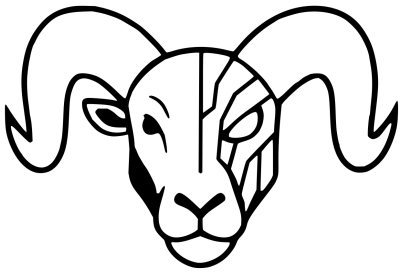
Rambots: Robust Robotic Systems for Outreach



Honors Thesis

Presented in Partial Fulfillment of the Requirements for the
University Honors Program
Colorado State University

By
Everett Serff
Walter Scott Jr. College of Engineering

Thesis Advisor: Soheil Fatehiboroujeni, Department of Mechanical Engineering

Thesis Reader: Ryan Bailey, Department of Civil Engineering

Spring 2026

Table of Contents

Abstract

The use of robotics in outreach and education has been shown to play an important role in motivating student interest in STEM, particularly when interactive experiences and mentorship are involved. Building on this foundation, the Rambot project is a multi-year, multidisciplinary effort focused on developing an accessible quadrupedal robot platform known as Sparky, which the team brings to K-12 STEM outreach events. This year's project focuses on improving the robot's functionality and outreach impact through the addition of new capabilities and forward-looking design work. Specifically, the team developed a manipulator arm capable of picking up objects and began early-stage design of a next-generation Rambot Mark II. The arm was designed using decision matrices, static analysis, and inverse kinematics, with validation through simulation in MATLAB, Python, and MuJoCo, along with CAD modeling, additive manufacturing, and custom PCB development. In parallel, the Mark II effort involved defining design requirements, creating conceptual models, and evaluating alternative leg architectures. Simulation results demonstrate that the manipulator arm can reliably reach target positions and perform object interaction tasks, supporting its potential use in automation and interactive demonstrations. Additionally, outreach experiences suggest that the addition of interactive features such as the arm increases engagement and interest among participants. The Mark II design process further identified a ball screw-driven leg mechanism as a promising approach for improving performance and robustness. This work contributes to the continued development of accessible robotics platforms by expanding functionality and strengthening their effectiveness as educational tools, while also laying the groundwork for a more advanced and capable system in future iterations.

Rambots: Who We Are

Rambots is a multi-year, multidisciplinary project with members across many of CSU's engineering disciplines, including computer, electrical, and mechanical engineers. The Rambot started as an open source project made by James Bruton. The team recreated it and has been improving on the project since it started. The main objective of this project, similar to previous years, is to apply the engineering skill set we developed over four years towards improving an affordable and educational outreach tool. In this way, we can generate interest in engineering for middle and high school students, while also growing as engineers.

This year the team was split into 3 sub-teams to effectively split the workload and achieve our many goals. Our general goals have been to improve the reliability of the Rambot and make additions to make it stand out as an original project rather than a recreation of the open source design. Each sub-team is dedicated to at least one of these goals. The Rambot sub-team is working on cable and code improvements to make it more reliable. The new Armbots sub-team is developing a new robot arm to mount on top of the Rambot to add interesting new functionality. Lastly, the Rambot Mark II subteam is trying to address all goals at once by starting a new Rambot from the ground up with original designs and incorporating everything we wish the Rambot had.

While I have contributed to every sub-team, my main involvement has been with the Armbots sub-team in terms of tangible work done, and I have been leading R&D for the Rambot Mark II sub-team. I joined Rambots because I have an interest in robotics, especially in robots that mimic living creatures. I follow many Youtubers who make robot dogs or other robots and it inspired me to make my own. The mission of the project resonated with me as well, being to inspire and engage young students in STEM. If it was not for the continued exposure I got to

engineering throughout middle and high school I would not have pursued engineering in college. Helping be a part of why a young person might get into STEM seems like a noble and important mission.

The Rambot is built off an open source model called OpenDogV3 by James Bruton, a popular robotics youtuber. Many students, including myself, are fans which means the Rambot often gets recognized as OpenDogV3, which becomes a point of criticism as the project appears to be building a kit rather than doing original work. I saw this as a chance to use my creativity to shift how people perceive the project, and the drive to do this was another motivating factor to joining. This paper discusses the engineering process that went into achieving our goals, how we conduct outreach, and how I have grown for being a part of it.

The Robot Arm

When I joined the team our primary ambition was to make a robot arm that can be mounted on top of the Rambot, affectionately named Sparky. This would allow it to interact with its environment in a way not previously possible, making it more engaging at outreach events. The ultimate goal of the arm is to enable Sparky to play fetch, retrieve objects from the ground, and perform tasks such as opening doors. When we started on this project we put together some research questions which will be addressed below.

What motors should be used?

Choosing motors is the first important step to making the arm. They will drive the motion and the arm will have to be designed around them. The following summarizes the research conducted on different motors and a detailed look into the decision making and design process.

Brushless DC Motors

"A BLDC motor uses an electronic commutation instead of brushes. It includes a rotor with fixed magnets and a stator with electromagnetics" [1]. Brushless motors have three phase wires, each corresponding to one phase of the stator windings. A motor controller energizes these windings in a timed sequence to create a rotating magnetic field that drives the rotor [1].

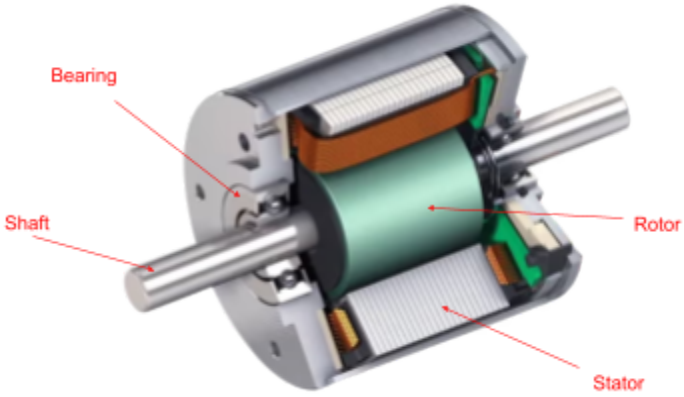


Figure 1: Components of a brushless motor, modified from [1]

For many robots, energy efficiency is critical. Brushless DC Motors (BLDCs) are the most efficient motors because they "can transform more electrical power into mechanical power than a brushed motor for the same input power because of the absence of friction of brushes" [1]. This is a big reason why they are used on the Rambot, the Rambot employs 12 motors and runs on batteries so BLDCs help maximize battery life. The lack of brushes means there is less friction as well, resulting in a longer lasting motor. They can typically produce high torque at low speeds, but typically require gearing to get higher torques.

