S.T.A.R.
SPOTLIGHT TRACKING WITH AUTOMATED RECOGNITION

Group 7 Authors

Gage Anderson	**Travis Nguyen**	**Lucio Villena**	**Jerison Lau**
Optical Engineering	*Optical Engineering*	*Computer Engineering*	*Electrical Engineering*

Review Committee

Dr. Jaesung Lee	ECE	Professor
Dr. Wei Sun	ECE	Professor

Advisors

Dr. Chung Yong Chan	ECE	Senior Lecturer
Dr. Aravinda Kar	CREOL	Professor

Table Of Contents

List of Tables

List of Figures

Chapter 1 - Executive Summary

The Spotlight Tracking with Automated Recognition (S.T.A.R.) system is a senior design project developed at the University of Central Florida by a multidisciplinary team of students in optical, electrical, and computer engineering. The project addresses a long-standing challenge in live entertainment, presentations, and theatrical productions; maintaining consistent, accurate spotlight tracking on performers without requiring a dedicated operator or expensive lighting console.

Traditional stage lighting relies on human operators or complex, high-cost control systems, both of which introduce limitations in precision, repeatability, and accessibility. S.T.A.R. presents an innovative solution by combining computer vision, real-time image processing, and motorized optics to create an autonomous moving-light system capable of following a performer dynamically. Using a CMOS imaging sensor and trained machine-learning algorithms, the system continuously identifies and tracks subjects on stage, directing a high-intensity spotlight via a two-axis motorized gimbal. This automation enables smooth, low-latency motion, achieving response times under 250 milliseconds while preserving consistent illumination on the selected performer.

The optical subsystem is modeled after the ETC Source Four ellipsoidal reflector design, a professional industry standard known for its efficiency and image quality. The S.T.A.R. system integrates a biconvex BK7 focusing lens and replaceable light source, initially using an HPL 757 halogen lamp with provisions for an LED upgrade. Motion control is achieved through high-torque servo motors driven by an ESP32-S3 microcontroller, while the Raspberry Pi Compute Module 5 executes real-time image recognition using the YOLO framework within OpenCV. Together, these subsystems enable optical precision, responsive tracking, and wireless user control via a mobile or web interface.

The system is designed for a variety of use cases, including theater performances, concerts, lectures, and live events, where it can autonomously follow presenters or performers with minimal setup. Its small, efficient design allows quick set-up and integration into pre-existing lighting networks for professional venues while remaining affordable and user-friendly for smaller productions and educational environments.

This document outlines the complete design process of the S.T.A.R. system, including the theoretical foundation, hardware and software research, and rationale behind key component selections. It details the architecture of the optical, electrical, and computational subsystems, supported by schematics and simulation data. Standards and design constraints relevant to safety, communication, and performance are discussed, followed by documentation on prototyping, fabrication, and testing. Administrative content such as the bill of materials, cost analysis, and project timeline is also presented. The report concludes with a summary of system achievements, lessons learned, and proposed improvements for future development phases.

Chapter 2 - Project Description

2.1 Background and Motivation

Stage productions, concerts, and presentations rely on spotlighting to direct the audience's attention to the appropriate subject and enhance performance quality. Traditionally, this task can be achieved via two options; a trained spotlight operator manning a heavy, large spotlight, or a costly lighting console that needs a programmer to operate [1]. While effective, these methods present limitations: consoles often cost upwards of $20,000, spot operators add to labor expenses, and manual control can introduce inconsistencies in timing and accuracy.

Advances in computer vision and security applications open new possibilities for automated spotlighting systems [2]. By utilizing imaging sensors, real-time subject tracking, and mobile control platforms, an autonomous spotlight can be made both affordable and reliable. There are many applications and systems where precision human tracking is essential for operation; utilizing these systems will be crucial in developing an accurate system.

The S.T.A.R. project aims to address these challenges. The system uses a CMOS camera and real-time human recognition algorithms to track performers on stage. A motorized gimbal directs the spotlight toward the tracked subject, ensuring consistent illumination. In addition, S.T.A.R. integrates wireless control via a mobile app or remote interface [3], eliminating the need for expensive lighting boards. S.T.A.R. demonstrates the feasibility of creating an accessible, user-friendly spotlight system by combining off-the-shelf imaging hardware, embedded processing, and motor control.

2.2 Existing Products & Related Work

CHROMA: Color Harmonization from Skin Tone Recognition

CHROMA was a previous Senior Design project completed at the University of Central Florida that focused on improving lighting conditions for photography. The goal of the project was to create a smart lighting system that could automatically adjust brightness and color temperature based on a subject’s skin tone. Using a small camera called the Image Capture Unit (ICU), the system analyzed the subject’s image and adjusted nearby LED panels to produce accurate and consistent lighting. This process helped photographers capture more natural-looking images, especially in situations with varying skin tones or inconsistent light.

The CHROMA system used a Raspberry Pi 5 Model B as its main controller and featured several RGBCCT LED panels capable of producing both warm and cool color temperatures. A wireless connection allowed communication between the camera and the light panels, so adjustments could be made in real time. The project demonstrated how combining image processing, adaptive lighting, and color science could improve

photography and make it more accessible to beginners. CHROMA's design approach is an example of how optical systems and embedded electronics can work together to automate lighting control and reduce the need for manual adjustments.

CHROMA could integrate into our S.T.A.R project in later iterations. Having a moving spotlight that could optimize colors depending on skin tone would be a groundbreaking innovation in stage lighting technology. Though CHROMA was designed for lighting panels in photography, it could still have a great impact in theatre, as optimizing the appearance of your performer is crucial. There would be great difficulty during implementation though, as the size of our light engine is significantly smaller than the one that was utilized in CHROMA. The imaging component though, is already built into S.T.A.R

Follow-Me 3D

Follow-me 3D is a tracking and control system used in large-scale productions. Unlike a standalone automated spotlight, Follow-Me is a software platform that integrates with existing moving head fixtures. It works by using cameras placed around the venue to track performers in three dimensions, then translates that data into a DMX signal which is sent to multiple moving lights. Operators can assign fixtures to specific performers, adjust beam parameters, and manage complex stage environments in real time. The system supports multi-fixture tracking, allowing one operator to control dozens of spotlights following several performers simultaneously.

This product displays the effectiveness of tracking systems with automated lighting instruments, but its complexity and price make it not suitable for small-scale usage. S.T.A.R. will fill this gap by utilizing the same camera tracking, while delivering a smaller scale system that can be used by anyone. S.T.A.R will have a single camera built into the unit, rather than needing multiple cameras around the perimeter of the stage. This makes the system much cheaper and easier to integrate.

MAC Viper Profile

The Viper Profile is a powerhouse in the moving light industry. Its mechanical system is a proven model for moving head lighting instruments. It utilizes a motorized yoke design, where stepper motors control both the pan and tilt directions. While it does not utilize any human tracking systems listed above, its intricate mechanics like the adjustable iris and motorized gimbal will be an inspiration of our design elements within S.T.A.R.

2.3 Goals and Objectives

At its foundation, the system aims to detect and track a single moving subject in lit environments, using a CMOS imaging system and real-time recognition algorithms to guide a two-axis motorized gimbal that delivers smooth spotlight tracking with less than 250 ms of latency. Advanced goals build upon this by refining multi-target detection, enabling user selection of subjects, optimizing gimbal motion for silent operation, and incorporating a wireless control interface for calibration and override. Stretch goals

further extend the system's functionality by supporting DMX/Ethernet integration for compatibility with professional lighting consoles, networking multiple units for coordinated coverage, and adapting to environmental challenges such as smoke, fog, or high-contrast lighting. Additional objectives include implementing built-in lighting effects, reducing optical distortion through post-processing, and exploring higher-wavelength detection capabilities such as thermal imaging in the far-infrared range. Together, these goals and objectives provide a clear roadmap for developing a robust, real-time, and versatile autonomous spotlighting system.

2.2.1 Goals

Basic Goals

1. Detect and track a single moving person 25 feet away.
2. Control a motorized gimbal to pan/tilt the spotlight toward the tracked subject's center of mass.
3. Provide smooth, low-latency spotlight movement (<300 ms total system delay).

Advanced Goals

1. Enable multi-target detection from 25 feet away with single-target spotlight focus.
2. Optimize gimbal motion for continuous and smooth operation.
3. Add wireless control interface (web app or mobile) for calibration and override.

Stretch Goals

1. Network multiple S.T.A.R. units for coordinated spotlight coverage.
2. Incorporate environmental adaptation, such as handling smoke, fog, or high-contrast lighting.

2.2.2 Objectives

Basic Objectives

1. Design a multi-lens system using commercial off the shelf components capable of providing a clear image of performers at the specified depth of field.
2. Use human recognition algorithms (i.e. computer vision or ML) to identify and track a single subject and communicate to a gimbal system for pan/tilt control.
3. Validate the system for latency through stage test scenarios with one performer.

Advanced Objectives

1. Refine the vision algorithm to detect multiple subjects and allow selection of which performer to follow by the user.
2. Calibrate the motor control to minimize jitter, noise, and overshoot in gimbal movement.
3. Design and test a mobile/remote app interface for manual override and basic spotlight control.

Stretch Objectives

1. Implement DMX/Ethernet support to allow for integration with existing lighting systems and console control.
2. Incorporate detection of higher wavelengths including that of the Far-Infrared (9µm - 12µm) range in order to provide thermal imaging.

2.4 Description of Features / Functionalities

There are many features and components of the S.T.A.R project that must integrate to create a fully functional, operating system. This section will do two things; show complete schematics of both the optical system and prototype, then explain the unit in its three main components; the rotating gimbal, imaging system, and illumination system.

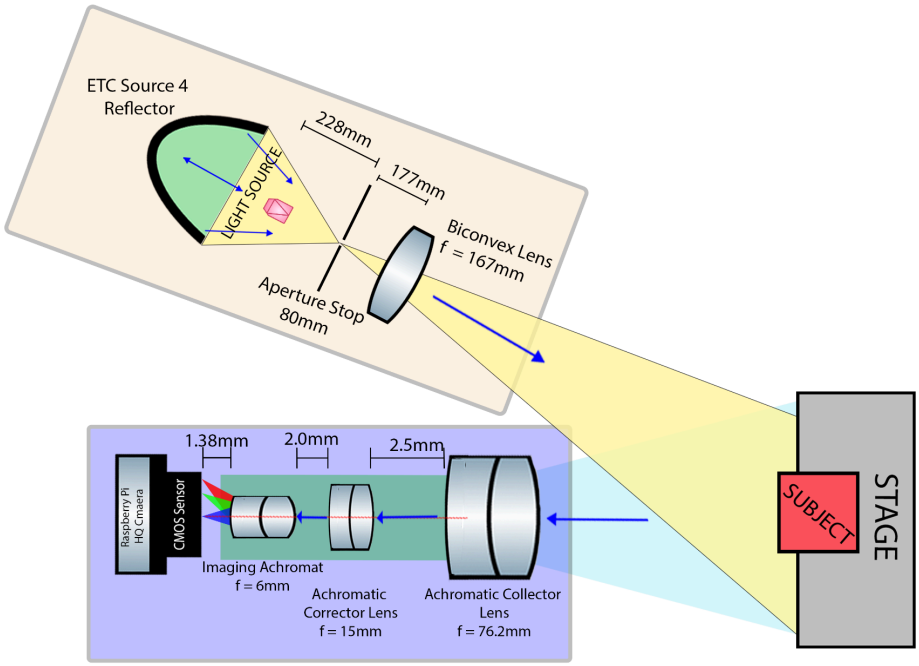


Figure 1.0 Optical Schematic

The figure above demonstrates all of the optical elements in the S.T.A.R system. The light yellow box shows the illumination system, from the illumination source to the focusing lens. The components in the blue box are the imaging system, from the sensor to the collector lens. The blue lines show that the light from the illumination system goes onto stage, and the light that reflects off of the subject (red box) goes back into the imaging system.

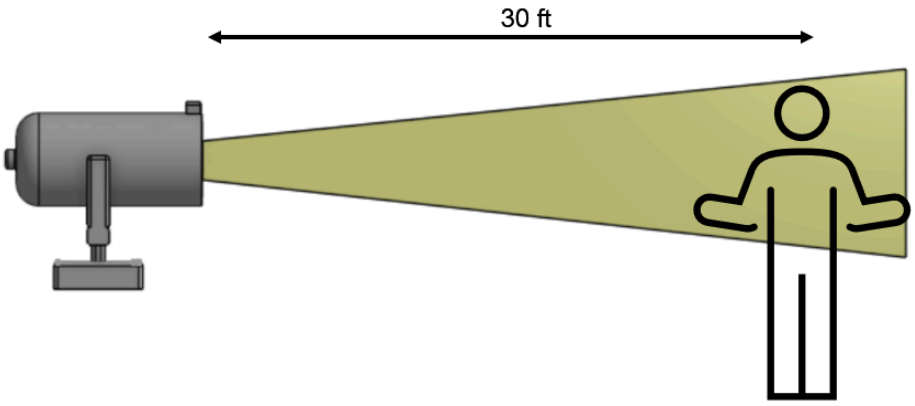


Figure 2.0 Prototype model of S.T.A.R.

The prototype shown above displays the S.T.A.R. motorized spotlight system capable of autonomous human tracking. The goal is to illuminate the entire upper half of the subject from 30ft to allow for hand motions/actions to be seen by the audience. The cylindrical housing contains the light source, reflector, and lens system. Inside the gimbal are servo motors for two-axis movement (pan and tilt), The base houses the control electronics, including the microcontroller, motor drivers, PCBs, and power sources. Our CMOS sensor system is mounted at the top of the output of the system. The system operates as a standalone unit, requiring only power and a simple user interface (remote or mobile app) rather than a traditional lighting console.

Rotating Gimbal

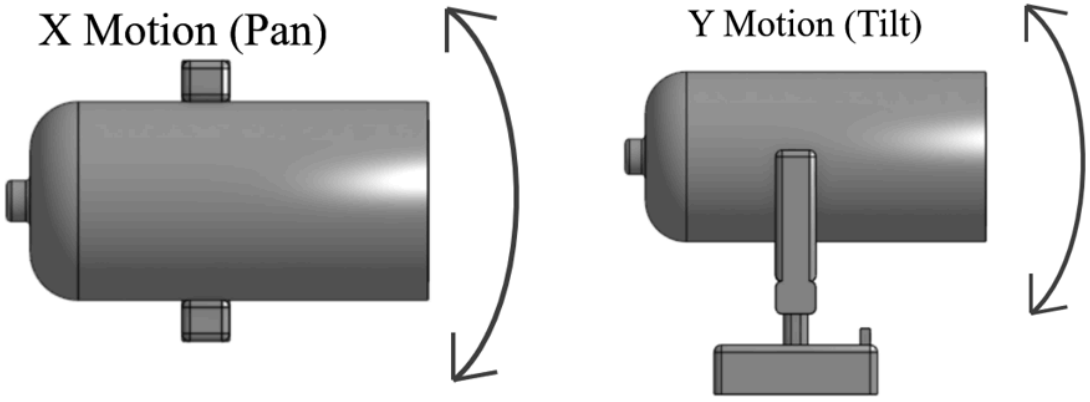


Figure 3.1: Gimbal motion axes of the S.T.A.R. system. The left view shows rotation about the X-axis (pan), and the right view shows rotation about the Y-axis (tilt).

A core feature of the S.T.A.R. system is the motorized gimbal that allows the spotlight to rotate in two separate axes: the horizontal and vertical direction. In professional stage lighting, motorized gimbals are most commonly found in moving head fixtures, such as the MAC Martin Viper. These instruments are some of the most versatile in the entertainment industry, and require the most skill to manipulate. These fixtures use multi-axis gimbal systems not only to provide pan and tilt movement, but also to

manipulate other parameters such as zoom, focus, and beam shaping. By contrast, S.T.A.R. aims to simplify the gimbal system by providing only the most essential element: pan and tilt. The gimbal serves as the method for subject tracking, translating the output of the vision system into physical movement of the illumination system. The design incorporates two degrees of freedom (pan and tilt):

- Pan (X-axis rotation): Allows the spotlight to sweep left and right across the stage.
- Tilt (Y-axis rotation): Adjusts the vertical angle to follow performers as they move closer or farther from the camera's elevation.

Human Detection

A key feature of the S.T.A.R. system is the ability to detect and track humans on stage in real time. This function eliminates the need for a dedicated spotlight operator and ensures consistent, automated coverage of performers. Human tracking will be achieved using a CMOS imaging system combined with computer vision algorithms that identify and follow targets as they move [4]. The camera will be mounted onto the rotating gimbal, so the subject will always be in view of the camera system [5].

To be able to track humans, the system will utilize a machine learning model from OpenCV that's trained for object detection. After taking input from the CMOS imaging system, it will communicate it through the MCU, where an application will be used alongside the system to help users select which human in view of the camera will be targeted by the system.The output of the application should then be able to track the selected human, adjusting the camera movement (pan and tilt) as the human moves [5].

Illumination System

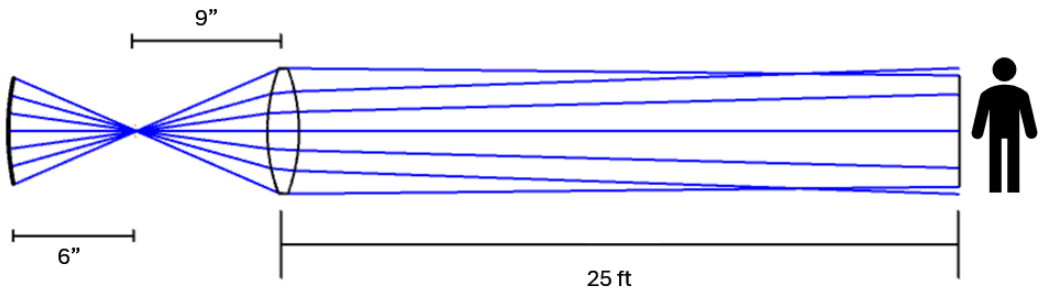


Figure 4 (from left to right) Illumination reflector, gate, and lens; pointing towards subject

The most critical component of the S.T.A.R. system is its illumination system, which is based on the ellipsoidal reflector spotlight (ERS) design (see Fig. 5). Ellipsoidal reflector systems have been a pillar of stage lighting for decades, with the release of the Source Four ERS by Electronic Theatre Controls (ETC) in 1992, setting the industry standard. The ERS illumination system is unique in that it utilizes a combination of lenses and

mirrors - an ellipse shaped mirror focuses the light to a point, and lenses are used to collimate past the focal point. Traditionally, these fixtures use a tungsten-halogen four-filament lamp, which provides a very high output, but also generates significant heat and requires frequent replacement. In contrast, the S.T.A.R. system adopts an LED emitter. Unlike filament lamps, the LED produces a large number of lumens in all directions with far greater efficiency, lower heat generation, and an extended operational lifespan.

2.5 Specifications

Table 1.0: Component Specifications

OPTICAL COMPONENTS			
Component	**Parameter**	**Specification**	**Unit**
Spotlight System			
Ellipsoid Reflector	1. Efficiency 2. Cone Angle 3. Focal Length	1. 90 2. 19 3. 237	1. % 2. Degrees 3. mm
LED Light Source	1. Intensity 2. Color Quality 3. Power	1. 5000 2. 85 3. 155	1. Lumens 2. CRI 3. Watts
Focusing Lens	1. Focal Length 2. Spot Size 3. Material	1. 177 2. 0.5-3 3. BK7	1. mm 2. m 3. NA
Imaging Camera Lens System			
Achromatic Doublet Collection Lens (MgF2 coated)	1. Diameter 2. Focal Length	1. 25.4 2. 76.2	1. mm 2. mm
Achromatic Doublet Corrector Lens (UV-VIS coated)	1. Diameter 2. Focal Length	1. 6.25 2. 15	1. mm 2. mm
Achromatic Doublet Imaging Lens (VIS-NIR coated)	1. Diameter 2. Focal Length	1. 4 2. 6	1. mm 2. mm

Sensor (Sony IMX477 sensor)	1. Pixel Size 2. Megapixels 3. Wavelength Range	1. 1.55 x 1.55 2. 12.3 3. 400 - 700	1. µm 2. Mp 3. nm
ELECTRICAL COMPONENTS			
Component	**Parameter**	**Description**	**Target value & unit(s)**
MCU (ESP32-S3)	Operating Voltage	Runs motor control	3.3V
SoM (Raspberry Pi Compute Module 5)	Operating Voltage	Runs Computer Vision and MCU programming	5V
Motor (Pan/Tilt)	Speed, Standing Torque, Operating Voltage, Stall Current	2-axis servo system	0.12sec/60° @ 14V, 50kg-cm @ 14V, 10-14V, 5500 mA
Motors (Shutter Mechanism)	Standing Torque, Operating Voltage, Stall Current	Illumination shutter system for blacking out light source	21.5kg-cm @ 6.8V, 5-6.8V, 2200 mA
PWM Motor Controller (PCA9685)	Max PWM Outputs, Operating Frequencies	Maximum number of servos that can be controlled, PWM frequency output range	16, Up to 1.6KHz
Power Supply (Raspberry Pi)	Power Output	DC power supply, USB-C connector	5.1V, 15W
Power Supply (ESP32 and Motors)	Power Output	DC power supply	24V, 25A
WAGO Connector (2606-1102)	Rated Voltage, Rated Current	For connecting 24V power supply to PCB	300V, 31A

Table 1.1: Overall System Specifications

Parameter	Description	Target Value & Unit(s)
Tracking Latency	Time delay between performer movement and spotlight response	≤ 250 ms (basic), ≤ 150 ms (advanced)
Tracking Accuracy	Angular error between spotlight center and the selected performer's center of mass	≤ 5° RMS at 10 m
Field of View (FOV)	The angular viewable area that the sensor can image	> 45°
Depth of Focus	The acceptable focus depth that a performer can move on a stage	> 3ft
Tracking Range	Distance at which subject can be reliably detected and tracked	25ft (± 2ft)
Frames Per Second (fps)	How many frames are captured by the camera per second	50 fps @ 1920x1080p
Lighting Conditions	Minimum illumination where tracking is possible	≥ 1000 Lumens
Spotlight Beam Size	Effective beam diameter at 10m	0.5m - 1m adjustable (>5° half cone)
Pan/Tilt Range	Gimbal rotation coverage	Pan: 180° max, Tilt: 120° min
Pan/Tilt Speed	Spotlight movement speed	≥ 180°/s pan, ≥ 120°/s tilt
Operating Power Consumption	Wattage consumed by system during operation (NOT including illumination source)	~ 611Watts
System Weight	Total weight of system (NOT including illumination source)	≤ 20 lbs
System Cost	Total cost of prototype build (not including redundancies)	≤ $800 prototype

2.6 Hardware Block Diagram

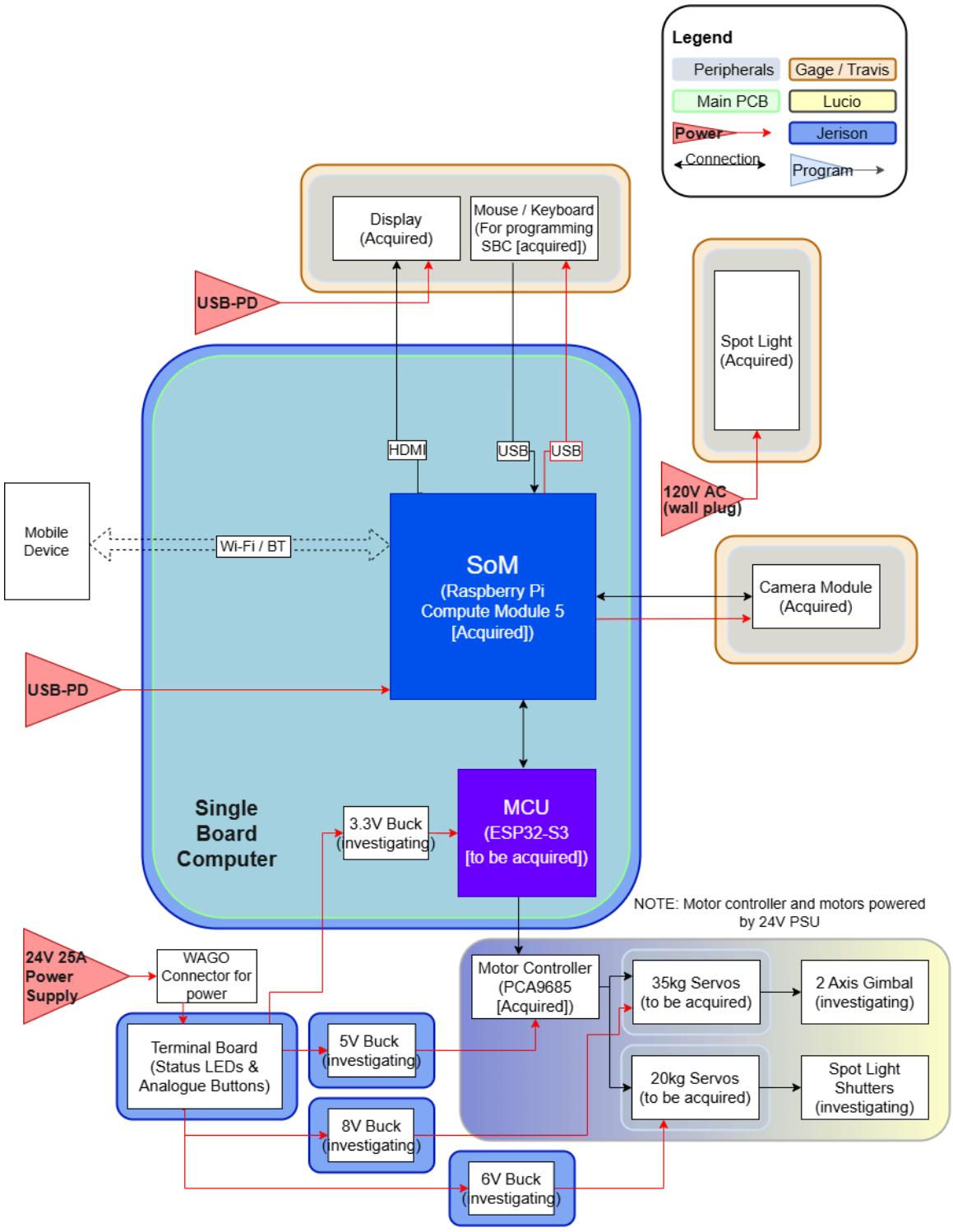


Figure 5.0: Hardware Block Diagram

Does NOT include lens system. Please refer to section 2.9 for diagrams of optical components.

2.7 Software Diagram

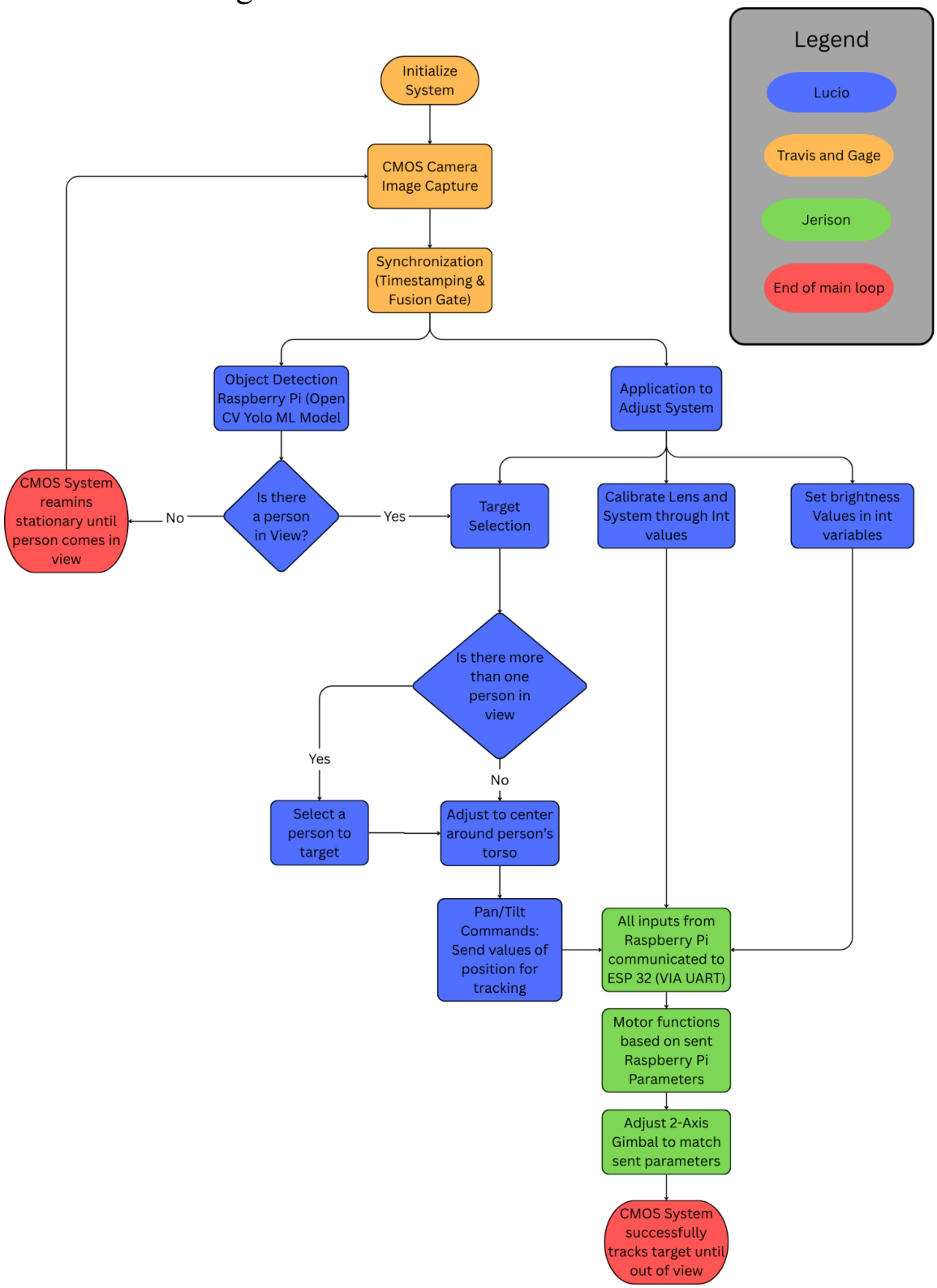


Figure 5.1: Software Block Diagram

2.9 Optical Components

Illumination system

The illumination system is composed mainly of two components: the source and reflector, and the focusing lens. The reflector is modeled after the ETC Source Four design, which is an elliptical housing coated with a reflective material to maximize light collection and projection efficiency. The lighting source can vary, as long as it is small enough to fit in the housing. For S.T.A.R., the Source Four lamp will be used. For the S.T.A.R. prototype, a Source Four lamp has been selected due to its high lumen output and strong overall efficiency. The primary drawback of this lamp is its heat generation, so a further investigation and testing of LED sources will be appropriate.

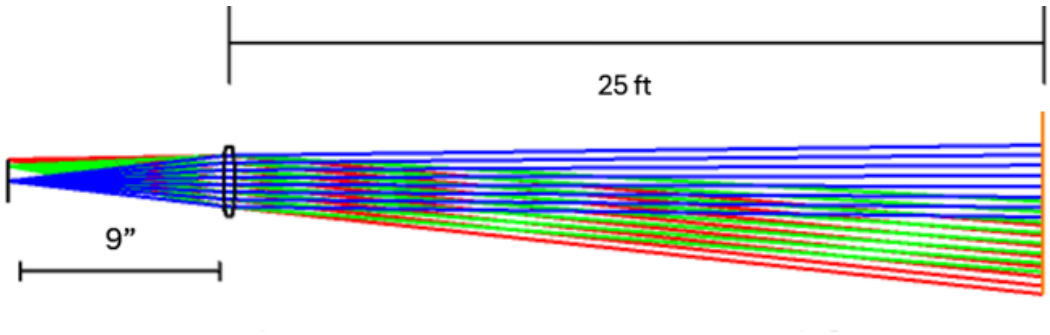


Figure 6.0 Cross section for output lens of S.T.A.R.

To be as cost efficient as possible, a biconvex lens is used to achieve a roughly 8.5 degree output half cone angle. Considering the variety of scenarios that S.T.A.R. will be used in, 8.5 degrees seems to be an optimal angle for varying distances. The lens will be placed at the end of the assembly. If we find that this angle is too wide for large distances, an iris will be placed at the gate of the optical assembly to reduce spot size.

Imaging System

The imaging system consists of a lens system and a CMOS sensor. This is a crucial optical element for the system as it determines the efficiency of the spotlight tracking. Glass choice is vital for building an optical setup to meet the parameters scoped for the resultant image. For this scope we are aiming to reduce as many aberrations as possible, leading to the design choice of using two types of glass: a crown lens and a flint lens. To maintain cost efficiency we explore the option of Commercial Off the Shelf Lenses (COTS) which provide a low cost alternative in comparison to a custom built lens.

The objective is to build a lens system that focuses light to a focal plane equivalent to the sensor size used. In this case we are using the **Raspberry Pi High Quality Camera 12.3 megapixel Sony IMX477 sensor**,

Table 2.0 Specifications of tracking camera

Raspberry Pi High Quality Camera 12.3 megapixel Sony IMX477 sensor	
Sensor Size	Horizontal: 6.32mm Vertical: 4.74mm
Pixel Size	1.55µm × 1.55µm
Megapixels	12.3
Wavelength Range	~ 400nm - 700nm
Lens sensor format	1/2.3” (7.9mm) or larger
IR Cut Filter	Integrated

For the crown lens, a common glass of choice to consider is an N-BK7 which when uncoated provides a transmission wavelength in the visible light spectrum to the near infrared wavelengths.

