Anthropomorphic Robot Manipulator with Gripper Claw and 4 DOF (Fall 2023)

Sidney Kantor
Electrical Engineering, University of North Dakota Member IEEE

***Abstract*—Ever since the purchase of my 3D printer many years ago, I have been fascinated with stepper motors. From CNC machines to pick-and-place machines, stepper motors boast incredible characteristics, such as torque, speed, and high-precision repeatability. In addition, the inherent design of stepper motors allows them to have little to no backlash. This characteristic, combined with the fact that stepper motors can be micro-stepped, makes them the ideal choice for robotics applications. With a background in mechanical design and embedded development, I set out to design and build a sophisticated and capable anthropomorphic robot manipulator with 4 degrees of freedom (DOF) that approaches industrial level performance. This paper documents my journey from the original concept to its ultimate realization.**

***Index Terms*—Robot Manipulator, Anthropomorphic, 4 DOF, Stepper Motors, Planetary Gear Reduction, STM32, Microcontroller, 3D Printed, Digital Stepper Driver, Gripper Claw Of Death**

I. INTRODUCTION

Stepper motors, amazing little machines, have always fascinated me, and using them to build a sophisticated anthropomorphic robot manipulator posed an exciting challenge. Specifically, I aimed to construct a manipulator with four degrees of freedom (DOF), with a servo-driven gripper-claw representing the 4th DOF. After extensive research, I refined my approach and gained a clear understanding of what was going to be required. My strategy involved utilizing stepper motors with built-in planetary gear reduction gearboxes to ensure the robot manipulator could lift objects of significant mass without losing steps. Working with these stepper motors, with their unique physical form, required a custom design for the robot manipulator links and joints. I did the design in Fusion 360, and 3D printed the parts on my Prusa 3D printer.

To drive these power-hungry stepper motors, I sourced three high-end, highly configurable digital stepper drivers, powered by my benchtop Siglent SPD3303X-E DC power supply. Although I have experience in embedded development using Arduino and Microchip PICs, I had been eager to explore the STM32 embedded development ecosystem. Consequently, for the controller, I sourced an STM32 Nucleo development board, capable of running at 180 MHz, and employed the STM32CubeIDE for all firmware development.

II. DESIGN/BUILD

A. Stepper Motors

The stepper motors utilized have planetary gear reduction gearboxes with 19.19:1 ratio allowing them to produce incredible torque. Dimensions of the motor were taken using electronic calipers and they were used when designing the 3D printed links and joints.

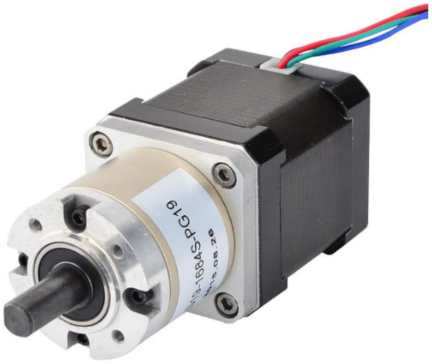


Fig. 1. High Torque NEMA 17 Stepper Motor with 19.19:1 Planetary Gearbox

Flanges were sourced that provide a strong connection between the stepper motor shaft and the 3D printed joints.

Fig. 2. Shaft Flanges

B. Electronics

For past projects with stepper motors I have used inexpensive motor drivers that were based on the A4988 IC. These were fine for small projects but had power and reliability issues. Therefore, for this project, I went with three DM542T high-end digital stepper drivers from StepperOnline. These drivers can take up to 50 volts input and can source nearly 5 amps of current. The drivers are nearly silent and the fine control these drivers offer is remarkable.

