The Popcorn Button: Project Romulus

Jacob Cochran, Tryn Dunne, Andrew Morrison, Elijah Smith

Abstract

This paper outlines the development and creation of a handheld, automatic-aiming system for Airsoft. This was accomplished by creating a comfortable frame that supports a horizontally mounted 3-RPS robot that attaches to any Airsoft gun using a picatinny rail. A Raspberry Pi running stereo vision combined with an image recognition model allows the system to detect and aim at archery targets using inverse kinematics. The system was able to be deployed remotely and consistently aim within 1 inch of the center of the target at 25 feet away.

Table of Contents

Background

Recreational Airsoft is a sport and community shared throughout the world. Airsoft has deep hobbyist and DIY culture. There is a plethora of open-source modifications that enthusiasts have created to improve performance, such as changes to the motors, various sights and attachments, modifications of triggers, and many more. This has also led to several startups and its own small industry. Despite the many attachments and redesigns that these DIY enthusiasts have created, there has yet to be an attachment or modification dedicated to significantly improving the aim and stability of existing recreational Airsoft guns. Airsoft demands high hand-eye coordination and reaction times. For those with disabilities it could be impossible to compete in Airsoft. If a system could be designed to aim for the user, Airsoft could be much more accessible to a large group of people.

Some hobbyists have demonstrated aim-assistance platforms in both Nerf and Airsoft [1], [2]. However, both systems required a complete redesign of the underlying weapon. Therefore, these systems are a platform in and of themselves. The cost of Airsoft weapons can be prohibitive; therefore, it is desirable to create a removable mounting system capable of interfacing with existing Airsoft weapons. To this end, we designed Romulus as a complete tracking and positioning system designed to interface with existing Airsoft guns that helps recreational Airsoft players increase their accuracy. Romulus is designed as an attachment system into which users can attach their existing weapons, therefore, to provide maximum compatibility the mounting interfaces must be standardized. Fortunately, there are two predominant standardized attachment mounting systems to most Airsoft weapons conform: the picatinny rail system and the M-LOK system. Therefore, Romulus is designed to interface with these standardized mounts, with an emphasis on the picatinny rail system due to its prevalence among popular and medium-cost Airsoft replicas.

Romulus is a system that features both complex hardware and software. With each member of the group being a computer engineer and most also being an electrical engineer we were uniquely qualified to take on this challenge. Jacob, having the most experience with research and development, took over the design of the robot arm and any programming necessary. Tryn with his experience in the FUN series and AI hardware course oversaw the circuit board and helped Elijah with the AI training. Andrew spearheaded the frame, overall assembly, and hardware debugging, due to his troubleshooting experience over the summer. Together we had all the necessary skills to design Romulus.

Description of Project

I. Performance Objectives

Romulus is made to assist users with aiming an Airsoft gun. To facilitate this, it must be able to find the intended target and lock onto it quickly. Romulus must be accurate, ideally it should be able to hit the bullseye of the target every time it is used. Airsoft isn't a short game either, so Romulus must have a long battery life and hot swappable batteries to allow for long gameplay periods. Comfort is another important factor; Romulus must be comfortable to hold for dozens of minutes up to hours. Above all Romulus must be reliable, in a fast-paced game of Airsoft, Romulus will be put under stress in many ways, and it must meet all the above objectives without failure.

II. Technical Overview

Romulus is, at its core, a handheld robotic device used to point and track a target in space. A complete render of final design is shown in Figure 1, as well as our physical implementation in Figure 2 and Figure 3. The user holds the handle and base-platform. Then, a moving platform, connected to the base platform by linear actuators, is repositioned to point the moving platform at our target-of-interest. The two platforms are joined by revolute and spherical joints, allowing vertical and horizontal rotation, as well as inward and outward translation. In its primary application, the target of interest is an archery-style bullseye target, and the moving platform is designed to hold an Airsoft gun. However, most testing was done with a laser-pointer in lieu of an Airsoft gun.

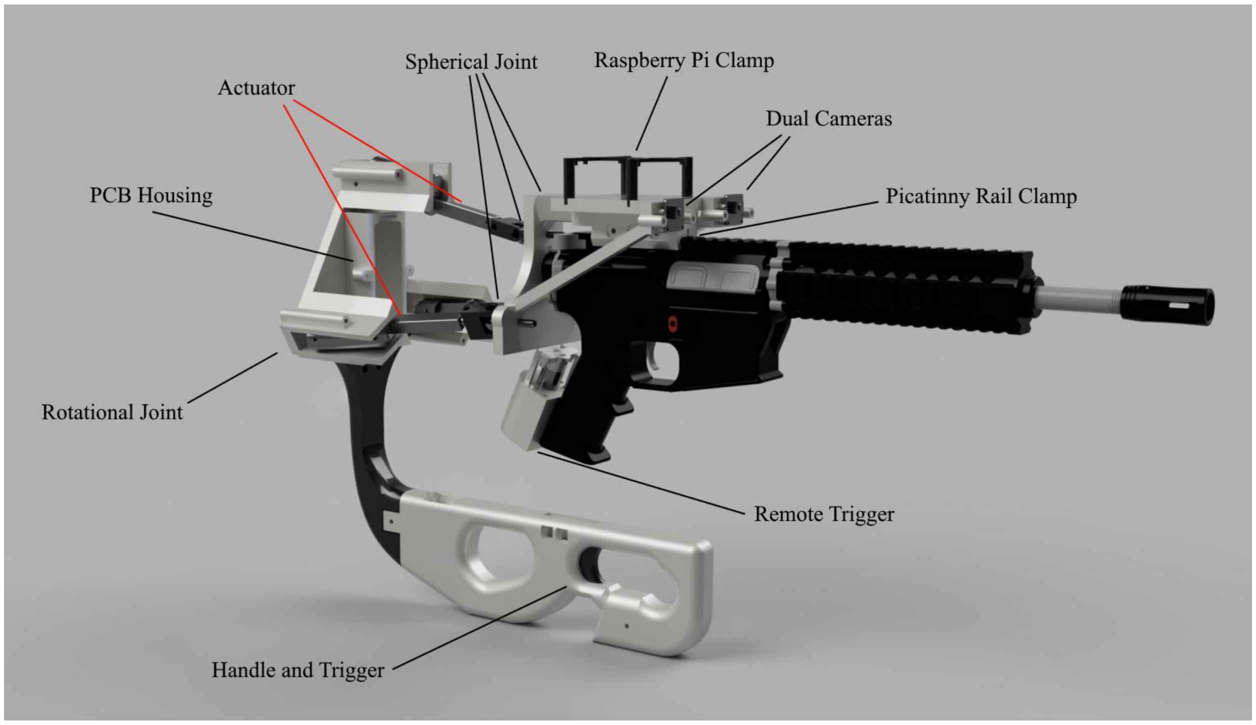


Figure 1. Full 3D-Render of Romulus System with Airsoft Gun Attached. Annotated are the main components and subsystems.

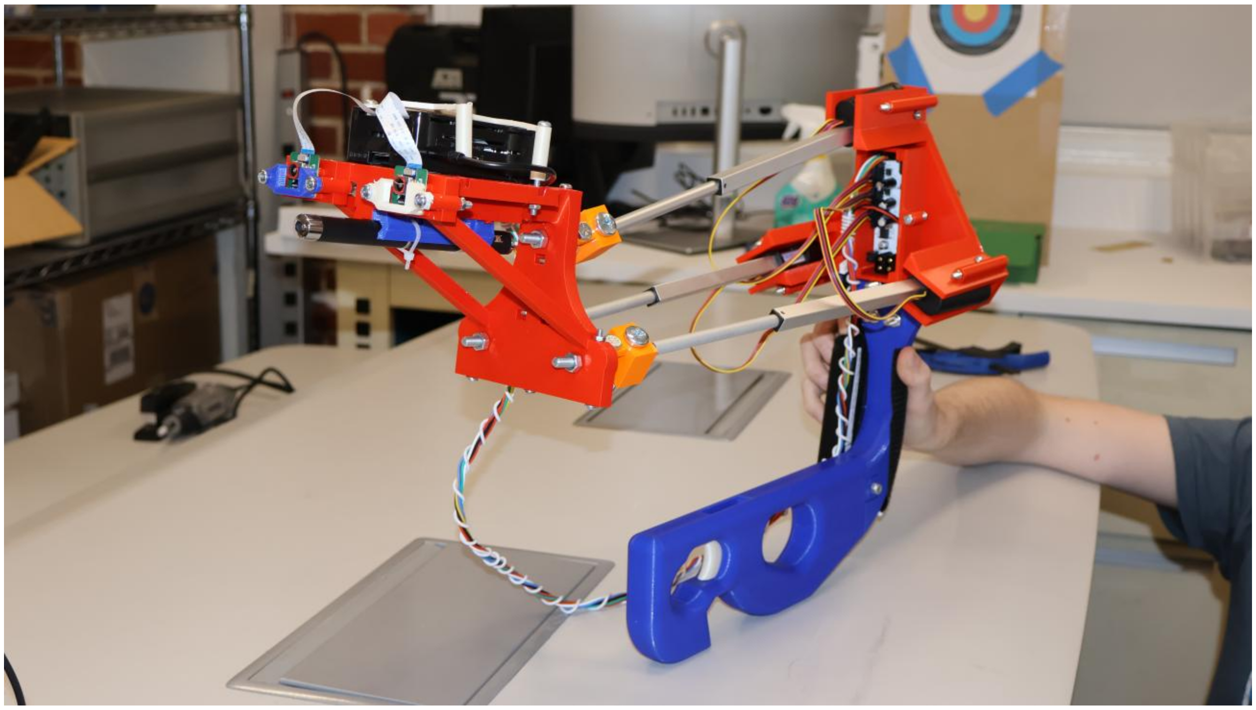


Figure 2. Image of completed Romulus Prototype. Note the addition of a physical cord connecting the PCB system to the Raspberry Pi. Also shown are batteries with Velcro connections to the holder.

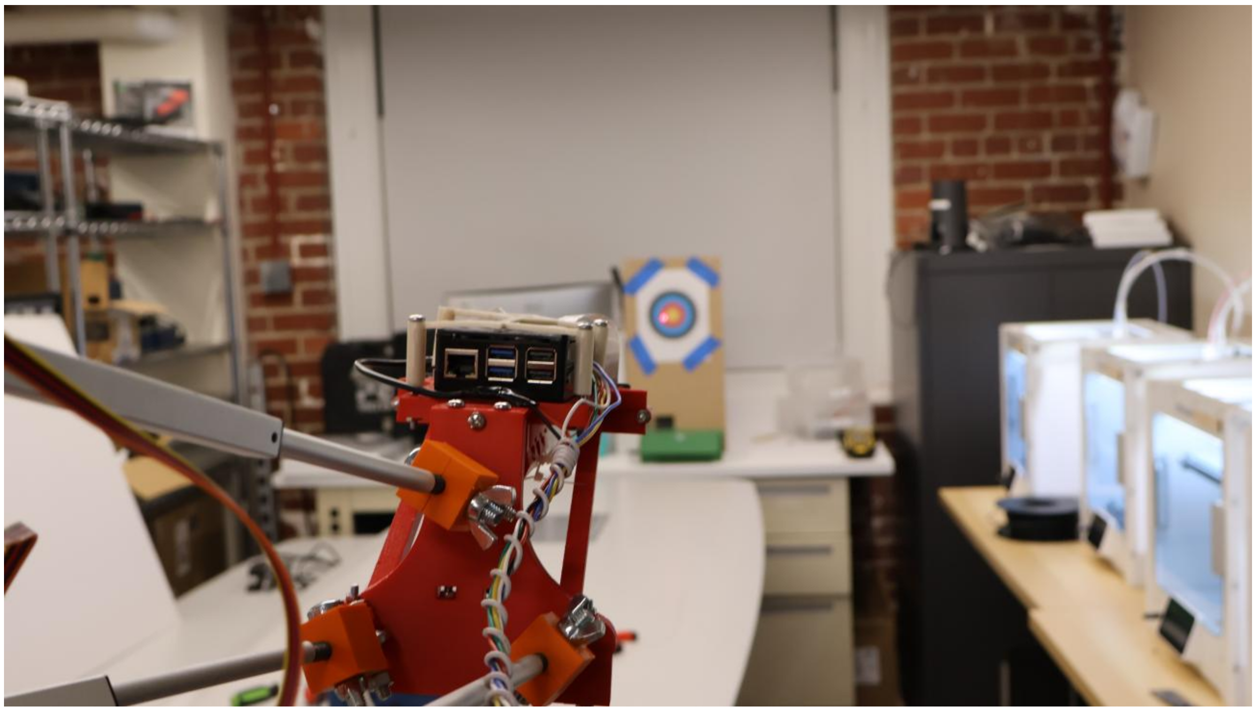


Figure 3. Image of the functional Romulus system from an over-the-shoulder point-of-view. Shown is a laser pointer on the target as it tracks.

There are several primary electronic and software systems used to realize Romulus. At the beginning of signal flow, information begins on the Raspberry Pi5 and the moving platform with a pair of stereo cameras. These cameras continuously capture input frames which are fed into an object detection model. This object detector then identifies the relative position of objects of interest (primarily archery targets) in the image. Bounding boxes are extracted from these detections which yield their relative position as pixels. However, to convert these relative pixels to a real-world angle, the physical distance of the object is needed. This depth perception is done using stereovision. That is, two cameras simultaneously capture the same image from shifted perspectives. By using the differences, or *disparity*, between the two images, we can perceive depth. In essence, it’s identical to human depth perception.

Having captured target depth and coordinates in camera pixels, simple trigonometry lets us determine the true cartesian coordinates of the target relative to the left camera. From this, the ideal upward angle (tilt) and horizontal angle (pan) required to adjust aim are calculated. This then becomes the reference signal for a digital controller. In this implementation, simple proportional-integral-derivative (PID) control was used. The output of this PID controller is an instantaneous tilt and pan control output. These control outputs are then processed through analytical inverse kinematics, converting a desired platform orientation angle and extension to linear actuator extensions.

These extensions are then transferred over SPI to our PCB, which essentially is a digital servomotor system. Our PCB consists of buck converters for battery power regulation and down conversion, three H-Bridge motor controllers, and an STM32 microcontroller. Each of the three servomotor systems is implemented as an independent proportional-derivative (PD) controller. The three actuator extension values received over SPI are the target reference position. The actuators have a potentiometer built into their stroke that outputs a voltage proportional to how far the actuator is extended. This position feedback is used by the STM, which sets an output speed and direction for each actuator's H-Bridge that realizes these speed and direction values in each actuator.

Of course, as each actuator moves the end-effector platform is spatially reoriented. This changes the relative angle perceived between the camera and target, requiring a new orientation angle to be dynamically updated. Because the angles calculated from the camera are relative to the end-effector's current orientation rather than the absolute orientation relative to the base, an estimator of the current absolute angle is desirable. To measure this, during each SPI exchange the current actuator positions are exchanged with the Pi in full duplex. These actuator lengths can be used to calculate the orientation and position of the moving platform with forward-kinematics.

III. Engineering Design

The Parallel Robot

The heart of Romulus is the robotics used to move the reorient the end-effector platform to point at its desired target. While we initially considered creating a custom architecture, background research indicated such a design would likely be difficult to analytically model, control, and would likely have instabilities [1] [2]. Typically, mechanisms are described by their degrees of freedom, that is the number of ways in which the joint can move unconstrained [5]. Three translational (3T) and three rotational (3R) degrees of freedom are needed to fully traverse space. For our application, we required at least two degrees of rotational freedom. The ability to rotate vertically (tilt) and horizontally (pan).

Robotic architectures are described as series, parallel, or hybrid mechanisms. A *link* is any rigid body, which are connected by *joints*, forming a *kinetic chain*; if multiple paths exist between links, then the chain is a *closed loop* [6]. A *parallel* mechanism consists of many parallel chains between links, while a *serial* mechanism is purely *open loop* [7]. While both parallel and serial mechanisms can achieve the required degrees of freedom in our system design, each offers unique advantages and disadvantages. Typically, serial robotic configurations are well suited when a large workspace is desired, as well as a simpler analytical analysis. However, these serial manipulators typically incur significant cost in weight, and low stiffness (which combined with serial structure) tends to propagate positioning error throughout the system [5], [6]. Meanwhile, parallel mechanisms tend to be best suited to compact, accurate, and high-speed environments [6], [7], [8]. Overall, the advantages prescribed to parallel manipulators were well suited to our use case.

While we only nominally required two rotational degrees of freedom, we were worried about structural integrity in a system with only two points of contact, especially due to the weight and torque applied by an Airsoft gun. Additionally, accurate control may have proven difficult. The forward and inverse kinematics, dynamics, and stability of parallel configurations are often complex derivations, lacking analytically closed-form solutions [6]. Additional points of instability may also exist within the workspace, known as singularities, which are difficult to predict [3], [6]. There are several types of singularities, but typically these are points in the workspace at which a degree of freedom can be instantaneously lost. To model this alone could have proven time intensive. Therefore, an important design criterion in our selection was that the architecture be well-studied, preferably with parametric design criteria that would allow us to accurately to predict the size of its safe workspace, that is without singularities or other instabilities.

An excellent candidate proved to be the 3-RPS configuration, which provides one translational and two rotational degrees of freedom [9] See Figure 4. Significant research has been dedicated to the design of 3-RPS systems to optimize workspaces for singularity avoidance [10], [11]. This robot design consists of two parallel platforms connected by three independent chains of *revolute*, *prismatic*, and *spherical* joints. Revolute joints allow a single axis of rotation, prismatic extend forwards in single translational direction, and spherical allow all three angles of rotation.

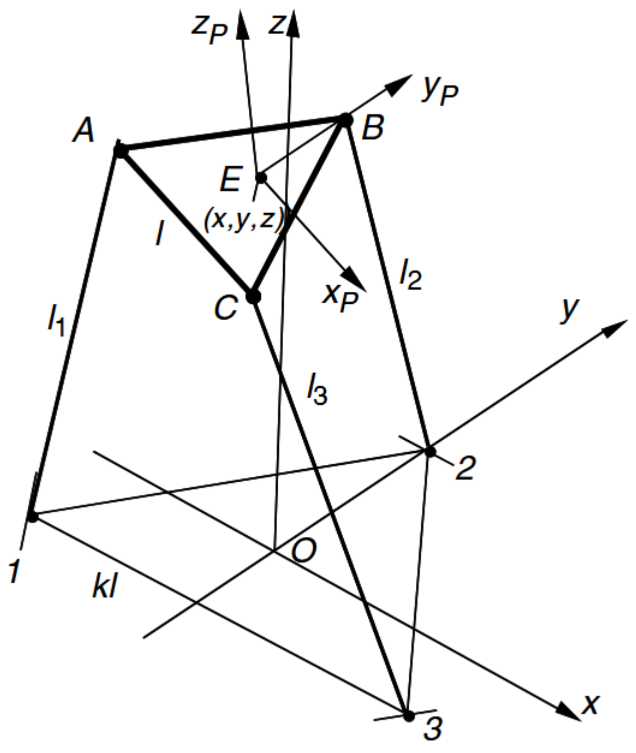


Figure 4. Basic 3-RPS Architecture. The bottom plane is fixed, consisting of three revolute joints labeled 1, 2, and 3. The top platform, also called the moving platform or end-effector, consists of spherical joints at A, B, and C. Joining the two are three linear actuators (prismatic joints), labeled $\boldsymbol{l_1}$, $\boldsymbol{l_2}$, and $\boldsymbol{l_3}$. Tilt is defined as rotation of $\boldsymbol{Z_p}$ along the fixed YZ plane, whereas pan is rotation of $\boldsymbol{Z_p}$ about the XZ plane. (Source: Alexei Sokolov).

There are several approaches to controlling a parallel robot, each of which is a tradeoff between design complexity, accuracy, and responsivity. For very-stiff end effectors in which the end-effort encounters little resistance, a simple positional control can be sufficient. In this system, a target reference position for the moving platform is set. Then inverse kinematics are calculated, in which the joint lengths l_1, l_2, and l_3 are determined for a desired orientation. For the 3-RPS mechanism, especially one with symmetry, the inverse kinematics has a closed form analytical solution [2],[7]. Forward kinematics describes the converse; that is, given some input lengths l_i, determining the position and orientation of the moving platform. However, forward kinematics typically does not have an analytical solution in parallel robots [6]. The forward kinematics for the 3-RPS architecture consists of a system of non-linear trigonometric equations, which is typically solved by numerical methods [8], [12].

Several implementations of inverse kinematics were performed in our project, including a custom inverse kinematics formulation based on the available research. The main difference between these representations being the coordinate systems used to relate the moving platform to the stationary base. For instance, Cartesian coordinates and ZYZ Euler angles were used in [6]. Cylindrical, joint, and a hybrid coordinate system reminiscent of spherical coordinates was used in [8]. Finally, for the main paper implemented in the final design, ZXY Euler angles were used [12]. In this coordinate system, the pan, tilt, and extension are the input unconstrained variables used to calculate the extension of each linear actuator, see Figure 4 through Figure 7. Another difference, however, which limits each of these inverse kinematics formulations is the effective domain of the inverse-kinematics function's definition [8]. While the formulation used did not provide a theoretical domain of definition, in practice the constraints imposed by physical joints proved to be the primary limiting factor on range of motion.

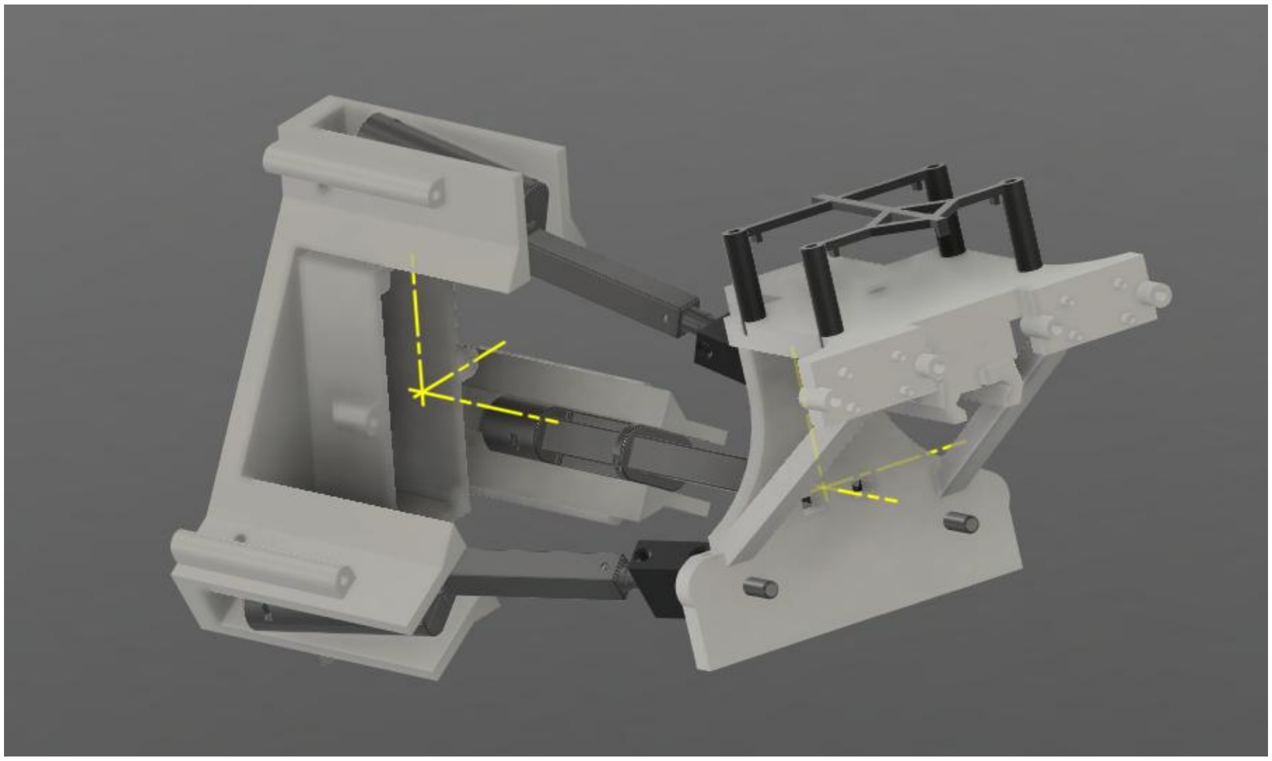


Figure 5. Isometric Rendering of 3-RPS Romulus Design. Shown as yellow lines are the two Cartesian coordinate systems (left) is the fixed coordinate system of the base, (right) is the moving coordinate system of the end-effector. The position is defined as the vector connecting their centers. The normal vector of the moving frame is defined as the z-component (pointing towards the reader). The orientation Euler angles are defined as the angles of this normal vector about each of the principle XYZ axes of the fixed base.

