Preliminary Design Report

Trey Dufrene Zack Johnson David Orcutt Alan Wallingford Ryan Warner

Submitted in Partial Fulfillment of the Requirements of:

ME407 Preliminary Design – Fall 2019

Meiosis

College of Engineering
Embry-Riddle Aeronautical University
Prescott, AZ

Abstract

The Manipulator for Educational Institutions with Open Source Integrated Systems (MEIOSIS) aims to increase the accessibility of robotics to secondary educational institutions and hobbyists. Accordantly, the manipulator is 3D printed in PLA with aluminum tube supports and costs the end-user less than $1000. The manipulator has six links and a base. The base houses a Raspberry Pi 3B and power supply. The Raspberry Pi controls seven Dynamixel smart servos with position feedback and proportional derivative control. Six MX-12W servos actuate six rotational joints, while one AX-12A servo actuates the removable end-effector. They provide the manipulator a position repeatability within 2mm of the previous pose. The manipulator can draw as well as perform pick and place operations within its dexterous workspace which is a hemispherical sub-shell of the reachable workspace of 280 mm thickness. The manipulator's operation is controlled by open-source software.

Contents

List of Figures

List of Tables

List Of Acronyms and Abbreviations

FK	:	Forward Kinematics
IK	:	Inverse Kinematics
PD	:	Proportional Derivative

Notation

$^{\text{Frame}}_{\text{From}}$	r $_{\text{To}}$	:	Direction Vectors
$^{\text{From}}$	T $_{\text{To}}$	:	Direction Cosine (Transformation) Matrices
	$c_{\theta_{nm}}$	:	$\cos(\theta_n + \theta_m)$
	$s_{\theta_{nm}}$	:	$\sin(\theta_n + \theta_m)$

1 Introduction

The Rethink Robotics Sawyer manipulator retails for $29,000 [4]. The Prescott Embry-Riddle robotics lab has 1 and the University's average student pays $35,654 annually – after financial aid [2]. Further, $25,000 to $35,000 is a common price for manipulators with a reach of approximately 1 meter and payload capacity below 1 kg [3]. While Trossen Robotics offers low cost manipulators between one sixth and one sixtieth of Sawyer's cost, for accuracy of 1mm, the cost is still around $2,000. For secondary education institutions, robotics is cost prohibitive. Consequently, students have less opportunities in high school to learn about robotics. The Manipulator for Educational Institutions with Open Source Integrated Systems (MEIOSIS) aims to increase the accessibility of robotics education to secondary education institutions and hobbyists.

In biology, meiosis is a process of cell division similar to mitosis. However, unlike mitosis, the resulting daughter cells are used to create new life during sexual recombination. Meiosis occurs in two sets of cell division: meiosis I and meiosis II. Analogously, MEIOSIS has a Preliminary and Detail semester. During meiosis I, chromosomes pair up and swap genetic material (DNA) in preparation for cell division. Then, during meiosis II, the parent cell splits into two haploids, which each split into two gametes. The gamete cells have half the number of chromosomes compared to a cell produced by mitosis and are not identical to their parent cell. When combined with another gamete, a new, unique cell emerges. MEIOSIS does not change the DNA of robotics. However, in combining with user needs and incorporating innovative design elements, MEIOSIS brings new life to robotics education.

This document outlines MEIOSIS's solution for increasing the accessibility of robotics education, current project status, and plans for the Detail semester. Specifically, it is subdivided into four sections: Requirements, Conceptual Design, Specifications, and Preliminary Design. The Preliminary Design comprises the majority of the document. It explains the evolution of MEIOSIS's manipulator, noting design decisions and changes made from simulations, tested parts, and motor calculations. Finally, an appendix contains simulation code and drawings for all parts.

2 Requirements

The requirements of the system concisely define the capabilities the system must possess in order to solve the stated problem.

2.1 Hardware

The following requirements are hardware specific and dictate the physical constraints the system must adhere to.

2.1.1 The system shall cost the end-user no more than $1000.

2.1.2 The system shall be fully dexterous without being kinematically redundant.

To create a system with the intention of advancing education, it must be complex enough to encourage higher level problem solving, as well as be capable enough (dexterous) in a broad spectrum of tasks — in the interest of remaining useful in addition to retaining the interest of students.

2.1.3 The system end effector shall maintain a positional accuracy magnitude of ± 1 mm and an orientation accuracy of $\pm 5^\circ$ eigen angle from the base frame.

To ensure that the robot has educational value, the accuracy must be defined so that any desired positions and movements are achieved.

2.1.4 The system end effector shall maintain a pose repeatability magnitude between 0.1—1.5 mm for the position and $\pm 4^\circ$ eigen angle from the base frame for the orientation.

This is to ensure a robot that can execute the same movement commands repeatedly and have the same results every time.

2.1.5 The system's reachable workspace shall be a hemisphere with a radius of 300-700 mm.

This workspace will provide enough movement to manipulate objects in order to perform basic tasks.

2.1.6 The system's dexterous workspace shall contain a hemispherical shell within the reachable workspace with a thickness of 280 mm.

2.1.7 The system shall have a removable end effector capable of picking and placing a low-odor chisel tip Expo dry erase marker.

This creates a robot capable of performing a variety of basic tasks, which enhances its educational value.

2.1.8 The system shall be able to write with a low-odor chisel tip Expo dry erase marker.

2.2 Software

2.2.1 The system shall be open source.

This will create an easily obtainable, low cost method of distributing the system's source code, which may be modified for personal use.

2.2.2 The system shall be capable of operating given only desired end effector cartesian coordinates specified with respect to the base frame.

This simplicity makes the system of use to inexperienced users.

3 Conceptual Design

The terminator T-2000 is a science-fiction spectacle of a robot – until you see the price. Channeling the inspiration many high school students may have for robotics, MEIOSIS robotics aims to provide an affordable manipulator to educators and enthusiasts. MEIOSIS uses primarily 3-D printed components and easily accessible materials. Among these materials are a Raspberry PI, smart servos and metal tubing. These features create an open-source manipulator accessible to the public to further robotics education.

3.1 Physical System Overview

The physical design of the robotic manipulator is shown in Figures *1, 2, 3,* and *4*.

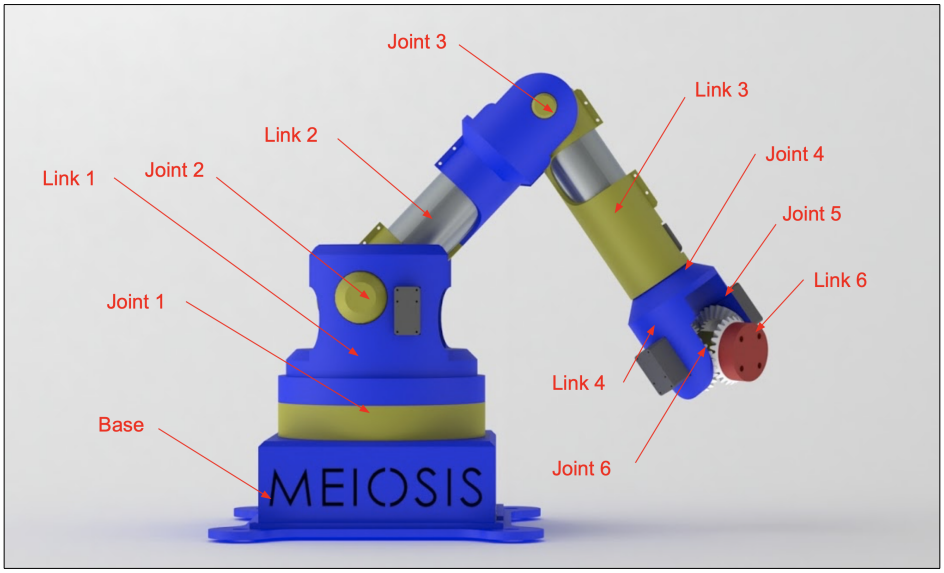


Figure 1: Overall System Conceptual Design

The colored links in *Figure 1* distinguish the different joints and links of the manipulator. The overall reach of the robot is 582.5 mm. This length was chosen to decrease material cost and weight while still satisfying requirement 2.1.2 and 2.1.5, allowing the manipulating to pick and place objects to perform basic tasks. The base of the robot is made to contain the Raspberry Pi and other electrical components.

3.1.1 Base

The base of the manipulator will house several of the electronic components, such as the computational system, power supply, and motor controller. A cross section of the base can be seen in *Figure 2*.

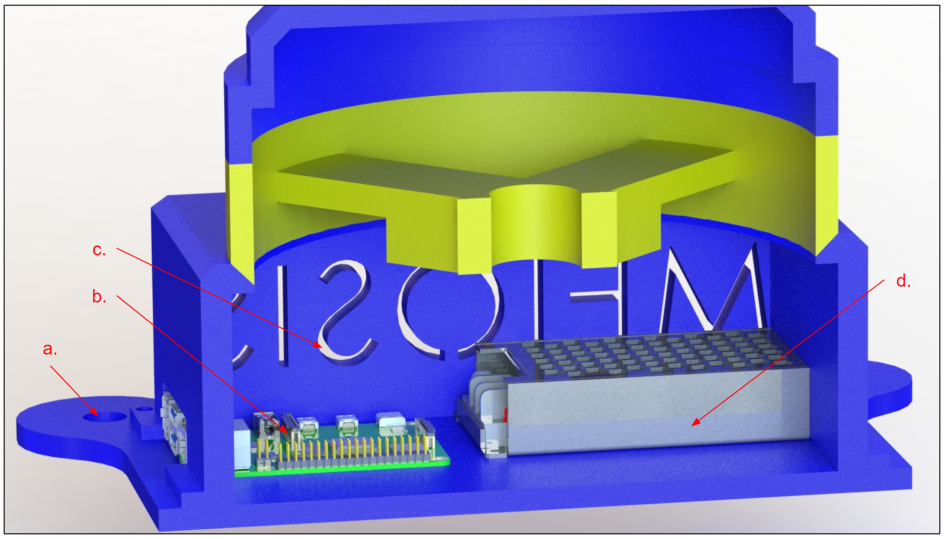


Figure 2: Manipulator Base with Call-outs

From *Figure 2*,

a. Base Supports: The base supports are located at each corner of the base and allow the base of the manipulator to be securely attached to a variety of surfaces with either standard fasteners or suction cups.

b. Computational System: The computational system is a Raspberry Pi; it is housed in the base, which allows the Raspberry Pi to be more easily accessible. The primary reason for this system being chosen is to fulfill the budget requirement, 2.1.1. The Raspberry Pi computes the manipulator's inverse kinematics and sends the angle commands to the servos.

c. Airflow Cutouts: The side of the base hsd cutouts to allow for airflow through the base; since the power supply is housed inside of the base as well as the computational system, the temperature must be regulated to prevent overheating.

d. Power Supply: The power supply is housed in the base as well, which allows the power supply to be more accessible and therefore more modifiable, so the end-user can easily expand the system to fulfill their needs.

3.1.2 Links

Figure 3 shows the links and their key features.

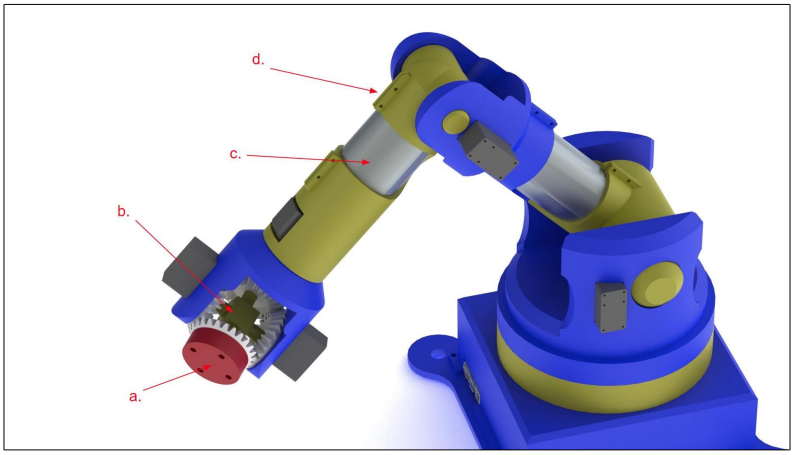


Figure 3: Drawing Showing Key Features of Design

In *Figure 3*, call-out (a) shows the connection point for the end effector. The mounting layout is the standard used by the Sawyer manipulator. The end effector mounting layout may be adjusted to accommodate lower cost, more accessible end effectors. Call-out (b) shows the differential gearbox that is used in the manipulator's wrist, saving space and weight. The manipulator has aluminum tubing as support in the links (c) and are attached to the 3D printed portion of the robot using clamp joints (d) tightened by screws.

Figure 4 depicts the cross section of link 2 for the manipulator.

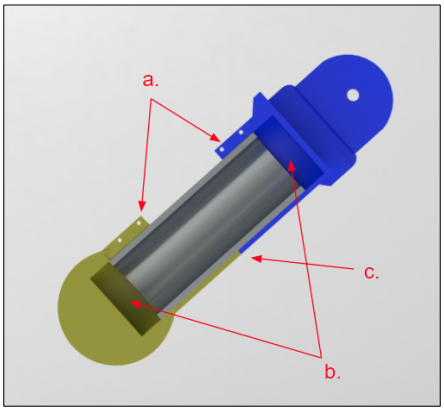


Figure 4: Drawing Showing Link Cross Section

The cross section seen in *Figure 4* shows the internal design for links two and three. It features two clamps that hold a hollow aluminum bar in place (a) and allows for gaps between the aluminum tube and the 3D printed call-out (b). The proper length is dictated

by the 3D printed guides lining up at call-out (c). This design allows for imprecision in the manufacturing of the aluminum tube.

3.2 System Functions

The system can be divided into two subsystems: the electrical and software systems. The electrical subsystem includes the wiring and hardware computational components, power system, actuators with drivers, and sensors. The software subsystem includes the algorithm for the computational system.

3.3 Electrical

Figure 5 is the block diagram for the electrical system of the manipulator.

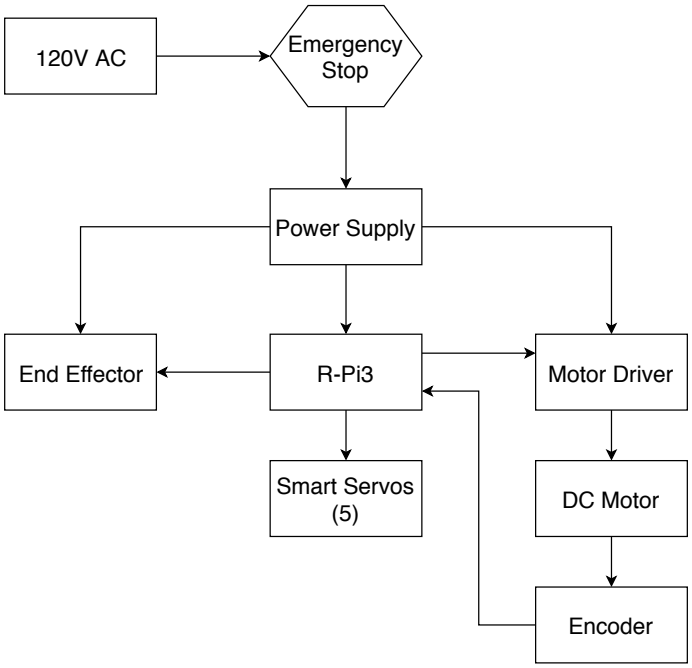


Figure 5: Electrical System Block Diagram

Figure 5 shows that the electrical systems of the manipulator consists of three main components: the power supply, the Raspberry Pi, and the servos. Power is supplied by the 120V AC from standard wall outlets. A power supply converts the AC voltage to the required voltages for each component. One component is the Raspberry Pi, which performs calculations for motor control. The Raspberry Pi sends signals to the DC motor driver and the five smart servos. The smart servos have an on-board controller, so no feedback will be necessary. However, the first joint, between the base and the first link, is actuated by a DC motor with an encoder to minimize cost.

3.4 Software

Figure 6 shows the software flowchart for the system.

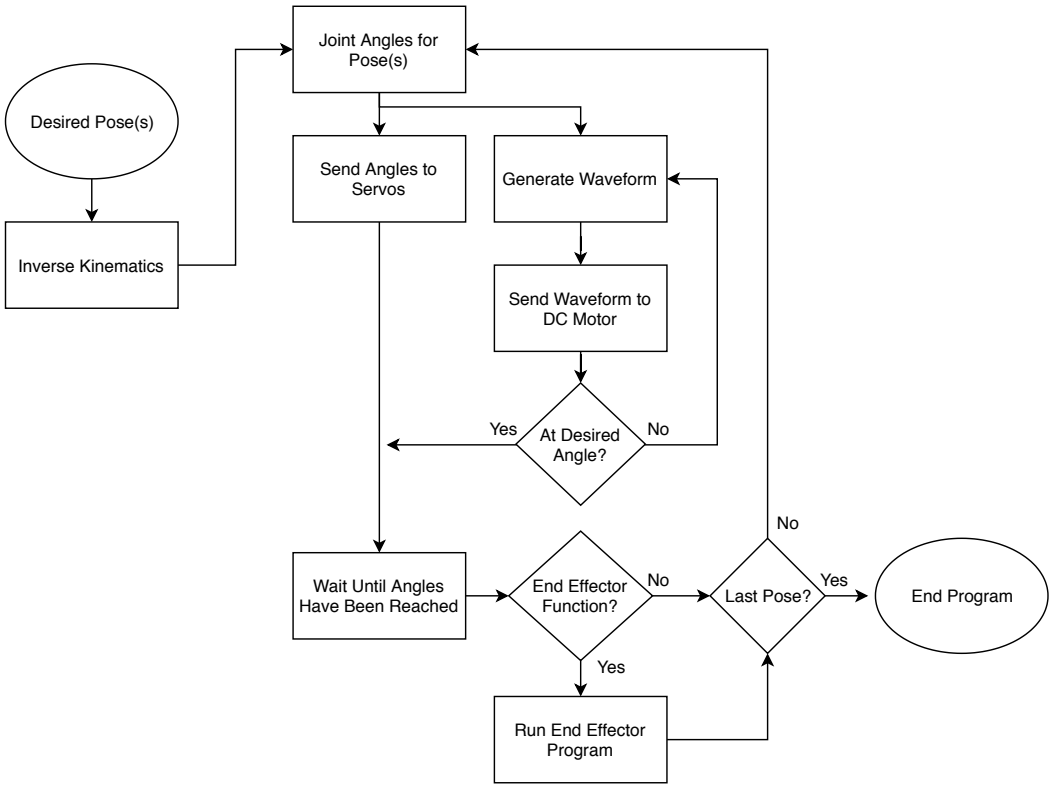


Figure 6: Software Flowchart

Figure 6 shows that the software receives the desired pose or poses that the user would like the manipulator to reach. Next, the Raspberry Pi uses inverse kinematics to calculate the necessary joint angles. The wave-forms/desired angles are sent to the respective drivers/motors, and positional information is be sent back to the Raspberry Pi to adjust the DC motor angle. When the motors have reached their desired pose, the Raspberry Pi actuates the end effector if it is specified by the user. The system then checks to see if there are any more poses to reach and either repeats the steps in the motor control section given the desired angles of the new pose or ends the program if the last pose has been reached.

3.5 Decision Matrices

Each team member's conceptual design provides a variety of different options for select issues, and in order to narrow down the options into the best choice decision matrices are used to make unobjective decisions about each individual topic. The weights of each criteria are defined by the average of our votes on how important the criteria is on a scale of one to ten. The same method is used to rate the different options on each criteria. The most important decision that needs to be made is what configuration the manipulator will have as the configuration has an affect on almost all the design choices. There are five potential configurations, and the scores of each option can be seen in *Table 1*.

The results from *Table 1* show that the solely rotational configuration is the best option,

Table 1: Configuration Decision Matrix

Configuration	**Cost**	**Educational Value**	**Ease of Use**	**Ease to Program**	**Ease of Manufacture**	**Total**
Weighting	10	8	7	2	6	
Cartesian w/ Spherical Wrist	5	9	10	8	4	232
RRR w/ Spherical Wrist	8	9	7	7	8	**263**
Cylindrical	7	10	7	8	7	257
SCARA	6	9	8	8	6	240
Spherical	7	10	7	7	6	249

followed closely by the cylindrical. Having translational joints is too be problematic for manufacturing and more expensive, and that is enough of an issue that the rotational manipulator has the advantage over the rest of the options.

Another design choice that needs to be made is what material the manipulator will be constructed out of. The main choices are to 3D print everything, manufacture the manipulator out of aluminum, or use a combination of the two. Cost and accessibility are the two highest weighted criteria as we deem them to be the most important aspects, followed by weight and manufacturability. Durability has the lowest weight because the manipulator should not have enough forces acting on it so that the strength of PLA compared to aluminum should not necessarily affect the manipulator. The grades of each material are shown in *Table 2*.

Table 2: Material Choice Decision Matrix

Material	**Cost**	**Weight**	**Accessibility**	**Manufacturability**	**Durability**	**Total**
Weighting	9.2	6.6	8	6.4	5.4	
3D Printed	8	9	8	9.4	5	284.16
Aluminum	4	5.8	5.8	6.6	9.2	213.4
Combination	8	7.8	7	9	9.2	**288.36**

As seen in *Table 2*, the winning material is a combination of 3D printing and aluminum. The combination allows the manipulator to have 3D printed parts that would be hard to manufacture out of aluminum or that would not necessarily need to be extremely durable, and allows the use of aluminum for the aspects of the manipulator that need to be strong and durable. This option allows for the best aspects of both materials to be utilized.

The internal computing system for the manipulator is another decision that needs to be made, so another decision matrix is used to decide from the six options available. Because the manipulator is intended to be low cost and used by primary education institutions, cost and ease of use are the two highest weighted criteria. Having a lot of capabilities and being able to function independently are factored in, but are deemed of secondary importance which can be seen in *Table 3*.

Table 3: Computing Choice Decision Matrix

Computing System	**Cost**	**Capabilities**	**Ease of Use**	**Independence**	**Total**
Weights	10	4.2	7.6	4.2	
RPi	7.9	6	9	10	**214.6**
Jetson	1	7	7	9	130.4
RPi + Arduino	4.8	7	8	10	180.2
Arduino	9	4	9	3	187.8
B Black	1.2	7	7	10	136.6
B Green	5	7	6	9	162.8

Table 3 shows that the Raspberry Pi is the winner by a significant margin, with only the Arduino coming close to it. While the Pi is not the most capable, its cost is very good and its ease of use and ability to independant functionality edge it over the other more capable competition. The arduino is the most cost effective, but its lack of functionality and low independence are detrimental to its score.

One of the other designs that needs to be selected was the way that the end effectors are attached to the end of the manipulator, and the different categories that are used to grade each option are ease of use, manufacturability, and durability, as seen in *Table 4*.

Table 4: End Effector Attachment Design Decision Matrix

EE Attachment	**Ease of Use**	**Manufacturability**	**Durability**	**Total**
Weighting	3.6	5	7	
Screw Connections	6.8	8.8	9.8	**137.08**
Snap Fit Joint	8.6	5	2.2	71.36
Threaded End Effector	6.3	5.4	7.3	100.78

From *Table 4*, the optimal design turns out to be screw connections in the end effector. The snap fit joint would be easier to use, but to create a functional snap fit connection would be difficult and would likely not be too durable. The threaded end effector is difficult to manufacture and not durable as well, so having connections for screw on the end effector is the clear choice.

4 Specifications

With the intention of making robotics education more accessible, The Manipulator for Educational Institutions with Open Source Integrated Systems (MEIOSIS) intends to provide high school educators and robot enthusiasts with a low cost manipulator. The system should be usable by novice students. It should also be modifiable to create a sustainably increased understanding of robotics. While MEIOSIS may not fully emulate industrial manipulators, it aims to provide more students with access to robotics education.

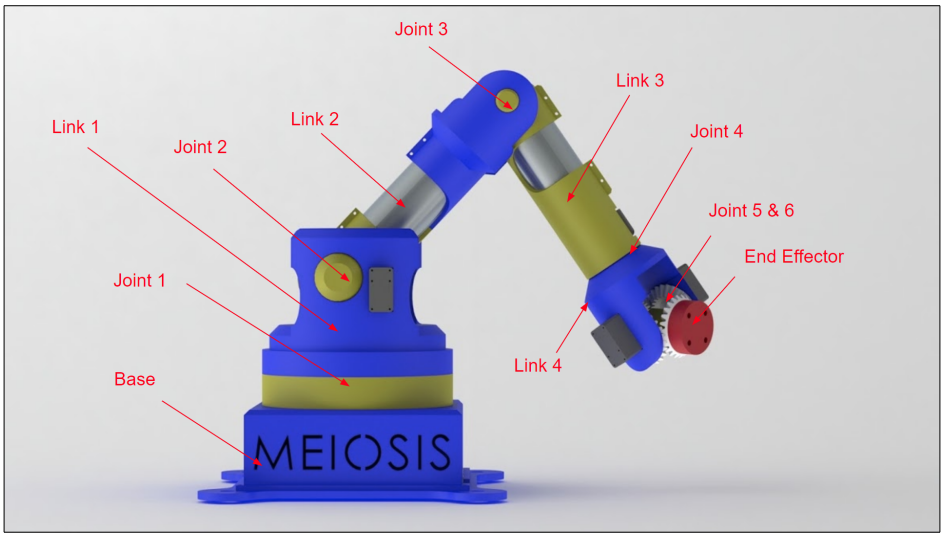


Figure 7: Overview of Physical System

The design seen in *Figure 7* is based on our conceptual design. It features four links and six joints for rotation and will be referenced throughout this document. The base of the manipulator and end-effector can also be seen in the figure.

4.1 Design Requirements

The specifications of the system are strictly based on the requirements defined previously. The requirements are divided into two primary categories, hardware and software.

4.2 Hardware

The following requirements and specifications are hardware specific and dictate the physical constaints the system must adhere to.

4.2.1 The system shall cost the end-user no more than $1000.

4.2.1.a The cost for the MEIOSIS team to develop the manipulator shall cost no more than $800.

4.2.2 The system shall be fully dexterous without being kinematically redundant.

4.2.2.a The system shall consist of six rotational joints connected by four links. The last three joints will create a spherical wrist.

As defined [6], "A manipulator having more than six DOF is referred to as a kinematically redundant manipulator (5)." A manipulator with less than six degrees of freedom will not be fully dexterous within it's workspace. *Figure 9* (see subsection 4.2.6, p. 14) shows a six degree-of-freedom rotary manipulator with it's coordinate frames in zeroed positions. The joint and link locations are seen in *Figure 7* (see section 4, p. 11).

4.2.2.b The system shall have no link offsets.

Link offsets as seen in *Figure 8* are commonly used to avoid singularities. However, having a link offset prevents the manipulator's dexterous workspace from being a complete hemispherical shell.

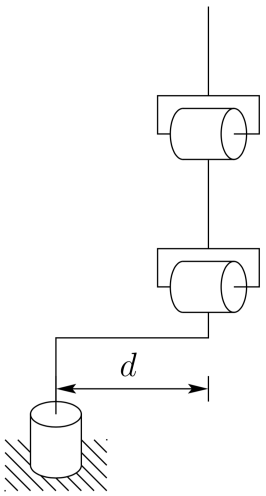


Figure 8: Elbow Manipulator Configuration with Link Offset [6]

As shown in *Figure 8*, the line directly above the first joint of the manipulator is offset such that the axes of the other joints are unable to become collinear with the base axis; this prevents singularity but causes a void in the detxterous workspace.

4.2.3 The system end effector shall maintain a positional accuracy magnitude of ± 1 mm and an orientation accuracy of $\pm 5^{\circ}$ eigen angle from the base frame.

To ensure that the robot has educational value, the accuracy must be defined so that any desired positions and movements are achieved with sufficient accuracy.

4.2.3.a The system shall accommodate a process in which the end user can calibrate the

end effector position and orientation to within 0.5 mm and 1 degree of the manipulator's precision.

The addition of a calibration process allows the removal of any systematic errors, such as drift. The theoretical limit of the calibration process is the difference between the precision and accuracy metrics of the system.

4.2.4 The system end effector shall maintain a pose repeatability magnitude between 0.1—1.5 mm for the position and $\pm 4^\circ$ eigen angle from the base frame for the orientation.

4.2.4.a Joint one and two of the system shall possess an angle error of no more than .025 degrees.

Being that joint one and two are the first two rotational elements in the system, their error will propagate the most to the end effector's position.

4.2.4.b Joint three of the system shall possess an angle error of no more than .03 degrees.

Since joint three is closer to the end effector it's error will not propagate as severely throughout the system.

4.2.4.c Joints four, five, and six shall possess an angle error of no more than .29 degrees.

The spherical wrist is the closest to the end effector's final position and therefore has the least error propagation.

4.2.5 The system's reachable workspace shall be a hemisphere with a radius of 300-700 mm.

This workspace will provide enough movement to manipulate objects in order to perform basic tasks.

4.2.5.a The length of link one, two, three, four, and the wrist shall be 220.8 mm, 250 mm, 200 mm, 80 mm, and 52.5 mm respectively.

This results in a total height of 220.8 mm with a total reach of 582.5 mm in the zeroed configuration as shown in the configuration represented in *Figure 9*.

4.2.6 The system's dexterous workspace shall contain a hemispherical shell within the reachable workspace with a thickness of 280 mm.

This workspace will provide enough movement to manipulate objects in order to perform basic tasks. 280mm is slightly greater than the length of letter paper.

