more than 2A on their own, we are well within the range of the max 4A. Its efficiency is around where our team needs about 90% to 95%.

3.2.5.2 - TPS565247

This synchronous buck regulator has a possible output voltage of .6V to 7V. It has a similar max current to the TPS564242 of 4A but has a better efficiency range with it being 92% to 97%. This is another common 12V regulator typically used for small systems, cameras, and routers. It has multiple operating modes for us to access, such as going for lower noise or going for higher efficiency. It has the compact design and robust protections we want allowing us to focus on overall design instead of just this part. Drawbacks are similar to other SOT563 parts, such as thermal limits and needing a careful layout when designing.

3.2.5.3 - TPS563200

The main advantage of this type of regulator is the optimal minimized BOM it needs to function. Its small BOM allows it to have a smaller footprint, making it perfect for embedded systems and small systems. It has an output range of .76V to 7V and has a max current of 3A. This max current could be an issue if we want to add more peripherals but for now it's in the running mostly for its size and BOM. It's a six-pin configuration but still has less features compared to other parts on this list

3.2.5.4 - TPS563210A

This buck regulator runs with a max current of 2A or 3A with 8 pins. It has small common power rails that need careful consideration when it comes to set up. It has a good power balance allowing it to be easy to implement. Its small size helps as well to keep its efficiency at around 90%. It has similar thermal constraints but could be harder to maintain due to its lower current capacity

3.2.5.5 - TPS566238

This 6A synchronous buck option has a feature that differs from the rest; its continuous conduction option. This type of regulator is meant to handle multiple loads and high inrush current if it exists in the system. We will be paying for this upgrade due to its higher BOM count, higher price, and higher footprint to build. We wont need something on this size but it's good to consider in case issues come up down the line. Its efficiency range is still among higher ones being at 90% to 95%

3.2.5.6 - LMR60430

This regulator is an automotive-grade buck allowing a much wider range of output from 1V to 20V and has low electromagnetic interference (EMI) in case this comes up with servo or motor issues later. It does have a lower max current of around 3A but efficiency is better, being around 90% to 95%. This model also has better thermal handling, allowing us to run near max current output if needed, whereas many other models could start to deteriorate. The system is bigger than normal models and has a slightly higher cost as well. If EMI isn't critical and VIN isn't changing much, this part would be overkill for what we need to do.

3.2.5.7 - TPS62133

This high switching frequency model of a buck converter is small and compact. It has very high efficiency at lower loads, and the switching frequency enables tiny magnetics in the system if that is something we are looking for. It has a voltage output range of .9V to 6V and runs at a max current of 3A. Its efficiency is the best yet getting up to 100% but again this part seems to be above what our system needs. We need a simple input and one output regulator which this part can do but others can do it in a simpler way.

Table 30: Regulator Part Comparison table

Part Number	Output Voltage Range	Max Current(A)	Efficiency

TPS564242	.6V - 7V	4A	90% - 95%
TPS565247	.6V - 7V	6A	92% - 98%
TPS563200	.76V - 7V	3A	88% - 92%
TPS563210A	.76V - 7V	3A	88% - 92%
TPS566238	.6V - 7V	6A	90% - 98%
LMR60430	1V – 20V	3A	90% - 95%
TPS62133	.9V - 6V	3A	90% - 100%(mode dependent

3.2.5.8 - Decision on Regulators

We have decided to go with the **TPS565247** due to its higher max current and higher efficiency. Its design is simple too for both the 5V and the 3.3V regulator schematics. Both can be small and compact allowing easier board design and more room for other peripherals. We will keep in mind TPS566238 due to its max current being so high and the fact that it's made for more inrush current type of applications. We may need this down the line if the regulators get effected by the inrush current from our motors but both systems will be isolated from each other so this should not be an issue

3.3 - Computer Vision Software

For our project we need computer vision because we need a way to process and understand what is in the images that our stationary cameras output. Computer vision lets us take the raw pixel data and turn it into useful information that can signal our system to initiate. Our main goal with this is to use blob detection, which finds areas in the image that stand out from the background and groups them together. By doing this, our system can pick out objects of interest and treat them as targets for further analysis. This makes computer vision a key part of our project, since it bridges the gap between simply capturing an image and using it for decision making

3.3.1 – Computer Vision Libraries

For our project, we have two main options when it comes to computer vision software. One option is to port over an existing library, which would allow us to use pre-built functions and tools that are already available. The other option is to develop our own custom library, where we would write the specific algorithms needed for our system from the ground up. Both approaches give us a way to handle image processing and blob detection on our embedded board, and the choice depends on how we want to balance development effort with system requirements.

3.3.1.1 – Open-Source Computer Vision Libraries

There are many libraries that offer blob detection algorithms for free online. These libraries come with built-in functions optimized for different tasks such as image filtering, edge detection, and blob detection. Using libraries will save a lot of development time and ensure that functionality is reliable and efficient. By using a library, more time and energy can be focused on applying the tools to our project, rather than spending time coding the low-level operations. This can help to expand our project in the future, since these libraries come with more than just the required functionality. Using a library will cut down on development time, ensure reliable and efficient algorithms, simplify overall system design, and provide a scalable architecture.

3.3.1.2 – Custom-Made Computer Vision Library

Building our own computer vision library from scratch will allow us to have more control over how the algorithms are written and optimized for our specific use case. A custom library will be designed to only include the necessary functions needed for blob detection, ensuring that memory space is utilized optimally. A custom library will also allow for modifications to the blob detection algorithm, which can be necessary if more vigorous constraints are placed onto the blob detection algorithm. A custom library with the specific hardware components in mind will allow for easy integration onto our MCU. Creating a custom library will cut down on wasted memory, while offering control over the functions that are supported, as well as how those functions operate.

Table 30 - Computer Vision Libraries

Option	Pros	Cons
Open-Source Computer Vision Libraries	Reliable and efficient code, less development time, documentation provided, simple to implement, scalable	Source files take up significant memory, may include unnecessary functions and code, hard to optimize for specific use cases and hardware, hard to manage dependency links
Custom-Made Computer Vision Library	Fully optimized to support specific hardware components, small and manageable memory footprint, can be tailored for speed or efficiency, depending on resource restrictions.	Requires significant development time, increased risk of bugs or errors, limited flexibility with new system requirements

3.3.1.3 - Decision of Computer Vision Libraries

Based on the table, we have decided to create our own custom-made computer vision library. The main driver of this choice was memory restrictions. Embedded boards come with limited memory, and many online libraries require multiple megabytes of flash memory to store the library alone. When a running process links those libraries to the program code and combines that with the storage space needed for the heap, stack, and runtime data, the overall process size could increase beyond what the MCU can support. The second important factor is that we want to be able to optimize the blob detection algorithm to avoid detecting objects that are either too large or too small, which is a feature we may not be able to guarantee with online libraries. The third factor is that these algorithms are well documented online, giving us many different places to find code and support for building a complete library. The final factor is that it will be easiest to ensure that the input and output of the different functions of our library will work well with the MCU. This means that our input and output will be decided by the limitations of our MCU, and our output will be customized to best be passed down the software pipeline.

3.3.2 – Blob Detection Algorithms

There are two types of object detection algorithms, single-stage object detection and two-stage object detection. Single-stage algorithms detect objects with one pass through the image. This type of algorithm is most suitable for real-time or resource-consumed systems. The downside of this type of object detection is that it is generally less accurate. Two-stage object detection uses two passes of the image to make predictions. The first pass sets up possible predictions for where objects may be, and the second pass refines those predictions. This approach is much more accurate than single stage but ends up being much more computationally expensive.

There are two common metrics used to evaluate the precision of each model, Intersection of Union (IoU) and the Average Precision (AP) metrics. The IoU is calculated by taking the bounding box of the prediction and the bounding box of the ground truth and overlapping the two. Taking the area of the intersection between the two and dividing that the total area covered by the bounding boxes provides a numerical measure of how close the prediction is to the ground truth. The AP metric measures precision and recall. Precision measures the proportion of predicted boxes that correspond to the actual object. Recall measures the fraction of true objects that were correctly detected. High precision means there were few false alarms, and high recall means there were few object misses. We then use IoU and compare it to a threshold, usually 0.5, and if it is greater, then it is a true positive, and if it is less, then it is a false positive. Using the false positive and true positive numbers, we can graph the precision vs recall curve and take the integral of this curve to get our AP for each class, or specific object we are detecting. If there are multiple classes, then take the average of all APs to determine the mean AP (mAP).

3.3.2.1 – YOLO Algorithm

One algorithm is called YOLO, or you only look at it once. YOLO uses 24 convolutional layers followed by 2 fully linked layers within the convolutional neural network. The first 20 layers are pretrained on ImageNet for feature extraction. This model is then converted to perform detection by adding convolutional layers and connected layers. YOLO divides the image into an SxS grid, where if the center of an object falls into a cell, that cell is responsible for detecting the object. Each cell makes multiple predictions, with the one with the highest IoU. This bounding box is now to sole predictor for that object. Then, non-maximum suppression is used to remove redundant boxes if needed. YOLO V7 can run on images with a max resolution of 608 by 608 pixels, meaning it can detect smaller objects.

3.3.2.2 – Faster R-CNN Algorithm

Another algorithm is called Faster R-CNN. Faster R-CNN is a two-stage object detection algorithm that begins with a backbone network to extract feature maps from the input image. After the feature map is generated, a Region Proposal Network slides over it to predict both the probability that a region contains an object and the coordinates of the proposed bounding boxes. Because the RPN and detection network are trained jointly, Faster R-CNN supports end-to-end training, which improves efficiency and accuracy. The next step is to pool each variable-sized Region of Interest into a fixed-size feature map. Finally, a detection network classifies each proposed region and refines the bounding box coordinates.

3.3.2.3 – Color-Based Blob Detection Algorithm

The final algorithm is a basic color-based blob detection method. This algorithm doesn't rely on any CNNs and just outputs an array of bounding boxes that are based off color thresholds. It works by taking the image as a 2D array of pixels and looping through each pixel, determining if that pixel is part of a blob or not. A second 2D array tracks visited pixels to avoid redundant processing. As the program iterates through the image, any unvisited pixels within the color threshold will trigger the flood fill function. The flood fill function uses depth first search to go through the neighboring pixels that haven't been visited and are within the color threshold, adding them to the current blob.

Table 31: Algorithm Comparison table

Feature	**YOLOv5-N (r6.1)**	**Faster R-CNN**	**Color-Based Blob Detection**
Algorithm Type	Single-Stage Detector	Two-Stage Detector	Classical

Detection Method	CNN	Region Proposal Network and CNN	Pixel color thresholding and flood fill
Accuracy (mAP)	28.0%	> 70%	N/A
FLOPs (Billion)	4.5	~8-9	~0.002
Maximum Resolution	640x640	No maximum	No maximum
Computational Cost	Medium (Runs on CPU and GPU)	High (Runs mostly on GPU)	Low (Runs solely on CPU)
Model Size	7.2 MB	12.6MB (MobileNet)	Tens of Kilobytes
Training Requirement	Requires labeled dataset and training	Requires labeled dataset and training	No training required
Effect from external lighting / background	Low	Low	High
Implementation Complexity	Moderate	High	Simple

3.3.2.4 - Decision of Blob Detection Algorithms

Based on the table, we decided to use a color-based blob detection algorithm that uses thresholding and flood-filling. The main reason for this choice is that the other two algorithms use a CNN, which will take up too much memory space. CNNs require storage for both the model parameters and the intermediate feature maps that are generated during the convolution steps. The running processes, the CNN model size, and the intermediate feature maps will require more resources than what the ESP32 can offer. The second reason that we chose to use the thresholding algorithm is that the stationary cameras act like the first pass in a two-stage detector. The purpose of the cameras is to find regions of interest for further investigation, rather than classifying the objects. The third reason for choosing the thresholding is that the algorithm is much easier to implement, requiring only a few small functions, rather than large CNN models. This will cut down on development time, as no network will need to be trained with labeled data sets, and there are plenty of online resources available to aid in the making of this function.

3.3.3 – Wired and Wireless Communication

Our project uses three cameras to detect and track objects that are nearby. Two of these cameras are responsible for detecting if a threat is nearby, while the third camera is mounted to a pan-tilt mechanism and is used for dealing with the threat. To effectively signal and position the pan-tilt mechanism, it is necessary to transmit position and status data between the cameras and their processing units. Therefore, choosing an appropriate communication method, either wired or wireless, is critical for the operation of our system. This decision will impact speed, reliability, and flexibility of the overall system.

3.3.3.1 – Wired Communication

Wired communication refers to the transmission of data over physical connections such as USB, Ethernet, or serial interfaces between two devices. They generally offer a reliable and stable method for transmitting data because the physical medium prevents interference, noise, and signal loss. Another advantage to wired connections is consistent and high bandwidth. They have their own private communication channels, which allow for maximum data transfer speed. For example, one protocol, USB 3.0, allows a transfer rate of 5 Gbit/s. They also have a smaller power footprint because they do not need to generate radio signals, maintain wireless links, or use amplifiers, antennas, or other RF components.

Wired connections do have limitations. One disadvantage is physical cables. The cables can restrict movement and placement of hardware. If the system is complex enough, cable management can also negatively affect development and debugging. Wired also requires physical ports to connect to. These physical ports limit the number of peripherals that can be connected to the processor.

3.3.3.2 – Wireless Communication

Wireless communication refers to the transmission of data between two devices via electromagnetic waves, eliminating the need for physical wires. This flexibility makes it easier to position hardware and allows for free movement, simplifying installation, and reducing the complexity of wire management. Without wires, reconfiguring and relocating hardware components don’t require significant rewiring, saving time when setting up or making changes. Popular wireless technologies, such as Wi-Fi, Bluetooth, and ZigBee, provide communication over a reasonable range depending on the environment, making them necessary for systems where wired connections are impractical. Wireless connections also support multiple devices on the same network, creating scalable setups that can easily add new sensors and peripherals to the network in the future.

There are some major limitations to wireless communication. It is prone to interference, as shared frequency channels and environmental factors reduce bandwidth and introduce errors. Signal quality can also be degraded by the mediums, such as water and air, that the electromagnetic waves travel through. Additionally, these networks are inherently

less secure because the signals are broadcast to the world. Wireless systems also use a lot more power to broadcast radio signals and maintain network connections.

Table 32: Communication Technology Comparison

Feature	Wired Communication	Wireless Communication
Reliability and Stability	Highly reliable with minimal interference	Prone to interference, disconnection, and signal degradation.
Bandwidth and Speed	High, consistent speed with maximum bandwidth CPU clock allows	ZigBee: 250 Kbit/s Bluetooth: 50 Mbit/s Wi-Fi: > 100 Mbit/s
Power Consumption	No extra power needed	Requires power to generate signals and maintain connections
Installation and Flexibility	Complex installation with restricted movement and constrained placement	Easy installation with free movement and flexible placement
Range	Limited by cable length	ZigBee: 33 feet Bluetooth: 33 feet Wi-Fi: 30 - 50 feet

3.3.3.3 - Decision on Communication Interface

Based on the table, we decided to use wired communication because the problems a wireless connection solves are not present in this project. Although the overall system is made of different subsections, the total area that will be covered is small enough that physical wires won't restrict movement or placement. There will be no need to use a long-range communication protocol if all devices are within a couple of inches of each other. The reduced wire complexity is also not needed because there are only a few peripherals that will be connected. While the added flexibility for movement and placement would be nice, the bandwidth and speed that wired connections offer satisfy the system's needs to be fast and accurate. Power is another metric that cannot be ignored on a small-scale embedded project. Wired connections use less power, which is important when resources are constrained. Although support for wireless connections will be simple

because the ESP32 comes with built-in wireless capabilities, it will not be needed for this project, and wired connections will suffice.

3.3.4 – AI Vision Cameras

For this project, an AI vision camera is required to control the pan-tilt mechanism to automatically locate and track objects in real time. An AI vision camera is needed over a traditional camera because traditional cameras output raw image data, but our project requires this data be processed using object detection and recognition. An AI vision camera will help us avoid performing expensive computations by using its built-in processing to identify objects and output bounding boxes directly. These bounding boxes contain data on the objects' position relative to the camera, which can then be used to control the pan-tilt mechanism to orient itself towards the object. This approach allows for reliable and efficient autonomous tracking without needing to implement external computers, making the system design more compact, cheap, and suitable for embedded applications.

3.3.4.1 – OAK-D Series

One popular camera is Luxonis' OAK-D series. This camera is advertised for real-time tasks such as object detection, tracking, and depth estimation. Each camera integrates an Intel Movidius Myriad X VPU which handles the onboard neural networks without relying on an external processor. The OAK-D cameras provide color and depth data simultaneously, allowing for more precise object location predictions and distance measurement. Pretrained deep learning models like YOLO and neural network architectures like MobileNet come installed and can transfer bounding box data and labels directly via a USB 3.0 connection.

However, the OAK-D series comes with a few drawbacks for embedded applications. The modules consume more power compared to other smaller AI vision cameras, and their higher bandwidth interfaces may require a more powerful host processor if full data streams are needed. The cost of OAK-D devices is also significantly higher than simpler AI cameras. The functions that this camera provides line up with its cost, but it may be too much for this project.

3.3.4.2 - Arduino Nicla Vision

The Arduino Nicla Vision, another AI vision camera. It is designed for low power machine learning and computer vision tasks. It uses a 2-megapixel color camera with an STM32H747 dual-core processor to perform real-time object detection and classification directly on the device. An integrated 6 axis IMU provides accelerometer and gyroscope data that can be used to compensate for tilting and improve stability during object tracking. The camera supports lightweight neural networks that output bounding box coordinates and object labels through UART, SPI, and USB interfaces. This allows for easy integration with the ESP32, where the bounding box data will be converted into signals that control the pan-tilt mechanism.

While the Arduino Nicla Vision offers onboard AI processing in a compact piece of hardware, it does lack the capabilities to perform complex neural networks. There is very little onboard RAM and no built-in accelerator, meaning this camera is only suitable for lightweight models and simple detection tasks, rather than deep network models such as YOLO or SSD. The inference speed is also slow compared to other cameras, at around 10 to 20 frames per second. The camera resolution is also 2 megapixels, which may be restricted for some use cases. Additionally, the integrated 6-axis IMU does not contain a magnetometer, so its orientation estimates will drift over time. Overall, this model is useful for low-power and moderately demanding embedded applications.

3.3.4.3 – HuskyLens AI Machine Vision Sensor

The HuskyLens AI Machine Vision Sensor is designed for computer vision applications for embedded systems. It comes with built-in face recognition, object classification, color detection, line tracking, and object following. The camera also features a 2-inch IPS display that provides real-time visuals from the camera, which is highly useful for debugging and demonstrations. The HuskyLens comes with support for UART and I2C communications, which can be easily configured using the cameras for custom GUI. The camera also comes with the ability to train models directly on the device, simplifying the process for learning custom objects by removing external hardware and datasets.

There are some potential flaws with the HuskyLens. First, it has low resolution, only about 320x240, meaning it will struggle to detect small objects or overlapping objects. The camera also has low processing power, restricting the cameras' ability to perform advanced computer vision tasks, like object detection with multiple classes or at large distances. Additionally, the frame rate is limited by the lack of an accelerator and low onboard memory. Overall, this camera is good for small scale projects with lightweight embedded systems.

Table 33: AI Vision Part Comparison Table

Feature	**OAK-D**	**Arduino Nicla Vision**	**HuskyLens AI Machine Vision Sensor**
Processor	Intel Movidius Myriad X VPU	STM32H747AII6 Dual-core Arm Cortex- M7/M4	Kendryte K210
AI Image Resolution	300x300, 416x416, 640x640	96x96, 128x128, 224x224	320x240
Depth Sensing	Yes	Yes	No
DFOV	81°	80°	70°

FPS	~20-30	~5-15	~30
AI Algorithms	Object detection, Facial recognition, Semantic segmentation, Pose estimation, Depth-based 3D localization, Gesture & hand tracking	Image classification, Simple object detection, Color or motion detection, Gesture recognition	Facial recognition, Object tracking, Color recognition, Line tracking, Object classification
Onboard Memory	8GB flash, 1GB RAM	2MB flash, 1MB RAM	8MB flash, 6MB RAM
Communication Interfaces	USB 3.0	Wi-Fi, Bluetooth, I2C, UART, SPI, USC 3.0	UART/I2C, SPI
Power Supply	5V	3.5V – 5.5V	3.3V – 5.0V
Current Consumption	1 A	105 mA	230 mA – 320 mA
Additional Peripherals	9-Axis IMU, Stereo depth perception,	6-Axis IMU, Time of Flight sensor, Microphone	2’ IPS display, pre-programmed modes
Output	Image frames, depth maps, bounding boxes, labels, confidence scores, 3D coordinates, IMU data	Image frames, IMU data, bounding boxes, labels, confidence scores	Bounding boxes, labels, confidence scores
Price	$299	$80	$35

3.3.4.4 - Decision on AI Vision Camera

Based off the table, we have decided to go ahead and use the HuskyLens AI Machine Vision Sensor because of its low cost, high speed, and integrated AI functions. The HuskyLens comes with built-in color tracking which will be used to follow the targeted object. The only data that is needed for this operation is the bounding box coordinates of the object, which is the sole output from the HuskyLens. The other cameras offer more versatile outputs, such as image frames and IMU data, but this data is obsolete since we

won't be performing any external computations on the image data, and we will use a separate IMU to control the movement of the pan-tilt mechanism. Another feature that the other cameras offer that the HuskyLens does not is depth sensing. Depth sensing is not necessary for this project because the target object will be of a known size. However, if the target object is not a known size, then depth sensing would be necessary.

The HuskyLens also runs at 30 FPS due to its moderate memory and RAM, along with its lightweight AI algorithms. This frame rate is necessary to ensure that the system gets up-to-date coordinates of the object, allowing for more precise and continuous movements. The QVGA image resolution of the HuskyLens is the bare minimum needed to ensure that the camera can detect objects up to 10 feet away. With a cost of $35 per unit, the HuskyLens camera is the best choice.