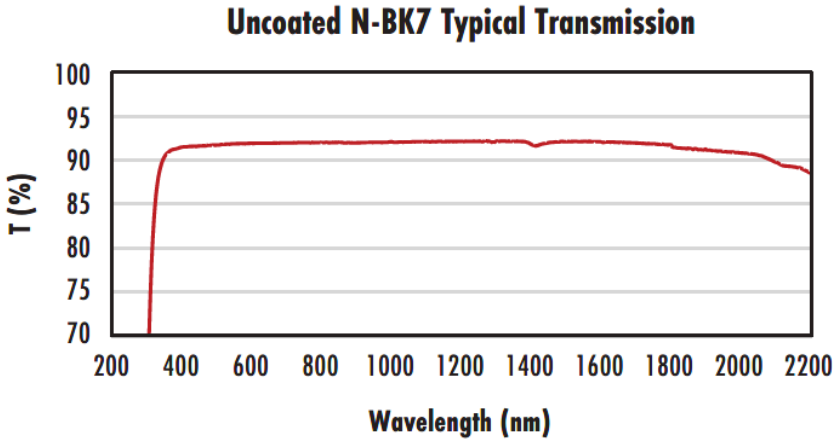


Figure 7.0: Typical Wavelength Transmissions of an Uncoated N-BK7 per Edmund Optics

As mentioned prior the lenses used are commercially off the shelf lenses. From Edmund Optics, the first frontal lens is the collector lens, gathering light to the optical system. This achromatic doublet features both a crown and a flint lens using substrates N-BK7 and N-SF5, respectively to minimize aberrations. The lens has a large diameter compared to other lenses within the system as it enables for more light to enter. The second lens is another achromatic doublet from Edmund Optics with an N-FK5 and F2HT substrate that serves as the corrector lens to mitigate the aberrations formed by the other optics, and produce a clearer image. The third and final lens of this system is also from Edmund

Optics, with substrates N-BAF10 and N-SF10 and a smaller diameter and focal length to minimize the effective focal length of the overall system, and maintain its compactness.

2.9 House of Quality

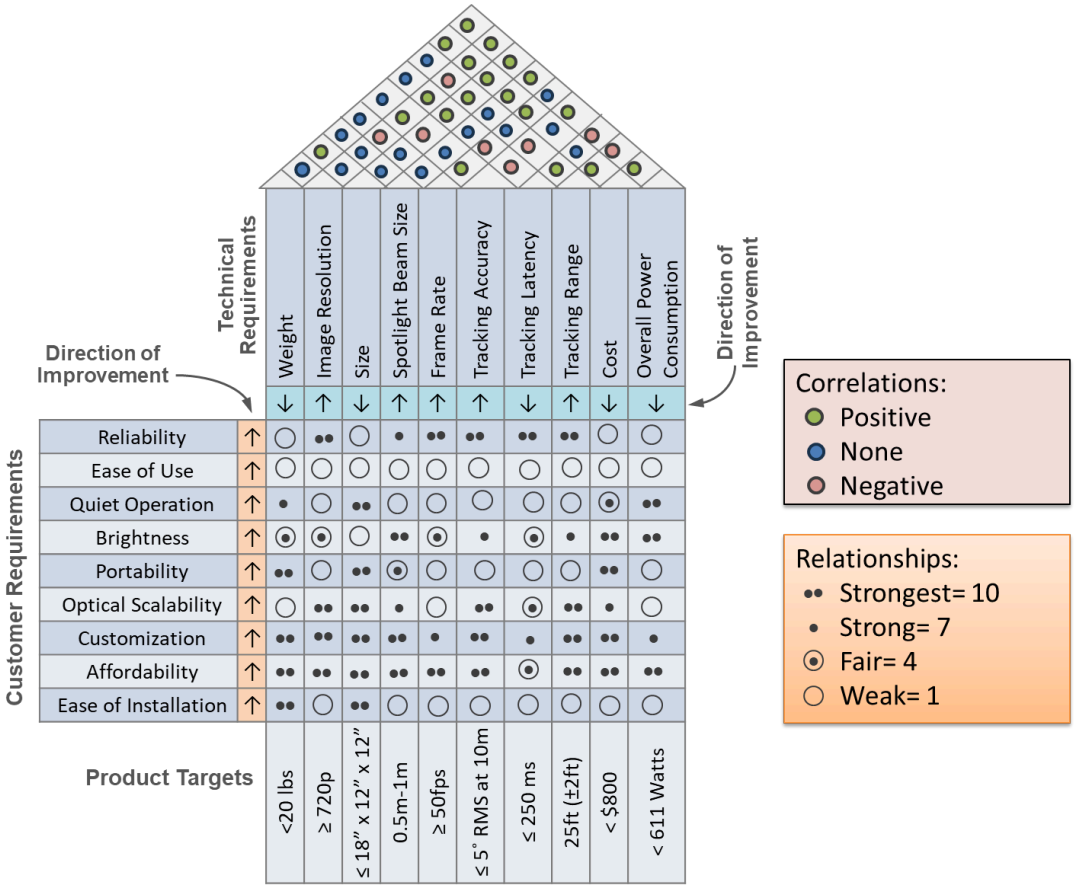


Figure 8.0: House of Quality

Chapter 3 - Research and Investigation

3.1 Illumination System

3.1.1 Reflecting vs. Refracting Optics

Generally, optical components can be divided into two main categories based on how they manipulate light: reflectors, which redirect light using mirrors or reflective surfaces, and refracting systems, which bend light using glass or other transmissive optical material. Both categories aim to achieve the same goal; image formation, beam shaping, or focussing. Their underlying physics, performance trade-offs, and material constraints differ significantly. Understanding these distinctions is crucial in selecting the appropriate configuration for a given optical system.

Fundamental Principles

Refracting optical systems operate on the fundamental physical principle known as Snell's Law of Refraction, which describes how light changes direction when passing between materials with different refractive indices. The law is expressed as:

$$n_1 \bullet sin(\theta_1) = n_2 \bullet sin(\theta_2) \quad (1)$$

where $n1$ and n2 are the refractive indices of the initial and secondary media, respectively, and θ1 and θ2 are the corresponding angles of incidence and refraction, measured relative to the normal at the surface interface.

A lens is formed from two or more refracting surfaces, typically spherical or aspheric, designed to redirect incident light rays to a specific focal region. The optical power of a lens is described by the Lensmaker's equation:

$$\frac{1}{f} = (n - 1)(\frac{1}{R1} - \frac{1}{R2}) \quad (2)$$

Where f is the focal length, n is the refractive index of the lens material, and R1 and R2 are the radii of curvature of the two surfaces. Because refraction depends on both material and surface geometry, lens designers can very precisely control how rays converge to form an image or spot.

Unlike refracting optics, which bend light as it passes through transparent materials, reflective optics redirect light entirely by bouncing the incident rays off of a surface. Some reflective optical components are completely flat, to be applied in redirecting incoming beams. Other reflective surfaces are designed with a curvature, to control the direction and convergence of light rays with its shaped surface. The curvature of a reflective optical surface can be aspherical or spherical as well, with the focal length of a spherical mirror simply being

$$f = \frac{R}{2} \quad (3)$$

This means that the focal point of a spherical mirror lies halfway between the mirror's surface and its center of curvature [15].

There are many other reasons why you would want to choose reflective optics rather than lenses. Reflectors provide a way to focus light while limiting aberrations - unwanted effects during the focusing process - much more than a lens would. Reflective optics are much better at handling higher temperatures, which is important when working with high powered sources.

3.1.2 Illumination System Design

Cold Mirror Reflector

A key component of the illumination system is the cold mirror reflector, which collects light from the source and focuses it through the aperture while simultaneously managing the system's thermal load. In high-intensity lighting systems, such as those using powerful LEDs or discharge lamps, a significant portion of the emitted energy lies outside the visible spectrum as infrared radiation. If not properly managed, this infrared energy can heat the surrounding optical components, degrading performance or shortening the lifespan of the fixture [16]. The cold mirror reflector solves this problem by reflecting visible light forward while transmitting infrared radiation through the back surface, which can then be dispersed via a fan.

The cold mirror is typically constructed from a borosilicate glass substrate, such as BK7, which is coated with multiple thin dielectric layers. These alternating high and low refractive index materials such as titanium dioxide (TiO_2) and silicon dioxide (SiO_2) are deposited in carefully calculated thicknesses to create wavelength-selective interference effects [17]. The result is a mirror that achieves high reflectance in the visible spectrum (approximately 400–700 nm) and high transmittance in the infrared region (above 750 nm). This design allows the reflector to maintain a bright, efficient optical output while significantly reducing the thermal energy directed toward the front lens and aperture.

The geometric design of the cold mirror reflector is ellipsoidal, meaning it has two focal points that govern how light is collected and directed. When the LED or lamp source is placed at the first focal point, the reflected rays converge at the second focal point, forming a concentrated beam and imaging plane as shown in figure 10:

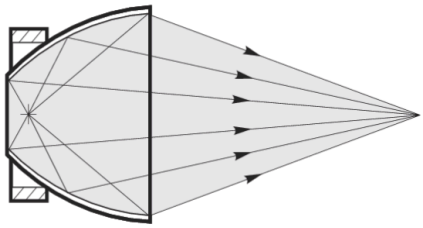


Figure 10. Ellipsoidal reflector diagram showing how light from the first focal point reflects to the second focal point. Adapted from [18].

This geometry maximizes optical efficiency by minimizing light spill and ensuring that nearly all usable light is redirected toward the focusing lens. Because the mirror reflects only visible light, the beam that reaches the second focal point remains cooler, protecting downstream components such as lenses and filters from thermal stress.

Because we aim to make our optical system as compact as possible, we want the f/# of the reflector to be as small as possible, giving us an optically "fast" system that will

deliver a wide, bright cone of light at the gate. The relationship between f/# and NA is given by the equation:

$$f/\# = \frac{1}{2NA} \tag{4}$$

They are inversely proportional, so a larger numerical aperture will contribute to a smaller f/#, making the system much faster.

Using a cold mirror reflector also helps maintain color stability within the projected beam. By preventing the forward projection of infrared radiation, the system preserves the true color temperature of the light source. This ensures consistent illumination quality and prevents warm color shifts that could otherwise occur as optical components heat up. The selective spectral behavior of the coating therefore not only enhances thermal performance but also contributes to the optical accuracy and visual consistency of the overall system.

In practical terms, the cold mirror reflector forms the foundation of the illumination assembly, performing the dual roles of light collection and thermal filtration. Its material and coating design enable high reflectivity for visible wavelengths while passively managing heat dissipation, allowing for a more compact and reliable optical housing. By combining high optical efficiency, durability, and selective spectral reflection, the cold mirror reflector provides a critical balance between brightness and thermal safety, making it an ideal choice for high-performance lighting systems where clarity and stability are essential.

Focusing Lens

The focusing lens is the final optical element in the S.T.A.R. illumination assembly and plays a critical role in shaping the beam projected onto the stage. After light is collected and redirected by the ellipsoidal cold mirror reflector, it converges toward the gate where the intermediate image is formed. The focusing lens then refines this light into a well-defined, uniform beam with the desired angular spread. Because the optical performance of the entire fixture depends on the accuracy and clarity of this beam, the selection of the lens geometry, focal length, and material was conducted with particular care.

The lens assembly must meet several objectives simultaneously. It must produce a controlled beam with an approximate full output angle of 19°, maintain the high luminous output of the LED source, and minimize optical aberrations to ensure consistent illumination across the entire field. The lens also needs to remain stable under moderate thermal conditions from residual heat within the system while providing a means for precise mechanical adjustment to fine-tune focus. These combined optical and mechanical requirements defined the foundation for the focusing lens design.

The focal length of the lens primarily determines the relationship between the gate and the projection distance, dictating how strongly the beam converges or diverges [20]. In general, a shorter focal length produces a wider beam with greater divergence, while a

longer focal length results in a narrower, more concentrated output. Similar to the reflector unit, the focusing lens should operate with a relatively small focal length relative to the system's numerical aperture (NA) to achieve a bright and efficient beam.

To begin, the numerical aperture of the lens can be expressed as:

$$NA = n(sin(\frac{\theta}{2})) \tag{5}$$

where n is the refractive index of the medium (equal to 1 for air), and θ is the half-angle of the converging cone of light.

Next, the focal length is determined by the geometric relationship between the projection distance and the beam angle, given by:

$$f = \frac{r}{tan(\theta/2)} \tag{6}$$

Where r is the radius of the aperture (25mm), and θ is the output throw angle (19 degrees). Solving this equation with our parameters gives us a focal length of 167mm.

The final and most important mechanical parameter of the focusing lens is its entrance pupil diameter (EPD), which defines the clear aperture of the lens and ultimately determines its physical size, mass, and cost. It can be calculated using the relationship:

$$EPD = \frac{f}{f/\#} \tag{7}$$

This equation gives us an EPD of 103mm. This result means that the focusing lens requires a minimum diameter of 103 mm to capture the entire light cone without vignetting. A lens of this size offers a balance between optical performance and manufacturability, large enough to efficiently pass the collected light, yet compact enough for integration within the fixture housing.

With all of these parameters defined, we have enough information to model the lens system that will be used in the S.T.A.R system, shown in figure 11:

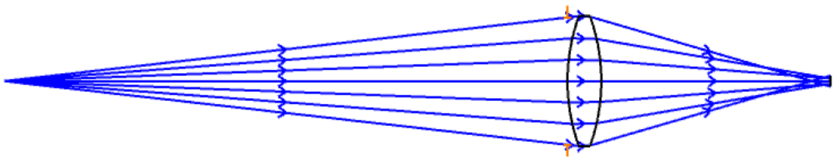


Figure 11: Focusing Lens Simulated in Zemax Opticstudio

Gate

The gate represents the image plane of the illumination system, located between the ellipsoidal reflector and the focusing lens. At this position, light collected by the reflector

converges to form a sharp, controlled focal point before being refocused by the front lens into the output beam. Because this region defines where the system produces its most accurate and concentrated image, the gate serves as the optimal location for inserting image-forming and beam-shaping components. In professional lighting systems, this is where gobos, irises, and shutters are placed to sculpt or pattern the light with precision.

A gobo, short for "goes before optics," [21] is a thin template or stencil placed at the gate to shape the light beam into a specific pattern or image. When light passes through the gobo, the design is projected and magnified by the focusing lens onto the stage. Gobos are commonly made of stainless steel for simple, high-contrast patterns or dichroic glass for detailed, multicolor imagery [22]. Their interchangeability allows for easy customization, enabling projection of holiday themes, logos, scenic textures, or any design tailored to a particular event.

To maintain compatibility with existing professional standards, the gate in the S.T.A.R. system is designed to accept B-size gobos, matching the format used in ETC Source Four fixtures. This ensures compatibility with a wide range of commercially available gobos, allowing designers to use both stock and custom templates without modification. Although most infrared radiation is filtered by the cold mirror reflector, the gate region still experiences moderate heat, so ventilation is incorporated to protect the gobo and surrounding components. The result is a stable, modular system that maintains sharp, consistent projection while offering full creative flexibility.

Internal Baffles

One unique aspect of the optical design of the S.T.A.R. illumination system is the inclusion of internal baffles, also known as light traps, within the lens tube assembly. These are a series of annular, matte-black structures positioned along the interior walls of the focusing lens housing [23]. Their primary purpose is to absorb stray light, suppress internal reflections, and enhance overall image contrast. In high-intensity optical systems such as this, even a small amount of unwanted reflection can lead to visible artifacts, reduced beam sharpness, and loss of contrast. The baffles serve as passive optical filters, ensuring that only light within the designed optical cone contributes to the final projected image.

When working with imaging and illumination systems, one of the most common challenges is the presence of ghost reflections, which are faint, secondary images caused by light reflecting multiple times between lens surfaces or the inner housing walls. These reflections can produce blurred halos, double edges, or unwanted light spill that degrade image quality. By placing baffles along the optical path as shown in figure 12, these stray rays are intercepted and absorbed before they can reach the projection plane. This results in a cleaner, higher-contrast beam with sharply defined edges, which is especially important when projecting gobos or patterns.

Incorporating internal baffles into the design ensures that the S.T.A.R. system maintains the same high level of image fidelity and beam quality found in professional-grade fixtures such as the ETC Source Four. By controlling stray light within the housing, the

system achieves greater optical efficiency, improved visual contrast, and a distinctly professional output that is free of unwanted glare or ghosting artifacts.

Figure 12: internal baffles in a stage lighting system

3.1.3 Materials

Focusing Lens

The choice of optical material plays a critical role in determining the performance, durability, and efficiency of the illumination system. Each lens material has distinct optical and mechanical properties; such as refractive index, dispersion, transmission range, and thermal stability. All of which influence how light is refracted and transmitted through the system. For the S.T.A.R. focusing lens, it was important to select a material capable of maintaining high optical clarity and withstanding the heat generated by the LED source and reflector assembly. To evaluate potential options, several candidate materials were analyzed, including optical plastics, BK7 crown glass, and SF2 dense flint glass. The following table summarizes the optical characteristics and suitability of each material for use in the S.T.A.R. system.

Table 2: Comparison of Optical Materials for Focusing Lens

Material	Refractive Index (n)	Abbe Number	Thermal Stability	Application / Notes
Optical Plastic (Acrylic)	~1.49–1.59	High (low dispersion)	Very low	Lightweight, inexpensive alternative for low-power systems; suitable for prototypes or environments with minimal heat exposure.

N-BK7 (Borosilicate Crown Glass)	1.5168 @ 587.6 nm	64 (moderate dispersion)	High	Industry-standard optical glass; excellent clarity, durability, and transmission; used in most general-purpose imaging and illumination systems.
SF2 (Dense Flint Glass)	1.6477 @ 587.6 nm	33.8 (High dispersion)	High	High-index material; useful for compact optics or achromatic doublets when paired with crown glass; more expensive and heavier than BK7.

Among the compared materials, N-BK7 was selected for the S.T.A.R. system's focusing lens. It offers an ideal balance of optical performance, mechanical stability, and cost-effectiveness. BK7 provides high visible and near-infrared transmission, a moderate refractive index, and low dispersion, making it suitable for imaging without introducing significant chromatic aberration. While SF2 provides a higher refractive index, its increased dispersion and weight make it less favorable for compact optical systems. Plastic lenses, though lightweight and inexpensive, were excluded due to their limited thermal tolerance under high-intensity illumination.

Reflector

The choice of reflector material has a significant impact on both the optical performance and thermal efficiency of the illumination system. The reflector's purpose is not only to redirect light from the source toward the gate but also to control how much infrared (IR) radiation and heat are transmitted through the optical path. Materials differ in how they reflect visible light, manage heat, and withstand prolonged exposure to high temperatures. Factors such as reflectivity, spectral selectivity, durability, and manufacturability must all be balanced to ensure long-term performance and safety. To determine the most suitable material for the S.T.A.R. illumination system, several common reflector types were compared, including polished aluminum, coated aluminum, dichroic (cold-mirror) glass, and reflective polymer films. The following table summarizes their relative optical and thermal characteristics.

Table 2.1: Comparison of Reflector Material Types

Material	Reflectivity (Visible)	Thermal Stability	Spectral Behavior	Notes Applications
Bare Aluminum	~85–90%	Moderate; may oxidize	Reflects visible + IR	Common, low cost, but projects heat forward.

Aluminum with Enhanced Coating (Al + dielectric)	~93–95%	Slightly better than bare Aluminum	Reflects visible; partial IR removal	Higher performance than bare metal reflector; improved durability.
Dichroic Glass (Cold Mirror)	~95–98% visible; IR transmitted	High; excellent stability	Reflects visible, transmits IR	Ideal for heat management in thermal systems.
Silicone / Polymer Reflective Film	~97% (depending on coating)	Good (varies by film type)	Reflectivity varies by film	Used in cost-sensitive applications; limited thermal endurance.

After evaluating these materials, the dichroic cold mirror reflector was chosen for the S.T.A.R. system. This material provides exceptional visible-light reflectivity (approximately 95–98%) [24] while allowing unwanted infrared radiation to pass through the back of the reflector, preventing heat buildup near the gate and lens assembly. The glass substrate offers excellent dimensional and thermal stability, ensuring stability under continuous operation. Unlike bare or coated aluminum, which can oxidize or discolor over time, the dielectric coating on the cold mirror is stable and maintains performance for thousands of hours.

Although polymer reflective films and metallic coatings are cost-effective and lightweight, they cannot withstand the high temperatures produced by a concentrated LED source. The cold mirror reflector, by contrast, combines high optical efficiency with robust heat management, resulting in a brighter, cooler, and more stable beam. For these reasons, the dichroic cold mirror was determined to be the optimal material for the reflector in the S.T.A.R. illumination system.

Instrument Housing Material

The outer body of the S.T.A.R. illumination system has to fill many roles; it provides mechanical support for the optical components, protects the internal electronics, and ensures safe operation under high temperature conditions. The housing must therefore be structurally rigid, lightweight, and thermally resilient, while also minimizing the total weight to allow smooth operation on its motorized gimbal assembly.

Because the enclosure is 3D printed, the choice of filament determines how well the system withstands heat generated by the LED and reflector, as well as how easily complex features such as mounts, ventilation channels, and cable paths can be fabricated. Several thermoplastics were evaluated for this purpose, including standard PLA, ABS,

PETG, HT-PLA, and polycarbonate (PC). Each material offers distinct trade-offs in printability, strength, and thermal behavior. The following table summarizes their key characteristics and suitability for use in the S.T.A.R. illumination system.

Table 2.2: Comparison of 3D Printing Materials for Fixture Housing

Material	Heat Resistance	Density / Weight	Printability / Cost	Notes / Suitability
Standard PLA	~60–65 °C glass transition	~1.24–1.30 g/cm³	Very easy, low cost	Excellent for prototypes, but poor heat resistance for lighting housings.
ABS	~100–110 °C deflection temperature	~1.04–1.08 g/cm³	Moderate (warps; enclosure needed)	Better heat resistance than PLA but still marginal near high radiant heat sources.
PETG	~70–80 °C heat resistance	~1.27 g/cm³	Good printability, moderate cost	Balanced properties, but not ideal for prolonged exposure to high heat.
HT-PLA (High-Temperature PLA)	Up to ~150 °C when annealed	~1.24–1.30 g/cm³	Similar to PLA; slightly higher cost	Engineered for heat applications; lightweight and thermally resistive
PC (Polycarbonate)	Glass transition ~145–150 °C	~1.20–1.22 g/cm³	Difficult print (high temp, enclosure needed)	Excellent heat and strength properties but higher cost and complexity.

After comparing several common 3D-printing materials, HT-PLA was chosen for the S.T.A.R. housing due to its unique combination of thermal resistance, low weight, and

printability. Once annealed, HT-PLA achieves a glass transition temperature of up to 150C, allowing it to maintain structural integrity even in the thermally intense environment surrounding the LED module and reflector [25]. This represents a substantial improvement over standard PLA and PETG, which begin to soften around 60–80 °C, and provides sufficient safety margin for extended operation. The improved heat tolerance ensures that the housing will not warp, deform, or lose alignment when exposed to the fixture's radiant heat.

Beyond thermal performance, HTPLA provides excellent dimensional stability and ease of fabrication. Its compatibility with standard FDM 3D printing allows for easy prototyping and precise geometric constraints without the need for costly machining or molds. The ability to print custom component mounts, internal airflow channels, and complex mechanical interfaces directly into the design simplifies assembly and reduces part count.

Overall, the choice of HTPLA offers an optimal balance between thermal resistance, strength, and manufacturability. It supports reiteration during the development phase while maintaining the heat tolerance and rigidity needed for reliable performance in a lighting environment. This makes it a practical and efficient material choice for the S.T.A.R. system's housing, combining the benefits of lightweight design with sufficient durability for continuous optical operation.

3.1.4 Illumination Source Comparison

Selecting the appropriate illumination source is one of the most critical aspects of designing a system like the S.T.A.R. project. The light engine defines the optical, thermal, and mechanical behavior of the entire fixture. The choice of light engine directly affects brightness, color rendering, efficiency, and even physical parameters such as system weight, length, and power distribution. There are many different illumination technologies on the market, such as traditional halogen-tungsten lamps, high-power LED COB modules, and modern RGBW light engines. Each unique engine presents advantages and limitations that must be carefully evaluated. Understanding the characteristics and constraints of each source type allows for a more efficient and optimized optical design, ensuring proper integration with other subsystems. The following subchapters discuss these illumination technologies in detail, outlining their respective principles, strengths, and trade-offs, followed by a direct comparison to determine the most suitable source for the S.T.A.R. illumination system.

LED COB Modules

Chip-on-Board (COB) LED modules are compact, high-efficiency light engines that integrate multiple LED dies onto a single substrate, creating a bright, uniform emission surface. This design minimizes optical losses and improves thermal conductivity compared to traditional discrete LED arrays, allowing for higher luminous density and simpler optical alignment. With efficiencies exceeding 90 lm/W and lifetimes over 25,000 hours [26], COB modules provide an excellent balance of brightness, energy efficiency, and reliability. Their continuous, even output eliminates visible hotspots,

producing a clean beam ideal for projection and imaging applications. Below is a table that demonstrates the differences between a COB module and a traditional LED array;

Table 3: Comparison of LED Light Source Technologies

Source Type	Efficiency (lm/W)	Lifetime (hours)	CRI Range	Thermal Load
Discrete LED Array	60–80	10,000–15,000	80–90	Moderate
LED COB Module	90–110	25,000+	80–95	Low–Moderate

COB modules also offer exceptional design flexibility, supporting a wide range of color temperatures and CRI values while remaining compact enough for lightweight optical housings. Although they require a constant-current driver and heat sink, these integration needs are minimal compared to the overall performance benefits. For the S.T.A.R. illumination system, the COB LED module was selected for its high lumen output, low thermal load, and long service life, making it the most practical and efficient choice for a gimbal-mounted optical assembly.

Halogen-Tungsten Lamp

The halogen-tungsten lamp has been a staple in stage lighting for decades, most notably popularized by the ETC Source Four lighting fixture. These lamps are praised for their extremely bright output and their warm, natural quality of light they produce. These characteristics are highly valued by lighting designers for creating visually appealing and inviting lighting scenes. Even with the rise of LED technology, many manufacturers continue to emulate the distinctive look and warmth of halogen-tungsten illumination to satisfy designers seeking the original look. Optically, these sources are ideal; the spectrum they produce matches almost perfectly to the blackbody spectrum, as shown in figure 13:

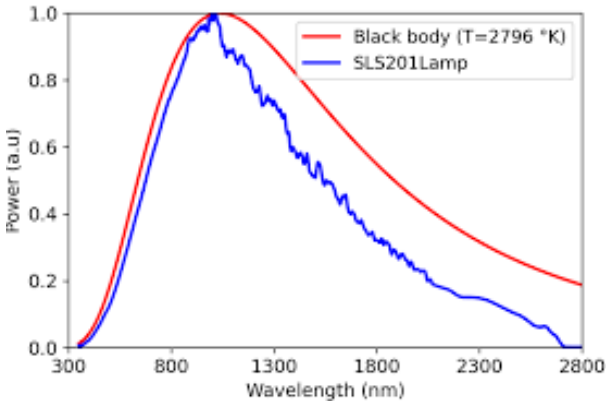


Figure 13: Spectral power distribution comparison between a blackbody source at 2796K and a halogen lamp. Adapted from [27]

However, despite their appealing light quality, halogen-tungsten lamps come with significant drawbacks. The most notable is their extreme heat generation, which far exceeds that of modern solid-state light sources. A large portion of their electrical power is converted into infrared radiation rather than visible light, leading to surface temperatures that can reach 518 °F (270 °C). This high thermal output not only reduces efficiency but also imposes strict design requirements for ventilation, heat shielding, and material selection—making halogen lamps less practical for compact or automated systems like the S.T.A.R. unit. They are also extremely sensitive to handling; touching the surface even once will transfer oils onto the glass housing, causing irregular heating and ultimately lead to an explosion of the lamp.

RGBW Engine

RGBW light engines combine red, green, blue, and white LEDs into a single integrated system, enabling full-spectrum color mixing and adjustable white-light output. By varying the intensity of each color channel, the engine can produce a nearly limitless range of hues and color temperatures—from warm tungsten-like ambers to bright, daylight-balanced whites. This makes RGBW technology highly attractive in entertainment and architectural lighting, where flexibility and dynamic control are prioritized. The inclusion of a dedicated white emitter also improves color rendering and brightness compared to traditional RGB systems, which often struggle to produce clean, high-intensity whites. This inclusion still does not come close to the robustness of a traditional incandescent source, as shown by the spectrum of an RGBW engine in figure 14

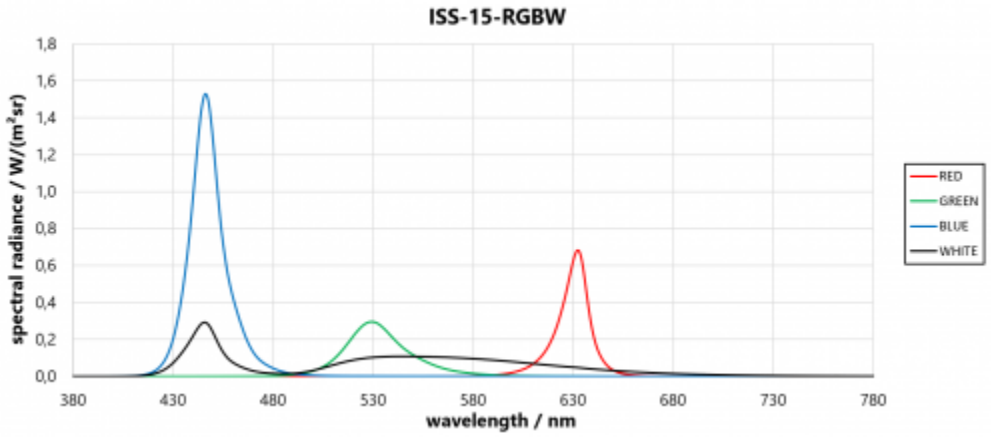


Figure 14: Spectral radiance of an RGBW LED engine, showing individual red, green, blue, and white emission peaks. Adapted from [28]

Despite their versatility, RGBW engines introduce increased system complexity. Each color channel requires precise current control and calibration to achieve consistent color balance and intensity. This demands a specialized driver and often digital signal control

such as DMX-512, increasing both the cost and electronic bearing of the system. Additionally, because the total electrical power is divided among multiple emitters, the overall luminous output of an RGBW engine is typically lower than that of a comparable single-source COB module. While RGBW fixtures are excellent for applications that benefit from color effects or tunable white output, their added cost, control requirements, and reduced simplicity make them less ideal for compact, fixed white-light systems like the S.T.A.R. unit.

Comparison

After evaluating multiple illumination technologies, it became clear that each source offers unique strengths and limitations depending on the intended application. Halogen-tungsten lamps excel in color rendering and produce the warm, natural tones favored by many lighting designers but suffer from extreme heat output and short lifespans. RGBW light engines provide unmatched color versatility and dynamic control but introduce additional complexity, cost, and lower overall brightness. In contrast, LED COB modules offer the best balance of efficiency, thermal stability, and luminous performance, making them ideal for compact and automated systems. The following table summarizes the performance characteristics of these illumination sources, highlighting the criteria that guided the selection of the most suitable light engine for the S.T.A.R. illumination system.

Table 4: Comparison of Halogen, LED COB, and RGBW Light Sources for Stage Illumination

Criteria	HPL 757 Halogen Lamp	High-Power LED COB Module	Integrated RGBW Engine	Best Option
Luminous Output (lm)	≈ 5000	4500–6000	3500–5000	LED COB Module
Efficiency (lm/W)	≈ 18	≈ 90	≈ 70	LED COB Module
CRI	> 85	80–90	> 90	Integrated RGBW Engine
Thermal Load	High	Moderate	Low	Integrated RGBW Engine
Lifetime (hours)	300–500	25,000+	20,000+	LED COB Module

Cost (USD)	~$20	~$80	~$150	HPL 757 Halogen Lamp
Integration Ease	Direct fit to housing	Requires adapter + driver	Custom driver needed	HPL 757 Halogen Lamp

Furthermore, the COB module's broad, uniform emission pattern pairs exceptionally well with the ellipsoidal cold mirror reflector used in the S.T.A.R. optical assembly. The reflector's geometry is designed to collect light from an extended source and redirect it toward the system's focal plane with high efficiency. Because the COB emits over a relatively wide solid angle, its light distribution closely matches the acceptance profile of the ellipsoidal reflector, minimizing spill light and maximizing usable flux at the gate.

3.2 Imaging System

3.2.1 Technical and Theoretical Considerations

Object Magnification

Before selecting hardware components for lenses, it is important to dive into the math and theory driving the decisions to be made. A preliminary factor to consider is the target object magnification needed to properly image the performing stage onto the sensor. This target magnification can be directly calculated using the height of the output image, and the height of the object using the equation below.

$$M = \frac{h_i}{h_o} \tag{8}$$

When using this equation it is important to make note that h_o and h_i are the width of the stage and the width of the sensor, respectively. With a target stage size of 25ft or 7620mm, and a horizontal sensor size of 6.32mm, the magnification can be calculated as a 1250x demagnification. Such magnification is hard to achieve with a two lens system, but could be easier to achieve with a three lens system.

With the calculated magnification, another variable that can be equated is the back focal length of the lens system to the sensor. There is another ratio that relates magnification. The distance from the object to the lens (u), as well as the distance of the lens to the image/sensor (v) can also be used as shown below.

$$M = \frac{u}{v} \tag{9}$$

For the values of magnification and the fixed value of *v,* the distance u can be calculated as 6.1mm. This is verifiable through the thin lens equation where

$$\frac{1}{f} = \frac{1}{u} + \frac{1}{v} \tag{10}$$

As our distance from the object to the lens is significantly greater than our distance from the image to the lens ($v >> u$), it is a safe approximation to make that our focal length $f = u$. Therefore when viewing our lens system overall, we are seeking for an effective focal length (EFL) of about 6.1mm.