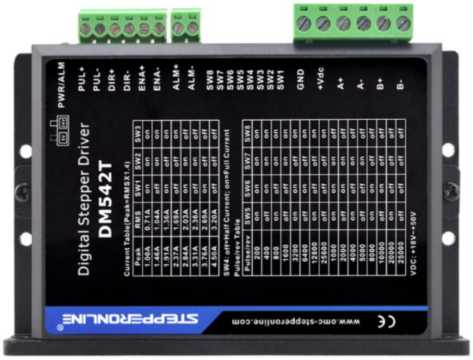


Fig. 3. Digital Stepper Driver DM542T

To power all of the electronics, a benchtop Siglent SPD3303X-E DC power supply was used.

Fig. 4. Siglent SPD3303X-E DC Power Supply

To ensure that I was able to utilize all of the performance of the high-end hardware, I chose a 32 bit microcontroller from STM that is capable of running at 180 MHz. This allowed me to use complex and resource intensive algorithms for controlling the acceleration and deceleration of the stepper motors without any performance issues.

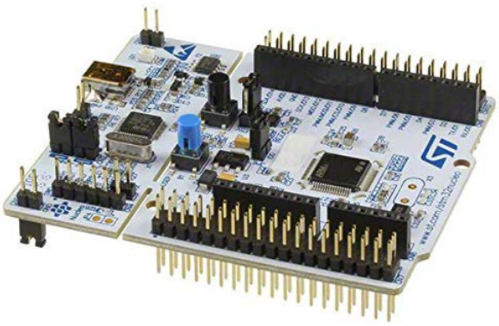


Fig. 5. STM32 Nucleo-64 Development Board

C. 3D Printed Components

Every robotic manipulator needs to have a strong foundation, so for the base I designed and 3D printed a structure that would allow the stepper motor body to be inserted and kept secure via an interference fit and locking rim. This is the first of three revolute joints and it allows the robotic manipulator to rotate about the z-axis.

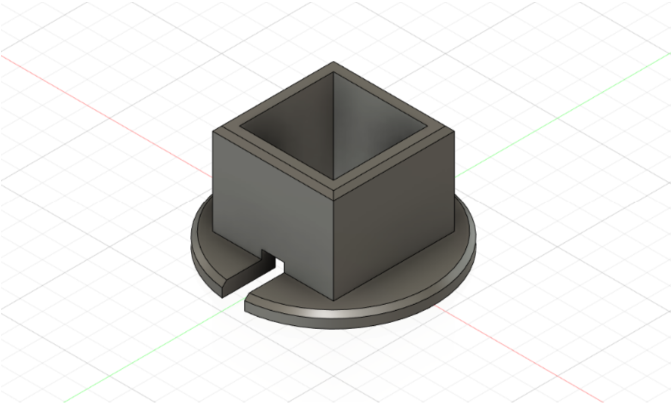


Fig. 6. 3D Printed Base

Brackets were designed and 3D printed that connect the stepper motors to other stepper motors or links. It can be seen that the recessed pockets allow the shaft flanges to seat very tightly into the bracket eliminating any backlash or play.

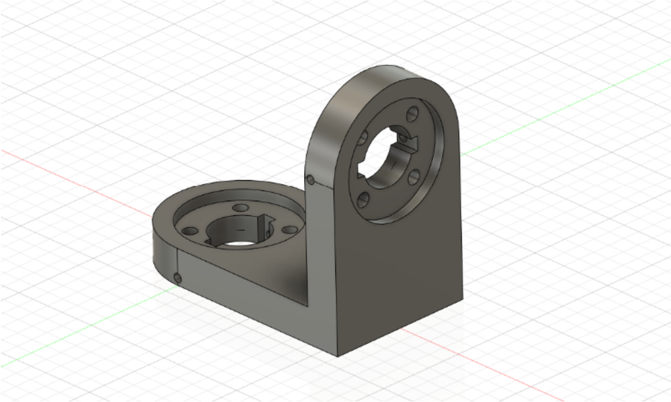


Fig. 7. 3D Printed Bracket

This is the main link that connects and holds two of the steppers motors rigidly. It was designed and 3D printed as well.

