Final Project

Team 3
Joseph Page, Joshua Silva, Natalia Jenkins, Garrison Veith
04/28/2025

Executive Summary

In this project, we developed an autonomous putt-putt golf system designed to sink a golf ball placed by a user anywhere along a custom board.The system integrates a physical putting mechanism, a camera subsystem for ball detection, and a Graphical User Interface(GUI) that allows the user to interact with the system. Our work involved coordinating the mechanical, software, and computer vision components to function seamlessly as one system. Throughout the project, we each contributed to several different subsystems while learning the value of clear communication to drive the project forward.

Engineering Requirements

- Camera subsystem must identify golf ball and hole locations to an accuracy of 98%
- Device must be able to deliver enough force to hit a ball to these distances 95% of the time
- Force should be variable to not overshoot or undershoot given changes in distance
- System must be able to correctly distinguish between three holes for use in calculations of positioning system with 95% accuracy
- User should have an easy interface to be able to choose which hole they want to sink the ball in
- System start button should be able to be pressed once the user has chosen the hole
- Receive a notification of completion and run-time information pop-up once striking motion has been completed
- Overall system('putter') should fit within 2.5x1 foot area
- System should enter into work space no more than 2"
- System should not exceed 70 dB
- Maximum power consumption should not exceed 15W
- Upon request from user, the system should move to desired location and strike the ball within 30 seconds
- Motors, gears and anything else similar should be covered or out of easy reach
- interfaces should be easily understood
- Use PID based closed loop control for accurate motor positioning within 3mm
- Must implement DC motor, Encoder, Camera

Design Details

The system, in its whole, is designed for Automated Putt Putt. The main parameters are that the user is to place and to retrieve the ball, but that the system is able to automate the hitting process and allow the user to select the golf ball and golf hall that they want through a GUI. To implement this design, several subsystems are necessary in order for components to receive the necessary data to effectively aim and hit the ball into the golf hole, with more of the parameters being discussed earlier in the Engineering Requirements.

The first subsystem is the camera, seen below in Figure 1, being necessary in order to detect where the golf balls & holes are located in order to proceed with calculations to hit the ball into the hole. A MATLAB .m file runs the necessary script in order to capture the golf board & the positions of the golf balls & holes.

Figure 1: Camera Subsystem - Brio 100

Before further processing, the captured image undergoes a perspective correction. MATLAB performs this correction by applying a projective transformation to the camera image. Four hardcoded corner points on the raw image (top-left, top-right, bottom-right, bottom-left) are mapped to a rectangle of fixed size (700×350 pixels). The fitgeotrans function in MATLAB computes the transformation matrix, and imwarp applies it to warp the original image into a top-down, rectangular view of the board, removing perspective distortion. This step is necessary to eliminate distortion from the camera angle and ensure consistent, accurate measurements on the board. After perspective correction, the image is cropped using hardcoded points to isolate only the putting board and remove surrounding objects.

In order to determine the positions, a technique called Background Subtraction is implemented. The technique requires two pictures, like a before and after, in order to detect the golf balls & holes. The camera does this by taking a picture of the golf board with a black background underneath the holes, and then taking a picture with a colored piece of paper underneath the board with the golf balls placed at the desired location. Through doing this, the script is able to tell the differences between the two images, grabbing the centroids and sending them to the GUI.

Before sending the centroids to the GUI, it is necessary for the MATLAB script to eliminate noise taken from the camera's pictures. Noise can range from additional shavings on the board, a slight shift of the board, or an object near the board that was captured by the camera. To eliminate this noise, the code uses erosion & dilation. Erosion removes the additional noise, with the assumption that this noise is

“smaller” than the golf ball & golf hole. This eliminates the noise from the background subtraction, but can be detrimental to the detection of centroids due to altering the shape of the detected golf ball and hole.