3.2.2 Sensor Technology Comparison

CCDs vs CMOS

In order to properly display an image, a choice in sensor must be made. There are two prominent types of sensors when considering an imaging system: a charge-coupled device (CCD) and a complimentary metal-oxide semiconductor (CMOS). These devices both produce an electrical signal from incident light received. Since we are aiming to use our image for human-tracking, it is important to consider the advantages and disadvantages of both sensors.

CCDs provide benefit in the pixel to pixel variation allowing for more precise optical measurements and less noise in the image produced, which is important as the applications for the project would most likely be used in a low light setting. This is achieved as the charge is transferred and read on a single amplifier. Although there are benefits, there are some drawbacks. CCDs tend to process at slower frame rates due to the readout architecture, and require a higher consumption of power. They also tend to draw more heat to the system, which is important to consider as the design needs to heavily monitor heat transfer from the illumination source.

CMOS cameras provide a parallel readout that allows for an increased frame rate, ideal for real-time imaging. For the pixel array in a CMOS sensor, each pixel contains its own amplifier with an analog-to-digital converter, which supports the faster frame rate. Since we are focusing on tracking motion on a stage, frame-rate is a key parameter to account for, making it a great choice to consider. These sensors also tend to exert less heat, removing the necessity of a method to cool the product. Overall, the choice between a CMOS and a CCD was determined by the factor of framerate speed and temperature, conditions that are necessary to consider for the scope of the project so communication between the real-time motion and the human tracking can be more efficient. To save costs, the CMOS sensor technology is the route we considered for this project, and thus we explored cameras using them.

Table 5: Comparison of CCD and CMOS

Specification	CDD	CMOS
Frame rate	Slower	Faster
Temperature control	Requires cooling	Does not require cooling
Power consumption	High power usage	Low power usage

Output noise level	Low	High
Cost	More expensive	Less expensive

Global shutter vs rolling shutter

When reviewing various sensor technologies, there are two primary capturing techniques used by sensors that define how light is captured over time. Global shutters capture an entire scene at a precise instant in time as each pixel in an array begins and ends their exposure in a simultaneous fashion. In doing so, image distortions are put to a minimum as real time movement in the object does not affect how each pixel sees it over time. Rolling shutters on the contrary, reads and exposes an image line-by-line sequentially. With this process, each row of pixels has a slight offset in time in the microseconds as there is real-time motion in the object. This can cause slightly more distortion effects when imaging. It is more common to see rolling shutters in most consumer CMOS applications. The simpler pixel readout technique tends to output lower costs, and show to be more compact. Since S.T.A.R. is aiming to rely on a Raspberry Pi system camera, the compactness and cost of these sensors are the factors we prioritized when looking for a camera.

ArduCam Pi Hawk-eye for Raspberry Pi

During the initial search for an imaging sensor, the attention of a high megapixel camera seemed to be beneficial for our design. With a higher sensor size, and thus a higher megapixel count, more incident light can be captured by the sensor, allowing for discrete sampling points. This means the optical image that is formed on the sensor would contain a finer image. With a high megapixel count, and a pixel size of 0.8 μm × 0.8 μm, the camera is able to perform 2x2 binning. This sensor-level technique takes a 2x2 array of pixels and merges them to form one effective pixel, reducing the output image by a factor of 4. Through this, the total still resolution is effectively 16mp, which is a deduction, but each pixel grants an increase in signal as the total light is from the previously combined array. As more light is captured by each pixel of the sensor, it is important to compare the ratio of signal-to-noise (SNR). WIth a 4x greater signal input, the noise of the overall image is decreased, and thus this 2x2 binning technique is very useful in low-light imaging, an environment that S.T.A.R. would likely be used in.

Raspberry Pi HQ Camera

Another point of interest for an imaging sensor was the Raspberry Pi High Quality (HQ) Camera, a popular imaging module built using the Sony IMX477 sensor. With a 12.3 megapixel CMOS sensor, this camera has the capabilities of communicating with a Raspberry Pi Compute Module 5, the latest models of those MCUs. The sensor uses a backside-illuminated CMOS architecture, where light sensitivity is improved and noise is reduced, making it stronger for low-light performance, and high dynamic range. This camera supports C- and CS-mount interchangeable lenses, increasing the amount of optical control onto the system, an important feat, as S.T.A.R. requires a great

magnification to image a large stage in its practical application. Similar to the Hawk-eye, the HQ camera also uses a rolling shutter as opposed to global shutter, which may cause some distortions for more fast-paced scenes.

Imaging Sensor Selection

In the final selection of the sensor for S.T.A.R., there were many benefits of each camera we considered. The ArduCam Pi Hawk-eye provides a larger sensor size accompanied by a greater megapixel count, allowing for 2x2 barreling for less noise and greater low-light performance. However, the choice we made was the Raspberry Pi HQ Camera, primarily due to its ability to incorporate C and CS-mount lenses. With the flexibility of changing lenses, more optical control is gained, and thus the effective Field of View, depth of field, and magnification can precisely be customized to fit the needs of S.T.A.R. With this, more degrees of freedom are open to the optical design of the overall system to meet the target parameters, and goals of the design.

Table 6: Specification Comparison of ArduCam Pi Hawk-eye vs. Raspberry Pi HQ Camera

Specification	ArduCam Pi Hawk-eye	Raspberry Pi HQ Camera
Net Price	$65	$50
Shutter Type	Rolling Shutter	Rolling Shutter
Sensor	Sony IMX686	Sony IMX477
Sensor Resolution	9152 x 6944 pixels	4056 x 3040 pixels
Still Resolution	64 megapixels	12.3 megapixels
Horizontal Image Area	Not Specified	6.287mm
Vertical Image Area	Not Specified	4.712mm
Pixel Size	0.8 µm × 0.8 µm	1.55 µm × 1.55 µm
Horizontal Field of View	74 degrees	Lens Dependent
Vertical Field of View	55 degrees	Lens Dependent
Lens Mount	Non-interchangable	C/CS or M12 Mount

3.2.4 Stage Imaging System Design

Glass Materials and Geometry

Before considering lenses used to build the imaging system, it is critical to consider the variations a lens can be fabricated in from geometry to material. Glass material is an important aspect of lens design, as it has a major role in apparent aberrations in the output image. There are two primary lens categories to consider when building a lens system, a crown glass and a flint glass. Crown glass offers low-dispersion material characterized with a relatively low refractive index ($n < 1.50$) and a high Abbe number causing incident light to bend moderately and colors to separate less, minimizing the chromatic aberration in a lens system. On the contrary, flint glass offers a high refractive index with high dispersion, bending light more strongly with a greater degree of separation between colors, causing greater chromatic aberration. With a high refractive index, these glasses are beneficial for compact lens design systems. Both glass types have its essentials in a lens design, and it is important to consider this when making our selection of lenses for the tracking camera.

With the two primary types of glass to choose from, the geometry of a lens is also an important factor to consider. A singlet lens is made from a single piece of glass, and is typically a convex or concave element that focuses or diverges light based on its radii of curvature. As we are primarily relying on commercial off the shelf lenses over custom fabricated lenses, cost is an important factor when choosing the lenses we will use for the system. From commercial lens retailers, singlets are typically easier to manufacture and tend to be a more cost-effective choice for lens design. On the downside, they suffer from chromatic aberration, as the refractive index of these lenses tend to change with varying wavelengths, causing a degree of separation between the focus distances of different colors of light. This would cause more color fringing in our output image, which is not ideal for a program to accurately identify humans. In contrast to this, a doublet combines two lenses either cemented together or air-spaced. Doublets are typically composed of both a crown and a flint glass lens to cancel out the chromatic dispersion caused by the individual singlet of the lenses. This causes doublets to significantly reduce chromatic aberration allowing for a sharper and more accurate image that can further be improved for other aberrations. Although at a higher cost margin than its singlet counterpart, achromatic doublets are the primary choice we seek when choosing lenses to build our system.

Wavelength and Coatings

Before looking into a lens, it is important to be aware of the target wavelength that we are imaging. Since we are relying on the visible wavelength spectrum, the aim is to “block” out incident light at wavelengths beyond or below 300nm-750nm by reflecting them. As we are using commercial off the shelf lenses, there is much variety to the types of coatings provided. The main coatings we considered were those with a transmission range from the UV-NIR region. These wavelengths provide minimal surface reflection at the targeted wavelengths and intentional reflection at wavelengths outside of those ranges to to help mitigate sensitive wavelengths to our IMX477 sensor.

Collector Lens Selection

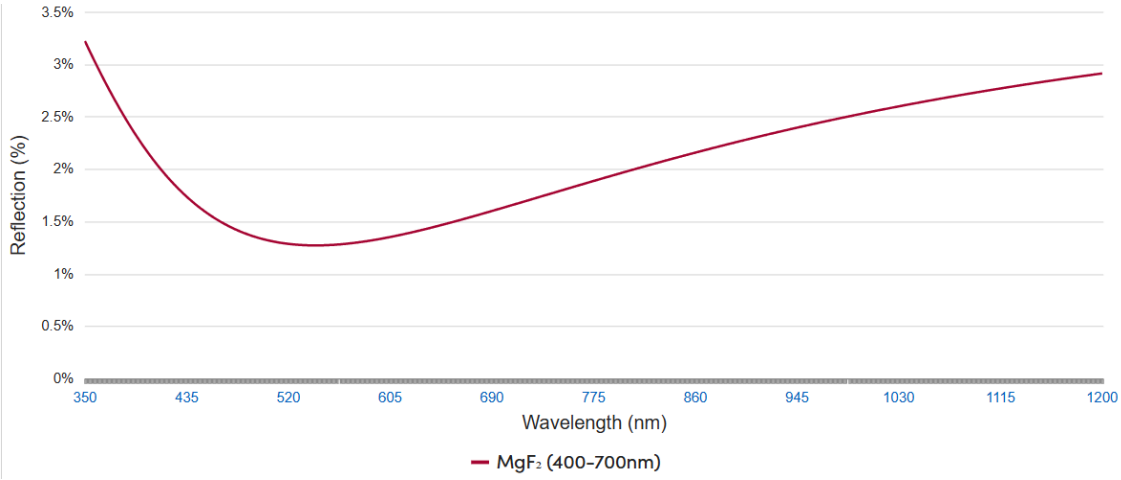
Since we are imaging from a target distance of >25ft, the collector lens ideally should have the maximum light collection of the system. A collector lens with a large diameter is sought for as it improves the amount of photons to reach the sensor, and to reduce vignetting of the overall image used for the tracking software. This lens ultimately determines the optical efficiency of our imaging system, as it defines how much light the overall system can accept, also known as the numerical aperture (NA). Providing a lens that promotes a high NA allows for a greater field cone angle that can transmit the collected light into the rest of the system. The formula relating the NA and the maximum cone angle is as follows.

$$NA = n \times sin(\theta) \quad (11)$$

It is important to note that *n* refers to the refractive index of the medium that the light is propagating in. In our case, the medium is air with a refractive index of 1, making NA a direct relation to the angle.

Edmund Optic's #49-768 lens provides a large diameter relative to the system we are using. With a 25.4mm diameter, and a 76.2mm focal length, enough light can be collected to ensure a clear image is formed. The lens contains an MgF2 coating which has a low reflectivity rate at wavelengths from 300nm-400nm as shown in the figure below.

Figure 15: Reflectance (%) of an MgF2 Coated Lens



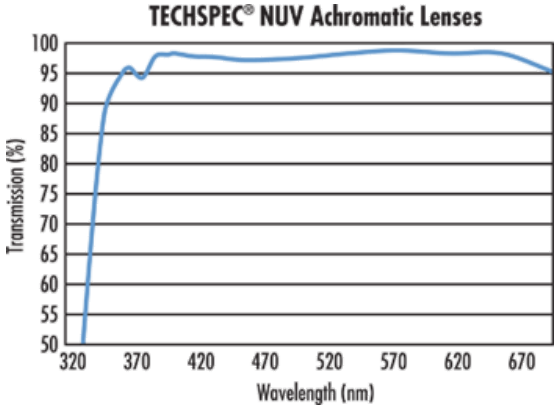
Corrector Lens Selection

The second lens of the design is the corrector lens, primarily used to eliminate aberrations by other optical components. This lens is pivotal for the system design as it improves the image quality by compensating for distortions such as spherical aberration, coma, and field curvature. Through this, corrector lenses extend the usable field of view of an imaging system and allow for other optical components to be used, without mitigating the quality of the output image. In an "ideal" lens, the shape is aspheric where the surfaces are contoured to counteract spherical aberration. Spherical aberration is a distortion caused by light focusing at different points rather than at the center as you leave off-axis.

Compared to traditional spherical lenses, asphere lenses have curvatures that gradually change across the surface to maintain a uniform focus across the field and correct for off-axis distortions.

Edmund Optic's TECHSPEC #84-328 is the lens we considered for this design. It is an achromatic doublet that contains a UV-VIS coating with a 6.25mm diameter, and a 15mm effective focal length. The substrates of this lens contain an N-FK5 as the first crown element of the doublet and an F2HT as the flint element. This is the lens we chose as our corrector lens.

Figure 16: Transmission of a UV-VIS Coated Lens

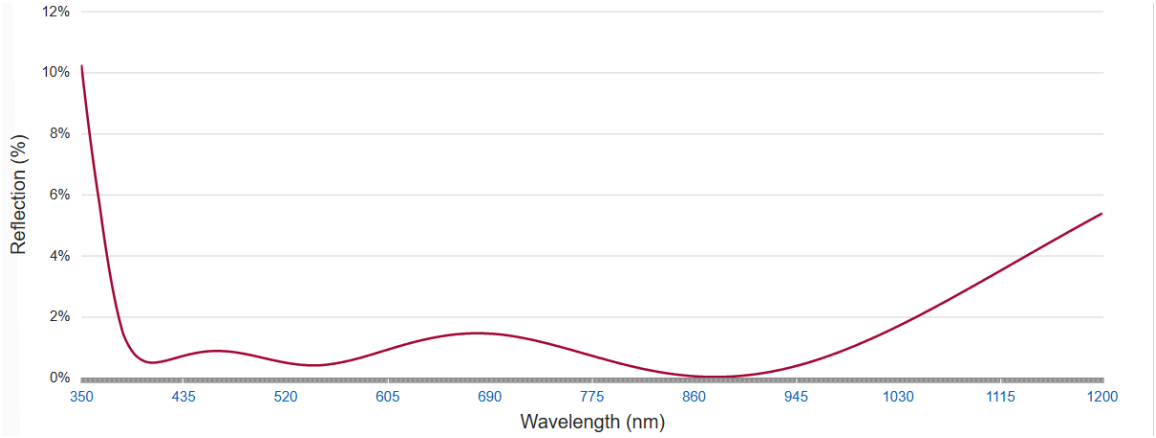


Imaging Lens Selection

The imaging lens acts as the final lens for this commercial off the shelf three lens system. The aim of this lens is to control the effective focal length of the overall system to achieve the desired magnification. With the previously calculated 6.1mm design, it is imperative that our effective focal length reaches that target. Choosing a proper imaging lens is important as it serves as the determinant of the focus and resolution of our final output image onto the sensor plane. Based on its position, this lens also helps us achieve the magnification needed for S.T.A.R. to image the stage. Since this element is closest to the detector, it is highly subject to internal reflections, so we aimed to find a lens with a VIS-NIR anti-reflection coating.

The lens selected for this component is Edmund Optic's TECHSPEC #63-714 which contains the anti-reflection coating we sought for, as well as the shape of an achromatic lens. For the targeted wavelengths our sensor can accept, the coating provides an extremely minimum reflection percentage, targeting under 2% from 400nm-860nm, as shown in figure 17. This helps prevent the back reflections that can be incident on our imaging sensor.

Figure 17: Reflectance (%) of a VIS-NIR Coated Lens



3.2.5 Existing Camera Imaging Technologies

CCTV Surveillance Cameras

CCTV lens systems are a compact, multi-element optical design that serves as the imaging core of many surveillance and machine-vision cameras. Often in the C-mount or CS-mount fixture types, the optical design is heavily complex. Including typically between four and eight glass elements, these systems incorporate achromatic doublets, or aspheric surfaces in order to keep a compact fixture capable of reducing aberrations. These cameras typically operate at fast apertures such as f/1.4 to f/2.8 in order to accommodate low-light surveillance. For S.T.A.R., this would be a great system to incorporate, as the compactness and low-light capabilities would be suitable for this design. However, with the complexity of the design, and the amount of optical components involved, cost plays a factor into choosing this as the basis for our design, as it is hard to achieve the quality and compactness without the use of an aspheric lens, an expensive component to purchase off the shelf.

3.3 Software

The S.T.A.R. spotlight tracking system uses a cross-platform mobile app paired with a lightweight Raspberry Pi server to deliver low-latency control and feedback over Wi-Fi. The app provides start/stop, manual override, and target selection; the Pi brokers real-time commands to the ESP32 and streams sensor/vision feedback to the app. To meet system specs (e.g., ≤250 ms end-to-end latency; stretch goal ≤150 ms), the software stack prioritizes predictable frame times, minimal per-message overhead, and stable reconnection behavior under transient network loss.

3.3.1 Computer Vision Tools

3.3.1.1 OpenCV

OpenCV provides a full pipeline on the Raspberry Pi—camera capture, image processing, DNN inference for person detection, classical trackers like CSRT/KCF, and lightweight filtering—under one API. This keeps frame times predictable and minimizes glue code, letting us detect a person with a compact model, convert the box to a torso region of interest, track between detector frames to reduce jitter, and emit stable UART commands. Its model-agnostic DNN runtime means we can swap MobileNet-SSD and YOLO-tiny/nano without changing the rest of the system, delivering an efficient, edge-friendly path to meet the ≤150–250 ms budget.

3.3.1.2 TensorFlow Lite

TensorFlow Lite is a compact inference engine that runs TensorFlow models efficiently on edge devices using quantization and hardware delegates when available. It excels at fast, predictable forward passes for small detectors, but it focuses on inference only, so camera I/O, preprocessing, tracking, and visualization still rely on OpenCV. In our pipeline it serves as an optional accelerator for the detector step if a TFLite model proves faster than OpenCV's built-in DNN backend, while the rest of the stages stay in OpenCV for simplicity.

3.3.1.3 PyTorch / TorchScript

PyTorch is a research-first deep learning framework that enables rapid training and experimentation, with TorchScript or ExecuTorch used to package models for deployment. It offers strong flexibility for custom detectors and fine-tuning but introduces heavier packaging and conversion steps on the Pi compared to TFLite. For S.T.A.R., PyTorch is most efficient when we need to train or customize a model, after which the exported artifact can be dropped into the same OpenCV-based runtime for capture, post-processing, tracking, and command generation.

3.3.1.4 MediaPipe

MediaPipe provides optimized, ready-made perception graphs such as Pose, Hands, and Face that run in real time on mobile and edge hardware. It is highly efficient for landmark extraction and can deliver upper-body keypoints useful for refining a torso target, but it is more task-oriented and less flexible for arbitrary detectors. Within S.T.A.R., MediaPipe is an optional add-on when landmark precision is needed; otherwise, an OpenCV detector plus tracker remains simpler and more efficient for end-to-end control.

Table 7– Computer Vision Tool Comparison

Criterion	OpenCV	Tensorflow	PyTorch /	MediaPipe

		LIte	Torchscript	
Primary Role	End-to-end CV toolkit	Edge inference for TF Models	Research DL; deploy via Torchscript/Executorch	Prebuilt perception graphs (pose/face/hands)
Edge Readiness	High (CPU/NEON; broad backends)	High (quantization, delegates)	Medium (heavier packaging on Pi)	High (mobile/edge focused)
Pipeline breadth	Wide (capture→infer→track→filter)	Narrow (inference only)	Narrow→Medium	Medium (task-focused)
Integration effort	Low	Medium	Medium-High	Medium

Choice: Use OpenCV as the unified runtime: detect the "person" class with a compact DNN (MobileNet-SSD or YOLO-tiny/nano via OpenCV DNN), convert to a torso ROI with a simple geometric slice of the box, then track the ROI between detections (KCF/CSRT) and low-pass filter the target.

3.3.2 Cross-Platform Mobile Application Frameworks

3.3.2.1 Flutter (Dart)

Flutter compiles Dart AOT to native ARM and renders via its own C/C++ engine (Impeller/Skia), avoiding a JS–native bridge and yielding consistent, pixel-perfect rendering across platforms. Because Flutter owns the entire rendering pipeline, frame scheduling is highly deterministic: animation and gesture handling share a single 16.67 ms frame budget (60 FPS) with fewer cross-thread/context switches than bridge-based approaches. AOT improves cold start and reduces JIT warm-up hiccups during first interactions, while Dart isolates help offload CPU work (e.g., JSON encode/decode, telemetry parsing) off the main thread to reduce jank under rapid control streams. Trade-offs include larger binary sizes and the need to learn Dart; platform-specific integrations (Wi-Fi/Bluetooth, camera) use plugins/native channels that should be profiled when pushing high-frequency updates.

Visually rich, custom UI with steady 60 FPS while streaming control events or rendering overlays (reticles, status toasts) where consistent look-and-feel across iOS/Android matters.

3.3.2.2 React Native (JavaScript)

React Native uses JavaScript/JSX to drive native UI widgets via a bridge between JS and platform modules, leveraging existing web/React experience and adopting each OS's native look-and-feel out of the box. The JS–native bridge introduces scheduling/serialization overhead under bursty, high-frequency updates (e.g., joystick deltas at 60–120 Hz). To keep frames under 16.67 ms, heavy parsing/work should move to native modules or TurboModules/JSI, and control messages should be batched (coalesced per frame) to reduce bridge chatter. Startup is fast if the bundle is cached, though initial interaction can contend with JS engine warm-up/GC. RN excels at native UI fidelity with minimal custom drawing; for fully custom realtime overlays, a Canvas/Skia layer or GL-backed view is recommended. Teams strong in JS/React, native look preferred, and control messages can be rate-limited/coalesced to keep bridge traffic predictable.

Table 8 – Cross-Platform Frameworks (Performance-centric)

Criterion	Flutter	React Native
Frame-time predictability	High (engine-owned pipeline)	Medium (bridge variability)
Per-event overhead	Low (in-engine)	Medium (bridge serialization)
Startup behavior	AOT helps cold start; larger APK/IPA	Fast with cached bundle; JS init/GC spikes possible
Heavy UI animation	Very strong at 60 FPS	Good; prefer native/Skia views for custom
Team ramp-up	Learn Dart; great tooling	Leverage JS/React skills
Best fit here	Custom, consistent UI w/ frequent updates	Native feel; updates coalesced/batched

3.3.3 Raspberry Pi Software Stack

3.3.3.1 Language & CV: Python + OpenCV on Pi

Python on the Pi enables rapid development and strong OpenCV support (detection/tracking, camera I/O). On a small CPU budget, practical patterns include: (1) downscaling frames for detection (e.g., 320–640 px short side) then mapping centroids to full-res; (2) running heavy detection at a lower cadence (10–20 Hz) while a lightweight tracker (centroid/Kalman) interpolates at camera rate; (3) placing vision and I/O into

separate asyncio tasks with bounded queues to avoid head-of-line blocking. These keep the control loop responsive even as camera fps increases.

3.3.3.2 Backend Framework: FastAPI (ASGI)

FastAPI with Uvicorn keeps control messages on an async event loop, avoiding one-thread-per-request overhead typical of WSGI. WebSocket handlers can push/await without blocking, structured models (Pydantic) serialize efficiently, and Swagger UI accelerates testing. On a Pi, this lowers CPU per message and improves tail latencies under bursts compared to synchronous frameworks, while keeping concurrency/backpressure straightforward for live streams.

3.3.3.3 Alternative: Flask (WSGI)

Flask is simple and familiar for REST, but real-time push usually requires Flask-SocketIO with eventlet/gevent, adding complexity/overhead. Under frequent small messages, WSGI worker/thread contention widens latency variance. It remains a solid choice for low-frequency configuration endpoints but is less ideal for sustained streaming.

Table 9 – Python Web Frameworks on Pi (Performance)

Criterion	FastAPI (ASGI)	Flask (WSGI)
Concurrency	Async event loop	Threads/processes
WebSockets	Native (Starlette)	Add-on (SocketIO + eventlet/gevent)
CPU per msg	Lower (no thread handoff)	Higher (context switching)
Backpressure	Await/queues in-loop	Manual; risk of blocking
Best role	Real-time control + REST	Light REST endpoints

3.3.4 Wireless Communication Methods

3.3.4.1 Wi-Fi (Infrastructure LAN)

Phone and Raspberry Pi join the same access point (venue Wi-Fi). Provides the most stable bandwidth and lowest jitter for continuous control plus optional preview/video. Native IP simplifies discovery (mDNS) and secure sessions (TLS). Ideal for demonstrations and multi-observer dashboards.

3.3.4.2 Wi-Fi Direct / Pi Hotspot

Raspberry Pi hosts its own secured SSID; the app connects directly in field scenarios without venue Wi-Fi. Delivers similar performance to infrastructure mode with slightly more connection management (SSID credentials, captive portal avoidance). Recommended as the offline fallback.

3.3.4.3 Bluetooth Classic (SPP/RFCOMM)

Serial-over-Bluetooth provides straightforward command channels with simple framing. Throughput and profile overhead make it less suitable for video/preview and high-rate telemetry, but it can support low-rate control and status where RF congestion makes Wi-Fi unreliable.

3.3.4.4 Bluetooth Low Energy (BLE/GATT)

Optimized for low power and small messages. Excellent for pairing, wake, and minimal control (e.g., start/stop, mode toggles), but constrained MTU and throughput make it a poor fit for streaming telemetry or preview. Useful as a companion/control plane—not the data plane.

Table 10 – Wireless Communication (Performance)

Criterion	**Wi-fi (LAN)**	**Wi-fi Direct / Pi Hotspot**	**BT Classic (SPP)**	**BLE (GATT)**
Setup	Join AP; mDNS	Join Pi SSID	Pair; open SPP	Pair/bond; discover GATT
Range	Good (via AP)	Short–moderate	Short–moderate	Short
Throughput	50–200+ Mbps	20–100+ Mbps	~1–3 Mbps	~0.125–2 Mbps
Latency	Low, stable	Low; RF-dependent	Moderate	Low (small MTU)

Multi-Client	Easy (IP)	Mostly single ctrl	Limited	Limited
Security	WPA2/3 + TLS	WPA2 PSK + app auth	Link key	Pairing + encrypt
Best Use	Primary link (control + telemetry + preview)	Offline fallback	Basic control/status	Companion for quick-pair/minimal control

Choice: Use Wi-Fi as the primary wireless link—Infrastructure LAN when available, with Pi Hotspot as the no-infrastructure fallback. Keep BLE as an optional companion channel for quick-pair, wake, or minimal control in high-interference environments; avoid Bluetooth Classic and BLE for any video/preview or high-rate telemetry.

3.3.5 Communication Methods

We compared HTTP/REST, WebSocket, and MQTT for app↔Pi command/telemetry. For tight closed-loop control, persistent bidirectional messaging is preferred to meet the ≤150–250 ms end-to-end budget.

3.3.5.1 HTTP/REST

Stateless request/response calls are ideal for configuration and infrequent actions. Each command incurs HTTP method/headers and, unless kept-alive, connection/TLS handshakes; even with keep-alive, request framing and server routing add non-trivial overhead. Polling for feedback (vs push) either increases latency (long intervals) or CPU/network (short intervals). This is suitable for setup, logs, and firmware/version checks, not for continuous joystick/pose streaming.

3.3.5.2 WebSocket

A single HTTP upgrade creates a long-lived, full-duplex TCP channel with very small per-message headers. Both directions can push asynchronously with minimal context switching, enabling high-rate control (e.g., 20–100 Hz deltas), immediate acknowledgments, and event-driven backpressure. Each connected client holds state on the server; with one controller per spotlight this is trivial. Mature libraries provide heartbeats and robust reconnects.

3.3.5.1 MQTT

MQTT provides lightweight pub/sub via a broker (e.g., Mosquitto) with QoS levels and retained messages, excellent for multi-client dashboards or intermittent connectivity. Headers are tiny and latency is low; the broker adds a minor LAN hop. QoS 1/2 add ack rounds for delivery guarantees; tune per topic (e.g., control at QoS 0, telemetry at QoS 1). Overhead is broker management—often unnecessary for a single app↔spotlight pair today, but the clean path if multiple operators or a cloud mirror are added later.

Table 11 – Control Channel Comparison

Criterion	HTTP/REST	WebSocket	MQTT
Setup cost	Request per action; keep-alive helps	One upgrade; then steady state	Broker + client connects
Push capability	No (client-pull only)	Yes (bi-directional)	Yes (publish/subscribe)
Per-msg overhead	High (headers)	Low (frame header)	Low (tiny header)
Backpressure	Ad-hoc	Event-loop awaits/queues	Broker buffer + QoS acks
Reconnect	Per request	Heartbeats & fast resume	Retained messages + QoS
Best use here	Config, logs	Primary real-time control	Future multi-client/IoT

Choice: WebSocket for low-latency control; REST for configuration; MQTT reserved for multi-client expansion.

3.3.6 Embedded Communication

3.3.6.1 UART

A framed, line-oriented protocol (start byte, command ID, payload, checksum, end byte) with sequence numbers and ACK/NACK provides reliable delivery at 115.2–921.6 kbps. This comfortably supports control deltas (pan/tilt targets, mode changes) at tens of Hz with negligible link latency. Timeouts and limited retries on the Pi side align with the app's heartbeat so faults propagate cleanly.

3.3.6.2 I2C

Best used on the ESP32's local bus for devices such as a PCA9685 PWM/servo driver or auxiliary sensors. It reduces wiring inside the actuator enclosure and leverages addressing on a short, well-terminated bus. Not preferred for the Pi↔ESP32 interconnect due to open-drain signaling, pull-up tuning, and shared-bus jitter.

3.3.6.3 SPI

Offers the highest throughput and most deterministic timing via a dedicated clock. It becomes attractive only if future requirements include bulk binary transfers (e.g., dense motion profiles or high-rate sensor data) across the Pi↔ESP32 boundary. The additional wiring and chip-select handling are unnecessary for the present command/ack workload.

Table 12 – Embedded Communication Comparison

Method	**Wires**	**Clocking**	**Practical Rate**	**Determinism & Jitter**	**Pros**	**Cons**
UART	TX, RX, GND	Asynchronous	115.2 kbps–1 Mbps	Good with framed packets and acks	Very simple wiring; easy debugging; resilient	No shared addressing; half-duplex per line pair
I2C	SCL, SDA (+pull-ups), GND	Synchronous (open-drain)	100 kHz/400 kHz/1 MHz	Bus arbitration; moderate jitter under load	Two-wire multi-drop; addressing	Pull-ups + bus integrity; not ideal for long runs
SPI	SCLK, MOSI, MISO, CS, GND	Synchronous (push-pull)	1–20+ MHz	Very good, clocked	Highest throughput; deterministic	More wires; chip select mgmt

Choice: UART for the Pi↔ESP32 primary control channel; I²C reserved for ESP32-local peripherals; SPI deferred for possible high-bandwidth future extensions.

3.3.7 ESP32 Firmware

3.3.7.1 Language & Runtime (C/C++)

C/C++ with Arduino core or ESP-IDF provides deterministic access to UART, PWM (LEDC), GPIO, timers, and FreeRTOS. Lower abstraction yields tighter loops and lower jitter than interpreted environments—ideal for servo timing. For PWM/LEDC, select frequency/resolution to preserve duty precision and avoid servo chatter (e.g., 50–300 Hz typical for hobby servos; higher update rates demand careful filtering). For the UART link, 115 200 bps is a simple baseline; compact, coalesced frames keep well below saturation. If dense telemetry is later required, 460 800–921 600 bps is available with ESP-IDF (validate cable integrity/level shifting). With FreeRTOS, separate UART RX/TX, motion control, and safety/limit checks into distinct tasks with priorities; use queues/semaphores to bound latency and protect timing.

3.3.7.2 IDE/Tooling

Arduino IDE accelerates iteration and serial debugging through simple builds and a rich library ecosystem. PlatformIO + ESP-IDF is available for lower-level drivers, custom build flags, and partition tuning when tighter control or optimization is needed.

Table 13 – Platform Language Trade-offs

Criterion	C/C++ (ESP32)	Python (Pi)
Execution	Native (no VM)	Interpreted
Jitter	Very low (timers/ISRs)	Low–moderate (GC/GIL; mitigated with async)
Hardware access	Direct (ESP-IDF/Arduino)	Via bindings (serial, GPIO)
Role	Motor control, PWM, UART	Vision, orchestration, server

3.3.8 End-to-End Interaction & Timing Budget (How it Works)

1. **App → Pi (Control).** The app maintains a WebSocket and streams control deltas (start/stop, joystick, target). Rate-limit (e.g., 30–60 Hz) and coalesce per frame to keep UI smooth. The server acknowledges immediately to timestamp round-trip.

2. **Pi (Server).** FastAPI routes messages to async handlers that write compact binary frames over UART to the ESP32 (e.g., header, command, checksum).

Backpressure is handled via asyncio queues to avoid blocking the WebSocket loop.

3. **ESP32 (Actuation).** Firmware parses the frame and updates LEDC targets; limit-switch logic clamps setpoints. Servo updates run on a periodic timer; UART RX uses a ring buffer to avoid drops.

4. **Pi (Feedback).** The vision loop (Python/OpenCV) computes torso centroid and tracking confidence; a lightweight tracker interpolates between detector hits. Feedback is emitted to the app over the same WebSocket and throttled to what the UI can render smoothly.

