HURON MQP Report

Rachel Dancy - Electrical & Computer Engineering

Jonathan Gong - Robotics Engineering

Aislin Hanscom - Robotics Engineering

Cameron Huneke - Robotics Engineering

Peter Lam - Electrical & Computer Engineering

Curtis Lee - Robotics Engineering

John Marcotte - Mechanical & Electrical & Computer Engineering

Rahil Parikh - Computer Science & Robotics Engineering

Angelo Ruggeri - Computer Science & Robotics Engineering

Brendyn Sang - Robotics & Mechanical Engineering

Kyle Staubi - Biomedical Engineering

Mohammad Madhi Agheli Hajiabadi (Advisor)

Karen Troy (Co-Advisor)

Cagdas Onal (Co-Advisor)

Markus Nemitz (Co-Advisor)

William R. Michalson (Co-Advisor)

Table of Figures

1.0 Introduction

Humanoid robots, like humans, have been evolving over time, especially as more and more research involving the myriad uses, ranging from personal assistance to search and rescue, (*Top 10 Examples of Humanoid Robots - ASME*, n.d., p. 10) of such robots has been brought to everyone's attention. Nonetheless, the biggest challenge with humanoid robots is to mimic human-like reactions, control, and the ability to adapt to the ever-changing environment. Additionally, while that is the biggest challenge within the field, the biggest barrier to entering the field of robotics itself is the immense cost involved in research and development.

The problem we faced was creating a bipedal robot that fits human dimensions. Many humanoid robots currently in existence are not of adult human proportions, with most only being around five feet tall. However, the additional constraint of our project is to create a functional walking and balancing robot that is closer in size to an adult while also utilizing a small budget and minimizing the weight to reduce strain on the motors. This is an essential problem because designing and building a robot more similar in shape to humans allows for testing balance and walking conditions that are more like humans. A set of human legs allows for a continuation project where an upper body can later be added, and with a full robot, more complex tasks can be tested, such as moving to a specified location to complete a task, one example of this is walking up to a desk to pick up a box. We have already seen the kind of impact humanoid bipeds have had on the world with the popularity of Boston Dynamics' Atlas robot, which often demonstrates the agility, mobility, and strength of their robot through various obstacle courses and programmed tasks. The goal of our project is to design, fabricate, and program a humanoid life-sized set of legs to balance and walk by the end of C term.

2.0 Motivation

As this is a brand-new project starting from scratch, our goal in terms of motion is to balance and react accordingly to any external forces. This means that if the legs are pushed, it should be able to correct itself back to an upright position. Then, if there is time, being able to take a few steps would be ideal, however, getting the robot to balance properly is more important since it cannot walk without balancing first. The balancing and walking motion should look more human-like and not have an unnatural appearance when moving. In terms of appearance, the robot should mimic human proportions as close as possible. The robot should not be too bulky or have any "non-human" features, as it should be able to wear pants and fit into shoes.

The reason for these goals is that having a humanoid robot would mean that it could be used for research purposes. While saying "research purposes" is broad, it also means there are a lot of opportunities for the robot to be used. There could be tasks that humans currently do that would be less dangerous if a robot were to do it. It could also be used to work in collaboration with humans in a work or personal environment in the next few years. These are just a couple of examples but having proof of concept of a humanoid robot would lead to a successful project in the long run.

Eventually, we want our humanoid robot, named HURON, to be used for disaster relief that requires navigation of terrain too dangerous for humans. Hence, it is important that we, and future MQPs, can successfully produce the lower half of a self-balancing robot that can react to forces accordingly and regain its balance, while walking with a human-like gait. To reach this point, we want to make sure our team can build a solid foundation for future MQPs to improve and build off.

3.0 Background

This chapter contains all the research that all three subteams have done. Each subteam divided their research into different sections that were referenced throughout the team's work during the terms.

3.1 Humanoid Robot Design

The first thing our team did was research designs that have been done for humanoid robots in the past. There were a few key parts of the robot that needed to be researched. One was the degrees of freedom (DOF) of the legs, as we wanted to see what was the optimal DOF that other robots used. Another aspect was the overall leg structure, where we looked at designs of other humanoid legs to gauge what materials are being used and how the joints are constructed. As we continued throughout the term, we learned the importance of positioning with joint axes, leading us to look more closely into how other designs have their joints oriented. Lastly, we researched how human bones are structured as we wanted to make our robot's legs mimic human legs as much as possible.

3.1.1 Degrees of Freedom

One of the first robots that we researched was the KHR-3 robot. It had 3 degrees of freedom (DOF) in the hip (roll, pitch, yaw), 1 in the knee (pitch), and 2 in the ankle (roll and pitch) for a total of 12 DOF in the legs. They had a similar goal of imitating human walking and therefore landed on the configuration shown in Figure 3.1 (Ill-Woo Park et al., 2005).

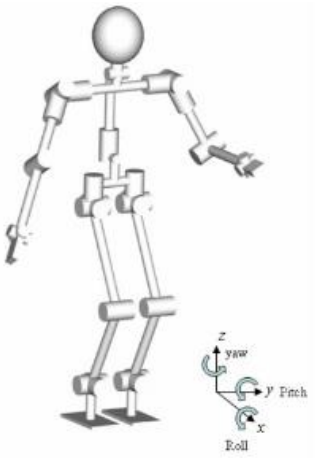


Figure 3.1. DOF for KHR-3 Robot

The next robot we looked at was HRP-2L, which also used a 6 DOF per leg set up. While HRP-2L is a bipedal, it doesn't fall into the category of humanoid due to the shape of its legs and the large mass at its hips, which is shown in Figure 3.2.

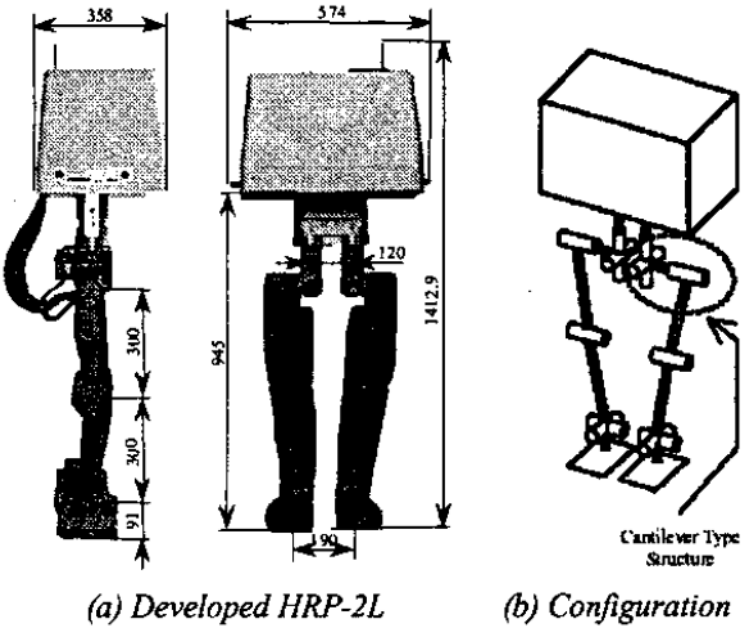


Figure 3.2. The HRP-2L robot with its configurations

The trend of using 12 DOF for the legs continued as we researched other robots. 12 DOF appeared to be the optimal number for mimicking human capabilities in a robot. For example, with the AUTOMI robot (pictured in Figure 3.3 below), the research paper states that they wanted their robot to be capable of walking and running, meaning their decision to go with 12 DOF is for achieving these capabilities (Varshney et al., 2019).

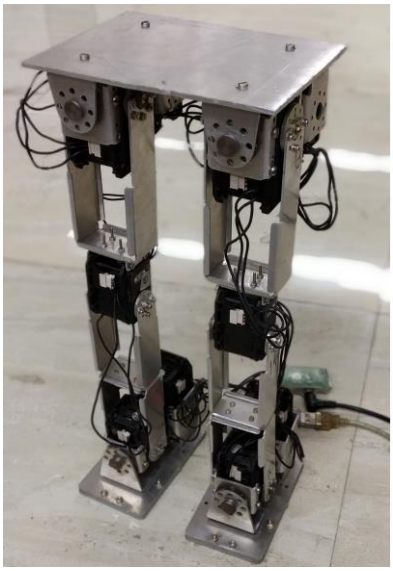


Figure 3.3. Image of the AUTOMI Robot's legs

12 DOF is backed further by the paper *Proposal Kinematics and Principle of Humanoid Robot GAIT*. Figure 2.4 shows an image taken from the paper that analyzes the motion of humans walking in terms of a robot. When taking a step, one leg maintains contact with the ground while the other leg must change position with touching the ground. In addition, the center of gravity does not stay over the supporting foot while walking, which prevents collapsing. The paper then proposes that if a robot needs to mimic human walking, the best DOF numbers to use is 3 for the hip, 1 for the knee, and 2 for the ankle (Świć, 2009).

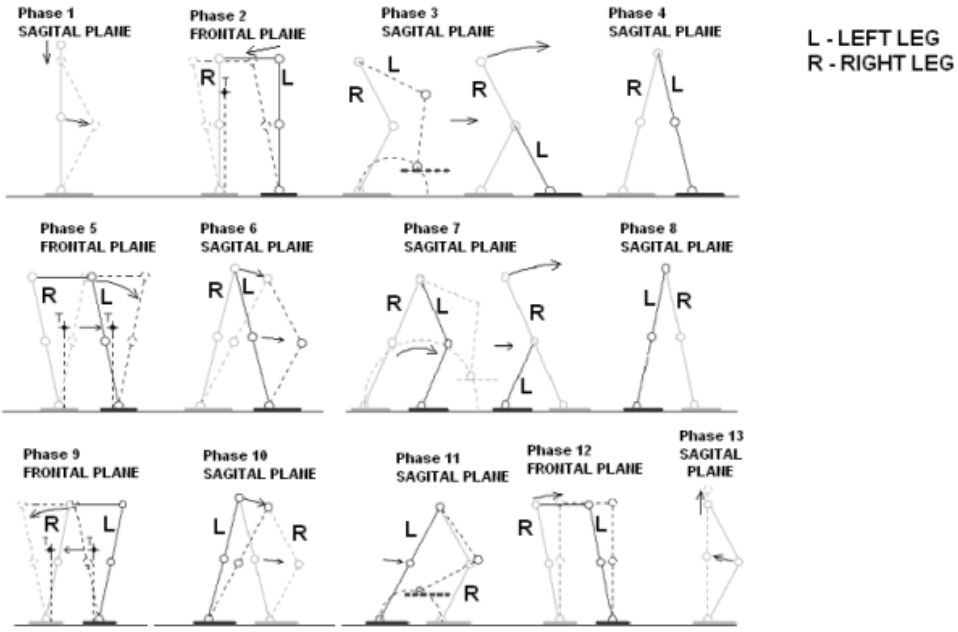


Figure 3.4. Simplified diagram of human walking motion

We also wanted to see if 3D printing robots made a difference in the DOFs. In some cases, it did not. This is shown in the NU-biped robot, which had 12 DOF of freedom, as shown in the left of Figure 2.5 (Folgheraiter & Aubakir, 2018). In contrast, the Poppy robot, another 3D printed robot, only utilized 10 DOF in their legs (M. Lapeyre et al., 2013). They did not have any ankle roll in their robot; however, it was not designed to be a full-scale humanoid robot as it was only 83cm tall.

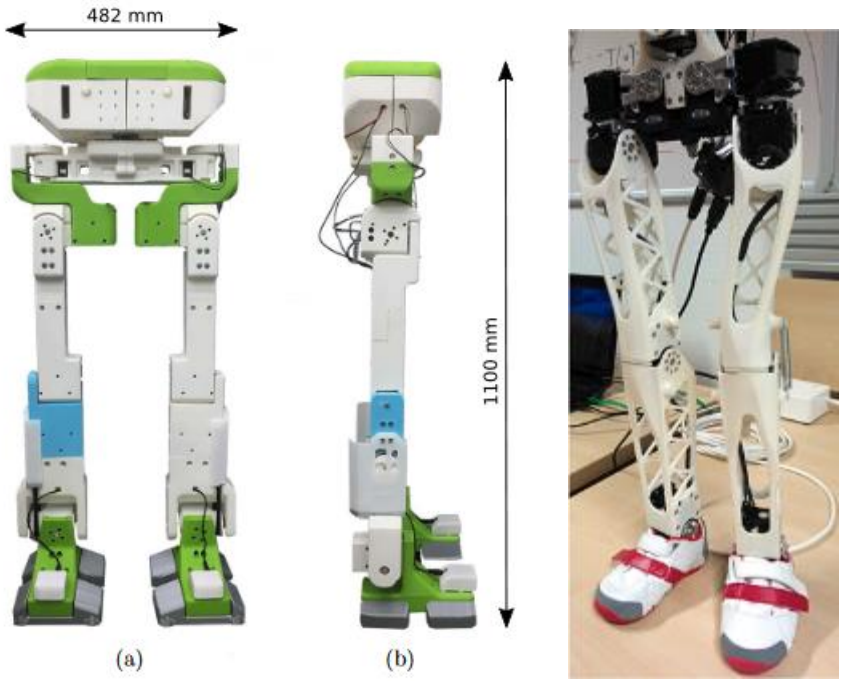


Figure 3.5. Design of NU-Biped robot (Left), Design of the Poppy Robot (Right).

Some robots used more than 12 DOF, with 7 DOF in each leg instead of the usual 6. One example is the WABIAN-2 robot, which adds a yaw to the ankle, as shown in Figure 2.6. The reasoning for the ankle yaw is a bit vague in the paper, however, they do mention that one of the goals of the robot was to push a walk-assist machine (Yu Ogura et al., 2006). Having an ankle yaw could provide extra balancing when doing a pushing motion, such as in the case of slippage, where an ankle yaw could help correct the ankle positioning.

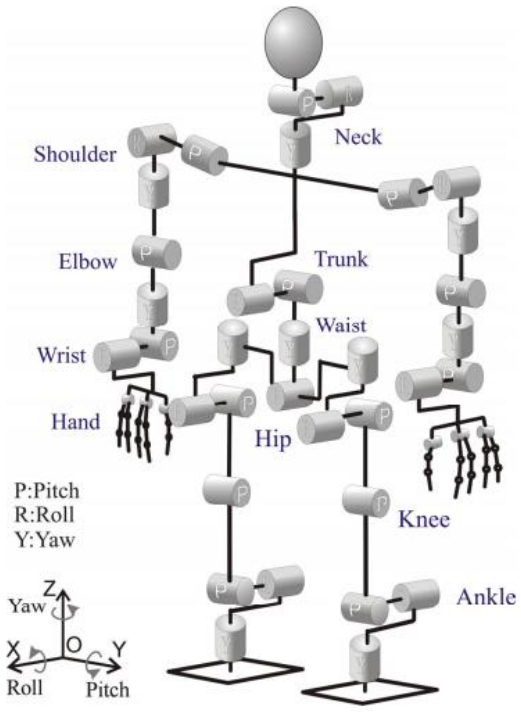
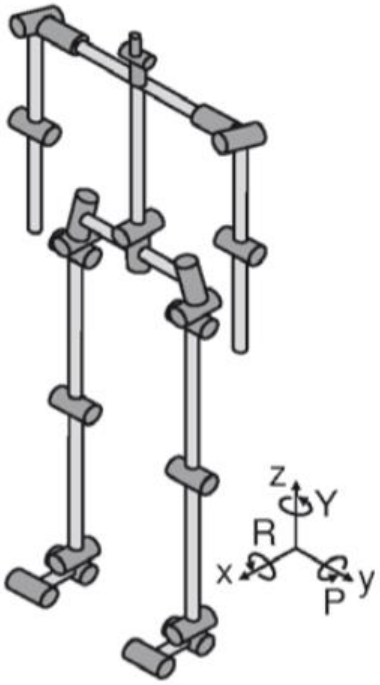


Figure 3.6. DOF of the WABIAN-2 Robot

Another robot that had 14 DOF was the Lola robot, but instead of adding to the ankle, it had actuation for the toes (Buschmann et al., 2009). When walking, the leg that is in motion can be described as the "swing leg", which influences the center of mass of the robot and is important for balancing. Having an actuated toe joint "allows the swing leg to be in a more extended configuration", which "stabilizes the robot and facilitates forward roll across the forefoot" (Buschmann et al., 2009). The DOF configuration is shown in Figure 2.7. It is worth noting that "very few humanoid robots [have] actively driven toe joints" despite the benefits it provides (Buschmann et al., 2009). However, based on the research done on other robots (discussed in more detail next), they function just fine without them.



Joint	DoF
Head	3
Shoulder	2
Elbow	1
Pelvis	2
Hip	3
Knee	1
Ankle	2
Toe	1
Total	**25**

Figure 3.7. The joint orientations and locations on Lola.

3.1.2 Leg Structure

Our interior leg structure was based primarily on the human bone structure, which are represented with rods in the robot. This design was shown in several of the papers we researched, such as the paper by M. Lapeyre et al. on the Poppy Humanoid Robot (M. Lapeyre et al., 2013). In their paper, they created a humanoid robot that had a 3D-printed mesh that bent inward at the thigh to mimic a humans femur bone, shown in Figure 2.8. For the poppy robot, they also looked at how the femur's motion changes during a normal walking gait with the center of mass changing with the bent inward femur, see Figure 2.9. Since the femur puts the knees and lower legs almost directly in line with the hips, it reduces the torque that is applied to the lower joints, which instead only must deal with the force of the human weight directed downward.

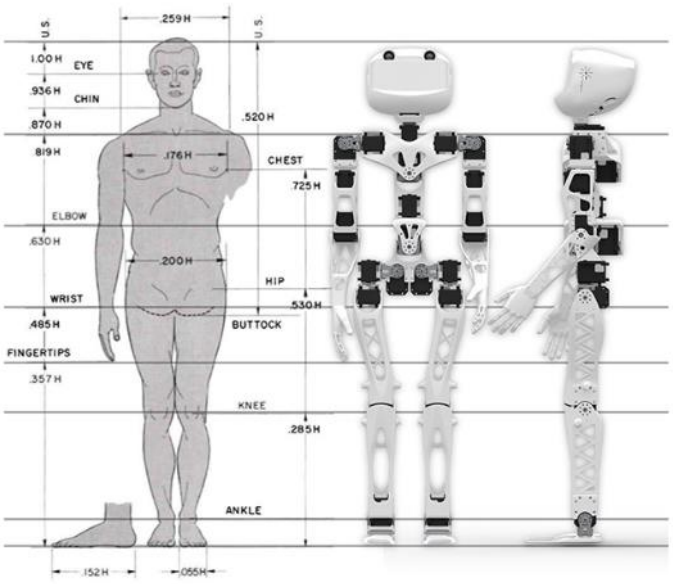


Figure 3.8. The Human proportions utilized for the design of Poppy.

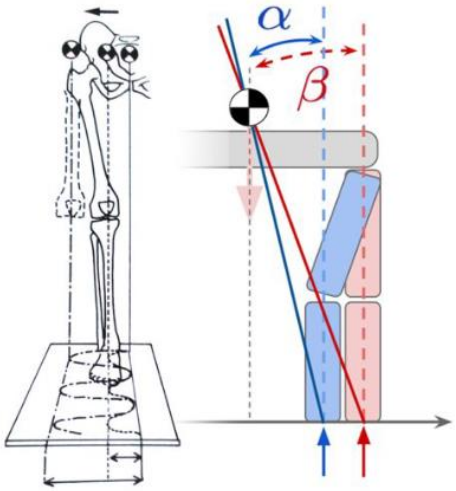


Figure 3.9. Human anatomy and the effect on the center of gravity during a walking gait vs the model dynamic model used for Poppy.

However, while the Poppy robot is a great example of trying to mimic a human with a robot, this robots total height was only .84 m, which is less than the average length of a humans lower body which is why we cannot utilize the same technique of primarily 3D printing the legs and using more cost effective motors (M. Lapeyre et al., 2013). Another point of which the Poppy

robot fails in the humanoid category is that the robot is based on only 10 degrees of freedom, where it lacks a roll joint for the ankle.

The next robot reviewed for leg structure was the NU-biped as pictured above in the left of Figure 2.5. This was another example of a lightweight robot, made with 3D printed parts, however, unlike Poppy, the NU-biped utilized the 12 DOF and is closer in human proportions with the legs measuring at 1.1 m with a weight of 10.8 kg without batteries (Folgheraiter & Aubakir, 2018). However, while this is ideally the size of human legs, the actual structure of the legs is very inhuman. There are very bulky calves and ankles, and the foot is a large square which is incapable of fitting into a shoe. Another negative of this design was the cost of the robot, the total estimated cost was approximately $20,000 (Folgheraiter & Aubakir, 2018).

To create their robot, they reviewed several pros and cons of different bipedal robots from the last few years with the most notable robots being the WABIAN-2, AISIMO, BHR-5, HRP-4, Poppy, HUBO, and Atlas (Folgheraiter & Aubakir, 2018). The main points this paper looked at was the height and weight of each of these robots, which range from a height of 0.85m to 1.62m tall including the upper body, the smallest the 3D printed Poppy robot and the tallest the BHR-5 (Folgheraiter & Aubakir, 2018). In terms of weight, the BHR-5 was the heaviest with a weight of 65.0 kg and Poppy was the lightest at only 3.5 kg (Folgheraiter & Aubakir, 2018).

The next robot reviewed was ERNIE, which is a simplified bipedal robot initially created to test the usability of spring joints at the knee (Yang et al., 2008). In terms of leg structure, ERNIE is inhuman and not a true consideration for the design of the leg structure, see Figure 2.10.

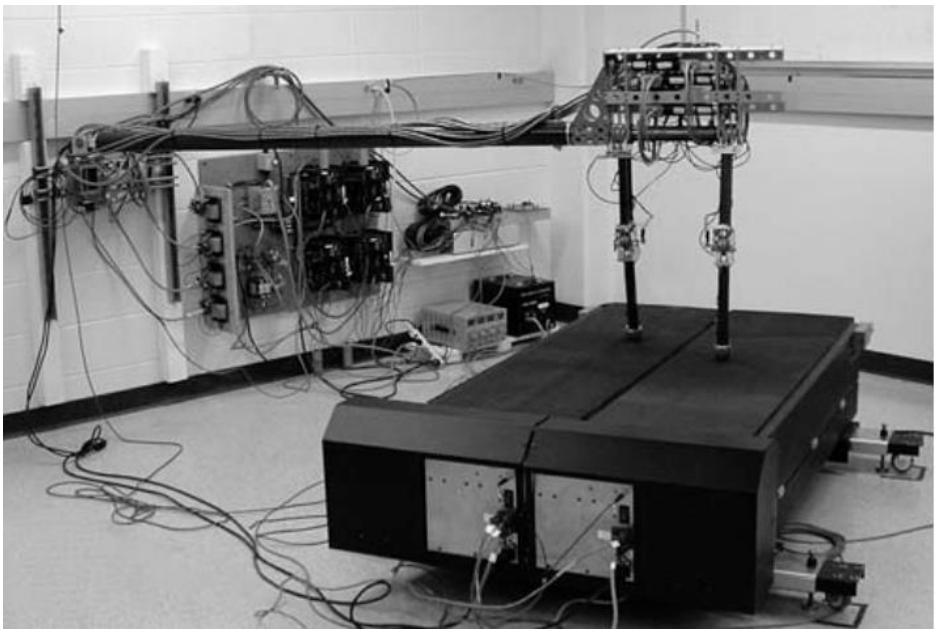
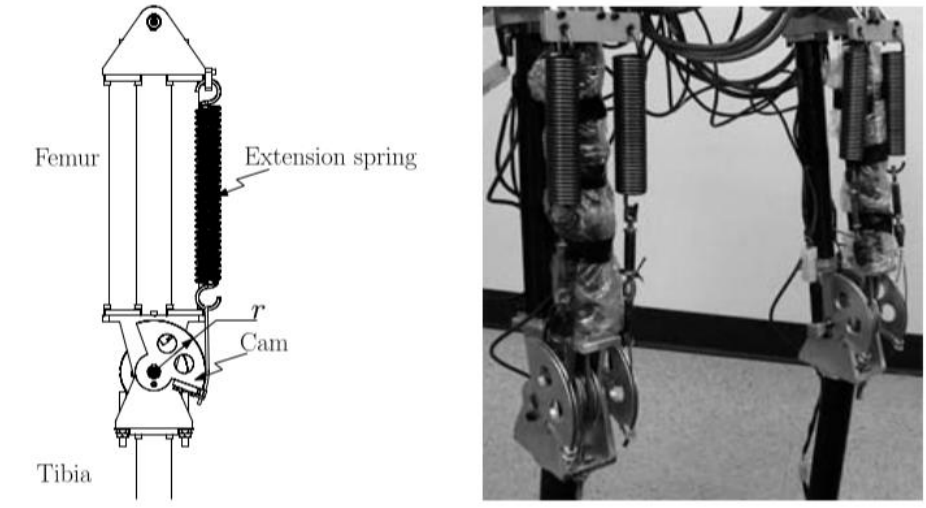


Figure 3.10. The biped prototype ERNIE with its experimental setup.

The interesting design idea that ERNIE poses is moving the center of mass further up the body as well as testing different compliance joints for the knee. The main principle that was tested for a compliant joint was using springs with different spring constants. They tested 3 different spring constants and one test without a spring, and shows the idea behind the joint more in Figure 2.11 (Yang et al., 2008).



(a) Schematic of springs in parallel with knee joint actuators

(b) Actual assembly of springs in parallel with knee joint

Figure 3.11. The schematic and actual assembly of the parallel spring knee joint.

However, while this was an interesting concept, this joint set up would not be able to hold the weight of the robot since ERNIE was very lightweight due to it being an inhuman design. It had carbon fiber rods and a lack of ankles or feet since they were only focusing on the knee design.

Next, we investigated the humanoid robot Lola, which was a full humanoid robot with an upper body designed to walk, see Figure 2.12. The basis of this project was to create a humanoid robot and based on the report and the images included within, their design was humanoid but also more expensive than the budget we are working with. This project was one of the best starting documentations for HURON, and the main take away from this project was the shaping and sizes of the legs (Buschmann et al., 2009).

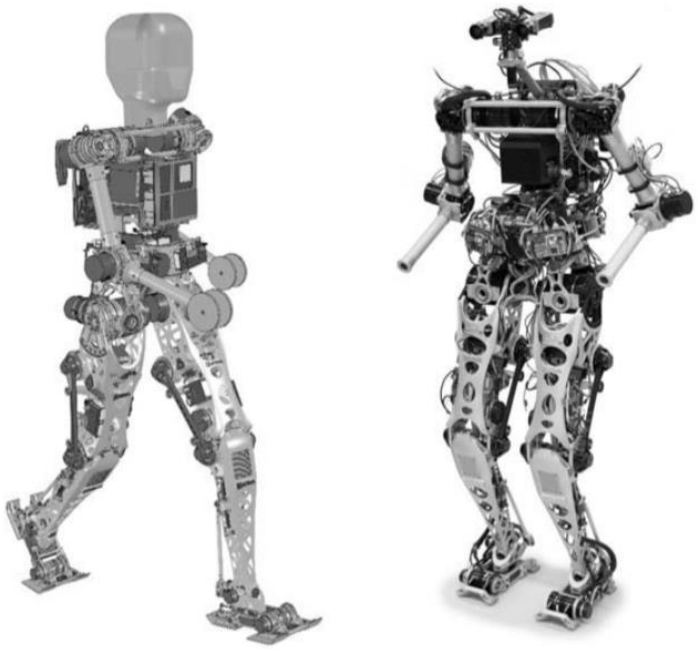


Figure 3.12. The simulated and physical set up of the humanoid walking robot Lola.

From the overview of the leg structures, we then looked more at the individual sections of the leg to see what they used and how it could be applied to our project. We discovered that they used timing belts and linear carriages for the lower limbs, see Figure 2.13. The other unique aspect of Lola was they created an actuated toe joint to help mimic a human gait more effectively, more specifics on the actuated toe are shown in Figure 2.14. They also utilized having a center of mass (COM) that resided high up the body, which being one of many papers to mention that having the COM higher was beneficial, really encouraged us to move as much mass as possible higher up the body (Buschmann et al., 2009).

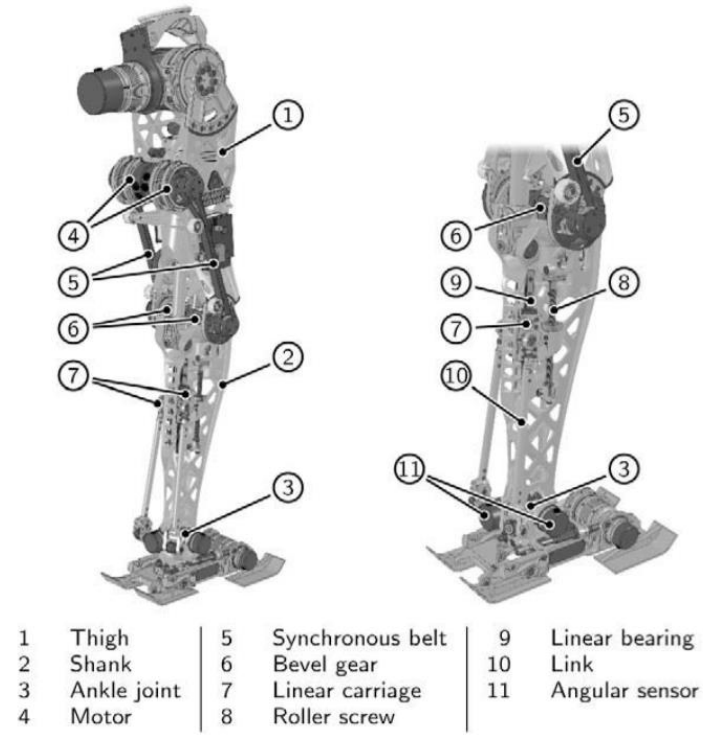


Figure 3.13. The ankle actuations of the Lola.

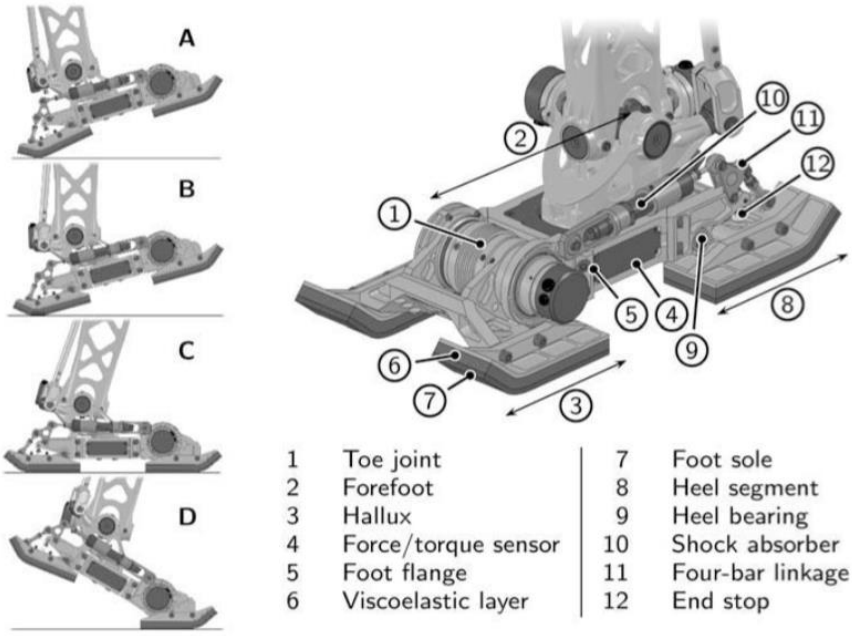


Figure 3.14. The foot design of Lola with an actuated toe.

In the HRP-2L robot (from Figure 2.5), they created a leg module set up which was based more on a cantilever structure. This allowed them to angle the legs inward reducing the torques on each of the lower limbs.

The second idea that we found interesting from this project, was the setup of their ankle and foot. Their foot pad was designed for uneven terrain and allows for the foot to tilt in several directions while also utilizing a six-axis force sensor to identify how it should balance, see Figure 2.16. They also included more information about the dimensions and the range of angles each joint could reach, which didn't mimic a human's range very closely, but gave a starting idea of the considerations necessary for designing the joints, see Figure 2.15.

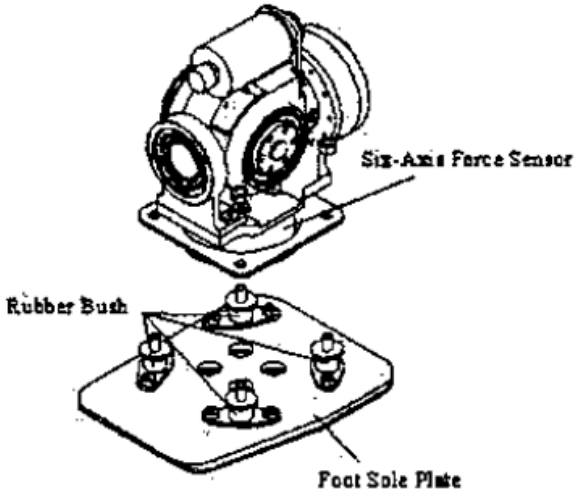


Figure 3.15. The mechanism for the foot to deal with rough terrain.

***Table 1:** Dimensional Specifications of Leg Module*

Legs	6 D.O.F. / Leg (Hip: 3 Knee: 1 Ankle: 2)	
	Upper leg length:	300 [mm]
	Lower leg length:	300 [mm]
	Ankle length:	91 [mm]
	Length between hip joints:	120 [mm]
Weight	Legs: 8.6 [kg/leg] × 2 [legs] =	17.2 [kg]
	Batteries:	11.4 [kg]
	Controller Box:	7.0 [kg]
	Dummy Weights:	22.6 [kg]

***Table 2:** Working Angle of Leg Module*

Hip	Yaw	-45 deg. to 45 deg.
	Roll	-45 deg. to 20 deg.
	Pitch	-135 deg. to 42 deg.
Knee	Pitch	-0 deg. to 150 deg.
Ankle	Pitch	-100 deg. to 42 deg.
	Roll	-20 deg. to 35 deg.

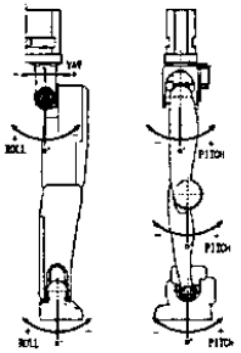


Figure 3.16. The dimensions and joint angles are established for the HRP-2L legs.

The next major section of the leg structure we considered was the foot design, and we found a paper detailing how one group created a flexible foot, and this generated a few initial ideas of how we would create the base support (Jianxi Li et al., 2008). This foot was unique, as it passively absorbs impact force when the robot is walking, and most of the absorption was done by the rubber bushes and pads on the bottom of the foot, see Figure 2.17 for exploded view and Figure 2.18 for collapsed view.

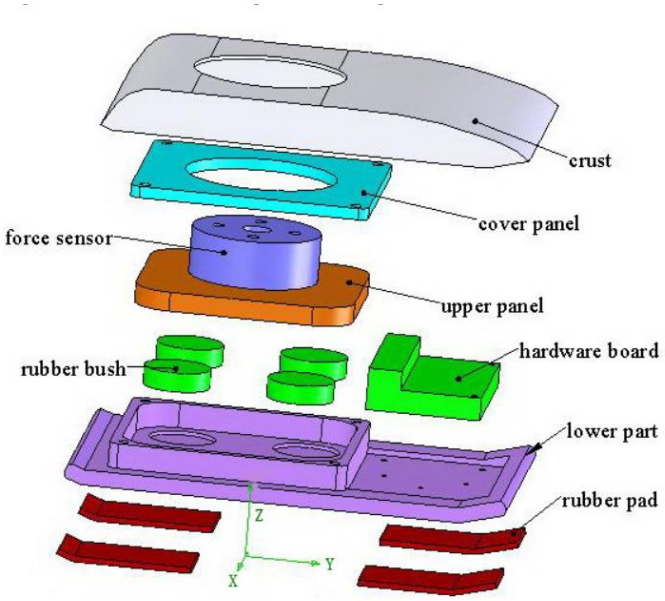


Figure 3.17. The flexible foot of the BHR-2 humanoid robot with parts exploded.

The casing they put around the main components was a rigid material, while the actual plate of the robot foot was stainless steel (Jianxi Li et al., 2008). They also noted that the toes section should curve upward to help stabilize on the ground even when the robot is taking a step and the heel begins to lift.

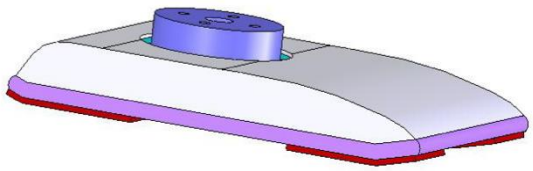


Figure 3.18. The flexible foot of the BHR-2 humanoid robot.

The next robot we researched into for leg struct was the KHR-3 robot, which has an overall human shape, but the actual leg sections are less humanoid but still functionally similar, see Figures 2.1 and 2.19 (Ill-Woo Park et al., 2005). Again, one of the main takeaways from this experiment was the need to make the center of gravity higher on the robot, with most of the weight centralized at the hips. This robot had an overall height of 4.1 ft, which is still over a foot shorter than average height and the joints were mainly actuated by use of a pulley-belt mechanism.

Figure 3.19. The humanoid robot KHR-3.

The last robot we reviewed was the AUTOMI, which was more humanoid, while also still being fairly blocked out and robotic, the physical build of the robot and the simulated robot with the upper body are shown in Figures 2.3 above and 2.20 below. This robot was also short for a human, with legs only measuring 72cm (Varshney et al., 2019). This robot was primarily made of aluminum for the frame, while the connections and fasteners were made from mild steel. One of the unique parts of the design here was the addition of four rectangular attachments to the bottom of the foot to prevent the robot from becoming flat-footed, as contact forces in the center of the foot are not ideal.

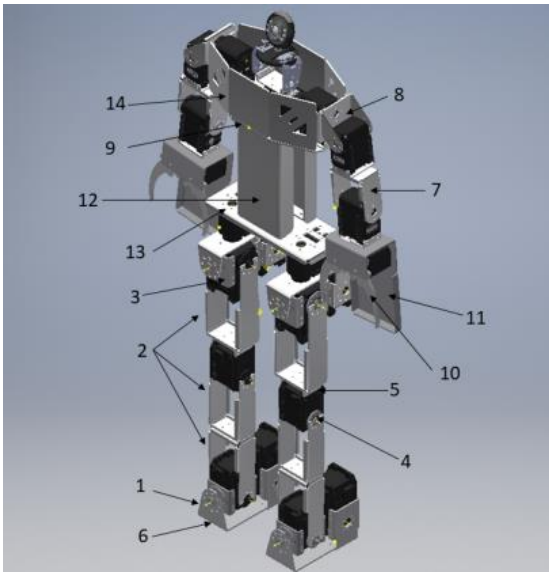


Figure 3.20. The labeled parts of AUTOMI in the CAD environment.

3.1.3 Joint Orientations and Motion

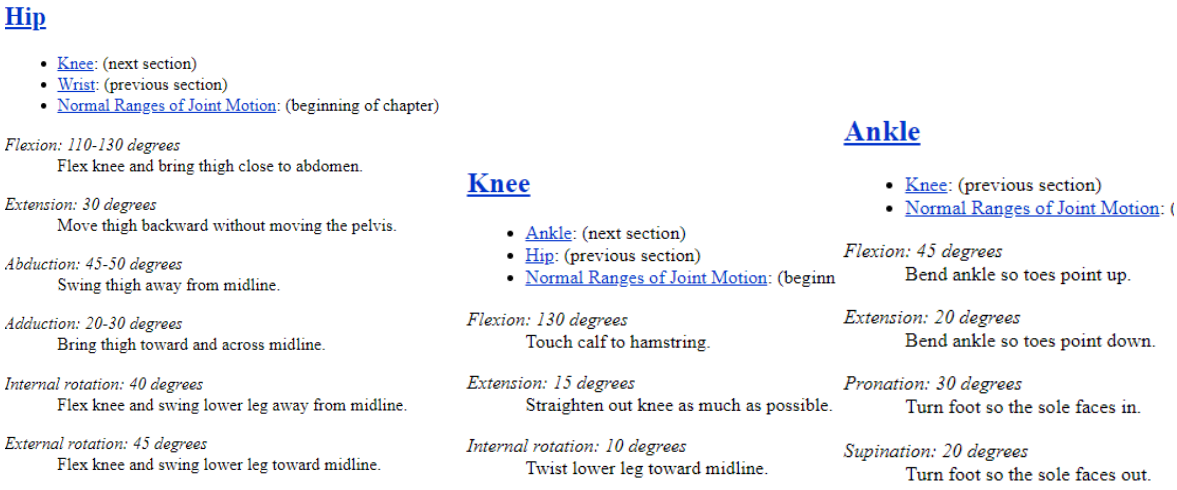
The joint orientations were a concern addressed through additional research because the orientation is a huge contributor to the calculations for controlling the robot. For more straightforward control implementation, having the axis of orientation of each joint for the three human joints, hip, knee, and ankle, intersect at each location was vital. Due to this consideration, several different iterations were considered to account for the rotation of each joint.

An important aspect of the joints was knowing the necessary range of motion of every joint. The *Design and Control of a Humanoid Robot* has a table showing the degrees of rotation they used which is pictured below (Chew, n.d.). They describe these angles as “similar to a human” but do not go into further detail on the angles. However, from a first look, the angles do look reasonable based on what humans are capable of. For example, we can bend our knees far inwards, but cannot bend outwards. The ankle angles also reflect this as we can pitch our ankles downwards more than up. Also, ankle roll does not need to be a lot considering that we lose balance early on by just bending our ankles side to side.

Hip	Roll Pitch Yaw	-60° to 45° -130° to 45° -90° to 60°
Knee	Pitch	0° to 150°
Ankle	Roll Pitch	-20° to 30° -30° to 80°

Figure 3.21: Example of angle rotations of a robot

As a comparison, we researched what normal human range of motion is for the joints in the legs. A research paper from MIT shows the average range of motion based on a study conducted, although they use different terminology than "roll", "pitch", and "yaw" (Appleton, n.d.). The degrees here are like the degrees of the robot above, with some difference of a few degrees. One interesting thing to note is that the MIT paper states that extending the ankle is 20 degrees and bending the ankle up is 45 degrees, which is odd since it feels like the ankle pitch down should be more than up pitch. However, Figure 2.22 reflects this idea, so there could be some difference in research.



Hip

- Knee: (next section)
- Wrist: (previous section)
- Normal Ranges of Joint Motion: (beginning of chapter)

Flexion: 110-130 degrees
Flex knee and bring thigh close to abdomen.

Extension: 30 degrees
Move thigh backward without moving the pelvis.

Abduction: 45-50 degrees
Swing thigh away from midline.

Adduction: 20-30 degrees
Bring thigh toward and across midline.

Internal rotation: 40 degrees
Flex knee and swing lower leg away from midline.

External rotation: 45 degrees
Flex knee and swing lower leg toward midline.

Knee

- Ankle: (next section)
- Hip: (previous section)
- Normal Ranges of Joint Motion: (beginn

Flexion: 130 degrees
Touch calf to hamstring.

Extension: 15 degrees
Straighten out knee as much as possible.

Internal rotation: 10 degrees
Twist lower leg toward midline.

Ankle

- Knee: (previous section)
- Normal Ranges of Joint Motion: (

Flexion: 45 degrees
Bend ankle so toes point up.

Extension: 20 degrees
Bend ankle so toes point down.

Pronation: 30 degrees
Turn foot so the sole faces in.

Supination: 20 degrees
Turn foot so the sole faces out.

Figure 3.22: Average Human Range of Motion

3.1.3.1 Hip Joint

The hip joints typically have 3 DOF by rotating in yaw, roll, and pitch, making it one of the more challenging designs of the leg. One of the most interesting parts of the NU-biped robot was the hip set up they chose to implement which created a compact area of the hip with all of the

joint axes intersecting, which is shown below in Figure 2.23. Having all the axes intersect helps with the theoretical part as it simplifies “the solution of the leg inverse kinematics” which “guarantee[s] a closed-form solution” (Folgheraiter & Aubakir, 2018).

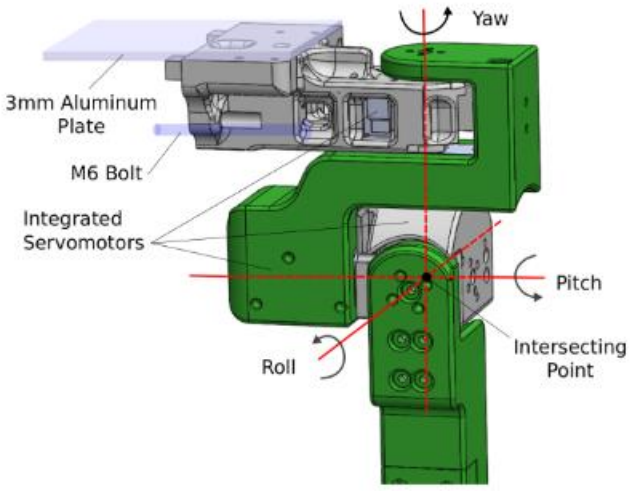


Figure 3.23. The Hip mounting of the NU-biped project, with intersection hip axes.

Another solution for getting the joint axes to intersect was having the hip yaw at an angle. Figure 2.24 shows how the Lola robot has their hip joints arranged. The roll and pitch are intersected by having the motors placed on the same plane. What’s interesting about that is it means the roll motor has to stick out the back of the leg, which does not seem ideal in terms of looking like a human. What is compact is having the yaw motor at an angle, which allows the axis to intersect with the other two, as shown by the red lines in the figure. This helps with having a smaller width in the hip, which is ideal for looking like a human (S.Lohmeier et al., 2006).

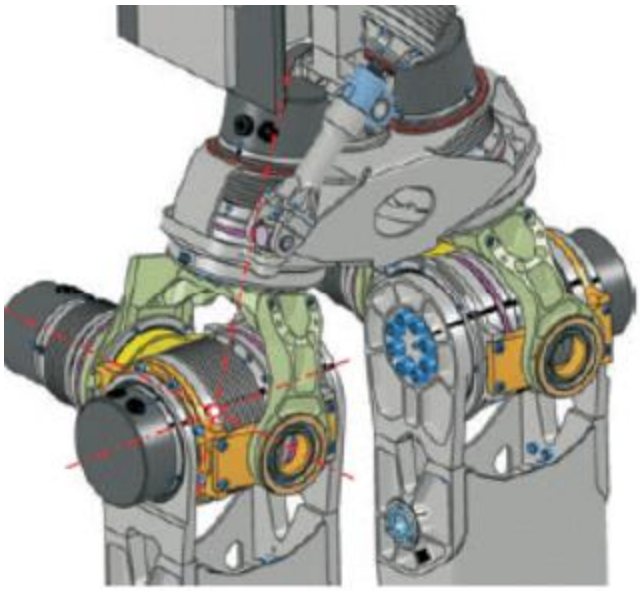


Figure 3.24: Hip Joint of the Lola

The KHR-3 Robot also had intersecting axes, but instead had the hip roll and yaw positioned on the same place instead of the pitch and roll (Ill-Woo Park et al., 2005). Unlike the Lola robot, the roll motor doesn't stick out that much due to using a compact design. Since the pitch motor is below the other two axes, a belt system was used to connect between the hip actuators.

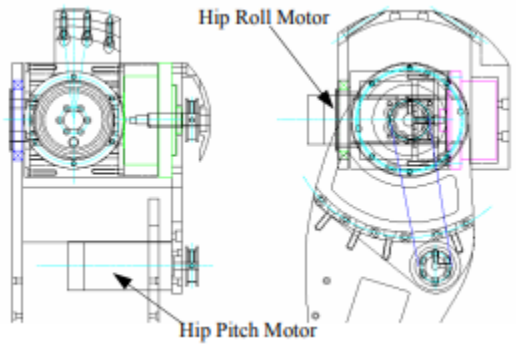


Figure 3.25: KHR-3 Robot Hip Axes

The AUTOMI robot's hip design seems like one to avoid because it is bulky. Referencing back to Figures 2.3 and 2.20, the hip joints do still intersect, but they are attached to a large metal sheet to act as the pelvis of the robot (Varshney et al., 2019). The issue comes from the actuators they use because they are big, meaning the brackets that are holding them need to be large. This takes up a lot of space, so while the design does work, it wouldn't be ideal for a humanoid robot.

3.1.3.2 Knee Joint

As mentioned in Chapter 2.1 (Degrees of Freedom), the knee only has pitch as the axis of rotation, leading our focus to be more on the design structure of knee joints. One of the first knee designs, shown in Figure 2.26, featured a motor and a harmonic drive in a "stacked" configuration (Chew, n.d.). The motor turns the harmonic drive through a pulley system which then turns the bottom bracket along the pitch axes. As for structural integrity, there are multiple brackets to support the weight of the robot above, but those could easily be modified to our needs.

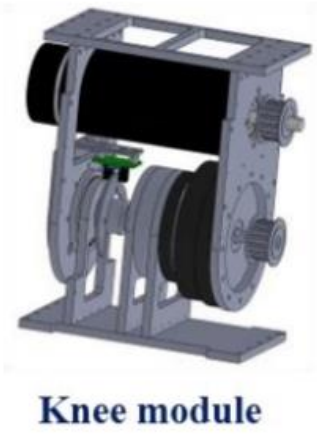


Figure 3.26: Knee Joint Design from (Chew, n.d.)

There were not many variations with the knee due to only having a pitch rotation. In addition, the design fit with what we were going with because of our budget restraints, which will be discussed more in the Design Process chapter. That being said, we did look at the knee design for the NU-Biped robot (Figure 2.5), which was more compact than the design above due to their actuation choice of servo motors (Folgheraiter & Aubakir, 2018). Considering that the robot was mainly 3D printed, these motors worked for this robot. Other robots also used one actuator and no external gearbox without being 3D printed. The AUTOMI robot discussed previously had a similar knee design with the NU-Biped, but the motor they used was bulky because it included a "Reduction Gearhead + Controller + Driver + Network" (Varshney et al., 2019). Essentially, they both are good designs but only if there was budget for expensive actuators.

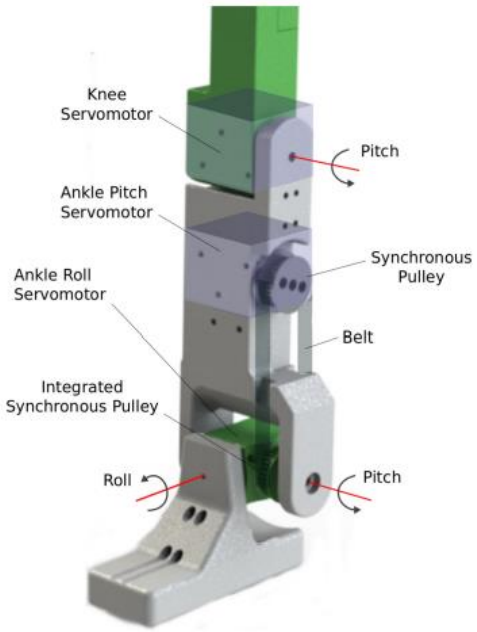


Figure 3.27: NU-Biped Knee and Ankle Joint

There was one case we saw with the KHR-3 Robot where they used 2 motors for the knee "because [the] knee joint actuator needs high speed and torque" when bending the leg (Ill-Woo Park et al., 2005). This is an interesting choice as having two motors for the pitch make the robot bulkier. In addition, most of the robots looked at can accomplish the knee pitch through just one motor. That being said, having two motors did allow for their knee to bend faster, which is a benefit when considering how humans walk.

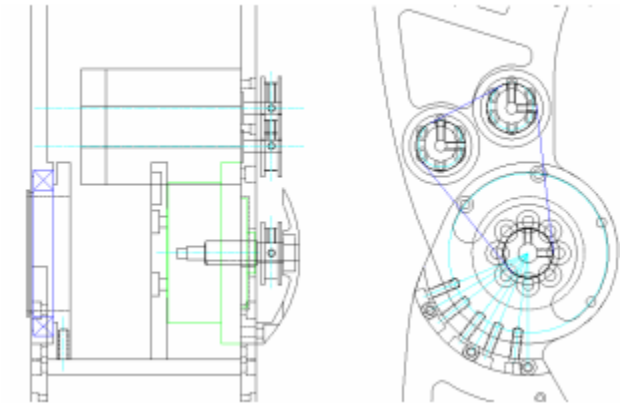


Figure 3.28: KHR-3 Knee Design

Some robots also used linear actuators for the knee joint. Figure 2.29 shows one example of a linear actuator being used for the knee in the Lola robot, which depicts a “muscle-like” mechanism. Some of Lola’s joints, including the knee and ankle, use a ballscrew drive, which has a “high efficiency, no backlash, no stick-slip and silent operation” They compare this to harmonic drives, where they note that while ballscrews are more efficient than harmonic drives, they are more complicated due to the surrounding construction being affected (S.Lohmeier et al., 2006).

The issue with linear actuators is their cost. For example, using ballscrew drive linear actuators, like the one used above, is typically a few hundred dollars. Technically there are cheaper ones, but the issue is that their quality may not be reliable.

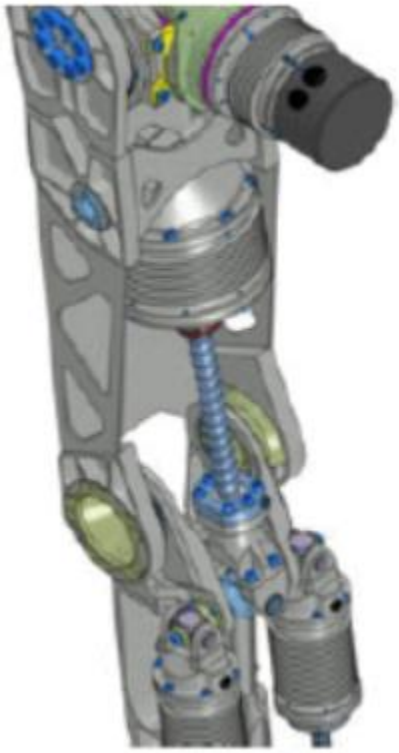


Figure 3.29: Linear actuator knee joint

3.1.3.3 Ankle Joint

Going back to the NU-biped robot from Figure 2.5, it had the roll and pitch axes intersect by using a pulley system (Folgheraiter & Aubakir, 2018). This involved putting the pitch motor right below the knee, where it connected to the ankle via a pulley system. Doing this prevents the need for a bulky ankle as having both motors in the ankle would take up a lot of space. This way, there was space for the roll motor to be put in the ankle.

This pulley system is also used in the KHR-3 robot, as shown in Figure 2.30 (Ill-Woo Park et al., 2005). However, there is a slight difference with the placement of the roll motor, as in this robot the roll motor is placed above the axis of rotation. In the paper, they explain their reasoning for this as not wanting to "transfer heat to the F/T sensor" because the sensor they used was "sensitive to temperature variance" (Ill-Woo Park et al., 2005).

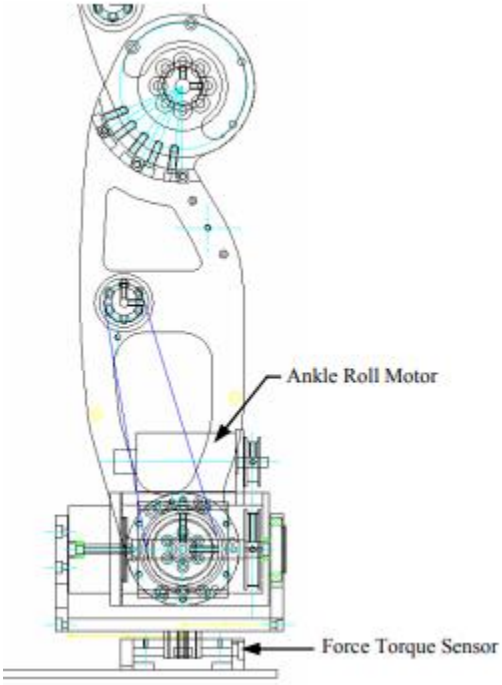


Figure 3.30: KHR-3 Ankle Design

Another design had both the ankle and roll motors much closer to each other. In the *Design and Control of a Humanoid Robot*, they also used a belt and pulley system, but instead of having the roll motor further up in the leg, they mounted it right above the pitch motor, as depicted in Figure 2.31 (Chew, n.d.). This design is interesting due to the way their robot is attached in general. Looking at the right picture below, the connection between the knee and the ankle is a large U-shape bracket. Both the ankle motors are attached to another bracket that extends to the foot to allow for the pitch and roll movements. Unfortunately, this makes for a bulky design because of the spacing needed between the brackets to fit both motors.

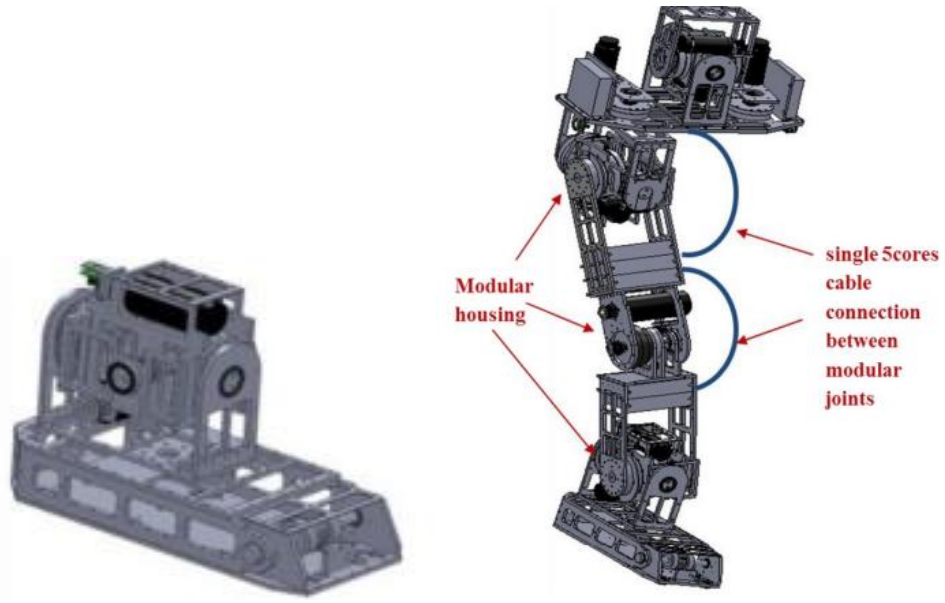


Figure 3.31: Ankle design with bracket attachments

Another design concept researched was with using linkages to actuate the roll and pitch. Going back to the Lola robot (Figure 2.13), the ankle is actuated by a motor near the knee, but it is connected to a link that attaches to the back of ankle. Therefore, when the linkage is moved down, the foot will pitch upwards, and vice versa. There is still a motor near the foot to do the roll, but having the linkage do the pitch actuation is cool design choice.

The HERMES robot used a similar design, as shown in Figure 2.32 (Wang et al., 2015). Instead of a large linkage, they use a thing rod which is attached by a screw to the back of the ankle. While that does not provide a lot of torque, the HERMES robot also was not human sized and consisted of some 3D printed elements, which cuts down on the weight.

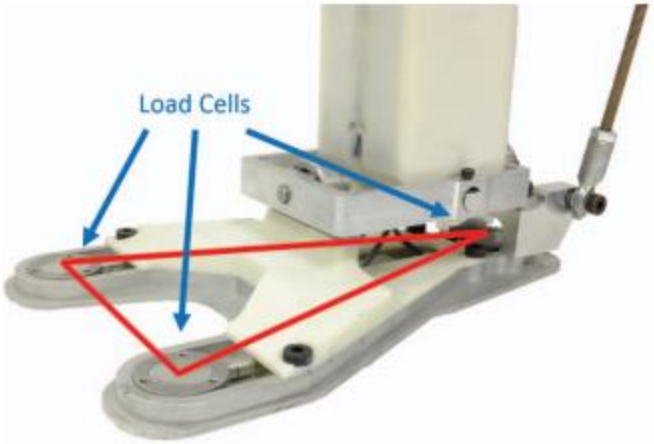


Figure 3.32: HERMES Ankle Design

Oregon State had an interesting robot we looked at in terms of the ankle. Their robot, Cassie, was built to run long distances, and they utilized a fourbar mechanism that starts from the hip as opposed to under the knee, as shown in Figure 2.33 (Oregon State University, 2021). While this is a cool concept, the issue is that this robot was not designed to be humanoid. They designed this robot to be fully optimized for running, which means not designing to look like a human. For example, the knees bend inwards instead of out, which is what is commonly used on other bipedal robots that are meant for speed.

Figure 3.33: Screenshot of Cassie from Oregon State's video

3.1.4 Human Leg Bones

Since the goal of the project is to make a robot as humanoid as possible which guided our research on human bone structure. There are three main bones that we consider, the femur, the fibula, and the tibia. The femur is the bone in the thigh while the tibia and fibula are the bones in the calf. We found that the length of the average fibula was 16.62 cm with a diameter of 1.227 cm, the tibia had dimensions of 16.62 cm in length and 3.1 cm in diameter, and the femur had dimensions of 24.9 cm in length with a diameter of 3.81 cm (Aitken, 2021; Cristofolini & Viceconti, 2000; Peter, 2019).

Another consideration with mimicking bone structure was the tilt that is featured in the femur. The tibia and fibula are vertically aligned in the calf, but the femur connects to the hip and slants inward to connect to the knee. When researching this tilt, which is called the Q Angle, we found that the Poppy robot utilized a 6-degree tilt; however, based on the dimensions of the robot we decided to proceed with a required tilt of 10.27 degrees, which is similar to the recommended tilt of a femur for men which is 13 degrees (Carreiro, 2009; M. Lapeyre et al., 2013). This degree angle would be chosen if it wouldn't cause interference between the knees; however, to give the knee joints enough clearance for motion, only 10 degrees were used.

3.1.5 Harmonic Drive

Based on the research into other bipedal humanoid robots, we noticed that many of them used harmonic drive gearboxes at the joints. The reason being is that harmonic drives can increase torque ratios drastically, are more compact than other gearboxes and are also not back drivable which is extremely useful as this means the robot does not have to be constantly powered on to remain standing (*Harmonic Drive® Strain Wave Gear - Zero Backlash | Harmonic Drive | Harmonic Drive*, n.d.). A harmonic drive, also known as a strain wave gear, operates by having an internal circular gear that meshes with a flexspline gear that is deformed into an ellipse by a wave generator, see Figure 2.34. When the wave generator spins, this changes the deformation of the flexspline which then meshes with different sections of the internal circular gear (Chen et al., 2019; Dabhi et al., 2019). This works because the internal circular gear has two more teeth than the flexspline, meaning when the flexspline rotates it skips a tooth creating a reduction ratio of half the number of teeth of the flexspline (Dabhi et al., 2019; How to Mechatronics, 2020; Kondo & Takada, 1990).

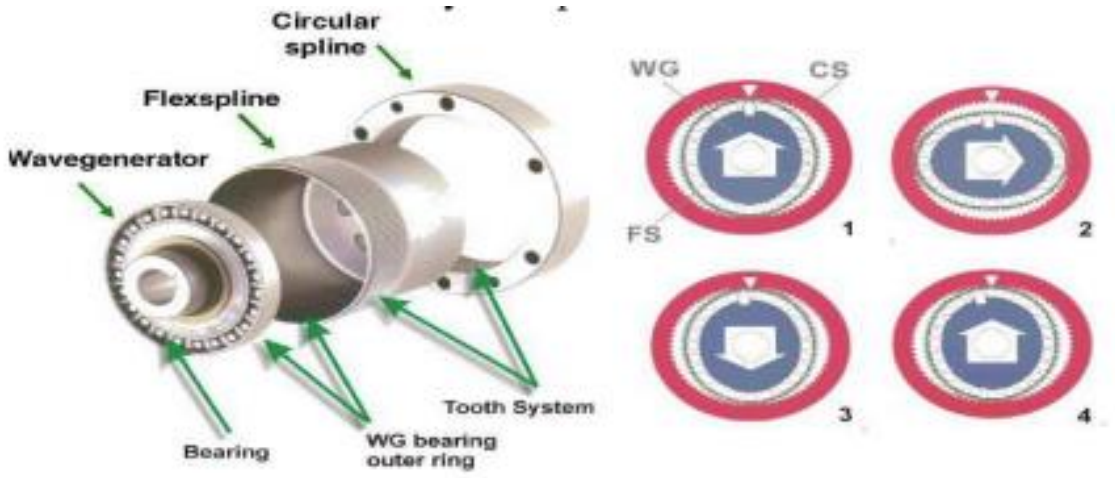


Figure 3.34. An exploded view of the main parts of a harmonic drive (left) with a sequence of how the wave generator deforms the flexspline (right).

With the initial idea of utilizing, we investigated adding harmonic drives to the project, but the drives available to purchase are in the realm of $1000 each. Based on the limitations of budget and the availability of 3D printing, we then investigated how to create our own harmonic drive. This involved researching the tooth profiles necessary to correctly mesh the gears, looking into the differences of modulus and pressure angles, which control how big the harmonic drive is as well as how (Chen et al., 2019; How to Mechatronics, 2020; Kondo & Takada, 1990; Ma et al., 2018). Based on the initial research, we were able to 3D print a version of a harmonic drive and testing with was continued further in iteration 1.

3.2 Humanoid Robot Control

The robot's actuation was key for it to be able to not only handle external forces but also react accordingly. Thus, the control of the humanoid robot can be divided into two main components and multiple sub-components:

1. Calculating forces for fundamental stability
2. Actuation to react to said forces:
    1. Motors
    2. Motor controllers/External encoders
    3. Power supply
    4. Computing

3.2.1 Forces

The possible external forces acting on HURON would be any kind of pushes that we may give it in order to fully test and demonstrate its reactive balancing capabilities. The internal forces on each joint have been thoroughly researched in one of the research papers that we found. Table 2 below lists the torques and speeds for each joint with a maximum / minimum moment arm (Hyon et al., 2017).

TABLE II
EXPECTED JOINT SPECIFICATION (14 MPA PRESSURE SUPPLY)

Joint	RoM [°]	Torque [N-m] @14 MPa supply	Velocity [°/s] @7 MPa load
TFE[1]	−42/54	645/302	397/947
TAA[1]	−15/15	445/203	586/1026
TR	−25/25	291/210	448/497
HR[2]	−20/20	330/223	585/397
HAA[2]	−40/20	294/190	444/548
HFE	−100/24.5	262/116	497/978
KFE	0 / −125	232/95	561/978
AFE[3]	−50/35	540/430	483/913
AAA[3]	−14.5/14.5	420/203	616/1026

Note: See Fig. 2 for the joint name convention. Joints with superscripts are coupled kinematically. The torques and velocities are evaluated respectively at the joint postures of maximum and minimum moment arm. See text for details.

Using this table above and the rest of the research article, we were able to come up with preliminary torque requirements needed from each joint in HURON which were used until a more detailed design was made later on. The hip needed 72 Nm, the knee needed 30 Nm, and the ankles needed 102 Nm. We used this minimum torque requirements to select the motors that were mentioned in the section below, 3.3.

3.2.2 Actuation

3.2.2.1 Motors

Common motor types used for humanoid robots are brushless dc (BLDC) (Buschmann et al., 2009; Jerry E. Pratt & Benjamin T. Krupp, 2004; Lee et al., 2014) and servomotors (Jerry E. Pratt & Benjamin T. Krupp, 2004; Lee et al., 2014; Tar & Veres, 2006; Yi et al., 2014). Stepper motors are also an option as introduced in (Tar & Veres, 2006). Direct drive is often not the choice for full-size humanoid robots, which means that the motor we plan to use must be compatible with the actuation mechanism the design team chooses. The discussed mechanism was a harmonic drive with a 100:1 gear ratio, which is subject to change as the project progresses. This means that we

can prioritize high speed motors over high torque motors. With this mechanism in mind, to select our motor type, we researched the advantages and disadvantages of each.

For servomotors, they are very inefficient at low speeds and high torque (Jerry E. Pratt & Benjamin T. Krupp, 2004). In addition, they can have jitter due to their closed-loop feedback design and are designed for holding positions ("DC Motor vs Stepper Motor vs Servo Motor - Which to Choose?," 2019). This is not ideal for our bipedal robot as it will require smooth movement and many continuous rotations for the actuation mechanism. This left the team with stepper motors and brushless dc motors. As stated in (*Brushless DC Motor vs. Stepper Motor*, n.d.) and ("DC Motor vs Stepper Motor vs Servo Motor - Which to Choose?," 2019), we can see that stepper motors have high torques at slow speeds. However, they can't maintain that torque across faster speeds and are extremely inefficient. In contrast, BLDC motors has both high speeds and high torque at all speeds. They are designed for high speed rotation, with speeds far exceeding stepper motors, and also come in a multitude of sizes which makes it easier to fit onto the robot. Therefore, ideally the team uses BLDC motors as it fits the use-case best. However, the biggest downside to BLDC motors is that they are very expensive. For example, the BLDC motors used on (Lee et al., 2014) for the hip actuators were about $780 per motor (*Online Shop for High Precise Drive Systems by Maxon | Maxon Group*, n.d.). Stepper motors are far cheaper as they are generally below $100, but they are geared more towards very high torques for very low speeds. The team explored the possibility of lowering the gear ratio of the actuation mechanism to accommodate for stepper motors as a backup choice.

3.2.2.2 Motor Controllers/External Encoders

The wide range of available motors require different types of motor controllers. For the two possible motor technologies we could use, stepper motors and brushless DC motors, there are two different motor controllers.

Stepper motors require a specific stepper motor driver to run. For high current applications evolving stepper motors, the driver's are often rather large. For example the CL86RS by Stepperonline has an 8A current limit, measures 5.9"x3.8"x2.0", and only controls 1 motor (*Brushless DC Motor vs. Stepper Motor*, n.d.). Some of the more expensive stepper drivers offer closed loop control through an added encoder in the stepper motor, to mitigate step loss in the motor.