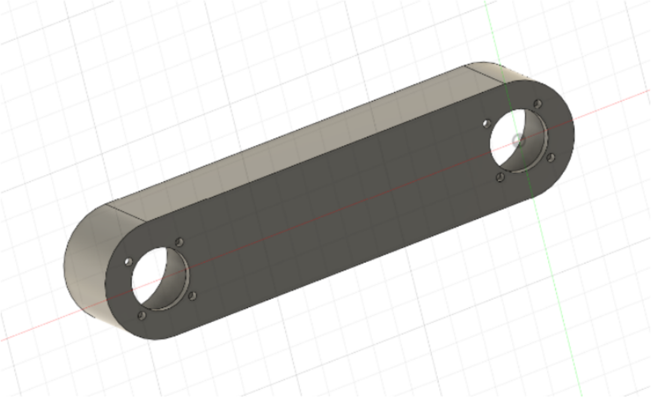


Fig. 8. 3D Printed Link

This is the final link in the sequence and it connects the 3rd revolute joint to the gripper-claw. It was designed and 3D printed to allow for a seamless attachment to the gripper-claw. It would have been nice to have a 4th revolute joint here to allow the gripper-claw to have wrist-like movement, however, I didn't want to over-complicate an already complex design. However, I may add one in the future.

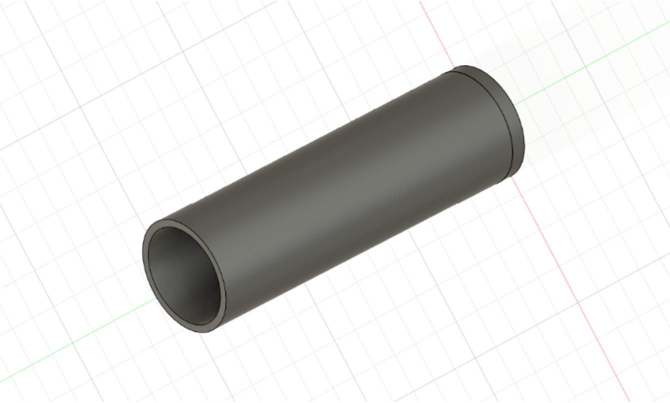


Fig. 9. 3D Printed Gripper Link

D. Gripper-Claw of Death

This is the gripper-claw of death. Well, not really. But it is a gripper-claw that is driven by a servo and serves as the robotic manipulator's end effector. Extensive research was done looking for the perfect gripper-claw but there are not a lot of good quality gripper-claws available at hobby-level prices. I started designing a custom gripper-claw, however due to time constraints, I chose to work with this gripper-claw.

Fig. 10. Gripper Claw of Death

E. Firmware

I developed the firmware from scratch, creating a robust command line interface for efficient communication with the micro-controller. While I won't delve deeply into technical specifics, some highlights include an object-oriented design where each link in the system is an autonomous object. This architecture enables each link to be aware of its own state as well as that of its connected links, facilitating precise control over commanded angles and synchronized joint movements. A key innovation is the implementation of a 'Waypoint', a class that encapsulates vital information about the robot's position and gripper state. This design allows for the construction of a Waypoint queue, streamlining the execution of movement sequences. The system processes each Waypoint in the queue, ensuring smooth acceleration and deceleration for each movement, thereby achieving efficient and coordinated robotic operations.