4.2.6.a The rotational limit of joint one, two, three, four, five, and six shall be $\pm 180^\circ$, -9.7°

to 177.5°, −150.6° *to* −19.3°, ±180°, −180° *to* −1.6°, *and* ±180° *respectively.*

The angles stated are with respect to the kinematic model shown in *Figure 9.* To be fully dexterous within our 280 mm dexterous workspace the manipulator must have the joint angles specified above. The joint limitations were calculated by iteratively verifying the orientation about every point within the quarter hemisphere cross section seen in *Figure 26* (see Appendix, p. 44).

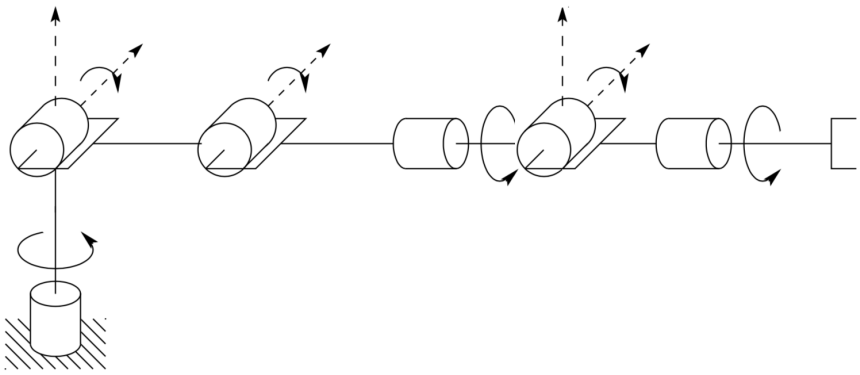


Figure 9: Kinematic Model Representing Zeroed Configuration [6]

4.2.7 The system shall have a removable end effector capable of picking and placing a low-odor chisel tip Expo dry erase marker.

This creates a robot capable of performing a variety of basic tasks, which enhances its educational value.

4.2.7.a The system shall use a parallel gripper that can close to 18mm.

The diameter of a low-odor chisel tip Expo dry erase marker is approximately 18 mm.

4.2.7.b The end effector shall attach to the manipulator using screws configured in a pattern that can accommodate a Dynamixel AX-12A servo.

It is expected that a majority of end effector styles will have to accommodate for a servo to facilitate actuation, therefore a pattern was chosen to standardize the mounting.

4.2.8 The system shall be able to write with a low-odor chisel tip Expo dry erase marker.

4.2.8.a The end effector shall be able to support 0.004 Newton meter moments about the axes normal to its gripping surfaces.

The coefficient of friction between the Expo marker and paper can be approximated and given the weight of an Expo marker the approximate grip strength of the end effector can

be calculated.

4.3 Software

The following requirements and specifications are software specific and determine the attributes of the operating system.

4.3.1 The system shall be open source.

This will create an easily obtainable, low cost method of distributing the system's source code, which may be modified for personal use.

4.3.1.a The software shall be hosted publicly on an online repository and maintain an MIT license for distribution.

This allows the end-user to freely download and modify the code without licensing. The MIT license disregards any legal obligation to code upkeep and documentation by the original author.

4.3.2 The system shall be capable of operating given only desired end effector cartesian coordinates specified with respect to the base frame.

4.3.2.a The system shall have a user interface capable of accepting the end-effector's desired cartesian position and Euler angle orientation as a six element row vector.

The system software interface facilitates an untrained user to operate without the advanced knoledge of the system's kinematics.

4.3.2.b The system shall be capable of performing floating point arithmetic.

The solution for the inverse kinematics requires the ability to perform high level arithmetic with little error.

5 Preliminary Design

The preliminary design section represents the most up-to-date status of the project and also encompasses information pertaining to several different specific design choices that had to be considered. The computer aided design section contains information regarding the physical design of the manipulator as well as noting key design features; the actuator analysis section contains information regarding actuator considerations and calculations since the servo motors play an essential role in the functionality of the final system.

5.1 Computer Aided Design

The manipulator has six degrees of freedom and features a spherical wrist. This design incorporates two different differential drive gearboxes that each control two degrees of freedom. This is to reduce the amount of torque required from the motors on the shoulder joint and reduce the size of the wrist. In order to achieve accuracy requirements, timing belts are incorporated into the design in order to control the first three degrees of freedom. The following figure shows the joint configuration used in the design of the manipulator.

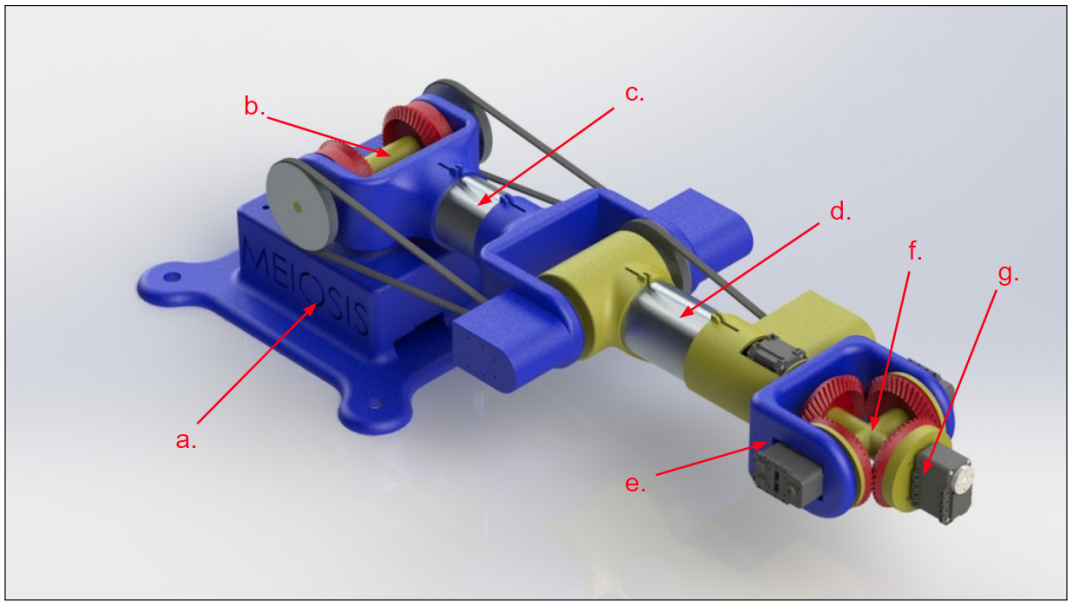


Figure 10: Manipulator in Zeroed Configuration with Callouts

Figure 10 shows the zeroed configuration for the manipulator with callouts showing the link locations.

a. The Manipulator Base is designed to hold the Raspberry Pi and power supply. The base features holes for airflow and I/O connections.

b. Link 1 & Differential Drive have three bevel gears to create the first differential drive rotational joint of the manipulator.

c. Link 2 has servos located at the end of the link to control the movement for the first two

degrees of freedom at the shoulder and are connected to link 1 using timing belts.

d. Link 3 contains two servos attached to the link that control the third degree of freedom (the elbow) and the fourth degree of freedom, which is the first degree of freedom in the wrist.

e. Link 4 houses the servos used to control the fifth and sixth degree of freedom, which are both in the wrist's differential drive gearbox.

f. Link 5 provides support for the wrist's differential drive gearbox.

g. The End Effector features connection points to hold a Dynamixel AX-12A smart servo.

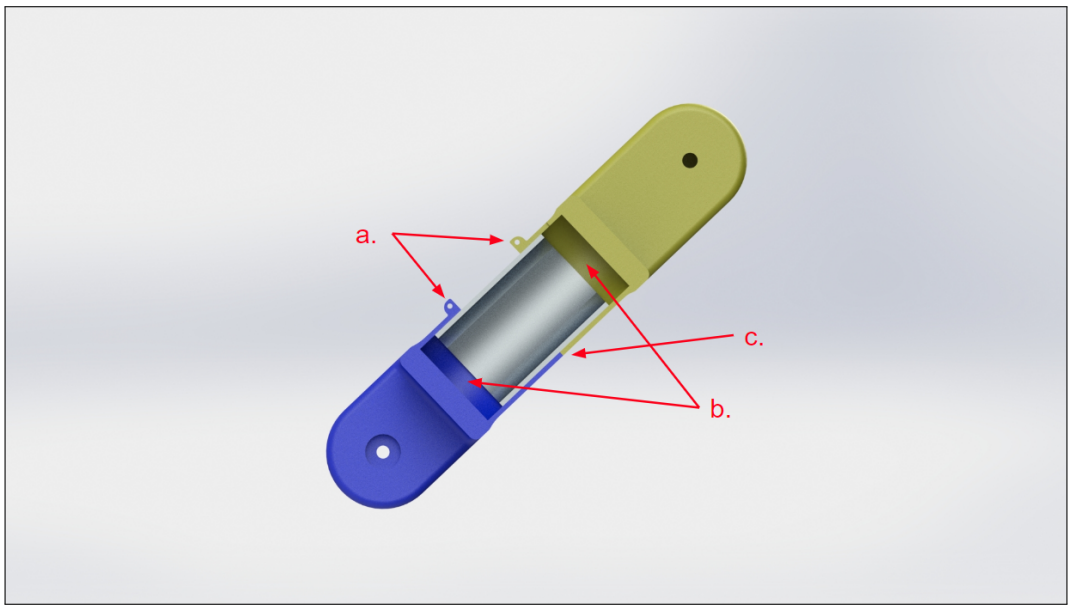


Figure 11: Link Cross Section

Figure 11 shows the cross-section for link two with callouts for key features in the design.

a. Clamping mechanisms connect the aluminum pipe to the 3-D printed parts. This allows for an easy way to assemble the manipulator while allowing for error in the length of the pipe when manufacturing the part.

b. Shows an example of the gaps between the aluminum pipe and the 3-D printed parts.

c. Shows where the 3-D printed parts line up to ensure that the link is the correct length and orientation.

Figure 12 contains an image of the design utilized to keep the assembly of the robot easy and practical.

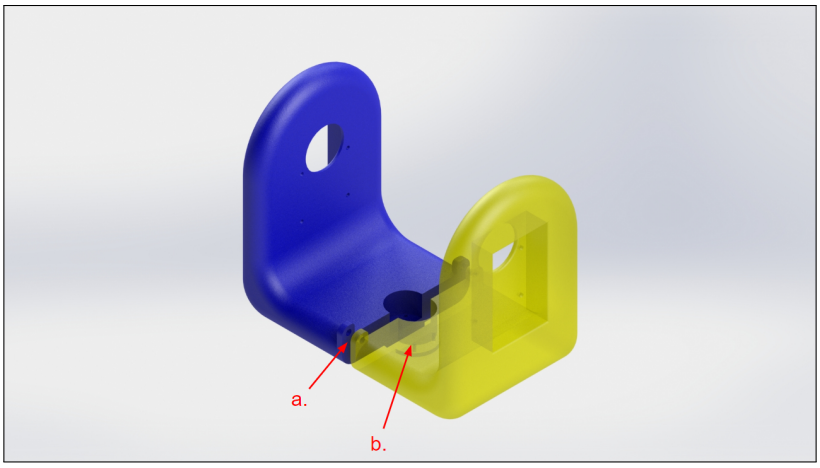


Figure 12: Link 4 Assembly Mechanism

Figure 12 shows the link in the wrist that utilizes a special design to ensure proper assembly. The link itself is two parts which connect together on both sides of the link at point (a), this allows for the differential gearbox to be assembled and inserted into the wrist without interference from link 4. Once assembled, the link attaches to the smart servo at point (b).

5.1.1 Pulleys

A 12 tooth and 120 tooth pulleys provide a 10:1 gear ratio and use a 0.25 in wide MXL belt. The center distance between the pulleys must be at least 5.43 in to have 5 teeth meshing, which is the minimum for reliable operation. Because of the high gear ratio, greater distances do not increase the number of teeth in mesh. The pulley and belt system for link 2 are represented in *Figure 13*.

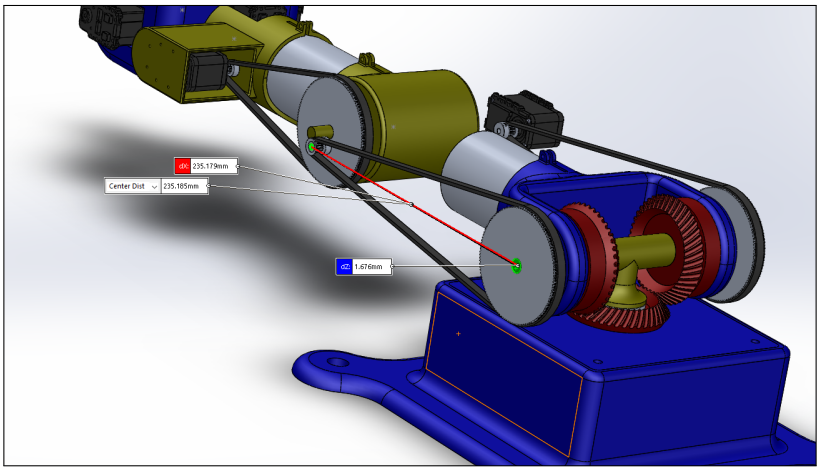


Figure 13: Pulley 1-2 Center Distance

Figure 13 shows the distance between the manipulator's first two pulleys is 235.185 mm, or 9.259 in, which corresponds to a belt with 300 teeth and a pitch diameter of 24 in. The pulley and belt system for link 3 are represented in *Figure 14*.

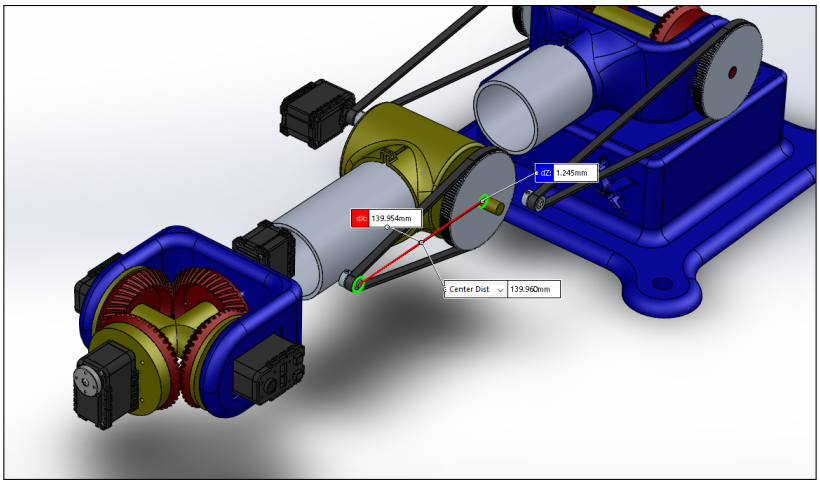


Figure 14: Pulley 3 Center Distance

Figure 14 shows the center distance between the second pulleys is 139.960 mm or 5.510 in. The corresponding belt has 208 teeth with a 16.6 in pitch diameter.

5.2 Actuator Analysis

Dynamixel does not provide a stall torque for the MX-12W servo, however it does provide stall torque data for two similar servos. The MX-12W's stall torque can be found from the stall torque data since the motors are the same. The stall torque of the MX-12W should be comparable to the AX-12W's stall torque. The MX-12W and AX-12W operate at the same 12 V voltage, 32:1 gear ratio, and 470 rpm no load output speed. The AX-12A also operates at 12 V, but with a 254:1 gear ratio and 59 rpm no load output speed. Since torque and current are inversely related, and if the AX-12A's and AX-12W's motors are the same, then when operating at the same voltage, the output speed of the AX-12W should equal to the output speed of the AX-12A multiplied by the ratio of the servos' gear ratios and the torque of the AX-12W should be the AX-12A's torque multiplied by the inverse of the gear ratios ratio:

$$59 \cdot \frac{254}{32} = 468 \cong 470$$

The AX-12A's and MX-12W's stall torques are 1.5Nm and 0.21Nm, respectively [7]:

$$1.5 \cdot \frac{32}{254} = 0.189 \cong 0.21\text{Nm}$$

Therefore, the AX-12A and MX-12W use the same motor. Additionally, if operating at the same voltage, the stall torque is defined by the no load output speed and gear ratio. Since the MX-12W and AX-12A operate at the same 12 V voltage with the same 32:1 gear ratio and 470 rpm no load output speed, their stall torques should be the same. More specifically, it is between the AX-12W's and the gear reduced AX-12A's stall torques of 0.21Nm and 0.189Nm, respectively. Therefore, the stall MX-12W's stall torque is within 0.01Nm of 0.2Nm. The manipulator is designed for the worst case scenario: 0.19 Nm.

The first differential faces its greatest torque when the manipulator is in the zeroed configuration. In the zeroed configuration, the arm is outstretched with gravity acting perpendicular to the arm's length. Since the manipulator must only support a chisel tip dry erase Expo marker, the maximum torque may be approximated as the manipulator's arm's mass acting acting near the centroid. More mass from the servos and longer aluminum support in link 3 place the center of mass closer to the end-effector than the base. Therefore, the point at which the arm's mass acts is assumed to be 273 mm from the base. These calculations will be further revised with mass parameters from the STL files used to generate the open and closed loop simulations. *Table 5* estimates the mass of the manipulator's arms.

Table 5: Manipulator Link 1 Mass Tabulations

Item	**Specifications**	**Weight (g)**	**Notes**
3-D Printed Materials	30% infill	353	100% infill: 1177g
Servos	7 low torque	385	55 $^{g}/_{servo}$
Aluminum Support	.75″ OD .125″ thick 1′ long	287	44.23 $^{g}/_{in^3}$
Bearings	5 bearings	60	12 $^{g}/_{bearing}$
End Effector		50	Approx.
Safety Factor		205	
	Total	1340	

The total mass of 1340g as seen in *Table 5*, yields a moment of 0.361 Nm or 3.685 kgm. To generate smooth motions, Dynamixel recommends operating at one fifth of the stall torque. Therefore, the actual maximum required torque is 1.805 Nm. An MX-12W provides 0.19 Nm of torque, necessitating a 10:1 gear ratio. While the AX-12A provides a greater torque of 1.5 Nm, it would also require gearing, The AX-12A is also only precise to 0.8 degrees, while the specifications require at least 0.29 degrees of precision for joints one and two. Further, the AX-12A only receives position feedback for 300 degrees of its rotation and cannot track multiple rotations, making it untenable for the base motor, which must rotate more than 360 degrees.

The third joint's torque requirements can be approximated in the same manner. However, links three, four, and five's masses are assumed to act 200 mm from the third joint. *Table 6* estimates the links' masses.

The resulting moment is 1.630 Nm, which can be satisfied by the same motor and gear ratio used on the base.

5.3 Forward Kinematics

The forward kinematics of the manipulator are described by the equations below, where the reference coordinate frames are given by *Figure 15*.

Table 6: Manipulator Link 2 Mass Tabulations

Item	**Specifications**	**Weight (g)**	**Notes**
3-D Printed Materials	30% infill	180	100% infill: 1177g
Servos	4 low torque	220	55 $^{g}/_{\text{servo}}$
Aluminum Support	.75″ OD .125″ thick 1′ long	140	44.23 $^{g}/_{\text{in}^3}$
Bearings	5 bearings	36	12 $^{g}/_{\text{bearing}}$
End Effector		50	Approx.
Safety		205	
	Total	831	

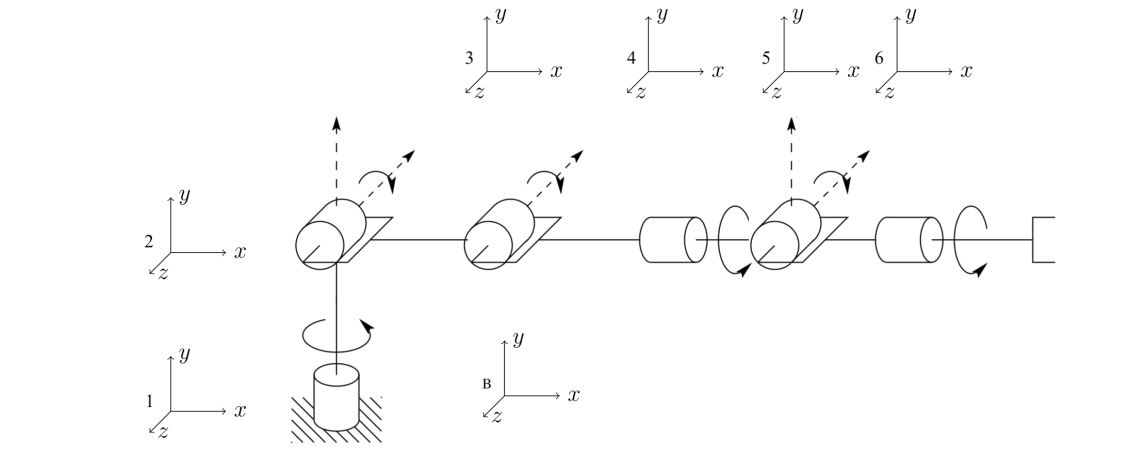


Figure 15: Coordinate Systems

Given the direction cosine matrices,

$$rotx(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}, \quad roty(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}$$

$$rotz(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \end{bmatrix}$$

The orientation of each link with respect to the inertial frame is given as:

$$\begin{aligned}
{}^{I}T_1 &= rotz(\theta_1) \\
{}^{I}T_2 &= rotz(\theta_1)\ rotx(\theta_2) \\
{}^{I}T_3 &= rotz(\theta_1)\ rotx(\theta_2)\ rotx(\theta_3) \\
{}^{I}T_4 &= rotz(\theta_1)\ rotx(\theta_2)\ rotx(\theta_3)\ roty(\theta_4) \\
{}^{I}T_5 &= rotz(\theta_1)\ rotx(\theta_2)\ rotx(\theta_3)\ roty(\theta_4)\ rotx(\theta_5) \\
{}^{I}T_6 &= rotz(\theta_1)\ rotx(\theta_2)\ rotx(\theta_3)\ roty(\theta_4)\ rotx(\theta_5)\ roty(\theta_6)
\end{aligned}$$

$${}^{I}T_1 = \begin{bmatrix} c_{\theta_1} & -s_{\theta_1} & 0 \\ s_{\theta_1} & c_{\theta_1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad {}^{I}T_2 = \begin{bmatrix} c_{\theta_1} & -c_{\theta_2}s_{\theta_1} & s_{\theta_1}s_{\theta_2} \\ s_{\theta_1} & c_{\theta_1}c_{\theta_2} & -c_{\theta_1}s_{\theta_2} \\ 0 & s_{\theta_2} & c_{\theta_2} \end{bmatrix} \quad {}^{I}T_3 = \begin{bmatrix} c_{\theta_1} & -c_{\theta_{23}}s_{\theta_1} & s_{\theta_{23}}s_{\theta_1} \\ s_{\theta_1} & c_{\theta_{23}}c_{\theta_1} & -s_{\theta_{23}}c_{\theta_1} \\ 0 & s_{\theta_{23}} & c_{\theta_{23}} \end{bmatrix}$$

$${}^{I}T_4 = \begin{bmatrix} c_{\theta_1}c_{\theta_4} - s_{\theta_{23}}s_{\theta_1}s_{\theta_4} & -c_{\theta_{23}}s_{\theta_1} & c_{\theta_1}s_{\theta_4} + s_{\theta_{23}}c_{\theta_4}s_{\theta_1} \\ c_{\theta_4}s_{\theta_1} + s_{\theta_{23}}c_{\theta_1}s_{\theta_4} & c_{\theta_{23}}c_{\theta_1} & s_{\theta_1}s_{\theta_4} - s_{\theta_{23}}c_{\theta_1}c_{\theta_4} \\ -c_{\theta_{23}}s_{\theta_4} & s_{\theta_{23}} & c_{\theta_{23}}c_{\theta_4} \end{bmatrix}$$

$${}^{I}T_5 = \begin{bmatrix} c_{\theta_1}c_{\theta_4} - s_{\theta_{23}}s_{\theta_1}s_{\theta_4} & s_{\theta_5}(c_{\theta_1}s_{\theta_4} + s_{\theta_{23}}c_{\theta_4}s_{\theta_1}) - c_{\theta_{23}}c_{\theta_5}s_{\theta_1} & c_{\theta_5}(c_{\theta_1}s_{\theta_4} + s_{\theta_{23}}c_{\theta_4}s_{\theta_1}) + c_{\theta_{23}}s_{\theta_1}s_{\theta_5} \\ c_{\theta_4}s_{\theta_1} + s_{\theta_{23}}c_{\theta_1}s_{\theta_4} & s_{\theta_5}(s_{\theta_1}s_{\theta_4} - s_{\theta_{23}}c_{\theta_1}c_{\theta_4}) + c_{\theta_{23}}c_{\theta_1}c_{\theta_5} & c_{\theta_5}(s_{\theta_1}s_{\theta_4} - s_{\theta_{23}}c_{\theta_1}c_{\theta_4}) - c_{\theta_{23}}c_{\theta_1}s_{\theta_5} \\ -c_{\theta_{23}}s_{\theta_4} & s_{\theta_{23}}c_{\theta_5} + c_{\theta_{23}}c_{\theta_4}s_{\theta_5} & c_{\theta_{23}}c_{\theta_4}c_{\theta_5} - s_{\theta_{23}}s_{\theta_5} \end{bmatrix}$$

$${}^{I}T_6 = \begin{bmatrix} {}^{I}T_{6(1,1)} & {}^{I}T_{6(1,2)} & {}^{I}T_{6(1,3)} \\ {}^{I}T_{6(2,1)} & {}^{I}T_{6(2,2)} & {}^{I}T_{6(2,3)} \\ {}^{I}T_{6(3,1)} & {}^{I}T_{6(3,2)} & {}^{I}T_{6(3,3)} \end{bmatrix}$$

$$\begin{aligned}
{}^{I}T_{6(1,1)} &= c_{\theta_6}(c_{\theta_1}c_{\theta_4} - s_{\theta_{23}}s_{\theta_1}s_{\theta_4}) - s_{\theta_6}(c_{\theta_5}(c_{\theta_1}s_{\theta_4} + s_{\theta_{23}}c_{\theta_4}s_{\theta_1}) + c_{\theta_{23}}s_{\theta_1}s_{\theta_5}) \\
{}^{I}T_{6(1,2)} &= s_{\theta_5}(c_{\theta_1}s_{\theta_4} + s_{\theta_{23}}c_{\theta_4}s_{\theta_1}) - c_{\theta_{23}}c_{\theta_5}s_{\theta_1} \\
{}^{I}T_{6(1,3)} &= c_{\theta_6}(c_{\theta_5}(c_{\theta_1}s_{\theta_4} + s_{\theta_{23}}c_{\theta_4}s_{\theta_1}) + c_{\theta_{23}}s_{\theta_1}s_{\theta_5}) + s_{\theta_6}(c_{\theta_1}c_{\theta_4} - s_{\theta_{23}}s_{\theta_1}s_{\theta_4}) \\
{}^{I}T_{6(2,1)} &= c_{\theta_6}(c_{\theta_4}s_{\theta_1} + s_{\theta_{23}}c_{\theta_1}s_{\theta_4}) - s_{\theta_6}(c_{\theta_5}(s_{\theta_1}s_{\theta_4} - s_{\theta_{23}}c_{\theta_1}c_{\theta_4}) - c_{\theta_{23}}c_{\theta_1}s_{\theta_5}) \\
{}^{I}T_{6(2,2)} &= s_{\theta_5}(s_{\theta_1}s_{\theta_4} - s_{\theta_{23}}c_{\theta_1}c_{\theta_4}) + c_{\theta_{23}}c_{\theta_1}c_{\theta_5} \\
{}^{I}T_{6(2,3)} &= s_{\theta_6}(c_{\theta_4}s_{\theta_1} + s_{\theta_{23}}c_{\theta_1}s_{\theta_4}) + c_{\theta_6}(c_{\theta_5}(s_{\theta_1}s_{\theta_4} - s_{\theta_{23}}c_{\theta_1}c_{\theta_4}) - c_{\theta_{23}}c_{\theta_1}s_{\theta_5}) \\
{}^{I}T_{6(3,1)} &= s_{\theta_6}(s_{\theta_{23}}s_{\theta_5} - c_{\theta_{23}}c_{\theta_4}c_{\theta_5}) - c_{\theta_{23}}c_{\theta_6}s_{\theta_4} \\
{}^{I}T_{6(3,2)} &= s_{\theta_{23}}c_{\theta_5} + c_{\theta_{23}}c_{\theta_4}s_{\theta_5} \\
{}^{I}T_{6(3,3)} &= c_{\theta_6}(s_{\theta_{23}}s_{\theta_5} - c_{\theta_{23}}c_{\theta_4}c_{\theta_5}) - c_{\theta_{23}}s_{\theta_4}s_{\theta_6}
\end{aligned}$$

Given the lengths of each of the manipulator links,

$$ {}^{I}_{B}r_1 = \begin{bmatrix} 0 \\ 0 \\ \ell_b \end{bmatrix} \quad {}^{1}_{1}r_2 = \begin{bmatrix} 0 \\ 0 \\ \ell_1 \end{bmatrix} \quad {}^{2}_{2}r_3 = \begin{bmatrix} 0 \\ \ell_2 \\ 0 \end{bmatrix} \quad {}^{3}_{3}r_4 = \begin{bmatrix} 0 \\ \ell_3 \\ 0 \end{bmatrix} \quad {}^{4}_{4}r_5 = \begin{bmatrix} 0 \\ \ell_4 \\ 0 \end{bmatrix} \quad {}^{5}_{5}r_6 = \begin{bmatrix} 0 \\ \ell_5 \\ 0 \end{bmatrix} $$