For brushless DC motors there are several options. The most common option is and ESC, or electronic speed controller. These devices are often less expensive, however they have some severe shortcomings. These controllers only control the speed of the motor, and don't track position. Some more advanced ESCs have the ability to track position data using a hall effect sensor, but this is merely to smooth up the movement of the motor and does not accurately track the position.

Another brushless DC motor controller is the ODrive 3.6 by ODrive Robotics. The ODrive provides accurate position, velocity, and torque control with the help of an external encoder (*ODrive*, n.d.). Each controller controls two motors and measures 5.5"x2.0". These controllers have several communication interfaces including USB, UART, and CAN. Similar low cost robotics projects used the ODrive BLDC motor controller ("OpenDog Electronics & Initial Code," 2018).

The ODrive requires an external encoder to track the position of the motor. Rotary encoders, and magnetic encoders are both valid options supported by the ODrive. Some BLDC motors have a hall effect sensor built in, and claim to provide position control. While the ODrive does support the use of a hall effect sensor, it does not provide accurate positional control, and the position may drift over time (*Encoders — ODrive Pro Documentation 0.6.6 Documentation*, n.d.).

3.2.2.3 Power Supply

In order for our robot to function freely, it requires an independent power source. Upon researching the different ways to power a robot of this scale, we found that we can either select a standalone type battery pack, or a wired power setup. We believe that the best option is to go with a standalone type battery in order to avoid the influence of a wire, as well as the physical limitations of travel. Robots of a similar scale, according to our research, were powered by 12s or custom batteries. The prices ranged from roughly $500 to several thousand, depending on the application for motor speed and the number of motors for degrees of freedom. We found some options on the lower end of that price range by looking for 12s Eskate batteries, which appear to have the power and current requirements for our application.

3.2.2.4 Computing

In our research, we found that robotic systems with similar functionality are controlled using recent models of Raspberry Pi, in combination primarily with C/C++ and python, aided by software libraries like ROS. The Raspberry Pi serves as the central computer for these systems

due to its computing abilities, reliability, and vast documentation. Most often, the operating systems are either Windows, Linux, or a combination of the two.

3.3 Humanoid Robot Sensing and Balancing

3.3.1 Balancing Mechanisms

One of the most pressing reasons for the need for cutting-edge perception and recognition technologies in humanoid robots is for their balance. Bipedal robots are inherently much more unstable than multi-legged or wheeled robots, and so need special techniques to remain an upright position. Broadly, there are two paradigms for robotic balancing, static and dynamic balance. Static balancing involves the robot's ability to remain standing in an upright position, while dynamic balancing allows the robot to use its own trajectory of motion to balance itself. One of the most widely used dynamic balancing mechanisms in humanoid robots is calculating the Zero Moment Point (ZMP) and matching that with the Center of Pressure (COP) by using force sensors and on-board computers.

The ZMP is a point in 3-dimensional space where the gravitational moments of all points on the robot sum to zero. ZMP control algorithms rely on the theory that the ZMP coincides with the COP when the ZMP is projected in the XY-plane on the floor. The ZMP can be measured and changed to alter the Center of Mass (COM). If the COM is found to be on a trajectory going outside of the support polygon, or outside of the box containing its contact with the ground, then the robot is unstable and requires ZMP correction.

The ZMP and COP are measured using a suite of sensors throughout the robot. Most ZMP-balancing robots include Inertial Mass Units (IMUs), which allow for the orientation of certain limbs to be read. IMUs are usually accompanied by either 6-Axis Force/Torque or ground-contact force sensors to measure normal forces at ground. These sensors enable the robot's central computer to calculate the current state of the robot, and signal motors to supply torques to actuators for correctional movement to realign the COM.

There are two main methods for implementing ZMP-based control algorithms, one strategy focuses on the ankles, and the other utilizes the double-inverted pendulum at the hips. The ankle strategy involves generating torque at the ankles to move the COM back within the support

polygon. The hip strategy, on the other hand, represents the robot as a double inverted pendulum, with sections from the ankles to the hips and the hips to the head. Actuators at the hip are responsible for re-balancing the robot in this strategy. Most ZMP controllers switch between the ankle strategy and the hip strategy, depending on the magnitude of the torque required for the correction. Lower torque requirements employ the ankle strategy, while large torques require the hip strategy.

ZMP control algorithms allow for a humanoid to employ dynamic balancing, where the robot balances itself while it is moving. This is in opposition to static balancing, which requires the humanoid to remain still to determine its balance. Dynamic balancing allows for a more human-like gait than static balancing, which requires the robot's COM to always be over its feet. This year, our MQP team is implementing an FFSM control algorithm for reactive balancing, which is a static balancing method. To implement a human-like gait, more additions would need to be made to HURON. Specifically, IMUs would need to be added to the robot's sensor architecture to ensure the orientation of the robot is known, and an upper half would need to be built to implement the hip strategy into the control algorithm. The following sections describe several types of sensors which can be used to implement a static balancing system, and an explanation of the final FFSM-FSR based system used for the HURON project.

3.3.2 Sensors for Balancing Robots

To act, reason, and interact like humans, humanoids must take input from the environment around them. There are several different kinds of sensors which allow humanoids to do this. Sensors consist of three components, the sensing element, conversion element, and electrical circuits (Ren et al., 2020). The sensing element directly perceives the measured quantity and converts the signal into other values following a definite relationship, while the conversion elements convert the non-electric quantity sensed by the sensitive element into electrical quantity (Ren et al., 2020). To understand how balanced the humanoid robot is and how its weight is distributed, we need to implement force sensors on the feet. Therefore, we considered four options: liquid metal sensors, piezoelectric sensors, load cells, and force sensing resistors (FSRs).

3.3.2.1 Liquid Metal Sensors

Liquid metal sensors, an innovation in the field of soft electronics, have been developed to easily fit into small spaces to detect and respond to touch with greater sensitivityLiquid metal

refers to the constituent metal whose melting point is below room temperature, which is usually Gallium. This fluid sensor allows for great adaptability and conductivity. However, there is a risk of the metal leaking from the soft encapsulation (Ren et al., 2020). These are used for detecting mechanical stress such as stretching, pressing, and bending as shown in Figure 3.35 below.

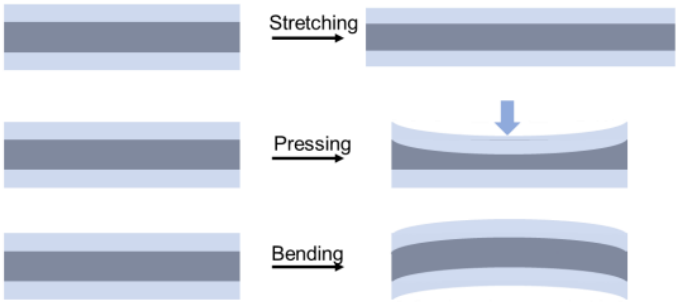


Figure 3.35: Liquid Metal Sensor's Ability to Detect Change [5]

Liquid metal sensors are good for detecting human activities such as finger bending or swallowing as their fluid nature allows them to combine the advantages of traditional artificial skin and sensors. Unfortunately, this is a developing technology, and the price of a liquid metal sensor is quite expensive, ranging from $800-1,500.

3.3.2.2 Piezoelectric Sensors

Piezoelectric sensors operate from the piezoelectric effect, which allows certain materials to gain an electric charge under mechanical stress. Piezoelectric sensors are best for measuring changes in load, as under static pressure the induced electric charge will return to neutral. Additionally, protective circuit designs must be utilized as high loads can induce voltages of up to 1,000V (Ren et al., 2020). These sensors are mostly used in acoustics to measure pressure variations in sound but do have applications for measuring physical forces. Piezoelectric sensors are manufactured as either large cylindrical transducers or as thin-film sensors, and range dramatically in price from around $20 on the low end to $500 on the high end. For our purposes, these sensors would be under a load which would induce voltages which could cause damage to the robot, and their application for measuring changes in pressure rather than static pressure would not suit our needs when the robot is standing still.

3.3.2.3 Load Cells

Load cells are a type of strain gauge sensor mainly used for robotics research. They come in two forms: multi-axis force sensors, which can measure forces in multiple directions, and single-axis load cells, which measure force uniaxially (*How Do Force Sensors Work? And What Are Their Benefits?*, n.d.). These are one of the first sensors designed to measure force due to their relative simplicity; load cells are just strain gauges connected to a PCB to convert a mechanical force to a proportional electrical output (*Continuum Robots and Tactile Sensors*, 2023). Despite their simplicity, they are very accurate and readily available, but they are bulky in size and expensive. The bulk of these sensors was the main reason they were ruled out for our design, as the sensors on the feet of the biped needed to be as unobtrusive as possible.

3.3.2.4 Force Sensing Resistors

The last sensors we researched were force sensing resistors, which in our research was the most common method for measuring external forces on a humanoid robot. These sensors can change their resistance in proportion to the amount of mechanical stress they experience; higher loads decrease resistance. FSRs are put together with two layers of conductive materials along with substrate film. On each layer, a conductive material is applied, followed by a layer of conductive ink. Adhesive is then used to laminate the two layers of substrate together to form the force sensor. FSRs have a non-linear force-resistance relationship which must be compensated either by analog circuit design or in software. However, Flexiforce sensors present a better response compared to most other FSRs in terms of linearity, repeatability, time drift, and dynamic accuracy. The price of one FSR ranges from $8-25 depending on the limitations of the FSR. These sensors were ultimately chosen for HURON.

3.3.2.5 Decision Matrix

To compare each of the sensors we researched and determine the best option, we created a decision matrix. The criteria we used to rate each sensor are shown below.

Table 3.1: Decision Matrix Criterion for Force Sensor

Criterion	Meaning and Scoring	Reason for Criterion
Design Integration - Software	Software needed to use this sensor and if it is compatible with the rest of the software design	If the software cannot integrate to the rest of the robot, it is no use

	(1-10, easy-difficult)	
Design Integration - Hardware	Hardware needed to use this sensor and if it is compatible with the rest of the hardware (1-10, easy-difficult)	If the hardware cannot integrate to the rest of the robot, it is no use
Price	How expensive it is (1-10, expensive-cheap)	Must remain within the budget
Accuracy	How accuracy and precise it is (1-10, least accurate-most accurate)	The sensor is what is making the robot stand so it needs to be accurate
Durability	How many times can it be used without breaking or drifting in resistance. (1-10, least durable-most durable)	This project will be continued for future years with few repairs
Linearity of Output	The linearity between force and output (1-10, nonlinear-linear)	A linear output will be the easiest to understand and most accurately telling of what is happening
Force Range/Threshold	Amount of force it can hold or read (1-10, low weight- high weight)	This robot is initially 40kg for only the lower half, so it must withhold a large amount of weight

Based on our research and decision matrix, we concluded that FSRs were our best option as it received the highest score from our decision matrix.

Table 3.2: Decision Matrix for Force Sensor

		Sensor			
Criterion	**Weight**	**Liquid Metal Sensors**	**Piezoelectric**	**Load Cell**	**FSR**
Design Integration - Software	10	5	4	3	5
Design Integration - Hardware	10	5	5	1	5
Price	5	1	1	2	5
Accuracy	6	3	4	3	2
Durability	8	3	5	5	3
Linearity of Output	10	2	3	5	2
Force Range/Threshold	5	1	5	5	4
Complexity	5	2	1	5	5
Total	**x**	**182**	**219**	**208**	**226**

3.3.3 FSR Placement

To control the balance of the robot, measure the center of pressure, and the contact forces while in motion, force sensitive resistors (FSRs) will be attached to the bottoms of the robot's feet. This posed the questions of where these sensors should be placed, how many sensors were needed, and how these sensors should be implemented. Several applications of FSRs were researched to come up with the ultimate design of HURON, including the measurement of human gait cycles and previous iterations of the bipedal robot.

In one study, Folgheraiter et. al. designed a flat foot and implemented four force sensors at the foot's planar (Folgheraiter & Aubakir, 2018). Additionally, a rubber layer was glued on the contact surface of the sensor to increase the friction with the floor. This design can be seen in the figure below.

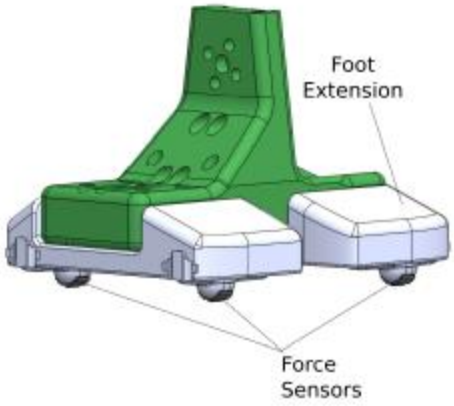


Figure 3.36: Foot Design of Lightweight Full-Scale Biped Robot

It is noted that these sensors are raised and are the only part of the foot touching the ground. This puts all the force on these sensors (giving more accurate readings) and makes calibration easier when changing environments. These force sensors were then connected to two microcontroller boards (ATmega328 microcontroller), one for each foot, to determine and filter the analog signals provided by the FSRs. These microcontrollers monitored the signals at a frequency of 1kHz. This design gave insight into all of our questions as it used 4 sensors with two in the front and two in the back with these sensors the only point of contact with the ground (Folgheraiter & Aubakir, 2018).

Malvade et. al. discussed an electronic insole system design using force sensing resistors to monitor pressure wirelessly and save the data (P. S. Malvade et al., 2017). This design used four

sensors, one at the heel, one at the lateral part, one at the metatarsal head and one at the anterior. The pressure measurements are then used to analyze the pressure distribution. The figure below shows the setup of where these sensors were located.

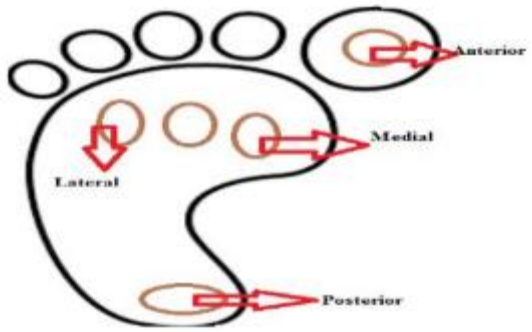


Figure 3.37: Sensor Locations for an Electronic Insole

Although the design of the robot should not have toes, we are attempting to create a human-like gait. Therefore, the pressure points on the feet will be similar to those of a human (P. S. Malvade et al., 2017). This again supports having a low number of sensors and having them in the front and back of the feet.

Application of Force Sensing Resistor (FSR) in Design of Pressure Scanning System for Plantar Pressure Measurement discusses the development of a low-cost dynamic pressure scanner. This provides insight into the peak pressure points of a human foot which are the heel, 1st, 3rd, 5th Metatarsal head, and the toe. However, this research found that to make this design more robust a few more points should be added to provide a complete coverage of pressure distribution on shoe soles. Therefore, this development used 8 force sensing resistors on each sole. A voltage divider circuit with unity gain amplifier was designed for these FSRs (Rana, 2010). The divider circuit and Op-amp LM-324 were designed so that the output of the circuit increases with increase in applied pressure even though the resistance varies inversely. Eight identical amplifiers were incorporated on a single board for parallel processing of plantar pressure data from each of the 8 FSRs. Implementing the amplifiers helps to improve the linearity, but it was still found that the response was nonlinear with a higher-pressure range. To compensate for this, they used the quadratic linearization function…

$$Y = AX + BX^2 + C$$

Y is the applied value of pressure
X is the measured output in volts
A and B are the slope coefficients

C is the intercept value

This helped the team make their output more linear which is seen in Figure 3.38 below

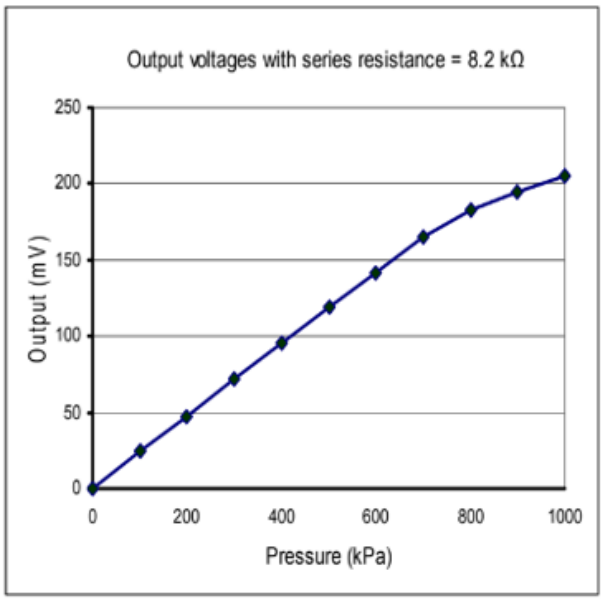


Figure 1.2 Results of circuit testing with compensation resistor

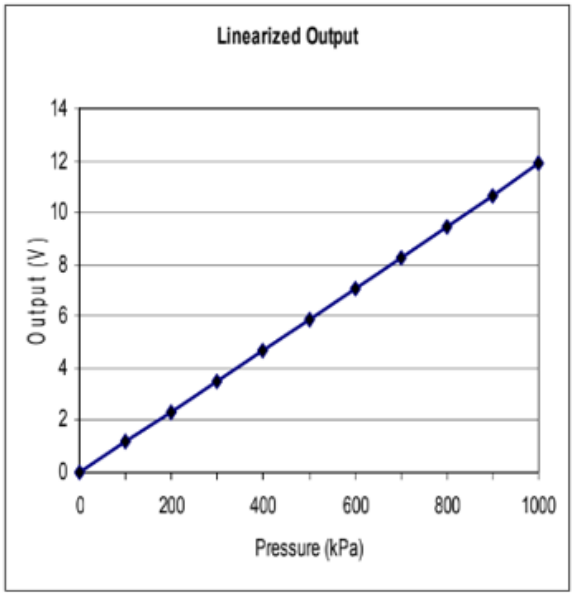


Figure 1.3 Linearized data plotted, offers a good linearity range

Figure 3.38: Initial Voltage Readings of FSR vs. Quadratic Linearization Voltage Readings of FSR (Rana, 2010)

After determining the location and number of sensors, the team placed the sensors in the specified locations and carefully glued the sensors to a soft rubber plate. These sensors had rubber plates on top and below them for protection and the rubber plates were then glued together (Rana, 2010). After experimentation, the team found that the following figure represents the locations where there was a significant difference in pressure, this is shown in the figure below.

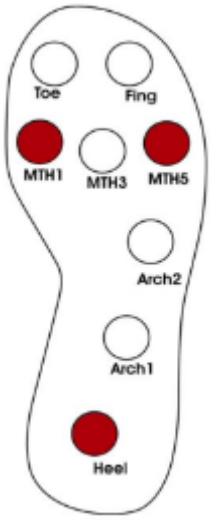


Figure 3.39: Red Spots Represents Significant Pressure Points for Foot Pressure Scanner (Rana, 2010)

Wibowo et. al. researched a static load measuring device using FSRs (Wibowo et al., 2020). The measuring instrument was designed in the form of a pressure plate platform which consists of 30 FSR 402 sensors, 15 for each foot. To obtain a fully covered load distribution of the foot, the measurements were divided into four areas, heel area, middle, front without the radius of the foot, and the radius of the foot (Wibowo et al., 2020). After experimenting with this design, the team found the following analysis of the location of pressure points on the foot.

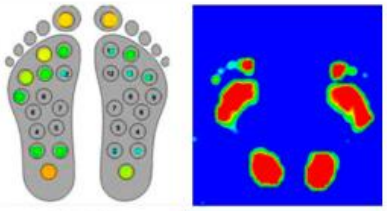
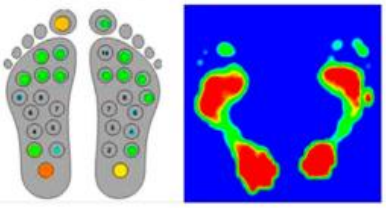
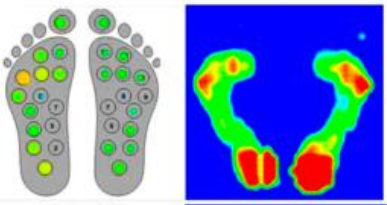
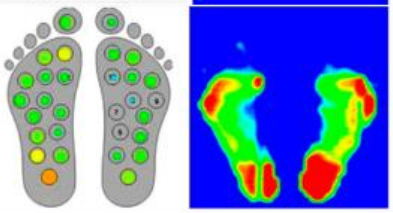
Type Soles of Foot	Display of Designed Measuring Instruments and MAFPM	Total Load in Each Area of Foot
High Arch		Subject No.: 1 Total load of the foot (gram) **Area / Left / Right** Rear 1202 1926 Middle 0 0 Front 1534 3472 Thumb 787 784
		Subject No.: 8 Total load of the foot (gram) **Area / Left / Right** Rear 1804 1979 Middle 689 0 Front 2953 2943 Thumb 756 820
Normal Arch		Subject No.: 2 Total load of the foot (gram) **Area / Left / Right** Rear 1598 1458 Middle 890 1156 Front 2462 4220 Thumb 516 631
		Subject No.: 6 Total load of the foot (gram) **Area / Left / Right** Rear 2077 2164 Middle 849 1243 Front 2276 4079 Thumb 354 411
Low Arch		Subject No.: 3 Total load of the foot (gram) **Area / Left / Right** Rear 1967 2102 Middle 1473 2188 Front 3478 4260 Thumb 766 824
		Subject No.: 5 Total load of the foot (gram) **Area / Left / Right** Rear 1797 2233 Middle 1104 2217 Front 2851 4158 Thumb 424 641

Figure 3.40: Pressure Points of Foot Based on Type of Arch (Wibowo et al., 2020)

A four FSR sensors system was chosen to provide sufficient information to balance our robot. The location of these sensors was in the front and back of the feet. Additionally, the sensors were the only point of contact with the ground. With the sensors picking up adequate signal from ground-contact forces, the following section discusses the theory behind converting these measurements to measurements of stability.

3.3.4 Foot Force Stability Margin

The overall goal of the sensor system was to communicate the stability of the robot so the robot can balance, walk, and react to external perturbations. Therefore, this involved measuring the force distribution on the feet, creating a quantitative way to measure the stability of robot, and decide how the robot should react at all times. When reacting to external forces, the robot may need to move its center of gravity (CG) or take a step. The stability of the robot depends on the geometric position of CG with respect to the foot contacts as well as the height of the robot. Therefore, to go from an unstable to stable state the geometry of the robot must move the CG by either applying torques to the ankles or hip or take a step. To ensure the stability of the robot is constantly monitored a real-time stability control system was integrated. This system utilized an array of FSR sensors, which changed their electric resistance in response to mechanical stress. By arranging the FSRs in a square on the base of each foot, an algorithm was created that could determine the overall stability and necessary corrective direction of movement for the robot (Agheli & Nestinger, 2012). In addition to these measurements, the algorithm was also capable of making the decision to take a step or move the center of gravity, depending on the stability being above or below a certain threshold.

The foot force stability margin (FFSM) was used as the measure for the current stability of the robot. FFSM provides a rapid sense of stability based solely on the forces read in at the feet on each FSR. This measurement allowed the robot to determine loss of stability, as well as the effectiveness of any recovery protocol the robot executed to return to a stable configuration. The FFSM mathematical formula is shown in the equation below:

$$FFSM = \prod_{i=1}^{n} \frac{f_i}{\bar{f}}, 0 \leq FFSM \leq 1 \quad (3.1)$$

Where n is equal to 8 (the number of FSR sensors).

The FFSM is defined as the product of individual normal foot force fractions to the average of all normal foot forces. n is the number of sensors in contact with the ground and $\bar{f}$ is $f_{total\ force}/n$ or the average of all normal foot force magnitudes.

To satisfy the maximum stability state of the robot, the product is normalized between 0 and 1 as shown in Equation 3.1. A number close to 0 represents an unstable state and 1 represents the maximum stable state. The FFSM represents how close the system is to the unstable or maximally

stability state. An FFSM equal to one only occurs when the foot forces are evenly distributed, or the standard deviation of foot force magnitude is zero, as this is the optimal balancing state which the robot is always trying to achieve.

The figure below theorizes a stable and unstable state of the robot from reading the amount of force on all 8 FSRs. On the left, all the FSRs are reading a similar amount of weight while on the right, the amount of weight on each FSR varies much more. When using the FFSM equation to calculate the stability of the stable set of feet on the left, the stability is calculated to be about 0.8. When using the equation for FFSM to calculate the stability of the stable set of feet on the left, the stability is calculated to be about 0.82. When using the equation for FFSM to calculate the stability of the unstable set of feet on the right, the stability is calculated to be about 0.66. This was easily implemented onto the robot by reading the force reading on all eight FSRs, finding the average force between all eight FSRs, and using the equation for FFSM.

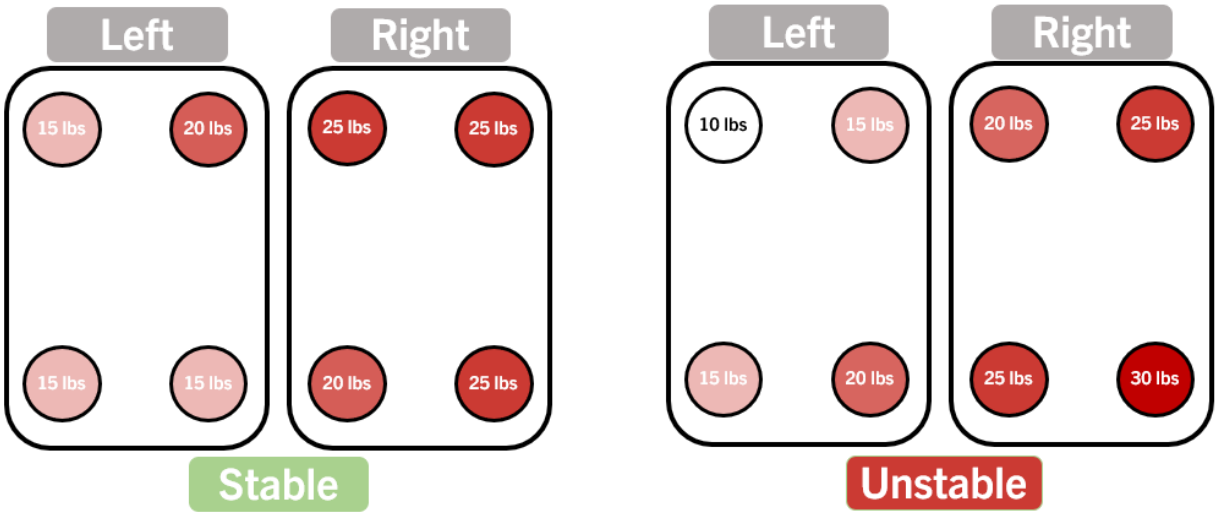


Figure 3.41: Theorizing a Stable and Unstable State From Reading Robot's FSRs

Another important aspect for determining the stability of the robot was determining the directions of the external perturbation. This information was necessary for the robot to take corrective action and return to a stable state.

The implementation of the angle measurement for HURON involved calculating the angle on each foot, then averaging these two angles to find the overall angle of external perturbation.

The direction of an external force was measured as an offset from the FSR with the maximum normal foot forces (Agheli & Nestinger, 2012). Therefore, the direction of the external force was determined based on which FSR experiences the maximum normal foot force. The direction of the external force can be found using the force ratio between the two adjacent sensors to the sensor experiencing the maximum force, given by

$$\eta = \frac{f_{i-1} - f_{i+1}}{(f_{i-1} + f_{i+1}) - 2f_i} \tag{3.2}$$

Based on the geometry of the robot, the angle offset from the max FSR can be given by

$$\gamma = \frac{\eta \cdot \varphi}{2} \tag{3.3}$$

Where φ is the angular distance between each sensor which depends on the geometrical constraints of the robot. The angular distance was determined to be:

$$\varphi = \frac{\pi}{2} \tag{3.4}$$

Because this was the angle made between each FSR and the center of the foot.

To implement this system, each FSR on the left and right foot was assigned a degree value of 45, 135, 225, and 315 degrees. These values are represented by α in the below equation. In real time, the FSR experiencing the maximum force is identified, and the adjacent FSR force readings are also pulled. The calculations of Equations 3.2 and 3.3 are then performed, where the former represents the offset of direction of external force from the maximum FSR, and the latter the absolute angle given by the maximum FSR. The system then calculates the angle of external force by adding these two values:

$$\theta = \alpha + \gamma \tag{3.5}$$

Where θ is the angle of external perturbation.

The figure below provides a theoretical explanation of determining the angle at which the robot is leaning. The top center of the foot is 0°, the top right FSR is 45°, the bottom right FSR is 135°, the bottom left FSR is 225°, and the top left FSR is 315°. Since both feet should be experiencing about the same angle, this figure only includes one theoretical foot. The angle is calculated for each foot and then averaged to find the final angle on which the robot is leaning.

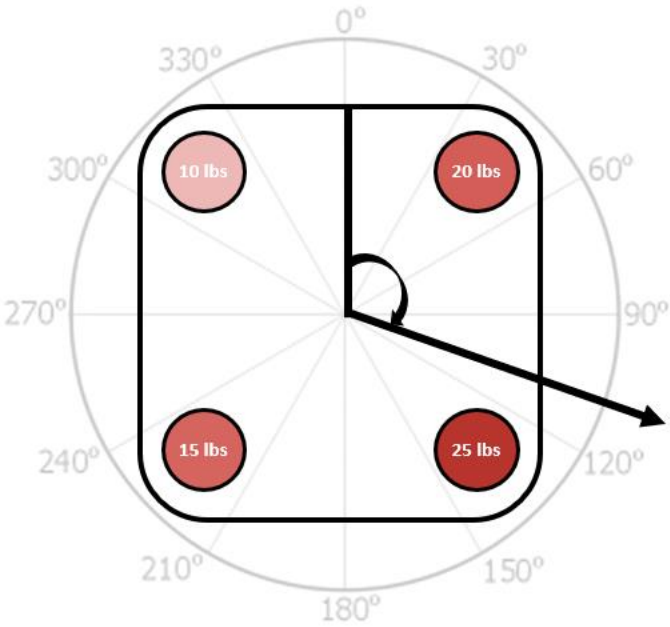


Figure 3.42: Theoretically Determining the Angle the Robot is Leaning from FSRs

The angle was measured by determining the index of the FSR which was experiencing the maximum force at any given time. This was implemented with the robot through simple logic to pull the indices and values of the adjacent FSRs from the 2 by 2 matrix. After the relative angle was calculated using an offset from the FSR experiencing maximum force, this was added to the absolute angle to give an angle of external perturbation. This calculation was performed for the right and left foot separately, then averaged to give an angle for the whole robot system.

4.0 Robot Design Process

The process for designing and implementing a humanoid system is an iterative process, meaning for each change most of the model and calculations generated need to be reworked to accommodate. This idea is shown below in Figure 4.1.

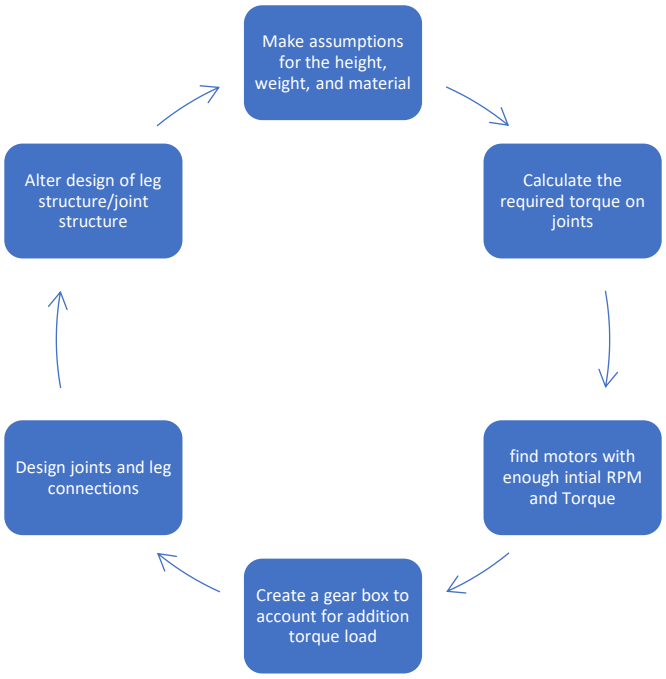


Figure 4.1. The iterative cycle of design for the lower body.

During A Term, we went through three major iterations, which will be explored in depth below, covering topics such as proportions, joint order and orientation, the CAD models in SolidWorks, materials used, estimated cost, which motors the model is based on, and the gearbox used.

4.1 Iteration 1

The first iteration of our design for the HURON robot featured the main ideas of first designing for human proportions in mind, not simply the orientations and relative locations of the joints. This iteration also includes the initial research and testing of a harmonic drive based on initially selected motor possibilities. At this stage, the primary budget was still considered to be around $3000, but discussions of raising to $5000 had already begun.

4.1.1 Proportions

The first thing we had to consider when first designing the lower body is to figure out the relative proportions between the sections of the legs. Some research into the generalized proportions was first done, and we also measured the lengths of one of our teammates to get a physical idea of how the proportions related to each other (López, 2014).

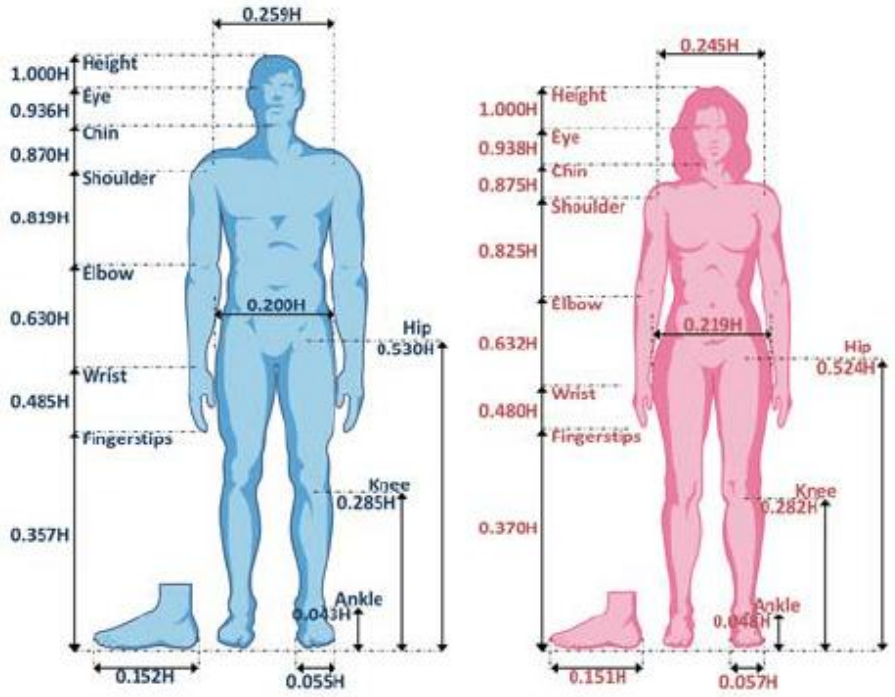


Figure 4.2. The proportions of human limbs relative to overall height of the person.

Based on the proportions shown above in Figure 4.2, we used the height of our teammate, 5’10”, to calculate the relative distances between joints. This gave a distance from floor to hip to be 94.2 cm, distance to knee 50.6 cm, and distance to ankle of 7.6 cm. From these values we found that the length of the femur should be 43.6 cm, length of tibia and fibula lengths should be 43 cm. However, the overall height of the lower legs is closer to 1.1 m tall with the added area for the pelvis where the batteries would rest.

From these initial measurements of the generalized shape and size of a human, this then gave us the initial constraints of size for developing a humanoid set of legs. However, an important note is that these calculations are a generalized set of proportions and will not be representative of all people that are 5’10” as the proportions vary widely from person to person. With this generalized shape, we were able to then set constraints for the size of mechanisms we can utilize,

and cost of materials can be narrowed down based on the expected weight of the internal workings, like gearboxes, motors, and batteries.

4.1.2 Joint Order and Positions

The initial setup of joint orders was the hip had the roll, pitch, yaw. However, upon looking at this design further we noted that this order would make the robot waddle more, so we switched to the order yaw, pitch, roll, which had a more similar motion to a human, shown in Figure 4.3, and was the order used for the NU-biped (Folgheraiter & Aubakir, 2018). Based on the estimated sizes of the motors and the inclusion of the gearboxes, we found that it would be extremely difficult to have the joints on top of each other, making the spacing between joints more crucial, which is visible for the hip and ankle joints especially.

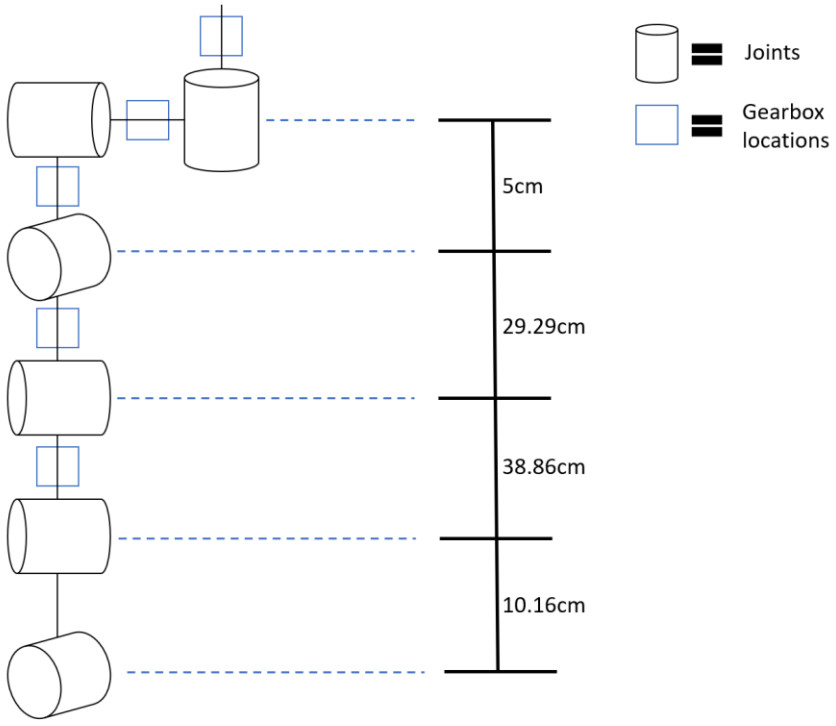


Figure 4.3. The order of joints with initial estimates of distance based on size of the gearboxes and the motors.

The locations of the splits were also done with consideration of where these joints would fit into the human proportions we had established. The last consideration that we started off with was also how to create enough torque for the joints to move the legs, which is why the inclusion of gearboxes was vital for designing and placing the joints.

4.1.3 CAD

Using the order of joints discussed in the section above, we made a rough sketch of the robot to simulate the motion of our robot. The model is a 12 DOF humanoid robot's lower body that uses blocks for the leg "bones and cylindrical pins to rotate about as shown in the figure below. These cylindrical pins were mated together such that they were overlapping each other. Although this model could not be used for the actual design of the legs, they gave us a good idea on what would happen if the joints were ordered with the positions and joint configuration as mentioned in the above section.

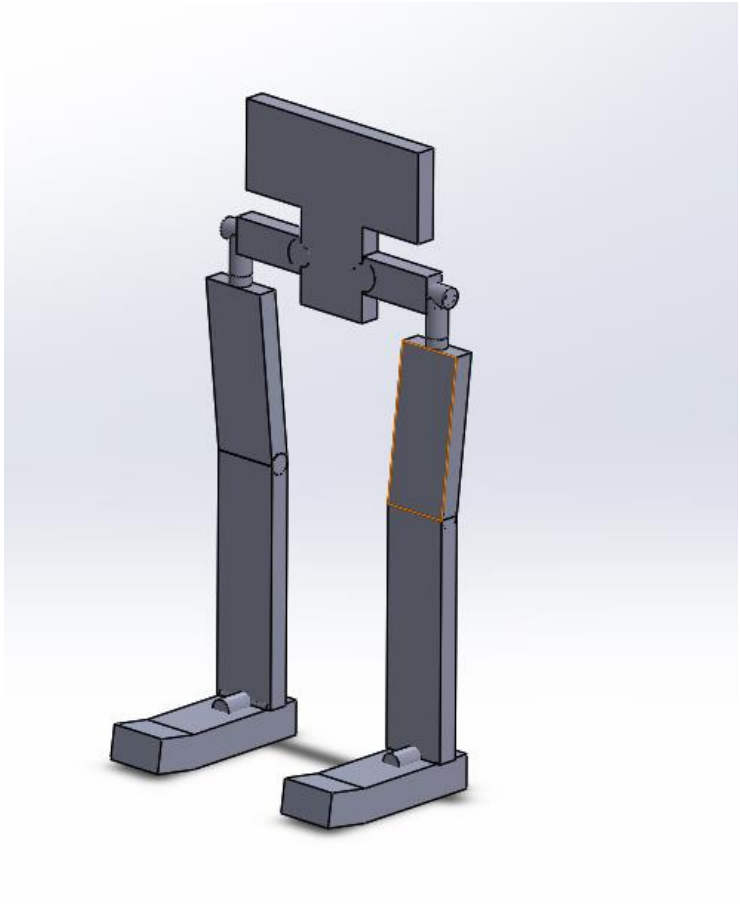


Figure 4.4. The first rough sketch of the legs with stand-in joint locations as simple cylinders.

4.1.4 Materials

The first materials we investigated using to build iteration 1 was PLA and aluminum. The PLA is primarily for creating the shell that will encase the legs to give them a more human appearance. The aluminum is used to build the links between the actuators. Based on initial research, we chose Aluminum 6061, and the links are based on 1" rods. The yield strength of

aluminum rods is 42,000 psi, which is more than the yield strength of a human femur bone which is 16,500 psi, which has a cross section of 4.4 cm^2. The cost of printing was based on the rates in the makerspace which for PLA it costs $0.04/gram printed while if we were to use Nylon, then the cost is $7/in^3 (*Prototyping Lab: WPI Makerspace*, n.d.). This gave us the initial weight calculations which are shown in Appendix C.

4.1.5 Cost

The initial budget for this project was $3000, with half of this being provided by the department and half provided by the students in place of textbook costs for the entire project. This is a very tight budget, and as a result discussions of finding new sources of funding were one of the first topics covered for iteration 1, with the goal to raise at least an additional $2000. So based on the newer budget of $5000, the estimated cost of parts is shown below in Figure 4.5. At this point in the design, the goal was to primarily 3D print parts, such as the shell of the robot, and the joint connections.

Budget: $5000	Total Cost	$4,620.93		
Item		**Cost**	Quantity	
DESIGN TEAM				
6606.44cm^3 (10.16 x 32 x 20.32cm) for hip; nylon; 30% density		$7/in^3	6606.44cm^3	$846.61
1" diameter Aluminum 6061 rods		$51.82	3	$155.46
if the surface area of the leg is 499.5 in^2 and we had 20% density for the mesh; thickness of the PLA layer is 1 in		$0.04/gram	10231.70	$818.54
		$1,820.61		
SENSOR TEAM: FSR Sensor		**Cost**	**Quantity**	**Total**
8 FSR Sensors		$8.29	8	$66.32
2 Arduino Unos		$27.00	2	$54.00
Total		**$120.32**		
CONTROL TEAM		Cost	Quantity	Total
Strong Brushless 3 Phase Motor		$200.00	4	$800.00
Medium hip		$140.00	2	$280.00
weak knee		$80.00	2	$160.00
Strong Ankle		$200.00	2	$400.00
medium ankle		$140.00	2	$280.00
weak ankle		$80.00	2	$160.00
Computer		$400.00	1	$400.00
Battery		$200.00	1	$200.00
Total cost		**$2,680.00**		

Figure 4.5. Initial budget table based on PLA casing with aluminum rods for bones and the additional costs from the other teams.

4.1.6 Motor Stand-ins

At this point of the project, we were just starting to think about the dimensions and required torques of the robot's legs and joints, so we only had a rough idea on the constraints of the motor in terms of dimensions and torque requirement. Based on the initial torque calculations done in Appendix A, we knew that we wanted to have as high of a torque as possible with a rough dimensional constraint of 103mm x 50mmØ. This dimension would allow the motors to fit comfortably within the proportions of a human's body. Any additional torque needed in each joint could be accomplished with a gearbox which will be discussed more in the next section. Since motors are also the bridge between the mechanical and electrical sides of the robot, the Controls team were responsible for finding a high torque motor with the necessary RPM to move the legs with a human gait while the Design team was responsible for figuring out the dimensional constraints.

4.1.7 Harmonic Drive

Based on the literature review, we discovered that most modern legged robots utilize harmonic drives on the joints to increase torque because they possess large gear ratios while taking up minimal space. However, because these gearboxes are so specialized, they are also very expensive as the companies which do sell them do not include the price of the gearboxes and are considered a special-order item. Therefore, instead of trying to raise money to afford such expensive gearboxes, we discovered that it is possible to 3D print a harmonic drive, which was the more cost-effective solution since we require 10 gearboxes for the lower body.

From the initial research into harmonic drives along with the selected motor, we established that a gear ratio of 1:400 was required at the maximum torque locations for the joints which experienced the most torque. This gear ratio was based on relatively cheap motors that had very high RPM but had stall torques with magnitudes in the milli-Newton meters. However, this was calculated for when the robot had an estimated weight of 40kg. Since the estimated weight of the robot and maximum stall of torque of the selected motors could significantly change the gearbox and overall maximum torque calculations, we had to recalculate the required torques of the robot since the center of mass location also changes using the preliminarily selected motor that the Controls team found. These calculations determined that we would want a 400:1 gear ratio to provide the required torque with a factor of safety of at least 2. Although the theory of the harmonic drive mechanism could resolve the issue of cost and limited space, a prototype would need to be

made to test the mechanical strength of the gearbox. This prototype will be discussed more in iteration 2.

4.2 Iteration 2

In the second iteration, the main changes from the first iteration were the changing in joint configurations as well as testing new joint designs to make the overall dimensions smaller. The other important changes were that the costing was based on the newer budget, potentially $7000 over the $5000 budget discussed in iteration 1. Finally at this point is when the CAD of the robot increased, several potential designs were created, and the testing of the harmonic drive prototype began in earnest.

4.2.1 Proportions

At this point of the project, we finalized the dimensions of the robot's legs to be modeled after a 5' 10" person. Not only does this model close to an average American male, but one of our team members closely resemble a 5' 10" male. This made it easier to quickly make some measurements for different parts of the body as a reference. These references were used to make the initial geometries of certain parts of the leg.

4.2.2 Joint Order and Positions

During design iteration 2, We redesigned the hips several times. The improvement between the first two versions was changing the design to have intersecting axes of rotation. Intersecting axes at a joint simplifies forward and inverse kinematics calculations for joint positions, so it was beneficial to redesign the hip with this in mind. While hip version 2 had intersecting axes the resulting total hip width was 60mm, which was significantly wider than our design target. Hip version 3 managed to keep intersecting axes and reduced the width of the hip down to approximately 48mm. This was accomplished partially thanks to a new joint order for the hip.

The second hip iteration realigned the axes but kept the same order of joints. The third hip iteration changed the joint order, as shown below:

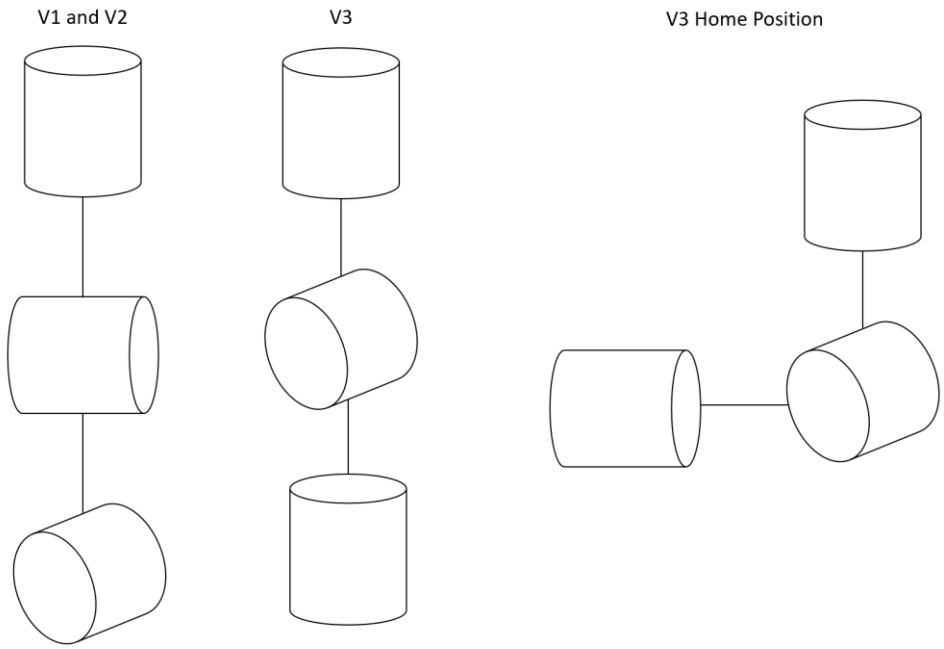


Figure 4.6: Joint orientation differences between Hip 2 and Hip 3

This new order allowed us to maintain the intersecting axes of rotation while moving some of the bulk out of the immediate area, which allowed us to make a narrower hip.

The knee joint stayed int the same orientation and general location as initially sketched in iteration 1. The first ankle design had intersecting axes but maintaining intersecting axes and keeping the foot small enough to fit in a shoe proved difficult for the designs at the time and so a second design had a small separation between axes.

4.2.3 CAD

For this iteration, a major portion of the work was to model designs for the different joints and skeletal structures of the legs of the robot. We worked on creating a right leg before mirroring the design for the left leg. Based on the joint positions discussed in the above section, the overall shape of the legs was finalized since we were modeling the size of the robot after a 5’10” person. The modeling was split into four categories: hip joint, knee joint, ankle joint, and leg bones. Since the Sensors team was still doing research and tests with the pressure sensor, we left the foot as a single plate of metal as a place holder until we finalized the design with the Sensors team.

In terms of the modeling process, we set a standard of constraints for each section. The design had to accommodate a 103mm x 50mm Ø motor and a 90mm x 90mm Ø harmonic drive gearbox. If the motor cannot directly connect to the gearbox, a 1:1 pulley using timing belts would

be used. For the brackets to support the joints, the thickness of each side of the bracket is 10mm which ensures that the brackets can support the different stresses experienced by the robot. This thickness was based on our engineering intuition as we did not know what the total weight of the robot would be as well as the concentration of stress experienced by each bracket. With these standard limitations and constraints, we were able to standardize the design of each section of the robot, making it easier to assemble the different parts of the robot in the master assembly of the legs.

Throughout the modeling process, different parts of each section were redesigned multiple times due to changing circumstances and overall design goals like the order of the joints, overall dimensions of the legs, and stress concentrations. These sub-iterations of each section are discussed in detail in the following sections.

4.2.3.1 Hip Joints

The hip joint was the most complex joint designed during Iteration 2, as it had to fit 3 degrees of freedom in as narrow of an area as possible. The initial design, shown below in Figure 4.7, was a relatively compact design but the axes of rotation did not intersect, which leads to more complicated kinematics. Additionally, the second stage of the bracket had design issues that likely would have led to it not surviving future design iterations, namely the mounting and rotation of the bracket completely encircling the harmonic drive. This mount likely would not have been structurally sound enough to support the weight of the robot.

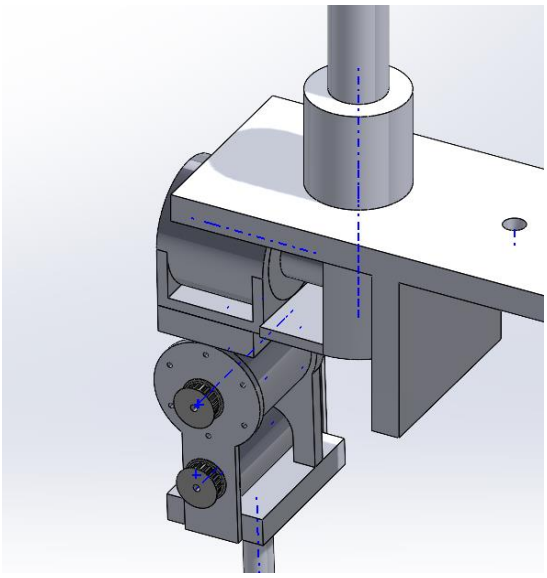


Figure 4.7: The first iteration CAD of the hip joint where the axes of rotation do not align.

Hip version 2 was a major redesign of the hip joint. The focus of version 2 was to have a hip joint with intersecting axes, to simplify the robot's kinematic calculations. The design for this hip iteration was loosely based off the design of the LOLA robot. In addition to being wide, hip version 2 likely would have been difficult to manufacture and assemble.

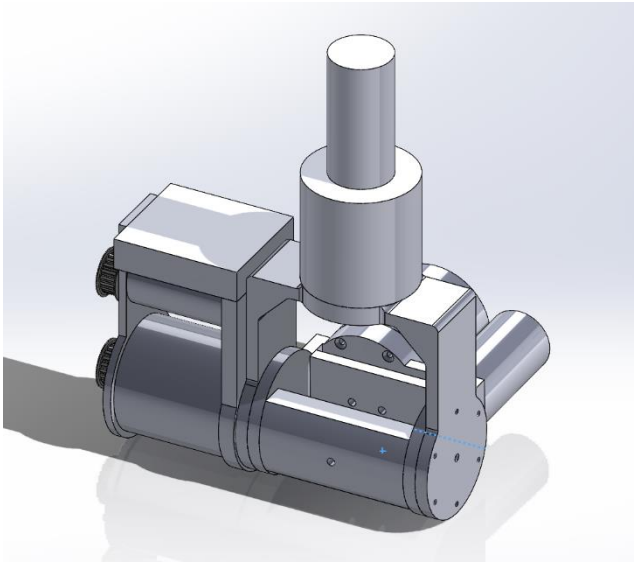


Figure 4.8: The second iteration CAD of the hip joint with a more compact design and the joints intersecting.

The primary design goal of the third version of the hip was to make the hips narrower than the second version. Additionally, we were provided an alternative way to orient the roll, pitch, and yaw motors, by changing the actuation order and having the home position be 90 degrees off to

the side. As an added benefit to this new order, we were able to emulate human bone structure by having an angled “femur” between the hip and the knee. The angle from this femur additionally serves to move the knee directly under the hip, simplifying the kinematics of the leg.

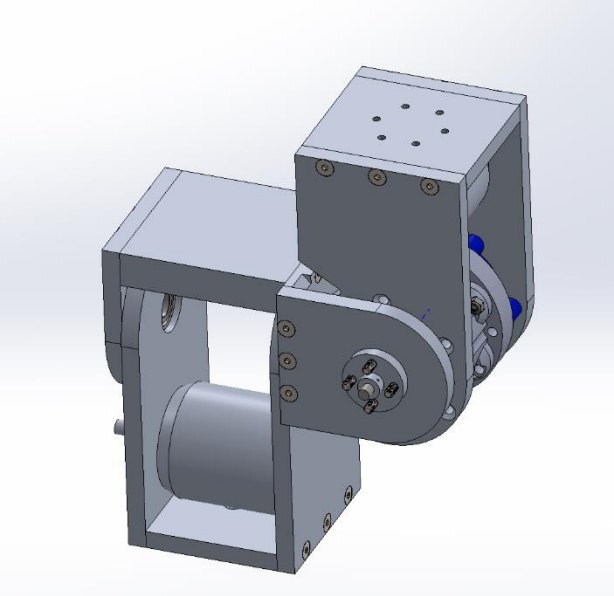


Figure 4.9. The third iteration of CAD for the hip, sporting stand in harmonic drive and motors.

4.2.3.2 Knee Joint

Version 1 of the knee joint was heavily inspired from the *Design and Control of a Humanoid Robot* paper by Chew, which was shown in Figure 2.26. We went with this design as we were currently working on the harmonic drive in this design iteration, and we needed a way to have the motor and harmonic drive together without taking too much space. Attaching the motor and the harmonic drive horizontally was not ideal because that would be too long length-wise, which would not be human like. Therefore, the design of stacking the motor and harmonic drive works for us as it solves the issue of fitting within human proportions.

This led us to our first version of the knee, which is shown in the figure below. We did our best to mimic the knee joint from the paper, but we couldn’t mimic the bracket attachments due to the size of our harmonic drive. At this point, the drive was larger than the motor stand-in we had, meaning there was only room to fit a couple of brackets.

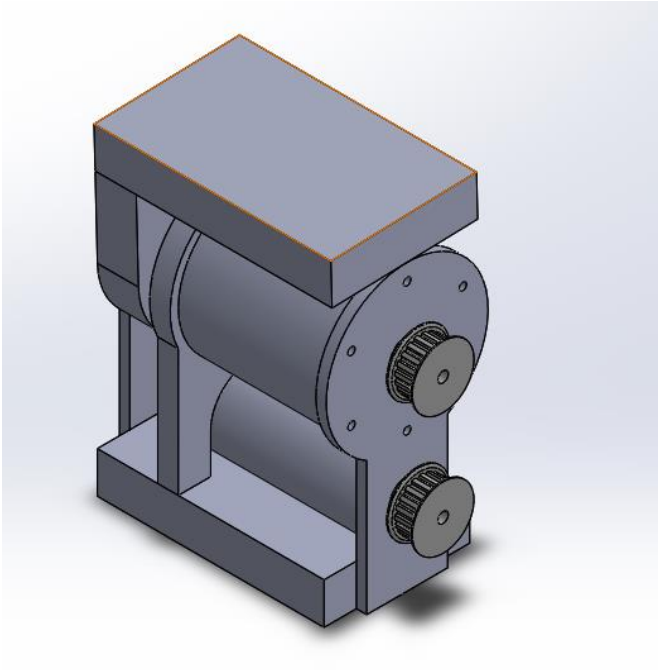


Figure 4.10. The 3D modeled knee joint with harmonic drive and motor stand-ins and connection points.

The main issue with this design is that that harmonic drive was only supported by one bracket at the end, which is not good for support purposes. Since the weight of the robot above the knee is pushing on the knee joint, having only one small bracket would not suffice.

This led us to the second version of the knee, which attempted to solve the support issue by modifying the brackets from version 1. The main difference was the mount for the motor. Since it was smaller than the harmonic drive, we decided to make the mount encase the motor, which allowed for the mount to have more surface area for the harmonic drive. This way, the force of the robot is spread out more and is not concentrated on a small bracket, which gives the knee more support. There were also some slight modifications to the brackets where the pulleys were, as we wanted to account for the motor axle that the pulleys are attached to.

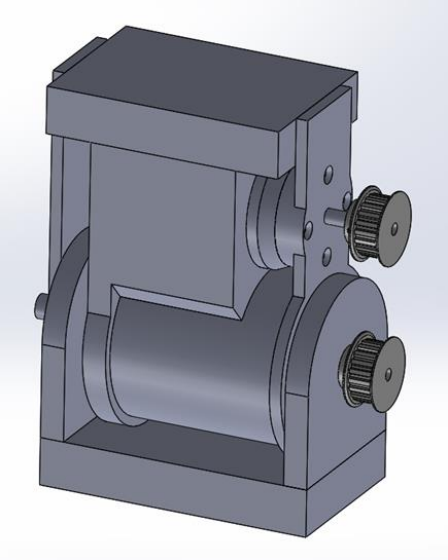


Figure 4.11. The second iteration of knee joint with more structural support and mounting locations for the motor.

4.2.3.3 Ankle Joint

For the first version of the ankle, we decided to utilize the knee joint as the pitch to simplify the design for us. This meant we had to build around the pitch to figure out how the roll will fit. This was a challenge as the pitch was bulky for an ankle, and we were working towards trying to fit the foot in the shoe. Since we had not gotten to the foot yet, we made a foot stand-in, which was just a rectangle piece with the same dimensions as a size 10 US foot. The other challenge was getting the axes to intersect using the current pitch joint. After one of our meetings with the advisors, we were informed that it is not entirely necessary to have the ankle joints intersecting. Therefore, we designed the roll under this assumption.

We naively assumed that the ankle roll did not need much torque based on our initial calculations (see Appendix A), meaning we did not utilize the harmonic drive. Therefore, we designed a simple casing for the motor to go inside, which then attached to the bottom of the pitch joint. This theoretically works as the roll would move the foot, which was attached by a couple brackets, and the whole roll mechanism could pitch up and down.

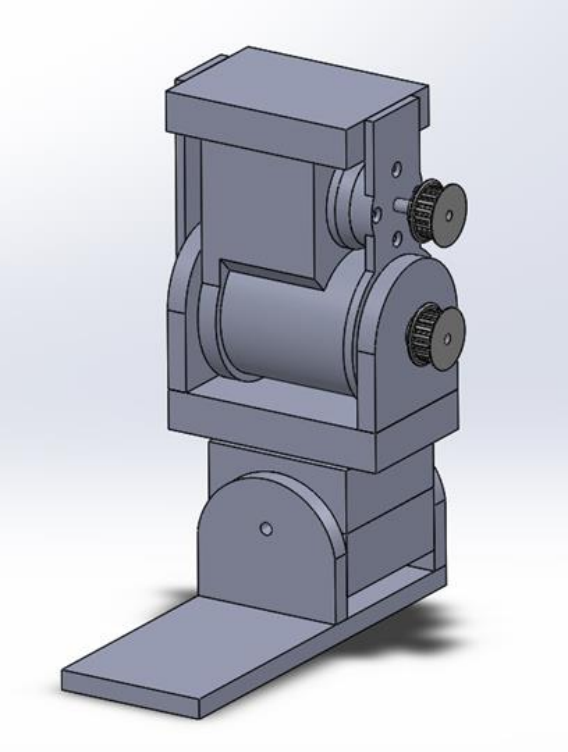


Figure 4.12. First iteration of ankle joint

There were a few issues with the design. First, it would not fit into a foot based on the way it was designed. The roll mechanism was too big and would not fit into the opening of a shoe, especially a size 10 US shoe, which is what we were basing the dimensions off. Secondly, the point where the ankle pitches would not look correct as a human. Assuming this was operational, if the ankle were to pitch up or down, it would constantly look like the robot's ankles were breaking in half, which is not ideal. There is also the incorrect assumption of the ankle roll torque requirements, but this would be further discussed during iteration 3.

4.2.3.4 Leg Bones

As discussed in the Literature review (specifically Chapter 2.4), there is a tilt angle in the femur that we wanted to mimic. To do this, we made a rod and cut the ends at the specified angle that was calculated in Chapter 2.4, which is shown in the figure below.

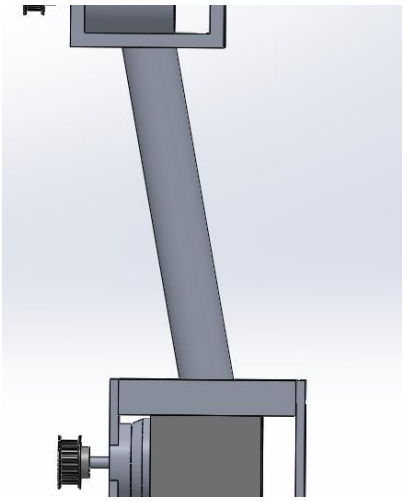


Figure 4.13: Femur bone in the robot

The tibia and fibula are also shown below, following the research that was done in Chapter 2.4, where we made the tibia thicker in diameter.

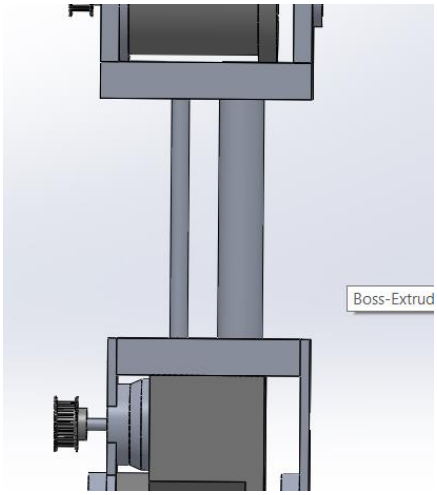


Figure 4.14: Tibia and Fibula Bones in the robot

4.2.4 Materials

During design iteration 2, our team considered the possibility of 3D a larger selection of parts, however, due to the size of the robot and the load-critical nature of many of the parts, we decided to use 6061 aluminum instead. Aluminum has a high strength to weight ratio, making higher factors of safety easier to achieve, and it is relatively fast to machine compared to harder metals like steel. 6061 is a common multipurpose alloy and usually has the lowest cost per unit on sites like McMaster Carr.

4.2.5 Cost

At this stage of the design process, we had revised the budget to then be based on the total budget of $7000, due to the addition of outside funding, and the pitch paper developed by the controls and sensor teams to give to companies to gain monetary or part donations. In the revised budget of costing, our section is more detailed as we had started to discuss the form of the joints, as well as connections, and hardware. Based on the requirements of the other teams, we limited our section of the budget to be around $3000, which can be seen with more detail below in Figure 4.15.

Budget: $7000			Total Cost	$5,773.79
Item	**Used For**	**Cost**	**Quantity**	**Total**
Aluminum Rod - 1.5" x 6ft	Tibia, Femur	$103.90	1	$103.90
Aluminum Rod - 0.5" x 3ft	Fibula	$6.04	1	$6.04
Aluminum Sheet - .5" x 24" x 24"	Joint brackets	$382.26	4	$1,529.04
1/2" Shaft Bearing	Joint brackets	$7.11	18	$127.98
1/4" Shaft Bearing	Joint brackets	$6.79	18	$122.22
M2 2.5mm Set Screws 25 pack	Joint brackets	$10.00	1	$10.00
1/4" Steel Rod 6 ft	Joint brackets	$8.02	1	$8.02
1/2" Steel Rod 1 ft	Ankle roll joint	$5.29	1	$5.29
M5x.08x25mm flat screws (100ct.)	Everything - not drive	$12.47	3	$37.41
M5x.08x25mm rounded screws (100ct.)	Harmonic Drive	$14.99	2	$29.98
M5 nylock nuts (100ct.)	Mostly everything	$16.62	3	$49.86
M4x0.7x18mm rounded screws (100ct.)	Motor screws	$11.29	1	$11.29
3D Printing	Everything	$200.00	1	$200.00
M5x0.8x35mm rounded screws (25ct.)	Harmonic Drive	$7.63	2	$15.26
14 Tooth Pulley	Joint brackets	$20.64	10	$206.40
28 Tooth Pulley	Joint brackets	$37.43	10	$374.30
15" Timing Belt	Joint brackets	$6.08	10	$60.80
				$0.00
Design Total Cost		Goal: <$3000		**$2,897.79**

Figure 4.15. The design budget based on a total budget of $7000.

Based on the updated idea of using a harmonic drive, we determined that we could 3D print a harmonic drive, to reduce not only overall cost, but also weight. At this point we also had to include the pulley systems that would translate the power of the motors to the drives. The cost break down of the other teams is shown below in Figure 4.16.

SENSOR TEAM: FSR Sensor		Cost	Quantity	Total
8 FSR Sensors (100Lbs)		$22.00	8	$176.00
				$0.00
2 Rubber Insoles		$10.00	2	$20.00
Sensor Total Cost				**$196.00**
CONTROL TEAM		**Cost**	**Quantity**	**Total**
Strong Brushless 3 Phase Motor		$200.00	4	$800.00
Medium hip		$140.00	2	$280.00
weak knee		$80.00	2	$160.00
Strong Ankle		$200.00	2	$400.00
medium ankle		$140.00	2	$280.00
weak ankle		$80.00	2	$160.00
Computer		$400.00	1	$400.00
Battery		$200.00	1	$200.00
Control Total cost				**$2,680.00**

Figure 4.16. The updated budgets based on the $7000 budget for controls and sensors.

4.2.6 Motor Stand-ins

We were still using the same motor stand-ins with dimensions of 103mm x 50mmØ. At the time of design for Iteration 2, we were still waiting for a finalized decision from the Controls team on their selection for a motor.

4.2.7 Harmonic Drive

While we were modeling different parts of the robot, part of the team was also modeling and prototyping a 3D-printed harmonic drive gearbox. There are three major components to the harmonic drive: the flexspline, the circular spline, and the wave generator. The input of the harmonic drive rotates the wave generator which then deforms the flexspline to contact two opposite sides of the circular spline which is a rigid internal gear. As the wave generator spins, the teeth of the flexspline mesh with a circular spline where each tooth of the flexspline has a harmonic trajectory relative to the circular spline teeth. This “tooth skipping” results in a much higher gear ratio than traditional spur gears within a much smaller area.

Although the theory of the harmonic drive fits well with the design goals of the robot, we wanted to make a prototype of the harmonic drive to not only confirm the design of our model but to also test the mechanical strength of 3D printing filaments available in the Innovation Studio Makerspace Prototyping Lab. For the flexspline, we used an equation-based tooth profile where we would need to define the number of teeth, module, and pressure angle to create the gear part of a harmonic drive gearbox. This allowed us to modify the flexspline as needed to achieve the correct gear ratio we needed. The same was done with the circular spline, but instead of creating the teeth,

the tooth profile was cut out of a circular ring. For the wave generator, an elliptical shape was made where it would have pins to hold small 8mm bearings. These bearings were what generated the harmonic motion of the flexspline as it moved around the circular spline. The rest of the components of the harmonic drive were made to accommodate for these three major components.

For this prototype, the design was a 2-stage stacked harmonic drive with a gear ratio of 400:1 with each stage being a 20:1 gear ratio. This would ensure that even relatively weak motors would be able to achieve the required torque of the robot.