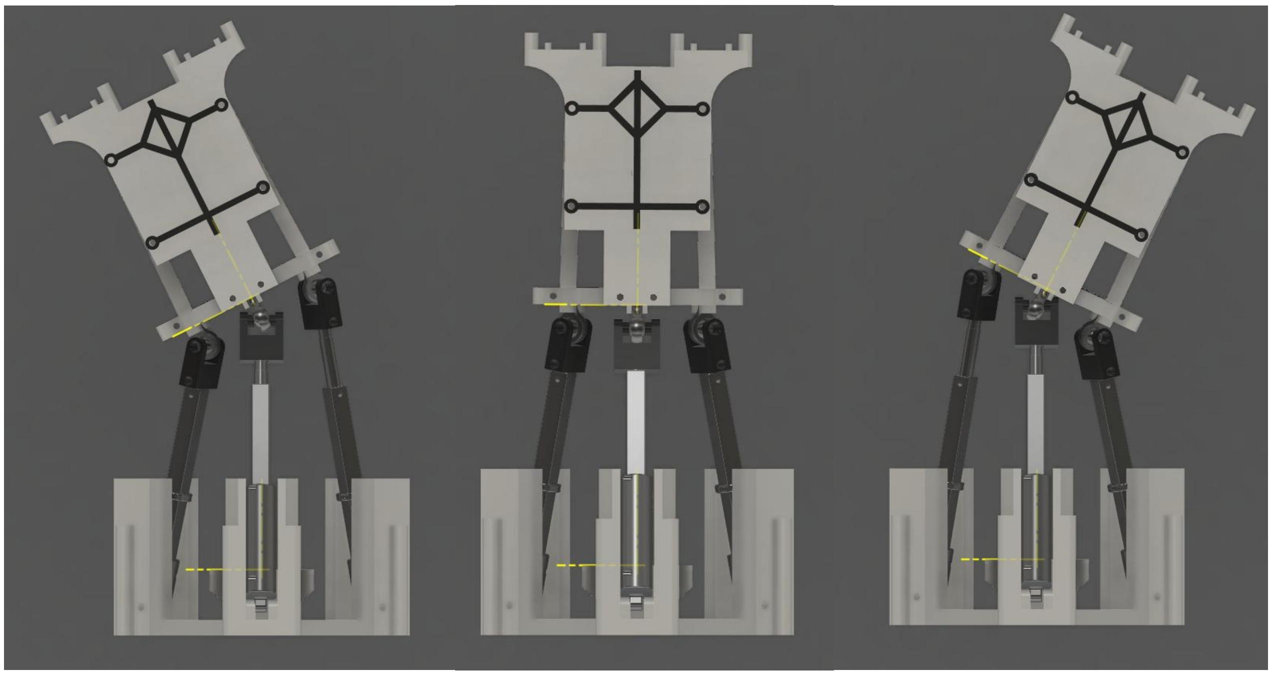


Figure 6. Render of Pure-Pan Motion in 3-RPS Romulus Design. Left is a negative pan, middle is the neutral position, while right is a positive pan value in the ZXY Euler-angle convention.

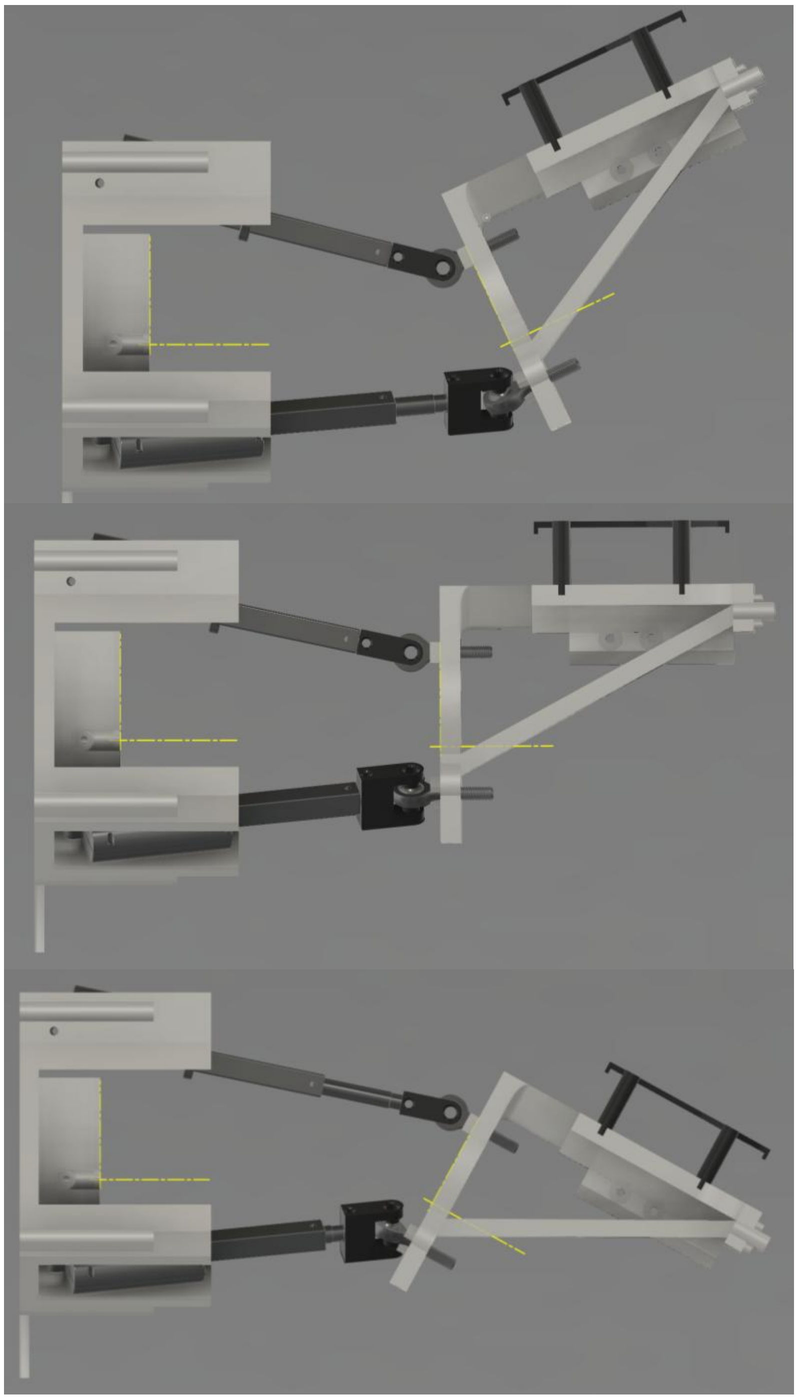


Figure 7. Render of Pure-Tilt Motion in 3-RPS Romulus Design. Top is a negative tilt, middle is the neutral position, while bottom is a positive tilt value in the ZXY Euler-angle convention.

Numerical methods of approximating the forward kinematics of the 3-RPS system were implemented to some degree of success. Several numerical solving techniques were employed, with the most successful being an iterative least-squares regression approach. However, it proved to be somewhat unreliable and susceptible to error. Therefore, in the final approach an empirical estimation approach was used instead. In this process, a massive grid of pan, tilt, and extension values are given as inputs to the system. After waiting an interval of time long enough to guarantee that the system reaches steady state (nonmoving), the extension of each linear actuator was measured via the microcontroller’s potentiometer reading transferred over SPI. These were then saved on a large, compressed, h5 file lookup table. At runtime given a set of measured lengths, this lookup table can then be searched to find the nearest corresponding saved actuator lengths. This was done in SciPy using a cKDTree which is a fast structured used to index and quickly find the nearest corresponding k points in a large set of data. To improve the resolution in angle, some interpolation, such as weighted average was done on the nearest k points, rather than simply extracting the closest stored value. However, the primary challenge with this approach became recording a reasonably high resolution of input angles, as recording several points required for moderate fidelity took several hours of continuous runtime. Additionally, there were concerns that the very non-linear relationship between angle and joint lengths would cause interpolation strategies to introduce additional error.

Stereovision and Depth

In our initial proposal, depth was to be measured using a laser rangefinder. This depth measurement was also initially only thought to be required for drop-correction. That is, to adjust the upward angle of the device to account for the Airsoft pellet's bullet drop as a function of distance. While this primary application is still equally valid in our final implementation, we also ultimately needed to range to accurately calculate the error in angle between the camera's current position and a detected target. To visualize why depth is required to analytically calculate relative rotation of the end-effector, consider the derivation in Figure 8.

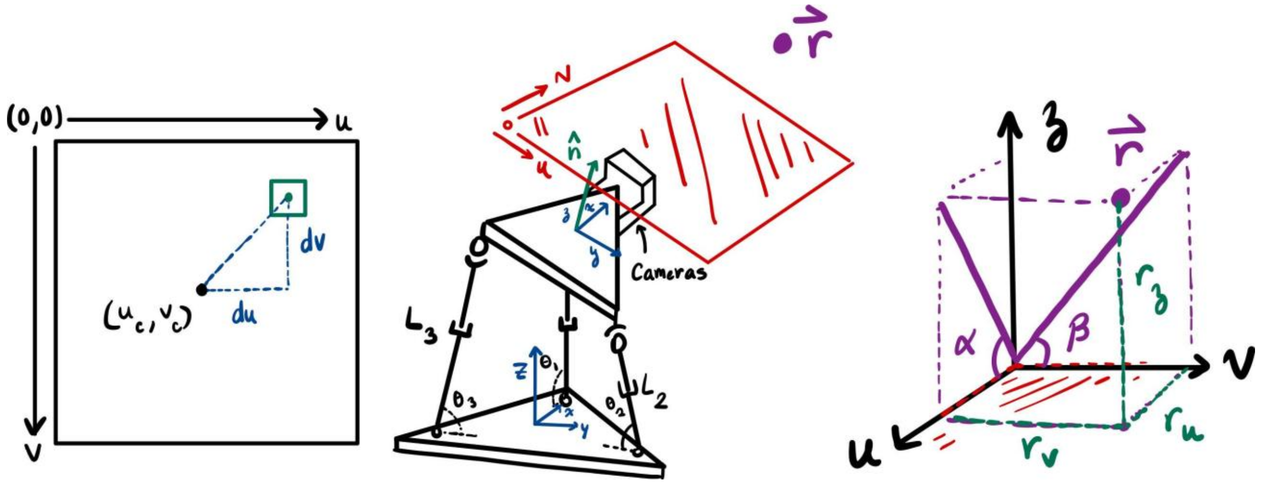


Figure 8. Simplified derivation of trigonometry used to translate from camera coordinates to *relative* control angles, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. Shown on the left is the detection of an object relative to the camera center's center in an image. In the middle is the relative position of the camera on the 3-RPS system, the challenge is given a projected image of some object of interest in 3D-space $\vec{\boldsymbol{r}}$, what relative angles $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ are needed to rotate the moving frame's normal vector $\hat{\boldsymbol{n}}$ to align with the target. The right image shows the camera image plane (the uv plane), extended into 3D space. From this extension, if the distance is known, trigonometry can be used to find these angles.

From the derivation, finding the relative pan and tilt angles is simple trigonometry if the Cartesian position of the object relative to the cameras is known. Some additional geometric correction is needed however, as inverse-kinematics is performed relative to the center of the moving platform, but these angles are relative to the camera's lens. However, if Cartesian coordinates are found for the camera, these corrections are simple corrections to the lengths of r_v, r_u, r_z.

$$\tan(\alpha) = \frac{r_z}{r_v} \qquad \tan(\beta) = \frac{r_z}{r_u}$$

However, a single camera on its own cannot measure distance accurately. If the size of a target is fixed, the distance can be approximated; however, our implementation goals required *Romulus* to function with targets of arbitrary size. Another approach is to use the focus of an autofocusing camera, intrinsic camera parameters, and optics to estimate distance. However, this implementation yielded very poor estimates of range, while only working over a minimal range of distances. The approach we ultimately decided on, and the approach we all use in our day-to-day lives, is that of stereovision.

By including an additional camera spaced horizontally apart from the first (Figure 10) distance can be estimated. Both cameras will see the same object within the frame, however, because of their physical offset, they will perceive the same object differently. This in practice can be done with any two cameras positioned arbitrarily if both see the same object [13]. However, in the ideal case, the two cameras are perfectly identical and aligned exactly so that they are perfectly horizontally separated and aligned in every other direction [13] (see Figure 10). In this case, an object in one camera will be only horizontally translated in the other camera's perception, as shown in

Figure 9.

Then, the disparity in pixels, d, is related to the distance Z by both the focal length f of the cameras and baseline B separating the cameras by the following,

$$d = x - x' = \frac{Bf}{Z}$$

Where the focal length, f, is an intrinsic property of each camera that ideally should not depend on the camera's focus. We are using the IMX-519 camera from Arducam, which was rated to have a nominal focus at 4.28 mm [14], however, because it is mechanically-focused the effective focal length varies based on its current depth-of-focus.

A *very* primitive first pass at a solution was to extract bounding box x-coordinates from each model detection and use those to estimate disparity.

In practice, even a well-designed stereo camera rig will be imperfect. The cameras may not be perfectly aligned and are certainly non-identical. The biggest driver of error however is distortion added by each camera's lens, which varies even more widely between cameras. These seemingly minute imperfections have substantial effects on the measured results.

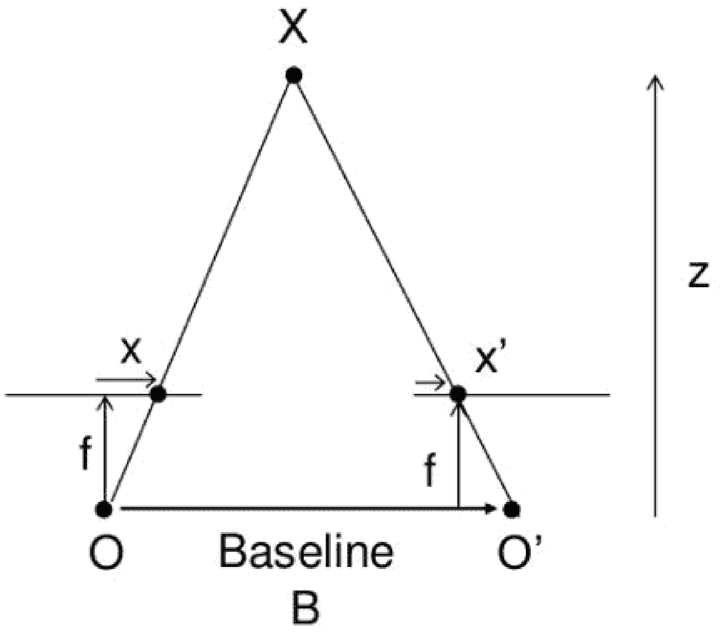


Figure 9. Ideal Stereo camera configuration with two cameras separated by a perfect horizontal separation. The baseline, $\boldsymbol{B}$, is the distance between the two cameras, $\boldsymbol{O}$ and $\boldsymbol{O'}$. The focal length $\boldsymbol{f}$ corresponds to the virtual plane in front of the camera for which each image is captured. The object, $\boldsymbol{X}$, is observed by both cameras as $\boldsymbol{x}$ pixels in $\boldsymbol{O}$ and $\boldsymbol{x'}$ pixels in $\boldsymbol{O'}$. The distance $\boldsymbol{z}$ is defined as the distance between the baseline and object. (Source OpenCV).

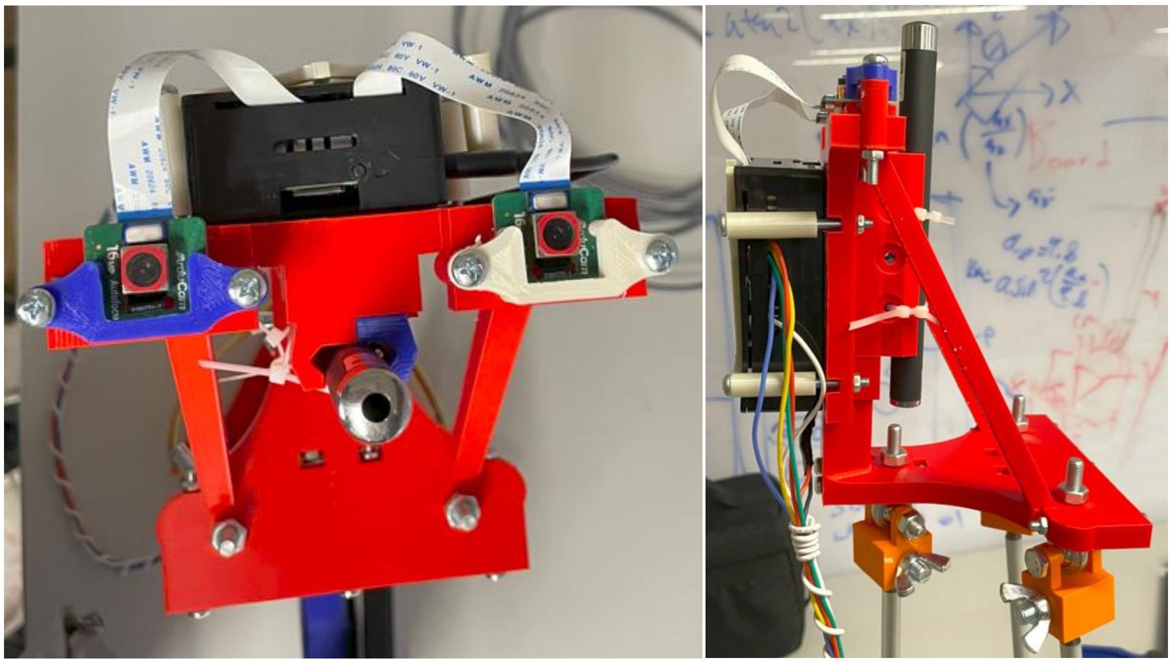


Figure 10. Top-down and profile views of Romulus' end effector. Two cables connect both cameras to the Raspberry Pi5. A laser pointer is zip tied into the picatinny rail for testing purposes. Note that both cameras are positioned on the same horizontal plane. The baseline is defined as the invisible line connecting the center of both cameras, which is approximately the width of the Raspberry Pi in the above image.

Therefore, the first step to effectively measure distance is to account for these imperfections. This process is referred to as stereo rectification [15]. Essentially, in stereo rectification the distortion and non-idealities of the stereo rig are measured through calibration against a known metric. The goal is to convert input images with distortion and imperfect alignment and convert them to a pair of idealized stereo images. The first step in calibration measures the distortion and effective focal length of each camera individually by taking several images of checkerboard patterns in a grid with the size and number of edges known [16], see Figure 11.

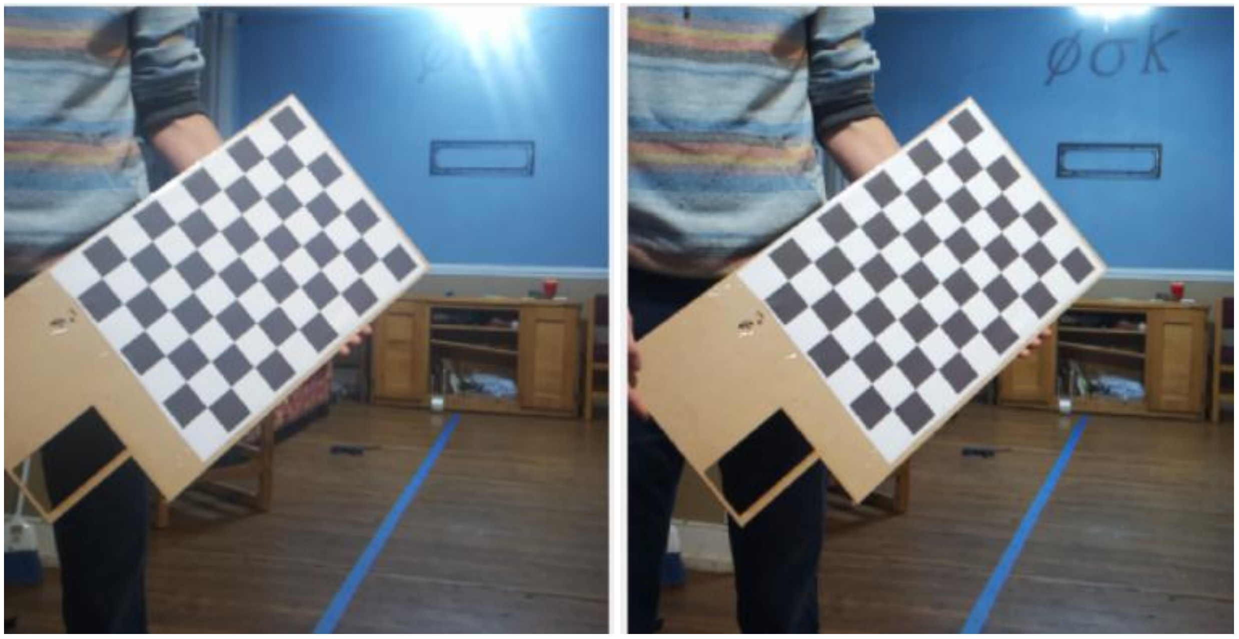


Figure 11. Calibration data taken prior to performing stereo rectification. Note that there is a slight perspective skew between the two.

The next step is to rectify the cameras against each other, and several approaches exist. The high-level idea of rectification is to find the transformation map which would transform one camera's image to be identical to the other [17]. The method used in our implementation was built on the calibration and stereo-rectification in [17], [18]. These transformations between cameras are referred to as *rectification maps*, which are used to transform the left and right camera feed to an idealized pair of stereo images, as shown by the connecting lines in Figure 16.

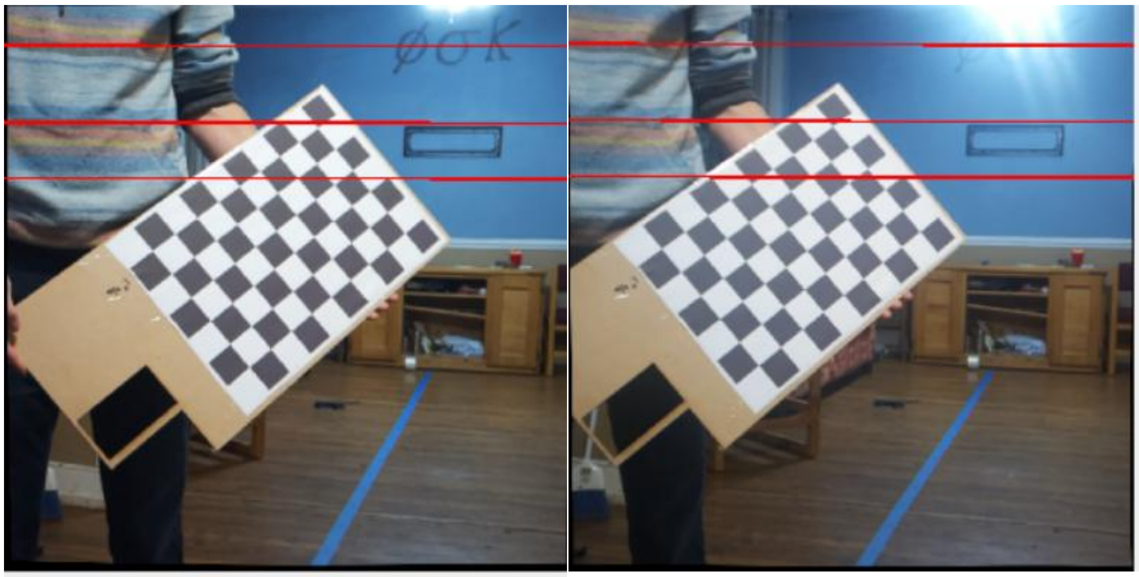


Figure 12. Computed connecting lines showing that the calibration was able to correctly rectify both images. Note the skew has been removed and the connecting lines are more horizontal. Also, the black lines on the bottom show how the image was slightly rotated to match the two.

Having rectified the left and right images, the disparity d can then be computed as the pointwise difference, in pixels, of an object in the left and right rectified images. That is, the disparity d in x and x' for each object in space. In theory, this is enough. By running our image classification model on both cameras, applying the transformation map, and measuring the disparity, distance can be estimated. However, in practice this yields mediocre estimates with very poor resolution in the distance that can be measured. That is, we could only really measure accurately to only about a foot. First, object detection, especially in early iterations, did not give the *exact* same pixel coordinate for the detection run on each camera. After moving far enough away from both cameras' poor resolution in the model made the disparity quickly trend towards zero (infinite distance). The other main reason is that pixels are quantized, so there is a fundamental limit on how many pixels will differ for small changes in distance.

To address both limitations, the commonly used approach is a block-matching algorithm. The purpose of a block-matching algorithm is to scan horizontally along both the left and right rectified images to find corresponding groups of pixels, and thus features, between them [15]. As groups of pixels are compared, there is an inherent smoothing that allows the effective disparity resolution to be increased beyond single pixels, which is implemented in OpenCV's block-matching algorithms [15]. These block-matching algorithms can then be used to compute a disparity map which is a representation that shows the difference between blocks in the right camera relative to the left (dominant) camera. Distance for each pixel is then theoretically inversely proportional to the value of each pixel in the disparity map. This was shown by representing disparity using a color map in Figure 13.

