Augmented Robotic Manipulator (A.R.M.)

|  |  |
| --- | --- |
| Branson Elliott | William Prody |
| *Department of Electrical Engineering* | *Department of Electrical Engineering* |
| *University of North Dakota* | *University of North Dakota* |
| *Grand Forks, ND 58202* | *Grand Forks, ND, 58202* |
| branson.p.elliott@und.edu | william.prody@und.edu |

|  |
| --- |
| Garrett Tjernagel |
| *Department of Electrical Engineering* |
| *University of North Dakota* |
| *Grand Forks, ND 58202* |
| garrett.tjernagel@und.edu |

*ABSTRACT*

The Augmented Robotic Manipulator is designed to mirror the movements of a human arm. Researching similar implementations of this idea revealed a few points that could be improved upon. The ARM and controller transmit data wirelessly, they both are self contained systems, and the response time of the system is as close to real time as possible. This will be achieved using an arm brace that captures human input movement data using various sensors like gyroscopes, potentiometers, and flex sensors located at key areas on the brace. With a gyroscope on the shoulder and flat of the hand, flex sensors on the fingers, and a potentiometer on the elbow, six degrees of freedom can be measured. This data will be filtered and packaged into a data array to be sent to the arm wirelessly via a radio transmitter. The ARM captures its current positional data using similar sensors, receives the target position data pack from the controller, and calculates differences between the two to be used in a simplified PID loop to move the ARM to the target position. The physical ARM and controller are modeled in Fusion360 and 3D printed using PLA FFF printing methods.

*INDEX TERMS* Robotic, wireless, sensors, gyroscopes, PID, data, Arduino, Kinematics, Control Flow, Circuit Design, 3D Modeling, 3D Printing

1. *INTRODUCTION*

Continued improvement in the field of robotics leads to ever more impressive engineering and design. Improvements to the speed, size, and access to technology allows more and more people to innovate. This paper describes the work and research done to create a robotic arm controlled by gyroscopic data sampling. To achieve high speed while maintaining acceptable levels of accuracy and precision, development of kinematics, modeling, control systems, and power systems are all required to provide an end user with as seamless of an experience controlling a robotic arm with as little training as possible.

1. *METHODS AND THEORY*

In this section we will describe the process taken in designing the mirror movement robotic arm using. The order in which the project design was done followed the process described by Puig et al.

1. **Problem Definition**

When it comes to the area of robotic modeling based on human functions, one of the more difficult parts to imitate is the arm and hand. There has been some extensive research done in this area over recent years, however, it seems that there haven’t been any studies that tried to replicate a human arm from shoulder to fingertips. Our group decided to take on this task and try to replicate the function of a human arm to be as accurate and responsive as possible. The applications for the use of this controlled arm could be in environments that are too dangerous for humans, or advancement in healthcare.

1. **Concept Design**

The beginning stage for designing the arm started with discussing where rotation and how many degrees of freedom (DOF) were needed. According to H. Kim et al., human arm motion is redundant in the fact that it contains 7 DOF, however it only requires 6 DOF to position the wrist and orient the palm in 3D space. This became a driving force in our decision to design for 6 DOF arm. We also aimed to create the most realistic feel possible so the arm will be mounted on a vertical pedestal to represent the arm attached to a body torso.

The joint motion of the arm will be driven through a series of rotational actuators. The shoulder joint will sit atop the pedestal and contain two degrees of freedom. The combination of these two degrees of freedom working simultaneously will create a third DOF. At the elbow joint, there will be a single rotating motion to create the curl of the forearm up to the bicep and vice versa. We will apply a forearm rotating motion to supplement for wrist rotation and palm orientation. Based on the anatomy of the human arm, when the wrist is rotated, the forearm rotates likewise. This concept again applies a realistic feel to the model, and adds a fifth DOF. The final degree of freedom will come at the combination of the wrist and base of the hand. The motion will represent the pitch of the wrist, giving it the freedom to move up and down.

When it came to the hand a few ideas were thrown around especially in regards to thumb movement. For the finger actuation, there was the possibility to have motors for rotating each finger joint, however there would not be enough room nor money to implement this. Cable-driven actuation was chosen for the fingers instead. The thumb was originally going to be designed using a ball joint in the hand and a motor to rotate the joint. Instead the adduction and abduction of the thumb was chosen to be imitated using a sweeping motion along with cable driven actuation.

1. *RELATED RESEARCH*

So far, the majority of robotic manipulators have been only small sections of the body like the hand. In regards to the hand, there are multiple robotic hands that have been made but most of them used tripod type grasping. The other robotic arms that were reviewed seemed to have a fairly slow response time to input. Ideally, the response time for this arm will be as close to zero as possible. Other arms also use wired connections to control the robotic arm, and this project aims to be able to control the arm wirelessly. Along with this comes a unique feature to this arm in which the end goal is for any person to be able to slip on the controlling gauntlet and have a natural feel to the control.

For many of the naturally controlled robotic arms, meaning those controlled by mimicking human movement, many were large and bulky when it came to design and operation. L. Chen et al describe ways to implement AI enabled wearable manipulator controllers. The group plans to use some of the ideas here focusing mainly on the wearable side of things with less emphasis on the AI as that would be harder to implement, despite the merit and enhancements that the AI could provide. S. Alvarez-Rodriguez and F. G. Peña Lecona discuss using an nth degree order system of sensors to control an nth-Degrees of Freedom (DOF) arm. The group found some of the ideas in their work especially valuable since we are using 3 main sensors, 4 feedback sensors, as well as 5 auxiliary sensors to control our system, not to mention all wirelessly to enable freedom of movement for the user. W. G. Hao et al also talk about control of a 6-DOF arm with efficient trajectory planning and speed control, which can be implemented into the groups system effectively as the group seeks to implement a finely controlled arm as efficiently as possible. Building again off the wireless nature of the groups design, N. RNaveen specifically talks about Arduino communication protocols, of which are specifically useful for the design of the ARM.

1. *EQUATIONS*

In order to find the kinematics of the hand, the transformation matrix first had to be found from the DH table. The DH table is found by analyzing the joints of the arm and hand. The number of joints determines the number of steps needed to go from the first joint to the last. Each DH table has a spot for rotation in Z(theta), translation in z(d), rotation in x(alpha), and translation in x(a), along with the variable. An example of the table is shown in Table 1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Order |  | d | a |  | Var. |
| 0-1 | 0 | d1 | 0 | 0 | 1 |
| 1-2 | 15 dgs | 0 | d2 | 0 | 2 |
| 2-3 | 0 | d3 | 0 | 0 | 3 |

Table 1 - DH Table Format

After the DH table is calculated the homogeneous transformation matrix can be found. This is done by multiplying the different operations in each row of the DH table together. Equation (1) shows the transformation matrix, , for the first row of a DH table.

|  |  |  |
| --- | --- | --- |
|  |  |  |

When the transformation matrix for each step was found, the overall transformation matrix was the product of each step multiplied in order. So for a DH table with three steps, the overall transform would be,

|  |  |  |
| --- | --- | --- |
|  |  |  |

Using the overall transformation and the transformation for each step, the jacobian can be found to describe the forward and inverse motion of the system. The jacobian of a system is found with the equation,

|  |  |  |
| --- | --- | --- |
|  |  |  |

Where,

|  |  |  |
| --- | --- | --- |
|  |  |  |

and,

|  |  |  |
| --- | --- | --- |
|  |  |  |

for revolute joints.

1. *DESIGN*
2. **Hand**
3. Control

The control of the hand is fairly simple. First, the finger actuation motion is controlled by steel and rubber wires that act like tendons in the hand. Each finger has three joints with the steel cable and rubber cable tied off at the tip of the finger. The rubber cable for each finger will be tied off at the wrist joint when the hand is in its resting position. The steel cable for each finger is tied to a servo motor in the forearm of the robot. When motion from the control glove is input, the servo motor will turn and pull on the steel cable which will bend the joints of the fingers. When the controlling finger is moved back to its rest position, the tension in the rubber cable pulls the robotic finger back to the rest position. The fingers will not have any side to side motion and therefore will just have one degree of freedom to bend down. The thumb is different from the other fingers due to the extra degree of freedom in the thumb. With the sweeping motion, the control glove will register the side to side inputs of the thumb for the sweeping motion in the robotic thumb. As far as the bending motion, all the fingers have one degree of freedom for curling the fingers down.

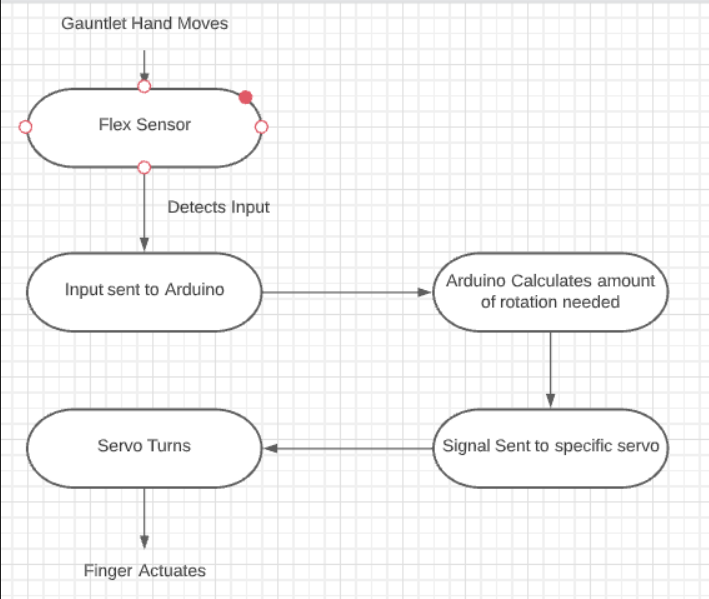


Figure 1 - Control Flow Diagram

1. Materials

The hand itself will be 3-D printed from the CAD model which will require printing material. For the actuation of the fingers both fishing line and rubber cable is needed for the hand. Finally, a 9mm bolt is needed for the thumb sweeping as well as the wrist. There will need to be 5 servo motors for the hand, one for each finger, and 1 for the wrist.

1. Modeling

When it came to modeling the hand, the group decided to make it as close to the same size as a regular human hand as possible. L. Tian et al. offers the idea to measure the length of each “link” on a human hand which is what was done to scale the 3-D model (n. pag.). The hand model was created using tinkercad and will be converted to an STL file when the hand is printed. H. Mnyusiwalla et al. points out that, when modeling a tendon based actuating hand, the designer needs to be careful of friction with the tendons. They go on to advise designers to minimize sliding surfaces, using pulleys, and making the tendon paths as straight as possible (p. 812). In implementation the straightest paths from the finger to the wrist were made and guides were inserted for the steel and rubber cables. There was not enough room in the fingers and hand for pulleys however, and the least number of guides possible were used to help reduce the friction. A gyroscope mount is included on the wrist and one hook was attached to the base of the thumb to get the sweeping motion. The 3-D model of the hand is shown below in figure 1.