Once the model of the harmonic drive was finished, we 3D printed each part using PLA with an infill of 20%. Structural parts like the casing and circular spline were printed with a thickness of 6 layers of walls to make them much stiffer than printing with standard printing settings. For the flexspline, we printed with 3 layers of walls to make the component both flexible and strong. Once the parts were printed, the gearbox was assembled and ready for mechanical strength testing.

To test the harmonic drive, a crank and lever arm were attached to the input and output shafts of the gearbox, respectively, and weights of various masses were attached to each of the lever arm to simulate a torque. Since the gearbox was designed to be as compact as possible, it was not able to be mounted to a vice grip. Therefore, we handheld the gearbox while taping weights from the recreational center to the lever arm. Once the weight was attached, one of us manually lifted the gearbox, keeping the lever arm as parallel to the ground as possible. We started from attaching a 5lb weight to a 60lb weight, which was the maximum weight available. This meant that our harmonic drive was able to support about 30N*m of torque in static, so we moved on to dynamic mechanical strength. For this test, we modified the prototype to attach it to two metal plates on the input side and output shaft of the harmonic drive. The metal plate attached to the input side of the harmonic drive was mounted to a vice grip. The manual crank of the harmonic drive was then rotated to actuate the gearbox and rotate the metal plate on the output shaft downwards, pressing down on a scale until the harmonic drive broke. The readings for the scale can then be translated to how much force the gearbox is applying to the scale in which the torque of the gearbox can then be calculated.

With the initial prototype of the harmonic drive, there were two points of failure: the output torque of the flexspline and the meshing of the circular spine and the flexspline. These points of failure were then modified and reinforced appropriately for the next iteration.

4.3 Iteration 3

4.3.1 Proportions

In this iteration, we changed the height of the legs to 1.1m or 3’ 7”. Our robot still resembles close to a 5’ 10” person because the height from the bottom of the foot to the hip is about 1m, but the extra 0.1m includes part of the torso of the robot. Our final lower body dimensions ended up being 1100mm x 500mm x 160mm, not including the length of the foot which was finalized to fit within a size 10 shoe.

4.3.2 Joint Order and Position

The main change in joints was the ankle joint and the foot. The ankle joints were switched, along with the positions and type of actuation. Based on the size of a human ankle, we found that we had to abandon the original design which was very bulky and didn’t fit in human proportions. As a replacement for this design, we switched to creating a four-bar mechanism that would actuate the pitch of the foot, which will be discussed further in the next section.

4.3.3 CAD

For this iteration, we used the same process as Iteration 2, but this time, we were designing with the new joint order and positions in mind and making sure that the hip and ankle joints had intersecting axis of rotation. We were also attempting to make the overall hip design more compact in order to make the legs fit within human proportions.

In terms of the standards used, the only differences between the standards used in Iteration 2 and this iteration are the size of the gearbox, motor, and centers of the pulley with everything else staying the same. Since the harmonic drive gearbox is only one stage, the total length of the gearbox changed to 73mm with the diameter remaining at 90mm. The reason for this change will be discussed in more detail in the next Harmonic Drive section. At this point of the project, the Controls team also finalized their motor selection, so for the model, we used a placeholder with a size of 74mm x 63mm Ø. Lastly, since the harmonic drive required a 2:1 pulley, we selected a set of pulleys and calculated the distance between the centers of the two pulleys based on the minimum

distance required between the centers for the gearbox and motor dimensions and the available timing belts on McMaster Carr. These calculations are shown in Appendix E.

4.3.3.1 Hip Joints

For the hip joints, the only major change was that it accommodated for a single stage harmonic drive. Since this change was relatively easy to adjust in the model, the brackets for the hip joints were adjusted directly in the manufacturing model.

4.3.3.2 Knee Joints

Some small changes were made to accommodate the single-stage harmonic drive, but those changes were directly done on the manufacturable parts (discussed further in Chapter 6.2.2).

4.3.3.3 Ankle Joints

Version 3 of the ankle was an attempt to get the joint axes to intersect while keeping the same pitch design. After another meeting with the advisors, there was some concern with how far away the roll and pitch axes were in the previous version. Therefore, the best course of action was to just find a way to make the axes intersect. The challenge was that the pitch design we were using was bulky, and there were not many options to attach the roll axis to the pitch.

We settled with trying to attach a shaft to the bottom plate of the pitch and have the "home" position of the pitch be bent. From there, the shaft is attached to the motor via an adapter, as the motor shaft is a different diameter to the roll shaft. We made a bracket for the motor to mount on, plus an additional bracket in between to help support the shaft, as the weight of the entire robot is being put on the ankle.

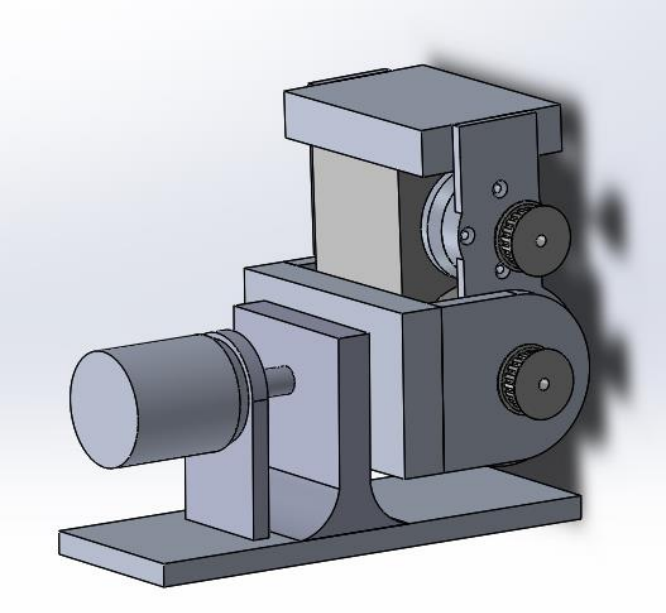


Figure 4.17: Ankle V3

This design still had problems and did not last long before switching to a different design. The first issue is that there is no support in the back of the ankle. While we knew that would be an issue, we did not have space to put any support in the back without making the foot bigger than it should be. In addition, we still did not have a gearbox for the roll motor, which at this point was when we realized that the roll motor needs way more torque than originally thought. We were going based off our initial torque calculations, which were incorrect in terms of the ankle roll.

4.3.3.4 Leg Bones

At this point in the iteration, we did not change the leg bones as we were still using the same femur, tibia, and fibula from the previous iteration. We did not run into any issues with the leg bones while going through our third iteration of joint designs.

4.3.3.5 Foot

Up until this point, a stand-in foot was being, where it had the basic dimensions of a foot. However, since we were going into version 3 of our parts, we began designing a foot that would be more suitable for our robot in terms of functionality. Since we wanted the foot to fit into a shoe, we decided to model our foot after an insole of a shoe, as shown below. The front of the foot was curved upwards slightly to act as the “toes” of the foot, since we are not going to have the toes actuated due to spacing reasons.

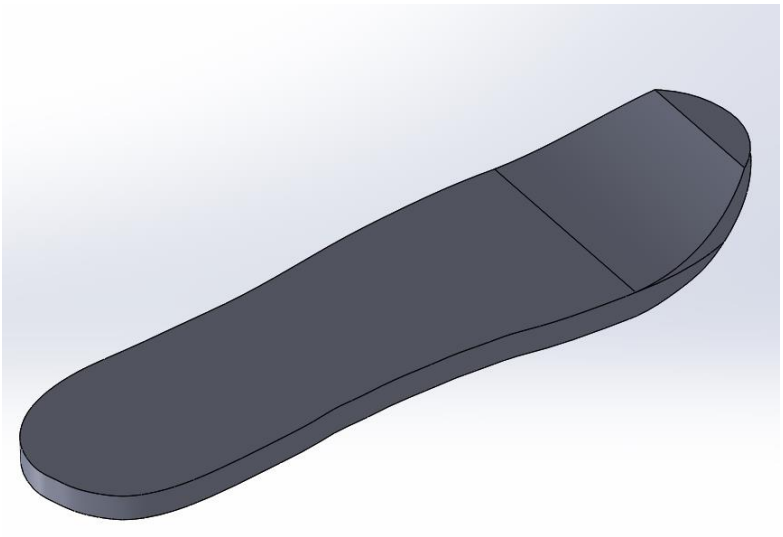


Figure 4.18: Initial Foot Design (based off size 10 US men's) without holes for the rubber stoppers

Along with the foot came the design for attaching the FSR sensors that the sensors team had been working with. They had informed us that they would like to have some rubber padding between the sensor and the ground as they did not want to risk having the sensor break. Therefore, we came up with a design to have a rubber padding push against the sensor when the foot meets the ground, which is shown below. In the diagram, the red represents the rubber pad, the blue is the FSR sensor, and the black is the foot/other aluminum plates.

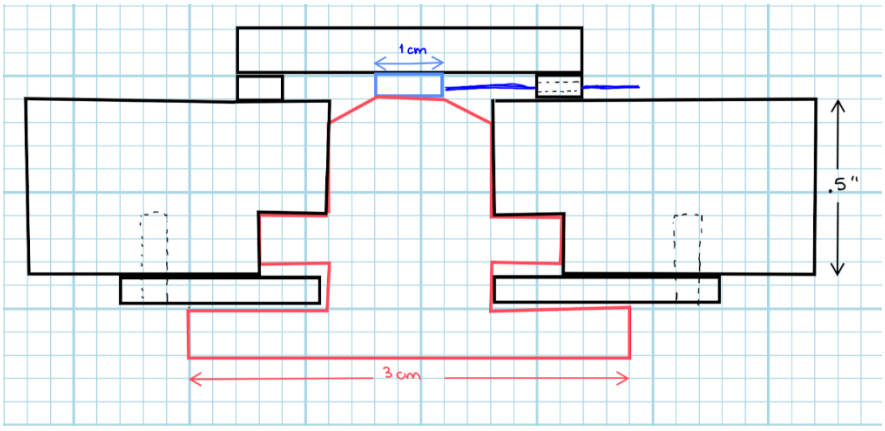


Figure 4.19. Detailed sketch of the rubber inserted into the foot piece with the sensor braced.

The idea of the design is that the rubber is slotted between some plates that allow it to move up and down slightly. When the rubber is in the "down" position, it is barely touching the sensor. When the rubber is pushed up (i.e., the robot takes a step), it contacts the sensor. However, the rubber padding can technically move up more, which is done on purpose as we wanted to ensure

that the rubber is making good contact with the sensor or else we risk having bad readings. With the sensor, the idea is to have it slotted underneath an aluminum plate, where the wire is able to stick out through the slot to connect to whatever circuit sensors has.

After making this design, it was then implemented onto the foot as shown in the figures below. The top figure shows the plates with the slots for the sensors to go under. These plates also act as the attachment points for the ankle, which is further discussed in the manufacturability chapter. The bottom figure shows the placement points for each of the rubber padding, which were verified by the sensors team as the optimal placements for the FSR sensors.

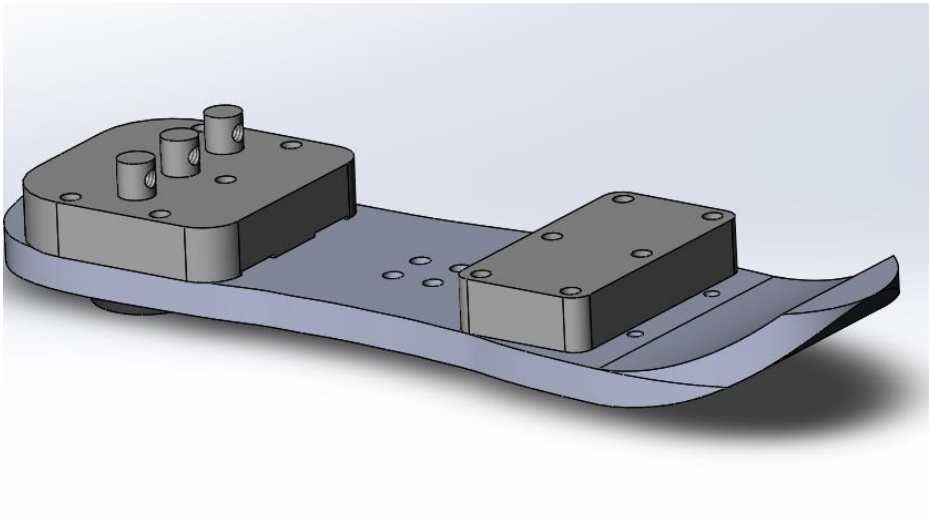


Figure 4.20: Foot design with the sensor attachments

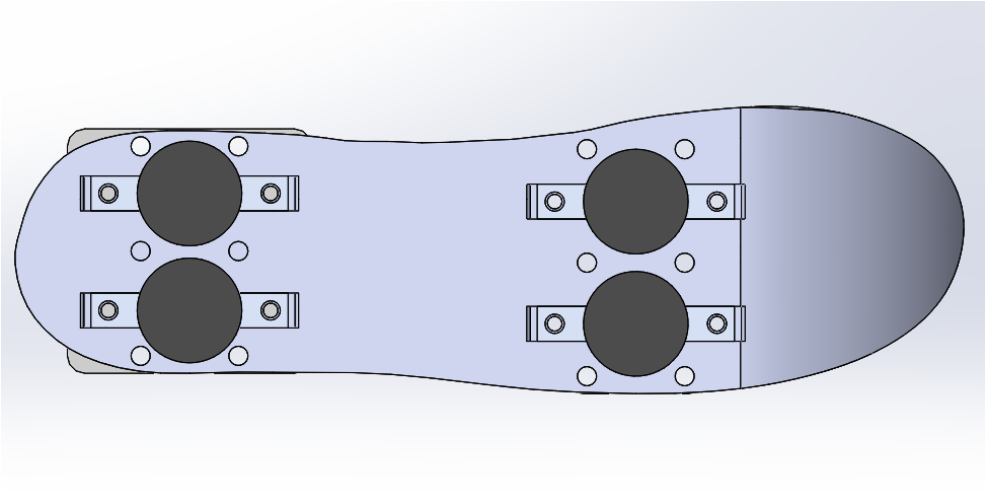


Figure 4.21: Bottom of the foot design with sensor attachments

4.3.4 Materials

Between design iterations 2 and 3, we did not feel the need to re-evaluate our materials selection, so we continued to design with 6061 aluminum alloy in mind, while occasionally looking to see if any parts could viably be made with 3D printed PLA.

4.3.5 Cost

With iteration 3, based on using a 2-stage 40:1 planetary gearbox and a 1:2 pulley system, the updated budget of the design requirements is shown below in Figure 4.22. An alternate supplier was found for the 24”x24” sheet metal stock, with a lower price. The cost breakdown from the sensor and control teams is shown in Figure 4.23.

Budget: $7000			Total Cost	$7,260.79
Item	**Used For**	**Cost**	**Quantity**	**Total**
Aluminum Rod - 1.5" x 6ft	Tibia, Femur	$103.90	1	$103.90
Aluminum Rod - 0.5" x 3ft	Fibula	$6.04	1	$6.04
Aluminum Sheet - .5" x 24" x 24"	Joint brackets	$233.25	4	$933.00
1/2" Shaft Bearing	Joint brackets	$7.11	18	$127.98
1/4" Shaft Bearing	Joint brackets	$6.79	18	$122.22
M2 2.5mm Set Screws 25 pack	Joint brackets	$10.00	1	$10.00
1/4" Steel Rod 6 ft	Joint brackets	$8.02	1	$8.02
1/2" Steel Rod 1 ft	Ankle roll joint	$5.29	1	$5.29
M5x.08x25mm flat screws (100ct.)	Everything - not drive	$12.47	3	$37.41
M5x.08x25mm rounded screws (100ct.)	Harmonic Drive	$14.99	2	$29.98
M5 nylock nuts (100ct.)	Mostly everything	$16.62	3	$49.86
M4x0.7x18mm rounded screws (100ct.)	Motor screws	$11.29	1	$11.29
3D Printing	Everything	$50.00	1	$50.00
M5x0.8x35mm rounded screws (25ct.)	Harmonic Drive	$7.63	2	$15.26
14 Tooth Pulley	Joint brackets	$20.64	10	$206.40
28 Tooth Pulley	Joint brackets	$37.43	10	$374.30
15" Timing Belt	Joint brackets	$6.08	10	$60.80
10mm-10mm Universal Joint	Ankle 6-bar linkage	$11.99	8	$95.92
VersaPlanetary Gearbox	Actuation	$90.00	10	$900.00
8-32 1" flat screws (50)	Gearboxes	$12.06	2.00	$24.12
Design Total Cost		Goal: <$3000		**$3,171.79**

Figure 4.22. The iteration 3 cost breakdown with the single-stage harmonic drive and ratioed pulley system.

SENSOR TEAM: FSR Sensor		Cost	Quantity	Total
8 FSR Sensors (100Lbs)		$22.00	8	$176.00
				$0.00
2 Rubber Insoles		$10.00	2	$20.00
Sensor Total Cost				**$196.00**
CONTROL TEAM		**Cost**	**Quantity**	**Total**
Motor		$115.00	12	$1,380.00
Motor Controller		$250.00	6	$1,500.00
Encoders		$20.00	12	$240.00
CAN Transciever		$23.00	1	$23.00
Battery / BMS		$500.00	1	$500.00
Cabling		$250.00	1	$250.00
Control Total cost		Goal: <= 4000		**$3,893.00**

Figure 4.23. The cost breakdown of control and sensor teams.

4.3.6 Motors

During design Iteration 3, the controls team made a final decision on a motor, the Flipsky E-Board Motor 6374/190KV. The 6374 is a brushless DC motor that runs on 3 phase DC current, has a stall torque of 8 N*m, runs at a maximum voltage of 50V, and can spin at a speed of 190 rpm/V. Assuming a required static joint torque of 160 N*m, we can run the motor at 2 N*m and a gear ratio of 80:1 to actuate the robot.

4.3.7 Harmonic Drive

Although the mass of the robot and each section was still changing, we used the mass of the full lower body assembly in Iteration 2 as a reference to calculate the required torque, and since the Controls team was able to finalize their motor selection, we were able to finalize on the required gear ratio. From our Iteration 2 lower body assembly, the torque required to lift the pitch of the hip out 90 degrees was 80N*m. Therefore, we decided to use 160N*m of torque as our reference for the torque requirement of each joint. This ensured that even if there was a joint that needed more than 80N*m of torque, the design would still be able handle the difference. Since the motors have a stall torque of 8N*m, we estimated that the nominal torque would be 4N*m since we don't want our motors to be anywhere close to stall torque. Therefore, we needed a 40:1 gear ratio from the output of the motor to the output of the gearbox. Based on the design from Iteration 2, every motor must connect to the gearbox with a timing belt pulley in order to keep the overall design within human proportions. Therefore, we could use a single stage of the harmonic drive gearbox which has a 20:1 gear ratio and have the timing belt pulley have a 2:1 ratio to achieve the total 40:1 gear ratio.

From the harmonic drive gearbox design in Iteration 2, we increased the structural integrity of the points of failure by increasing the thickness. Additionally, one of the observations we made while doing the initial torque tests of the gearbox was that the circular spline was flexing along with the flexspline which made the gearbox easily backdrivable. Since this is one of the key features of the gearbox, we decided to increase the thickness of the circular spline's perimeter to resemble more of a circle rather than a flower. Due to this increased stiffness of the circular spline, we found that it was incompatible with the flexspline and wave generator for the previous design, so we had to revisit the tooth profile of the flexspline and circular spline.

From our further research, the circular spline has a different tooth profile than the flexspline where the rigid circular spline has a conjugate tooth profile to the flexspline. This aspect of the harmonic drive gearbox is one of the trade secrets of companies that manufacture these gearboxes. However, we were able to find multiple papers on how to calculate the trajectory of the flexspline teeth and conjugate tooth profile. Although these tooth profiles may not be as efficient as the gearboxes sold by other companies, we might have still been able to use the tooth profiles for our robot. Based on the equations in Appendix D, we were looking for the pressure angle of the conjugate tooth profile. Throughout the calculation, we used the parameters of the existing flexspline we made as well as certain assumptions for the parameters we wanted to optimize while still leaving room for error in 3D printing the components. These assumptions were based on how the variables are defined in the reference paper and our intuition of the manufacturing tolerances of 3D printing. By the end of the calculations and after adjusting the assumptions a few times, we ended up with a pressure angle of 29 degrees.

After adjusting the model of the circular spline to reflect the 29-degree pressure angle, the prototype was assembled again and ready for torque testing. We used the same testing set up as Iteration 2, but instead of pushing the output metal plate downward, we were trying to lift a backpack at the end of the lever arm which has a measurable weight. Unfortunately, after attempting to manually crank the gearbox to lift the backpack, the flexspline immediately shattered which we suspected the issue coming from the material of the flexspline. Since PLA is a brittle material, it doesn't handle a lot of strain very well, so when attempting to lift the backpack, not only was the component weak due to the wave generator forcing the flexspline to bend but it also had to resist the stress of torque applied on the output lever arm. Therefore, we made the same

flexspline out of nylon since nylon has a much higher resistance to mechanical strain, meaning it can flex without compromising the strength of the component. However, after testing this new prototype, the flexspline still broke at about 24N*m of torque. This meant that we would need to find a different way of increasing the torque of each joint's transmission while still being within a reasonable budget.

Since the 3D printed harmonic drive gearbox won't be able to withstand the torsional loads experienced by each joint, we decided to look for planetary gearboxes that could achieve the necessary loads while also being cost effective. Therefore, we are currently getting one gearbox to verify and test the mechanical loads. These tests will have to be conducted in B term.

4.4 Redesign for Manufacturability

During the second design iteration, we started to realize that many of the parts we were designing would not be easily manufacturable out of the stock we wanted to use. While many of the parts could have been milled out of large blocks of aluminum, this would incur significant milling times and increased material costs, due to the wasted material. Therefore, we decided to redesign the parts to cut them out of sheet stock and make finishing passes on a mill. With these new goals in mind, we focused on making each of the more complex parts manufacturable out of sheet stock during design iteration 3, and made our new parts fit to our new manufacturability goals.

4.4.1 Overall Design

For the last step of our design process, we broke down the model of each bracket and component into smaller parts so that we could easily machine them. We decided to break them into small pieces rather than making each bracket from one block of stock material in order to reduce the total cost of the robot. Making smaller pieces will reduce the amount of wasted material as well as make the overall design of the robot more modular. This allows us to make any minor adjustments or changes without needing to completely remove an entire section of the robot.

Like the last iteration of modeling, we made a set of standards to follow to make assembly easier. When breaking a standard U-shaped bracket down into smaller parts, the sides of the bracket are to stay the same size such that the fasteners are screwed into the bottom section of the bracket. We also decided to use M5 screws to fasten the brackets together since M5 screws have a high tensile strength while still being small enough to keep the overall design compact. Each bracket component is also fastened together with 3 or more M5 screws to distribute the load experienced by the screws. The length of each screw was also defaulted to 25mm in thread length since it was the longest screw that was also the most cost effective; however, if a certain bracket

design required a different size screw, we would attempt to keep the number of different sized screws to a minimum.

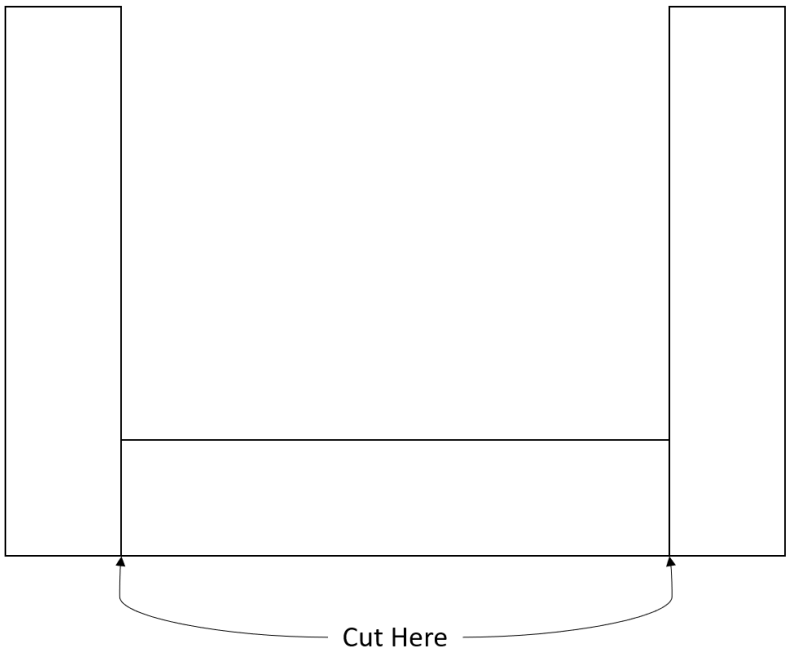


Figure 4.24: Bracket splitting standard

4.4.1.1 Joints

Several of the joints were redesigned in sectioned pieces that could be manufactured easily with the available tools in Washburn and in Prescott, but each part in the SolidWorks model was redone to be manufacturable, which includes the milling or cutting of the pieces as well as the holes for attachments. There were also some major changes to the ankle and foot design from the last iteration, which will be detailed further below.

4.4.1.2 Hip

The hip originally consisted of 3 pieces, with 2 U-shaped brackets and one that resembled 2 U-shaped brackets fused together at a 90-degree angle. After the redesign, the first and third stages were split into 3 flat pieces (not counting hardware), and the second stage was split into 5 pieces. An exploded view of the second stage can be found below in Figure 4.25.

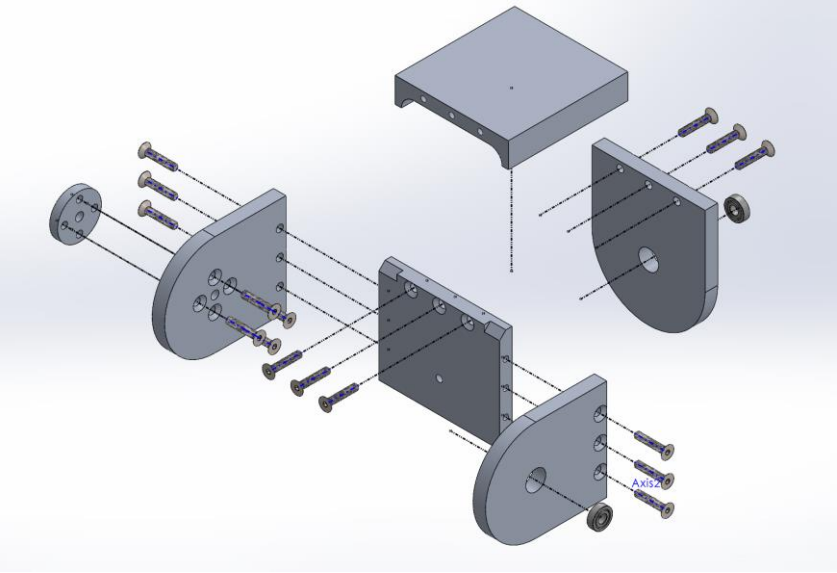


Figure 4.25: Exploded view of Hip Bracket Stage 2

4.4.1.3 Knee

Over the course of the manufacturability redesign, the knee was slightly simplified. It consists of 2 u-shaped brackets when fully assembled, with the motor above the gearbox. Other than that, not much changed from the previous version.

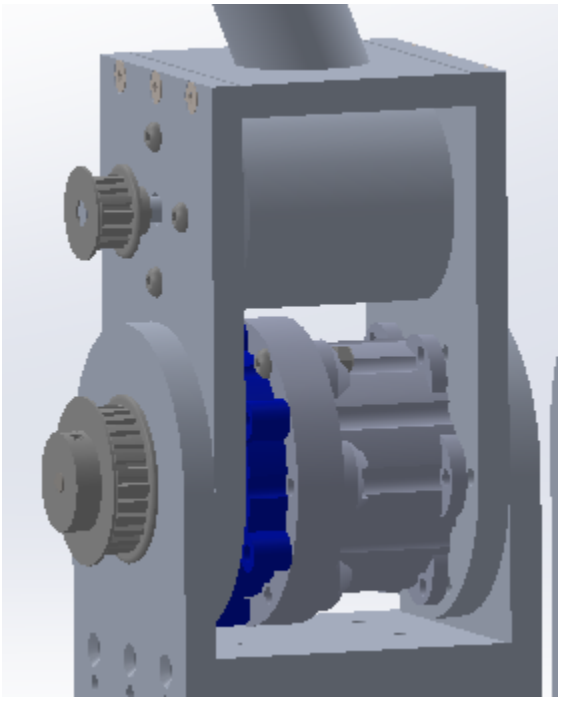


Figure 4.26: Manufacturable Knee Joint

4.4.1.4 Ankle

During the next redesign, we realized that the ankle needed to provide more torque than we were allowing for in our initial designs. Therefore, we went for a total redesign, going through several possibilities before finally settling on a 6-bar linkage, capable of actuating both pitch and roll with two motors controlling the one linkage. The linkage utilizes motor and gearbox mounting which follows the manufacturability standard, and the links relate to universal joints. The ankle joint can be seen below in Figure 4.274

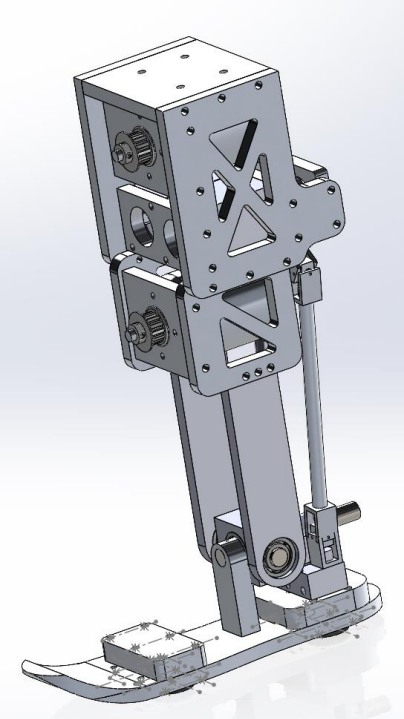


Figure 4.27: Full ankle and foot assembly

4.5 Cost

The overall budgeting table did not change between the iteration 3 and the new manufacturing redesign, other than a change from using the 3D printed harmonic drive gearboxes to using the VEX versaplanetary gearboxes. The cost spreadsheets in Figures 4.28 and 4.29 already reflects this change.

Budget: $7000			Total Cost	$7,543.67
Item	**Used For**	**Cost**	**Quantity**	**Total**
Aluminum Rod - 1.5" x 6ft	Tibia, Femur	$103.90	1	$103.90
Aluminum Rod - 0.5" x 3ft	Fibula	$6.04	1	$6.04
Aluminum Sheet - .5" x 24" x 24"	Joint brackets	$233.25	4	$933.00
1/2" Shaft Bearing	Joint brackets	$7.11	18	$127.98
1/4" Shaft Bearing	Joint brackets	$6.79	18	$122.22
M2 2.5mm Set Screws 25 pack	Joint brackets	$10.00	1	$10.00
1/4" Steel Rod 6 ft	Joint brackets	$8.02	1	$8.02
1/2" Steel Rod 1 ft	Ankle roll joint	$5.29	1	$5.29
M5x.08x25mm flat screws (100ct.)	Everything - not drive	$12.47	3	$37.41
M5x.08x25mm rounded screws (100ct.)	Harmonic Drive	$14.99	2	$29.98
M5 nylock nuts (100ct.)	Mostly everything	$16.62	3	$49.86
M4x0.7x18mm rounded screws (100ct.)	Motor screws	$11.29	1	$11.29
3D Printing	Everything	$300.00	1	$300.00
M5x0.8x35mm rounded screws (25ct.)	Harmonic Drive	$7.63	2	$15.26
14 Tooth Pulley	Joint brackets	$20.64	10	$206.40
28 Tooth Pulley	Joint brackets	$37.43	10	$374.30
15" Timing Belt	Joint brackets	$6.08	10	$60.80
10mm-10mm Universal Joint	Ankle 6-bar linkage	$11.99	8	$95.92
VersaPlanetary Gearbox	Actuation	$90	10	$900
Design Total Cost		Goal: <$3000		**$3,397.67**

Figure 4.28. The Design Cost Table.

SENSOR TEAM: FSR Sensor		**Cost**	**Quantity**	**Total**
8 FSR Sensors (100Lbs)		$22.00	8	$176.00
				$0.00
2 Rubber Insoles		$10.00	2	$20.00
Sensor Total Cost				**$196.00**
CONTROL TEAM		**Cost**	**Quantity**	**Total**
Motor		$115.00	12	$1,380.00
Motor Controller		$250.00	6	$1,500.00
Encoders		$20.00	12	$240.00
CAN Transciever		$80.00	1	$80.00
Battery / BMS		$500.00	1	$500.00
Cabling		$250.00	1	$250.00
Control Total cost		Goal: <= 4000		**$3,950.00**

Figure 4.29. The sensor and control team cost tables.

4.6 Weight and Joint Torques

Based on the SolidWorks model, the total weight of the robot is a little over 27kg; however, due to the multiple changes and lack of time, we could not finish the model to include the correct shafts and use planetary gearboxes instead of harmonic drive gearboxes. This means that we can only use the 27kg as a reference. The true mass of the robot is likely closer to 29kg or 30kg. Although the model is not complete, we can still use the model as a reference to calculate and locate the center of mass of different sections of the robot. Therefore, based on the model and

estimated additional weight due to the missing parts, we made a diagram of the joints and linkages connecting them, putting the center of mass locations and mass at the respective locations on the corresponding linkage.

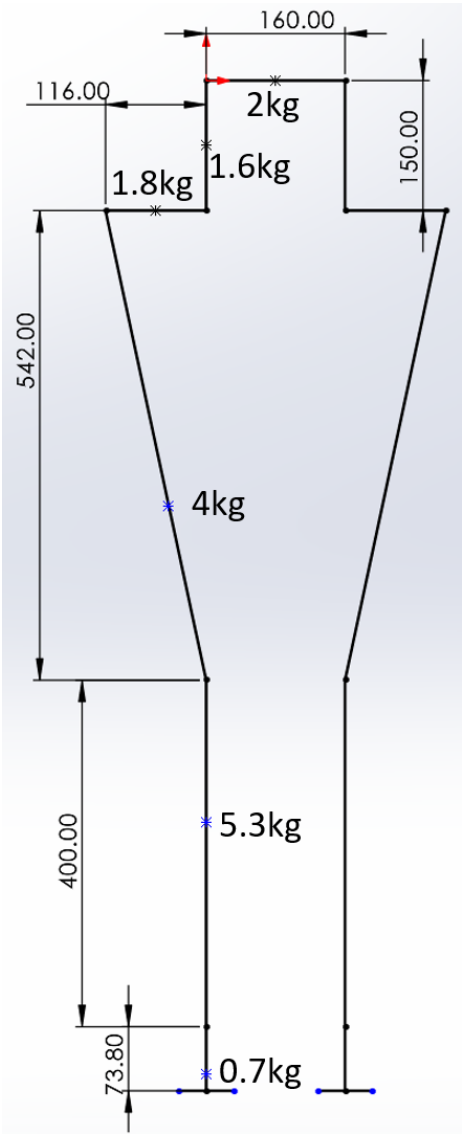


Figure 4.30: Center of mass diagram of robot linkages

Using the diagram, we could calculate the amount of required torque each motor would need to apply given a certain configuration. Due to the constantly changes and time constraints, we analyzed the model in a static state where none of the linkages are moving. The configurations we analyzed were made such that the joint being analyzed would experience the most amount of torque possible. This meant that the configurations are the worst-case scenarios, so if the calculated torque required is less than the maximum torque of the transmission of 320Nm, the joint will be

able to move the necessary linkages at a reasonable angular acceleration. Additionally, even if the static torque analysis determines that the required torque is either close to or more than 320Nm, the planetary gearbox that we are planning on testing has the optional configuration of having a 100:1 gear ratio without increasing the weight or size which means that we could increase the maximum torque output of the transmission to 1600Nm without needing to redesign the brackets. For each configuration, we're assuming that the stability of the robot does not affect the analysis which means that one of the robot's feet is grounded to the floor. This assumption means that we are ensuring that the robot's joints will be able to resist movements that could cause stability. For example, if the robot was attempting to take a step but was doing a split, the joints would be able to resist the legs from moving further apart from each other. Additionally, the analysis only accounts for the weight of the skeleton. We know that additional weight will be added to the top of the robot for the battery and electronics that the Controls team will need to attach; however, since we don't know exactly how much additional weight this is, we assumed that if the required torque that we calculate in the robot's worst case scenario is less than the maximum torque of the transmission, then there would still be a enough torque for the expected movements and configurations of the robot. Lastly, we also used the maximum range of motion (ROM) of our model as the limits to the angles of the joints.

Starting with the hip joints, the worst-case scenario for the pitch of the hip is went the robot lifts one leg as far as possible. The configuration and calculations can be seen in Appendix B. Since the resulting required torque was 53.099Nm, there would not be an issue with the angular acceleration for the dynamic motion of the robot's legs with the 40:1 transmission.

For the hip roll, the worst-case scenario was when the robot moves apart its legs as far as the ROM allows it. Since this configuration requires both hip roll joint to apply torque, the torques were analyzed one at a time. Like the hip pitch joint, the hip roll joint's maximum torque requirement is relatively low at 60.799Nm as shown in the calculations in Appendix B.

For the hip yaw, the static configuration was negligible compared to amount of weight that the other joints would need to resist. Therefore, it was more beneficial to calculate the highest angular acceleration that the joint could apply which we calculated to be 8.2 rad/s^2 since the moment of inertia is just the masses of the linkages that were not along the hip yaw's axis of rotation. These calculations were also assuming that the hip roll was not fully extended, but these

calculations provide a reference for the maximum angular acceleration for the Controls team to verify that their models are physically achievable.

Since the knee joint has the same transmission as the hip pitch joint, we could assume that the knee can handle the required torque since there are less forces affecting the necessary torque compared to the hip pitch joint. Therefore, we could also assume that it can actuate the motion of knee with the necessary angular acceleration that the Controls team may require.

For the ankle joints, we had to analyze both the torque applied at the joint and the necessary torques that the transmission would need to apply for the 6-bar linkage to apply the analyzed torque of the joint. Starting with ankle roll joint, the worst-case scenario is when the robot's legs are stretching its hips as wide as possible. The configuration and calculations of the required torque at the joint and corresponding torques required by the 6-bar linkage actuators are shown in Appendix B. Although the resulting maximum torque calculated is relatively close to the stall torque of the transmission, this joint configuration is not expected when attempting to walk, and therefore, it is unlikely to require an increase in the gear ratio. However, the ankle pitch torque requirement has a much bigger issue.

Due to the movement of taking a step, a bipedal robot's center of mass moves away from the back foot. This configuration is exaggerated and drawn in Appendix B in order to simulate ethe worst-case scenario. As shown in the calculations, the required torque to actuate the pitch of the robot at the worst-case scenario exceeds the transmission's maximum torque output, but we could simply increase the gear ratio of the planetary gearbox to 80:1 to have a total gear ratio of 160:1 in order to reach much higher than the required torque in static. However, while we were solving for the static loads on the 6-bar linkage, we found that the loads experienced by each link far exceed the maximum stress aluminum linkages can handle. This is due to the fact that our center of mass is applying a moment about the ankle joint, so the 6-bar linkage has to apply an equivalent and opposite moment in order to maintain the joint configuration in the diagram. As shown in the diagram, the center of mass is about 800mm from the ankle joint along the x-axis where as the force perpendicular to the ground that the 6-bar linkage has to apply is only about 16mm from the ankle joint along the x-axis. Since the 6-bar linkage has a distance that is 50 times less than the distance of the center of mass, the 6-bar linkage must apply about 50 times the force of the center of mass which is over 13,000N of force. Not only would the actuating link of the

ankle pitch and roll break, but the links of the 6-bar linkage, the universal joints, and set screws would break. Even for the expected joint configuration for stepping where the robot's center of mass is only half a stride away from the ankle joint which is about 500mm, the 6-bar linkage would still need to handle 9000N of force which is still far beyond the maximum stress that it could handle. Unfortunately, we could not simply change the material of the links to carbon steel or increase the thickness of the links because that would still not resolve the issue of the stress on the universal joints and set screws of the mechanism. In order to resolve the issue, the best course of action is to find a way to increase the distance between the actuating joint of the 6-bar linkage and ankle joint. However, the distance required to reduce the stress on the 6-bar linkage enough to account for a margin of error and unexpected loads would be at least 18cm. This would make the ankles and calves of our robot exceed human proportions which is one of the main differentiations of this robot compared to other humanoid and bipedal robots. This is a major issue, but we will have to quickly resolve it in B-term since we had to start the torque calculations towards the end of A-term.

4.7 Finite Element Analysis of Selected Components

To ensure that load-critical components within the robot would be able to stand up to the extremes of the forces applied, we ran Finite Element Analysis (FEA) studies on a selection of parts and some simplified assemblies. Load cases and connections were simplified for the purposes of FEA, and the load magnitudes used were either 300N, 450N, or 600N, depending on the location of a part on the robot. The further from the hips, the higher the number. These loads were chosen based on an assumption that the robot would weigh 60kg. The final weight of the robot appears to be closer to 30kg, so the factors of safety calculated with these studies can be doubled. Additionally, some parts were given a worst-case scenario load of twice what they would expect to experience, specifically in the case of mirrored parts, and those parts will also have a doubled factor of safety.

When showing FEA results in SolidWorks, there is an option to show the deformation of the part, to easier identify the ways in which the part deflects when under load. However, since the magnitude of deformation is proportional to the magnitude of the load, there is a "deformation scale", which is a multiplier that indicates how much more or less visually deformed a part is than would be observed in practice. This deformation was turned on for all images taken of FEA results for ease of interpretation from the figures.

FEA configuration was assisted by a consultation with Dr. Erica Stults in the WPI Academic Research and Computing department.

4.7.1 Pelvis Block

The pelvis block is the central point of the robot, where both legs are mounted. As a result, at any given point, the pelvis block will only have to support the weight of one leg at a time, so this analysis was configured as such: one hip yaw hole was fixed in place, and the other had a surface split around the hole and had a 300N downward force applied. The selected material for the pelvis block was 6061 Aluminum. The setup can be seen below in Figure 4.31.

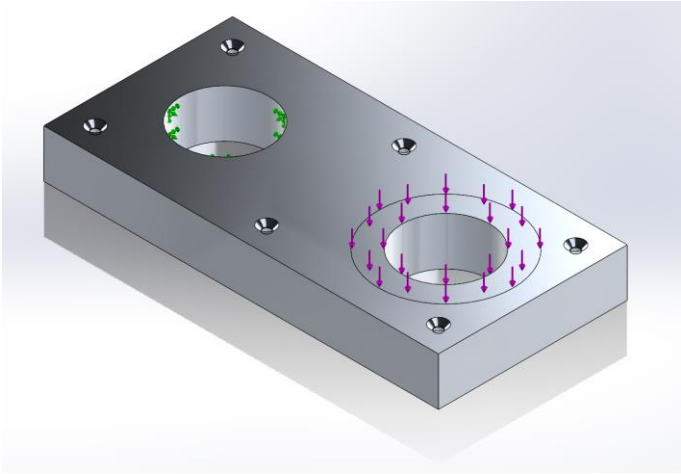


Figure 4.31: Pelvis Block FEA configuration

Once run, the study showed maximum stress of 348.7MPa, well below the yield strength of 6061 Aluminum (listed in SolidWorks as 5.515GPa), giving this part a factor of safety of 15.8. The maximum displacement found in the study was .01829mm, which is effectively negligible. most of the stress was concentrated along the inside of the fixed yaw hole, on the portion closest to the center of the pelvis block. The deformation scale of this study was 1265.1 times real deformation.

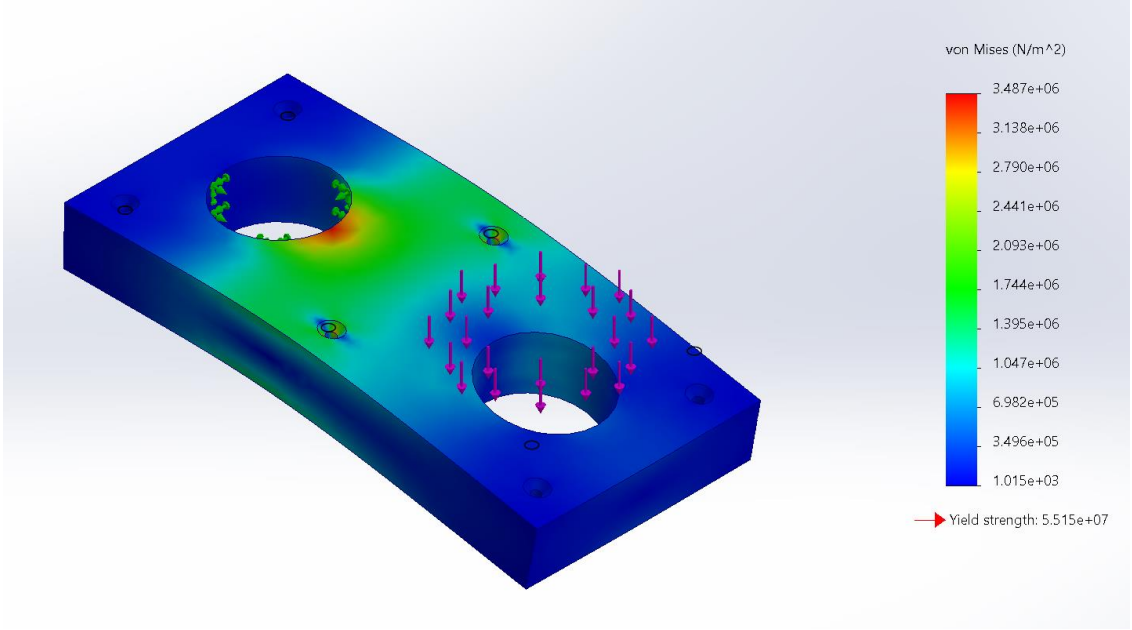


Figure 4.32: Stress results of the pelvis block FEA

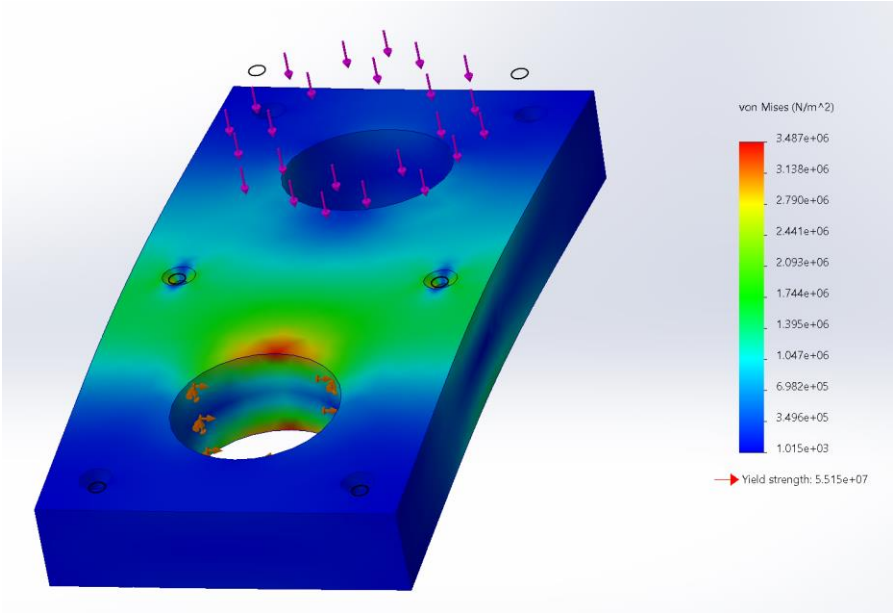


Figure 4.33: Alternate view of Pelvis block stress results

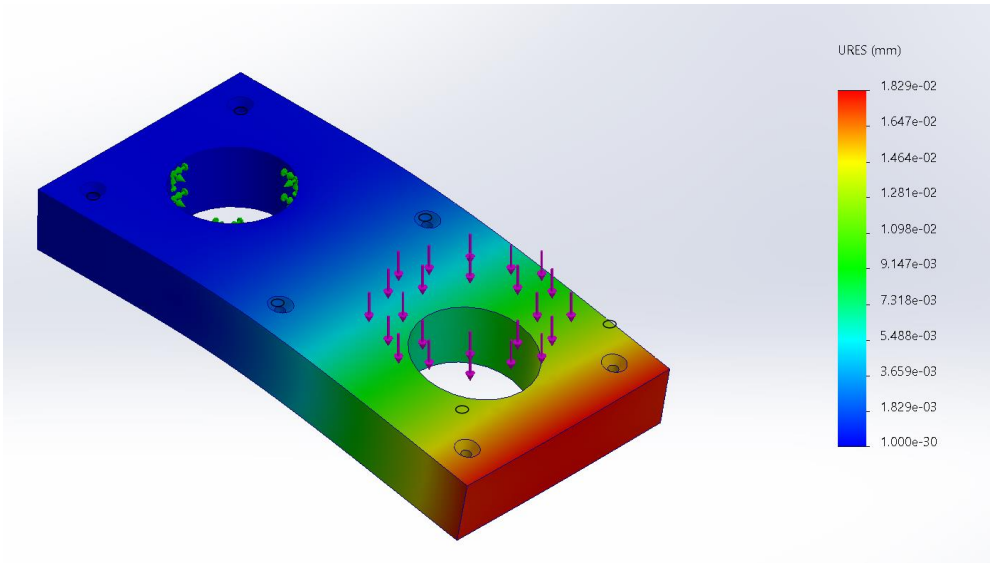


Figure 4.34: Displacement results for the pelvis block FEA

A second study with force applied in an upwards direction to simulate standing on the opposite leg was omitted, because the near-perfect symmetry of the part would result in an almost identical result.

4.7.2 Hip Yaw Mounts

The two parts analyzed in this section are designed to hold the hip brackets onto the pelvis block. Because of the way these parts would be assembled onto the robot, both brackets will only experience downwards forces, when the leg of the same side is not grounded.

The top hip yaw mount has a small chamfer that was set as fixed geometry, since it would be forced up against a bearing in the pelvis block. The inside of the cylinders representing the screw holes then had a 300N force applied, distributed amongst the 6 holes, and directed downwards. The fixed chamfer is shown in Figure 4.35, and the defined forces are shown in Figure 4.36.

This study was initially run with 6061 aluminum set as the material, but the results showed such a significant factor of safety that the study was re-run with a custom PLA material, whose properties can be found in Appendix F.

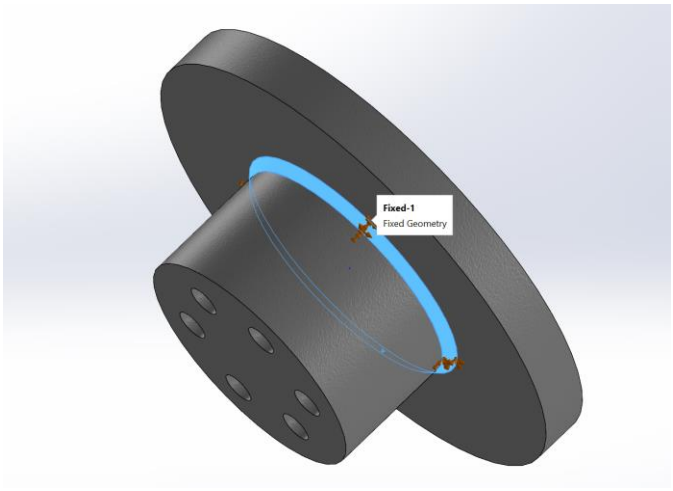


Figure 4.35: Top hip yaw mount fixed chamfer

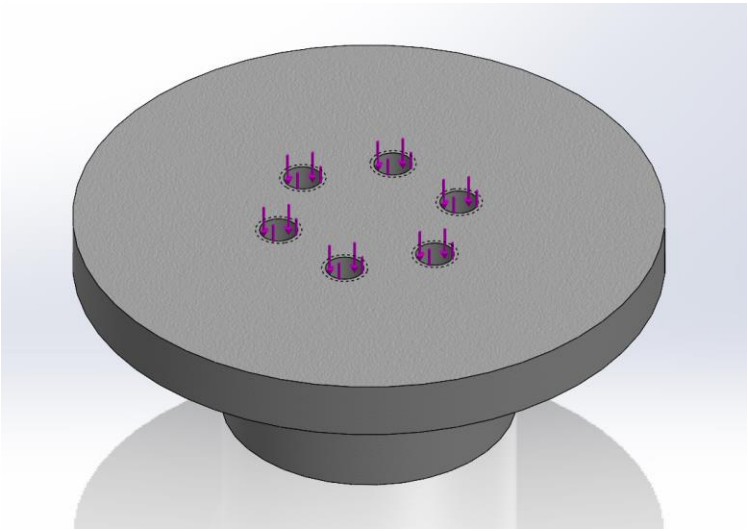


Figure 4.36: Top hip yaw mount force configuration

Once run, the FEA shows that the stresses are concentrated around the bottom of the chamfer, with additional increases in stress inside the screw holes. The deformation of the part is also mostly concentrated in the center, with the center of the part being pulled down and the edges of the rim being rotated up.

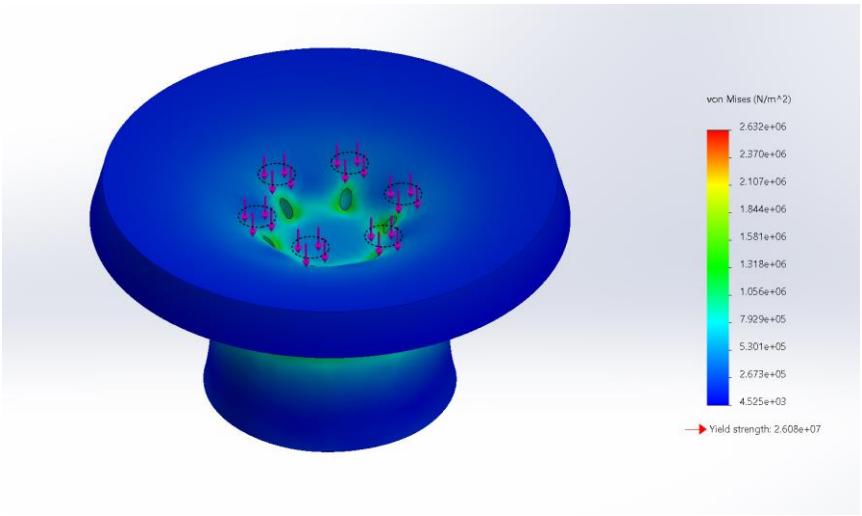


Figure 4.37: Stress results of the top hip yaw mount

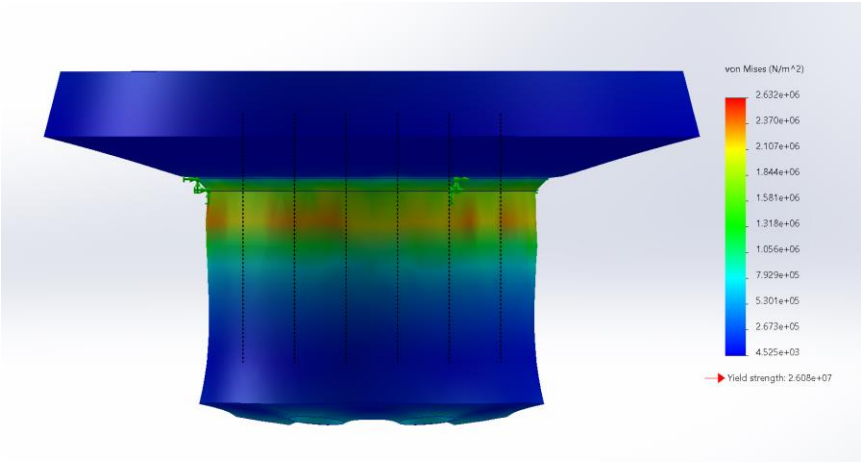


Figure 4.38: Alternate view of the top hip yaw mount stress results

The deformation scale for this study was 1086.15, and the maximum deflection of the part was less than 5 thousandths of a millimeter. The maximum stress experienced by the part was 263.2MPa, resulting in a safety factor of 9.9.

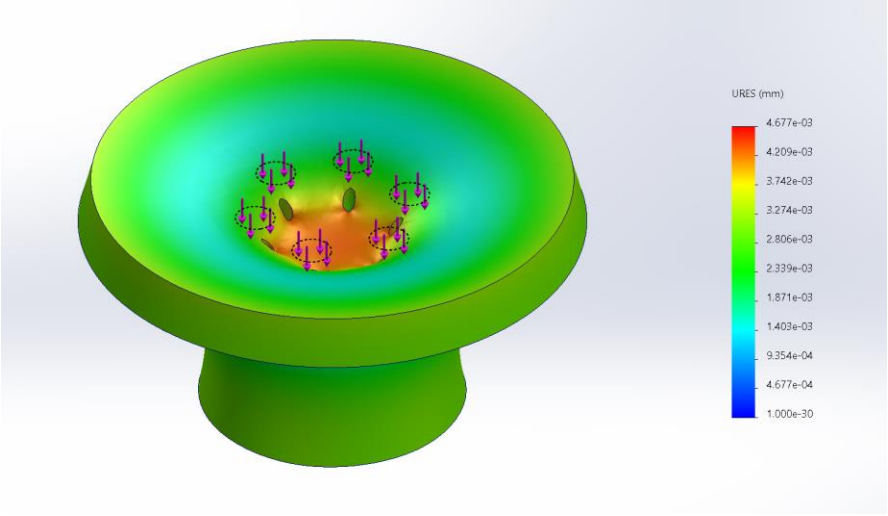


Figure 4.39: Deflection results of the top hip yaw mount

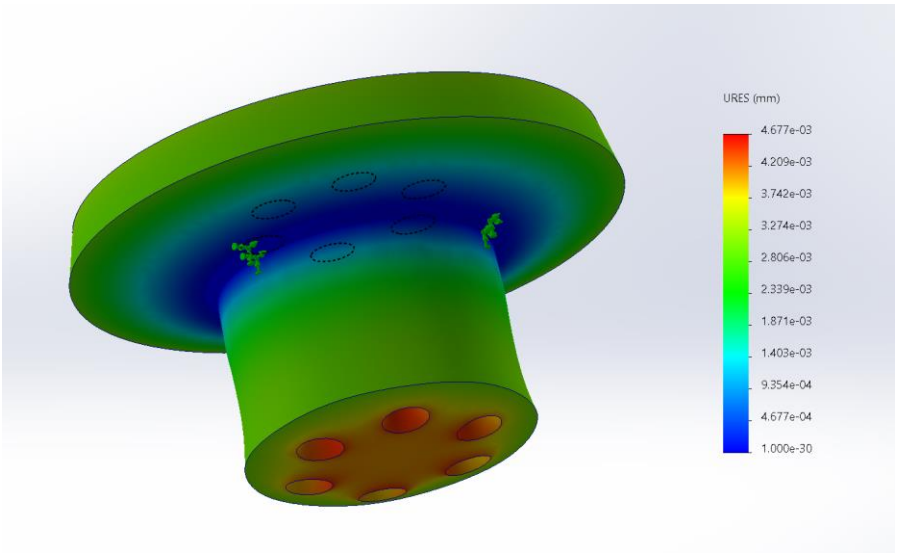


Figure 4.40: Alternate view of the top hip yaw mount deflection results

4.7.3 Hip Bracket Stage 1

The first stage of the hip bracket experiences at most half of the weight of the robot at any given moment. The bracket is split into 3 pieces in a U-shape, meaning that either side should only experience half of the force from the weight at once, but to stress test the parts, a load of 300N was still applied to a single side. The first study run on one side of hip bracket stage 1 applied the force, distributed equally among three bearing loads placed inside the bolt holes at the top, while the shaft hole was fixed in place. Bearing loads is a feature in SolidWorks simulations that apply distributed force along a cylindrical face to simulate the force a cylindrical object would if pushed

in a certain direction. The material for this part was set to 6061 Aluminum. The force and fixture setup can be seen below in Figure 4.41.

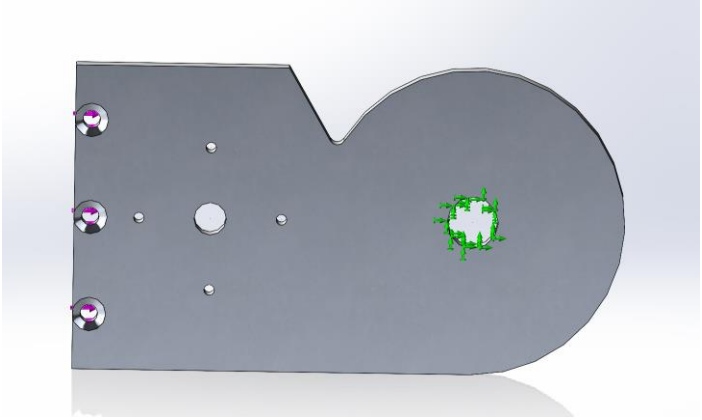


Figure 4.41: Hip bracket stage 1 FEA configuration

The FEA results indicate that the stress on the part concentrates around the divot cut into the part, as well as an increased stress in various areas on the back of the part. The maximum stress experienced by the part was 363.1MPa, which provides a safety factor of 15.2. The deformation scale of this study was 2913.52, and the maximum deflection experienced was about .006mm. Stress results can be seen in Figures 4.42 and 4.43, and deformation results can be seen in Figure 4.44.

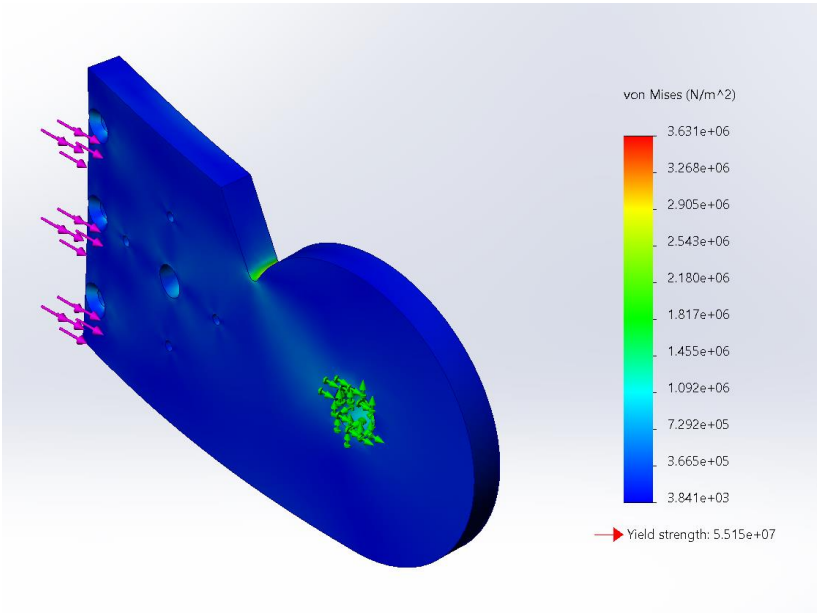


Figure 4.42: Stress results for the hip bracket stage 1

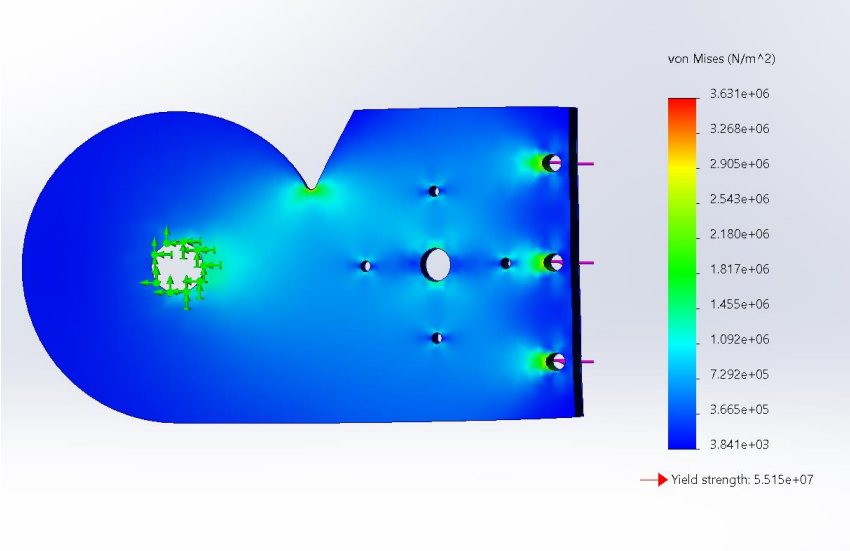


Figure 4.43: Rear view of stress results for hip bracket stage 1

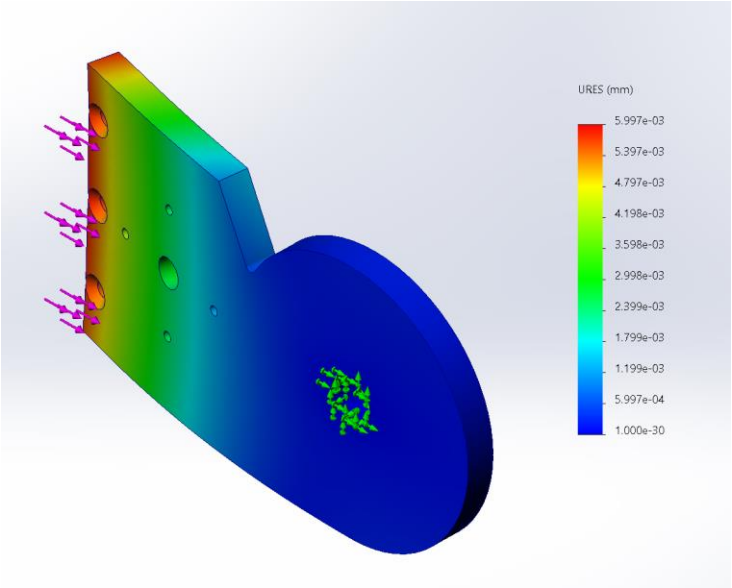


Figure 4.44: Deformation results for hip bracket stage 1

There are two similar pieces in hip bracket stage 1, only one of which had FEA run on it. Because of the large factor of safety, the small changes in the design from one bracket to the other are not expected to cause significant enough structural changes which would result in the other part failing under the same load cases.

4.7.4 Hip Bracket Stage 2

The second stage of the hip has a more complicated shape than the first or third stages, therefore FEA was run using the single part used in robot design Iteration 2, rather than running

an analysis on each individual piece. Upwards forces totaling 300N were applied to the holes that the third stage would rotate about using bearing loads. Screw holes which secured the bracket to the shaft it rotates about were fully fixed in place, and a hole for a bearing opposite of the screw holes was fixed so that it could rotate, but not translate. The fixture and load configuration can be seen in Figure 4.45.

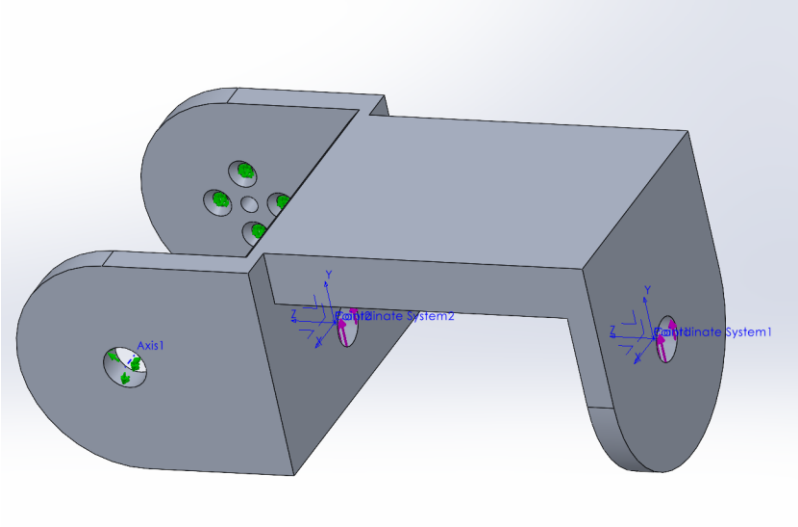


Figure 4.45: Hip bracket stage 2 FEA configuration

The results of the analysis show that the stress concentrates around a corner near the fixed side of the bracket, as seen in Figures 4.46 and 4.47. The maximum stress experienced by the part is 3.419GPa, resulting in a safety factor of only 1.6. However, this stress is only felt in one of the four screw holes, as seen in Figure 4.48, and is not experienced at the corner specified before. The maximum stress experienced in that area is closer to approximately 2GPa, for a factor of safety of 2.8 for those areas.