F. Specifications

- Anthropomorphic with 3 revolute joints and a gripper-claw (4 DOF)
- NEMA 17 Stepper Motors with Planetary Gear Reduction 19.19:1
- Backlash at No-load: $\leq 1°$
- Maximum Permissible Torque: 3 Nm (425 oz-in)
- Momentary Permissible Torque: 5 Nm (708 oz-in)
- 800 steps/revolution @ motor shaft
- 800 * 19.19 = 15352 steps/revolution @ output shaft
- Resolution: 360°/ 15352 = 0.0235°
- DM542T Digital Stepper Drivers
- Range of motion
- Waist Joint: $-270° \leq \theta \leq 270°$
- Shoulder Joint: $-26° \leq \theta \leq 180°$
- Elbow Joint: $-110° \leq \theta \leq 110°$
- Gripper-Claw: $0° \leq \theta \leq 100°$

G. Robot Manipulator

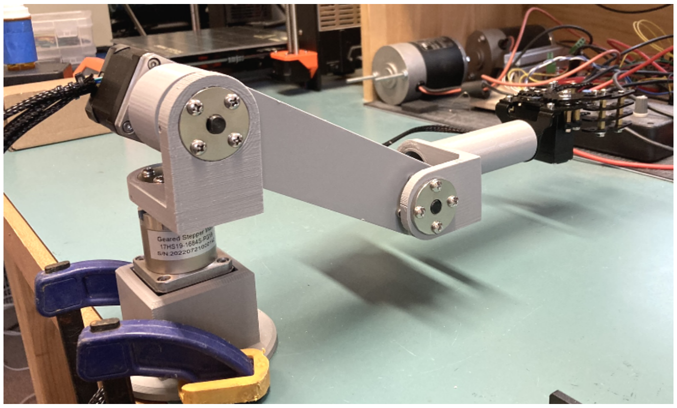


Fig. 11. Assembled Robot

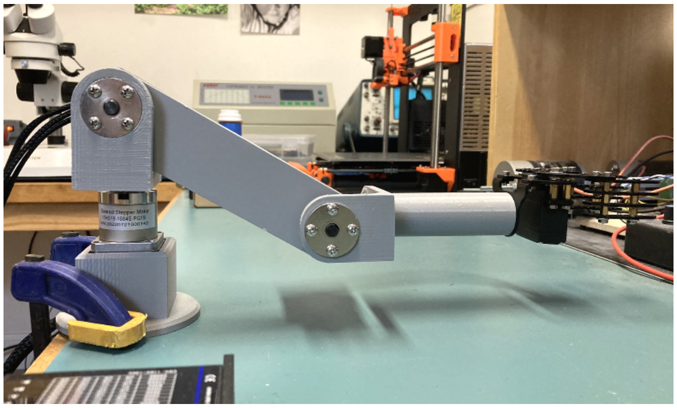


Fig. 12. Assembled Robot

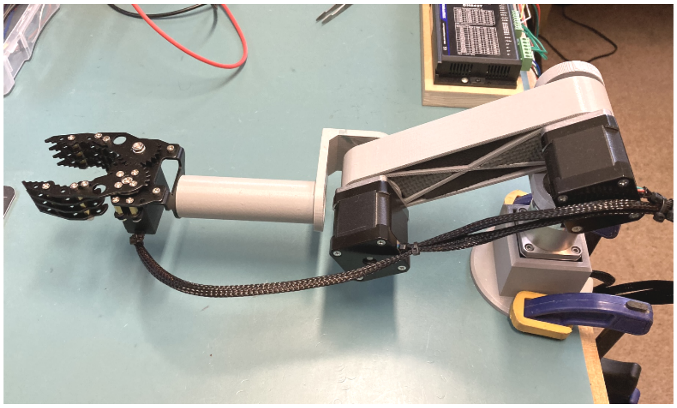


Fig. 13. Assembled Robot

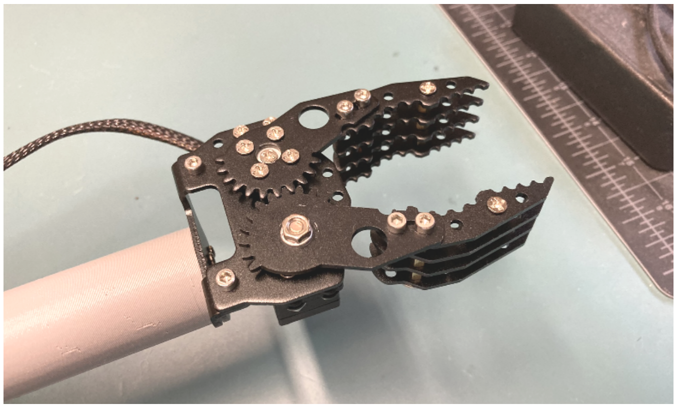


Fig. 14. Gripper Claw End Effector

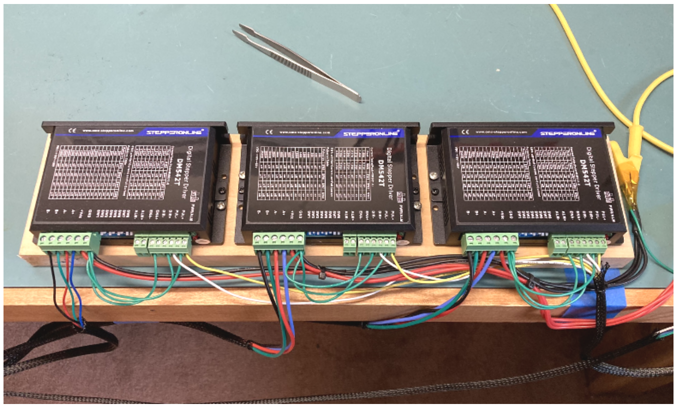


Fig. 15. Stepper Drivers

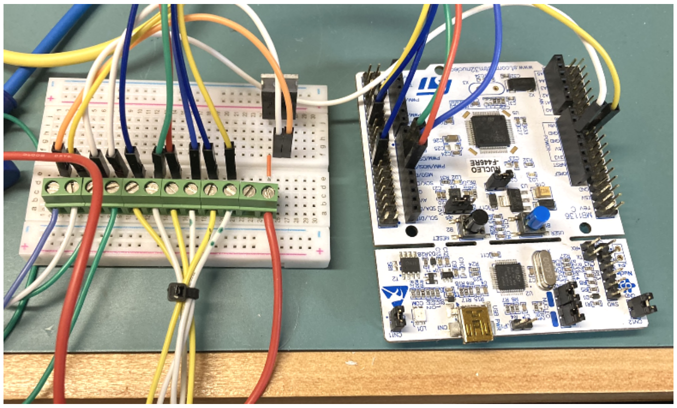


Fig. 16. Micro-controller and Bus Board

III. Forward Kinematics

A. Diagrams

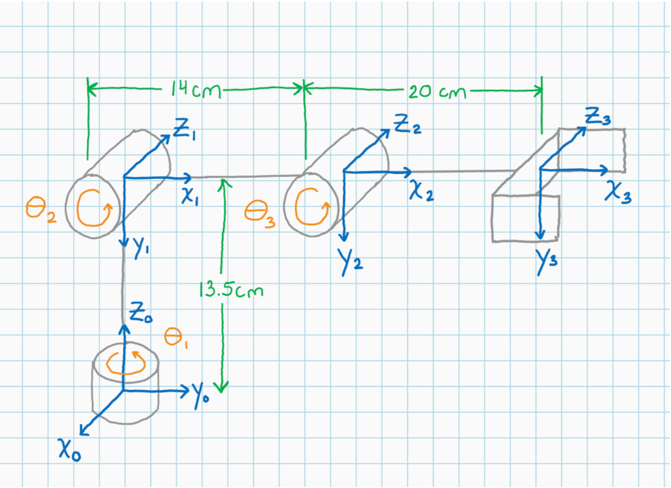
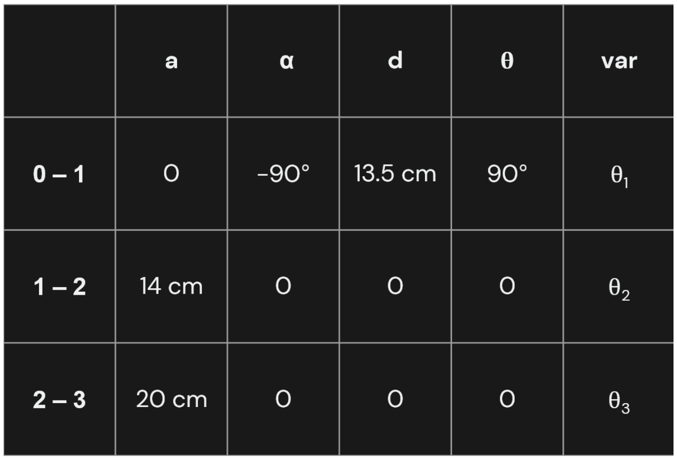