A key disadvantage of brushless motors is their complexity, needing to be connected to an often bulky motor controller and require an encoder for the highest accuracy [1].

Servo Motors

A servo motor is a type of DC motor that comes with gearing and position control. Its essential components include a DC motor, a potentiometer, a gearbox, and a control circuit. The figure below shows a typical servo configuration.

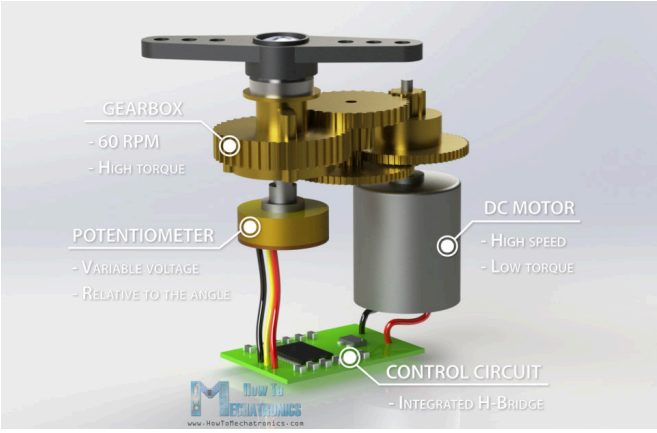


Figure 2: Servo motor configuration [2]

When the output shaft rotates, so does the potentiometer which gives feedback on the output's position to the control circuit. The feedback guides an error amplifier that "compares current and desired positions, generating an error voltage that powers the motor until the error is zero, facilitating accurate rotation" [2]. Due to the potentiometer, most servos have a limited range, typically 180 degrees or 270 degrees, which is acceptable for robot arm applications since the arm joints do not need higher range of motion than that.

A servo motor also has 3 wires, but instead of each wire controlling a phase of a stator winding, 2 of the wires are just the positive and negative terminals of the power supply, and one wire is the signal. The signal supplied to the motor is in the form of Pulse Width Modulation (PWM), meaning the signal is pulses of electrical signals of different lengths. "These pulses have a width that varies between 1 to 2 milliseconds, and they are sent repeatedly at a rate of 50 times per second to the servo motor. The adjustment of the pulse width serves as a means to effectively control the position of the rotating shaft in the servo motor" [2].

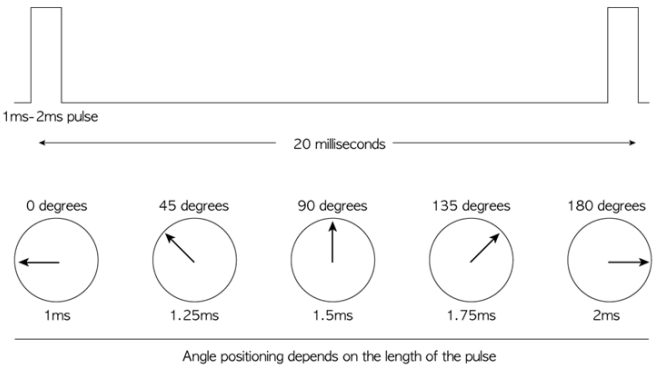


Figure 3: How PWM length affects the angle of a servo motor [2]

There are 2 types of servo motors, analog and digital. The main difference is the frequency in which they can have an output. An analog servo operates at 50 Hz, outputting signals only once every 20 milliseconds, which is slow for many control systems [2]. A digital servo is capable of 500 hz, giving it faster response and better resolution.

Servo motors are advantageous to the application of a robot arm because they provide fast precision control without much complexity, being easy to integrate into a system. They are also fairly energy efficient, especially if smaller ones are used [2]. It is also advantageous that they are specced according to their torque, meaning that once required torques are calculated it is easy to find a servo that meets or exceeds the requirements. Another factor to consider is that many robot arms in industry utilize servos, emphasizing their viability for the application [3].

Some disadvantages are that servos land on the pricier side when compared to DC or stepper motors. This can be partially overcome in the Armbots project by getting smaller servos for each joint since each joint will experience less torque, but this introduces a new issue of power distribution since each servo may require a different maximum voltage.

Stepper Motors

The distinguishing characteristic of a stepper motor is that its shaft rotates through steps. Each rotation has a set amount of steps, for example a Nema-17 has 200 steps per revolution. This gives fairly precise angle control because it is known that each step is 1.8 degrees [4]. Like other motors, steppers have a moving rotor and fixed stators. Each pair of stators is one step and is winded together. The operating principle is "the rotor aligns with the magnetic field created by the current flowing in the coil when one or more stator phases are energized. The rotor can be turned a certain degree to obtain the desired final position by sequentially delivering different phases" [4]. This principle is visualized in Figure 5 below.

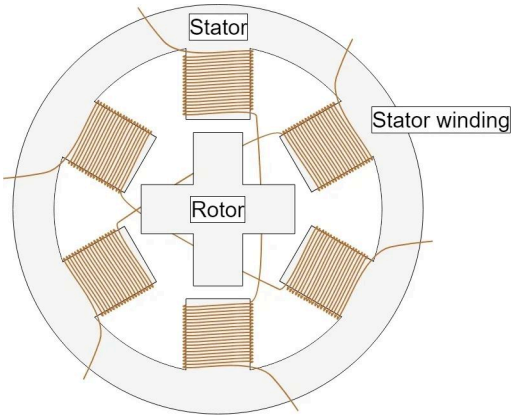


Figure 4: Internal configuration of a stepper motor [4]

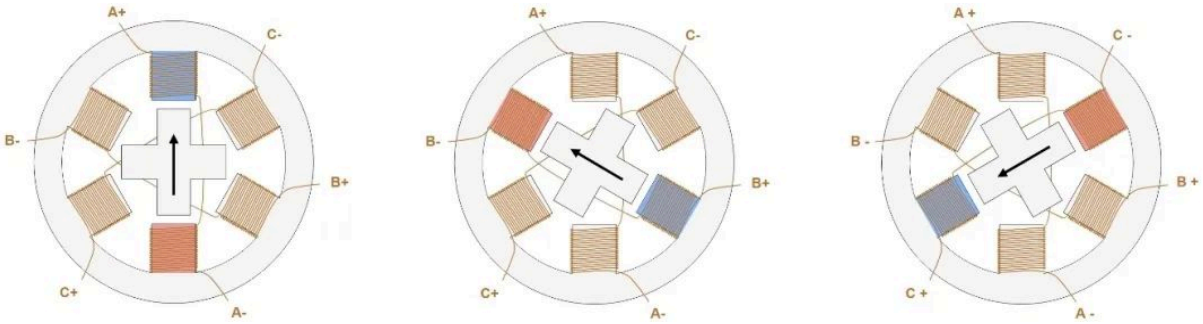


Figure 5: A demonstration of the operating principle of a stepper motor through energizing pairs of coils to turn a rotor [4]

The key properties of a stepper motor are that it has high precision without need for closed loop control. Unlike servos, they are capable of continuous 360 degree rotation. They also exhibit high holding torque [1].