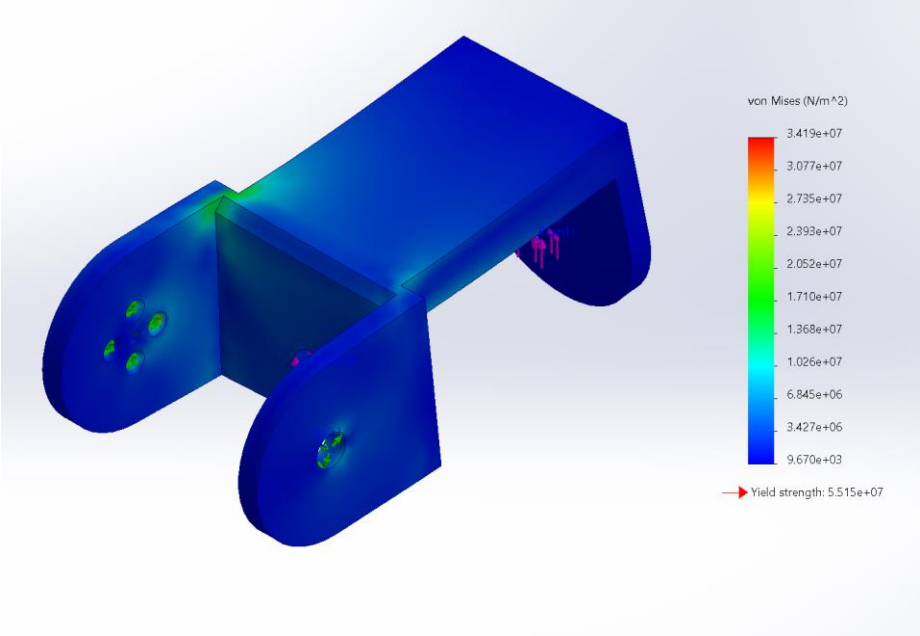


Figure 4.46: Stress results of the hip bracket stage 2

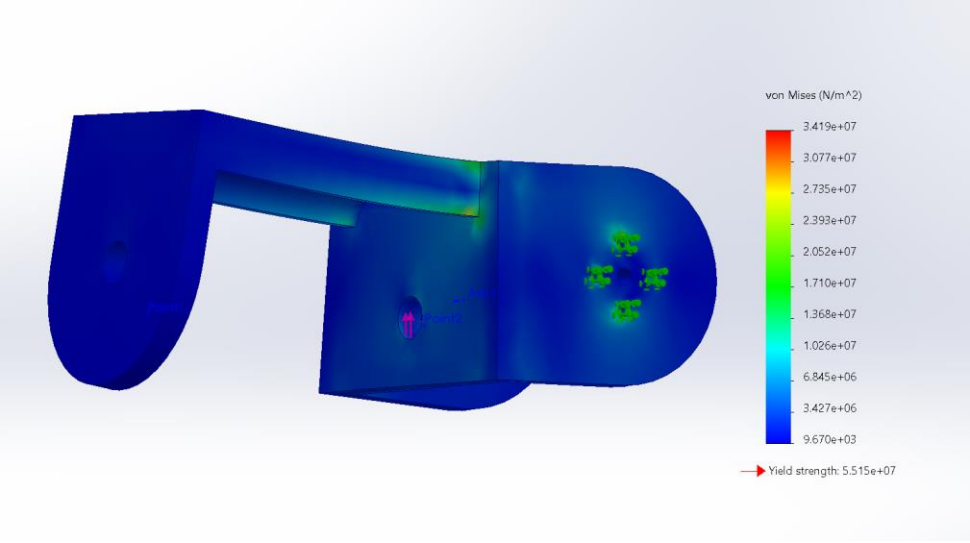


Figure 4.47: Alternate view of stress results of the hip bracket stage 2

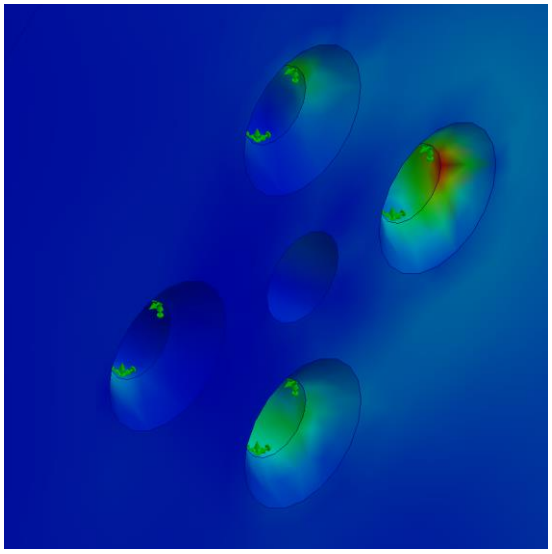


Figure 4.48: Zoomed in view of screw hole stresses on hip bracket stage 2

The deformation scale for this study was 109.32, and the part experienced a maximum deflection of .22mm.

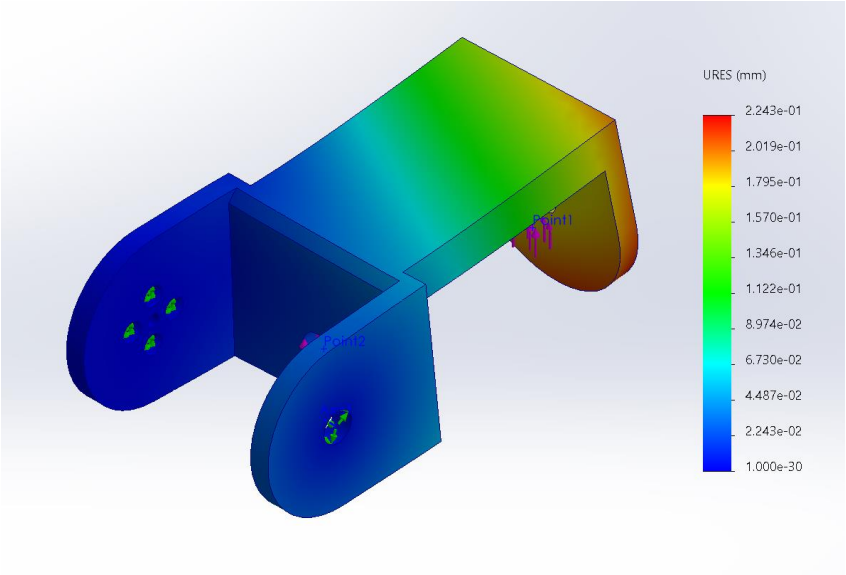


Figure 4.49: Deformation results of the hip bracket stage 2

4.7.5 Femur and Knee Top Plate

To create a more accurate analysis for the behavior of the femur and knee top plate under load, an assembly was used instead of the single part analysis used in other sections. Global interactions were set to “bonded,” meaning that SolidWorks treats the assembly as one piece when creating a mesh for the analysis.

A 450N force was applied to the top of the femur. 450N was selected because due to the physical location of this part, at most 75% of the robot's weight would be applied here. The screw holes in the side of the top plate were set to be fixed, as they were the only point of contact with the rest of the robot underneath. The force and fixture configuration can be seen in Figure 4.50 below.

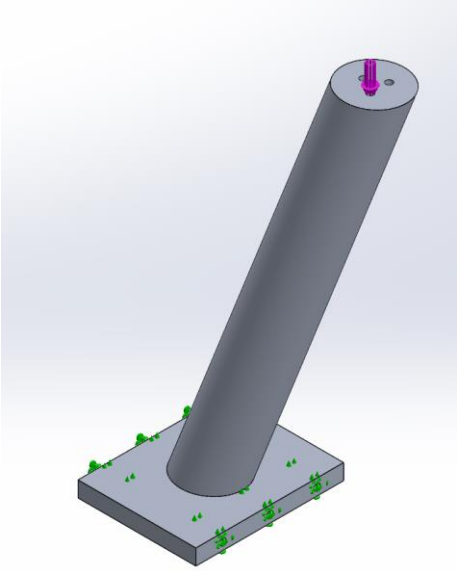


Figure 4.50: Femur and knee top plate FEA configuration

The results of the study show that the forces are concentrated where the femur meets the top plate on the side with the acute angle, and inside the screw holes. The maximum stress experienced by the parts was 3.914GPa, for a safety factor of 1.4, but most of the top plate and femur only experience up to 2.5GPa, which produces a safety factor of 2.2. The distribution of stress can be seen in Figure 4.51, and the concentrated maximum stress can be seen in Figure 4.52.

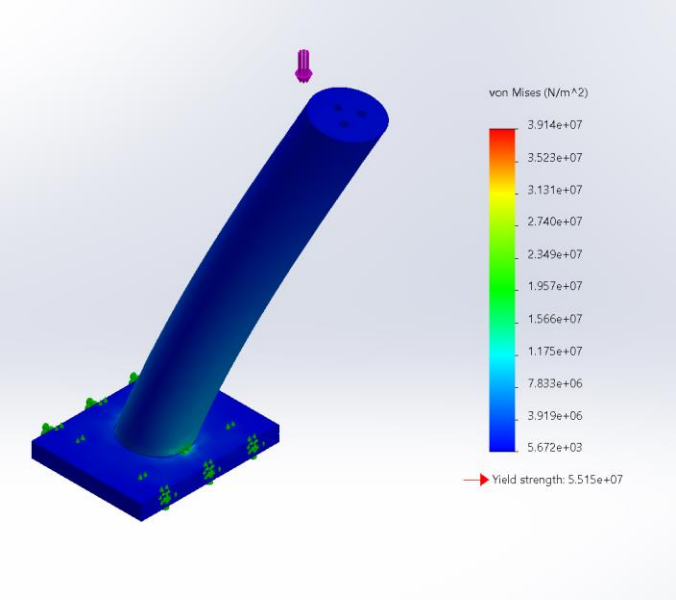


Figure 4.51: Stress results of the femur and knee top plate

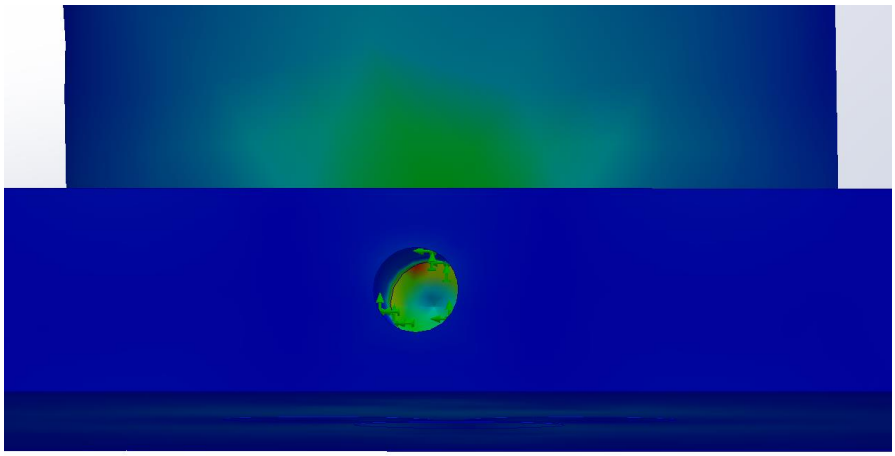


Figure 4.52: Alternate view of concentrated stresses in the femur/knee top plate analysis

The deformation scale of this study was 149.7, and the maximum deflection experienced by the femur was .2mm, as shown in Figure 4.53.

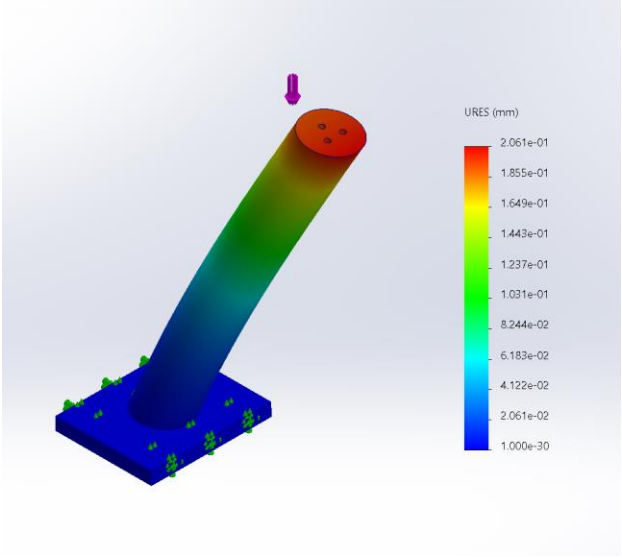


Figure 4.53: Deformation results of the femur and knee top plate

4.7.6 Ankle Supporting Links

There are 3 pieces that make up the supporting links for the ankle. Each of these pieces holds up to approximately the weight of the entire robot, but there are always 2 of these pieces in the structure, supporting half of the weight each. Therefore, the load magnitude used for each piece was 300N.

The first piece is the very top of what constitutes the “ankle assembly”. A 300N force was applied along the top edge to emulate the force coming from the screw holes and the bending the load would cause. The bottommost screw holes, which are where the next piece connects, were fixed in place. The setup can be seen in Figure 4.54.

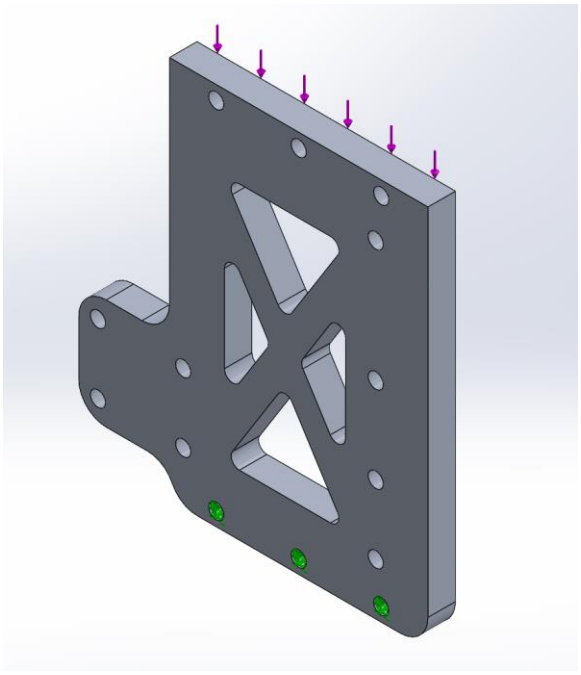


Figure 4.54: FEA configuration of ankle support part 1

The maximum stress seen in the analysis was 831.7MPa, concentrated on the back of the part, as shown in Figures 4.55 and 4.56. The factor of safety is 6.6, and the maximum deflection in the part is .038mm, and the deformation scale in this study was 367.5. The deformation results can be seen below in Figure 4.57.

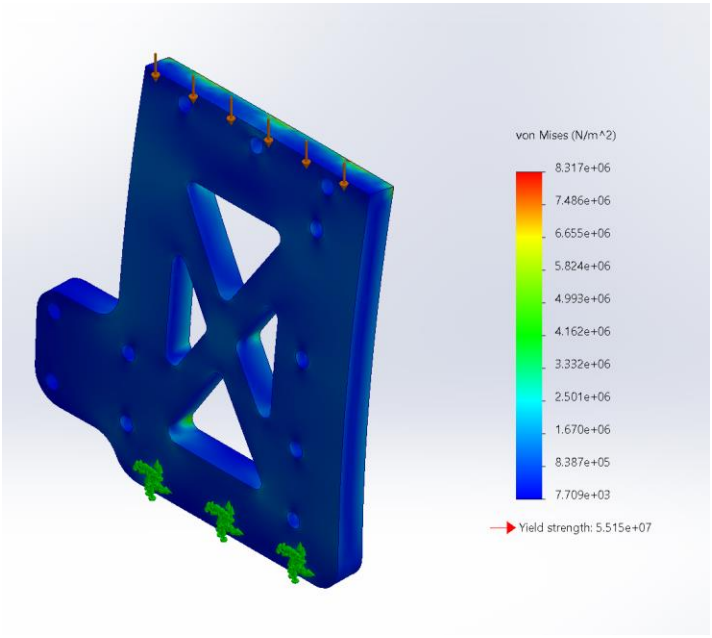


Figure 4.55: Stress results of the ankle support part 1

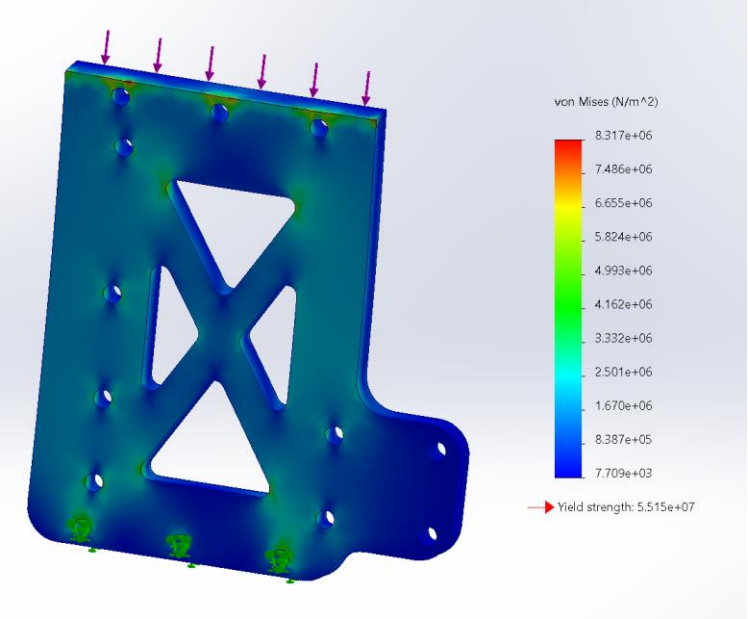


Figure 4.56: Alternate view of stress results of the ankle support part 1

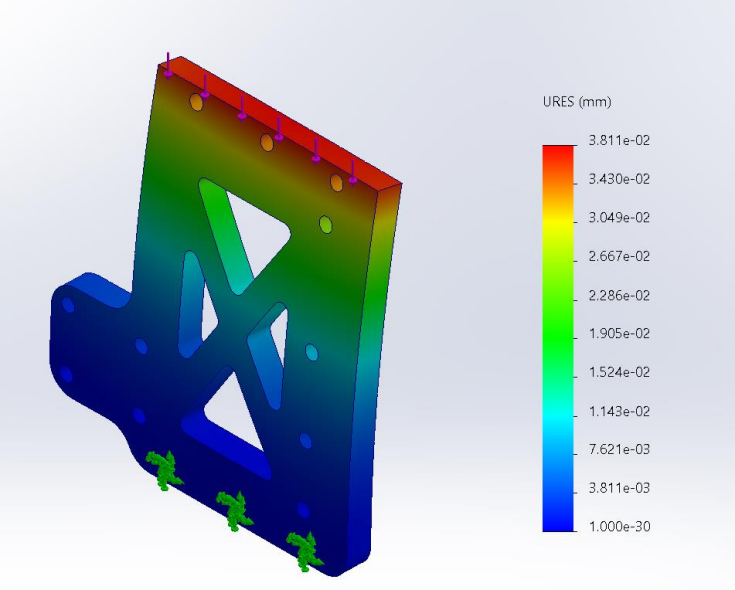


Figure 4.57: Deformation results of the ankle support part 1

The next part has a similar configuration, 300N edge load and fixed screw holes leading to the next part, shown in Figure 4.58.

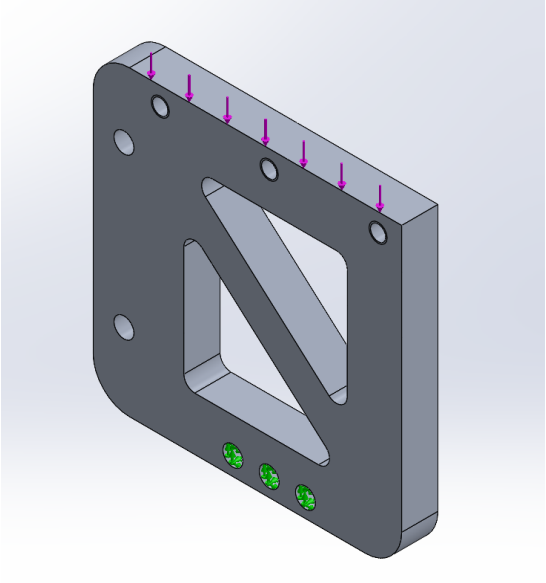


Figure 4.58: FEA configuration for ankle support part 2

The maximum stress experienced by the part was 1.142GPa, leading to a safety factor of 4.8. The stress results can be seen in Figure 4.59. The maximum deflection was .028mm and can be seen in Figure 4.60.

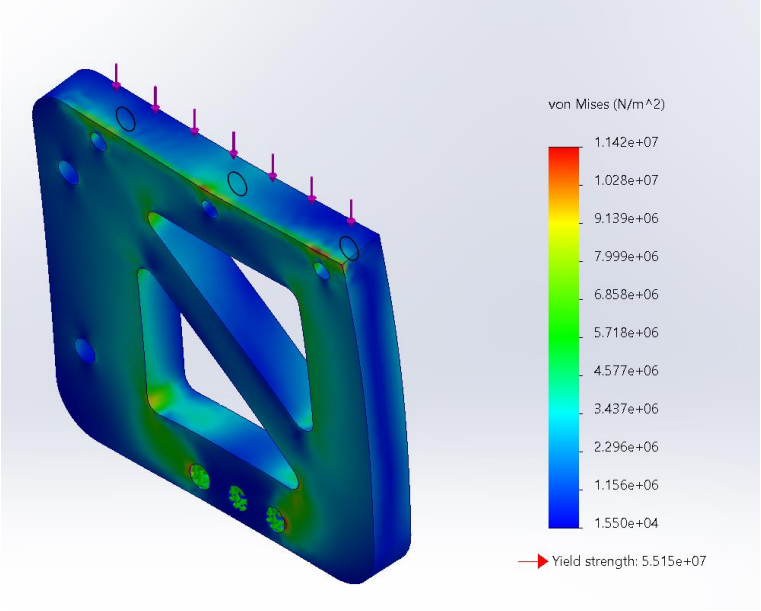


Figure 4.59: Stress results for ankle link part 2

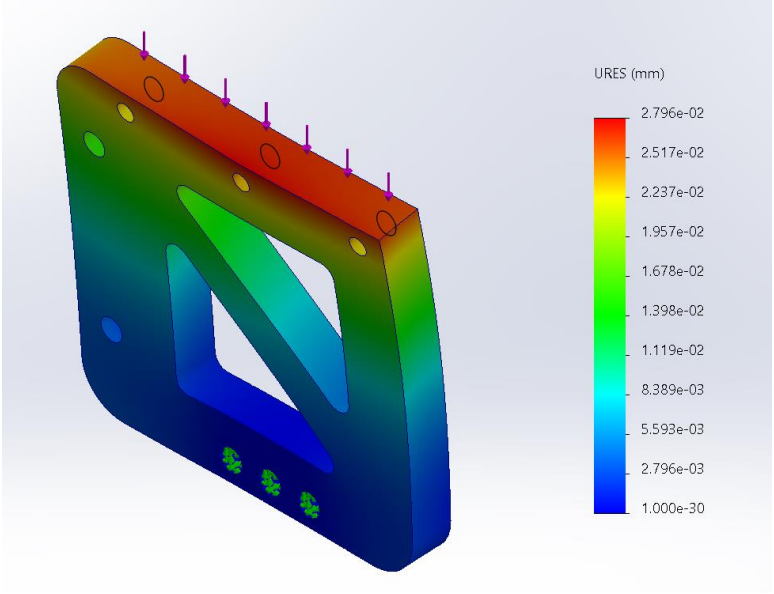


Figure 4.60: Deformation results for ankle link part 2

The third ankle support holds a bearing to connect to the foot. The hole for this bearing was fixed, and a total of 300N of bearing loads were applied to the screw holes above. The configuration is in Figure 4.61.

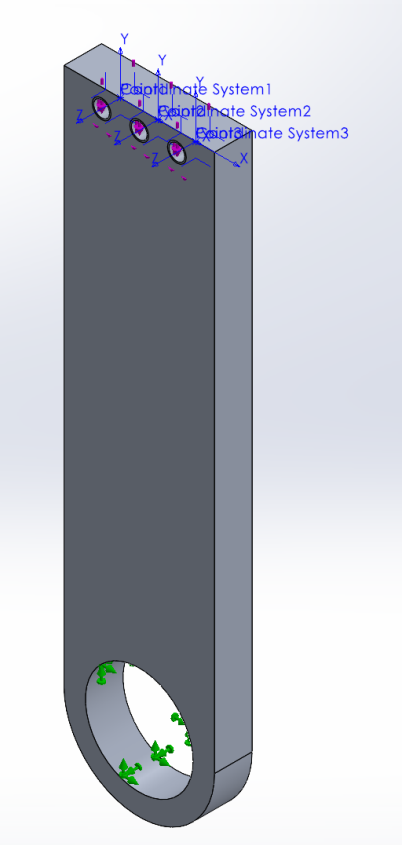


Figure 4.61: FEA configuration of the ankle support part 3

The maximum stress experienced by the part was 260MPa. This results in a factor of safety of 21.2. The stress results can be seen in Figure 4.62. The maximum deformation of the part was .0015mm, and the deformation scale of the study was 10854.9. the deformation plot can be seen in Figure 4.63.

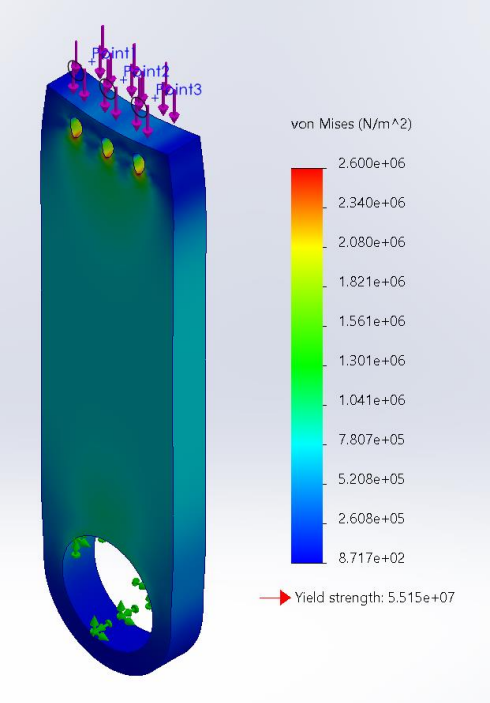


Figure 4.62: Stress results of the ankle support part 3

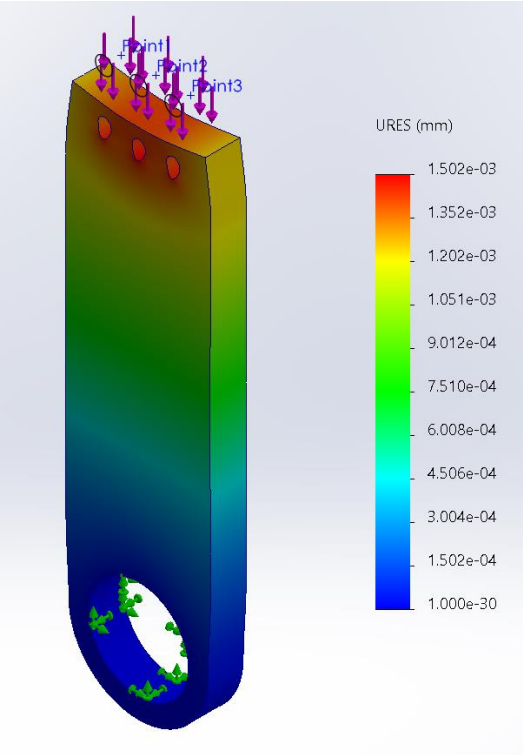


Figure 4.63: Deformation results of the ankle support part 3

4.8 Manufacturing Tools

When designing for manufacturing, we also need to design around what kind of machines we plan to use to fabricate each of the parts. Initially, we thought we were going to have to use the mill to fabricate every piece, which would require significant amounts of machining time, so we investigated alternative methods to fabricate our parts out of metal. We discovered that Washburn Shops also has a plasma cutting table, which we considered using. However, upon inspection of some parts made from the plasma cutting table, we found that the quality would likely not be up to our standards, so we started looking at alternatives. One alternative that was recommended to us was to find a third-party fabricator to cut our parts with a waterjet, however, when getting quotes for our parts, the pricing was significantly outside of our budget, so that idea was scrapped quickly. We then looked around to see if anyone at WPI had access to a waterjet, and there happens to be one at the 85 Prescott Street offices, but with a 1'x1' work bed. To manufacture our parts, we will use a combination of the waterjet cutter and milling machine to make our parts. The waterjet will cut out the general outline and holes in one face of the parts, then the milling machine will face the parts, then drill and tap the rest of the holes.

5.0 Robot Control Preliminary Design

5.1 Motors

Through our background research, we concluded that BLDC motors are optimal except for pricing. To stay within our initial budget, we identified BLDC motors that were not much greater than ~$120 each. Availability was also a major consideration as getting the motors in the labs and acquiring replacements quickly would greatly assist testing. After research, the motors we selected are Flipsky Electric Scooter Motor 6374 (*Amazon.Com: Flipsky Electric Scooter Motor 6374 Offroad Electric Longboard Skateboards Motor BLDC Brushless Outrunner Motor 190kv 3500w for Electric Skateboard,Longboard,Escooter,Ebike Esk8 DIY Kit Parts : Sports & Outdoors*, n.d.). Currently, the motors are available on Amazon with ~4 day shipping at $115 each. The motors are specified to have 8Nm max torque which is more than enough for our given requirements with a harmonic drive (1.5Nm max torque). In addition, with 190Kv and a max voltage of at least 40V, the max speed is at least 7600 RPM which also meets our given speed criteria (1000 max RPM). The design team also provided size constraints of 16cm length and 10cm diameter which this motor also complies with (10.8cm length and 6cm diameter).

5.2 Motor Controllers/External Encoders

After deciding that brushless DC motors were the route we were going to take, we needed to decide on a motor controller that would suit our needs. ESCs lack the positional control necessary for our project, so the ODrive is the controller we will be going with. While this is on the pricier side, $250 to control two motors, it has the features necessary for our robot to function. In order to stay within our budget, we are going with the ODrive 3.6, which is the cheapest controller in the ODrive lineup due to the fact that it is being phased out. Despite that, it has the features we need, we just need to acquire them before it goes out of stock.

Our encoder of choice to be paired with the ODrive is the AS5047P-TS_EK_AB magnetic encoder. This option is cheaper than an equivalent rotary encoder and affords us a couple other benefits. Since it is a magnetic encoder, it is not connected to the shaft and has a slightly larger tolerance than a rotary encoder. It will also make installation easier since it doesn't need to be fitted to the shaft of the motor.

5.3 Power Supply

Based on our research, we believe it is best to pursue either a 12s battery or two 6s batteries run in series. Our thinking is that, based on our power and current requirements and that of similar projects, a 12s battery would surely provide enough voltage and current to be distributed throughout our system. Running two 6s batteries in series would provide the same voltage, however, they may be able to provide a higher discharge current, based on our research. The primary 12s battery that we believe is best for our cost restraints is the MOLICEL-P42A-12S4P-16.8AH, which is an Eskate battery that ranges in price from $349.00 to $699.00. Our 6s battery of choice is the MBoards 6s1p 40T Complete Battery Solution, which is also a 6s Eskate battery that is roughly $200. Both of these options meet our power requirements, however, more testing must be done before we decide if the 12s or two 6s in series is optimal.

5.4 Computing

We have selected the Raspberry Pi 3 for our computer for this project due to the vast documentation, its use in similar projects, and its computing capabilities. With this computer, along with our knowledge of C/C++ and python accompanied by ROS, we are confident that we can meet our requirements for this project. This selection is also consistent with several other projects of similar scale that we came across in our research.

5.5 Budget

Table 5.1: Controls Budget Table A Term

Item	Cost	Quantity	Total
Flipsky 6374 190kv 3500W BLDC Motor	$115.00	12	$1,380.00
ODrive 3.6 BLDC Motor Controller	$250.00	6	$1,500.00
AS5047P-TS_EK_AB Magnetic Encoder	$20.00	12	$240.00
2-Channel CAN Bus Raspberry Pi HAT	$23.00	1	$23.00
Battery / BMS	$500.00	1	$500
Raspberry Pi 3	$35.00	1	$35.00
Cabling	$250.00	1	$250
Controls Total cost	Goal: <= $4000		**$3,928**

6.0 Robot Sensing Design and Testing

6.1 Location of FSRs and Design of Foot

From our research, decision matrix, and budget we concluded that using four FSRs with two sensors in the back and two in the front (creating a rectangular configuration) was our best option for our force sensor design. Four sensors in this alignment are a commonly used design for humanoid robots, thus validating our decision. Additionally, when looking at where the weight is distributed on human feet it is mostly in the front and back of the foot. Since we plan to have a human-like gait, having sensors in the front and the back of the feet will help us determine which way the robot is leaning to keep it balanced when walking, standing, or pushed by external forces. The figure below shows the placement of the FSRs.

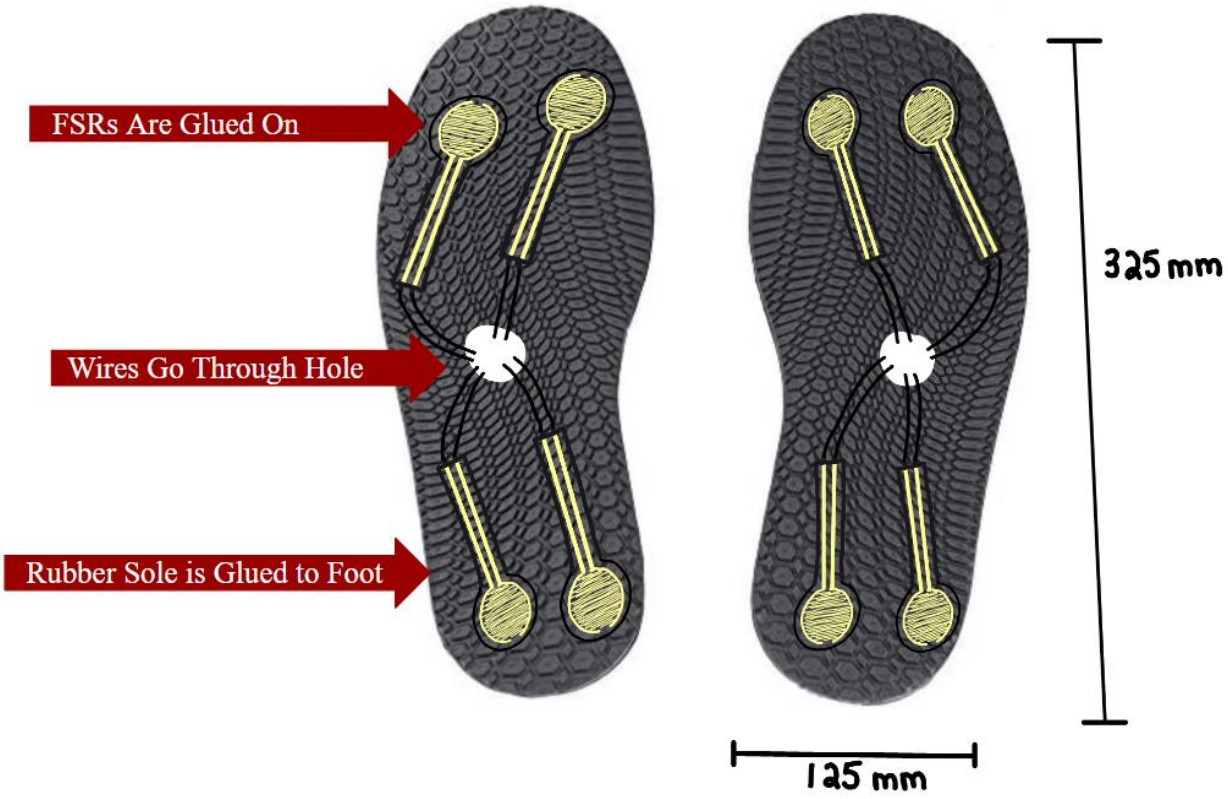


Figure 6.1: Initial Foot Design

The following design was chosen for HURON due to design and machining constraints.

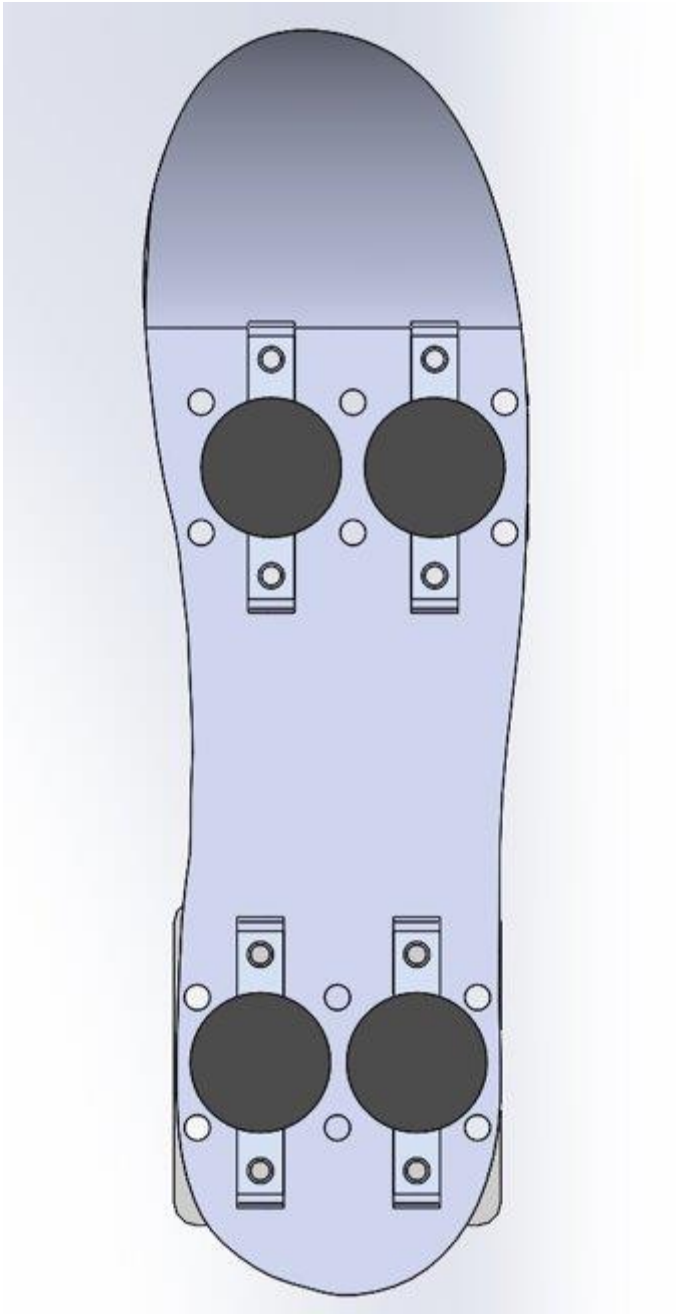


Figure 6.2: Foot Design (made by Design team)

From the initial design, the same locations of the FSRs are kept. However, the design was adjusted because the hole in the foot was not feasible. Instead, large rubber pads are the only point of contact with the ground. The figure below shows how this rubber is then connected to the FSR.

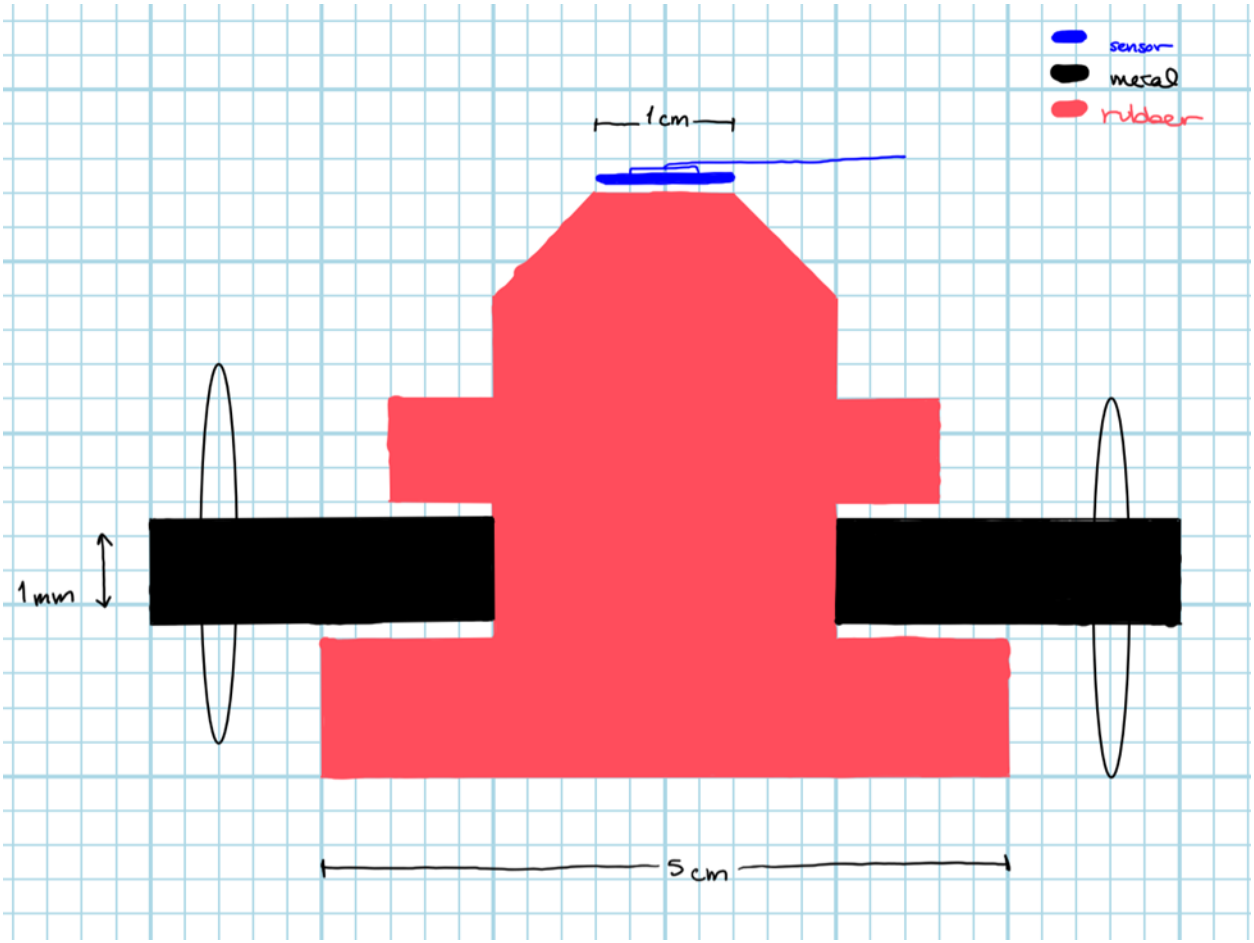


Figure 6.3: Side Inside View of Foot (Made by Design Team)

In Figure 6.3, black represents the metal comprising the foot and the red represents the rubber. The blue on top represents the FSR which is 0.375 inches or 0.9525 cm in diameter. All the weight from the floor will be transferred up the rubber to the top and the FSR will push up to a plate on top getting a force reading. There is then a slot on this plate where the rest of the FSR will come out of and the wiring and microcontroller will be clipped onto the leg of the robot.

6.2 Materials

The weight of the robot will be distributed between the four FSRs on each foot. However, depending on the robot's dynamic position, more weight may be on specific sensors. At the design phase, it was estimated that HURON will weigh 40 kg, which only includes the lower half of the humanoid. Since HURON is planned to be built on for the next several years, it will gain weight each year. Therefore, the highest weight rated FSRs of 100 lbs were chosen. If all the weight is evenly distributed throughout all eight sensors (four on each foot), there would be about 11 lbs or 5 kgs on each sensor (with the total weight of the robot being 40 kgs). However, the weight will

not be evenly distributed because that is not the case for humans, and the robot was designed to be as human-like as possible. Additionally, with dynamic motion more force will be exerted on some sensors while simultaneously exerting no force on other sensors. The commonly sold FSRs come in 1 lb, 5 lbs, 25 lbs, and 100 lbs. These weights are the limit of the FSRs and FSRs do not perform well close to their limits, so 100 lbs FSRs were the best option to avoid getting close to the limit. FlexiForce piezoresistive force sensor (FSR) from Tekscan were decided upon. Theoretically, the resistance changes from infinite to ~300kOhms. However, the resistance of the FSR at full load was measured to be about 35MOhms. These sensors are thin and flexible, but their resistance does not change while being flexed. Their resistance only changes when pressure is applied to the sensing pad. The length of the FSR is 8.5 inches, 0.1 inches wide, and the sensing pad is 0.375 inches in diameter (*FlexiForce Pressure Sensor - 100lbs. - SEN-08685 - SparkFun Electronics*, n.d.). The price of these FSRs is $22 and eight are required for measuring the force distribution, totaling the cost of FSRs to $176.

To get force readings and communicate with the rest of the robot a microcontroller was integrated into our system. A high sampling frequency of at least 1kHz, low cost, and ease of integration into the rest of the humanoid systems were important for microprocessor selection. Additionally, the microcontroller was required to have eight analog inputs to read the FSR signals. An Arduino Mega was chosen the model fits all the microcontroller requirements.

The remaining circuit materials consist of op-amps, resistors, and wires. Op-amps will keep the FSR output consistent and buffer the signal for smaller driving loads. The recommended op-amsp for Flexiforce FSRs are LM324, which are quad, rail-to-rail op-amps. Since the Arduino Mega has a 0V to 5V source voltage, this op-amp optimizes our signal allowing us to read from 0V to 5V. This op-amp is also priced very low at $1.50 and only two are required (1 per 4 FSRs).

6.3 Circuit Design

The two circuits were tested: a two-stage current to voltage circuit to create more linearity and a voltage divider into an op-amp. The FSR Voltage Divider is shown in the figure below.

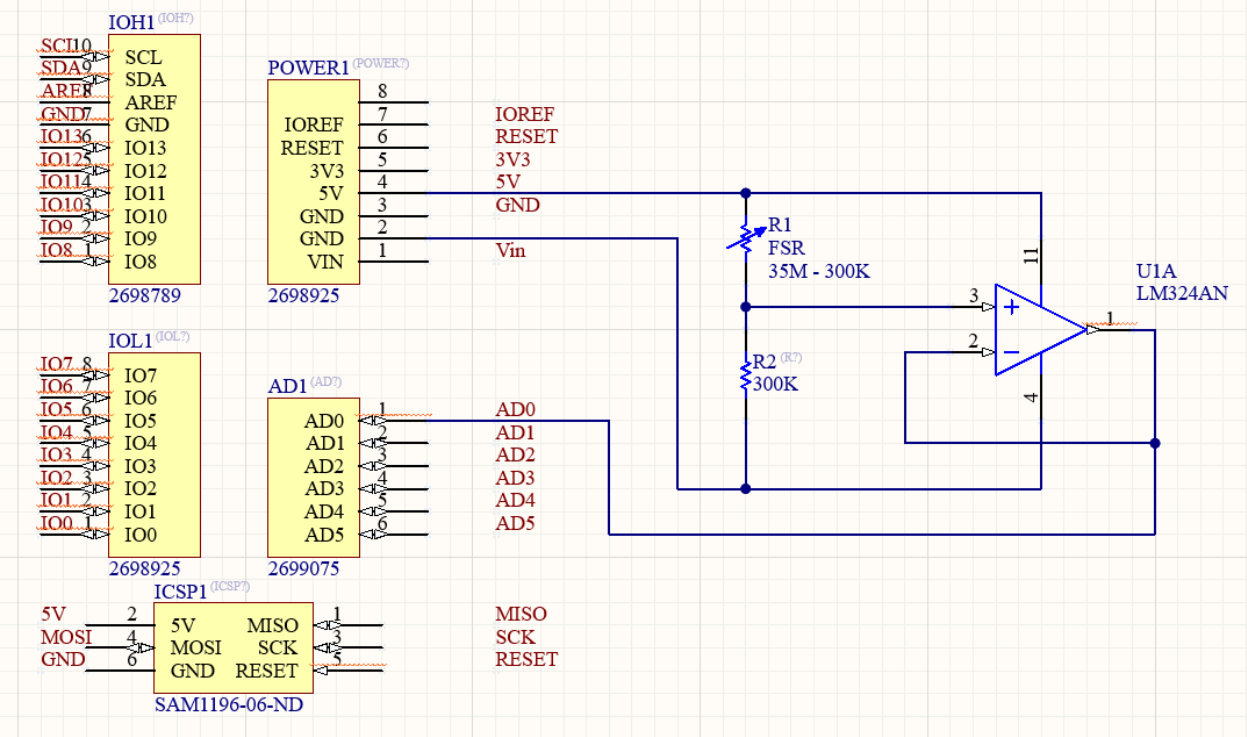


Figure 6.4: FSR Voltage Divider Schematic

This provides a simple force-to-voltage conversion where the output voltage increases with increasing force. The resistance of the FSR ranges from 300k Ohms (full load) to 35M Ohms or practically infinite resistance (no load). The measuring resistor, R2, was chosen to maximize the desired force sensitivity range and to limit current. The current through the FSR should be limited to less than 1mA/square cm of applied force. The output of the circuit can be represented by the equation below.

$$V_{OUT} = \frac{V^+}{1+\frac{R_{FSR}}{R_2}} \tag{6.1}$$

At full load, where the resistance of the FSR is approximately 300k Ohms, we wanted Vout to reach 5V. However, at no load, where the resistance of the FSR approaches 35M Ohms, we wanted Vout to reach 0V. In order to compensate for both the full load and no-load scenarios, a resistance of 300k Ohms was chosen for R2, which meant our output voltages would be slightly lower than 5V at maximum load and slightly higher than 0V at no load.

The current to voltage converter we designed is shown below. Since force and resistance have an inverse relationship when dealing with FSRs, the relationship between force and

conductance with an FSR is linear. Therefore, we created a current to voltage circuit to get a more linear relationship.

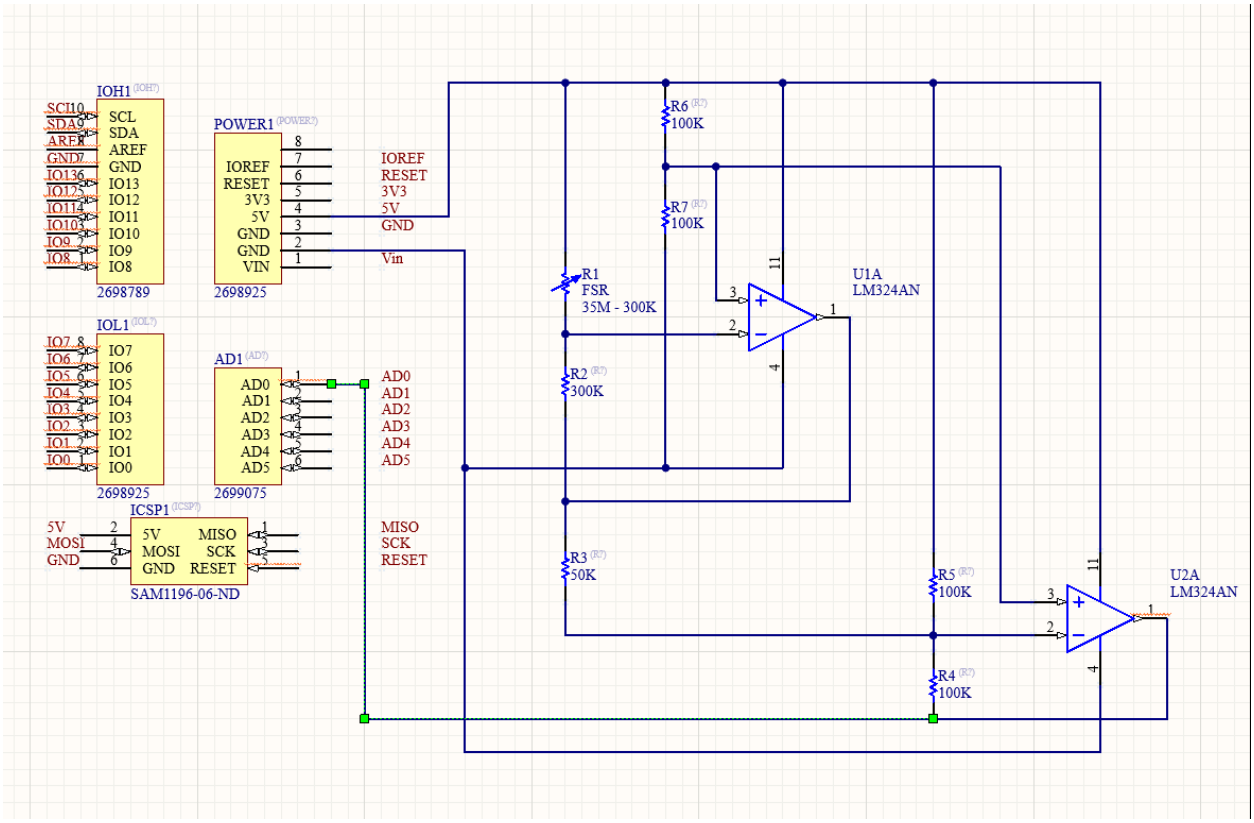


Figure 6.5: Current-to-Voltage Circuit Schematic

This circuit consists of two FSR current-to-voltage converters. The first stage can be shown with the circuit below.

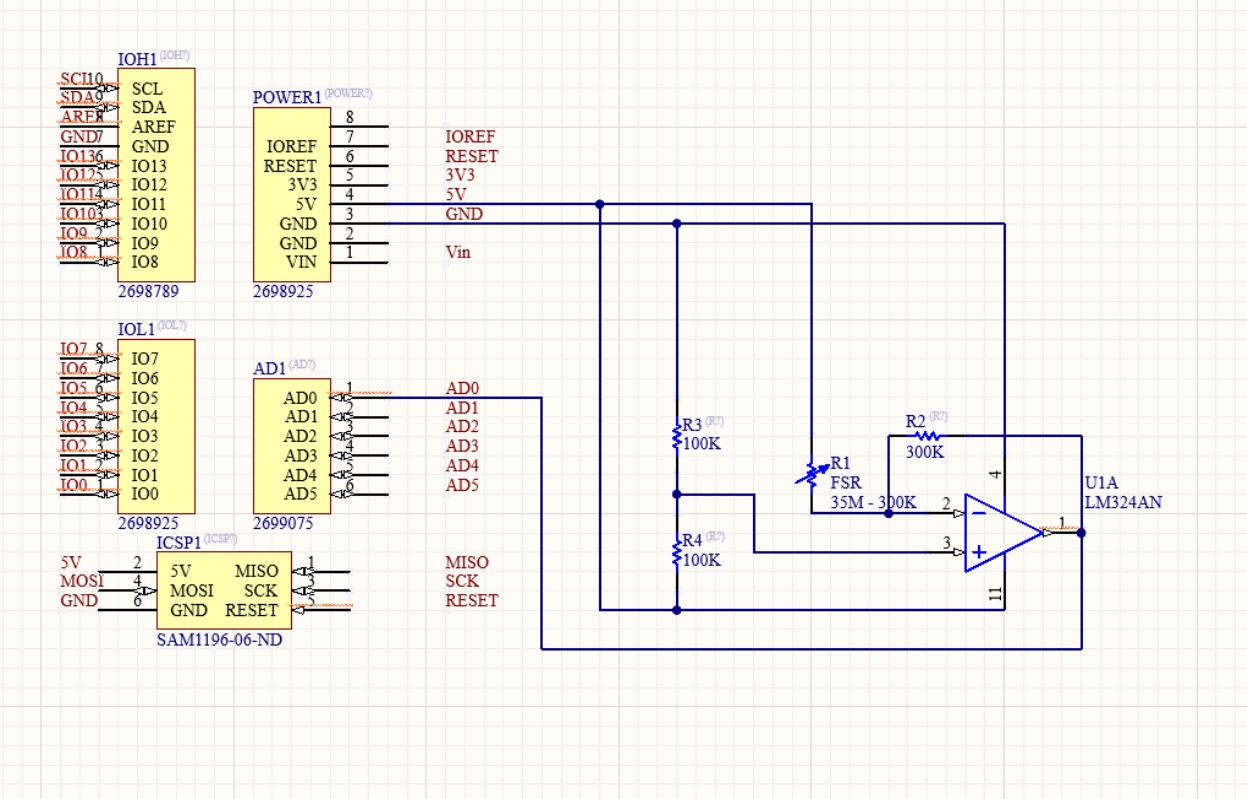


Figure 6.6: Circuit Schematic for Stage 1 of Current-to-Voltage Converter

Since we are using a single voltage input of 5V, we used a voltage divider with two matching resistors to input 2.5V into the positive input of the op-amp. We then matched the resistance of RG to 300k Ohm to increase sensitivity but avoid negative saturation. The output of this circuit is from 2.5V at no load to 0V at full load and can be represented by the equation below.

$$V_{OUT} = \frac{V_{ref}}{2} * \frac{1 - R_2}{R_{FSR}} \tag{6.2}$$

However, since we want Vout to be at 5V at full load and 0V at no load we needed a second stage. The second stage is represented by the circuit below.

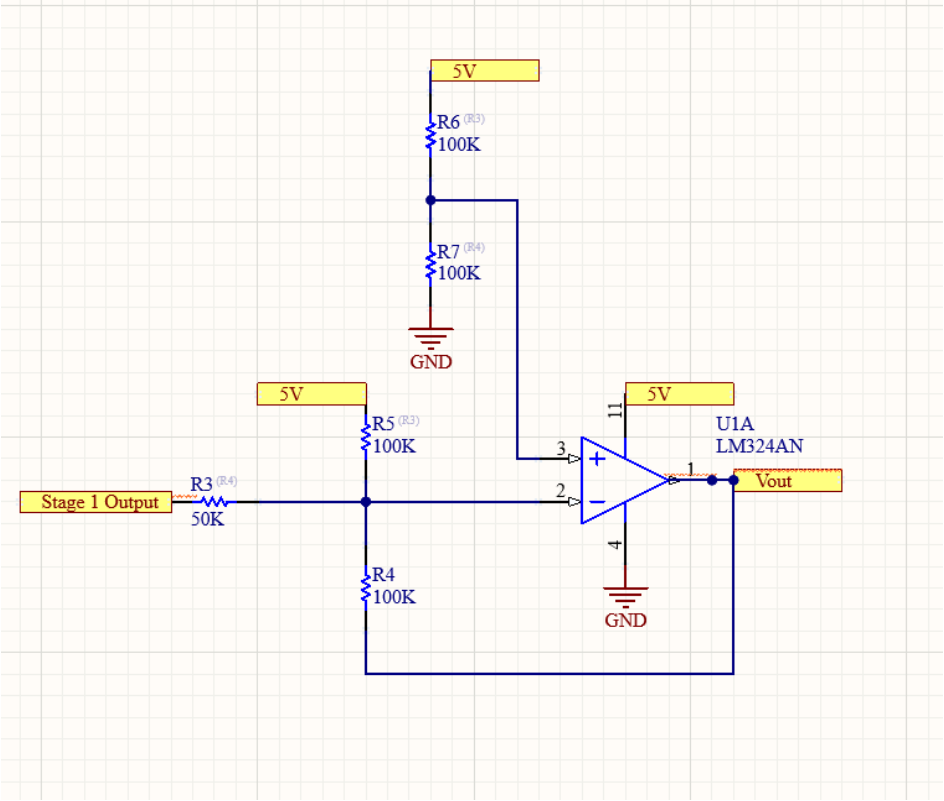


Figure 6.7: Circuit Schematic for Stage 2 of Current-toVoltage Converter

When full load is applied to the FSR, the output of the first stage is 0V, thus the input into the negative input of the second stage is 0V. This situation can be presented with the following equation.

$$V_{OUT} = \frac{V_{ref}}{2} * \frac{1+ R_2}{R_{FSR}} \tag{6.3}$$

Since R2 is equal to 300k Ohms and at full load RFSR is equal to 300k Ohms the output of the second stage is 5V. When no load is applied to the FSR, the output of the first stage is 2.5V and the positive input of the second stage is always 2.5V. Since both inputs of the op-amp are always equal, no current flows through the 50k resistor. Therefore, the following equation can be used to determine Vout.

$$V_{OUT} = \frac{V_{ref}}{2} * \frac{1- R_4}{R_5} \tag{6.4}$$

In this case, R4 and R5 are both 100k Ohms, so Vout goes to 0V at no load. All of the equations listed above were found through Interlink Electronics FSR guide (Solis, n.d.).

6.4 Testing

After designing two circuits, we tested them to determine the output voltage each circuit gave at incremental weights from 0 lbs to 100 lbs. This helped us determine which circuit performed the best. For our testing, we first created a testing mechanism to hold the weights, ensuring that all the weight was evenly being applied to the sensing pad on the FSR. The testing mechanism is shown in Figure 6.8, Figure 6.9 shows the top part of the testing mechanism up-side down, and Figure 6.10 shows a top view of the bottom part of the testing mechanism.

On top, the testing mechanism has a platform to hold the weights. A wooden dowel is then drilled into the platform. At the end of the dowel, there is a vinyl pad the same size as the sensing area on the FSR. When testing we placed the weights on the platform. The weight on the platform is then transferred to the dowel and on to the absorbing impact material. We found a vinyl material the same size as the sensing pad with a low elongation and low compression set, so we used this for testing. The dowel goes through a hole in a block which acts as the main guide for the dowel. The three blocks screwed onto the platform also act as additional guides. Since the dowel is longer than the additional guides, none of the weight is transferred to the guides.

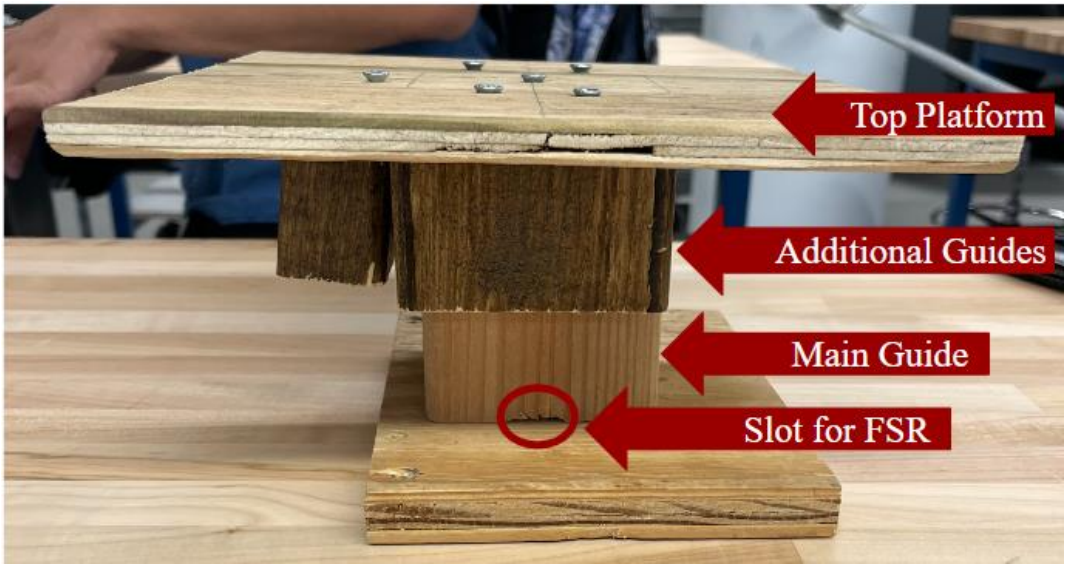


Figure 6.8: Testing Mechanism Labeled (Front View)

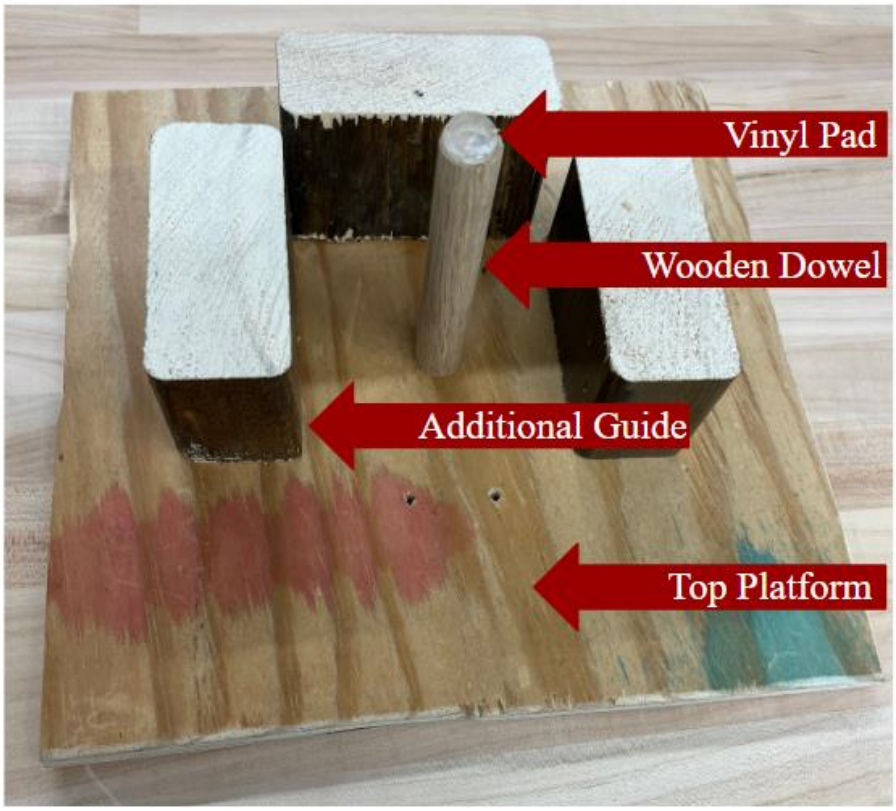


Figure 6.9: Top Part of Weighing Mechanism (Up-side Down)

Figure 6.10: Bottom Part of Weighing Mechanism (Top-View)

To perform our testing, we configured our circuit on a breadboard and slipped the FSR through the slot in the main guide (this slot is shown in Figure 6.8). Our entire testing setup is shown in Figure 6.11 below.

Figure 6.11: Testing Set Up

After configuring our set up, we placed weights on top of the weighing mechanism in increments of 5 lbs, starting from 0 lbs all the way up to 100lbs. While doing this we recorded the weight applied and voltage read. Unfortunately, our two-stage circuit design railed to 5V immediately at 5lbs, so our best performing circuit was the FSR voltage divider. The results for our voltage divider circuit are shown in the table and figure below. Also, please note that the reason for there being a reading with no weight added is because it is reading the weight of the weighing mechanism.

Table 6.1: Weight vs. Voltage Relationship of Force Sensing Resistor

Weight (lbs)	Voltage (mV)
0	200
5	750
10	1250
15	1570
20	1730
25	2150
30	2330
35	2500
40	2570
45	2650
50	2740
55	2820
60	2880
65	2920
70	2940
75	3000
80	3040
85	3060
90	3120
95	3140
100	3170

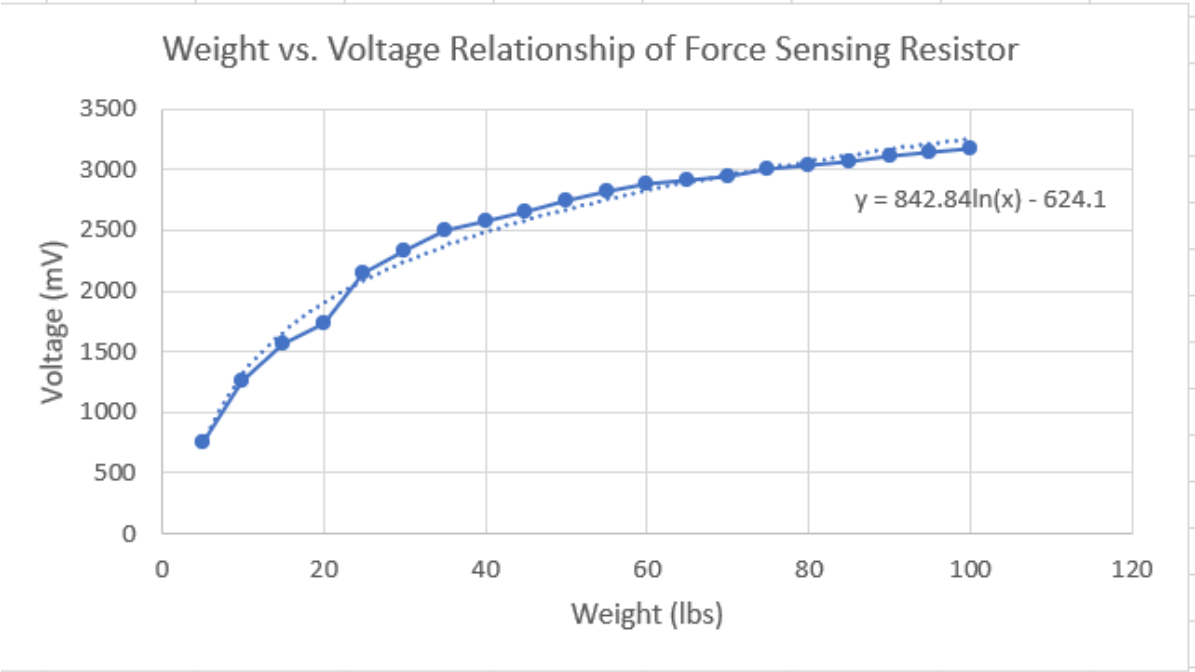


Figure 6.12: Weight vs. Voltage Relationship of Force Sensing Resistor

Although it is a logarithmic relationship, the performance of the FSR starts to become poor around 60 lbs and with the current weight of the robot the maximum weight on one FSR will most likely be around 40 lbs. 40 lbs could be the case if the robot is on one foot and only on the front or back part of the foot is touching the ground.

In the humanoid robot, we will get a voltage reading from the FSR and use this to determine how much force is on each FSR. Therefore, we used MATLAB to find the predicted weight from the voltage by finding the inverse of our best fit curve. The results from this are shown in the figure below.