Table 14 – Control-Loop Budget (Design Targets)

Segment	Typical Work	Budget Notes
App UI frame	Gesture → delta coalesce	Keep < 16.67 ms; avoid heavy JSON on main thread
App→Pi transport	WebSocket send	Small binary frames; heartbeat for loss detection
Pi handling	Async route → UART	Avoid blocking; bounded queues for bursts
UART	Bytes on wire	115 200 bps baseline; compact frames minimize latency
ESP32 actuation	Parse → PWM setpoint	Timer-driven; prioritize motion over logs
Feedback return	Telemetry/centroid	Throttle to UI frame rate

Selected Stack (recap with performance rationale)

- **Mobile**: Flutter (selected; deterministic 60 FPS); React Native as alt (native feel; batch/JSI for heavy work).\
- **Pi backend**: Python + FastAPI (ASGI) + OpenCV; OpenCV DNN (YOLO-tiny/MobileNet-SSD), detect-every-N with CSRT/KCF tracking and EMA/Kalman smoothing.
- **Communication**: Wi-Fi (LAN primary, Pi Hotspot fallback) + WebSocket for real-time; REST for config; MQTT later for multi-client.

- **ESP32 firmware**: C/C++ (Arduino/ESP-IDF) with FreeRTOS tasks; UART 115200 bps framed (ACK/NACK); LEDC PWM for servos; I²C for local peripherals.

3.8 Hardware

3.8.1 SoM

When selecting a computational platform, one of the key design decisions involves choosing between a System-on-Module (SoM) and a Single-Board Computer (SBC). Both serve the same fundamental purpose — providing the processing, memory, and interface capabilities needed to run advanced algorithms — but they differ significantly in flexibility, scalability, and integration complexity.

A System-on-Module, such as the Raspberry Pi Compute Module 5, integrates the processor, GPU, memory, and storage into a compact, self-contained board that interfaces with a custom-designed carrier board. This modular approach offers a high degree of design flexibility, allowing engineers to tailor the carrier PCB to the project's specific requirements. The carrier board can include only the necessary interfaces reducing system complexity and improving reliability. It also allows designers to implement custom power distribution and noise filtering circuits to enhance stability. While the initial engineering and PCB design costs are higher, the result is a more compact, efficient, and professional-grade embedded system that can be easily scaled or upgraded by replacing the SoM without redesigning the entire system.

In contrast, a Single-Board Computer, such as the Raspberry Pi 5, provides a complete computing platform with all necessary I/O ports, power management, and connectors on a single board. SBCs are highly advantageous for rapid prototyping and early-stage development, as they eliminate the need for custom hardware design. Developers can immediately begin testing software, algorithms, and peripherals such as the camera and motor controller without concern for hardware integration. SBCs also benefit from mature software support, well-documented drivers, and extensive community resources, which can accelerate debugging and reduce setup time. However, this convenience comes at the cost of reduced flexibility and a larger physical footprint. Because the I/O and power architecture are fixed, it becomes challenging to optimize the system for compact enclosures or specialized power domains. SBCs are also more susceptible to electrical noise and thermal inefficiencies when used in close proximity to high-power components.

The Compute Module 5 was originally chosen because of its excellent processing power for computer vision tasks and the ability to create a completely custom carrier board for I/O. However, highspeed boards were deemed too complex. HDMI needs very specific PCB requirements to function correctly. In the interest of cost-savings and time constraints, we have decided to not design a custom carrier board. Nevertheless, we will still be using a CM5 and CM5IO board. The PCB requirement will be satisfied by a custom MCU board, custom motor controller, and custom LED and switch terminal board.

Table 15 – SoM and SBC Comparison

Attribute	System-on-Module (SoM)	Single Board Computer (SBC)
Integration Level	Core computing components only (CPU, RAM, storage, GPU). Requires carrier board.	Fully integrated with I/O ports, power management, and interfaces on a single PCB.
Design Flexibility	Highly customizable—carrier board can be tailored for specific project needs.	Limited customization—fixed I/O layout and hardware configuration.
Scalability	Easier to upgrade by replacing the compute module without redesigning the carrier.	Typically requires replacing the entire board to upgrade hardware.
Manufacturing Cost	Higher initial design cost but scalable for production runs.	Lower cost for prototypes and small projects.
Thermal Management	Can be optimized with custom heat dissipation solutions.	Fixed thermal design based on vendor layout.

3.8.2 MCU

One of the most critical hardware decisions involves selecting between a microcontroller unit (MCU) and a field-programmable gate array (FPGA). Both technologies are capable of handling real-time control and signal processing tasks, but they differ substantially in terms of programmability, complexity, cost, and overall suitability for specific applications. The decision ultimately rests on balancing computational performance, design flexibility, and system efficiency.

FPGA development typically requires advanced expertise in hardware description languages (such as Verilog or VHDL), longer design cycles, and specialized toolchains. FPGAs also tend to consume more power and are substantially more expensive than MCUs, making them less practical. While it is technically feasible to implement motor control or communication logic on an FPGA, these functions are more efficiently executed by an MCU, which already provides built-in peripherals and optimized hardware support for such operations.

Table 16 – MCU and FPGA Comparison

Attribute	MCU (e.g., ESP32-S3)	FPGA
Processing Type	Sequential software execution on processor cores.	Parallel hardware logic execution with reconfigurable gates.

Programming Complexity	Easy—uses C/C++ or MicroPython.	High—requires HDL (VHDL/Verilog) and synthesis tools.
Development Speed	Rapid—large ecosystem and open-source libraries.	Slower—requires detailed logic design and timing closure.
Power Consumption	Low; ideal for embedded and battery-powered systems.	High; often requires cooling and significant power overhead.
Cost	Very low (<$10 for ESP32-S3).	High ($50–$200+ depending on FPGA).
Best For	General-purpose control, networking, real-time tasks.	High-speed signal processing or custom hardware pipelines.
Reason Chosen for S.T.A.R.	The ESP32-S3 provides real-time motor control, Wi-Fi, and I²C communication in a compact, low-cost package.	FPGA unnecessary for S.T.A.R.'s moderate control and vision needs.

For the S.T.A.R. system, the ESP32-S3 microcontroller was chosen over an FPGA because it offers a superior balance of computational capability, cost, and ease of integration.

The ESP32-S3 was chosen over other popular microcontrollers such as the STM32 family, Arduino Portenta, or Teensy 4.x series because it offers the best balance between performance, cost, wireless connectivity, and ecosystem maturity for the specific requirements of our system. While each alternative MCU family has its own strengths, the ESP32-S3's combination of computational capability, built-in wireless interfaces, and developer accessibility makes it uniquely well-suited for a distributed embedded system that demands both real-time control and networked communication.

Table 17 – ESP32 and STM32 Comparison

Attribute	**ESP32-S3**	**STM32 (e.g., STM32F4/F7)**
CPU Architecture	Dual-core Xtensa LX7 @ 240 MHz	ARM Cortex-M4/M7 @ up to 480 MHz

Wireless Connectivity	Integrated Wi-Fi + Bluetooth 5.0	Requires external modules
AI/ML Support	Built-in vector instructions for AI acceleration	Limited DSP capability
I/O Interfaces	I²C, SPI, UART, PWM, USB, ADC, DAC	Extensive peripheral support
Ease of Development	Excellent—supported by Arduino, ESP-IDF, and MicroPython	Good—requires STM32CubeIDE
Cost Efficiency	Very low (<$10)	Moderate ($10–$30)
Chosen for S.T.A.R.	Offers wireless connectivity, powerful dual-core processing, and efficient servo control via I²C (PCA9685).	STM32 better for high-precision control but costlier and lacks wireless integration.

3.8.3 Motors

Servo motors were chosen for the S.T.A.R. system due to their precision control, integrated feedback mechanisms, and smooth, predictable motion characteristics, all of which are essential for maintaining accurate spotlight positioning and stable target tracking. Unlike stepper motors, which move in discrete increments and can exhibit vibration or "jitter" during operation, servos provide continuous positional feedback via an internal potentiometer or encoder, allowing for closed-loop control. This ensures that the spotlight's pan and tilt axes remain steady, even under dynamic tracking conditions where frequent adjustments are required. The result is a system capable of fluid, responsive movement that mirrors professional-grade gimbal performance—ideal for live performance environments where smooth motion is crucial to visual quality.

Another advantage of using servo motors is their simplified control architecture. Each servo integrates a motor, feedback sensor, and driver circuit into one compact package, allowing for direct control via PWM (Pulse Width Modulation) signals. This drastically reduces the external circuitry required compared to stepper or DC motors, which often need complex H-bridge drivers and external encoders for feedback. By interfacing the servos through an Adafruit PCA9685 PWM driver, the system can control multiple channels simultaneously with high precision and minimal signal noise. Furthermore, servos inherently hold their position when idle, maintaining the spotlight's orientation without consuming additional processing power or continuous correction signals from the ESP32-S3 microcontroller.

The torque ratings for the selected servo motors were determined through a "more-is-better" mentality. We have chosen 35kg-cm servos for the gimbal. However, if they are found to be too weak in testing, we will swap out the servos for ones of higher torque rating. Since the stage will always be in front of the spotlight, 360 degree rotation is not necessary for our motors.

3.8.4 Motor Controllers

There are several available motor controller options for driving servo motors, each varying in complexity, channel capacity, communication interface, and control precision. Common servo control solutions include direct microcontroller PWM generation, dedicated servo driver ICs (such as the PCA9685 or TLC5940), and advanced motion control boards (such as Pololu Maestro or Adafruit 16-Channel PWM/Servo Driver). Each has its advantages, but for a project like S.T.A.R. —which requires reliable, synchronized, multi-axis servo control integrated with embedded processing—the Adafruit PCA9685 stands out as the most practical and efficient choice.

Direct PWM generation through a microcontroller, such as the ESP32-S3, is a simple and inexpensive method for controlling a few servos. The ESP32 can generate PWM signals using internal timers, allowing positional control via software. However, this approach becomes less practical as the number of servos increases, since PWM signal generation consumes significant processor resources and can introduce timing jitter if the CPU is simultaneously handling computationally intensive tasks—such as vision processing or wireless communication. For this reason, the S.T.A.R. system required a dedicated servo driver that offloads PWM signal generation from the main controller, freeing the ESP32-S3 to focus on system logic, camera interfacing, and communication tasks.

The Adafruit PCA9685 was selected due to its high channel count (16 channels), I2C communication interface, and precision 12-bit PWM resolution. These features provide smooth, accurate positional control and eliminate jitter in servo movement. Because it communicates over I2C, the PCA9685 can be easily interfaced with both the ESP32-S3 and the Raspberry Pi Compute Module 5, enabling a flexible architecture where the Raspberry Pi manages vision and tracking algorithms while the ESP32 handles real-time motion control. The PCA9685's ability to generate PWM signals independently of the microcontroller ensures consistent timing and reliable performance, even under heavy processing loads.

Another key advantage is that the PCA9685 includes a built-in oscillator, meaning it does not rely on the timing accuracy of the controlling microcontroller. This guarantees consistent servo response regardless of CPU load or software timing fluctuations. Additionally, the driver supports an external power source for the servo rail, allowing high-current motors to be powered independently of the logic circuitry—essential for protecting sensitive electronics like the Raspberry Pi CM5 and ESP32-S3 from voltage dips or noise caused by servo operation.

Alternatives, such as the Pololu Maestro or TLC5940, offer advanced features like scripting or current feedback, but they come at significantly higher cost and complexity. The Maestro, for example, is better suited for robotics applications that require trajectory planning or multi-servo choreography, while the PCA9685 provides a simpler and more cost-effective solution optimized for smooth PWM control and scalability.

In summary, the PCA9685 was chosen because it offers high-precision, multi-channel PWM generation over I2C, offloads timing control from the main processor, and supports independent servo power input—all in a compact, affordable package. These attributes make it ideally suited for S.T.A.R.'s two-axis servo gimbal system, where reliable, synchronized motion and system stability are paramount.

Table 18 – Motor Controller Comparison

Attribute	**PCA9685 (Adafruit)**	**TLC5940**	**Pololu Maestro**
Control Type	12-bit PWM over I2C	12-bit PWM over SPI	USB/UART control with scripting
Channels	16 independent servo channels	16 channels	6–24 channels (depending on model)
Communication Interface	I2C (addressable up to 62 devices)	SPI (faster, but fewer devices)	USB/UART (standalone controller)
Ease of Integration	Very easy—widely supported by Arduino, ESP32, and Raspberry Pi	Moderate—requires more configuration	Easy, but adds software complexity
Power Handling	Separate servo power rail; supports up to 6V	External current sink required	Integrated power management per servo
Cost	~$10	~$7	$25–$50
Chosen for S.T.A.R.	Offers high channel count, reliable timing, and I2C compatibility with ESP32-S3 for smooth servo control.	TLC5940 lacks flexibility for multi-servo synchronization.	Maestro provides advanced features but is too costly and over-specified.

3.8.5 PSU

The power supply selection was a critical design consideration, as it directly influences system stability, electrical safety, and performance reliability. After evaluating the total current draw and voltage requirements of the major subsystems—including the servo motors, motor controllers, and embedded electronics—a 24V 25A power supply unit (PSU) was selected. This configuration provides a total available power of 600W, which ensures adequate headroom for peak loads while maintaining efficiency and thermal stability during continuous operation.

The choice of 24V as the main supply voltage was driven by the power requirements of the high-torque servo motors used to drive the two-axis gimbal. These servos demand relatively high current, particularly during rapid acceleration and deceleration cycles when both torque and current draw peak. Operating them at 24V allows for more efficient torque delivery and reduced current flow compared to lower-voltage systems (such as 12V), which in turn minimizes resistive losses along the power lines.

The 25A current capacity was selected to provide sufficient margin above the nominal load to prevent voltage sag or instability during high-power transients. Servos, in particular, can draw several amps each under full load.

A 600W PSU provides flexibility for powering auxiliary electronics such as cooling fans, control boards, and sensor modules without the risk of overloading the main power rail. To enhance system reliability, individual subsystems are isolated using voltage regulators or buck converters that step down the 24V rail to the required logic levels (e.g., 5V for the PCA9685, 3.3V for the ESP32-S3).

In summary, the 24V 25A PSU was chosen to deliver a balance of power, efficiency, and headroom suitable for a system combining high-torque actuation with sensitive embedded control electronics. It ensures that the S.T.A.R. system can maintain stable operation during rapid servo movement while providing ample reserve power for future scalability and operational safety.

3.8.6 Buck Converter

A buck (step-down) switching converter is the preferred topology for converting our system's high-voltage rail (e.g., 24 V) down to the logic and servo rails (5 V, 3.3 V) primarily because of efficiency and thermal advantages. A buck converter achieves high efficiency by switching a transistor rapidly and using an inductor to store and release energy; only the difference between input and output power (plus converter losses) is dissipated as heat. This makes buck converters especially appropriate when the input-to-output voltage difference is large and when the load current is moderate to high — exactly the case when stepping 24 V down to 5 V or 3.3 V to feed servos, drivers, and SBCs. The high efficiency reduces heat generation, relaxes thermal management requirements (smaller heatsinks and fewer fans), and extends component life.

A linear regulator (LDO), by contrast, drops voltage by dissipating the voltage difference across a pass element as heat. The power loss is simply the voltage drop multiplied by current, so when Vin is much higher than Vout and currents are significant, the heat becomes excessive — sometimes impractically so. LDOs are attractive only in scenarios with small voltage drops, very low currents, or when very low output noise is required for sensitive analog circuits. In practice for S.T.A.R., using LDOs alone to step 24 V to logic rails would require large heatsinking and greatly increase thermal stress and inefficiency.

Table 19 – Buck and LDO Comparison

Attribute	**Buck Converter (Switching)**	**LDO (Linear Regulator)**
Efficiency	High (typically 80–95% depending on load and design).	Low when Vin ≫ Vout (efficiency = Vout/Vin).
Heat dissipation	Low (most energy converted, small losses).	High (dissipates (Vin−Vout)×I as heat).
Suitability for 24 V → 5 V at ≥1 A	Excellent — recommended for primary conversion.	Poor — generates excessive heat; not recommended.
Output noise & ripple	Switching noise and ripple present; requires filtering (good layout, LC filters, low-ESR caps).	Very low noise; excellent for sensitive analog/ADC/clock rails.
Transient response	Good, but requires loop compensation; faster switching designs respond well.	Very fast and clean transient response for small load steps.
Complexity and BOM	Higher: inductor, diode/synchronous FETs, switching IC, output/input capacitors, layout care.	Lower: single component, few external caps.
Size	Can be compact but requires inductor (board area).	Very compact, minimal components.
EMI considerations	Requires EMI mitigation (layout, filtering, shielding).	Minimal EMI.
Quiescent current (standby)	Can be low; some designs have µA standby, others higher.	Often very low quiescent versions exist (suitable for battery standby).

Cost	Moderate (higher component count).	Low (cheap and simple).

3.8.7 WAGOs

Wago connectors were chosen for the S.T.A.R. system primarily due to their reliability, safety, and ease of maintenance when dealing with multiple power and signal connections within a complex electromechanical setup. Unlike traditional screw terminals or soldered joints, Wago lever or push-in connectors provide a tool-free, vibration-resistant connection that ensures long-term electrical integrity — a crucial factor in systems subject to movement, such as a motorized pan-tilt spotlight.

From a technical standpoint, Wago connectors employ a spring-clamp mechanism that maintains consistent contact pressure on the wire conductor, even under thermal expansion or vibration. This design minimizes the risk of loose connections, arcing, or intermittent faults that could otherwise occur in high-current or mobile applications. In the S.T.A.R. project, this is particularly important since the 24V, 25A power rail supplies multiple subsystems (such as the servo motors, buck converters, and control electronics) that must remain electrically stable at all times.

Another major advantage is modularity and serviceability. The S.T.A.R. system may undergo iterative testing and component swaps during development. Wago connectors allow rapid prototyping and safe reconfiguration of the wiring harness without soldering or crimping tools, reducing downtime and minimizing the potential for wiring errors. The connectors also support wide wire gauge ranges and are compatible with both solid and stranded conductors, making them ideal for mixed-signal and power distribution setups found in this project.

In addition, the compact form factor and clean aesthetic of Wago connectors contribute to a neat and professional internal layout of the 3D-printed enclosure. This enhances not only safety and organization but also compliance with good electrical design practices, such as maintaining proper creepage and clearance distances between high- and low-voltage lines.

Chapter 4 - Standards and Design Constraints

4.1 Applicable Standards

Designing the S.T.A.R. system means balancing real-world engineering limits with a system that is safe, reliable, and practical. Even though this is a student project, the final device still needs to follow the general ideas behind lighting industry standards and safety guidelines. Since this is a lighting fixture with electrical components, moving parts, and bright illumination, the design process has to consider electrical safety, heat management,

optical performance, and user safety. This chapter explains the standards and constraints that shaped our decisions and how they influenced the design.

Even though student prototypes don't go through official certification processes, it's still important to understand the standards used in professional lighting equipment. These standards help guide design decisions so that the final system operates safely and reliably.

Electrical Safety – NFPA 70 (NEC)

NFPA 70 an important standard for the S.T.A.R. project because it focuses on how electrical systems are wired, grounded, and installed safely. Since S.T.A.R. uses a high-power LED source, control electronics, and motor drivers, the entire system needs to follow the NEC's rules for proper wiring, current limits, and overcurrent protection. This includes using the right wire gauge for the load, adding fuses or circuit breakers where needed, and making sure all metal parts of the fixture are properly grounded to prevent electric shock. The NEC also sets standards for cable insulation, strain relief, and connector types, which all apply when running power and control lines to the light.

For a project like S.T.A.R, these guidelines help ensure that the fixture can be used safely in a theater or event space without risk of electrical failure or fire. The NEC also includes sections specific to stage and entertainment lighting installations, which outline safe practices for temporary power setups, rigging, and portable equipment. Following these requirements not only keeps the system compliant with building and venue codes, but also ensures that anyone handling or maintaining the light can do so safely. In short, NFPA 70 provides the foundation for how S.T.A.R. should be wired and powered, ensuring the system is reliable, safe, and ready for professional use.

Lighting Fixture Safety — IEC 60598

The IEC 60598 luminaire safety standard connects directly to the S.T.A.R. project because it covers all the basic safety requirements that professional lighting fixtures need to follow. Since S.T.A.R. is designed to operate like a stage light with a high-power LED source, reflector, and motorized gimbal, it has to meet the same expectations for electrical and mechanical safety. This includes making sure the housing is properly insulated, that any exposed metal parts are grounded, and that the enclosure can handle heat from the LED without becoming a fire hazard. The parts of the standard that deal with mechanical construction also relate to S.T.A.R.'s gimbal system, since the fixture will likely be placed overhead during setup.

IEC 60598 also talks about things like thermal management, photobiological safety, and protection from dust or moisture. Those sections are especially important if S.T.A.R. is ever used outdoors or in touring environments. Following these guidelines helps ensure that the light is safe to operate around people and meets the same professional standards used by manufacturers like ETC. Overall, this standard gives S.T.A.R. a clear reference for what needs to be considered when turning a prototype into a reliable, real-world lighting product.

Optical Safety — IEC 62471

IEC 62471, titled “Photobiological Safety of Lamps and Lamp Systems,” is the international standard that defines how to evaluate the optical hazards produced by light sources. It focuses on the potential biological effects of radiation in the visible, ultraviolet (UV), and infrared (IR) ranges on both the eyes and skin. The standard applies to LEDs, lasers, and all other types of lamps, describing how to measure their spectral output and determine safe exposure limits.

This standard directly applies to S.T.A.R. because it uses a high-intensity LED source and optical system that focuses light into a controlled beam. Since the fixture is designed to track and follow performers, its light output will be directed towards people on stage, making eye and skin exposure an important consideration. Testing under IEC 62471 would determine the risk group classification for S.T.A.R.’s LED engine and help verify that it can be safely operated in close proximity to performers.

Design features such as lens selection, beam shaping, and diffuser coatings can all influence optical safety. By referencing IEC 62471, the S.T.A.R. team can make informed decisions about limiting hazardous brightness levels, minimizing blue-light exposure, and including appropriate warning labels if needed. In short, this standard ensures that S.T.A.R. delivers professional lighting performance without posing photobiological risks, aligning it with the same safety expectations as commercial LED stage fixtures.

Chapter 5 - Comparison of ChatGPT with other Similar Platforms

5.1 Introduction

Artificial intelligence (AI) has become a common tool in engineering education and professional work. It is used to help with tasks such as research, data organization, documentation, and design development. In UCF’s Senior Design course, AI tools like ChatGPT are used by students to generate project ideas, summarize technical topics, and improve report structure. These tools are also helpful for formatting documents, creating tables and figures, and simplifying large amounts of information. The main advantage of using AI in this course is efficiency; it allows students to focus more on engineering and design while spending less time on repetitive writing tasks. However, students must still verify results and apply their own judgment, as AI systems can make mistakes or present information that lacks context. When used correctly, AI supports learning by helping students communicate complex ideas clearly and stay organized throughout their project.

5.2 Overview of AI Language Platforms

Table 12 – Comparative Analysis of AI Assistant Platforms

Platform	Strengths	Limitations	Best Use
ChatGPT	Excellent contextual reasoning and report writing	Occasional factual inaccuracies	Technical documentation, brainstorming
Claude	Handles long documents and maintains tone	Limited technical depth	Summarizing long reports
Gemini	Real-time web access and multimedia use	Inconsistent with advanced topics, making confident decisions	Fact-checking and research
Copilot	integration for coding platforms like MATLAB	Not conversational or analytical	Software and coding projects

5.3 Use in Senior Design

5.3 Use in Illumination System Design

AI was a helpful tool during the design of the illumination system, mainly as a research and organizational assistant rather than a decision-maker. While it couldn't directly calculate optical parameters like focal length, F-number, or beam divergence given my specific components, it was useful for gathering background information and helping guide design choices. For example, AI was used to locate reliable data sheets for the light source, find specific optical materials, and summarize manufacturing standards related to LEDs and lenses. It was able to explain specific parameters of my design choices, like how a "cold mirror" was different from a dichroic mirror. Overall, it's good at information, but bad at optical calculations.

AI is absolutely useless when asking for help with Zemax Opticstudio. It cannot help with solving issues, how to complete tasks within the program, or explaining anything within the software. When I prompt ChatGPT to help me with something simple like where to find the fan plot, it tells me to click on menus and buttons that do not (nor ever have) existed. This is okay though, since Zemax is an old software that hasn't developed much over the past few decades and there is an abundance of information online about how to troubleshoot every possible problem that you can run into. In general, AI is pretty bad at helping with optics at all. It cannot plot any useful information (transmission spectrums, ray aberration, spot diagram…), its predictions on how light will behave and its ability to ray trace fall flat almost every time.

5.4 Use In Imaging System Design

Please refer to Appendix E [5a][5b] and [6a][6b] for the given prompt and outcomes of ChatGPT in research for the design of the imaging system.

ChatGPT [5a][5b] did an amazing job in dictating initial research on parameters set for S.T.A.R.. Aiding in explaining the concept of magnification, and verifying whether the target stage size could realistically be imaged, the LLM provided formulas which appear to be accurate when figuring out a target magnification. Indicating the need of an image and an object size, the formula the LLM provided served as the main subset of the magnification parameter we sought for. ChatGPT [6a][6b] also provided a great explanation to further the concept of demagnification. Indicating that a “short effective focal length” must be used to achieve demagnification on a 25ft stage, the goal of the commercial off the shelf lenses was made. The overall aim of the lenses we selected was to reduce the effective focal length and properly image the stage. Along with this, the LLM mentions that a small magnification occurs when the object to be imaged is significantly further than the image distance from the lens, which is an accurate explanation to match our design as we are capturing a stage 25ft away.

5.5 Use In Hardware Design

During early design phases, LLM was used to generate and refine descriptions for hardware subsystems such as the gimbal, optical housing, and motor control interfaces. It assisted in translating complex design details into formal technical language appropriate for an engineering report. Additionally, it helped standardize formatting, improve clarity, and ensure coherence between sections such as system objectives, specifications, and comparative component analyses. This allowed us to focus more on implementation and testing, while maintaining professional-quality documentation.

From a technical perspective, ChatGPT has been instrumental in assisting with system analysis and integration design. By leveraging the model’s ability to synthesize large amounts of technical information quickly, the design process became more efficient and better informed. Furthermore, ChatGPT provided valuable insight into design standards, hardware interfacing protocols, and control methodologies, reducing the time typically required for manual literature review.

For some examples, please refer to Appendix E [7a][7b], [8a][8b], and [9a][9b] for the given prompt and outcomes of ChatGPT in research for our hardware design.

5.6 Use In Software Design

LLM has been used as a productivity aid across the S.T.A.R. software stack—mobile app, Raspberry Pi backend, and ESP32 interface—without replacing engineering judgment. It helped us quickly draft architecture options (Flutter or React Native UI plus FastAPI on the Pi plus UART to ESP32), compare transports (REST vs WebSocket vs MQTT)

against our latency goals, and turn those choices into consistent, report-ready prose that aligns with Chapter 3.

On the implementation side, we used the LLM to scaffold boilerplate code and configs (for example, FastAPI project layout, WebSocket endpoints, simple health and metrics routes, and UART read and write loops) and to generate lightweight test stubs (mock frames, timing logs, and packet parsers) that we then completed and validated. For the mobile app, it accelerated early UI wireframes and happy-path flows (start, stop, manual override, target select), which we refined during integration.

For computer vision, we used the LLM to outline OpenCV pipeline options (capture to preprocess to detect or track to pan or tilt command) and to enumerate trade-offs (frame rate vs model size vs latency). Final algorithm choices, thresholds, and model selection were validated through our own experiments; the LLM served to organize options and surface issues such as bounding-box smoothing to avoid servo jitter.

It also assisted with documentation assets: turning interface definitions into clear API tables (command schema for Pi to and from ESP32 over UART), drafting sequence and state-transition descriptions for figures in Chapter 7, and standardizing terminology across chapters so the app, backend, and firmware sections stay consistent.

Limitations: we did not rely on the LLM for final algorithm design, performance claims, or optics and vision truths without testing. All code and specifications generated were treated as drafts and verified through experiments, profiling, and hardware tests.

Chapter 6 - Optical Design

6.1 Illumination System Optical Design

In the S.T.A.R. illumination system, the optical performance depends on several key parameters: the numerical aperture (NA), the focal ratio (f/#), the focal length (f), and the entrance pupil diameter (EPD). These parameters define how light from the LED COB module is collected, focused, and projected onto the stage. The following calculations outline how each value is determined and how they relate to the design of the optical system.

Numerical Aperture

The numerical aperture defines the range of angles over which the optical system can accept or emit light. It directly determines the cone of light that passes through the optical path. The equation for numerical aperture is:

$$NA = n \bullet sin(\theta/2) \tag{12}$$

where n is the refractive index of the medium (air = 1.0) and θ is the full cone angle of the beam. The reflector used in this system has a full cone angle of 36 degrees, so the half-angle is 18 degrees:

$$NA = n \bullet sin(18/2) = 0.31 \quad (13)$$

This value represents a relatively wide cone of light, suitable for a high-output stage fixture where coverage and brightness are prioritized over tight imaging resolution. The chosen NA ensures that the beam can effectively fill the aperture of the focusing lens while maintaining uniform illumination at the gate.

F-Number (f/#)

The f-number, also known as the focal ratio, describes how "fast" or "slow" an optical system is. It represents the relationship between the focal length and the entrance pupil diameter. For optical design, the f/# is linked to numerical aperture through:

$$NA = \frac{1}{2f/\#} \quad (14)$$

Rearranging for f/#:

$$f/\# = \frac{1}{2NA} \quad (15)$$

Since we have just calculated our numerical aperture value, we can plug it back into the equation to get

$$f/\# = \frac{1}{2(.31)} = 1.618 \quad (16)$$

An f/# of 1.6 indicates a relatively "fast" optical system, meaning it can collect and project a large amount of light. For illumination purposes, this is ideal because it increases light throughput, which is directly related to the brightness of the beam projected on the stage.

Focal Length

The focal length is the distance from the optical element to the point where light rays converge into focus. For the S.T.A.R. system, the focusing lens creates a 19-degree output beam, matching the design of commercial stage spotlights. The focal length can be estimated using the formula:

$$f = \frac{r}{tan(\theta/2)} \quad (17)$$

where r is the radius of the aperture (25 mm) and θ is the output beam angle (19°). Substituting the known values:

$$f = \frac{25mm}{tan(19/2)} = 167mm \quad (18)$$

If you refer to figure 4, this distance (167mm) is very close to the distance from the aperture to the lens. This is on purpose; it's just that the light is *coming from* a focal point, and not *going to* a focal point. Light can go either way through a lens, it just depends on how your system is set up. The way we have it, putting the lens exactly 167mm away from the aperture would give us a perfectly collimated output. Since we don't want that, and instead want an output cone of 19°, we move the lens 10mm further away. The 10mm was determined via Zemax simulation, since there is no single equation (or set of equations) that could determine that exact distance.

Entrance Pupil Diameter (EPD)

The entrance pupil diameter defines the physical size of the optical aperture required to achieve a given focal ratio and focal length. It is important to us because it determines our lens diameter. Usually, the lens diameter is not equal to the EPD, but the simplicity of this particular subsystem allows us to make this decision. It can be determined from the equation:

$$EPD = \frac{f}{f/\#} \quad (19)$$

Substituting the calculated values:

$$EPD = \frac{167mm}{1.618} = 103mm \quad (20)$$

This means the lens assembly must have a minimum clear aperture of 103 mm (in our case, the diameter is equal to the clear aperture). This value also provides the mechanical constraint for the lens mount and determines the outer diameter of the optical tube. Choosing a lens of this size ensures the beam is not vignetted or cut off at the edges, preserving brightness uniformity.

Illumination Source

The ETC Source 4WRD is used as the LED illumination module for this project. Since it replaces a traditional tungsten Source Four lamp, understanding its electrical behavior helps ensure safe operation and proper power planning. The 4WRD unit operates from a standard AC power source and is designed to run on a switched, non-dimmable circuit.

The unit consumes roughly 150 watts at full output. For a 120-volt supply, this results in a steady current of approximately 1.25 amps, using the basic electrical equation:

$$I = \frac{P}{V} \quad (21)$$

Substituting values:

$$I = \frac{150W}{120V} = .63A \quad (22)$$

This is a fairly light load compared to traditional stage lighting fixtures, making the Source 4WRD more power-efficient and easier to power from standard outlets.

For heat calculations, LED fixtures still generate heat even though they are more efficient than tungsten lamps. The Source 4WRD produces roughly 450–500 BTU per hour [29]. This helps estimate ventilation needs and ensures that adjacent components, especially 3D-printed parts or sensitive electronics, remain below their temperature limits. Although this heat level is lower than an incandescent Source Four, it still needs active or passive airflow around the light engine to avoid heat buildup.