3.3.5 - Stationary Camera Choices

The stationary cameras will be used for target detection and coordinate retrieval. They will provide visual input to detect and classify objects approaching the camera's field of view. Due to the cameras being immobile and stationed on the base of the mount, we must consider factors such as interface compatibility, resolution, overhead processing, and power consumption over things like portability.

3.3.5.1 - Wired Camera Modules

The wired camera modules will connect directly to the microcontroller through the hardware interfaces such as DVP, SPI, I2C, or UART that we implement onto the custom PCB. These modules are typically lower power, allow for direct control, and can provide raw or compressed images that we will use for processing to identify objects for our target and send the coordinates to the main controller using lightweight image detection or TinyML. Some examples of these cameras include the OV2640/5640 DVP modules or Arducam Mega SPI series.

3.3.5.2 - Wireless/Network Cameras

These camera modules connect via Wi-Fi, such as the Wyze or ESP32-CAM modules that internally process video streaming and compression. They send data to the master controller over Wi-Fi using UDP or TCP. These cameras typically offer higher resolution, but it requires more power and significant amounts of bandwidth and is not MCU-friendly. Thus, it is better suited for tasks like surveillance and not low-latency onboard detection. Since the MCU that we are using only has around 32KB of memory, it becomes difficult to store and process high resolution images, making it not ideal for us to use these types of cameras.

3.3.5.3 - AI-Integrated Cameras

These types of modules include the HuskyLens, which embed AI models directly for tasks such as object detection, line tracking, or face recognition. These modules offload the image processing from the MCU and output simplified data (i.e. bounding boxes or object IDs). Although these modules can greatly reduce the processing load and reaction time, the units are very expensive and provide less access to raw image data.

3.3.5.4 - USB Cameras

USB cameras such as the Wyze Cam are able to stream in high resolution and frame rates. The cost is not too bad at about $15-35, and it is easy to integrate as it is usually plug-and-play. However, our MCU the ESP32-S3 does not support UVC (USB Video Class), so it would require extra hardware to integrate it into our project.

Table 34: Stationary Camera Options Table

Option	**Pros**	**Cons**	**Features**	**Cost**	**TinyML Suitability**
Wired Camera Modules	Low power consumption, MCU control	MCU resource heavy, interface limits	Direct MCU interface, raw/compressed image output	$5-30	Strong (scalable resolution, lightweight processing)
Wireless /Network Cameras	Easy integration, high resolution	High bandwidth, high power consumption, not MCU-friendly	Onboard compression and streaming	$20-60	Poor (overhead is too high for the MCU)
AI-Integrated Cameras	Offloads tasks from MCU, easy to use	Expensive, limited ML options to built-in models, no raw data access	Built-in AI models, sends results only	$50-150	Neutral (Does not need or run tinyML)
USB Cameras	Cheap, high frame rates	Our MCU lacks UVC support, requires extra hardware	High-resolution, plug-and-play	$15-35	Poor (not directly usable by MCU)

3.3.5.5 - Decision of Stationary Camera Technology

From the table above, we weighed the pros and cons of each option and decided that it would be best to use a wired camera module that connects via SPI, UART, I2C, or DVP interfaces. Thus, giving us the following options to choose from:|

3.3.6 - Turret Camera

3.3.6.1 - Arducam Mega 3MP SPI Camera

The Arducam Mega 3MP connects via SPI, allowing multiple modules to operate on a single ESP32-S3 MCU by using separate chip-select lines. It provides a 3MP resolution (2048x1536) that can be scaled down and has built-in JPEG compression, which eases strain on the MCU's memory. It features autofocus and IR cut filter capabilities that help ensure accurate detection across various distances and lighting conditions. The scalable resolution allows for tuning speed or detail depending on the application. Even though SPI is slower than DVP connections, making high frame-rate image capturing limited, the trade-off allows for dual-camera operation on one MCU. The JPEG compression and scalable resolution make it ideal for tinyML applications by feeding reduced image sizes into a custom/pre-made model for blob or object detection. The cost is around $35, making it mid-range compared to the other options.

3.3.6.2 - Arducam Mega 5MP SPI Camera

This is the higher resolution version of the 3MP SPI camera mentioned above, as it offers 5MP (2592x1944) resolution. The Arducam Mega 5MP provides greater detail but has slower performance at full resolution. Other than the high resolution, the Arducam Mega 5MP shares the same SPI advantages, flexibility, and compression support as the 3MP version. This camera is moderately suitable for tinyML as the high resolution is overkill for our detection tasks and will need to be scaled down anyways, although it will offer better raw image quality for classification. The cost is around the same as the 3MP version, with it being around $35-45.

3.3.6.3 - OV2640 Camera Module

The OV2640 is a 2MP (1600x1200) camera that connects via a DVP parallel interface. The OV2640 supports multiple image formats including YUV, RGB, and compressed. The resolution can also be scaled down significantly (i.e. 40x30), making it efficient for lightweight blob detection. Using UXGA can achieve 15 fps and 30 fps at SVGA. The problem lies with the ESP32-S3 as it only has one DVP interface, requiring us to use one OV2640 per MCU without additional hardware. The Ov2640 is strongly suitable for TinyML blob detection or simple classification with lower resolution. The cost of one unit is around $5, making it a very low-cost option.

3.3.6.4 - OV5640 Camera Module

The OV5640 is a 5MP (2592x1944) sensor that connects via the DVP interface. It supports a wide range of frame rates (i.e. 30 fps at 1080p, 60 fps at 720p, 120 fps at QVGA) and includes autofocus and adjustable image quality settings. The OV5640 poses the same problem as the OV2640 module with the ESP32-S3 only having a single DVP

bus, making multi-camera setups difficult. The OV5640 also has a rolling-shutter which can cause distortion when tracking fast-moving targets. With its higher resolution, it is not as suitable for TinyML as the OV2640, as the resolution would have to be scaled down significantly before processing causing some overhead. The cost is around $15-20, making it more expensive than the OV2640, but cheaper than the Arducam modules.

3.3.6.5 - HuskyLens AI Vision Sensor

The HuskyLens is an AI-enabled vision module that outputs recognition results directly over I2C or UART. It has built-in models for object tracking, color detection, face recognition, and others allowing for minimal programming effort. This module eliminates the need for image processing on the MCU, but because the raw image data is not accessible, the flexibility is restricted. Since the HuskyLens has built-in AI models, the TinyML is not needed, meaning that we would be limited in options, but also offload heavy tasks like object recognition from the MCU. The HuskyLens is significantly more expensive than the other options, being around $50-60.

Table 35 - Detailed Camera Options Table

Features	**Arducam Mega 3MP**	**Arducam Mega 5MP**	**OV2640**	**OV5640**	**HuskyLens**
Interface	SPI	SPI	DVP	DVP	UART/I2C
Max Resolution	2048x1536	2592x1944	1600x1200	2592x1944	1600x1200
Frame Rate	15 fps at full resolution, scalable	Slower at full resolution, scalable	15 fps at UXGA, 30 fps at SVGA	30 fps at 1080p, 60 fps at 720p, 120 fps at QVGA	Not raw video
Power	185mW-585mW	182mW-650mW	125mW-140mW	18mW-294mW	1.06W-1.15W

Multi-Camera Support on ESP32-S3	Yes	Yes	No	No	Yes
Features	Autofocus, IR cut filter, JPEG compression	Same as Arducam Mega 3MP with greater image detail	Blob detection friendly	Autofocus, image controls, image compression	None
TinyML Suitability	Strong Scalable resolution	Moderate (Needs to be scaled down for lightweight detection)	Strong (low resolution scaling)	Weak (High resolution adds overhead, rolling shutter)	Neutral (no need for MCU processing)
Cost	~$35	~$35-40	~$5	~$15-20	~$50-60

3.3.6.6 - Decision of Camera Technology

After reviewing the options listed above in depth, we have decided to go with the Arducam Mega 3MP SPI for stationary cameras. Even though the OV2640 and OV5640 are lower cost and support higher frame rates, the ESP32-S3 can only support one DVP module per MCU, making a dual-camera setup impractical without increasing the hardware demands. The Arducam Mega 3MP on the other hand uses the SPI bus which the ESP32-S3 can share across multiple devices with separate chip-select lines, allowing us to use two cameras simultaneously on one MCU, helping to reduce complexity, power consumption, and cost.

The 3MP resolution is good enough for detecting approaching targets. It also has built-in JPEG compression and scalable resolution which help save space in the ESP32-S3's limited RAM of 32MB. Even though the SPI connection is slower than DVP, it does not particularly affect our application as we are focusing on detection rather than high-speed video.

The other options, such as USB cameras, were excluded due to compatibility issues with the ESP32-S3. The HuskyLens AI Vision Sensor would ease the workload of the MCU as it has built-in AI models and onboard detection/recognition; it was not chosen due to its cost and the inability to access the raw image data. The Arducam Mega 3MP was chosen over the 5MP version since the resolution would have to be scaled down significantly (lower than 3MP) to process and run lightweight detection on the images. Thus, the Arducam Mega 3MP camera gives the best balance between multi-camera support, flexibility, ease of integration, and resolution for our project's requirements.

3.3.7 - MCU Choices

The Microcontroller Unit (MCU) is responsible for controlling all of the turret functions including the pan-tilt mount servo motors, stationary cameras, main camera, and peripherals (i.e. lights and alarms). An MCU that is capable of handling real-time image detection, wireless communication, and has enough storage to store low resolution images for lightweight detection is required for this project. It should also have sufficient GPIO support in order to control multiple peripherals while drawing less power. Due to the use of multiple cameras and sensors, processing capability and memory capacity were key factors when selecting an MCU.

When choosing the MCU, we needed flexibility to support multiple communication protocols such as UART, I2C, SPI, and more so that we had the ability to connect multiple components to one MCU and allow for a range of options when considering components. The ease of implementation with other components and cost were additional considerations.

3.3.7.1 - MCU vs SBC

When choosing a controller for this project, a Microcontroller Unit (MCU) and a Single-Board Computer (SBC) were considered. The factors that were taken into consideration included cost, processing requirements in terms of this project, and power constraints. MCUs are more suited for real-time control with low latency, and SBCs are more suitable for applications that require heavier computations like large language models. However, since the project requires performing simple image detection using TinyML, controlling servo motors for a pan-tilt mount, and other real time tasks, then an MCU is a suitable choice for this project.

Table 36: MCU vs SBC

Options	**Pros**	**Cons**	**Features**
MCU	Real-time control, draws low power, low cost (~$5-20)	Limited RAM and CPU power to run	Supports RTOS, compact, able to run TinyML,

		high resolution image processing	supports multiple interfaces
SBC	Powerful CPU, may include GPU, supports heavier computations (i.e. advanced AI)	Requires more power, high cost	Able to run heavy tasks like automation or complex vision tasks, runs full Operating System (i.e., Linux)

From this comparison, an MCU was selected for the turret system. It meets the performance requirements for real-time tasks and is able to process lightweight detection models. The MCU allows for efficient power use and low costs considering budget constraints. Even though an SBC is more powerful than an MCU and would allow us to run advanced image processing with AI (even on live video streams), it is not necessary to implement it in our project. We also have a constraint that we can only use the chip and implement it onto a custom PCB that will include all the connectors, and the controls for the tasks will be flashed into memory.

3.3.7.2 - MCU Options

After determining that an MCU is the best choice for this project, multiple MCU chips were evaluated to determine the most suitable one. Several factors were considered such as memory capacity, support for GPIO, support for communication between devices and peripherals, processor performance, and power consumption. Only the MCU chips are going to be implemented in this project, since we are custom designing the PCBs. This allows us flexibility over the layout and interface design. The MCUs that we evaluated are the ESP32-S3R8V, Arduino Uno R3 (ATMega328p), Arduino Mega 2560 (ATMega2560), and Raspberry Pi Pico 2 (RP2350).

3.3.7.3 - ESP32-S3-WROOM-N32R16V

The ESP32-S3-WROOM-2-N32R16V features a dual-core 240 MHz processor, giving it the ability to handle vision tasks to an extent, control multiple peripherals, and communication between devices simultaneously. Having a dual core means that we can run tasks in parallel using each core such as having one core manage motor controls and sensors, while the other core can handle lightweight image detection using TinyML. The ESP32-S3 includes 33 GPIO pins that allow interfacing with several devices including servo motors, sensors, and cameras through UART, SPI, DVP, and other interfacing options. Its support for Wi-Fi and Bluetooth 5.0 provides us with a flexible way to communicate and connect without additional hardware. It is a cost-effective option with

it being around $5-15. The area where the ESP32-S3-WROOM-2 falls short is memory size (32MB of flash RAM and 16MB of PSRAM), which limits our ability to run heavy machine learning models on high resolution images. However, that can be overlooked with the fact that it can still process low resolution images for simple blob detection or image recognition using TinyML.

3.3.7.4 - Arduino Uno R3 (ATMega328p)

The Arduino Uno R3's ATMega328p chip is an 8-bit microcontroller that runs at 16 MHz with around 32KB of flash RAM and 2KB of SRAM. The memory capacity alone makes this MCU unsuitable for this project as it would not be able to perform real-time image processing and lightweight detection. They can't even hold images of the lowest resolution. The speed of the SPI interface is also too slow with it being about half of the CPU clock speed (~8 MHz), making the data transfer rate low affecting our reaction times and may not meet the project's target goals.

3.3.7.5 - Arduino Mega 2560 (ATMega2560)

The ATMega2560 offers more SRAM than the ATMega328p (~8KB) and more GPIO support, allowing for connections with more devices and peripherals. The 8-bit architecture and 16 MHz processing speed it possesses is not enough for this project's image processing application. The ATMega2560 also lacks support for Wi-Fi and Bluetooth, reducing the options for connectivity. While this MCU is acceptable for applications requiring more I/O tasks, it poses the same problems as the ATMega328p and is not suitable for this project.

3.3.7.6 - Raspberry Pi Pico 2 (RP2350)

The Raspberry Pi Pico 2 features the RP2350 microcontroller that possesses a dual-core ARM Cortex-M33 processor that runs at 150 MHz. The RP2350 also includes an SRAM with the capacity of up to 520KB and 16MB of flash/PSRAM, which is less than the ESP32-S3-WROOM-2, but more than the Arduino microcontrollers. It still provides adequate memory for image processing at low resolution and real-time control. It features a multitude of interfacing options including UART, SPI, I2C, etc. The lack of support for Wi-Fi and Bluetooth limits connectivity and communication between devices (i.e. sending and receiving image data through UDP/TCP), which also requires us to add additional hardware like an external module if we need that ability. Although the RP2350 is a strong option for this project, it still falls short of the features and capabilities that the ESP32-S3-WROOM-2-N32R16V offers.

Table 37: MCU Choices

Features	ESP32-S3-WROOM-2-N32R16V	Arduino Uno R3 (ATMega328p)	Arduino Mega 2560 (ATMega2560)	Raspberry Pi Pico 2 (RP2350)
CPU/Lock Speed	Dual-core Xtensa LX7 at 240MHz	Single core 8-bit AVR RISC-based at 16 MHz	Single core 8-bit AVR at 16 MHz	Dual-core ARM Cortex-M33 at 150 MHz
RAM/Flash Memory	512 KB SRAM and 16 MB of flash memory	2 KB SRAM and 32 KB of flash memory	8 KB SRAM and supports 64,128, or 256 KB of flash memory	520 KB SRAM and up to 16 MB of flash memory
Pros	Dual-core CPU for parallel tasks, able to run TinyML, able to support multiple devices, supports Wi-Fi/Bluetooth	Low power, ease of integration, simple	Same as the ATMega328p, allows for more I/O tasks with high GPIO count	Provides more RAM for processing low resolution images, dual core allows for parallel tasks, and flexible options for GPIO
Cons	Limited RAM for high resolution ML, draws more power	Not enough RAM for any image tasks, slow clock speed	Too slow for real-time image processing, not enough memory storage	Does not support Wi-Fi or Bluetooth, slow USB transfer

				speeds since it uses USB 1.1
Interface Options	33 GPIO pins, DVP, SPI, UART, I2C	23 GPIO pins, supports UART, SPI, I2C	54 GPIO pins, can support UART, SPI, I2C	30-48 GPIO pins (depending on model), can support UART, I2C, SPI, or USB 1.1
Voltage	3.0-3.6V	2.7-5.5V	1.8-5.5V	1.8-5V
Current	Ranges from ~50 to 100mA	~10-20mA	~20m	~15-100mA
Cost	$5 - $15	~$2-5	~$1-5	~$1-10

3.3.7.7 - MCU Decision

After evaluating the MCU options above, the ESP32-S3-WROOM-2 was chosen as the MCU for this project. Given the data above, it was deduced that ESP32 provides the best balance between connectivity, power efficiency, and performance. The dual-core architecture it possesses allows for multiple tasks to be performed simultaneously such as having one core manage the controls for the motors and peripherals, while the other can run lightweight ML models on real-time low-resolution images. Although the ESP32-S3-WROOM-2 has enough RAM to run TinyML applications (i.e. blob detection and tracking), it is limited when trying to perform heavier computations with complex machine learning algorithms, which is what is needed for this project. The RP2350 was a strong contender for the MCU choice, however it falls short of the ESP32 with its lack of wireless communication and slower clock speeds. The ATMega328p and ATMega2560 are seen to be unsuitable for this project as their limited memory and low processing power makes it difficult to handle real-time image data even at the lowest resolution. Their high GPIO pin count, however, makes them ideal for peripheral only controls for smaller systems. Thus, it has been decided that the ESP32-S3-WROOM-2-N32R16V is the best option for this project's requirements.

4.0 Standards and Design Constraints

4.1 IPC-2221/IPC-2222: Generic Standard on Printed Board Design

4.1.1 Standard Overview

IPC-2221 covers the generic standards need to produce printed circuit boards, and IPC-2222 extends that by discussing certain design requirements such as spacing of width tracing and ways to get around EMF. Many standards and requirements listed in this standard are needed for us due to the design of our Access box PCB but also our turret PCB as well. We need these so our boards can perform as expected during the demonstration but to also keep us and people around us safe during testing and when power is applied to our system. These standards also help to ensure the manufacturability of our board.

4.1.2 Conductor Spacing and Width Requirements

When designing our boards, something we need to keep in mind is the conductor spacing and width requirements between traces and pads. The reason we need to look out for this during design is possible shortage of trace, possible arc current, and impedance mismatch. Shortage of trace is especially hard when it comes to the physical implementation of the MCU we have chosen. When designing the traces going out of this MCU, we need to keep in mind which GPIO pins are being used because if we put too many pins too close to each other, this could cause the trace to short. Effects of this could be as simple as peripherals not responding if it's a signal-to-signal trace. For power to ground traces, these need extra care when being made. Issues with this trace short could cause the regulators to not turn on, causing them to heat and break down. Traces could burn as well along the board causing signal to not go through. To mitigate these possible issues, we will make sure each trace in the 5V and 3.3V sections have at least 7 mil spacing between each tracing, allowing for fixes of traces if needed as well. The minimum assigned by IPC-2221 is 5.1 mil between traces, but we will add an overhead to be safe. In the 12V and 7V rails we will use a space of at least 20 mils between traces. This is to ensure the higher current going through does not jump even under extreme conditions that we could experience.

4.1.3 Trace Width for Current Capacity

Not only must we worry about the length between traces, we need to worry about the width of traces themselves, especially with our motors needing an inrush current of 20A. IPC-2221 provides formulas and other materials to help determine what current would be needed and the amount of copper that is needed as well so that the board can perform as expected. What we will need to find the proper trace is the required current; temperature rise above ambient, copper weight, and whether the trace is internal or external. This can be represented by this formula below

$W= (I \wedge (0.744))/(k * \Delta T \wedge 0.44)$

I = current, k = .024 for external layers, .048 for internal layers and ΔT is change in temperature above ambient. Our motors will be running at 20A start up and a continuous 10A. We will need a trace of at least 5mm or 200mil to run the 10A safely with a 1 oz copper pour and a 10°C rise in temperature. For the 20A region, it is recommended that

we have a 2 oz copper thickness to help with stability and proper conduction. We will need a trace of at least 150mils for 1 oz copper and around 75 to 80 mils for 2 oz copper. For the rest of our board, such as MCU GPIO, LEDs, and Cameras, we can use a standard of 10 mils which is generally industry standard. The standard and formula say we only need a few mils(1 to 3 mil) but we will run well above to allow for a more robust system but still give us plenty of space to move components around as needed.

4.1.4 Via Design and Thermal Management

Vias are plated through holes connecting different layers of the board. This is important to consider for both signal routing and power distribution between layers of our boards. The via has a current capacity based off the barrel plating thickness, the diameter of the via, and the thickness of the board. This is important, especially in our project due to the servos and motors. Our other peripherals of LEDS, Cameras, and buzzer can all work with a standard via hole in Fusion360 but won't work with the motors and servos due to the higher current they will draw. Because of this, we will need to focus on routing for this section first and worry about the peripherals later in design. Avoiding vias in the turret section will be the easiest approach. Thermal vias need to be considered as well due to the high heat dissipation some components will output such as the voltage regulators or the motor drivers. We are required to have thermal vias directly under the thermal pad of surface mount components. The via diameter must be .3 to .5mm, and arrays of vias must be utilized when needed. We can also connect to internal copper planes to help with heat spreaders. For our design, when needed, we will create vias connected to the ground plane to help with the heat sink.

4.2 IEC 62368-1: Audio and Video, Information and Communication Technology Equipment

4.2.1 - Standard Overview

This standard covers many safety requirements needed when conducting tests in an electrical environment. It breaks up the safety requirements into two parts; the primary circuits, which is the circuit connected directly to the AC source, and the secondary circuits anything powered by the DC outputs of the AC source. Since we will be using a power bank, its connection will be from the wall of 120VAC and the output will be 12VDC, so both requirements must be taken into consideration when designing schematics, PCBs, and wiring of all electronics. It is recommended as well in automated systems that there is emergency stop procedures in place. Our system has automated detection but is not automated firing. To be safe, we will have easily accessible switches to all the power connections we have. Each motor will have its own fuse and switch, and the access box will contain a switch for each voltage rail to help with the insulation of any issues in the system but to also help with stopping anything if an emergency were to occur.