To fix this issue from erosion, dilation is done in order to restore the previous shape of the golf ball and hole, allowing regionprops to correctly detect the centroids of the golf ball and holes. From this, the script then differentiates between the golf ball & holes through the RGB value of the centroid's position of the “current” image, that of the golf board with visible balls & holes. If the RGB value is close enough to white (255 255 255), then the script determines that centroid is a ball, otherwise it labels it as a hole. Once done, the subsystem is done running & is able to supply the GUI with correct locations of the golf balls & holes.

Another important feature for the subsystems to communicate with each other is the GUI, being implemented as a MATLAB App, seen in Figure 2. While not physical hardware, it is necessary in order to satisfy Customer Requirements for a User Interface. The GUI allows the user to visually see what is going on in the board, as it plots the centroids onto the taken image of the board, and to select which golf balls & holes they wish to score with. With how the software interacts, the GUI acts as the glue between each subsystem, acting as an intermediary between the camera subsystem & the motors’ Simulink model.

The GUI takes in the centroid data from the camera subsystem & displays it to the user, once the user has selected their golf ball & hole, the GUI calculates the trajectory necessary for the ball to go in. The calculation treats the scenario as a triangle, taking the slope between the selected ball & hole to find the slope. From the slope, inverse tangent calculates the necessary angle to aim the arm, and slope intercept formula then calculates the rail position. For hitting the ball, a simple angle is given to tell the servo motor to hit. After these calculations, the GUI sends the parameters to the Simulink model, pausing between each send to allow each motor to go to the appropriate position before the next motor runs.

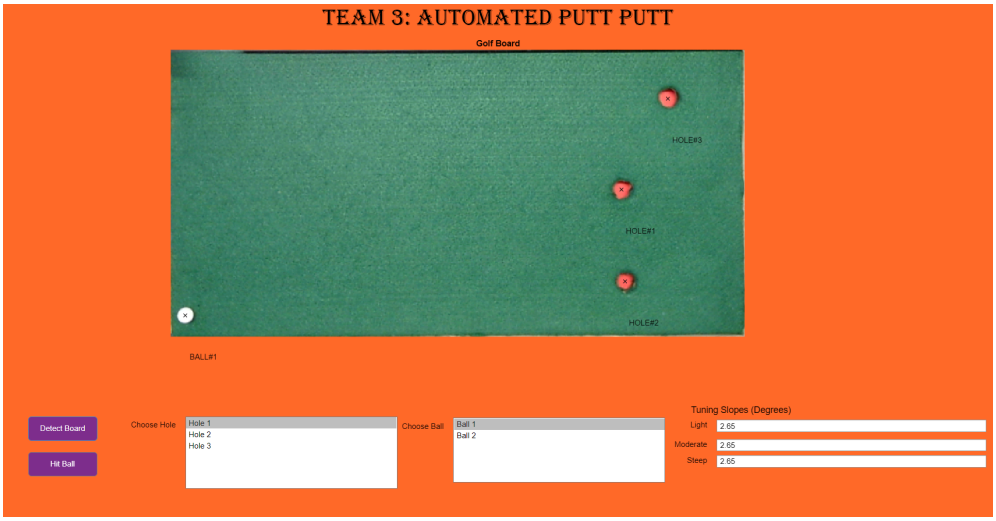


Figure 2: Graphical User Interface

The Motors receive some of its power and all of its instructions from the Arduino Mega 2560, which communicates with the Simulink model to send parameters from the GUI into the desired Arduino Pins that the motors are listening to. Both can be seen below in Figure 3 and Figure 4 respectively.