Table 13: Power and Thermal Specifications for the ETC Source 4WRD Light Engine

Parameter	Value
Input Voltage	100–240 VAC
Power	~150 W
Current @ 120V	~1.25 A
Inrush Current	~32 A
Heat Output	~450–500 BTU/hr
Circuit Type	Switched / Non-Dim

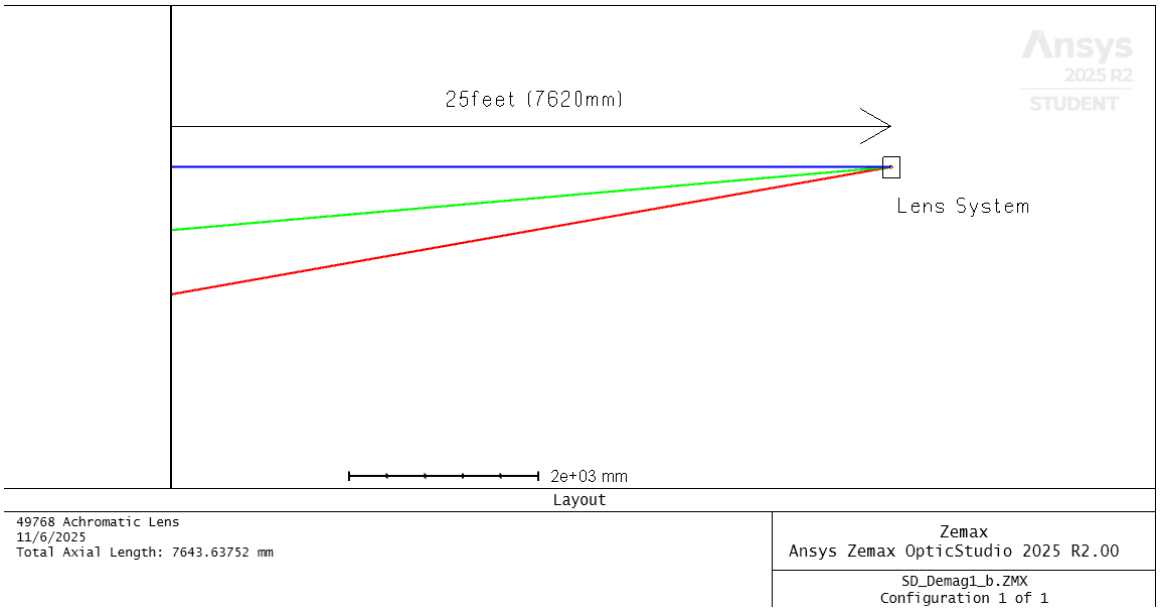
6.2 Imaging System Optical Design

Ansys Zemax Optic Studio Simulation

With the selected lenses to build the imaging system, there are several variables to account for to ensure that the target parameters are met. The main criteria to be met is the magnification of the overall system, aiming to hit a 1250x demagnification for a 25feet wide stage placed 25 feet away from the optical system. Much of the target values we are seeking for such as effective focal length and f/# can all be simulated using Ansys' Zemax Optic Studio, a program for lens design. Referring back to the lenses and their parameters in Table 1.0, there are a few variables that can be controlled in order to optimize the output image produced by the system. Since we are using commercially off the shelf lenses as opposed to custom lenses, the only degree of freedom to vary is the

distances between each lens, and the distance from the sensor. The simulation design and results are shown below in Figure 18.1, Figure 18.2, and Figure 19.

Figure 18.1 Zemax Simulation of Selected COTS Lenses



Please note that the lens system begins inside the box. The magnified simulation of the lens system is shown in Figure 18.2.

Figure 18.2 Zemax Simulation of Selected COTS Lenses

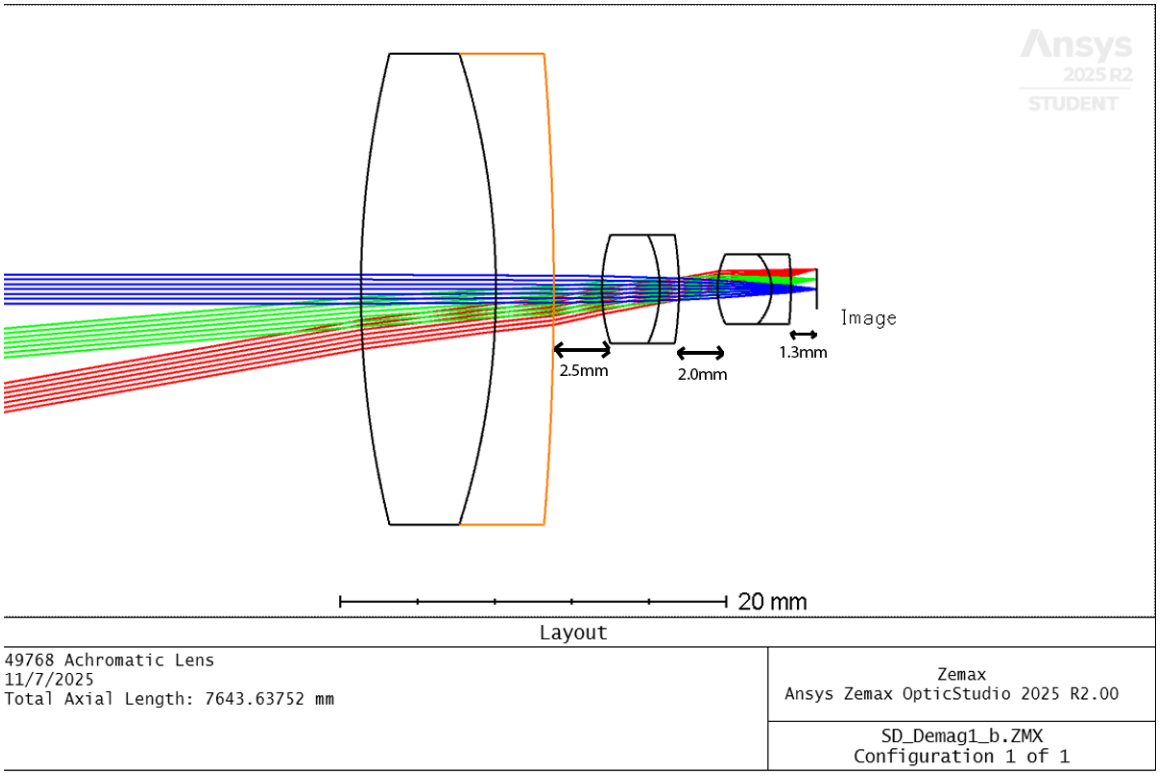


Figure 19. Zemax Simulated Extended Scene Analysis

It is important to note that the aperture stop is placed after the second achromatic doublet, with a diameter of 1.5mm, which is what causes the rays in the figure to not reach the full diameter of the first lens.

Through this simulation, and optimizing the system, it was found that the distance between the last surface of the system and the sensor that produces the smallest RMS

spot size was at a distance of 1.338mm. It is pivotal to acknowledge that this distance does not equate to the effective focal length of the system, but rather the back focal length. For the effective focal length, that can be calculated with the knowledge of both image and object planes. For a multi-lens system, the image of the first lens becomes the object of the second lens, and the same concept applies for each lens. Using Zemax, this focal length is already calculated, with an optimized distance between the achromatic collector lens and the achromatic corrector lens to be 2.5mm, and the distance between the corrector lens and the imaging lens to be 2.0mm.

With these distances, and the focal lengths of each lens, the effective focal length of the lens system is 5.904 as calculated by Zemax.

Table 14: Reported Calculated Values of the Imaging System from Zemax

Parameter	Reported Value
Entrance Pupil Diameter (EPD)	1.5mm
Effective Focal Length (EFL)	5.9mm

With these values, various numerical parameters can be calculated in order to verify that the target parameters have been met.

Magnification Calculations

From the object height of 25 feet or 7620mm, and a sensor size of 6.29mm horizontally, the theoretical target magnification can be calculated using

$$M = \frac{h_i}{h_o} = \frac{6.29}{7620} = \frac{1}{1250} \tag{23}$$

This result means a 1250x magnification system is needed to properly image the width of the stage. With the results of the Zemax report, we can calculate the true magnification the imaging system achieves, found by dividing the effective focal length over the distance of the stage to the system.

$$M = \frac{h_i}{h_o} = \frac{5.9}{7620} = \frac{1}{1290} \tag{24}$$

Although not precisely a 1250x demagnification, the system achieves a 1290x demagnification, which is about 3% higher with a slightly smaller image than the theoretical image, suitable for our design.

F/#

The working f/# of the system can also be determined from the Zemax report, using both the given effective focal length and the entrance pupil diameter. The calculated f/# can be found by dividing the effective focal length over the entrance pupil diameter. As aforementioned, the entrance pupil diameter was previously defined in the system from the aperture stop placed between the corrector and the imaging lens, pre-defined as 1.5mm. Thus, the working f/# of the system is:

$$f/\# = \frac{EFL}{EPD} = \frac{5.9}{1.5} = f/3.9 \quad (25)$$

The calculated value of f/3.9 is a perfectly reasonable speed that is acceptable for a compact imaging lens assembly, providing expected depth of field for an imaging system.

Chapter 7 - Hardware Design

7.1 Single Board Computer Design

The CM5 has 28 general purpose I/O (GPIO) pins which correspond to the GPIO pins on the 40way header. Since the CM5 can support up to five UART connections, we will be using one of those connections to transmit data to the ESP32. In figure 20, the Raspberry Pi Compute Module 5 transmits data to the ESP32-S3 using GPIO14 (Function a4, UART0_TX). The ESP32 receives that data using pin 36 - RXD0 (U0RXD).

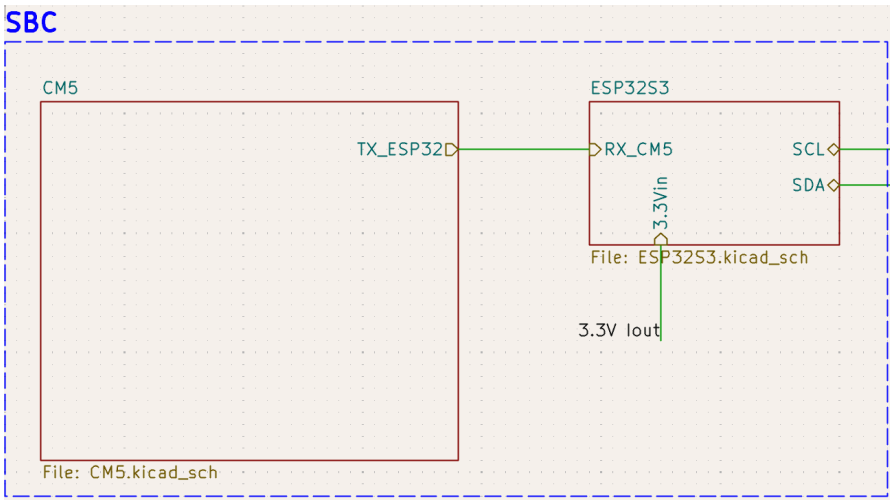


Figure 20: High-level SBC diagram

The system on module (CM5) and microcontroller (ESP32) are what make up our single board computer. The Compute Module 5 handles all of the computer-vision tasks then transmits that information to the ESP32 where it handles all of the motor control for gimbal movement and other motor functions. We will now look into the schematics and design of our CM5 carrier board and ESP32 respectively.

7.1.1 CM5IO Carrier Board

A system on module is a compact module containing the CPU, RAM, and storage. However, It needs to be connected to a "carrier board" to function properly. Raspberry Pi does offer a readily available carrier board. We used this for testing and our final product. The Raspberry Pi CM5IO carrier board is shown in the figures below.

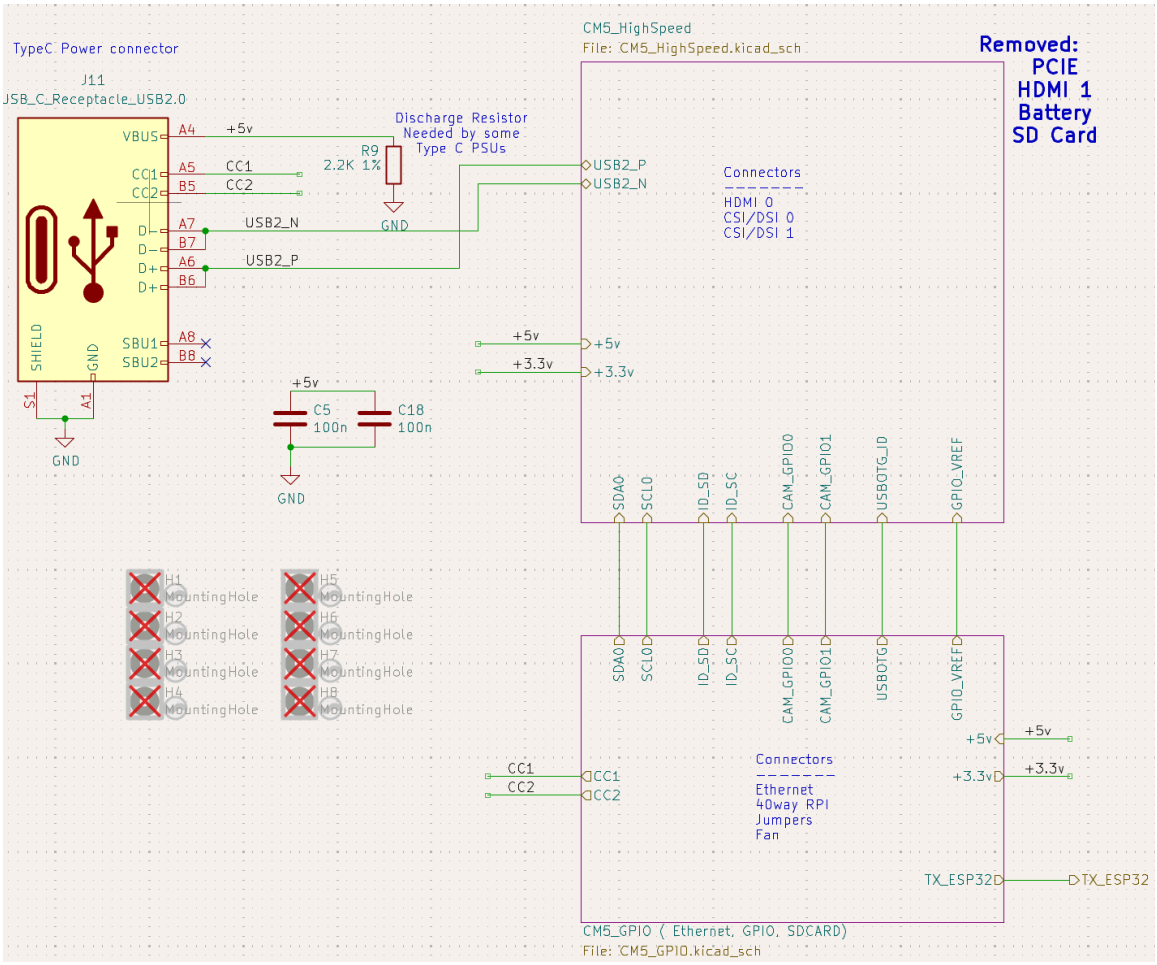


Figure 21: High-level Carrier Board Design

In figure 21, the schematic is split into three main sections. On the top left, a USB-C receptacle is used to provide power delivery (+5V), transmit/receive data (D-/D+), and send configuration channel signals (CC1, CC2) to negotiate power delivery. On the top right, the highspeed connections (HDMI, Camera, USB) and are contained. Finally, the bottom right holds general purpose I/O pins and connections such as fan headers, ethernet port, and jumper pins. Figure 22 shows the entire highspeed schematic sheet. Figure 23 shows the entire GPIO schematic sheet.

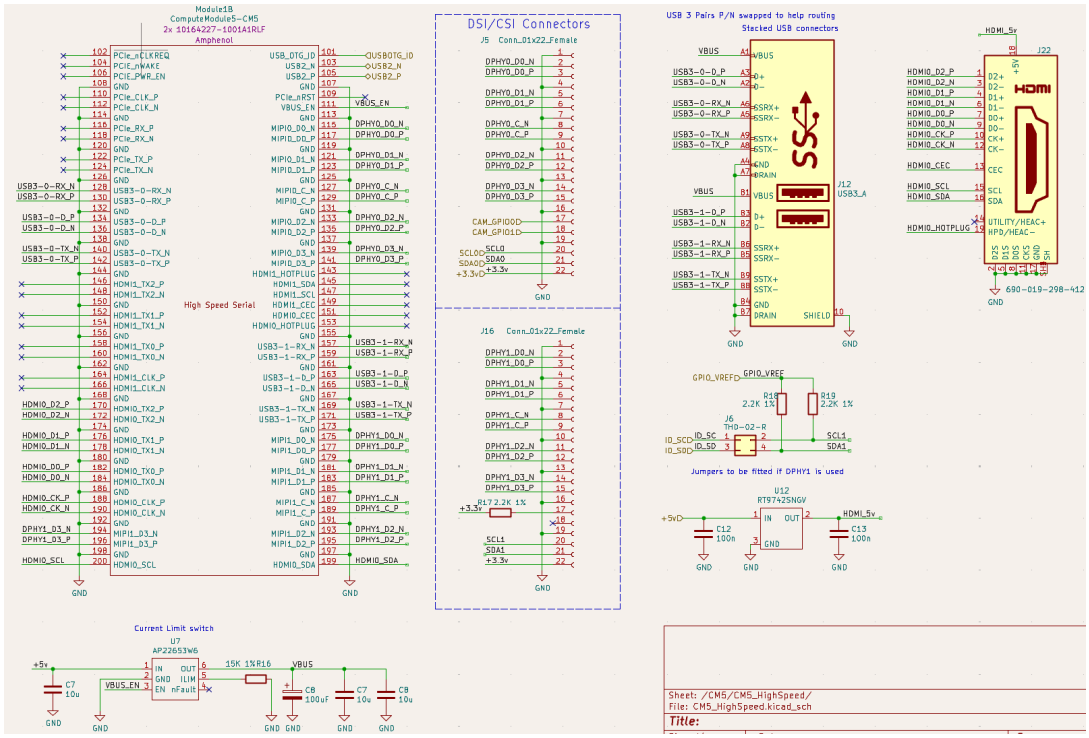


Figure 22: Highspeed Carrier Board Schematic

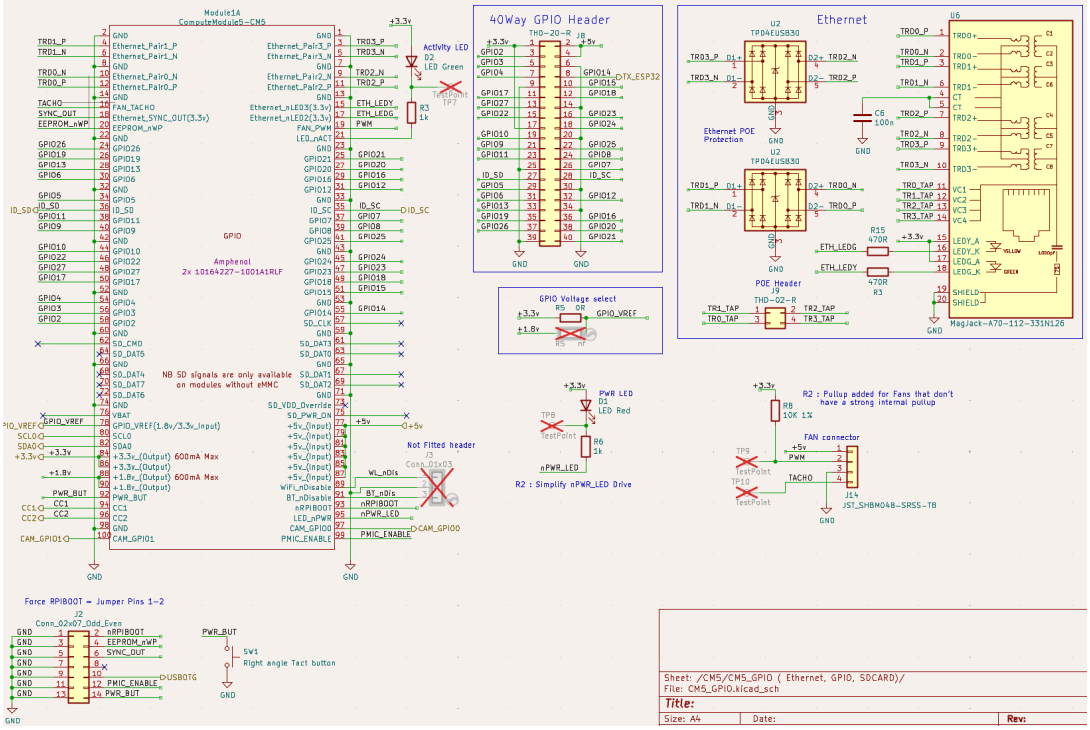


Figure 23: GPIO Carrier Board Schematic

7.1.2 ESP32

The ESP32 microcontroller serves as the primary motor control unit within the S.T.A.R. system, responsible for driving the 2-axis servo gimbal that orients the spotlight. Real-time positional commands are received from the Raspberry Pi Compute Module through a UART serial interface, ensuring low-latency and deterministic response for spotlight adjustments. Figure 24 shows the entire MCU schematic sheet.

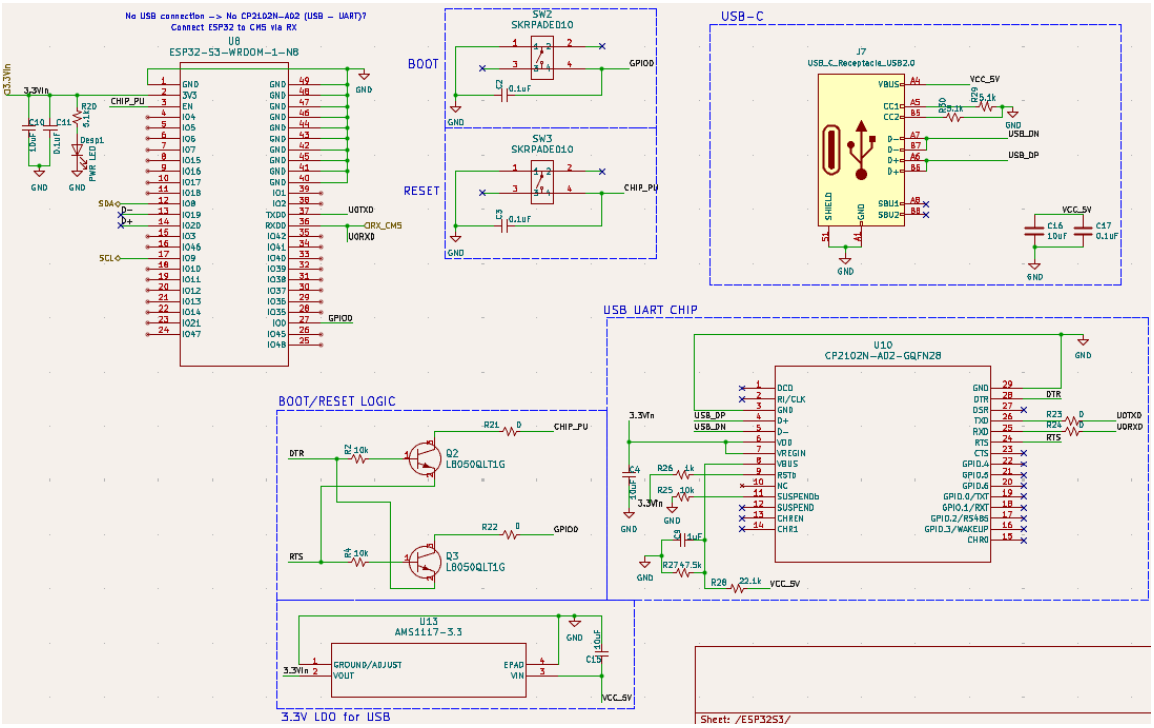


Figure 24: ESP32 Schematic

The ESP32 and its components will be on its own custom PCB. 3.3V Power will be supplied by the 600W PSU through a buck converter discussed later in this chapter. While the ESP32 will receive real-time commands from the Raspberry Pi, a USB-C USB port has been added for programming redundancy.

7.2 Motor Control

The Adafruit PCA9685 PWM driver is utilized as our dedicated servo control interface, providing precise and hardware-timed pulse generation for the servo motors. Operating via an I2C communication bus, the PCA9685 offloads the real-time PWM computation from the ESP32, enabling smoother and more stable servo motion without burdening the microcontroller's internal timers. The use of the PCA9685 ensures deterministic timing, reduced signal noise, and scalable control, all critical for achieving the fluid and responsive spotlight tracking performance required by the system. For this semester, a standalone PCA9685 board will be used. We will integrate the PCA9685 onto a custom controller board. Figure 25 shows the entire PCA9685 schematic sheet.

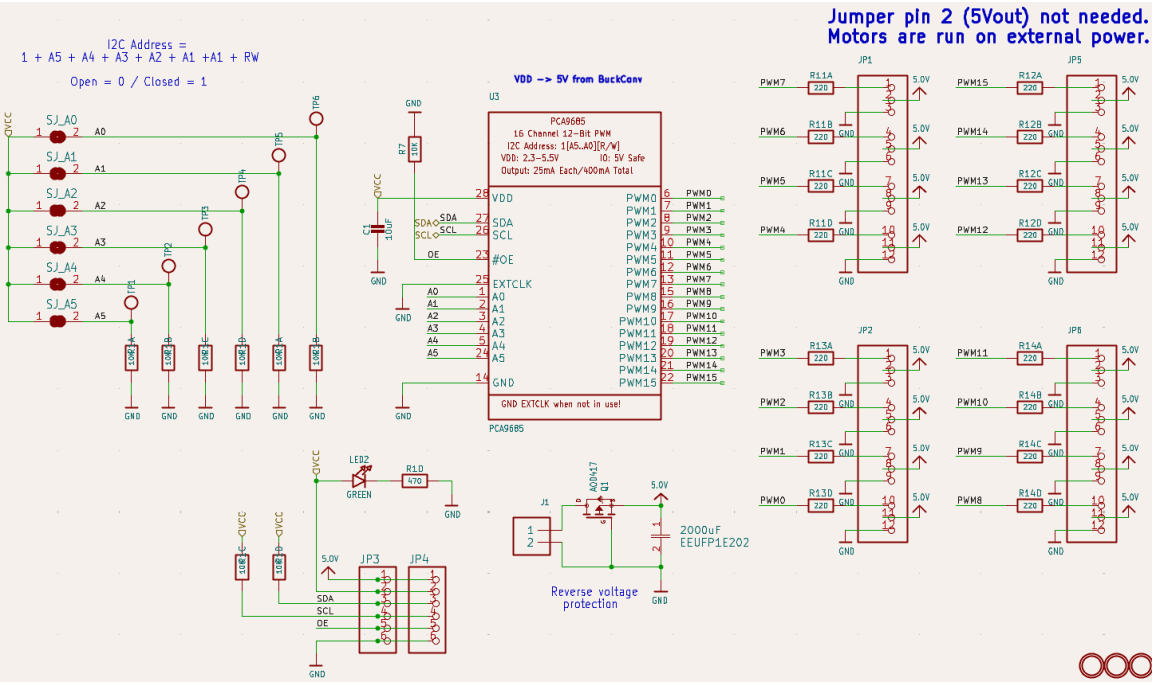


Figure 25: PCA9685 Schematic

The PWM controller needs a 5V supply. This will be provided by the 600W PSU. Since the PCA9685 has an output voltage limit of 6V, we must power the motors with an external source. This will be covered later in the chapter where we discuss the buck converters used for voltage regulation.

7.3 Power Delivery/ Electrical Power System

7.3.1 24V 25A 600W PSU

To provide ample power to our system, we needed a power supply unit with a high current output. Since there are multiple high torque servo motors being used, we needed enough amperage in case all of the motors reach their stall current at the same time. The PSU plugs into a typical 110V/220V AC power outlet. Output voltage is adjustable from 0-24V DC. Appropriately rated wire will be attached to the screw terminals, then connected to the WAGO connector on the PCB. The PSU will be bought off amazon and not custom built in the interest of time and cost savings.

7.3.2 WAGO Connector

We are using the WAGO 2604-1102 PCB terminal block. It provides us with more than enough rated current and voltage for our application. It also comes with the added benefit of convenience with its push in cage clip lever system. Figure 26 shows the schematic for the WAGO connector.

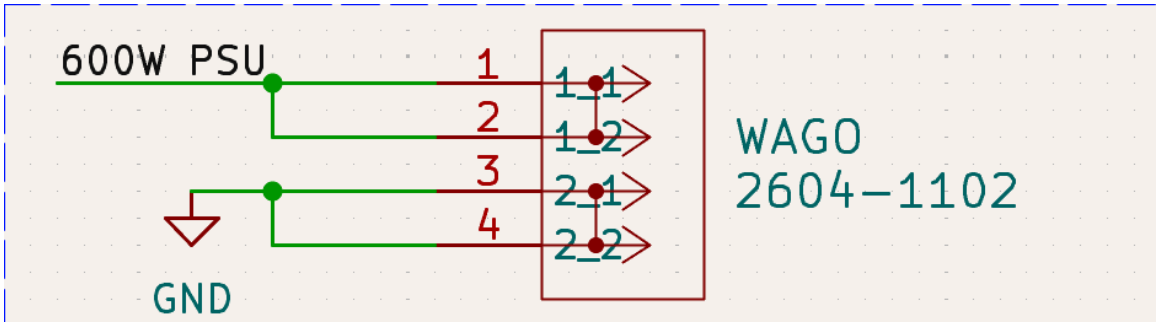


Figure 26: WAGO Schematic

7.3.3 Buck Converters

A buck converter is a type of switching voltage regulator designed to efficiently reduce a higher input voltage to a lower output voltage. Buck converters are commonly used to power low-voltage components such as microcontrollers, sensors, and communication modules from higher-voltage sources (e.g., 12V or 9V supplies) and provide relatively high levels of efficiency (85% - 95%). For our hardware design, we have used Texas Instrument components for all of our buck converters. It was extremely helpful being able to use the TI WEBENCH power designer to come up with the buck converter circuitry.

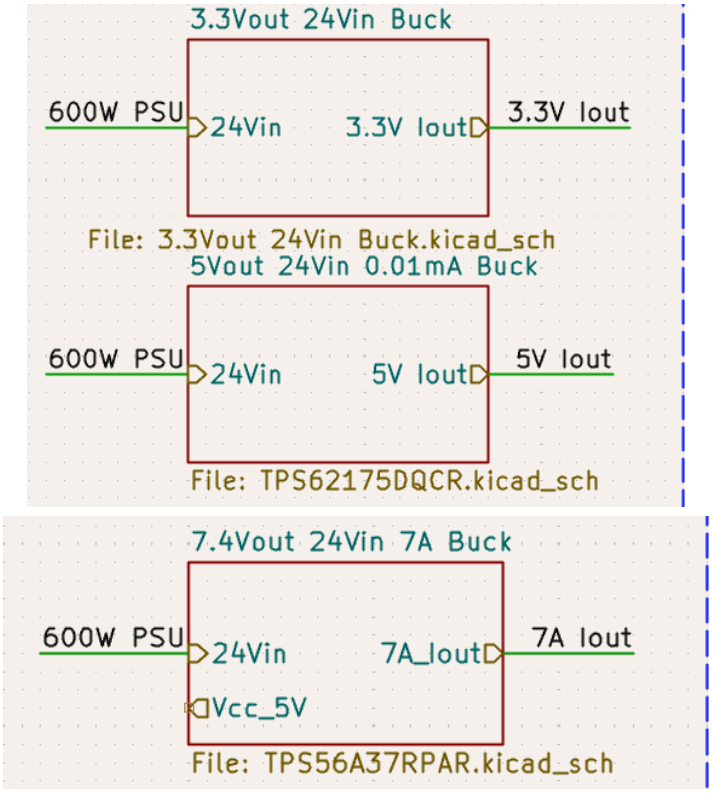


Figure 27: Buck Converter Hardware Blocks

Above in figure 27, we have the high-level blocks of our buck converters. All of them receive voltage from our 600W PSU and step it down to the desired voltage and current output. We will go into detail for each of them in the figures below.

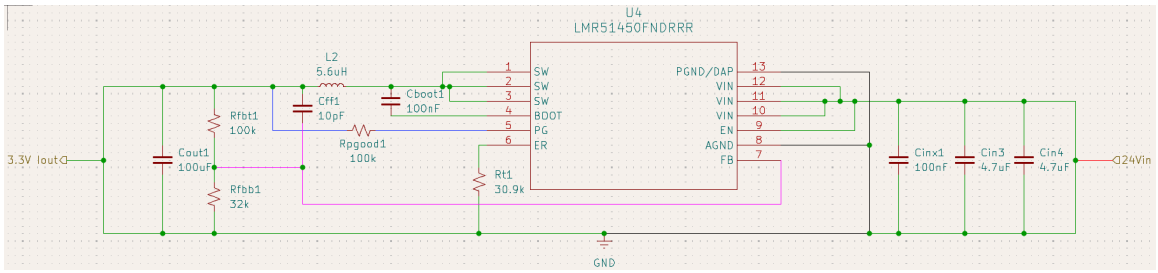


Figure 28: LMR51450 Buck Converter

In figure 28, we use a TI LMR51450 to take our PSU's 24V 25A down to 3.3Vout at 2A Iout. This buck converter will be used to regulate voltage to our ESP32. In figure 29, we use a TI TPS62175 to step down the PSU to 5Vout at 10mA Iout. This one is used to regulate power to the PCA9685.

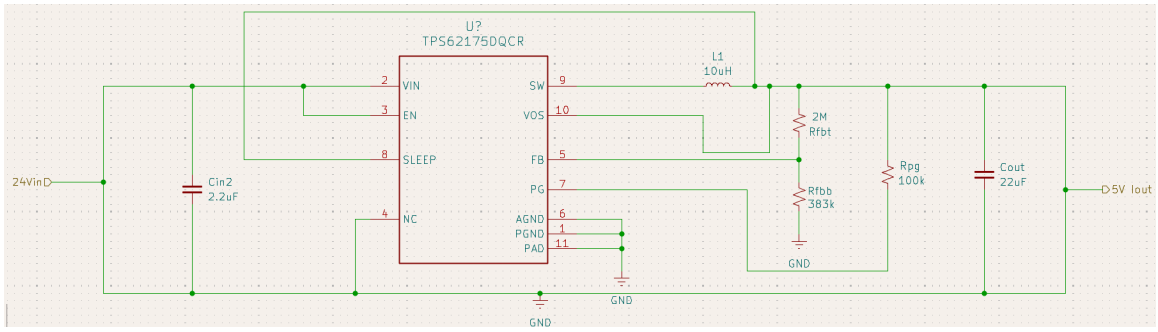


Figure 29: TPS62175 Buck Converter

For our motors, we will use similar TI components. Below in figure 30, we use a TI TPS56A37 to take our PSU's 24V 25A down to 7.4Vout at 7A Iout. This buck converter will be used to regulate voltage to our 35kg motors. We use the same TI TPS56A37 to regulate power to the 25kg servo motors.