4.2.2 - Primary Circuit Safety requirements

We will not be designing the primary circuit aka the power bank but how to power, proper isolation, and proper handling of this power bank will be needed especially due to the high input and output that we will be handling in this situation. The use of fuses, proper grounding, and physical barriers will be needed to properly protect our input from

any issues. Fuses can help detect when too much power is coming into the system. The fuse will trip when too much power comes in, causing the system to shut down a stop operating. The AC wall cord we choose needs to have a proper grounding pin to help with safety of shocks that could occur when plugging and unplugging this device. The physical barrier we are enacting for this power bank is to make it isolated from our system. We will physically have it at least 2 feet away from any active electrical elements to help with EMF and any other transfers that could affect our system. This is in compliance with IEC 62369-1 as it suggests having the AC source and the user interface section not be in close contact with each other.

4.2.3 Secondary Circuit Requirements

The Secondary circuits will be anything connected to the output of the AC source. This includes our motors, ESC, Huskey lens, laser diode, and our access box. Fuses are recommended to help with the mitigation of high temperature rise in our circuits which could cause harm to the device and user. It also suggests that making bigger traces and paths allows for extra temperate rise without harm to the system. We plan on connecting a fuse to the output of the ESC due to the motors being the most dangerous part of our system. This fuse will serve as a warning that something has gone wrong with the motors, and we must turn off our system and access the issues.

4.3 - ANSI/NFPA 79: Electrical Standard for Industrial Machinery

4.3.1 - Standard Overview

This standard created by the National Fire Protection agency addresses the general industrial electrical safety for industrial machinery. Although this is only a senior design project and we won't be working with industrial machinery, there are still standards we should abide by and look out for during the creation of this project. Specifically we should pay attention to safety regarding the sizing of wires to use going out of our access box, power system protection, and also protection of our system since it has the use of motors and servos.

4.3.2 Wire Sizing and Current capacity

ANSI/NFPA 79 has tables that specify ratings for different sizing of wires and the current they can carry with 60°C or 75°C insulation. Due to heat loss from the current flowing through the wire, proper precautions need to be taken to avoid potential fire starters in the circuit or on the wire itself. Below is a table for common gauge wires with the current carrying capacity.

Table 38: Wire comparison

AWG	**Size Cross-Section(mm^2)**	**Current Carrying capacity**
22	.324	3A
20	.519	7A
18	.823	10A

16	1.31	15A
14	2.08	20A
12	3.31	30A

This applies to our approach to design since we need multiple wires coming out of our access box so that we can power our system. 22-guage or 20-gauge wire would suffice for most of our system, but the servo and motors are the biggest issue. These require high amperage that is not recommended to run on PCBs, but it is possible. An alternative we need to be aware of is directly wiring the power bank to our ECS which power the motors. We would need to choose an appropriate wire to do so, and these standards can help decide which wire can sustain 10A for a long period of time and which can handle a quick inrush of current if needed.

Another factor to look at is the derating factor of bundled wires. The more wires that are bundled together, the less current should be carried through these wires. The standard by NFPA lists different bundles of conducts should only use a certain percentage of the total current carrying capacity. Four to six conductor bundles should have 80% of its base current capacity; seven to nine conductor bundles should have 70% of its base current capacity, and ten to twenty conductor bundles should have 50% of its base current capacity. This is important since bundling wires would help with the presentation and neatness of our project. However, by doing this, the 3.3V and 5V power components would now need a higher current capacity rating of wire.

4.4 IEC 61010-1: Safety Requirements for Electrical Equipment for Measurement, Control, and Laboratory Use

4.4.1 Standard Overview

IEC 61010-1 establishes safety principles for laboratory and control equipment operating from mains or DC power. It focuses on preventing electric shock, overheating, and mechanical injury. Since our turret operates indoors using a 120V AC to 12V DC power supply and includes powered electronics, this standard is directly relevant. The main objective is to ensure user protection through proper insulation, enclosures, and current limiting components.

The standard covers several key areas that apply to our project. First, it defines requirements for basic and supplementary insulation between different voltage levels to prevent users from contacting hazardous voltages. Second, it establishes temperature limits for accessible surfaces and internal components to prevent burns and fire hazards. Third, it addresses mechanical hazards like moving parts, sharp edges, and pinch points that could cause injury during normal operation.

For educational and laboratory equipment like our turret, IEC 61010-1 is particularly important since the users may not have detailed electrical knowledge. The standard helps ensure that even if someone makes a mistake during operation or tries to access internal

components, the risk of serious injury remains low. It also provides guidelines for proper labeling, documentation, and emergency shutdown mechanisms.

4.4.2 Application to the Turret

Our access box and power electronics are enclosed in a PETG housing that isolates all live terminals. The flywheel and servo assemblies are mechanically shielded to prevent hand contact with moving parts. All accessible circuits operate below 60V DC, satisfying Safety Extra Low Voltage (SELV) requirements. Power lines include fuses rated below the conductor's thermal limit to prevent overheating or fire during a fault condition.

The enclosure design ensures that users cannot accidentally contact any voltage higher than 12V without using tools to open the access panel. Even with the panel open, the layout positions high current terminals away from the opening, and all connections use insulated terminals or connectors. The 12V distribution uses properly rated wire with appropriate insulation thickness, and all splices are made with insulated crimp connectors rather than exposed solder joints.

Temperature management follows IEC 61010-1 guidelines for accessible surface temperatures. Components that generate significant heat (ESCs, voltage regulators, motors) are positioned with adequate spacing for natural convection cooling. The ESCs are mounted to metal standoffs that act as heatsinks, and the enclosure design includes ventilation slots positioned to allow airflow without exposing users to moving parts or live circuits.

The mechanical shielding around the flywheels completely encloses the spinning components with only the dart feed opening exposed. This opening is sized to accept foam darts but is too small for fingers to reach the rotating wheels. The servo mechanisms are similarly enclosed, with only the output shafts exposed, and those shafts have smooth surfaces without pinch points or sharp edges.

4.5 IEC 60204-1: Safety of Machinery – Electrical Equipment of Machines

4.5.1 Standard Overview

IEC 60204-1 defines requirements for safe wiring, circuit protection, and emergency stop systems in electrically controlled machines. Though our system is small, the automated motion of servos and flywheels qualifies it as machinery in principle. The standard addresses how to properly implement control circuits, emergency stops, and protective measures for machines with electrical actuation.

Key aspects of IEC 60204-1 include proper circuit separation between control logic and power circuits, requirements for emergency stop functionality, and guidelines for wiring practices that prevent failures. The standard also covers proper labeling of all electrical components and connections, color coding of wires, and strain relief for cables that experience movement.

For our turret project, even though it's a demonstration system rather than industrial machinery, following IEC 60204-1 principles ensures the system behaves predictably and can be safely shut down in an emergency. The standard emphasis on proper wiring

practices also helps prevent intermittent failures that could cause unpredictable behavior during demonstrations.

4.5.2 Application to the Turret

The project adheres to IEC 60204-1 by isolating logic and power circuits, labeling all connectors, and providing current protection on each voltage rail. The turret includes a single emergency stop button that immediately cuts power to the ESCs and servos while leaving the low voltage logic powered for diagnostics. All wiring is color coded, strain relieved, and routed away from mechanical motion to prevent abrasion and accidental shorts during operation.

The E-Stop implementation follows the standard requirements for Category 0 emergency stops. When pressed, the E-Stop button breaks power to all motion related circuits (flywheel ESCs, servo regulators) through a mechanical contact that cannot be overridden by software. This ensures the turret stops moving even if the microcontroller has crashed or is running faulty code. The E-Stop button is a latching type that requires deliberate reset, preventing accidental restart after an emergency shutdown.

Circuit separation keeps the 12V power distribution physically separated from the 5V and 3.3V logic circuits on the PCB. Power and signal grounds are connected at a single star point to prevent ground loops while maintaining proper circuit isolation. High current paths use dedicated copper pours or heavy gauge wire, while signal traces are routed away from noisy power switching circuits to minimize electromagnetic interference.

Wire color coding follows standard conventions: red for positive power, black for ground, and various colors for signals with documentation mapping each color to its function. All connectors are labeled on both the board and the cable with their function and voltage level. This makes troubleshooting easier and reduces the risk of connecting power to the wrong circuits during maintenance or modifications.

Strain relief is implemented at all cable entry points using grommets, cable ties, or 3D printed cable guides. Cables that move with the turret (servo power, camera connections) use flexible silicone wire and are routed through cable carriers that limit bending radius and prevent the cables from snagging on moving parts. The routing keeps all cables away from the flywheel area and positions them so they can't be pinched by the rotating turret base.

4.6 IEC 60825-1: Safety of Laser Products

4.6.1 Standard Overview

IEC 60825-1 classifies and regulates all visible light lasers based on optical power, wavelength, and safety precautions. It specifies labeling, emission limits, and user protection requirements for Class 1 through Class 4 laser devices. The classification system helps users understand the potential hazards and what safety measures are required for each class.

Class 1 lasers are completely safe under all conditions of normal use. Class 2 lasers (like ours) emit visible light where eye protection is normally afforded by aversion responses

including the blink reflex. Class 3R lasers are low risk, but direct viewing should be avoided. Class 3B and Class 4 lasers are hazardous and require extensive safety controls.

The standard also defines requirements for laser product labeling, including warning labels, aperture labels, and explanatory labels. It specifies the format and content of these labels to ensure consistency across all laser products. For Class 2 lasers, the standard requires specific warning text and the laser radiation symbol but does not require protective housings or interlocks that higher classes need.

4.6.2 Application to the Laser Subsystem

Our turret uses a Class 2 green laser module (Quarton VLM-520-52 LPT, 520nm, 1mW) for aim indication. This class is considered eye safe for brief exposure due to the natural blink reflex. To comply with IEC 60825-1, the module will carry proper labeling, and its mounting includes a shroud that limits beam spread and prevents the laser from crossing the audience area during rotation. The laser is powered from the regulated 5V rail and can be electronically disabled by the microcontroller during idle states.

The laser labeling includes a warning label stating "CAUTION: CLASS 2 LASER PRODUCT" with the standard laser warning symbol. The label also includes the wavelength (520nm) and maximum output power (1mW). An aperture label near the laser opening warns "LASER APERTURE" to prevent users from looking directly into the opening. These labels are permanent and positioned where they remain visible during normal operation.

The mechanical shroud serves multiple safety functions. First, it restricts the laser beam to a forward-facing cone, preventing the beam from sweeping sideways across observers as the turret pans. The shroud extends approximately 20mm in front of the laser aperture and has an opening angle of about 30 degrees, wide enough for the aiming task but narrow enough to prevent wide angle exposure. Second, the shroud acts as a physical barrier that makes it difficult to position your eye directly in front of the laser aperture.

Software control provides additional safety. The laser only activates when the system is in active tracking mode and has a valid target lock. During idle states, system startup, or fault conditions, the laser remains off. The microcontroller can disable the laser instantly by pulling the enable pin low, which happens automatically if the E-Stop is pressed or if the system detects an error condition. This ensures the laser doesn't remain on if the turret malfunctions or loses control.

The mounting position places the laser near the center of the turret launcher assembly, aimed parallel to the dart trajectory. This positioning ensures the laser dot indicates where the dart will go, but it also means the laser is never pointed upward at steep angles where it could more easily catch someone's eye. The maximum elevation angle is mechanically limited to prevent the turret from pointing the laser at faces of standing observers.

4.6 - Communication Protocol Standards

4.6.1 - Standard Overview

The communication systems implemented in this project are required to transmit data between the two MCUs (master and secondary), main camera (HuskyLens), servo motors, stationary cameras, sensors, and other peripherals. Due to the reliance on various low-level serial communication interfaces (i.e. SPI, UART, etc.) and optional wireless links over Wi-Fi or Bluetooth, using established electrical and communication standards allow for a more reliable and stable operation of the turret system.

The turret system primarily uses SPI, UART, and I^2C serial communication protocols, as each performs specific actions based on distance, data rate, required position, and protocol support (how many connections are available on the MCU). There is Wi-Fi and Bluetooth compatibility available on the MCU that can be used for communication between the two MCUs (i.e. send coordinate data to master MCU for fine tracking), however for a more reliable and low-latency option we will connect the MCUs via a wired connection. These component-level serial communication standards are established by several organizations/companies including Motorola, EIA/TIA, and NXP that define safe operating voltage levels, timing of data transmission, and bus configurations in embedded systems.

Standards applied to this project design:

EIA/TIA-232 and EIA/TIA-485:

This standard defines the characteristics for using UART, which is an asynchronous serial communication.

Motorola SPI Standard:

This standard describes a synchronous serial communication protocol (SPI) with high-speed data transfer between master and slave devices that is suitable for peripherals like cameras in embedded systems.

NXP I^2C Standard (UM10204):

This standard defines the two-wire serial communication protocol (I^2C) that is primarily used for sensors and peripherals. Following the above standards will allow for reduced noise disturbance, prevent signal contention, and maintain integrity on custom designed PCB traces. With the combination of standards, proper line termination, isolation, and other factors, we can ensure that the system has robust, stable, and reliable communication between components.

4.6.2 - Wired Communication Protocol and Implementation

4.6.2.1 - I^2C Communication

The Inter-Integrated Circuit (I^2C)communication protocol allows for low-speed sensor data transmission, primarily from the IMU readings. I^2C provides a two-wire interface Serial Clock Line (SCL) and Serial Data Line (SDA) and is compatible with multiple devices interfacing using a shared bus. Pull up resistors (between I^2C) are connected on

both lines to prevent signal interference between devices. The cable length can be adjusted to fit the proportions of our system and be minimized to ensure clock stability by reducing the line capacitance to around 400 pF and below, which is recommended according to the standard. With its ability to support multiple devices, there are bound to be collisions when transmitting data, so to prevent that the protocol implements a mechanism that monitors the bus for any incoming traffic and if a transmission is detected it will withdraw. This protocol was chosen for its ease of use and efficiency, making it ideal for short distance communication.

4.6.2.2 - SPI Communication

The Serial Peripheral Interface (SPI) communication protocol allows for high-speed data transfer between stationary cameras and MCU to perform tasks like capturing real-time image data. For multiple device use, each slave device will connect to a dedicated chip-select (CS) line to prevent data collision and will have a pull-up resistor attached to prevent unwanted activation when MCU is waking up. It operates using a 4-wire interface that is for the CS, MOSI (master out slave in), MISO (master in slave out), and SCLK (serial clock) lines which are routed directly on the PCB to help maintain clock integrity when operating at higher frequencies. The SPI bus will operate up to 8 MHz (maximum clock speed of Arducam) with 3.3V operating voltage, balancing data throughput and signal integrity. When switching between cameras, it is recommended to use decoupling capacitors near the power source for noise filtering. To prevent noise disturbance/interference caused by high-current motor lines, proper grounding and resistance will be integrated into our custom PCB. Given SPI's ability for high-speed data transmission, support of half and full duplex data exchange between a master and multiple slave devices, and since the master controls the data flows which means that it does not require separate mechanisms to handle it which makes SPI protocol ideal for handling devices like the stationary cameras.

4.6.2.3 - UART Communication

UART will be used for serial communication between the master MCU and the HuskyLens AI vision sensor. This connection allows for streaming data continuously at varying baud rates using independent TX and RX lines. The standard recommends including series resistors near MCU pins to reduce EMI/RFI. Its ease of use, reliable point to point communication, debugging capabilities, and ability for asynchronous operation (helps timing among multiple devices) are the reasons UART communication was chosen.

4.6.3 - Safety Measures

Mechanisms are deployed in this project to detect, correct, or recover certain errors that can occur when communicating between multiple devices simultaneously. The SPI and UART data packets should include some kind of CRC or checksum validation in case any bit-level errors occur due to either noise or data corruption and ensure the integrity of the data during transmission. The MCU has a watchdog timer that will act as a fail-safe mechanism if the system becomes unresponsive. For cases like bus hang when using I^2C where the device on the bus holds one of the lines (SDA or SCL) in a low state, which prevents other devices from communicating on the bus, a toggle on the clock line can be

placed to reset the communication without having to do a full reboot, saving a lot of time. Defaults could also be implemented in case of communication loss with critical peripherals where the system can enter a default state to prevent damage or uncontrolled processes.

4.7 ISO/IEC/IEEE 12207:2017 Systems and software engineering — Software life cycle processes

4.7.1 Standard Overview

This standard establishes a comprehensive framework for defining, managing, and implementing software throughout the entire project life cycle. It outlines the processes required for obtaining third party software, developing software, system maintenance, and retirement of the system. There is a strong emphasis on quality assurance, configuration management, and verification of functionality. This standard provides a common structure for organizations and project-level software processes, ensuring that there is consistency and traceability between all stages of the life cycle, from initial requirements to deployment. By following this standard, our team can be certain that the development of our project will be fluid, adhere to industry's best practices, and the final output will meet the defined requirements for the system. This standard is essential for projects that demand reliability, maintainability, and formal documentation of software engineering practices.

4.7.2 Primary Life Cycle Processes

This standard organizes development into a clearly defined life cycle process: acquisition, development, operation, and maintenance. Acquisition and development help with deciding what the requirements are, overall design of the system, as well as creating the actual code, then testing the code, and then integrating the code. This is also when it is decided what third-party software will be used. The acquisition and development stage ensure that the software meets system requirements and performs in a safe and expected manner. The second stage, operation, involves monitoring the software as it is deployed in its intended environment. This stage ensures that it is reliable and user-friendly. The last stage is maintenance. This stage deals with bug fixes and requirements changes.

Having this architecture for our project will help to ensure that each software change is documented and easily reversible if a critical error is produced. By following this life cycle of development, it helps us to focus our time effectively, such as only writing software once it is decided exactly what the software needs to do.

4.7.3 Supporting Processes

Another process defined by this standard is the tools, methods, and assurance that are activated that make the life cycle effective and reliable. This includes documentation and configuration management, verification and validation, quality assurance, and problem resolution with risk management. Documentation and configuration management provide clear and verbose documents explaining what changes were made and by who. This helps track versions and software updates. Verification and validation check if the software matches the determined design and specifications, such as its speed and accuracy, as well as if it does what it is supposed to do. Quality assurance ensures that the system meets established quality standards, such as safety measurements. Lastly, problem resolution with risk management helps identify, track, and resolve bugs while weighing the amount of risk that each bug produces. This helps determine how likely a bug is to happen and if it is worth fixing the bug or not.

This supporting process will help to iron out how to go about each stage of the development life cycle. Having clear configuration management will ensure that each change is accounted for, as well as the supporting documentation to explain why it was needed. Performing tests that confirm that we have a quality product that meets the defined requirements will help reduce future errors and offer predictable performance across all stages of the life cycle. The standard also explains how to format code so that it is structured, readable, and maintainable, which is important as all developers will be working together to edit files and peer review each other's changes before committing a change.

4.8 ISO/IEC 14882:2024 Programming languages — C++

4.8.1 Standard Overview

The standard discusses the specifications and requirements for the C++ programming language, and ensures consistency, reliability, and portability over all software and hardware platforms. In this standard, a formal definition of C++ language syntax, standard libraries, and core features such as object-oriented programming and memory management are established. There are distinct differences laid out between the core language and its associated data types, operators, and control flow and those of the standard library, which help to provide reusable components for IO, concurrency, and common functions. Because our system involves low-level hardware interaction with high performance image processing, it is important that we adhere to these standards to ensure that our C++ code behaves in a predictable manner and is compatible among different compilers for embedded environments. This standard will also ensure safe programming practices by using type safety, defined behaviors, and error handling when handling live data from external devices. Implementing this standard will help us to

create reliable software that is easily maintained and compliant with the official C++ language.

4.9 ISO 10218-1 Robotics — Safety requirements

4.9.1 Standard Overview

This standard covers the safety requirements for industrial robots, focusing on the design and construction of the robotic system to protect operators and nearby personnel. It defines limits for robot movement, force, and speed, as well as adding protective measures such as emergency stops and guards. For our system, this standard will be relevant because of the pan-tilt mechanism and attached turret that involve automated motion. This motion can become a risk if not properly controlled. By following ISO 10218-1, we can implement safeguards to make sure that all moving parts stop on command if there was an error or emergency, and the system operates within safe mechanical limits. These safeguards include switches that can isolate power to the motors and design motion control software to limit the maximum speed and power. Adhering to this standard will help keep the operators, system, and surrounding environment safe during testing and normal operation.

4.9.2 Safe Motion and Operator Safety

For our project, ISO 10218-1 will guide the design and operation of the pan-tilt mechanism and turret by enforcing strict motion limits with emergency stop mechanisms and clear outlines of where the robot can and cannot reach. When programming the motors and actuators, we will ensure that the movement stays within the defined speed and accepted force limits. There will also be accessible switches that can immediately cut power to the system in case of an emergency. This standard will force us to use software to create safety zones, which means our algorithms need to be intelligent enough to distinguish humans from potential threats. This will avoid any accidental harm to the surrounding people. By following this standard, we will create a motion control system that is effective at detecting only the necessary objects, as well as creating a safe environment for operators and nearby personnel.

4.10 ISO/IEC 61966-2-1 sRGB Color Space

4.10.1 Standard Overview

This standard defines sRGB, which is a dedicated color space for red, green, and blue images that ensure consistent color representation among cameras, displays, and other devices. ISO/IEC 61699-2-1 specifies the chromaticity of the primary colors, the white point, and the gamma curve used to map linear RGB values to the non-linear format typically used in digital imaging. For our project, this is important because the stationary cameras output RGB frames, and we need the color data to be interpreted consistently

when performing image processing and object detection. Adhering to the sRGB standard ensures that the numeric RGB values correspond accurately to perceived colors, which is crucial when bounding boxes are generated based on detected objects or when additional visual indicators like LEDs are used to represent distances or states. By following this standard, we can reduce errors caused by inconsistent color interpretation between cameras and processing modules.

4.10.2 RGB Data Representation and Processing

When handling RGB images from the cameras, a key consideration is how pixel values are represented and processed. According to the sRGB standard, each pixel is composed of three channels, each with values typically ranging from 0 to 255 for 8-bit images. The standard defines a gamma-corrected transfer function, which ensures that the perceived brightness of colors is consistent and proportional to the numeric values. For our image processing pipeline, this means that the software can reliably convert the raw RGB frames into the formats needed for object detection algorithms, such as normalizing pixel values or converting them to alternative color spaces if required. Following sRGB also helps maintain accuracy when overlaying bounding boxes or sending visual cues to external devices, such as LED indicators or display outputs, ensuring that the processed information accurately reflects the camera data.