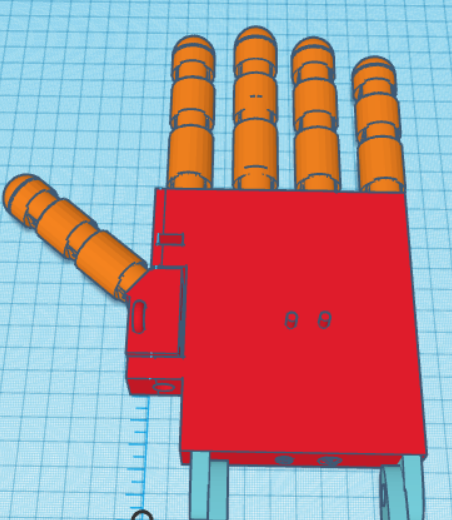


Figure 2 - Top View of Hand

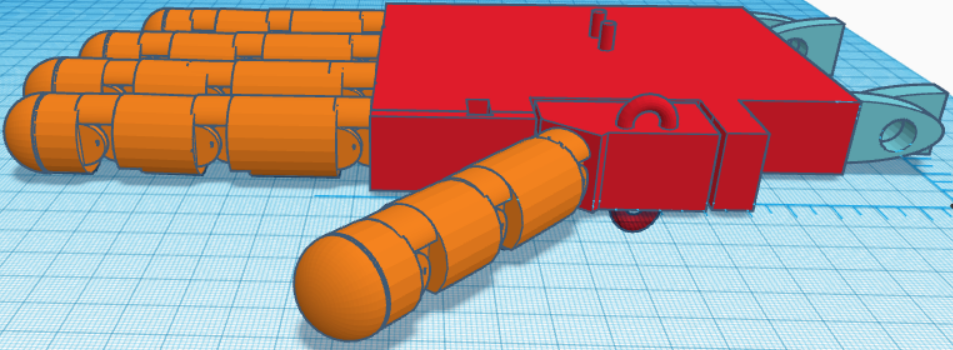


Figure 3 - Side View of Hand

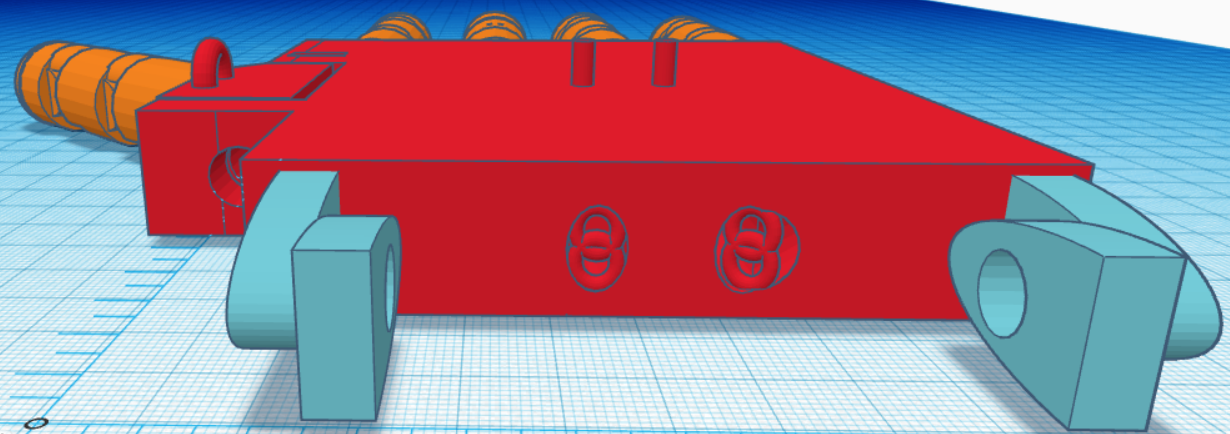


Figure 4 - Backside View of Hand

When viewing the side of the hand, one can see the stops on the backside of each joint of each finger. This will keep the fingers from being able to go past the resting point. There are notches in the links of the fingers as well in order to keep the links from stopping each other. Finally, in the back view one can see the two openings for the steel and rubber cables along with some of the guides. There is also a 9mm hole for the thumb adduction and abduction.

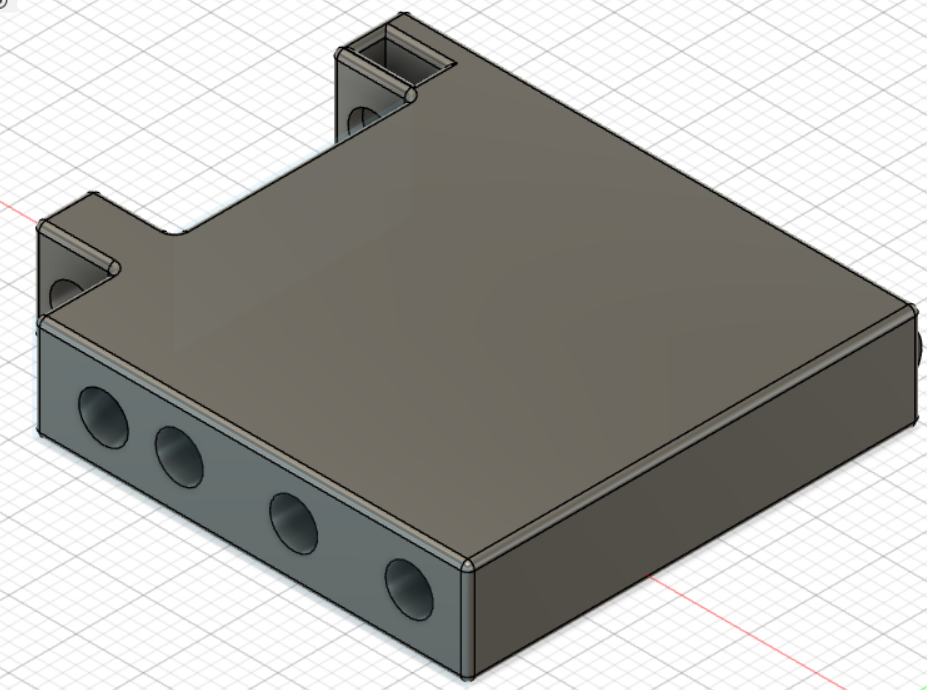
After the first version of the model was printed, the fingers were assembled and attached to the hand. When examining the size of the hand to the amount of actuation obtained from the fingers, it appeared that the first version of the hand was too large. The hand was remodeled in Fusion 360 to be smaller and have more efficient lines for the cables. The original hand was approximately 5.5 inches long and 4.5 inches wide. Version 2 of the palm of the hand was shrunk down to approximately 3.2 inches long and 3.1 inches wide. This way the full extension of the fingers reaches down to over halfway down the hand. The second rendering of the palm of the hand is shown in 

Figure 5 - Palm of Hand V2

In the second version of the palm model, there is no hinge for the wrist operation. The operation of the wrist was changed by coordinating with other group members. The group came up with the idea of using a piece to fit directly around the base of the hand which will be connected to a separate hinge. This piece can be seen in the wrist assembly under the body section of the robotic arm design. Holes will be cut into the wrist attachment to allow the cables performing finger actuation to run down to the forearm where the servo motors for actuation control will be mounted.

One of the most important aspects for the function of hand grasping is the thumb swivel piece. This allows the robotic thumb to help mimic the side to side motion of the thumb when gripping objects. The first version of the thumb swivel had the thumb mounted near the top portion of the swivel, similar to how a human hand works. When printed and assembled, the real life application of where the thumb is mounted on the swivel did not provide an accurate grasping motion. Due to this, the thumb swivel was redesigned to have the thumb mounted towards the bottom of the swivel, closer to the base of the hand. This will allow the thumb to come straight across the hand and provide a much better grasp for objects. The second version of the thumb swivel is shown below.

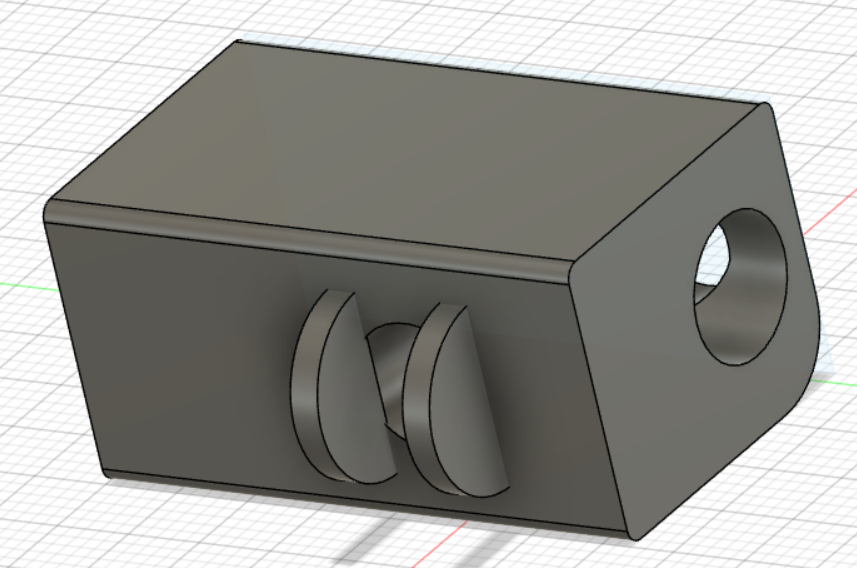


Figure 6 - Thumb Swivel V2

The final model that will need to be printed to make the hand fully operational is the mount for the servo motors. A mount was created that was able to perfectly fit over top of the forearm piece and keep itself snug within that area. The mount contained 6 slots for 5 servo motors and holes for each of the servo motors wiring to be routed through. There are also holes near the top of the mount in which the cables for hand actuation are able to be routed in order to keep them from getting tangled. The servo mount model can be seen below in Figure 7.

. 

Figure 7 - Servo mount for finger actuating servo motors

1. Kinematics

In order to calculate and analyze the motion of the hand, the kinematics of the hand were evaluated and will be explained in the following paragraphs. The first step for the kinematics is the DH table. Two separate DH tables were made for the hand, one that works for the pinkie, middle, ring, and index fingers, and one that works for the thumb motion. Both of the DH tables can be seen below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Order |  | d | a |  | Var |
| 0-1 | 0 | 0 | d1 | 0 | 1 |
| 1-2 | 0 | 0 | d2 | 0 | 2 |
| 2-3 | 0 | 0 | d3 | 0 | 3 |

Table 2 - DH table of fingers

The index finger and the ring finger are the same length so the d1-d3 values will stay the same for each of those. For the middle and pinky fingers, the d1-d3 values will increase and decrease in size respectfully. The DH table for the thumb is shown in table 3.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Order | Theta | d | a |  | Var |
| 0-1 | 0 | 0 | 0 | - | 1 |
| 1-2 | - | 0 | d2 | 0 | 2 |
| 2-3 | 0 | 0 | d3 | 0 | 3 |
| 3-4 | 0 | 0 | d4 | 0 | 4 |

Table 3 - DH table of thumb

The lengths for each distance were measured on tinkercad and a value of the different lengths in inches can be found in table 4. The distances found were measured from the middle of one joint to the middle of the next joint.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Finger | d1 | d2 | d3 | d4 |
| Index and ring | 1.35 | 1.1 | 0.75 | N/A |
| middle | 1.47 | 1.1 | 0.69 | N/A |
| pinky | 1.17 | 0.96 | 0.5 | N/A |
| thumb | 0 | 1.2 | 0.96 | 0.66 |

Table 4 - Distance values measured for each finger

After the DH tables have been made, the overall transformation matrix can be found. This is done by finding the homogeneous transformation matrix for each step and then multiplying them together. This was done using MATLAB and the code and results can be seen in Appendix D.

In order to find the transform matrix for each of the different fingers, the d1-d3 values from Table 4 would be plugged in.

To see the resulting transform matrices for the fingers and thumb see Appendix D. After the transform matrices are found, the jacobian can be calculated. The code for calculating the Jacobian for the fingers and thumb is shown below and the resulting Jacobian Matrices can be found in Appendix E.

5) Circuit Model

The circuit model for the hand involves the servo motors to actuate the fingers, an arduino, and a voltage bus.