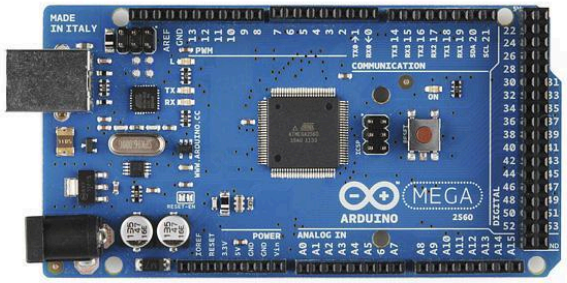


Figure 3: Arduino Mega 2560

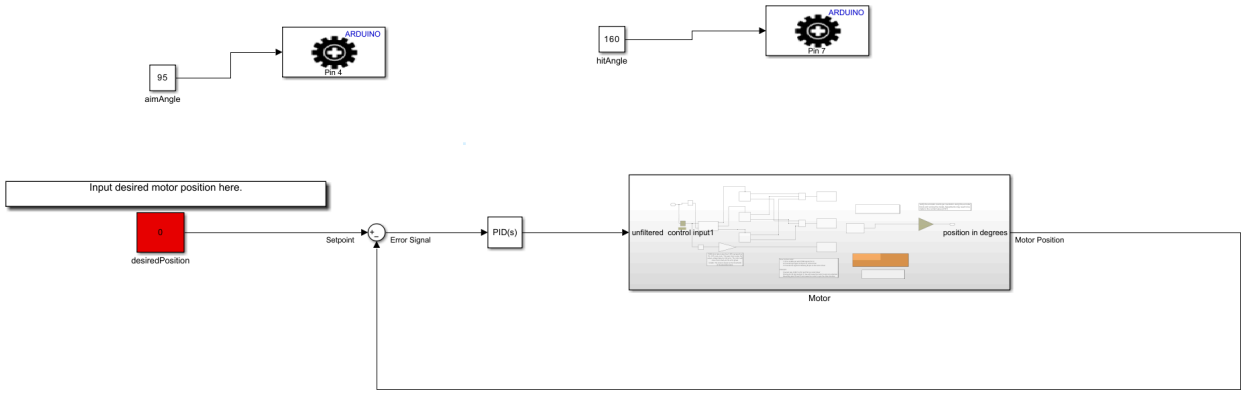


Figure 4: Simulink Model of the Motors

Our project's physical design offered a unique solution to a simple problem statement. During early conversations as a group we came up with design aspects that we liked and put the best of our initial thoughts together and found the solution that we generally stuck to throughout our project. We wanted a design that had physical resemblance to a paddle for the hitting apparatus and some method of moving along a straight line and a rotational axis as well that can drive the paddle for angled shots. Our design ended up using a cart, seen in figure 5 and 6, belt, and DC motor for the movement along our rail, seen in figure 7. A servo motor with mount for our rotational movement and an additional servo motor attached to that mount. The final servo had a 3D printed golf club attached to it for the hitting mechanism.

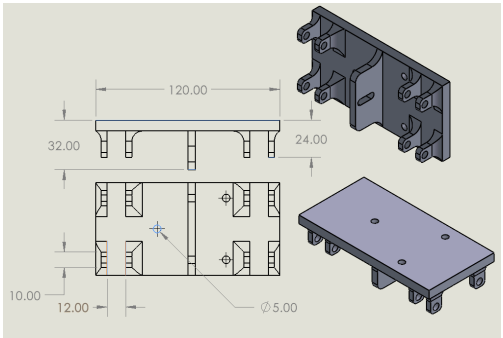


Figure 5: Bottom half of cart. Attached to the belt and has wheels to roll in the rail.

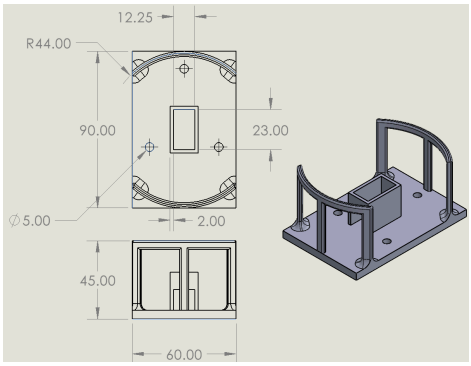


Figure 6: Top half of cart. Held the rotation servo motor and the rotation platform.