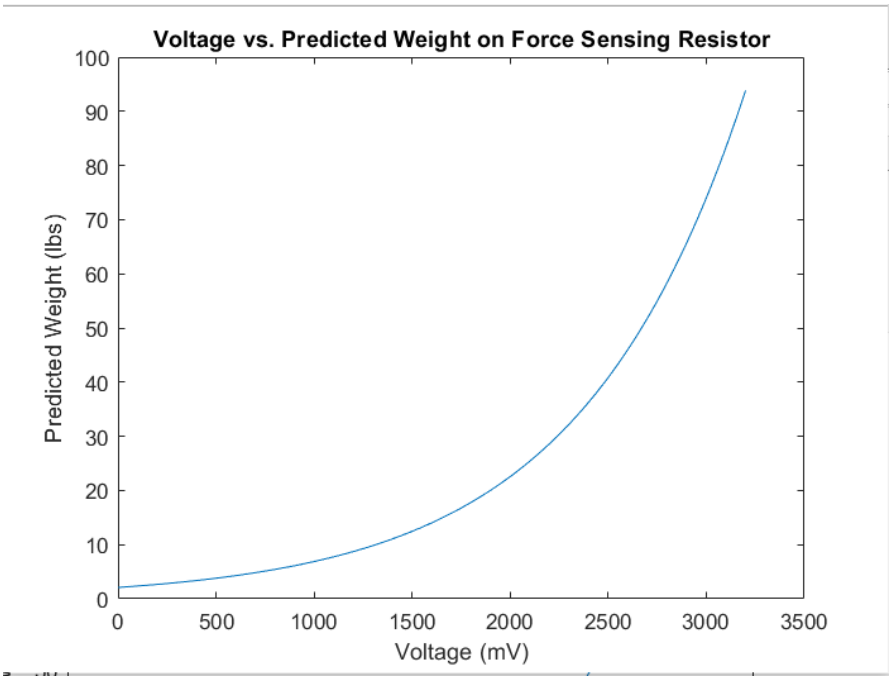


Figure 6.13: Voltage vs. Predicted Weight on Force Sensing Resistor

After finalizing what type of circuit would help utilize the FSR to meet our needs, the sensor team's next task was to determine the optimal resistor value for representing the voltage to weight relationship. For this testing, the sensor team used the same testing device discussed in figures 6.8 through 6.11. The team tested 6 different resistors for the voltage follower. This includes 3kΩ, 10kΩ, 30kΩ, 51kΩ, 75kΩ, and 300kΩ resistors. The procedure for testing these resistors is the same as the previous testing but was repeated from start to finish for each resistor.

The results from these tests are shown below in Figures 6.14 to 6.19. After plotting our findings, we found a regression that best matched our real data. This resulted in a linear regression for the circuits with a 3kΩ and 10kΩ resistor and a logarithmic regression for the circuits with a 30kΩ, 51kΩ, 75kΩ, and 300kΩ resistor. The figures below show the real data in a darker blue and the regression in a lighter blue.

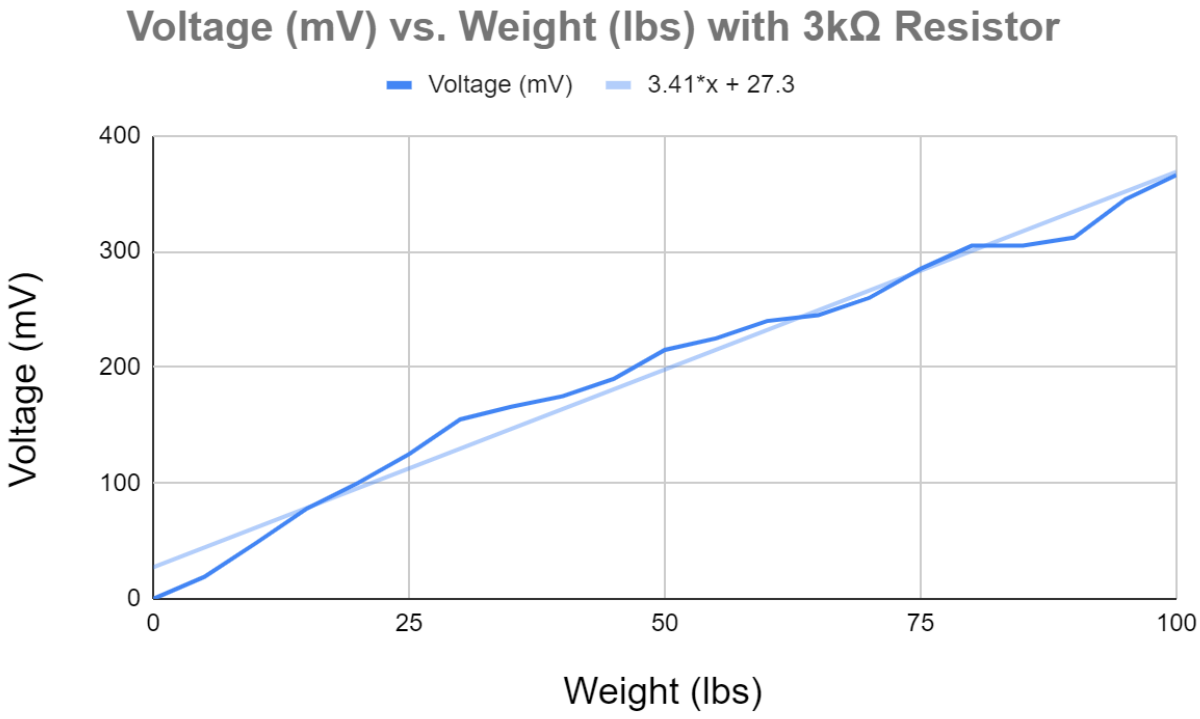


Figure 6.14: Voltage vs. Weight Relationship with a 3kΩ Resistor

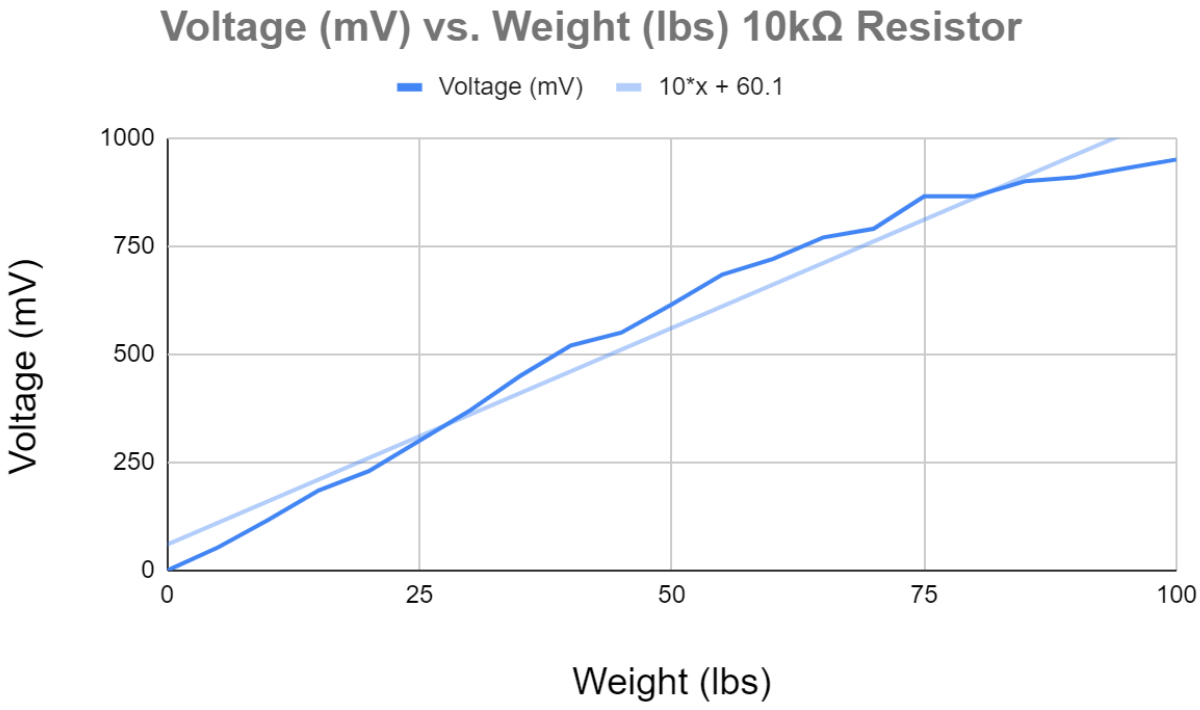


Figure 6.15: Voltage vs Weight Relationship with a 10kΩ resistor

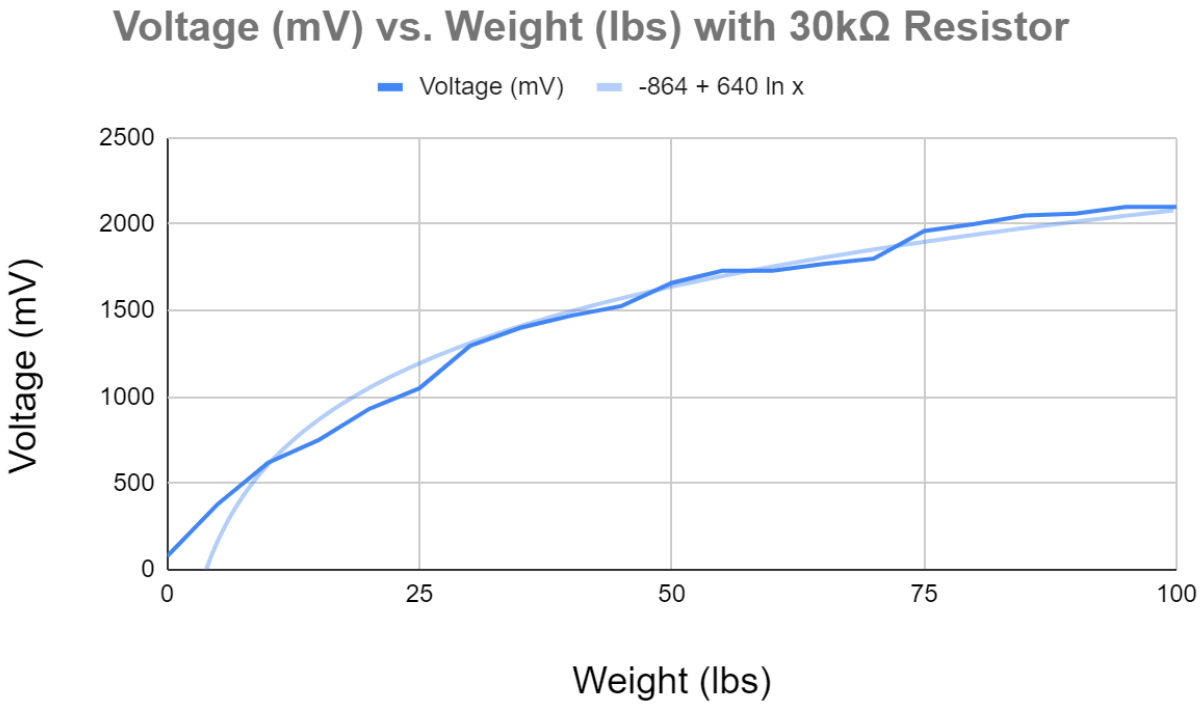


Figure 6.16: Voltage vs Weight Relationship with 30kΩ Resistor

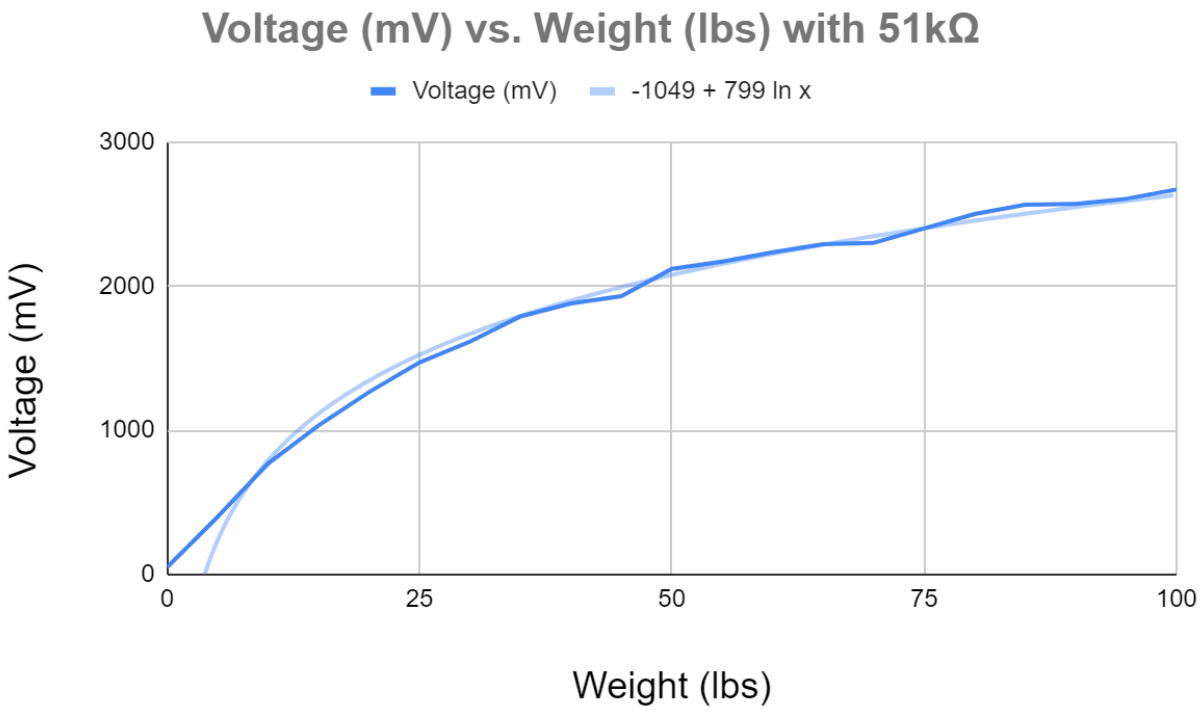


Figure 6.17: Voltage vs Weight Relationship with 51kΩ Resistor

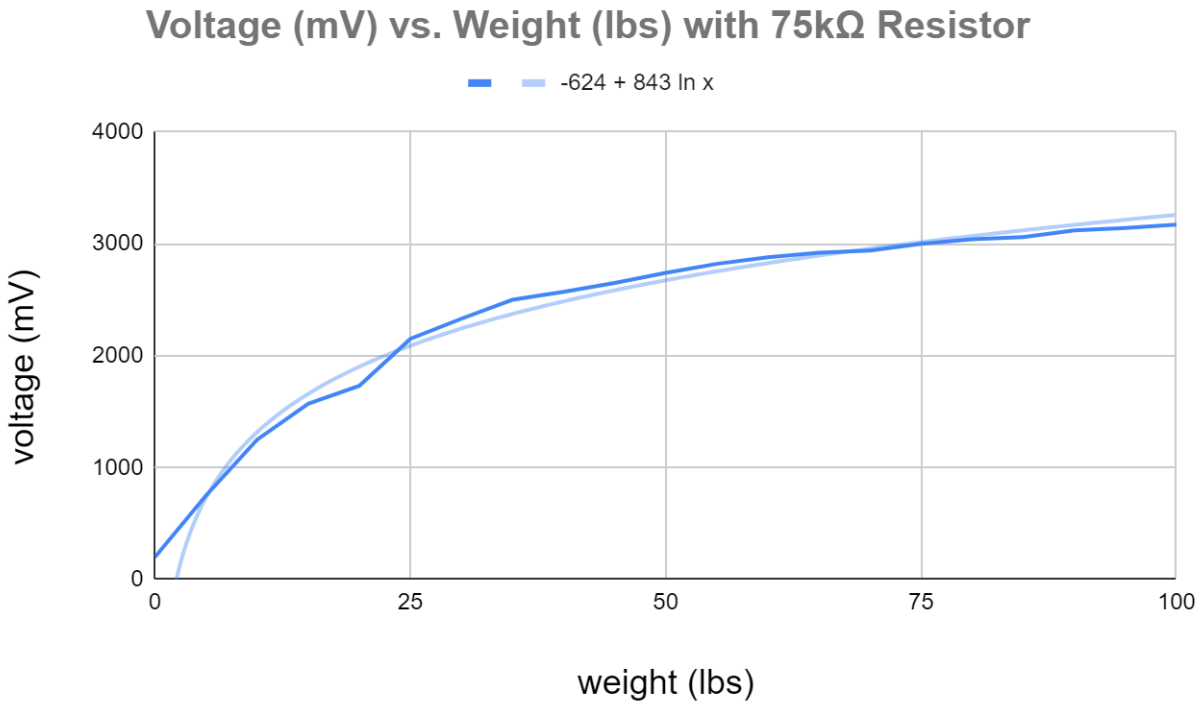


Figure 6.18: Voltage vs Weight with 75kΩ Resistor

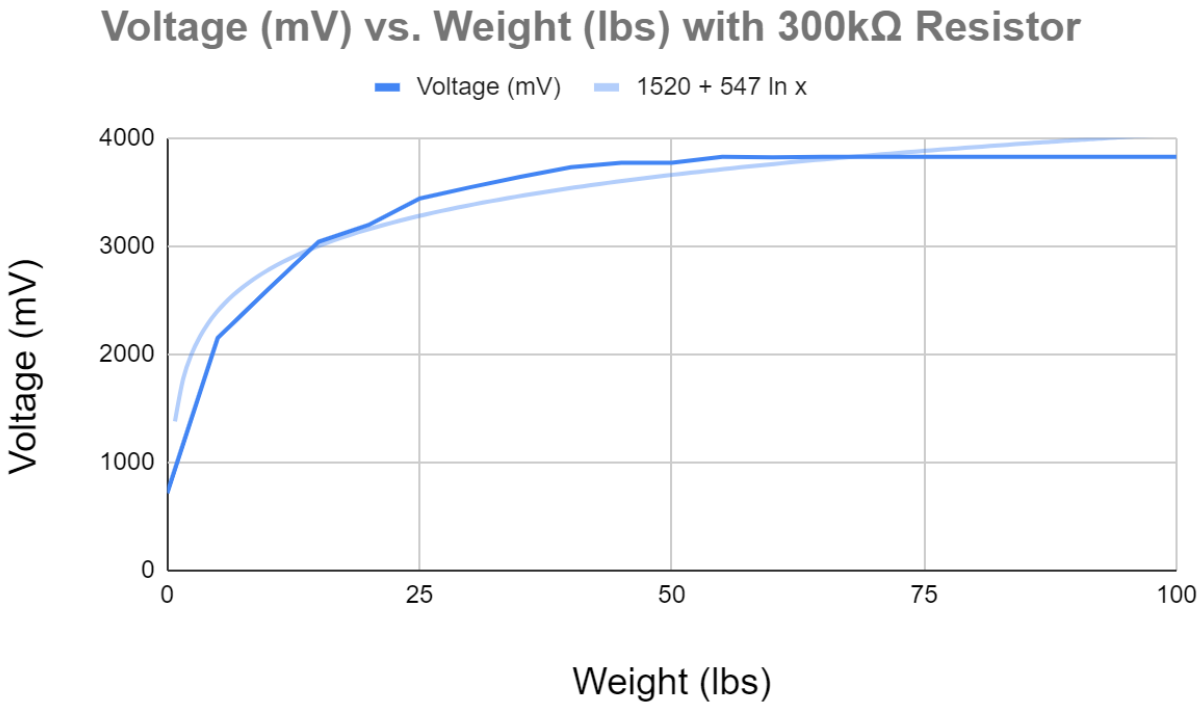


Figure 6.19: Voltage vs Weight with 300kΩ Resistor

With these results, we wanted to find the resistor value that obtained a regression that differed the least from the real data and gave a large range of voltage. Having a large range of voltage would make it easier to detect changes once implemented into our robot and a close regression would help us to accurately determine how much force the robot is experiencing and where. This led us to the 51kΩ resistor as it has a voltage range of about 2.7V and its logarithmic regression was on top of the data for most of the experimentation.

After taking a closer look at the results from the 51k resistor, we determined that we could more accurately represent the voltage to weight relationship through splitting our data into two sections and creating a linear regression from 0-10 lbs and a logarithmic regression from 10-100 lbs. This is shown in figures 6.20 and 6.21.

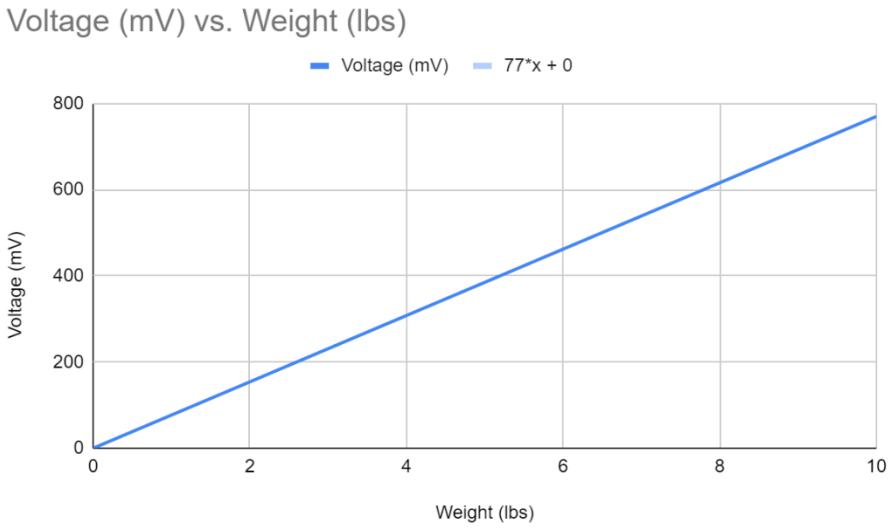


Figure 6.20: Weight vs voltage relationship with a 51kΩ resistor from 0-10lbs

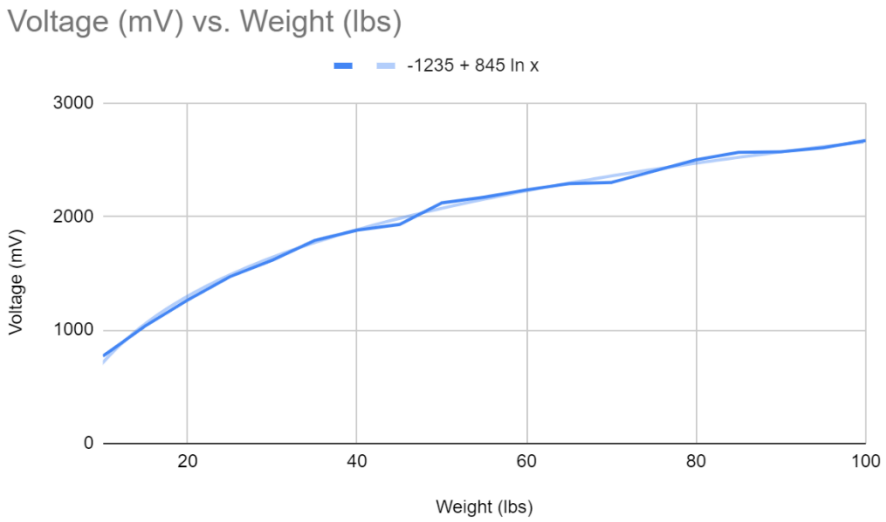


Figure 6.21: Weight vs Voltage Relationship with a 51k resistor from 10-100lbs

As shown in these figures it creates even less error as there are only about 3 spots where the regression differs from the real data at around 50 lbs and 70lbs. However, these errors could have also been due to human error during testing. Overall, with a 51k resistor we can accurately create two equations to represent the weight to voltage relationship with a wide voltage range. After arriving at our two equations, we then needed to find the inverse of to develop a relationship for finding the amount of weight on each resistor based on the voltage reading. The voltage reading at 10lbs was 770mV, so the inequalities were created based on this.

$$Force = \frac{voltage}{77},\ if\ voltage < 770 \tag{6.2}$$

$$Force = e^{\frac{voltage + 1235}{845}}, if\ voltage \geq 770 \tag{6.3}$$

6.5 Finalizing Circuit

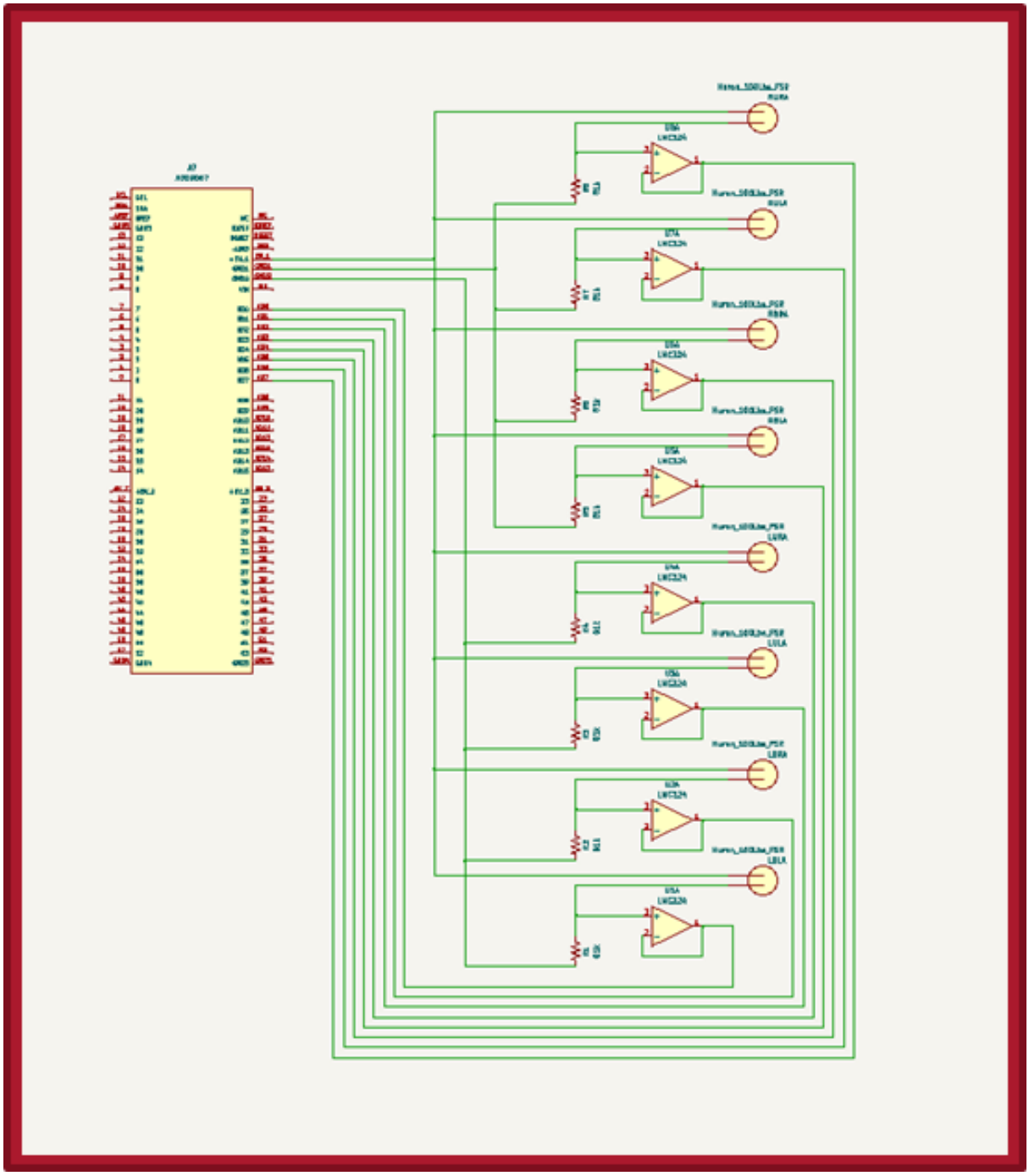


Figure 6.22: Feet Sensor Circuit Schematic

After finalizing the circuit along with the code, we needed to integrate the components more securely and permanently to place onto the robot. The final circuit (figure 6.22) consists of an Arduino Mega. One sensor corner uses a voltage divider followed by an op-amp voltage follower. The voltage divider consists of a 51K Ohm resistor along with the FSR (50M - 300K Ohms) and the op-amp is in a LMC324. In total there are 8 of these circuits that output to an analog input in the Mega (A0 for left foot back left corner, A1 for left foot back right corner, A2 for left foot upper left corner, A3 for left foot upper right corner, A4 for right foot back left corner, A5 for right foot back right corner, A6 for right foot upper left corner, and A7 for right foot upper right corner).

To permanently attach the circuit, the quickest and cheapest solution was to solder the components onto a perfboard. The team was able to order perfboards from amazon and cut them into (2 x 1.2 in) pieces.

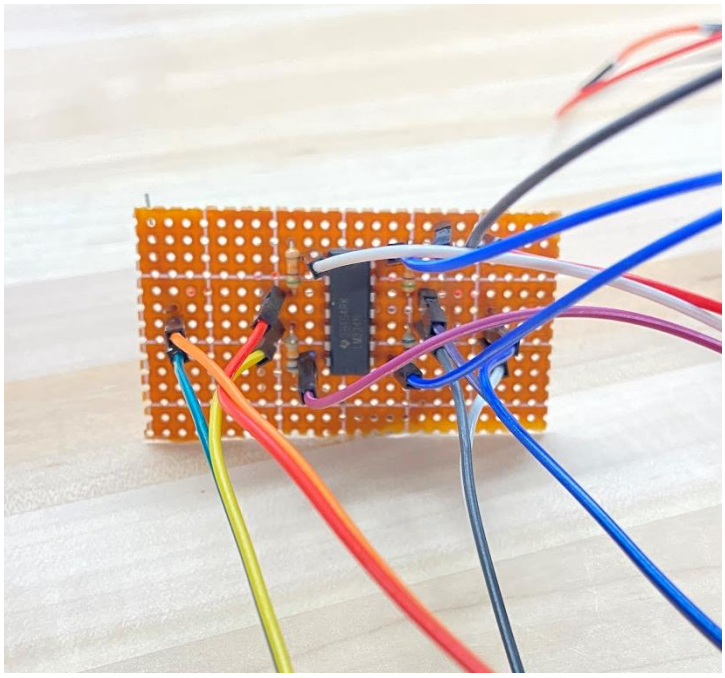


Figure 6.23: Front View Perfboard

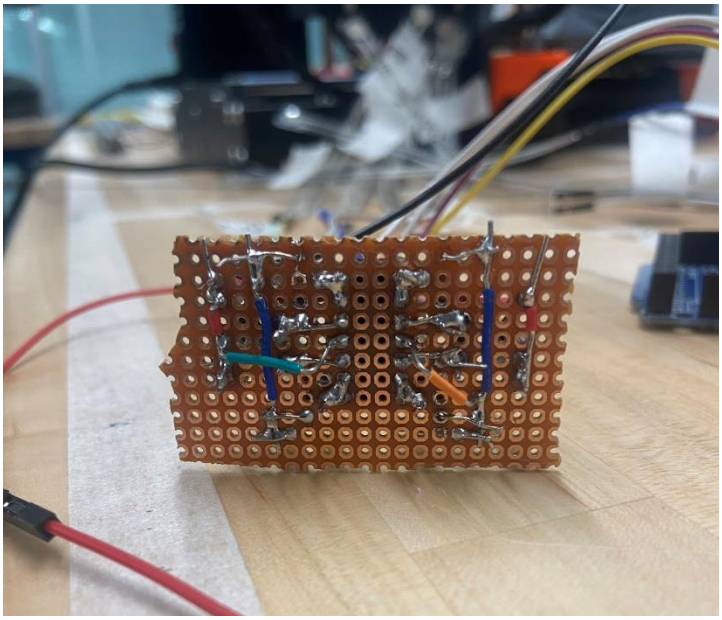


Figure 6.24: Back View Perfboard

As seen in figures 6.23 and 6.24, there are 2 perfboards, 1 for each foot. Each board has an LMC324 soldered in the centers followed by the voltage divider consisting of a 51K Ohm resistor and a 50M to 300K Ohm FSR. To note, the FSRs are not soldered directly to the circuit in case any fixes or changes in the future need to be made, instead they are attached using male to female wires which are soldered in. There are also extra wires which are for attaching to the Arduino Mega analog input ports, 5V power port, and the ground ports. After soldering the components, the circuits were checked to make sure the correct connections were made and if any changes in values were seen.

6.6 Verification of Stability Measurement and Angle of Lean

After theorizing how to quantitatively measure the stability of the robot and determine the angle it is leaning, the team verified if the calculations were accurate. The team first programmed an Arduino to calculate the stability and angle of lean. Then connected the Arduino to the two circuits containing all eight FSRs. All eight FSRs were attached to a plank of wood in the same formation it would be on the feet of the robot, (2) 2x2 matrix. After creating the setup, one member of the team stood on the planks of wood and leaned in different directions and stood in different stable and unstable ways. Another person on the team watched the stability and angle change while the other member altered their stance on the wood. This set up is shown in the figure below.

Figure 6.25: FFSM data collection. FSRs were attached to the bottom of two pieces of wood, representing the left and right feet.

The data from this verification test was saved and visualized in the graphs below. The top graph shows the stability from 0 to 1 and the bottom graph shows the angle of lean from 0° to 360°.

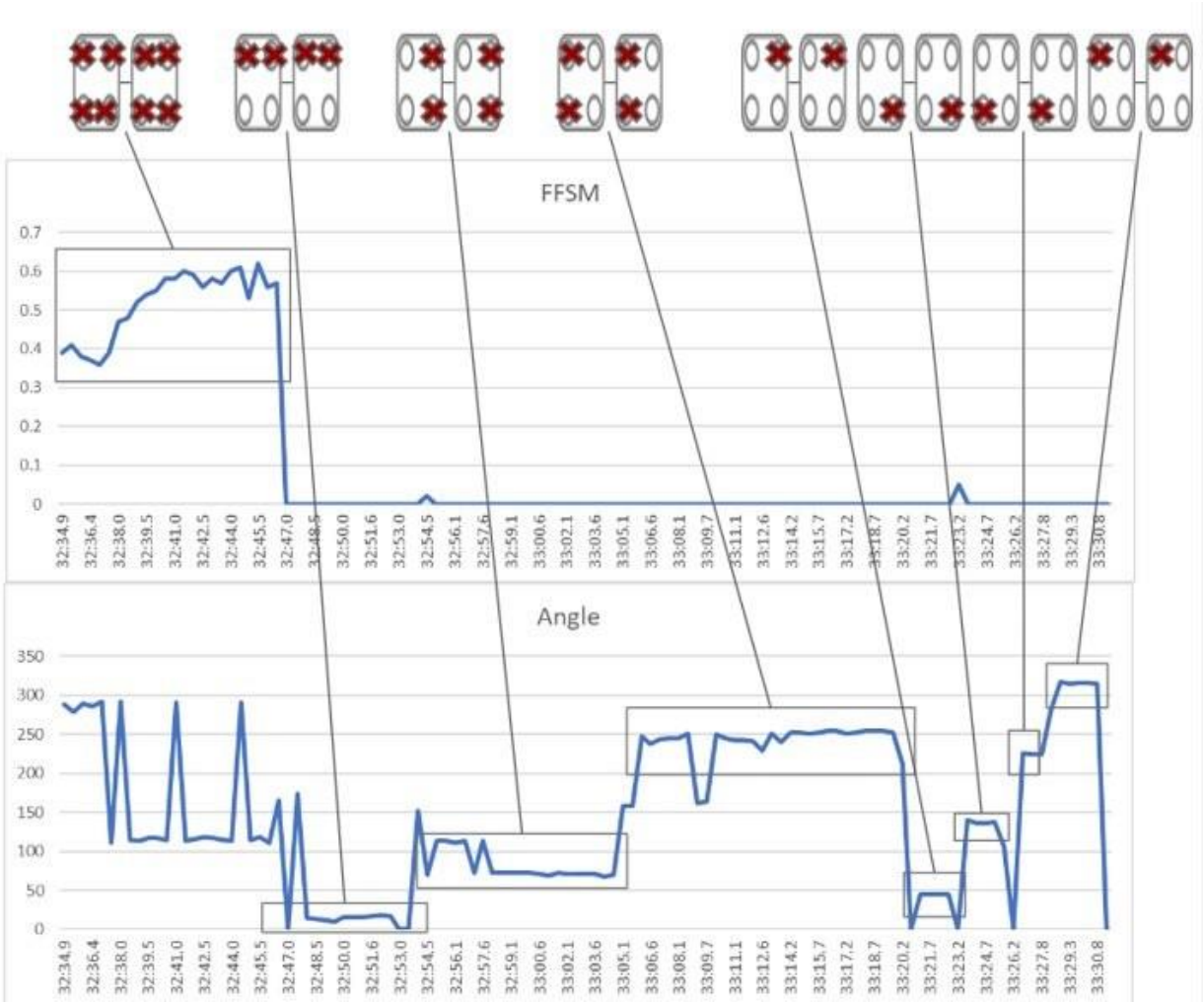


Figure 6.26: FFSM testing. Top-most diagrams represent the two-foot model, where red X's mark applied pressure. The top graph shows the calculated FFSM value, while the bottom graph shows the Angle value.

The figure above demonstrates the accuracy of the FFSM system. This test started by exerting an approximately equal force on all FSRs, and a high FFSM of around 0.6 was recorded meaning the system was stable. Then the testing moved onto ensuring the logic was correct for each scenario of the angle. The results demonstrated that the angle increased as the team selectively applied pressure in the clockwise direction on each foot. Sudden spikes in the data can be attributed to the distribution of pressure throughout our plank of wood, which itself may have been slightly bent. These issues were mitigated using custom force stoppers.

7.0 Manufacturing Process

At the beginning of B term, the design team began the manufacturing process. Since we still needed to finalize the ankle design from the previous term, we split into two “sub groups” to work on the ankle, and to also start learning CAM, as none of us have had real CAM experience before.

7.1 Finalizing Ankle Design

As mentioned previously in chapter 4.6 (weight and joint torques), the ankle needed a redesign based on the force issues. There were a couple options that were discussed amongst the team. One option was using cables instead of the aluminum shafts. There were some cons with this. One was that there would be a limited rotation of motion. Second, the pulleys would have to be in a non-parallel configuration. In addition, we’d need to buy better gearboxes to withstand the high torques and there’s the concern of the set screws holding the load.

The other option was to use linear actuators. Now, previously, the design team did not look to much into this option in A term due to time constraints and initial concerns with pricing. However, after discussion with one of our advisors, linear actuators seemed like the best option that would solve the load issue, assuming we can find ones for a decent price. However, the linear actuators still had the problem of needing universal joints that could handle the load. Buying universal joints was not an option as not only were they too expensive for our needs, but we also needed them to be custom fitted for our robot.

7.1.1 Linear Actuators

After a lot of searching, we ended up finding linear actuators that was best suited for our needs considering the limitations we were dealing with. The linear actuators we found could do up to 3500N of force, which fits the needs of the robot. They also were not too big, which was an issue we had with a lot of other linear actuators because we still needed to fit within human proportions. The actuators will stick out a slight amount when pitched upwards, but it is only roughly 12mm and is not a significant amount. The cost was also manageable and still fit within our budget as we needed to get 4, with 2 per leg.

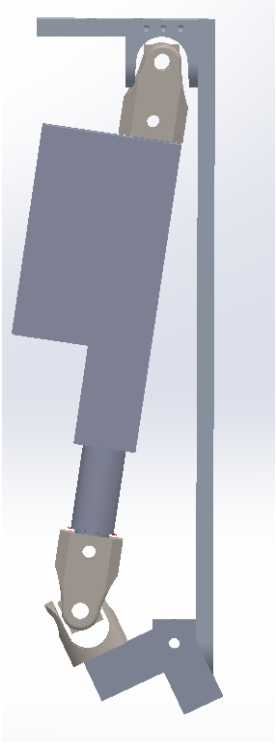


Figure 7.1: Ankle Pitch Down (actuators stick out slightly)

The main issue with these actuators were the speed. Using the speeds given online, we created a spreadsheet to see how fast it would take for the actuators to pitch from -20 to 20 degrees as a test. Based on the results, it would take roughly 4-6 seconds (depending on the load of the robot) to go from -20 to 20 degrees. This obviously is not ideal, however, given the constraints that we had with the size limitations and budget, it was the best option we had. Other linear actuators that we found had a common problem of being too big/long or did not have enough force.

No Roll; pure pitch		
Stroke (mr	Angle (from horizontal)	RPM
10	-27.51	NA
14	-18.36	1.525
18	-11.47	1.148333
22	-5.61	0.976667
26	-0.35	0.876667
30	4.52	0.811667
34	9.1	0.763333
38	13.48	0.73
42	17.71	0.705
46	21.84	0.688333
50	25.88	0.673333

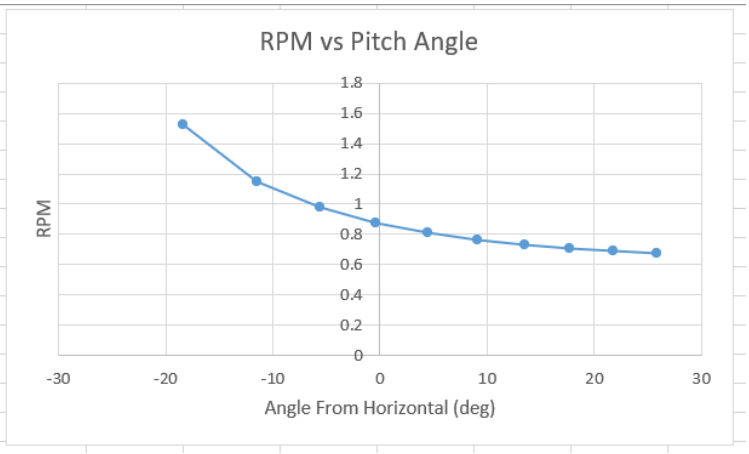


Figure 7.2: Ankle pitch rpm calculations

7.1.2 Final Design

The final ankle design can be seen in the figure below. There are a few things to note. The gray colored pieces are aluminum, while the brown color is steel. The universal joints at the bottom needed to be steel as they had to fit needle-nose bearings, which required a larger diameter and therefore could not be made in aluminum without breaking. In addition, the ground link (the triangular shaped piece in the front of the ankle) needed to be this shape because of the perpendicular force that happens on the ankle when the robot is taking a step. We found that a triangle shape was able to handle the perpendicular force without breaking, and it could still be made out of aluminum. We could have made the ground link steel and kept it as a simple rectangular prism, but with steel being so heavy, we wanted to keep as many aluminum parts as possible.

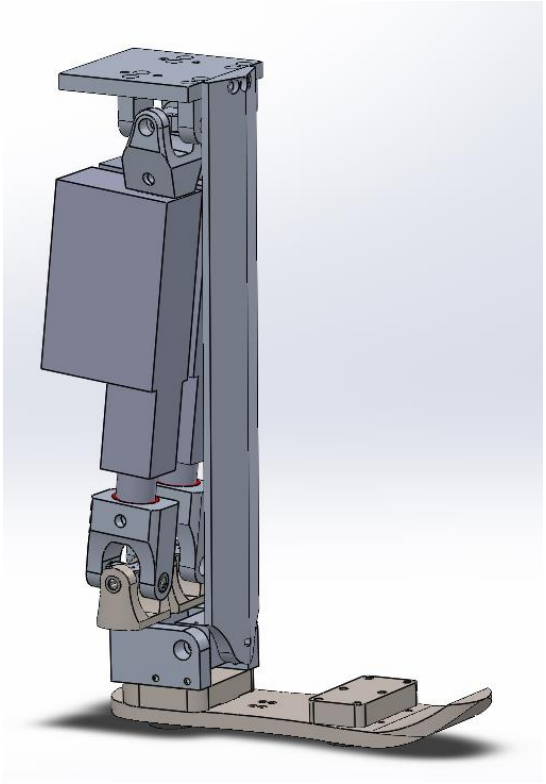


Figure 7.3: Final Ankle Design

The final lower body design can be seen below.

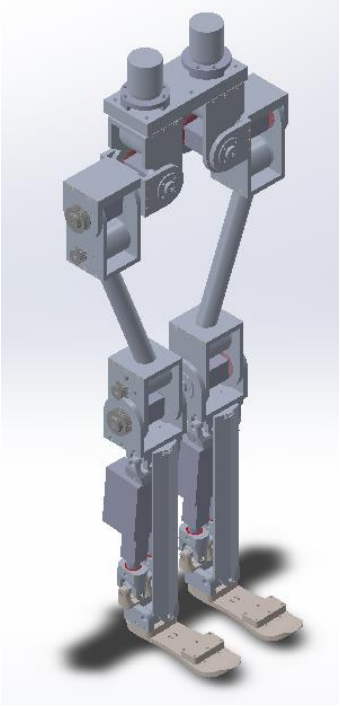


Figure 7.4: Final lower body design

7.2 Learning CAM

Before we could machine, we needed to learn how to CAM our parts to prep them for the mills. Since none of us had real experience with machining before, we knew the best course of action was to talk to lab monitors in Washburn about CAM and what the best software is to use. At first, we started out with using Esprit as that is what most ME1800 students use when they machine, and majority of the lab monitors have that class experience. However, after speaking with one of the lab monitors, we were told that Fusion would be a better option for us because it is a lot easier to use, and given our time constraints, we needed to learn how to use CAM as fast as possible.

Despite this, learning how to use CAM took longer than expected because there are a lot of nuances that goes into CAM. The major one was feeds & speeds. In the machining world, this is a well-known aspect of machining that can take a lot of tuning to get right. Essentially, each tool that is made has certain settings that the tool can operate in without breaking. This includes things such as how much stepover the tool can do, the amount that the tool can take at a time, and so on. This information can be found on the WPI manufacturing website, so we were advised to look at

that as a reference when making our tools in Fusion, but this would require more tuning as will be discussed in chapter 7.4.

Another aspect that required some thought was how many operations it would take to machine a part. In essence, a part needs to be flipped over multiple times as the vise can only grip one side at a time. In addition, the vice needs to have a flat surface to grip on, otherwise it could risk the part coming off during machining. In Fusion, each operation is known as a setup, where each setup has its own tools and toolpaths based on the type of operation that is needed. Below is an example of the CAM for one of the robot's parts. On average, each part required 4 operations to be completed based on the CAM.

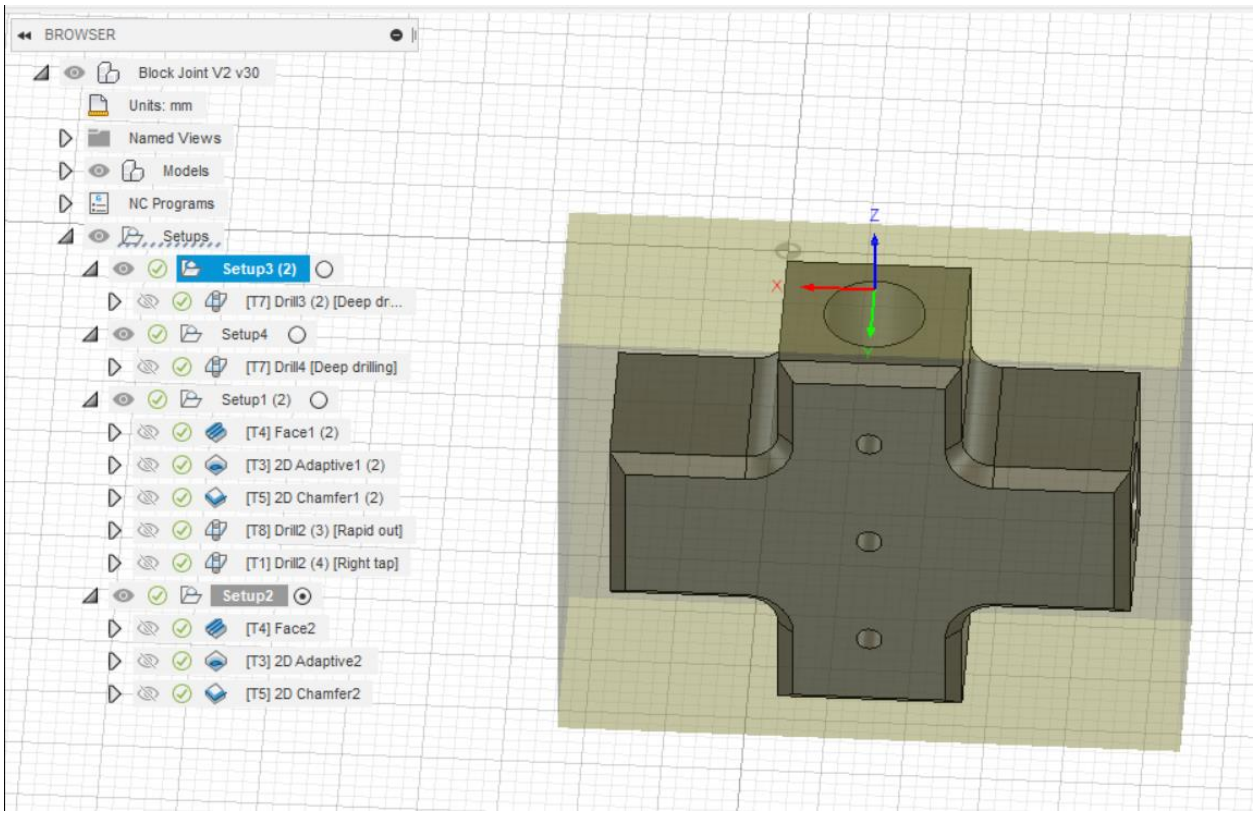


Figure 7.5: Example CAM in Fusion

7.3 Learning to Machine/Waterjet

7.3.1 Machining

A lot of what we learned with machining came from trying to machine a failed part. In general, the actual process of setting up and machining a part in the mills was not difficult to learn, but just a lot of steps to memorize. For each operation, the tools and the piece had to be probed,

along with simulating the CAM on the mill to ensure that the piece was set up properly. From there, it is just a “press and go” situation, but the only thing you need to check is the “distance to go”. Essentially, the mills can display the distance that it thinks the tool still has to go before it comes into contact with the part. This is important to check as if the distance is off, then the operation will not be successful.

Once we learned the steps with machining, we tried to machine one of our parts using the CAM we worked on. This revealed a lot of learning points with our CAM as there were many issues that came up with machining this part. For example, with the face mill, we were taking too much of a bite with each pass, which is not great for the tool as the face mill is only designed to take a little off at a time. In addition, we also learned that the tools should have a difference in their stepdown and stepover depending on how they are cutting. Basically, if the tool is doing a contour, its settings should be different than if the tool is cutting into a part with its full width. We also learned that we would need to remeasure our stock for each CAM. At first, when we were cutting stock for the test part, we set the stock length in the CAM to be what we planned to cut the stock out as. Our original idea was that since each part needed to be machined multiple times, we could have one set of gcode (the code that the mills use to read the CAM) per part. However, we did not consider the fact that it is nearly impossible to cut stock at the exact measured length. This was reflected in the test part we machined, as once we finished all the operations, we noticed that the stock was smaller than what we had inputted into the CAM, meaning the edges of the part were cut off a bit.

Figure 7.6: Failed Machine Part

7.3.2 Waterjet

Since machining is a lengthy process, we realized that some of our parts could be done in the waterjet, which would be much faster. After meeting with the machine shop assistant at 50 Prescott, we found that all we needed to provide were dxf files of the parts we wanted to cut, and from there it can be put into the waterjet software for it to cut.

Specifically, the .dxf file would be loaded into a program called ProtoLayout. In here, parts of the .dxf file that were unnecessary, such as the SOLIDWORKS logo, were removed. Once extra lines were removed, parts could be rearranged, and then a path could be generated for the water jet to follow. This path would then be loaded into the program ProtoMake where the actual control of the water jet takes place. ProtoMake both calibrates the water-jet, and provides a simulation of the water-jet pathing to ensure that the stock will be cut correctly.

To prep for water-jetting, we took all the pieces that could be waterjet (mainly the hip and knee brackets) and organized them onto 1ft x 1ft sheets, as that was the max size the waterjet could hold. This was done in Solidworks, as shown below.

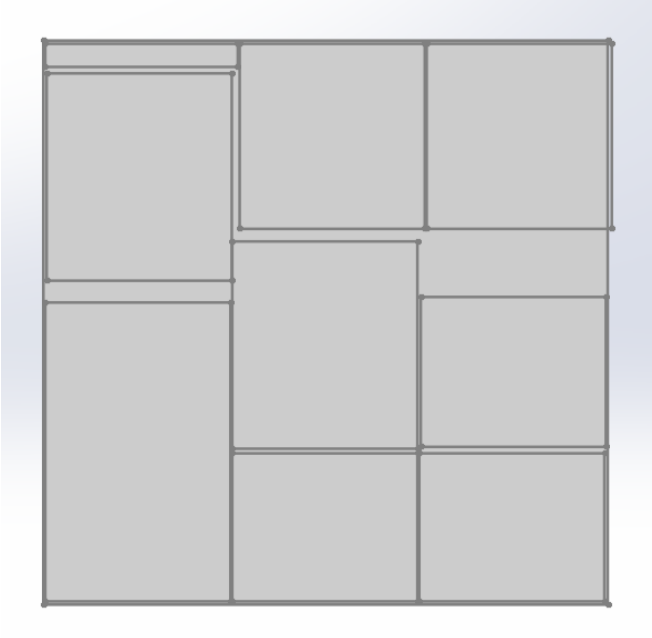


Figure 7.7: Waterjet stock layout

7.4 CAM/Machining Process

7.4.1 Stock Setup

As mentioned previously, each piece needed to have its own gcode due to the slight variations in stock size. Therefore, we came up with a numbering system for the gcode to make it simpler to know which gcode goes to what part. The basic idea is that the gcode will always be a 4-digit number. The first two numbers refer to what part it is, which we defined in a spreadsheet, For example, the ground link has the first two numbers as "01". The third number refers to the piece number. Back to the ground link, since we needed two ground links, the third number will either be a "1" or "2" based on if it is the first or second ground link piece. The last number refers to the operation number. Therefore, the gcode "0112" would refer to the 2nd operation of the first ground link piece. This naming convention was put into a spreadsheet, which we used to keep track of which operations were done and what still needed to be done. We also enlisted help from several members of the controls and sensor subteams, as with the large number of parts that needed to be machined, it would take too long if it was just the four design team members.

As we cut the stock for each piece, we measured the stock size as accurately as possible and inputted into our spreadsheet, which would then be adjusted in the CAM. Since each operation can take time, we had columns where someone could put their initials, indicating that they finished an operation. That way, if someone was only able to do 2 operations, someone else can see that and can pick up on the 3rd operation, and so on.

General Information						Process 1					Process 2				
Fusion Part Name	Section of Robot	Wave Number	Stock Starting Size	Piece Number	Quantity	Your Initials	Machine	Code/File Link	Start Date	End Date	Your Initials	Machine	Code/File Link	Start Date	End Date
Aluminum 2024 Bar 1 (620mm x 101mm x 38mm)															
Ground Link	ankle	1	382mm x 101mm x 19mm	01_	2		Mill	Done-JKG							
[redacted]			402.5mm x 101.6mm x 15.83mm	011_		JKG		0111	29-Nov	30-Nov	JKG		0112	30-Nov	30-Nov
			399mm x 101.6mm x 18.3mm	012_		JKG		0121	2-Dec	2-Dec	JKG		0122	2-Dec	2-Dec
Aluminum 2024 Bar 2 (620mm x 101mm x38mm)															
Bracket Stage 2 Piece 3	hip	1	96 mm x 101 mm x 38 mm	02__	2		Mill	Done-AH				Mill	Done-AH		
[redacted]			94.59 x 101.7 x 38.17	021_		JM		0211	2-Dec	2-Dec	JM		0212	2-Dec	2-Dec
			95.14 x 101.6 x 38.18	022_		JM		0221	4-Dec	4-Dec	JM		0222	4-Dec	4-Dec
Ankle Roll Actuation Link V3	ankle	1	80mm x 101mm x 38mm	03__	2		Mill	Done-AH				Mill	Done-AH		
[redacted]			79.6 x 101.7 x 38.17	031_		RD		0311			RD		0312		
			79.78 x 101.7 x 38.17	032_		RD		0321 (WIP)	1-Dec	1-Dec	RD		0322	1-Dec	1-Dec
Ankle Pitch and Roll Coupling V2	ankle	1	50.5mm x 32mm x 30mm	04__	2		Mill	WIP-AH				Mill			
[redacted]			50.73 x 47.55 x 38.17	041_		RD		0411	8-Dec	8-Dec	RD		0412	8-Dec	8-Dec
			51.18 x 46.48 x 38.19	042_		AJH			7-Dec	7-Dec	AJH			7-Dec	7-Dec
			51.2 x 49.88 x 38.17	042_		AJH		0421	11.29	11.29	AJH		0422	11.29	11.29

Figure 7.8: Snippet from our Manufacturing Spreadsheet

In addition to the spreadsheet, we also laid out all the cut stock onto a table with each part laid on top of their respective gcode label. That way, someone can easily find the piece they are looking for and know which gcode is used for that piece without having to check the spreadsheet.

Figure 7.9: Stock layout

7.4.2 Changes due to CAM

Our original intention was to have all the CAM "finalized" before we began machining, as that way, we could just focus on machining straight for a couple weeks. Considering that we had help from the other subteams, we did not want them to worry about the CAM not being correct. That being said, we unfortunately ran into some issues with the parts that required CAM to be adjusted during the machining period, which caused delays in our timeline.

For the most part, we had figured out the feeds and speeds from machining the test part. However, there were still some adjustments that needed to be made while machining. One of them was with the cutting feedrate. Essentially, the feedrate we had for some of the tools was too fast, meaning that the tool was moving across the part faster than it could theoretically handle. This was caught early on, so no tools were damaged. Despite this, there were some syncing errors with the tools in Fusion, meaning we had to go through each part and manually check to make sure the correct settings were applied.

The cutting heights was another adjustment point that had to be made. In Fusion, one of the settings that is important is the "bottom height", which essentially means the lowest point that the tool will cut down to during an operation. This is important because when the part is fixed in the vise, you want to make sure the tool is not cutting down to the point where it will interfere with the vise. With some of the thinner parts, the bottom height had to be adjusted after physically

placing the part in the vise as at times, the clearance was less than 1mm, which is too close for comfort.

Another point of adjustment was the order of the operations we had for certain parts. This was mainly applicable to one of our parts, which was called the block joint. The block joint consisted of four operations, with the last two being a simple drill to make a hole. However, when it got to the drilling operations, the part ended up getting warped, as shown in the figure below. What we eventually found out was that because the first two operations involved facing and clearing out stock to make the piece smaller, the aluminum that the drill was going through was too thin, which caused the piece to get warped. Therefore, we made the drilling operations first, which solved the issue. This kind of issue was something that we could not have known about until we started machining, which shows that sometimes there is no way to "finalize" CAM without physically machining the part.

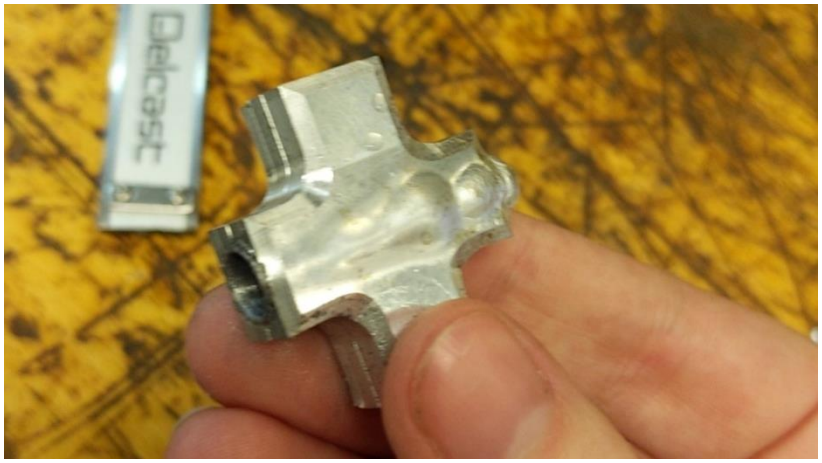


Figure 7.10: Warped Block Joint

7.4.3 Changes due to Manufacturability

Besides changes in CAM, there were also some other manufacture related changes that had to be made while we were machining. One of those changes involved the size of the taps that we were using. For the block joints, we originally were using M2 taps due to the part being a small size. However, during the machining process, the M2 tap proved to be too fragile and broke easily. This could have been a feeds and speeds setting issue, however, we only had one more M2 tap to work with, as the machine shop did not have any M2 taps. Therefore, we had to adjust the CAM

to use M3 taps, as they were slightly bigger and should not break as easily, and the machine shop had more of them. This switch worked for the remainder of the block joints.

Taps continued to be an issue when working with steel parts. In general, machining steel became more of a pain than previously anticipated due to the difference in feeds and speeds and the nature of steel being more difficult to machine in general. Some of our steel parts required tapped holes, however, we were not sure how tapping steel would go and if the taps would break easily. After talking to a lab monitor, we tested tapping a hole on a scrap piece of steel, and that ended up working. However, this must have been a fluke, as when doing a tapping operation on one of our steel parts, the tap broke and got stuck inside the part. After some discussion, we found that the tap was most likely breaking because it was trying to drill into excess steel shards that were being produced when initially tapping. Since taps are not made to drill into material, and since steel is more durable, the tap therefore broke. Thus, it might be better to hand tap the steel, as that way, we can tap incrementally, pausing to clear out the excess steel that is produced.

Figure 7.11: Broken Tap (left hole) in Steel Part

Besides the taps, there were other setbacks with machining the steel. As mentioned before, the feeds and speeds were difficult with steel mainly because they were different than aluminum. While the WPI manufacturing labs website had recommendations for feeds and speeds on steel, the issue was that not many of the lab monitors had much experience working with steel. Therefore, we had to be extra careful with machining the steel because we did not have a lot of extra stock to

work with due to the price, and there was not much help we could get without just trial and error. Since we had already broken some tools from the machine shop, we decided to purchase a titanium coated drill bit for steel, as we wanted to maximize our chances with machining steel moving forwards. In general, there was a lot of uncertainty in the beginning with steel because we had to figure things out ourselves, but since we successfully machined a couple steel parts, we felt more confident moving forwards. The main issue was with tapping the steel.

In addition to steel, there were some adjustments that had to be made to some of the aluminum parts, some due to our own fault. One issue was the hole sizes for fitting bearings. Some of our parts had holes for bearings to fit inside, which were for shafts to slide through. However, we made a slight error in measurements when initially designing, as we found out after machining one part that the holes were off by half a millimeter, which was just enough to not fit the bearings in. Unfortunately, we realized that this offset in measurement was also found in other parts that had bearing holes. This was adjusted for the remainder of the parts, but we had to fix the other holes. This was doable, but additional CAM had to be made for it, which added to our overall delay. Similarly, some of the bearing holes were too big, which unfortunately could not be fixed by machining. Instead, these could be fixed by 3D printing small spacers, but we did not have time to do this during B term.

Figure 7.12: Bearing Size Issue in Aluminum Part

7.4.4 Waterjet Parts

The general procedure for the waterjet parts was to waterjet the pieces out of 1ft x 1ft stock, then either countersink or tap the holes, depending on the piece (sometimes needing both). If some pieces needed holes on the side of the piece, then those needed to be drilled also.

Each piece needed to be 10mm based on our design. Unfortunately, the stock we were able to get came at ½ inch thick, meaning we needed to face it down about 2.7mm. In addition, the stock came in 2ft x 2ft sheets, meaning we needed to cut the stock to 1ft x 1ft sheets first before facing. The only way to cut sheets that large was using the plasma cutter in Washburn, as there was no other machine that could fit that stock. Learning how to use the plasma cutter was not difficult, but it was a lengthy process because we were told to read an instruction guide online, which some figuring out. We also had to make special CAM for the plasma cutter to tell it what cutting pattern to do (see Appendix G for plasma cutter setup).

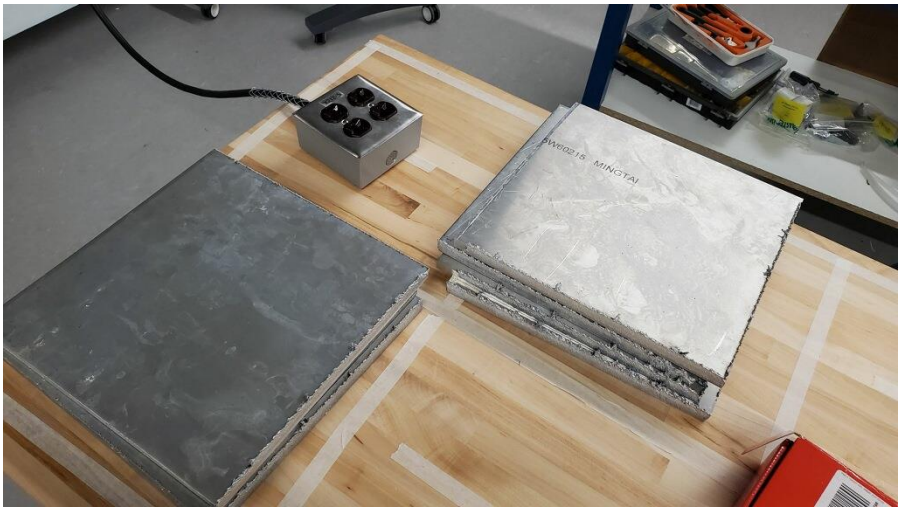


Figure 7.13: Waterjet stock Cut to 1ft x 1ft

Once that was finished, we moved on to facing the 1ft x 1ft stock down to 10mm. We used the VM machine, as the normal mills we used were too small. However, when we began facing, we noticed that it produced a loud, unpleasant sound that sounded like something was breaking. After speaking to a lab monitor, we were informed that because the piece we were facing was large, a lot of the piece was not being physically supported by the vise, which causes chatter. This essentially means the piece is vibrating ever so slightly, which causes the loud noise. While the chatter was not harmful to the machine or the tool, it was a problem for the stock, as due to the

vibrations, it caused the surface to be uneven and less than 10mm, despite the CAM programming otherwise. Therefore, we had to pivot to just cutting the pieces with the waterjet first, then facing each induvial piece afterwards. While this took more time, it was better for the quality of the pieces overall.

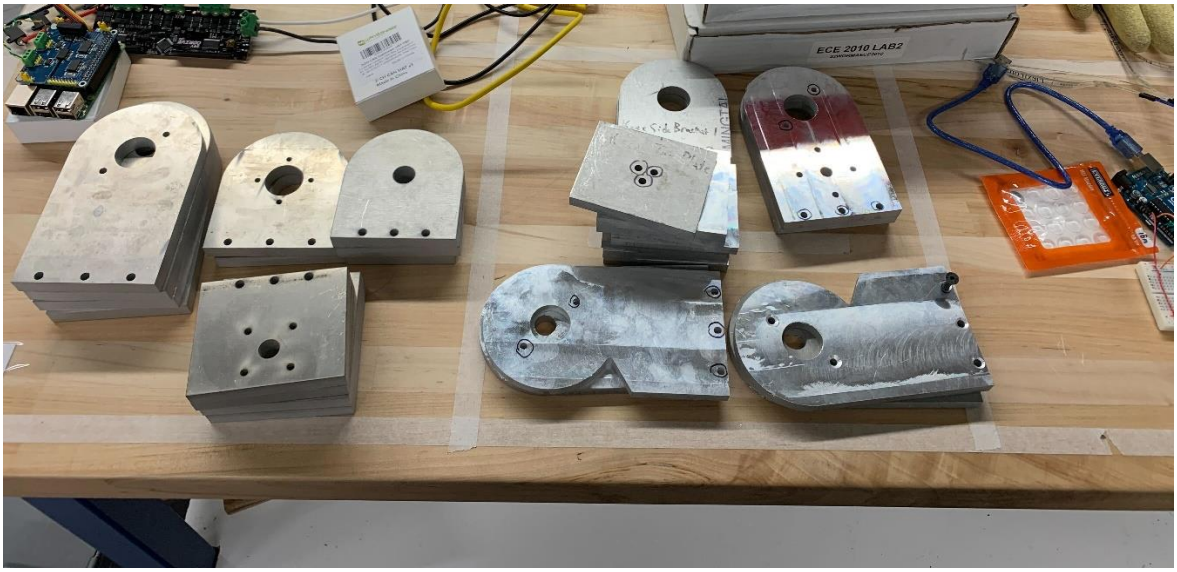


Figure 7.14: Waterjet Pieces Cut

8.0 Assembly Documentation

In this section, we detail based on the subassembly of the robot, the assembly instructions to document the process, and the pieces of the robot. The four major sections of the robot are detailed below in the following order: ankle, knee, hip, and test rig assembly. During each step, each picture shows the assembly of the section parts, with the caption listing the names of the pieces found in the SolidWorks assembly. The assembly shown in each figure will be further explained before the image appears to assist with the documentation and assembly of the robot sufficiently. Below in Figures 8.1 and 8.2 are all the parts for the left and right legs laid out in the general area of assembly for the general reference of part locations. The assembly pictures are all from the assembly of the right leg.

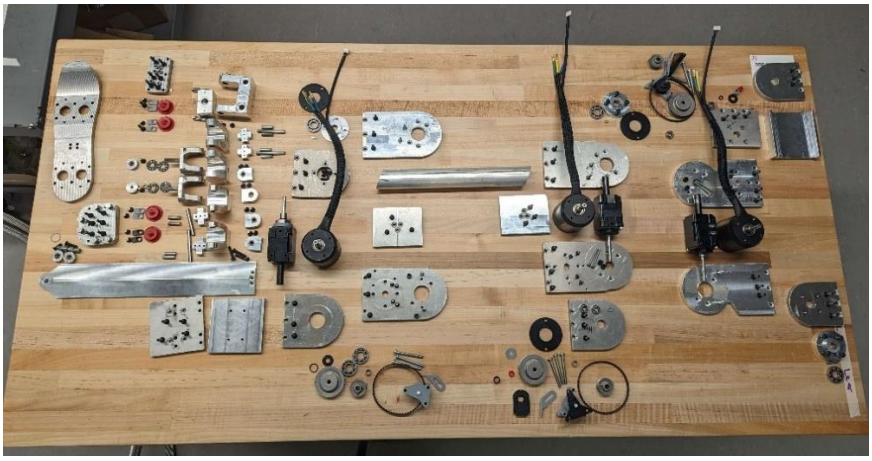


Figure 8.1: All parts laid out in general order for the left leg.