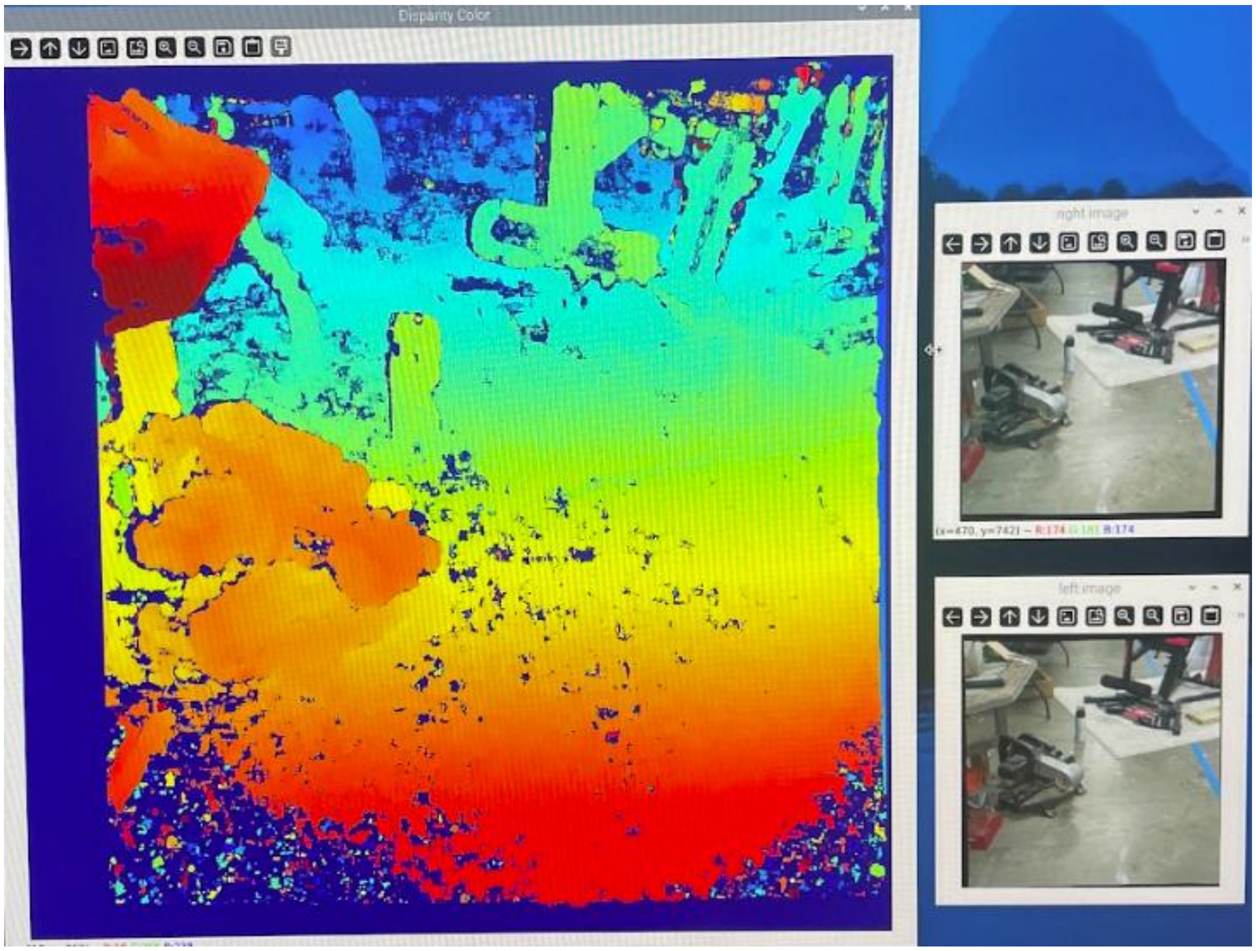


Figure 13. An example disparity map rendering for stereo camera setup used. Note the left and right camera images shown to the right. The disparity shows near objects in warmer red hue, and further objects in a cooler color.

Given a disparity map, extracting the corresponding distance can additionally be done in two primary ways, each with their own inherent tradeoffs. The first uses OpenCV's reprojectImageTo3D function to convert a depth map and calibrated camera parameters to 3D coordinates for each pixel in the left-camera. The downside is primarily computational expense, and therefore processing time. The upside is that no additional camera calibration is required. This method was also far more prone to errors in the initial set of calibration data and would sometimes take several attempts at calibration to function. Because changing the camera's focus entailed recalibration for each focus point, entailing a large amount of initial calibration to be performed and validated. Ultimately, we were only able to get this method working for some focal lengths. However, this method did yield *incredibly* accurate Cartesian coordinates for objects in real-world space. Horizontal and vertical positions were measured to be accurate within approximately a quarter of an inch up to around twenty-five feet.

The second approach entails an additional step of calibration after stereo rectification. In this method, an object is placed several known distances from the stereo cameras and the disparity is measured. Ideally, there is an inverse relationship between distance and disparity, therefore the relationship between depth and the reciprocal of disparity should be linear. Therefore, a linear regression was performed and saved for each calibrated focus value. An example measurement setup used is shown in Figure 14. The results for a single focal distance calibration are shown in Figure 15.

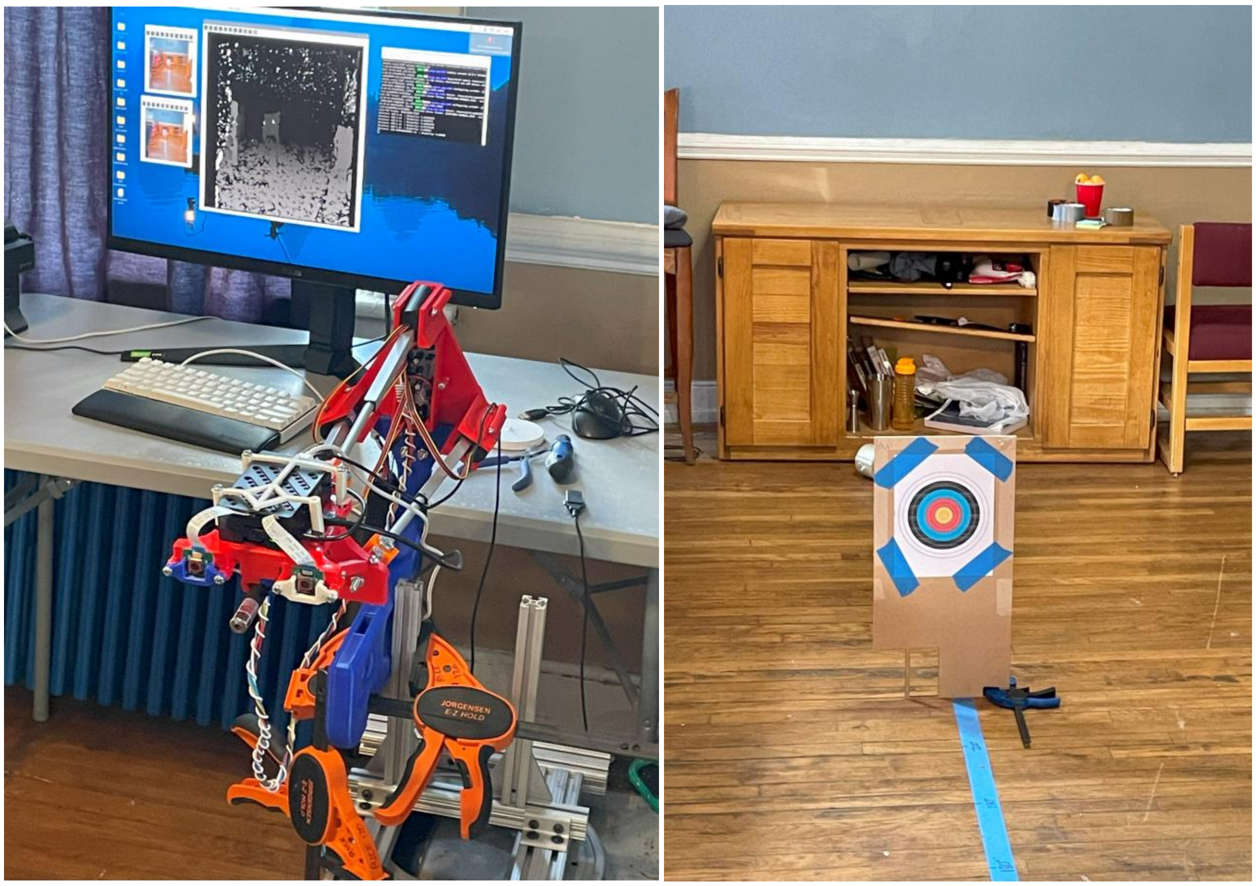


Figure 14. Example distance calibration and regression setup. Shown on the left is *Romulus* statically mounted to a base platform using clamps. Shown on the right is a target spaced some distance away. The blue tape is a track with 10-inch increments marked. The program in the top-left is a gray-scale depth map used to facilitate data collection. The target was moved several times along the track to generate points for the regression.

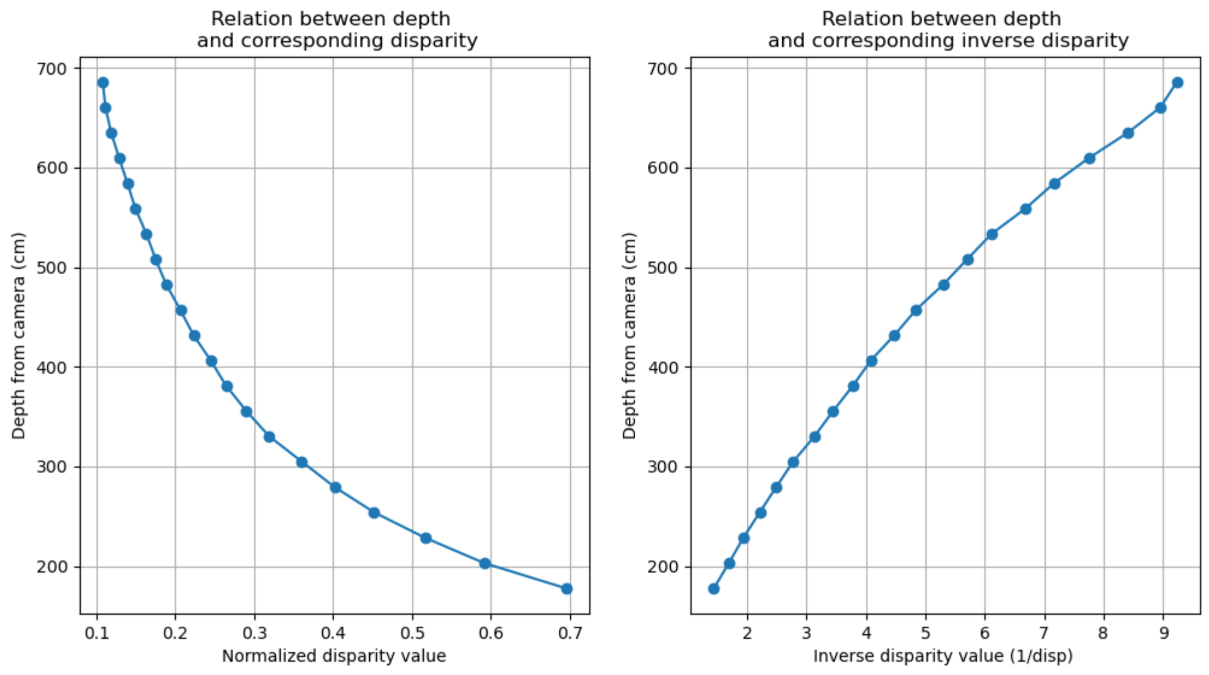


Figure 15. Example disparity-to-distance regression generated during testing. Note the mostly linear relationship between the reciprocal of disparity and distance observed on the right. The slope and intercept were then used for real-time distance estimation.

Additionally, for some poorer stereo rectification calibration, the observed relationship proved to be quasi-quadratic. While this indicated a poorer calibration, measurements taken showed that despite a poor initial calibration, the estimated distances were still quite accurate.

Object Detection

For the system to detect whether a visual input contained a desired target, a large dataset was obtained to train an object detection model. This dataset contained pictures of a colored archery target with various camera angles and lighting conditions to provide a good foundation for detection. The service our team used for acquiring and eventually training this dataset was Edge Impulse. This platform was chosen due to its ease of use and fast deployment.

After acquiring a dataset, bounding boxes were drawn and labels added for the model to correctly identify the edges of the targets. A split of 80% of images were used for training and 20% of images were used for testing.

Note that the images are taken with different backgrounds and distances away from the target. The bounding boxes are drawn such that they envelop the entirety of the target.

Initially, for the system to detect a target, concerns regarding computational efficiency and low latency were crucial for choosing an object detection model. For these reasons, a You Only Look Once (YOLO) model was initially chosen. Furthermore, YOLO as an object detection model has multiple different versions due to continuous iterations over the years since its inception, with each version containing various levels of size and complexity in their implementations. However, despite these benefits, problems were raised in our implementation of YOLO for Romulus. Firstly, Edge Impulse does not have a built-in YOLO model training system, which meant the requirement of community builds of these models to be used. Additionally, when training the YOLO model, the platform required a considerable amount of time to train and often would result in lower detection accuracy than desired. Finally, after finishing a training session, the resulting model would have high inference times upwards of 100 milliseconds for poor image detection resolution. Due to this, our team made the switch to use a different object detection model for our implementation.

The resulting decision was to use a Faster Objects, More Objects (FOMO) model. FOMO proved to be much faster, more accurate, and provided much lower inference times averaging 50 milliseconds compared to its YOLO counterpart. Using FOMO, multiple models were constructed to optimize the best use of image resolution and dataset information.

<u>Robotic Control</u>

The main question of the robotics control system boils down to this: *given a target, how should you move to get there?* Several approaches were attempted to various degrees of success throughout this project. In a purely open-loop control system, a target position is specified without any feedback over time. Ideally, such a system will stabilize at the desired target after some interval of time. In a feedback-based control system, the state of the system is continuously observed, and is ideally used to force the system to stabilize.

In the very first iteration of *Romulus*, a single camera and feedback control system was able to successfully track a target; however, without depth information would be unstable or slow at ranges of distance. In the ideal case, our tracking performance should be distance agnostic. That is, rotating upwards ten degrees at five feet should be identical to rotating ten degrees at twenty.

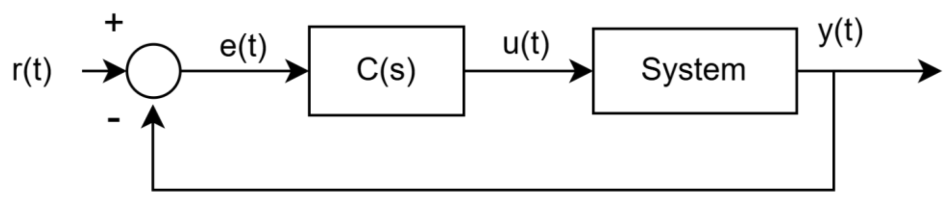
After successfully implementing distance-finding with stereovision and some trigonometry, we were indeed able to succeed here.

In fact, the distance-finding and coordinate extraction was precise enough that pure open-loop control was to some degree successfully implemented. In this control architecture, an interface was written that allowed a user to click an arbitrary point on the camera and the robot would rotate to point at that position based solely on the initial reference coordinates generated from that single reading. However, pure open-loop control presents three design challenges: (1) what happens when the target is moving, (2) how to correct for measurement error, (3) how *long* will it take to stabilize before you can process new input?

These design challenges are where the concept of closed-loop feedback becomes important. One simple feedback system that can be added is to use forward-kinematics to solve the third challenge. Essentially, we can measure the time between the initial reference signal input and the robot reaching those target actuation angles. However, these first two challenges are not quite as simple. What if the initial depth map reading had small measurement error? Well then, we'd never stabilize on the target. To make things more complicated, what if the target is moving from side to side? In this case, the robot would be guaranteed to lag towards a moving target. New input wouldn't be allowed until it stabilized at the previous location.

So that begs the question, how can we continuously take in new camera input while still tracking a moving target efficiently? To do this, we utilized the relatively simple but effective feedback control system of proportional-integral-derivative (PID) controls, see Figure 16. In this type of controller an error signal $e(t)$ is determined based on the current output $y(t)$ and a reference input or setpoint, $r(t)$. First, a reference signal is first applied to a process. In this case, the reference signal is the center of the camera frame for all time, which was represented as zero pan, zero tilt. The system output is the center of the bounding-box coordinates extracted from the object detector[1], calculated using distance and trigonometry as the relative angle to center. The error is then the difference between these bounding boxes and center frame. The complete block-level control implementation used is shown in Figure 16.

[1] Minor corrections are added to account for robot geometry since the center-camera does not correspond 1:1 with the true center for the end-effector mounted in the picatinny rail.



$$u = k_{\mathrm{P}} e + k_{\mathrm{I}} \int_0^t e(\tau)\mathrm{d}\tau + k_{\mathrm{D}} \frac{\mathrm{d}e}{\mathrm{d}t} \qquad C(s) = \frac{U(s)}{E(s)} = k_{\mathrm{P}} + \frac{k_{\mathrm{P}}}{T_{\mathrm{I}} s} + k_{\mathrm{P}} T_{\mathrm{D}} s.$$

Figure 16. Basic Block-level Topology of PID Control. C(s) represents the PID control signal as a function of error in the frequency domain. Shown on the bottom are the control output u(t) as a function of time, and the control output C(s) as a function of frequency. These are both continuous time implementations, while the digital PID controller was implemented in discrete time, however the design idea is identical.

The discrete-time implementation of PID controls is quite simple. The iteration time is simply the measured time between each object detection. The derivative is simply the current error minus the previous error divided by the iteration time. The integration is a rolling summation of the current error divided by iteration time. Finally, the proportional term is simply the current error. Each term is then multiplied its respective gain (K_p, K_i, K_d) and summed.

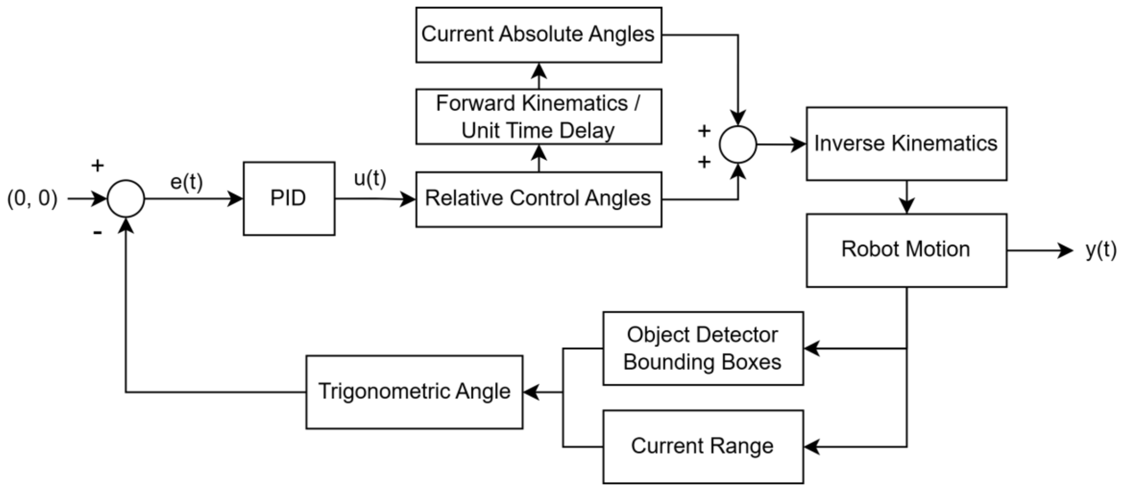


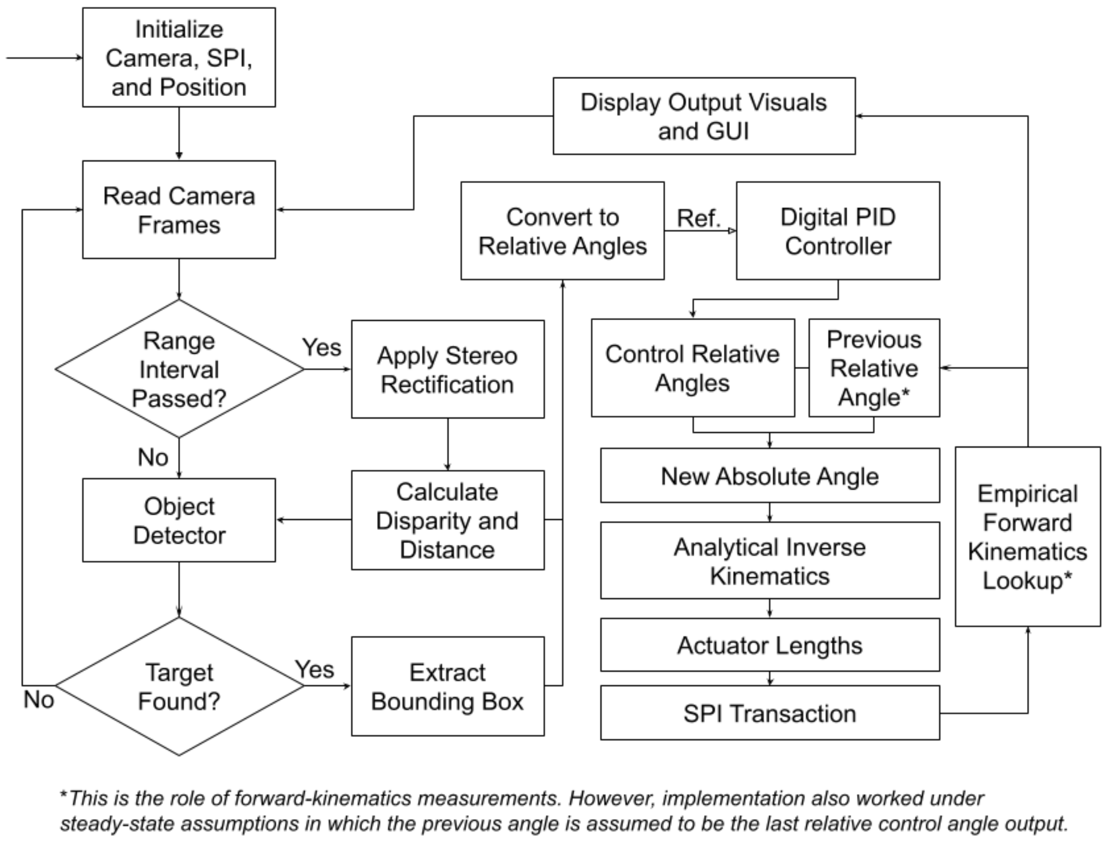
Figure 17. Block-level overview of Romulus' PID control implementation and feedback. Note that the current range is periodically calculated to save computational expenses, whereas object detection is performed with each control iteration. Additionally, for PID control the time per iteration is measured, with this being the time between successful detections (and therefore control output generations). Additionally, forward kinematics was only used in some of the final implementations, as the absolute angle can be assumed to be roughly equal to the current previous control output.

In practice, a separate PID controller was implemented for the robot's pan (horizontal camera axis), and its tilt (vertical camera axis). However, while the implementation of PID controls is quite simple, the main design challenge with PID controls is choosing the appropriate gain terms K_p, K_i, and K_d as they determine the stability and tracking speed of the system. This process is called tuning and is typically done through empirical measurement.

To facilitate the tuning process, an interface was first created. In the experimental setup, *Romulus* was fixed in place with a laser-pointer mounted in its end-effector. This was typically done using a similar setup as shown in Figure 18. To tune the pan, the bullseye is then moved to some horizontal displacement from the home position (zero absolute pan, zero absolute tilt). To tune the tilt, an initial vertical displacement was chosen instead. Once both were independently tuned, tweaks were made to the overall tuning by introducing both simultaneously with both initial vertical and horizontal error. The interface was written to have sliders for each PID constant. Once a PID gain is changed, the robot resets home position, waits, then begins tracking the target under these new controller gains. To evaluate the performance, we considered if it stabilized, the time to both marginally and fully stabilize, and the overshoot observed.

In practice, we used the following tuning heuristic approach. First all other terms are nulled, and the proportional gain is increased to its maximum stability point. Then, integral gain is slowly introduced while slightly backing off proportional gain. This should allow the controller to stabilize without substantially overshooting. Finally derivative gain is introduced along with integral gain to increase tracking speed while maintaining stability. This process is very iterative, however, tends to yield reasonable tuning parameters. Several well-known tuning approaches also exist, such as the Zeigler-Nichols tuning method [19]. However, none were implemented due to the ease of experimentation and relative success of this basic approach.

The complete control flow of *Romulus'* function is detailed in Figure 18. Almost all processing is performed on the Raspberry Pi, with the microcontroller functioning solely as our linear servomotor system.



Initialize Camera, SPI, and Position
Read Camera Frames
Range Interval Passed?
Yes
No
Apply Stereo Rectification
Calculate Disparity and Distance
Object Detector
Target Found?
Yes
No
Extract Bounding Box
Display Output Visuals and GUI
Convert to Relative Angles
Ref.
Digital PID Controller
Control Relative Angles
Previous Relative Angle*
New Absolute Angle
Analytical Inverse Kinematics
Actuator Lengths
SPI Transaction
Empirical Forward Kinematics Lookup*
*This is the role of forward-kinematics measurements. However, implementation also worked under steady-state assumptions in which the previous angle is assumed to be the last relative control angle output.

Figure 18. Block level diagram of digital tracking main program for *Romulus*. Note empirical forward kinematics was not used in all finalized implementations. Additionally, the digital PID controller was tuned prior. Not shown are relevant timing checks, such as the interval over which a new stereovision depth calculation is performed. This implementation was created using both methods of distance estimation from block-matching, however, the fitted parameters generally performed quicker in real-time applications.

Linear Servomotor Implementation

To control each of the linear actuators for the 3-RPS robot, we implemented our own custom linear servomotor system. While linear servomotor systems are commercially available, typically the added cost, interfaceability, and size presents challenges. When researching products, it was incredibly difficult to find linear servomotor systems which both met our specifications for strength, speed, and cost, while also providing adequate interface and size. Several of these systems implement many unnecessary features at the expense of size and cost. For instance, many implemented remote control packages or other proprietary control interfaces. Therefore, we decided to implement a custom servomotor system.

A servomotor system consists of a motor, a controller, and a driver. In this system, a target position is specified as input to the controller. The controller then provides input to the motor driver circuit, which sets the appropriate voltage and polarity needed. Additionally, feedback is measured from the motor continuously, which allows the controller to adjust its input to the motor drivers over time.