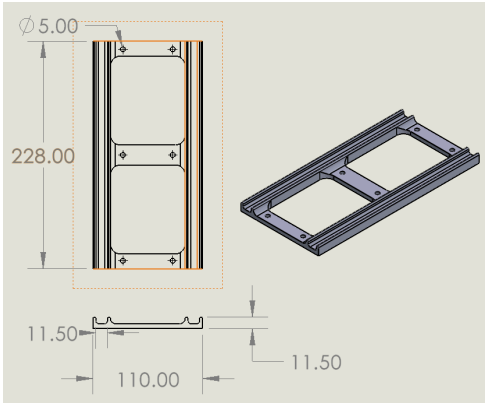


Figure 7: Rail, two were used in the final design.

Our motor position calculations involve considerable dependency on our hardware and their spatial locations. Whether that be the camera, any of our motors, or the distance from our board and assumptions we make along the way. The camera's placement determined the overall viewpoint of our pictures we could operate on. With our camera mount we aimed to have the camera capture as little extra space outside the board to maximize its efficiency and accuracy in centroid calculations. We also aimed to have the camera as centered as possible with respect to the middle of the board. This allowed for a considerably reliable method of locating our centroids.

Since our camera only picks up locations on our board and our putting mechanism exists outside this region we had to consider the offsets this would require us to incorporate. For simplicity we implemented a zero location along our rail and considered this location the farthest over our putter would ever need to be. This allowed us to isolate the problem to essentially a quadrant where we only deal with positive values of x and y along our coordinates in our image. This means that the DC motor's default position before any action is taken would exist at the origin.

To determine the location that the DC motor will end up at we consider the slope that the chosen hole and ball location will create and we consider the offset values we have found. To determine these offsets we measured out the start of the board to the point of rotation on our cart in the resting position. Taking those two distances and finding the resolution per area(pixels per inch) we calculated our pixel offsets in the x and y direction. There were additional calculations made to convert the rotational

movement of the DC motor into a certain distance that our cart needs to travel. From the customer requirements it was important for us to consider a few details before much of our printing. That being specifically the sharpest angle we would have from a potential hole and ball placement. The hole was said to be placed anywhere from 1’ to 2’ away from the putter side of the board. From this we found the maximum length our rail would need to be and used that length for our eventual design. This would ensure that we did all we could to design for any case within the customers needs.

We also need to consider our angle calculation. This angle calculation will determine where the putter aims before it takes the shot, the rotation platform is seen in figure 8. It makes use of the slope calculation found earlier and the inverse tangent function calculation. Due to the servo motor's behaviour we implemented an offset that would allow the servo to point either direction whether it was given a negative or positive angle. Since we already accounted for an optimal offset in our DC motor rail position calculation we have ensured that our putter will always be centered to allow for a consistent shot every time no matter the extremity of the angle.

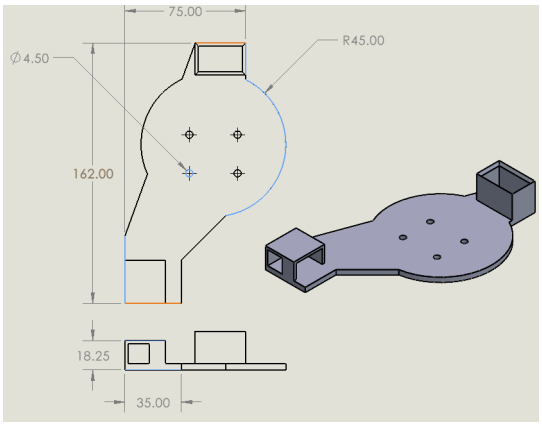


Figure 8: Rotation platform

The servo motor that controls the putter, seen in Figure 9, is considerably simpler than the previous two uses of motors. This motor is simply switching from one location to the next location. The speed at which this response occurred determined how hard the ball was hit. Through various iterations it was found that an increase in voltage resulted in a significantly harder hit. We also could add additional control in how far we pulled the servo motor arm back. Having the arm farther back resulted in a harder hit as well.