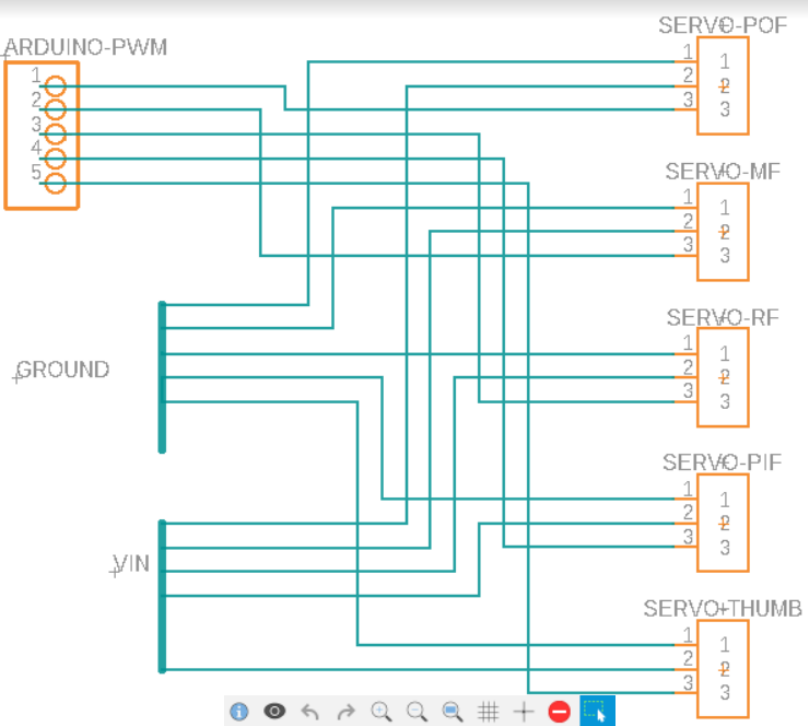


Figure 7 - Hand Servo Circuit Diagram

The servo motors will be powered using the wall outlet power supply which gives 24 V and 5A. A buck converter will be used to step down the voltage to 7V which is close to the maximum 7.2V that the servo motors use when operating at maximum stall torque. Each of the servo motors operate on approximately 450 mA when at maximum torque, so 5 A will be plenty of current to power the servo motors.

6) Code

The code for controlling the hand relates the input from the flex sensors to an amount of rotation in each servo. The amount of rotation is calculated from the main control code and is sent to the code for the hand. This code writes the servo motor a specific rotation amount to actuate the fingers.



Figure 8 - Hand Control Code

1. **Robotic Arm**
2. Control

Control of the arm’s mechanical function is done through a series of electric drives, 2 arduino microcontrollers, multiple gyroscopes, radio transceivers, and feedback signals. As a group, we designed for 6 degrees of freedom which requires 4 separate joints. The joints of discussion are shoulder, elbow, forearm, and wrist. As stated above, the goal was to create a robotic arm that mirrors the movement to that of the gauntlet controller. The arm gauntlet controller will contain an Arduino Due microcontroller, 2 gyroscopes, a radio transceiver, and a battery power supply. The robotic arm will also utilize an Arduino Due, 2 gyroscopes, a radio transceiver, 6 servo motors, and a power supply.

The physical location of the gyroscopes on both the robotic arm and the gauntlet are as follows: one on the upper arm link and one on the lower arm link. The two gyroscopes allow us to place where the user’s arm is located at any point in time using vector math to place the arm in space. Positional data values are taken from the gyroscopes of the gauntlet and transmitted from the gauntlet microcontroller to the robotic arm microcontroller via radio communication. This positional data is sampled multiple times in a matter of milliseconds for precise locating before getting relayed to the arm. The radio transceiver located on the arm receives positional data from the gauntlet. The arm microcontroller processes data, computes movement, and relays signals to the relevant motors to drive movement of the arm. At the same time, the arm gyroscopes are sampled for feedback positional data which is fed into a control loop compared with the received data in order to acquire precise movement of the motors.

1. Components and Structure

Once the goals for the arm structure were in place the joint parts and links were designed via Fusion 360 and 3D printed with PLA. This provides a lightweight, durable structure for the rotational motion and movement of each arm joint. Additionally, the design of the arm extended from shoulder to the tips of the fingers but it is placed on a two foot pedestal to imitate the torso of a human body and how the arm realistically moves.

The servo motors to drive joint motion are MG995, 20KG metal gear servos. The shoulder contains three motors, while the other three joints each have one. Due to the 180° rotation of the servo motors, there will be some limitations in the arm movement. However, due to strategic planning and placement of the motors, we were able to limit this as much as possible to maximize the arms range of motion with respect to human movement.

The microcontroller used to send various signals through the arm circuit is an Arduino DUE. Due to motors located throughout the arm, the arduino due is placed at a centralized location on the upper arm link to eliminate excessive lengths of wires from obstructing the robot's movement.

In the realm of communicating with the controlling gauntlet, radio transceivers are used. The transceiver on the gauntlet sends data packets in the form of position vector to the transceiver on the arm. The arm transceiver receives the data packets and orients the arm accordingly. The arm transceiver is located on the upper arm link near the microcontroller.

Lastly, the power supply circuit for the components needed to be designed. This includes a wall outlet power supply of 24V and 5A. From that supply, a buck converter is used to supply 7V to a power terminal block which is then connected to each of the servo motors and the Arduino DUE. This voltage gives the motors maximum torque within the recommended voltage range.

1. Modeling

As stated before, when modeling the arm, the goal was to achieve 6 DOF. We also aimed to design an arm that has movements as similar to a human arm as possible. These objectives have guided us to create a robotic arm that behaves as similar to a human arm as time and money will allow. As stated previously, the arm has some limitations in motion due to the use of 180° servos. The following figures show the structure of each joint with the servo motors included.