The linear actuators we chose were Actuonix's P16 series [20], as they were a great compromise between high-speed, strength, and cost. They have a built-in potentiometer used for position-sensing feedback. By setting an input reference voltage, the current length of the actuator can then be measured as a proportion of the input voltage using simple voltage division. The internal motor for these actuators is a simple brushed DC motor, which has the advantage of being relatively light, cheap, and easy to control. The speed of each motor is directly proportional to average voltage applied, while the direction they move is simply the polarity of the applied voltage.

To drive power to each linear actuator, we use the DRV8874-Q1 H-Bridge from Texas Instruments [21]. The basic function of an H-Bridge is four transistors applied across the terminals of a load device, in this case our linear actuators brushed-dc motor, see Figure 19. By applying a pulse-width-modulated input voltage to a pair of gates, the average voltage can be set. The duty cycle of the pulses corresponds directly to the average voltage, and therefore speed. Applying the pulse-width modulation to the opposite transistor pair creates the same average voltage but of the opposite polarity, which allows the direction to be reversed.

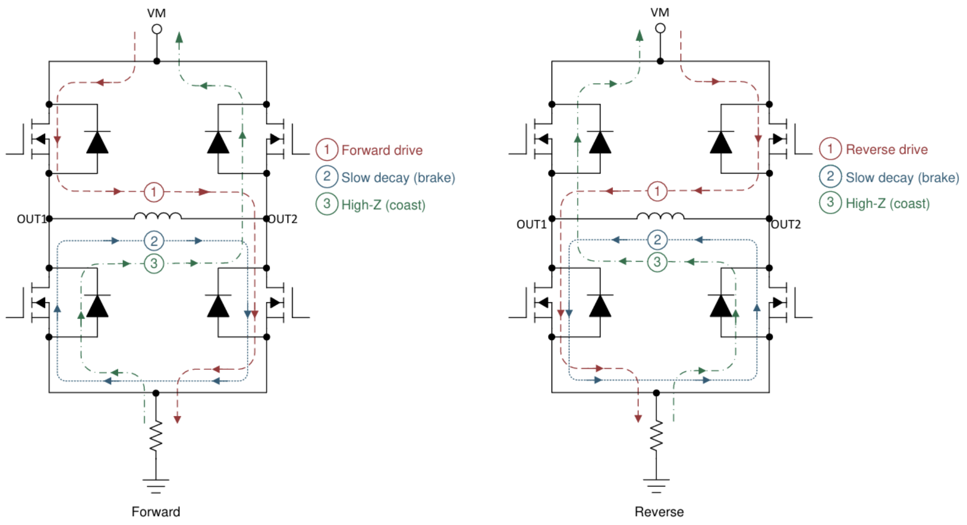


Figure 19. Basic function of an H-Bridge. A ladder of transistors is applied across the leads of a brushed DC-motor. By pulse width modulating the gates of the top left and bottom right transistor, a target average forward voltage can be driven through the motor in one polarity. Then, by driving a pulse width modulation to the opposite gates, an average reverse voltage can be applied to drive the motor in the opposite direction. Diagram taken from Texas Instruments, **[21]**.

Initially, we intended to implement our own H-bridges with discrete transistors. However, there were several desirable reasons to use an off-the-shelf component. First, initially we considered using dynamic force control in the robotic design, which necessitated measuring the current through each H-Bridge. Additionally, for high-power transistors, driving the gate-capacitance with digital inputs can be non-trivial. Finally, and most important, is the effective dissipation and regulation of the high-peak power through each motor. Because motors appear like a highly inductive load, through which the current cannot instantaneously change, there is a large surge current that must be dissipated when they stop or change direction. This can potentially cause damage to other components if not properly handled.

However, this Texas Instruments Chip ultimately gave us a good compromise between interfacility, robustness, and features. It has integrated current sensing to estimate force. It also features an integrated charge-pump that boosts digital inputs to drive the H-bridge transistors. Controlling its speed is still done with pulse-width-modulation as an ‘enable’ signal. The direction is a separate digital input called ‘phase’, summarized in Figure 20.

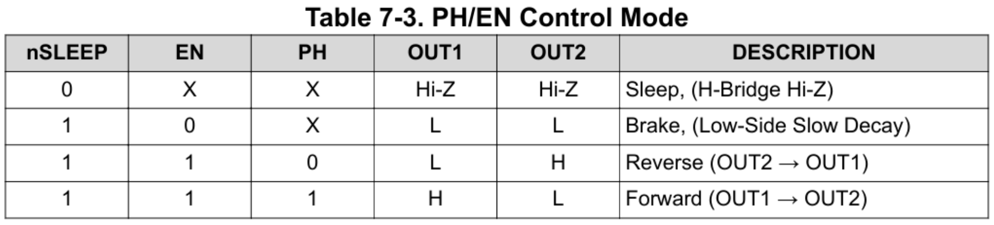


Table 7-3. PH/EN Control Mode

nSLEEP	EN	PH	OUT1	OUT2	DESCRIPTION
0	X	X	Hi-Z	Hi-Z	Sleep, (H-Bridge Hi-Z)
1	0	X	L	L	Brake, (Low-Side Slow Decay)
1	1	0	L	H	Reverse (OUT2 → OUT1)
1	1	1	H	L	Forward (OUT1 → OUT2)

Figure 20. Motor Driver digital control inputs from Texas Instruments Datasheet **[21]**. Note sleep input is driven high to enable the device. Enable is pulse-width-modulated to set the speed. If not input, the motors enter a brake state. The phase pin, which uses digital GPIO sets the direction based on its state.

The last element of the servomotor system is the controller. For this, we use the STM32G071RBT6 microcontroller [22]. The primary function of the microcontroller is to provide the feedback and control system needed to quickly, but accurately, position each of the robots’ linear actuators. The microcontroller receives desired reference positions from the Raspberry Pi through SPI transactions. The microcontroller also reads the current position of each actuator by measuring the potentiometer reference voltage with analog-to-digital converters. The difference in current position and reference position is considered the error signal. Using this error signal, the microcontroller uses proportional-derivative control to quickly force each actuator to the correct length. The sign of the error determines the phase (direction of motion), while its magnitude determines the pulse width (average voltage and speed). These are fed to the phase, enable, and sleep pins of each H-Bridge chip. The process is summarized in Figure 21, which shows the full code execution process performed on the microcontroller.

The microcontroller receives reference positions from the Raspberry Pi using SPI configured as a full-duplex slave device. Direct memory access (DMA) was used to efficiently perform each SPI transaction with the Raspberry Pi. In each communication, the packet width was fixed and a redundancy key unique to the Raspberry Pi and microcontroller was added for data integrity and monitoring. Because the hardware overlays written for the Raspberry Pi's SPI drivers do not always correctly hold the SPI data-lines inactive, a chip-select signal proved important to have for SPI. This chip select signal, NSS, is correctly toggled at the start and end of each SPI transaction. This was helpful to ensure SPI DMA on the microcontroller was not spuriously triggered. In each exchange, the STM writes the current actuator lengths, while the Raspberry Pi writes a new set of lengths.

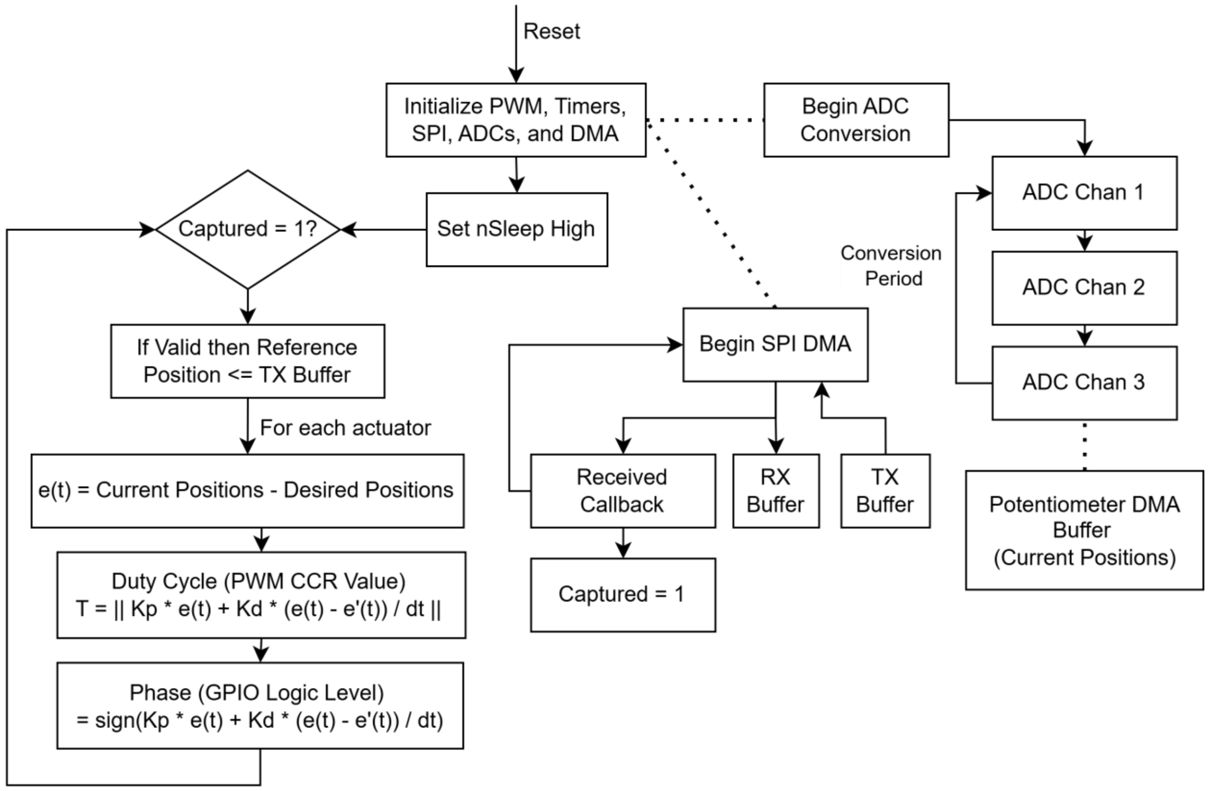

Reset
Initialize PWM, Timers, SPI, ADCs, and DMA
Begin ADC Conversion
ADC Chan 1
ADC Chan 2
ADC Chan 3
Conversion Period
Potentiometer DMA Buffer (Current Positions)
Set nSleep High
Captured = 1?
If Valid then Reference Position <= TX Buffer
For each actuator
e(t) = Current Positions - Desired Positions
Duty Cycle (PWM CCR Value) T = || Kp * e(t) + Kd * (e(t) - e'(t)) / dt ||
Phase (GPIO Logic Level) = sign(Kp * e(t) + Kd * (e(t) - e'(t)) / dt)
Begin SPI DMA
Received Callback
RX Buffer
TX Buffer
Captured = 1

Figure 21. Software Control flow on STM Microcontroller. Dotted lines indicate processes accelerated on separate hardware components, as direct memory access was used for analog-to-digital conversion, and SPI transactions. Callbacks are included in the main code to set flags checked during each iteration of the main loop.

Analog to digital conversion was also done with DMA and continuously converted in the background. By continuously sampling each channel the current length of each actuator is measured. A signal ADC was used, however, it has separate multiplexed channels for each device. All were read into a circular buffer of current positions.

The main digital outputs were to the motor driver H-bridges. Two GPIO output pins were configured to enable the device (nSLEEP=1) and set the direction of motor movement (PH). The PD controller was a value of desired motor velocity based on the current positioning error. Therefore, the sign determined the digital value of the direction pin (PH). The speed was controlled using the hardware accelerated PWM channels of the same hardware timer. The period was set to be 20 kHz, with the duty cycle determined using a count-up register. Also, inside of each PWM channel is another register that sets the effective duty cycle. When the count is below this register's value, the PWM output is high, while when it exceeds this value, its output is low. Therefore, this capture-compare register was set for each timer channel to set the effective duty cycle, and therefore speed for each actuator's H-Bridge. Overall, this converted a speed generated by the PD controller to a duty cycle as a fraction of the maximum counter compare register value, and an input to the enable pin of each actuator's motor driver.

Frame and Joints

In designing the frame, we worked under two major design principles, one being that we wanted lightweight custom parts. This required the use of 3d printing. The second principle is a result of the first, the design program we used, Fusion 360, has no way to reliably model 3d-printed components due to the design process printing in layers instead of a homogenous solid. This led us to print a component and test it physically rather than in a simulator, then iterate based on the failure points observed. This led to many failed components but allowed for rapid iteration of parts and a final product that had spawned from thorough stress testing.

The frame has four critical sections, the base triangle mounted against the shoulder, the handle, the end effector, and the joints that connect them all. To start we prioritized making the base, joints, and end effector. This would allow us to immediately start testing and prototyping the inverse kinematics of the robot arm. Pictured below, in Figure 22, is the first iteration of the robot arm we utilized for testing. For the Spherical joints we chose the highest rotation spherical rod end that McMaster-Carr offered, so that we would have as much freedom of movement as possible. These ended up being ¼" rod ends as picture in Figure 22.

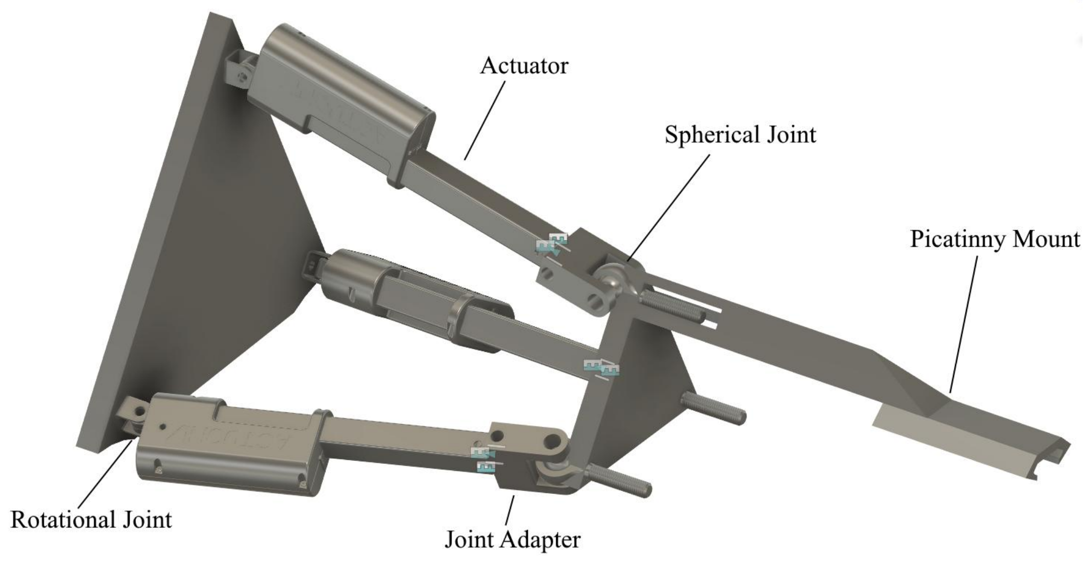


Figure 22. First iteration of robotic arm used for prototyping and testing.

For this first iteration we chose a scaling factor of two times between the two platforms, so that the robot had plenty of freedom in its movement. For the rotational joints we used the base metal joints that came with the actuators, thinking that they would be stronger than anything we could make. However as soon as we assembled the arm, we realized these joints would be unusable, due to their extremely loose tolerances. This is why we moved to integral rotational joints in future iterations, as seen below in Figure 23. Another problem with this design is the thin arm meant to support the Airsoft rifle. This arm almost immediately experienced a delamination along the 3d-print layers. This is a consistent problem we experienced when using 3D printing, because the plastic is deposited in layers, it requires much less force to separate these layers compared to breaking layers in half.

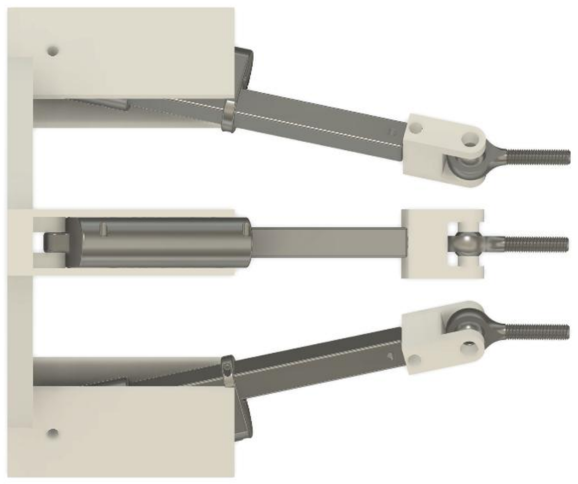


Figure 23. The second iteration of the base of Romulus, take note of the much lower tolerance rotational joints, and guides for the actuator base.

In the second iteration of the base, we incorporated integral joints as well as guides for the actuators themselves. The reason for these guides is that the actuators themselves have very low tolerance on the hole used for rotational motion. This hole is 4.25mm but is made for M4 machine screws so there are .25mm of slop in the actuators themselves that couldn't be affected by a joint. After much tuning we found that a tolerance of 0.1mm in most of our components allowed for parts to fit while still being tight and secure. However, we did not account for the amount of forces these guides would have to survive, and they almost immediately cracked and bent along print layers due to this force.

In the final design, pictured in Figure 24, we increased the thickness near the edges, like an I-Beam and added sleeves for long machine screws to act like rebar and support the plastic. These reinforced the joint and stopped any movement. Additionally, at this point in time the PCB was finalized so we incorporated an integral housing to hold it in place. This used custom standoffs to allow for the PCB to be swapped in and out and had an open face for wires to connect to the PCB.

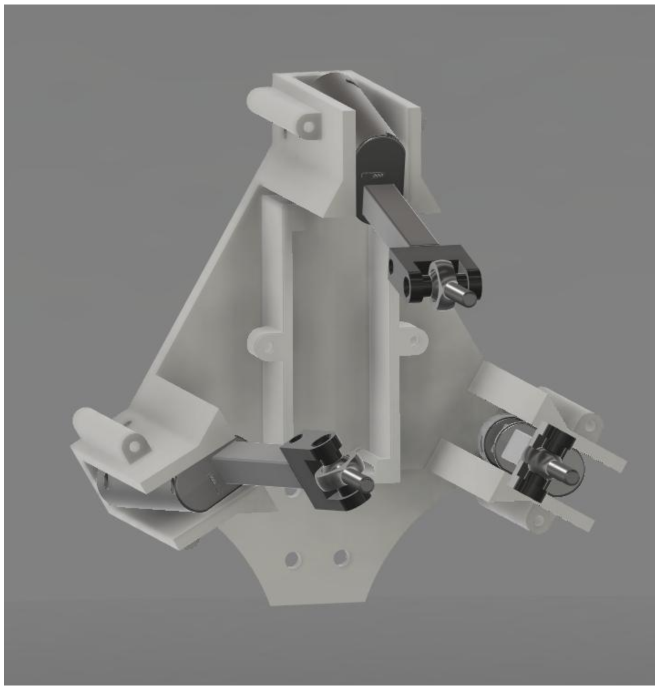


Figure 24. Final version of the base. This version improved joints and added the PCB cover.

The first end effector had many problems, even beyond structural issues, there was no way to connect the Airsoft gun to the picatinny mount, nowhere for the camera, and nowhere for the rangefinder. The next revision, pictured in Figure 25, attempts to fix all of these problems.

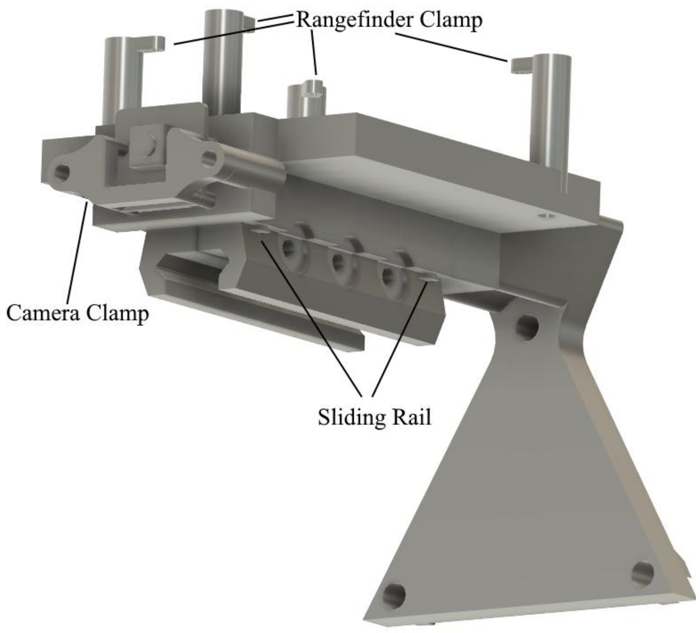


Figure 25. The next generation of the end effector featuring many improvements.

The arm is much thicker and no longer has to be hollow to allow for room for the spherical joint. The picatinny mount is now made of two separate components. The one on the left is integral to the end effector, while the one on the right slides into place along plastic rails and squeezes the picatinny rail through the screws going through both sides. The camera is now placed front and center and held in place by a clamp that squeezes the mounting holes present on the PCB.

This design had two main problems. The first problem was a catastrophic structural failure that occurred at the topmost spherical joint hole where it necks down. This fault occurred along a layer plane, but also because of an inclusion that occurred during printing. This inclusion effectively made a gap in the layer that wouldn’t adhere as well, which made a weak point in the overall structure. The other problem came when we switched from the rangefinder to stereo depth mapping. This system level change brought in two complications, first a second camera mount had to be added, and second the rangefinder clamp was obsolete so it could be replaced with a mount for the Raspberry Pi.

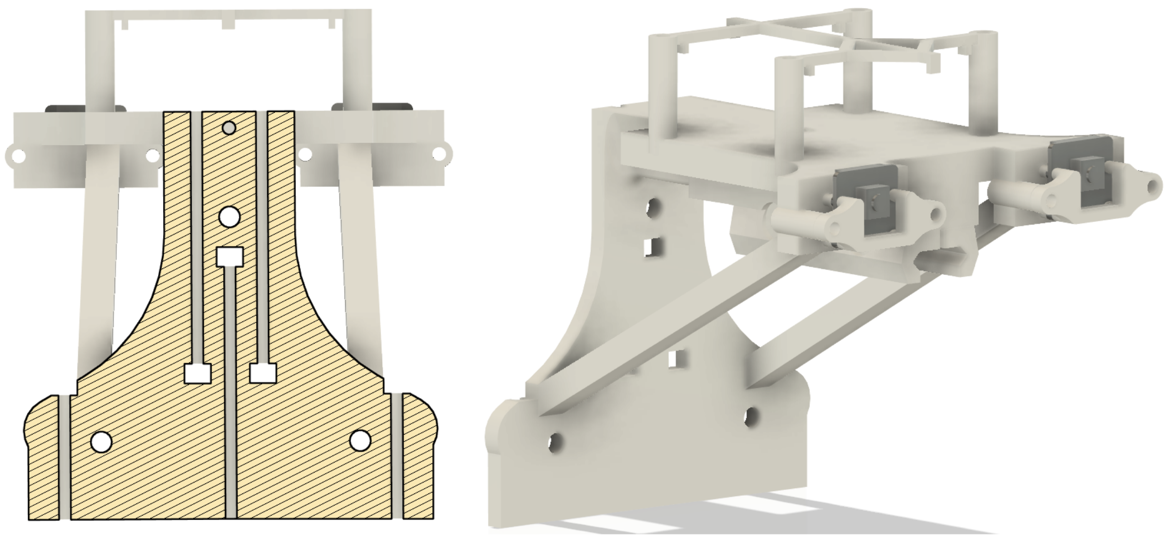


Figure 26. Final end-effecter design, with a cross section to the left showing where screw reinforcements have been added.

The final version of the end affecter pictured above in Figure 26 fixes both of these issues as well as several others. To address structural support three changes were made, first the thickness of plastic around each hole was increased. Second, holes were added for screws to run the length of the end affecter to squeeze the printer layers together and spread force along them. Lastly, plastic support struts have been added between the camera attachment point and the triangular base. These changes fixed all the structural issues being experienced. This is easily demonstrated with the FEA pictured in Figure 27 showing the stress when 50 pounds of force are applied to the end effector.