4.11 - Time Constraints

4.11.1 - Overview

This project has two timed constraints, with one of them being the development process that includes the research, design, testing, and implementation of the turret system over the course of two semesters. The other time constraint is the real-time system that requires the turret system to respond to real-time events within a certain time frame. To successfully build this turret system, a schedule should be followed closely and be able to accommodate any issues that may arise.

4.11.2 - Development Time Constraints

This project is taking place over two consecutive 16-week semesters. This should give us adequate time to plan and develop our system with some headroom for any problems that may occur. The first semester will focus on research, design, and prototyping, while the second semester focuses on implementing, testing, and finalizing the turret system. Proper scheduling of our time is key to completing a functional turret system by having time constraints that will help with prioritizing tasks and deciding on designs.

The turret system consists of multiple components, including two microcontrollers, two to three PCBs that we are custom designing, three cameras, two servo motors for a pan-tilt mount, and several other peripherals making it a complex system as we need proper coordination between software and hardware. Given the complexity, problems with

designing the PCB schematics and implementing the firmware are bound to happen which will lead to delays in completing this project. Therefore, it is important that we communicate with each other and set deadlines to ensure that no bottlenecks occur, causing delays later in the project.

A 32-week limit will restrict our ability to optimize the system. Ideally, we would have more time for testing, tuning, and optimization, however we need to focus more on achieving a functional system that meets the requirements we set in the proposal. By the end of the first semester, a working prototype should have been developed. By the end of the second semester, the turret system should be stable and demonstrable even if certain features must be removed or postponed due to the time constraints.

The use of custom PCBs can significantly impact the time we need to complete this project. The time it takes to draw up schematics, ordering, and fabricating the PCB introduces a challenge when scheduling as each part can take several weeks. That does not include the time it would take in cases of design errors, which could set back progress as it would require redesigning and ordering a new PCB. Therefore, it is important that the schematic is reviewed thoroughly and prototyping is completed early, so we can mitigate any risks.

Other factors that need to be considered include part shortages, shipping delays, or lab availability (limited space). The ways to counter some of these problems are to source components early on (during the first semester) and to have backup plans in case a component cannot be sourced. This is discussed with the group on a regular basis to identify the problems that could potentially cause the schedule to be thrown off course. Unexpected problems such as bugs in the software, mismatches of communication protocols, or failure of components are issues that must be addressed as quickly as possible. When integrating the subsystems, we must make sure that each part is operating correctly and cohesively, as this poses a great risk for time delays. This is combatted through carefully planning and testing each subsystem before integrating into the final turret system, avoiding the need to rework the whole system.

The effect of all these time constraints dictates that a disciplined approach to time management is key. To stay on track, the team will create a schedule, divide labor, and spend a great deal of time researching early in the process. Maintaining communication throughout both semesters, dynamically planning, and continually testing the system, we will be able to successfully complete the project within the timeframe while developing a robust system.

4.11.3 - Real-Time Systems Constraints

In addition to the time constraints related to development, there are also real-time system time constraints that must be considered to ensure the system is responsive and operates

smoothly. The feedback and response of the system is important for safety, reliability, and performance. This includes visual and auditory feedback from LEDs and buzzers to activate within 500ms of an event trigger (i.e. target detection). The synchronization of alert mechanisms with real-time events prevents any delays that may occur.

The stationary cameras and vision sensor module also require timing constraints when performing their roles. To detect targets and respond effectively, the turret system has to process the images received from the stationary cameras, send the data to the main controller, and react accordingly within 2 seconds of detecting the target movement. The response time takes the image processing using a light machine learning model (e.g. blob detection or color recognition), data transmission to servo motors on pan-tilt mount, and peripherals. A delay in this process could cause inaccuracy when tracking and detecting.

To address these requirements, task prioritization, communication between components, and rate of transferring data within the system (between MCUs and peripherals) must be taken into account when developing the software. To achieve these goals, the ESP32-S3 MCU was chosen for its dual-core capability allowing for task separation, such as controlling the servo motors, operating the stationary cameras, processing image data, tracking, and LED signaling using FreeRTOS (Real-Time Operating System) allowing for task prioritization of time-sensitive tasks from less time sensitive ones. Thus, we can minimize the number of interruptions or bottlenecks in data transfer across communication protocols (i.e. SPI or UART) while maintaining a deterministic performance. The real-time system time constraints significantly affect decisions regarding the design involving optimization, communication protocols, and MCU selection to ensure the system is able to operate consistently and reliably under several conditions.

4.12 Economic Constraints

This project also faces a financial constraint which involves how the overall budget is managed and ensures that all components and materials required for the turret system can be obtained without going over the agreed-upon budget. As the four of us are all undergraduate students, and this project is not sponsored, the cost of all materials comes out of our own pockets, therefore we have limited our budget to about $1000 USD.

The economic constraint will significantly influence the development cycle as well as the overall specifications of our system, as every decision must be weighed by performance and by cost, which in turn could lower the overall quality of the product to fit within the budget. Hardware will need to be chosen based on affordability, rather than pure performance, and software will need to be optimized and written for systems that offer fewer computing resources. Limited funding may restrict the use of high-end components and sensors, which could have been used to increase accuracy, speed, and durability. This constraint will require us to mark careful tradeoffs between quality and affordability.

Additionally, it may also limit the number of prototypes we can build and test, which will slow down our ability to detect and fix early design flaws.

Resource allocation will need to be carefully managed, ensuring that the essential elements such as microcontrollers, motors, and cameras are prioritized and given the most flexibility when it comes to their price. By prioritizing these elements, we can spend more money getting the core functionality of the system working, while potentially losing some of the less important functions. Additionally, the team may need to look at repurposing existing parts and technologies that would be free of charge and integrating those parts where needed. By using technologies that the team already possesses, such as old motors from past projects, we can reduce our overheads, but with the added cost of implementing technology that may not have been manufactured to integrate with a project like ours.

One way this constraint has affected our team is when making the decision of what to use for our color-based blob detection algorithms. The most efficient algorithms rely on deep neural networks, with millions of parameters and billions of floating-point operations. The amount of memory to hold the network, as well as the processing power to compute those floating-point operations would have required an integrated accelerator such as a GPU or TPU, as well as hundreds of megabytes, potentially even gigabytes of flash storage and RAM. The added performance and quality that a deep neural network provides for image recognition and object detection would development time and drastically increase accuracy and speed, but each of those added parts can cost anywhere from 20 – 30 dollars for additional RAM, or hundreds of dollars for a mediocre GPU. Because of the added costs of these parts, it was decided to go with a simpler color-based blob detection algorithm that doesn't require a deep neural network, sacrificing quality for a cheaper price.

4.13 Design Constraints

When designing our system, we are constrained by the internal mechanisms and performance of the electronics we choose. One such limitation is for the Arducam 3MP camera. This camera can capture high quality images, but this will result in large data files that exceed the data handling capabilities of the ESP32 MCU. The camera's SPI interface has a limited data transfer speed due to its internal clock speed of 8 MHz. This can create a bottleneck during image transmission, which will drastically slow down our system. For the maximum resolution of 3MP and a maximum transfer rate of 1MB/s, an image would take roughly 9 seconds to transfer from the Arducam to the ESP32. This limitation affects how quickly the ESP32 can receive and process new images, potentially causing unwanted latency between image capture and system response. This limitation

causes us to decrease the image quality from the Arducam, resulting in less data per image, potentially decreasing accuracy, and worse overall performance.

The ESP32-S3 microcontroller also presents another hardware constraint. Due to its restricted computational power and available memory, processing large images for object detection can quickly use up all 512 KB of onboard RAM. This may cause the device to struggle with performing real-time object detection while also controlling the rest of the system. This limitation could result in dropped frames, slower decision time, or delays in system responses. To help mediate this constraint, an ESP32 board with a dedicated "external" RAM module was chosen. This module includes 16MBs of PSRAM, connected to the main RAM through an SPI interface, running on a 120MHz clock. This allows for 15MB/s of data transfer within main RAM and the PSRAM. All operations are performed on main RAM, therefore we are limited by how fast the PSRAM and RAM can communicate, however 15MB/s should be more than enough to perform the necessary calculations.

Chapter 5: Application of ChatGPT with Similar Platforms

Since the release of ChatGPT in 2022, it has evolved into a useful tool using ChatGPT 5.0 that uses advanced reasoning and logic to solve problems and answer prompts. Since that time, there has been a spark of evolution in the AI field and multiple large language models (LLMs) have been created and released by Google, DeepSeek, Claude, etc. With all these LLMs available, it should simplify the amount of research needed as they search multiple sources and sites to formulate a logical and reasonable solution/answer. This has certainly been our experience when using multiple LLMs. However, each LLM has a limit to the amount of information they can output accurately. In my experience, ChatGPT has difficulty handling high level math involving circuits and such, but it excels at basic research by providing potential outcomes, several options, and tradeoffs without requiring too much detail in the prompt. The other LLMs such as Google AI were also able to provide sufficient information for a given prompt, however some required more elaboration such as my goal and what my expected outcome would be while some were able to deduce what I wanted and gave me more than what was asked by providing examples and possibilities for several applications. In most cases, the LLMs were able to provide sources for their information, but some struggled as they did not have access due to copyright issues, making an LLM like Google AI ideal for research since it has more access to a wide range of sources. The use cases ranged from member to member as can be seen in the case studies below, but in general each member used multiple large language models to perform different tasks for consistency and for accuracy we double checked the information and did follow up research.

5.1 - Case Study 1

For Case Study 1 please refer to the following prompt "Given the workload of controlling two servo motors, two cameras whose image data will be processed using light machine

learning model (i.e. blob detection), an main camera for fine tracking (AI vision sensor), several peripherals (LEDs, buzzers, etc.), and other controls, what MCU will be best for this architecture" and the outcomes from ChatGPT and Google AI.

ChatGPT gave a brief but detailed list of recommendations, along with reasons for the recommendations and a list of potential architectures that could be considered when designing this project. The amount of detail is appreciated as it allows us to see the pros and cons of each MCU option, making it easier to choose one. As I did give the large language models some project specific items such as being able to use an AI vision sensor, it was general enough to provide me with various options that will excel in multiple areas of this project. Even though the options given by the ChatGPT large language model are detailed enough to get a rough overview of the MCUs, they do not provide datasheet information which would be an in-depth look at each option. As I did not specify an architecture for the system, the large language model gave a few recommendations as to how the system can be designed. This allowed for more insight into what MCU to choose, depending on factors like ease of use, cost, control, etc. The information that ChatGPT did provide for each MCU seems to be factual and sufficient to meet industry standards. It provided an answer with a detailed face-level overview of the options that required extra research, which matched the specificity of the prompt provided.

When Google AI was given the same prompt, it provided an answer similar to ChatGPT's large language model. The AI model included a few options ranging from SoCs (System-on-Chip) to SBCs (Single-board computer) with pros and cons, as well as architecture options similar to ChatGPT. The information that Google AI provides is not as detailed as it does not provide details about each options feature (i.e. processor cores, types of memory, ability to run machine learning models, etc.) that ChatGPT offers despite being given the exact same prompts and its access to a larger amount of sources. Although both options were given the same prompts, each one gave similar information on the types of MCU to choose for our application, however, the ChatGPT large language model is preferred over Google AI since it offered more detailed information and insight into the path we could take to build this system depending on what we want the MCU to prioritize.

5.2 - Case Study 2

For Case Study 2 please refer to the following prompt "What stationary cameras are compatible with multiple MCUs that will be able to connect via a standard communication protocol and whose image data is able to be processed through a tinyML model" and the outcomes from ChatGPT and Claude AI.

ChatGPT and Claude AI were both given the same prompt above, and the results between the large language models are vastly different. ChatGPT gave a much more detailed list of recommended cameras, types of cameras, specific modules, and tradeoffs. The Claude AI large language model gave a very direct answer to the prompt. It gave a brief list of camera modules with communication protocols.

ChatGPT did a great job of elaborating on the prompt it was given by thinking of several possible paths that the user could take and providing relevant sources for the user to do

additional research on the topic. Given the broad scope of the prompt, ChatGPT was able to build on it and provide lots of useful information that would typically be found after researching deeper on the topic. It gave examples of camera modules using different communication protocols with explanations of why they would be good (i.e. compatibility, throughput, etc.). ChatGPT also provided tradeoffs to consider such as memory for image processing, protocol choices, or bandwidth vs latency. The information provided is much more detailed than the information output by Claude AI.

Claude AI gave a direct response to the prompt above, by giving information pertaining only to the cameras. This was unlike ChatGPT, whose large language model provided detailed information on the camera modules and possible uses for them given the application. The Claude AI model did provide relevant sources for further research on the camera module recommendations but only gives a simple bullet point list of the surface level information that requires more research. Unless I specifically ask for certain information, Claude will not expand on the options it recommended.

Although both provide adequate answers to the prompt above, ChatGPT is the preferred choice. Its ability to provide a detailed list of recommendations with considerations for several different applications/scenarios while also providing relevant sources of information makes it easier to choose an option. While Claude gives sufficient information, it only provides a surface level view of the options requiring much more research compared to ChatGPT.

5.3 - Case Study 3

For Case Study 3 please refer to the following prompt “for a 2 MCU architecture using ESP32-S3, how are the tasks split between both MCUs that will allow us to meet specific time requirements” and the outcomes from ChatGPT and Google AI.

Both large language models were able to provide feasible solutions to problems pertaining to task prioritization in a 2 MCU architecture. ChatGPT gave a general list of communication protocols that could be used for intercommunication, gave math to calculate throughput depending on the communication protocol, a list of tasks that are listed from high to low priority, an example table showing a potential architecture with timing, a possible implementation method, and the recommended split. Even though the amount of information is sufficient, it required further research to check the if the math provided was correct (seems correct at first glance but need to double check) and since no relevant sources were cited, all the information needs to be checked in depth. This is different from the solution offered by Google’s LLM.

Google AI provided a similar answer and tailored it specifically to the ESP32-S3 which was given in the prompt. It was able to pull information from relevant sources directly such as MCU capabilities from Espressif (company that makes ESP32-S3) and formulate a potential solution for how the tasks can be split between the MCUs. This makes it easier to visualize a possible path that we could take when designing the software. Aside from the numerous sources (i.e. datasheets and videos), Google’s LLM did not take into account the types of peripherals and the communication protocols they use, which can significantly affect the throughput and latency of the system. So, Google AI requires more specific instructions to get additional information and compared to ChatGPT it

focuses more on the technical aspects of the MCU, while ChatGPT's LLM was able to provide more information than the question asked such as showing math to calculate rate of data flow using SPI as well as the throughput which can significantly impact the time requirements, however the accuracy of it is still questionable.

Given its ability to provide numerous relevant sources and helpful information pertaining to the prompt, Google AI's answer is preferable over the one provided by ChatGPT. Although the ChatGPT LLM provides extra information beyond the scope of the question, it usually lacks any source of material for information like the math to calculate latency or ESP32 capabilities. Meanwhile, the Google LLM was capable of providing relevant information about the capabilities of the MCU and how the tasks could be split, while providing sources where further research could be performed should the user wish to delve deeper into the topic.

5.4 - Case Study 4

For Case Study 4 please refer to the following prompt "Given that I am using an ESP32, what is the most feasible to do object detection on objects of a certain color: YOLO, Faster-RCNN, or threshold blob-detection. Give me the reasons to use each, as well as why they may or may not be possible" and the outcomes from ChatGPT and Copilot.

ChatGPT and Copilot were both given the above prompt, and the answers were similar, with both leading to the same answer. However, ChatGPT's answer included a more detailed explanation for each part of the prompt, while Copilot provided a small table that briefly went over the given options. ChatGPT created bullet points for each algorithm and split the bullet points into sections based on why I would use the algorithm, why it may not be possible, and added a section on potential workarounds. Copilot provided the mentioned table, with about one sentence worth of data for the description, the pros, and the cons. Copilot also produced a detailed paragraph explain why some algorithms cannot be used and another paragraph explaining why an algorithm can be used.

ChatGPT gave an informative answer that explained in detail what each algorithm did. This is important because it helps the user to understand why the algorithm might have certain pros or cons. When explaining the reasons to use each algorithm, ChatGPT listed multiple bullet points of high-level reasons. This is beneficial when trying to understand what the algorithm does and what is needed for it, but the answers lacked any real statistics to back main points, such as accuracy. Regardless, the listed points give a blunt explanation of what each algorithm has to offer. The next set of bullet points provided go over the limitations of each algorithm. The bullets provided by ChatGPT quickly explain why something like Faster-RCNN will not be possible on an ESP32 board. This is beneficial to this project because it avoids wasted time learning and implementing an algorithm that ultimately will not work. The bullets also include numbers, such as minimum memory requirements, like the need for tens of megabytes of RAM to store the YOLO model. ChatGPT also included potential workarounds, which were not a part of the prompt. This bonus gives us ideas to research if the algorithm in question might be necessary, even if it cannot run natively on the ESP32. After all this, ChatGPT then provided a small table that summarized each of the points from the bullet list. This table

alone is beneficial to group characteristics together for each algorithm. Overall, the output from ChatGPT was much more useful than that provided by Copilot.

The output from Copilot consisted of a small table, then two paragraphs explain its choice. The provided table lacked a useful description of each algorithm, forcing us to look more into algorithms that may not be used. The pros and cons of the table were sufficient but lacked any in-depth reasoning. For example, the cons for threshold blob detection only describe its lack of robustness, without mentioning other important qualities such as accuracy, or explaining why it lacks robustness. Copilot also did not mention any direct workarounds; however, it did offer a hybrid approach that combines threshold object detection with one of the CNNs. After the table, Copilot provided some bullet points as to why YOLO and Faster-RCNN won't work. These bullet points went into more detail but lacked any raw numbers for the algorithms and why they may not work. The reasons for using threshold blob detection were well written and gave some new information, such as image qualities that will work with it. The information given was sufficient to explain the final verdict, but it lacked any deep reasons and statistics.

Although both AIs provided the same result, ChatGPT provided a much more elaborate response to the prompt. Both results lead us to the answer that the prompt was asking for, but the ChatGPT one explained in greater detail the pros and cons of each option. The addition of workarounds was also very helpful, because it gave the foundation for a follow up question if the workaround was something worth trying. The Copilot answer was sufficient but would not suffice if more information was needed, or I ultimately wanted to make the decision instead of the AI.

Ch 6. Hardware Design

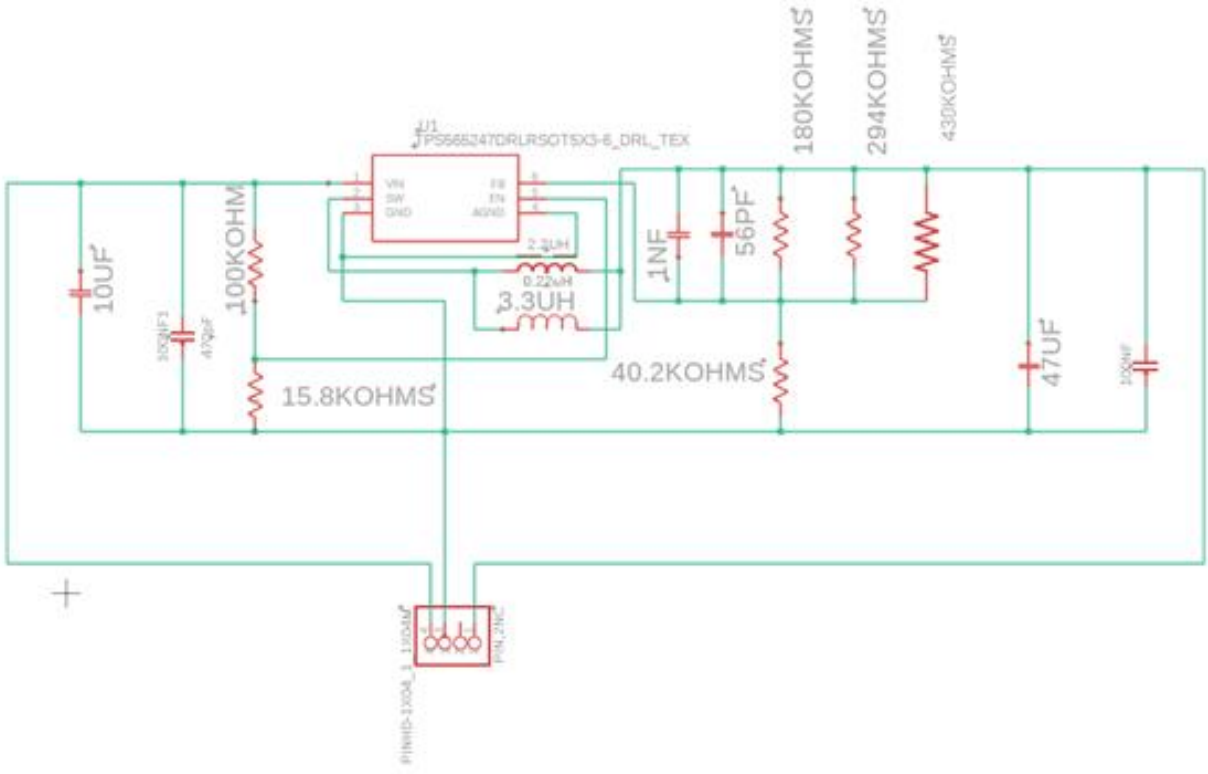
6.1 - Overview of regulators

We decided to create a unique design and try to be cost effective when it comes to needing order regulators. We decided to go with a "plug and play" approach when it comes to the output of our boards. When ordering on websites such as JLCPCB there is a minimum count of 5 per order. It is always good to have backups which we plan on having anyway; however, we will only be using no more than two regulators at each time depending on how we build our access box and our turret PCB which will be discussed later in this section. If we need to go through a second iteration/design as well, we have to order all three sets again for the 3.3V, 5V, and 7V rails. This could start to get expensive quickly, so we went with different approaches. We decided to have all the possible combinations of resistors, capacitors, and inductors each regulator would need all on one board. The output of our regulator is dependent on the voltage divider created by the two resistors at the output of our TPS565247 IC. For the 7V, it requires a different output inductor, and capacitor is the only major difference between the 3.3V and 5V. This way when we do our initial order to test, we only need to order the minimum

amount of 5 boards. We can then test and see if our design worked for each output and if it does order more to complete what we need and to have extras. If we need to redo the circuit, we can place a single order again to test. If none of this works, we can of course go to the normal circuit of just the required components on board. This use of plug and play does have a huge benefit however we could have some major cost if this doesn't work out. We are hoping this approach works out the way we want which would allow us to save on cost and difficulty when it comes to board design. We will see each section highlighted below for each rail. Everything between each regulator rail will have the same layout and components except for the orange boxes which indicate what changes are changing between circuits.