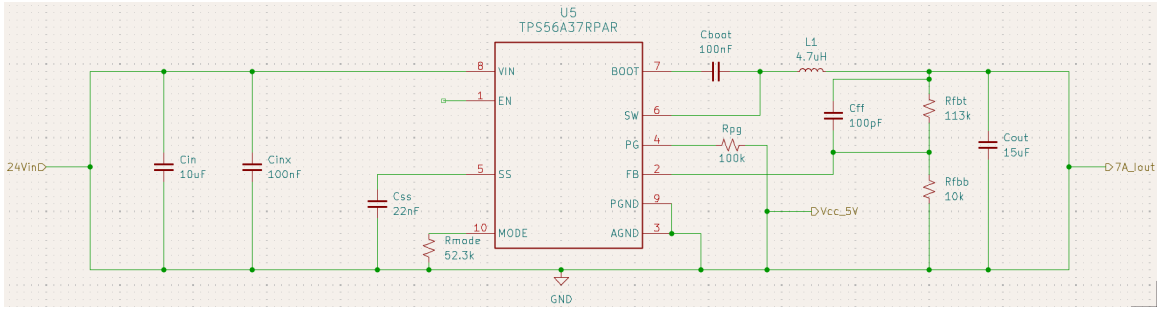


Figure 30: TPS56A37 Buck Converter

7.3.4 CM5 Power

The Raspberry Pi Compute Module 5 (CM5) should be powered by a dedicated power source. It has been reported that the CM5's sensitive electronics can become unstable when the module is configured to both supply and receive power simultaneously. To prevent potential backfeeding and electrical noise, it is considered best practice to isolate the CM5's power supply from other system components. In this design, the CM5 is not powered by the 600W PSU used for other subsystems; instead, it is powered by a 27W USB Type-C power adapter provided by Raspberry Pi. The schematic shown below illustrates the power protection circuitry implemented for the CM5 to ensure stable and reliable operation.

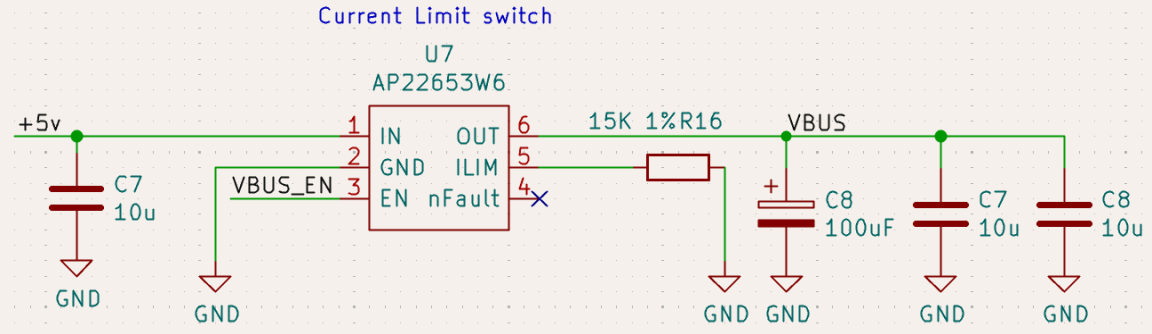


Figure 32: CM5 USB-C Current Limit Switch

7.4 Terminal Board

To reach the PCB requirements, we will be adding a terminal board to hold additional peripherals such as status LEDs for each buck converter and power / kill switches. Voltage and current sensing components at the output of each buck converter that displays on the terminal board may be an addition in SD2 if it deemed to be a significant QOL improvement.

7.5 Gimbal Design

The proposed design for a two-axis pan–tilt gimbal is intended to provide precise, stable, and responsive motion control for a Source Four 19° stage light. Three 35kg-cm servo motors will be used for the gimbal. Two servos will be used for the tilt axis, one for each side of the stage light. The remaining servos will be used for the pan axis. Due to time constraints, a smaller scale spot light will be used for Senior Design 1 Demo. In later revisions, proper servos will be used that can handle the weight of the Source Four 19° stage light.

Mechanically, the gimbal consists of two main rotational stages: a pan axis located at the base and a tilt axis integrated within the upper stage. The pan assembly use a turntable bearing to relieve some for the frictional forces. The tilt axis will use a pair of roller bearings, designed to handle the moment loads imposed by the weight of the light fixture. Both axes will be printed from PETG because it is easy to manufacturing and cheap to repproduce.

A high-ratio planetary gearbox may be implemented at each axis if deemed necessary or benificial. The gearbox provides the necessary torque multiplication while maintaining low backlash for precise angular control. A holding brake may also be included on the tilt axis to secure the light in position during power loss or system faults.

The gimbal control architecture consists of three major subsystems: motor control, vision processing, and power distribution. Motor drives interface directly with the servos, providing current, velocity, and position control loops. The drives receive angular setpoints from an ESP32 microcontroller, which executes low-latency position control and communicates with the Raspberry Pi Compute Module 5. The Compute Module handles the higher-level vision processing and target tracking, transmitting position commands to the ESP32 via a UART.

In testing and verification, the system will be evaluated for static holding torque, smoothness of motion, and positional accuracy. Dynamic performance will be measured in terms of latency, overshoot, and response to rapid direction changes, ensuring that the spotlight tracks targets seamlessly during live operation. Acoustic testing will confirm that motor and gearbox noise remain below acceptable thresholds for stage environments, while safety testing will verify reliable operation under power interruptions and limit conditions. The final design aims to combine robust mechanical construction, precise servo control, and responsive real-time tracking to create a professional-grade, automated spotlight system suitable for theater and live event applications.

7.6 System Housing Design

The electronic control system—including the Raspberry Pi Compute Module 5, ESP32 motor controller, PCA9685 PWM driver, and supporting power circuitry will be housed within a custom 3D-printed enclosure designed. The enclosure serves as both a protective shell and a thermal management solution for the sensitive electronics that power and control the S.T.A.R. system.

The enclosure is constructed from polyethylene terephthalate glycol (PETG), chosen for its balance between mechanical strength, thermal stability, and printability. PETG's higher temperature tolerance compared to standard PLA prevents deformation during prolonged system operation. The housing is printed with 3 mm wall thickness and >10% infill density, providing sufficient rigidity while maintaining a manageable weight for gimbal-mounted or portable configurations. The design incorporates threaded brass inserts at mounting points to ensure repeated assembly and disassembly without wear to the plastic threads. Cable management channels and strain relief clips are integrated into the design to ensure secure routing of power and signal lines, reducing the risk of accidental disconnection or noise coupling.

Thermal management is achieved through strategically placed ventilation slots and optional 40 mm low-noise fans positioned near the PSU, motor driver sections, and CM5

carrier board. Passive heat dissipation is enhanced by vent fins located along the enclosure's upper perimeter, promoting natural convection.

Chapter 8 – Software Design

8.1 Software Architecture

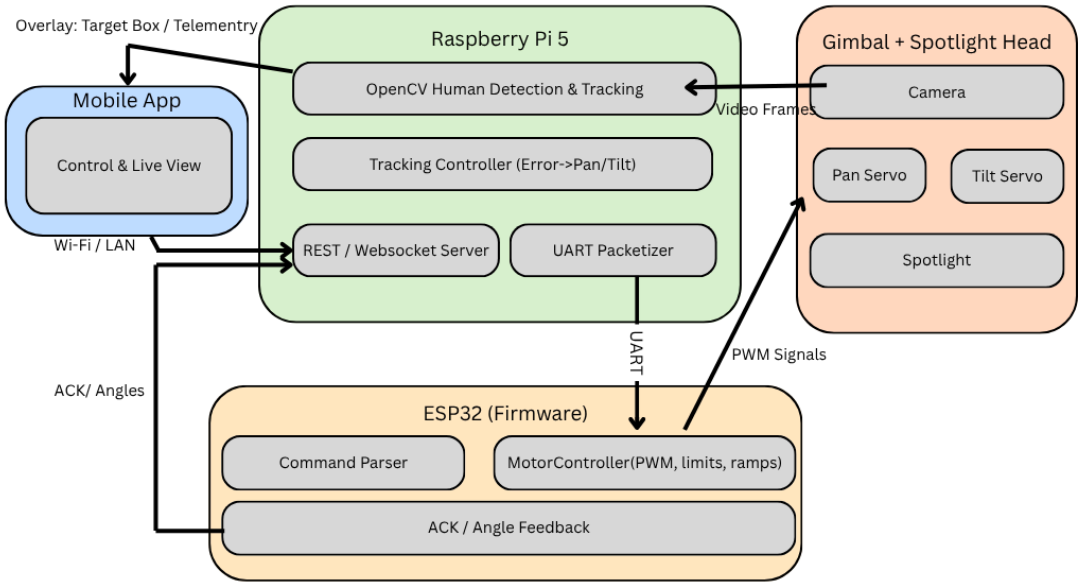


Figure 33: S.T.A.R. Software Architecture

The software is organized into three main components: a Raspberry Pi 5 for high-level vision processing and control logic, an ESP32-S3 microcontroller for low-level motor control, and a mobile application for user interface. The Raspberry Pi receives live video from a camera mounted on the gimbal and runs a computer vision algorithm (using an OpenCV-trained model) to detect and track a performer. Based on the target's position in the frame, the Pi computes the required pan/tilt adjustment and sends corresponding commands over a UART serial link to the ESP32 microcontroller. The ESP32 firmware interprets these commands to drive the pan and tilt motors of the gimbal, directing the spotlight accordingly. Meanwhile, the mobile app connects to the Pi's onboard server via Wi-Fi (either through a local network or the Pi acting as a hotspot) to provide real-time user control and monitoring. This modular architecture ensures each subsystem is specialized – the Pi handles intensive image processing, the MCU handles real-time actuation, and the app handles user interactions – while all three communicate to achieve smooth automated spotlight tracking.

The Raspberry Pi 5 serves as the central controller and “brain” of the system. It runs a full Linux OS, enabling the use of high-level libraries for computer vision (OpenCV, etc.) and networking. The Pi processes video frames from the

MP CMOS camera (mounted on the gimbal) in real time, applying a machine learning-based human detection model to identify the performer. By mounting the camera directly on the moving spotlight head, the vision system always stays aligned with the light beam. Once a target is identified, the Pi calculates the target’s offset from the center of the frame. This offset is translated into pan and tilt angles needed to re-center the spotlight on the target. A control loop on the Pi continuously performs this capture-detect-compute cycle, effectively tracking the subject as they move.

To minimize latency between detection and response, the Pi immediately communicates the needed adjustments to the ESP32-S3 MCU. The Pi and MCU communicate over a serial UART interface using a lightweight protocol (e.g. simple command strings or packets). For example, the Pi might send commands like “PAN=30” or “TILT=-15” indicating the servo positions or increments required. The design targets a total system response under ~250–300 ms, so efficient communication and processing are critical. The software design emphasizes synchronization between the imaging and motor subsystems – the Pi’s vision loop and the MCU’s motion control loop must operate in harmony. Any delay in image processing, command transmission, or motor response can introduce tracking error, so the code is optimized to reduce overhead. For instance, the communication protocol is kept simple, and the Pi might batch small commands or use binary messages to avoid latency. Acknowledgement messages from the MCU are used to confirm execution, ensuring the Pi knows when a command has been received and applied. If an acknowledgment is not received within a timeout, the Pi can retry or alert the user, adding robustness to the system.

On the ESP32 microcontroller side, the firmware is structured to handle real-time motor control. The MCU code parses incoming commands from the Pi and drives the servo motors for pan and tilt accordingly. Because the spotlight must move smoothly and precisely, the MCU generates appropriate PWM signals to achieve the commanded angles with minimal overshoot or jitter. The software incorporates calibration constants and acceleration profiles to smooth out motion – an advanced objective of the project was to calibrate motor control for minimal jitter and overshoot. The MCU may also enforce mechanical limits to prevent the gimbal from rotating beyond its range. The separation of responsibilities (vision on Pi vs. motor control on MCU) allows the system to leverage the Pi’s computational power for image analysis while benefiting from the real-time responsiveness of the MCU for actuation.

8.2 Vision Algorithm and Target Tracking

The vision algorithm is at the core of S.T.A.R.’s autonomous tracking capability. It uses computer vision techniques to detect a human subject in the camera’s field of view and continuously track that subject’s position. Initially, the system employs a pretrained object detection model (from OpenCV’s library or similar), configured to recognize

human figures. Each frame captured by the Raspberry Pi's camera is fed into this model. The output is the coordinates (e.g. bounding box) of the detected performer in the image. From this, the software determines the target's centroid position relative to the frame center. The goal is to keep the performer centered; thus, the difference between the target's position and the image center (the tracking error) is computed each frame.

To translate this image-space error into physical motion, the Pi's software applies geometric calibration that relates pixels to angular movement. This calibration accounts for the camera's field of view and the gimbal's angular resolution. For example, if the target appears 10% to the right of center, the Pi might calculate that the pan motor needs to rotate a certain number of degrees right to compensate. The Pi then sends these pan/tilt angle adjustments to the ESP32. This forms a closed-loop control: as the target moves, the camera image shifts, the Pi computes new angles, and the motors correct the spotlight's aim. The loop iteration time is kept as short as possible (tens of milliseconds) to ensure smooth tracking without lag.

Importantly, the vision software is designed to be robust against typical stage conditions. It performs frame-to-frame tracking, possibly using techniques like Kalman filtering or optical flow to predict motion in between detection frames, reducing jitter. If the detection model temporarily loses the target (e.g. due to occlusion or stage lighting effects), the software can momentarily extrapolate the target's motion or widen the search region to reacquire. In addition, the algorithm can distinguish between multiple people on stage. If multiple potential targets are detected, the system either automatically chooses one (e.g. the largest or central figure) or waits for user selection via the app. The ability to handle multiple detections and allow user choice is an advanced feature in the design. By default, the system focuses on one performer at a time to mimic a traditional spotlight operator focusing on the lead performer.

Tracking a fast-moving performer requires not just detection but also predictive motion handling. The software may incorporate a predictive filter so that the gimbal can begin moving toward the anticipated next position of the target, rather than reacting purely after the fact. This helps reduce latency in practice. Because the overall system latency includes camera exposure, image processing, command sending, and motor acceleration, any prediction in software helps keep the spotlight aligned in real-time. As noted in the design goals, achieving a smooth, low-latency spotlight movement was a primary objective.

Another aspect of the vision design is reacquisition and recovery. If the performer leaves the frame or the system is toggled to Auto mode without a known target, the software enters an idle scanning behavior. In this state, the camera can pan slowly (or the system can use a wide-angle search) to look for a human silhouette. This "patrol" mode prevents the system from doing nothing if auto-tracking is active but no target is currently locked. As soon as a person is detected, tracking will resume. Conversely, when the performer stops moving, the system stabilizes the spotlight on that position without oscillation, thanks to the damping in the control logic. Overall, the vision and tracking algorithm ensures the spotlight smoothly follows the chosen subject, keeping them illuminated center-stage without manual intervention.

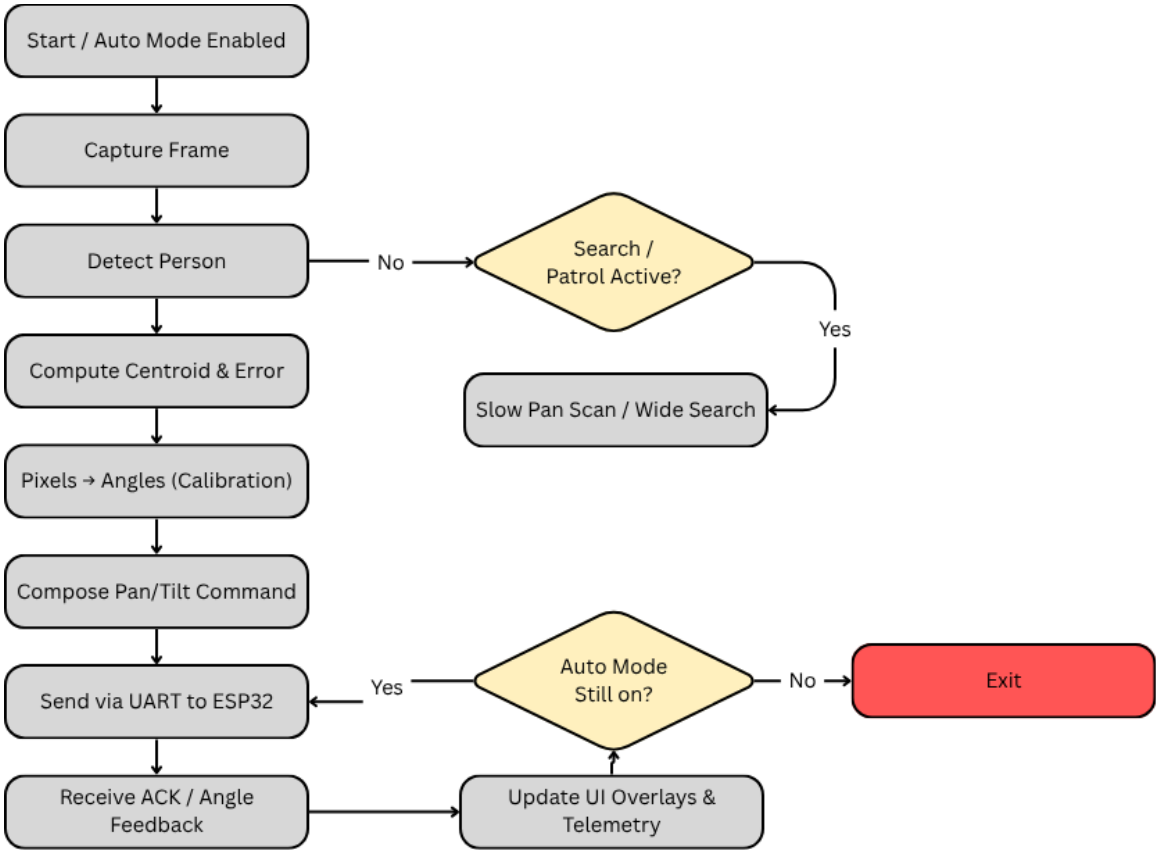


Figure 34: Continuous Tracking Loop Flowchart

The Raspberry Pi continuously executes this loop during Auto-Tracking mode. Each cycle begins with capturing a camera frame and running the detection/tracking algorithm to locate the target. If the target is found, the Pi calculates the necessary pan/tilt adjustment and sends a command to the ESP32 to move the gimbal accordingly. If no target is detected (e.g. target temporarily out of view), the system can hold the previous spotlight position or perform a slow scan until a target reappears. This loop repeats indefinitely at a high frequency, enabling real-time tracking updates. The loop terminates only when tracking is disabled (such as when the system is turned off or switched to Manual mode). The continuous nature of this loop is what allows S.T.A.R. to follow dynamic movements smoothly, with the Pi making incremental adjustments and the ESP32 moving the motors on the fly.

8.3 Microcontroller Firmware and Motor Control

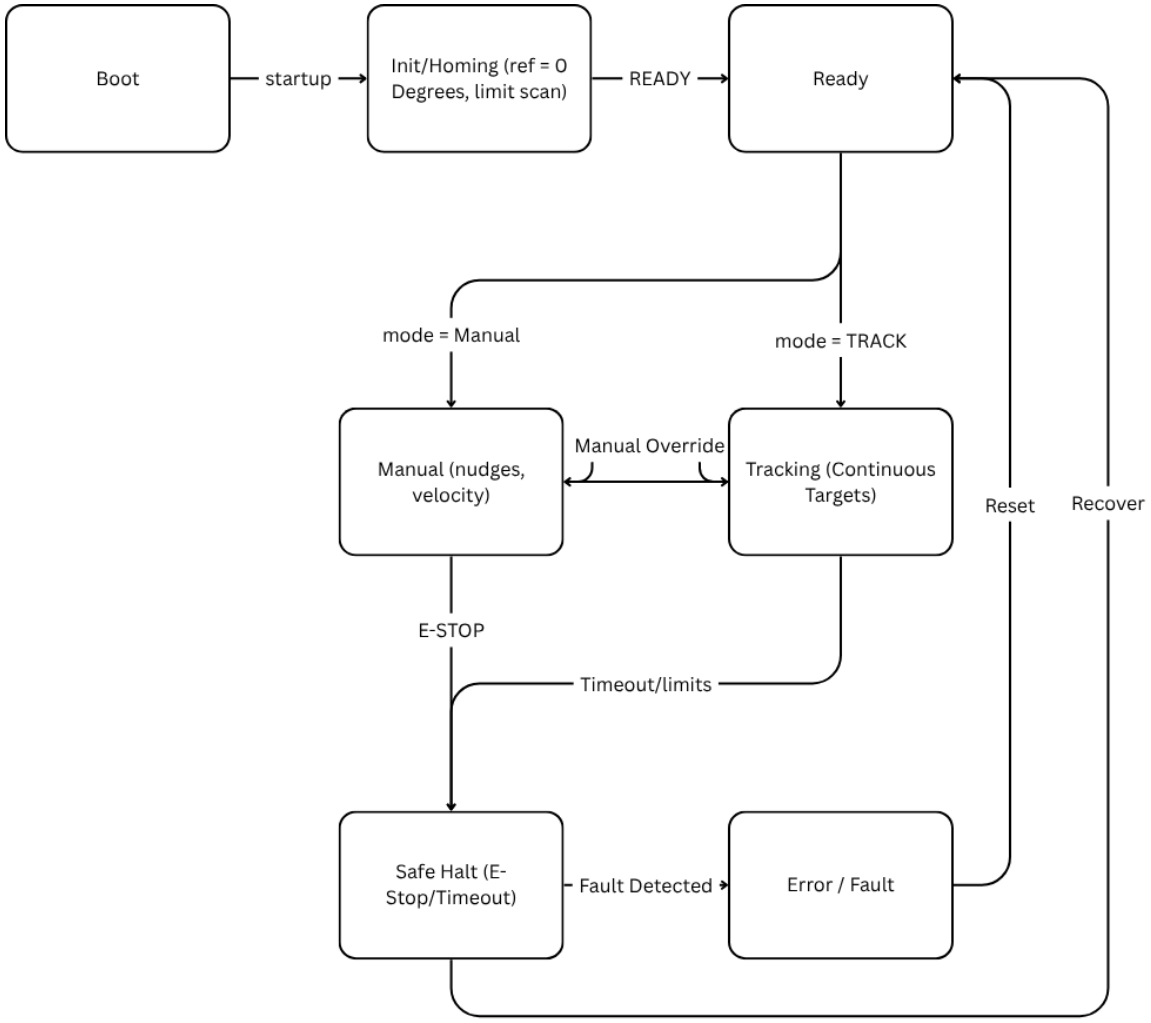


Figure 35: State Machine Flowchart

The ESP32-S3 microcontroller runs the firmware responsible for gimbal motor control and executing commands from the Pi. Its software design is lightweight and real-time oriented, often structured as an infinite loop or interrupt-driven tasks that listen for incoming commands over UART and adjust motors accordingly. On receiving a pan/tilt command, the MCU's firmware parses the message (for example, extracting an angle or movement value). It then interfaces with the motor drivers to produce the required motion. The pan and tilt axes are controlled by high-torque servo motors; these may be driven either directly via PWM signals from the ESP32's timers or through a dedicated servo controller IC, depending on the hardware design. In either case, the firmware generates the correct pulse widths corresponding to the desired angles.

Because smooth motion is critical for professional-looking spotlight operation, the MCU software likely implements motion smoothing. For instance, instead of jumping directly to a new angle, it can increment the servo position in small steps if the change is large, or

limit the speed to avoid sudden jerks. This can be achieved through a simple ramping algorithm or by utilizing the servos' speed control features. Additionally, the firmware can monitor the current position (using either servo feedback or by keeping track of the last commanded position, if the servos are open-loop) to decide how much to move on each update. If the servos or gimbal have feedback sensors (like potentiometers or encoders), the MCU would run a feedback control loop (PID controller) to accurately reach and hold the commanded position. In the initial design, standard hobby servos with built-in feedback are used, so the ESP32 mainly sends positional commands and the servo's internal controller holds the angle.

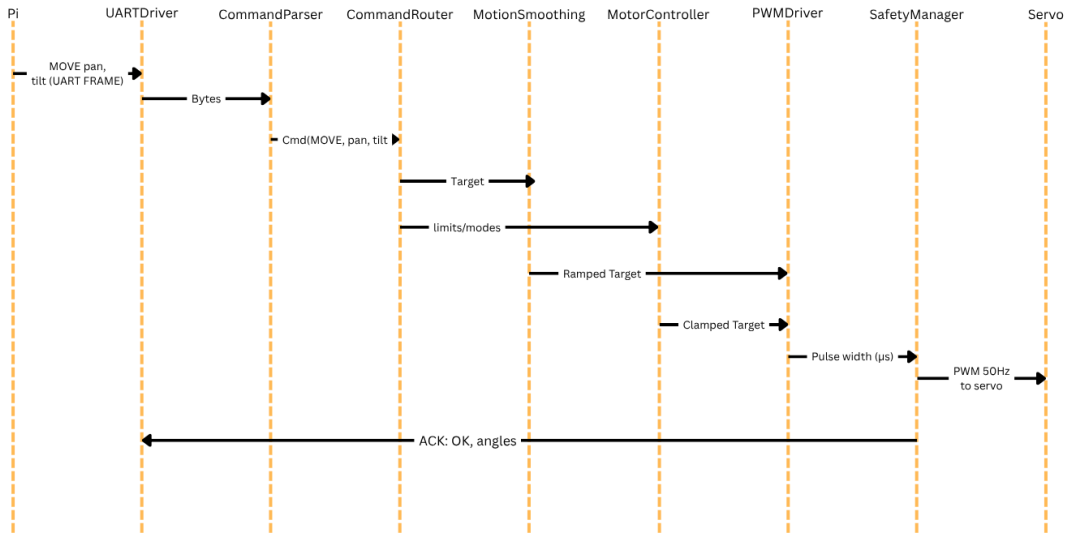


Figure 36: Move Sequence Flowchart

The communication protocol between the Pi and ESP32 is a key part of the firmware design. The team opted for a simple custom protocol over UART for reliability and ease of debugging. For example, a command frame might be a short ASCII string or binary packet containing a command identifier and parameters (angle values, etc.), terminated by a newline or special byte. The ESP32 continuously listens on the UART; when a full command is received, it is decoded and dispatched to the appropriate handler. The firmware likely has a command parser module that can recognize commands such as "SET_PAN", "SET_TILT", "MOVE" (for combined pan/tilt), "CALIBRATE", or "STOP". This modular design (as illustrated in the software class diagram) separates the parsing logic from the motor driving logic. For instance, a CommandParser component processes incoming messages and then calls a MotorController component with the parsed parameters, reflecting a clean separation of concerns in the firmware architecture.

To support two-way communication, the ESP32 also sends feedback or acknowledgments to the Pi. After executing a movement, the MCU might reply with an "OK" status or the new angle positions. For example, if the Pi sends a "HOME" command to center the gimbal, the ESP32 will carry out the homing routine (moving to preset reference angles) and then respond with a confirmation that homing is complete. This feedback mechanism is important for synchronization – it prevents the Pi from queueing up commands too

quickly and allows the Pi to update the user interface with current angles or status. The Pi's software uses these acknowledgments to confirm actions. If no acknowledgment is received within a short timeframe, the Pi knows something went wrong (as noted, the system has safety checks for missed acknowledgments).

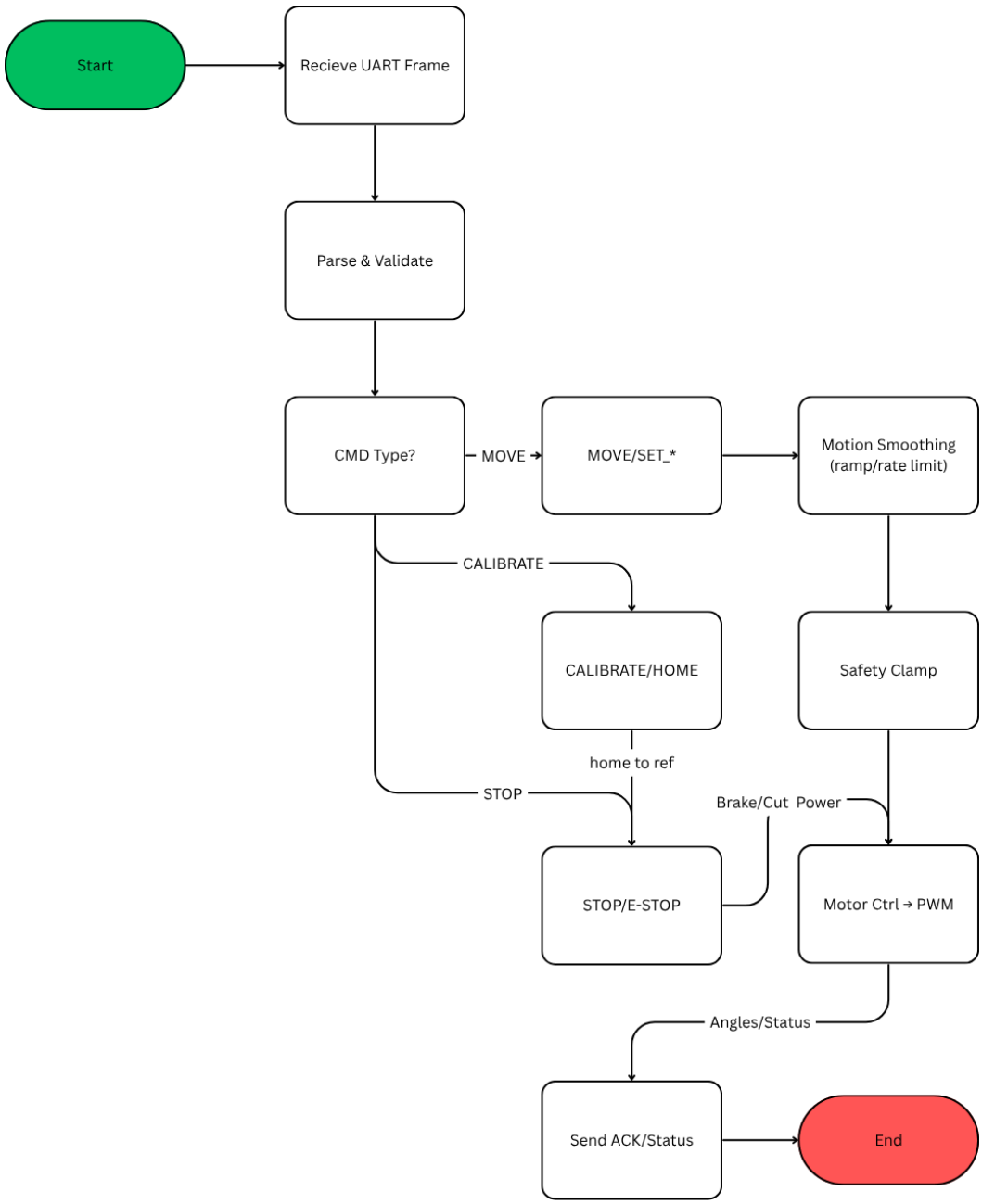


Figure 37: Activity- Handle Incoming Command Diagram

During startup, the MCU may perform an initialization routine. This could involve moving the gimbal to a known reference position (e.g., pan=0 and tilt=0, which might correspond to a "home" pointing direction). Limit switches or preset servo positions can be used to establish this reference. The firmware might include a calibration sequence to

zero out any positional error. Once initialized, the MCU signals readiness to the Pi. Thereafter, it operates in either tracking mode (accepting continuous angle updates) or manual mode (accepting direct motor commands from the user, such as "nudge left/right"). In tracking mode, the MCU essentially slaves to the Pi's stream of commands, acting as a servo controller. In manual mode, it still receives commands from the Pi, but those commands originate from user inputs (joystick or buttons) rather than the vision algorithm. The firmware itself does not fundamentally change between modes – the distinction is mostly on the Pi's side – but the rate and pattern of commands differ. Manual commands might be discrete moves or velocity commands, whereas auto-tracking commands are continuous position updates aiming to follow a target.

A critical aspect of the motor control design is ensuring safe operation. The software incorporates safeguards such as: ignoring any command that would rotate the light past its physical end stops, gradually decelerating the motors as they approach limits, and preventing oscillations. If, for example, the performer moves rapidly and the vision system overshoots, the MCU can smooth out the motion by not instantly responding to large erratic changes. This is part of the calibration to minimize jitter. The result is a stable spotlight that moves fluidly, akin to being operated by a skilled human operator. Furthermore, in the event of an emergency stop (user hits a stop button or system off), the MCU can immediately cut power to the motors or gently brake them to avoid any hazardous movement.