Aside from this we made three other changes in this iteration. The first two just involved turning the rangefinder mount into a mount for the Raspberry Pi and adding a second camera mount. The third change was making the base triangle of the end affecter bigger. We decided to change the 2x ration to a 1.75x ratio. This increased the size of the workspace, especially in the vertical direction and allowed for more structural reinforcement

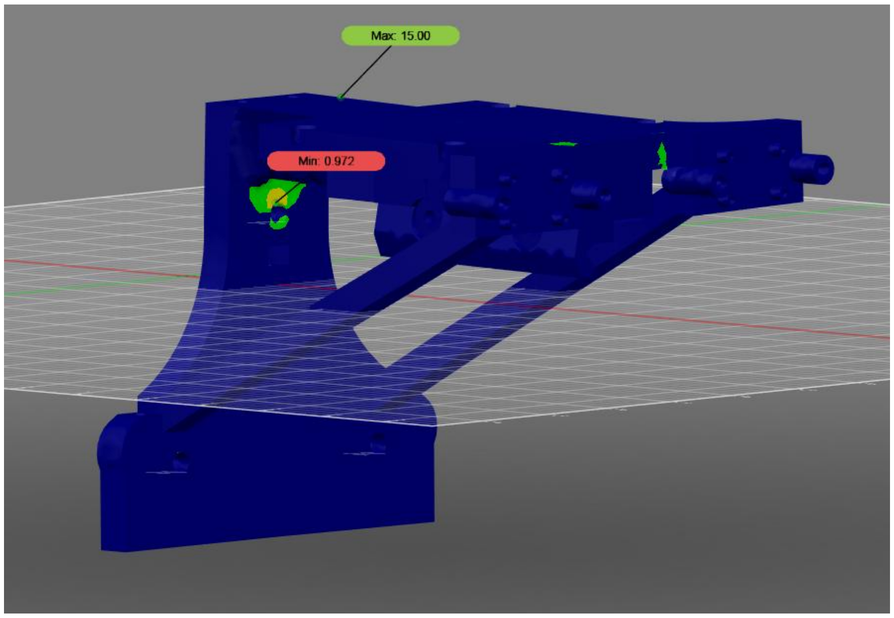


Figure 27. FEA analysis of the final end affecter. The most structurally sound is colored in blue and then green, yellow, and finally red

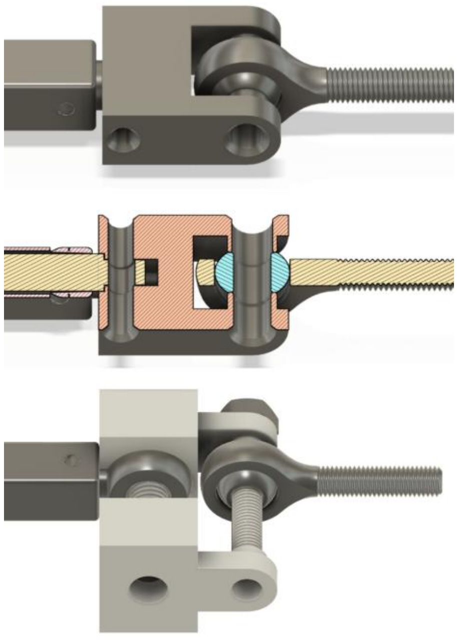


Figure 28. Joint, below that is the corresponding cross section, below that is the final joint.

With the design for the base and end affecter covered they now needed to be connected together. The actuators came with a screw in plastic end piece with a hole for mounting. This hole however used M4 screws and needed to be adapted to the ¼" hole used for the spherical joints. To do this we 3d printed a custom adaptor pictured in the top two images of Figure 28. Once tolerances were dialed in these worked really well and had almost no slop. However, the plastic end piece that came with the actuators broke during testing. These end pieces use M8 machine screw threading, so we replaced them with metal M8 rod ends and made new adapters. These can support much more weight than the plastic ones that came with the actuators. Additionally, they stayed screwed in for longer periods of time and don't experience any wear compared to the plastic joints.

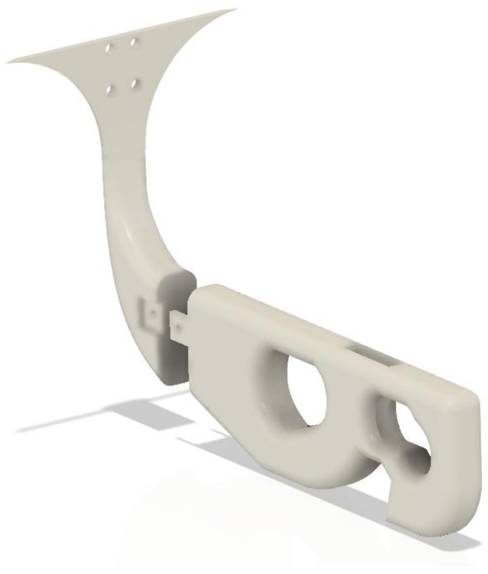


Figure 29. First iteration of handle and linkage to connect it to the base.

The last major components of the frame are the handle and the linkage to connect it to the base of the 3-RPS arm, pictured in Figure 29. When first printed out the linkage was 10cm thick and only featured 20% infill compared to the 100% used on almost every other part. We figured this would be fine but were quickly proven wrong. During a bout of user testing to judge the intuitiveness and comfortability of the handle the linkage snapped right below the bolt connections. To address this, we increased the infill to 100% and doubled the thickness to 20cm, because strength is proportional to cross sectional surface area this increased the strength of the component by a factor of 10.

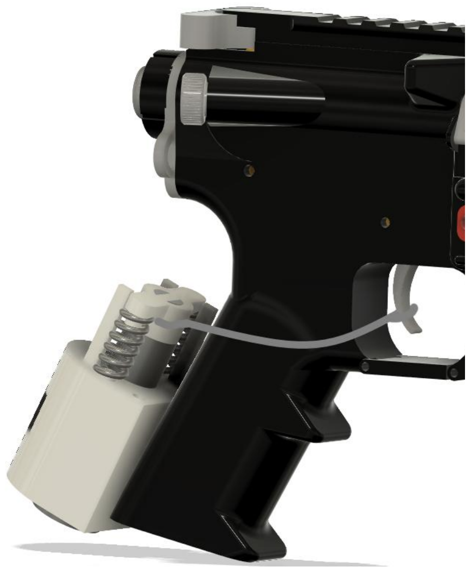


Figure 30. Trigger mechanism attached to Airsoft handle.

The handle is modeled off of a P90 handle due to its short form factor. This allows for the Airsoft gun to aim down without hitting the handle. In Figure 30, there is a slot in the handle, this is used to install the trigger and the button that activates the remote trigger.

The remote trigger, pictured in Figure 30. Uses a solenoid to pull the trigger of the Airsoft gun on command. This solenoid is powered by the 12V battery and triggered by a button in the handle. When activated the solenoid pulls the trigger using a string to connect the two. Lastly after the trigger is released and the solenoid deactivated the springs return it to a resting position so it can be activated again. This assembly is connected to the handle of the Airsoft gun using Velcro straps, for quick and easy assembly.

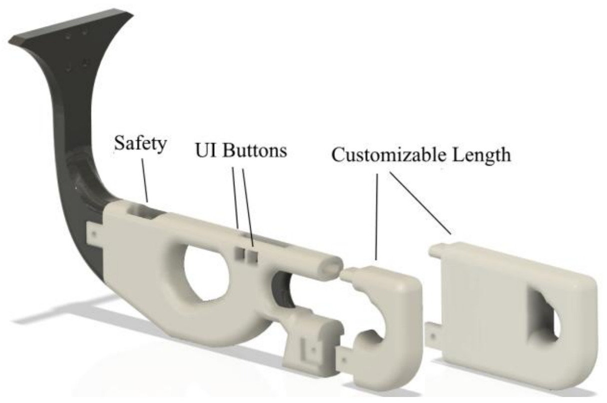


Figure 31. Final version of handle with quality-of-life features for the end user.

The final version of the handle, pictured in Figure 31wasn't made to address any problems, but to add features intended for an end user. First and foremost a safety was added so that when being transported the trigger can be pulled but it won't do anything. Beyond that UI buttons were also added right above the trigger. These buttons are intended to add features such as reset for the 3-RPS arm or a button to turn off the tracking all together. Lastly the end of the handle was changed to be detachable so that the length of the handle could be extended to whatever the user was comfortable with. This last change came in part because the center of mass will change with an Airsoft gun attached to the end affecter, and in traditional handle designs one hand is positioned before the center of mass and one hand after. If both hands were behind the center of mass, it would be much more unwieldy and extremely uncomfortable.

Schematic

The block-diagram circuit schematic for this project is seen in Figure 33.

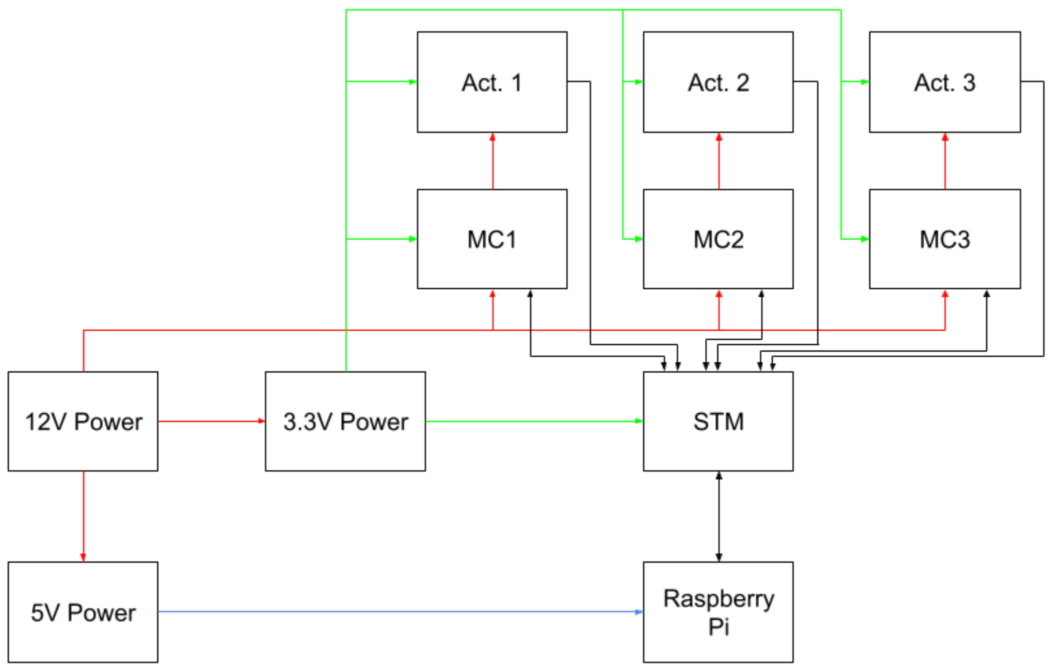


Figure 33. Block-Diagram for Circuit Schematic. Note that red arrows represent 12V power signals, blue represents 5V, green represents 3.3V, and black represents data. MC stands for motor controller and Act. stands for linear actuator.

The 12V power supply, consisting of an RC battery, connects to two DC-DC voltage converters that step the voltage down to 5V and 3.3V. The 12V supply is also connected to each of the 3 motor drivers to drive the linear actuators. The 5V power is connected to the Raspberry Pi. The 3.3V power is used to power the STM32 microcontroller and as a voltage reference for the motor drivers and linear actuator potentiometers. The Raspberry Pi is connected to the STM32 and communicates through SPI. The STM32 connects to each motor driver, each one driving their respective linear actuator. Each linear actuator's potentiometer is connected directly to the STM32.

This general circuit diagram has stayed close to unchanged throughout the whole project's lifetime. However, the implementation of this circuit has changed and developed with time.

The first design choices made were what computer, microcontroller, linear actuators, and motor drivers should be used.

Linear actuators with the strength and speed needed are expensive, leading us to pick actuators to buy at the start and use for the entire project. This means we selected actuators that would give us the most flexibility and must last through the whole semester. The Actuonix P16-P 100mm 22:1 12V linear actuator shown in Figure 34, was chosen over other linear actuators. The 100mm stroke gives enough flexibility that we will not need to worry about being constrained on length. The gear ratio of 22:1 provided the fastest speed at which the actuators could extend or retract. One of the most important features of this actuator is the potentiometer feedback. This allows us to create our own servos by writing control logic, whereas linear servos are much more expensive. Each linear actuator came out to be within our budget at $90 each.

Figure 34. Actuonix P16-P 100mm Linear Actuator.

With the linear actuators selected, the next step was to find a motor driver that would support the 12V and up to 1 amp of current that each linear actuator will use. The DRV8874 H-bridge motor controller was selected due to the robustness of the chip, driving up to 6 amps and controlled through pulse-width modulation. This allows the STM32 to easily control these motor drivers. Another reason was that these chips have many power protections for when the linear actuators turn off. When motors turn off, they create a large inductance that floats the ground to a higher voltage, which could damage other components. This was the primary reason that we chose to use the DRV8874 over building our own H-bridge circuits.

A Raspberry Pi was chosen to be used because of the computing power available. The initial and final design used image classification models to find the targets. To accurately control the 3-RPS system, many intensive calculations are required to properly model and control the robot. The initial plan was to use the Pi itself to control the motor drivers; however, this requires the Pi to use ADCs to read the actuator potentiometers which it does not have built in. Instead of using a

Raspberry Pi ADC peripheral, a STM32 microcontroller, STM32G071RBT6 was chosen due to both lower cost and familiarity within the group.

The initial circuit was prototyped on a breadboard. The schematic for each motor driver was taken directly from the DRV8874 datasheet seen in Figure 35, [21]. The values were also taken directly from the recommended values in the datasheet.

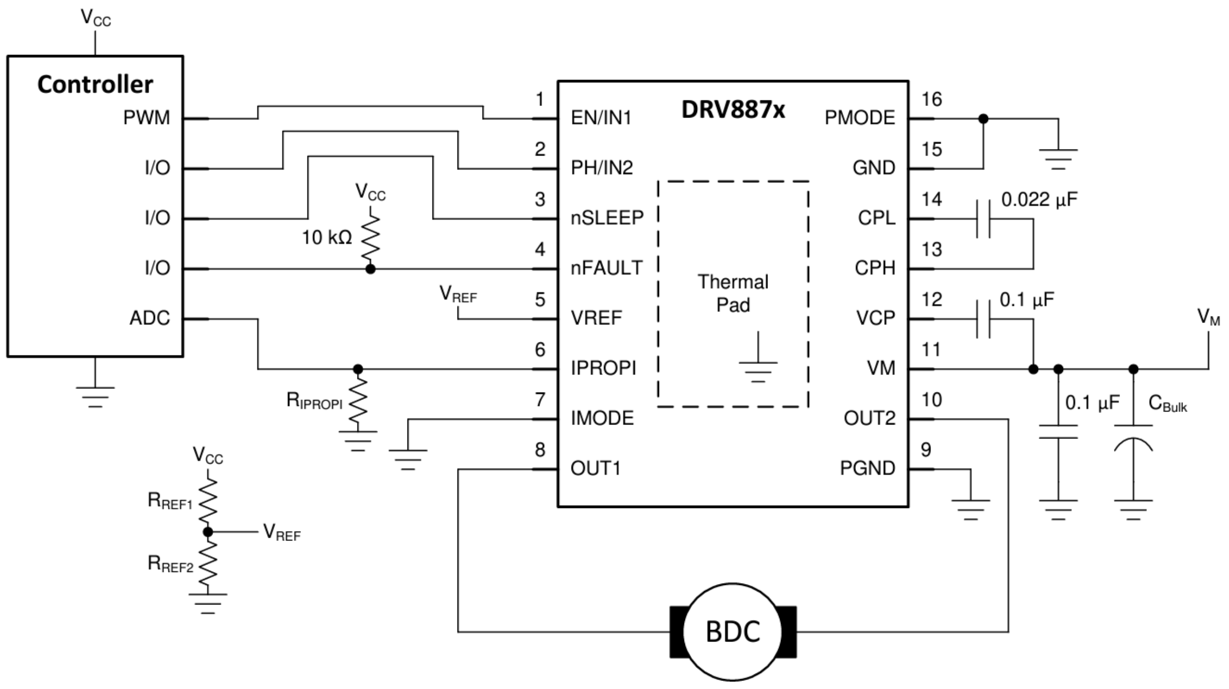


Figure 35. Motor Driver Schematic.

The motor drivers and linear actuator connectors were put onto a breadboard. The Raspberry Pi and STM32 debug board were then connected to the breadboard and to each other. All parts were powered via direct power supplies. Part of the breadboarded circuit (the motor drivers) can be seen below in Figure 36.

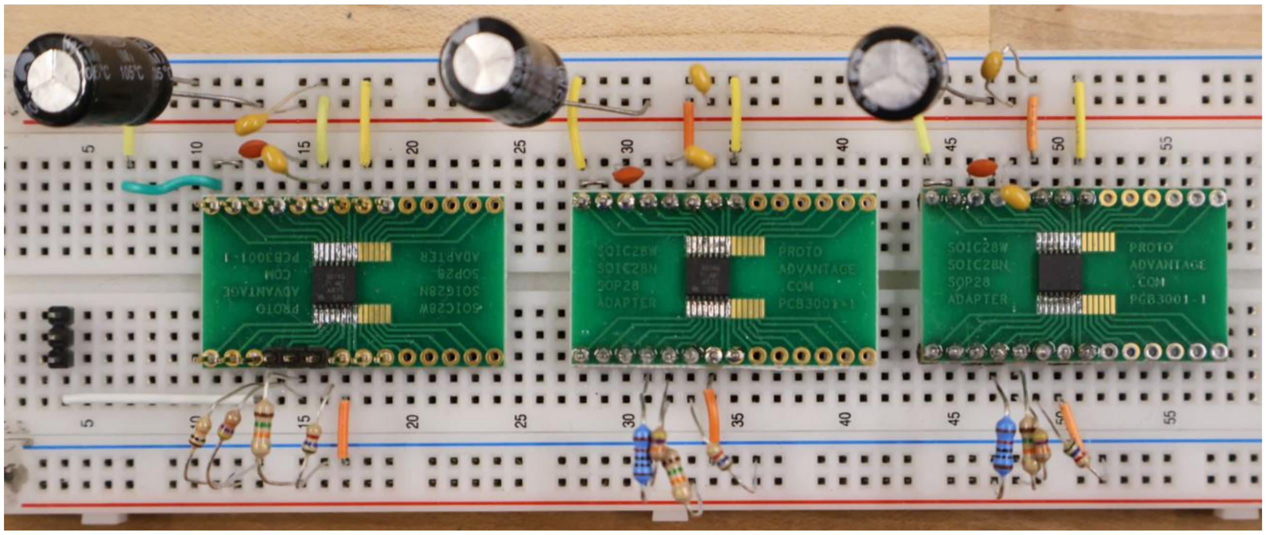


Figure 36. Breadboard testing with surface mount motor driver circuits. A breakout board was purchased to adapt the surface mount chips to a through hole package.

During testing, this initial design did not work. After debugging, it turns out that the CPL-CPH capacitor in the motor driver datasheet is the wrong value. This caused the motor driver to not work. Decreasing the value from 22 nF to 2.2 nF fixed the issue.

This initial prototype proved that our general circuit schematic worked and was able to effectively control the 3RPS system. It also showed an error in the motor driver datasheet that would have been extremely hard to debug if we started directly with a custom printed circuit board (PCB).

The next step was to create a custom PCB that has two main functions: consolidating the circuit elements into a more compact space and allowing the system to be mobile. This required more design decisions to be made regarding batteries and voltage converters.

One of our design requirements was that the system must be mobile. This meant that every electronic used needs to be powered via battery. Within the N.I. lounge, two 11.1V RC batteries were found that should work for this project. While the actuators can work up to 12V, they will work at 11.1V at a slightly slower speed. These batteries had 4000 and 5000 mAh of charge. An estimated power calculation was performed by assuming each actuator would draw 1A each and the Raspberry Pi would draw 2A at 5V. This calculation estimated that each battery should last a little bit over an hour and should be an underestimate, meaning these batteries should provide enough battery. The batteries need a male XT-60 connector on the PCB to connect.

The next design consideration was converting the battery's voltage to 5V and 3.3V. The TLVM13610 with a 3V to 36V Input and 1V to 10V output was chosen to be used for both conversions. The TLVM13610 allows the output voltage to be set by a referencing circuit and can provide up to 8 amps of output current. This means that we should never have to worry about our power converters being current limited. Another feature is that even if the voltage input drops as the batteries drain, the output voltage will stay consistent regardless. The TLVM13610 is also well documented with example schematics for both 5V and 3.3V. The schematic for the DC-DC voltage converter found in the datasheet can be seen below in Figure 37 [23].

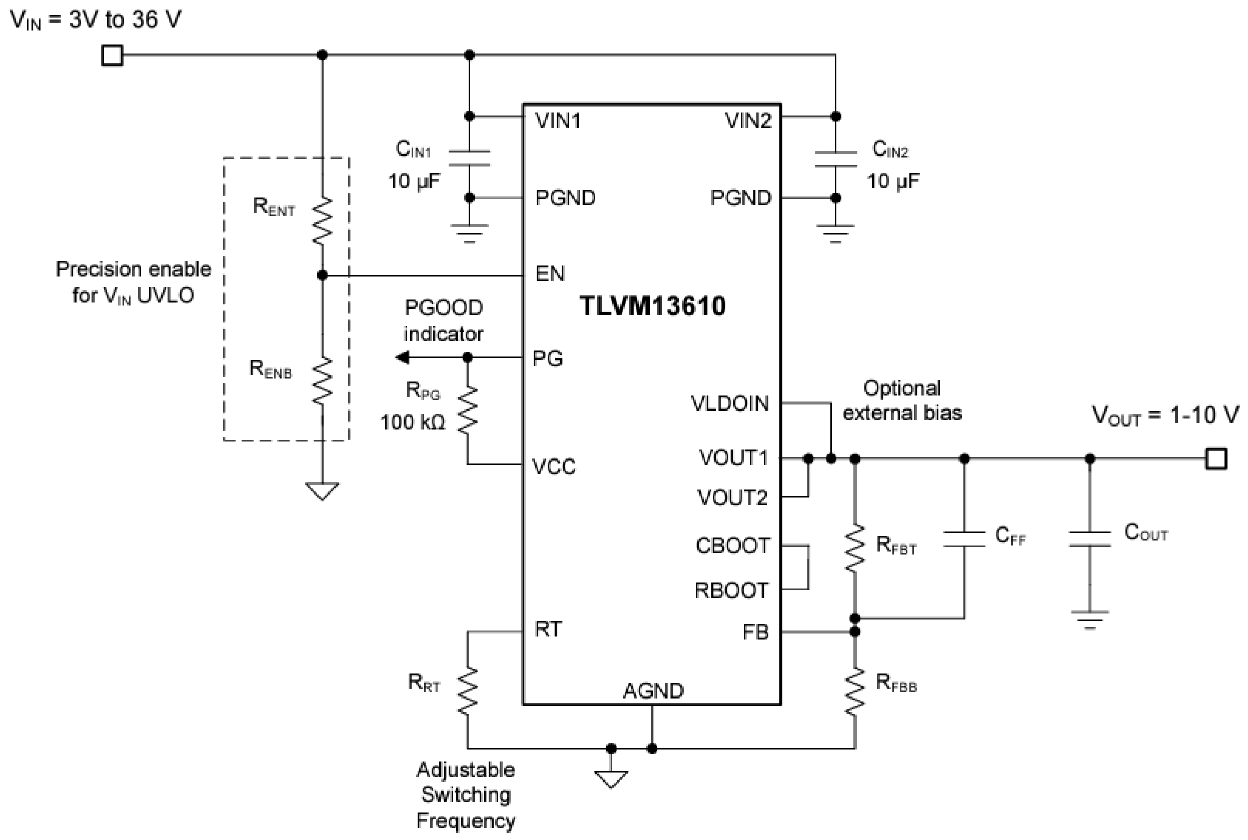


Figure 37. TLVM13610 DC-DC Voltage Converter Schematic.

An initial PCB was then developed to provide power to every system and house the STM, motor drivers, power circuitry, and linear actuator connections. This PCB used jumpers to isolate and debug STM signals, functioning as a prototype board. The initial PCB design can be seen below in Figure 38. The size of the PCB was 3.85" by 1.425".

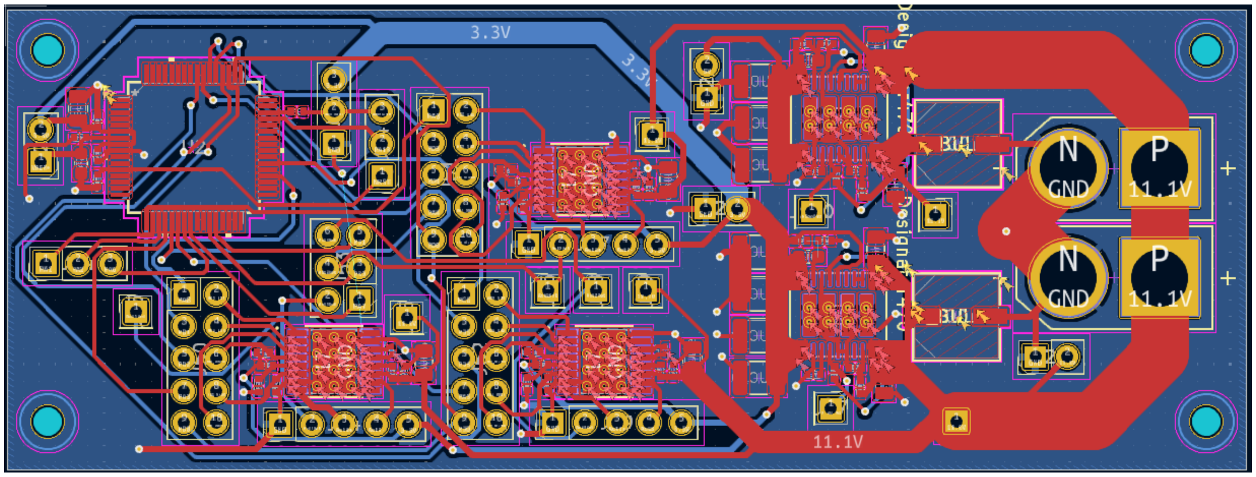


Figure 38. Initial PCB Design.

This PCB was then manufactured and assembled by JLCPCB. However, JLCPCB did not have the TLVM13610 modules in stock. Waiting for those parts to arrive at JLCPCB to assemble would have taken an extra 2 weeks minimum, leading us to order this part and solder it ourselves. The TLVM13610 turned out to be incredibly hard to hand-solder, taking hours to get correct. The physical assembled circuit can be seen below in Figure 39.