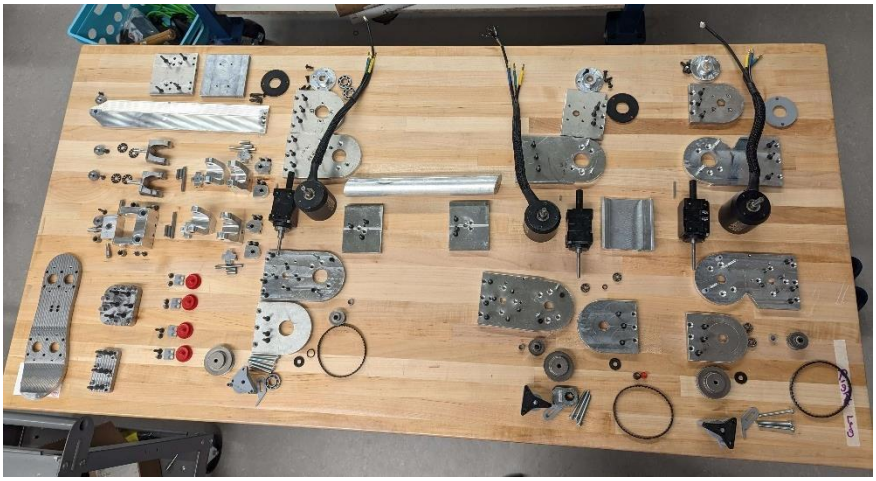


Figure 8.2: All parts laid out for the right leg.

8.1 Ankle Assembly

The first section that should be assembled during ankle assembly is the ankle pitch and roll coupling assembled with its bearings and inserted into the ankle roll actuation link. Once the piece is within the actuation link, then slipping the shafts through the bearings and should be set screwed into place, screwing tightly. Next, the bottom joint connectors should be inserted into the two main holes in the actuation link with thrust bearings between the top contact and on the bottom between the actuation link and the rotational adjustment piece, screwing into place tightly, before loosening slightly for a free spin. Once these are in place, the actuation link can be mounted on the ankle block, screwing into place with two screws on three sides: left, back, and right. This completed section is shown below in Fig 8.3, and a front view of the assembly is shown in Fig 8.4.

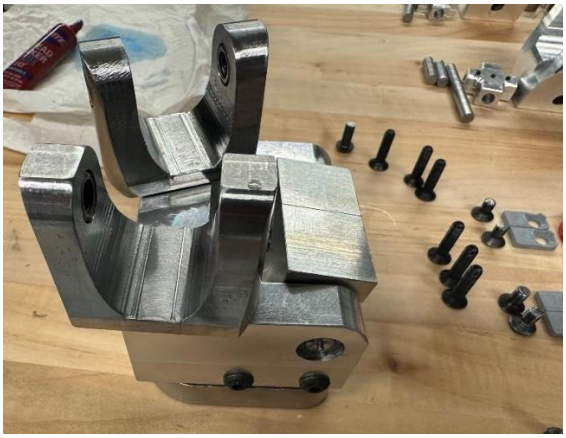


Figure 8.3: Connection between the Ankle block, Ankle Roll Actuation Link, Ankle Pitch and Roll Coupling, and Bottom Joint Connector V3

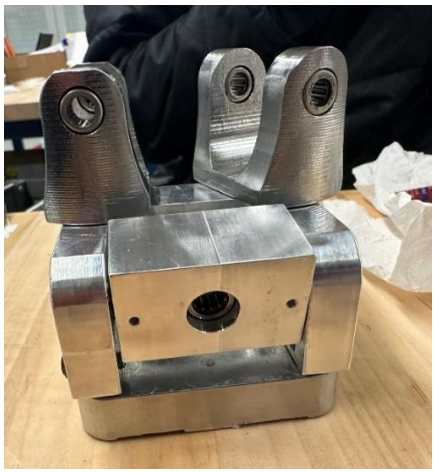


Figure 8.4: Front view of connection between the Ankle block, Ankle Roll Actuation Link, Ankle Pitch and Roll Coupling, and Bottom Joint Connector V3

The first ankle assembly is then mounted onto the foot, using six flathead M5 screws connecting the bottom of the ankle block to the top of the foot at the back. These screws should be screwed in fully, resulting in a tight fit. The same process is followed with the top block, using the same type of screws, mounted to the front of the foot, with nyloc nuts on the top to secure the screws. All the screws should sit flush with the bottom of the foot. This is shown in Figure 8.5.

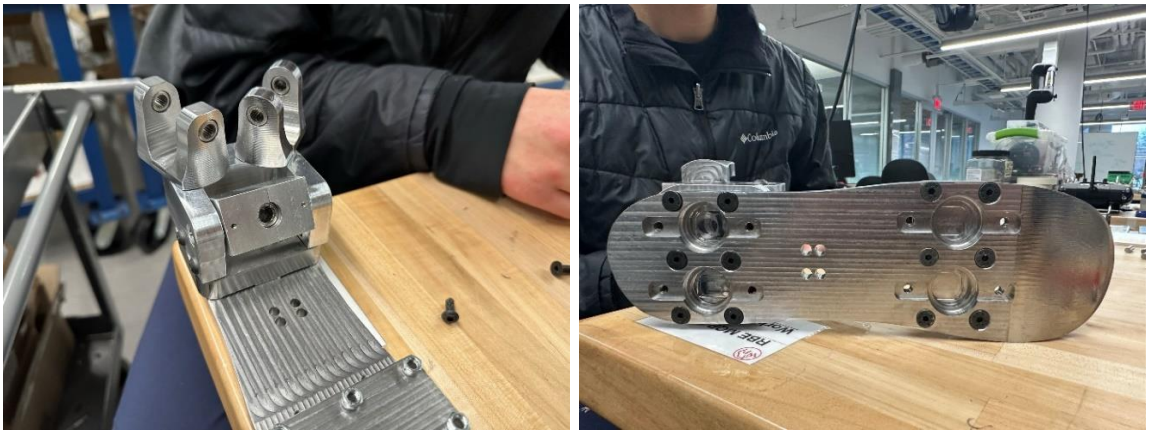


Figure 8.5: Connecting the foot to the Ankle Block and the Top Block (Left) with M5 countersink screws in the bottom of the foot (right)

Next, the attachment shaft for the ground link is slid into the ankle roll coupling and secured to the piece using a rotation adjustment piece at the back of the ankle roll coupling. Secure tightly. This is shown in Figures 8.6 and 8.7. Once the shaft is secure, slide the ground link onto the attachment shaft, and using another rotation adjustment piece, screw in tightly while ensuring the piece can still rotate, see Figure 8.8. Try twisting around the ground link to ensure this section is assembled correctly; moving from side to side and front to back should be easy.

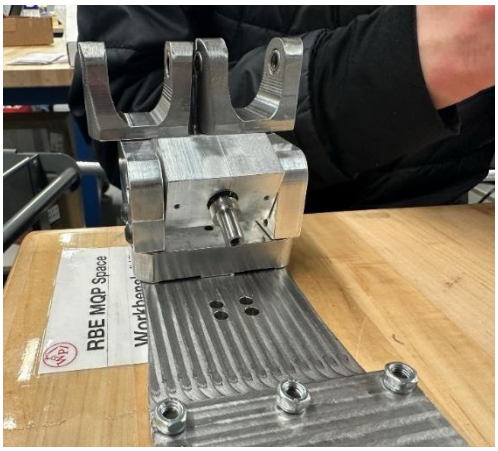


Figure 8.6: Attaching shaft into the Ankle Roll Coupling

Figure 8.7: Attaching Bottom Mount of Rotational Adjustment to the back of the Ankle Roll Coupling

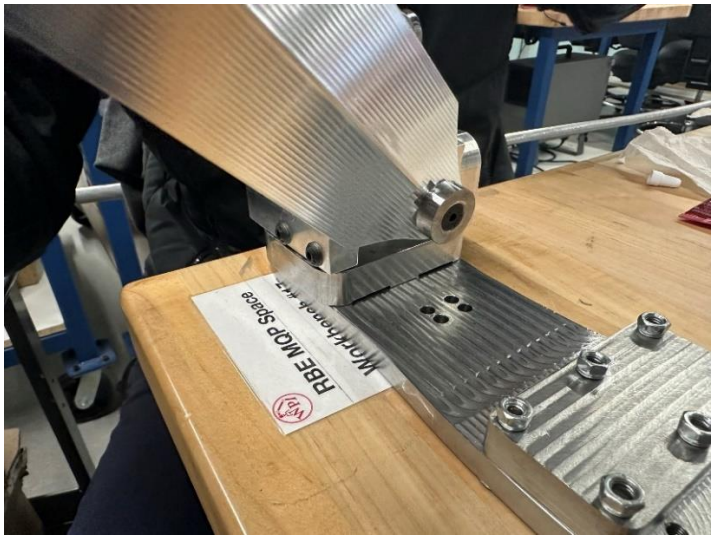


Figure 8.8: Attaching the Ground Link to the front of the Ankle Roll Coupling via a Bottom Mount of Rotational Adjustment Piece

As a note during the assembly documentation, the assembly instructions above do not mention the linear actuators as there were problems receiving the correct ones. As a result, 0.5” aluminum rods were made to replace the actuators as a temporary replacement in order for the robot to be able to stand on its own. Additionally, in order to add back in the 4 DOF that was lost, a spring compression system was used on the left calf of the robot. During initial testing, the spring system was found to be too stiff as the springs had a spring constant of about 80lbf/in. The spring stiffness was chosen based on very quick estimations rather than a full dynamic analysis due to the lack of time that the Controls team had to program with the physical robot.

8.2 Knee Assembly

The next stage of assembly is the knee, which is built starting from the ankle assembly. However, these two sections can be separated and done individually by moving the first step, attaching the ankle top plate to the ground plate, to be the final step instead. The ankle top plate is attached to the ground link at the top, with three M5 button head screws from the top of the ground link into the side of the top plate, as shown in Figure 8.9. This step's critical part is ensuring that the four sets of two countersinks face upward during assembly.

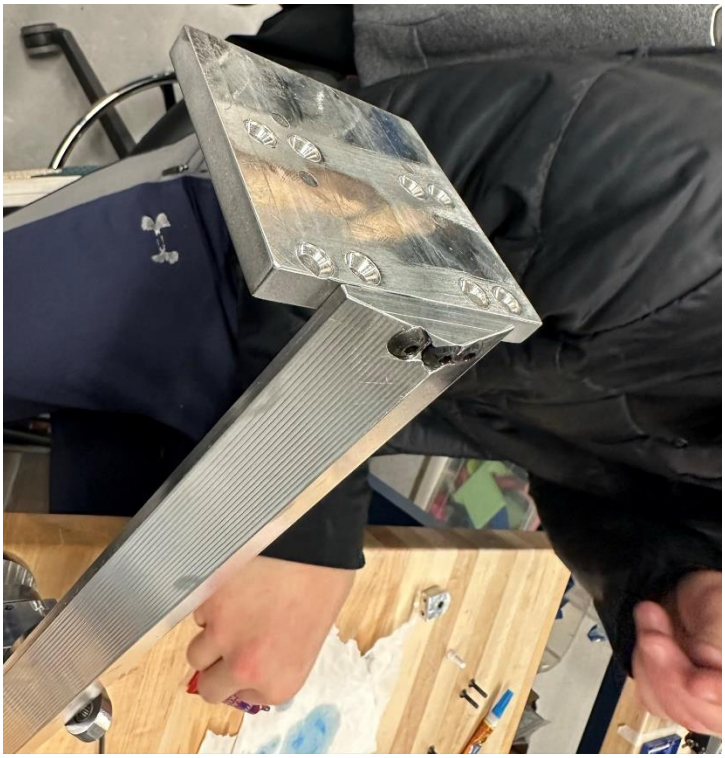


Figure 8.9: Attaching Ankle Top Plate to the top of the Ground Link with M5 Button Head screws

Once the top plate is attached to the ground link, using the shortened countersink M5 screws, attach the u-joint attachments to the top plate. There should be four u-joint attachments, each with a bearing installed. Make sure that the flat side of the bearing faces inward toward the other u-joint; see Figure 8.10 and 8.11 for reference. Once all four u-joints are attached, perform a quick test to ensure the block joint can slide between them freely. Then the knee bottom plate can be mounted to the ankle top plate using countersink M5 screws. Ensure that pieces are lined up and that the holes on the sides of the knee bottom plate face outward to the left and right of the

ground link. They will be attached by screwing up from the ankle top plate to the knee bottom plate.

Figure 8.10: Attaching 4 Ankle Top Plate U-Joint Attachments to the bottom of the Ankle Top Plate (Left) with M5 Countersink Screws from the top (right)

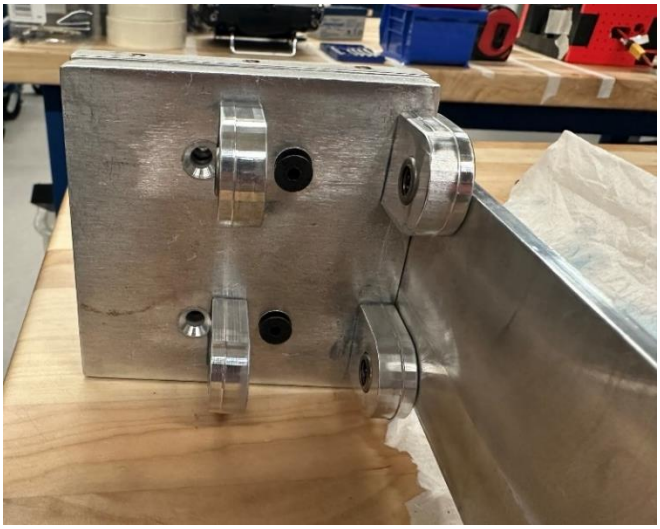


Figure 8.11: Attaching the Bottom Plate on top of the Ankle Top Plate using M5 Countersink Screws (bottom view)

Next, insert two small shafts through the joint connector with the block joint inside the joint connector, lining up the holes. This process is shown in Figure 8.12. Once the shafts are within the bearings on the joint connectors and do not block the other through-hole in the block joint, secure the shafts using the set screws. For one leg, this process should be completed two times. Then, this piece can be inserted with the block joint lined up inside the u-joint attachments on the bottom of the ankle top joint, shown in Figure 8.13. With the holes lined up, slide a long shaft through the u-joint attachments until the shaft is relatively centered, then carefully screw in

the set screw to hold the shaft in place. This process is difficult due to the number of parts in this area, and it may take several minutes to secure the shaft fully. Once the assembly is complete, twist the joints ensuring free motion up, down, and side to side. Repeat this process for the second universal joint.

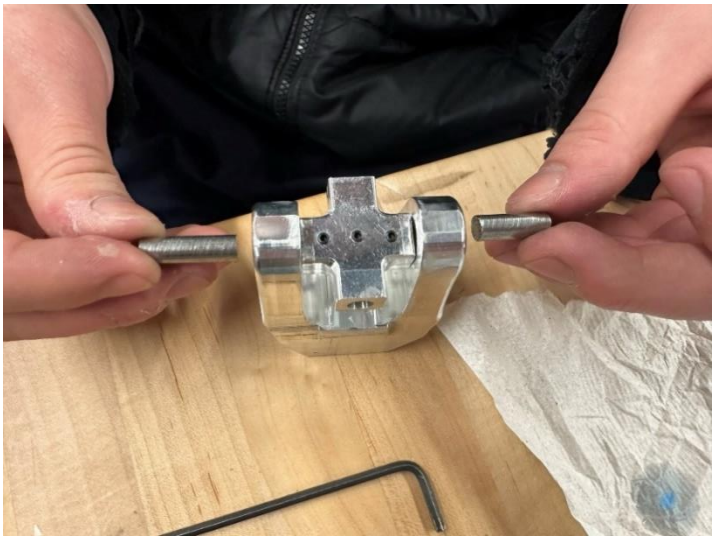


Figure 8.12: Putting two small shafts into a Joint Connector with a Block Joint in between

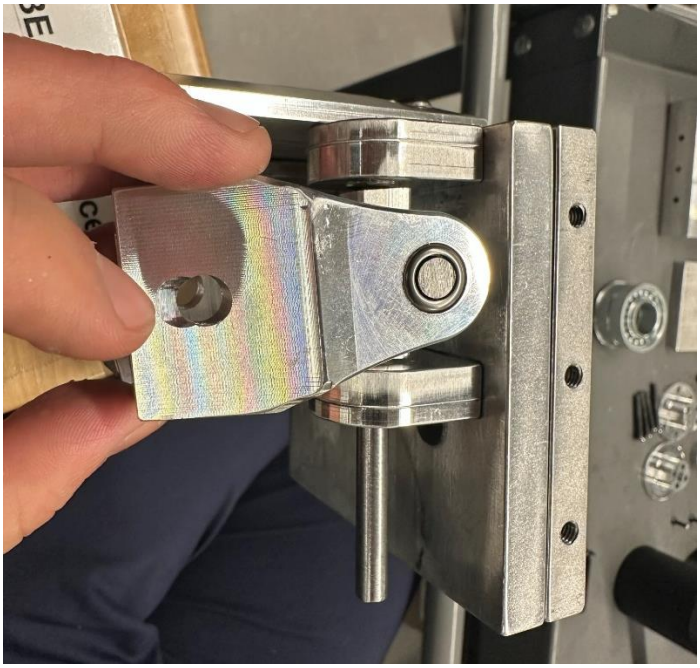


Figure 8.13: Putting a Joint Connector between the Ankle Top Plate U-Joint Attachments with a long shaft going through

Figure 8.14: Both Bottom Joint Connectors attached to the Ankle Top Plate U-Joint Attachments

Next, attach the exterior side bracket, also called knee Side Bracket 1 to the exterior side of the bottom knee plate with M5 countersink screws, shown in Figure 8.15. On the exterior knee plate, knee side plate 1, line up the gearbox into the top hole with the curve on the outside. The output shaft should be lined up through the hole; this is the longer metallic shaft. There are two sets of attachment holes; use the two horizontal holes. Insert the 3D printed standoffs, called screw sleeve spacers, between the gearbox and the plate, line up the 8-32 long screws into the holes of the gearbox, and screw in tightly; this process is shown in part a of Figure 8.16. Next, insert the motor into the other hole, and line up the mounting holes. Ensure the wires are facing toward the gearbox and on the back side of the robot; see Figure 8.16, part b, and Figure 8.17 for reference. Use the M4 countersink cut screws, secure them tightly, and test the motor's spin; it should spin freely.

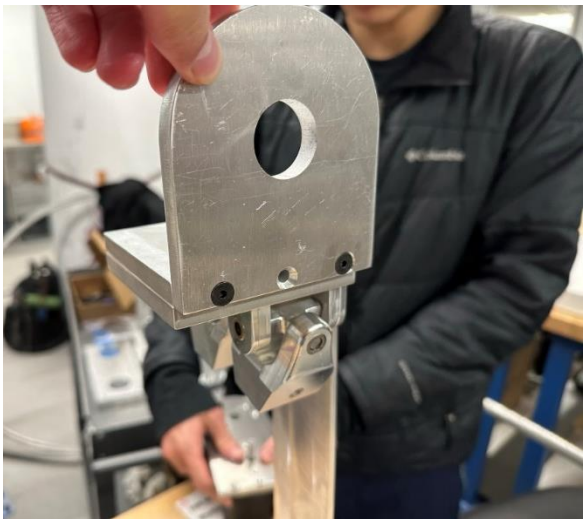


Figure 8.15: Attaching Side Bracket 1 to the Bottom Plate with M5 Countersink Screws

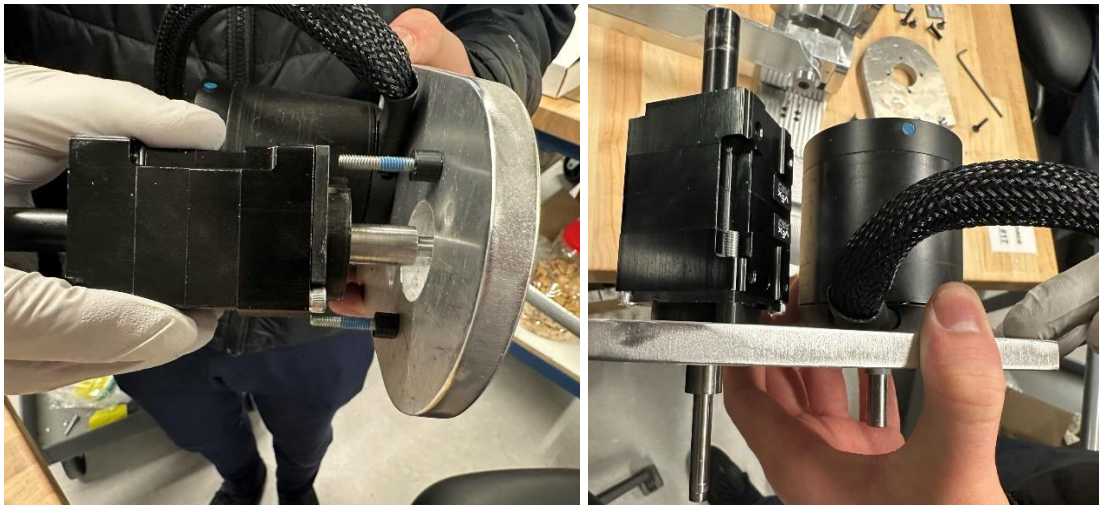


Figure 8.16: Attaching a Gearbox using screw sleeve spacers into the Side Plate 1(left) and a motor below using M4 Screws (right)

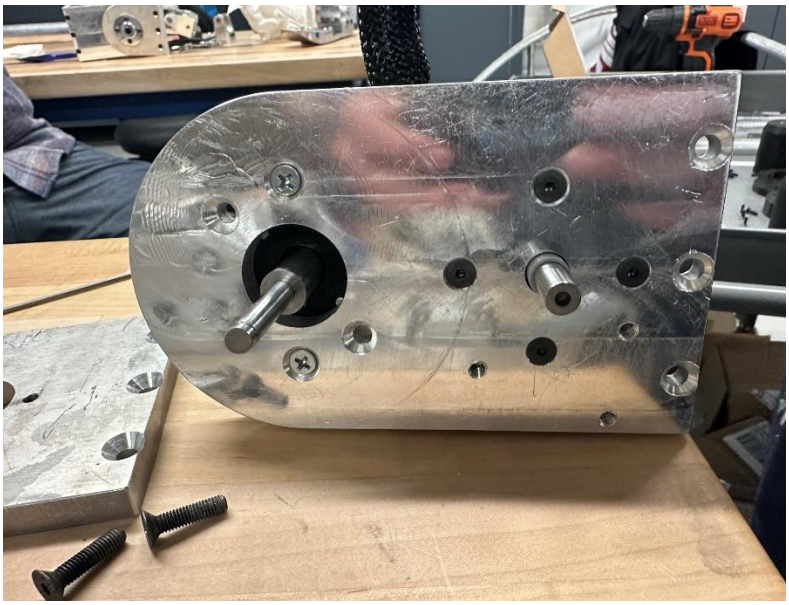


Figure 8.17: Side view of the Side Plate 1 with a gearbox and motor attached

Once the gearbox and motor are attached to Side Plate 1, slide on Side Plate 2 onto the other side of the pair. Slide the large PLA spacer onto the gearbox shaft then align Side Plate 2 onto the shaft, shown in Figure 8.18. Using countersink M5 screws, screw side plate 2, the spacer, and the gearbox, and make sure the piece is secure. Once the plates are attached, slide two bearings and a spacer onto the shaft, with the spacer between them, as shown in Figure 8.19. With the plates assembled and ensuring that bearings don't slip off, put the plates within the knee bracket assembled on the ground link. The shaft for the gearbox should go through the plate and the bracket with the bearings in both pieces, as shown in Figure 8.20.

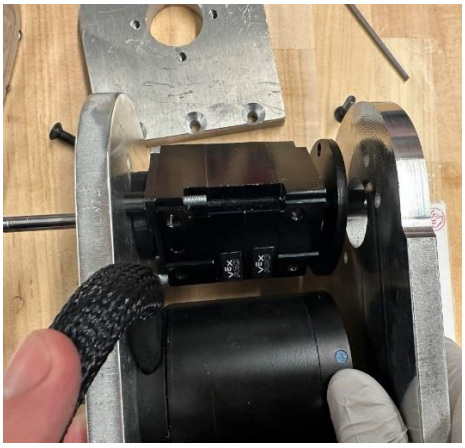


Figure 8.18: Attaching Side Plate 2 with a gearbox spacer to the gearbox

Figure 8.19: Bearing and spacer layout that goes on the gearbox shaft

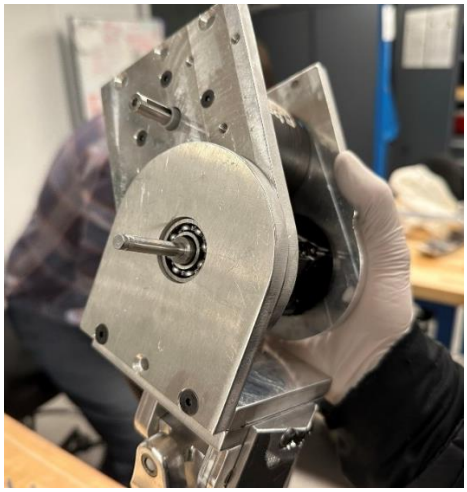


Figure 8.20: Attaching Side Plate 1 to Side Bracket 1 using M5 Countersink Screws

With the plates within the brackets, attach Side Bracket 2 to the knee bottom plate using countersink M5 screws as shown in Figure 8.21. Insert a key into the gearbox output shaft, then slide on the knee output shaft connector using M5 buttonhead screws; it should be a snug fit and should spin the gearbox without wiggling, shown in Figure 8.22.

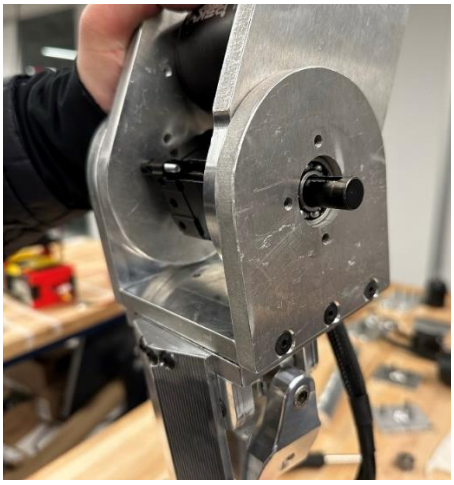


Figure 8.21: Attaching Side Bracket 2 to Side Plate 2 with M5 Countersink Screws

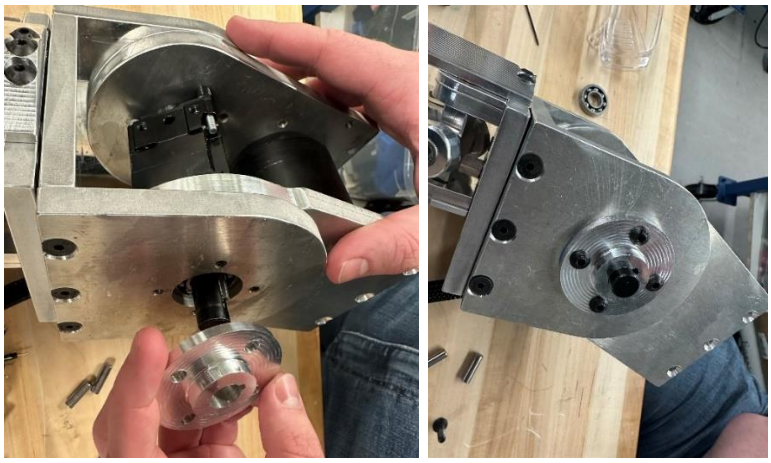


Figure 8.22: Attaching the Knee Output Shaft connector with a key in the gearbox shaft (left) using M5 Button head Screws(right)

Next, on the other side of the knee, working with the exterior of the knee, slide on another bearing and a PLA spacer on the gearbox shaft before sliding the large pulley onto the shaft, as shown in Figure 8.23. Ensure that the pulley sits fully down on the shaft, with a section of the shaft sticking out, then fix the pulley to the shaft, fully screwing in both set screws. Then on the motor shaft, slide the PLA shaft collar on first, then slide on the smaller pulley with the orientation inverted, as seen in Figure 8.25. Next, assemble the tensioner, which is shown in Figure 8.24. Line up the general orientation with the holes on the bracket. Then, carefully slip on the timing belt, wrapping around the larger pulley first, then over the smaller pulley and the tensioner, shown in Figure 8.25. Once the belt is in place, secure the small pulley with both set screws. Then fix the tensioner into place using three long 8-32s.

Figure 8.23: Attaching a bearing, spacer, and large pulley on the gearbox input shaft

Figure 8.24: Layout for assembling a tensioner for the pulleys

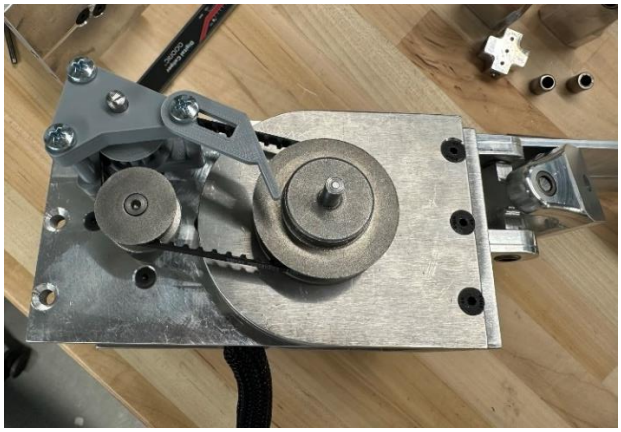


Figure 8.25: Attaching the tensioner and belt to the pulleys

8.3 Hip Assembly

Once the knee area is almost complete, the only piece missing is the top plate of the knee, which should be assembled onto the femur first, before being attached to the knee. Line up the

holes on the top plate with the holes on the end of the femur. Once attached, the slant of the femur should be in the direction of edges with the holes on the side. Attach the Knee Top Plate and the Bracket Stage 3 Top Plate onto the other side. Attach both with countersink M5 screws, making sure both plates are parallel, and the holes should be on the same side. The plate's attachment and the section's overall look are shown in Figure 8.26 and 8.27.

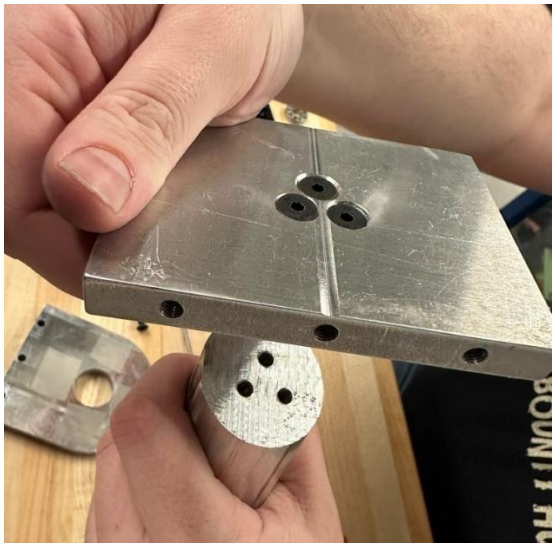


Figure 8.26: Attaching the Top Plate onto the Femur with M5 Countersink Screws

Figure 8.27: Femur with Bracket Stage 3 Top and Top Plate Attached

Next, repeat the same process undertaken with the knee, by placing the gearbox and the motor into bracket stage 3 in plate, which is shown below in Figure 8.28. Use the M4 countersink screws for the motor and the 8-32 screws with the PLA screw sleeves for the gearbox. Then attach the bracket stage (BS) inside plate to the BS 3 Top Plate with three M5 countersink screws. The next step is to slide the PLA spacer onto the gearbox shaft, then attach the BS 3 out plate on the

other side of the motor and gearbox. Screw the plate to the gearbox with the 8-32 screws, then attach the plate to the BS 3 Top Plate with three countersink M5 screws, as shown in Figure 8.29.

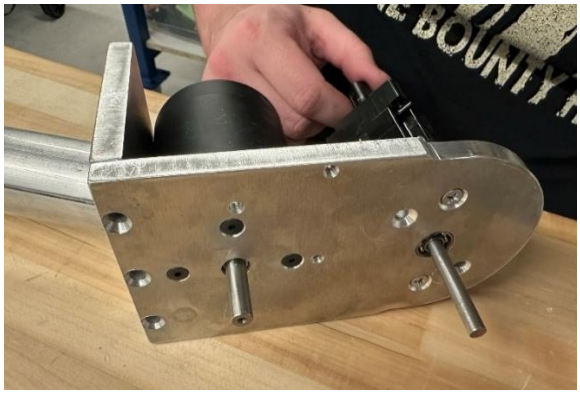


Figure 8.28: Attaching Bracket Stage 3 In to Bracket Stage 3 Top with a motor and gearbox (attached the same way as the knee)

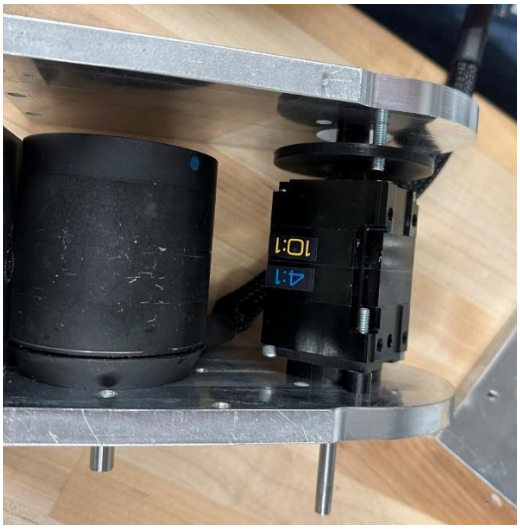


Figure 8.29: Attaching gearbox to Bracket Stage 3 Out with spacer

The next step is to build stage 2 of the hip around stage 3, starting with the output side of the BS2. Slide the BS2 Out Plate onto the gearbox shaft, then secure in place by putting the output shaft adapter onto the shaft and securing with a key. Then screw the output shaft connector to the BS2 Piece 2 with four M5 cut buttonhead screws, as shown in Figure 8.30. Then secure BS2 Piece 3 over the BS3 setup, the curve of the piece, should round the section. The piece should be connected with three M5 countersink screws to piece 2, shown in Figure 8.31.

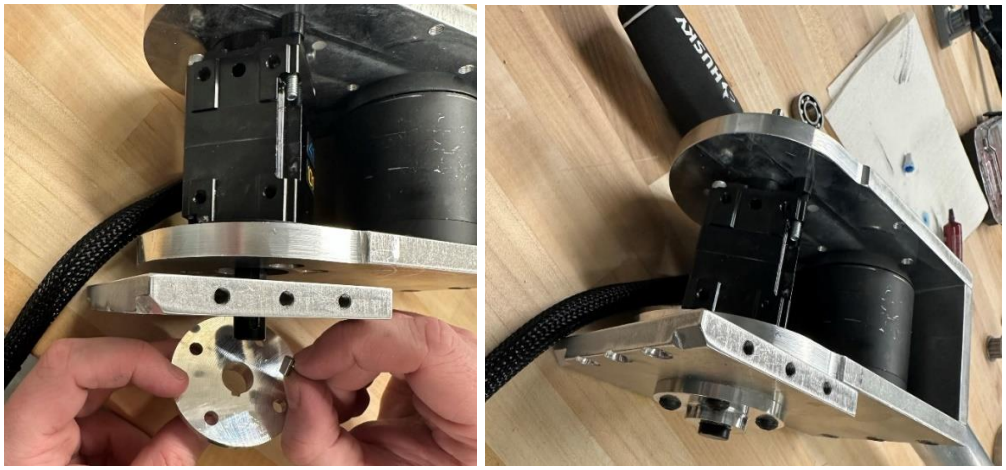


Figure 8.30: Attaching a Knee Output Shaft Connector and BS2 Piece 2 to the Bracket Stage 3 Out (left) with M5 Button head Screws (right)

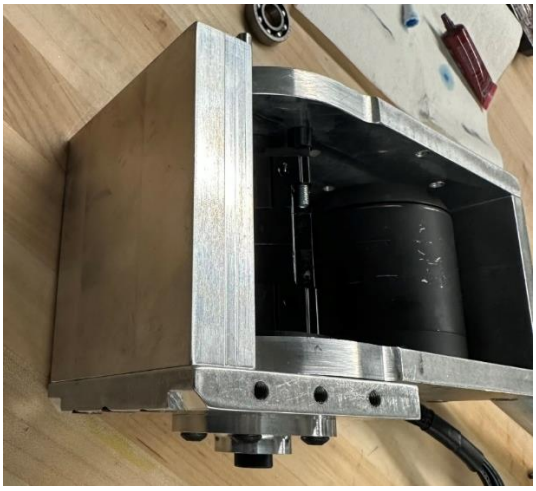


Figure 8.31: Attaching BS2 Piece 3 to BS2 Piece 2 with M5 Countersink Screws

The next stage of assembly entails attaching the next stage, BS2 to the other side of the BS3 assembly. Slide on two bearings and a spacer onto the shaft of the gearbox, with the spacer in between the bearings, then attach the BS2 Piece 1 onto the shaft and attach to Piece 3 with three countersink screws, tightly fastened as shown in Figure 8.32. Next slide on the shaft collar onto the motor shaft, before affixing the small pulley to the shaft inverted, and the large pulley to the gearbox shaft with a PLA spacer between the pulley and the plate, as shown in Figure 8.33. Fix both pulleys in place by fully tightening the two set screws on each of the pulleys. Then, following the same process as with the knee, ensure the timing belt is wrapped around the pulleys and then attach the tensioner mount with the long 8-32 screws, with the tensioner pulley in the center with the shaft fixing it into place, shown in Figure 8.34.

Figure 8.32: Attaching a bearing, spacer, bearing, and BS2 Piece 1 to Bracket Stage 3 In (left) with M5 Countersink Screws (right)

Figure 8.33: Attaching both pulleys on Bracket Stage 3 In and BS2 Piece 1

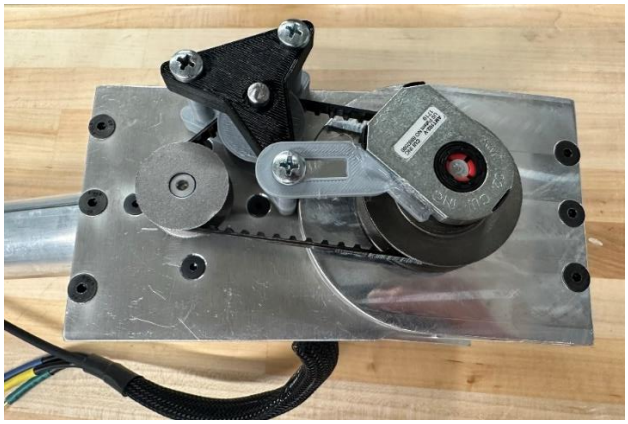


Figure 8.34: Attaching the tensioner and belt

Next, start assembling the stage 1 of the hip, which can be started separate of the rest of the assembly up to this point. Attach the motor and gearbox to the BS1 in piece, using M4

countersink screws for the motor and 8-32 screws for the gearbox, with the PLA screw sleeves between the gearbox and the plate, as shown in Figure 8.35. Next, attach the BS1 Out Plate to the other side of the motor and gearbox, ensuring the large PLA spacer is attached to the gearbox before the plate, then slide the plate onto the shaft, as shown in Figure 8.36.

Figure 8.35: Attaching a motor and gearbox to Bracket Stage 1 In

Figure 8.36: Attaching Bracket Stage 1 Out to the gearbox

With the hip stage 1 assembled, the next part of assembly must be done with the other sections of the robot as well. On the rest of the robot, attach BS2 Piece 2 and Piece 1 with three M5 countersink screws, then slide the assembled hip stage 1 into place, the gearbox shaft going through the hole, as shown in Figure 8.37. Slide the output shaft connector onto the shaft, and then screw into place with four M5 button head screws, shown in Figure 8.38.

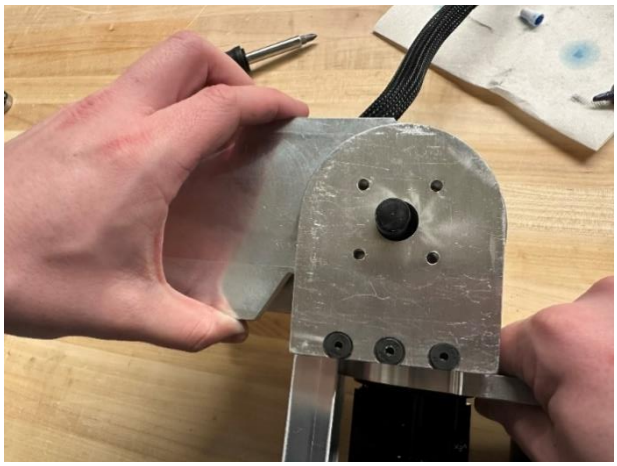


Figure 8.37: Attaching BS2 Piece 1 to BS2 Piece 2, with Bracket Stage 1 Out to BS2 with M5 Countersink Screws

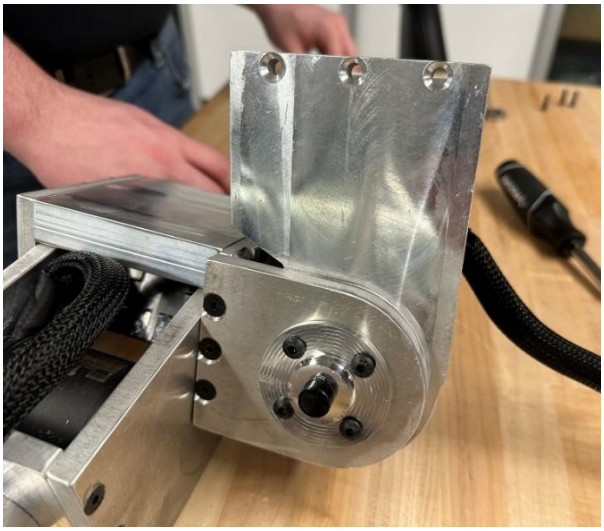


Figure 8.38: Attaching a Knee Output Shaft Connector to BS2 Piece 1 with M5 Button head Screws

Then on the other side, fix the BS2 Piece 2 to Piece 3, with stage 1 in between the bracket, the gearbox input shaft should be in the hole and attach with three M5 countersink screws, as shown in Figure 8.39. Slide on the PLA shaft collar on the gearbox and the motor shaft collar, then slide on the spacer and then the pulleys (large pulley on the gearbox and small pulley on the motor), securing with the set screws. Thread on the timing belt and then slide the tensioner into place. Once the timing belt is around all the pulleys, screw the tensioner into place with the three long 8-32s, shown in Figure 8.40, with the encoder holder in place.

Figure 8.39: Attaching BS2 Piece 2 to BS2 Piece 2 and on Bracket Stage 1 In with M5 Countersink Screws

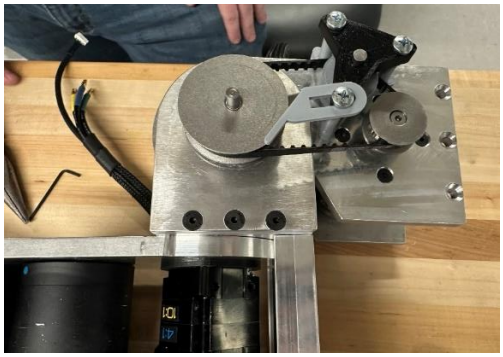


Figure 8.40: Attaching the pulleys and tensioner

Next, set up the top piece of the hip, mounting the Hip Yaw Bottom to the stage one top piece with M5 button screws, as shown in Figure 8.41. Then, insert the large bearings into the pelvis, as shown in Figure 8.42. Next, with the Bottom Hip Yaw on one side, and on the other the top hip yaw, these should line up as shown in Figure 8.43. Attach both pieces using four M5 button screws, and should twist easily, this is shown in Figure 8.44.

Figure 8.41: Attaching Bracket Stage 1 Top and Hip Yaw Mount Bottom with M5 Button head screws

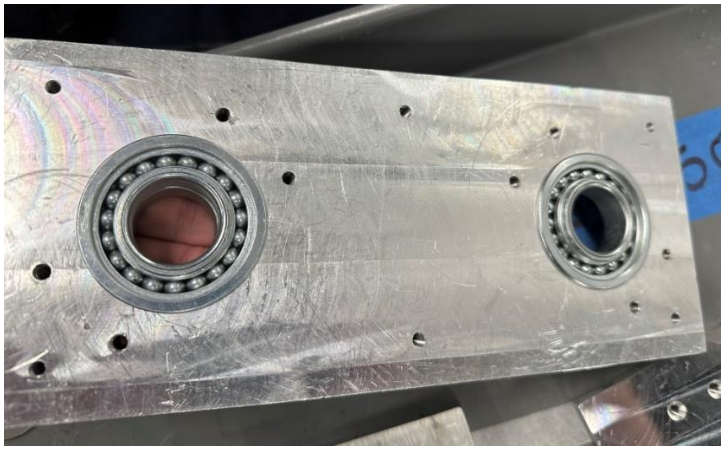


Figure 8.42: Putting Bearings in both sides of the Pelvis

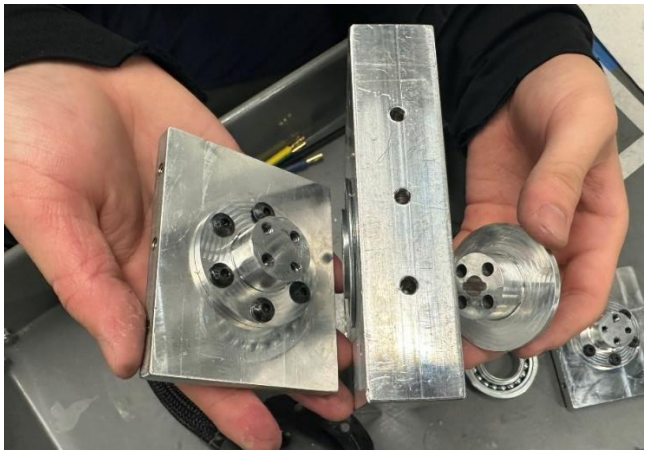


Figure 8.43: Putting Hip Yaw Mount Top and Bottom into the Pelvis

Figure 8.44: Screwing the Hip Yaw Mount Top/Bottom together with M5 Button Head Screws

The next step is to attach the top motors, the 140 motors, onto the PLA motor mount using four M4 countersink screws, shown below in Figure 8.45. Once the motors are attached to the mount, insert the key into the motor shaft and insert from the Top Yaw Mount, sliding stiffly into place, then screw the PLA mount into place using four M5 button screws to the pelvis, as shown in Figure 8.46. When twisting the motor, the Bottom Yaw Mount should also twist, but not the pelvis.

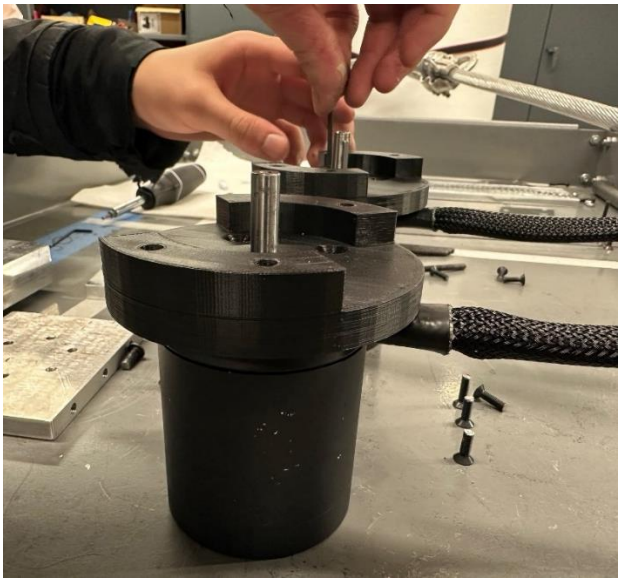


Figure 8.45: Attaching Yaw Motor Mount to a motor

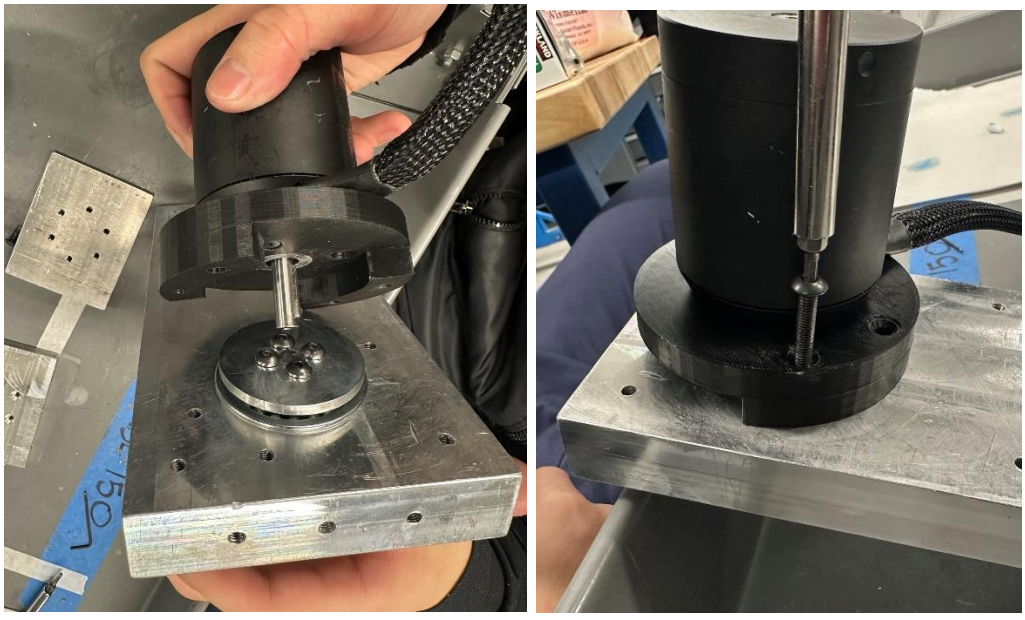


Figure 8.46: Putting Motor onto the Pelvis (Left) using M5 Button head Screws (right)

8.4 Test Rig Mount Assembly

The next stage is to assemble the test rig mount, which are vertical attachments to the pelvis on either side. Both sides are mounted using M5 button head screws, as shown in Figure 8.47. Then insert the rig shaft into the holes at the top, as shown in Figure 8.48, then secure the rod in place by attaching the screwing plates to the outsides with six M5 button head screws, as shown in Figure 8.49.

Figure 8.47: Attaching Test Rig Side plates onto the pelvis (left) using M5 Button Head Screws (right)

Figure 8.48: Putting Steel Rod through the Test Rig Side Plates

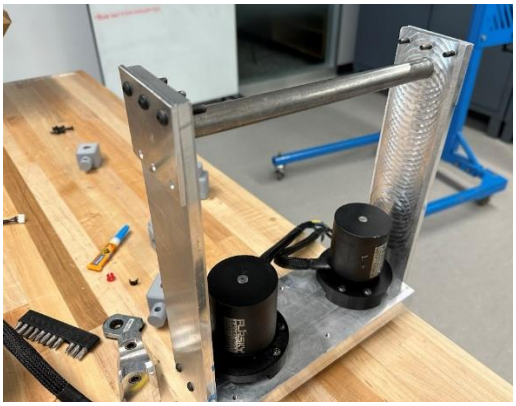


Figure 8.49: Screwing Plates onto the Test Rig Side Plates

Once the rod is secured, attach the Hip Stage 1 Top Plate to the rest of stage 1 underneath the pelvis. This can be achieved easier if the robot is set up on the test rig, and another person is holding the leg. Attach the Top Plate to the sides of Hip Stage 1, using six countersink screws, shown below in Figure 8.50. Once this process is repeated for the other leg, the robot is then complete.

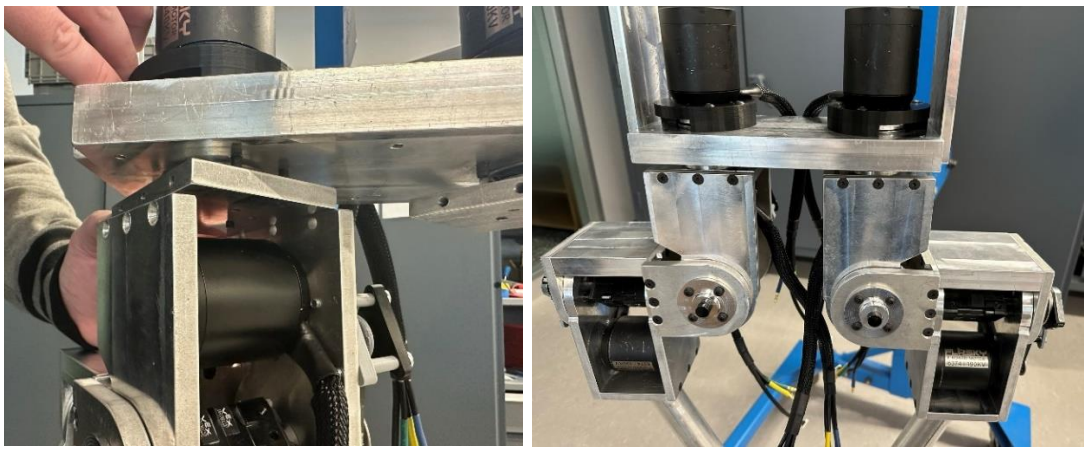


Figure 8.50: Putting the Leg Assembly onto Bracket Stage 1 Top (left) with M5 Countersink Screws (right)

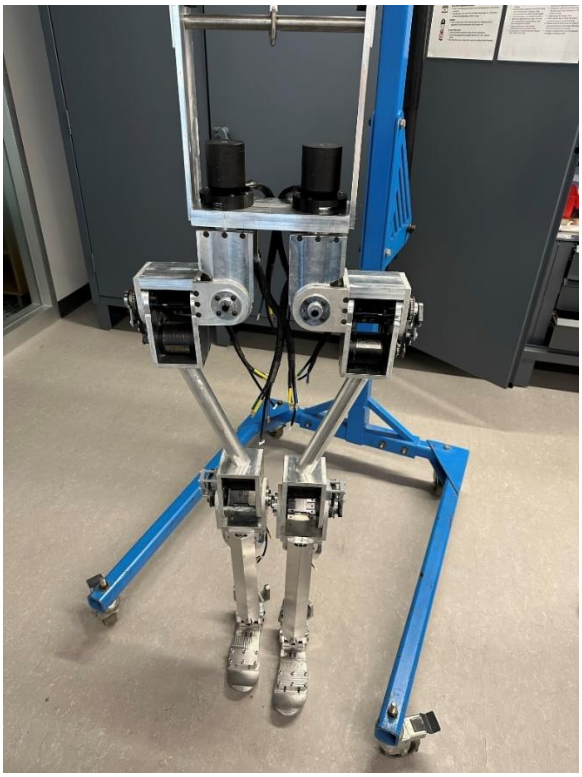


Figure 8.51: Repeat Process for other leg

9.0 Sensor Integration on HURON

9.1 Mounting Sensors Onto HURON

The FSR system was mounted on the robot using custom made stoppers which isolate most ground contact forces onto the FSRs. These stoppers were also manufactured to match the diameter of the FSR so that full contact could be achieved.

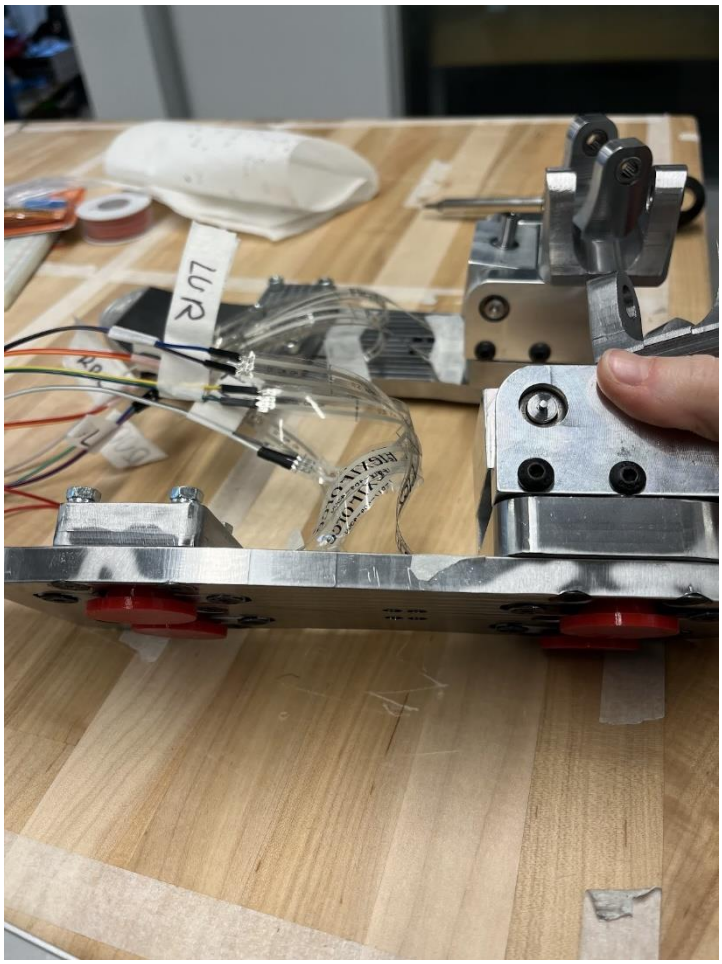


Figure 9.1: FSR Stoppers. Custom made stoppers isolate ground forces onto each FSR.

The design of the foot included four slits into which the FSRs could be inserted. At the bottom of the feet, there are stoppers that isolate the forces to the FSRs. To ensure the FSRs stay in place while in motion, these stoppers are screwed in. While the rest of the FSR was taped onto the foot to ensure that no pulling would move the FSR. Each foot also has its own analog circuit soldered to a perf board. Each perf board is taped to the top on the feet where the toes would be, as shown in the figure below.

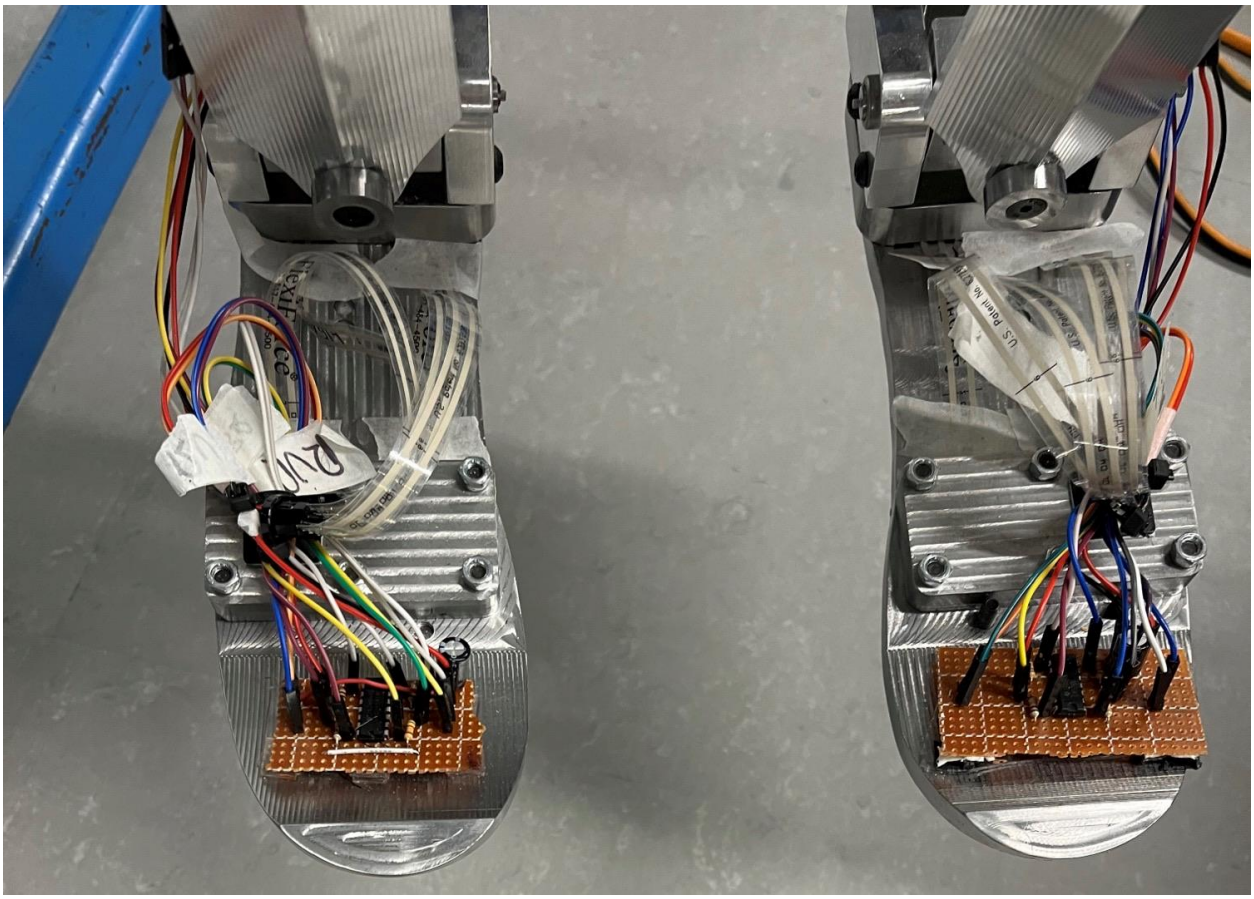


Figure 9.2: Perf Boards Mounted onto Feet

From there, the FSRS are connected to the inputs of the perf boards using male and female connected wires. With the connections of the FSR complete, the next step will be connecting the perf board to the Arduino, this is done by connecting the male output and power wires to the 6-conductor encoder cables (1 for voltage input, 1 for ground, and 4 for the FSR outputs). The cables run up, along the body connected to the main bunch of wires using zip ties, and end connected to the inputs of the Arduino board (1 wire that splits for voltage input, 2 for ground, and 8 for the analog inputs) as shown in the figure below. All of this is done so that the analog data presented from the perf boards can be processed and turned into digital data for the controls team to use.

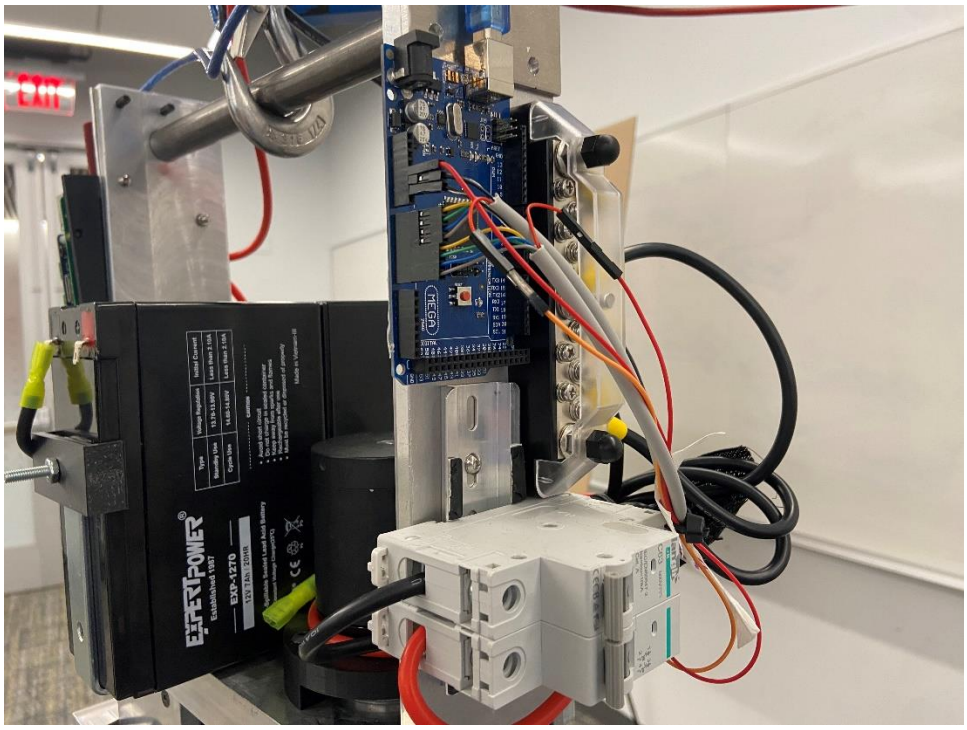


Figure 9.3: FSR Wire Connection to Arduino Mega

9.2 Verification of Voltage to Force Conversion

After the sensor system was mounted to the robot, the team verified that the conversion between voltage to force was correct. This was important because when making reactive decisions, the weight on the FSRs are taken into account. Therefore, if the weight readings are off the distance the robot needs to move to regain stability would also be off. Additionally, since the initial test set up was not the same exact environment as the robot is in due to different materials it was very likely that the conversion was slightly off. Therefore, the team retested the voltage to force conversion in the same environment the robot is in. The test set up is shown in the figure below.

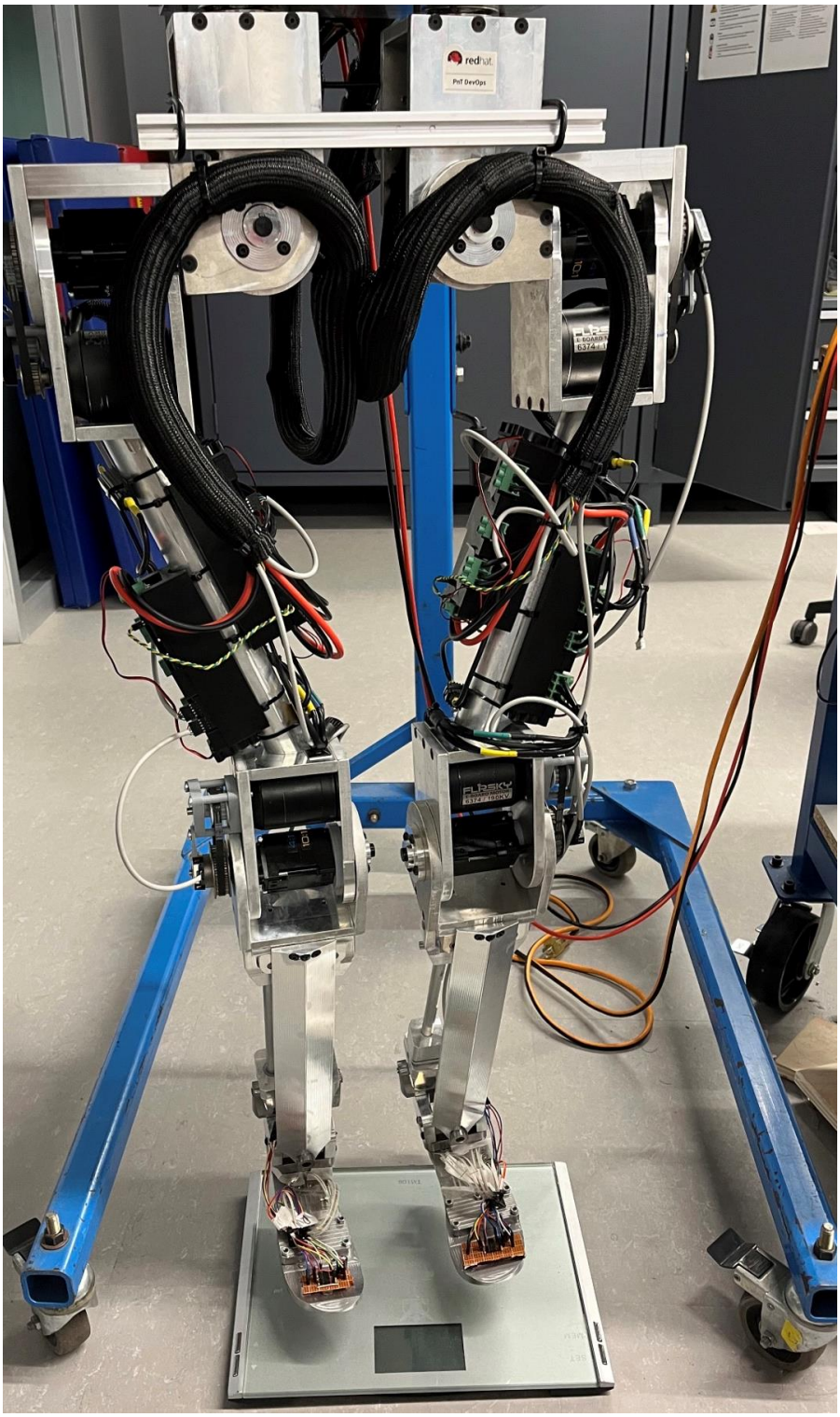


Figure 9.4: Test Set Up for Verifying the Voltage to Force Conversion

To ensure accuracy of the voltage to force conversion, the sensor system was kept on the robot the same way it will be used when the robot is standing or walking. To create a relationship between the voltage to force, a human scale was used and the following procedure was followed:

- Calibrate the scale
- Lower robot onto scale using the engine lift (as you can see in the figure above the robot scale was directly below the robot and the robot is levitating/ being held by the engine lift)
- Record the weight on the scale
- Record the voltage reading on all eight FSRs
    a. Record 5-8 sets of readings
- Repeat steps 1-4 while lowering the robot more to put more weight onto the scale

After gathering a significant amount of data with weight readings of the robot from 0 to 96lbs the team began to analyze the results. The first step in analyzing was to average the voltage readings for each FSR from the 5-8 sets recorded. The team decided to collect 5-8 steps because often times the readings jump around due to noise and the robot not being completely stable. Now that each FSR had an average voltage reading corresponding to a weight in pounds, the proportion of the weight that each FSR endured at each weight was determined. With this proportion and the total weight that was recorded from the scale the amount voltage on each FSR could be mapped to a force reading in pounds. This was then plotted with voltage on the x-axis and weight on the y-axis and a linear regression was applied to create an accurate voltage to force conversion. This graph is shown in the figure below.