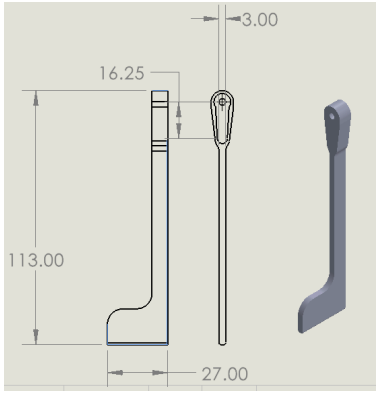


Figure 9: The Design of the Putter

Analysis of Final Prototype Performance

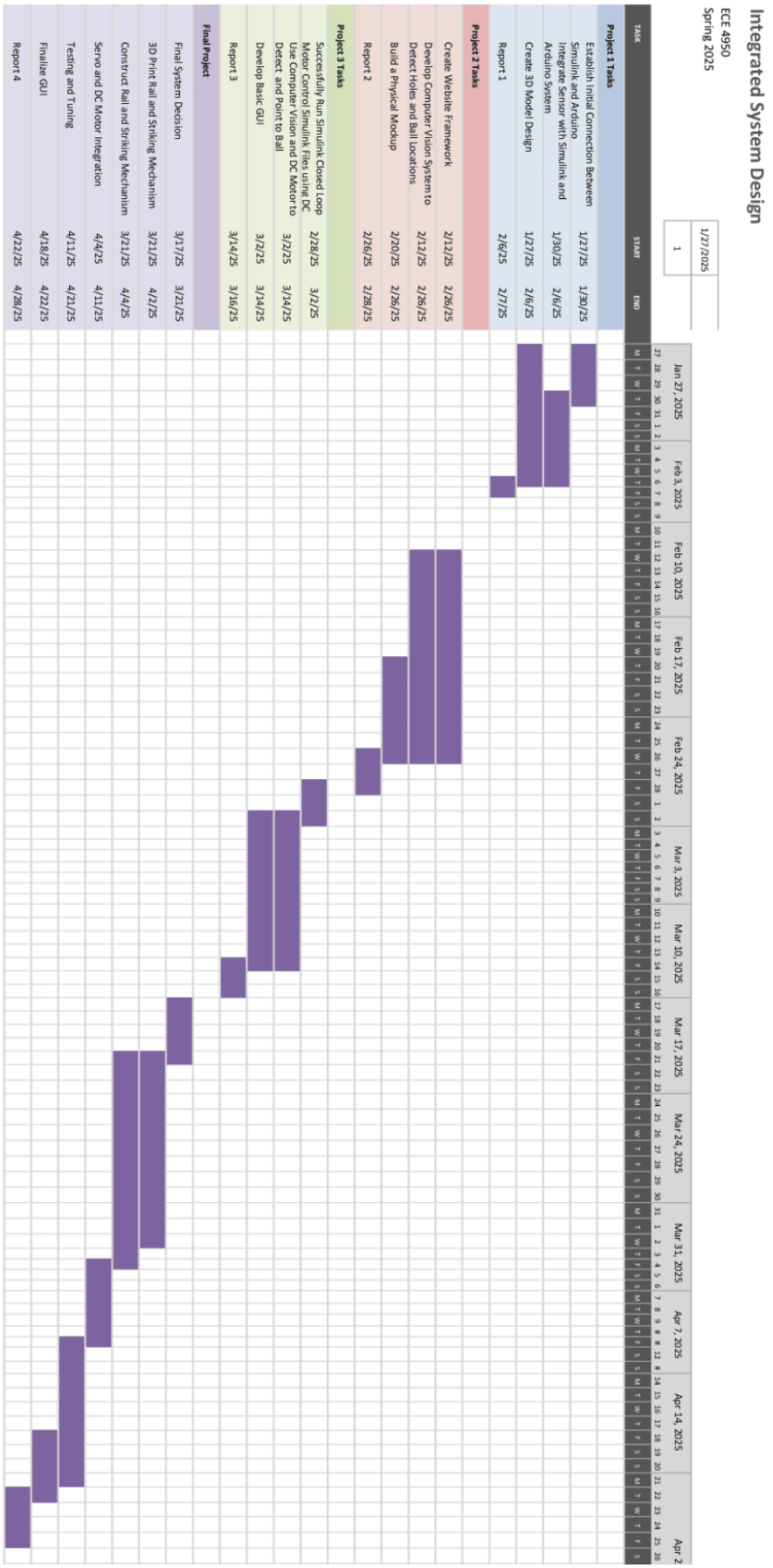
The system successfully met all customer requirements. Table 1 lists the customer requirements and provides an explanation of how the design met each requirement.