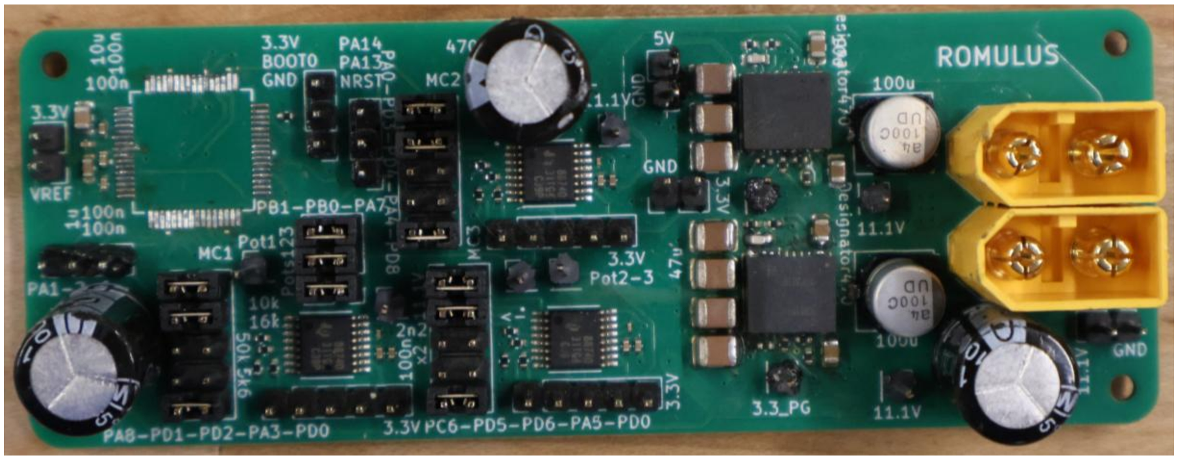


Figure 39. Assembled Initial PCB. Note this was taken after the STM was desoldered.

This initial PCB had a few flaws with it. First, multiple STMs and TLVM13610s were fried in the debugging process. This was primarily due to the TLVM13610 output being connected directly to the 3.3V trace with no jumper between and the soldering being nearly impossible because the pads are underneath the package. When flashing the STM using the ST-Link, the 3.3V pin on it drives power. However, we did not know this at the time and put the pin on the board. This fried the

TLVM13610s as the output was being driven. The STM would also sometimes get fried if the TLVM13610 had poor soldering. This required us to desolder and resolder new components many times. This culminated in a pin of an STM evaporating after 40-60 minutes of use, seen in Figure 40. We believe that this was caused by a short created by repeated desoldering and soldering.

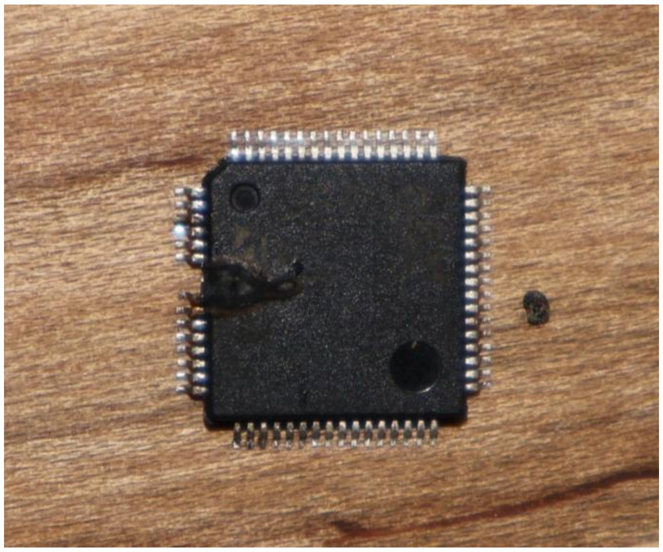


Figure 40. STM with Vaporized Pin.

While debugging, the reference voltage connection on the STM was called into question as it was connected directly to 3.3V. A bigger flaw in the PCB was that the intended pinout for SPI did not include the chip select pin. This meant that a workaround was required to get SPI to transmit data between the STM and the Pi. The PWM connections for each motor driver were on different timers which was harder to work with than if they were on one.

While this PCB never enabled the system to be completely handheld due to the soldering issues listed above, we were able to effectively use the robot and connect all components together through the PCB. This included figuring out how to flash the STM32 through serial wire debug. We also verified that the power converting circuitry worked.

For the final PCB, the above issues were addressed. A jumper connecting the output of the voltage converters to the rest of the board was added, allowing easier debugging. The jumpers between the motor drivers and the STM were also gotten rid of in the final board iteration. Another improvement was using Molex connectors instead of GPIO pins. This provides a much more secure connection that will not come loose, which is important when moving around with the system.

The STM connections were completely overhauled, including adding ferrite bead for the reference voltage, the actuator potentiometers all being on the same channel of the ADC, and the addition of the SPI chip select pin. The final pinout for the STM can be seen below in Figure 41.

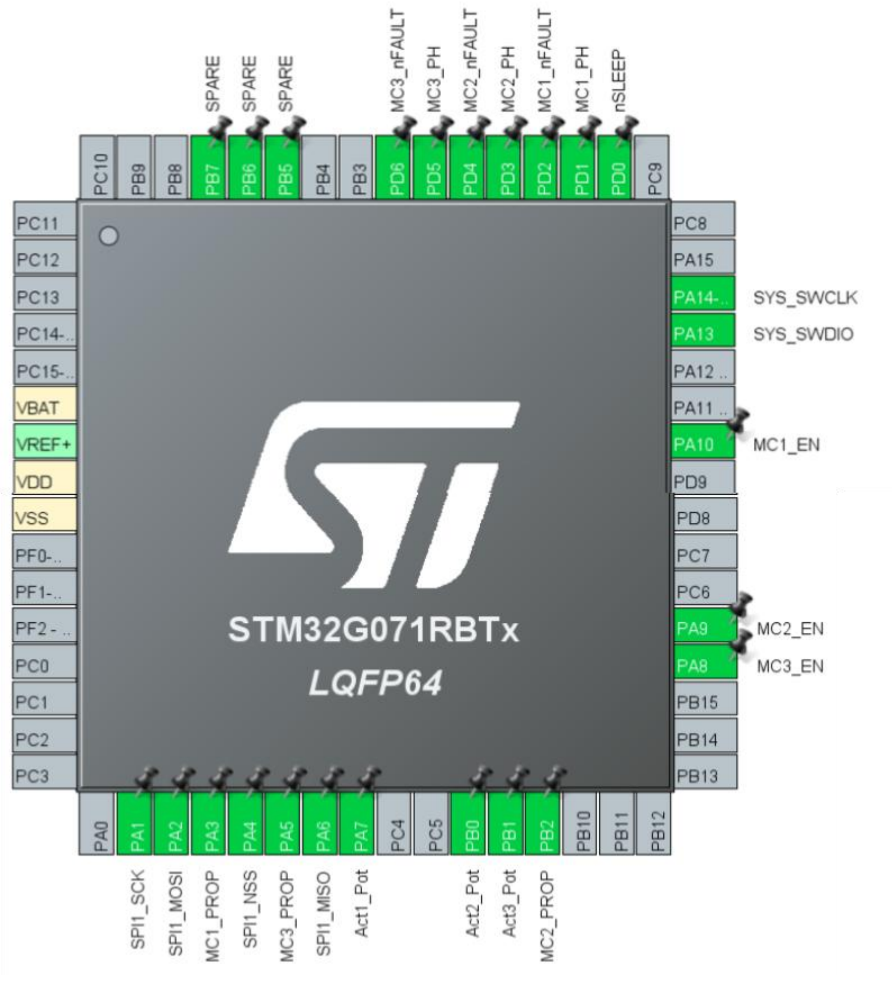


Figure 41. Pinout for the STM via the .ioc file.

This pinout pictures how the STM32 is connected to each motor controller’s EN, PH, nFault, prop, and nSleep pin. Each EN pin is connected to a pin on the STM32 that runs on Timer1 which allows for pulse width modulation. The STM32 has ADC pins connected to the wiper of each linear actuator’s potentiometer. The connections for serial wire debugging which includes SWCLK, SWDIO, and nRST are also enabled. There are 3 miscellaneous GPIO pins enabled at the top for backup use. The power circuitry for the STM32 is shown below in Figure 42, including the ferrite bead for the reference voltage.

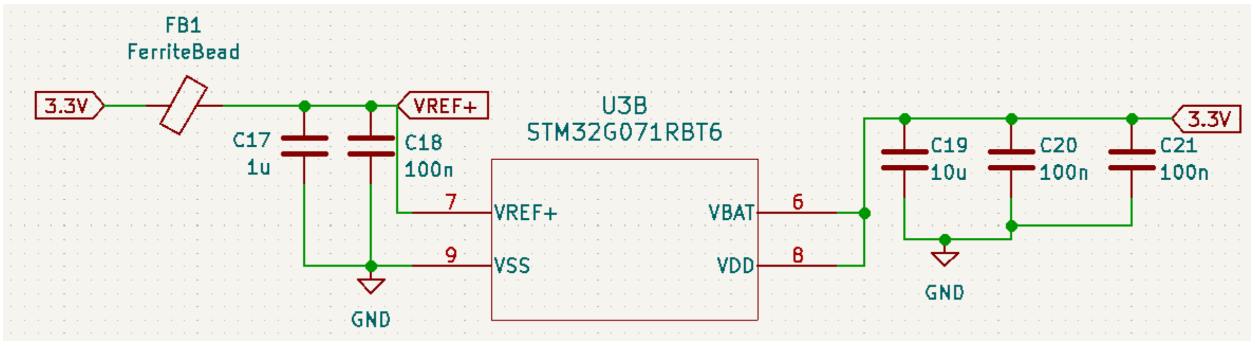


Figure 42. STM32 Power Circuitry.

The final PCB layout is seen below in Figure 43.

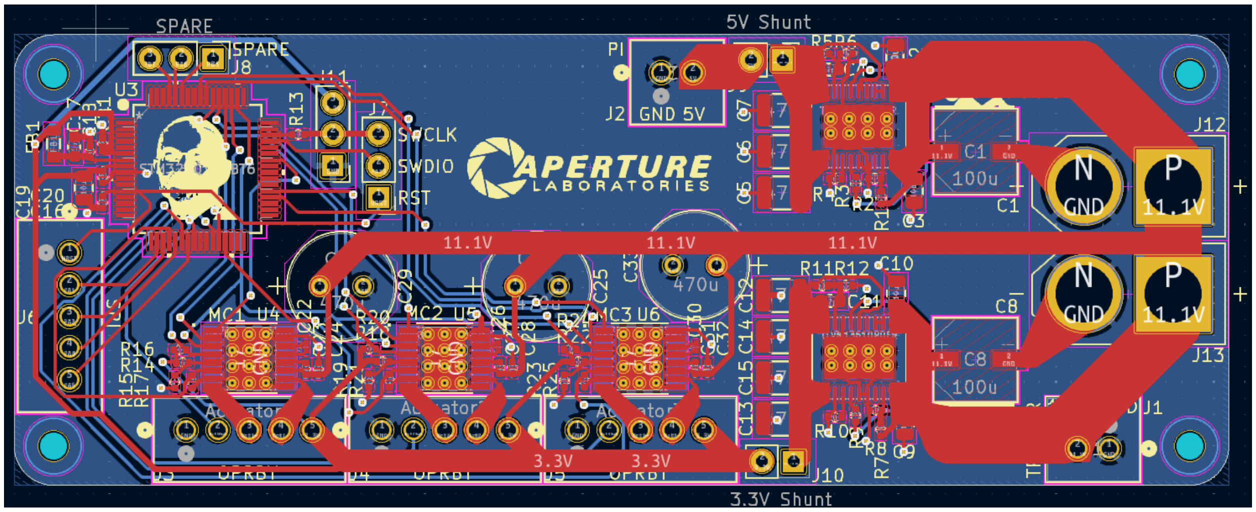


Figure 43. Final PCB Layout.

This PCB follows all the schematics shown throughout this section. The lack of jumpers freed up a lot of space for cleaner wiring. However, the board was kept the same size due to a mount for the old footprint being present in the frame.

There are four 5-pin Molex connections for the connection between the Pi and STM32 and a connector for each linear actuator. The 5-pin connector between the Pi and the STM32 is made up of 4 SPI pins (SCK, MISO, MOSI, CS) and the nRST pin. This will allow the Pi to reset the STM if necessary.

There are two 2-pin Molex connectors, one carrying a 5V power supply to the Pi and the other carrying the 11.1V power to a trigger system. The Pi is connected to this power supply via a custom Molex to USB-C connection. The trigger system will connect via standard wires with a Molex header.

There are two 3-pin GPIO headers to the right of the STM32. The one closest to the STM, from top to bottom, has 3.3V, the Boot0 pin, and ground. The intent for this jumper is to use a jumper to set Boot0 if you debugging is needed. The farther 3-pins correspond to the three signals required for serial wire debugging which is used to flash a program. This includes the SWCLK (the same pin as Boot0), SWDIO, and nRST. This allows the STM32 to be flashed using an ST-Link.

This board was then purchased from PCBWay. PCBWay was used instead of JLCPCB because we were not getting assembly service, and they were able to process the board much faster than JLCPCB. We then ordered all the parts and got the final PCB assembled by WWW Electronics. The BOM sent to WWW is found in Appendix A. By using a local assembly company, the TLVM13610 chips that were incredibly hard to solder by hand were able to be installed with good connections. This solved the problem with hand soldering them that was experienced with the initial board. Note that WWW soldered all the surface mount components. All through hole components were soldered by hand. The assembled PCB can be seen below in Figure 44.

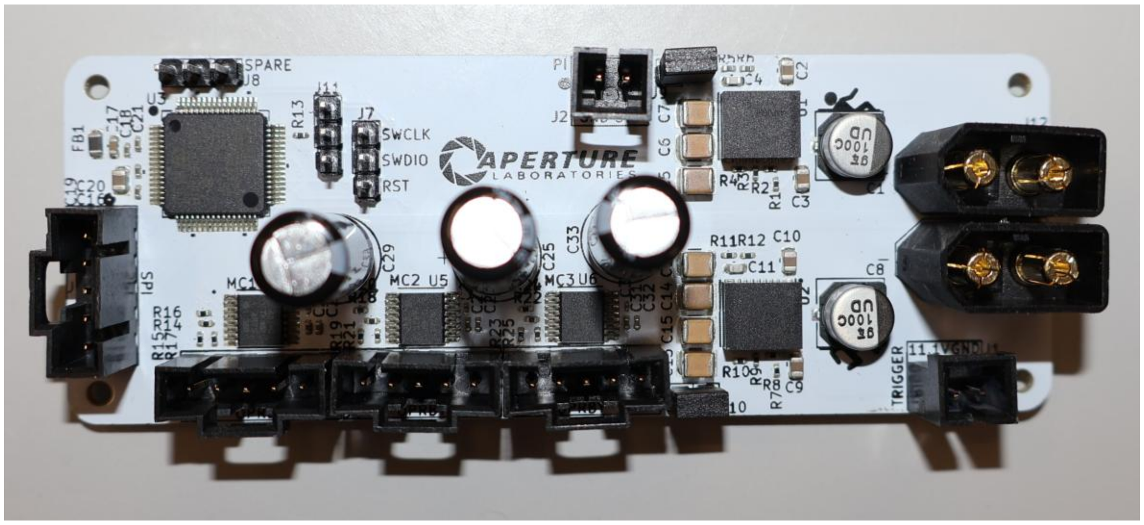


Figure 44. Picture of the assembled final PCB.

This PCB, once the STM was flashed, worked perfectly without needing to debug or troubleshoot.

To flash the STM, the following process was used. First, an ST-Link v2 was purchased from Amazon. Then, verify that the 3.3V power is not shunted. The SWSCK, SWDIO, RST, 3.3V, and GND wires are then connected to the PCB. Alternatively, the 3.3V connection should not be used if you are powering the STM through the on-board power supply.

Project Considerations

Physical Constraints:

In the design of Romulus there were several design constraints that had to be accounted for. Starting with the frame, this had to be made out of 3d printed plastic due to the limited time and budget of the project. One of the downsides of 3d printing is that everything is printed in layers, these layers reduce the strength of the material and require support material to support any overhangs in the design. This led to many of the design decisions and problems encountered while designing the frame.

The Raspberry Pi was also a source of many constraints. The Pi and specifically the Pi 5 model theoretically has hardware level communication protocol support for most common protocols, like SPI and UART. However, the Pi 5 changed the hardware implementation for these protocols, this meant that there was only one working SPI library and one working GPIO library, which significantly limited our programming options. Furthermore, UART refused to work, which is a common issue with the Pi 5. This is the main reason we chose to use SPI rather than UART. Lastly, while the Pi is much faster than any microcontroller, it is still limited, especially in terms of AI because it has no hardware optimization for that purpose. This means that Romulus is very CPU limited, especially with the complex image analysis used to calculate depth.

For the PCB, there were a few primary constraints. First, JLCPCB and PCBWay both had individual tolerances when it came to manufacturing PCBs. This primarily effected clearances between traces or pins and the size of through-holes. These constraints were standard and were easy to work around, not requiring explicit adaptations. However, the biggest constraints in terms of the PCB came with manufacturing and assembly time. JLCPCB was able to create and assemble our board but did not have all the necessary chips in stock (most notably the TLVM13610). This led to many problems down the road with soldering as that specific chip was incredibly hard to hand-solder. Waiting for JLCPCB to source this chip would have delayed the initial boards by 2 weeks, giving no time for final changes. Because of this time constraint, the final PCBs were ordered from PCBWay which offered the fastest manufacturing and shipping. The parts, which were all available between DigiKey and Mouser, were sourced independently. The boards and parts were then delivered to WWW Electronics to assemble. Ordering through PCBWay and assembling locally was the most time efficient solution and allowed the final boards to be ready after Thanksgiving.

There were also many software tools used in the production of Romulus. For software development, Visual Studio Code was used extensively. This allowed for code to be run remotely on the Pi from other computers, which was essential to the development process. Fusion 360 was used as the CAD design tool, as well as to create final renders, and perform finite element analysis. This allowed everyone to access the CAD files from the shared project and allowed for a one-stop-shop for anything related to the frame. Both the Ultimaker S3 and S5 were used to print out the CAD files, to slice these files and prepare them for printing we used the Ultimaker Cura software. For circuit and PCB schematics, KiCAD was used. KiCAD was chosen due to familiarity with the program, a plethora of resources and help, and it is free. All component models were compatible with KiCAD as well.

Bringing this prototype to production would require solving many more challenges. To start, the frame would have to be redesigned to be compatible with injection molding manufacturing. This is necessary to reduce cost and achieve a low tolerance final product. Next, the AI model would have to switched to a more practical target, such as other Airsoft players, so that users will have a use for it. The actuators and PCB can stay the same, as simply ordering them in bulk will bring the cost down significantly. Several quality-of-life features are necessary, mainly more UI buttons so that the user can do more than turn Romulus on and off. Ideally there would be ways of changing the AI model on the fly or a way to turn off tracking all together. With these changes Romulus could be brought forth to the market at large.

Social Impact

On the face of it Romulus seems like it would be a threat to the safety of the general public, if it were to be misused with a real firearm. For several reasons, which shall be expounded upon, this is not the case. First and foremost, Romulus was not designed to handle the constraints imposed by real firearms. Although Airsoft guns attempt to mimic firearms, they aren’t entirely accurate, generally being lower weight, and having no recoil. If a firearm were to be attached to Romulus the increased weight would throw off the PD controller for the actuators. This would make Romulus less accurate and unresponsive. The presence of recoil would also be disastrous for Romulus, as it would at least throw off any calibration or tuning and more than likely cause the catastrophic failure of many of the parts. These factors mean that if a real firearm were to be attached to Romulus it would not function nearly as well if at all.

Beyond safety Romulus could have a much broader impact on Airsoft as a whole. Its foremost purpose is to promote accessibility in Airsoft. Airsoft is a very physical sport that demands high levels of hand eye coordination and reaction time, which just aren’t possible for some people. Romulus will allow these people to close the gap in skill and be competitive with other Airsoft players. This could allow people with disabilities or just poor coordination to join the Airsoft community. One downside however is that Romulus is expensive. Even if it were put into mass production, Romulus would still cost on the order of hundreds of dollars. This cost would be prohibitively expensive for many people and could cause a divide in Airsoft players, between those who can afford it and other gadgets and those who can’t. Admittedly Airsoft is already a very expensive sport where members are willing to spend large sums of money, so this is not a likely outcome. Additionally, Romulus is about equivalent to a slightly skilled player, so while this would allow for people to play who otherwise wouldn’t have, it will not overwhelm the dedicated players.

Outside of Airsoft players there are other stakeholders that must be considered. Most critically the Airsoft manufacturers, and other DIYers need to be considered. Assuming the success of Romulus, Airsoft manufacturers would have access to a wider market and increased sales, which would be a benefit to growing their business. Their sales would only increase as Romulus requires a user to already own an Airsoft gun. This means that Romulus users would have to buy Airsoft guns from the preexisting manufacturers. As for the DIY community, Romulus could spark a wave of innovation in the space. Currently it is rare to see complex and novel robotics in the DIY field, due to the large amounts of domain knowledge required to make such devices. Romulus could lower that barrier to entry and allow for more unique and innovative designs from the community.

Lastly in the modern age of global warming, it is critical to consider the environmental impact of Romulus. The first consideration is the impact of Airsoft as the two are intrinsically linked. One would think that with the use of plastic pellets in Airsoft the environmental impact could be catastrophic. However, this is not the case as the vast majority of Airsoft fields require the use of biodegradable pellets and have done so for years. Outside of the pellets Romulus is made of plastic and has many electronics that would cause problems if thrown out. This is pretty much unavoidable due to the design requirements of Romulus. However, the waste can be avoided, by having robust customer support and repair services. In this way users can keep using Romulus for decades down the line without throwing anything away.

Project Costs

The final costs spent on this project can be seen in Appendix B. Notably, we were given a budget of $500 by the University. At the beginning of the project, we agreed as a group that we would be willing to spend some of our personal money to make a great product. The price of everything related to the project totals around $1682.72. However, if we exclude the Airsoft gun and Raspberry Pi that were already owned, the total comes out to $1336.81. This number was higher than expected but we believe it was necessary for the design process.

This price was driven higher by two main sources: the final PCB and extraneous components. The final PCB was created close to the end of the project and three were assembled. This meant that rush shipping was more expensive, and assembly was three times the price of getting one board assembled. This decision was made to ensure multiple backups if the PCB ended up breaking for any reason. Extraneous components include spare parts, the rangefinder, and the Raspberry Pi UPS. These are parts that were unused or ideas that did not pan out.

If this product was brought to scale, the estimated price per unit was calculated and broken down in Appendix C. Prices are based off the bulk (10,000 unit) price. The total price for our product comes to around $321.82. This price does not include the initial startup cost such as injection molds. PCB price was estimated using JLCPCB. The detailed price breakdown for the PCB components sourced from DigiKey can be seen in Appendix D. The frame cost was estimated using the price of ABS plastic for injection molding. Hardware was estimated by surveying various hardware suppliers. Note that bulk prices were not available for the Raspberry Pi 5 and the necessary cameras.

Intellectual Property

There exist many patents with material like our project. However, the scope and application of these patents concern different areas of focus than what we are attempting to accomplish. For instance, a patent filed by Reymond Clavel for a *Device for the movement and positioning of an element in space* describes a device that “enables the control of the three basic degrees of freedom in parallel from actuators arranged on a fixed support” [31]. This device is very similar to the linear actuator system we have implemented. However, the patent claims that the device must comprise of at least three linking means where one of the means is comprised of parallel bars. This differs from our project with respect to the structure of the linear actuator system. In our system, the actuators are directly connected to a base and moving structure rather than having an intermediary connection system by way of parallel bars. The patent’s original application was to not only rotate an element in three-dimensional space but also move the element translationally, adding another degree of freedom that our project does not require nor implement.

Another patent of relevance is *Target recognition system and target recognition method executed by the target recognition system* which was filed by Haike Guan [24]. This patent describes a target recognition system using stereovision that is able to recognize one or more desired targets. The system differs from our implementation with respect to the method of target acquisition. The patent claims that the target recognition method comprises of detecting a candidate set of recognition targets and calculating a range of threshold values for accurately identifying desired targets. This system differs in our use of stereovision and target acquisition where we determine physical distance away from a desired target to then feed into an established object recognition model. Due to the need of only tracking one target, there is no need to create a candidate set.

Lastly an additional patent concerning similar functionality to our project is *Shooting with aim assist*, filed by Philip Theriault [25]. This patent describes a shooting system involving a handheld firearm with targeting and aim assist capabilities. The system was created with the purpose of assisting soldiers in firing at enemy Unmanned Aerial Vehicles (UAV’s) given the frequency of encounters and the difficulty of anticipating their movements. The handheld shooting system uses a portable processing unit and power source similar to our own project. Additionally, the system makes continuous predictions based on external sensors added to the firearm and runs a targeting system to find the “optimal targeting vector” to adjust the user’s aim. The aim is adjusted by a “momentum transfer system” that applies dimensional movement to correct firearm aim. However, unlike our team’s design, this system as claimed by the patent has the linear motion applicators comprise of an electromagnet within a sleeve external to the firearm to displace mass selectively. Our device differs in this respect as it is structured to correct aim by the precise parallel motion of linear actuators.