The position of each link relative to the inertial frame is given as:

$$ {}^{I}_{B}r_1 = {}_{B}r_1 \qquad {}^{I}_{B}r_2 = r_1 + {}^{I}T_1 {}^{1}_{1}r_2 \qquad {}^{I}_{B}r_3 = r_2 + {}^{I}T_2 {}^{2}_{2}r_3 $$
$$ {}^{I}_{B}r_4 = r_3 + {}^{I}T_3 {}^{3}_{3}r_4 \qquad {}^{I}_{B}r_5 = r_4 + {}^{I}T_4 {}^{4}_{4}r_5 \qquad {}^{I}_{B}r_6 = r_5 + {}^{I}T_5 {}^{5}_{5}r_6 $$

$$ {}^{I}_{B}r_1 = \begin{bmatrix} 0 \\ 0 \\ \ell_b \end{bmatrix} \quad {}^{I}_{B}r_2 = \begin{bmatrix} 0 \\ 0 \\ \ell_b + \ell_1 \end{bmatrix} \quad {}^{I}_{B}r_3 = \begin{bmatrix} -\ell_2 c_{\theta_2} s_{\theta_1} \\ \ell_2 c_{\theta_{12}} \\ \ell_b + \ell_1 + \ell_2 s_{\theta_2} \end{bmatrix} \quad {}^{I}_{B}r_4 = \begin{bmatrix} -s_{\theta_1}(\ell_3 c_{\theta_{23}} + \ell_2 c_{\theta_2}) \\ c_{\theta_1}(\ell_3 c_{\theta_{23}} + \ell_2 c_{\theta_2}) \\ \ell_1 + \ell_b + \ell_3 s_{\theta_{23}} + \ell_2 s_{\theta_2} \end{bmatrix} $$

$$ {}^{I}_{B}r_5 = \begin{bmatrix} -s_{\theta_1}(\ell_3 c_{\theta_{23}} + \ell_4 c_{\theta_{23}} + \ell_2 c_{\theta_2}) \\ c_{\theta_1}(\ell_3 c_{\theta_{23}} + \ell_4 c_{\theta_{23}} + \ell_2 c_{\theta_2}) \\ \ell_1 + \ell_b + \ell_3 s_{\theta_{23}} + \ell_4 s_{\theta_{23}} + \ell_2 s_{\theta_2} \end{bmatrix} $$

$$ {}^{I}_{B}r_6 = \begin{bmatrix} \ell_5 c_{\theta_1} s_{\theta_4} s_{\theta_5} - \ell_4 c_{\theta_{23}} s_{\theta_1} - \ell_2 c_{\theta_2} s_{\theta_1} - \ell_5 c_{\theta_{23}} c_{\theta_5} s_{\theta_1} - \ell_3 c_{\theta_{23}} s_{\theta_1} \\ \qquad + \ell_5 c_{\theta_2} c_{\theta_4} s_{\theta_1} s_{\theta_3} s_{\theta_5} + \ell_5 c_{\theta_3} c_{\theta_4} s_{\theta_1} s_{\theta_2} s_{\theta_5} \\ \ell_5(s_{\theta_5}(s_{\theta_1} s_{\theta_4} - c_{\theta_4}(c_{\theta_1} c_{\theta_2} s_{\theta_3} + c_{\theta_1} c_{\theta_3} s_{\theta_2})) - c_{\theta_5}(c_{\theta_1} s_{\theta_2} s_{\theta_3} - c_{\theta_1} c_{\theta_2} c_{\theta_3})) \\ \qquad - \ell_3(c_{\theta_1} s_{\theta_2} s_{\theta_3} - c_{\theta_1} c_{\theta_2} c_{\theta_3}) - \ell_4(c_{\theta_1} s_{\theta_2} s_{\theta_3} - c_{\theta_1} c_{\theta_2} c_{\theta_3}) + \ell_2 c_{\theta_1} c_{\theta_2} \\ \ell_1 + \ell_b + \ell_3 s_{\theta_{23}} + \ell_4 s_{\theta_{23}} + \ell_2 s_{\theta_2} + \dfrac{\ell_5 c_{\theta_{23}} s_{\theta_{45}}}{2} + \ell_5 s_{\theta_{23}} c_{\theta_5} - \dfrac{\ell_5 s_{\theta_4 - \theta_5} c_{\theta_{23}}}{2} \end{bmatrix} $$

5.4 Velocity Kinematics

The translational and rotational velocties ($\dot{r}$ & ω) can be found given the geometric jacobian of the body and transformation matrix corresponding to it.

$$ \begin{bmatrix} {}^{B}_{B}\omega_I \\ {}^{I}_{B}\dot{r}_B \end{bmatrix} = J_B = \begin{bmatrix} {}^{I}T_B(:,3)^T \cdot \dfrac{\partial}{\partial \gamma} {}^{I}T_B(:,2) \\ {}^{I}T_B(:,1)^T \cdot \dfrac{\partial}{\partial \gamma} {}^{I}T_B(:,3) \\ {}^{I}T_B(:,2)^T \cdot \dfrac{\partial}{\partial \gamma} {}^{I}T_B(:,1) \end{bmatrix} $$

5.5 Inverse Kinematics

The inverse kinematics can be calculated given desired position and orientation vectors, o and R respectively.

$$ \begin{bmatrix} x_c & y_c & z_c \end{bmatrix} = o \ , \quad R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} $$

Inverse Position:

$$\theta_1 = \text{atan2}\left(x_c,\ y_c\right) - \pi/2$$
$$\theta_2 = \text{atan2}\left(z_c - \ell_1,\ \sqrt{x_c^2 + y_c^2}\right) - \text{atan2}\left(\ell_3 s_3,\ \ell_2 + \ell_3 c_3\right)$$
$$\theta_3 = \text{atan2}(-\sqrt{1 - D^2},\ D)$$
$$\text{where } D \equiv \frac{x_c^2 + y_c^2 + (z_c - \ell_1)^2 - \ell_2^2 - \ell_3^2}{2\ell_2\ell_3}$$

Inverse Orientation :

$${}^I T_3 = rotz(\theta_1)\ rotx(\theta_2)\ rotx(\theta_3)$$
$${}^3 T_6 = {}^I T_3^T R$$
$$\theta_4 = \text{atan2}\left({}^3T_{6(1,2)},\ {}^3T_{6(3,2)}\right)$$
$$\theta_5 = \text{atan2}\left({}^3T_{6(3,2)}/c_4,\ {}^3T_{6(2,2)}\right)$$
$$\theta_6 = \text{atan2}\left({}^3T_{6(2,1)},\ \ -{}^3T_{6(2,3)}\right)$$

5.6 Equations of Motion

Given robot dynamics described by $H(\gamma)\ddot{\gamma} + d(\gamma, \dot{\gamma}) + G(\gamma) = F_\gamma$, the equations of motion for the manipulator can be determined. Solving this equation for the acceleration, $\ddot{\gamma}$, gives:

$$\ddot{\gamma} = H(\gamma)^{-1}\left(F_\gamma - d(\gamma, \dot{\gamma}) - G(\gamma)\right) \tag{1}$$

Where H is the system mass matrix, F_γ is the vector of generalized forces, d is the vector of centripital and coriolis effects, and G is gravitational effects.

$$H(\gamma) = \sum_B^N J_B(\gamma)^T \begin{bmatrix} {}^B_B J & \mathring{S}({}^B_B\Gamma)\,{}^I T_B^T \\ {}^I T_B \mathring{S}({}^B_B\Gamma)^T & m_B I \end{bmatrix} J_B(\gamma)\ , \quad {}^B_B\Gamma = {}^B_B\, r_{cm} m_b\ , \quad \mathring{S}(\omega) r = (\omega \times r)$$

$$d(\gamma, \dot{\gamma}) = \sum_B^N J_B(\gamma)^T \begin{bmatrix} {}^B_B J & \mathring{S}({}^B_B\Gamma)\,{}^I T_B^T \\ {}^I T_B \mathring{S}({}^B_B\Gamma)^T & m_B I \end{bmatrix} \dot{J}_B(\gamma, \dot{\gamma})\dot{\gamma} + J_B(\gamma)^T \begin{bmatrix} {}^B_B\omega_I \times {}^B_B\, J^B_B \omega_I \\ {}^I T_B \left({}^B_B\omega_I \times ({}^B_B\omega_I \times {}^B_B\,\Gamma)\right) \end{bmatrix}$$

$$G(\gamma) = \left(\frac{\partial U({}^I r(\gamma))}{\partial \gamma}\right)^T\ , \quad U_B = \begin{bmatrix} 0 & 0 & g \end{bmatrix}\left({}^I_B r_B m_B + {}^I\, T_B\, {}^B_B\Gamma\right)$$

Where J_B is the jacobian of the body, Γ is the vector of first mass moments, m_B is the mass of the body, and ${}^B_B\omega_I$ is the rotational velocity of the body relative to the inertial frame.

5.6.1 Actuator Dynamics

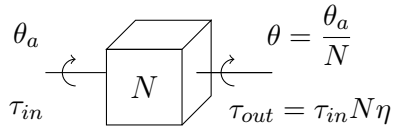
Given robot dynamics described by $H(\gamma)\ddot{\gamma} + n(\gamma, \dot{\gamma}) = \tau$, the torque, τ, provided by the servo motors is necessary to solve the closed loop dynamics of the system. Assuming the servo is driven by a D.C. motor with proportional derivative control,

$$\tau_a = K i_a = J_a \ddot{\theta}_a + b_a \dot{\theta}_a + \tau_L \tag{2}$$

Where τ_a is the actuator torque, K is the back-EMF constant, i_a is the motor current, J_a is the armature inertia, $\theta_a,\ \dot{\theta}_a, \ddot{\theta}_a$ is the motor position and it's first and second time derivatives respectively, b_a is the viscous friction coefficient, and τ_L is the torque available for the actuator to do work. The basic equation for a motor is known to be:

$$V_a = i_a R_a + K\dot{\theta}_a \tag{3}$$

Where V_a is the voltage applied to the actuator and R_a is the armature resistance. Given a gearbox with $^{\text{in}}/_{\text{out}}$ ratio N and efficiency η,



The motor equation (2) can be expressed in the output coordinates:

$$Ki_a = J_a N\ddot{\theta} + b_a N\dot{\theta} + \frac{\tau}{N\eta}$$

Substituing into equation (3) and solving for i_a:

$$i_a = \frac{J_a N}{K}\ddot{\theta} + b_a N\dot{\theta} + \frac{\tau}{N\eta}$$

$$V_a = \frac{R_a J_a N}{K}\ddot{\theta} + \frac{R_a b_a N}{K}\dot{\theta} + \frac{R_a}{KN\eta}\tau + KN\dot{\theta} \tag{4}$$

Assuming PD control, $V_a = K_p(\theta - \theta_d) + K_d\dot{\theta}$, where θ_d is the desired orientation of the actuator, the following solution is found by setting the PD solution equal to equation (4). After collecting like terms:

$$\frac{R_a J_a N}{K}\ddot{\theta} + \left(\frac{R_a J_a N}{K} - K_d + KN\right)\dot{\theta} - K_p\theta = -K_p\theta_d - \frac{R_a}{KN\eta}\tau \tag{5}$$

The following parameters of the system can be obtained by applying a step input to the system with $\tau = 0$ and measuring the characteristics of it's response. Denoting ζ as the damping ratio and ω_n as the natural frequency of the system,

$$\%\ \text{Overshoot} = \left(\frac{\theta_{max} - \theta_{ss}}{\theta_{ss}}\right) \times 100\ , \quad \zeta = \frac{-\ln(^{\%\text{OS}}/_{100})}{\sqrt{\pi^2 + \ln^2(^{\%\text{OS}}/_{100})}}\ , \quad \omega_n = \frac{\pi}{T_p\sqrt{1-\zeta^2}}$$

Given $\theta_{max},\ \theta_{ss}$, and T_p as measured parameters of the system's max output, steady state, and time to peak, respectively.

Refactoring equation (5) and equating with the general solution for a second order system given by $\ddot{\theta} + 2\zeta\omega_n\dot{\theta} + \omega_n^2\theta = \omega_n^2\theta_d$, the following solutions are found:

$$2\zeta\omega_n = \frac{b_a}{J_a} - \frac{KK_d}{R_aJ_aN} + \frac{K^2}{R_aJ_a} \qquad (6) \qquad\qquad \omega_n^2 = \frac{-KK_p}{R_aJ_aN} \qquad (7)$$

Performing a similar experiment as previously described, except with a known inertial load $\tau = J_m\ddot{\theta}$, the following parameters can be found:

$$\alpha_m \equiv 2\zeta\omega_n = \frac{R_ab_aN^2\eta - KK_dN\eta + K^2N^2\eta}{R_aJ_aN^2\eta + R_aJ_m}\ , \quad \beta_m \equiv \omega_n = -\frac{KK_pN\eta}{R_aJ_aN^2\eta + R_aJ_m}$$

$$\begin{bmatrix} 1 & -(\alpha_1J_1 + \beta_1J_1) \\ 1 & -(\alpha_2J_2 + \beta_2J_2) \\ \vdots & \vdots \end{bmatrix} \begin{bmatrix} \dfrac{R_ab_aN^2\eta - KK_dN\eta + K^2N^2\eta - KK_pN\eta}{R_aJ_aN^2\eta} \\ \\ \dfrac{1}{J_aN^2\eta} \end{bmatrix} = \begin{bmatrix} \alpha_1 + \beta_1 \\ \alpha_2 + \beta_2 \\ \vdots \end{bmatrix} \qquad (8)$$

With multiple datasets (varying inertial loads, J_m), the solutions of equation (8) can be found using the least-squares method, yeilding

$$\frac{R_ab_aN - KK_d + K^2N\eta - KK_p}{R_aJ_aN} \qquad (9) \qquad\qquad \frac{1}{J_aN^2\eta} \qquad (10)$$

Finally, the coefficients of the second order system (11) are known:

$$\underbrace{\left(J_aN^2\eta\right)}_{{}^{1}\!/_{(10)}\ = C_1}\ddot{\theta} + \underbrace{\left(\frac{R_ab_aN^2\eta - KK_dN\eta + K^2N^2\eta}{R_a}\right)}_{{}^{(6)}\!/_{(10)}\ = C_2}\dot{\theta} - \underbrace{\left(\frac{KK_pN\eta}{R_a}\right)}_{{}^{(7)}\!/_{(10)}\ = C_3}\theta + \underbrace{\left(\frac{KK_pN\eta}{R_a}\right)}_{{}^{(7)}\!/_{(10)}\ = C_3}\theta_d = -\tau \qquad (11)$$

The MATLAB code implementing this process can be found in the Appendix (see section 5.13, p. 44, *Listing 1*). The torque provided by the servo can now be solved for, given the current position (θ), velocity ($\dot{\theta}$), angular acceleration ($\ddot{\theta}$), and desired position (θ_d) are known.

Given the equation of motion for the dynamical response of the system (1), substituting in the solution obtained for the motor dynamics and solving for the acceleration,

$$\left(H + J_aN^2\eta\right)^{-1}\left[\left(B - \frac{R_ab_aN^2\eta - KK_dN\eta + K^2N^2\eta}{R_a}\right)\dot{\gamma} - \left(\frac{KK_pN\eta}{R_a}\right)(\gamma_d - \gamma) - n\right] = \ddot{\gamma}$$

Where

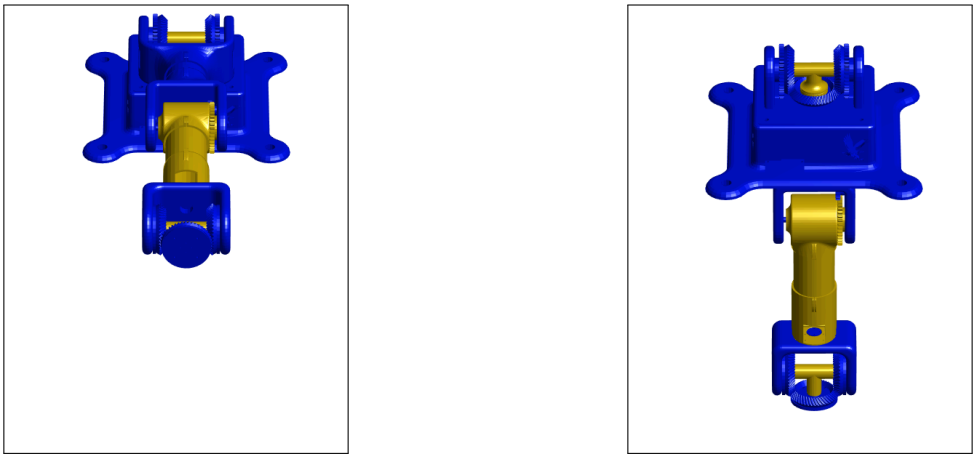
$$n(\gamma, \dot{\gamma}) = d(\gamma, \dot{\gamma}) + G(\gamma) + C\mathrm{sgn}(\dot{\gamma})$$

5.7 Open-Loop Simulation

The equations of motion described in equation (1) can be integrated to simulate the motion of the system. For the open loop control, no input torque is supplied, meaning the system responds only to gravity. Due to the complexity of the equations of motion, they will be integrated numerically using a 4th Order Runge-Kutta method algorithm [5]. Since no input force is supplied, the equations of motion reduce down to the relation described in equation (12).

$$\ddot{\gamma} = H(\gamma)^{-1}(-d(\gamma, \dot{\gamma}) - G(\gamma)) \tag{12}$$

The resulting simulation shows the manipulator assembly starting in the zeroed configuration, (*Figure 16a*) then "falling" due to the forces of gravity acting on the links (*Figure 16b*). The manipulator continues to oscillate analogous to a simple pendulum, being that there are no frictional forces accounted for.



(a) Frame Snapshot near Simulation Initiation

(b) Frame Snapshot near Simulation Termination

Figure 16: Open-Loop Control Simulation Animation Snapshots

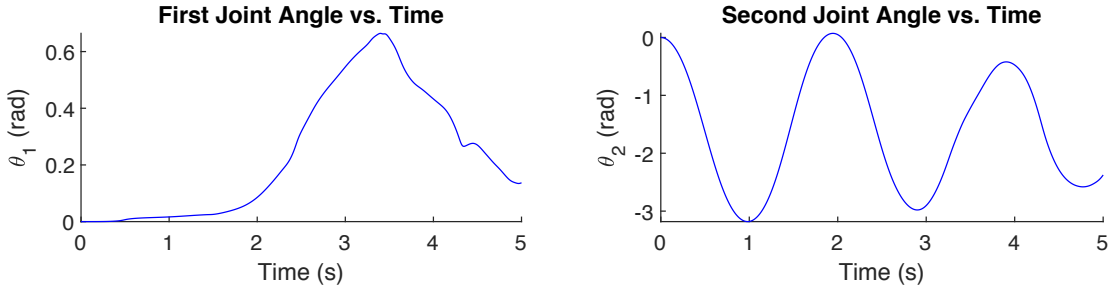


Figure 17: Joint Angles vs Time in Open-Loop Simulation

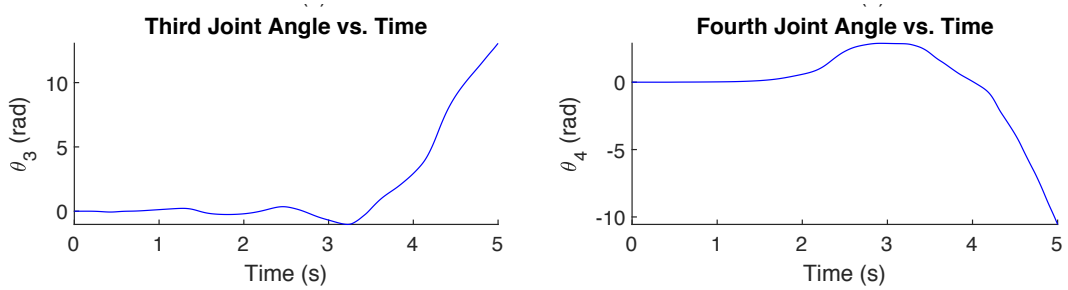


Figure 17 (cont.): Joint Angles vs Time in Open-Loop Simulation

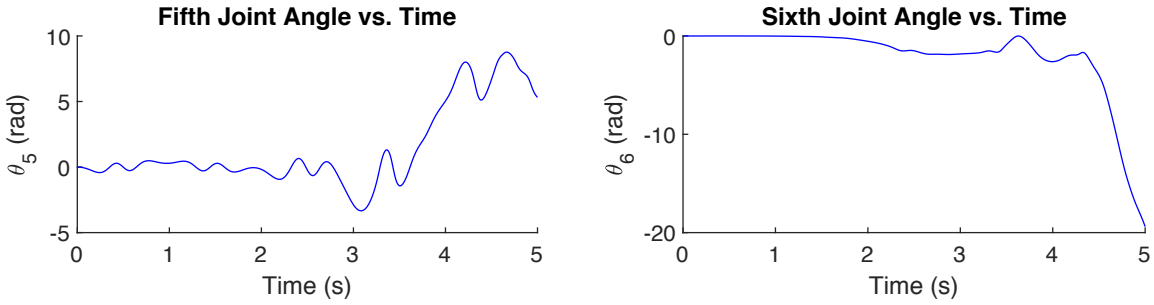


Figure 17 (cont.): Joint Angles vs Time in Open-Loop Simulation

5.8 Closed-Loop Simulation

The motor equation (11) gives an expression for the motor torques, however the system dynamics are defined in terms of geometric joint angles. The inclusion of differential drive systems means that the joint angles, γ do not directly correspond to motor rotations, θ. However, there is a linear relation between them, described in equation (13).

$$\gamma = A\theta \quad \text{where} \quad A = \begin{bmatrix} 1/(2N) & 1/(2N) & 0 & 0 & 0 & 0 \\ 1/(2N) & -1/(2N) & 0 & 0 & 0 & 0 \\ 0 & 0 & -1/N & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1/2 & 1/2 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \end{bmatrix} \tag{13}$$

Equation (13) can be used to map the joint angles to the motor angles. The gear ratio of 1:10 is represented by the variable N. Similarly, the motor angles can be determined by multiplying both sides of equation (13) by the inverse of matrix A, giving the following relation.

$$\theta = A^{-1}\gamma \tag{14}$$

It is important to note that the virtual work done by the joint torques (F_γ) and the virtual work done by the motor torques (F_θ) are equal. Using equation (14), a linear relation between

the joint torques and motor torques can be determined.

$$\begin{aligned}
\delta W = F_\theta^T \delta\theta = F_\gamma^T \delta\gamma, \text{ where } \delta\gamma = A\delta\theta \\
F_\theta^T \delta\theta = F_\gamma^T (A\delta\theta) \\
F_\theta^T = F_\gamma^T A \\
\left(F_\theta^T\right)^T = \left(F_\gamma^T A\right)^T \\
F_\theta = A^T F_\gamma \Leftrightarrow F_\gamma = A^{-T} F_\theta
\end{aligned}$$

Using this equation, a relation can be determined between the motor dynamics and the system dynamics given in equation (11) and equation (1) respectively.

$$\begin{aligned}
H(\gamma)\ddot{\gamma} + d(\gamma, \dot{\gamma}) + G(\gamma) = -A^{-T}\left(C_1 A^{-1}\ddot{\gamma} + C_2 A^{-1}\dot{\gamma} + C_3\theta_d - C_3 A^{-1}\gamma\right) \\
\ddot{\gamma} = H(\gamma)^{-1}\left(-A^{-T}\left(C_1 A^{-1}\ddot{\gamma} + C_2 A^{-1}\dot{\gamma} + C_3\theta_d - C_3 A^{-1}\gamma\right) - d(\gamma, \dot{\gamma}) - G(\gamma)\right) \qquad (15)
\end{aligned}$$

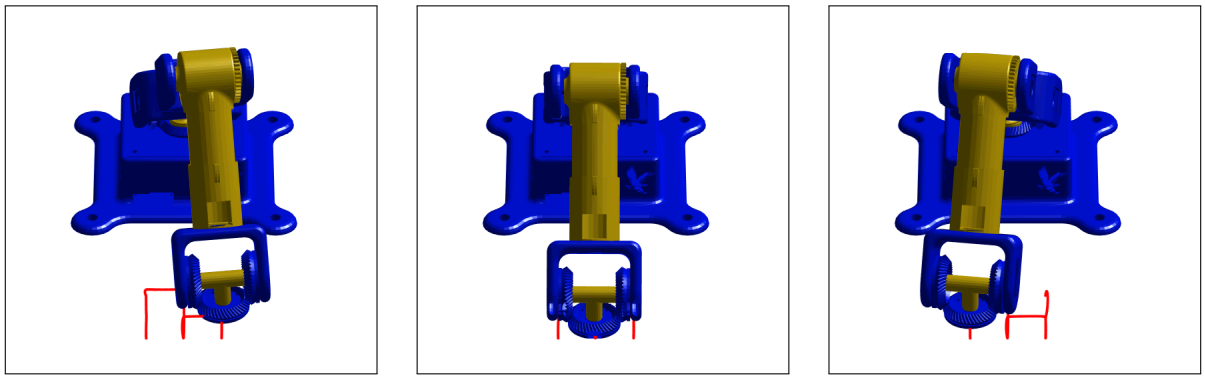
Because this equation includes the motor model, which in turn includes an internal PD controller, this equation can be integrated to solve for the system response given a desired motor angle input, θ_d. However, doing so will not result in the desired system response. This control scheme does not have any compensation for the inertia of the links, and it is also lacking gravity compensation. This can be remedied by modifying the input to the motors, θ_d. A new input, u, is defined such that gravity can be compensated. Thus, the motor input term in equation (15) must include both compensation for gravity and the desired motor angle.

$$\begin{aligned}
A^{-T} C_3 u = G(\gamma) + d(\gamma, \dot{\gamma}) + A^{-T} C_3 \theta_d \\
u = \left(A^{-T} C_3\right)^{-1}\left(G(\gamma) + d(\gamma, \dot{\gamma})\right) + \theta_d \qquad (16)
\end{aligned}$$

With this new motor input, the closed loop control system equations of motion are given as:

$$\ddot{\gamma} = H(\gamma)^{-1}\left(-A^{-T}\left(C_1 A^{-1}\ddot{\gamma} + C_2 A^{-1}\dot{\gamma} + C_3 u - C_3 A^{-1}\gamma\right) - d(\gamma, \dot{\gamma}) - G(\gamma)\right) \qquad (17)$$

Equation (17) can then be integrated to solve for the system response given desired motor angles.



(a) Frame Snapshot near Simulation Initiation

(b) Frame Snapshot near Simulation Middle

(c) Frame Snapshot near Simulation Termination

Figure 18: Closed-Loop Control Simulation Animation Snapshots

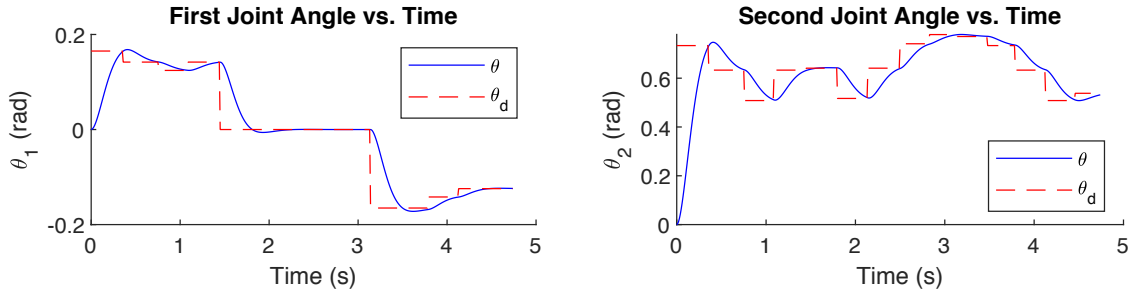


Figure 19: Joint Angles vs Time in Closed-Loop Simulation

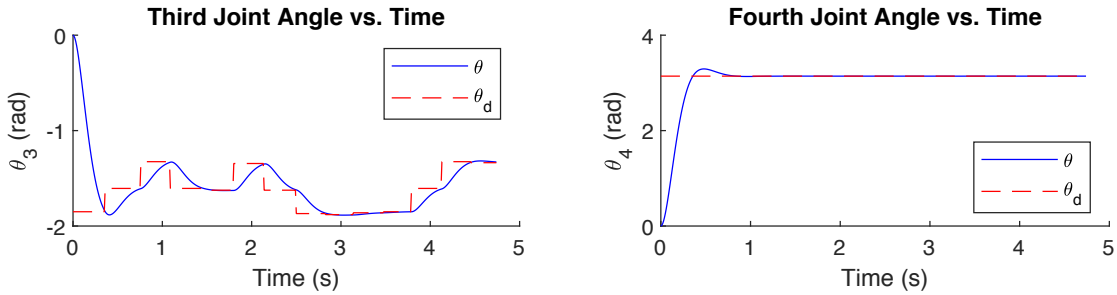


Figure 19 (cont.): Joint Angles vs Time in Closed-Loop Simulation

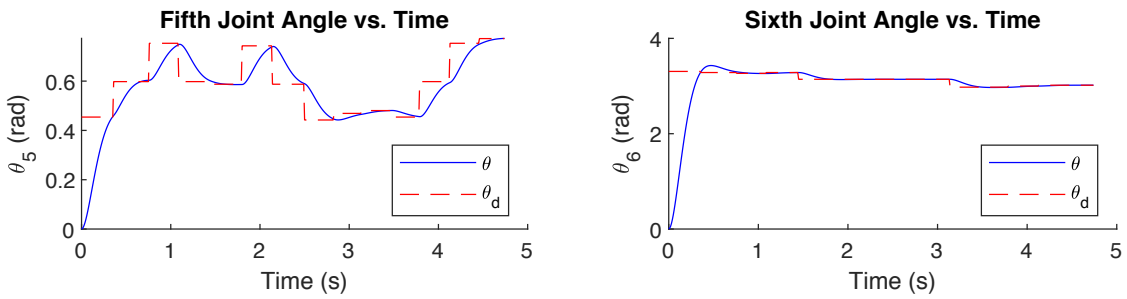


Figure 19 (cont.): Joint Angles vs Time in Closed-Loop Simulation

5.9 ANSYS

With 100% infill, 3D printed PLA has a maximum shear stress of 13.6 kpsi. The manipulator applies a load of 13N in the negative y-direction. Without gears in the base differential, the differential support would bear the load on its bearing mounts. *Figure 20* shows the manipulator's differential support could experience up to 97 kPa or 0.014 kpsi of shear stress, which is less than the maximum shear stress of PLA with 100% infill. Since some of the manipulator's mass is supported by the gears, the actual shear experienced by the differential support will be less.