6.1.1 - 3.3V Rail

Figure 4 – Regulator Design for 3.3V rail



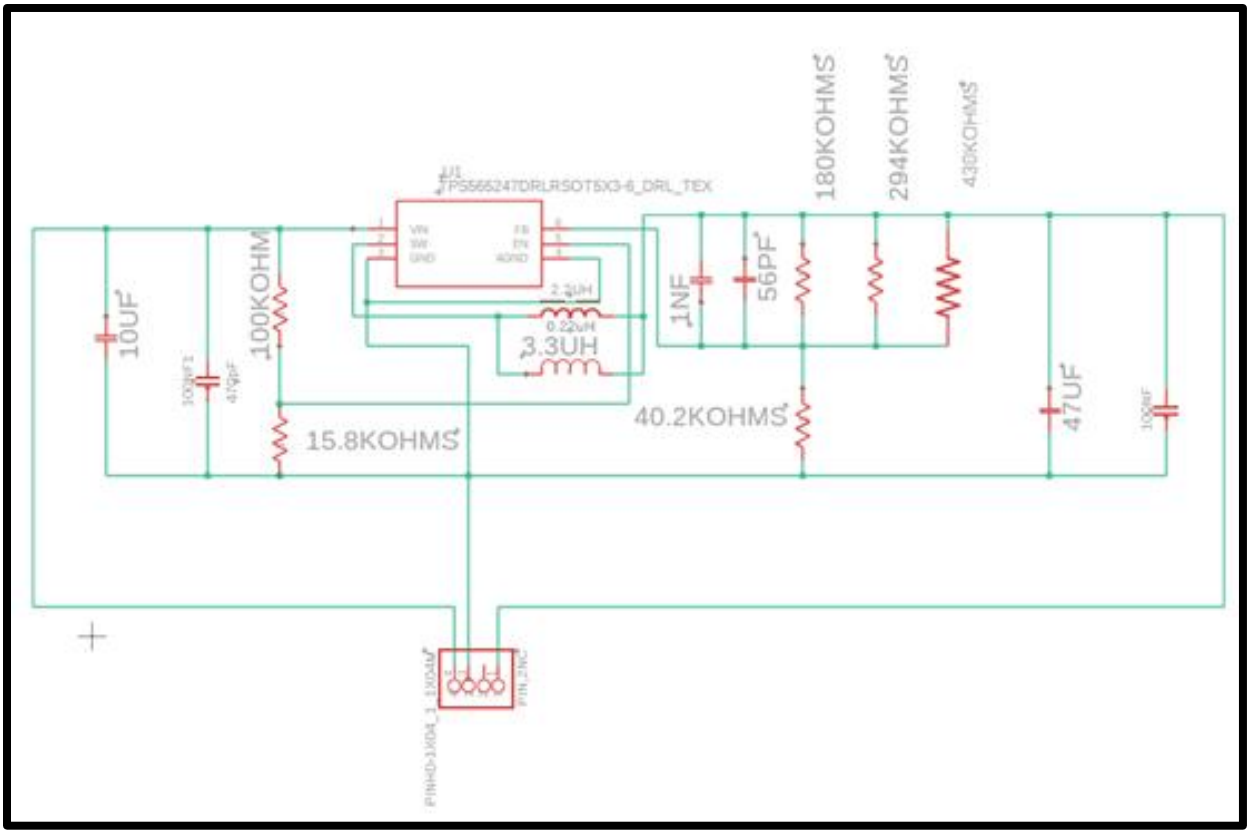
We are using the formula of Vout = Vref * (1+Rout/Rin) which is extracted from our datasheet for the TPS565247 to find the ratio of each R1 and R2 that will be needed at our output to achieve our desired voltage. As we see above, we will be using the 2.2uH inductor, a 56pF capacitor and R1 is 180k ohms and R2 is 40.2k ohms. Our IC also has a Vref of .6, let's see if we achieve our Vout.

*Vout = .6V * (1 + 180k/40.2k) = 3.2866V*

As we see, we have a proper 3.3V expected output. Our datasheet only calls for a single capacitor both on the input and output. To help with stabilization of our circuit, we decided to include some extra capacitors on both ends. This will help keep a steady frequency for the voltage coming along with the current. The output will have a better transient response and be more stable around the 3.3V line instead of varying with the help of the extra capacitors. We added these in every circuit since there are more benefits than cost to adding these in.

6.1.2 - 5V Rail

Figure 5 – Regulator design for 5V rail



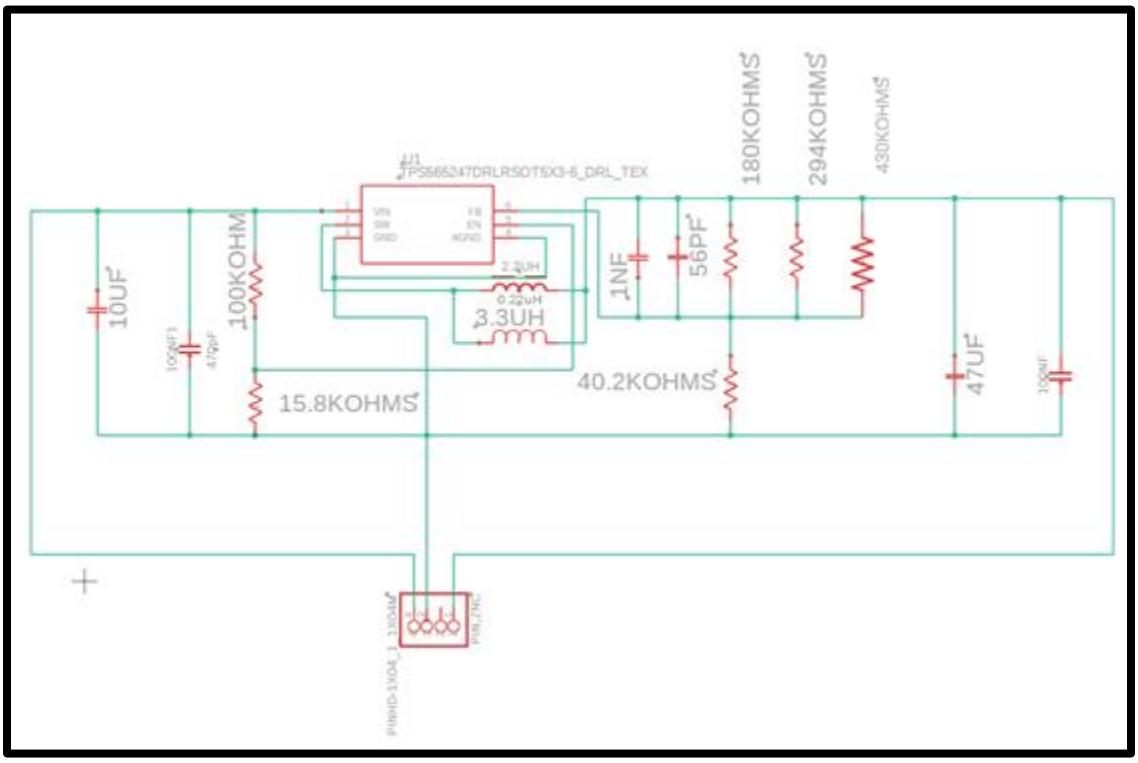
For our 5V rail we will keep the 2.2uH and our 56pF. Due to the voltage only increasing by 1.7V from our 3.3V rail, we feel these components are sufficient for our design to stabilize the frequency, keep a steady transient response and keep our voltage at our expected value. The R1, however, needs to change since we increased our voltage. R2 stays the same at 40.2k ohms, but R1 now increases to 294kohms. Below, we use our formula from the IC datasheet to check if this ratio is correct.

*Vout = .6V * (1 + 294k/40.2k) = 4.9881V*

We have achieved our 5V rail using the ratio mentioned above. In both the 5V and 7V rail, we will have more current come through our circuit compared to the 3.3V path. To help accommodate, we have made our traces be at 20mils allowing ample current to flow in our circuit with little worries of overheating. The IC pins are close together which forced us to use 12mil traces near there but shortly after, we connected 20mils so there is not too much congestion on the 12mil traces.

6.1.3 - 7V rail

Figure 6 – Regulator design for 7V rail



Here we have our 7V rail. This time we need to increase the inductor and capacitor. We are now running our IC around its max output, meaning it will be more susceptible to noise and stability issues. To help with that, we included a higher inductor of 3.3uH and increased the CFF capacitor to 1nF. Both of these changes should help the IC run at a more stable and safer limit of 7V, which will power our servos. Since this is the only thing, this regulator will be supplying, we have decided that having this IC at its max voltage output should be ok. The servos we choose can run anywhere between 4.8V to 7.4V. The issue we run into is the inverse relation of current to voltage. Since we feel current may be an issue already on our boards, we want to limit it as much as possible so we will run the servo as high as possible. If needed, we can take the output voltage of this regulator to around 6V or 6.5V if the regulator starts to burn up. Our servos will still be

fully operational with this voltage; all we need to change is the R1 to fit the ratio of our needed output voltage.

6.2 - Access Box

The use of our access box is meant to act as the brain of our system, transporting information, and relaying information to the user. We plan on having our main power switch and arm/disarm switch on here as well. This way our system is on and can alert us to objects in the area, but we won't track the objects until the arm switch is flipped on. Now we plan on having two MCUs, one on the turret PCB and another on the access box. The reason for this is the motion tracking algorithm and the data space needed for the cameras we are using will be extensive. To help with CPU utilization and latency, we have split the MCU functions. The access box will house the camera data MCU along with user peripherals such as LEDs, switches etc. The other MCU on the turret PCB will handle taking the camera data from the first MCU and we will use the turret PCB to power and control the turret along with configuring the data for our Husky Lens sensor. Now the question is do we want to 3D-print this access box or do we want to buy a premade box of some kind. Let's look at the two choices in detail.

6.2.1 - Pelican box

A premade box option we are looking at is a pelican box. This box comes with some built in padding and is made to transfer electronic equipment over long distances due to allowing foam to be in this box to hold such elements nice and tight and has a hard exterior. A hard exterior would be much more realistic as this configuration would allow for the more portable features of our system to happen with increased protection when transporting the system. We can house multiple layers of PCBs and only have the user interface PCB on top to help with wiring/display issues. We would need to take extra precautions when making the PCBs as well in this case. The first issue is making sure the PCBs are small enough to fit in the dimensions of the box. We want our system to be small as part of our requirements, but the box limits would be extra requirements not originally accounted for. This could cause some issues down the line when it comes to design. We also need to consider spacing more in this case with external wire connections. With the thick cover of the external part of the box, making holes big enough for wires but also small enough that not too much foreign debris gets in there both when the wires are plugged in and when they are not such as when we are transporting the system. Having covers for these holes when nothing being used is something that needs to be considered in the price since finding specific hole coverings for this would be no easy challenge. Making the PCBs small enough to fit on one side of the box also allows us to house the turret and the Sentry station in the box when it's not being used as well. This way this pelican box can house everything; allowing the perfect

portable box that can set up and tear down our system quickly and keep it nice and organized all in one place with extra protection in case issues arise during transportation.

6.2.2 - 3D printed box

With this option a lot gets flipped in terms of pros and cons. Here our issue is transportation and protection of the system, but freedom of use for the system is increased. Using the pelican box, we can either choose the route of deciding on the box first then fitting the PCBs to this size, or we can make the PCBs then buy a box big enough to fit what we need. Using a 3D printed box allows for the ifs to be taken care of. Designing a box would happen after PCB creation allowing more freedom of design spacing to be utilized with the PCBs. This box would be one dimensional only housing the main PCB. This main PCB would contain the user peripherals along with the regulators needed for each component and the main MCU. This allows the box to be small, and we would have minimal wires coming out of this box as well. The only wires coming out of the box should be the wires needed to power the Sentry cameras along with sending data and the power wire for the turret PCB. The turret PCB would be with the turret which would help with all the wires needed to power this portion of the system. Doing the pelican box, we will need multiple wires to come out of the box if we try to do the stacking design as mentioned where in the 3D printed version, complication of wires would be cut down significantly. The issue with these choices comes down to the protection and transportation of the system. We plan on having all of our elements, the access box, Housing station, and turret, be separated, which again will help with designing. But transporting this system could cause significant issues to function if this is not done correctly. Wires could come undone, or something could be bent on the housing station that was not meant to be.

6.2.3 - Pelican box mixed with 3D printed box

A possible solution to our problem is mixing both scenarios described above. We could design the access box and our system to the fullest, not worrying about room for now. Once the initial testing is done, we can go back into our system and scale it down to make sure it is well below our requirements to fit in our dimensions. From there we can then look online to see if we can find some pelican boxes that fit all our parts of the system into this container. This solution could help cover each con in the scenarios explained above. The protection of transportation would be there, assuming we use foam to separate the parts. The wires coming out of the box would not be an issue since everything is still housed in the access box and turret. We would not need to worry about foreign debris coming into holes we do not expect while transporting either since the pelican box will be tightly sealed. The issue with this choice is cost. We can stay in our design requirements of being small fairly easily but redesigning to fit into a box is not a priority but a luxury.

We can transport everything “unsafely” but just make sure to be careful when we are transporting each part. This choice will only be exercised later if we are well within budget and lots of this project has gone smoothly and accordingly to the plan.

6.2.4 - Decision on Access box

We have decided on 3D printing for our access box. This option provides the easiest way to implement our project with minimal hassle. We can make changes as needed and any modifications that need to happen are cheap to implement. We need room to change especially when it comes to switches and LEDs since these are the hardest to get correct when moving them from a surface mount component to a through-hole component that must be seen up top to be controlled by us.

6.3 - Overview of setup for Access Box

Below you will see the block diagram for our access box. This box is meant as the brain of our system, giving us feedback in real time based off what the cameras are seeing and is in charge of distributing power along with sending data from the cameras to the turret PCB so the turret can start its process of locking on to the enemy if there is one in range. The access box will take 12V from the power bank then distribute 12V to the turret PCB. The reason for this is we want a readily accessible switch that is near the controls for the user. The turret is not planned to be very far away from our system, but we plan on having the access box be user friendly and keeping controls in one place and everything else should be automated. We will have a switch for each regulator to help with the controls of our system and to try to isolate portions of the circuits when we can in case any rise in temperature occurs that is not expected or if the motors produce too much EMF, we have a quick way to protect the system from further harm. The access box is broken into a few different parts. The switch section meant to control power for the entire board, the regulator section which is where our regulator circuits will connect via through hole female headers, the MCU which is where the esp32 is along with its reprogramming circuit. The buzzer circuit will be activated during mission critical times and finally the status LED section which will have multiple LEDs for any type of situation our system will encounter that should be known by the user. Below you will find the overview of the entire schematic and the rest of this section will look at each circuit individually and explain our choices.

Figure 7: Access Block Diagram with Legend.

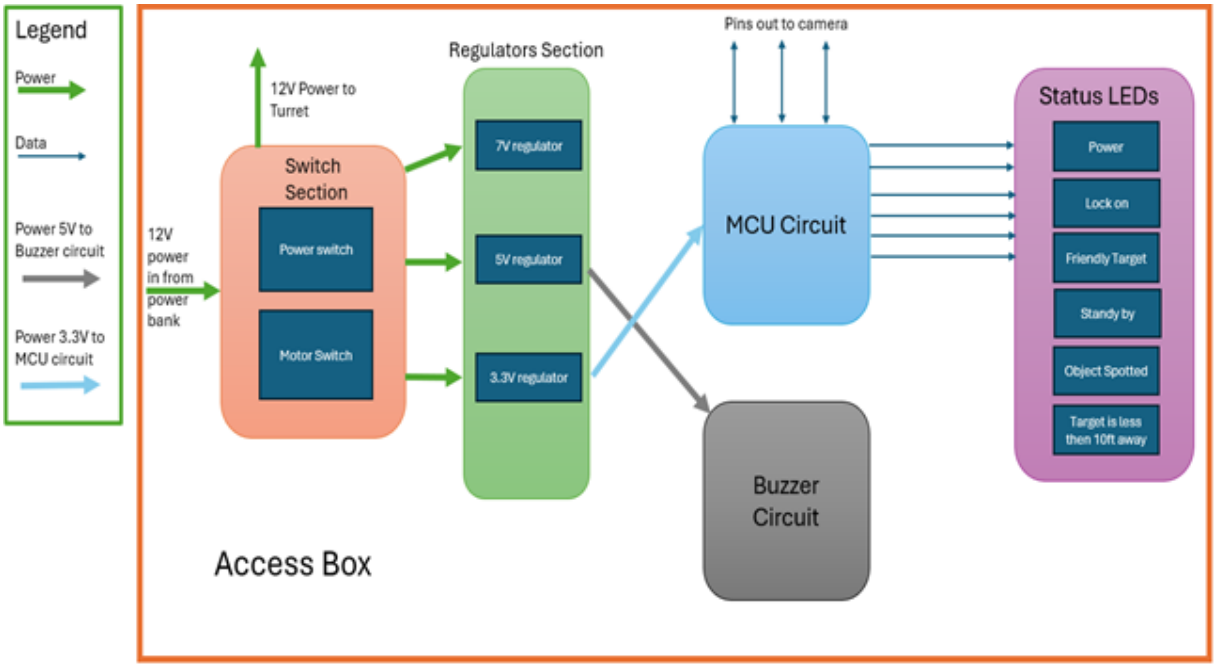
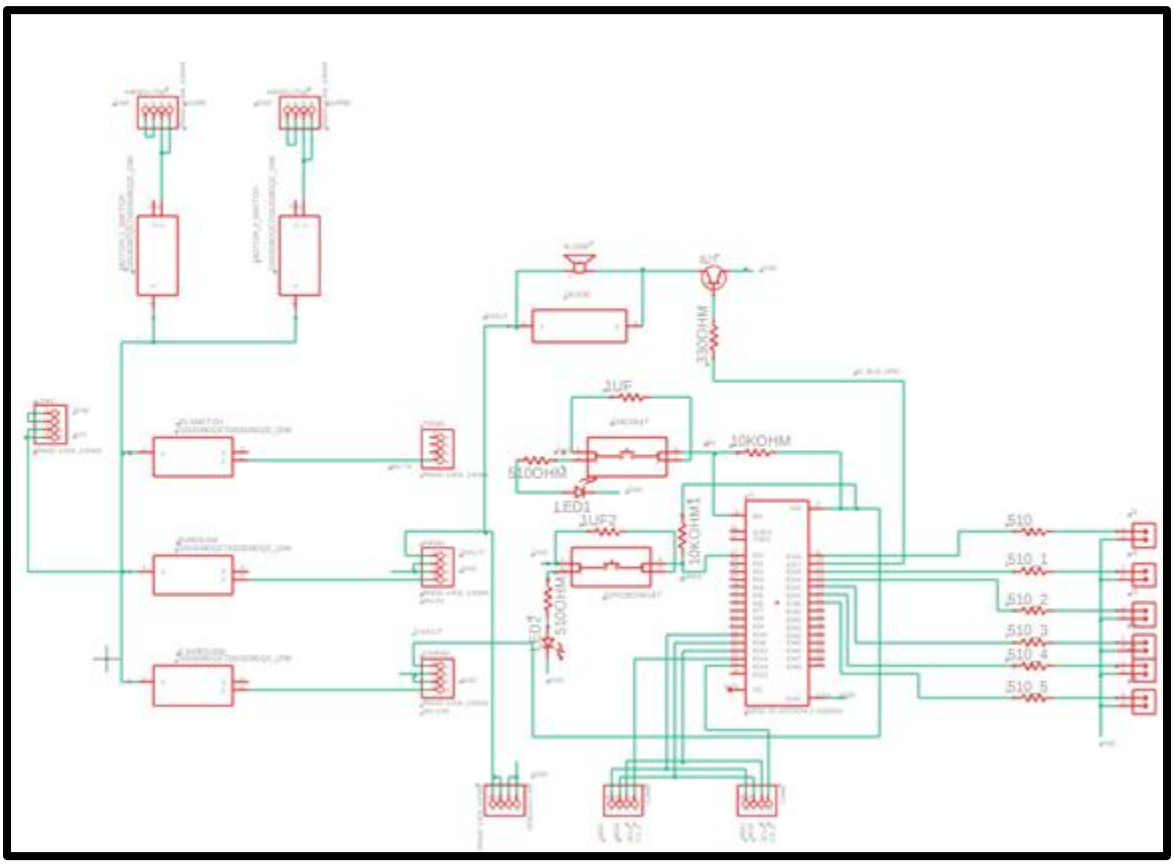