Although these patents cover similar intentions and implementations, we believe that our project is patentable due to the novelty of our system. Our project implements a more niche approach to portable aim assistance that these patents do not cover along with the combination of multiple subsystems that make a patent desirable.

External Standards

For this project, there will be a few standards involved. First, the picatinny rail that this system interfaces with needs to follow the NATO and Mil-spec standard [26]. This will ensure that this system can work with multiple Airsoft guns as the mounting solution will follow the global standard.

Because our project demonstration will involve lasers, there are different classes of lasers that we need to understand. The safety and classification of lasers defined in ANSI Z136.1-2022 will help us identify what type of laser we should use and all the safety precautions and procedures we should employ while using it [27].

For the manufacturing of the power PCB, we will need to follow IPC-2221A to ensure that our board fits the recommend spec [28].This should help with the thermals of the board overall and give guidance for how wide the traces should be to account for high currents. Lastly, there are safety requirements set by IEC 62133-2:2017 for portable lithium batteries [29]. Meeting these requirements will be incredibly important to keep the user safe as we will be using a lithium battery to power the whole system.

For machine learning, there are only general rules for modelling found in the ISO/IEC-23053 [30]. This should give us a baseline understanding and rules to follow while implementing our image classification model.

Results and Testing

Overall, in the design process of Romulus our testing philosophy can be summarized as: test early, test often, and test upwards. We were adamant in our early conception of the project that we needed to knock out as many systems as possible in parallel and to identify challenges as early as possible. The main electrical systems we used were adapted from manufacturer application notes, which made the transition from conception to design seamless.

Component Testing

One of our most effective design decisions was to breadboard-test our linear servo implementation. To do this, we began with a development board model of the STM32 microcontroller. While our motor driver chip was unavailable in through-hole packages, we were able to get an adaptor PCB that let us test with it [21]. Following a bottom-up strategy we first tried minimal code implementation to get a single linear actuator moving. This proved to be a challenge, as noted elsewhere, this let us identify that the given capacitance values were incorrect to properly supply the chips internal charge pump. To identify these hardware-related issues an approach of following the fault-indication pins and conditions given in the data sheet proved effective. By scoping each potential condition, and ruling each out one-by-one, this issue was diagnosed effectively.

In these early iterations, inefficient code implementations were used and verified before any optimization. Features were also added one at a time and tested after each addition. While tedious this was an effective strategy for troubleshooting and testing. This let us build a single linear actuator PD controller from the ground up. To test it, we had it track to a single initial position repeatedly. Once we demonstrated this was functional, optimizations such as direct memory access and hardware acceleration were used to replace inefficient strategies. We then added the other two motor drivers before proceeding, again validating their independent functionality.

After this design process, communication between the Raspberry Pi and the STM was established in an independent test project from motor control. Due to considerable issues, most notably driver transparency for the Raspberry Pi, this process was tedious. However, by testing independently, we were able to efficiently identify the root of these problems. After establishing communication, the PD controllers for all three motors were simultaneously tested to ensure they effectively and quickly tracked to their target reference positions.

Next, inverse kinematics was implemented. Kinematics were verified analytically in both matplotlib graphing, as well as through empirical testing. One effective strategy was using the input from a game controller's joysticks to control the pan, tilt, and elevation of the robot. This provided both an intuitive way of verifying both its speed and accuracy met our design requirements.

One design aspect we continuously struggled with throughout the project was how to optimally size the platforms of the 3-RPS robot. The ratio sets the effective size of the workspace. That is, the maximum pan and tilt possible. Analytical derivations existed in research but were difficult to implement or seemed unapplicable to our use cases. However, tests such as using a controller did provide some helpful iterative experimentation in our design process. Especially when it came to defining a safe boundary for the workspace to prevent damaging the system. In our first iteration of the platform, overconstraining system did result in fracturing the base platform and several cracks in the plastic-joint interfaces. However, these hardware failures also provided opportunities for our mechanical design to iterate and improve.

Considerable time was devoted to the use of an off-the-shelf rangefinder module (due to cost). However, ultimately it was infeasible to interface with. This led us to a massive design pivot towards stereovision. The testing and validation of stereovision proved to be immensely difficult and sensitive to setup errors. It took an embarrassingly long time to deduce that the camera's built-in focus caused repeated calibration failures. For stereovision to work correctly both cameras should be focused identically and maintain the same focal length as their calibrated value. This was especially difficult to find as the process appeared to be intermittently working. Even despite being very incorrectly configured. This is one role where background research in the design and testing process proved to be crucial. Many of the solutions found came from pursuing a deeper understanding of background research and the subject matter (this also proved to be true for kinematics).

Combined with repeated bottom-up testing and implementation and further background research we were able to get stereovision successfully implemented. One incredibly important test proved to be validation of stereo rectification calibration. To do this, we ensured that the images were correctly horizontally rectified by drawing horizontal lines between objects in each of the two images. Then, with improved calibration, and several testing approaches to extract disparity and distance, we were able to successfully range-find. In fact, during one experimental setup, distance measurements were accurate within under a quarter of an inch at approximately twenty-five feet. Some success was also had generating 3D point-cloud renderings of the testing environment.

PCB

After soldering on the last couple components to the PCB we started testing by doing a visual inspection of each solder joint. Of particular importance were the pins on the voltage regulators. The regulators used a footprint that positioned every pin underneath the package where they couldn't be physically accessed. This added complexity made visual inspection even more critical, as a bad connection would be hard to spot and disastrous for the board. Once every solder connection seemed sound, we used a DC power supply to slowly increase the input voltage while measuring the output of the voltage regulators. This allowed us to spot a short if there was a current spike, or quickly reduce the supply voltage if the regulators output the wrong voltage. Once the voltage regulators output the correct voltage and the board was drawing reasonable current, we moved on to testing the STM.

We started by attaching the power supply to the board so that the STM would be powered. Then we attempted to flash the STM, if the STM could be flashed it was working, however if it couldn't be recognized it needed to be replaced. Lastly, to test the motor drivers we ran a simple program on the STM to move them up and down, this would quickly show if the motor drivers are working correctly. When the final PCB arrived the testing system stayed the same, with the exception that the power supply and STM could now be tested separately. This is because the second PCB utilized a jumper pin to connect the VDD pin of the STM to the voltage regulator. This ensured we couldn't back drive voltage into the voltage regulator while flashing the STM, and that we couldn't burn out the STM while testing the voltage regulator. Using this method, we were able to verify that we had a fully functional PCB.

User Testing

As part of our system design for the frame we wanted Romulus to be comfortable and intuitive to users. To conduct these tests, we handed the frame to users without any instructions and observed how they handled it, as seen in Figure 45.

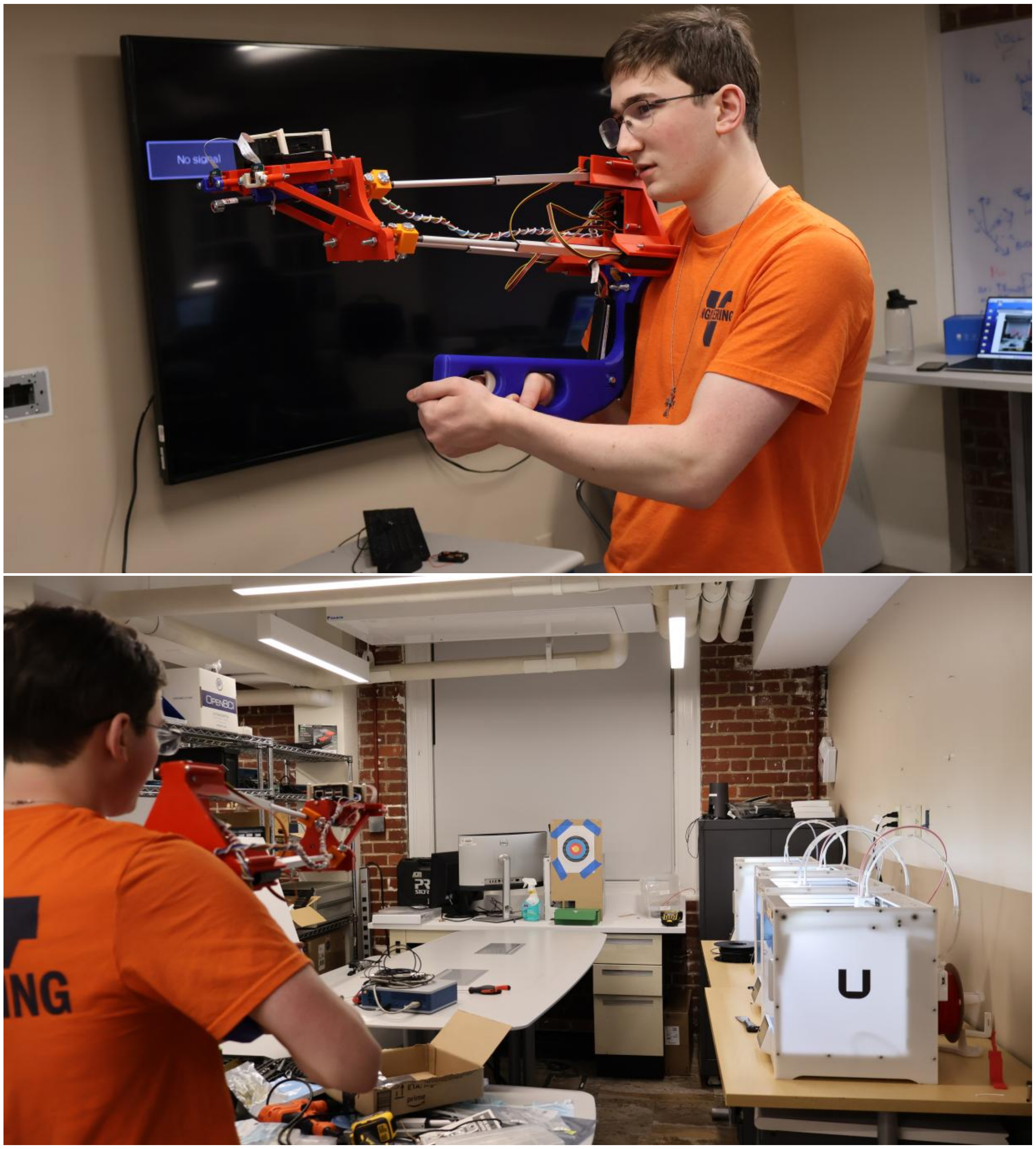
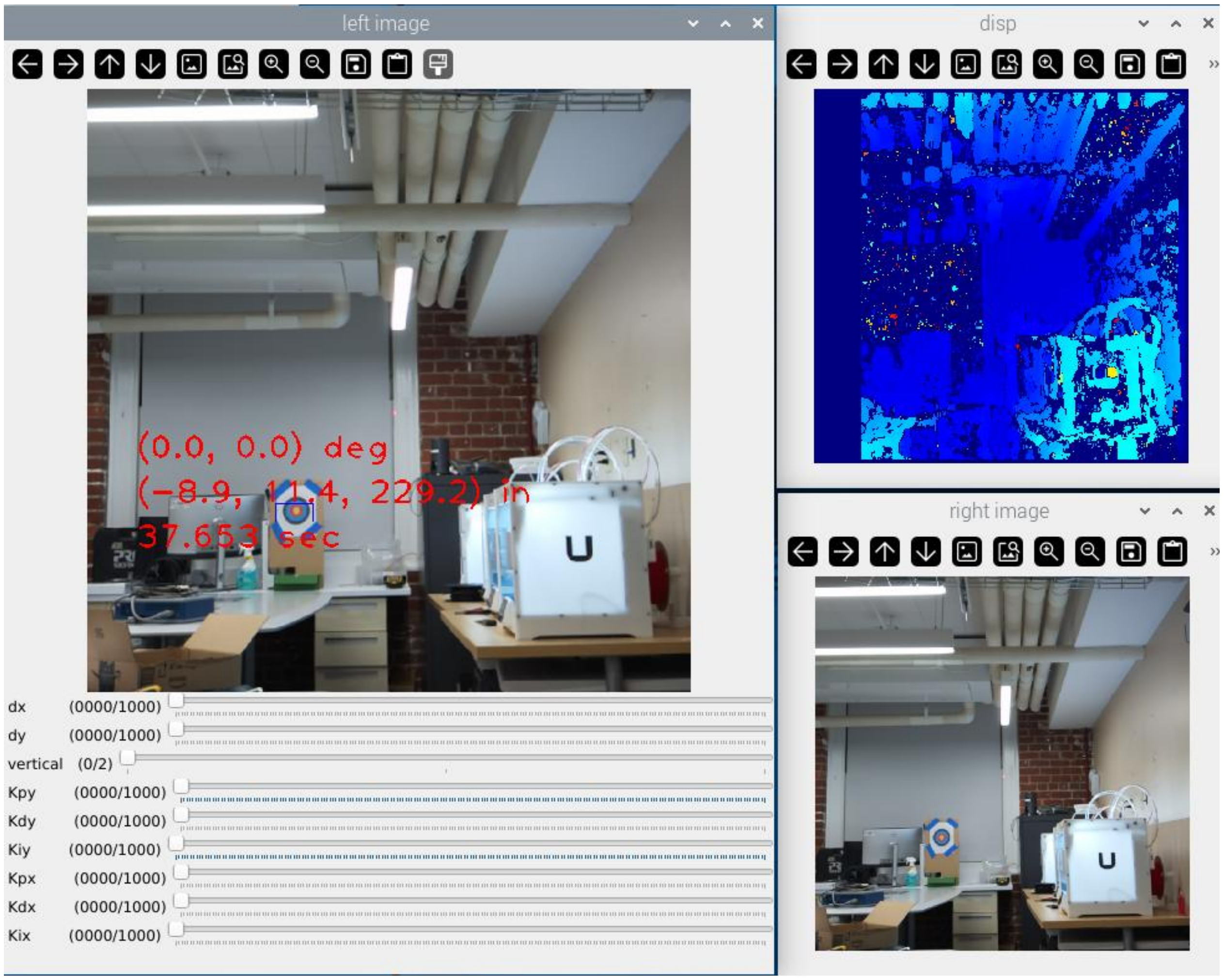


Figure 45. User testing with fully operational Romulus.

We found that most users who were experienced with firearms handled it intuitively, for those who weren’t they didn’t immediately know where to put their hands on the handle but picked it up with some guidance. Universally users found the frame comfortable to hold once they figured out how to. During one of the early on testing phases the linkage of the frame between the handle and the base snapped. This was very helpful as it drove us to increase the strength of this part, and it has held up fine since.

Full System Testing

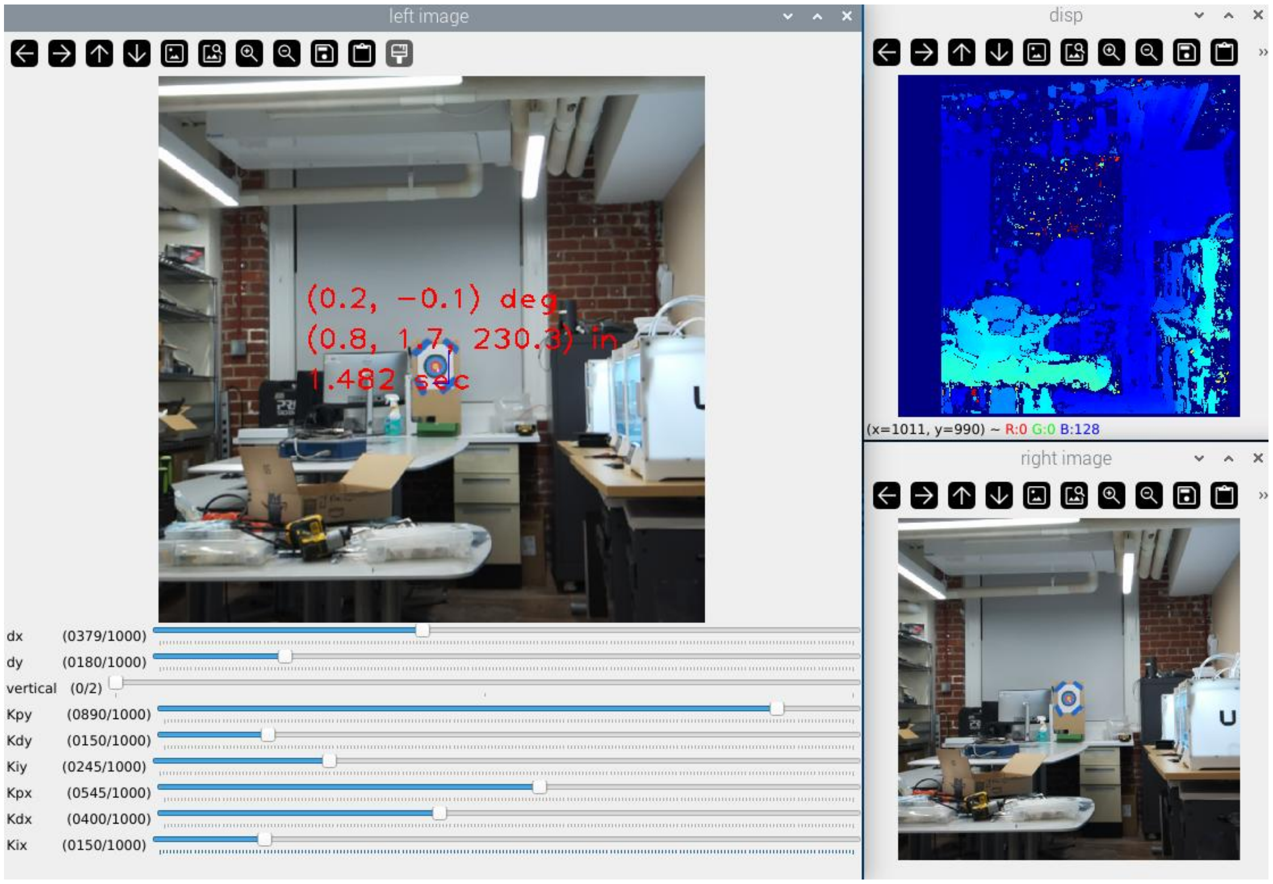
In the proposal we outline a primary metric of success: to at minimum track to a stationary target given an initial displacement. This was undisputedly successful. Our approach to *tune* the PID controller was on its own, completion of this goal. To validate the tracking speed, a testbench program was written based on our tuner. An initial horizontal and vertical error were introduced. Then, the time to track and stabilize on the target was measured, see Figure 46 and Figure 47.



```
(0.0, 0.0) deg
(-8.9, 11.4, 229.2) in
37.653 sec
```

```
dx        (0000/1000)
dy        (0000/1000)
vertical  (0/2)
Kpy       (0000/1000)
Kdy       (0000/1000)
Kiy       (0000/1000)
Kpx       (0000/1000)
Kdx       (0000/1000)
Kix       (0000/1000)
```

Figure 46. A result showing the measurement setup used to calculate tracking time for a static target. Notice the laser pointer in the left image pointed at a roughly equal horizontal and vertical distance from the target. This interface was based on the same used for efficiently tuning PID parameters, which are the sliders in the bottom of the left image window. Also Shown in the left window is a bounding box appearing over the detected target. Note the relative angle observed at as the top annotation in degrees. Below are the calculated (and verified) cartesian distances (r_x, r_y, r_z) measured in inches. Above these distances are the relative angles output by the PID controller. Note that they are zero because the gains were nulled (as can be seen by the sliders). The vertical window switched between both axes, vertical only, and horizontal only. The slider for dx and dy are correction factors used to account for differences in the robot's geometry and the coordinate bases for inverse kinematics. The upper right image is a disparity map, where closer objects have warmer hues. The lower right image is the live camera feed.



```
(0.2, −0.1) deg
(0.8, 1.7, 230.3) in
1.482 sec
```

```
(x=1011, y=990) ~ R:0 G:0 B:128
```

```
dx        (0379/1000)
dy        (0180/1000)
vertical  (0/2)
Kpy       (0890/1000)
Kdy       (0150/1000)
Kiy       (0245/1000)
Kpx       (0545/1000)
Kdx       (0400/1000)
Kix       (0150/1000)
```

Figure 47. The tracking performance results of a tuned PID controller implementation. Note the measured time to track the target and fully stabilized on a target, based on the initial displacement from above, was observed to be 1.482 seconds. This test was performed just under twenty feet away from the target (as can be seen from the r_z coordinate of 230 inches)

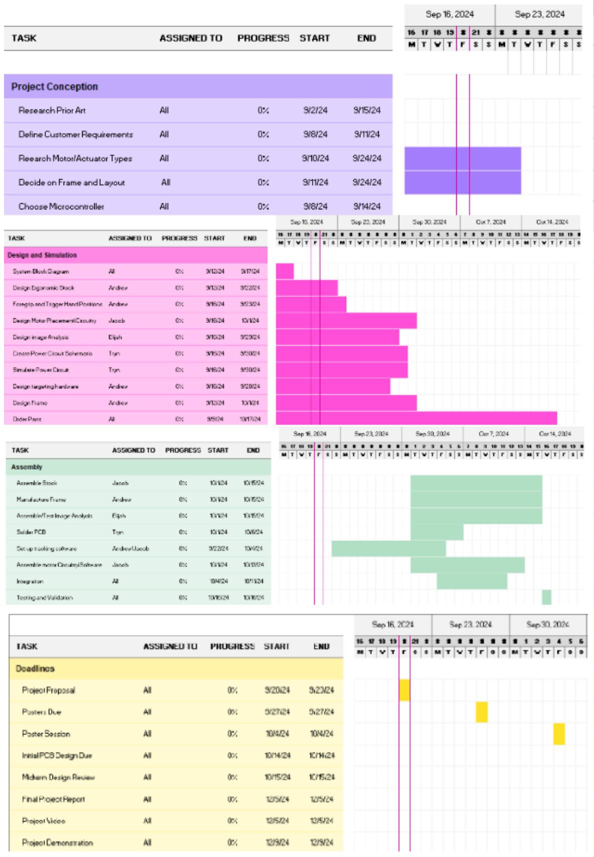
Additionally, a reach goal defined in our proposal to track a moving target. While we did not create a quantitative metric to evaluate this. From our testing, the system was able to track a slow-moving target. The primary limitation was that our control approach was purely reactive. Therefore, over some threshold of speed, the processing and input delay of the system causes it to lag significantly behind a moving target.

Overall, we were able to achieve all our initial expectations as outlined in the proposal. The system was successfully able to track a target, which ultimately for our testing was a small target on printer-paper. Our original expectation was to use a larger target for testing. Therefore, we were quite pleased to successfully track and stabilize within the one-inch yellow circle of the target at a range of twenty feet in approximately 1.5 seconds. Additionally, in use we were found that the batteries lasted for over two hours continuously. This is much longer than we expected and blows our goal out of the water.

Furthermore, we ultimately had deliverables that we never anticipated at the outset of this project. In our initial proposal ranging was to be done using a laser rangefinder, which we found to be a costly and opaque system. Instead, we pivoted to stereovision which was both considerably cheaper and more accurate. The laser rangefinder was only accurate to within ten centimeters, whereas we achieved an accuracy of a quarter inch at twenty-five feet. Additionally, our distance estimates can extract the position of *all* objects in line of sight, rather than the single object detection potential of a laser rangefinder (which would have needed to pass over the target prior to detecting its range). Therefore, we were quite pleased with these results.

Unfortunately, the main limitation in our results was any concrete testing with the Airsoft gun. One feature we proposed to account for is the drop of an Airsoft pellet with distance. The math was implemented to perform these correction factors. However, no final testing was ever performed to do this validation.