Table 1: Customer Requirements and System Performance

Customer Requirement	**Met? (Yes/No)**	**Explanation**
System must hit ball into selected hole	Yes	Successfully hits the ball into the selected hole using calculated trajectory and servo motors.
System must allow user to select ball and hole via GUI	Yes	GUI correctly displays ball/hole options and allows users to select.
System must work for multiple ball placements	Yes	Works for all placements, including extreme corner cases.
System must be easy to use	Yes	GUI is intuitive and user-friendly.
Hit ball within 5 seconds of user selection	Yes	System is able to perform the hit in under 5 seconds.
System must start after user clicks Start button	Yes	System waits for the user to click the “Detect Board” button before detecting the board. System waits for the user to click the “Hit Ball” button before running.
System must stay within 2.5'x1' workspace, only entering up to 2 inches	Yes	System stays fully within board boundaries.
System must be quiet, efficient, fast, safe, and user-friendly	Yes	System is fast and user-friendly; operation is quiet. Safety ensured by use of low-force motors and controlled motion.
DC motor with encoder and camera must be used	Yes	System uses a DC motor with encoder for cart movement and a camera for detecting golf ball and hole positions.

Project Schedule/Gantt Chart

Link to Gantt Chart File: Gantt Chart.xlsx



ECE 4950 Project 4 – Customer Requirements and Final Design Parameters

Use the guidelines below to complete your report and add at the end of your report.
Group Member Last Names: Page, Silva, Jenkins, Veith

Score	Pts	
	5	**General Format - Professional Looking Document/Preparation (whole document)** a) Fonts, margins (11pt, times new roman, single spaced. 1" margins on all sides). b) Spelling and grammar are correct c) Layout of pictures – all figures need numbers and captions and must be referenced in the text d) Follows the page limitations below. e) References. Use IEEE reference format. f) This grading sheet is included as the final page.
	0	**Page 1: Title, Group Name, Group Members, and Date** **Executive Summary** (1 concise, well-written paragraph) Provide an overview of this project. Briefly describe what you did and what you learned.
	5	**Page 2: Engineering Requirements** (<1 page) Bulleted list of Final Design Engineering Requirements
	10	**Pages: 3-7: Design Details (<5 pages)** Describe a system that can be built including System Architecture and System Integration based on the Engineering Requirements. Do not include data sheets or software code.
	10	**Page 8: Analysis of Final Prototype Performance** (<1 page) Did it succeed or fail to meet customer requirements? What went wrong and what happened in the design process to allow this problem? Make a table of the customer requirements and address how well your design met these expectations.
	5	**Page 9: Project Schedule/Gantt Chart** (<1 page) Create a schedule (Gantt chart) that shows the tasks and schedule for your project. Start from the very beginning of your project and extend to the end (completing final report and presentation).
		Page 10 This grading sheet is included as the final page.
	50	Laboratory demonstration of your prototype (evaluated by instructor and TAs). Evaluator will manipulate the interface and evaluate how well the system provides the timing and display functions (i.e. how well does the closed loop control work). Is it well built? Neat wiring? (.6 * the prototype evaluation score)
	15	Professional-looking video explaining the working of the designed system.
	15	Rating by reviewers after competition
	15	Poster for Demo containing the following from the report: ● Executive Summary ● Customer and Engineering Requirements ● Software Explanation ● Prototype Design Explanation ● Performance Analysis