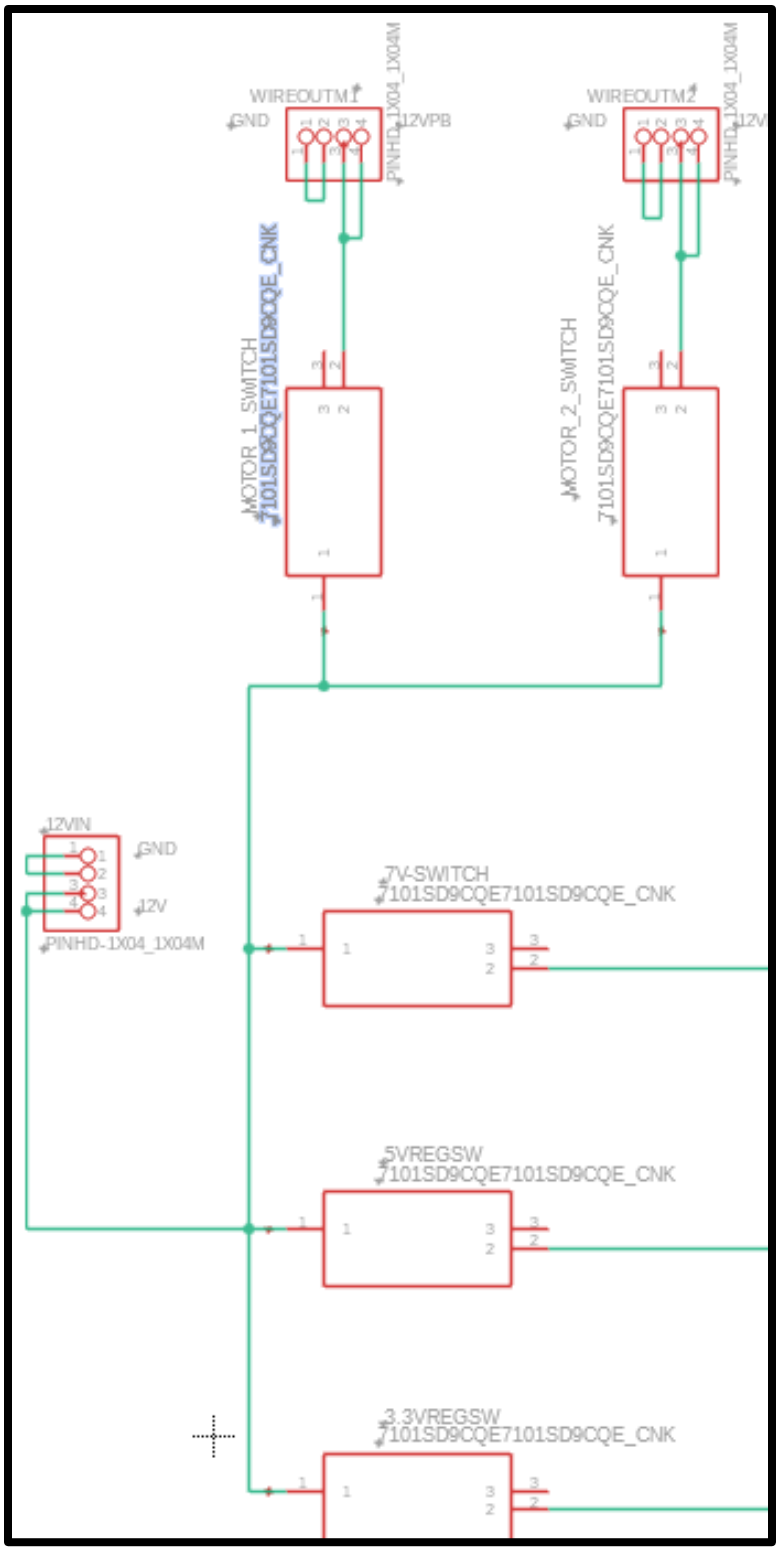
Figure 8 - Full schematic of access box



6.3.1 - Switch section

Let’s start with the switch section for our first circuit. This circuit is meant to simply control the flow of power into the system. Referring to figure xx below, the three right switches towards the bottom control the flow of power into our regulator circuits. This is meant to help with test cases and shielding for debugging in the future. We may only want to test certain sections at one time or another and not power the rest of the system. So, we split up power to help with scenarios such as those. We also have multiple switches in case of needing to shut off the system we can do this quickly and only turn off portions of the circuits if needed. We also split up the switches to help with the handling of current into our system. Each switch is rated for 5A, and that is just about the current draw of access box. To help with the headroom and to stay safe, breaking up the regulator's power with its own dividual switches made more sense. A more robust and cleaner way is to only use one switch to power them all, but we are taking a safer, more sectional way of things as we have with the entirety of this project. The top two switches will be controlling the motors over at the turret. We plan on having the switches on the access box again so we can have a centralized control box instead of needing to have one person at our access box and one person at the turret. Due to the higher current pulled power the motors, servos, and ESC again it makes more sense to break up the switches so we can stay closer to the rating of the switch of 5A. We will need relays as well, which will be discussed further in the turret PCB section of this chapter.

Figure 9 – Switch circuit

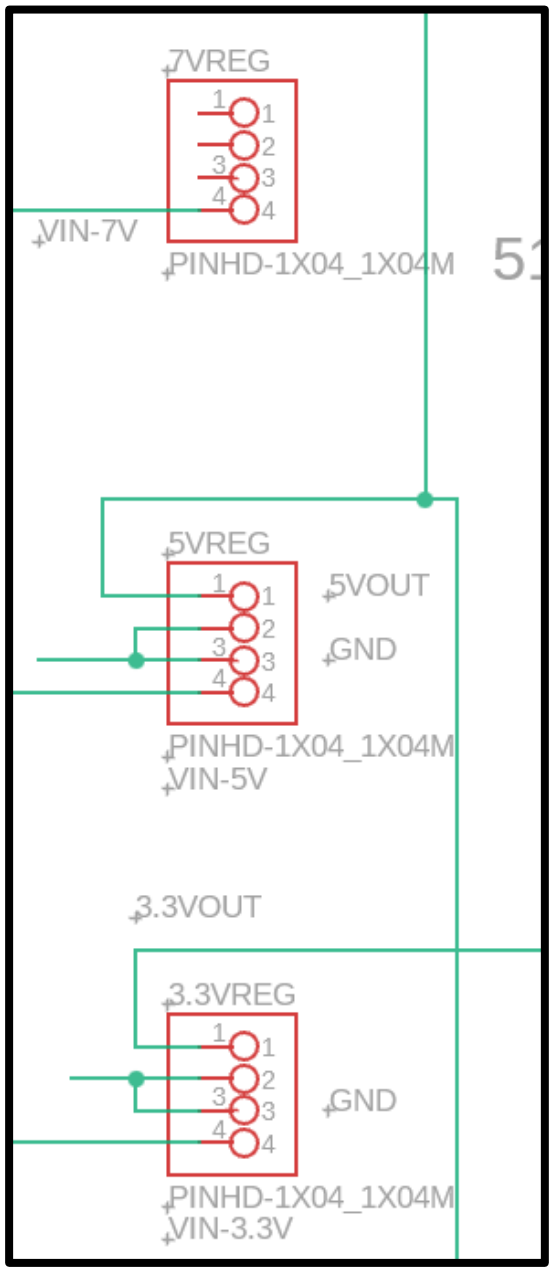


6.3.2 - Regulator Section

As discussed in the regulator section, we separated the regulators from our board to help with costs in case issues were to arise. In this portion of our access box, we have through

holes which will have female headers attached so we can attach the male headers on the regulators to our access box PCB. Pin one will take the 12V from its respective switch, pin 2 and 3 will be connected to ground, and pin 4 will be our expected output voltage for each regulator. Below you can see a more zoomed in version

Figure 10 – Regulator section of Access box

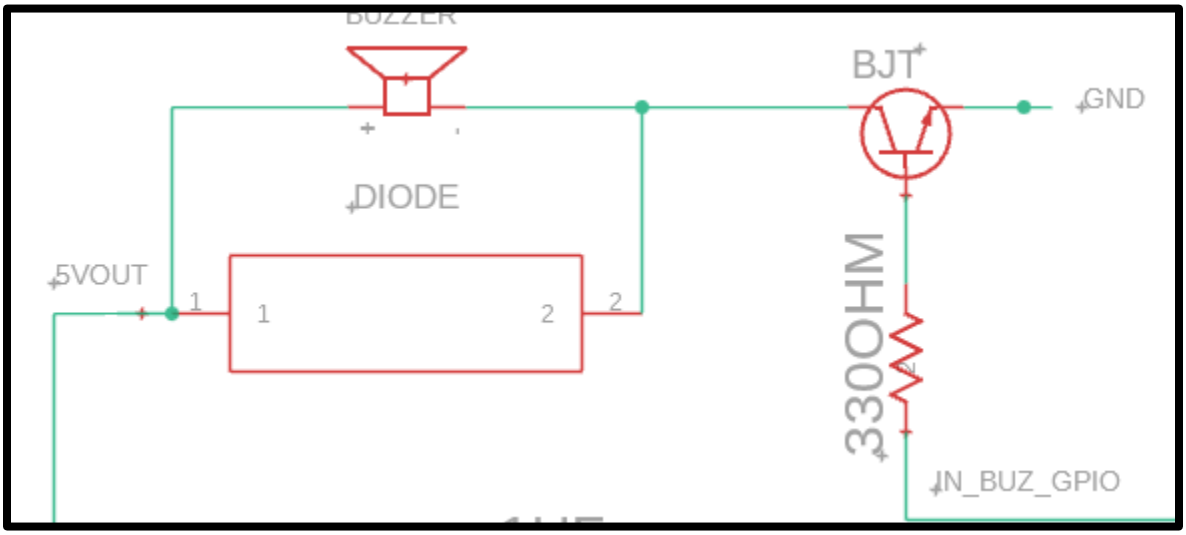


6.3.3 - Buzzer Circuit

Our buzzer is here for mission critical alerts to the user. This will be saved for actions such as enemy lock on achieved or when a new target has entered the area. This buzzer

produces a magnetic field so due to this we must use the circuit that is given in the datasheet based on the buzzer we choose. It has a few components such as the buzzer itself, a diode, a MOSFET transistor, and a resistor of 330 ohms. The reason for the buzzer is to produce the sound we want during these alerts as explained before. We have what's called a flywheel diode which is there to help counteract the magnetic field that is created by the buzzer when it switches from the on to off position. The transistor acts as a switch, and the 330-ohm resistor is to help protect the transistor from current being too high. 5V is fed into the positive terminal of the buzzer and a GPIO pin is fed into the 330ohm resistor which is connected to the base of our transistor. The emitter is then connected to ground, completing our circuit. Our input must be a square wave, aka we must turn on the GPIO that our buzzer is connected to, add a delay, then turn it low and loop to get the buzzer sound. We cannot just apply a GPIO pin and put it in a constant high position like an LED can handle; this signal must be in the form of a square wave.

Figure 11 – Buzzer Circuit



6.3.4 - MCU circuit

In our project, we are not allowed to use a development board which means we need a specific way to program our chips. The ESP32 we have chosen has a few different ways to achieve this however we have decided to use this circuit to do that. As we can see, our circuit has switches on the ESP32 along with some LEDs. These switches are here so we can set the ESP32 into programming mode, which will allow us to flash code from the development board to the bare chip so we can update our design as time goes on. We need two switches in this case, one for the enable signal and one for the GPIO0 pin. The GPIO0 pin will act as our BOOT button which in tandem with the enable signal will set the esp32 into programming mode. It will stay in this mode until code has been sent to the bare chip through RXD0 and TXD0 which will be connected to similar pins on the development board of the ESP32. We have pulled the switches high using a pull up

resistor and have a status LED to show that these have been pulled up. We will be using surface mount LEDs here instead of through holes since we don't need to see the lights when the cover of the access box is on. We will take the board out to reprogram it when needed as the point of the project is to just flip a switch and it be self-sufficient from there with predetermined targets. To put our ESP32 into programming mode, we will hold the enable in a low position and flip the switch for GPIO0 from high to low and back to high which puts it into our programming mode so we can change the behavior of the system as needed.

Figure 12 – MCU circuit

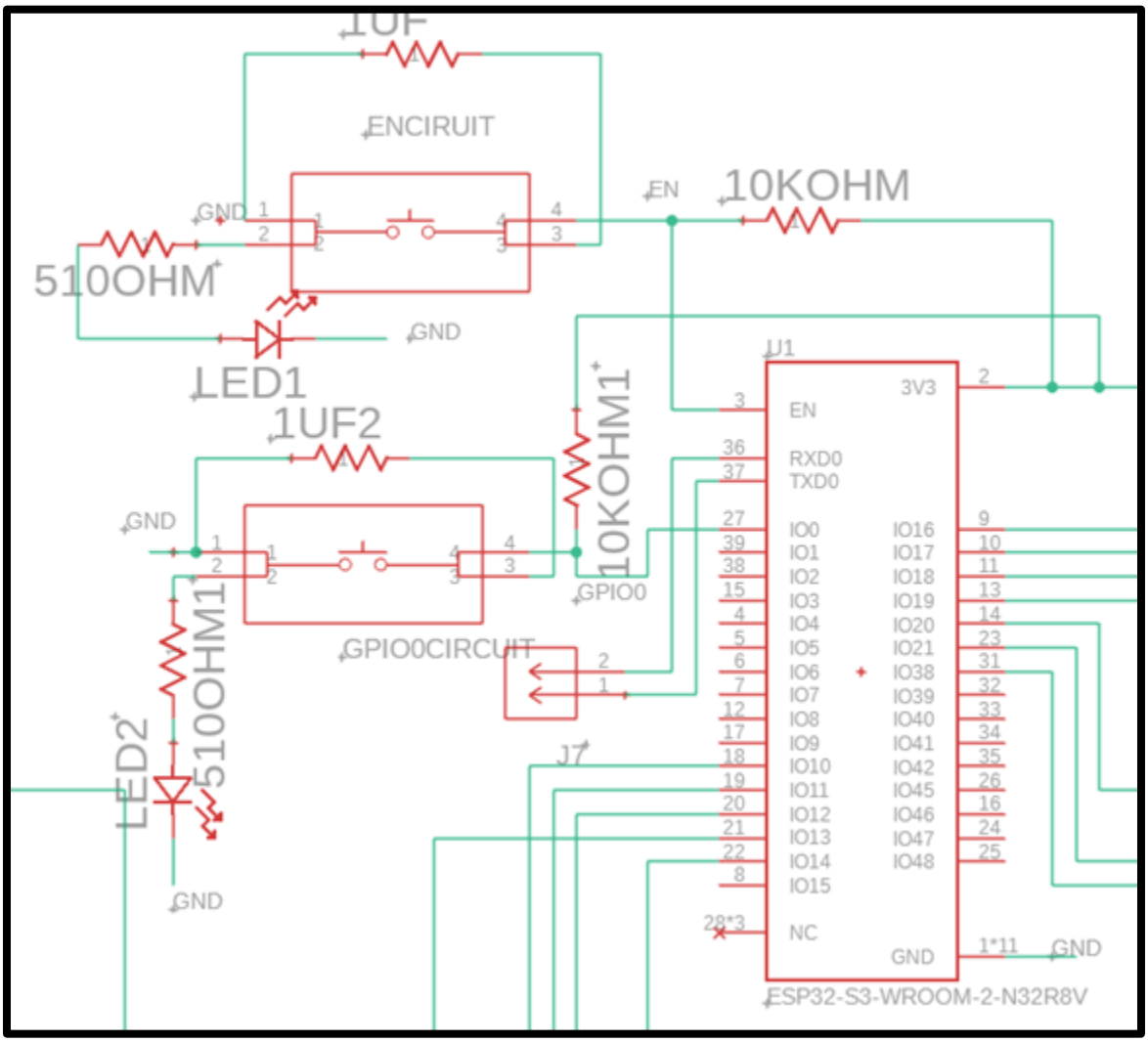
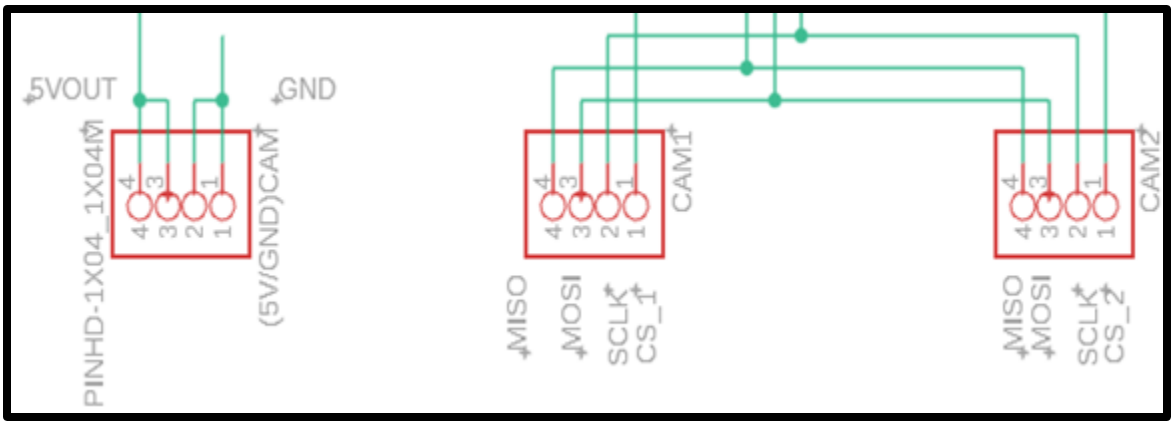


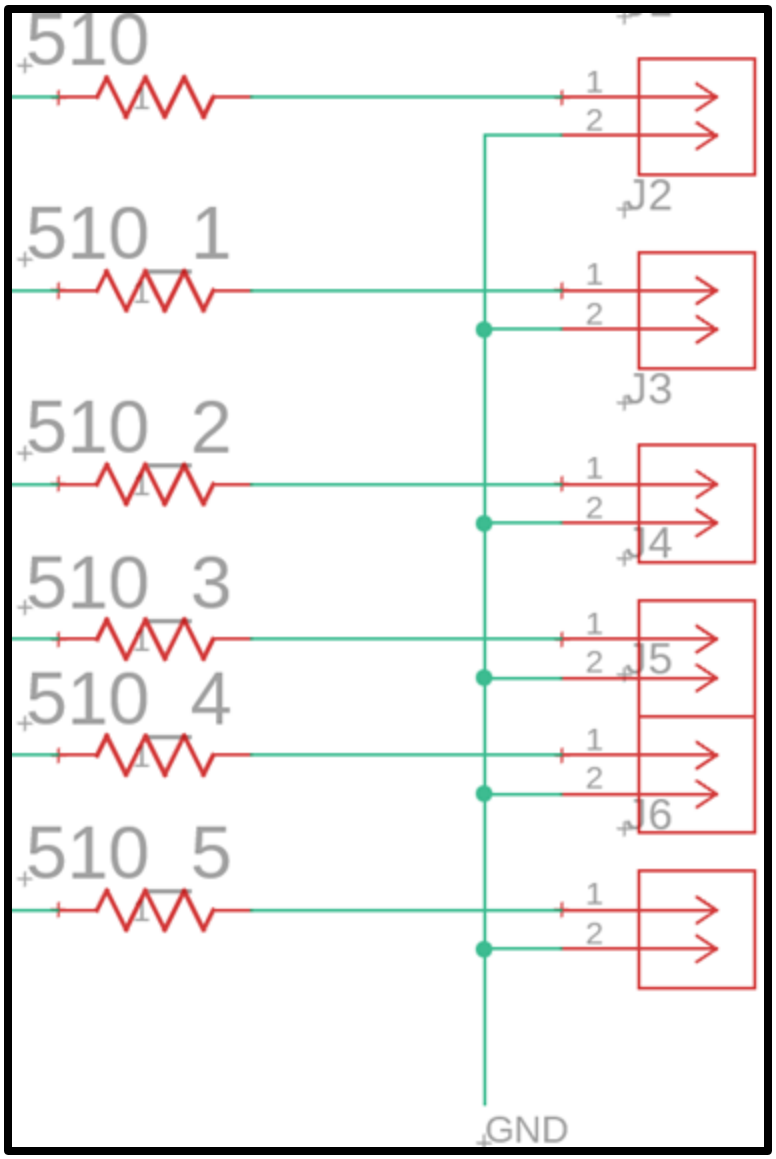
Figure 13 – Pins for cameras



6.3.5 - Status LED section

Our last section is the status LED section. This section is meant to serve as the main way to alert the user as to what is happening. We plan on having at least 6 LEDs to show a few different scenarios. One will be for power and knowing the power bank voltage is received before our switches have even been turned on. One LED will be for when our turret has been locked on to a target. At this moment our buzzer circuit will trigger as well. We want to be able to distinguish between enemies and friendly targets, so we will have another LED for if the camera is tracking a friendly target or an enemy target. We will break this up into a few LEDs if an enemy is spotted. First, we will turn on an LED if an object is in range of detection, next we will then identify a friend or enemy. If it's an enemy, no LED will turn on until the target is within 10 feet, at which point the camera will then send data to the turret to start its lock on process. Our last LED is a standby signal to show that power is in the system; everything is operational, but no objects are within the cameras field of view. Our LEDs are plugged into our two-pin female header and have a current limiting resistor attached to GPIO pins from the ESP32.

Figure 14 –



6.4 Power Delivery / Electrical Power System

The turret is powered exclusively from mains via a 12V, 50A industrial enclosed supply (Mean Well LRS-600-12). Eliminating batteries simplifies compliance and removes charging risks while giving ample headroom for simultaneous flywheel spin ups. The 12V bus enters the access box through a fused inlet and lands on a relay power board that performs high current switching, branching, and protection. From there, the bus splits into three paths: 1. two independent branches to the BLDC ESCs each with its own inline fuse and bulk capacitors, and 2. a regulator branch for auxiliary rails.

Two point of load regulators derive the low voltage rails. A synchronous buck converts to 7.0 to 7.4V for the high torque servos (SF3218MG) and another synchronous buck provides 5V for logic, camera, and the aim laser. Rails are monitored with simple indicator circuits so that a technician can verify power state at a glance. To preserve bus stability during motor events, each ESC input is locally decoupled with 2x 2200 to 3300

µF low ESR capacitors and the 12V trunk is clamped with a TVS diode (approximately SMBJ16A) near the supply entry. This arrangement keeps the 12V rail stiff during simultaneous ramps while isolating digital loads from motor transients..

Bring-up behavior

On power up, only logic rails energize. Once the access box MCU finishes self checks, it asserts the main relay to feed the ESC/servo bus. Emergency stop will open the relay coil supply and removing power from all actuators. With Soft Start enabled in the ESCs and local bulk capacitance, measured currents are expected to be around 24 to 36A during the first 0.2 to 0.4 seconds when both flywheels spin up simultaneously, and 6 to 10A steady, leaving margin for dart contact spikes and servo corrections.