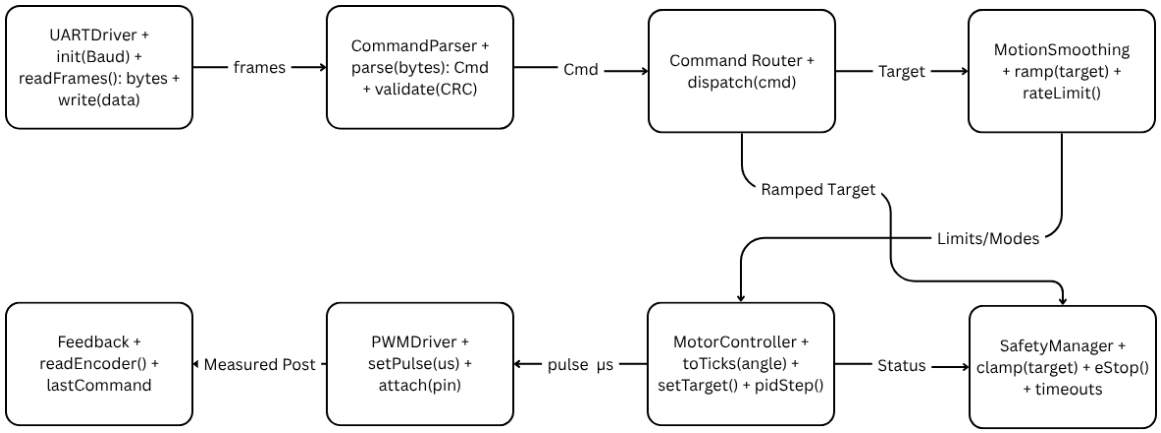


Figure 38: ESP Class Diagram

8.4 Mobile Application Interface and Wireless Communication

The mobile application is the user's gateway to monitor and control S.T.A.R. in real time. It was developed with a focus on ease of use, given that stage operators may need to quickly override or adjust the system during a live performance. Upon launching the app, the user is presented with a clean home screen that prominently features a live video feed from the spotlight's camera. This feed allows the operator to see exactly what the spotlight "sees" – essentially acting as the system's eyes. Overlaid on the video feed are useful indicators such as a bounding box around the tracked target and textual readouts of the current pan/tilt angles or zoom level (if applicable). If the system is not currently

tracking anyone, the overlay might show a message like “No target” to make clear that the spotlight is idle. A status bar on the app shows connection status (whether the app is connected to the Pi), system battery level or power status, and the active mode (Auto or Manual). These real-time telemetry features fulfill the design goal of real-time monitoring for the user, ensuring they have confidence that the system is working as intended.

User controls are intuitive and accessible directly from the main screen or a minimal menu. The primary controls include a Power toggle to turn the system on or off, a Mode toggle to switch between Automatic tracking and Manual control, target selection inputs, and manual movement controls. The Power On/Off control is typically a prominent button or switch icon. When the user toggles it on, the app initiates the system startup sequence (connecting to the Pi and enabling tracking). Toggling it off sends a shutdown command that disables tracking and can even safely power down the hardware. The app provides visual feedback of the power state (e.g., showing a green “ON” indicator or a red “OFF/disconnected” status). The Auto vs Manual mode switch is another key UI element. In Auto mode, the app indicates that the system is handling targeting automatically; in this state, manual control widgets may be hidden or disabled. In Manual mode, the app prominently displays manual control options (such as arrow buttons for up/down/left/right or a virtual joystick) and pauses any autonomous adjustments. The mode switch changes color or label to clearly show the current mode, and the transition happens immediately from the user’s perspective (with possibly a small confirmation like a toast message “Manual Control Engaged”).

For target selection, the app implements a very intuitive mechanism: the user can simply tap on the live video feed to designate a new target. If multiple people are visible, the app may also present thumbnails or a list of detected targets (e.g., “Performer A”, “Performer B”) for the user to pick from. Tapping on a person’s image or selecting a name sends a command to the Pi with either the coordinates of the tap or an ID corresponding to the chosen target. The Pi then switches the tracking algorithm to focus on that new target, and the app highlights the selection (e.g., drawing a box around the chosen person). This tap-to-track feature aligns with the goal of having straightforward override control – even while the system is in Auto mode, the user can reassign the spotlight with a quick tap if, say, the main performer changes.

In Manual mode, the app essentially turns into a remote joystick for the spotlight. It provides controls to pan and tilt the light at will. This can be done with on-screen arrow buttons or a drag interface. For example, pressing a “Left” button might continuously send a small pan-left command to the Pi (which forwards it to the ESP32) until the button is released. Similarly, there could be a slider for tilt or a two-axis virtual pad. In the implementation, when the user presses a manual control, the app sends repeated or sustained commands like “MANUAL_PAN_LEFT” to the Pi, which immediately relays them to the MCU to move the motor in real time. The user can observe the effect in the video feed as the camera moves. The design ensures that while in Manual mode, the automatic tracking loop on the Pi is paused, so it does not fight the user’s commands. The UI also prevents target selection actions in manual mode (or implicitly switches to auto if a target is selected) since tracking a target is only applicable in Auto mode.

Other features accessible via the app include a spotlight toggle (to turn off the illumination temporarily without shutting the tracking off – useful for blackout cues), a brightness slider to dim the light if hardware supports it, and a calibration/homing button. The calibration function, when invoked, sends a command for the system to recalibrate – the Pi instructs the ESP32 to move the gimbal to reference positions (like pan center, tilt floor) to re-zero any drift. During this calibration routine, the app might grey out other controls and show a progress indicator until homing is complete. All these controls are arranged in the app for quick access, likely using distinct icons and possibly a tab or accordion layout for more advanced settings (so the main screen remains uncluttered for critical functions).

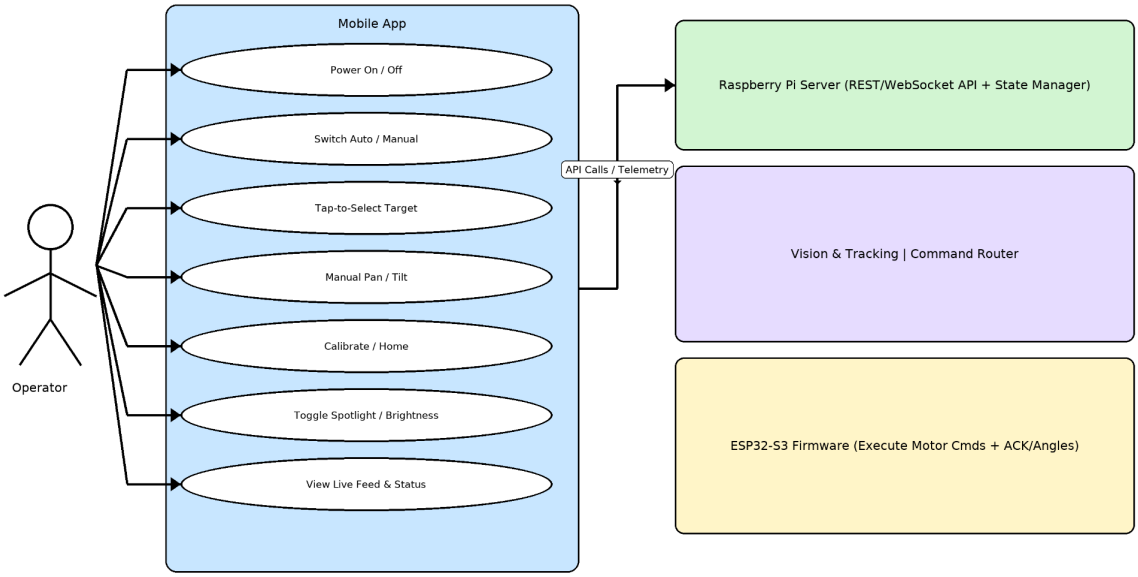


Figure 39: User Interaction Use Case Diagram

The primary user (spotlight operator) interacts with the S.T.A.R. system through the mobile app. Key use cases include powering the system on/off, switching between Auto and Manual modes, selecting a target to follow, manually controlling the spotlight direction, initiating a gimbal calibration routine, and adjusting the spotlight's brightness or beam as needed. This diagram illustrates these interactions as use cases, all facilitated by the app's interface. The app communicates the user's actions to the Raspberry Pi (over Wi-Fi), which in turn controls the spotlight hardware. By providing these use cases, the software meets the user's needs for both autonomous operation and manual override control in a performance scenario.

Under the hood, the communication between the app and the Raspberry Pi is handled via standard network protocols. The Pi runs a server (for example, a RESTful API service or WebSocket server) that the app connects to. When the app starts, it discovers the Pi on the network – either by the Pi's known Wi-Fi SSID (if it's acting as a hotspot) or via mDNS/Bonjour if on a common Wi-Fi network. Once the app finds the Pi's IP address or hostname, it initiates a handshake. This might be a simple HTTP GET request to a known

endpoint or a dedicated “HELLO” message if using a socket connection. The Pi responds to acknowledge the connection and may send back initial status information (e.g., “System Idle” or current mode and tracking status). If any authentication is required (for example, a passcode so only authorized users can control the spotlight), the app will send credentials during this handshake and the Pi will validate them. After connecting, the app maintains a live link – if using WebSockets or MQTT, this is a persistent connection, whereas if using HTTP, the app will periodically poll or send commands as needed.

The system is designed to handle connectivity issues gracefully. For instance, if the app goes out of range or the Wi-Fi signal is lost, the Pi can either pause the tracking or continue autonomously with last known instructions. When connectivity is restored, the app will resynchronize by fetching the latest state from the Pi (which mode it’s in, what target is being tracked, etc.). The software also includes a heartbeat mechanism: the Pi regularly checks that the ESP32 microcontroller is online and responding, and it can relay this status to the app. In practice, this means the Pi might send a periodic ping to the MCU (“PING”) and expect a “PONG” reply. If the MCU becomes unresponsive, the Pi flags an error that is shown on the app (alerting the user that the motor controller is not communicating). Similarly, the app could ping the Pi or simply detect a lost connection if the video stream or commands time out, then prompt the user to check the system. By integrating the mobile app via wireless communication, S.T.A.R.’s software allows for a high degree of user control without tethering the user to the device. The operator can be anywhere in the venue (within Wi-Fi range) and still monitor the spotlight’s view and intervene as needed. This wireless interface replaces the traditional physical knobs and levers of a spotlight or the expensive lighting console setups in large venues, thereby greatly increasing the system’s flexibility and ease of deployment.

8.5 System Modes and State Management

In software terms, the S.T.A.R. system operates in a few distinct modes or states that determine how input is handled and how the control logic runs. The primary states are Idle, Auto-Tracking, and Manual Override. The system can be modeled as a finite state machine (FSM) with these states, as illustrated in Figure 7.4. When the system is first powered on and initialized (e.g., the Pi and camera are on, and motors homed), it typically starts in an Idle state. In Idle, the spotlight is powered and ready but not actively tracking any target – essentially awaiting user commands. The Idle state occurs in scenarios such as: right after turning on (before a target is selected), or if the system has lost its target and is waiting, or if the system is intentionally paused without switching off. In Idle, the motors may hold their last position or a default “park” position, and the vision algorithm may either be idle or running in the background looking for potential targets.

From Idle, the user can command the system into either of the active modes. Auto-Tracking (Auto mode) is entered when the user selects a target or toggles the system into autonomous mode (assuming a target is available). In the Tracking state, the vision-processing loop on the Pi is active and continuously controlling the motors to follow the subject. The transition from Idle to Tracking might be triggered by the user

tapping a person in the camera view (causing the system to lock on and start tracking) or pressing an "Auto" button which instructs the Pi to automatically pick up a target (for example, the first person it sees). Once in Tracking, the system will remain so until something causes a state change – either the user intervenes or the target is lost. If the performer being tracked exits the camera view and no other target is auto-selected, the system may effectively revert to an Idle-like behavior (no active target) while still technically in Auto mode. In practice, we can consider that a loss of target causes a transition to an Idle state (or at least a distinct sub-state "Idle (no target)" within the Auto mode). The software could be designed to do a slow pan scan when no target is present, but it will not actively track until a new target is acquired.

The other major state is Manual Override (Manual mode). This is entered when the user toggles the mode switch to Manual on the app. The transition from either Idle or Tracking into Manual causes the Pi to halt any automatic tracking operations immediately. In Manual state, the Pi essentially passes all control to the user: it will accept manual pan/tilt commands from the app and forward them to the MCU, without running the vision algorithm for target following. The spotlight now moves only when the user instructs it. Internally, the Pi's software might disable or pause threads related to vision tracking upon entering Manual mode. If the user was in the middle of tracking someone and switches to Manual, the system will hold the spotlight at the last known position until the user manually moves it. Manual mode is typically used when the operator wants to take full control – for example, to point the light at something the vision system is not trained to recognize, or to pre-position the light.

Switching back to Auto from Manual is also a defined transition. When the user toggles to Auto mode again, the system must resume the tracking algorithm. If a previous target was being followed before Manual mode, the Pi will attempt to reacquire that same target in the camera's view. If that target is still visible, tracking seamlessly resumes (perhaps with a brief search to lock on). If the target has moved out of frame or is no longer valid, the system either stays idle awaiting a new target selection or automatically picks another detectable target (depending on implementation). The design tries to make this transition smooth – for instance, if the spotlight is already pointed near the target when Auto is re-engaged, the system will lock on quickly without a sudden jump. If the spotlight was manually pointed elsewhere, the Pi may need to rapidly slew the gimbal to find the performer again, which could cause a noticeable jump unless handled carefully. One possible strategy is to slowly interpolate from the manual position to where the target is, to avoid startling movements (this was considered in design to ensure performer safety and audience comfort).

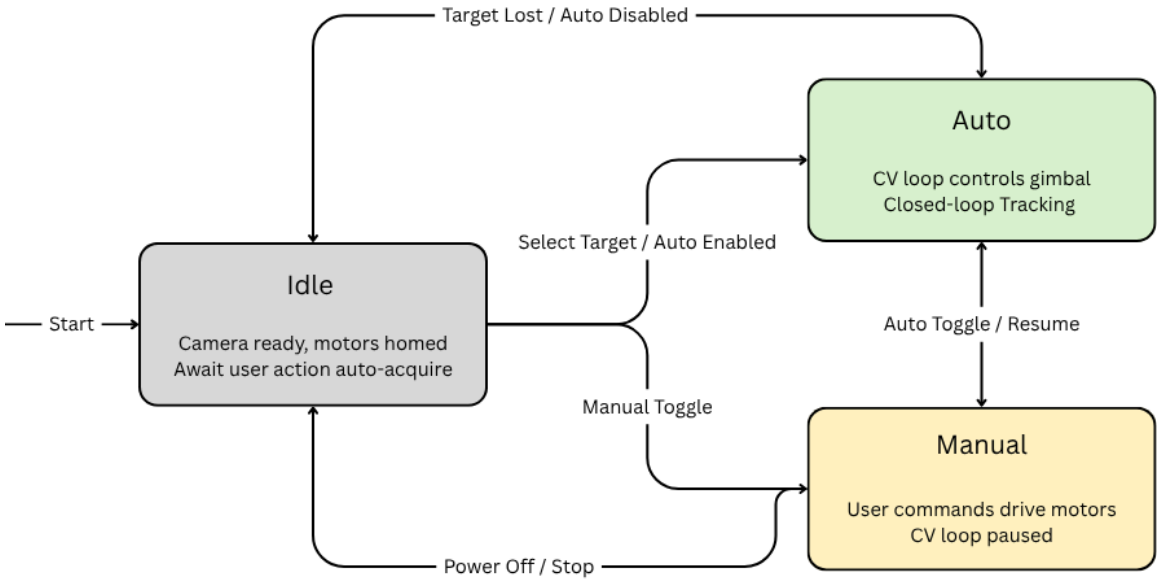


Figure 40: State Diagram of System Modes

The software transitions between three main states: Idle, Auto-Tracking, and Manual Override. Upon power-up, the system begins in Idle (after initialization). From Idle, the user can enter Auto-Tracking mode by selecting a target or enabling autonomous tracking, or enter Manual mode by taking direct control. In Auto-Tracking state, the system's vision algorithm actively controls the gimbal to follow the target until the user switches mode or the target is lost (which causes a transition back to Idle). In Manual state, all automatic tracking is suspended and the user's inputs directly move the gimbal; switching back to Auto will resume tracking of the last or a new target. This state machine ensures that automatic and manual controls do not conflict – only one mode is active at a time – and defines how the system responds to mode change events.

The mode management logic in the software makes certain guarantees for safety and performance. For one, when in Manual mode, any incoming automatic adjustment commands are ignored or buffered. The user should never feel the system "fighting" their input. Similarly, in Auto mode, manual joystick commands are not processed (or the app UI may not even send them) to prevent unintended interference. The system also implements a debounce on mode switching – if the user flips modes rapidly, the software might enforce a short delay (e.g., 1 second) between mode changes to allow the system to stabilize. This prevents oscillation or rapid toggling that could confuse the control loops. Additionally, when no target is being tracked in Auto mode, the system either stays idle or performs a gentle scan as mentioned, but it will not arbitrarily jump to Manual or turn off without user command. Each state transition is initiated by a clear event (user action or target loss). This predictability is important for the operator to trust the automation: they know that the system won't spontaneously change modes without input.

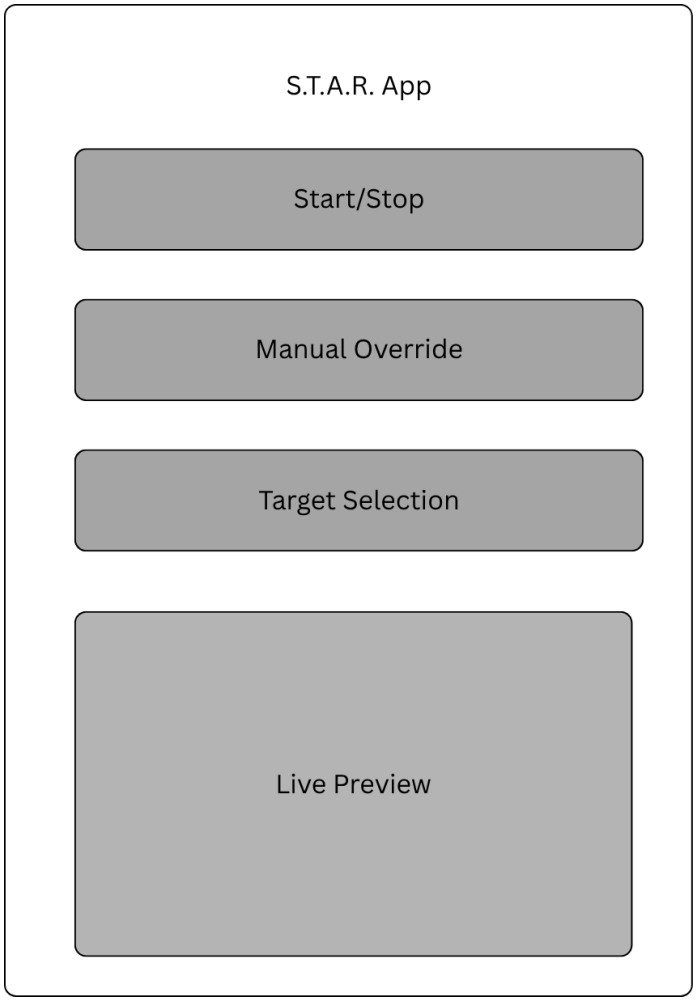
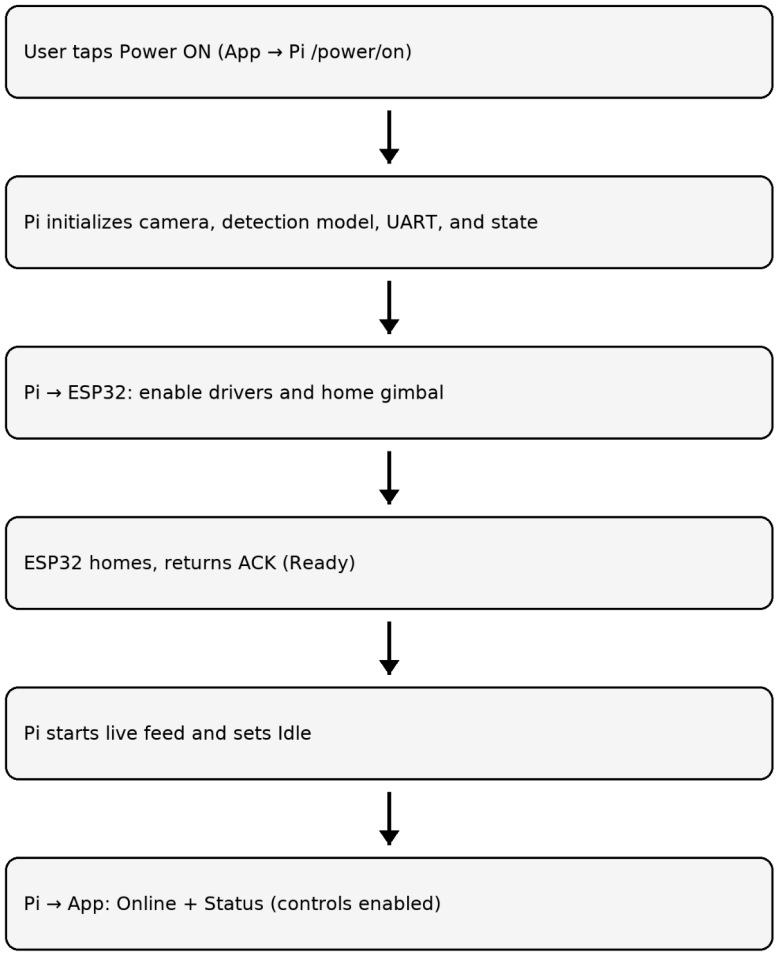


Figure 41: App Layout

8.6 Control Flowcharts and Sequence Diagrams

To further clarify the software design, this section presents key flowcharts of how the system operates in response to certain events. These sequences tie together the subsystems (app, Pi, MCU) and demonstrate the interactions described above in a step-by-step manner. By visualizing these flows, one can trace how a user action propagates through the software and hardware to achieve the desired outcome.

8.6.1 System Startup and Shutdown Sequence



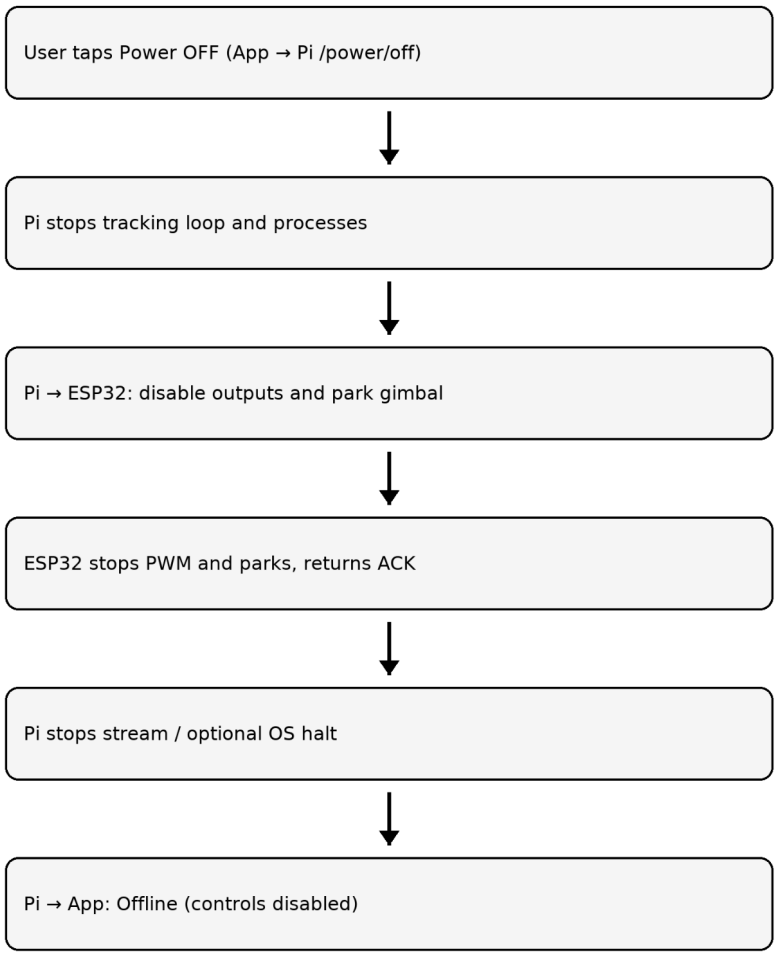


Figure 42: System Power-On and Power-Off Flowcharts

This diagram illustrates the sequence of operations when the user turns the S.T.A.R. system on (left) and off (right) via the mobile app. (Power On): The user tapping the "Power On" button in the app triggers the app to send a system-on request to the Raspberry Pi. The Pi then initializes all subsystems – it establishes communication with the ESP32, initializes the camera and vision algorithm, and sets up internal state variables. Next, the Pi sends a command to the ESP32 to enable the motor drivers and prepare the gimbal (for example, release any motor hold and run a self-test or homing routine). The ESP32, upon successful activation, replies with an acknowledgment, which the Pi relays back to the app. Finally, the app updates its UI to indicate the system is online – it may now display the live video feed and allow the user to select targets or modes. At this point the system enters an Idle/standby state, powered up and ready for action.

(Power Off): When the user presses "Power Off" in the app, a shutdown command is sent to the Pi. The Pi then signals all processes to stop – it halts the tracking loop and informs the ESP32 to disable motor outputs (and turn off the spotlight lamp if applicable). The ESP32 carries out these steps (e.g., stopping PWM signals to the servos, potentially

moving the gimbal to a safe “park” angle) and responds with a shutdown acknowledgment. The Pi may then either fully shut down its OS or at least inform the app that shutdown is complete. The app reflects this by changing the status to “Off/disconnected” and disabling controls until the system is turned on again. Safety checks are present throughout – if any subsystem fails to respond during power-up or power-down, the app will alert the user of the error. This sequence ensures an orderly and reliable startup/shutdown, protecting the hardware (for instance, ensuring motors aren’t active when they shouldn’t be, and the light turns off gracefully).

During power-on, one important step is the initialization of the vision system on the Pi. The software likely loads the trained detection model into memory and may perform an initial camera exposure/white-balance adjustment. This can take a short moment, which is why the app might show a “Starting…” indicator until the camera feed is live. In the power-off flow, the Pi’s shutdown command to the ESP32 could include turning off the LED spotlight to prevent any unwanted light output as the motors power down. The ESP32’s firmware ensures that the gimbal comes to rest – for example, if it was mid-move when the shutdown occurred, it stops smoothly to avoid any sudden jerk or mechanical stress.

8.6.2 Target Selection and Tracking Update Sequence

Another important scenario is when the user selects a new target to track while the system is running, or when the system automatically switches targets. The flow is as follows: the user selection on the app (tapping a target or choosing from a list) causes the app to send a SELECT_TARGET command to the Pi with the specified target’s identifier or screen coordinates. The Raspberry Pi then processes this command – if the target ID corresponds to a detection the vision algorithm already knows, it simply locks onto that. If coordinates are given (from a tap), the Pi directs the vision algorithm to focus on that region of the image to find the nearest object. Once the new target is acquired, the Pi resets or updates its tracking parameters for that target (e.g., it might initialize a new tracking filter). It may also send an immediate motor command to roughly center the gimbal on the new target if the change is large, to speed up acquisition. The continuous tracking loop (Figure 7.2) then takes over to fine-tune and continue following. The ESP32 receives the updated pan/tilt commands and moves the motors, resulting in the spotlight shifting to illuminate the new subject. The app, upon a successful switch, updates its UI highlights – e.g., highlighting the newly selected target’s name or drawing a new bounding box – to confirm the change to the user. All of this happens within a second or two, allowing relatively quick hand-offs between performers.

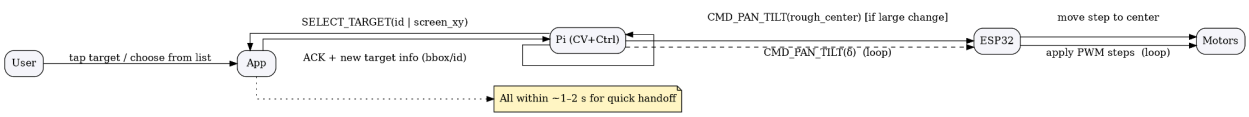


Figure 43: Target Selection Sequence

If the user rapidly wants to switch again, the system can handle it; however, to maintain smoothness, the software might impose a short delay or smoothing as the gimbal moves to the new position. For example, if the user taps a far-left person then immediately taps a far-right person, the Pi will try to avoid a sudden whip-pan. It could either ramp the motion or momentarily slow down the change. This behavior falls under the earlier mentioned motion smoothing and is part of delivering a professional look. In fully automatic operation (with multi-target detection enabled), the system could decide on its own to switch targets under certain conditions (though the base design keeps one target until user changes it). For instance, if the current target leaves the stage and another person enters, the software might switch to the new person after a configurable delay. However, such behavior would be carefully controlled to avoid unpredictable changes; the user's manual selection always takes priority.

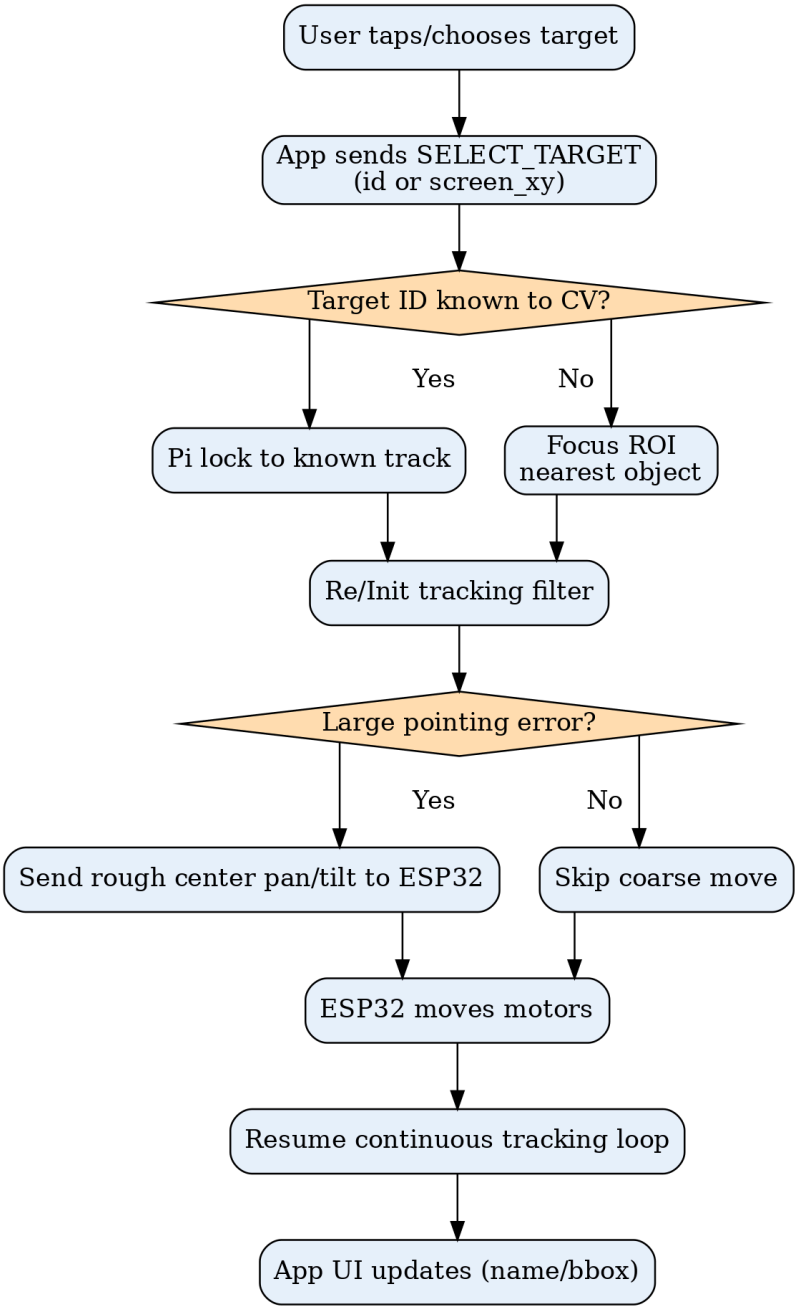


Figure 44: Continuous Tracking Loop Flowchart

8.6.3 Manual Override Control Sequence

When the user takes manual control, the flow of information changes slightly from the normal tracking loop. Instead of the vision algorithm driving the motors, the user inputs drive them. For example, suppose the spotlight is currently tracking someone, and the user toggles to Manual mode and presses the “pan left” button on the app. The sequence would be: the app sends a SET_MODE(MANUAL) command to the Pi, which the Pi

acknowledges by stopping the vision loop. Now in Manual state, the app's "pan left" press causes it to send a MANUAL_PAN_LEFT message (perhaps repeatedly, or with a parameter like an angle increment) to the Pi. The Pi, which is essentially pass-through in manual mode, forwards this command to the ESP32 immediately. The ESP32 then rotates the pan motor to the left by a small step. If the user holds the button, multiple commands stream and the motor continues to move until released. Throughout this, the Pi might still be reading the camera (especially if streaming video to the app), but it is not using those images to control anything – it could optionally still show the video so the user sees where they are aiming.

During manual movement, the ESP32 can send feedback to the Pi about the current angles (particularly if it has encoders or if it just uses the last commanded value). The Pi can use this to update any angle display on the app in real time. This creates a feedback loop where the user sees the spotlight moving in the live feed and also sees numeric angle changes or a moving joystick indicator reflecting the new position. The processing load on the Pi in manual mode is much lighter, since it's not running the CV algorithm; it mostly shuffles commands between the app and MCU.

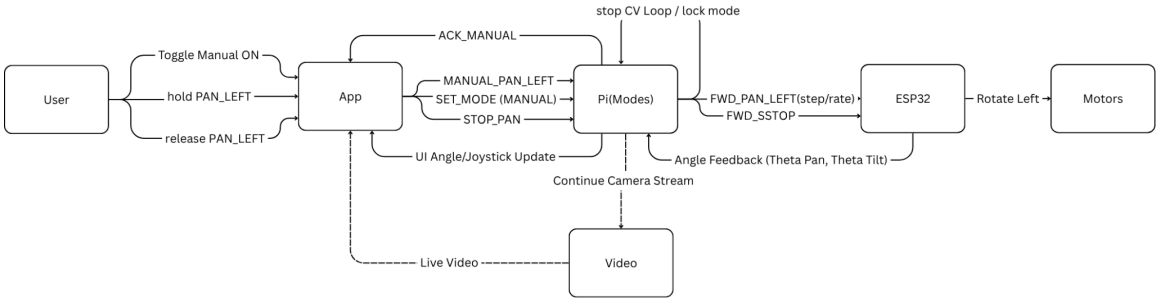


Figure 45: Manual Override Sequence

When the user decides to return to Auto, they toggle the switch back. The app sends SET_MODE(AUTO) to the Pi, which re-engages the tracking software. At this moment, the Pi has to determine what to track. Often it will attempt to lock onto the same subject that was last tracked (before manual mode) if that subject is still reasonably in view. If the spotlight was manually pointed way off, the target might not be in frame; in that case the system could either do nothing (wait for user to select a target) or attempt a guess like scanning or choosing a visible person. In a user-driven workflow, typically the operator would manually point the spotlight toward the next target and then hit Auto, effectively handing off the target to the system. The transition is designed to be as smooth as possible – for example, if the user had already centered the new target manually before switching to Auto, the system will simply take over and maintain that lock without any sudden movement. Software-wise, re-entering Auto mode means the Pi resumes sending continuous tracking commands and the app disables the manual controls again.