Some disadvantages are that they are often large, especially high torque steppers. They also are more difficult to control, as they require a motor driver and have 4 input wires. Lastly, in theory the open-loop precision should be good enough, but there is no way to guarantee precision without encoders.

Decisions and Design

In order to decide on what motors to use for the arm joints, we put as many motor types as we could think of into a decision matrix with as many relevant evaluation factors as possible. This would allow us to make a more objective decision about which motors best suit the application. As seen in the table below, the chosen motor type for the arm joints was servos.

Motors											
	Torque	Accuracy	Weight	Power requirement	Cost	Speed	Size/volume	Connections	Integration complexity	Backdrivability	Final Weight
Weight	0.60	0.40	0.30	0.30	0.15	0.05	0.20	0.05	0.15	0.35	
BLDC + Encoder	4	4	4	5	2	5	3	2	3	4	9.8
Industrial Servo	5	5	4	5	3	4	4	5	5	4	11.55
Stepper	3	4	2	3	4	2	2	1	1	1	6.55
Hobby Servo Motor	1	5	5	5	5	2	5	5	5	4	9.85
DC Motor and encoder	4	4	2	3	3	4	2	3	3	4	8.55
AC motor	4	1	1	1	1	3	1	1	1	1	4.45

Table 1: Motor decision matrix for arm joints

In summary, the decision mostly came down to BLDC motors versus digital servo motors. Servos were preferred because they have a lower integration complexity, having the gearing, position control, and motor driver built in. If the team chose BLDCs, getting ODrive motor drivers would be required, as well as making a gearing system which would likely have to be 3D printed for cost effectiveness. All of those factors would lead to a much bigger and heavier arm if BLDCs were used and would slow down development.

The arm does not use servos exclusively, however. The base and the claw both require motors that have a continuous range of motion and good holding torque with good position control. As discussed in the stepper motors section, stepper motors are fit for this application. To overcome the low torque capabilities of a stepper motor in the base, a gearing system was created with a 5:1 reduction, increasing the torque by 5 times while also giving even greater resolution to the base turn since one step is now 0.36 degrees instead of 1.8 degrees.

In order to choose what servo motors to get, torque estimations needed to be performed. Servo motors are specced by their torque rating so it is easy to tell what motor to get from torque estimations. First it was important to find out how long the arm needed to be. A simple model was developed where the platform is the height of the dog while it is crouching and we know we want it to at least touch the ground six inches away. This gives a minimum arm length estimate. We wanted to be able to reach that point but still have much of the arm raised up so we chose lengths of 15 inches for link one, 15 inches for link two, and six inches for link 3.

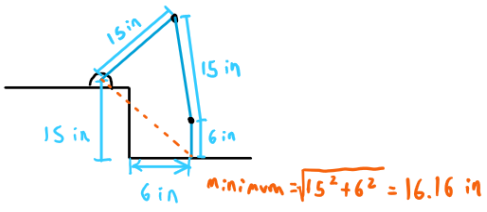


Figure 6: Estimating arm length sketch

Now that the lengths were determined, the torques could be estimated by putting the arm in its worst case scenario in terms of torque, which is fully extended out to the side. From here the sum of the torques can be calculated as:

$$\Sigma M_O = -\frac{L_1}{2}W_1 - L_1W_4 - (L_1 + \frac{L_2}{2})W_2 - (L_1 + L_2)W_5 - (L_1 + L_2 + \frac{L_3}{2})W_3 - (L_1 + L_2 + L_3)W_6 = 0 \quad (1)$$

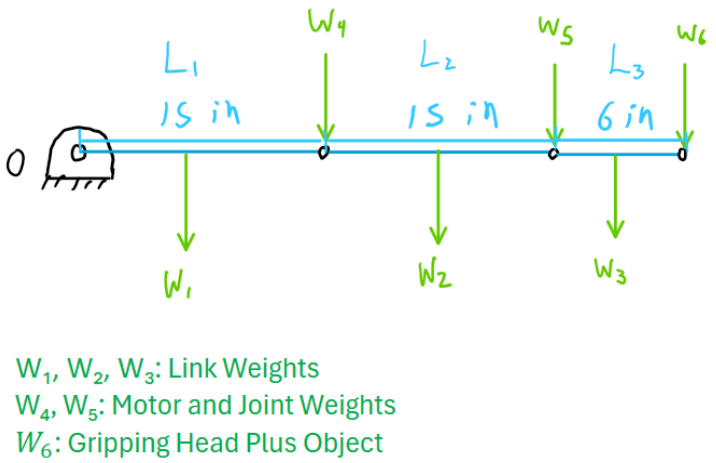


Figure 7: Free Body Diagram of the Arm

In order to easily play around with different weights and lengths of the arm, a MATLAB script was created using the equation above to allow users to put in the density and cross section of a link material, and put in motor weights as well. This allowed for us to iterate through different material choices and motors. After an initial estimation was completed, we were able to find motors that matched those specifications and added their weight into the program to run it again to make sure the torques still worked out. The final torque estimations assuming 4"x2" PETG (material selection will be discussed in a later section) links were as follows:

```
Estimated Torque of Shoulder Motor (kg*cm)  = 339.074

Estimated Torque of Elbow (kg*cm)  = 126.491

Estimated Torque of Wrist (kg*cm)  = 16.5773
```

Figure 8: Output of MATLAB Script Estimating Torques of Each Joint

Available digital servos that exceed required specifications were as follows:

Location	Name and torque rating	Image	Price
Shoulder	550kg*cm DOCYKE High Torque RC Servo, 2 in 1 Servo and Motor 16V~24V High Voltage Full Metal Gear Digital Servo		$129.99
Elbow	Annimos 160 kg*cm servo (270 Deg motion)		$44.85
Wrist	Annimos 35 kg*cm servo		$33.87

Table 2: Chosen Arm Joint Servo Motors

Finally, the stepper motors chosen were these two:

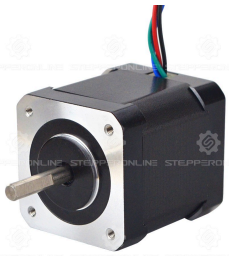
Location	Name and torque rating	Image	Price
Base	Nema 17 Bipolar 0.9deg 46Ncm(65.1oz.in) 2A 42x42x48mm 4 Wires		$11.41
Claw	E Series Nema 17 Bipolar 1.8deg 17Ncm(24.07oz.in) 1A 42x42x23mm 4 Wires		$4.43

Table 3: Chosen Stepper Motors

What material should be used?