The sequencing is important because if we tried to power everything at once, the inrush current from the ESCs could cause the power supply to fault or the voltage to dip enough that the microcontroller resets. By bringing up logic first and waiting for it to stabilize, we ensure the system can properly control the relay before high current loads come online. This also gives the microcontroller time to run its startup checks and make sure everything is in a safe state before motors get power.

6.5 Turret PCB

The turret side control PCB is built around an module and handles all the low voltage I/O for sensors and actuators at the head. The module runs from a local 3.3V regulator and uses standard boot control with 10 kΩ pull-ups on both EN and GPIO0. We added momentary RESET and BOOT buttons so you can drop into the ROM downloader when you need to flash new firmware.

For sensing, everything shares a 3.3V I²C bus. The HuskyLens vision module connects to SDA and SCL while getting its 5V on the power pin. The IMU ties into the same bus with CS strapped high for I²C mode, and SA0 sets the address. Keeping sensors on one bus simplifies the wiring and lets us add more devices later without running out of GPIO pins.

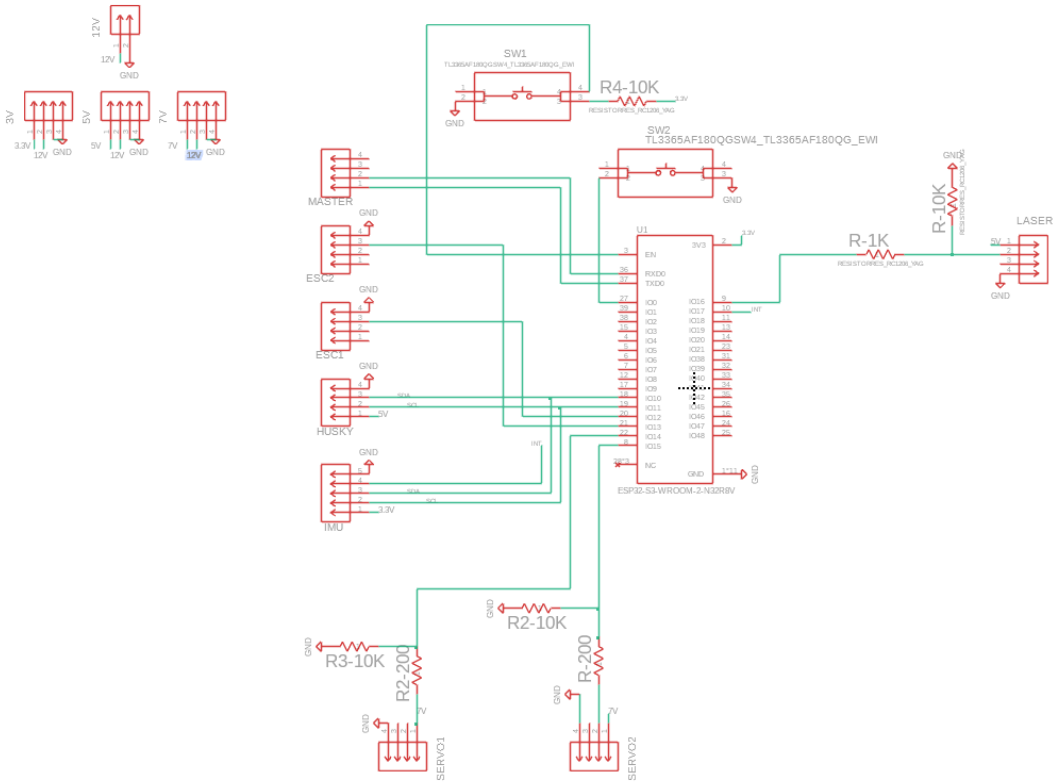
The actuation breakouts are arranged as dedicated labeled headers. The two ESC throttle ports each get an MCU PWM signal that runs through about 220 Ω series resistance, and there's a 10 kΩ pulldown at the connector to guarantee the motors stay idle on boot. The pan and tilt servos follow the same signal treatment and pull their power from the external servo rail. The laser module gets input driven through 1 kΩ series with a 10 kΩ pulldown to keep it off by default.

For coordination between boards, we added a MASTER link that carries TX, RX, and GND across a short harness so the access box controller can exchange commands and status with the turret MCU. This keeps the system modular and lets us test each board independently.

All the connectors use keyed JST-XH for signals with clearly printed legends showing voltage and function. This speeds up assembly and makes it nearly impossible to plug things in backwards during late night debugging sessions.

The grounding scheme ties all logic returns at a single star point to the incoming ground from the access box. E-Stop action is handled upstream on the power board where it belongs. The turret PCB enforces safe defaults through its pull-ups and pull-downs, so even if the firmware crashes, nothing dangerous happens.

Figure 15 – Turret PCB



6.6 Rail Monitoring and Thermal Considerations

Each rail includes a status LED and a simple undervoltage indicator so that undervoltage conditions are immediately visible during tests. Components with notable dissipation like the ESCs, 7.4V buck are mounted and provided vent placement to dissipate heat. Vent paths are oriented to move air past the ESC heat sinks without exposing users to moving parts. Harness drops are reduced by keeping high current runs short and, for the 12V trunk, by using ring terminals or high current screw terminals with proper strain relief.

The thermal management is particularly important for the ESCs since they're switching high currents at high frequency The ventilation slots are positioned to create natural convection flow without creating a direct path to touch moving parts or live circuits.

We actually measured the ESC temperatures during a 5 minute continuous run and saw them hit around 65 to 70°C on the heatsink fins. That's well within their rating but warm enough that you wouldn't want to grab one right after a demo. The metal mounting helps

a lot, in early tests without the metal standoffs the ESCs got noticeably hotter. The enclosure itself acts like a big heatsink, spreading the thermal load across a larger area.

6.7 BLDC Flywheel Drive

The launcher uses two BLDC outrunners (2312, 1250 KV), one per flywheel, each driven by its own Hobbywing Skywalker 40A ESC. The controller inputs are standard RC PWM signals from the turret MCU with a shared ground. Signal lines are routed as twisted pairs with small series resistors (100 to 220 Ω) at the MCU pins to reduce ringing. The ESCs are configured with the LED program card: Brake OFF, Soft/Medium Start, Timing Low, Governor OFF, and LVC disabled (NiMH mode) for PSU operation. These settings minimize inrush, avoid regenerative braking, and keep throttle predictable.

Electrical hygiene focuses on keeping motor current loops local to the ESCs. Each ESC branch includes a 20 to 25A slow blow fuse, and the wiring from ESC to motor uses a compact three conductor connector with locking retention. The system runs both motors simultaneously. The chosen PSU provides plenty of headroom to handle the combined startup current without issues.

The series resistors on the PWM lines are there to damp any ringing on the signal edges. Without them, the relatively long wire runs to the ESCs can act like transmission lines and create reflections that show up as noise on the PWM signal. The 100 to 220Ω range is enough to damp the ringing without affecting the signal levels that the ESCs expect. We tried a few different values during testing and settled on 150Ω as a good middle ground.

6.8 Pan/Tilt Servo Actuation

A pair of SF3218MG digital servos provide pan and tilt. Pan is geared 1:1 to preserve a full 270° sweep while maintaining control authority. Tilt uses 30T:90T (3:1) to multiply torque and hold angle against gravity with reduced buzz. Both servos run from the dedicated 7.0 to 7.4V rail and are commanded directly from the turret MCU's timer PWM outputs. To reduce steady current and oscillation, motion profiles use gentle acceleration/deceleration and the tilt axis is mechanically balanced about its pivot. Cabling to the moving head is dressed with a service loop and kept clear of flywheels.

The servo power rail is separate from the logic rail for a good reason. When servos move under load, they can draw several amps in short bursts, which would cause significant voltage droop on a shared 5V rail. That droop could reset the microcontroller or cause cameras and sensors to behave erratically. By giving the servos their own regulated 7V rail, we isolate their current spikes from the sensitive digital circuits. We learned this lesson the hard way in an early prototype where we tried to run everything off 5V and kept getting random resets when the servos moved quickly.

The 3:1 gearing on the tilt axis was chosen after testing a few different ratios. Direct drive didn't have enough torque to hold the launcher assembly steady, especially at shallow elevation angles where gravity creates maximum moment. The 3:1 gives enough

mechanical advantage that the servo can hold position without constantly drawing current to fight gravity.

The service loop in the servo cabling is important for reliability. As the turret pans back and forth, the cables need to flex without creating stress on the solder joints or connectors. The service loop provides extra cable length arranged in a gentle curve that can absorb the motion without tight bending.

6.8 Relay Power Control PCB

Power switching and distribution are consolidated on the turret PCB. The board carries a high current 12V SPST automotive relay for the actuator bus. The logic header accepts a GPIO from the access box MCU, but the relay coil supply also routes through the E-Stop chain so a press mechanically drops motor power independent of firmware. Keeping this function on a dedicated PCB shortens high current paths, simplifies service, and clarifies safety inspections.

The automotive relay is rated for 40A continuous which gives us margin above our expected peak loads. These relays are designed for harsh environments (under the hood of a car) so they handle the inrush currents and EMI from our motors without problems. The flyback diodes across the coil are critical because when the relay turns off, the collapsing magnetic field in the coil creates a voltage spike that can damage the driver transistor if not clamped.

The relay makes a satisfying click when it energizes, which is actually useful feedback during testing. You can hear when the system transitions from standby to active mode.

The E-Stop integration is done at the hardware level for safety. The E-Stop button physically breaks the connection to the relay coil power, which means pressing it will always cut motor power even if the microcontroller has crashed or the firmware is stuck in a loop.

Chapter 7: Software Design

There is one system for this project that handles all automated tasks and requires no human-machine interface (HMI). This system runs on code flashed to both ESP32s, each of them working together autonomously. The software on these MCUs is the brain that controls all the hardware components, ensuring that the system functions as desired. It is responsible for configuring external sensors, processing data, and controlling motors, LEDs, and the laser diode. The software itself controls multiple different components; therefore, it is important to split the whole system apart into smaller subsystems that are easier to manage and integrate. This project utilizes a two MCU architecture, therefore the software will be split between the two ESP32-S3 microcontrollers as each MCU will have its own tasks to perform. Each MCU is responsible for a separate set of tasks to ensure the performance of the turret system is reliable and consistent under real-time conditions. The first ESP32-S3 (master) microcontroller will be responsible for motor control (pan-tilt mount), peripherals (LEDs and buzzers), HuskyLens AI vision sensor for fine-tracking of target, and communication between the two controllers. The second

MCU is primarily for image capturing and processing through lightweight image detection and transmits that data to the main controller.

Integrating both software components together forms the turret system to ensure that it has a quick response time with accurate tracking abilities, provides timely visual and auditory feedback through several LEDs and buzzers, and each component in the system has the ability to work together seamlessly. This section will outline how each system was designed and integrated into our project.

7.1 - Primary MCU

The primary ESP32-S3 is the central controller of the system as it handles most of the control tasks. Its tasks involve handling servo motors for a stable pan-tilt mount, processing high-level image data incoming from the secondary MCU and HuskyLens, as well as managing the feedback mechanisms. It will also undertake control of safety mechanisms and state transitions (idle, tracking, etc.).

Goals:

1. Control the pan and tilt servo motors using PWM signals and a PID algorithm for smooth and stable motion.
2. Fine tracking using positional data retrieved from the HuskyLens via the I2C communication protocol.
3. Auditory and visual feedback mechanisms that are able to activate within 500ms of detecting a target or an error.
4. Receive target data from the secondary MCU that takes input from two stationary cameras via the SPI interface and incorporates it into the tracking algorithm for the HuskyLens to use.
5. Handle real-time user input for cases such as emergency shutdown.

The tasks are divided between cores (dual-core CPU) using FreeRTOS (Real Time Operating System). It allows for parallel task operation by separating control, feedback, and communication into individual tasks. The tasks are prioritized accordingly with servo motor control and safety monitoring at the highest priority to maintain deterministic timing, and the auditory and visual feedback mechanisms such as the LEDs and buzzers along with I2C communication (HuskyLens) operate at a lower priority but maintains responsiveness. External triggers such as target detection, loss of signal, or lock on handled immediately using interrupt routines. This setup allows for tasks to be performed parallelly while maintaining the stability of the system and operating in real time consistently.

7.2 Secondary MCU

The secondary ESP32-S3 will primarily handle image processing from the two stationary cameras that connect via the SPI interface. The primary function of these cameras is to detect any incoming objects and localize (i.e. obtain coordinates) the targets for processing. These images are run through lightweight TinyML (Tiny Machine Learning) models that are specifically designed for embedded systems, as MCUs don’t typically

have the memory capacity to support heavy machine learning models. Once the images are processed, the coordinate data will be transmitted to the primary MCU for fine tracking and actuation.

Goals:

1. Acquire image data from stationary cameras via SPI communication protocol
2. Scale image resolution or grayscale the image to conserve memory resources.
3. Perform blob detection and color recognition to identify potential targets.
4. Get the results (i.e. bounding box size, coordinates, confidence levels) and send them to the master MCU using SPI, UART, or over Wi-Fi using ESP-NOW depending on which connection gives the quickest data transmission rate with low latency.
5. Capture images and process them in under 2 seconds per frame.

The secondary ESP32-S3 is solely dedicated to vision processing by running asynchronous (cannot capture images simultaneously) image capturing and inference tasks. Image data is captured efficiently using dynamic memory allocated (DMA) SPI data transfers. Since the TinyML models are memory heavy, the MCU is not able to perform too many other tasks simultaneously, therefore it focuses primarily on processing images. It accomplishes this by running the model in a processing loop, while transmitting the data to the primary MCU only when new, valid detections are made.

This significantly reduces the workload of the master ESP32-S3 by offloading the heavy processing tasks to the secondary ESP32. This allows the master controller to focus on maintaining stable and precise motion control and feedback from multiple peripherals.

7.3 - Inter-MCU Communication

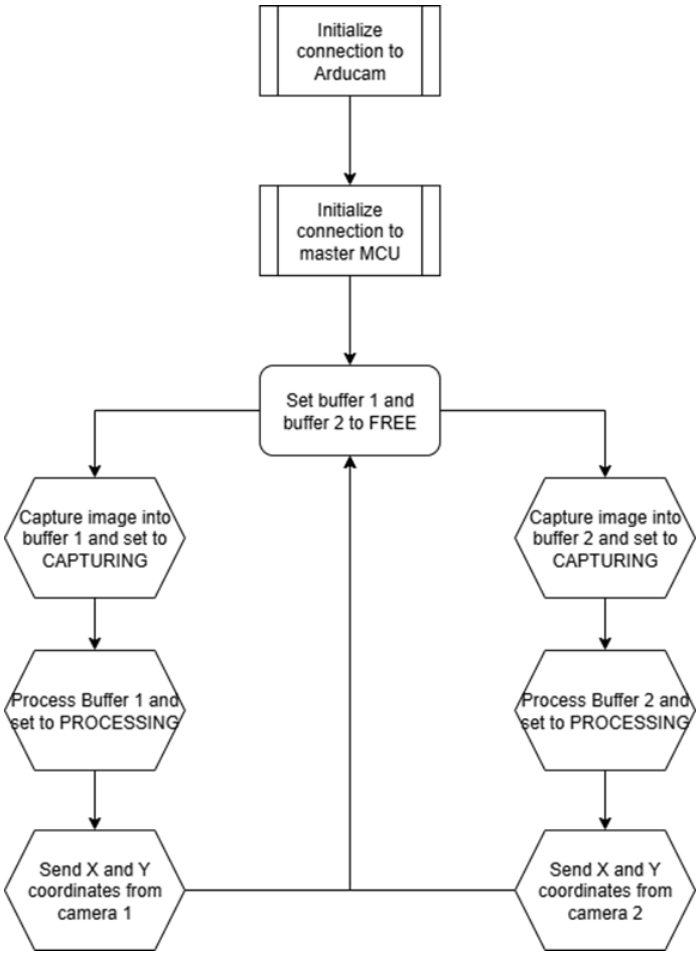
To communicate between the master and secondary ESP32s, there are several options for implementation such as communicating via SPI, UART, or ESP-NOW (wireless). Each option has its advantages and disadvantages and is chosen based on speed and latency. SPI is the preferred interface for its speed and low-latency capabilities and would have the primary ESP32 as the master and the secondary ESP32 as the slave device. The data packets sent between MCUs are structured in a frame format to maintain integrity and synchronizes them:

Field	Description
Header Byte	Start of frame
X-Coordinate	Target’s x position
Y-Coordinate	Target’s y position
Flags	Status bits (i.e. not detected, error)
Checksum	CRC or XOR for validation

The main ESP32 will continuously poll the interface for new data incoming from the secondary MCU. If a valid detection is received, then they are implemented into the loop to track the target. However, if no detection is received before a timeout, then the system will search or go idle until a data transmission is available to be received. This design is reliable and has low overhead when transmitting data between MCUs, allowing for minimized latency.

7.4 Stationary Camera Sensor Software

The stationary camera sensors are used to actively scan the surroundings for potential threats. The cameras connect to the ESP32 via SPI pins, which gives the software full control over the operation of these cameras. After initializing both cameras, the MCU requests an image from a camera. Once it receives this image, the image is processed and outputs an array of bounding box data structures. These data structs are then sent over serial communication to the main ESP32, which will handle the data on its own. After sending the processed image over, the system resets and requests an image from the second camera. The process repeats again, only stopping when the system is shut down. The system relies on strict timing, making sure that only one camera is accessed at a time. By following a pipelined architecture, we can avoid idling the CPU while it waits on IO by processing an image while the second camera begins taking its image. To ensure real-time performance, there will need to be efficient timing control.



7.4.1 Camera Interface Module

The software for the camera interface handles configuration, initialization, and communication. To establish a connection to the cameras, we use the specific ESP32 SPI pins. We can then utilize Arduinos SPI.h library to easily define what pins will be used for SCLK, MISO, MOSI, and CS. After writing a byte to the cameras test register, in our case 0x00, we can verify the connection by reading the register and confirming it contains the sent value.

To configure and initialize the cameras, there are prebuilt libraries for the Arducam Mega 3MP camera that handles the hardware specific instructions. The Arducam_Mega.h file in Arduino provides a simple instruction set that allows us to use defined constants to configure the camera's quality and output data stream format when taking a photo.

After the connection has been established and the camera is ready to take a photo, the software will take a photo, then transfer the raw bytes of each row into a buffer. Once the buffer is full, it will be transferred to a second buffer, then it will take the next row. This new buffer will compare itself with the old buffer and go row by row, performing a

scanline flood fill by comparing pixels with their neighbors. This way, an image only needs to be processed one row at a time, greatly reducing the amount of memory usage.

7.4.2 Image Processing Module

The image processing module performs threshold-based color blob detection. The input to module will be two image buffers, one relating to the current row in the image, and one for the previous row. A predefined RGB value will act as a threshold, and then using a scanline flood fill, we will go over every pixel and check to see if it is above that threshold, meaning it is the expected color, and then checking if its neighbors are the same color as well. This will be used to create bounding box data structures, which will store the top left corner and the bottom right corner, as well as the number of pixels in the box. By knowing the size of the box, we can get an idea for the distance to the object because the object will be of a known size. This will be important later on.

After finding all the bounding boxes, we must choose which box to track. On the first run, the largest bounding box will be chosen, as this will most likely be the closest object. This is as simple as looping through each box and returning the largest. If there is already an object being tracked, then we will remember the box from the previous frame, then find the box that is closest to it. This is as simple as getting to the center of the previous box and using the distance formula to the center of every new box, then taking whichever has the shortest distance. Because our systems desire to track objects closer than farther ones, there will be a threshold applied where if the largest box is a considerable amount more than the previously tracked box, then the largest box will become the main target.

It is important to note that the bounding boxes will be relative to the image taken. Because of this, we need to convert those coordinates to be relative to the camera itself and then convert those coordinates to be relative to a global center. To do this, take the physical width and divide it by the image width in pixels, then do the same with the height, and this will give you the scale of mm/pixel.

$$\text{Scalex} = \text{Physical Width} / \text{Image Width in Pixels}$$
$$\text{Scaley} = \text{Physical Height} / \text{Image Height in Pixels}$$

Then convert this to X and Y coordinates by multiplying the X and Y of the bounding box by the scale. This will be relative to the center of the image, rather than the top left corner.

$$\text{Xcam} = (x - \text{width} / 2) * \text{Scalex}$$
$$\text{Ycam} = (y - \text{height} / 2) * \text{Scaley}$$

We can then get a global X and Y by using a 2D Rigid Body transformation.

$$\text{Xglobal} = \text{Xcam} * \cos(\text{theta}) - \text{Ycam} * \sin(\text{theta}) + \text{Xcam\_origin}$$
$$\text{Yglobal} = \text{Xcam} * \sin(\text{theta}) - \text{Ycam} * \cos(\text{theta}) + \text{Ycam\_origin}$$

These coordinates will be sent over to the main MCU, which will be able to process and use it.

7.4.3 Communication Module

To send the data between the two MCUs, a standard protocol must be followed. Because only an X and a Y value will need to be transmitted, only two bytes of raw data will be sent. To start the transmission, a 2-byte header will be sent that lets the main MCU know that it is receiving packets. The next 2 bytes will be used to send the bounding box center coordinate, and the last byte will be a checksum byte, used to confirm that packets were sent correctly.

Because of how often transmissions occur, time will not be wasted to ensure that the receiver gets the correct data. If the receiver gets corrupt data or there is data loss, then it will simply wait for the next frame to come through, as the next frame will include any updated data. Because of this system, communication between MCUs only needs to be one way.

7.4.4 System Scheduler

To ensure that the system flows smoothly, the scheduling will be handled via state machines and triggers. Instead of the system running based off a set timer, such as every 200 milliseconds it takes a picture and sends the bounding boxes, we will have a main control loop that keeps track of the current state of the process. This way, when one image is sent, another image capture can start immediately. The ESP32 will likely not be able to handle a full pipeline architecture, therefore it will need a double buffer pipeline, where one buffer will take an image and process it, then the next buffer will do the same. While the first buffer is being processed, the second buffer will store its image. Once the first buffer is processed, the data will immediately be sent over. The second buffer can then begin processing while the first buffer stores a new image.