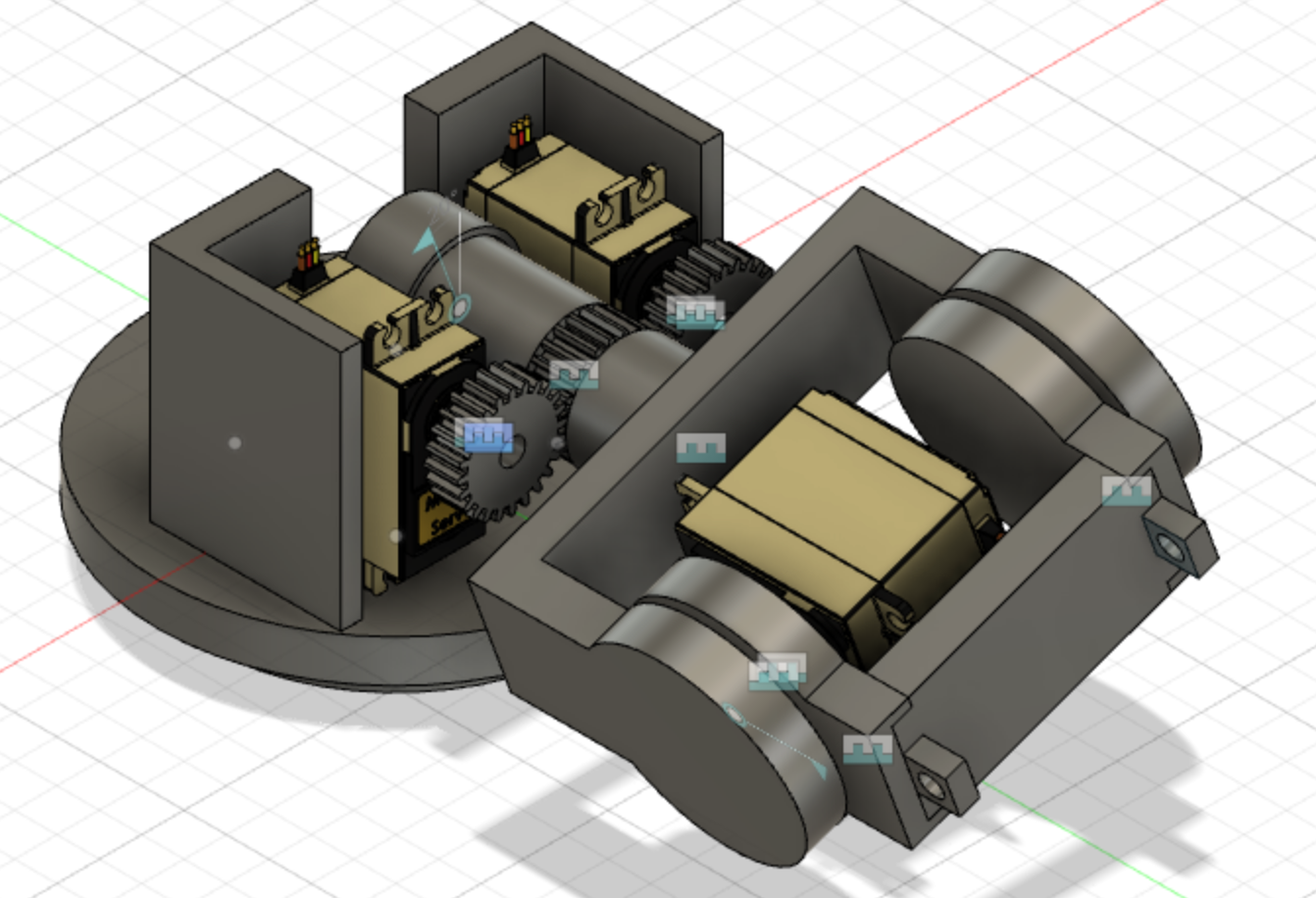


Figure 9 - Shoulder Assembly

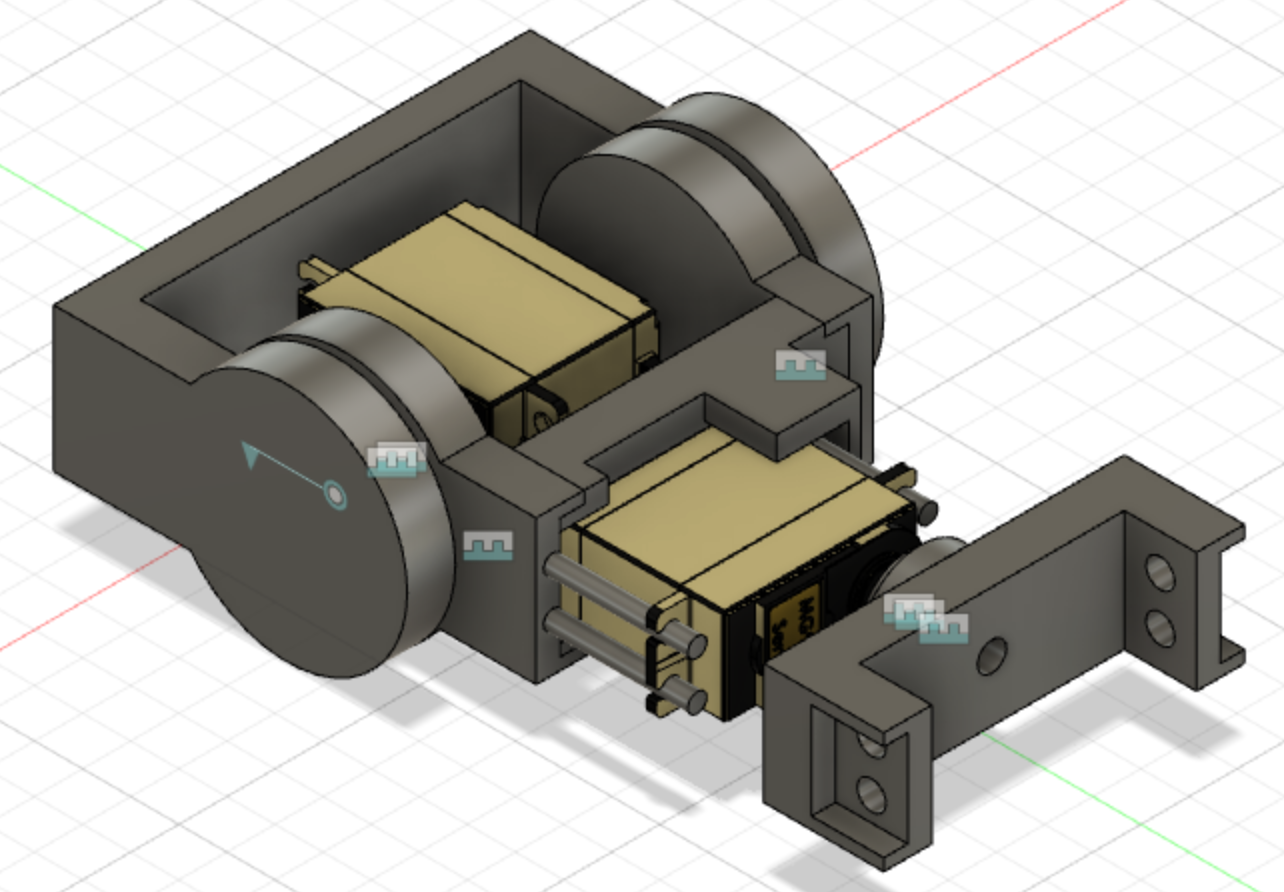


Figure 10 - Elbow/Forearm Assembly

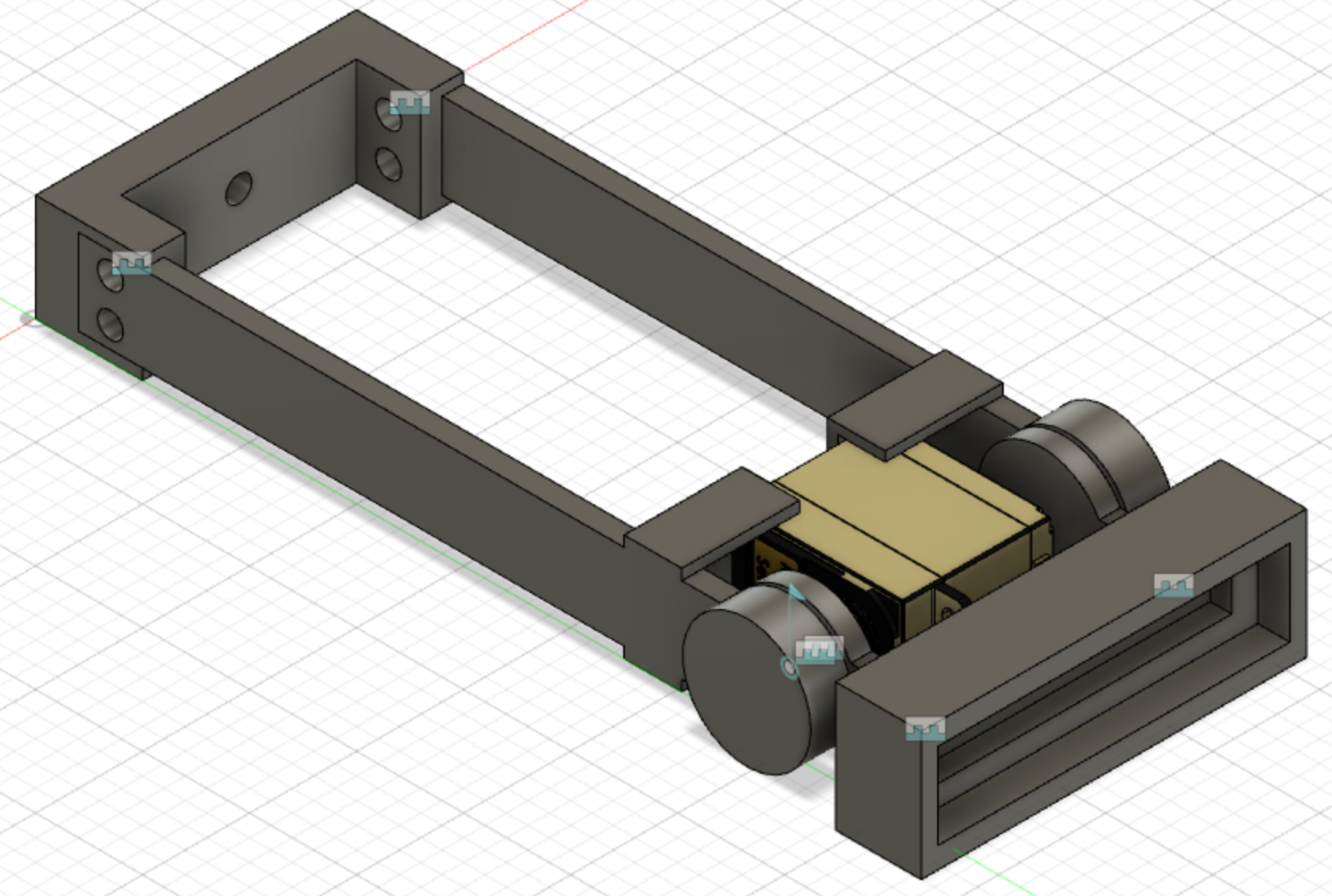


Figure 11 - Wrist Assembly

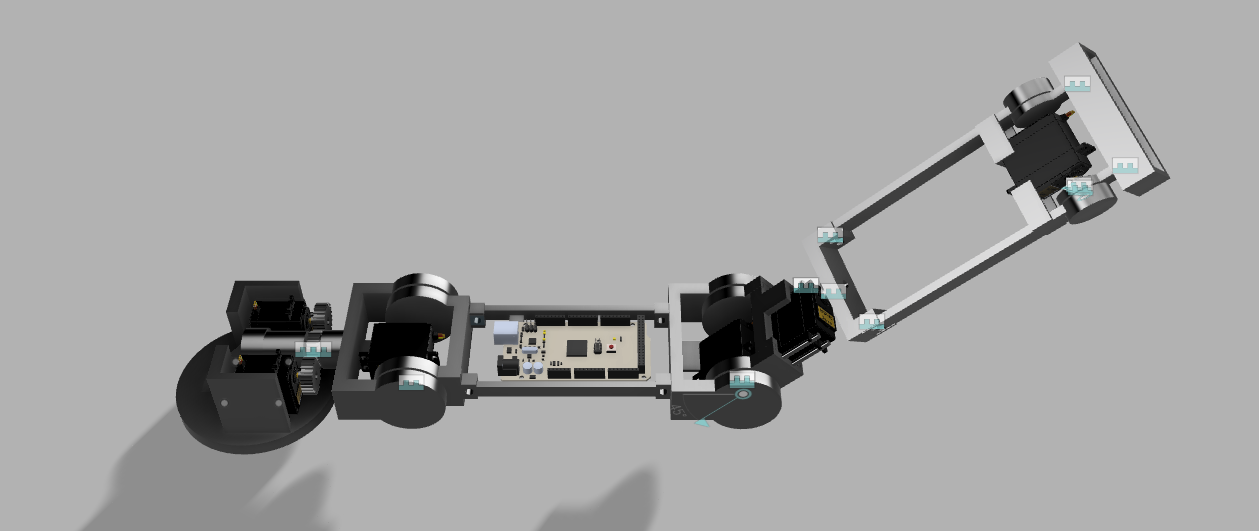


Figure 12 - Arm Assembly

1. Circuit Model

Designing the circuit for the arm was fairly straight forward. As stated previously, the servos as well as the Arduino DUE operate on a 7V source. This source will come from a wall outlet adapter that supplies 24V and 5A. A buck converter will step down the voltage to the usable level of 7V. The signal ports of the servos are connected to digital PWM pins. Both of the gyroscopes are powered by the arduino 3.3V output. The gyroscopes have two other pins that are connected to the analog pins of the arduino: SCL and SDA. SCL is the serial clock pin used for providing a clock pulse for inter-integrated circuit communication (I2C). SDA is the serial data pin used to transfer data through I2C. I2C is used to establish communication between two or more ICs. The SCL and SDA pins of each gyroscope are connected to the analog pins A5 and A4, respectively, on the Arduino. The radio transceiver used to communicate with the gauntlet controller is to be connected to the Arduino as well. The operating voltage of the transceiver is 1.9-3.6V so the VCC pin is connected to the 3.3V source on the Arduino. MOSI, MISO, and SCK are the Serial Peripheral Interface (SPI) pins and are connected to the SPI pins of the Arduino. SPI is used to communicate between two arduino microcontrollers. Thus, there is a master and a slave controller which is the gauntlet Arduino and the arm Arduino, respectively. The CSN and CE pins of the transceiver are connected to two separate digital pins on the Arduino. CSN and CE are used for setting the module in active mode and switching between command/transmit mode. The theoretical circuit schematic for the arm is shown in figure 16 below.

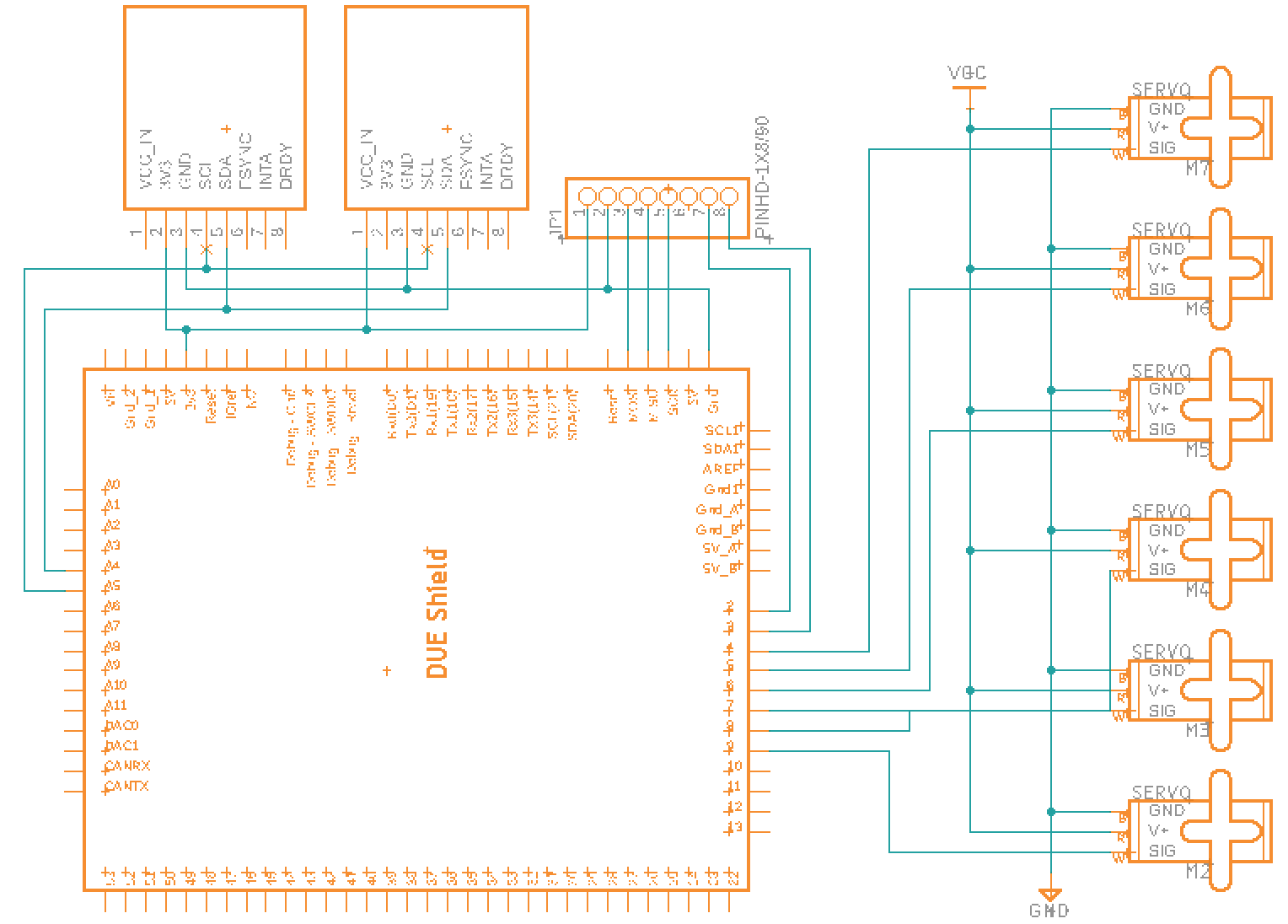


Figure 13 - Circuit Schematic for the Robotic Arm

1. Kinematics

To determine the orientation and location of the end effector (hand) in 3D space, the kinematics for the arm have been evaluated. We were able to calculate the DH table which helps us determine the various angles and lengths from each motion of the robotic arm. The DH table can be seen in Table 5 below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Order | 𝜽(z) | d(z) | a(x) | 𝛼(x) | Var |
| 0-1 | 0 | 0 | a1 | -90 | 𝜽1 |
| 1-2 | 0 | 0 | a2 | 0 | 𝜽2 |
| 2-2.5 | 0 | 0 | a3 | 90 | 0 |
| 2.5-3 | 90 | 0 | 0 | 90 | 𝜽3 |
| 3-4 | 0 | d4 | 0 | -90 | 𝜽4 |
| 4-4.5 | 0 | 0 | 0 | 90 | 0 |
| 4.5-5 | 0 | d5 | 0 | 0 | 𝛄 |