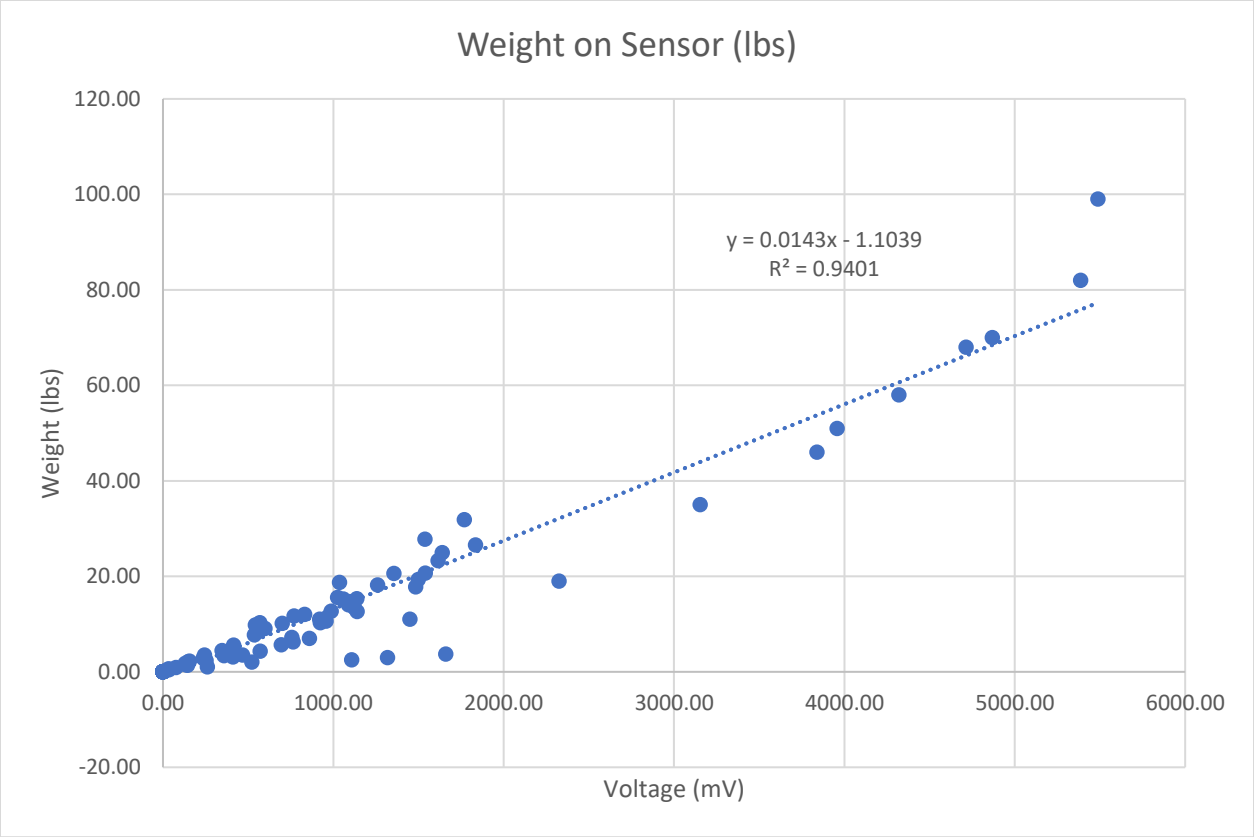


Figure 9.5 Final Voltage to Force Conversion

At this point, the team verified both the theory for determining the stability of the robot, angle it is leaning, and accuracy of the force readings. Additionally, the system was mounted onto the robot. From here the team was able to move onto creating a reactive stability recovery system.

9.3 Stability Recovery and Reactive Balancing

The FFSM system developed determines how stable the robot is and the angle at which it is leaning. A decision matrix was then developed using the stability, angle of lean, and geometric limitations of the robot. This decision matrix determines how the robot should move to maintain stability and communicate with the control system.

To return to a stable state once unbalanced, the robot must either move its CG or take a step. Normally, moving its CG would be represented by moving its ankle torque or moving its hips. However, due to money and shipping delays we were not able to use linear actuators for the

robot's calves, so the robot cannot move its ankles thus the robot must always take a step if it is unbalanced. In the future, this robot can be improved to have linear actuators, so the ankles can move, and the stability system can be improved.

Ideally, the robot would either move its CG or take a step. In this case there would be two thresholds, the stability threshold and the excessive normal force threshold. If the stability of the robot falls below the stability threshold but none of the FSRs have exceeded the excessive normal force threshold, then the robot's CG must be moved to return to a stable state. On the other hand, if the stability of the robot has fallen below the stability threshold and some of the FSRs have exceeded the excessive normal force threshold then the robot must take a step to restabilize. Since the robot has two feet with four FSRs on each foot, totaling to eight FSRs, the robot is at its most stable state when each FSR is reading ⅛ of the total weight of the robot. The minimum stability requirement is when the robot is standing on one foot with all its weight evenly distributed between the four FSRs on that foot. Using this information, the excessive normal force threshold is defined as the ¼ of the weight of the robot. If one of the FSR's readings exceeds this threshold the foot that the FSR is under may need to be displaced to maintain stability. The stability threshold is a number between 0 and 1 (calculate using Equation 3.1) and should be very close to 1, which means the robot is completely stable.

Due to the robot not being able to move its ankles, the robot is still held by an engine lift and is most stable on mats. Unfortunately, this also means the robot was not able to achieve a very high stable state. Currently, the stability threshold is 0.7. This number was determined from finding a high stability the robot could easily achieve without being pushed. The current functionality calculates the stability of the robot and compares it to the stability threshold to determine if the robot must take a step to restabilize. When determining the distance to move the foot, the FSRs above the excessive normal force threshold are used. In the future, this should be updated to also decide if the robot must take a step or only needs to move the CG accordingly.

In the case that the robot needs to take a step to restabilize, there are three subsequent calculations needed including which leg needs to move, the distance at which the leg needs to move, and the angle at which the leg needs to move.

If the robot is unstable, the first key piece of information is determining which leg must move to regain a stable state. This decision is based on the angle the robot is leaning. If the robot

is leaning towards the right, $0 \leq angle \leq 180$, the robot must take a step with its left foot. If the robot is leaning towards the left, $180 < angle \leq 360$, the robot must take a step with its right foot. In both cases, the robot would take a step in the direction it is leaning. This is based off human reactions when regaining balance with a step. In the future, when moving the CG can be used to regain balance, the robot would push against the angle it is leaning to regain balance through moving its ankles, knees, and hips.

After determining the leg that needs to move and at what angle, the distance at which the leg needs to move is then required. The distance that the leg needs to move when restabilizing is the reaction magnitude of the foot. To calculate this, the maximum magnitude in the direction of the theta with respect to each leg first needs to be defined. This is represented by D_{maxi}, where i is either left or right, this is determined by the geometry of the robot or how far each leg can move. The figure below uses a top-down view to describe the meaning of D_{max}. Since the robot has two legs, it has a D_{max} for the right leg, D_{max_right}, and a D_{max} for the left leg, D_{max_left}, representing the maximum distance each leg can move in the direction of the calculated angle theta. Through experimentation and including a safety factor, the D_{max_right} and D_{max_right} were both determined to be 150 cm.

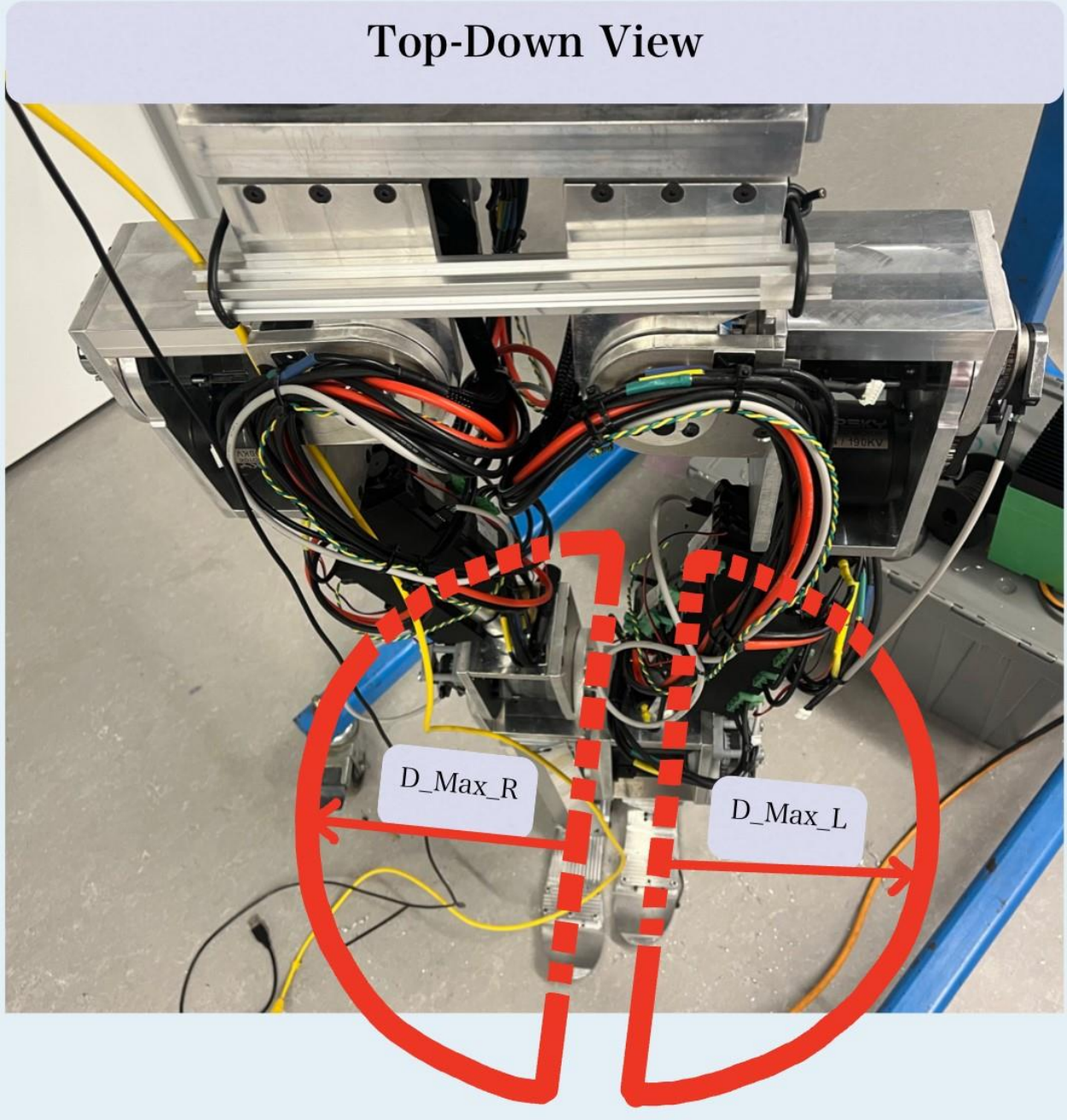


Figure 9.6 FSRs Locations On Robot Feet.

The direction indicated by theta, which we described in section 16.0, is the direction the *i*th foot should move away from the CG to enhance system stability. The reaction magnitude of the foot is given by:

$$D_i = R_i \cdot D_{\max i} \tag{9.1}$$

Where:

$$R_i = \frac{1}{k}\sum_{i=1}^{k} r_n, \text{ n = 1, …, k} \quad (9.2)$$

In Equation 9.1, i represents the left and right feet. Thus, Equation 9.1 calculates the magnitude to move the left foot and the right foot. But first, R_i and D_{maxi} must be calculated. The previous paragraph and Figure 9.6 described how to calculate D_{maxi}, which is based on the geometric limitations on the robot. In Equation 9.2, k is the number of FSRs that are above the excessive normal force threshold. For each FSR that is above the excessive normal force, r_n must be calculated. The Equation for r is shown below.

$$r = \frac{f - f_{min}}{f_{max} - f_{min}} \quad (9.3)$$

In Equation 9.3, f is the force reading that the FSR above the excessive normal force is reading, f_{min} is the minimum force the of the eight FSRs that robot is experiencing, and f_{max} is the maximum force of the eight FSRs that the robot is experiencing. Once calculating r for each FSR that is above the excessive normal force threshold using Equation 9.3, finding the average of r for each foot using Equation 9.2, the distance to move each foot can be determined using Equation 9.1.

As stated above, with the current functionality of the robot, if the robot is unstable, it must take a step to restabilize. In the future, this should be updated to also move the CG if the robot is unstable. In the case that the CG can be moved to regain balance, Equation 9.1 – 9.3 can still be used but will require a different set of actions when moving the joints. Although the current functionality only takes a step when regaining balance, Figures 9.7 and 9.8 summarize how to restabilize when taking a step and when moving the CG. In the future, these methods can be utilized.

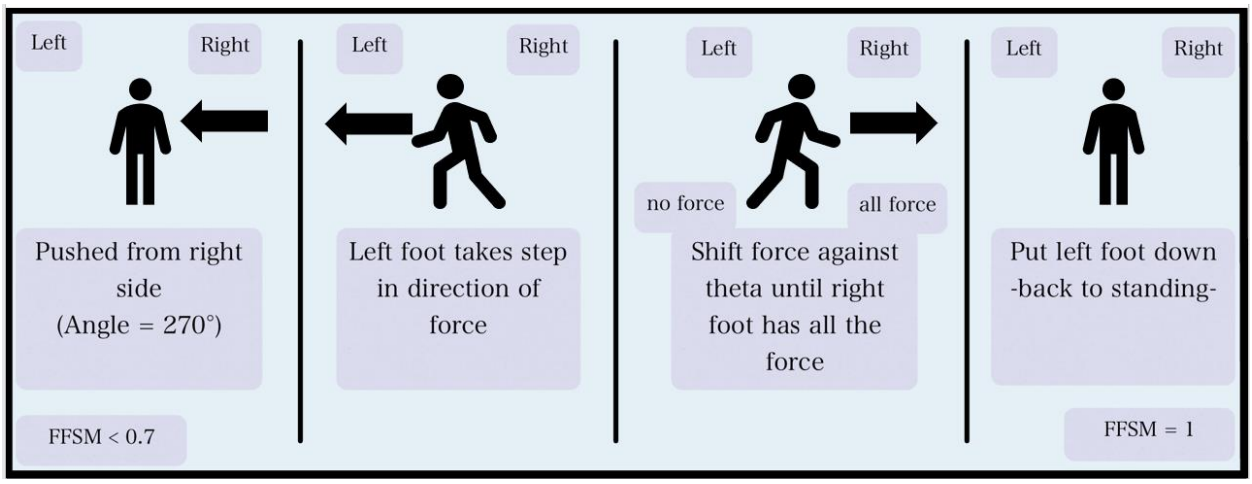


Figure 9.7: Restabilizing by Taking a Step

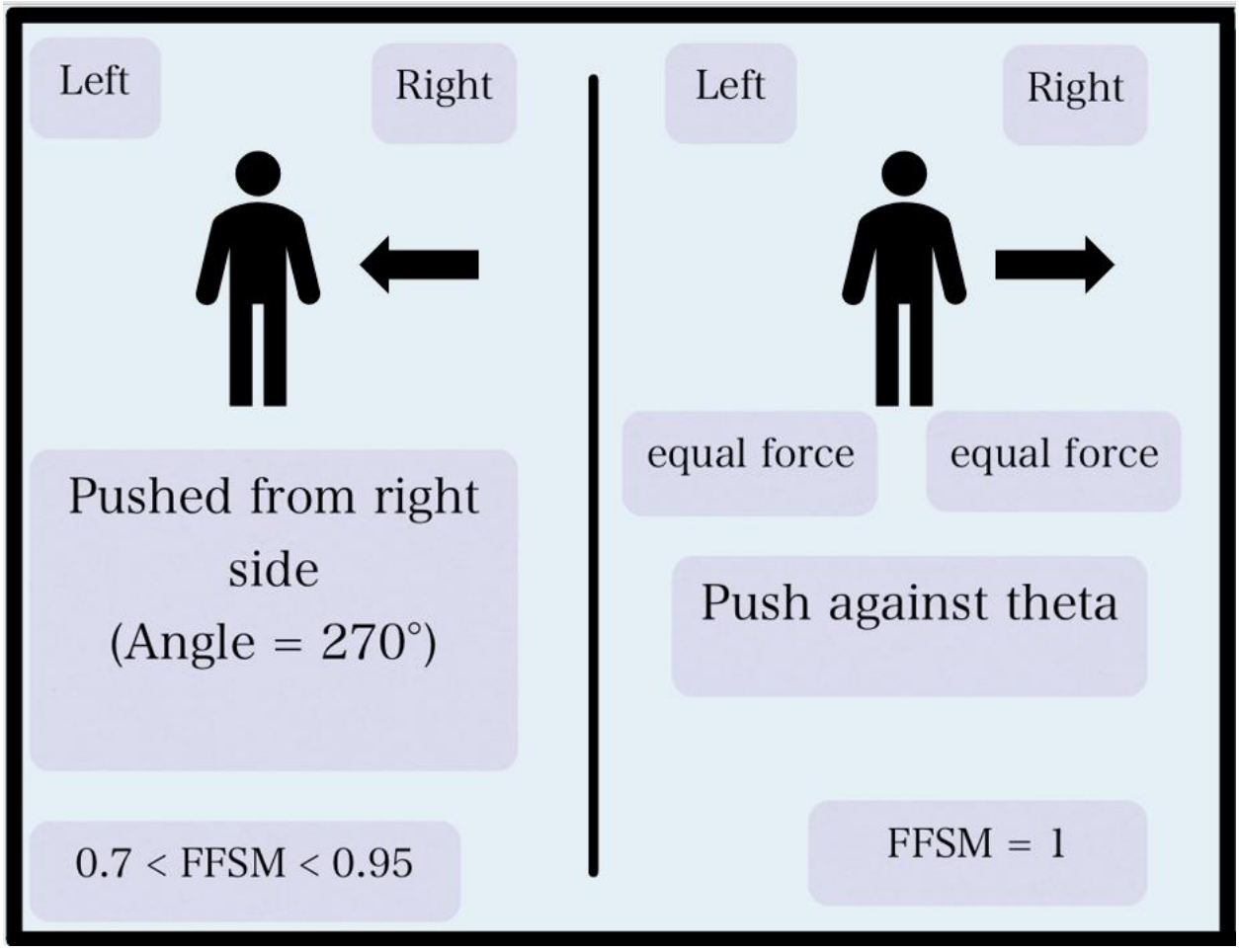


Figure 9.8: Restabilizing by Moving CG

Figure 9.7 shows how the robot should react if it is pushed from the right side (reading an angle of 270) and FFSM is reading less than 0.7. As of right now the limit of 0.7 is an arbitrary

number and a more accurate measurement will be made through experimentation. However, if the robot is pushed and is unstable enough where it must take a step, the foot experiencing more force should move in the direction of the force. While the foot is in the air, the other foot should shift against the direction of the force until the foot is experiencing the whole force of the robot. Finally, the foot that is in the air should be put down and the robot will be back to a stable standing state.

Figure 9.8 shows how the robot should react if it is pushed from the right and FFSM is reading between 0.7 and 0.95. In this case the robot should push against the force in the opposite direction of the force until both feet are experiencing the same amount of weight and in a stable standing state.

Overall, the entire sensor system is summarized in the diagram below. This is a constant loop, but the system starts in the top left corner at read foot force sensors. Once the decision for moving the robot is sent to the control system, this loop repeats.

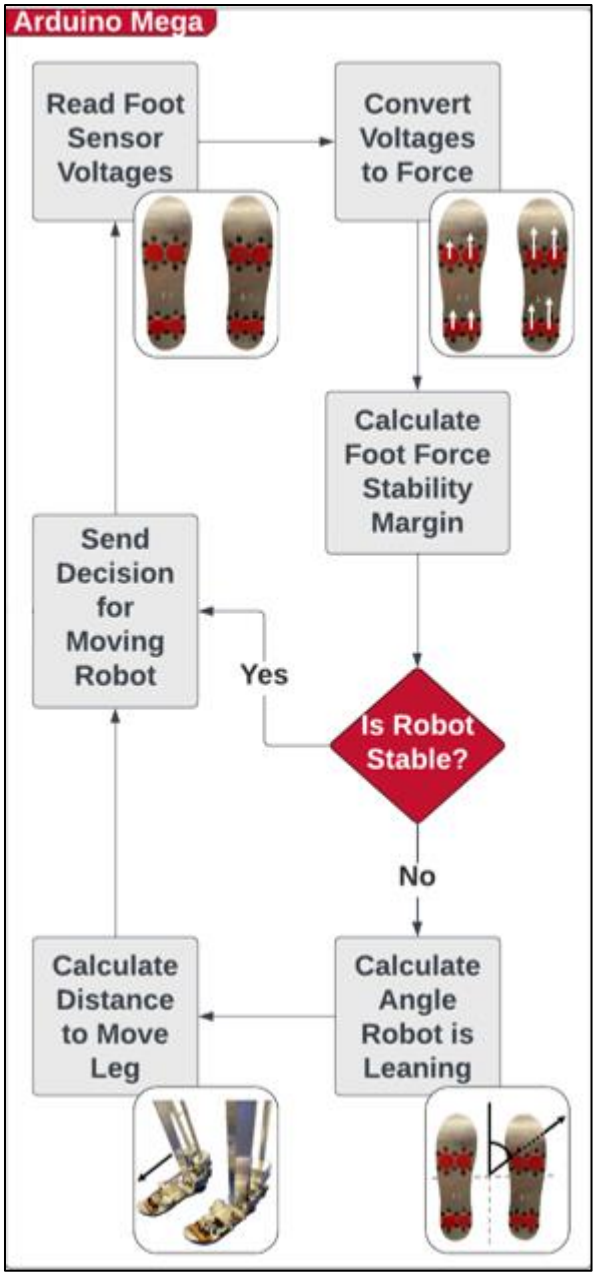


Figure 9.9: Sensor System Process

The decision that is sent to controls contains a number between 0 and 3 which represents how the robot should move, an angle representing the angle at which the robot should move, and the distance that the robot needs to move. The decision key for how the robot should move is summarized in the table below and the Arduino program for reactive balancing can be found in 24.3.1 Appendix C: Reactive Balancing.

Table 9.1 Decision Key for Control

Decision Number	Description
0	One or more feet are off the ground
1	Robot is stable
2	Take a step with left leg
3	Take a step with right leg

10.0 Robot Control Design and Simulation

10.1 Control System

Below in Figure is the map of the system hardware that will control the robot. In the middle of the Diagram is a Raspberry Pi 3, which will be the central controller of the robot that will interface with the other peripherals. The Raspberry Pi OS is Ubuntu Server 20.04 which will allow us to run ROS Noetic. The Raspberry Pi is setup to be controlled over ethernet, so we do not need to plug in a monitor to the robot to work with the Raspberry Pi.

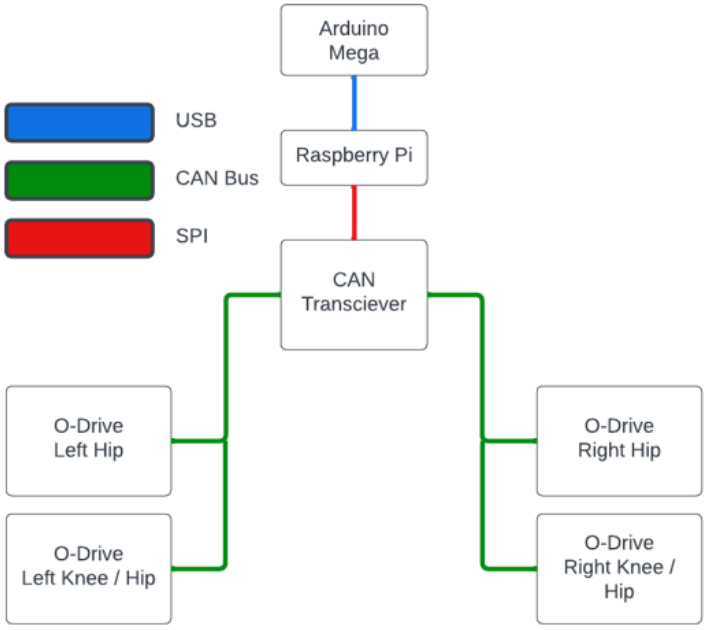


Figure 10.1: Control Hardware System Overview

There are several different peripherals that need to interface with the Raspberry Pi. The Microcontroller that interprets force sensor data from the feet will communicate with the Raspberry Pi over USB. The O-Drive motor controllers will need to be placed at several points throughout the robot's legs, so it is important that we use a communication bus to control these. The O-Drives enable control over a CAN bus, however the Raspberry Pi is not natively compatible with CAN communication. We have elected to use the Waveshare 2 Channel CAN transceiver to bridge the connection between the O-Drives and the Raspberry Pi. We chose this because it is designed to be used with the Raspberry Pi, and has 2 channels, allowing us to use 1 CAN bus for each leg to simplify our wiring.

10.2 Power Supply

There are several factors that go into choosing a voltage for the supply. The Flipsky BLDC motors that we have elected to use are compatible with many voltage levels. The O-Drive motor controllers are limited to 56V, which gives us plenty of different voltages to work with. The Linear actuators that will actuate the ankles are only available in 12V and 24V versions, so one of these voltages should be used. Higher voltages would allow us to actuate the joints using BLDC motors faster, so we are opting for a 24V source since this is the highest voltage source we can use, while remaining compatible with the linear actuators. With the Lead Acid batteries, we will need to use two 12V packages to achieve the 24V supply voltage. For the Li-Po battery, the closest commonly available voltage is 22.2V nominal voltage from a 6S package.

Lead Acid batteries have a constant output voltage that sharply decreases at the end of its discharge range. The plot below shows the characteristics for the lead-acid battery we are considering using. When we are using them, we will discharge them through most of the linear region and recharge them once the voltage starts to rapidly decrease.

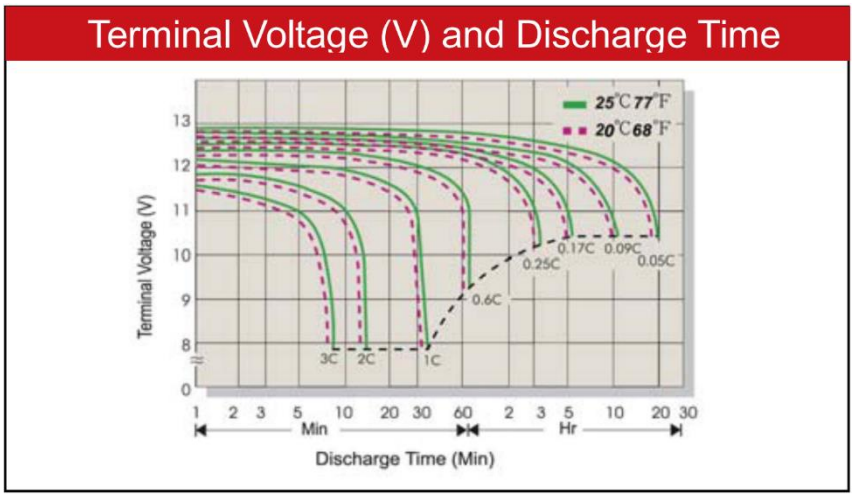


Figure 10.2: Terminal Voltage vs Discharge rate for lead-acid batteries

Li-Po batteries have a large range that they discharge over. Each cell, with a 3.7V nomial voltage can range from 4.2V at maximum charge to 3.4V at the minimum allowable voltage. For the 6S battery we are planning on using, this equates to a voltage range of 25.2V at maximum charge to 20.4V at minimum charge. With Li-Po batteries, discharging the battery to below 3.0V can permanently damage the battery. Below is the discharge curve for a typical 3.7V Li-Po cell.

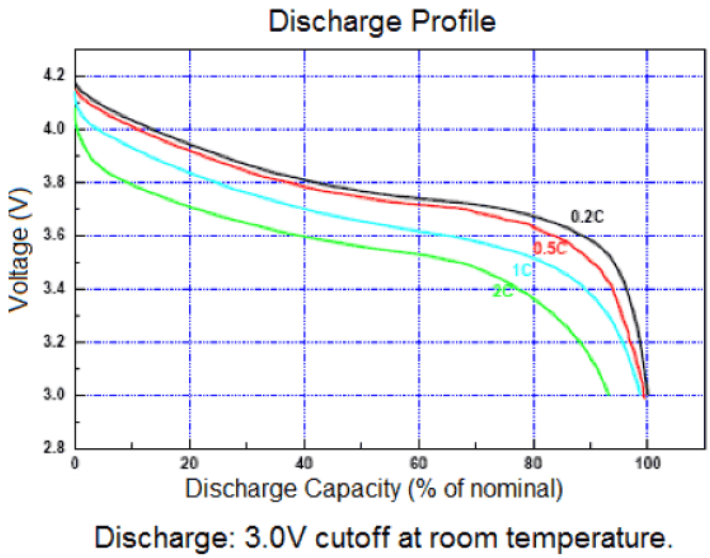


Figure 10.3: Terminal Voltage vs Discharge capacity for 3.7V Nominal Li-Po Cell

Size is another factor that needs to be considered when selecting which battery to use. The 7AH lead-acid batteries that are commonly available measure 151mm x 65mm x 94mm. We would need two of these in series to reach the voltage we need, so this factor must be taken into account when looking at size. The largest capacity Li-Po battery that we have considered, a 30AH 6s Li-Po from Gens Tattu, measures 218mm x 120.5mm x 60mm. Both of these fit within the specified torso dimensions with a semi-minor axis of 95mm and a semi major axis of 150mm.

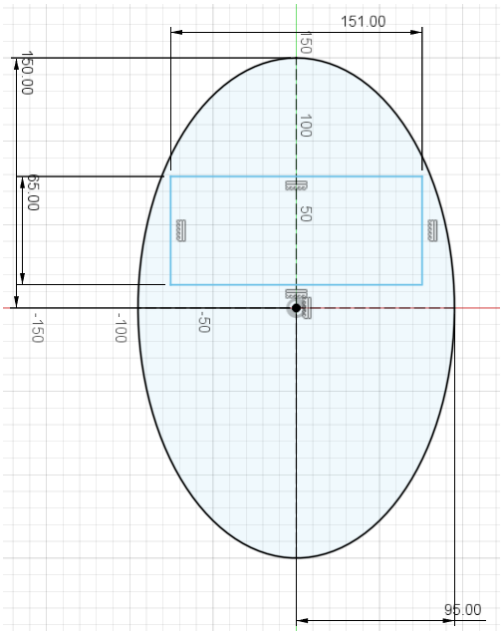


Figure 10.4: 7 AH Lead Acid battery dimensions inside given torso size

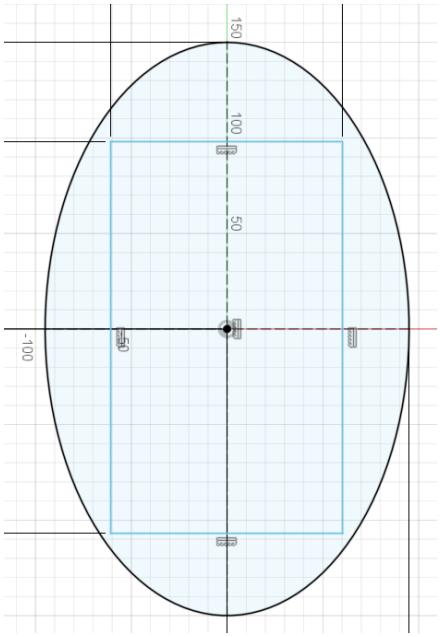


Figure 10.5: 30 AH Li-Po battery dimensions inside given torso size

The capacity of the batteries determines the amount of time that the robot will be able to run for. Without knowing how much current each joint will be drawing, we will estimate the power consumed by the motors using the mechanical energy of the motor. This can be found using the equation:

$$P_{mech} = \tau[Nm] \times \omega[rad/s] \quad (10.1)$$

Since the efficiency of the motor and joints is unknown, we will multiply the power consumption by a safety factor of 1.5 to estimate the power consumed considering losses. Using the peak torque and speed of the motors, we can find the peak power consumed by each BLDC motor, and then multiply it by 8 for each of the BLDC motors used. The linear actuators have a given power, which is multiplied by 4 for the 4 linear actuators in the ankle. These powers are summed to give the peak possible power consumption for the robot.

The capacity of batteries is given in Amp-Hours, however we need to convert this to Watt-Hours to calculate the run time based off of our calculated power consumption. The capacity can be converted using the following equation:

$$Capacity\ [Wh] = Capacity[Ah] \times V \quad (10.2)$$

The runtime, in minutes, can then be calculated by the equation:

$$Runtime = \frac{Capacity[Wh] \times 60}{P_{mech\ total}} \quad (10.3)$$

This returns the runtime of the robot if each of the joints were to be run at maximum power 100% of the time. This is not a realistic estimate since the robot will not be running at maximum power 100% of the time. To account for this discrepancy, the run-time for each battery was calculated at several different values of power, each a different percentage of the peak power draw. The table below shows the runtimes for several different Li-Po battery capacities, and the 7 AH Lead Acid Battery.

Table 10.1: Battery Runtime Estimate

	Runtime (minutes) 100% Power	**Runtime (minutes) 50% Power**	**Runtime (minutes) 25% Power**	**Runtime (minutes) 10% Power**
30 AH Li-Po	8.019960791	16.03992158	32.07984317	80.19960791
22 AH Li-Po	5.88130458	11.76260916	23.52521832	58.8130458
9 AH Li-Po	2.405988237	4.811976475	9.62395295	24.05988237
7 AH Lead Acid	1.871324185	3.742648369	7.485296739	18.71324185

There is not a requirement for minimum runtime, but the longer the runtime, the easier it will be to test and use the robot. The choice between which battery we will use from the table above comes down to funding. Ideally, we would use the largest Li-Po battery available, but these are the most expensive batteries. For example, the 30AH Gens Tattu 6S Li-Po is listed for $533. The best budget option is the 7AH Lead Acid battery which can be found online for $20-$30 online depending on the manufacturer. Fortunately, these batteries should be interchangeable, since both would use a charger external to the robot instead of battery management circuitry built into the robot.

These batteries will power the motors and actuators on the robot as is, but the control circuitry will need a 5V source for power. We will need a 24V to 5V DC-DC converter which are commonly available from many different manufacturers. A 63A DC Circuit breaker will serve as the main power switch, and protection between the batteries and the robot.

10.3 Simulation in Gazebo

In order to test any control algorithms without a physical system, we opted to attempt to load our Solidworks model into Gazebo, a simulation tool with realistic physics. This posed several challenges for us. Firstly, Solidworks does not support exporting to .URDF files, which are the necessary files for Gazebo. After attempting to write our own .URDF files and a file

converter system, we stumbled across an online file converter tool, promoted by ROS documentation. The converter existed in a github repository, that after downloaded, would add an add-in to export to .URDF within Solidworks. However, this converter did not work as well as expected. The converter was optimized for Solidworks 2021, which is not ideal as our Solidworks model was developed in Solidworks 2022, and the files are not backwards compatible. Due to this, we have limited functionality in the converter tool. For instance, there is an option when selecting joint locations for the converter to automatically generate the coordinates. When we choose this option, however, the converter throws several errors, and fails to generate a .URDF file for us to use. This meant that we needed to calculate these coordinates by hand. This unfortunately became a little more intensive than expected, as the global origin for the Solidworks model is located behind and to the left of the model, rather than properly at the base plate itself. After doing some geometry to find the coordinates with respect to the base plate, we needed to edit the generated .URDF file to add the new coordinates, as well as redefine joint characteristics. By doing so, we were able to load into Gazebo a model with bends at the knees. We intend to continue experimentation with this, and ideally be able to control each joint within Gazebo itself.

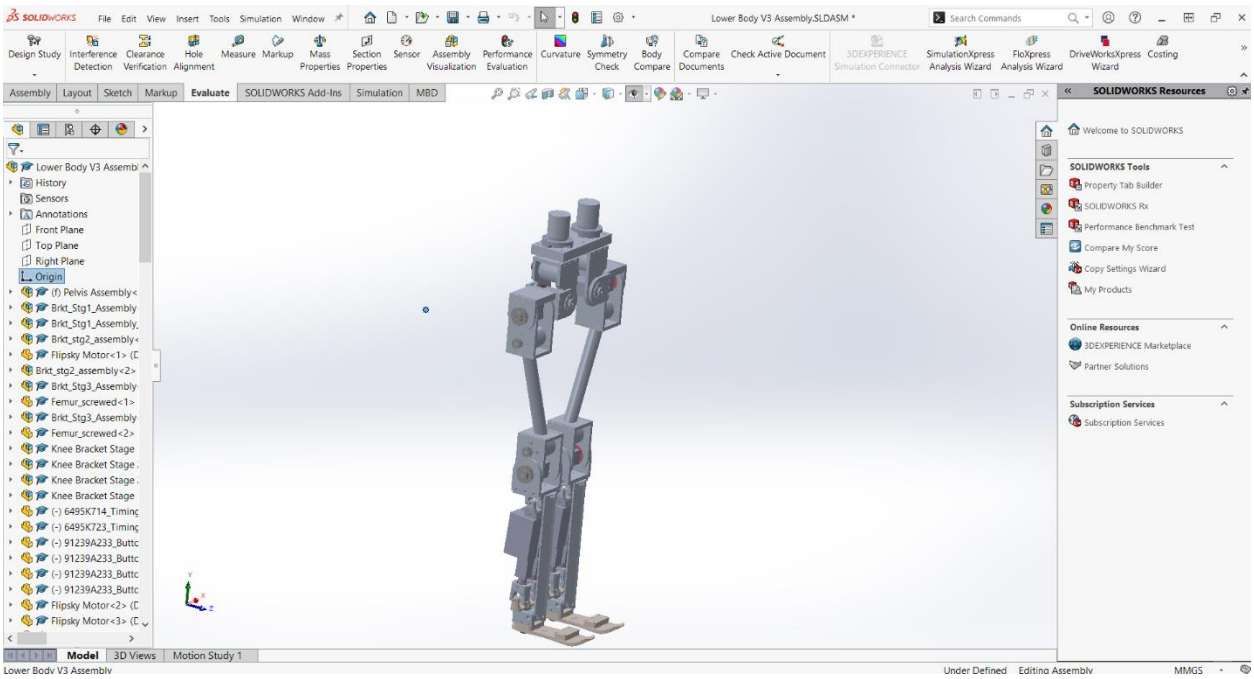


Figure 10.6: Solidworks Model with origin behind model

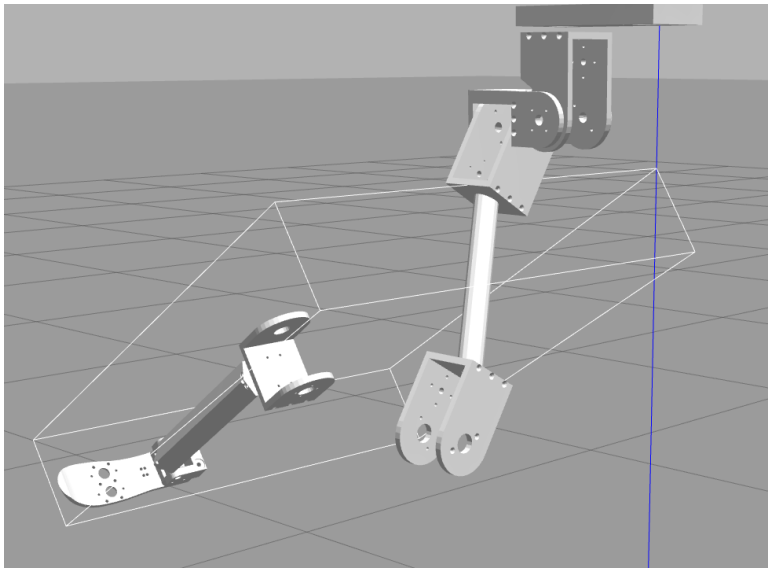


Figure 10.7: Joint with wrong joint origin in Gazebo

11.0 Motor Integration and Testing

As the design team was wrapping up their assembly, we worked with them to ensure that the transition into programming the robot was successful. The design team ran into some issues with belt sizing and tensioners. We worked with them to test the motors with the tensioners, and resolve issues with improper tension on the belts, as well as concentricity of the pulleys. As we began working on moving the legs of the robot, we worked with them through minor design flaws such as loose shafts, and interference that prevented the limbs from moving freely.

In order to test the torque capabilities of the motors, as well as determining how to control multiple motors simultaneously, we developed a testing rig. This test setup consisted of a 3D printed winch and hubs designed for different motor axle sizes in case the setup was used for other motor tests. The minimum required torque, given by the design team, was 4 Nm. The winch itself had a diameter of 19 cm which was calculated to be the general size where a common weight (~20lbs) could be attached to attain a goal torque of 7 Nm. This goal stems from the max output torque of the system, divided by the torque ratio provided by the pulleys and gearboxes. The winch was connected to the motor by compression and a slot for the motor key. Attached to the winch were thick, Timberland style shoelaces. This way we would be able to test the load and torque limits of the motors during operation, without fault from the laces. Objects of varying weights and standard weights were added to determine the motor's torque abilities, and the motor was run towards incrementally further positions until the weight was suspended into the air. If the weight was stably suspended with no issues from the ODrive, the motor was considered capable of providing the given torque. Testing proved that the motors were able to provide a maximum of just over 6 Nm of torque under load, which is above the 4 Nm of torque we needed at the motor, proving that the Flipsky 190 kV motors would work for our application with a safety net.

Figure 11.1: Motor Torque Test

During extensive testing, the insulation around the wire between the motor and ODrive began to smoke, due to excessive heating of the wire. In addition, the ODrive MOSFET temperatures were around 100 C, which was the upper limit that these chips can handle. When the motor was stalled by the weight it was lifting, it was drawing approximately 60 Amps from the ODrive. The wire used in the test setup was not suitable for the high currents involved. To address these issues, small fans were added to direct air at the ODrive and heat sinks were installed. In addition, larger wire gauges with silicone insulation were used when wiring the robot to ensure that the wires did not overheat and start smoking. Overall, the testing rig was effective in determining the motor's torque abilities, despite overheating issues. The addition of fans and heat sinks helped to address the issue and allowed us to continue testing the motor's capabilities. These tests were able to help inform us of future design decisions and reassure us of the motors.

12.0 Wiring and Communication

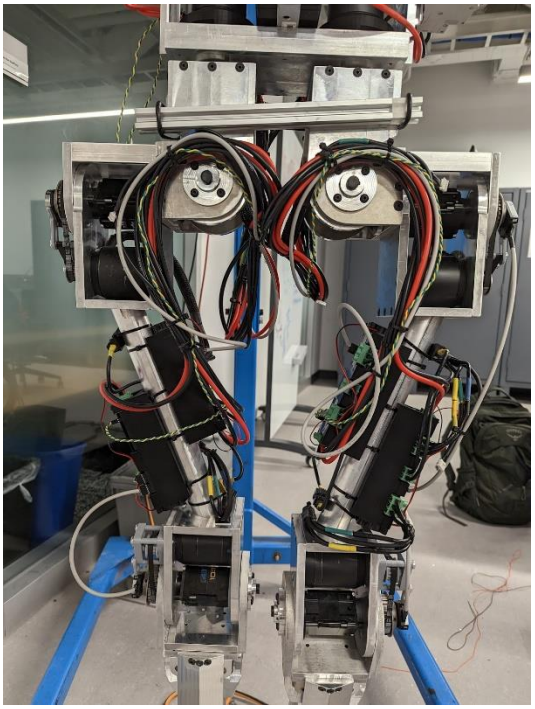


Figure 12.1: The front of HURON showing the wiring harness

There are several components that need to be connected to ensure that proper power distribution and communication occurs on the robot. The components were wired according to the wiring diagram found in the controls appendix. For power, we are using two 12V lead-acid batteries in series due to budget constraints, and the high current supply potential of these batteries. This provides the 24V necessary to power the motors. The battery is wired to the main power rails through a 63A DC circuit breaker. The circuit breaker acts as the primary power switch for the robot, and also protects against electronic faults that lead to excess current-draw from the source. The power terminal blocks were rated for 150 Amps, allowing plenty of overhead for our primary power source. 10 AWG silicone insulated wire was used for all power lines to ensure that the wires do not overheat, since there is large amounts of current going through these lines. To connect the motors to the Odrives, 12 AWG silicone insulated wire was used, since there is not quite as much power going through these as the primary power lines. Another benefit of using silicone coated wire for both of these is their flexible properties which will allow the joints to move without putting stress on the wire insulation.

Each motor requires a rotary encoder to operate. These sensors are prone to interference from the high-power motor wires that are running in close proximity to these wires. To mitigate noise, we are using 22 AWG 5 Conductor cables with a foil shield. The shield was grounded at both ends to ensure that the signal is transmitted successfully.

Our central processing unit, a Raspberry Pi 3, requires a 5V power source. To avoid having two separate voltage supplies on the robot, we are using a 7805CV voltage regulator to regulate the 24V power from the battery down to 5V. We are using a Waveshare 2 Channel CAN HAT for the Raspberry Pi. We are using 24 AWG silicone insulated wire to wire the CAN bus to the Odrives. The CAN wires were twisted together to mitigate the common mode interference that could interfere with the signal.

To test the performance of the CAN protocol for the system, two experiments were run. Setup for these experiments included assigning a unique CAN ID to each motor through the ODriveTool. The CAN ID is an identifier used to differentiate between different devices connected to the CAN bus.

The first experiment involved two motors being connected to a single ODrive using the CAN protocol. By sending commands to each motor, the team was able to verify that multiple motors could be controlled simultaneously on a singular ODrive.The second experiment involved two ODrives with a motor on each. This experiment allowed the team to verify that using multiple ODrives simultaneously caused no errors as both motors were given a command and ran with no issues.

Figure 12.2: The benchtop CAN bus test setup

Before the robot was programmed, we had to configure each of the Odrives for the motors, encoders, and other parameters that are a part of our system. This was done through the Odrive tool, which communicates with the controller over USB. We plugged in each of the motor controllers one by one and sent the commands shown in the controls appendix to the Odrive. Slight modifications, such as the CAN node IDs, so that each motor has its own ID, and motor KVs were changed depending on which motors the controller is communicating with.

For communication with the raspberry pi, we considered both USB and I2C. After considering the pros and cons of each, we opted for USB communication. With USB, we would have a higher data transfer rate, which we believed to be useful when uploading code and making frequent changes during testing. Also, we would avoid setting up an I2C protocol, which seemed appealing given our time constraints.

13.0 Code Structure

The two types of functionality we aimed to achieve with HURON were locomotion and reactive balancing. Locomotion requires that we are able to accurately control the position of the feet through the 12 actuators on the robot. For the reactive balancing, we need the data from the foot sensors, and the ability to actuate the joints. To successfully balance and move, the process looks like this:

1. Receive data from sensors implanted in the foot and analyze that data.
2. Use the analysis as an input into equations we derived as part of inverse kinematics which then resulted in an output of robot joint positions.
3. Send commands to every joint to move to those positions.

In terms of implementation, the program is written in python. How commands are sent from the Raspberry Pi to the ODrives is through CAN messages. Each CAN message has an arbitration ID, where bytes of data can be found. As there are specific CAN IDs and arbitration IDs for each motor (found online in the ODrive documentation for CAN protocols), the team created a motorCon class to handle sending and receiving messages related to each motor. These classes are instantiated in a main file where their functions are called to control the robot. Currently there are functions for setting axes state, moving the motor, and checking if motor movement is complete.

There is also a function for inverse kinematics that takes in target positions for the hip and outputs the joint values for one leg to achieve that hip position. This function is written in the main file as it's not specific to each motor. The plan for this function is to convert a task space trajectory into a joint space trajectory. Once the joint space trajectory is acquired, we can send the joint positions consecutively through the motorCon functions for each object instantiated.

```
import can
import cantools
import time

class motorCon:

    def __init__(self, canID, axisID):
        self.db = cantools.database.load_file("odrive-cansimple.dbc")
        self.bus = can.Bus(canID, bustype="socketcan")
        self.axis = axisID
        self.states = {"idle": 0x01, "calib": 0x03, "closeloop": 0x08}

        self.change_state("calib")
        print("Calibrating...")
        time.sleep(15)

        self.send_cmd('Set_Controller_Mode', {'Input_Mode':1, 'Control_Mode':3})

        self.change_state("closeloop")
        print("Entering closed loop")
        time.sleep(1)

        self.send_cmd('Set_Limits', {'Velocity_Limit':10.0, 'Current_Limit':70.0})

    def move_motor(self, pos, vel, tor):
        msg = self.db.get_message_by_name('Set_Input_Pos')
        data = msg.encode({'Input_Pos':pos, 'Vel_FF':vel, 'Torque_FF':tor})
        msg = can.Message(arbitration_id=self.axis << 5 | msg.frame_id, data=data, is_extended_id=False)
        self.bus.send(msg)

    def kill_motor(self):
        self.change_state("idle")
        print("killed motor")

    def send_cmd(self, name_of_command, input):
        msg = self.db.get_message_by_name(name_of_command)
        data = msg.encode(input)
        msg = can.Message(arbitration_id=self.axis << 5 | msg.frame_id, is_extended_id=False, data=data)
        self.bus.send(msg)

    def change_state(self, s):
        try:
            self.states[s]
        except:
            print("unknown state" + s + "inputed into change_state()")

        self.send_cmd('Set_Axis_State', {'Axis_Requested_State': self.states[s]})
```

Figure 13.1: Example code showing the motorCon class

14.0 Simulation of Inverse Kinematics

Thus, firstly, we needed the math to bridge steps 1 and 3. Ibrahim, the graduate student we are working with, did the math for the inverse kinematics of our system. His equations derive the joint angles for each joint on the leg. We use these theta values as an input into our motor commands to send every motor to its respective theta value.

We then verified these equations in MATLAB as we tried to first simulate the solution before implementing it on the robot. To start, we setup our joints, added the dh parameters, and added link lengths such that we could simulate our robot. We then set up a trajectory as we asked our simulated leg to move a certain distance in a given number of step intervals. Essentially, we asked our robot to take a step as if it were walking.

Initially we ran into some errors, where some of the joints were moving in unexpected trajectories. Once a math error in the inverse kinematics was resolved, the sequence of movements shown below were achieved in the MATLAB simulation.

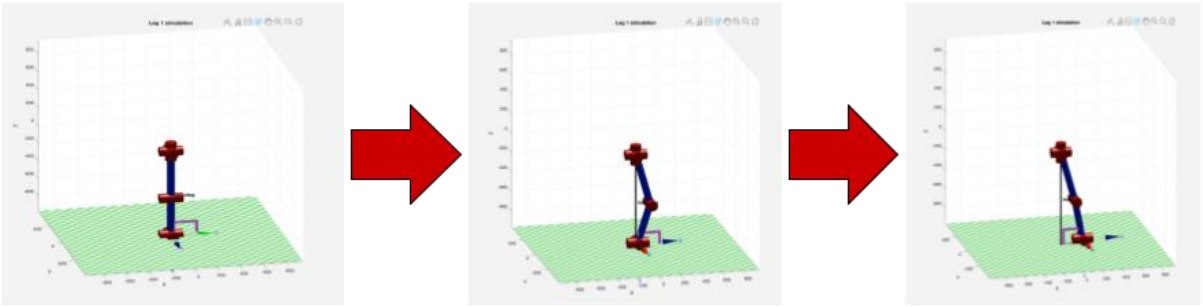


Figure 14.1: MATLAB inverse kinematics simulation

15.0 Testing on whole system: Legs move

To test the whole system, the design team requested a testing rig to hold the robot up as it walked. The design parameters were to have about 7ft of horizontal space for the robot to walk across, and to have a safety factor of 10 for holding the weight of the robot (~60kg). In order to stay within budget, the controls team designed a testing rig in CAD with plans to use wood, joist hangers, and a beam trolley with an enclosed beam track. Other alternatives were considered too, such as finding an I-beam somewhere on campus and attaching the robot onto it with an I-beam trolley. However, the team learned of a beam trolley already on campus used for an older robotics project in Washburn. After meeting with the WPI RBE Lab Manager, permission was acquired to use the setup as our testing rig for testing the walking. To transport the robot to different locations and to hold the robot upright, the team acquired an engine lift.

Figure 15.1: Preliminary test stand design

Initially, with the whole robot wired and assembled, the team tested calibrating the left knee motor. This was done through the Raspberry Pi by changing the ODrive axis state to full calibration sequence. In the code, the team sent a message through CAN with the arbitration ID found in the ODrive documentation. Each arbitration ID was different for each axis as well, so a function needed to be created such that the axis ID would be taken into account when calling with a specific arbitration ID. The arbitration IDs are in hex, and a noticeable pattern was that

arbitration IDs for the same info but for different axes were offset by a multiple of 32 (eg. axis 0 heartbeat = 0x01, axis 1 heartbeat = 0x21, etc.). Due to the encoder not being directly on the axis of the motor but rather on the gearbox input axis, the ODrive configuration needed to be modified such that the number of ticks per motor revolution was modified by half.

Once the calibration for one joint was successful, we attempted to move it through commands from the Raspberry Pi. This was done in the same fashion as setting the calibration mode in terms of CAN messaging, but with different arbitration IDs. Each axis needed to be set into Closed-Loop Control mode, and then a target position would be sent after. To begin, the team sent a very small value for position to see if the knee could handle the torques and how much the knee would move by. As things seemed stable, the team incrementally increased the position value, up to about 10 revolutions where the knee looked sufficiently raised for walking. Then the knee was set back to the 0 position to see if there was any drifting.

After testing with one joint was successful, the team attempted two joints simultaneously on the left leg (hip and knee). At first, the team calibrated each motor separately which took a lot of time (~15 seconds each consecutively). Knowing that this process would linearly increase with all the motors, the team looked into simultaneous calibration however, there were some issues. When all of the motors were calibrated at the same time, the ODrive would through an error indicating the motor phases were out of balance. This was due to the excess power going through the controller with both motors calibrating. To remedy this, we calibrated 1 motor on each controller at a time. With simultaneous calibration achieved, the team attempted to move each motor consecutively. By following the same steps as with a single joint but for each, the team incrementally moved the motors until a sufficient walking position was achieved. After, the team attempted simultaneous motor movement on one leg which ended successfully.

Final testing for the full leg assembly was to make the motors move to different positions like a path. An issue that arose with this was that there was no queue for the ODrives so sending a position in code would overwrite the previous command if the previous command wasn't complete yet. A quick band-aid fix was to add sleeps in between each command, however smooth movement with this method would require calculating exactly how much time was needed for each movement. Therefore, the team looked into a message type that the CAN protocol sends to let the system know that the motors had reached a targeted position. This

would allow for new commands to be sent only when this completion message is received. A solution the team has settled on so far is reading from the Encoder Estimates message, and checking if the current position is within some tolerance of the target position. Once this was achieved, the team applied this process to the other leg. In the end, the team was able to make the robot move in a walking motion, albeit without trajectories or inverse kinematics.

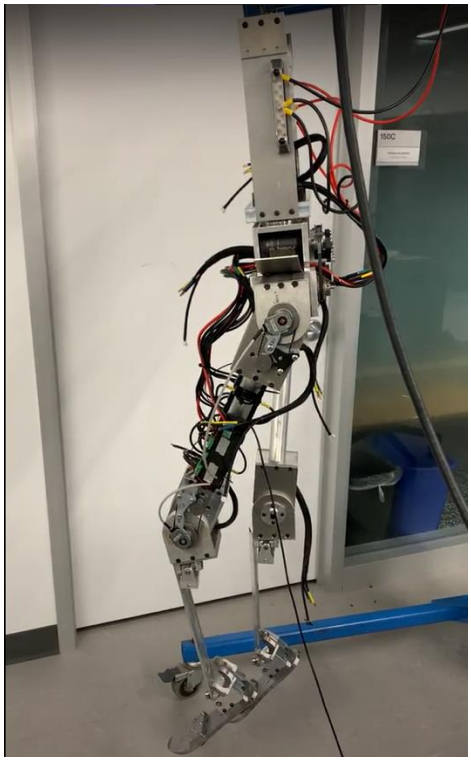


Figure 15.2: HURON with the left leg bent at the hip and knee through code

16.0 Raspberry Pi without Ethernet

Initially, the Raspberry Pi was set up for ssh access through an ethernet port. However, we wanted to find a way to access the Raspberry Pi without an ethernet port in case there were no ethernet ports available. A proposed solution was to use an adapter between ethernet and USB. However, an issue arose where the Raspberry Pi would not connect with our laptops. Multiple adapters and laptops with different operating systems were tested, but neither terminal nor PuTTY allowed for a connection and resulted in a "connection timed out" error.

To overcome this issue, we found a working solution by connecting the Raspberry Pi to an external display using the HDMI port and using the USB ports to connect an external keyboard and mouse. With this setup, we could pull the code from the repository to update it and command the Raspberry Pi to run the code on the robot. However, the Raspberry Pi did not have a connection to Wi-Fi set up, which made updating from the repository impossible. In this case, we manually updated the code by transferring the USB from the Raspberry Pi to a laptop.

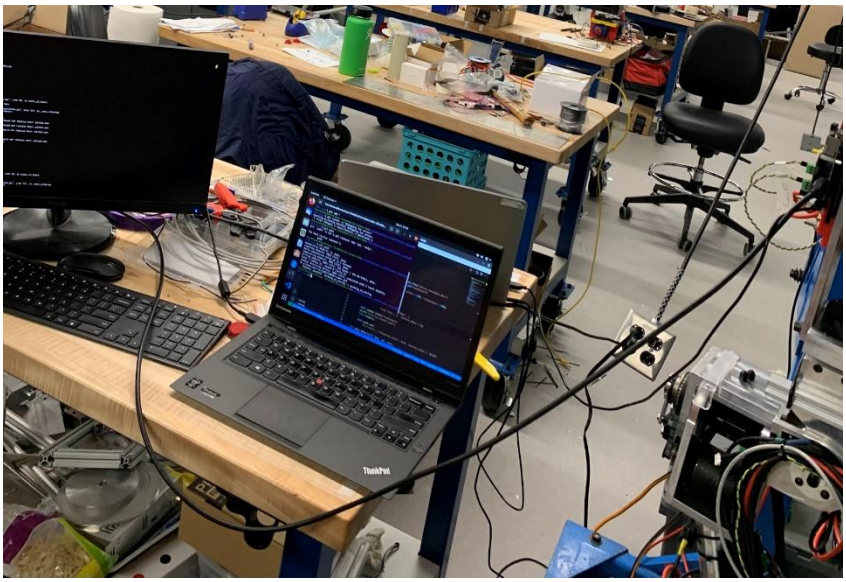


Figure 16.1: The programming setup without an ethernet connection.

17.0 Final Control Algorithm

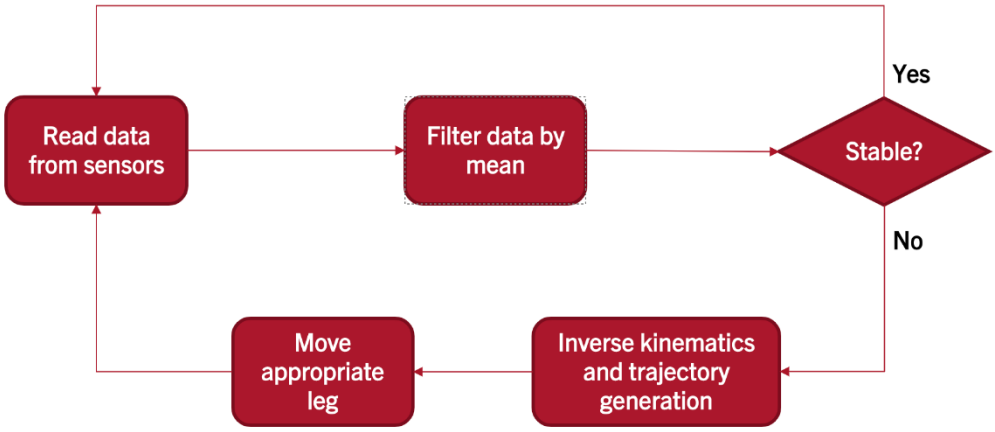


Figure 17.1: Control Algorithm Flow Chart

The implementation of our control algorithm is illustrated in the figure above. The first step of the process involves waiting for data from the sensors. After receiving the data, we apply a mean filter to reduce the effect of noise. We filtered out erroneous sensor data by setting a threshold to reject any data outside of the expected range. Once the data has been filtered, the algorithm returns two pieces of information: a decision and a distance to move. The decision tells us the state of the robot, whether it is stable or unstable, and if unstable, which leg to move. Subsequently, we use this decision to move the legs the required distance using inverse kinematics and trajectory generation. This process is then repeated in a loop.

While implementing our control algorithm, we encountered several challenges that needed to be overcome. One significant challenge was motor calibration errors, which we resolved by fine-tuning the motor control parameters. Another challenge we faced was ensuring the stability of the robot's gait over time. To accomplish this, we repeated the cycle of waiting for sensor data, filtering it, making a decision, and moving the legs using inverse kinematics and trajectory generation for every successive step taken by the robot.

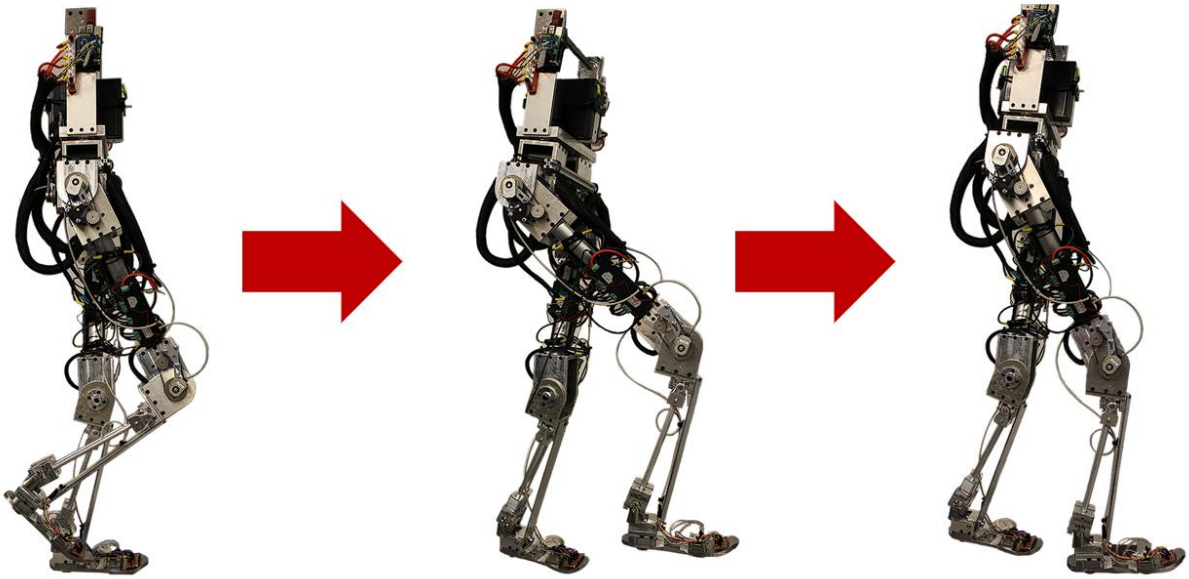


Figure 17.2: Robot Taking a Step in Response to an External Force

18.0 Pitch Paper and Sponsorship Work

After budgeting the robot, it was apparent that the project would not have enough funding to be completed. As a result, we were tasked with creating a pitch paper and email template to request sponsorships from companies and organizations. To help complete this, the sensor team collaborated with the executive director of corporate relations at WPI, Dave Orthendahl. Dave would help edit our pitch paper and our email template to make them more appealing to readers. The role of the pitch paper was to be a quick and easy display that we could share with many companies where they can be informed about our project and what we needed. Our final pitch paper consists of information about the project and the impact that the project would have, the project requirements, timeline, the budget, team and contact, and a sponsorship section with sponsorship levels. The pitch paper includes pictures of the CAD models, tables, and diagrams. The reason we needed an email template was so that everyone would be able to create a uniform and concise email to companies that would effectively educate readers about an MQP, what our MQP was, and what we needed from the company. Our final email template consists of 4 paragraphs that give information about WPI MQPs, what our project is, what we are looking for and what the donation would mean to us, and extra sources of information.

To help obtain sponsors for the project, we emailed many companies and organizations through their website, connections, and social media. We collaborated with members with the controls team and Dave to create a running spreadsheet. This would help organize the list of companies that we have contacted, who in the company we have contacted, whether they have responded, and what we have received from them. So far, we have obtained $500 dollars and discounts on certain parts. Dave is also assisting with this as he has many connections with WPI alumni in certain companies. We are making a list of WPI alumni from certain companies that Dave may possibly know to help us contact more companies. We have also worked with Prof. Agehli and a RBE admin, Sharon, to create a WPI sponsorship account to hold our sponsorship

donations.

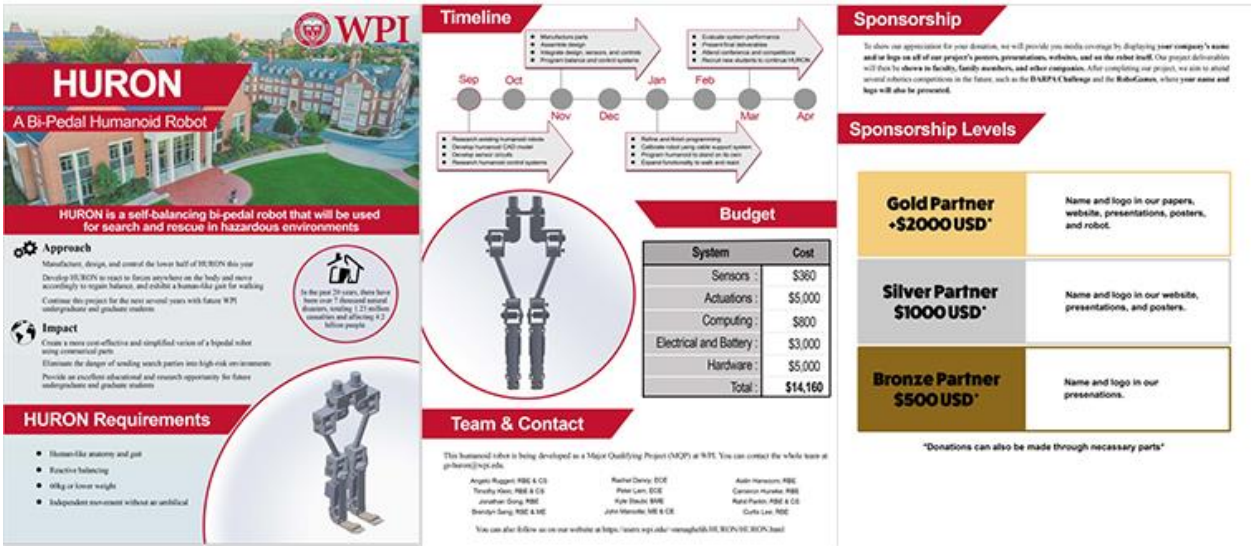


Figure 18.1: Pitch Paper

Over the months, we have attempted to contact many other companies using connections, the help of Dave Ortendahl, and LinkedIn. However, the response rate was very limited, we've had responses only from Analog Devices, Red Hat, and Raytheon at the moment. With Analog Devices, the team and an employee are attempting to find a path that would lead to donations that could apply to our project. With Raytheon, there is a similar situation where we are trying to find the correct person to talk to, to help possibly fund our project, so far, we are trying to contact the Raytheon campus manager for WPI to see if there is any program that Raytheon does to help donate money to WPI projects. Red Hat is our closest lead so far, as they have discussed within their company if there was a possibility to donate money. So far, they are hesitant as we are an undergraduate program running under a university, and they have mainly donated to FIRST robotic programs in the past. There have been many discussions about the state and position of our project, but no final decision yet. Throughout the process, we also needed to retrieve a 501(c) for proof that we are a non-profit project and make ourselves more applicable to potential sponsors.

19.0 Conclusions

Huron is the lower half of a full-sized humanoid robot that has been designed and modeled after a 5ft10in male. After the fabrication process, Huron weighs 96lbs and stands at a height of 56.7in. The robot is equipped with 8 Force Sensitive Resistor (FSR) sensors, each capable of detecting a maximum force of 200lbs for each foot. These sensors play a critical role in calculating the Foot Force Stability Margin (FFSM) and detecting instability in the robot's gait.

If Huron detects any instability, it can take corrective steps using its advanced algorithms with a reaction time of 2.5s, starting from sensing imbalance to touchdown of the supporting leg. This capability allows the robot to maintain balance and stability even in challenging environments. When walking, the robot can cover a maximum stride length of 7.9in and has a net gait time of 4s.

20.0 Next Steps

20.1 Design Team

In terms of the mechanical design of HURON, the main goal that next year's MQP team should focus on is the redesign of the ankle. As mentioned previously, the ankle was unable to be completed due to complications with orders on the linear actuators, leaving the robot in its current state of only having 8 DOF. However, the linear actuator design is still not feasible in the long term as they tend to be too slow at the constrained size, considering that the robot will eventually need to be able to balance and walk on its own (i.e., without the need for the test rig). The main drawback that was faced this year was budget and time constraints, so having next year's team focus on the ankle design should be feasible as most of the budget will be concentrated on improving one joint instead of being spread throughout the entire leg.

Assuming this case of being able to focus the budget on one joint, the team should design around the use of more expensive actuation methods, such as the use of harmonic drives. Having harmonic drives could work in tangent with the current motors to provide enough torque and still have the desired speed for walking and balancing. With this, a new design could be done while keeping in mind the constraint of being within human proportions; however, the main obstacle would be funding as harmonic drives and the necessary material would be quite expensive. Additionally, in order to reduce inaccuracies in the force sensors while the robot walks on varying terrain, a certain amount of compliance should be added to either the foot or calf as even with correct calculations, the machine tolerances throughout the whole system could affect the sensor readings.