Selecting a material for the robot links is another critical design decision. The material determines how the links will be manufactured, how much budget needs to be allocated, and how the parts should be designed. This section examines some of the materials considered and why we chose 3D printed PETG.

The decision can be explained by our decision matrix. We compared each option by their weight, strength, cost, manufacturability, fracture resistance, time to make, joinability, and stiffness.

Material									
	Weight	Strength	Cost	Manufacturability	Fracture Resistance	Time to make	Joinability	Stiffness	Final Weight
Weight	0.50	0.30	0.30	0.35	0.20	0.20	0.15	0.40	
PLA	4	2	5	5	1	4	4	5	9.45
PETG	4	3	5	5	1	4	4	5	9.75
Sheet Al	3	4	4	3	5	3	3	4	8.6
Steel	1	5	3	3	5	1	2	5	7.45
Carbon Fiber	5	4	1	1	5	2	4	5	8.35
TPU	4	3	3	5	5	4	4	2	8.75
Cast Metal	1	4	2	1	5	1	1	5	6

Table 4: Material Decision Matrix Choosing PETG

PLA was a close contender to PETG. We chose PETG because the website Simplify3D [5] provides a great comparison chart for the different 3D printing materials, and demonstrates that PETG has higher durability and fatigue resistance than PLA which is what matters most for a robot arm that will repeatedly go through motions after a period of sitting idle.

What software should be used?

After the robot arm is constructed there needs to be some way to control it. Software plays an important role in this as it is what facilitates the programming, inverse kinematics, and interprets inputs into outputs for the arm. Below are software options we considered and what our decisions ultimately were.

Arduino

Arduino is a beginner-friendly IDE primarily designed for simple, standalone sketches. While it supports multiple files via tabs, its build process (which merges files alphabetically) is prone to errors in complex projects. It lacks integrated version control and advanced code

navigation tools, making it ideal for quick prototyping and hardware testing but inefficient for a collaborative team environment. We will use it only for early-stage motor testing.

PlatformIO For VS Code

PlatformIO is also an open source IDE built on Microsoft Visual Studio Code, providing a professional grade development ecosystem. Being integrated into VS Code provides out of the box functionality including git integration, making it much more ideal for a team environment. Team members can pull the project source, make contributions, and push them back for the rest of the team to review and run. PlatformIO additionally includes a .ini file within the workspace that automatically configures all of the libraries the project uses and other settings. This is a much more streamlined process than manually downloading all of them, as required in the Arduino IDE. Although Arduino has basic multi-file support, it is not as powerful as VS Code's folder based organization, which allows for true modular programming. PlatformIO utilizes true C++ programming, allowing you to keep your implementation logic and interfaces distinct and manageable. You can create header files with a .h extension that have functions, class definitions, constants, etc. These can be included in the implementation source files with a .cpp extension. This makes code easier to navigate since it is split up into parts instead of one big program. PlatformIO's features, combined with VS Code give it the advantage over Arduino IDE for our purposes.

Robot Operating System (ROS)

We came across a paper from Ben Hazeem et al. [6] that makes a strong argument for using ROS for our application. As described in the paper, “The ROS is an open-source program for writing robot software for simulation applications and running robot hardware in

experimentation. Recently, ROS has become one of the mainstream robot systems; it contains libraries, tools, and conventions in order to simplify the process of controlling complicated robots using a diversity of robotic platforms" [6]. Additionally, it "provides packages to simulate multi-DOF robot manipulators, such as solvers for both forward and inverse kinematic models; and dynamic motions" [6]. These make it ideal for our application, however the program seems to come with a steep learning curve. The goal is to implement ROS eventually, but prioritize VS Code and PlatformIO as the team is familiar with it.

Another important aspect of our workflow is simulation. Our team currently uses MuJoCo to simulate the robot arm because of its high-fidelity physics and accurate contact modeling. While MuJoCo is not inherently part of ROS, recent work has shown that the two can be integrated seamlessly. As described in "MuJoCo ROS: Integrating ROS with the MuJoCo Engine for Accurate and Scalable Robotic Simulation," MuJoCo is considered "one of the most promising engines for high-fidelity simulation due to its advanced contact modeling, numerical stability, and computational performance" [7]. The authors also note that MuJoCo ROS "combines MuJoCo's advanced physics with ROS's modular environment" [7], enabling researchers to use realistic physics while still benefiting from ROS's standardized controllers, planners, and communication tools. This means that once our team transitions to ROS, our existing MuJoCo simulation pipeline should integrate smoothly, allowing us to test ROS-based control strategies in simulation before deploying them on the physical arm. This compatibility will make the eventual shift to ROS significantly easier and will help ensure that our control algorithms behave consistently between simulation and hardware.

Decisions

In summary, for this year's team, the chosen software for programming the arm will be VS Code with PlatformIO. Arduino will be used for some small tests and the team hopes next year's team will be able to build on our infrastructure and implement ROS into the arm.

What microcontroller should be used?

The microcontroller will be the most important piece of hardware, as this is what is programmed and sends commands to the motors. The options considered are ESP32, Arduino, Teensy, and Raspberry Pi. An ESP32-S3 was chosen because it is small, has bluetooth and WiFi capabilities, and a dual core processor. Arduinos lack the memory this project needs and ones comparable to ESP32s are significantly more expensive. A comparable Teensy, like the Teensy 4.1, does not have integrated wireless and only has a single core, and that core is overpowered for our application. The Raspberry Pi Pico 2 W is comparable with an ESP32-S3 as it also features built-in wifi, bluetooth, and a dual core processor. The ESP32-S3 was selected because its dual-core setup is designed to "set and forget" the Wi-Fi. It uses a built-in operating system to keep the wireless connection running on one core without ever slowing down the motor movements you have programmed on the other. In contrast, while the Pico 2 W is also dual-core, it lacks the specialized "motor-control" hardware found in the ESP32. The ESP32 includes dedicated circuits specifically designed to handle motor timing and safety at a hardware level, making it a more reliable "all-in-one" brain for a robot arm.