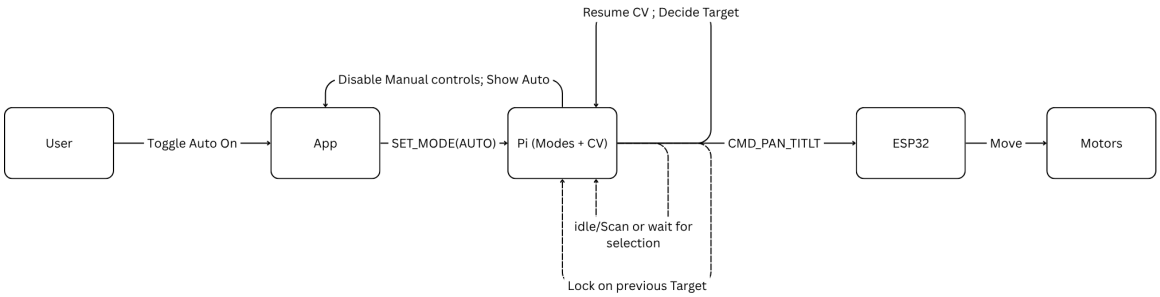


Figure 46: Mode Transition Sequence

Throughout manual/auto toggling, the software ensures that only the appropriate inputs are acted on. The system also prevents any conflicting situation (like a scenario where a leftover tracking thread tries to move the motors while in manual – which is prevented by the mode logic locks). In summary, the manual override sequence allows the user to seamlessly interject their control into the system and just as easily hand control back to the autonomous system, a critical feature for real-world use where lighting directors might need to make artistic adjustments on the fly. Overall, the software design – from high-level architecture down to detailed sequences – provides a comprehensive framework for automated spotlight control. It modularizes functionality into the vision processing on the Pi, real-time motor control on the MCU, and user interaction through the app, tying everything together with robust communication protocols. The included diagrams and descriptions illustrate how each part of the software contributes to the system's goals: reliably tracking a performer with low latency, while allowing an operator to monitor and override the system effortlessly. Through careful state management, flow control, and subsystem integration, the S.T.A.R. project's software meets the technical requirements and provides a user-friendly, performance-ready solution for automated spotlight tracking.

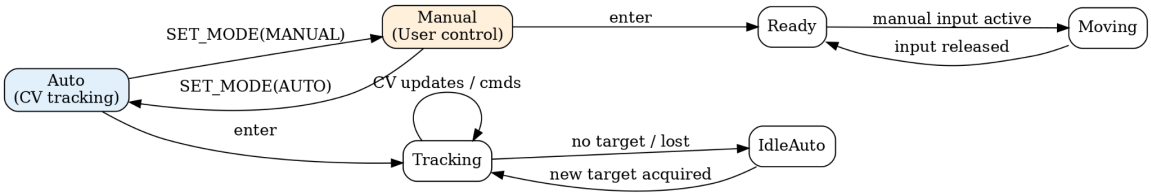


Figure 47: System Modes State Diagram

Chapter 10 - Administrative Content

10.1 Budget and Financing

The projected budget for this project is $1,700, which reflects both the technical complexity and the specialized hardware requirements necessary to achieve reliable real-time spotlight tracking. A significant portion of this cost comes from PCB manufacturing, as custom-designed boards are required to integrate the MCU, ICs, etc.

into a compact, stage-ready system. Not to mention the cost of international shipping. Additionally, the optical components, including a high-quality LED emitter, reflector, and focusing lenses, contribute heavily to the expense, since precision glass optics and lighting elements for professional-grade illumination are costly. The motorized gimbal system also represents a major investment, as it must use high-torque, low-noise servo motors to provide smooth, precise, and silent operation suitable for performance environments. Beyond hardware, funds must also be allocated for testing, prototyping, and iteration, as initial prototypes may require redesigns or component replacements. Compared to off-the-shelf lighting systems, this budget remains reasonable, as it accounts for both performance-level functionality and the flexibility of a custom-engineered design. The group has agreed to split the total cost of this project equally between all 4 members.

Table 15: General BOM

Part Name	Vendor	$/unit or lot	QTY	Note
ESP32-S3 Chip	Digikey	$5.00	1	
Raspberry Pi Compute Mod. 5	Digikey	$65.00	3	65-80 depending on model
Servo Motors (shutter mech.) 4x	Amazon	$20.00	1	
Servo Motors (high torque)	Amazon	$60.00	3	50-110 depending on model
Portable Monitor	Amazon	$60.00	1	
RP Camera HQ	Digikey	$50.00	1	
HPL757 Lamp	ETC	$19.00	1	
Illumination lens	ETC	$60.00	1	
Source Four Reflector	ETC	$65.00	1	
VIS-NIR Achromatic Lens	Edmund Optics	$79.38	1	
Uncoated Plano-Concave Lens	Edmund Optics	$28.62	1	
Misc. PCB components	Digikey	$150	1	Rough estimate
PCB Fabrication	JLCPCB	$300	1	Rough estimate
EST. TOTAL		**$1,104.00**		Will most likely exceed est. total due to unforeseen costs

The general Bill of Materials (BOM) for the S.T.A.R. system lists all major electrical, optical, and mechanical components used in the design. Core control hardware includes an ESP32-S3 microcontroller and Raspberry Pi Compute Module 5 units, which handle

system processing and communication. The optical assembly consists of components sourced mainly from ETC and Edmund Optics, such as the HPL757 lamp, illumination lens, Source Four reflector, and precision achromatic and plano-concave lenses for imaging. The system also includes several servo motors for movement, a portable monitor for visual interface, and an RP Camera HQ for image capture.

Electronics and fabrication costs account for a significant portion of the budget. Miscellaneous PCB components and fabrication services from Digikey and JLCPCB were estimated at around $450 combined. The total projected cost of the S.T.A.R. system is approximately $1,104, though the team anticipates that final expenses may exceed this estimate due to unforeseen costs such as replacement parts, additional wiring, or updated module versions. Overall, the BOM demonstrates a balanced mix of off-the-shelf components and specialized optics to meet the system's illumination and imaging requirements.

10.2 Project Milestones

The milestones for the S.T.A.R. project outline the planned timeline and deliverables for both semesters of the Senior Design sequence. These checkpoints help the team stay on schedule by dividing the project into specific phases, each with measurable goals. During Senior Design I (Fall 2025), the main focus is on concept development, research, and preliminary prototyping. Early weeks involve group formation, selecting a project idea, and drafting the Divide and Conquer (D&C) document. Later milestones include research on key components such as the CMOS camera, microcontroller, and motorized gimbal, followed by basic integration testing and prototype development. The semester concludes with the submission of the final report and demonstration video.

Table 16: Milestones Fall 2025

Senior Design I - Fall 2025			
Week	Milestone	Dates	Deliverable
1-2	Group Formation and Project Idea	8/18-8/29	Finalized Project Idea with Required Components
3	D&C Document	9/1 - 9/5	Finished D&C Document
4	D&C Revision	9/8	Revised D&C Document
5-6	Research on CMOS Camera and MCU	9/9 - 9/21	Selection of CMOS and MCU
5-6	Research on ML model for object tracking and how to interface it with MCU	9/9 - 9/21	Demo application of of object tracking

Senior Design I - Fall 2025			
7-8	Research motorized GIMBAL, PTU, and PCB Design	9/22 - 10/5	Demo application of Spotlight Tracking Person
9	Research on lowlight optimization	10/6 - 10/19	Demo application of Spotlight Tracking Person in near dark
10	Research on Wireless and UI Interface	10/20-10/26	App/web control demo
11-15	Integration testing	10/27 - 11/17	Prototype built
16	Final Report and Demo Video Due	11/25	Full Report Due

The Senior Design II (Spring 2026) milestones will focus on hardware implementation, testing, and system integration. Major tasks include PCB fabrication and testing, assembling the prototype, and refining system software. Once integration is complete, the team will conduct full system debugging and prepare for the final presentation. These milestones ensure that each stage of the project builds toward a complete and functional product, keeping development organized and on track for the end-of-semester demonstration.

Table 17: Milestones Spring 2026

Senior Design II - Spring 2026			
Week	Milestone	Dates	Deliverable
TBD	PCB Testing	TBD	Test PCB
TBD	Integrated Code testing	TBD	Have working application for PCB
TBD	PCB Assembly	TBD	Assembled Prototype
TBD	PCB Testing and Debugging	TBD	Assemble Device
TBD	Final Presentation	TBD	Present final product

In order to ensure an efficient workflow, we have assigned specific tasks to each member of the group based on their area of expertise. The photonics engineers, Travis and Gage,

are responsible for the optical and imaging aspects of the project. Their work includes lens design, optical calculations, selecting and integrating the image sensor and camera module, and developing the illumination and shutter systems. They also manage the system display output and production of the demo video. The computer engineer, Lucio, is responsible for the software portion of the system. His tasks include developing the computer vision algorithms, designing the user interface, coding the system logic, and managing motor control for the gimbal. He also maintains the Senior Design website. The electrical engineer, Jerison, focuses on PCB design, electrical circuitry, and gimbal motor control. He is also in charge of 3D modeling and printing mechanical components for housing and support. This distribution of work allows each team member to specialize in their field while ensuring smooth integration across the optical, electrical, and software systems.

Table 18: Work Distribution Table

Photonics Engineers	**Responsibilities**
Travis and Gage	Lens design
	Optical calculations
	Image sensor / Camera module
	System display output
	Illumination shutter system
	Demo video production
Computer Engineer	**Responsibilities**
Lucio	Computer vision algorithms
	System UI
	System logic coding
	Motor command code
	Senior Design website
Electrical Engineer	**Responsibilities**
Jerison	PCB design
	Motor logic for gimbal
	Illumination shutter system

	3D design & printing
	Circuitry & electrical components

Chapter 11 - Conclusion

The S.T.A.R. (Spotlight Tracking with Automated Recognition) system represents a multidisciplinary senior design project developed at the University of Central Florida. This project integrates principles of photonics, electrical engineering, and computer engineering to create an autonomous spotlight capable of detecting and tracking human subjects in real time. Building upon the legacy of previous UCF projects such as CHROMA, and existing products such as the ETC Source Four, S.T.A.R. extends the concept of intelligent lighting into the live performance environment by combining optical precision, embedded control, and computer vision into a single self-contained unit. The motivation for this design stemmed from the need to reduce reliance on manual spotlight operation in stage productions and presentations, offering a more affordable and accessible alternative to large-scale automated lighting systems used in professional theaters.

Summary of Design and Integration

The system merges multiple engineering disciplines into a cohesive prototype. The optical subsystem was modeled after the ETC Source Four ellipsoidal reflector spotlight, a well-established standard in stage lighting. Through analysis of reflecting versus refracting optics, the design implements an ellipsoidal cold mirror reflector paired with a biconvex focusing lens to achieve an output beam angle of approximately 19°. The use of N-BK7 glass provides high transmission efficiency in the visible spectrum, while internal baffles minimize stray light for improved beam contrast. This optical design ensures that the illumination maintains professional quality while remaining compact and thermally stable.

The imaging subsystem employs a Raspberry Pi High Quality Camera with a Sony IMX477 CMOS sensor, chosen for its high frame rate, sensitivity, and affordability. The imaging optics use commercial off-the-shelf lenses to achieve the required magnification and minimize aberrations. Through this system, the camera captures continuous video for human detection and tracking. On the computational side, a Raspberry Pi Compute Module 5 executes the tracking algorithm using OpenCV and a trained YOLO model, while an ESP32-S3 microcontroller handles gimbal motor control and communication. Together, these subsystems form an integrated pipeline, from image capture and recognition to motorized actuation that achieves tracking accuracy within 5° RMS at 10 m and latency under 250 milliseconds.

The mechanical and electrical systems were designed for reliability, heat management, and modularity. High-torque servo motors provide smooth two-axis pan and tilt

movement, while a secondary set of servos drives the shutter mechanism. The PCB design integrates motor drivers, logic control, and communication interfaces within a compact footprint. HT-PLA was selected as the housing material for its lightweight strength and heat resistance, allowing the fixture to remain durable without the need for metal fabrication. The modular construction of the system supports easy maintenance and future upgrades, including potential LED light-engine integration.

Interdisciplinary Collaboration

Development of S.T.A.R. will require consistent coordination between team members specializing in photonics, electrical engineering, and computer engineering. The photonics engineers will focus on optical design, sensor integration, and illumination system analysis; the computer engineer will develop the vision algorithm, system interface, and embedded control logic; and the electrical engineer will design the power systems, PCB layout, and mechanical housing. This division of labor allows the team to efficiently address each subsystem's challenges while maintaining consistent system-level communication. Integration testing, simulation, and modeling ensure that each component will reliably function within the complete assembly.

This collaboration reflects the interdisciplinary nature of modern engineering projects. Optical engineers must understand electronic control systems, while software developers must account for real-world optical constraints such as focus, field of view, and latency. The success of the S.T.A.R. project demonstrates how these specialties intersect to produce a practical, automated system capable of meeting both technical and creative requirements.

Research Contributions and Learning Outcomes

Throughout development, the team explored and compared multiple technologies to determine the most effective balance between performance, cost, and complexity. Research into illumination systems evaluated halogen, LED COB, and RGBW light engines, ultimately identifying the LED COB module as the most efficient choice for future iterations. Optical design research applied principles such as Snell's Law, the Lensmaker's equation, and numerical aperture calculations to determine lens geometries and focal lengths. Similarly, research on sensor technologies compared CCD and CMOS architectures to assess trade-offs between noise performance, speed, and power consumption.

Beyond technical understanding, the project provides significant hands-on experience with professional design tools and fabrication techniques. The team will gain practical knowledge in optical simulation, PCB layout, 3D modeling, and embedded programming. The experience of managing procurement, budgeting, and interdisciplinary collaboration will also reinforce professional engineering practices, which are skills that extend beyond the classroom into industry applications.

Challenges and Future Improvements

As with any complex engineering system, the development of S.T.A.R. will involve numerous technical and logistical challenges. Optical alignment, heat management, and signal synchronization between the imaging and motor subsystems will require careful calibration. Latency reduction will be particularly demanding, as delays could arise at multiple stages—image processing, command transmission, or mechanical response. Further optimization of software algorithms and microcontroller communication protocols may yield even faster and smoother tracking.

For future iterations, several enhancements have been identified. The first involves replacing the halogen light source with a high-power LED COB module to improve efficiency and longevity while maintaining brightness. The second focuses on refining the gimbal mechanics with brushless motors and encoders for higher precision and reduced noise. Expanding the computer-vision model to support multi-subject tracking and user selection would further enhance versatility. Additional goals include implementing DMX/Ethernet control for compatibility with professional lighting consoles, improving the mobile UI for real-time calibration, and developing thermal management solutions such as passive heat sinks or forced air cooling.

Broader Impact and Future Applications

S.T.A.R. demonstrates how automation and computer vision can enhance accessibility in live performance technology. The same principles applied here—autonomous motion tracking, intelligent illumination, and wireless control—can be extended to other environments such as lecture halls, film studios, and industrial inspection systems. By leveraging affordable components and open-source software, the project lowers the barrier for integrating intelligent lighting into small-scale venues or educational settings. In doing so, it bridges the gap between professional automation systems and entry-level stage technology, enabling a broader range of users to benefit from advanced optical engineering.

The project also contributes to the growing intersection between photonics and computer science. As imaging sensors and embedded processors become increasingly powerful, autonomous optical systems like S.T.A.R. will continue to evolve. The team's work provides a foundation for future UCF students to build upon, whether by refining the optics, optimizing code performance, or expanding functionality through artificial intelligence and networking.

Final Remarks

In conclusion, the S.T.A.R. project represents the integration of optical design, embedded electronics, and computer vision into a unified system capable of dynamic, real-time human tracking. It demonstrates the practical application of engineering in solving real-world problems while emphasizing affordability, modularity, and user accessibility. Through detailed research, structured milestones, and collaborative teamwork, the project

will achieve its foundational objectives and establish a platform for future innovation in automated lighting. S.T.A.R. not only showcases the technical abilities of its designers but also serves as an example of how multidisciplinary collaboration can transform a conceptual idea into a functional, impactful engineering solution.

Appendix A - References

[1] T. B. Greenslade and S. G. Heisler, “Stage Lighting Instruments,” *The Physics Teacher*, vol. 13, no. 9, pp. 548–551, 1975.

[2] A. P. Kshirsagar and H. Azath, “A Comprehensive ATM Security Framework for Detecting Abnormal Human Activity via Granger Causality-Inspired Graph Neural Network Optimized with Eagle-Strategy Supply-Demand Optimization,” *Expert Systems with Applications*, vol. 272, 2025.

[3] R. Holzer, “YOLO — Object Detection,” *OpenCV Tutorial*, 2019. [Online]. Available: https://opencv-tutorial.readthedocs.io/en/latest/yolo/yolo.html

[4] “The Ultimate Guide to Lens Design Forms: The Types of Optical Systems in a Lens Designer’s Toolbox,” *Pencil of Rays*. [Online]. Available: https://www.pencilofrays.com/lens-design-forms/

[5] M. Kaur, et al., “Security Camera with Frame Detection,” in *Proc. 2024 Int. Conf. on Emerging Smart Computing and Informatics (ESCI)*, IEEE, 2024, pp. 1–5.

[15] D. Korsch, *Reflective Optics*. Boston, MA: Academic Press, 1991.

[16] Z. Djafar, A. Z. Salsabila, and W. H. Piarah, “Performance Comparison Between Hot Mirror and Cold Mirror as a Beam Splitter on Photovoltaic-Thermoelectric Generator Hybrid Using LabVIEW Simulator,” *Heat and Technology*, vol. 39, no. 5, pp. 1609–1617, 2021.

[17] A. Q. Gadhban, H. M. Roomy, and S. K. Taha, “Design High Performance Multilayers Cold Mirror,” *J. Phys.: Conf. Ser.*, vol. 1879, no. 3, p. 032086, 2021.

[18] Newport Corporation, “Paraboloidal and Ellipsoidal Reflectors.” [Online]. Available: https://www.newport.com/f/paraboloidal-and-ellipsoidal-reflectors

[20] I. G. Priest and U.S. National Bureau of Standards, *A New Method of Determining the Focal Length of a Converging Lens*. Washington, DC: U.S. Dept. of Commerce and Labor, Bureau of Standards, 1909.

[21] Electronic Theatre Controls Inc., “Gobo Holder Comparison for Source Four.” [Online]. Available: https://support.etcconnect.com/ETC/Fixtures/Source_Four/Source_Four_ERS_and_HID/Gobo_Holder_Comparison_for_Source_Four

[22] “Landscape Lighting Assembly Having Stackable Gobo Sections,” *U.S. Patent Application 20190078761*, 2019.

[23] “LED Light Fixture for Indirect Lighting with Adaptable Baffle Structure,” *U.S. Patent Application 20190309917*, 2019.

[24] J. A. Jimenez, “Enhanced NIR Emission from Nd^{3+} Ions in Plasmonic & Dichroic Cu Nanocomposite Glass,” *Chemical Physics Letters*, vol. 824, 2023.

[25] A. S. H. Md Yasir, et al., “Effect of Heat Treatment on Mechanical Properties and Dimensional Accuracy of 3D-Printed Black Carbon Fiber HTPLA,” *Heliyon*, vol. 10, no. 11, 2024.

[26] E. Juntunen, et al., “Copper-Core MCPCB with Thermal Vias for High-Power COB LED Modules,” *IEEE Trans. Power Electron.*, vol. 29, no. 3, pp. 1410–1417, 2014.

[27] R. D. Araguillin, et al., “Spectral Power Distribution of the Tungsten-Halogen Lamp (Blue) Compared with the Blackbody Spectrum with the Equivalent Color Temperature (Red),” *ResearchGate*, 2025. [Online]. Available: https://www.researchgate.net/figure/Spectral-power-distribution-of-the-tungsten-halogen-lamp-blue-compared-with-the_fig2_382400303

[28] Gigahertz-Optik GmbH, “ISS-15-RGBW.” [Online]. Available: https://www.gigahertz-optik.com/en-us/product/iss-15-rgbw/

[29] Electronic Theatre Controls Inc., “Source 4WRD Color Tech Specs.” [Online]. Available: https://www.etcconnect.com/Products/Legacy/Lighting-Fixtures/S4WRD-C/Tech-Specs.aspx

[30] Flutter, “Build UIs.” [Online]. Available: https://docs.flutter.dev/ui

[31] Flutter, “Performance Best Practices.” [Online]. Available: https://docs.flutter.dev/perf

[32] Meta, “React Native — Getting Started.” [Online]. Available: https://reactnative.dev/docs/getting-started

[33] Meta, “React Native — Optimizing Performance.” [Online]. Available: https://reactnative.dev/docs/optimizing-performance

[34] Google, “Dart Language Tour.” [Online]. Available: https://dart.dev/guides/language/language-tour

[35] Meta, “React Native — Architecture (JSI, TurboModules).” [Online]. Available: https://reactnative.dev/architecture

[36] R. T. Fielding, M. Nottingham, and J. Reschke, “HTTP Semantics,” *RFC 9110*, Jun. 2022. [Online]. Available: https://www.rfc-editor.org/rfc/rfc9110

[37] M. Thomson and C. Benfield, “HTTP/2,” *RFC 9113*, Jun. 2022. [Online]. Available: https://www.rfc-editor.org/rfc/rfc9113

[38] I. Fette and A. Melnikov, “The WebSocket Protocol,” *RFC 6455*, Dec. 2011. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6455

[39] OASIS, “MQTT Version 3.1.1,” *OASIS Standard*, Dec. 2015. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

[40] OASIS, “MQTT Version 5.0,” *OASIS Standard*, Mar. 2019. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html

[41] Eclipse Foundation, “Eclipse Mosquitto — An Open Source MQTT Broker.” [Online]. Available: https://mosquitto.org/

[42] S. Tiangolo, “FastAPI User Guide.” [Online]. Available: https://fastapi.tiangolo.com/

[43] Pallets, “Flask Documentation.” [Online]. Available: https://flask.palletsprojects.com/

[44] Python Software Foundation, “Asyncio — Asynchronous I/O.” [Online]. Available: https://docs.python.org/3/library/asyncio.html

[45] OpenCV Team, “OpenCV Documentation and Tutorials.” [Online]. Available: https://docs.opencv.org/

[46] Ultralytics, “YOLOv8 Documentation.” [Online]. Available: https://docs.ultralytics.com/

[47] Raspberry Pi Ltd., “Cameras — Documentation (libcamera).” [Online]. Available: https://www.raspberrypi.com/documentation/computers/cameras.html

[48] Espressif Systems, “ESP-IDF Programming Guide.” [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/

[49] Espressif Systems, “Arduino Core for ESP32 — Documentation.” [Online]. Available: https://docs.espressif.com/projects/arduino-esp32/en/latest/

[50] Adafruit, “PCA9685 16-Channel PWM Servo Driver.” [Online]. Available: https://learn.adafruit.com/16-channel-pwm-servo-driver

[51] Flutter, “Navigation and Routing.” [Online]. Available: https://docs.flutter.dev/ui/navigation

[52] Meta, “React Native — Networking and WebSocket.” [Online]. Available: https://reactnative.dev/docs/network

[53] Google, “Material Design 3 — Design System.” [Online]. Available: https://m3.material.io/

Appendix E - ChatGPT Prompts

[1a] ***Prompt:*** "*Explain the differences between a COB lighting module and an RGBW engine. I understand that they are both LED driven, but what are some key differences? Please explain in one paragraph*"

[1b] ***Response:*** *"A COB (Chip-on-Board) lighting module and an RGBW engine are both LED-based systems, but they serve different purposes and are built differently. A COB module packs many small LED dies tightly together on a single substrate to create a large, uniform light-emitting surface. This design produces very high brightness and smooth, even output, making COBs ideal for applications like stage fixtures, spotlights, and general illumination where consistent white light and strong punch are needed. In contrast, an RGBW engine uses separate red, green, blue, and white LEDs—often arranged in clusters or mixed through optics—to create a wide range of colors, including white light through color mixing. RGBW engines offer precise color control and dynamic effects, but typically have less uniformity and may show color separation without proper optics or diffusion. In short, COBs prioritize brightness and uniform white output, while RGBW engines prioritize color flexibility and tunability."*

[2a] ***Prompt:*** *"Explain the reasoning behind putting baffles in a lighting fixture. I understand it helps with reflection and stray light, but please provide a quick summary on the benefits"*

[2b] ***Response:*** *"Baffles are added to lighting fixtures to control and shape the light more precisely by blocking stray rays and reducing internal reflections. They help prevent light from spilling where it isn't wanted, improving beam definition and maintaining clean edges, which is especially important in stage and architectural lighting. Baffles also reduce glare by shielding the light source from direct view at certain angles, making the output more comfortable for audiences or occupants. Additionally, they enhance contrast and image clarity in optical systems by limiting light that would otherwise wash out the beam or reduce sharpness. Overall, baffles provide better beam quality, visual comfort, and more professional, controlled lighting performance."*

[3a] ***Prompt:*** *"What is the heat output of an ETC Source Four tungsten-halogen lamp?*

[3b] ***Response:*** *"The ETC Source Four (halogen version) emits a total heat output that can be calculated as* ***3.412 BTU per hour per watt*** *of electrical input. Electronic Theatre Controls Inc*

- *For the 575 W HPL lamp: ~1,962 BTU/hr. Electronic Theatre Controls Inc*
- *For the 750 W HPL lamp: ~2,559 BTU/hr. Electronic Theatre Controls Inc*

If you tell me the exact wattage of your lamp I can compute the exact BTU/hr for your specific fixture.

[4a] ***Prompt:*** *"I want to expand on reasons why you would want to use a lens instead of a reflector. Can you write a paragraph or so explaining this?"*

[4b] ***Response:*** *"While reflectors are highly efficient for collecting and redirecting light, they alone cannot achieve the level of* ***beam precision and focus control*** *required in professional illumination systems. Reflective surfaces shape light primarily through geometric reflection, which is ideal for collimation or spreading light broadly, but they lack the fine spatial control necessary to form a sharply defined, adjustable beam. A* ***refractive lens****, on the other hand, allows for precise manipulation of light convergence and divergence by altering the optical path through changes in refractive index. This enables the user to fine-tune the focus position and beam diameter, producing a crisp edge and uniform intensity distribution across the illuminated field. Unlike a reflector, a lens provides a continuously variable focus range simply by changing the axial distance between the lens and the light source or gate plane. For the S.T.A.R. system, this flexibility is crucial— it allows the same optical assembly to adapt to different stage sizes and performer distances without mechanical reconfiguration of the reflector. Additionally, lenses can correct minor beam aberrations introduced by the reflector and improve overall image quality when projecting through gobos or irises, making them indispensable for applications that demand precision and visual consistency."*

[5a] ***Prompt:*** *"What does magnification do for a lens design?"*
[5b] ***Response:*** *"In a lens design, magnification describes how the size of the image formed on the sensor relates to the size of the object being viewed. For an imaging system like one capturing a theater stage, magnification tells you how much smaller the stage appears when projected onto a camera sensor. It is defined as the ratio of image size to object size, often expressed as* $M = \frac{image\ size}{object\ size}$*, and is typically negative for real inverted images. When the magnification is less than one in magnitude — that is, when* $|M| < 1$ *— the image is a demagnified version of the scene. The magnification also determines the field of view (FOV): a smaller magnification allows more of the scene to fit within the frame, while a larger magnification narrows the FOV and zooms in on smaller details. In practical terms, magnification bridges the geometry of the object, the sensor, and the focal length — guiding how lenses are selected and spaced to ensure that the desired area of the stage is imaged sharply onto the sensor.*

[6a] ***Prompt:*** *"Can you elaborate on what demagnification indicates?"*
[6b] ***Response:*** *"Demagnification is the process by which a lens system forms an image that is smaller than the actual object being viewed. In optical design, it refers to a* ***magnification magnitude less than one (|M| < 1)****, meaning the image on the sensor is a scaled-down version of the original scene. This occurs when the* ***object distance*** *(the distance from the lens to the scene) is much greater than the* ***image distance*** *(the distance from the lens to the sensor). In stage or theater imaging, demagnification is essential because a very large object — such as a 25-foot-wide stage — must be captured onto a small camera sensor only a few millimeters across. To achieve this, lenses are chosen or combined to create a* ***short effective focal length****, which allows the system to project a wide field of view while keeping the image sharply focused on the sensor. The degree of demagnification directly controls how much of the scene fits within the image: stronger demagnification means a wider portion of the stage is visible. However, designing for extreme demagnification introduces challenges, such as maintaining image sharpness*

across the field and controlling aberrations, since rays from wide angles must still converge precisely onto the small sensor plane.

[7a] ***Prompt:*** *"What type of connector would you recommend specifically for 25A at 24V—are there any brands or models you suggest?"*
[7b] ***Response:*** *"For a 25A, 24V application, you'll want to use connectors that are specifically rated for high current. Here are a few types and brands that are commonly recommended:*

*1. **Anderson Power Products**: Their Powerpole connectors are popular for high-current applications. They are modular and can handle up to 45A, making them suitable for your needs.*

*2. **Amphenol**: Known for their robust industrial connectors, Amphenol offers various models that can handle high current, such as the Amphe-Power series.*

*3. **Molex**: The Molex Mini-Fit Jr. series is another option, designed for high-current applications up to 9A per circuit, but they can be used in parallel configurations to handle higher currents.*

*4. **TE Connectivity**: Their Dynamic Series connectors are designed for high-current applications and offer reliable performance.*

*5. **WAGO**: WAGO's lever-operated connectors are easy to use and can handle high currents, suitable for quick and secure connections.*

When selecting a connector, ensure it is rated for at least 25A and is compatible with your wire gauge. Additionally, consider factors like ease of use, environmental resistance, and mechanical durability based on your specific application needs."

[8a] ***Prompt:*** *"How do I calculate the required PCB trace width for 25A current at 24V, and are there any online tools you recommend for this?"*
[8b] ***Response:*** *"Calculating the required PCB trace width for a specific current involves considering factors such as the current carrying capacity, temperature rise, and the thickness of the copper. The IPC-2221 standard provides guidelines for trace width calculations. Here's a basic approach:*

Calculation Steps:
*1. **Determine Copper Thickness**: Common copper thicknesses are 1 oz/ft² (35 µm) and 2 oz/ft² (70 µm).*

*2. **Temperature Rise**: Decide on an acceptable temperature rise, typically 10°C to 20°C above ambient.*

*3. **Use IPC-2221 Formula**: The IPC-2221 standard provides formulas to calculate trace width based on current, copper thickness, and temperature rise.*

Online Tools:
Several online calculators can help you determine the trace width:

*1. **Saturn PCB Design Toolkit**: A popular tool that provides comprehensive calculations for trace width, impedance, and more.*

*2. **Advanced Circuits Trace Width Calculator**: Offers a simple interface to input current, copper thickness, and temperature rise to get the trace width.*

*3. **EasyEDA Trace Width Calculator**: An easy-to-use tool for quick calculations based on IPC standards.*

Example Calculation:
For a 1 oz copper thickness and a 10°C temperature rise, you might find that a trace width of approximately 200 mils (5 mm) is required for 25A. However, this can vary based on specific conditions and standards.

Using these tools and guidelines, you can accurately calculate the trace width needed for your PCB design to safely handle 25A at 24V."

[9a] ***Prompt:*** *"how large would the PCB traces be to handle 25A and 8A"*
[9b] ***Response:*** *"To determine the appropriate PCB trace width for handling specific currents, you can use the IPC-2221 standard as a guideline. This standard provides formulas and tables for calculating trace width based on current, copper thickness, and allowable temperature rise. Here's a general approach:*

Factors to Consider:
*1. **Copper Thickness**: Common copper thicknesses are 1 oz/ft² (35 µm) and 2 oz/ft² (70 µm).*
*2. **Temperature Rise**: Decide on an acceptable temperature rise, typically 10°C to 20°C above ambient.*

Calculating Trace Width:
For a rough estimate, you can use online calculators or the IPC-2221 formula. Here are typical values for 1 oz copper thickness:

*1. **25A Current**:*
  *- **External Layers**: Approximately 400 mils (10 mm) for a 10°C temperature rise.*
  *- **Internal Layers**: Approximately 800 mils (20 mm) for a 10°C temperature rise.*

*2. **8A Current**:*
  *- **External Layers**: Approximately 100 mils (2.5 mm) for a 10°C temperature rise.*
  *- **Internal Layers**: Approximately 200 mils (5 mm) for a 10°C temperature rise.*

Online Tools:

*- **Saturn PCB Design Toolkit**: Offers comprehensive calculations for trace width, impedance, and more.*
*- **Advanced Circuits Trace Width Calculator**: Provides a simple interface to input current, copper thickness, and temperature rise to get the trace width.*

Considerations:
*- **Safety Margin**: It's often wise to use a slightly wider trace than calculated to provide a safety margin.*
*- **Thermal Management**: Ensure adequate cooling and consider using thicker copper if space is limited.*
*- **Testing**: Validate the design with thermal and electrical testing to ensure reliability.*

These calculations provide a starting point, but always verify with detailed design tools and testing to ensure your PCB can handle the required currents safely."