Table 5 - DH Table of the arm

**C. Control Systems**

1. Gauntlet Controller Background

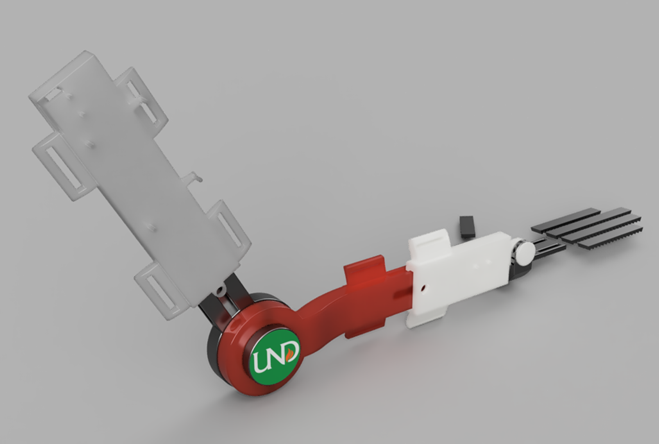
The control system for the A.R.M Project is a big focus point for the project. Our goal is to create a controller that can capture human arm movements at the joint levels and transmit that data wirelessly to a robotic arm that can then match the captured movements, ideally with as little lag as possible. Research showed there were many ways to capture human movements to translate to digital mediums like robots or computer programs. The most prevalent of which is motion capture, however similar to A. F. Panaite et al (16), I. Prayudi et al (17), J. A. Corrales et al (18), M. K. Kim et al (19), and Shintemirov et al (20), we sought to achieve a similar effect at a lower budget. Implementing IMU chips we can capture the movement of the operator's arm and translate that into data for the robot to follow. With that in mind, the first thing done for the controller, referred to in this paper as the Gauntlet, was creating a list of requirements which are as follows:

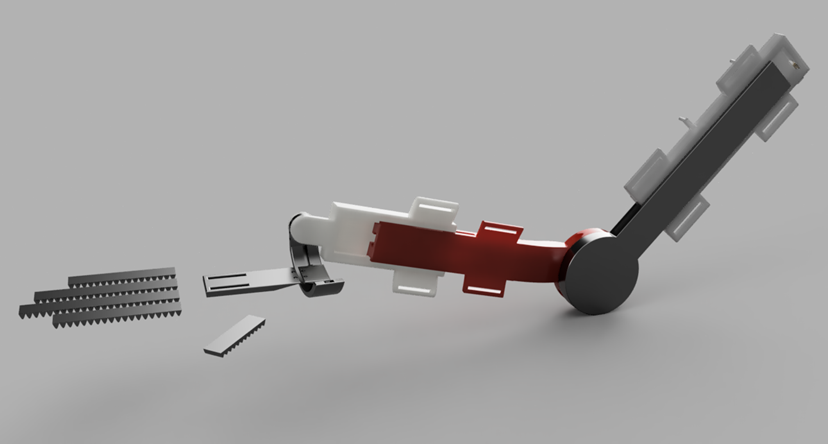
1. Gauntlet captures all movements of the operator’s arm, including:
   1. Shoulder joint rotation in two dimensions
   2. Elbow rotation in one dimension
   3. Forearm rotation in one dimension
   4. Wrist rotation in one dimension
   5. Thumb movement in one dimension
   6. Finger movement in one dimension
2. Gauntlet is completely wireless for portability and ease of use
3. Gauntlet’s data transmission is fast and reliable
4. Gauntlet uses as little hardware as necessary
5. Gauntlet size is adjustable in size so many users can operate it

With the system requirements agreed upon, the components were chosen. An Arduino Due will be used as the system controller because it is a larger Arduino board for starters and we have a lot of components to interface. The Arduino Due also has the fastest processor currently available to the Arduino controller family as well as a large amount of SRAM and digital pins to control all the servos in the project. To transmit the data, a nRF24L01 radio transceiver was chosen for easy interfacing to the Arduino system as well as having communication rates up to 2Mbps at lower ranges which is perfect for our ideal use scenario. To tackle points 1a-1c and 4, a GY512 MPU6050 was chosen because of familiarity with the board and how to collect data from it as well as it being fairly inexpensive. The gyroscopes can be used to measure changes in angles from initialization. If placed correctly on the Gauntlet, it can be used to determine the change in angle of the shoulder joint, and in conjunction with a second one placed on the forearm, the elbow rotation can be measured. At the joint of the wrist, a potentiometer will be implemented to determine the angle of rotation of the wrist by measuring the applied voltage across it and converting it into angular data, and a similar procedure will be done for the fingers by way of flex sensors acting as variable resistors, and converting the changing voltage values into angular rotation data. The entire controller will be powered by two lithium ion batteries in series to raise the input voltage to a nominal 7.2V at a full charge. The Due operates at a minimum voltage of 6V, and at 2,480mAh, the batteries should be able to power the Gauntlet for much longer than is necessary.

2) Gauntlet Controller Design

With the chosen components in mind, modeling the Gauntlet in Fusion 360 started. To meet requirement 5, the general design idea was adapted from the various aforementioned sources since their designs were tried and true and so that different sized arms could fit in the controller. The final design is shown below:

Figure 14 - Gauntlet V1 Model

Figure 15 - Gauntlet V1 Model

Each segment of the upper and lower arm consists of a rail and a slider. The two rails are connected at the ends, one with a post allowing the other to be fixed to it allowing for rotation in one dimension like the elbow. The rails are equipped with a slot running down the center of it allowing clearance for the head of a bolt to slide through it. The sliders have a hole for the bolt to fit through so a nut can be secured over the top. The piece can be slid up and down the rails to fit the user before being tightened down. At the end of the forearm slider is a similar post for another piece to fit around allowing for wrist rotation in one dimension. It was decided by the group to limit the movement of the wrist to one dimension to reduce the amount of servos used thereby reducing weight and complexity of design while still meeting most of the requirements for movement and control. The wrist has a half cuff rail to allow rotation of the wrist along the forearm axis when fully extended. From there a hinged slider is strapped to the hand. From there, there are ridged strips of flexible material that the flex sensors can be attached to with the wires routed underneath the hand hinge slider

.

3) Systems Control Flow - Gauntlet

With the design completed, the control flow was drawn out. It was during this process that the need for a visual status indicator as well as a manual multipurpose button were added to the design. The Gauntlet’s block diagram is shown below:

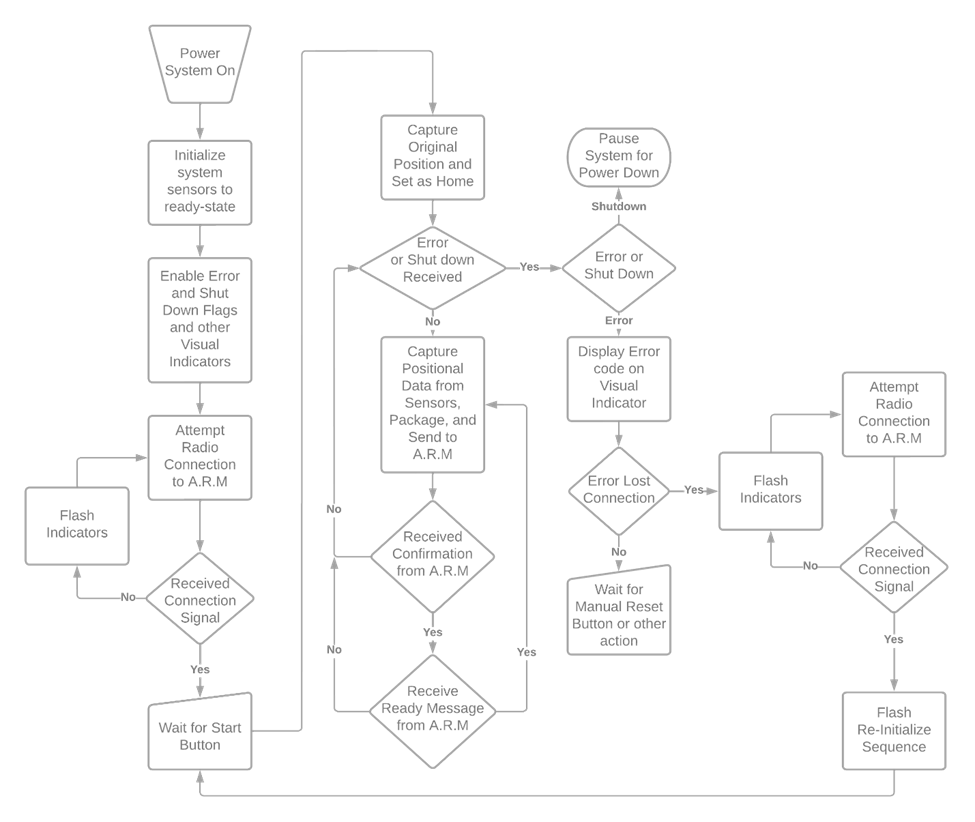


Figure 16 - Gauntlet Control Flow

The system will be powered on by flipping the switch connecting everything to the battery. Upon doing so the system will start its initialization sequence and attempt to connect to the controller flashing indication signals for feedback to the operator. Once connection is established, the system waits for the user to press the start button, of which will be pressed once the user has posed themselves in the correct orientation. After pressing the start button, the Gauntlet enters a control loop of collecting data, packaging the data, and communicating with the A.R.M while also checking for errors that may arise. In the event that connection to the A.R.M is lost, the Gauntlet will return to connection mode and attempt to re-connect to the A.R.M. If any other error is thrown, the status indicator will show the corresponding code to the user and wait for a manual reset.

4) Systems Control Flow - A.R.M

The control flow for the A.R.M has a similar structure to the Gauntlet, but inverts the sending and receiving messages from packing data to unpacking it, as well as integrating the limit switches into the design. The control flow is shown below:

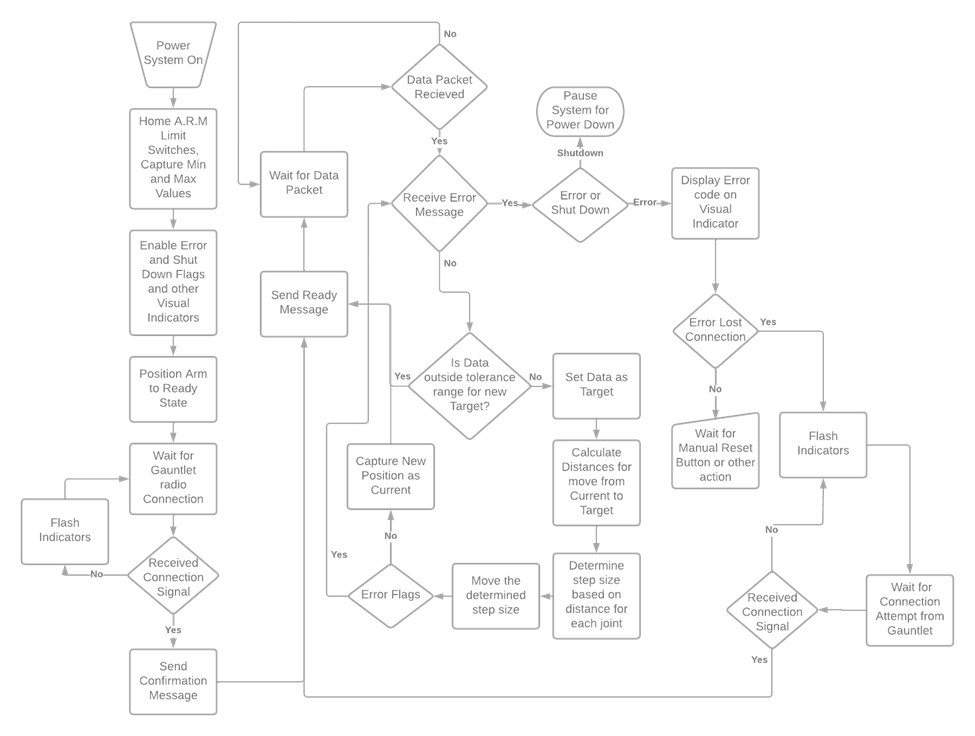


Figure 17 - ARM Control Flow

Akin to the Gauntlet, the system powers on after receiving power and a switch is thrown. After that the A.R.M homes itself via the limit switches and enables the different error flags and interrupts. The A.R.M then waits for communication from the Gauntlet to be established prior to moving itself to the ready position. After sending a ready message to the Gauntlet, the A.R.M initializes its control loop. The control loop is a relatively simple configurable PID loop. It has its current position at home saved from the original limit switch initialization and sits waiting for the data packet from the Gauntlet to tell itself where the new target position is. Once it receives the data packet, it checks if the new target data is inside or outside of a configurable tolerance range. The tolerance range is set so that the A.R.M isn’t constantly moving and jittery. If the new target data is outside the tolerance range, it calculates the distance between the new target position and the current position. It then sets the step size for movement based on how large or small the distances are. It then moves one step in the direction of the target position and sends a message to the gauntlet asking for new target data. It will check its error flags and interrupts and wait for the next data packet from the Gauntlet. This loop has a delay rate that can be configured. This will help pseudo smooth the A.R.M’s movement.