Fig. 17. Kinematic Diagram

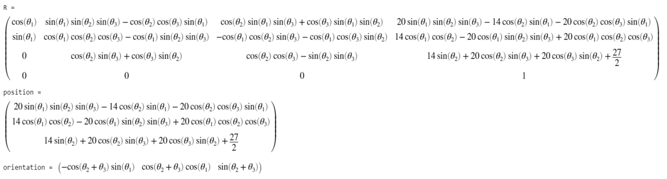


	a	α	d	θ	var
0 – 1	0	-90°	13.5 cm	90°	θ_1
1 – 2	14 cm	0	0	0	θ_2
2 – 3	20 cm	0	0	0	θ_3

Fig. 18. DH-Parameters

B. Transformation Matrices

Although the DH-parameters work well for the forward kinematics I preferred using the homogeneous transformation matrices as they seemed more intuitive.



R =

$$\begin{pmatrix} \cos(\theta_1) & \sin(\theta_1)\sin(\theta_2)\sin(\theta_3) - \cos(\theta_2)\cos(\theta_3)\sin(\theta_1) & \cos(\theta_2)\sin(\theta_1)\sin(\theta_3) + \cos(\theta_3)\sin(\theta_1)\sin(\theta_2) & 20\sin(\theta_1)\sin(\theta_2)\sin(\theta_3) - 14\cos(\theta_2)\sin(\theta_1) - 20\cos(\theta_2)\cos(\theta_3)\sin(\theta_1) \\ \sin(\theta_1) & \cos(\theta_1)\cos(\theta_2)\cos(\theta_3) - \cos(\theta_1)\sin(\theta_2)\sin(\theta_3) & -\cos(\theta_1)\cos(\theta_2)\sin(\theta_3) - \cos(\theta_1)\cos(\theta_3)\sin(\theta_2) & 14\cos(\theta_1)\cos(\theta_2) - 20\cos(\theta_1)\sin(\theta_2)\sin(\theta_3) + 20\cos(\theta_1)\cos(\theta_2)\cos(\theta_3) \\ 0 & \cos(\theta_2)\sin(\theta_3) + \cos(\theta_3)\sin(\theta_2) & \cos(\theta_2)\cos(\theta_3) - \sin(\theta_2)\sin(\theta_3) & 14\sin(\theta_2) + 20\cos(\theta_2)\sin(\theta_3) + 20\cos(\theta_3)\sin(\theta_2) + \frac{27}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

position =

$$\begin{pmatrix} 20\sin(\theta_1)\sin(\theta_2)\sin(\theta_3) - 14\cos(\theta_2)\sin(\theta_1) - 20\cos(\theta_2)\cos(\theta_3)\sin(\theta_1) \\ 14\cos(\theta_1)\cos(\theta_2) - 20\cos(\theta_1)\sin(\theta_2)\sin(\theta_3) + 20\cos(\theta_1)\cos(\theta_2)\cos(\theta_3) \\ 14\sin(\theta_2) + 20\cos(\theta_2)\sin(\theta_3) + 20\cos(\theta_3)\sin(\theta_2) + \frac{27}{2} \end{pmatrix}$$

orientation = $\begin{pmatrix} -\cos(\theta_2+\theta_3)\sin(\theta_1) & \cos(\theta_2+\theta_3)\cos(\theta_1) & \sin(\theta_2+\theta_3) \end{pmatrix}$

Fig. 19. Homogeneous Transformation Matrix, Position Matrix and Orientation Matrix

IV. Inverse Kinematics

To solve for the inverse kinematics the graphics approach using trigonometry was utilized which resulted in the equations below.