Making the arm

After making our crucial decisions on the arm, we could start designing and building it. Jason was assigned to model and make the arm links, Tom was assigned to make the claw, and I was assigned to make the base. The base came with a few design questions as well. The first was how will the base be turned by the motor? I decided to put the motor in the center of the base with its shaft facing down where a gear could be attached to it and create a gear drive to increase the torque and transmit the power to the base housing. This meant the housing needed some way to rotate, which needed a bearing. I chose a lazy susan bearing with a 6 inch diameter since the motor and gearing system would be able to fit inside it, and it is able to support 500 lbs making it well suited for the application. Below is a sketch of how it was planned to be used and the final implementation.

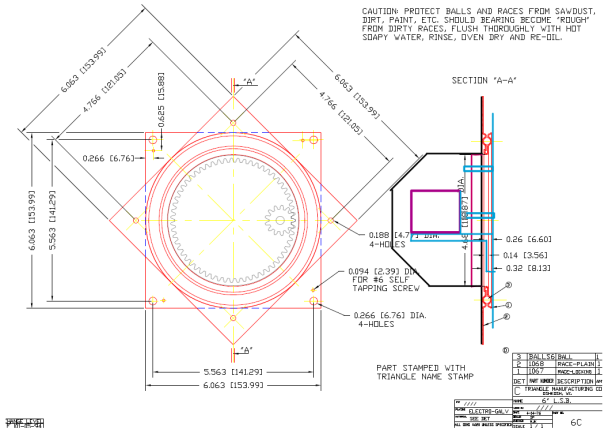


Figure 9: Sketch of how the gear and motor setup was planned for the base

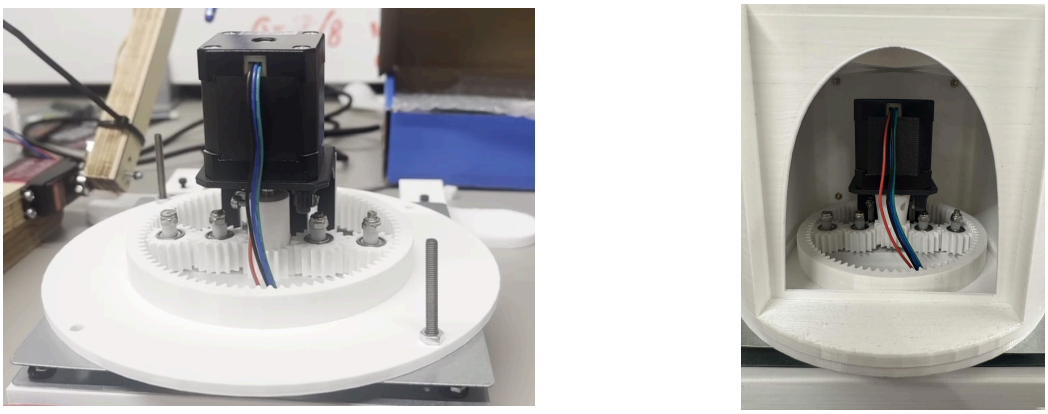


Figure 10: Finished implementation of gear system, motor, and bearing

The number of teeth on the gears and diametral pitch was found through iteratively trying different combinations until a match was found that provided a gear ratio greater than 3:1 and had shared centers for the motor gear and ring gear on the Gear Generator website [8]. The hole in the bearing was 4 inches in diameter so I knew to make the pitch diameter of the ring gear to be 4, providing a good basis from which I could iterate. The working combination was spur gears of 16 teeth, diametral pitch = 20, pitch diameter = 0.8, and pressure angle = 20; while the internal gear was 80 teeth, diametral pitch = 20, pitch diameter = 4, and pressure angle = 20.

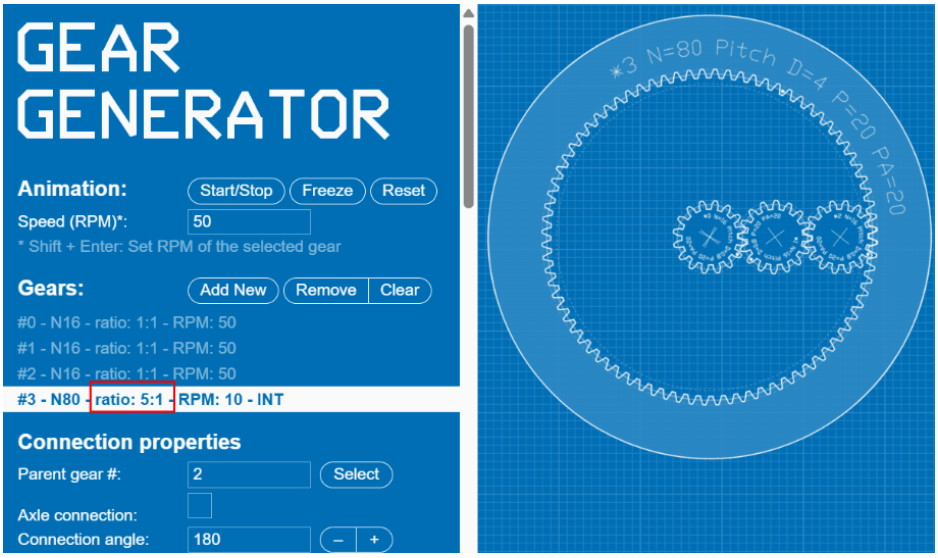


Figure 11: How Gear Generator was used to make the gears [8]

The final iteration of the base housing featured holes that line up with the holes in the bearing and ring gear with threaded inserts to make the screws hold reliably. It features standoffs to allow the mounting of the custom PCB and a window to see the gears for diagnosing issues and demonstrating how it works at outreach events. The base was fun to make because it included some creative design choices and was designed as much for aesthetics as it was for functionality.