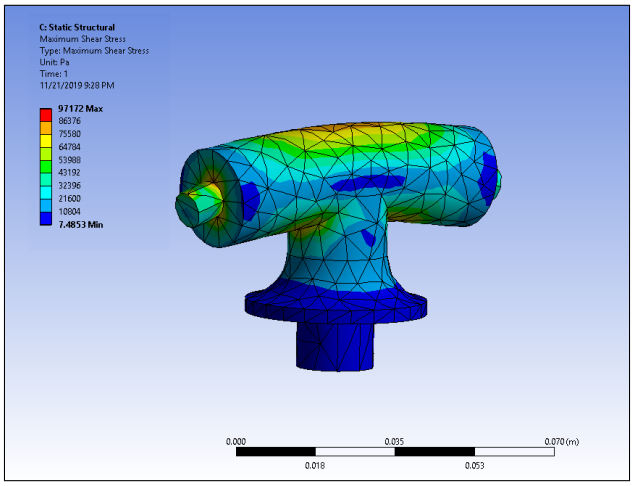


Figure 20: T-Bar ANSYS FEA

To simulate a dynamical loading situation where the manipulator would be under the largest amount of stress, gravitational forces and an outward force (parallel to the arm direction in it's zeroed configuration) were applied to the structure. This situation represents the worst-case loading scenario, such as the manipulator swinging while outstretched. The supports and simulated forces can be seen in the ANSYS image capture shown in *Figure 21*.

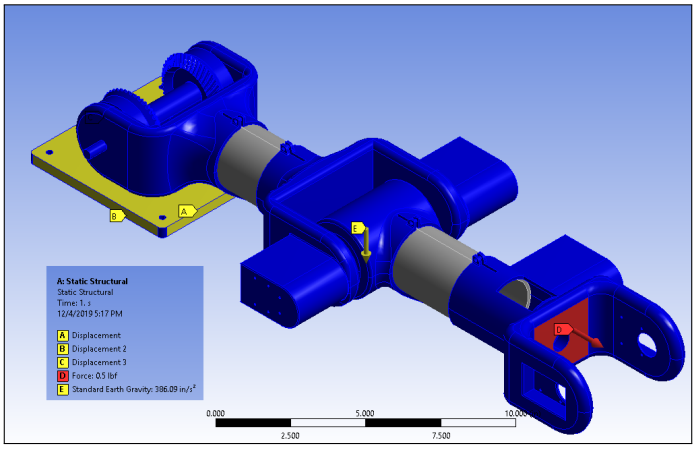
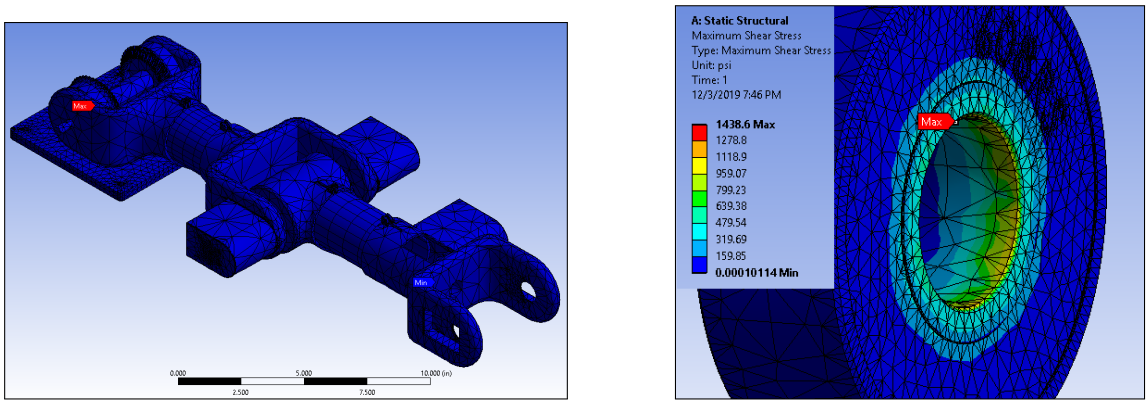


Figure 21: ANSYS Simulated Forces Image Capture

As shown in *Figure 21*, the red arrow is the outward force simulating centrifugal forces, the yellow arrow represents gravity acting at the manipulator's center of mass, and the yellow highlighted faces show the fixed support at the base.

The dynamical loadings resulted in a maximum shear stress at the shoulder differential bearing, as seen in *Figure 22a*; a close-up image of the bearing analysis can be seen in *Figure 22b*.



(a) ANSYS Full View of Maximum Shear (b) ANSYS Bearing Shear Stress Close-up

Figure 22: ANSYS FEA of Dynamical Loading Scenario

To further validate that the structure is capable of handling alternating stresses, a fatigue test was also performed showing the life of the manipulator handles a minimum of 1e6 cycles, as seen in *Figure 23*, showing it is unlikely to fail due to material yeilding.

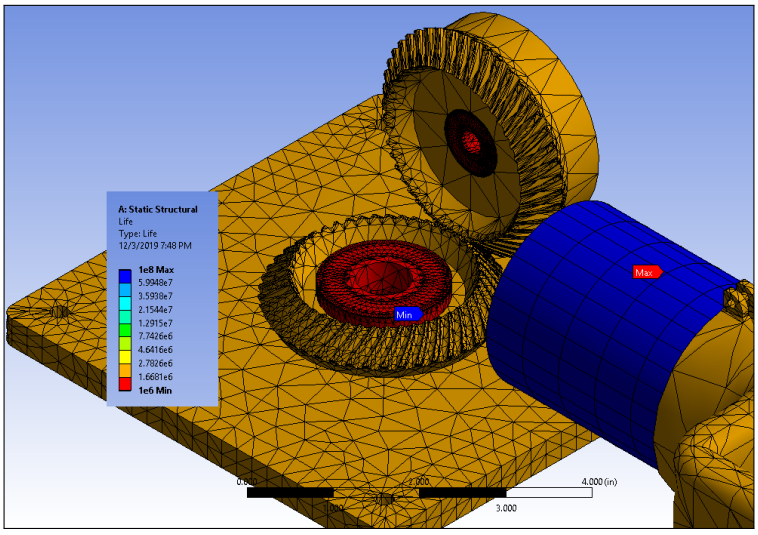


Figure 23: ANSYS Fatigue Test

As seen in *Figure 23*, the lower bearing of the differential drive on the shoulder of the manipulator would be the most likely component to fail under repeating loadings.

5.10 Electrical Schematic

The electrical schematic shown in *Figure 24* consists of a power supply that receives 120V AC that is standardly supplied from wall outlets. The power supply has two outputs, a 5V terminal that has a max current output of 3A, and a 12V terminal that has a max output of 7A. The Raspberry Pi 3B is powered by the 5V terminal, and since the max current draw from the Pi is 1.2A the maximum current draw from the terminal is not surpassed. The six MX-12W servos used in the manipulator, as well as the AX-12A servo used in the end effector, are connected in parallel across the 12V terminal from the power supply. Since the MX-12W has a stall current of .6A and the AX-12A has a stall current of 1.5A, the maximum current to be pulled from the 12V terminal would be 5.1A, less than the 7A the supply can output. In order to communicate to the servos, a U2D2 communication converter is connected to the Raspberry Pi which creates a serial connection used by the servos, allowing the servos to be daisy-chained together. The Raspberry Pi also connects to an external PC so that the user can communicate with the Pi using a mouse, keyboard, and monitor.

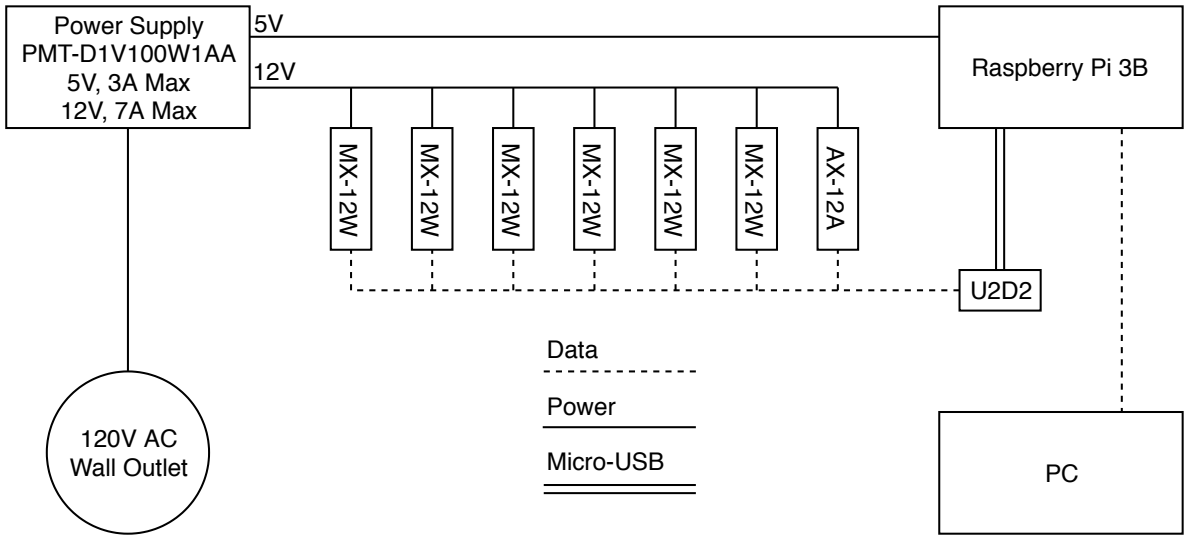


Figure 24: Electrical Schematic

5.11 Software Flowchart

The software the system uses takes in a row vector of position, orientation, path type, and end effector function information for all points to be traveled through and runs the manipulator through the desired points following the specified paths requested. The general overview of the code is shown in *Figure 25*.

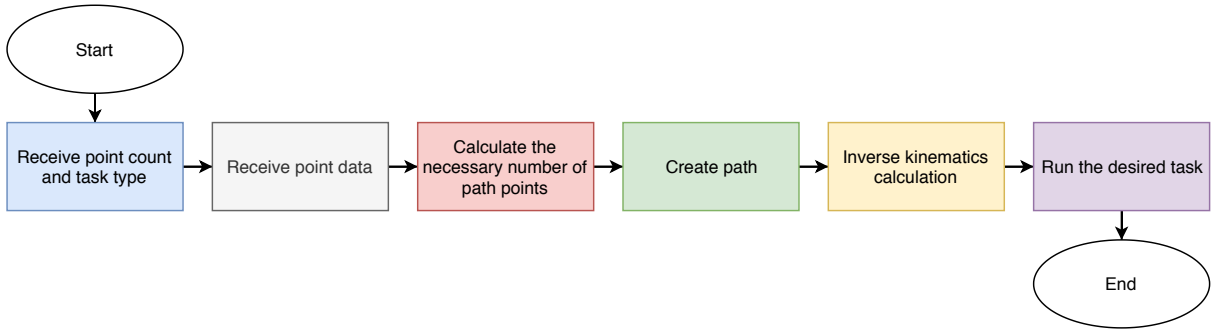


Figure 25: Software Flowchart

Figure 25 shows that the software for the manipulator is broken up into six subsections, two sections that receive data, three that do calculations, and one that runs the specified task.

The first subsection of the software works to receive the number of points the user is inputting as well as the general task the user is completing, shown in *Figure 25a*.

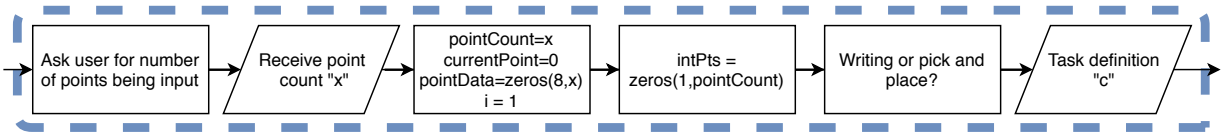


Figure 25a: Software Flowchart Subsection 1

Figure 25a specifies that the software prompts the user for the number of points that the manipulator will travel through and stores the input as a variable, in this case 'x'. The 'x' variable is only used to help preallocate data vectors so that the size of the vector does not change with each input. The software also receives the task specification as either a 0 for cartesian straight line pathing or a 1 for a straight line in the joint space and stores this value in the variable 'c'.

The second general block in the software flowchart works to receive and store the necessary data for the points the user is inputting depending on the path type as seen in *Figure 25b*.

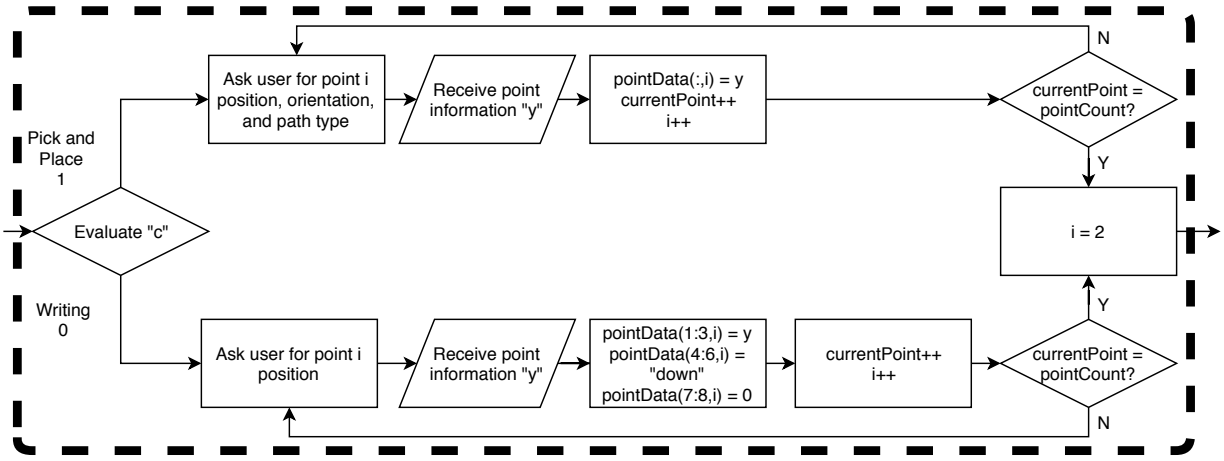


Figure 25b: Software Flowchart Subsection 2

Figure 25b shows that the path type variable 'c' is used to determine what information is necessary to collect. If the user is doing a writing task, the software only collects the x, y, and z distances for the point and assumes that the end effector orientation will be facing down so that the marker is vertical. If the user is doing pick and place, the software prompts the user for the x, y, z, phi, theta, psi, path type, and end effector data. The software loops until all the points have been input.

The next block in the software flowchart calculates the total points necessary to complete the task. The overview for this section can be seen in *Figure 25c*.

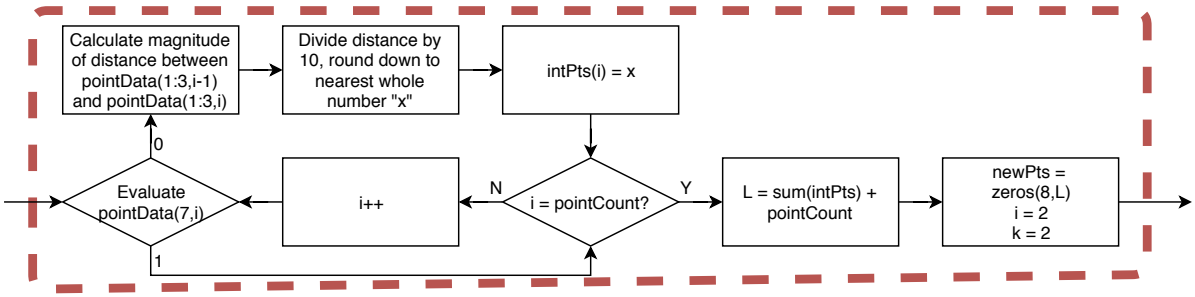


Figure 25c: Software Flowchart Subsection 3

As seen in *Figure 25c*, the section of software calculates the distance between the current point and the previous point if the path type is cartesian straight line and divides the distance by ten to find the number of centimeters between the two points. This value is stored as the necessary number of intermediate points, and the software will loop through until every point has been checked. The section of code also stores the total number of points that will be used as the variable level for later use.

The fourth code block in the flowchart creates and stores the necessary intermediate points along the desired path, shown in *Figure 25d*.

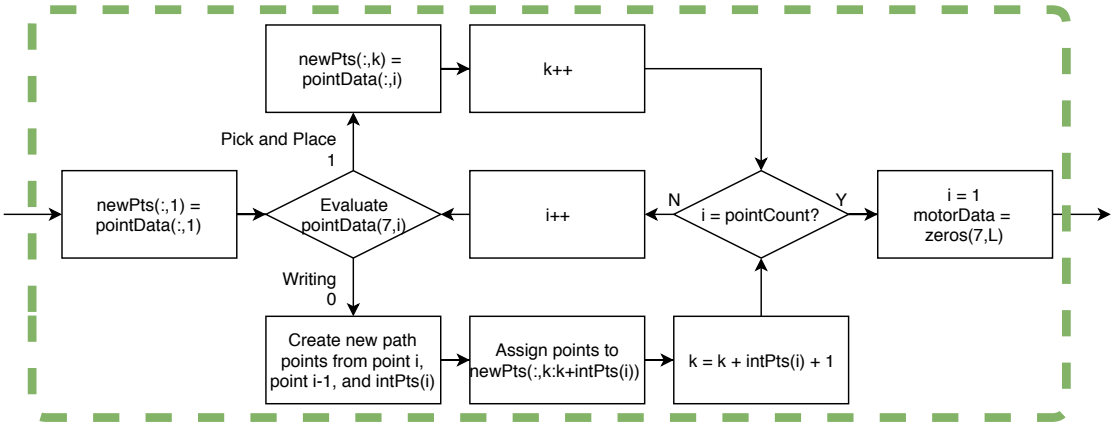


Figure 25d: Software Flowchart Subsection 4

The code shown in *Figure 25d* creates points every centimeter if the path type is cartesian straight line using the number of path points stored for each point from the previous block of software. This ensures that a straight line will be followed between the two user input points. If the path type is a straight line in the joint space, the software does not add any intermediate points since the path seen in the cartesian space does not matter.

The fifth code block in the flowchart calculates inverse kinematics of the points defined in the previous block of code and stores the angles as counts that can be used by the servos. The overview of this section can be seen in *Figure 25e*.

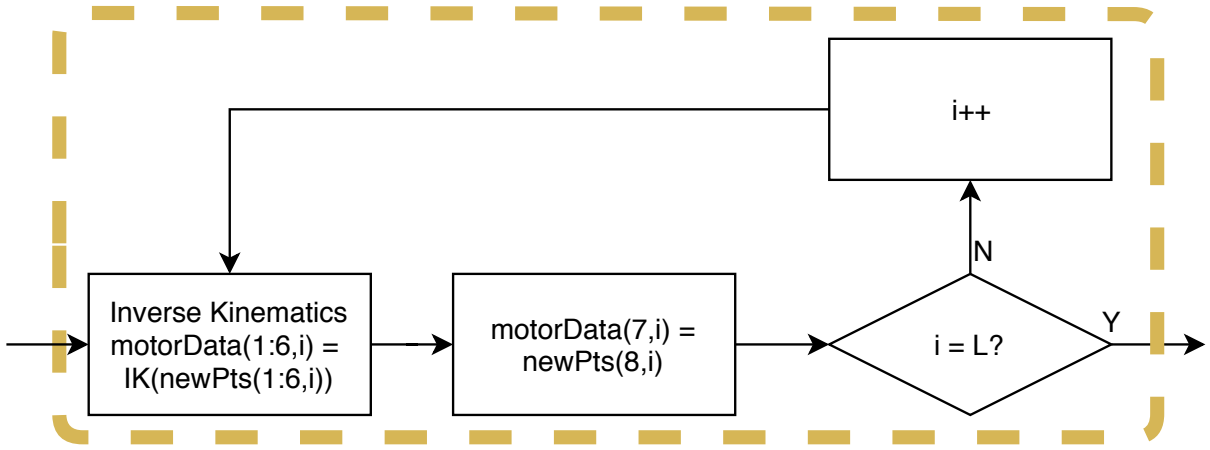


Figure 25e: Software Flowchart Subsection 5

Figure 25e shows that the new points found in the prior section of code are run through an inverse kinematics function that will output the necessary counts the servos can utilize. The code iterates through each point until the inverse kinematics have been calculated for all points.

The final block in the software diagram runs the manipulator through the desired task, with this section of code requiring user input at certain stages depending on the path type, seen in *Figure 25f*.

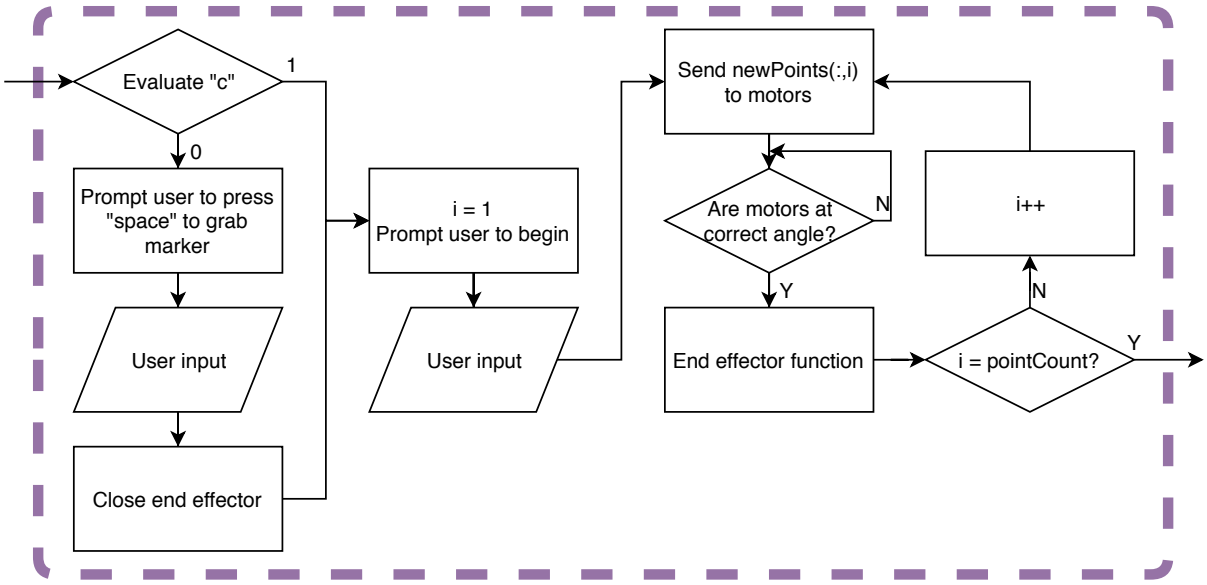


Figure 25f: Software Flowchart Subsection 6

The code in *Figure 25f* prompts the user to press space to close the end effector and grab the marker if drawing was the specified task, otherwise the software jumps straight into prompting the user to begin the task, and when the user begins the task the counts for each position are sent to the servos one at a time. The counts for the next position are not sent to the servos until the servos have reached the desired positions and the end effector function has been completed if there is one.

5.12 Project Status and Future

— Print Differential

— Cut Aluminum Supports

— Print Base

— Print Link 1

— Test Link 1 and Differential

— Print Differential Support

— Test Differential

— Test Link 1 Pulleys

— Test Link 1 with Motors

— Test Motor Control of Differential

— Test Link 1 With Load

— Implement Kinematics in Python

— Test Kinemtics with Motors

5.13 Parts List

Table 7 lists the parts MEIOSIS will require to build the manipulator. The total cost is $629.22 including shipping. Specification 1.1b required MEIOSOS's cost to develop the manipulator be less than $800. In addition to pulley belts for the current configuration, *Table 7* allocates $20 for two additional belt sizes to increase joint 1/2's and joint 3's torque by a factor of 10. One belt size and the belt must be purchased in packs of 3 from Automation Direct. *Table 7* accounts for increased cable lengths of 500 mm to communication bus signals from motor 2 to 3 and motor 3 to 4 and 350 mm to communication bus signals from motor 5 to 6. Motors are identified in [ZACK'S FIGURE]. Aside from electronic hardware, Figure ZZ accounts for physical hardware: To 100 threaded press-fit inserts. The manipulator will require between 21 and 46 inserts. The majority of inserts attach MX-12W servos to the manipulator. Mounting hardware accompanies each servo.

Table 7: MEIOSIS Bill of Materials with Costs

Part	**Retailer**	**Quantity**	**Unit Cost (USD)**	**Total Cost (USD)**
3 pack, 300 tooth	Automation Direct	1	11.5	11.5
3 pack, 208 tooth		1	9.5	28.5
Base second belts		1	10	10
Link 3 second belts		1	10	10
MX-12W	Trossen Robotics	6	65.9	395.4
500 mm, 1/2 pulleys		2	3.95	7.9
350 mm, 3 pulley		1	2.95	2.95
EE		1	24.95	24.95
Pi 3 B	Amazon	1	37.99	37.99
Bearings		1	8.99	8.99
2 Sch 10 12" Al tube	Industrial Metal Sales	1	2.99	2.99
12 V, 5 V power supply	Digi-Key Electronics	1	43.21	43.21
Automation Direct	Shipping	—	—	0
Trossen Robotics				13.15
Amazon				0
Industrial Metal Sales				26.36
Digi-Key				8.99
			Total	629.22

In addition to the costs listed in *Table 7*, *Table 8* shows all further costs for the end-user. Since the Embry-Riddle robotics lab has 3D printing available without affecting MEIOSIS's

$800 budget, *Table 8* accounts for outsourced 3D printing costs sufficient to print the entire manipulator with six sets of pulleys. If the end-user owns a 3D printer, the 3D printing cost would effectively reduce to filament cost. Additionally, *Table 8* assumes the end-user does not already possess an AX-12A servo to be used with the end-effector. Further, *Table 8* assumes the manipulator would be more accessible to end-users by using a proprietary U2D2 communication module in lieu of a soldered or bread-board circuit. The robotics lab has an AX-12A servo and U2D2 communication module MEIOSIS will use. With the aforementioned additional costs, the MEIOSIS manipulator costs the end-user $1,007.98 including shipping costs. While $1,007.98 is slightly above the maximum cost of $1000 from specification 1.1a, it provides greater accessibility, which may be diminished by assuming end-users posess 3D printers.

Table 8: End-User Bill of Materials with Costs

Part	**Retailer**	**Quantity**	**Unit Cost (USD)**	**Total Cost (USD)**
Aforementioned Costs	*Table 7*	—	—	629.22
U2D2	Trossen Robotics	1	49.9	49.9
EE with servo	Trossen Robotics	1	64.95	64.95
3D PLA outsourcing (incl. shipping)	Craft Cloud	1	288.86	288.86
			Total	1007.98

Acknowledgements & Attributions

We would like to acknowledge the following people for their contibutions in creating this report.

— Dr. Isenberg

— Dr. Schipper

References

[1] Trossen robotics. URL *https://www.trossenrobotics.com/*.

[2] Embry-riddle aeronautical university-prescott. URL *https://www.collegetuitioncompare.com/edu/104586/embry-riddle-aeronautical-university-prescott/*.

[3] Range of robot cost – robot system cost series. URL *https://motioncontrolsrobotics.com/range-robot-cost/*.

[4] Sawyer: Rethink robotics unveils new robot. URL *https://spectrum.ieee.org/automaton/robotics/industrial-robots/sawyer-rethink-robotics-new-robot*.

[5] Michael Zeltkevic 1. Runge-kutta methods, 04 1998. URL *http://web.mit.edu/10.001/Web/Course_Notes/Differential_Equations_Notes/node5.html*.

[6] S. Hutchinson M. Spong and M. Vidyasager. *Robot Modeling and Control*. 2006.

[7] ROBOTIS. Ax-12a e-manual, 2019. URL *http://emanual.robotis.com/docs/en/dxl/ax/ax-12a/*.