5) Controller Circuit Diagrams

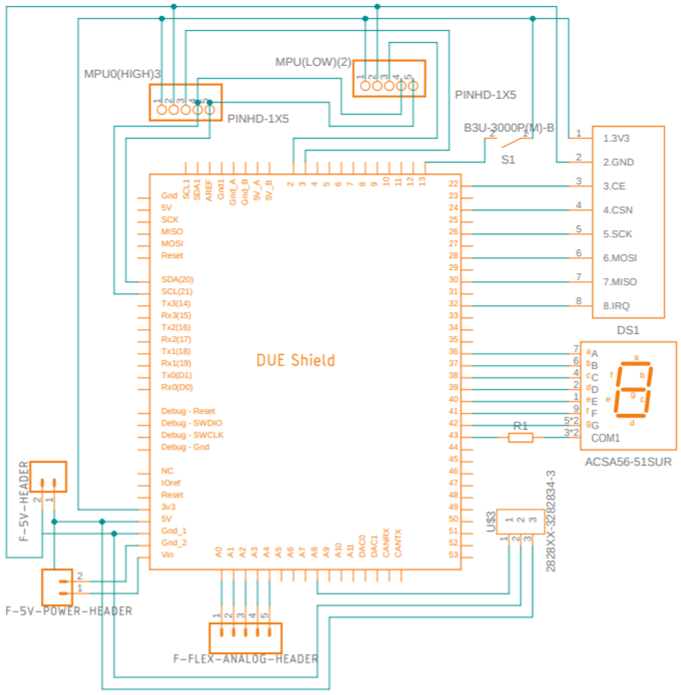


Figure 18 - PLC Circuit

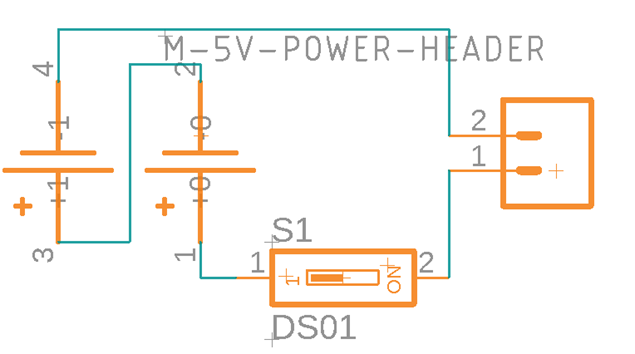
The controller circuit consists of an Arduino Due, the nRF24L01 radio transceiver, a seven segment display, a switch, and connectors for the various other components of the Gauntlet detailed below.

Figure 19 - Gauntlet Power Circuit

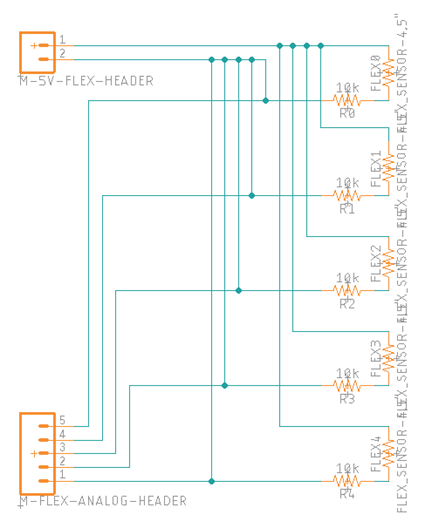
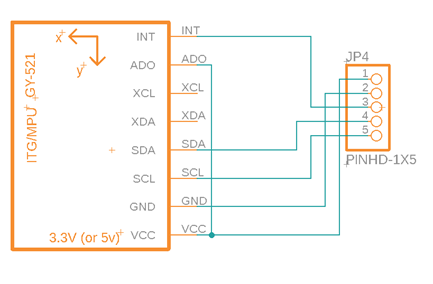
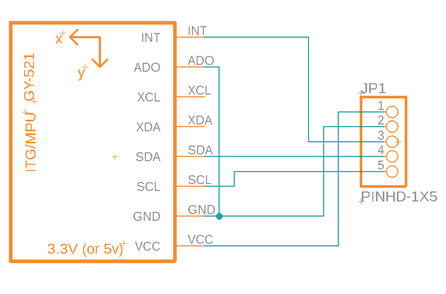
The battery circuit is simple enough with a battery holder putting two lithium cells in series with a switch.

Figure 20 - Gauntlet Flex Sensor Circuit

The flex sensors are all connected to a power bus and the voltage dividers are connected to a 5 pin connector to run back to the analog pins of the Arduino.

Figure 21 - Gauntlet Gyroscope Circuit

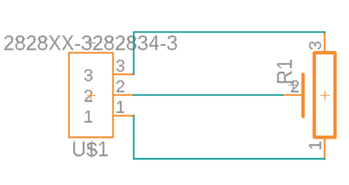
The first IMU (Top) located on the upper arm has its address pin grounded so that the Arduino can access it at register 0x69. The second IMU (Bottom) is located on the lower forearm, and its address pin is tied to the voltage in raising its address to 0x69. Otherwise, the SCL and SDA pins are going to be tied into the other IMUs SCL and SDA pins and they share a common voltage and ground. 

Figure 22 - Gauntlet Potentiometer Circuit

The final piece is the potentiometer which is a simple connection to a pin header that will tie back into the Arduino with the middle leg tying into one of the analog pins on the Arduino.

6) Arduino Code

The code was written in the Arduino IDE which is based in C++. The main libraries used were from Adafruit to interface with the MPU6050s better as well as the RF24 library to utilize the radio transmitter. Most of the functionality of each step of the control flows were made into their own respective functions. The final product didn’t have the full functionality that was expected. The entire code for both the ARM and gauntlet are on the github repository of which there is a link located in Appendix D.

**D. Power Systems**

1. Robotic Arm

The driving motors for the joint movement are MG995, 10KG servo motors that operate on a 7V supply. The Arduino DUE microcontroller has an recommended input voltage of 7-12V with a maximum input voltage level of 16V. Due to a low requirement of output supply, a 7V supply was used for both the servo motors and microcontroller. Our robotic arm will remain stationary on a solid surface so a wired power supply driven from a wall outlet was used. From the outlet, an AC to DC converter is connected to convert the current and step the voltage level down to 12V and 10A. A buck converter is used to further step the voltage down to 7V which is connected to a power bus terminal block. All of the servos as well as the Arduino DUE are connected to the power bus terminal block.

1. Gauntlet Controller

The gauntlet controller does not contain any motors but will have an Arduino DUE for signal processing. As stated above, the recommended input voltage of the microcontroller is 7-12V. Additionally, we want our gauntlet controller to be free of stationary wired connections as we need to be able to move around and operate the arm from various ranges. In order to achieve this, we are using two lithium ion batteries in series to power the gauntlet arduino microcontroller. This is mounted to the gauntlet controller to maintain free range of motion. Gyroscopes and flex sensors are connected to various pins on the microcontroller. The radio transceiver is connected to the serial data pins of the microcontroller.

1. *IMPLEMENTATION & TESTING*
2. **Arm**

The testing process for the arm started with building the structure of each joint and testing actuation individually. This was first done by hard coding angles for each servo motor and observing that the joint actuated accordingly. Once each joint motion was verified, we moved on to building the structure of the arm one joint at a time starting at the shoulder. Testing for the overall structure of the arm was done as each joint was added.

Starting with the shoulder pitch, we confirmed the joint functioned properly by relaying angles from the Arduino to the two servo motors. Once functionality was confirmed, the shoulder yaw joint was added to the pitch shaft thus connecting the two joints at the shoulder. Yaw was first tested individually, then both pitch and yaw were tested together to give a third degree of freedom at the shoulder.

From this point, the links between the yaw joint and elbow joint were connected to create the upper arm section, simulating the bicep/tricep area of the human arm. By doing this, it provided a place to mount the Arduino Due and power terminal bus. It also gave the wires of each servo a secure place to connect to the Arduino and power bus. From there, the elbow joint was similarly tested individually and then all three joints were tested together.

Once the elbow functionality was confirmed, the forearm rotational joint was incorporated with no added weight at the end of the arm. Similar testing was done to verify forearm rotation would work accordingly. Due to the rotation at this joint, the wires feeding back to the Arduino and power bus needed to be strategically routed in order to not impede motion. The lower arm links were then added at this joint to simulate the lower section of the human arm. This lower arm area is also where the hand servos were to be mounted to drive finger actuation. Before the hand servos were added, the wrist joint was first incorporated into the arm. The wires for the wrist motor were routed up the arm with the forearm wires. Again, this joint was first tested through sending various angles to the servo and then testing the wrist and forearm joints together. We continued to move up the arm and test all joints together to verify each joint was operating as designed in coordination with all other joints.

From this point, the mounting of the gyroscopes was strategically done to make sure the arm was able to accurately determine position data. To do this, one gyroscope was mounted on the upper arm link. This gave the ability to determine the position of the two shoulder joints. With this gyroscope placed close to the Arduino, the wires were easily routed to the microcontroller. The second gyroscope was placed on the back of the hand. This gyro provided the ability to determine both forearm and wrist position. The wires for this gyroscope routed back up the arm to the Arduino in the same manner as the forearm and wrist servo wires did. By using these two gyroscope locations, the position of each joint was successfully able to be determined. This position data was extrapolated with incoming data from the gauntlet controller to provide a target position for each motor to move to. The two gyroscopes were tested individually to confirm they both were reading proper data from the I2C bus and printing that data in the Arduino serial monitor. They both proved to successfully read data so the next step was to verify incoming data sent from the radio transceiver.

The radio transceiver was mounted on the upper arm link. This was done to keep it close to the Arduino and not create unnecessary wiring. The arm radio transceiver acted as the slave for the gauntlet tranceiver. In order to verify successful communication, we began by sending simple data packets from the gauntlet to the arm and then a received signal back to the gauntlet. Once successful communication was achieved, the hand was implemented into the arm to continue to test further.

1. **Hand**

The hand was one of the last parts of the project to be implemented. This was due to the cables for the hand needing to be cut to specific lengths when attached to the servo motors. Each of the three pieces for each finger were attached using a stripped piece of copper wire cut to a suitable length, and wired through the joints. The thumb swivel was connected into the hand using a steel bolt which was glued in towards the top of the hand, allowing thumb swivel rotation. All five of the servo motors that were used to perform finger actuation were placed inside the servo mount on the forearm. The servo mount had slots for the power, ground, and signal wires to run out the bottom. The wires were split into their respective colors and connected to a common pin. Using a DuPont kit wires were cut to the exact length needed for said connections to run up the side of the arm and be out of the way for the rest of the arm motion. The signal wires were routed into the nested arduino area, while the power and ground wires were connected to the power bus supplying approximately seven volts.