A. Diagrams

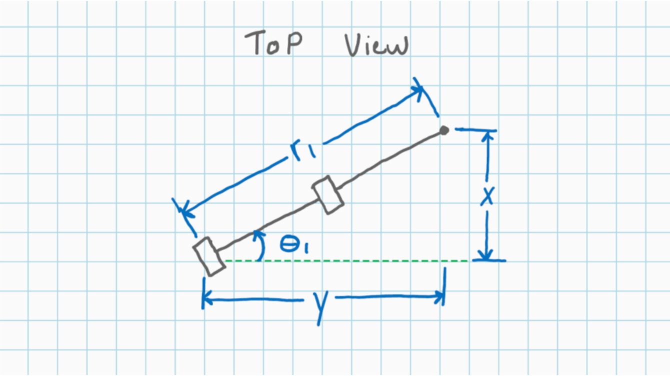


Fig. 20. Top View

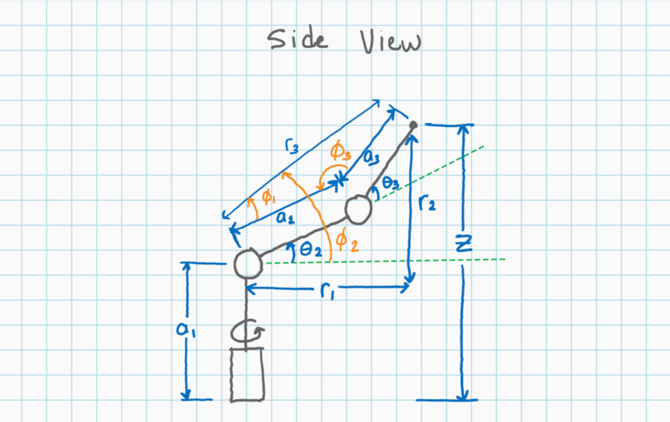


Fig. 21. Side View

B. Equations

$$\theta_1 = -atan2(x, y)$$

$$r_1 = sqrt(x^2 + y^2)$$

$$r_2 = z - a_1$$

$$\phi_2 = atan2(r_2, r_1)$$

$$r_3 = sqrt(r_1^2, r_2^2)$$

$$\phi_1 = acos((a_3^2 - a_2^2 - r_3^2)/(-2 * a_2 * r_3))$$

$$\theta_2 = \phi_2 - \phi_1$$

$$\phi_3 = acos((r_3^2 - a_2^2 - a_3^2)/(-2 * a_2 * a_3))$$

$$\theta_3 = \pi - \phi_3$$

V. SIMULATION

The simulation was done in Matlab using the Robotics Toolkit. This allowed me to build a robot manipulator based on a rigid body tree and I could add bodies and joints to this model using parameters that reflected the real world dimensions of the robot manipulator. Although I only used the most basic of this functionality, the Robotics Toolkit is quite capable and very powerful.