7.4.5 Software Flow

The first step is to initialize the entire system. This will be handled by the camera interface module, and it will set up the system. Once all communications between devices are established, the system will go into its active state. Inside the active state, there will be one main control loop that loops indefinitely. This loop will check the state of the buffers, then perform the appropriate action. The buffer can in one of three states: FREE, CAPTURING, PROCESSING. The loop will first check if buffer is free, and if so, it will start an image capture and move the state to capturing. The image capture will take time, and is done entirely by the hardware, leaving the CPU free to continue processing. Once an image has been captured, which will be detected by a control register in the camera, the buffer will be sent to the processing state. The loop will then check if a buffer is in the processing state. If so, it will call the processing function to get the bounding box; it will then immediately begin sending the data to the main MCU, eliminating the need for a sending state. This is because we want to avoid a scenario where buffer 1 is ready to be

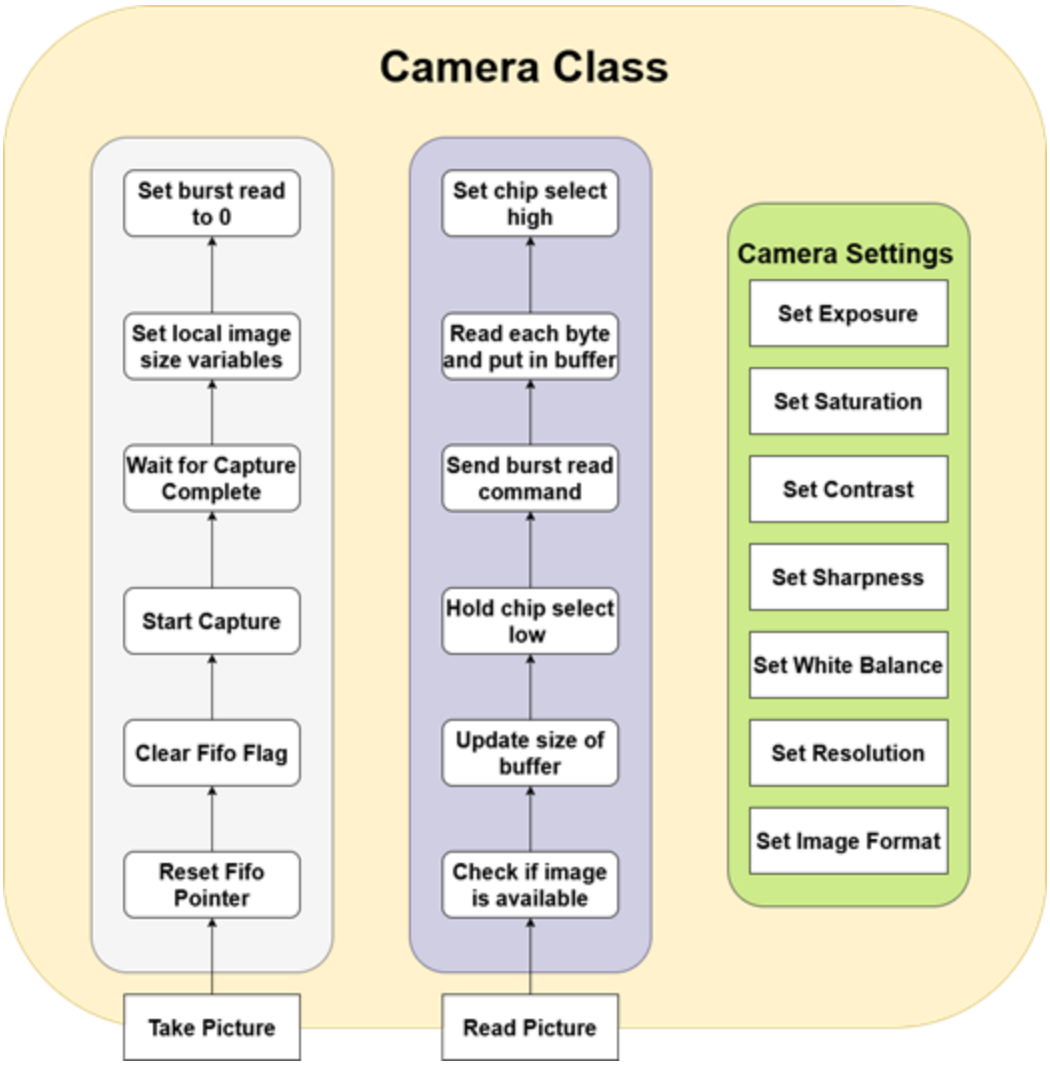
sent, but the CPU is processing buffer 2. After the data has been sent over, the loop will restart.

This flow will not be broken and cannot go in different paths. Additionally, because of the nature of the pipeline, it is not guaranteed that the images will be sent from different cameras in every frame. This is expected, and the main MCU won't care where the coordinates came from.

7.4.6 Camera Class

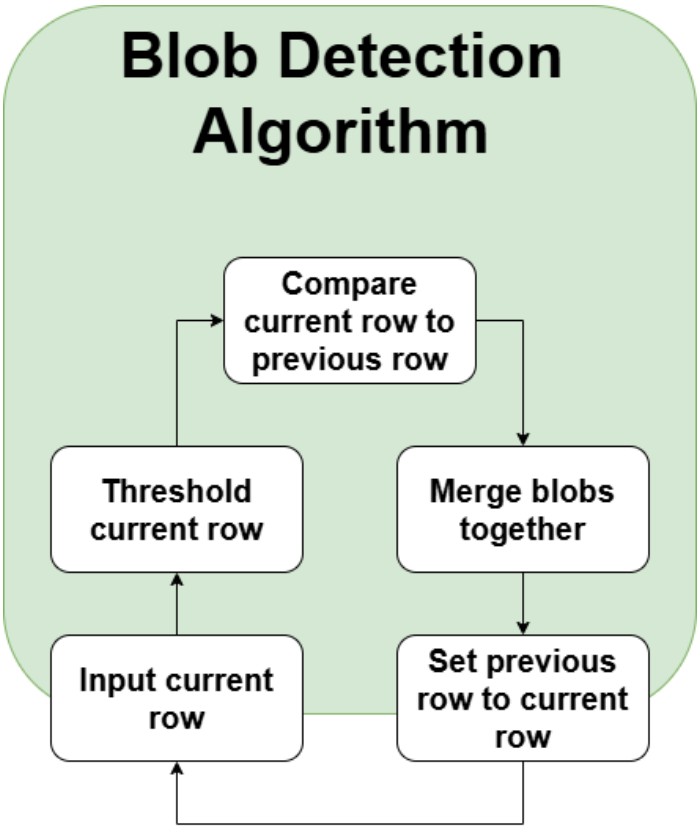
The camera class is used to interface with each stationary camera. It is fully independent of any other class or module. The camera can easily be configured using the different setters that can change its exposure, saturation, sharpness, contrast, white balance, resolution, and image format. When creating a camera object, the only required input is the chip select pin the SPI camera is attached to. Apart from the configuration functions, the two public functions that will be called is takePicture and readPicture. The takePicture function will set internal camera registers to the appropriate values and wait until an image has been captured. There will be no input or output to this function as everything is stored on the camera hardware. The readPicture function will take in a buffer and its size. It will then write the exact number of bytes to the buffer and increment the internal buffer read pointer. The output from this function will be the number of bytes written to the buffer, as this number can sometimes be less than the size of the buffer.

This class is fully independent from the rest of the code, meaning if requirements change for the system, this class will be unaffected. The sole purpose of this class is to take an image and read the image. It is the responsibility of different classes to determine what to do with that image and how to manipulate the output data.



4.7.8 Blob Detection Algorithm

The blob detection algorithm as defined earlier will go row by row, updating a global registry of all blobs as it passes through the image. There is a static table that tracks all blobs in the image, assigning them a unique integer label. Before assigning the label, a row must first run through the threshold function. Then the pixels that passed the threshold will be compared to the previous row's pixels. Once blobs have been assigned to each necessary pixel, any merge conflicts will be resolved. Then the previous row will be set to the current row and the process can continue again.



The thresholding function will determine how red the pixel is. This function takes in a pixel, which has a red, green, and blue value, then checks if the red value is above 130, while the green and blue values are below 60, or if the red value is more than twice the value of green and blue combined. These two conditions will ensure that red is the dominating color. If the pixel is past the threshold, that pixel will be marked as 1, indicating that it is red. If the pixel is not red, it will be labeled as 0. This will result in a new array, 1/3 the size of the input that signals what pixels will be added to a blob and which ones won't.

After each red pixel has been identified, the algorithm will loop through the current encoded row and compare it to the previous. The previous row will not be made up of 1s and 0s, but it will be made up of integer labels and 0s. These integer labels correspond to that pixel to a blob. The first step is to check if the current pixel is 1, if so, check if the adjacent pixels in previous row have a label. If they do, then set current pixel to that label. If no neighbors have a label, then create a new label and assign it to the current pixel. Then continue with the next pixel and repeat. Each time a pixel is added to a blob, the blob updates its size.

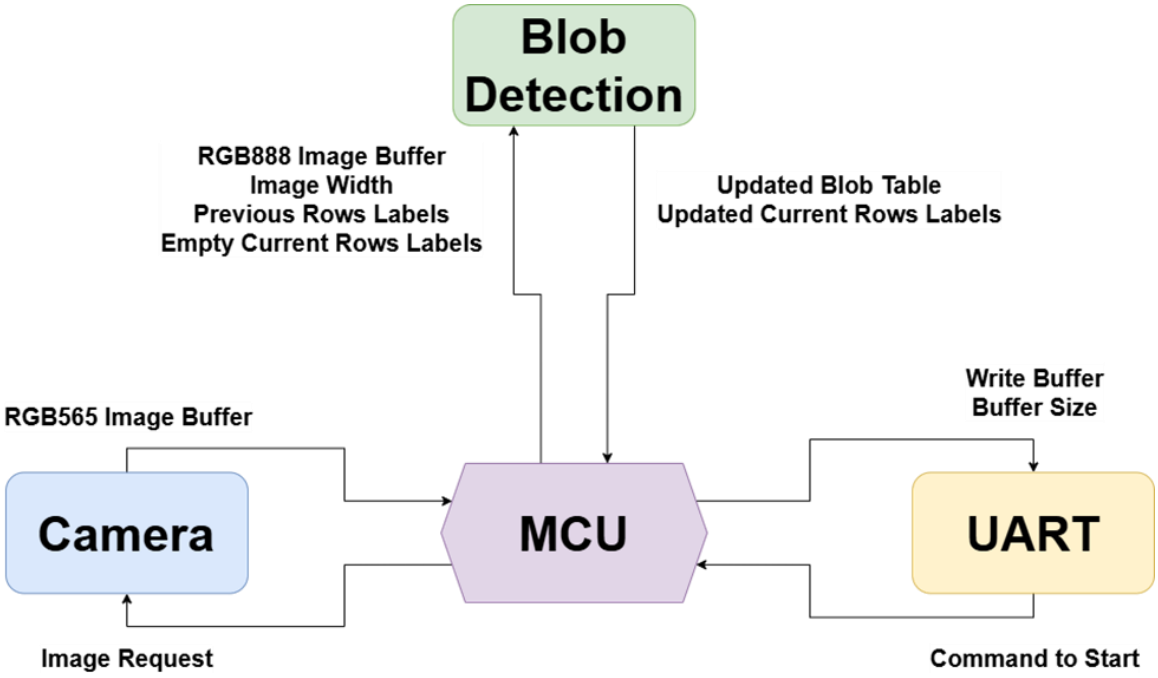
If two blobs are adjacent, then they will need to be merged. This can be accomplished by imagining that each blob is a set. When two sets need to be merged, a union operation can be performed. This operation chooses a set to become the parent of the other set and then updates its size accordingly. This way, when two blobs are combined, the labels don't need to be updated, just the parent of the blob needs to be updated. After the full

image is processed, only the parent blobs need to be accounted for as they will contain the updated dimensions.

Lastly, the previous row of labeled pixels needs to be updated to be the current row. The next buffer can then come in and repeat the process.

7.4.9 System Architecture

The whole system relies on the center MCU to perform the functions. Each module is standalone and has no dependency on any neighboring module. The MCU is the only component that is linked to all. This central hub will be responsible for sending requests to take a picture, read an image buffer, convert the image buffer to rgb888, then pass that image buffer into the blob detection algorithm, then finally output the bounding boxes through UART to any machine ready to receive them.



Ch 10 Administrative content

This section serves as guidance on how to navigate the roles of this project and the cost of the project as well. This is important for such a project as to see how the balance of workload, not just for the individual person but for the whole team, will look and fair out.

10.1 Bill of Materials

Below is our Bill of Materials for this project and the expected expenses for each item. We plan on the majority of our costs coming from PCB ordering. The reason for this is due to the high tariffs at the moment and the size of our boards. We are including a cost buffer to also allow for failure of parts or misaligning issues that could appear during this project.

Table 5 – BOM

Subsystem	Part Name	Quantity	Unit	Estimated Price per [$]
Hardware	Microcontroller	1	1	$5.06
Hardware	BLDC Motor	2	1	$57.26
Hardware	High Torque Servo	3	1	$17.99
Hardware	Electronic Speed Controller	2	1	$24.00
All	IMU	1	1	$12.95
Hardware	LEDs	1	50	$7.10
Hardware	Buzzer	1	1	$0.54
Hardware	Laser Diode	1	1	$18.95
Hardware	SG90 Servo	1	1	$3.62
Hardware	Resistors			
Hardware	PSU (Mean-Well 600 12v)	1	1	$80.60
Hardware	Emergency Button	1	1	$5.90
Hardware	5V Regulator	1	1	$0.44
Hardware	Mechanical Misc (bearings,springs)	1	1	$131.00
Hardware	3.3V Regulator	1	1	$0.44
Hardware	PLA Filament	1	1	$17.99
Software	Husky Lens	1	1	$54.90
Software	Cameras (OV5640)	2	1	$11.99
Hardware	PCBs	3	1	$150.00
All	20% Cost Buffer	1		$120.15
Total				**$1,150.11**

10.2 Project job description per person

Table 6 – Job Descriptions

Description	Name
Modeling of power PCB and access box controls	*Nick Martucci*

Modeling of 3D printed turret and turret PCB	*Carlos Garcia*
Functionality and compatibility of PCBs working	*Nick Martucci and Carlos Garcia*
Choosing MCU	*Devak Sharma*
Data transfer and saving between sentry cameras and HuskyLens	*Devak Sharma and Christian Solanet*
Motion tracking algorithm	*Devak Sharma and Christian Solanet*
Encoding of LEDs and button on access box and any other peripherals	*Christian Solanet*
Modeling of hardware to hold cameras	*Nick Martucci and Carlos Garcia*
Modeling of access box	*Nick Martucci and Carlos Garcia*
Motor Controls	*Devak Sharma*

10.3 Milestones for SD1 and SD2

Table 7 – SD1 milestones

Project idea and initial research	**Fall Week 1 to Week 3**
Divide and Conquer	Fall Week 2 to Week 3
Deep Research into parts and specs	Fall Week 3 to Week 5
Start drawing schematics and working on initial code generation	Fall Week 5 to Week 8
Midterm report	Fall Week 3 to Week 8
Start Testing physical breadboard and initial code	Fall Week 8 to week 12
Start PCB design work	Fall Week 12 to Week 14
Final Report	Fall Week 8 to Week 16

Table 8 – SD2 milestones

Fully working Breadboard prototype with at least initial laser diode detection of target	**Spring Week 1**
Order initial PCBs	Spring Week 1
Start soldering on components	Spring Week 3
Test all stress cases see how far range can work and redesign PCBs if needed	Spring Week 4 to Week 8
Final PCB Design	By Spring Week 10
Final Demo	Spring Week 16

Appendix

Appendix A - ReferenceS

C-RAM research - Project Convergence: Revolutionizing Targeting in Large-Scale Combat Operations

UCF Logo Picture - Ucf Logo, Identity, University, Crest, Emblem PNG Clipart HD

Espressif Systems, “ESP32-S3 Series Datasheet,” Version 2.0. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf

Microchip Technology, “ATmega328P 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash,” Datasheet. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

Raspberry Pi, “RP2350 A Microcontroller by Raspberry Pi,” Datasheet, 2025-07-29 [Online]. Available: https://datasheets.raspberrypi.com/rp2350/rp2350-datasheet.pdf

DFRobot, “HuskyLens AI Machine Vision Sensor,” DFRobot Wiki. [Online]. Available: https://wiki.dfrobot.com/HUSKYLENS_V1.0_SKU_SEN0305_SEN0336

Past Project Nerf dart turret – brushless – WiFi – WIP. Printables. , from https://www.printables.com/model/338574-nerf-dart-turret-brushless-wifi-wip

KEB America, “Stepper motor vs. servo motor: Which is right for your application?,” *KEB America Blog*, 2023 https://www.kebamerica.com/blog/stepper-motor-vs-servo-motor-which-is-right-for-your-application/

OMC-StepperOnline, “Brushless DC motors vs. stepper motors,” *StepperOnline Support*. https://www.omc-stepperonline.com/support/brushless-dc-motors-vs-stepper-motors

Anaheim Automation, “Top 3 motor technologies for compact machines: Stepper vs. servo vs. BLDC,” *Anaheim Automation Blog*, 2025. https://anaheimautomation.com/blog/post/top-3-motor-technologies-for-compact-machines-stepper-vs-servo-vs-bldc

Texas Instruments, “WEBENCH® Power Designer Tool,” Texas Instruments, [online]. Available: Power Designer. [Accessed: Sept. 18, 2025].

Asking AI about LEDs and buzzer components - Microsoft Copilot: Your AI companion

Allied Motion – Choosing Between Brush and Brushless DC Motors https://www.alliedmotion.com/choosing-between-brush-and-brushless-dc-motors/

Aspina Group – Advantages of Brushless DC Motors
https://eu.aspina-group.com/en/learning-zone/columns/what-is/019/

RCexplained – The Difference between Analog and Digital RC Servos
https://www.radiocontrolinfo.com/the-difference-between-analog-and-digital-rc-servos/

Industrial Automation Co. – Analog vs Digital Servos – Which Is the Better Option?
https://industrialautomationco.com/blogs/news/analog-vs-digital-servos-which-is-the-better-option

JouAV – Brushed vs Brushless DC Motor: What's the Difference?
https://www.jouav.com/blog/brushed-vs-brushless-motor.html

Mean Well – LRS-600 Series (600 W) Single Output Switching Power Supply Datasheet
https://files.trcelectronics.com/datasheets/lrs600.pdf

IEC – IEC 60825-1: Safety of Laser Products – Part 1: Equipment Classification and Requirements (Edition 3.0)
https://downloads.regulations.gov/FDA-2017-D-7011-0010/attachment_1.pdf

FDA – Laser Products – Conformance with IEC 60825-1 Ed. 3 and IEC 60601-2-22
https://www.fda.gov/media/110120/download

MGI Speedware – 40-40 Relay Specifications V2.0 (Motor-Load Rating Example)
https://mgispeedware.com/wp-content/uploads/2018/07/40-40-Relay-Specifications-V2.0.pdf

TE Connectivity – V23134 Series Power Relay Datasheet
https://www.mouser.com/datasheet/2/418/5/NG_DS_V23134-X0000-A001_0314_134_0314-844411.pdf

Digi-Key – Panasonic CB1-12V Automotive Relay (12 V, 40 A) Technical Summary
https://www.digikey.com/en/products/detail/panasonic-electric-works/CB1-12V/646971

Digi-Key – Adafruit 650 nm 5 mW Laser Diode Module (Red)https://www.digikey.com/en/products/detail/adafruit-industries-llc/1056/7035009

Datasheets for regulators

TPS56424x 3-V to 16-V Input Voltage, 4-A Synchronous Buck Converter in a SOT-563 Package datasheet

TPS56524x 3-V to 16-V Input Voltage, 5-A Synchronous Buck Converter in a SOT-563 Package datasheet (Rev. A)

TPS56x210A, 4.5-V to 17-V Input, 2-A, 3-A Synchronous Step-Down Voltage Regulator In 8-Pin SOT-23 datasheet

TPS56623x 3-V to 18-V Input, 6-A Synchronous Step-Down Voltage Regulator datasheet (Rev. B)

LMR60440 3V to 36V, 4A , Synchronous, Buck Converter Optimized for High Power Density, Low EMI, and Low Output Capacitance datasheet

TPS6213x 3-V to17-V, 3-A Step-Down Converter In 3-mm × 3-mm QFN Package datasheet (Rev. F)

TPS56x200 4.5-V to 17-V Input, 2-A, 3-A Synchronous Step-down Voltage Regulator in 6-Pin SOT-23 datasheet (Rev. E)

Datasheet - ISM330DHCX - iNEMO inertial module with embedded Machine Learning Core: always-on 3D accelerometer and 3D gyroscope with digital output for industrial applications

PPTC021LFBN-RC Sullins Connector Solutions | Connectors, Interconnects | DigiKey

SSQ-102-03-G-S Samtec Inc. | Connectors, Interconnects | DigiKey

SL-102-T-39 Samtec Inc. | Connectors, Interconnects | DigiKey

310-13-132-41-001000 DigiKey | Connectors, Interconnects | DigiKey

RS PRO Piezo Buzzer, 3-24VDC 66b5745f84cf99d25b4e2e20f736f74e.pdf

Datasheet for AI-1223-TWT - specs-AI-1223-TWT-5V-R.pdf

CEM-1206S Datasheet - Audio Transducers | Buzzers | Same Sky

KSC1815YTA onsemi | Discrete Semiconductor Products | DigiKey

IPC-2221: The Standard for Printed Circuit Board Design

IPC_2221.pdf

IEC 62368-1 Audio/video, information and communication technology equipment – Part 1: Safety requirements

Overview of NFPA 79

79 2024.pdf

NFPA 79: Electrical Standard for Industrial Machinery (2007)

Standard Multilayer Ceramic Caps.pdf

Espressif_Systems_esp32-s3_datasheet_en.pdf

ESP32-S3R16V Espressif Systems | Mouser

https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/6644/ESP32-S3-WROOM-2-N32R16V.pdf