In order to articulate the hand, runs of 20-pound braided fishing line were used to run through the fingers, hand, and finally connected to the servo motors. The fishing line cables were glued onto the fingertips of each finger as well as rubber cable. On the other end, the fishing line was run through the servo horns and tied off and glued on to make sure the cables did not come undone.

When it came to testing the hand, some simple code was written to rotate each of the servo motors individually. Some for loops for a sweeping motion were used with a delay of 45 ms in order to tighten the cable but not so fast that it could snap off. The hand testing worked well, there were only a couple of problems with slack in the line. These problems with the slack were simple enough to fix by rotating the horn to a starting position that kept the line tight.An unforeseen reaction was caught when the cables for finger actuation were tightened. As the cables tightened, the wrist was pulled up due to the cables running through it. This caused the team to decide that the wrist motor needed to be set to a constant 90° in order to keep the wrist from bending upwards while actuating the fingers. When testing the finger actuation it was decided that the wrist would have to be set to 90 degrees The rubber cables that were routed through the hand in order to bring it back to neutral position did not function as expected. There were some issues with routing through the paths in the hand and wrist attachment that seemed to interfere with their function. In order to fix this problem, new lengths of rubber cable were simply glued on the backs of the fingers which immensely helped with resetting the fingers to their initial position.

1. **Control System**

The control system was tested as it was being built in the Arduino IDE using the serial monitor. As each feature was implemented, the serial monitor was used to verify the feature was working correctly before moving to the next item. We began by initializing the gyroscopes and making sure that the data received was being read correctly if at all. The two gyroscopes operated on the I2C communication bus, and were separately addressed for reading data. The Adafruit library helped push the project along since it automatically manipulated the raw input to output angular rotation data. This data was sampled and averaged over a short time to create a simple integration that changed the angular rotation velocity to angular rotation position. This new data point was continually created and summed each loop in order to have current positional data as the controller was moved from its initialization position. This process was repeated for each yaw, pitch, and roll data point for the two gyroscopes. After that was reliably being read, the potentiometer at the elbow was next. This required no additional libraries to implement. The potentiometer requires ground, power, and its remaining leg can be utilized to pull the varying voltage values from on one of the arduino analog pins. This data coming in was easy to use as well. as we moved the arm to the lower and upper limits, measured the analog values at each point, and used the built in map() function to convert the data to usable rotational values. The finger flex sensors were implemented next. The circuit was simple to build, being a voltage divider circuit. A 10kOhm resistor was soldered to the power leg and one leg of the flex sensor, then a signal wire was connected between the resistor and the flex sensor, and finally a ground wire was connected to the remaining leg of the flex sensor. Similar to the potentiometer, the flex sensor values were read into the analog pins of the arduino and changed to a value between 0 and 100 to show percentage of flexion. Finally the aforementioned data was packaged into an integer array in order to send to the arm. This required a library designed for the NRF2401 library. These gyroscopes can only handle 32 bytes of data per message sent. Using the arduino sizeof() function returns the size of the data in bytes, by using that, we found that the gyroscope and potentiometer data was 22 bytes, so we added one more integer to the array as an indicator for the ARM to use to differentiate the data later. The flex sensor data was packaged into its own integer array with a second indicator integer as well.

1. *CONCLUSION*

Bringing all of the aforementioned components together, we began combined testing. We began with verifying that the controller and the ARM both were successfully reading their gyroscopes from the I2C bus and the data read was usable. Using that data to verify the radio communication was done next. Once the data coming in from the gyroscopes on the controller was verified, a new integer data array was created to contain the incoming data as new received data’. This data was used in combination with the ARM’s ‘current data’ to mathematically find the distance between the two creating the final ‘target data’. This target data was again checked against the current data in order to determine how far the ARM would need to move so that until the next received data pack comes in. We used this data to create a simple test function to determine the directions that the servos would need to move based on input. If the controller moved below the current ARM position, the serial monitor would print “Go Down” and if it was above the ARM’s current position the serial monitor would print “Go Up”. This was done for each servo, joint, and degree of freedom necessary. Once the directions were set up to correctly move the servos, we began testing movement using the controller as input. We began small using only the elbow to test the controller inputs.

The first few tests were unsuccessful. Nothing seemed to be working like it had previously. We tried many things to get the arm working: checking wiring to make sure it was connected correctly, making sure the controller and ARM were powered correctly, we tried editing the code in case some of the serial print commands were causing issues, and finally we started commenting out serial print lines thinking it would help as well as checking the delays in the code itself to see if there were issues.

1. *FUTURE WORK*

In order to further improve upon this project there are several aspects that could be changed. The first being the implementation of faster, stronger motors. The motors that our team used were easy to work with as well as inexpensive but did not possess the torque needed to fully capture the range of motion the team was working towards. The motors controlling the forearm, wrist, and fingers all had enough torque for their respective parts but the trouble came with the elbow and shoulder motors. So further analysis of stress on joints and required torque output would need to be done to correctly size the motors. The second aspect that could be improved upon would be the finger articulation. While the fingers were able to capture a good range of motion, it was very linear and could be designed to be much more precise. The motion of grasping objects was the goal and this could be accomplished with a couple of tweaks to the hand and finger design. Better cable routing or even a different system for finger actuation would be quite beneficial to the hand articulation.

For the controller’s physical design, the brace could be reworked in a few areas. The first being the elbow joint and how the brace attached to the arm could be redone since the current model slides around a little bit when it is worn and used by the operator. The material could be changed as well from 3D printed plastic to some sort of metal like aluminum to reduce the overall size, but allow it to stay relatively lightweight. Finally, moving to manufacture some PCBs to drop components in place rather than soldering wires and making jumper wire harnesses would be beneficial for both saving time as well as clarity for circuit integrity

References

[1] H. Mnyusiwalla, P. Vulliez, J.-P. Gazeau, and S. Zeghloul, “A new dexterous hand based on bio-inspired finger design for inside-hand manipulation,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 6, pp. 809–817, Jun. 2016.

[2] J. E. Puig, N. E. Rodriguez, and M. Ceccarelli, “A methodology for the design of robotic hands with multiple fingers,” *International Journal of Advanced Robotic Systems*, vol. 5, no. 2, p. 22, 2008.

[3] L. Tian, N. M. Thalman, D. Thalmann, and J. Zheng, “Nature grasping by a cable-driven under-actuated anthropomorphic robotic hand,” *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 17, no. 1, p. 1, 2019.

[4] L. Chen, H. Sun, W. Zhao, and T. Yu, “Robotic Arm Control System Based on AI Wearable Acceleration Sensor,” *Mathematical Problems in Engineering*, 03-Mar-2021. [Online]. Available: https://www.hindawi.com/journals/mpe/2021/5544375/. [Accessed: 12-Oct-2021]

[5] S. Alvarez-Rodríguez and F. G. Peña Lecona, “th Order Sensor Output to Control -DoF Serial Robot Arms,” *Journal of Sensors*, 11-Jan-2021. [Online]. https://www.hindawi.com/journals/js/2021/8884282/.[Accessed: 12-Oct-2021].

[6] W. G. Hao, Y. Y. Leck and L. C. Hun, "6-DOF PC-Based Robotic Arm (PC-ROBOARM) with efficient trajectory planning and speed control," 2011 4th International Conference on Mechatronics (ICOM), 2011, pp. 1-7, doi: 10.1109/ICOM.2011.5937171

[7] N. RNaveen, “Arduino Communication Protocols (Wired and Wireless for IoT),” *Technobyte*, 03-Nov-2019. [Online]. Available: https://technobyte.org/arduino-communication-protocols-wired-wireless-for-iot/. [Accessed: 12-Oct-2021].

[8] Talli, Amit, and Vinod Kumar V Meti. “Design, Simulation, and Analysis of a 6-Axis Robot Using Robot Visualization Software.” *IOP Science*, 2020, https://iopscience.iop.org/article/10.1088/1757-899X/872/1/012040.

[9] Pachaiyappan, S, et al. “Design and Analysis of an Articulated Robot Arm for Various Industrial Applications.” *IOSR Journal*, 2014, https://www.iosrjournals.org/iosr-jmce/papers/NCCAMABS/V

[10] Reddy, Gurudu Rishank, and Venkata Krishna Prashanth Eranki. “Design and Structural Analysis of a Robotic Arm.” *DIVA Portal*, 2016, https://www.diva-portal.org/smash/get/diva2:1068547/FULLTEXT02.

[11] P. Bora and V. Nandi, "Low cost shadow function based articulated robotic arm," *2015 International Conference on Energy, Power and Environment: Towards Sustainable Growth (ICEPE)*, 2015, pp. 1-4, doi: 10.1109/EPETSG.2015.7510079.

[12] H. Kim, L. M. Miller, N. Byl, G. M. Abrams and J. Rosen, "Redundancy Resolution of the Human Arm and an Upper Limb Exoskeleton," in *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 6, pp. 1770-1779, June 2012, doi: 10.1109/TBME.2012.2194489.

[13] “ServoWrite,” *Arduino*. [Online]. Available: https://www.arduino.cc/en/Reference/ServoWrite. [Accessed: 30-Mar-2022].

[14] *Single Chip 2.4 Ghz Transceiver NRF24L01*. Nordic Semiconductor ASA, Mar. 2006, https://www.sparkfun.com/datasheets/Components/nRF24L01\_prelim\_prod\_spec\_1\_2.pdf.

[15] “MPU6050 Accelerometer and Gyroscope Module.” *Components 101*, 17 Mar. 2021, https://components101.com/sensors/mpu6050-module.

[16] A. F. Panaite , M. Leba , M. L. Olar , R. C. Sibisanu and L. Pellegrini, “Human Arm Motion Capture using Gyroscopic Sensors,” 2021 MATEC Web of Conferences 343, 08007 (2021), doi: 10.1051/matecconf.2021.34308007.

[17] I. Prayudi and D. Kim, "Design and implementation of IMU-based human arm motion capture system," 2012 IEEE International Conference on Mechatronics and Automation, 2012, pp. 670-675, doi: 10.1109/ICMA.2012.6283221.

[18] J. A. Corrales, F. A. Candelas and F. Torres, "Hybrid tracking of human operators using IMU/UWB data fusion by a Kalman filter," 2008 3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI), 2008, pp. 193-200, doi: 10.1145/1349822.1349848.

[19] M. K. Kim, K. Ryu, Y. Oh, S. -R. Oh and K. Kim, "Implementation of real-time motion and force capturing system for tele-manipulation based on sEMG signals and IMU motion data," 2014 IEEE International Conference on Robotics and Automation (ICRA), 2014, pp. 5658-5664, doi: 10.1109/ICRA.2014.6907691.