A Appendix

A.I Relevant Figures and Materials

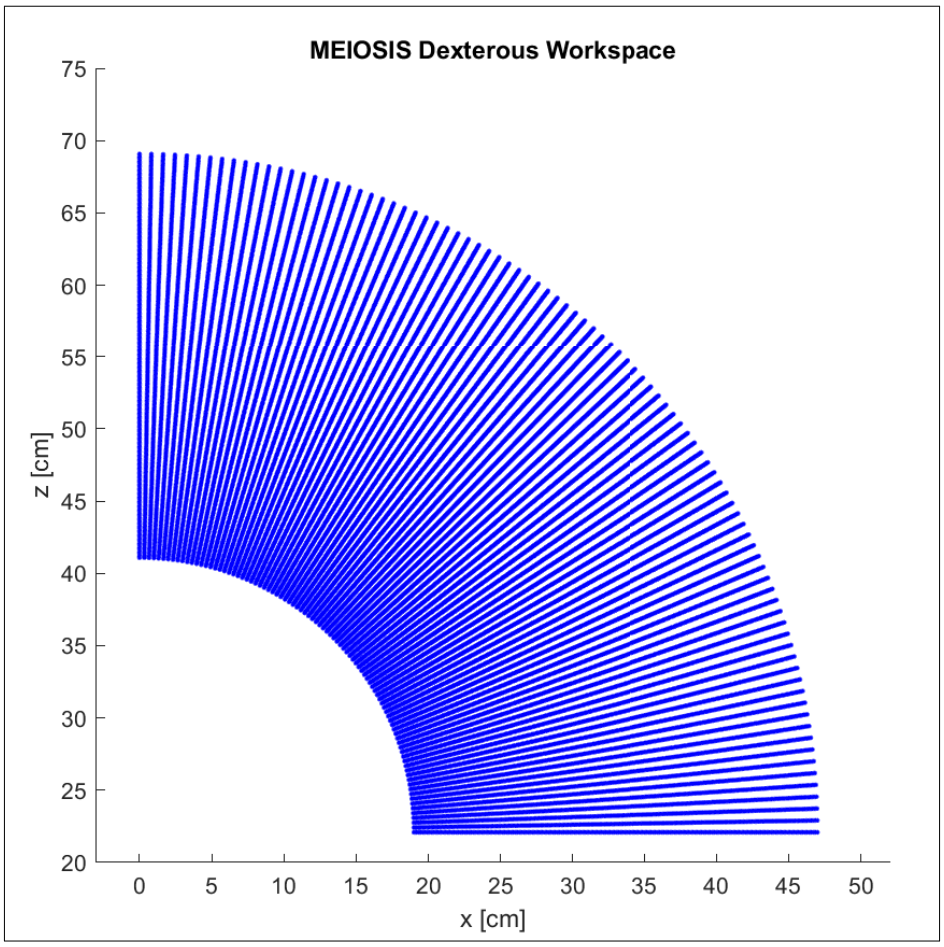
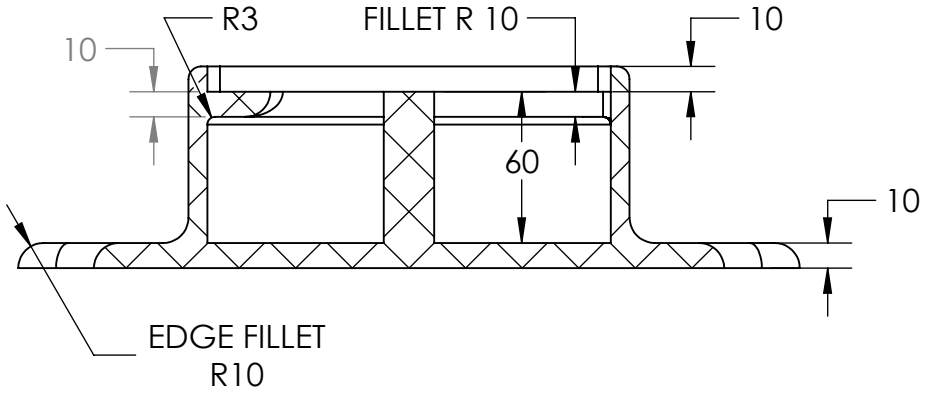
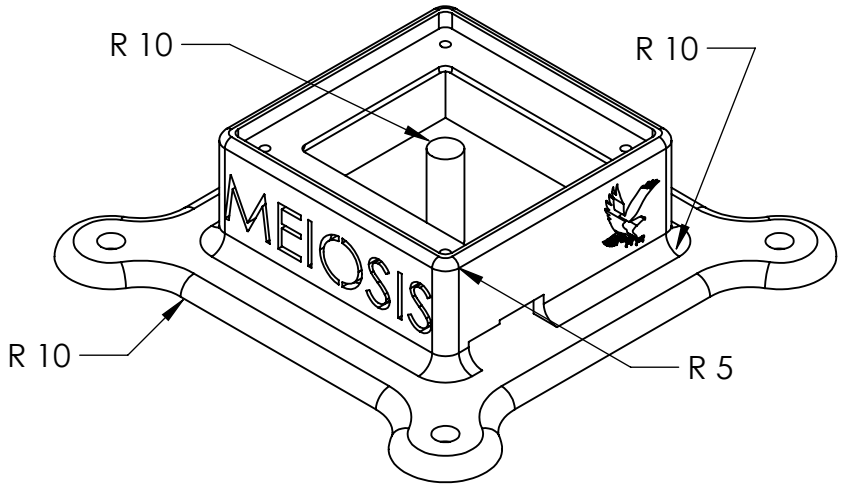


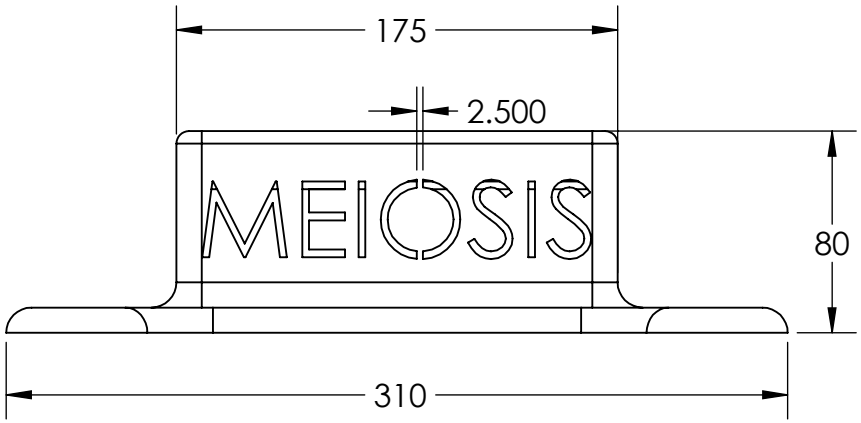
Figure 26: Cross Section of Dexterous Workspace Quadrant

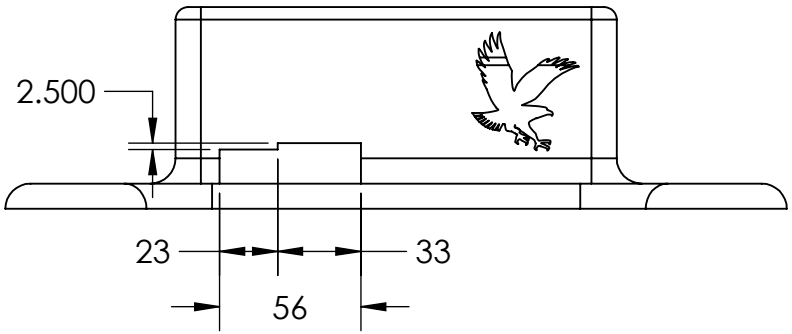
A.II CAD Drawings

The complete drawing package is attached.







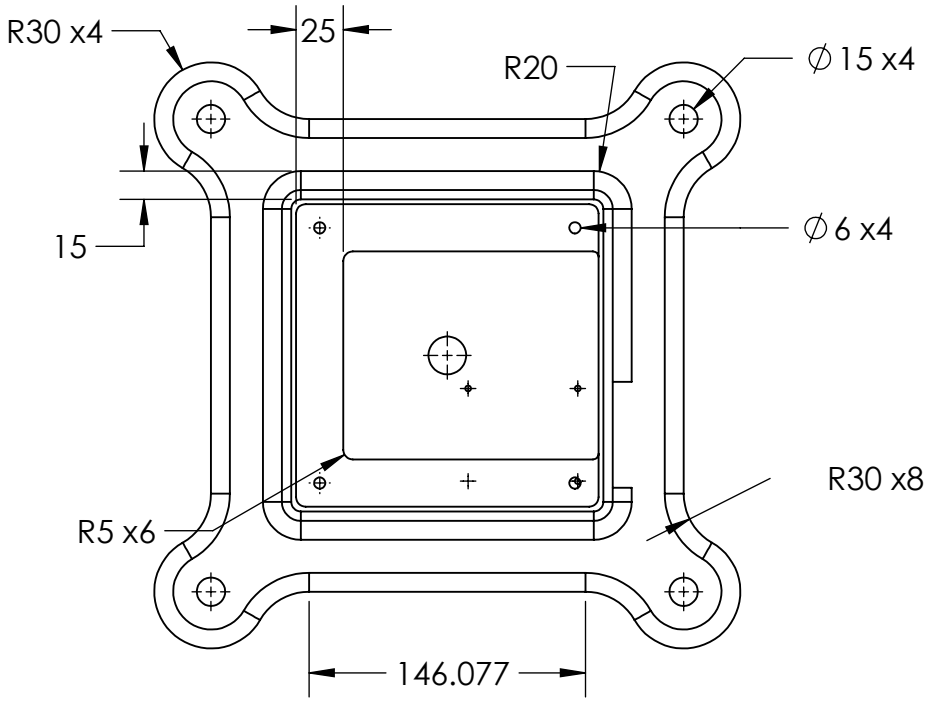


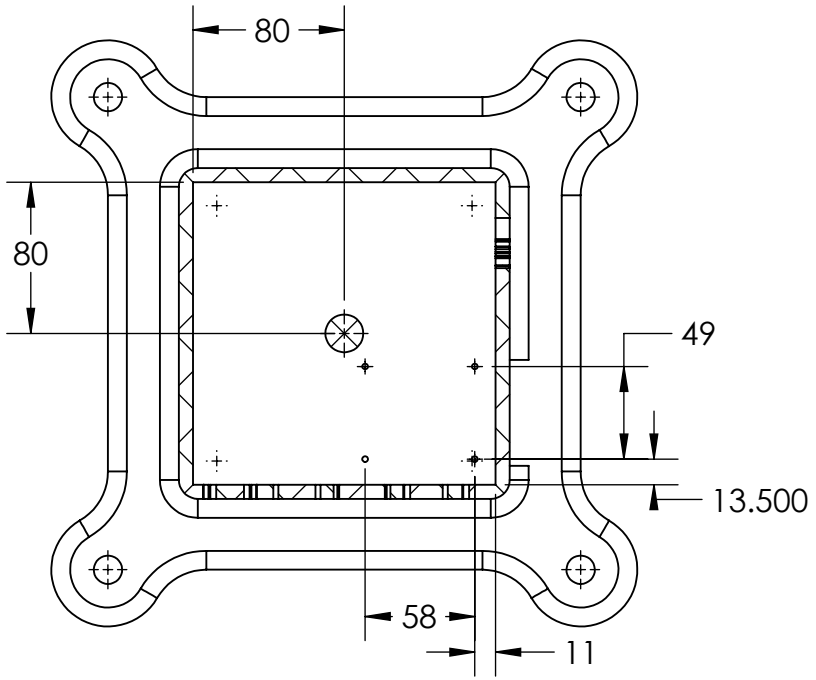
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN MILLIMETERS TOLERANCES: ±3.0	DRAWN	Ryan W.	11/13/2019
	CHECKED		
	ENG APPR.		
	MFG APPR.		
INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
MATERIAL: PLA, 30% Infill	COMMENTS:		
FINISH: N/A			
DO NOT SCALE DRAWING			

TITLE: Base Bottom

SIZE	DWG. NO.	REV
A		1.0

SCALE: 1:5 | WEIGHT: | SHEET 1 OF 18



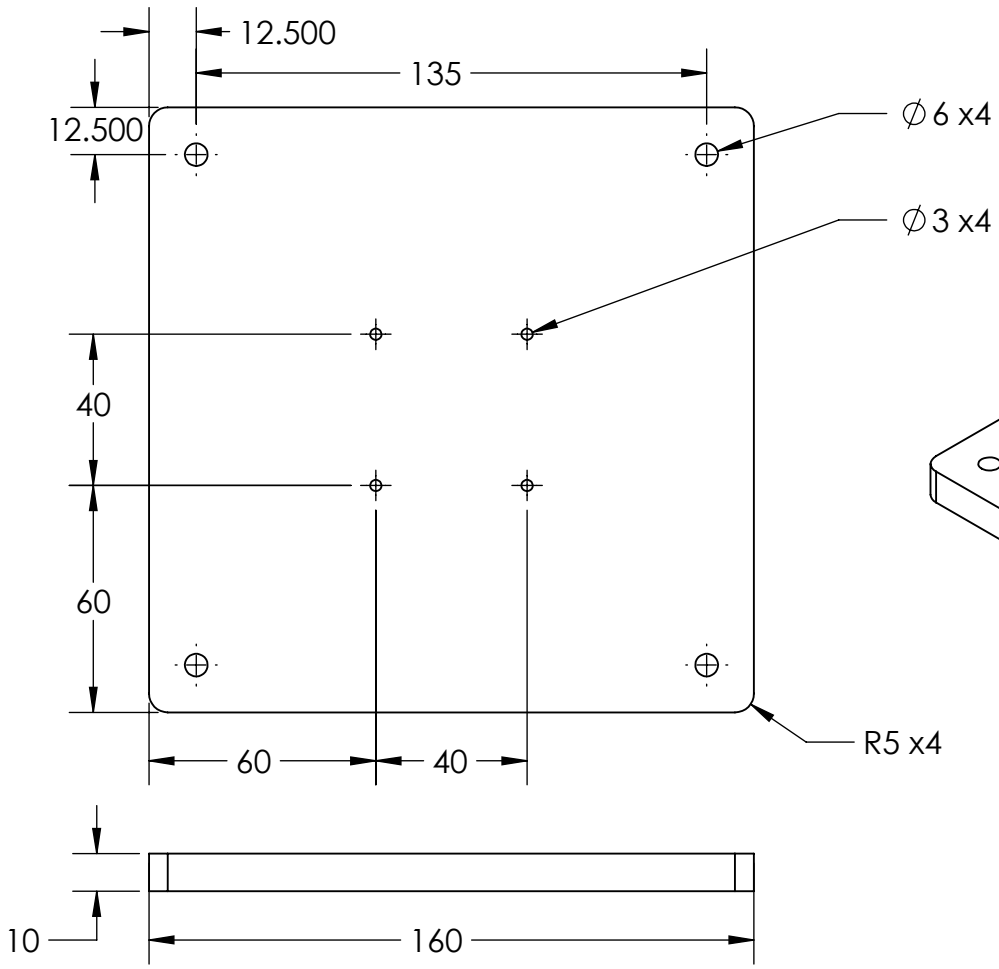


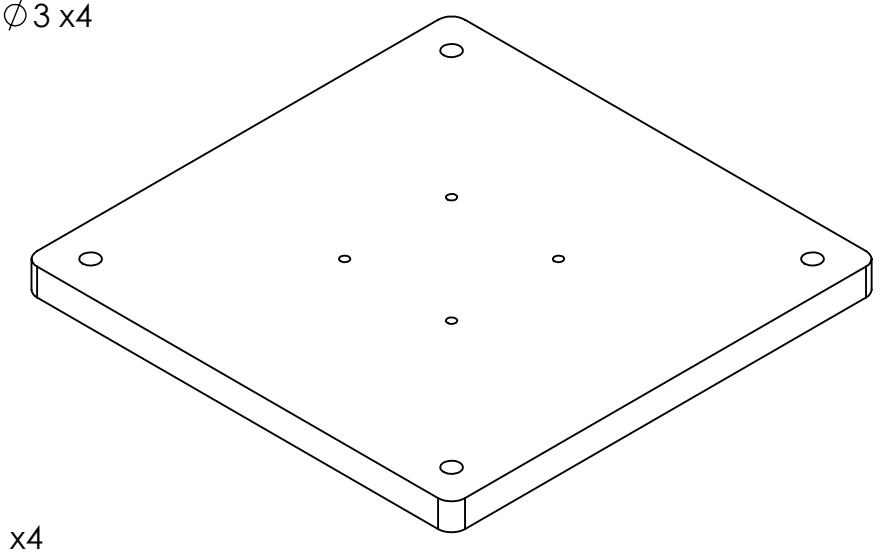
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN MILLIMETERS TOLERANCES: ±3.0	DRAWN	Ryan W.	11/13/2019
	CHECKED		
	ENG APPR.		
	MFG APPR.		
INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
MATERIAL: PLA, 30% Infill	COMMENTS:		
FINISH: N/A			
DO NOT SCALE DRAWING			

TITLE: Base Bottom

SIZE	DWG. NO.	REV
A		1.0

SCALE: 1:5 | WEIGHT: | SHEET 2 OF 18



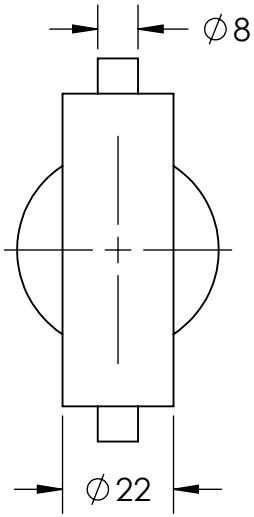


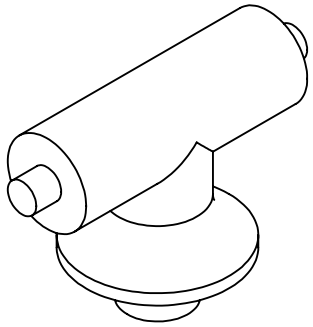
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN MILLIMETERS TOLERANCES: ±3.0	DRAWN	Ryan W.	11/8/2019
	CHECKED		
	ENG APPR.		
	MFG APPR.		
INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
MATERIAL: PLA, 30% Infill	COMMENTS:		
FINISH: N/A			
DO NOT SCALE DRAWING			

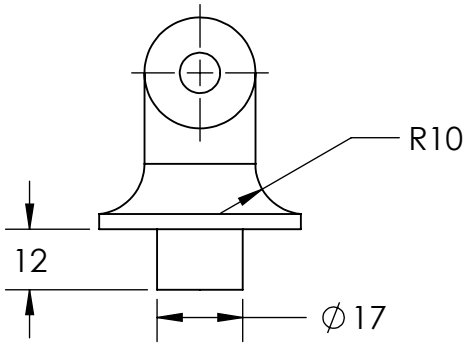
TITLE: Base Top

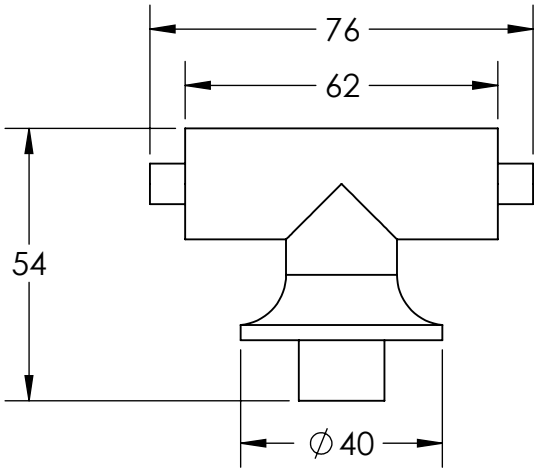
SIZE	DWG. NO.	REV
A		1.0

SCALE: 1:5 | WEIGHT: | SHEET 3 OF 18









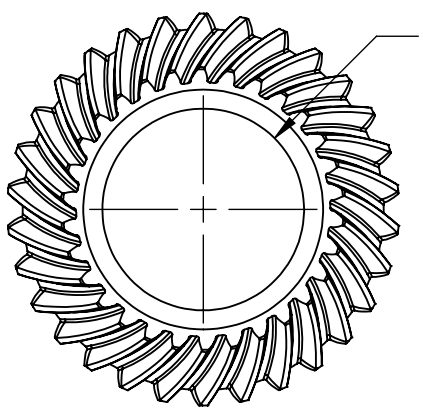
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN MILLIMETERS TOLERANCES: ±3.0	DRAWN	Ryan W.	11/8/2019
	CHECKED		
	ENG APPR.		
	MFG APPR.		
INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
MATERIAL PLA, 30% Infill	COMMENTS:		
FINISH N/A			
DO NOT SCALE DRAWING			

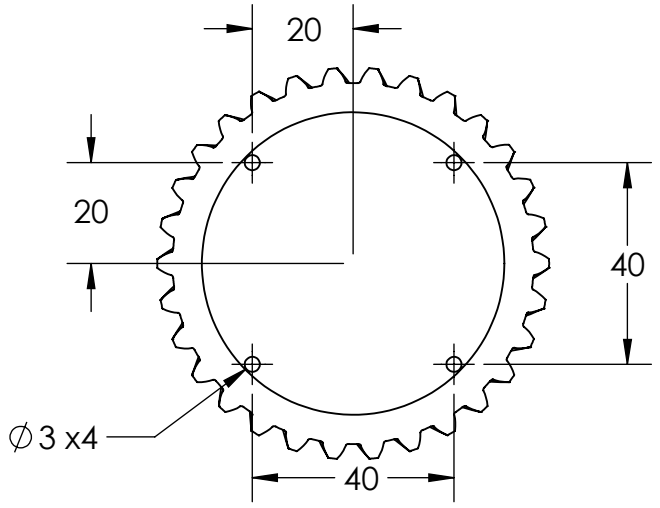
TITLE:

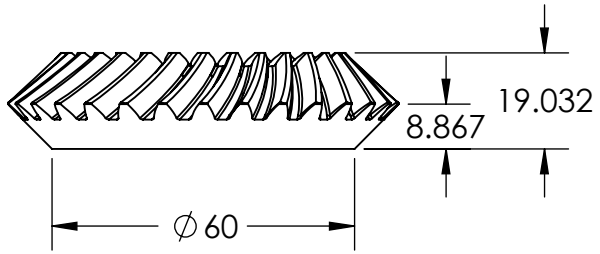
Shoulder Differential T-Bar

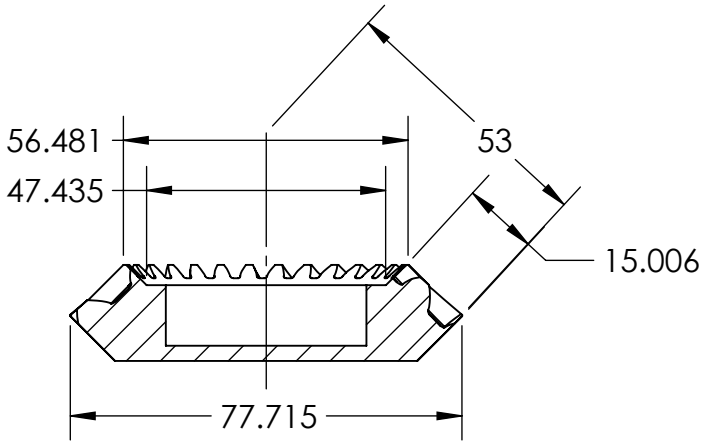
SIZE	DWG. NO.	REV
A		1.0

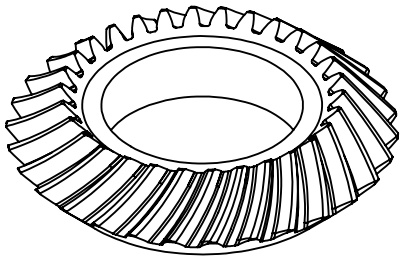
SCALE: 1:2	WEIGHT:	SHEET 4 OF 18











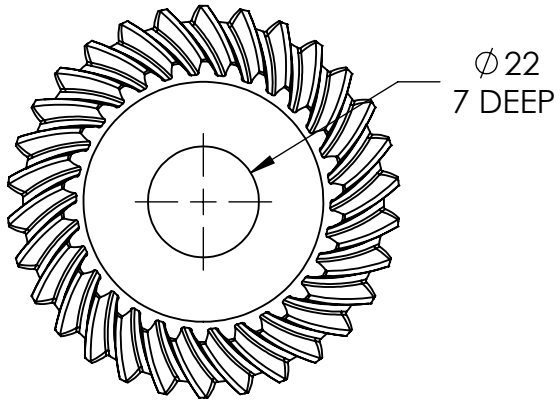
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN MILLIMETERS TOLERANCES: ±3.0	DRAWN	Ryan W.	11/8/2019
	CHECKED		
	ENG APPR.		
	MFG APPR.		
INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
MATERIAL: PLA, 20% Infill	COMMENTS:		
FINISH: N/A			
DO NOT SCALE DRAWING			

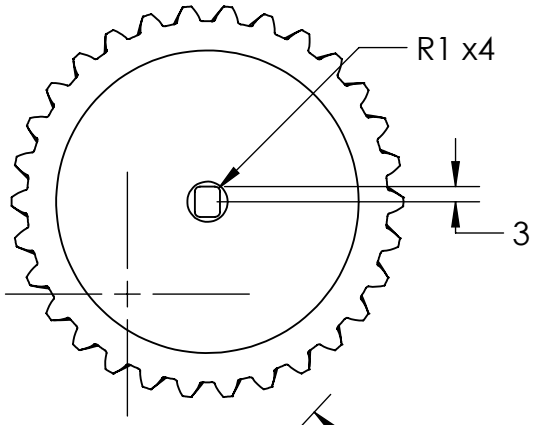
TITLE:

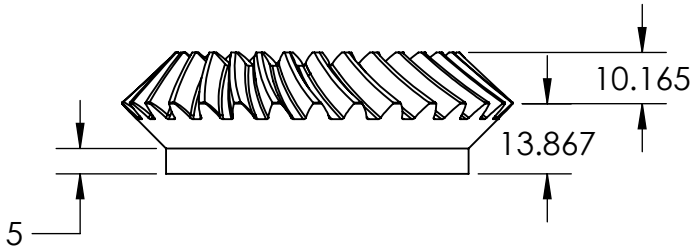
Shoulder Bevel Gear (1)

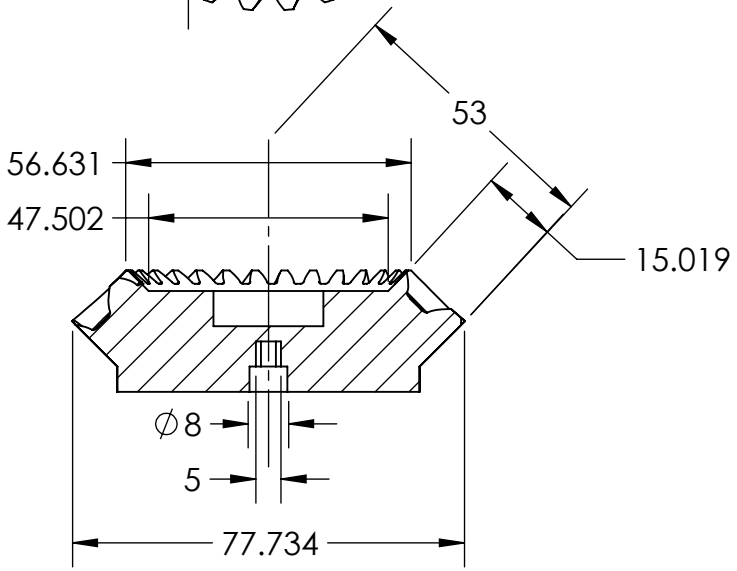
SIZE	DWG. NO.	REV
A		1.0

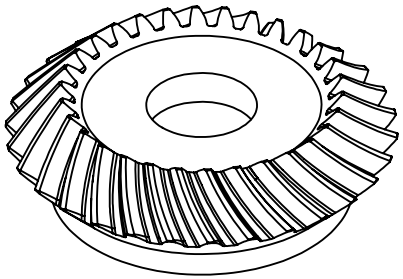
SCALE: 2:3	WEIGHT:	SHEET 5 OF 18











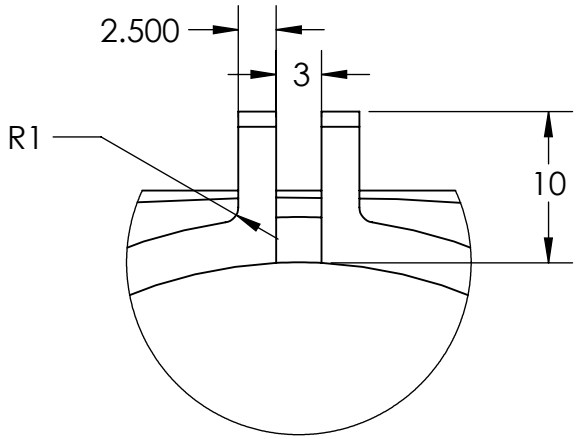
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN MILLIMETERS TOLERANCES: ±3.0	DRAWN	Ryan W.	11/8/2019
	CHECKED		
	ENG APPR.		
	MFG APPR.		
INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
MATERIAL: PLA, 20% Infill	COMMENTS:		
FINISH: N/A			
DO NOT SCALE DRAWING			

TITLE:

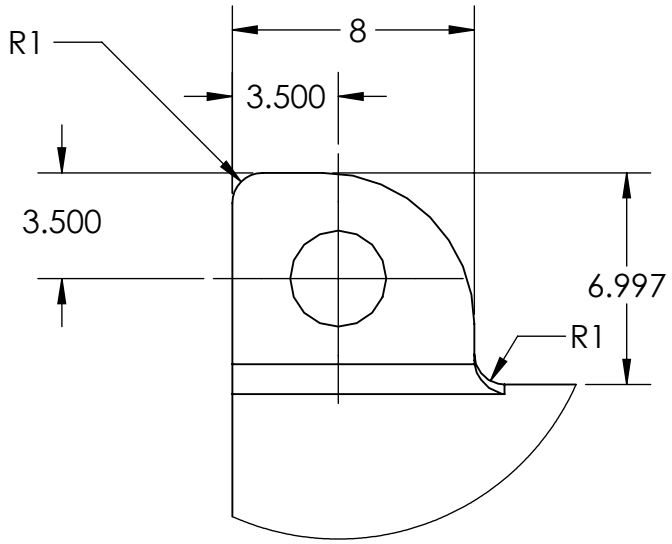
Shoulder Bevel Gear (2)