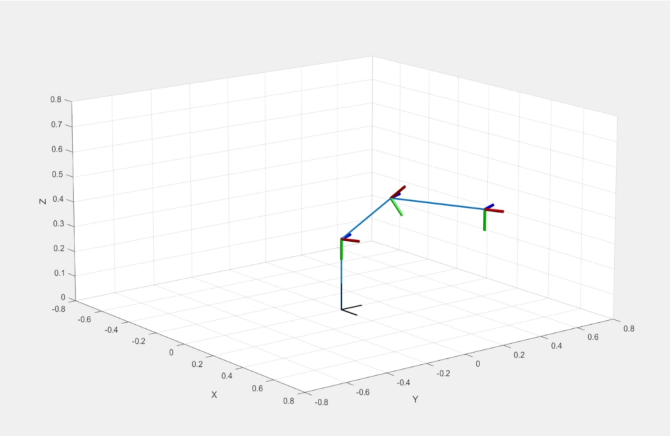


Fig. 22. Simulation in Matlab

VI. DEMONSTRATION

The main goal of the robotic manipulator was to swap two bottles that were sitting within the workspace. It would pick up the first bottle from it's platform and move it to a staging platform. It would then pick up the second bottle and place it at the first bottle's original location. It would then pick up the first bottle from the staging platform and place it at the second bottle's original location. The demonstration performed this swapping challenge at normal and ludicrous speeds. I say ludicrous because any faster and the robot would self-destruct. I was extremely satisfied with the performance of this robotic manipulator. The movements were extremely precise, insanely fast and repeatable. I feel strongly that I accomplished my original goal.

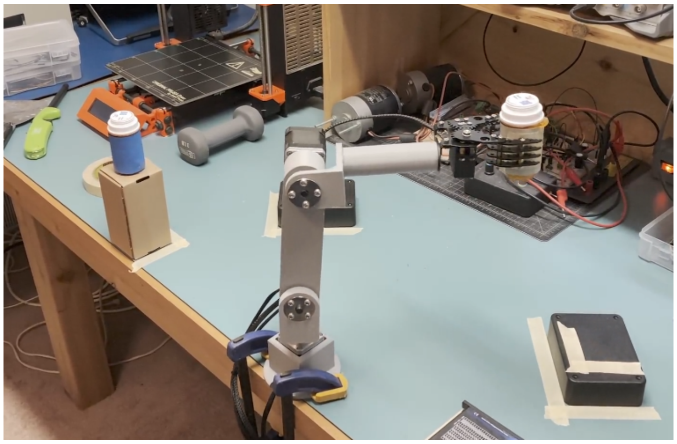


Fig. 23. Robotic Manipulator Swapping Two Bottles

VII. DISCUSSION

The simulation and the demonstration were nearly identical. The simulation was done first and the final demonstration was slightly different only due to the fact that the demonstration had much more advanced movements than the simulation.

VIII. CONCLUSION

Below are two key takeaways from this project regarding backlash and end effectors.

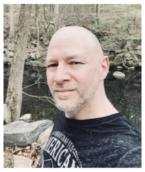
A. Backlash

Using stepper motors with planetary gear boxes was paramount to being able to build a sophisticated and capable robotic manipulator. However, even though the robot's joints had a resolution of 0.0235°, this high precision was negated due to the backlash in the gears. Backlash is the amount of clearance or "play" between mating gear teeth and it can make it difficult to get high precision and repeatable movements. There are drive mechanisms available that have little to no backlash, however they come at a premium cost.

B. End Effector

I was not satisfied with the end effector that was used in this project. It worked but not without issue. I started designing my own end effector, however due to time constraints I made the decision to just make the end effector that I had work. There were not many options for quality end effectors at the hobby level price point. After much research I also learned the challenges that come with choosing an end effector. The key takeaway here is that there is not one end effector that will work for all applications. End effectors have to be carefully chosen based on the specific application they are intended for.

ABOUT THE AUTHOR



Sidney Kantor has over 25 years' experience in the software industry as a software engineer. He held a role at his current company working in a research and development position, investigating the potential application of artificial intelligence within the context of the company's needs. Within this role he implemented a neural network using Python and Jupyter Notebook using only the native capabilities of the language. The neural network was trained on image recognition and was able to accurately identify handwritten numerical values zero through nine, regardless of the orientation. He is currently studying towards earning a Bachelor of Science (B.S) in electrical engineering from the University of North Dakota.