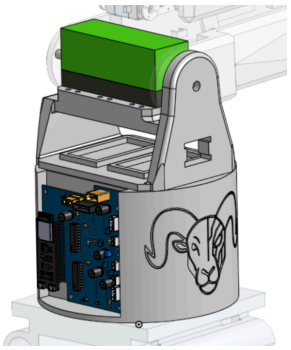


Figure 12: Base housing model

Simulating and inverse kinematics

A primary function of a robot arm is using it to automatically do tasks. This typically involves telling it some positions to go to, but how does the arm know how to get to that position? That is where inverse kinematics comes in. Inverse kinematics makes it so a user can input a coordinate point and then the angles the arm needs to have to achieve that position can be solved for. However, the arm created has bends in it while kinematic models typically use straight lines. The actual arm can be simplified into straight links by drawing lines from joint to joint. The actual motor position will then be off by a constant angle that can just be added to the inverse kinematic result when giving the output angle to the motors. Below is a visual representation of the virtual links and their length and offset.

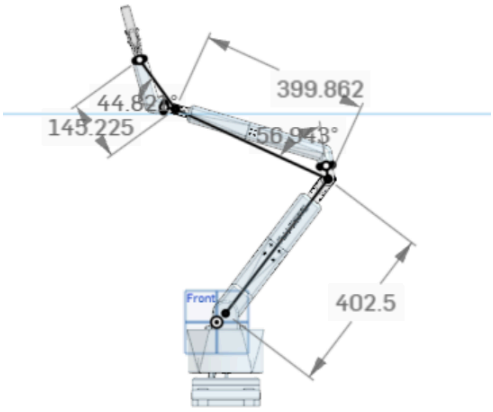


Figure 13: Simplified kinematic model of arm

After making this model, the inverse kinematics can be derived as below using the fundamentals of inverse kinematics from *Modern Robotics: Mechanics, Planning, and Control* [9].

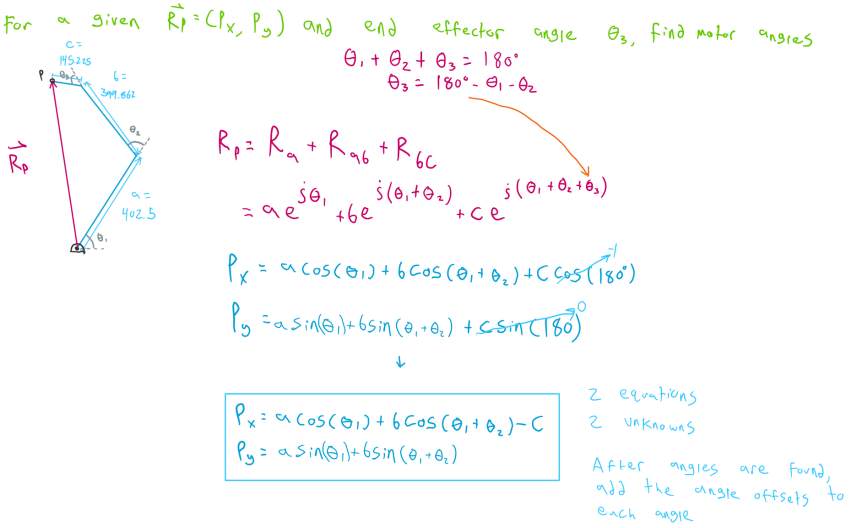


Figure 14: Inverse kinematics equations derivation

With these fundamentals, a MATLAB program was made to demonstrate the ability to follow paths with inverse kinematics, and then upgraded to be a Python program that uses MuJoCo to simulate the arm going to positions with realistic physics.

End product

The end product was mostly 3D printed out of PETG featuring a few metal features such as motor couplings on the base motor gear and on the shoulder motor and a metal pin for supporting the other end of the shoulder link. The final design utilizes a custom PCB which handles the voltage step downs for the microcontroller and different motors and controls the motors. Inverse kinematics were used to create a simulation of the arm where a user could input a coordinate and the arm would go there, which will be implemented to control the real arm as well. Below is an image of the finished product on its test stand, and below that is the intended final configuration of the arm on the Rambot.

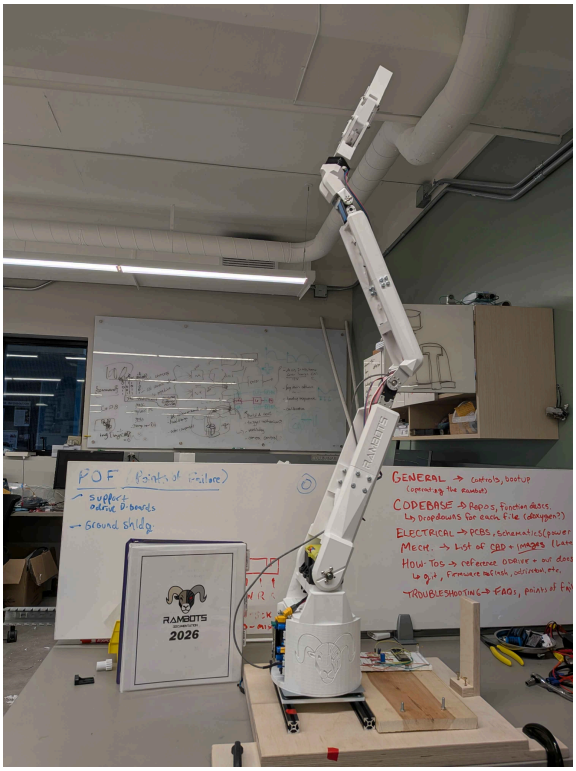


Figure 15: Finished robot arm on test stand

Figure 16: Final configuration for robot arm on top of the Rambot

Rambot Mark II

While the Rambot's original goal of an accessible robot that can be mostly 3D printed is admirable, CSU provides extensive manufacturing resources that the team should utilize and make a selling point to get more students at outreach events interested in CSU. Additionally, the main point of criticism from E-days judges over the years is that the project nearly always looks the same as last year. Our team addressed this by making lots of new additions to Sparky, but a more effective approach is to develop a new robot from the ground up featuring a completely original design, which also separates it from its open-source roots in OpenDogV3.

To improve the Rambot, it is important to identify problems and potential solutions to them. To do this, a survey was made for the team to identify what team members like, dislike, and would like to be added to Sparky. Additionally, a design questions document was made which laid out research questions and the research to fulfill it. An example is what leg configuration to use.

Leg Configurations

The current leg configuration of the Rambot consists of legs that are mirrored about the center which have their knees driven by pulleys.