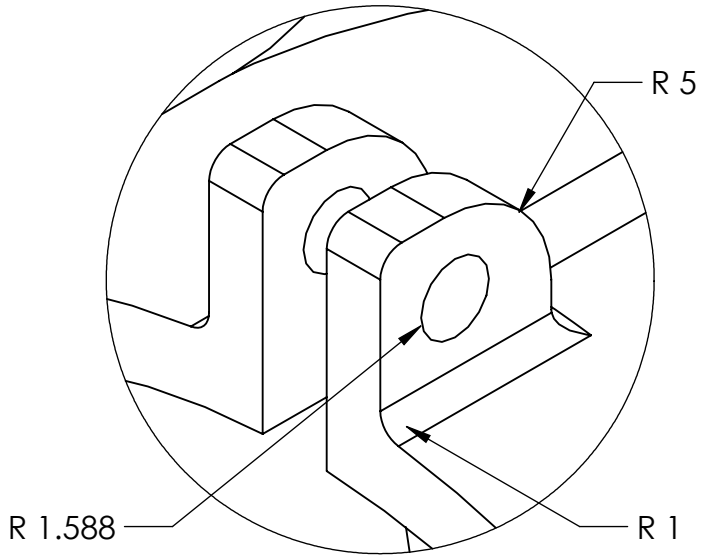
SIZE	DWG. NO.	REV
A		1.0

SCALE: 2:3	WEIGHT:	SHEET 6 OF 18



SCALE 2:1





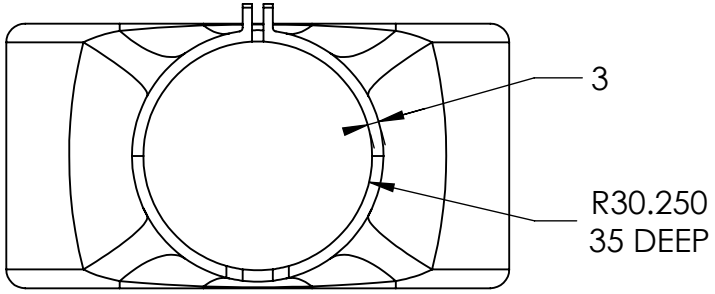
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN MILLIMETERS TOLERANCES: ±3.0	DRAWN	Ryan W.	11/20/2019
	CHECKED		
	ENG APPR.		
	MFG APPR.		
INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
MATERIAL: PLA	COMMENTS: THIS DRAWING SHOWS THE CLAMPING MECHANISM TO BE USED FOR LINK 2_A, LINK 2_B, LINK 3_A, AND LINK 3_B.		
FINISH: N/A			
DO NOT SCALE DRAWING			

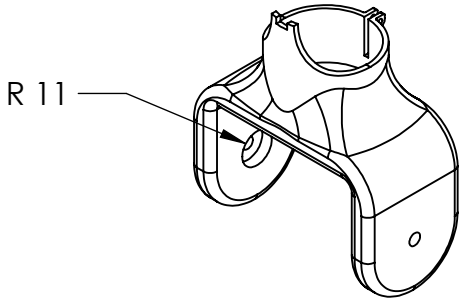
TITLE:

CLAMP DESIGN

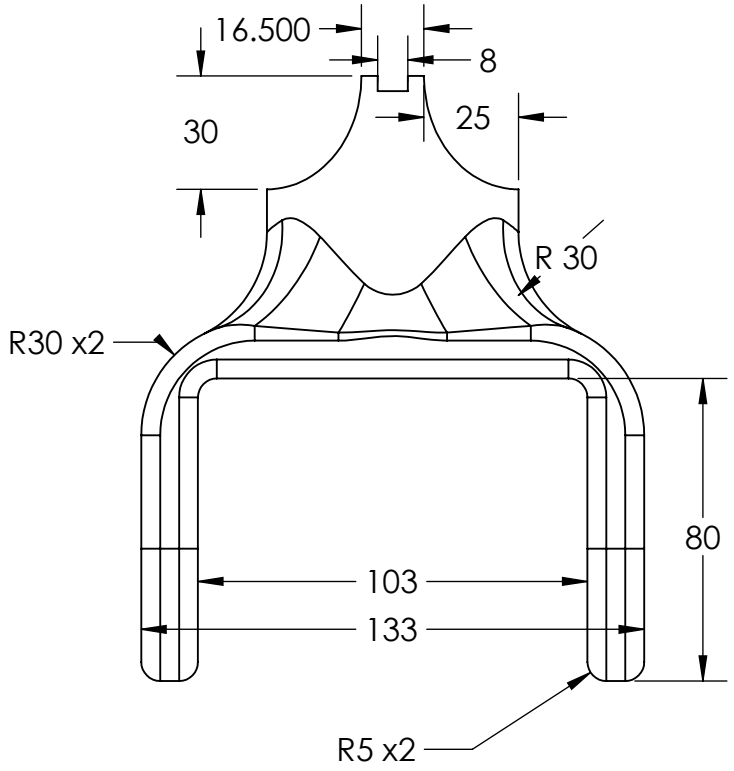
SIZE	DWG. NO.	REV
A		1.0

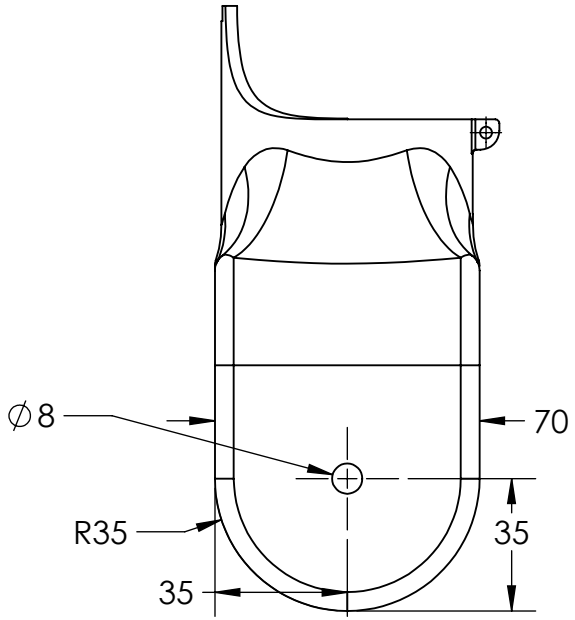
SCALE: 4:1	WEIGHT:	SHEET 7 OF 18





SCALE 1:4



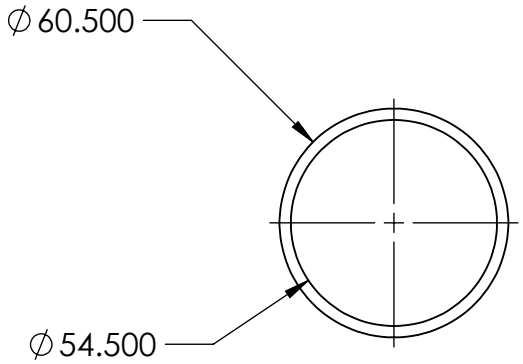


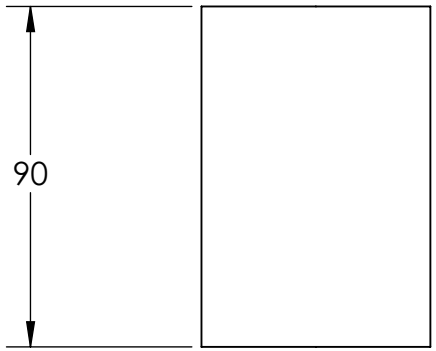
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN MILLIMETERS TOLERANCES: ±3.0	DRAWN	Ryan W.	11/11/2019
	CHECKED		
	ENG APPR.		
	MFG APPR.		
INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
MATERIAL: PLA, 30% Infill	COMMENTS:		
FINISH: N/A			
DO NOT SCALE DRAWING			

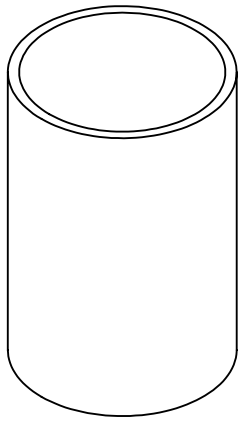
TITLE: Link 2_a

SIZE	DWG. NO.	REV
A		1.0

SCALE: 1:2	WEIGHT:	SHEET 8 OF 18





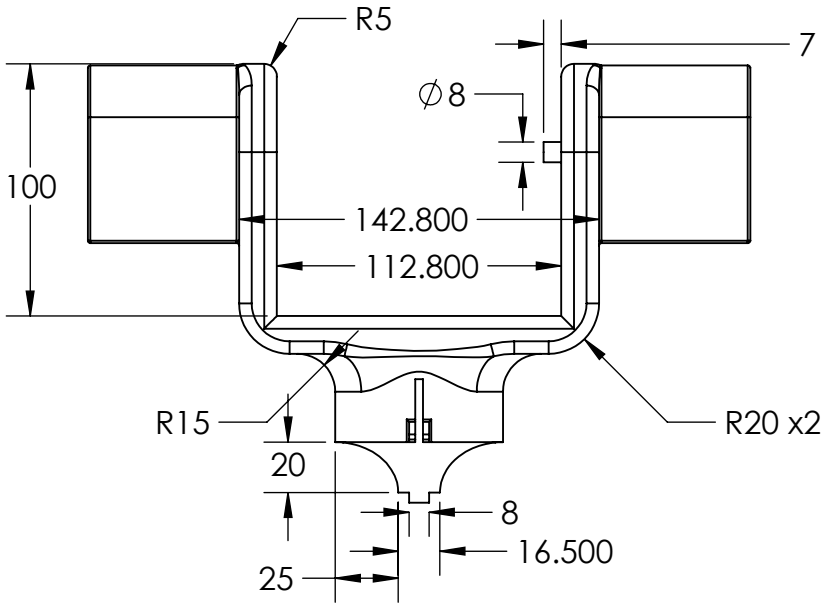


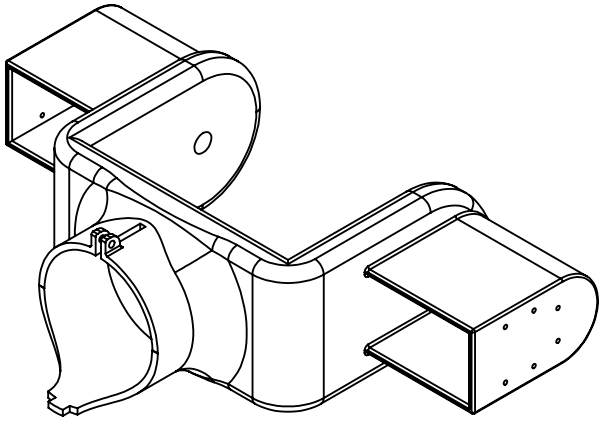
UNLESS OTHERWISE SPECIFIED:			DATE
DIMENSIONS ARE IN MILLIMETERS TOLERANCES: ±3.0	DRAWN	Ryan W.	11/11/2019
	CHECKED		
	ENG APPR.		
	MFG APPR.		
INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
MATERIAL: Aluminum	COMMENTS:		
FINISH: N/A			
DO NOT SCALE DRAWING			

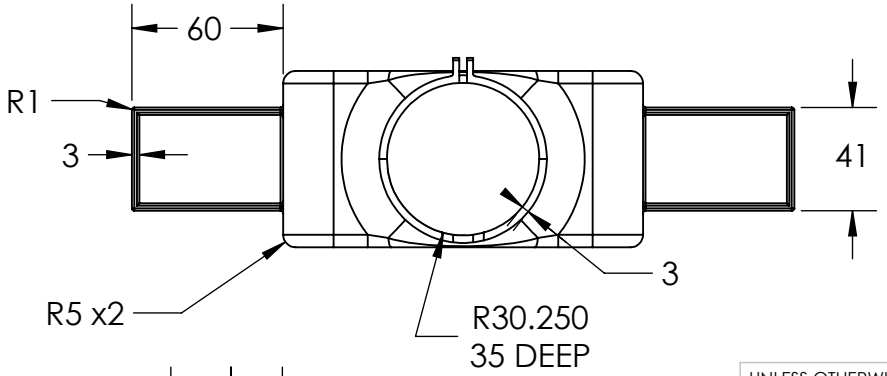
TITLE: Link 2 Tubing

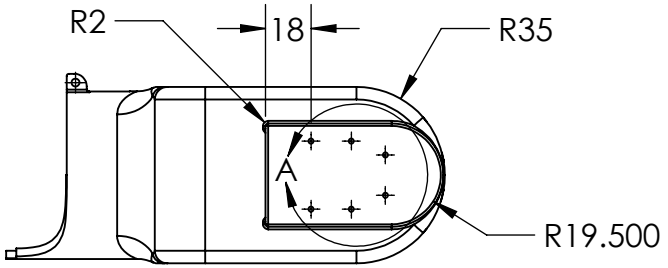
SIZE	DWG. NO.	REV
A		1.0

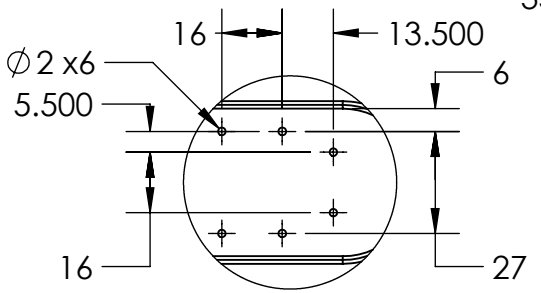
SCALE: 1:2	WEIGHT:	SHEET 9 OF 18











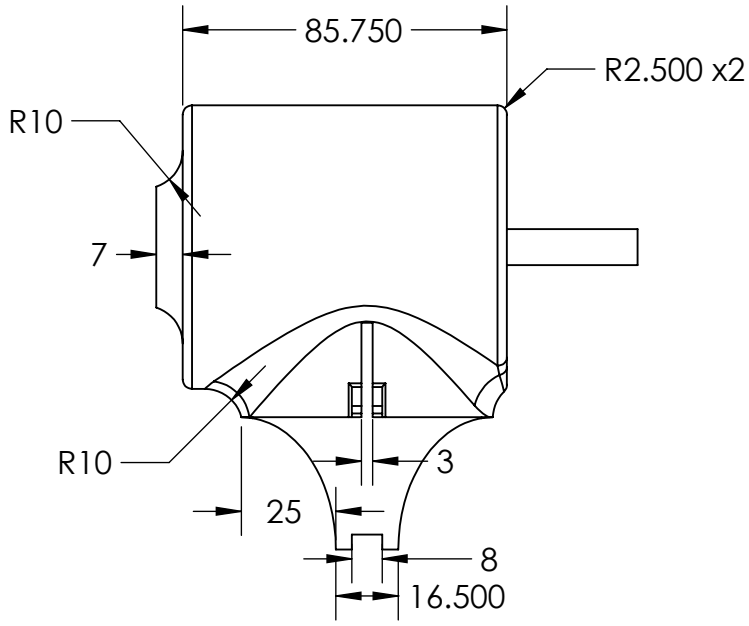
DETAIL A
SCALE 1 : 2

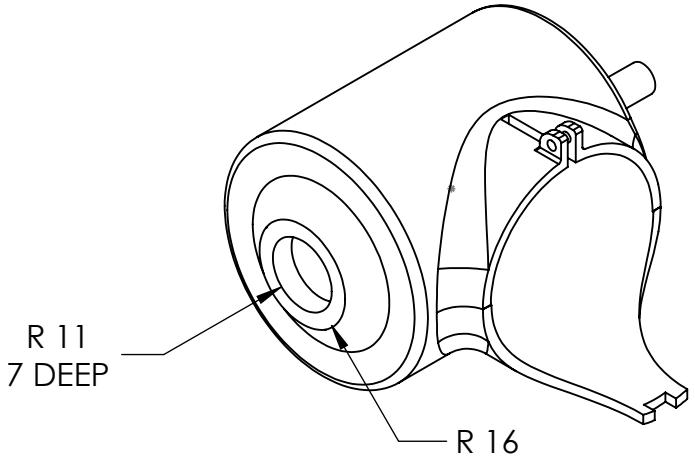
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN MILLIMETERS	DRAWN	Ryan W.	11/13/2019
TOLERANCES: ±3.0	CHECKED		
	ENG APPR.		
	MFG APPR.		
INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
MATERIAL: PLA, 30% Infill	COMMENTS:		
FINISH: N/A			
DO NOT SCALE DRAWING			

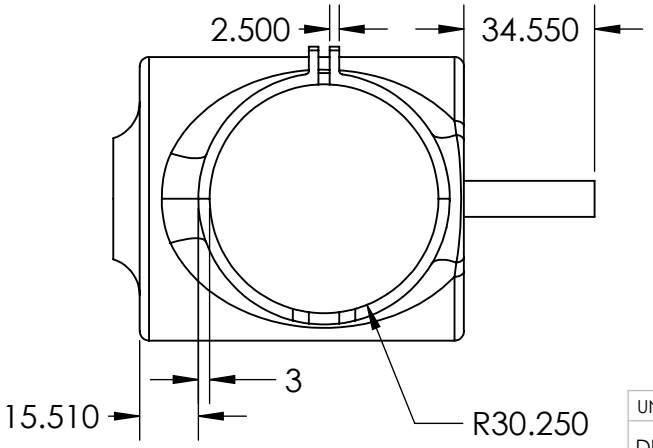
TITLE: Link 2_b

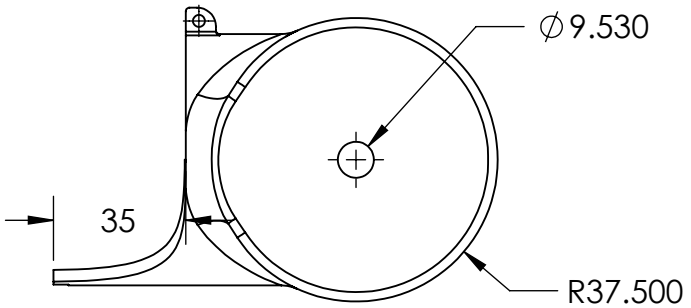
SIZE	DWG. NO.	REV
A		1.0

SCALE: 1:3 | WEIGHT: | SHEET 10 OF 18







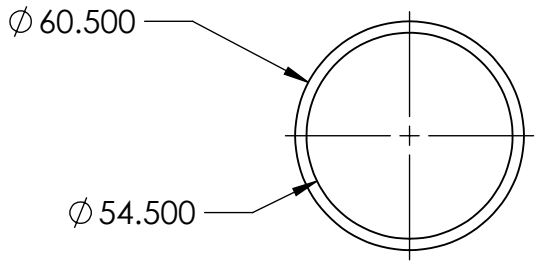


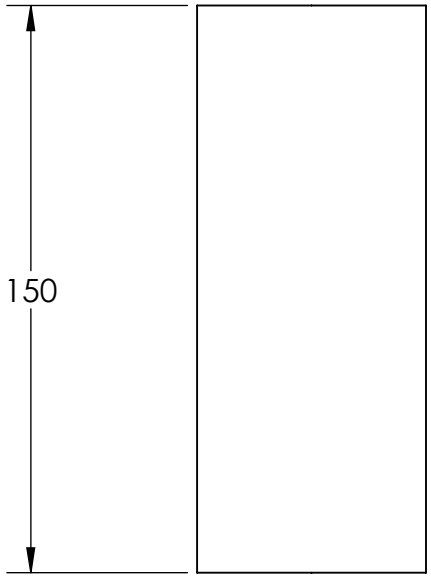
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN MILLIMETERS TOLERANCES: ±3.0	DRAWN	Ryan W.	11/11/2019
	CHECKED		
	ENG APPR.		
	MFG APPR.		
INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
MATERIAL: PLA, 30% Infill	COMMENTS:		
FINISH: N/A			
DO NOT SCALE DRAWING			

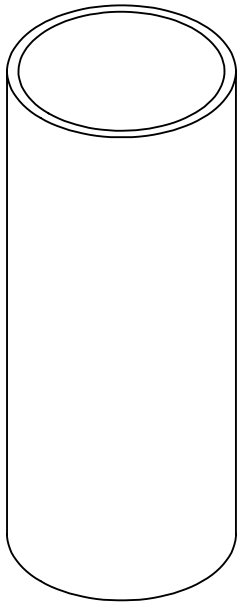
TITLE: Link 3_a

SIZE	DWG. NO.	REV
A		1.0

SCALE: 1:2 | WEIGHT: | SHEET 11 OF 18





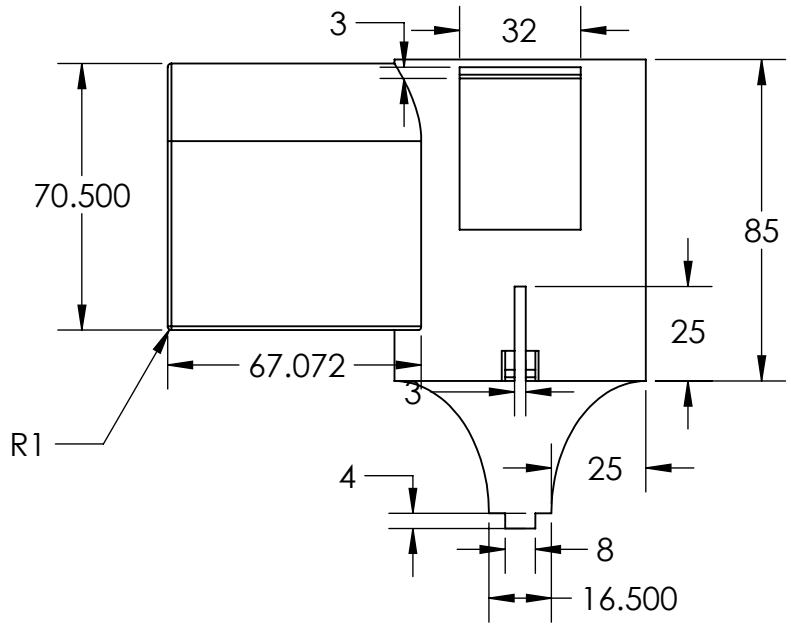


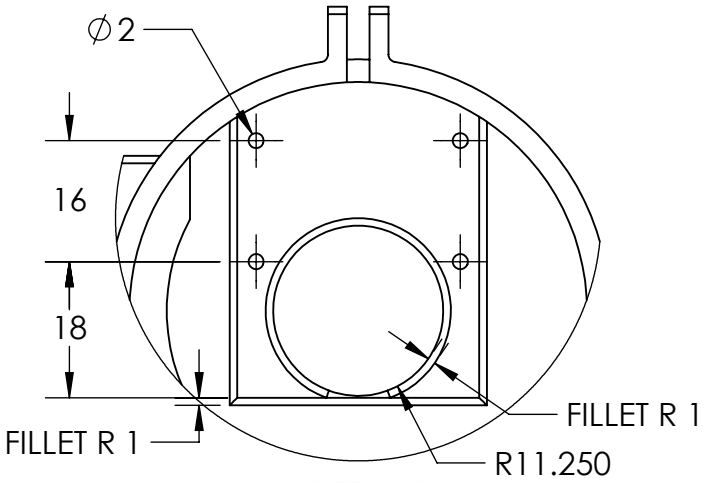
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN MILLIMETERS TOLERANCES: ±3.0	DRAWN	Ryan W.	11/11/2019
	CHECKED		
	ENG APPR.		
	MFG APPR.		
INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
MATERIAL: Aluminum	COMMENTS:		
FINISH: N/A			
DO NOT SCALE DRAWING			

TITLE: Link 3 Tubing

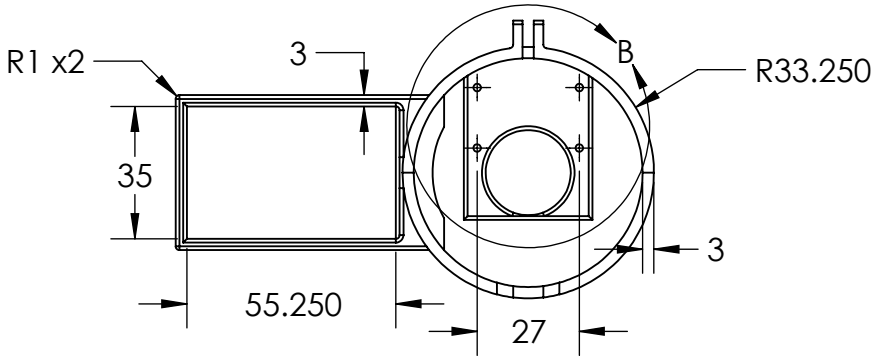
SIZE	DWG. NO.	REV
A		1.0

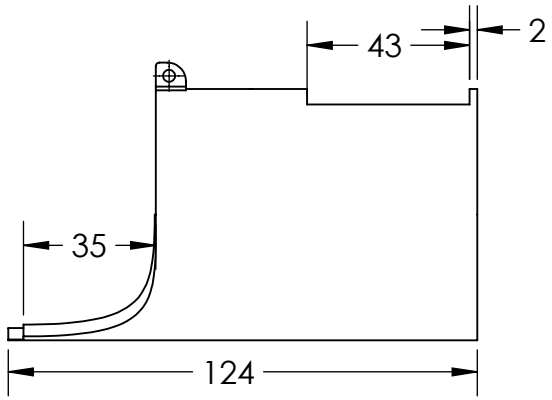
SCALE: 1:5	WEIGHT:	SHEET 12 OF 18

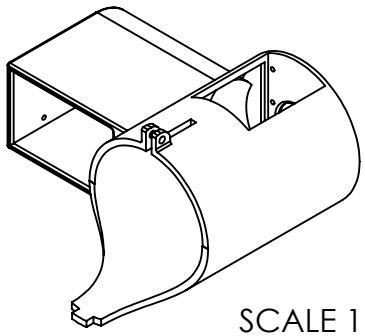




DETAIL B
SCALE 1 : 1







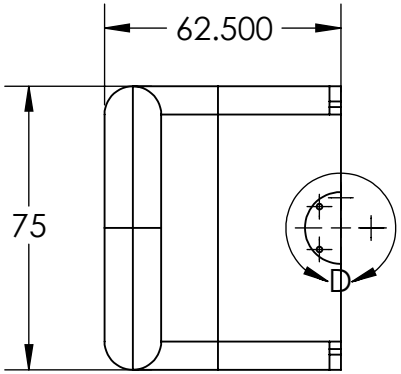
SCALE 1:3

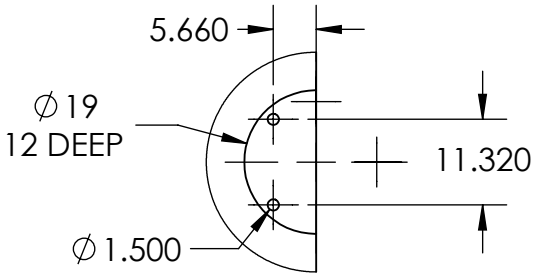
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN MILLIMETERS	DRAWN	Ryan W.	11/15/2019
TOLERANCES: ±3.0	CHECKED		
	ENG APPR.		
	MFG APPR.		
INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
MATERIAL PLA, 30% Infill	COMMENTS:		
FINISH N/A			
DO NOT SCALE DRAWING			