Project Timeline

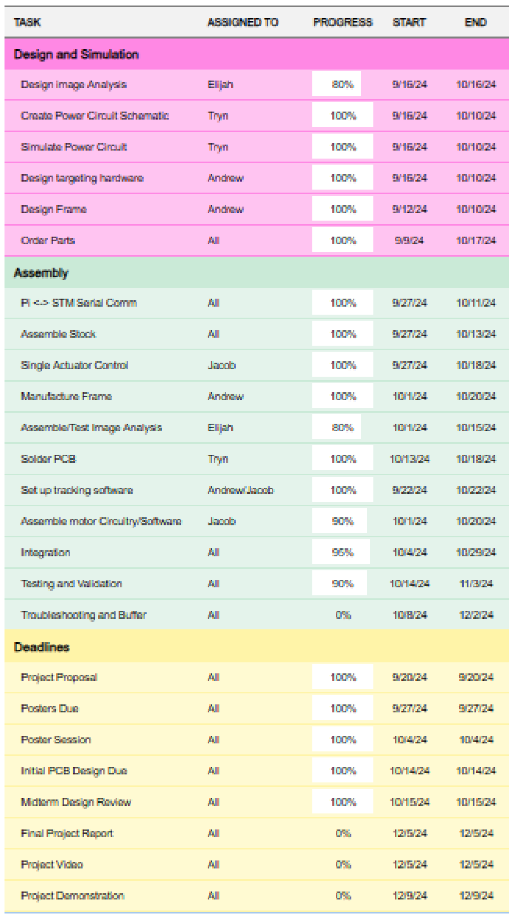


TASK	ASSIGNED TO	PROGRESS	START	END
Project Conception				
Research Prior Art	All	0%	9/2/24	9/15/24
Define Customer Requirements	All	0%	9/8/24	9/11/24
Reearch Motor/Actuator Types	All	0%	9/10/24	9/24/24
Decide on Frame and Layout	All	0%	9/11/24	9/24/24
Choose Microcontroller	All	0%	9/8/24	9/14/24

TASK	ASSIGNED TO	PROGRESS	START	END
Design and Simulation				
System Block Diagram	All	0%	9/12/24	9/17/24
Design Ergonomic Stock	Andrew	0%	9/12/24	9/22/24
Foregrip and Trigger Hand Positions	Andrew	0%	9/16/24	9/23/24
Design Motor Placement/Circuitry	Jacob	0%	9/16/24	10/1/24
Design image Analysis	Elijah	0%	9/16/24	9/29/24
Create Power Circuit Schematic	Tryn	0%	9/16/24	9/30/24
Simulate Power Circuit	Tryn	0%	9/16/24	9/30/24
Design targeting hardware	Andrew	0%	9/16/24	9/28/24
Design Frame	Andrew	0%	9/12/24	10/1/24
Order Parts	All	0%	9/9/24	10/17/24

TASK	ASSIGNED TO	PROGRESS	START	END
Assembly				
Assemble Stock	Jacob	0%	10/1/24	10/15/24
Manufacture Frame	Andrew	0%	10/1/24	10/15/24
Assemble/Test Image Analysis	Elijah	0%	10/1/24	10/15/24
Solder PCB	Tryn	0%	10/1/24	10/6/24
Set up tracking software	Andrew/Jacob	0%	9/22/24	10/4/24
Assemble motor Circuitry/Software	Jacob	0%	10/1/24	10/13/24
Integration	All	0%	10/4/24	10/11/24
Testing and Validation	All	0%	10/16/24	10/16/24

TASK	ASSIGNED TO	PROGRESS	START	END
Deadlines				
Project Proposal	All	0%	9/20/24	9/20/24
Posters Due	All	0%	9/27/24	9/27/24
Poster Session	All	0%	10/4/24	10/4/24
Initial PCB Design Due	All	0%	10/14/24	10/14/24
Midterm Design Review	All	0%	10/15/24	10/15/24
Final Project Report	All	0%	12/5/24	12/5/24
Project Video	All	0%	12/5/24	12/5/24
Project Demonstration	All	0%	12/9/24	12/9/24



TASK	ASSIGNED TO	PROGRESS	START	END
Design and Simulation				
Design image Analysis	Elijah	80%	9/16/24	10/16/24
Create Power Circuit Schematic	Tryn	100%	9/16/24	10/10/24
Simulate Power Circuit	Tryn	100%	9/16/24	10/10/24
Design targeting hardware	Andrew	100%	9/16/24	10/10/24
Design Frame	Andrew	100%	9/12/24	10/10/24
Order Parts	All	100%	9/9/24	10/17/24
Assembly				
Pi <-> STM Serial Comm	All	100%	9/27/24	10/11/24
Assemble Stock	All	100%	9/27/24	10/13/24
Single Actuator Control	Jacob	100%	9/27/24	10/18/24
Manufacture Frame	Andrew	100%	10/1/24	10/20/24
Assemble/Test Image Analysis	Elijah	80%	10/1/24	10/15/24
Solder PCB	Tryn	100%	10/13/24	10/18/24
Set up tracking software	Andrew/Jacob	100%	9/22/24	10/22/24
Assemble motor Circuitry/Software	Jacob	90%	10/1/24	10/20/24
Integration	All	95%	10/4/24	10/29/24
Testing and Validation	All	90%	10/14/24	11/3/24
Troubleshooting and Buffer	All	0%	10/8/24	12/2/24
Deadlines				
Project Proposal	All	100%	9/20/24	9/20/24
Posters Due	All	100%	9/27/24	9/27/24
Poster Session	All	100%	10/4/24	10/4/24
Initial PCB Design Due	All	100%	10/14/24	10/14/24
Midterm Design Review	All	100%	10/15/24	10/15/24
Final Project Report	All	0%	12/5/24	12/5/24
Project Video	All	0%	12/5/24	12/5/24
Project Demonstration	All	0%	12/9/24	12/9/24

Figure 48. Original Gantt Chart Timeline (left) vs Final Gantt Chart Timeline.

Reflection on Project

There were two main sources of improvement over the course of the semester that would have made our project significantly easier. The first is time management. It took us a long time to get rolling on the project. A large part of this is having to switch to this project at the last minute but another larger part is that we spent too much time in the beginning focusing on logistics deliverables rather than actual progress. If we could restart, we would have started immediately and rush ordered everything, it took almost two weeks from ordering the first set of parts to receiving them. By the end of the semester, we went for express shipping over all else, in a world where we did this from the beginning we would have had several weeks of extra time.

The second source of improvement is communication. At the start of the project, we didn’t do a good job of clearly and evenly dividing up the work. What ended up happening is several group members picking up slack without saying anything. Ideally, we would have come together, mentioned the work that needed doing and divvied it up evenly. Instead, the person who noticed whatever problem they found took up the extra work without comment. This led to an uneven workload that could have been avoided. Additionally, one of the group members consistently underperformed and the work that they weren’t getting done also had to be picked up. With better communication both between each other about workloads, and with our advisor about the amount of work accomplished, we would have been much more effective.

Future Work

In a perfect world, without monetary or time constraints we would redesign Romulus to be constructed using injection molding. This would allow us to use stronger, higher quality plastics with much tighter tolerances. It would also lower the price of production and facilitate mass production. We would also like to add more customizability into the design. Currently only the front handhold can be extended, in final production we would like to make a stock to add further customizability.

With the enhanced strength from injection molding and being able to do accurate FEA we would also be able to perform testing with actual Airsoft guns. As is there is no way to know if it will break or not until it does. With injection molding, the frame could be modeled properly, and we could have confidence in it holding up to the pressures from actual use.

We would also like to find a better way of attaching the remote trigger to the Airsoft gun. In its current state Velcro works but it loosens overtime. This means it needs to be readjusted periodically to continue to function. With more time We could look into more permanent and secure methods of affixing the trigger to the Airsoft gun.

There is so much to be said about improving the robotics and control aspects of this project. Given more time, the first step would be to optimize and identify bottlenecks in the live processing of image classification and stereovision. By removing processing time, the tracking performance can easily be improved. A simple, but large addition would be the inclusion of parallel processing wherever possible in the image processing pipeline. For instance, by handling stereovision on a separate core from image classification. Another massive optimization that can be made is pre-cropping images wherever possible prior remove unnecessary information prior to image processing. A robust approach that maximizes information loss with efficiency could dramatically improve the speed of these processes.

As mentioned in previous sections, while some effort was dedicated to planning the robot's workspace, we would have liked to dedicate substantially more time to this. While analytical methods exist, they often are difficult to translate into real-world situations. One proposed solution that we came up with, but did not have time to implement, was an empirical planning method. In this approach the current is continuously monitored using Romulus' motor drivers. These values are reported over SPI to the Raspberry Pi. The robot then tries to move as much as possible based on inverse kinematics, but after the current exceeds a certain thresholding (indicating that an actuator is stalling against a restrictive joint), the edges are saved. Additionally, by placing the bottom platform on a set of sliding rails, we could repeatedly alter platform sizes and create these measurements.

In terms of improved controls, a state-space model that accounts for system dynamics would be desirable to simple PID controls. Ideally, we could implement predictive tracking to estimate the motion of a moving target and intercept its trajectory preemptively. By combining this with removing unneeded processing delay, we believe we could *greatly* improve the processing delay in the system.

An idea to make our system able to withstand more weight would be to add a center beam connecting the two platforms. This would get rid of the translational degree of freedom but increase the structural stability of the system. The translational degree of freedom is not necessary for our design, meaning this would be a net boon to structural stability.

One of the biggest headaches we ran into with the PCB was the lack of power protection circuitry on the PCB. This led to multiple chips being fried, and while the final design did have improved protection, there is likely much more work to be done to make the circuitry more robust. One of the key examples would be to use diodes to prevent back-current or overvolting. This would have prevented the STM and DC-DC power converters from frying.

Another potential change is the DC-DC power converters. The TLVM13610 chips are very versatile chips, however, they are likely overkill and expensive. This component can be replaced with a cheaper DC-DC converter.

Statement of Work

<u>Jacob Cochran</u>

Overall, throughout this project I had a large role in the initial project conception, integration, and implementation. I was responsible for much of the background research used during project conception. Specifically, I performed much of background research in robotic design and control implementations. Ultimately, I was responsible for choosing the 3-RPS robot structure, as well as choosing us much of our initial system-level architecture. I chose our specific H-Bridge design and several of the chips for our linear servomotor implementation. I was responsible for breadboarding, our servomotor control algorithms, implementing inverse kinematics, stereovision, tracking, and all other software written for the project.

<u>Tryn Dunne</u>

My primary responsibility was to design and manufacture the electronic hardware of Romulus including schematics and printed circuit boards. This involved first coordinating design decisions that were already created such as the linear actuators. The PCB schematic was then created which involved laying out an STM microcontroller, three motor drivers circuits, and the power circuitry. Two boards were then laid out and manufactured, the initial and final designs. I coordinated communication with each company including the creation of assembly files like the BOM and assembly drawings. Making sure everything functioned correctly with the boards was my responsibility.

My secondary responsibilities were to help Elijah create the image recognition model, including providing feedback to make the model more accurate. I also helped Jacob as an extra hand whenever needed, such as for calibration or proof-reading equations.

<u>Andrew Morrison</u>

My main responsibility was the design and manufacture of every physical part of Romulus. Anything that was 3d-printed I designed and printed myself. Every piece of physical hardware I selected, every bolt and nut. Additionally, I handled any soldering or repair necessary for the PCB. This started with soldering the motor drivers to the initial breakout board for prototyping. For the final PCB this involved just through-hole components and making custom Molex wires. For the initial PCB I had to solder in the voltage regulators, and desolder then resolder the STM whenever it failed.

On the electrical side of things, I designed the trigger and attempted to interface with the rangefinder when we were still planning on using it. I also helped Tryn and Jacob when they needed it and helped bounce ideas off of each other.

Elijah Smith

My primary responsibility was the building of the object detection model. I performed data acquisition of target images and drew bounding boxes for model training. I created multiple versions of a working model for use of experimentation and implementation on the finished system.

My secondary responsibilities included much smaller and menial tasks. These included contacting different facilities early in project development for use of on-site testing and acquiring targets for in-person and demonstration use.

References

[1] 3DprintedLife, *I Made an Auto-Aiming Nerf Blaster - It CAN'T Miss*, (Oct. 19, 2020). Accessed: Dec. 06, 2024. [Online Video]. Available: https://www.youtube.com/watch?v=GoZubkaNvcE

[2] Excessive Overkill, *I Made Real-life Airsoft AIM-ASSIST: Aimbot V3*, (Apr. 06, 2024). Accessed: Nov. 07, 2024. [Online Video]. Available: https://www.youtube.com/watch?v=miRnNy7ZvIc

[3] C. Gosselin and J. Angeles, "Singularity Analysis of Closed-Loop Kinematic Chains," *Robot. Autom. IEEE Trans. On*, vol. 6, pp. 281–290, Jul. 1990, doi: 10.1109/70.56660.

[4] K.-M. Lee and D. K. Shah, "Dynamic analysis of a three-degrees-of-freedom in-parallel actuated manipulator," *IEEE J. Robot. Autom.*, vol. 4, no. 3, pp. 361–367, Jun. 1988, doi: 10.1109/56.797.

[5] S. K. I. Metodij, "COMPARISON OF THE CHARACTERISTICS BETWEEN SERIAL AND PARALLEL ROBOTS".

[6] H. D. Taghirad, *Parallel Robots: Mechanics and Control*. CRC Press, 2013.

[7] S. Parasuraman and P. C. J. Liang, "Development of RPS Parallel Manipulators," in *2010 Second International Conference on Computer and Network Technology*, Apr. 2010, pp. 600–605. doi: 10.1109/ICCNT.2010.123.

[8] A. Sokolov and P. Xirouchakis, "Kinematics of a 3-DOF parallel manipulator with an R-P-S joint structure," *Robotica*, vol. 23, no. 2, pp. 207–217, Mar. 2005, doi: 10.1017/S0263574704000773.

[9] K.-M. Lee and D. K. Shah, “Kinematic analysis of a three-degrees-of-freedom in-parallel actuated manipulator,” *IEEE J. Robot. Autom.*, vol. 4, no. 3, pp. 354–360, Jun. 1988, doi: 10.1109/56.796.

[10] A. Nayak, T. Stigger, M. L. Husty, P. Wenger, and S. Caro, “Operation mode analysis of 3-RPS parallel manipulators based on their design parameters,” *Comput. Aided Geom. Des.*, vol. 63, pp. 122–134, Jul. 2018, doi: 10.1016/j.cagd.2018.05.003.

[11] S. Maraje, L. Nurahmi, and S. Caro, “Operation modes comparison of a reconfigurable 3-PRS parallel manipulator based on kinematic performance,” in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers, 2016, p. V05BT07A055. Accessed: Dec. 04, 2024. [Online]. Available: https://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-abstract/IDETC-CIE2016/V05BT07A055/259253

[12] P. S. Rao and N. M. Rao, “POSITION ANALYSIS OF SPATIAL 3-RPS PARALLEL MANIPULATOR,” 2013.

[13] K. Sadekar, “Introduction to Epipolar Geometry and Stereo Vision | LearnOpenCV #.” Accessed: Dec. 05, 2024. [Online]. Available: https://learnopencv.com/introduction-to-epipolar-geometry-and-stereo-vision/

[14] J. Lee, “B0471 Datasheet.” Arducam, Aug. 17, 2023. [Online]. Available: https://www.arducam.com/downloads/datasheet/B0471_16MP_IMX519_USB3.0_Camera_Datasheet.pdf

[15] K. Sadekar, “Stereo Camera Depth Estimation With OpenCV (Python/C++).” Accessed: Dec. 05, 2024. [Online]. Available: https://learnopencv.com/depth-perception-using-stereo-camera-python-c/

[16] K. Sadekar, “Camera Calibration using OpenCV | LearnOpenCV #.” Accessed: Dec. 05, 2024. [Online]. Available: https://learnopencv.com/camera-calibration-using-opencv/

[17] P. Lafiosca and M. Ceccaroni, “Rectifying Homographies for Stereo Vision: Analytical Solution for Minimal Distortion,” in *Intelligent Computing*, K. Arai, Ed., Cham: Springer International Publishing, 2022, pp. 484–503. doi: 10.1007/978-3-031-10464-0_33.

[18] decadenza, *decadenza/SimpleStereo*. (Dec. 04, 2024). Python. Accessed: Dec. 05, 2024. [Online]. Available: https://github.com/decadenza/SimpleStereo

[19] “9.3: PID Tuning via Classical Methods,” Engineering LibreTexts. Accessed: Dec. 06, 2024. [Online]. Available: https://eng.libretexts.org/Bookshelves/Industrial_and_Systems_Engineering/Chemical_Process_Dynamics_and_Controls_(Woolf)/09%3A_Proportional-Integral-Derivative_(PID)_Control/9.03%3A_PID_Tuning_via_Classical_Methods

[20] “Actuonix P16 Datasheet.” Actuonix Motion Devices Inc., 2016. [Online]. Available: https://www.actuonix.com/assets/images/datasheets/ActuonixP16Datasheet.pdf

[21] “DRV8874-Q1 H-Bridge Motor Driver With Integrated Current Sense and Regulation.” Texas Instruments Inc., Jan. 2021. [Online]. Available: https://www.ti.com/lit/ds/symlink/drv8874-q1.pdf

[22] “STM32G071x8/xB.” STMicroelectronics, Nov. 2018. [Online]. Available: https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/726/STM32G071_Rev1_DS.pdf

[23] “TLVM13610 Datasheet.” Texas Instruments, 2023. [Online]. Available: https://ti.com/lit/ds/symlink/tlvm13610.pdf

[24] H. Guan, “Target recognition system and target recognition method executed by the target recognition system,” US20130322691A1, Dec. 05, 2013 Accessed: Dec. 06, 2024. [Online]. Available: https://patents.google.com/patent/US20130322691A1/en

[25] P. Theriault, “Shooting system with aim assist,” US9612088B2, Apr. 04, 2017 Accessed: Dec. 06, 2024. [Online]. Available: https://patents.google.com/patent/US9612088B2/en

[26] “MIL-STD-1913 DIMENSIONING ACCESSORY MOUNTING RAIL SMALL.” Accessed: Dec. 06, 2024. [Online]. Available: http://everyspec.com/MIL-STD/MIL-STD-1800-1999/MIL-STD-1913_17705/

[27] *ANSI Z136.1*, 2022. [Online]. Available: https://assets.lia.org/s3fs-public/pdf/ansi-standards/samples/Sample%20-%20Z136.1%20-2022-Digital.pdf

[28] *IPC-2221A*, 1998. [Online]. Available: https://www-eng.lbl.gov/~shuman/NEXT/CURRENT_DESIGN/TP/MATERIALS/IPC-2221A(L).pdf

[29] “Standard | IECEE.” Accessed: Dec. 06, 2024. [Online]. Available: https://www.iecee.org/certification/iec-standards/iec-62133-22017

[30] *ISO/IEC 23053*, 2022. [Online]. Available: https://cdn.standards.iteh.ai/samples/74438/dc54208373d643c191a657cf5eed9eaf/ISO-IEC-23053-2022.pdf

[31] R. Clavel, “Device for the movement and positioning of an element in space,” US4976582A, Dec. 11, 1990 Accessed: Dec. 06, 2024. [Online]. Available: https://patents.google.com/patent/US4976582A/en

Appendix

Appendix A. Final PCB BOM.

Qty	Reference	Value	Manufacturer's Part Number	Manufacturer	DNI	Description
2	C1,C8	100u	UUD1C101MCL1GS	Nichicon		100uF 16V SMD Electrolytic Capacitor
5	C2,C3,C9,C10,C19	10u	CL21B106KOQNNNF	Samsung electro-mechanics		16V 10uF X7R 0805 Ceramic Capacitor
1	C4	22p	06035C220JAT2A	Kyocera AVX		22pF Ceramic 0603 Capacitor X7R
7	C5,C6,C7,C12,C13,C14,C15	47u	GRM32ER71A476ME15L	Murata Electronics		47uF X7R 10V 1210 Capacitors
1	C11	47p	06035C470KAT2A	Kyocera AVX		47pF Ceramic 0603 Capacitor X7R
4	C16,C18,C20,C21	100n	GRM155R71C104KA88J	Murata Electronics		0.1uF Ceramic 16V X7R 0402 Capacitor
1	C17	1u	CL10A105KO8NNNC	Samsung electro-mechanics		1uF Ceramic 16V 0603 Capacitors
3	C22,C26,C30	2n2	GRM033R71C222KA88D	Murata Electronics		2.2nF 16V X7R 0201 Ceramic Capacitor
6	C23,C24,C27,C28,C31,C32	0.1u	GRM155R71C104KA88J	Murata Electronics		0.1uF Ceramic 16V X7R 0402 Capacitor

3	C25,C29,C33	DNI	EEU-FR1C102LB	Panasonic Electronic Components	DNI	470uF 16V capacitor
1	FB1	FerriteBead	MI0805K601R-10	Laird-Signal Integrity Products		Ferrite bead
1	J1	DNI	705430001	Molex	DNI	
1	J2	DNI	705430001	Molex	DNI	
1	J3	DNI	705430109	Molex	DNI	
1	J4	DNI	705430109	Molex	DNI	
1	J5	DNI	705430109	Molex	DNI	
1	J6	DNI	705430109	Molex	DNI	Generic connector, single row, 01x05, script generated (kicad-library-utils/schlib/autogen/connector/)
1	J7	DNI	PREC040SAAN-RC	Sullins Connector Solutions	DNI	Generic 2.54mm Pin Headers
1	J8	DNI	PREC040SAAN-RC	Sullins Connector Solutions	DNI	Generic 2.54mm Pin Headers
1	J9	DNI	PREC040SAAN-RC	Sullins Connector Solutions	DNI	Generic connector, single row, 01x02, script generated (kicad-library-utils/schlib/autogen/connector/)
1	J10	DNI	PREC040SAAN-RC	Sullins Connector Solutions	DNI	Generic connector, single row, 01x02, script generated (kicad-library-utils/schlib/autogen/connector/)
1	J11	DNI	PREC040SAAN-RC	Sullins Connector Solutions	DNI	Generic 2.54mm Pin Headers
2	J12,J13	DNI	XT60-M	AMASS	DNI	XT60-M Connector
2	R1,R7	187k	AC0402FR-07187KL	Yageo		187k +/-1% 0402 chip resistor
2	R2,R8	49k9	AC0402FR-0749K9L	Yageo		49.9k +/-1% 0402 chip resistor
4	R3,R5,R9,R11	100k	AC0402FR-07100KL	Yageo		100k +/-1% 0402 chip resistor

4	R4,R14,R18,R22	16k	AC0402FR-0716KL	Yageo		16k +/-1% 0402 resistor
1	R6	24k9	AC0402FR-0724K9L	Yageo		24.9k +/-1% 0402 chip resistor
1	R10	20k	AC0402FR-0720KL	Yageo		20k +/-1% 0402 resistor
1	R12	43k	AC0402FR-0743KL	Yageo		43k +/-1% 0402 chip resistor
4	R13,R16,R20,R24	10k	AC0402FR-0710KL	Yageo		10k +/-1% 0402 chip resistor
3	R15,R19,R23	50k	CRCW040250K0FKED	Vishay Dale		50k +/-1% 0402 resistor
3	R17,R21,R25	5k6	AC0402FR-075K6L	Yageo		5.6k +/-1% 0402 chip resistor
2	U1,U2	TLVM13610RDFR	TLVM13610RDFR	Texas Instruments		Power Converter
1	U3	STM32G071RBT6	STM32G071RBT6	STMicroelectronics		STM MCU
3	U4,U5,U6	DRV8874QPWPRQ1	DRV8874QPWPRQ1	Texas Instruments		Motor Driver

Appendix B. Final Cost Tabulation.

Description	Price
Final PCB Components Digikey	$149.69
Final PCB Components Mouser	$47.95
Final PCB Fabrication	$94.75
Final PCB Assembly	$154.50
Prototype PCBs	$152.44
Breadboard Components	$84.97
Airsoft Gun	$189.00
Magazine and Speedloader	$27.94
Raspberry Pi5	$128.99
Camera Cables	$16.98
XH2.54 Cable Inserts	$6.99
UPS Hat + Batteries	$57.88
Linear Actuators	$286.95
Arducam #2	$24.99
Hardware & charger	$53.00
Rangefinder	$62.64
Hardware	$19.88
Emergency Components	$74.20
Mouser Initial Parts	$49.00
Total Cost	$1,682.74
Prior Ownership	$345.93
Actual Cost	$1,336.81

Appendix C. Scaled Cost of Romulus.

Quantity	Part	Bulk Cost	Total Cost
3	Linear Actuators	$50.00	$150.00
1	Frame	$2.15	$2.15
1	PCB Components	$24.09	$24.09
1	PCB Fabrication	$0.23	$0.23
1	PCB Assembly	$0.36	$0.36
1	Raspberry Pi	$80.00	$80.00
2	Arducam 16MP	$25.00	$50.00
1	Hardware	$15.00	$15.00
	TOTAL COST	N/A	$321.82

Appendix D. Scaled Cost of Electrical Components.

Qty	Manufacturer's Part Number	Manufacturer	10k Bulk Price	Price per PCB
2	UUD1C101MCL1GS	Nichicon	$0.0596	$0.1192
5	CL21B106KOQNNNF	Samsung electro-mechanics	$0.0420	$0.2100
1	06035C220JAT2A	Kyocera AVX	$0.0134	$0.0134
7	GRM32ER71A476ME15L	Murata Electronics	$0.2750	$1.9252
1	06035C470KAT2A	Kyocera AVX	$0.0333	$0.0333
4	GRM155R71C104KA88J	Murata Electronics	$0.0040	$0.0160
1	CL10A105KO8NNNC	Samsung electro-mechanics	$0.0040	$0.0040
3	GRM033R71C222KA88D	Murata Electronics	$0.0030	$0.0089
6	GRM155R71C104KA88J	Murata Electronics	$0.0040	$0.0240
3	EEU-FR1C102LB	Panasonic Electronic Components	$0.1767	$0.5302
1	MI0805K601R-10	Laird-Signal Integrity Products	$0.0538	$0.0538
2	705430001	Molex	$0.4076	$0.8152
4	705430109	Molex	$0.6272	$2.5086
1	PREC040SAAN-RC	Sullins Connector Solutions	$0.2574	$0.2574
2	PRT-10474	SparkFun	$1.6000	$3.2000
2	AC0402FR-07187KL	Yageo	$0.0020	$0.0041
2	AC0402FR-0749K9L	Yageo	$0.0020	$0.0041
4	AC0402FR-07100KL	Yageo	$0.0031	$0.0122
4	AC0402FR-0716KL	Yageo	$0.0020	$0.0082
1	AC0402FR-0724K9L	Yageo	$0.0020	$0.0020
1	AC0402FR-0720KL	Yageo	$0.0031	$0.0031
1	AC0402FR-0743KL	Yageo	$0.0031	$0.0031
4	AC0402FR-0710KL	Yageo	$0.0020	$0.0082
3	CRCW040250K0FKED	Vishay Dale	$0.0080	$0.0239
3	AC0402FR-075K6L	Yageo	$0.0020	$0.0061
2	TLVM13610RDFR	Texas Instruments	$3.9951	$7.9902
1	STM32G071RBT6	STMicroelectronics	$1.9646	$1.9646
3	DRV8874QPWPRQ1	Texas Instruments	$1.4480	$4.3441