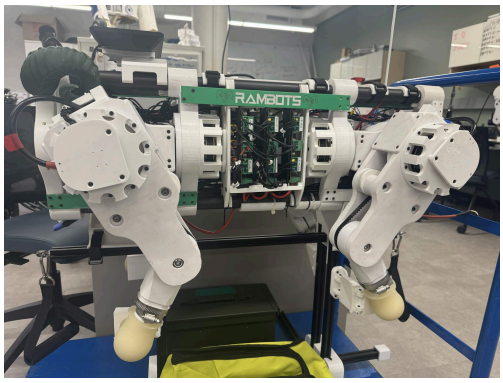


Figure 17: Current Rambot leg configuration

This design was chosen by the creator of the open-source model rather than our team. It uses pulleys as it allows the motor to stay close to the top which decreases the amount of torque needed to turn the leg, and pulleys are pretty efficient at transmitting rotation over a long distance. The legs are mirrored, pointing in at each other, because otherwise the center of mass shifts too far one way and it is hard for the robot to balance. Below are some other configurations considered for the Mark II.

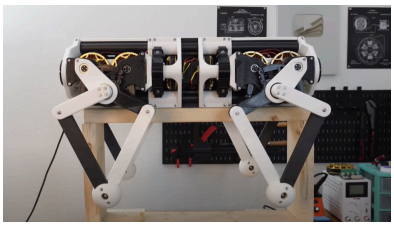
		 
Linkage driven leg	CARA 5 Bar linkage leg	Boston Dynamics ball screw driven leg

Table 5: Leg configurations considered for Rambot Mark II

Overall, we chose to use Boston Dynamic's leg configuration utilizing a ball screw. The ball screw is driven directly by a motor which linearly actuates the knee as a result. This method is more compact than all the others and ball screws are very reliable and efficient. This design is also standard in industry which gives the team confidence that this is the right direction. The design also differs from what the Rambot does in that both legs bend backwards. This is smart design because it allows the robot to easily climb stairs, which would be a capability the team should be open to including [10].

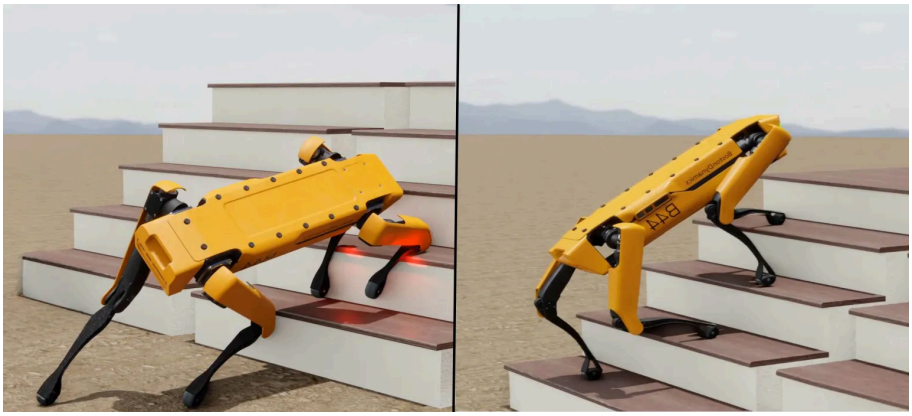


Figure 18: Forward vs backward bending legs on stairs [10]

I made a sketch of what a future Rambot Mark II might look like utilizing a VR sketching and modeling software called Gravity Sketch. The components are made translucent so the internal mechanisms are visible, including the ball screw mechanism.

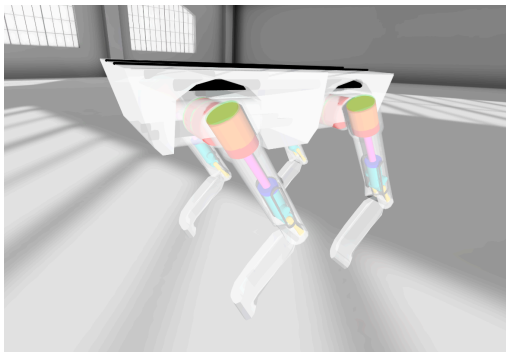


Figure 19: Sketch of Rambot Mark II showcasing internal mechanisms

Assembling a team

The overall goal of what we have done this year is to try to set up next year's team for success. Part of that included other team members making detailed documentation on the Rambot, and my contribution is setting out a solid foundation for next year's team to get started on an ambitious goal. Along with my project coordinator Joey, we have been finding members for next year’s team early and meeting with them to teach them about the project and getting

them started on the skills they will need, answer any questions they have, and go over what is already designed. By keeping an open connection with the next team they will be prepared for the task and ideally able to deliver, and pass on the preparation we gave them to the team after them.

Worldwide applications

This project may appear as building a robot dog for fun, and while it is fun, it has much more significance. Sparky is an outreach tool, and throughout the year we have brought Sparky to at least 7 outreach events, notable ones including: Festival on the Oval, CSU's homecoming all-university showcase to parents and the CSU community; Greely Bot Bash, a robotics showcase for high schoolers interested in robotics; and Choose CSU, a large-room showcase of our project to general prospective engineering students interested in coming to CSU. I participated in two outreach events, ECE Department Night and Experience ECE. At ECE Department Night, I engaged with first-year engineering students by demonstrating the project and explaining its functionality through hands-on interaction with the test leg, including demonstrations of closed-loop control. At Experience ECE, I presented the project to prospective students and their families, discussing both the technical aspects of the system and the opportunities available to contribute to similar projects at CSU. These experiences reinforced the value of interactive demonstrations in effectively communicating engineering concepts and increasing student interest.

STEM outreach is important because early exposure to science and engineering strongly influences long-term academic and career pathways. Prior research shows that "children, whose interest is triggered in STEM at school level, generally pursue STEM career, if their interest is

sustained through the years" [11]. However, student interest in STEM has been observed to decline during middle and high school, highlighting the need for continued engagement beyond the classroom. One study, "Teachers' Perspectives on College Students as STEM Promoters," examined how college students can help address this gap through outreach initiatives. In this study, engineering students conducted interactive sessions with schoolchildren across more than 600 schools, while teachers evaluated their effectiveness in promoting STEM engagement. The results showed that college students were effective at both engaging younger students and improving their perception of STEM, with strong outcomes in sustained interest and active participation. The study concludes that college students can serve as impactful STEM promoters, particularly through interactive, hands-on outreach that complements traditional classroom instruction [11].

Rambots aims to engage younger students to maintain their interest in STEM or maybe spark their interest for the first time while teaching them something valuable about robotics or engineering. Most importantly, teaching them the fact that robotics is something they can do for very low cost because many of the means to do it and learn it are accessible.

An important finding regarding this year's additions including the arm and eyes is that young students are engaged more than ever in the project and asking questions to learn more. At E-days, many young students were interested in the new arm and enjoyed playing the simulator, emphasizing that it was a good decision to develop it. We hope to keep this trajectory going and continue to engage kids in the project with new exciting additions.