TITLE: Link 3_b

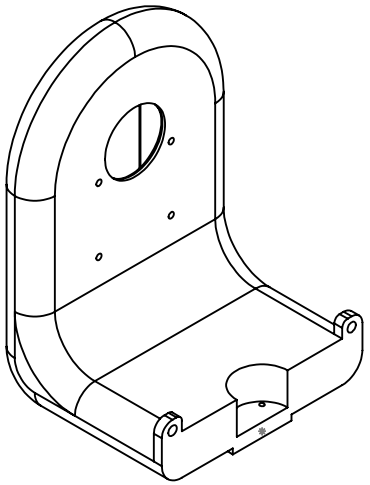
SIZE	DWG. NO.	REV
A		1.0

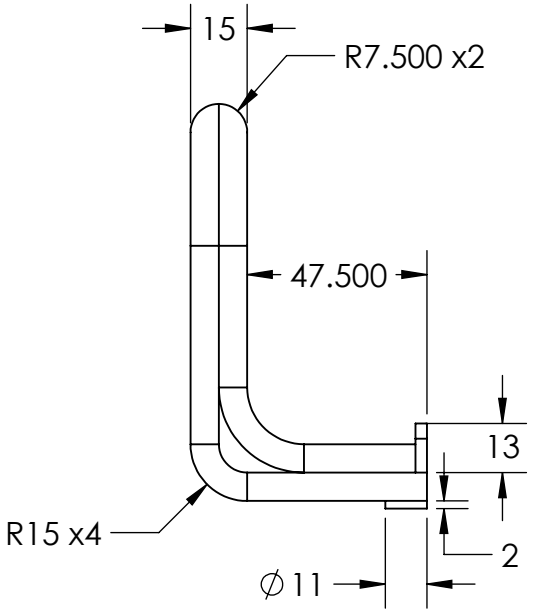
SCALE: 1:2 | WEIGHT: | SHEET 13 OF 18

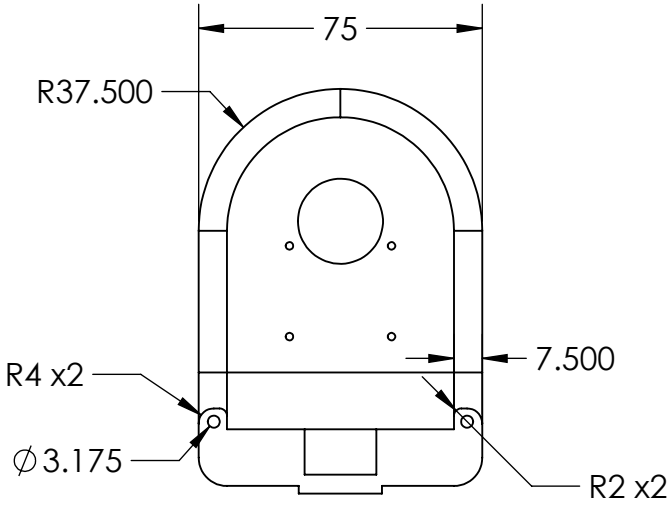




DETAIL D
SCALE 1 : 1





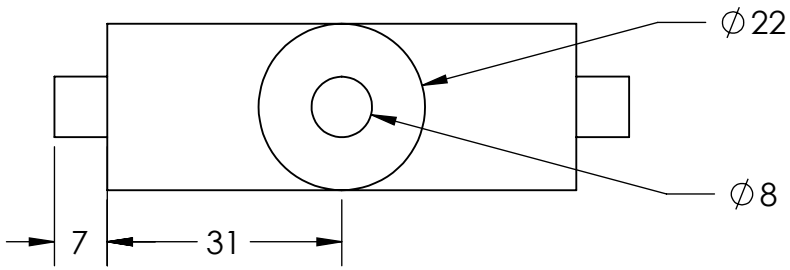


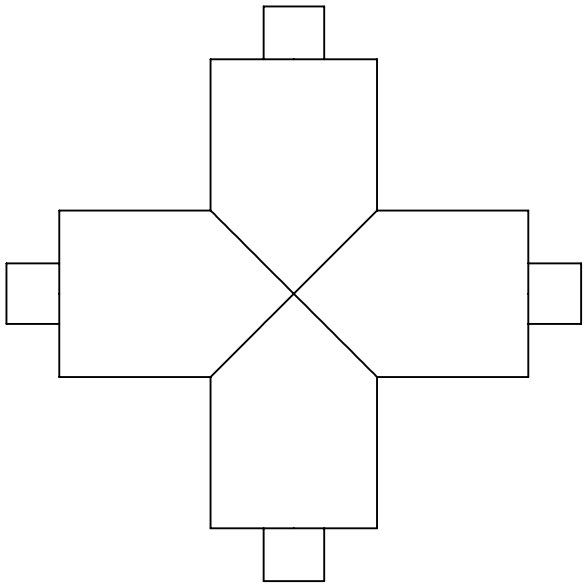
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN MILLIMETERS TOLERANCES: ±3.0	DRAWN	Ryan W.	11/11/2019
	CHECKED		
	ENG APPR.		
	MFG APPR.		
INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
MATERIAL: PLA, 30% Infill	COMMENTS:		
FINISH: N/A			
DO NOT SCALE DRAWING			

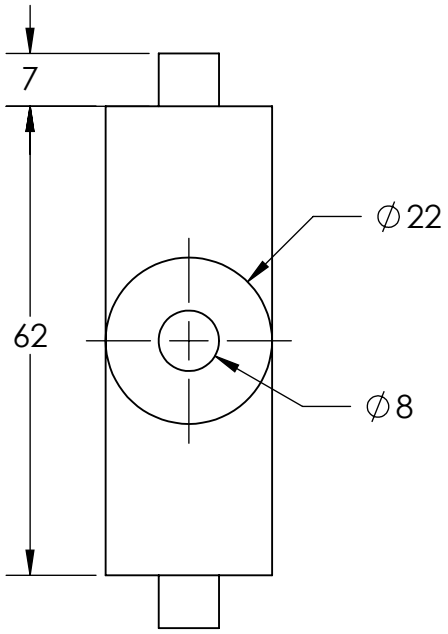
TITLE: Link 4

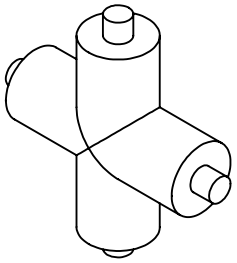
SIZE	DWG. NO.	REV
A		1.0

SCALE: 1:2 | WEIGHT: | SHEET 14 OF 18









SCALE 1:2

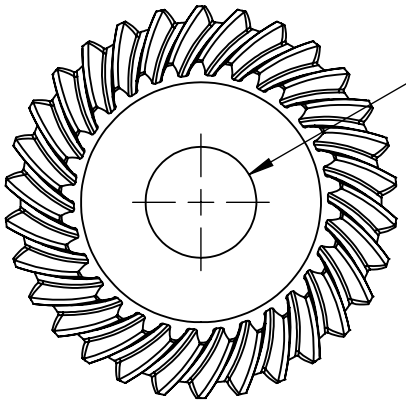
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN MILLIMETERS TOLERANCES: ±3.0	DRAWN	Ryan W.	11/8/2019
	CHECKED		
	ENG APPR.		
	MFG APPR.		
INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
MATERIAL: PLA, 30% Infill	COMMENTS:		
FINISH: N/A			
DO NOT SCALE DRAWING			

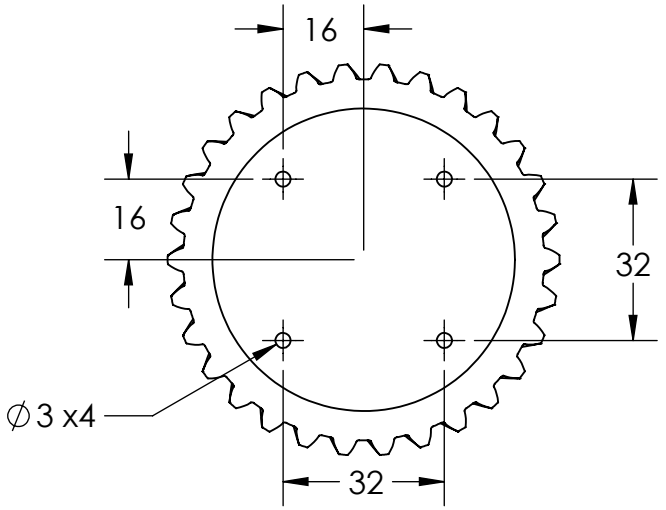
TITLE:

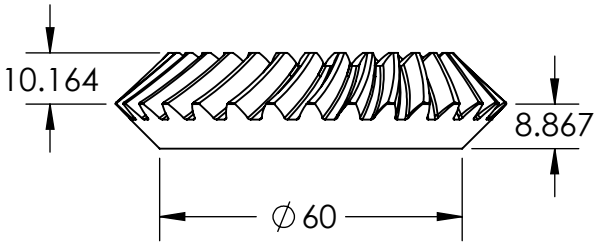
Wrist Differential Crossbar

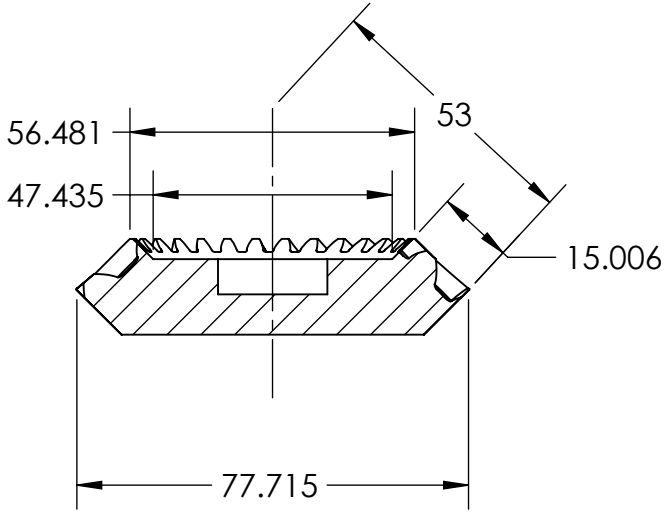
SIZE	DWG. NO.	REV
A		1.0

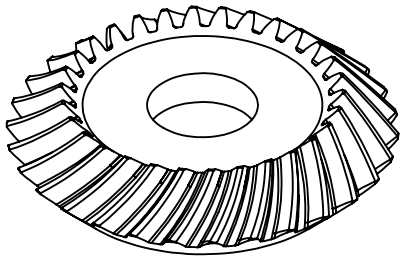
SCALE: 1:1	WEIGHT:	SHEET 15 OF 18











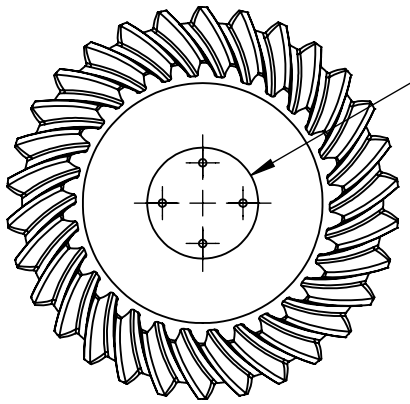
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN MILLIMETERS TOLERANCES: ±3.0	DRAWN	Ryan W.	11/8/2019
	CHECKED		
	ENG APPR.		
	MFG APPR.		
INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
MATERIAL: PLA, 20% Infill	COMMENTS:		
FINISH: N/A			
DO NOT SCALE DRAWING			

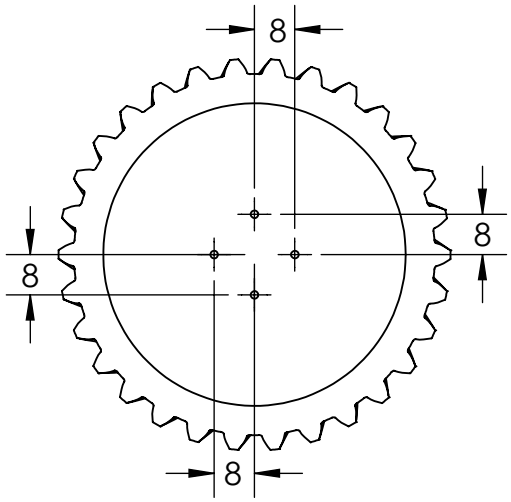
TITLE:

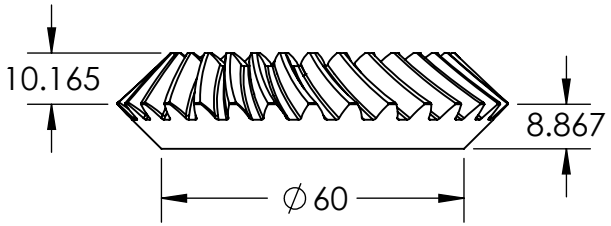
Wrist Bevel Gear (1)

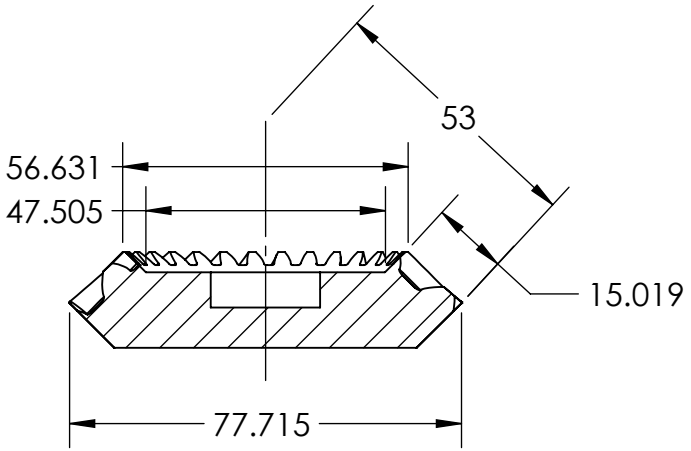
SIZE	DWG. NO.	REV
A		1.0

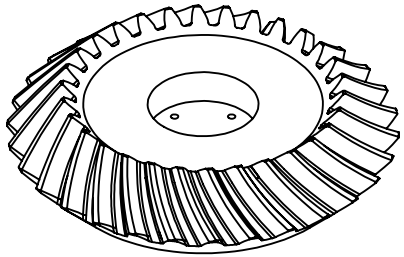
SCALE: 2:3 | WEIGHT: | SHEET 16 OF 18











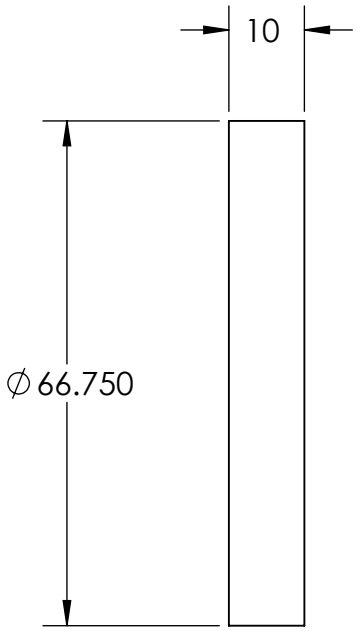
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN MILLIMETERS	DRAWN	Ryan W.	11/8/2019
TOLERANCES: ±3.0	CHECKED		
	ENG APPR.		
	MFG APPR.		
INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
MATERIAL: PLA, 20% Infill	COMMENTS:		
FINISH: N/A			
DO NOT SCALE DRAWING			

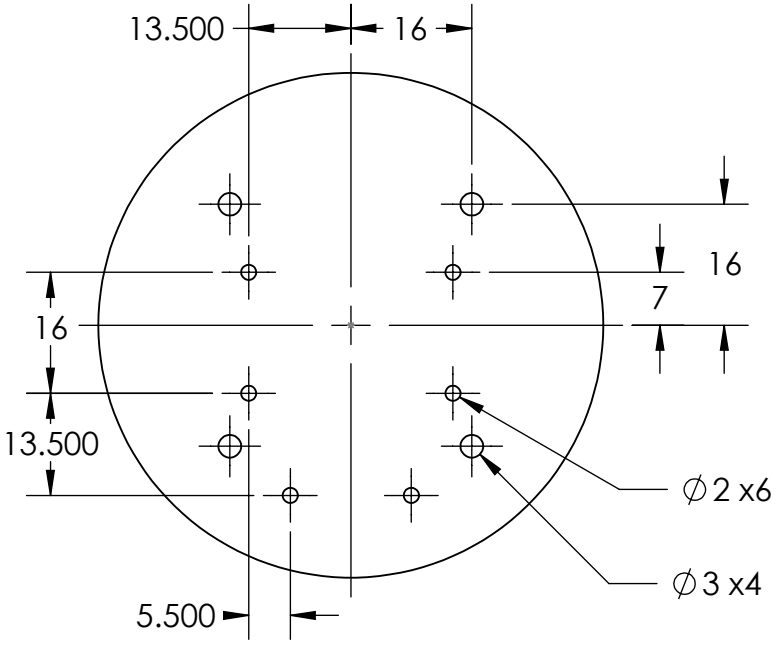
TITLE:

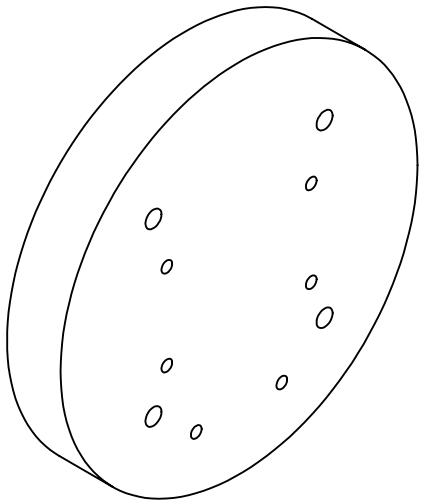
Wrist Bevel Gear (2)

SIZE	DWG. NO.	REV
A		1.0

SCALE: 2:3	WEIGHT:	SHEET 17 OF 18







UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN MILLIMETERS TOLERANCES: ±3.0	DRAWN	Ryan W.	11/11/2019
	CHECKED		
	ENG APPR.		
	MFG APPR.		
INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
MATERIAL: PLA, 20% Infill	COMMENTS:		
FINISH: N/A			
DO NOT SCALE DRAWING			

TITLE: End Effector

SIZE	DWG. NO.	REV
A		1.0

SCALE: 1:2 | WEIGHT: | SHEET 18 OF 18

A.III Salient Code

Listing 1: Actuator Dynamics MATLAB Code

```
% dynamixel motor model experiment
close all;clear;clc

% test loads mass moments of inertia
m = [.146 .088 0.108];          % mass (kg)
b = [.61277 .37227 0.28257];    % length (m)
h = [.01915 0.01915 0.0299];    % height (m)
J = (1/12)*m.*(h.^2 + b.^2);

% path to csv files relative to script
datapath = 'data/AX12A/';
files = dir(strcat(datapath,'*.csv'));
numFiles = length(files);
% initialize variables
[damp, wn, Tp] = deal(zeros(numFiles,1));

for ii = 1:numFiles
    % load experimental data, skip 5 header lines
        M = csvread(strcat(datapath, files(ii).name),5,0);
    % clean data by removing outliers
        nani = (find(diff(M(:,1)) > 100));
        M(nani,:) = [];
    % show response
         figure();
         plot(M(:,1),M(:,2))
         title('Experimental Data')
    % find % OS
        peak = max(M(:,2));                         % peak value
        peaki = find(M(:,2)==peak, 1, 'first');     % peak value index
        ss = M(end,2);                              % steady state
        os = ((peak - ss) / ss) * 100;              % % OS
    % damping ratio
        damp(ii) = -log(os/100) / sqrt(pi^2 + log(os/100)^2);
    % find where the motor begins responding
        start = M(find(diff(M(:,2)) > 1, 1, 'first'), 1);
    % time to peak
        Tp(ii) = (M(peaki,1) - start) / 1000;
    % natural frequency
        wn(ii) = pi / (sqrt(1 - damp(ii)^2)*Tp(ii));
end

sol = zeros(4,1);
```

```
% no load case, 2*zeta*omega_n
sol(1) = 2*mean(damp(end-2:end))*mean(wn(end-2:end));
% no load case, omega_n^2
sol(2) = mean(wn(end-2:end))^2;

% obtain average damping ratio and natural frequencies for load cases
zeta = [mean(damp(1:3));mean(damp(4:6));mean(damp(7:9))];
omegan = [mean(wn(1:3));mean(wn(4:6));mean(wn(7:9))];

alpha = 2.*zeta.*omegan;
beta = omegan.^2;
A = zeros(3,2);
b = zeros(3,1);

for jj = 1:3
    A(jj,:) = [1, -(alpha(jj)*J(jj) + beta(jj)*J(jj))];
    b(jj) = alpha(jj) + beta(jj);
end

sol(3:4) = A \ b;
```

Listing 2: Forward Kinematics MATLAB Function

```
function [r6,T6]= MeiosisFK(theta)

    %       Mapping between joint space and motor space
    N = 10;          %Gear Ratio
    A = [ 1/(2*N), 1/(2*N),   0, 0,   0,   0;
          1/(2*N),-1/(2*N),   0, 0,   0,   0;
                0,       0,-1/N, 0,   0,   0;
                0,       0,   0, 1,   0,   0;
                0,       0,   0, 0,-1/2, 1/2;
                0,       0,   0, 0, 1/2, 1/2];
    gamma = A*theta;

%     %Define Constants
%     LB = 12.275;
%     L1 = 0;
%     L2 = 25;
%     L3 = 20;
%     L4 = 7.2;
%     L5 = 0;
```

```
%     L6 = 5.3;

    %Relative Positions
    rBfromI = [ 0.00000000; 0.00000000; 0.00000000];
    r1fromB = [ 0.00000000; 0.00000000; 0.12275000];
    r2from1 = [ 0.00000000; 0.00000000; 0.00000000];
    r3from2 = [ 0.00000000; 0.25000000; 0.00000000];
    r4from3 = [ 0.00000000; 0.20000000; 0.00000000];
    r5from4 = [ 0.00000000; 0.07000000; 0.00000000];
    r6from5 = [ 0.00000000; 0.04750000; 0.00000000];
    %r7from6 = [0;       0;       0]; % dist. from 3rd wrist coor. frame to
       the end effector is 5.25 cm

    %Orientations wrt I:
    T1 = rotz(gamma(1));
    T2 = T1*rotx(gamma(2));
    T3 = T2*rotx(gamma(3));
    T4 = T3*roty(gamma(4));
    T5 = T4*rotx(gamma(5));
    T6 = T5*roty(gamma(6));

    %Positions wrt I:
    %rB = rBfromI;
    r1 = r1fromB;
    r2 = r1 + T1*r2from1;
    r3 = r2 + T2*r3from2;
    r4 = r3 + T3*r4from3;
    r5 = r4 + T4*r5from4;
    r6 = r5 + T5*r6from5;

end
```

Listing 3: Inverse Kinematics MATLAB Function

```
function [theta, error] = MeiosisIK(pos,R)

    eOff = [0;47.5;0];
    npos = pos - R*eOff;
    xc = npos(1);
    yc = npos(2);
    zc = npos(3);
    L1 = 122.75;
```

```
    d = 0;
    L2 = 250;
    L3 = 270;

    % Inverse Position
    if (xc^2 + yc^2 -d^2) < 0
        theta = 1000*[1;1;1;1;1;1];
        error = 1;
    else
        t1 = atan2(yc,xc) - atan2(d,sqrt(xc^2 + yc^2 -d^2)) - pi/2;
        D = (xc^2 + yc^2 - d^2 + (zc - L1)^2 - L2^2 - L3^2)/(2*L2*L3);
        t3 = atan2(-sqrt(1-D^2),D);
        t2 = atan2(zc - L1,sqrt(xc^2 + yc^2 - d^2)) - atan2(L3*sin(t3),L2 +
           L3*cos(t3));

        % Inverse Orientation
        T3 = rotz(t1)*rotx(t2)*rotx(t3);
        T = T3.'*R;
        t6 = atan2(T(2,1),-T(2,3));
        t4 = atan2(T(1,2),T(3,2));
        %t4 = atan2(sin(t4),cos(t4));

        if sin(t4) > -10e-6 && sin(t4) < 10e-6
            t5 = atan2(T(3,2)/cos(t4),T(2,2));
        else
            t5 = atan2(T(1,2)/sin(t4),T(2,2));
        end

        gamma = [t1,t2,t3,t4,t5,t6].';

        %       Mapping between joint space and motor space
        N = 10;         %Gear Ratio
%         B = [ N, N, 0, 0, 0, 0;
%               N,-N, 0, 0, 0, 0;
%               0, 0,-N, 0, 0, 0;
%               0, 0, 0, 1, 0, 0;
%               0, 0, 0, 0,-1, 1;
%               0, 0, 0, 0, 1, 1];
        A = [ 1/(2*N), 1/(2*N),   0, 0,   0,   0;
          1/(2*N),-1/(2*N),   0, 0,   0,   0;
                0,       0,-1/N, 0,   0,   0;
                0,       0,   0, 1,   0,   0;
                0,       0,   0, 0,-1/2, 1/2;
                0,       0,   0, 0, 1/2, 1/2];

```

```
        theta = A\gamma;
        error = 0;
    end
end
```

Listing 4: Closed Loop Animation

```
% Closed Loop Animation
% 11/11/2019
%   Adapted from code written by: Zack Johnson and Edward Pierce

clear all
close all
clc

dt = 0.01;                  %Integration timestep
delta_t = 4;                %Animation timestep
t = 0:dt:40;                %Simulation time array
b = zeros(12);              %Pre-allocate state variable for all time

%Define Desired Coordinates and Desired Joint angles
[xDes,yDes,zDes,gammad] = Meiosis_Name();
r6d = MeiosisFK(gammad(:,1));

%Define initial conditions
b(:,1) = [0;0;0;0;0;0;0;0;0;0;0;0];     %Initial position and velocity

%Define Control Parameters
tolerance = .001*ones(3,1);

jj = 1;

for ii = 1:(length(t)-1)

gamma = b(1:6,ii);
r6 = MeiosisFK(gamma);
if (abs(r6(3)) < .001)
    ink(:,ii) = r6;         %Keep track of where the marker writes
    draw(ii) = 1;           %Keep track of when the marker writes
else
    ink(:,ii) = [0;0;0];
    draw(ii) = 0;
```

```
end
Xe = r6 - r6d;

if (abs(Xe) <= tolerance)
    jj = jj + 1;
    if jj > length(gammad)
        break
    end
    r6d = MeiosisFK(gammad(:,jj));
end
gammadMat(:,ii) = gammad(:,jj);

k1 = Meiosis_robot1(b(:,ii),gammad(:,jj));
k2 = Meiosis_robot1(b(:,ii) + k1.*dt/2,gammad(:,jj));
k3 = Meiosis_robot1(b(:,ii) + k2.*dt/2,gammad(:,jj));
k4 = Meiosis_robot1(b(:,ii) + k3.*dt,gammad(:,jj));
b(:,ii+1) = b(:,ii) + dt*(k1./6 + k2./3 + k3./3 + k4./6);


end

% Animation
x = ink(1,:);
y = ink(2,:);
%z = r6Mat(3,:);
z = zeros(size(x));

figure('Name','Closed-Loop Animation','NumberTitle','off')
ii = 1;
for jj = 1:delta_t:29%(length(b))
%for jj = (length(b)-1):length(b)
clf(gcf)
plot3(x(1:jj),y(1:jj),z(1:jj),'r.')
hold on
Meiosis_draw2(b(1:6,jj))
F(ii) = getframe(gcf);
pause(0.001)
ii = ii + 1;
end

%      % Uncomment the following section to store the animation as a video
% writerObj = VideoWriter('DifferentialAnimation.avi');
% writerObj.FrameRate = 1/(dt*delta_t);
% open(writerObj);
%
```

```
% % write the frames to the video
% for i=1:length(F)
%     writeVideo(writerObj, F(i));
% end
%close(writerObj);

figure('Renderer', 'painters', 'Position', [10 100 700 700],'Name','Joint
    Angles vs. Time','NumberTitle','off')
Meiosis_Joint_Angle_plot(b(1:6,1:length(gammadMat)),gammadMat,t(1:length(
    gammadMat)),'Joint Angles vs. Time')
print('JointAnglePlot.pdf','-dpdf')
```