[20] Shintemirov, Almas et al. “An Open-Source 7-DOF Wireless Human Arm Motion-Tracking System for Use in Robotics Research.” Sensors (Basel, Switzerland) vol. 20,11 3082. 29 May. 2020, doi:10.3390/s20113082

APPENDIX

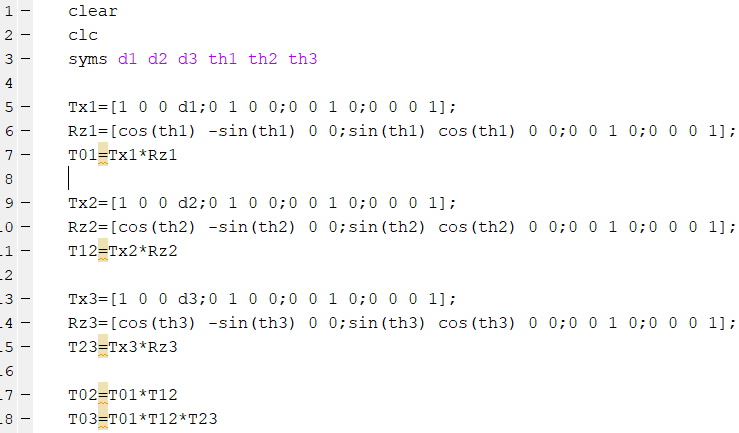
1. Hardware Parts List

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Part | Count | Cost/Unit |  | Part | Count | Cost/Unit |
| Arduino DUE | 2 | 40.30 | 1/4in machine screws | 1 | 1.49 |
| GY-512 MPU6050 | 4 | 2.99 | Bearings | 1 | 7.14 |
| 4PC 20kg Digital 180deg Servo Motors | 2 | 18.99 | Servo Mounts | 3 | 9.99 |
| 4PC NRF24L01 2.4G Radio Transmitter | 1 | 7.89 | Power Distribution Board | 1 | 6.99 |
| AC to DC Power Converter | 1 | 20.99 | Buck Converter | 1 | 3.00 |
| Lithium Ion Battery Holder | 1 | 6.99 |  |  |  |
| Misc Wire and Heat Shrink Kit | 1 | 12.99 |  |  |  |
| Dupont Kit | 1 | 11.99 |  |  |  |
| Connector Kit | 1 | 11.99 |  |  |  |
| Steel Cable | 1 | 17.99 |  |  |  |
| 2mm braided rubber cable | 1 | 6.99 |  |  |  |
| 8mm Bolt | 1 | 1.99 |  |  |  |
| Circular Mounting Bracket | 1 | 6.49 |  |  |  |
| 24in Rigid Pipe | 1 | 12.98 |  |  |  |
| 2x2 wood | 2 | 9.36 |  |  |  |
| Wood mounting bracket | 1 | 3.28 |  |  |  |
| 6mm machine screws | 1 | 1.49 |  |  |  |
| 10mm machine screws | 1 | 1.49 |  |  |  |

1. Fingers and Thumb Transform

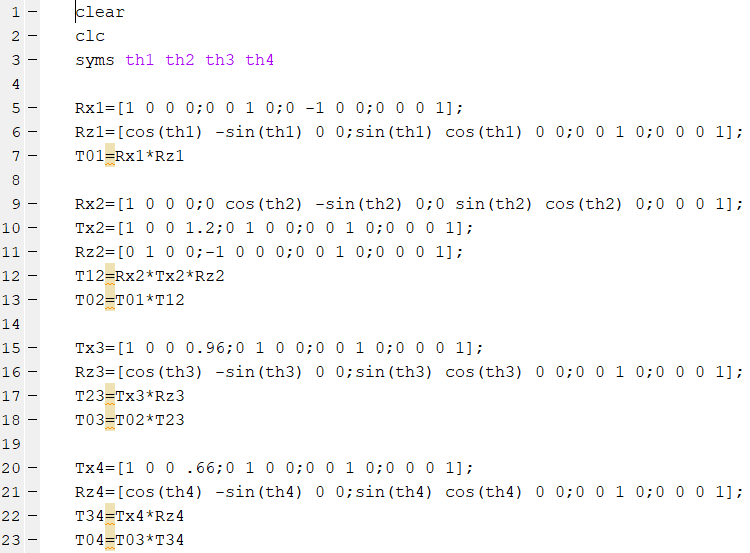
The transform for each of the fingers with the variables d1-d3 included are shown below.

|  |  |  |  |
| --- | --- | --- | --- |
| cos(th3)\*(cos(th1)\*cos(th2) - sin(th1)\*sin(th2)) - sin(th3)\*(cos(th1)\*sin(th2) + cos(th2)\*sin(th1)) | - cos(th3)\*(cos(th1)\*sin(th2) + cos(th2)\*sin(th1)) - sin(th3)\*(cos(th1)\*cos(th2) - sin(th1)\*sin(th2)) | 0 | d1 + d3\*(cos(th1)\*cos(th2) - sin(th1)\*sin(th2)) + d2\*cos(th1) |
| cos(th3)\*(cos(th1)\*sin(th2) + cos(th2)\*sin(th1)) + sin(th3)\*(cos(th1)\*cos(th2) - sin(th1)\*sin(th2)) | cos(th3)\*(cos(th1)\*cos(th2) - sin(th1)\*sin(th2)) - sin(th3)\*(cos(th1)\*sin(th2) + cos(th2)\*sin(th1)) | 0 | d3\*(cos(th1)\*sin(th2) + cos(th2)\*sin(th1)) + d2\*sin(th1) |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 |



Result of Finger Transform Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| cos(th4)\*(cos(th1)\*sin(th3) + cos(th2)\*cos(th3)\*sin(th1)) + sin(th4)\*(cos(th1)\*cos(th3) - cos(th2)\*sin(th1)\*sin(th3)) | cos(th4)\*(cos(th1)\*cos(th3) - cos(th2)\*sin(th1)\*sin(th3)) - sin(th4)\*(cos(th1)\*sin(th3) + cos(th2)\*cos(th3)\*sin(th1)) | sin(th1)\*sin(th2) | (6\*cos(th1))/5 + (24\*cos(th2)\*sin(th1))/25 + (33\*cos(th1)\*sin(th3))/50 + (33\*cos(th2)\*cos(th3)\*sin(th1))/50 |
| sin(th2)\*sin(th3)\*sin(th4) - cos(th3)\*cos(th4)\*sin(th2) | cos(th3)\*sin(th2)\*sin(th4) + cos(th4)\*sin(th2)\*sin(th3) | cos(th2) | - (24\*sin(th2))/25 - (33\*cos(th3)\*sin(th2))/50 |
| sin(th2)\*sin(th3)\*sin(th4) - cos(th3)\*cos(th4)\*sin(th2) | sin(th4)\*(sin(th1)\*sin(th3) - cos(th1)\*cos(th2)\*cos(th3)) - cos(th4)\*(cos(th3)\*sin(th1) + cos(th1)\*cos(th2)\*sin(th3)) | cos(th1)\*sin(th2) | (24\*cos(th1)\*cos(th2))/25 - (6\*sin(th1))/5 - (33\*sin(th1)\*sin(th3))/50 + (33\*cos(th1)\*cos(th2)\*cos(th3))/50 |
| 0 | 0 | 0 | 1 |

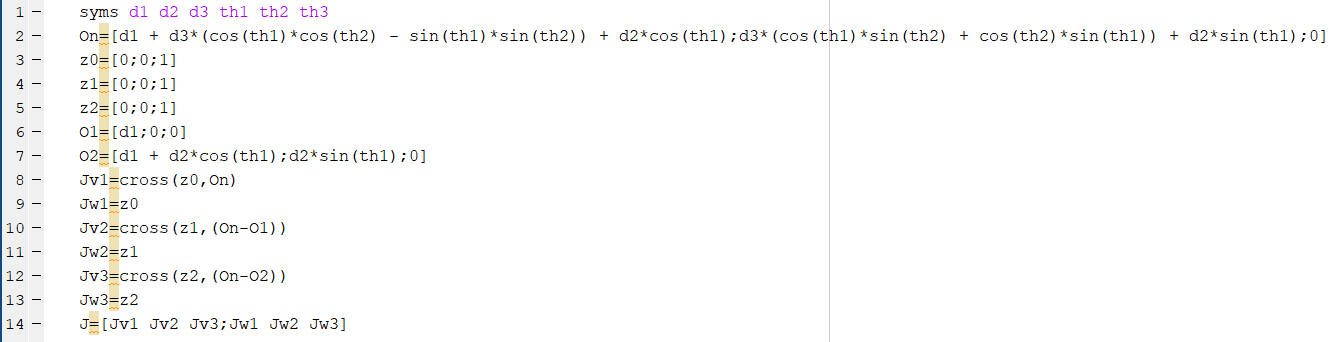


Resulting Transform Matrix for Thumb with d values substituted

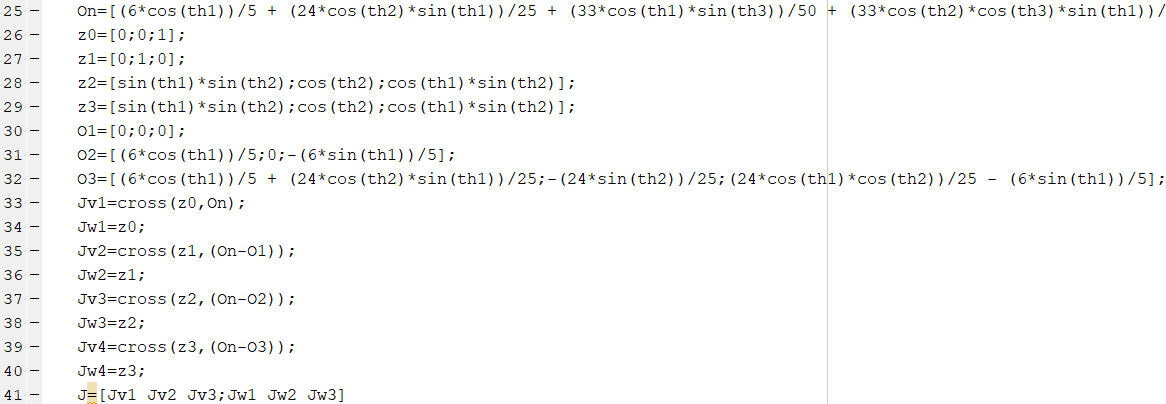
1. Fingers and Thumb Jacobian

|  |  |  |
| --- | --- | --- |
| - d3\*(cos(th1)\*sin(th2) + cos(th2)\*sin(th1)) - d2\*sin(th1) | - d3\*(cos(th1)\*sin(th2) + cos(th2)\*sin(th1)) - d2\*sin(th1) | -d3\*(cos(th1)\*sin(th2) + cos(th2)\*sin(th1)) |
| d1 + d3\*(cos(th1)\*cos(th2) - sin(th1)\*sin(th2)) + d2\*cos(th1) | d3\*(cos(th1)\*cos(th2) - sin(th1)\*sin(th2)) + d2\*cos(th1) | d3\*(cos(th1)\*cos(th2) - sin(th1)\*sin(th2)) |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

Jacobian Matrix for the Fingers



|  |  |  |
| --- | --- | --- |
| (24\*sin(th2))/25 + (33\*cos(th3)\*sin(th2))/50 | (24\*cos(th1)\*cos(th2))/25 - (6\*sin(th1))/5 - (33\*sin(th1)\*sin(th3))/50 + (33\*cos(th1)\*cos(th2)\*cos(th3))/50 | cos(th2)\*((24\*cos(th1)\*cos(th2))/25 - (33\*sin(th1)\*sin(th3))/50 + (33\*cos(th1)\*cos(th2)\*cos(th3))/50) + cos(th1)\*sin(th2)\*((24\*sin(th2))/25 + (33\*cos(th3)\*sin(th2))/50) |
|  |  |  |
| (6\*cos(th1))/5 + (24\*cos(th2)\*sin(th1))/25 + (33\*cos(th1)\*sin(th3))/50 + (33\*cos(th2)\*cos(th3)\*sin(th1))/50 | 0 | cos(th1)\*sin(th2)\*((24\*cos(th2)\*sin(th1))/25 + (33\*cos(th1)\*sin(th3))/50 + (33\*cos(th2)\*cos(th3)\*sin(th1))/50) - sin(th1)\*sin(th2)\*((24\*cos(th1)\*cos(th2))/25 - (33\*sin(th1)\*sin(th3))/50 + (33\*cos(th1)\*cos(th2)\*cos(th3))/50) |
| 0 | - (6\*cos(th1))/5 - (24\*cos(th2)\*sin(th1))/25 - (33\*cos(th1)\*sin(th3))/50 - (33\*cos(th2)\*cos(th3)\*sin(th1))/50 | - cos(th2)\*((24\*cos(th2)\*sin(th1))/25 + (33\*cos(th1)\*sin(th3))/50 + (33\*cos(th2)\*cos(th3)\*sin(th1))/50) - sin(th1)\*sin(th2)\*((24\*sin(th2))/25 + (33\*cos(th3)\*sin(th2))/50 |
| 0 | 0 | sin(th1)\*sin(th2) |
| 0 | 1 | cos(th2) |
| 1 | 0 | cos(th1)\*sin(th2) |



Jacobian Matrix For Thumb kinematics

1. Github code repository

<https://github.com/garrett-tjernagel/A.R.M_UNDCEMEE481SeniorDesignProject>