An additional application of this project is its potential to become something more like Boston Dynamics' robot dog, Spot. Spot is a versatile platform capable of navigating many types of terrain and environments, including environments humans cannot enter. The article

“Experimental evaluation of autonomous map-based Spot navigation in confined environments” [12] goes over how robots are used for exploration, inspection, and search and rescue missions where they face poorly lit, unstructured environments. Spot can be used for this application because its quadruped structure makes it able to navigate and recover from uneven terrain and it can be equipped with a lidar to enable simultaneous localization and mapping (SLAM) to map and navigate these areas. The Rambots team has been experimenting with lidar and SLAM, creating our own lidar module and interface, as well as creating research papers on the topic. Below is an example of our lidar interface which demonstrates detection of objects around the Rambot.

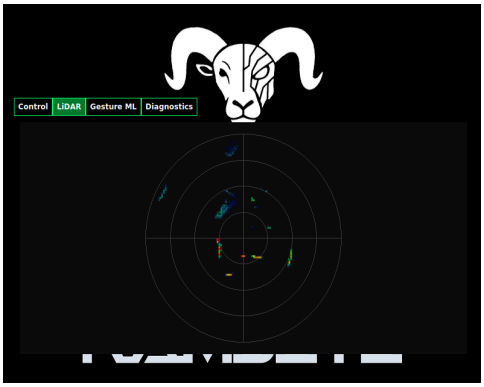


Figure 20: Rambot lidar interface

Furthermore, with the addition of the arm and the upcoming development of the Rambot Mark II, the team hopes future teams will be able to make the Rambot match the capabilities of Boston Dynamics’ Spot since it will utilize better hardware.

Reflection

This project was an extremely valuable experience that taught me many lessons, many of which I did not expect. The most important lesson was how to work with an interdisciplinary team and why it is important. The hardware and software side of the arm had so many considerations I never would have known to include and would have had us very stuck if it was not for the help and input of our electrical and computer engineering teammates and EIR mentors.

In addition, I developed a deeper understanding of the importance of power considerations in system design, particularly the need for consistency across components. Although power was included in our design matrices, it was primarily considered in terms of whether the system could be battery-powered, rather than whether the power requirements could be standardized across all components. As a result, different motors in the arm operated at significantly different voltage levels. For example, the wrist motor had a maximum voltage of approximately 8 volts, while the shoulder motor operated at up to 24 volts. Although this issue was ultimately managed through careful PCB design, it introduced unnecessary complexity that could have been avoided with more thorough planning. Additionally, the importance of component documentation became clear, as some motors, such as the Docyke motor in the shoulder, lacked sufficient documentation, making integration more difficult. Overall, this experience reinforced that engineering design involves more considerations than initially expected and that incorporating input from multiple disciplines is essential.

Additionally, this project strengthened my skills in timeline management. Successfully achieving project goals required developing realistic schedules, identifying key milestones, and

planning the sequence and duration of tasks needed to reach completion. I gained experience creating Gantt charts by defining a final deadline and working backward to allocate time for each phase of the project. This approach allowed for adjustments when additional time was available, enabling more flexibility in meeting intermediate deadlines.

The team also utilized weekly work plans to outline intended tasks and estimate the time required for completion. These plans were later updated to document the work accomplished and the actual time spent, providing a useful comparison between projected and real timelines. This process improved accountability and communication within the team, as it clearly conveyed individual responsibilities and progress. Weekly meetings were then used to review these plans, discuss challenges, and ensure alignment with overall project goals. Practice with these skills will greatly help in my future career and I am glad for the opportunity to practice skills that I will use in industry.

This will be a pivotal chapter in my life and educational career because I can look back on this and say this is the most rewarding and impactful project I have contributed to so far. The design work I did and the work the team did overall is something to be really proud of and will always stand as a testament to my capabilities to learn and make as an engineer.

Finally, the relationships I developed with my teammates are among the most meaningful outcomes of this experience. Their impact on my personal and professional development has been immeasurable, and I will carry forward the technical, communication, and planning skills I gained from this team into my future career.

Works Cited

[1] “8 Different Types of Robot Motors: What They Are, Why Use Them, and How to Choose Motor,” Mosrac.com, 2018. https://www.mosrac.com/resources/blog/robot-motors.html

[2] Ahmed, “Servo Motors,” Apr. 04, 2025. https://www.researchgate.net/publication/390492904_Servo_Motors

[3] H. Liu, *Robot systems for rail transit applications.* Elsevier, 2020. https://www.sciencedirect.com/book/monograph/9780128229682/robot-systems-for-rail-transit-applications

[4] K. Patel, “Stepper Motor Operation and Control: A Holistic Review,” *International Journal of Scientific Research in Science Engineering and Technology*, vol. 12, no. 5, pp. 187–193, Oct. 2025, doi: https://doi.org/10.32628/ijsrset25125412.

[5] Simplify3D, “Ultimate 3D Printing Material Properties Table,” Simplify3D, 2024. https://www.simplify3d.com/resources/materials-guide/properties-table/

[6] Z. B. Hazem, R. Ince, and S. Dilibal, “Joint control implementation of 4-DOF robotic arm using Robot Operating System,” in *2022 International Conference on Theoretical and Applied Computer Science and Engineering (ICTASCE)*, Ankara, Turkey, 2022, pp. 72–77, doi: 10.1109/ICTACSE50438.2022.10009733.

[7] D. P. Leins, R. Haschke and H. Ritter, "MuJoCo ROS: Integrating ROS with the MuJoCo Engine for Accurate and Scalable Robotic Simulation," 2025 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR), Palermo, Italy, 2025, pp. 1-6, doi: 10.1109/SIMPAR62925.2025.10979045.

[8] A. Vincze, “Involute spur gear generator and simulator,” *geargenerator.com*, 2014. https://geargenerator.com/

[9] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge, U.K.: Cambridge Univ. Press, 2017.

[10] S. Mathew, “Spot Robot : The Design details and working» Lesics Engineering Courses,” Jan. 04, 2024. https://sabinmathew.com/spot-robot-design-and-engineering/

[11] G. S. Mani, R. Jain, and T. Mote, “Teachers’ perspectives on college students as STEM promoters,” in 2024 First International Conference for Women in Computing (InCoWoCo), Pune, India, 2024, pp. 1–5, doi: 10.1109/InCoWoCo64194.2024.10863243.