Listing 5: MEIOSIS Drawing Function

```
function Meiosis_draw2(gamma,clr)

    if exist('clr','var')
        if (clr == 'clear') | (clr == 'Clear')
            clf(gcf);
        end
    end

    Floor_v = [-.3 .4 0;...
                .3 .4 0;...
               -.3 .2 0;...
                .3 .2 0];
    Floor_f = [1 2 4 3];

    set(gcf, 'Position', [50, 50, 950, 900])
    hold on
    patch('Faces',Floor_f,'Vertices',Floor_v,'EdgeColor','None','FaceColor',
        'white');%,'FaceAlpha',.5);


    [Base,  Base_f]    = stlread('STL/Base.stl');
    [Link1, Link1_f]   = stlread('STL/Link1.stl');
    [Link2, Link2_f]   = stlread('STL/Link2new.stl');
    [Link3, Link3_f]   = stlread('STL/Link3.stl');
    [Link4, Link4_f]   = stlread('STL/Link4.stl');
    [Link5, Link5_f]   = stlread('STL/Link5.stl');
    [Link6, Link6_f]   = stlread('STL/Link6.stl');
    [Gear1, Gear1_f]   = stlread('STL/Gear1.stl');
```

```
[Gear2, Gear2_f]   = stlread('STL/Gear2.stl');
[Gear3, Gear3_f]   = stlread('STL/Gear3.stl');
[Gear4, Gear4_f]   = stlread('STL/Gear4.stl');
[Link2tube, Link2tube_f]   = stlread('STL/Link2tube.stl');
[Link3tube, Link3tube_f]   = stlread('STL/Link3tube.stl');
[Pulley1, Pulley1_f] = stlread('STL/Pulley1.stl');
[Pulley2, Pulley2_f] = stlread('STL/Pulley2.stl');
[Link4motors, Link4motors_f]   = stlread('STL/Link4motors.stl');
[Link6motor, Link6motor_f]   = stlread('STL/Link6motor.stl');
[Link2belt, Link2belt_f]   = stlread('STL/Link2belt.stl');
[Link3belt, Link3belt_f]   = stlread('STL/Link3belt.stl');

%          Forward kinematics
%Relative Positions
rBfromI =  [ 0.00000000; 0.00000000; 0.00000000];
r1fromB =  [ 0.00000000; 0.00000000; 0.12275000];
r2from1 =  [ 0.00000000; 0.00000000; 0.00000000];
r3from2 =  [ 0.00000000; 0.26000000; 0.00000000];
r4from3 =  [ 0.00000000; 0.20000000; 0.00000000];
r5from4 =  [ 0.00000000; 0.07000000; 0.00000000];
r6from5 =  [ 0.00000000; 0.09600000; 0.00000000];
rG1from1 = [ 0.09055000; 0.00000000; 0.00000000];
rG2from1 = [-0.09055000; 0.00000000; 0.00000000];
rG3from5 = [ 0.04700000; 0.00000000; 0.00000000];
rG4from5 = [-0.04700000; 0.00000000; 0.00000000];

%Orientations wrt I
TB = eye(3);
T{1} = TB*rotz(gamma(1));
T{2} = T{1}*rotx(gamma(2));
T{3} = T{2}*rotx(gamma(3));
T{4} = T{3}*roty(gamma(4));
T{5} = T{4}*rotx(gamma(5));
T{6} = T{5}*roty(gamma(6));
T{7} = T{1}*rotx(-gamma(1));        %Gear 1
T{8} = T{1}*rotx(gamma(1));         %Gear 2
T{9} = T{5}*rotx(-gamma(6));        %Gear 3
T{10} = T{5}*rotx(gamma(6));        %Gear 4

%Positions wrt I
rB = rBfromI;
r{1} = rB + TB*r1fromB;
r{2} = r{1} + T{1}*r2from1;
r{3} = r{2} + T{2}*r3from2;
r{4} = r{3} + T{3}*r4from3;
```

```
    r{5} = r{4} + T{4}*r5from4;
    r{6} = r{5} + T{5}*r6from5;
    r{7} = r{1} + T{1}*rG1from1;        %Gear 1
    r{8} = r{1} + T{1}*rG2from1;        %Gear 2
    r{9} = r{5} + T{5}*rG3from5;        %Gear 3
    r{10}= r{5} + T{5}*rG4from5;        %Gear 4

    %Transform the stl coordinates based upon FK
    Base_v   = repmat(rB,1,length(Base)) + Base';
    Link1_v  = repmat(r{1},1,length(Link1)) + T{1}*Link1';
    Link2_v  = repmat(r{2},1,length(Link2)) + T{2}*Link2';
    Link2tube_v  = repmat(r{2},1,length(Link2tube)) + T{2}*Link2tube';
    Link3_v  = repmat(r{3},1,length(Link3)) + T{3}*Link3';
    Link3tube_v  = repmat(r{3},1,length(Link3tube)) + T{3}*Link3tube';
    Link4_v  = repmat(r{4},1,length(Link4)) + T{4}*Link4';
    Link5_v  = repmat(r{5},1,length(Link5)) + T{5}*Link5';
    Link6_v  = repmat(r{6},1,length(Link6)) + T{6}*Link6';
    Gear1_v  = repmat(r{7},1,length(Gear1)) + T{7}*Gear1';
    Gear2_v  = repmat(r{8},1,length(Gear2)) + T{8}*Gear2';
    Gear3_v  = repmat(r{9},1,length(Gear3)) + T{9}*Gear3';
    Gear4_v  = repmat(r{10},1,length(Gear4)) + T{10}*Gear4';
    Pulley1_v  = repmat(r{7},1,length(Pulley1)) + T{7}*Pulley1';
    Pulley2_v  = repmat(r{8},1,length(Pulley2)) + T{8}*Pulley2';
    Link4motors_v  = repmat(r{4},1,length(Link4motors)) + T{4}*Link4motors';
    Link6motor_v  = repmat(r{6},1,length(Link6motor)) + T{6}*Link6motor';
    Link2belt_v  = repmat(r{2},1,length(Link2belt)) + T{2}*Link2belt';
    Link3belt_v  = repmat(r{3},1,length(Link3belt)) + T{3}*Link3belt';

    patch('Faces',Base_f, 'Vertices',Base_v', 'EdgeColor','None','FaceColor'
       ,[0 0.082353 1]);
    patch('Faces',Link1_f,'Vertices',Link1_v','EdgeColor','None','FaceColor'
       ,[0.901961 0.756863 0.035294]);
    patch('Faces',Link2_f,'Vertices',Link2_v','EdgeColor','None','FaceColor'
       ,[0 0.082353 1]);
    patch('Faces',Link2tube_f,'Vertices',Link2tube_v','EdgeColor','None','
       FaceColor',[0.5774    0.5774    0.5774]);
    patch('Faces',Link3_f,'Vertices',Link3_v','EdgeColor','None','FaceColor'
       ,[0.901961 0.756863 0.035294]);
    patch('Faces',Link3tube_f,'Vertices',Link3tube_v','EdgeColor','None','
       FaceColor',[0.5774    0.5774    0.5774]);
    patch('Faces',Link4_f,'Vertices',Link4_v','EdgeColor','None','FaceColor'
       ,[0 0.082353 1]);
    patch('Faces',Link5_f,'Vertices',Link5_v','EdgeColor','None','FaceColor'
       ,[0.901961 0.756863 0.035294]);
    patch('Faces',Link6_f,'Vertices',Link6_v','EdgeColor','None','FaceColor'
```

```
        ,[1 0.082353 0.082353]);
    patch('Faces',Gear1_f,'Vertices',Gear1_v','EdgeColor','None','FaceColor'
        ,[1 0.082353 0.082353]);
    patch('Faces',Gear2_f,'Vertices',Gear2_v','EdgeColor','None','FaceColor'
        ,[1 0.082353 0.082353]);
    patch('Faces',Gear3_f,'Vertices',Gear3_v','EdgeColor','None','FaceColor'
        ,[1 0.082353 0.082353]);
    patch('Faces',Gear4_f,'Vertices',Gear4_v','EdgeColor','None','FaceColor'
        ,[1 0.082353 0.082353]);
    patch('Faces',Pulley1_f,'Vertices',Pulley1_v','EdgeColor','None','
        FaceColor',[0.5774    0.5774    0.5774]);
    patch('Faces',Pulley2_f,'Vertices',Pulley2_v','EdgeColor','None','
        FaceColor',[0.5774    0.5774    0.5774]);
    patch('Faces',Link4motors_f,'Vertices',Link4motors_v','EdgeColor','None'
        ,'FaceColor',[0.5774    0.5774    0.5774]);
    patch('Faces',Link6motor_f,'Vertices',Link6motor_v','EdgeColor','None','
        FaceColor',[0.5774    0.5774    0.5774]);
    patch('Faces',Link2belt_f,'Vertices',Link2belt_v','EdgeColor','None','
        FaceColor',[0 0 0]);
    patch('Faces',Link3belt_f,'Vertices',Link3belt_v','EdgeColor','None','
        FaceColor',[0 0 0]);

    axis equal
    camlight(180,90)
    set(gca,'xtick',[])
    set(gca,'xticklabel',[])
    set(gca,'XColor','none')
    set(gca,'ytick',[])
    set(gca,'yticklabel',[])
    set(gca,'YColor','none')
    set(gca,'ztick',[])
    set(gca,'zticklabel',[])
    set(gca,'ZColor','none')
    set(gca,'Color','none')

    set(gca,'projection','perspective')
    view(140,45)
    axis([-0.4 0.4 -0.2 0.7 0 0.4])
    %axis([-.1 .1 -.1 .1 0 .2])

    hold off
end
```

Listing 6: H, D, and G Calculations

```
function [H, d, G] = Meiosis_HdG(gamma,gammadot)

    %       Mass Parameters
    %Mass(kg)
    m{1} = 0.13954848;      %Link 1
    m{2} = 1.14358921;      %Link 2
    m{3} = 1.00606970;      %Link 3
    m{4} = 0.26834265;      %Link 4
    m{5} = 0.03933179;      %Link 5
    m{6} = 0.19893937;
    m{7} = 0.24224704;      %Gear 1
    m{8} = 0.24224704;      %Gear 2
    m{9} = 0.05035795;      %Gear 3
    m{10} = 0.05035795;

    %Center of mass for each link (m)
    rcm{1} = [ 0.00000000; 0.00000000;-0.02792483];     %Link 1
    rcm{2} = [ 0.00007758; 0.15623908;-0.00011610];     %Link 2
    rcm{3} = [-0.01528806; 0.07006082; 0.00110335];     %Link 3
    rcm{4} = [ 0.00000000; 0.03163485; 0.00000000];     %Link 4
    rcm{5} = [ 0.00000000; 0.00630115; 0.00000000];     %Link 5
    rcm{6} = [ 0.00000000;-0.03626349;-0.00034373];     %Link 6
    rcm{7} = [-0.02271539;-0.00001560; 0.00001565];     %Gear 1
    rcm{8} = [ 0.02271539;-0.00001560;-0.00001565];     %Gear 2
    rcm{9} = [-0.00678010; 0.00000000; 0.00000000];     %Gear 3
    rcm{10}= [ 0.00678010; 0.00000000; 0.00000000];     %Gear 4

    Gam{1} = rcm{1}*m{1};
    Gam{2} = rcm{2}*m{2};
    Gam{3} = rcm{3}*m{3};
    Gam{4} = rcm{4}*m{4};
    Gam{5} = rcm{5}*m{5};
    Gam{6} = rcm{6}*m{6};
    Gam{7} = rcm{7}*m{7};
    Gam{8} = rcm{8}*m{8};
    Gam{9} = rcm{9}*m{9};
    Gam{10} = rcm{10}*m{10};

    %Inertia Matrices
        J{1} = [ 0.00015462  0.00000000  0.00000000;        %Link 1
                 0.00000000      0.00016692  0.00000000;
                 0.00000000      0.00000000  0.00003819];
    J{2} = [ 0.03894999 -0.00002305  0.00000000;        %Link 2
```

```
        −0.00002305  0.00589287  0.00001585;
         0.00000000  0.00001585  0.04414151];
J{3} = [ 0.01143098  0.00169728  0.00000008;        %Link 3
             0.00169728      0.00185484 −0.00021794;
             0.00000008     −0.00021794  0.01259229];
J{4} = [ 0.00063966  0.00000000  0.00000000;        %Link 4
             0.00000000      0.00069715  0.00000000;
             0.00000000      0.00000000  0.00108401];
J{5} = [ 0.00000851  0.00000000  0.00000000;        %Link 5
             0.00000000      0.00001467  0.00000000;
             0.00000000      0.00000000  0.00002081];
J{6} = [ 0.00036378  0.00000000  0.00000000;        %Link 6
             0.00000000      0.00007212 −0.00000173;
             0.00000000     −0.00000173  0.00034916];
J{7} = [ 0.00014269 −0.00000005  0.00000005;        %Gear 1
            −0.00000005      0.00029726  0.00000000;
             0.00000005      0.00000000  0.00029726];
J{8} = [ 0.00014269  0.00000005  0.00000005;        %Gear 2
             0.00000005      0.00029726  0.00000000;
             0.00000005      0.00000000  0.00029726];
J{9} = [ 0.00003333  0.00000000  0.00000000;        %Gear 3
             0.00000000      0.00001986  0.00000000;
             0.00000000      0.00000000  0.00001986];
J{10}= [ 0.00003333  0.00000000  0.00000000;        %Gear 4
             0.00000000      0.00001986  0.00000000;
             0.00000000      0.00000000  0.00001986];


%          Forward kinematics
%Relative Positions
rBfromI =  [ 0.00000000; 0.00000000; 0.00000000];
r1fromB =  [ 0.00000000; 0.00000000; 0.12275000];
r2from1 =  [ 0.00000000; 0.00000000; 0.00000000];
r3from2 =  [ 0.00000000; 0.26000000; 0.00000000];
r4from3 =  [ 0.00000000; 0.20000000; 0.00000000];
r5from4 =  [ 0.00000000; 0.07000000; 0.00000000];
r6from5 =  [ 0.00000000; 0.09600000; 0.00000000];
rG1from1 = [ 0.09055000; 0.00000000; 0.00000000];
rG2from1 = [−0.09055000; 0.00000000; 0.00000000];
rG3from5 = [ 0.04700000; 0.00000000; 0.00000000];
rG4from5 = [−0.04700000; 0.00000000; 0.00000000];

%Store orientations in cell array
rn{1} = r1fromB;
rn{2} = r2from1;
```

```
rn{3} = r3from2;
rn{4} = r4from3;
rn{5} = r5from4;
rn{6} = r6from5;
rn{7} = rG1from1;
rn{8} = rG2from1;
rn{9} = rG3from5;
rn{10} = rG4from5;

%Orientations wrt I
TB = eye(3);
T{1} = TB*rotz(gamma(1));
T{2} = T{1}*rotx(gamma(2));
T{3} = T{2}*rotx(gamma(3));
T{4} = T{3}*roty(gamma(4));
T{5} = T{4}*rotx(gamma(5));
T{6} = T{5}*roty(gamma(6));
T{7} = T{1}*rotx(gamma(1));           %Gear 1
T{8} = T{1}*rotx(-gamma(1));          %Gear 2
T{9} = T{5}*rotx(-gamma(6));          %Gear 3
T{10} = T{5}*rotx(gamma(6));          %Gear 4

%Positions wrt I
rB = rBfromI;
r{1} = rB + TB*r1fromB;
r{2} = r{1} + T{1}*r2from1;
r{3} = r{2} + T{2}*r3from2;
r{4} = r{3} + T{3}*r4from3;
r{5} = r{4} + T{4}*r5from4;
r{6} = r{5} + T{5}*r6from5;
r{7} = r{1} + T{1}*rG1from1;        %Gear 1
r{8} = r{1} + T{1}*rG2from1;        %Gear 2
r{9} = r{5} + T{5}*rG3from5;        %Gear 3
r{10}= r{5} + T{5}*rG4from5;        %Gear 4


%              Recursive Kinematics
Jb0 = zeros(6);
Jbdot0 = zeros(6);

for ii = 1:10
    if ii == 1
        Jmat = Jb0;
        wI = Jmat(1:3,:)*gammadot;
        [Jb{ii},Jbdot{ii}] = Next_GeoJac_Meiosis(Jb0,Jbdot0,gamma,
```

```
            gammadot,TB,rn{ii},zeros(3,1),wI,ii);
134         elseif ii == 7
135             Jmat = Jb{1};
136             wI = Jmat(1:3,:)*gammadot;
137             [Jb{ii},Jbdot{ii}] = Next_GeoJac_Meiosis(Jb{1},Jbdot{1},gamma,
                    gammadot,T{1},rn{ii},zeros(3,1),wI,ii);
138         elseif ii == 8
139             Jmat = Jb{1};
140             wI = Jmat(1:3,:)*gammadot;
141             [Jb{ii},Jbdot{ii}] = Next_GeoJac_Meiosis(Jb{1},Jbdot{1},gamma,
                    gammadot,T{1},rn{ii},zeros(3,1),wI,ii);
142         elseif ii == 9
143             Jmat = Jb{5};
144             wI = Jmat(1:3,:)*gammadot;
145             [Jb{ii},Jbdot{ii}] = Next_GeoJac_Meiosis(Jb{1},Jbdot{5},gamma,
                    gammadot,T{5},rn{ii},zeros(3,1),wI,ii);
146         elseif ii == 10
147             Jmat = Jb{5};
148             wI = Jmat(1:3,:)*gammadot;
149             [Jb{ii},Jbdot{ii}] = Next_GeoJac_Meiosis(Jb{1},Jbdot{5},gamma,
                    gammadot,T{5},rn{ii},zeros(3,1),wI,ii);
150         else
151             Jmat = Jb{ii-1};
152             wI = Jmat(1:3,:)*gammadot;
153             [Jb{ii},Jbdot{ii}] = Next_GeoJac_Meiosis(Jb{ii-1},Jbdot{ii-1},
                    gamma,gammadot,T{ii-1},rn{ii},zeros(3,1),wI,ii);
154         end
155
156         Jmat = Jb{ii};
157         wI = Jmat(1:3,:)*gammadot;
158
159         Hmat = Jb{ii}.'*[J{ii},skew(Gam{ii})*T{ii}.';T{ii}*skew(Gam{ii}).',m
                {ii}*eye(3)]*Jb{ii};
160         dvec = Jb{ii}.'*[J{ii},skew(Gam{ii})*T{ii}.';T{ii}*skew(Gam{ii}).',m
                {ii}*eye(3)]*Jbdot{ii}*gammadot + Jb{ii}.'*[cross(wI,J{ii}*wI);T{
                ii}*cross(wI,cross(wI,Gam{ii}))];
161         g = Jb{ii}.'*[cross(rcm{ii},T{ii}.'*[0;0;-m{ii}*9.81]);[0;0;-m{ii
                }*9.81]];
162
163         if ii == 1
164             H = Hmat;
165             d = dvec;
166             G = g;
167         else
168             H = H + Hmat;
```

```
            d = d + dvec;
            G = G + g;
        end
    end

end
```

Listing 7: Joint Angle Plotting Code

```
function Meiosis_Joint_Angle_plot(b,gammad,t,Title)

    % Plot joint angle 1
    subplot(4,2,1)
    hold on
    plot(t, b(1,:),'b');
    plot(t, gammad(1,:),'r--')
    title('First Joint Angle vs. Time');
    xlabel('Time (s)');
    ylabel('\theta_1 (rad)');
    legend('\theta','\theta_d')

    % Plot joint angle 2
    subplot(4,2,2)
    hold on
    plot(t, b(2,:),'b')
    plot(t, gammad(2,:),'r--')
    title('Second Joint Angle vs. Time');
    xlabel('Time (s)');
    ylabel('\theta_2 (rad)');
    legend('\theta','\theta_d','Location','Southeast')

    % Plot joint 3 movment
    subplot(4,2,3)
    hold on
    plot(t, b(3,:),'b')
    plot(t, gammad(3,:),'r--')
    title('Third Joint Angle vs. Time');
    xlabel('Time (s)');
    ylabel('\theta_3 (rad)');
    legend('\theta','\theta_d')

    % Plot joint 4 movment
```

```
    subplot(4,2,4)
    hold on
    plot(t, b(4,:),'b')
    plot(t, gammad(4,:),'r--')
    title('Fourth Joint Angle vs. Time');
    xlabel('Time (s)');
    ylabel('\theta_4 (rad)');
    legend('\theta','\theta_d','Location','Southeast')

    % Plot joint 5 movment
    subplot(4,2,5)
    hold on
    plot(t, b(5,:),'b')
    plot(t, gammad(5,:),'r--')
    title('Fifth Joint Angle vs. Time');
    xlabel('Time (s)');
    ylabel('\theta_5 (rad)');
    legend('\theta','\theta_d','Location','Southeast')

    % Plot joint 6 movment
    subplot(4,2,6)
    hold on
    plot(t, b(6,:),'b')
    plot(t, gammad(6,:),'r--')
    title('Sixth Joint Angle vs. Time');
    xlabel('Time (s)');
    ylabel('\theta_6 (rad)');
    legend('\theta','\theta_d','Location','Southeast')


%     annotation('textbox', [0 0.9 1 0.1], ...
%     'String', Title, ...
%     'EdgeColor', 'none', ...
%     'HorizontalAlignment', 'center')

end
```

Listing 8: MEIOSIS Name Plotting

```
function [xDes,yDes,zDes,thetad] = Meiosis_Name2()

    %Define Desired Workspace Coordinates
```

```
4       [xM,yM,zM] = LetterM();
5       [xE,yE,zE] = LetterE();
6       [xI1,yI1,zI1] = LetterI1();
7       [xO,yO,zO] = LetterO();
8       [xS1,yS1,zS1] = LetterS1();
9       [xI2,yI2,zI2] = LetterI2();
10      [xS2,yS2,zS2] = LetterS2();
11
12      xDes = [xM,xE,xI1,xO,xS1,xI2,xS2];
13      yDes = [yM,yE,yI1,yO,yS1,yI2,yS2];
14      zDes = [zM,zE,zI1,zO,zS1,zI2,zS2];
15
16      R = rotx(-pi/2);
17
18
19      %Calculate Joint Angles
20      for ii = 1:length(xDes)
21          [thetad(:,ii),errorCheck] = MeiosisIK([xDes(ii);yDes(ii);zDes(ii)],R
                );
22          if errorCheck == 1
23              break
24          end
25      end
26  end
27
28  function [x,y,z] = LetterM()
29
30      %Define Desired Workspace Coordinates
31      x = [ 270, 270, 240, ];
32      y = [ 290, 290, 250, ];
33      z = [  10,   0, ];
34
35  end
36
37  function [x,y,z] = LetterE()
38
39      %Define Desired Workspace Coordinates
40      x = [210, 210, 210, 150, 150, 200, 200, 170, 170, 200, 200, 150, 150,
            210, 210];
41      y = [260, 260, 340, 340, 330, 330, 300, 300, 290, 290, 270, 270, 260,
            260, 260];
42      z = [ 10,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
            0,  10];
43
44  end
```

```
45
46 function [x,y,z] = LetterI1()
47
48     %Define Desired Workspace Coordinates
49     x = [130, 130, 130, 110, 110, 130, 130,  80,  80, 100, 100,  80,  80,
           130, 130];
50     y = [260, 260, 270, 270, 330, 330, 340, 340, 330, 330, 270, 270, 260,
           260, 260];
51     z = [ 10,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
           0,  10];
52
53 end
54
55 function [x,y,z] = LetterO()
56
57     %Define Desired Workspace Coordinates
58     x = [ 60,  60,  60,   0,   0,  60,  60,  50,  50,  50,  10,  10,  50,
           50];
59     y = [260, 260, 340, 340, 260, 260, 260, 270, 270, 330, 330, 270, 270,
           270];
60     z = [ 10,   0,   0,   0,   0,   0,  10,  10,   0,   0,   0,   0,   0,
           10];
61
62 end
63
64 function [x,y,z] = LetterS1()
65
66     %Define Desired Workspace Coordinates
67     x = [-20, -20, -20, -50, -50, -20, -20, -60, -60, -30, -30, -60, -60,
           -20, -20];
68     y = [260, 260, 300, 300, 330, 330, 340, 340, 290, 290, 270, 270, 260,
           260, 260];
69     z = [ 10,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
           0,  10];
70
71 end
72
73 function [x,y,z] = LetterI2()
74
75     %Define Desired Workspace Coordinates
76     [x,y,z] = LetterI1();
77     x = x - 210;
78
79 end
80
```

```
function [x,y,z] = LetterS2()

    %Define Desired Workspace Coordinates
    [x,y,z] = LetterS1();
    x = x - 130;

end
```

Listing 9: MEIOSIS Robot State Calculator

```
function bdot = Meiosis_robot1(b,gammad)

    %Preallocate variables
    bdot = zeros(12,1);

    %Variable reassignments
    gamma = b(1:6);
    gammadot = b(7:12);

    [H,d,G] = Meiosis_HdG(gamma,gammadot);
    C1 = diag((1 / 82.5459264810444)*[1 1 1 1 1 1]);
    C2 = diag((13.4787019763844 / 82.5459264810444)*[1 1 1 1 1 1]);
    C3 = diag((88.3252169803572 / 82.5459264810444)*[1 1 1 1 1 1]);

    N = 10;  %Gear Ratio
    A = [ 1/(2*N), 1/(2*N),   0, 0,   0,   0;
          1/(2*N),-1/(2*N),   0, 0,   0,   0;
                0,       0,-1/N, 0,   0,   0;
                0,       0,   0, 1,   0,   0;
                0,       0,   0, 0,-1/2, 1/2;
                0,       0,   0, 0, 1/2, 1/2];

    bdot(1:6) = b(7:12);
    u = (A.'\C3)\(G + d) + A\gammad;
    bdot(7:12) = (H + A.'\C1/A)\(-d - G - A\C2/A*gammadot - A.'\C3/A*gamma +
        A.'\C3*u);

end
```

Listing 10: MEIOSIS Velocity Calculator

```
function [dotr,w] = MeiosisVelocity(T,r,gamma,dotgamma)

    dotr = simplify(jacobian(r,gamma)*dotgamma);

    Jw = [T(:,3).'*jacobian(T(:,2),gamma);...
          T(:,1).'*jacobian(T(:,3),gamma);...
          T(:,2).'*jacobian(T(:,1),gamma)];

    M = simplify(Jw,'steps',200,'criterion','preferReal');

    w = simplify(M*dotgamma);

end
```

Listing 11: Geometric Jacobian Calculator

```
function [JbN,JbNdot] = Next_GeoJac_Meiosis(Jb,Jbdot,gamma,gammadot,TI,rn,
   rndot,wI,link)

    Ihat = zeros(3,length(gamma));
    Itilda = zeros(3,length(gamma));

    %link = 8 corresponds to the screen
    %
    switch link
        case 1
            Ihat(3,1) = 1;
            Tn = rotz(gamma(1));
        case 2
            Ihat(1,2) = 1;
            Tn = rotx(gamma(2));
        case 3
            Ihat(1,3) = 1;
            Tn = rotx(gamma(3));
        case 4
            Ihat(2,4) = 1;
            Tn = roty(gamma(4));
        case 5
            Ihat(1,5) = 1;
            Tn = rotx(gamma(5));
        case 6
```

```
            Ihat(2,6) = 1;
            Tn = roty(gamma(6));
        case 7                      %Gear 1
            Ihat(1,1) = 1;
            Tn = rotx(gamma(1));
        case 8                      %Gear 2
            Ihat(1,1) = 1;
            Tn = rotx(-gamma(1));
        case 9                      %Gear 3
            Ihat(1,1) = 1;
            Tn = rotx(-gamma(6));
        case 10                     %Gear 4
            Ihat(1,1) = 1;
            Tn = rotx(gamma(6));

    end

    %GeoJac Matrices
    mat1 = [Tn.',zeros(3);-TI*skew(rn),eye(3)];
    mat2 = [Ihat;TI*Itilda];

    JbN = mat1*Jb + mat2;

    %GeoJacdot matrices
    mat1 = [          -skew(Ihat*gammadot)*(Tn.'), zeros(3);
            -TI*skew(wI)*skew(rn) - TI*skew(rndot), zeros(3)];
    mat2 = [        Tn.', zeros(3);
            -TI*skew(rn), eye(3)];
    mat3 = [           zeros(3,6);
            TI*skew(wI)*Itilda];

    JbNdot = mat1*Jb + mat2*Jbdot + mat3;

end
```

Listing 12: Open-Loop Animation Code

```
% Open Loop Animation
% 11/11/2019
%   Adapted from code written by: Zack Johnson and Edward Pierce

clear all
```

```
close all
clc

dt = 0.001;                     %Integration timestep
fr = 24;                        %Animation Framerate
delta_t = round(1/(dt*fr));     %Animation timestep
t = 0:dt:30;                    %simulation time array
b = zeros(12,length(t));        %pre-allocate motor angles for all time
Va = zeros(6,length(t));        %pre-allocate input torque for all time

%Define initial conditions
b(:,1) = [0;0;0;0;0;0;0;0;0;0;0;0];     %Initial position and velocity

for ii = 1:(length(t)-1)
k1 = Meiosis_robot1(b(:,ii),Va(:,ii));
k2 = Meiosis_robot1(b(:,ii) + k1.*dt/2,Va(:,ii));
k3 = Meiosis_robot1(b(:,ii) + k2.*dt/2,Va(:,ii));
k4 = Meiosis_robot1(b(:,ii) + k3.*dt,Va(:,ii));
b(:,ii+1) = b(:,ii) + dt*(k1./6 + k2./3 + k3./3 + k4./6);
ii/length(t)
end

% for ii = delta_t:delta_t:(length(b))
jj = 1;
for ii = 1:delta_t:length(b)
Meiosis_draw2(b(1:7,ii),'clear',t(ii))
F(jj) = getframe(gcf);
jj = jj + 1;
end

 % Uncomment the following section to store the animation as a video
writerObj = VideoWriter('OpenLoopAnimation.avi');
writerObj.FrameRate = fr;
open(writerObj);

% write the frames to the video
for i=1:length(F)
writeVideo(writerObj, F(i));
end
close(writerObj);

figure('Renderer', 'painters', 'Position', [10 100 700 700],'Name','Joint
   Angles vs. Time','NumberTitle','off')
Meiosis_Joint_Angle_plot(b(1:6,:),t(1:length(b)),'Joint Angles vs. Time')
print('JointAnglePlot.pdf','-dpdf')
```

Listing 13: Velocity Kinematics Code

```
clear all
close all
clc

%Define Symbolic Variables
syms t1 t2 t3 t4 t5 t6 dt1 dt2 dt3 dt4 dt5 dt6 L1 L2 L3 L4 L5 L6
q = [t1 t2 t3 t4 t5 t6].';
dotq = [dt1 dt2 dt3 dt4 dt5 dt6].';

%Relative Positions
rBfromI = [  0;  0;  0];
r1fromB = [  0;  0; L1];    % Frame 1 is 122.75mm away from the Base frame
   in the Z direction
r2from1 = [  0;  0;  0];    % Frame 2 is centered on Frame 1
r3from2 = [  0; L2;  0];    % Frame 3 is 250mm away from Frame 2 in the Y
   direction
r4from3 = [  0; L3;  0];    % Frame 4 is 200mm away from Frame 3 in the Y
   direction
r5from4 = [  0; L4;  0];    % Frame 5 is 70mm away from Frame 4 in the Y
   direction
r6from5 = [  0; L6;  0];    % Frame 6 is 47.5mm away from Frame 5 in the Y
   direction

%Orientations wrt I:
T(:,:,1) = rotz(t1);
T(:,:,2) = T(:,:,1)*rotx(t2);
T(:,:,3) = T(:,:,2)*rotx(t3);
T(:,:,4) = T(:,:,3)*roty(t4);
T(:,:,5) = T(:,:,4)*rotx(t5);
T(:,:,6) = T(:,:,5)*roty(t6);

%Positions wrt I:
rB = rBfromI;
r(:,1) = rB + r1fromB;
r(:,2) = r(:,1) + T(:,:,1)*r2from1;
r(:,3) = r(:,2) + T(:,:,2)*r3from2;
r(:,4) = r(:,3) + T(:,:,3)*r4from3;
r(:,5) = r(:,4) + T(:,:,4)*r5from4;
```

```
r(:,6) = r(:,5) + T(:,:,5)*r6from5;


for ii = 1:6
    [dotr(:,ii),w(:,ii)] = MeiosisVelocity(T(:,:,ii),r(:,ii),q,dotq);
    disp(ii)
    disp(dotr(:,ii))
    disp(w(:,ii))

end
```