Aside from the improvements that can be made on the robot itself, a completely new test rig should be made using a rail system that was mentioned in Section 14.0 where the robot can walk without much external interference. Although small modifications to the current test rig can be done to allow the robot to take a couple steps, it does not allow extensive testing of the whole system.

20.2 Controls Team

As far as control is concerned, it is necessary that next year's team incorporates the remaining motors and ankle actuators into the inverse kinematic model of the robot. The installation of

absolute angle encoders on the top hip motors is also necessary to enable these motors to be actuated properly. By including these additional actuators, the robot will be able to level its feet and receive more accurate foot sensor readings. This will enhance the robot's stability and allow for more precise motion control.

Furthermore, having control of the ankle is essential for creating a more human-like walking gait. The current implementation of the staple-shaped trajectory has limited the robot's range of motion and fluidity of movement. By refining the ankle actuator control, next year's team can improve the robot's walking gait and achieve more natural movements. While the current system is functional, next year's team should strive to enhance the robot's actuation speed and refine the method of responding to external forces. This will help to create a more practical and effective reactive balancing system that can adapt to different environments and scenarios.

Building on this, the team should also focus on the reactive balancing component while the robot is walking. This will require more advanced control algorithms that can monitor and adjust the robot's balance with less noise while in the process of taking a step. Achieving this will ensure that the robot remains balanced and stable even when subjected to unexpected external forces during the walking gate.

20.3 Sensors Team

There are several improvements to the current sensor system which future MQPs could implement. First, the circuit used to pick up FSR signals could include a bandpass filter, to limit noise and DC offset. Second, other sensors could be implemented in the balancing scheme, including IMUs at the ankle and hip. This would allow for a third improvement; altering the Arduino script to include a dynamic balancing system utilizing ZMP measurements. Some mechanical improvements to the system could also be made, including designing housing for the perf boards and FSR wires on the feet.

21.0 Acknowledgements

The team would like to thank all the people and the groups who have made this project possible:

- Our sponsor, Red Hat, for providing funds to help the project
- Lambda Chi Alpha for donating to help the project
- Ibrahim Al-Tameemi for deriving the initial equations for the inverse kinematics
- Washburn Shops Staff for helping and providing CNC assistance and guidance
- Daniel Ali Tribaldos for guiding and assisting waterjet operations

22.0 Authorship

The breakdown of each section with list of authors who contributed to that section is shown below with authors listed in order from most contribution to least per section.

Table 22.1. Authorship Table

Task	Authors
Introduction	Aislin Hanscom, Jonathan Gong
Motivation	Jonathan Gong
Background	Jonathan Gong
Humanoid Robot Design	Jonathan Gong
Degrees of Freedom	Jonathan Gong, Aislin Hanscom
Leg Structure	Aislin Hanscom, Jonathan Gong
Joint Orientations & Motion	Jonathan Gong, Aislin Hanscom
Human Leg Bones	Aislin Hanscom
Harmonic Drive	Aislin Hanscom
Humanoid Robot Control	Rahil, Jack, Angelo, Curtis
Forces	Rahil
Actuation	Jack, Angelo, Curtis
Humanoid Robot Sensing and Balancing	Rachel and Kyle
Balancing Mechanisms	Kyle
Sensors for Balancing Robots	Rachel, Kyle, and Peter
FSR Placement	Rachel
Foot Force Stability Margin	Rachel and Kyle
Robot Design Process	Aislin Hanscom
Iteration 1	Aislin Hanscom
Proportions	Aislin Hanscom
Joint Order and Positions	Aislin Hanscom
CAD	Brendyn Sang
Materials	Aislin Hanscom
Cost	Aislin Hanscom
Motor Stand-ins	Brendyn Sang

Harmonic Drive	Brendyn Sang, Aislin Hanscom
Iteration 2	Aislin Hanscom
Proportions	Brendyn Sang
Joint Order and Positions	Cameron Huneke
CAD	Brendyn Sang
Hip Joints	Cameron Huneke
Knee Joints	Jonathan Gong
Ankle Joints	Jonathan Gong
Leg Bones	Jonathan Gong
Materials	Cameron Huneke
Cost	Aislin Hanscom
Motor Stand-ins	Brendyn Sang
Harmonic Drive	Brendyn Sang
Iteration 3	Aislin Hanscom
Proportions	Brendyn Sang
Joint Order and Positions	Aislin Hanscom
CAD	Brendyn Sang
Hip Joints	Brendyn Sang
Knee Joints	Jonathan Gong
Ankle Joints	Jonathan Gong
Leg Bones	Jonathan Gong
Foot	Jonathan Gong
Materials	Cameron Huneke
Cost	Aislin Hanscom
Motors	Cameron Huneke
Harmonic Drive	Brendyn Sang; Cameron Huneke
Redesign for Manufacturability	
Overall Design	Brendyn Sang
Joints	Cameron Huneke, Aislin Hanscom
Cost	Aislin Hanscom
Weight and Joint Torques	Brendyn Sang
Finite Element Analysis (FEA)	Cameron Huneke
Manufacturing Tools	Cameron Huneke
Robot Control Preliminary Design	Jack

Motors	Curtis
Motor Controllers/External Encoders	Jack
Power Supply	Jack
Computing	Angelo
Budget	Rahil
Robot Sensing Design and Testing	Rachel
Location of FSRs and Design of Foot	Rachel
Materials	Rachel
Circuit Design	Rachel and Peter
Testing	Rachel
Finalizing Circuit	Peter
Verification of Stability Measurement and Angle of Lean	Kyle
Manufacturing Process	Jonathan Gong
Finalizing Ankle Design	Jonathan Gong
Linear Actuators	Jonathan Gong
Final Design	Jonathan Gong
Learning CAM	Jonathan Gong
Learning to Machine/Waterjet	Jonathan Gong
Machining	Jonathan Gong
Waterjet	Jonathan Gong and Curtis
CAM/Machining Process	Jonathan Gong
Stock Setup	Jonathan Gong
Changes due to CAM	Jonathan Gong
Changes due to Manufacturability	Jonathan Gong
Waterjet Parts	Jonathan Gong
Assembly Documentation	Aislin Hanscom, Jonathan Gong
Ankle Assembly	Aislin Hanscom, Jonathan Gong
Knee Assembly	Aislin Hanscom, Jonathan Gong
Hip Assembly	Aislin Hanscom, Jonathan Gong
Test Rig Mount Assembly	Aislin Hanscom, Jonathan Gong

Sensor Integration on HURON	
Mounting Sensors Onto HURON	Kyle and Rachel
Verification of Voltage to Force Conversion	Rachel
Stability Recovery and Active Balancing	Rachel and Kyle
Robot Control Design and Simulation	Jack and Angelo
Control System	Jack
Power Supply	Jack
Simulation in Gazebo	Angelo
Motor Integration and Testing	Angelo
Wiring and Communication	Jack
Code Structure	Rahil and Curtis
Simulation of Inverse Kinematics	Rahil and Jack
Final Control Algorithm	Rahil and Angelo
Testing on Whole System: Legs Move	Curtis
Raspberry Pi without Ethernet	Angelo
Pitch Paper and Sponsorship Work	Peter
Conclusions	All
Next Steps	All
Acknowledgements	All
References	All
Design Team Appendices	Aislin, Jonathan, Brendyn, and Cameron
Controls Team Appendix	Jack
Sensor Team Appendix A	Rachel
Sensor Team Appendix B	Rachel and Kyle
Sensor Team Appendix C	Rachel, Kyle, and Peter
Formatting	Aislin Hanscom, Jonathan Gong

23.0 References

Agheli, M., & Nestinger, S. (2014). Closed-Form Solution for Constant-Orientation Workspace and Workspace-Based Design of Radially Symmetric Hexapod Robots. *Journal of Mechanisms and Robotics*, *6*. https://doi.org/10.1115

Agheli, M., & Nestinger, Stephen. S. (2016). Force-based stability margin for multi-legged robots. *Robotics and Autonomous Systems*, *83*, 138–149. https://doi.org/10.1016/j.robot.2016.05.012

Aitken, S. (2021). Normative Values for Femoral Length, Tibial Length, and the Femorotibial Ratio in Adults Using Standing Full-Length Radiography. *Osteology*, *1*, 86–91. https://doi.org/10.3390/osteology1020009

Amazon.com: Flipsky Electric Scooter Motor 6374 Offroad Electric Longboard Skateboards Motor BLDC Brushless Outrunner Motor 190kv 3500w for Electric Skateboard,Longboard,Escooter,Ebike Esk8 DIY Kit Parts: Sports & Outdoors. (n.d.). Retrieved April 25, 2023, from https://www.amazon.com/Flipsky-Hardened-Skateboard-Brushless-Accessories/dp/B08L5WDTSZ/ref=sr_1_7?keywords=skate%2Bmotor&qid=1663037094&sr=8-7&th=1

Appleton, B. (n.d.). *STRETCHING AND FLEXIBILITY - Normal Ranges of Joint Motion*. Retrieved October 12, 2022, from https://web.mit.edu/tkd/stretch/stretching_8.html

Biped Humanoid Robot WABIAN-2R. (n.d.). Retrieved April 25, 2023, from https://web.archive.org/web/20111106123739/http://www.takanishi.mech.waseda.ac.jp/top/research/wabian/index.htm

Brushless DC Motor vs. Stepper Motor. (n.d.). Retrieved April 25, 2023, from https://www.omc-stepperonline.com/support/brushless-dc-motors-vs-stepper-motors

Buschmann, T., Lohmeier, S., & Ulbrich, H. (2009). Humanoid robot Lola: Design and walking control. *Journal of Physiology, Paris*, *103*(3), 141–148. https://doi.org/10.1016/j.jphysparis.2009.07.008

Carreiro, J. E. (2009). Chapter Six—Lower leg. In J. E. Carreiro (Ed.), *Pediatric Manual Medicine* (pp. 273–327). Churchill Livingstone. https://doi.org/10.1016/B978-0-443-10308-7.00006-5

Chen, G., Li, H., & Liu, Y. (2019). Double-arc harmonic gear profile design and meshing analysis for multi-section conjugation. *Advances in Mechanical Engineering*, *11*(5), 1687814019850656. https://doi.org/10.1177/1687814019850656

Chew, W. T. (n.d.). *Design and Control of a Humanoid Robot*. 198.

Continuum Robots and Tactile Sensors. (2023, January 8). Tacuna Systems. https://tacunasystems.com/knowledge-base/continuum-robots-and-tactile-sensors/

Cristofolini, L., & Viceconti, M. (2000). Mechanical validation of whole bone composite tibia models. *Journal of Biomechanics*, *33*(3), 279–288. https://doi.org/10.1016/S0021-9290(99)00186-4

Dabhi, M. D., Bhatt, H. J., Pandya, P., & Scholar, P. (2019). *Design and Development of Harmonic Drive for Shredder Machine to Achieve Higher Gear Ratio up to 750:* 12.

DC Motor vs Stepper Motor vs Servo Motor—Which to choose? (2019, April 1). *Latest Open Tech From Seeed*. https://www.seeedstudio.com/blog/2019/04/01/choosing-the-right-motor-for-your-project-dc-vs-stepper-vs-servo-motors/

Encoders—ODrive Pro Documentation 0.6.6 documentation. (n.d.). Retrieved April 25, 2023, from https://docs.odriverobotics.com/v/latest/encoders.html

FlexiForce Pressure Sensor—100lbs. - SEN-08685—SparkFun Electronics. (n.d.). Retrieved April 25, 2023, from https://www.sparkfun.com/products/8685

Folgheraiter, M., & Aubakir, B. (2018). Design and Modeling of a Lightweight and Low Power Consumption Full-Scale Biped Robot. *International Journal of Humanoid Robotics*, *15*(05), 1850022. https://doi.org/10.1142/S0219843618500226

Harmonic Drive® strain wave gear—Zero backlash | Harmonic Drive | Harmonic Drive. (n.d.). Retrieved September 30, 2022, from https://www.harmonicdrive.net/technology

How Do Force Sensors Work? And What Are Their Benefits? (n.d.). Retrieved April 25, 2023, from https://roboticsandautomationnews.com/2020/04/22/how-do-force-sensors-work-and-their-benefits/31897/

How to Mechatronics (Director). (2020, June 18). *What is Strain Wave Gear a.k.a. Harmonic Drive? A Perfect Gear Set For Robotics Applications* [Youtube]. https://www.youtube.com/watch?v=xlnNj9F37MA

Hyon, S.-H., Suewaka, D., Torii, Y., & Oku, N. (2017). Design and Experimental Evaluation of a Fast Torque-Controlled Hydraulic Humanoid Robot. *IEEE/ASME Transactions on Mechatronics*, *22*(2), 623–634. https://doi.org/10.1109/TMECH.2016.2628870

I. -w. Park, J. -y. Kim, & J. -h. Oh. (2006). Online Biped Walking Pattern Generation for Humanoid Robot KHR-3(KAIST Humanoid Robot—3: HUBO). *2006 6th IEEE-RAS International Conference on Humanoid Robots*, 398–403. https://doi.org/10.1109/ICHR.2006.321303

Ill-Woo Park, Jung-Yup Kim, Jungho Lee, & Jun-Ho Oh. (2005). Mechanical design of humanoid robot platform KHR-3 (KAIST Humanoid Robot 3: HUBO). *5th IEEE-RAS International Conference on Humanoid Robots, 2005.*, 321–326. https://doi.org/10.1109/ICHR.2005.1573587

Jerry E. Pratt & Benjamin T. Krupp. (2004). *Series Elastic Actuators for legged robots*. *5422*, 135–144. https://doi.org/10.1117/12.548000

Jianxi Li, Qiang Huang, Weimin Zhang, Zhangguo Yu, & Kejie Li. (2008). Flexible foot design for a humanoid robot. *2008 IEEE International Conference on Automation and Logistics*, 1414–1419. https://doi.org/10.1109/ICAL.2008.4636375

K. Erbatur, A. Okazaki, K. Obiya, T. Takahashi, & A. Kawamura. (2002). A study on the zero moment point measurement for biped walking robots. *7th International Workshop on Advanced Motion Control. Proceedings (Cat. No.02TH8623)*, 431–436. https://doi.org/10.1109/AMC.2002.1026959

Kondo, K., & Takada, J. (1990). Study on Tooth Profiles of the Harmonic Drive. *Journal of Mechanical Design*, *112*(1), 131–137. https://doi.org/10.1115/1.2912570

Lee, B., Knabe, C., Orekhov, V., & Hong, D. (2014). Design of a Human-Like Range of Motion Hip Joint for Humanoid Robots. *Volume 5B: 38th Mechanisms and Robotics Conference*, V05BT08A018. https://doi.org/10.1115/DETC2014-35214

López, R. (2014). *A lower limb exoskeleton with hybrid actuation*.

M. Agheli & S. S. Nestinger. (2012). Study of the Foot Force Stability Margin for multi-legged/wheeled robots under dynamic situations. *Proceedings of 2012 IEEE/ASME 8th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications*, 99–104. https://doi.org/10.1109/MESA.2012.6275544

M. Lapeyre, P. Rouanet, & P. -Y. Oudeyer. (2013). The poppy humanoid robot: Leg design for biped locomotion. *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 349–356. https://doi.org/10.1109/IROS.2013.6696375

Ma, D., Wang, R., Rao, P., Sui, R., & Yan, S. (2018). Automated Analysis of Meshing Performance of Harmonic Drive Gears Under Various Operating Conditions. *IEEE Access*, *6*, 68137–68154. https://doi.org/10.1109/ACCESS.2018.2872797

ODrive. (n.d.). ODrive. Retrieved April 25, 2023, from https://odriverobotics.com

Online shop for high precise drive systems by maxon | maxon group. (n.d.). Retrieved April 25, 2023, from https://www.maxongroup.com/maxon/view/product/motor/ecmotor/ec4pole/305013

OpenDog Electronics & Initial Code. (2018, October 9). *XRobots*. http://www.xrobots.tech/opendog-electronics-initial-code/

Oregon State University (Director). (2021, July 27). *OSU Bipedal Robot First to Run 5K*. https://www.youtube.com/watch?v=a_YGPbWJO5g

P. S. Malvade, A. K. Joshi, & S. P. Madhe. (2017). IoT based monitoring of foot pressure using FSR sensor. *2017 International Conference on Communication and Signal Processing (ICCSP)*, 0635–0639. https://doi.org/10.1109/ICCSP.2017.8286435

Peter, E. (2019). Evaluation of osteometric parameters of fibula and talar facet morphometry in Telangana region. *Indian Journal of Clinical Anatomy and Physiology*, *2019*, 10429. https://doi.org/10.18231/j.ijcap.2019.109

Prototyping Lab: WPI Makerspace. (n.d.). Retrieved October 12, 2022, from https://canvas.wpi.edu/courses/9342/pages/Prototyping%20Lab?titleize=0

Rana, N. K. (2010). Application of Force Sensing Resistor (FSR) in Design of Pressure Scanning System for Plantar Pressure Measurement. *Proceedings of the 2009 Second International Conference on Computer and Electrical Engineering - Volume 02*, 678–685. https://doi.org/10.1109/ICCEE.2009.234

Ren, Y., Sun, X., & Liu, J. (2020). Advances in Liquid Metal-Enabled Flexible and Wearable Sensors. *Micromachines*, *11*(2), 200. https://doi.org/10.3390/mi11020200

Sardain, P., & Bessonnet, G. (2004). Forces Acting on a Biped Robot. Center of Pressure-Zero Moment Point. *Trans. Sys. Man Cyber. Part A*, *34*(5), 630–637. https://doi.org/10.1109/TSMCA.2004.832811

Solis, J. (n.d.). *FSR Integration Guide*.

Świć, A. (Ed.). (2009). *Modern techniques in mechanical engineering*.

Tar, A., & Veres, J. (2006). Design and Realization of a Biped Robot Using Stepper Motor Driven Joints. *2006 IEEE International Conference on Mechatronics*, 493–498.

Top 10 Examples of Humanoid Robots—ASME. (n.d.). Retrieved April 25, 2023, from https://www.asme.org/topics-resources/content/10-humanoid-robots-of-2020

Varshney, A., Singh, A., Das, R., Ranjan, S., Gupta, S., & Varshney, U. (2019). *Mechanical Design of Humanoid Robot, AUTOMI*.

Wang, A., Ramos, J., Mayo, J., Ubellacker, W., Cheung, J., & Kim, S. (2015). The HERMES humanoid system: A platform for full-body teleoperation with balance feedback. *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, 730–737. https://doi.org/10.1109/HUMANOIDS.2015.7363451

Wibowo, D. B., Suprihanto, A., Caesarendra, W., Khoeron, S., Glowacz, A., & Irfan, M. (2020). A Simple Foot Plantar Pressure Measurement Platform System Using Force-Sensing Resistors. *Applied System Innovation*, *3*(3), 33. https://doi.org/10.3390/asi3030033

Yang, T., Westervelt, E. R., Schmiedeler, J. P., & Bockbrader, R. A. (2008). Design and control of a planar bipedal robot ERNIE with parallel knee compliance. *Autonomous Robots*, *25*(4), 317–330. https://doi.org/10.1007/s10514-008-9096-5

Yi, S.-J., McGill, S., Vadakedathu, L., He, Q., Ha, I., Han, J., Song, H., Rouleau, M., Hong, D., & Lee, D. D. (2014). THOR-OP humanoid robot for DARPA Robotics Challenge Trials 2013. *2014 11th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 359–363. https://doi.org/10.1109/URAI.2014.7057369

Yu Ogura, Aikawa, H., Shimomura, K., Kondo, H., Morishima, A., Hun-ok Lim, & Takanishi, A. (2006). Development of a new humanoid robot WABIAN-2. *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, 76–81. https://doi.org/10.1109/ROBOT.2006.1641164

24.0 Appendices

24.1 Design Team Appendices

24.1.1 Appendix A: Initial Torque Calculations on Joints

These calculations are based on some initial assumptions of what materials were used, the size of the robot and a shell around the internal wiring. The initial assumptions we made were:

- Mass of 3D parts and aluminum stock are point masses
- Aluminum rods are length of full leg sections (ie, not accounting size of joint)
- Motors and gear boxes are massless

Based on these assumptions the values for each joint torque was as follows:

- Hip yaw
    - Minimum force/friction
        - T = 0
- Hip Pitch:
    - Max torque at 90 degrees straight
    - only have to worry about one leg
    - Tmax = [(17.145cm)(6.68kg) + (53.72cm)(7.57kg)+(98.31cm)(2.92)kg]*(9.81 m/s^2) = 79.29 Nm
- Hip Roll
    - Max torque at 90 degrees out to the right/left
    - Tmax = [(12.145cm)(6.68kg)+(48.72cm)(7.57kg)+(93.31cm)(2.92kg)]*(9.81 m/s^2) = 70.87 Nm
- Hip Knee
    - Max torque when knee is bent backwards at 90 degrees
    - Tmax = [(19.43)(7.57kg)+(19.43 + 25.16)(2.92kg)]*(9.81 m/s^2) = 27.2 Nm
- Hip Ankle
    - Max torque taken from a real human walking, using protractor to estimate angles of the leg
    - Tmax = [15cos(5)*2.92kg + 19.43cos(60)*7.57kg + (38.8cos(60)-17.145sin(10))*6.68kg + (38.86cos(60)-34.29cos(10) + 17.145sin(15))*6.68kg +

(38.86cos(60)*34.29cos(10)+34.29sin(15) + 19.43sin(15)*7.57kg +
(38.86cos(60)-34.29cos(10) + 34.29sin(15) + 38.86sin(15) + 15)*2.92kg]*(9.81
m/s^2) = 67.89 Nm

With these numbers in mind, we then created the initial torque requirements for each joint, which led us to the need for a Harmonic Drive.

24.1.2 Appendix B: Finalized Joint Torque Calculations

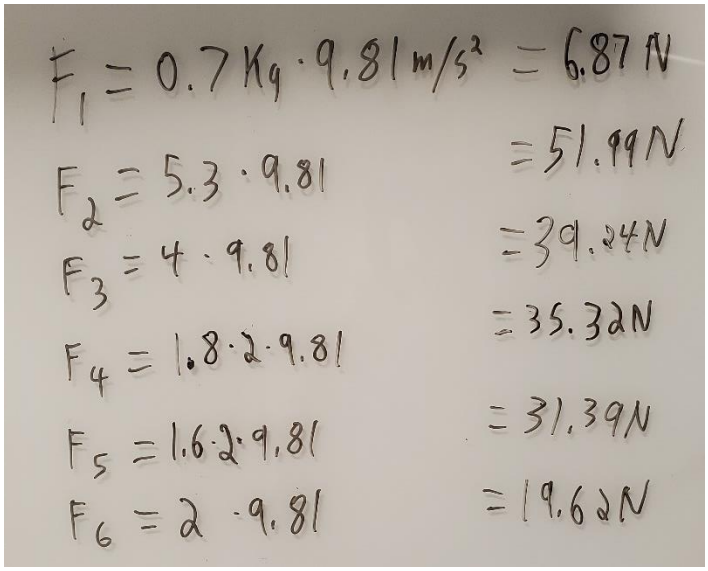


Figure 24.1: Force of different center of masses of joint linkages

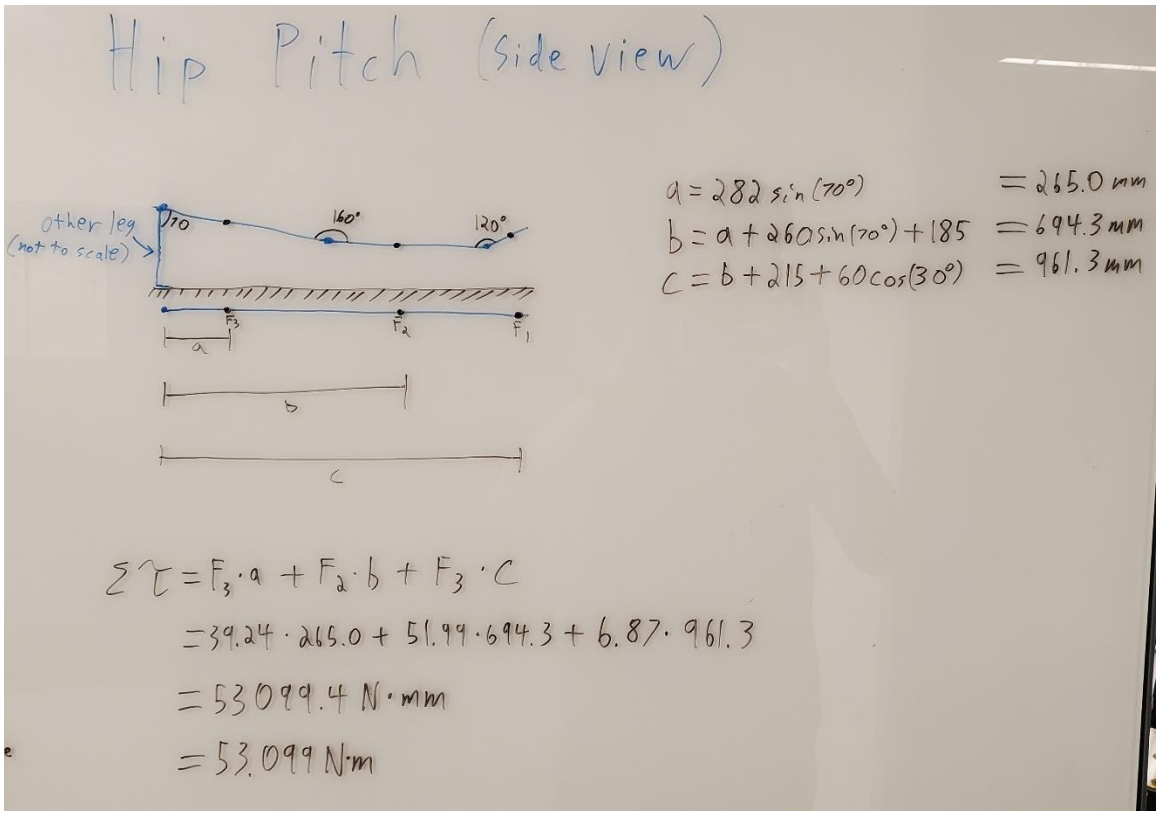


Figure 24.2: Hip pitch worst-case scenario torque calculations

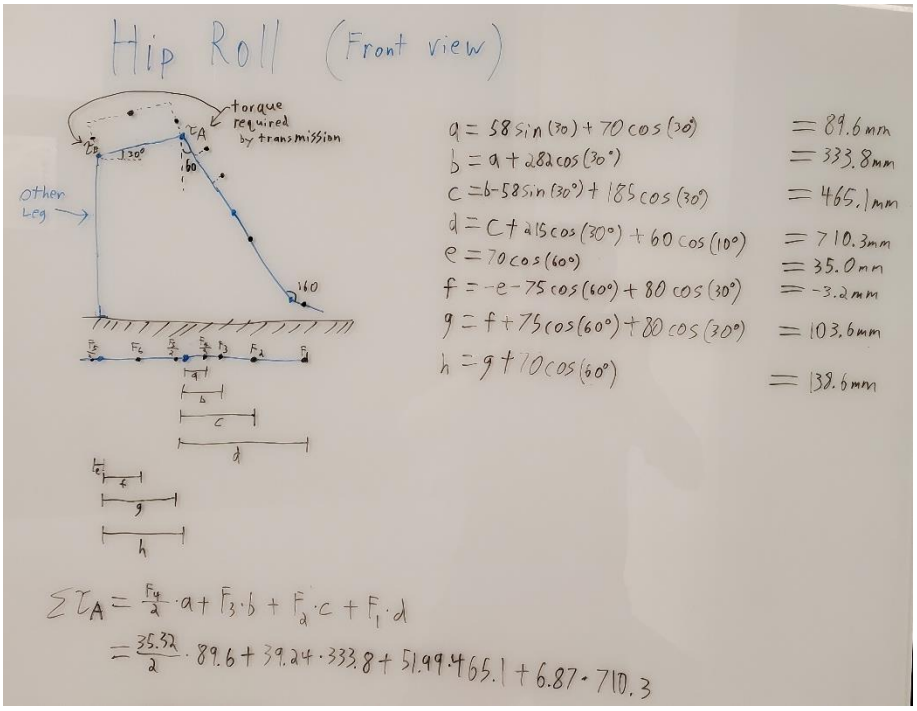


Figure 24.3: Hip roll worst-case scenario torque calculations

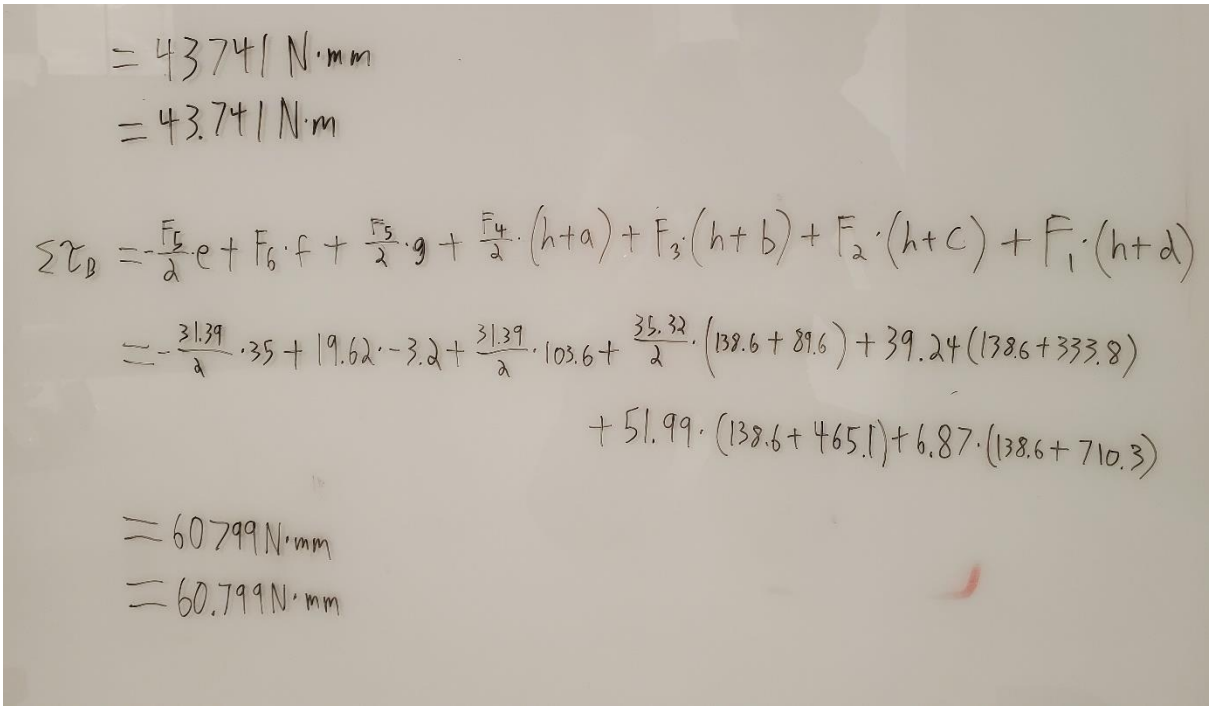


Figure 24.4: Hip roll worst-case scenario torque calculations cont.

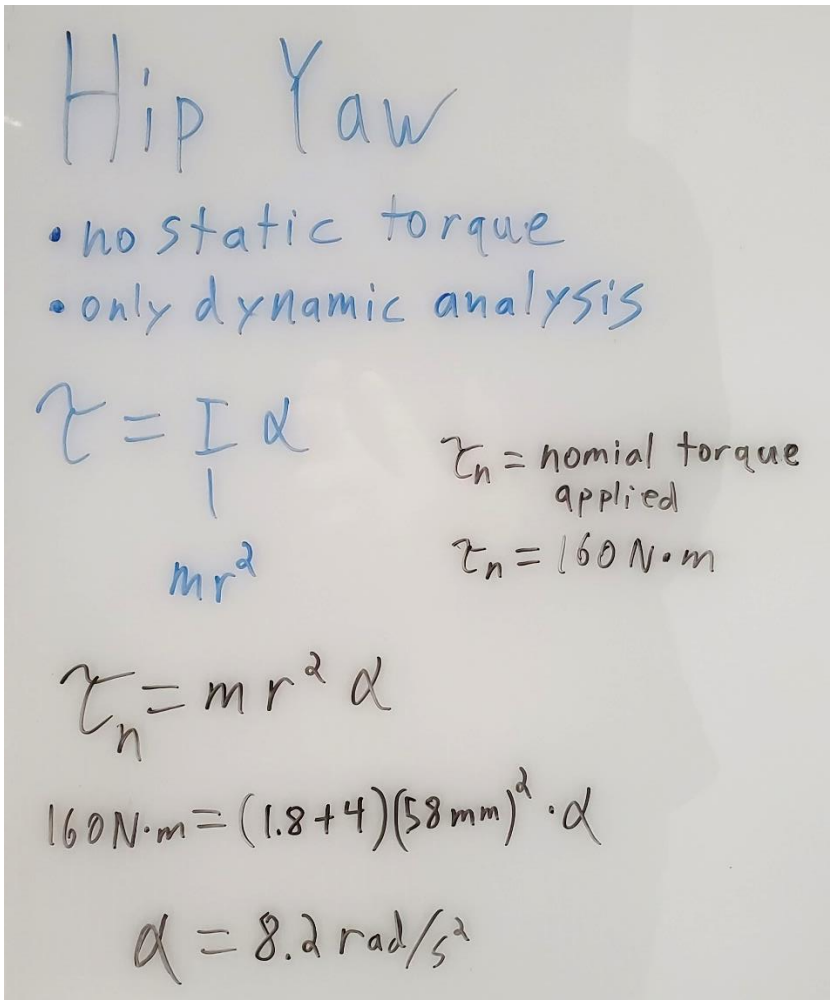


Figure 24.5: Hip yaw worst-case scenario dynamic torque calculations

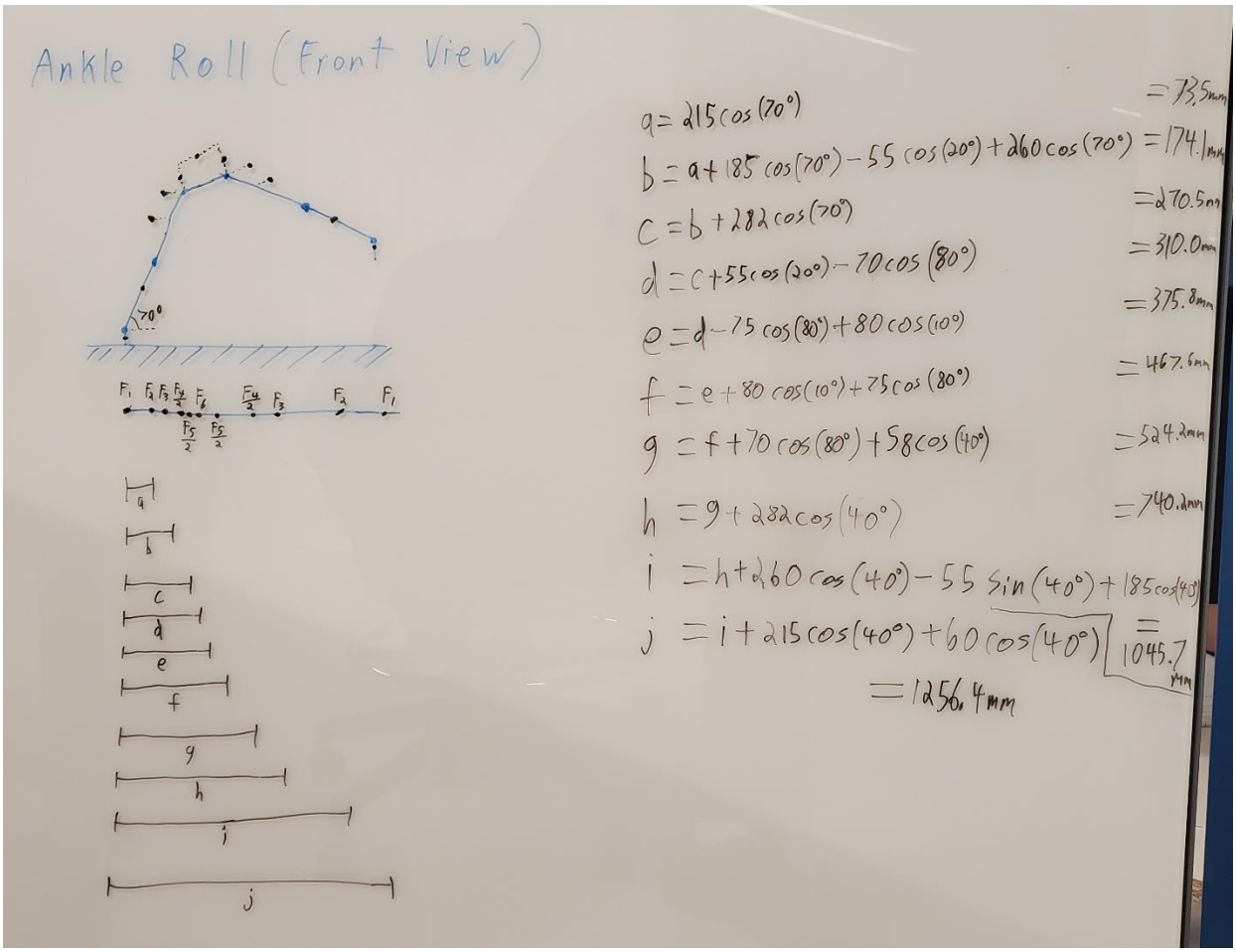


Figure 24.6: Ankle roll worst-case scenario torque calculations

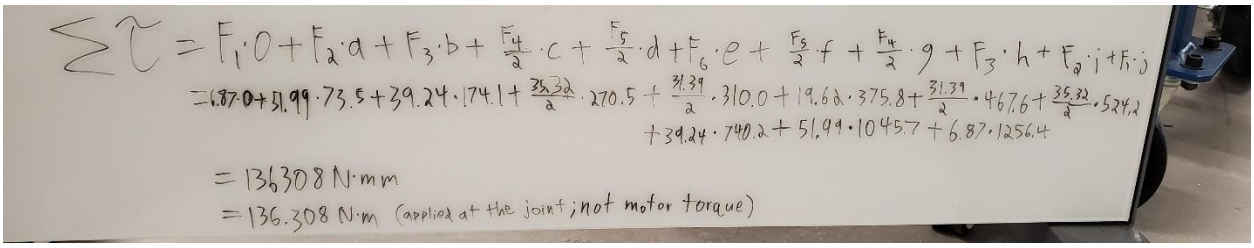


Figure 24.7: Ankle roll worst-case scenario torque calculations cont.

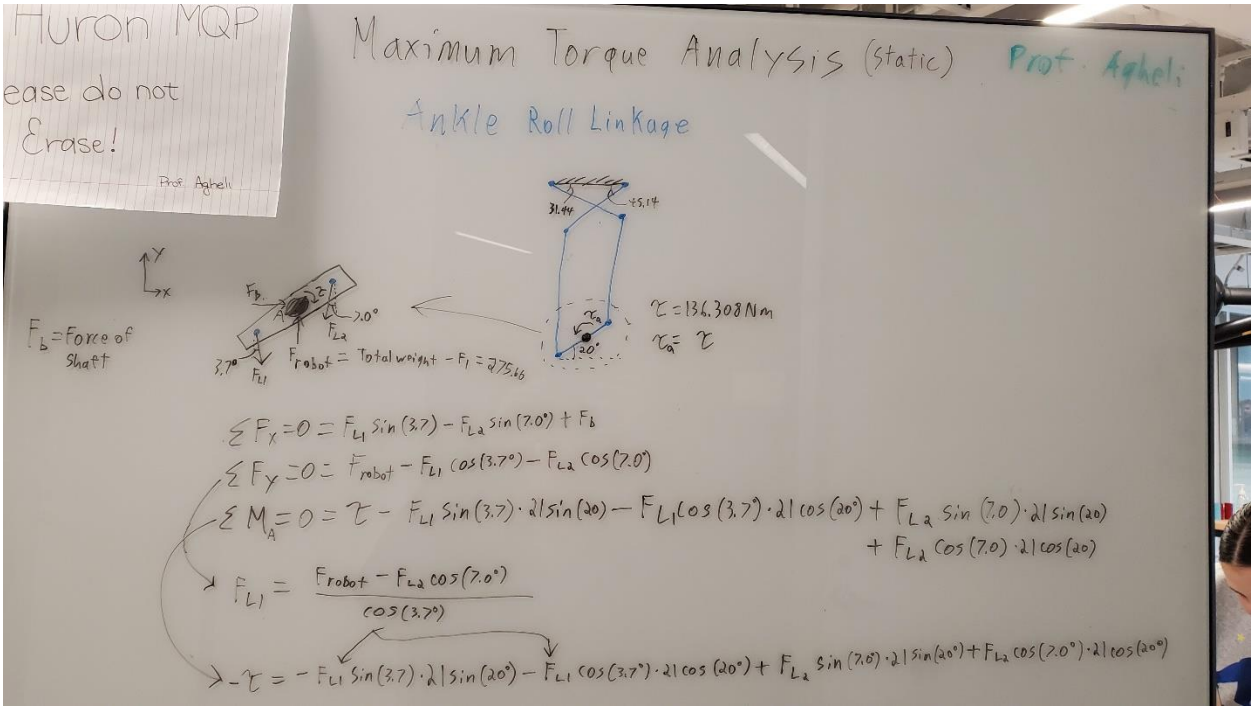


Figure 24.8: Ankle roll 6-bar linkage worst-case scenario torque calculations

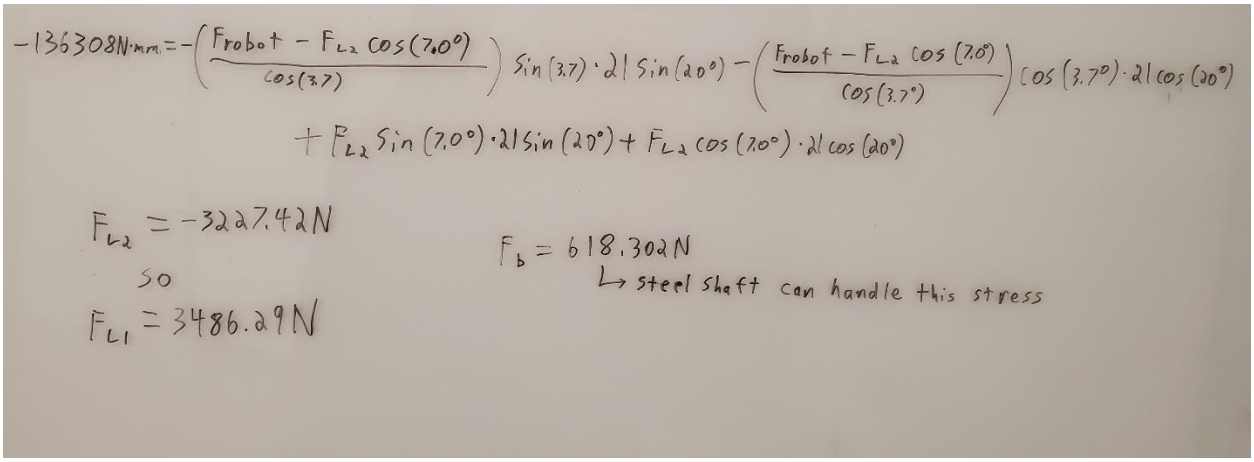


Figure 24.9: Ankle roll 6-bar linkage worst-case scenario torque calculations cont.

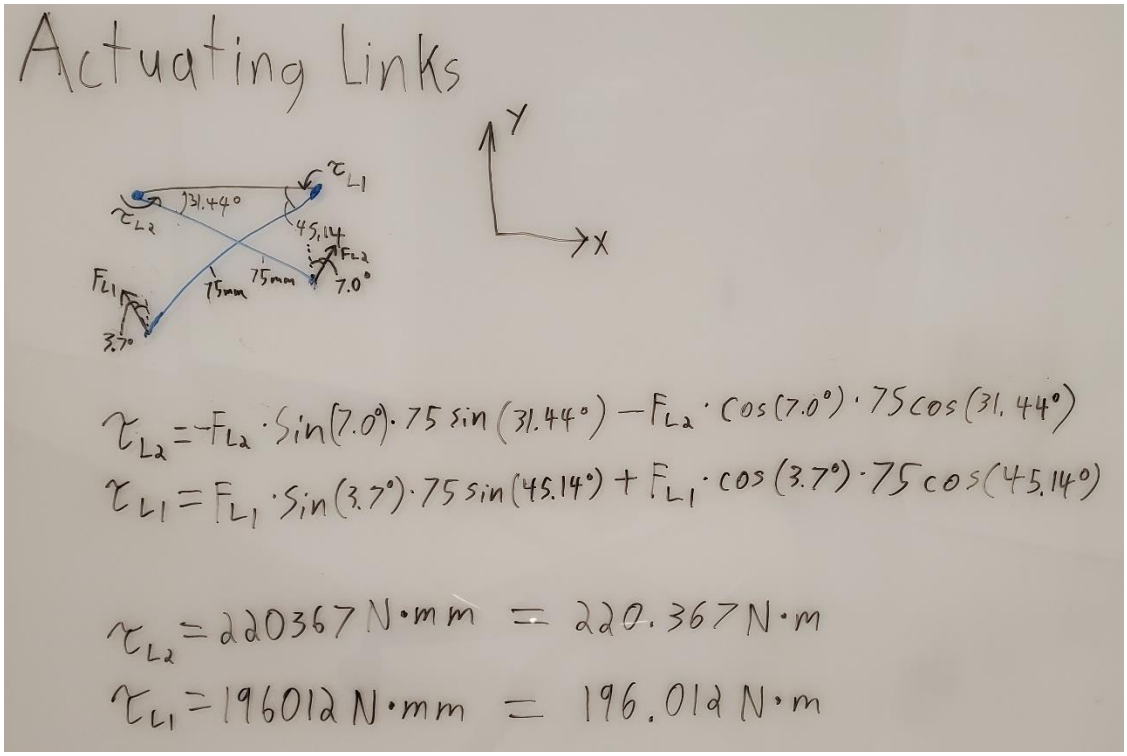


Figure 24.10: Ankle roll 6-bar linkage worst-case scenario torque calculations cont.

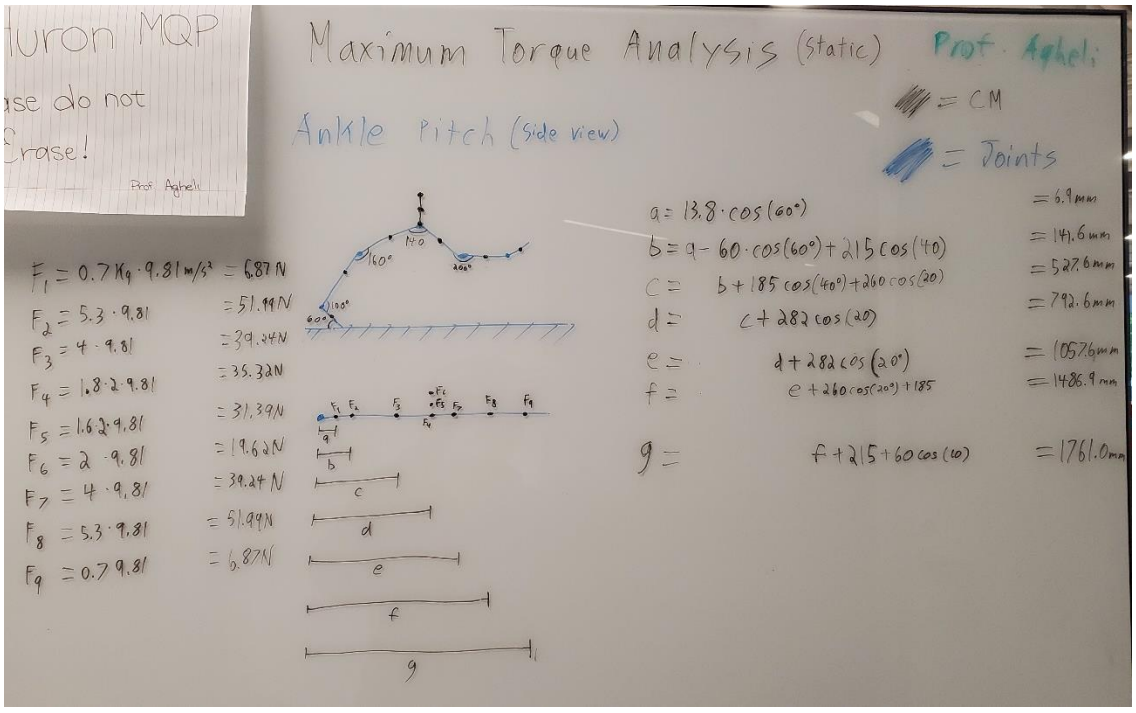


Figure 24.11: Ankle pitch worst-case scenario torque calculations

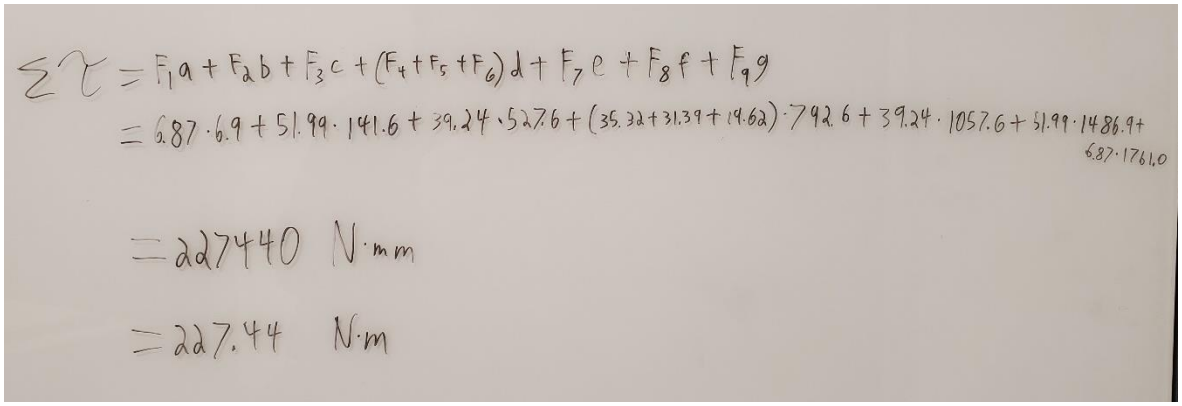


Figure 24.12: Ankle pitch worst-case scenario torque calculations cont.

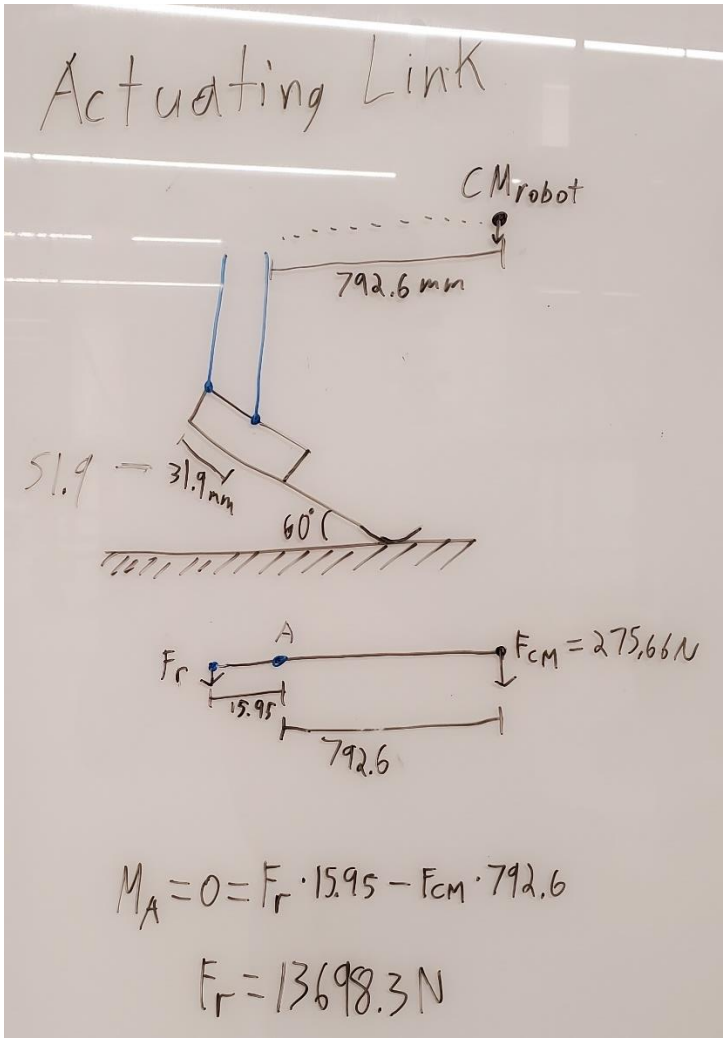


Figure 24.13: Ankle pitch 6-bar linkage worst-case scenario torque calculations

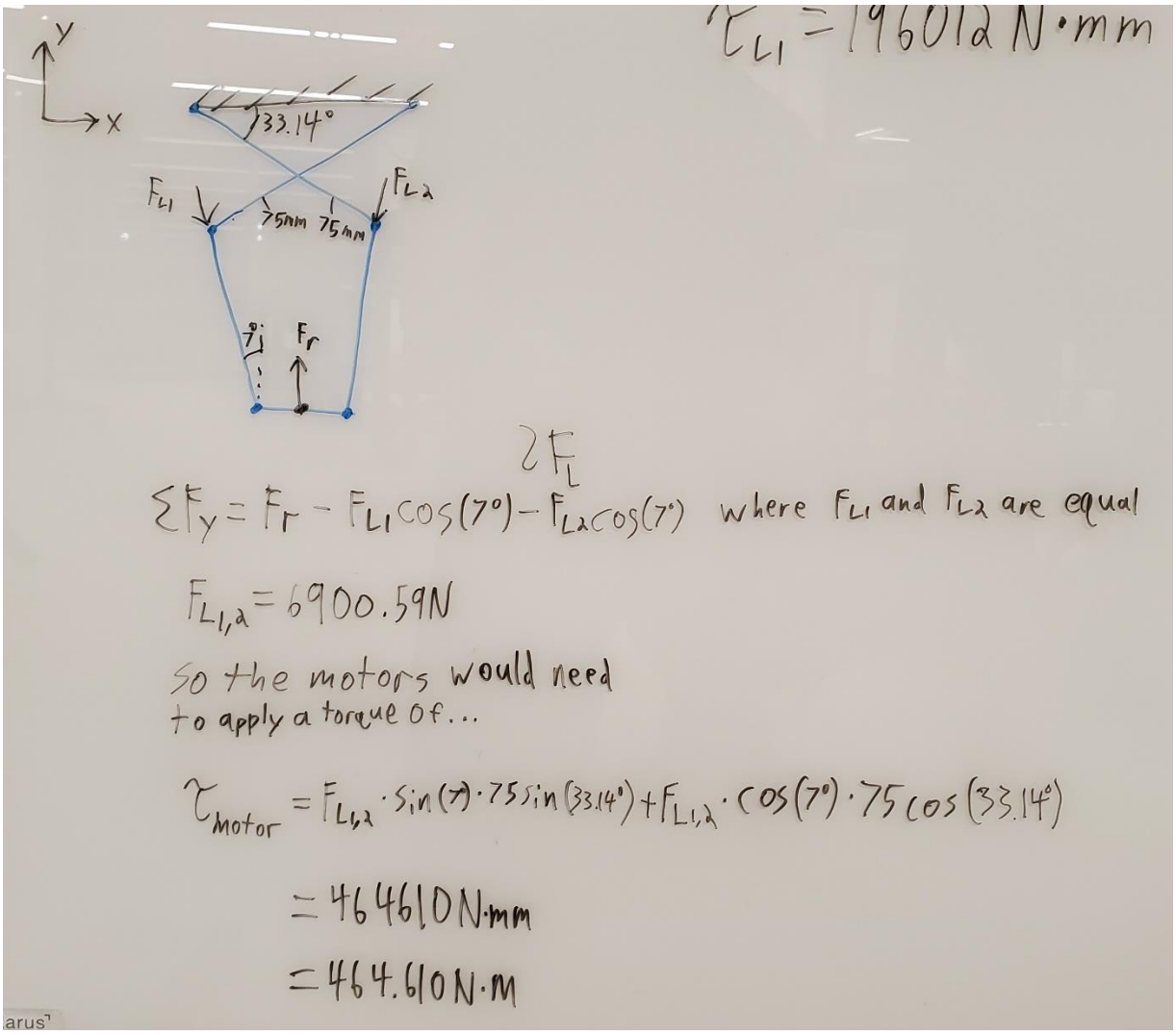


Figure 24.14: Ankle pitch 6-bar linkage worst-case scenario torque calculations cont.

24.1.3 Appendix C: Initial Weight Calculations

The density of aluminum rods is 2.7 g/cm, which is a constant used throughout the calculations.

$$Volume\ for\ a\ Cylinder:\ V = \pi * r^2 * l$$

$$Mass\ of\ a\ Cylinder:\ m = 3 * V * \rho$$

Aluminum Thigh:

$$V = \pi * \left(\frac{2.54}{2} cm\right) * (34.29\ cm) = \ 695\ cm$$

$$m = 3 * (695\ cm) * \left(2.7 \frac{g}{cm}\right) = 1.8765\ kg$$

Aluminum Calf:

$$V = \pi * \left(\frac{2.54}{2} cm\right) * (38.86\ cm) = \ 787.626\ cm$$

$$m = 3 * (787.626\ cm) * \left(2.7 \frac{g}{cm}\right) = 2.13\ kg$$

Masses of PLA shell:

$$m_{thigh} = \left(\frac{34.29}{73.15}\right) * 10231.7\ g = 4796.24\ g = 4.80\ kg$$

- $m_{calf} = \left(\frac{38.86}{73.15}\right) * \ 10231.7\ g = 5435.46\ g = 5.44\ kg$

Total Mass:

$$Thigh:\ \ m_{total} = 4.80\ kg + 1.88\ kg = 6.68\ kg$$
$$Calf:\ \ m_{total} = 5.44\ kg + 2.13\ kg = 7.57\ kg$$

Mass and Volume of the Foot:

$$V = (30\ cm) * (12\ cm) * (3\ cm) = 1080\ cm$$
$$m = V * \rho = (180\ cm) * \left(2.7 \frac{g}{cm}\right) = 2.916\ kg$$

This is based on the initial generalized size of the foot and will change with further specifications from the sensor team.

Hip Connector:

$$V = (16\ cm) * \pi * \left(\frac{2.54}{2} cm\right) = 81.073\ cm^2$$

$$m = 2 * V * \rho = (81.073\ cm) * \left(2.7 \frac{g}{cm}\right) = .438\ kg$$

24.1.4 Appendix D: Photographs and Videos of Harmonic Drive

24.1.4.1 Initial Testing of the Harmonic Drive with Two Stages

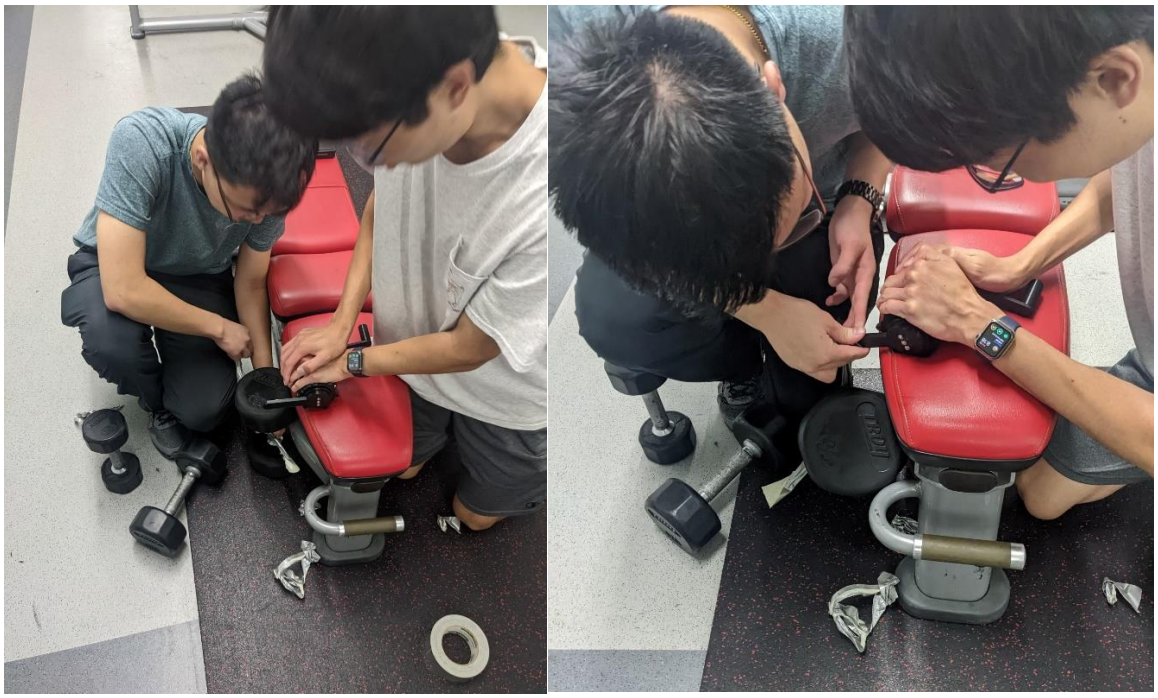


Figure 24.15. Brendyn and Jonathan setting up the testing conditions for the Harmonic Drive with a 20lb weight.

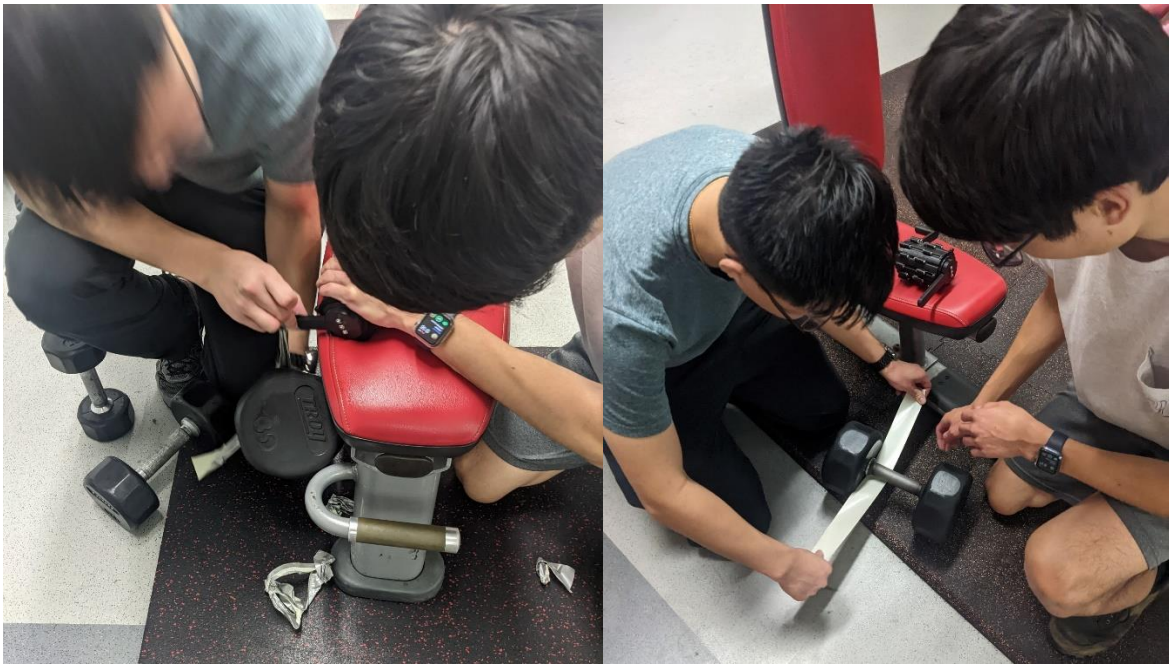


Figure 24.16. Brendyn and Jonathan attached the 35lb weight to the Harmonic Drive with duct tape.

Figure 24.17. Brendyn and Jonathan attached a 40lb weight to the Harmonic Drive.

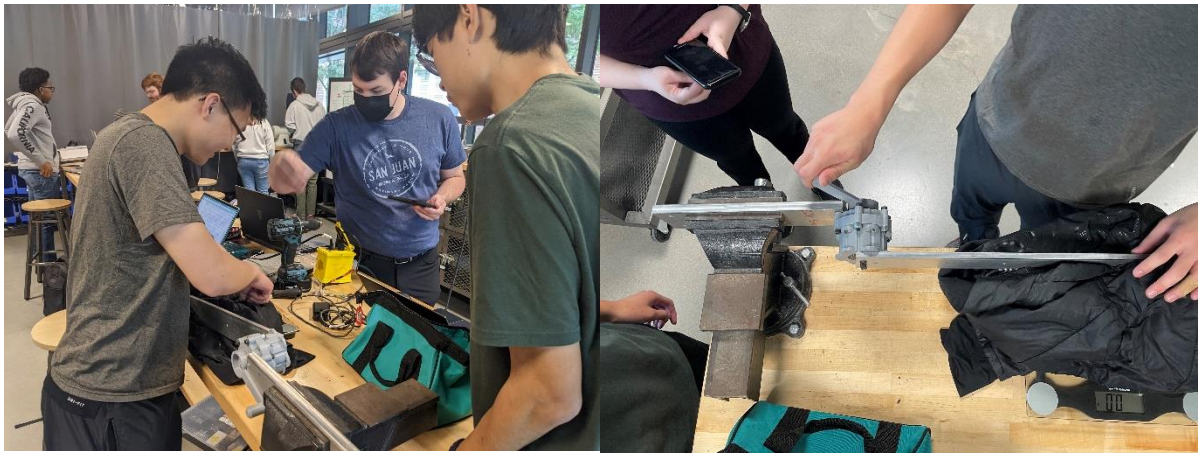


Figure 24.18. Testing of the one stage harmonic drive.

40lb Weight Test: https://youtu.be/R2AbgwErcdQ

50lb Weight Test: https://youtube.com/shorts/5_jue4xJc9E?feature=share

65lb Weight Test: https://youtube.com/shorts/jOE8PvlBb6o?feature=share

24.1.4.2 Harmonic Drive Conjugate Circular Spline Tooth Profile Calculations

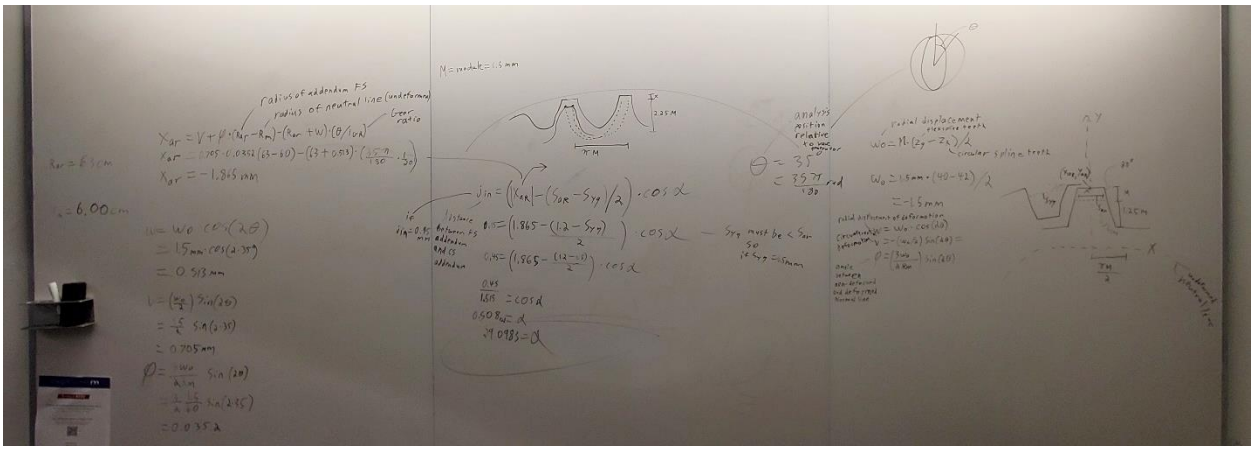


Figure 24.19: Conjugate tooth profile calculations

24.1.5 Appendix E: Pulley Calculations

24.1.5.1 MATLAB Code for 2:1 Pulley

```
%% Pulley Length Calculation
% constants
R = 45.2882;
r = 22.6314;

MinDist = 52.5;
C = MinDist:.01:120; % Range of possible distances between centers of rotation

L = [];

% Loops once for every .01mm change in distance between centers of rotation
% Values are stored in L
for i = 1:length(C)
   alpha = asin((R-r)/C(i));
   L(i) = (pi - 2*alpha)*r + (pi + 2*alpha)*R + 2*C(i)*sqrt(1-sin(alpha)^2);
end

% L was originally in mm, this converts it to inches
L_inches = 0.0393701 * L;

% find indices of viable axis-to-axis sizes
% based on McMaster Carr available belt sizes, most
% are only available in whole-number inch sizes
L_targets = find(L_inches - floor(L_inches) < .001);

% print belt size and center-to-center distance
L_inches(L_targets)
C(L_targets)
```

24.1.5.2 MATLAB Results for 2:1 Pulley

```
>> pulley_length_calculation

ans =
```

```
   13.0005   14.0001   14.0008   15.0003   16.0007   17.0000   17.0008   18.0006

ans =

   53.5500   67.2600   67.2700   80.6100   93.7700  106.8000  106.8100  119.7700
```

24.1.6 Appendix F: Custom SolidWorks PLA material properties

Elastic Modulus: 2.5GPa

Poisson's Ratio: .36

Shear Modulus: 318.9MPa

Mass Density: 1250 kg/m^3

Tensile Strength: 615MPa

Yield Strength: 2.608GPa

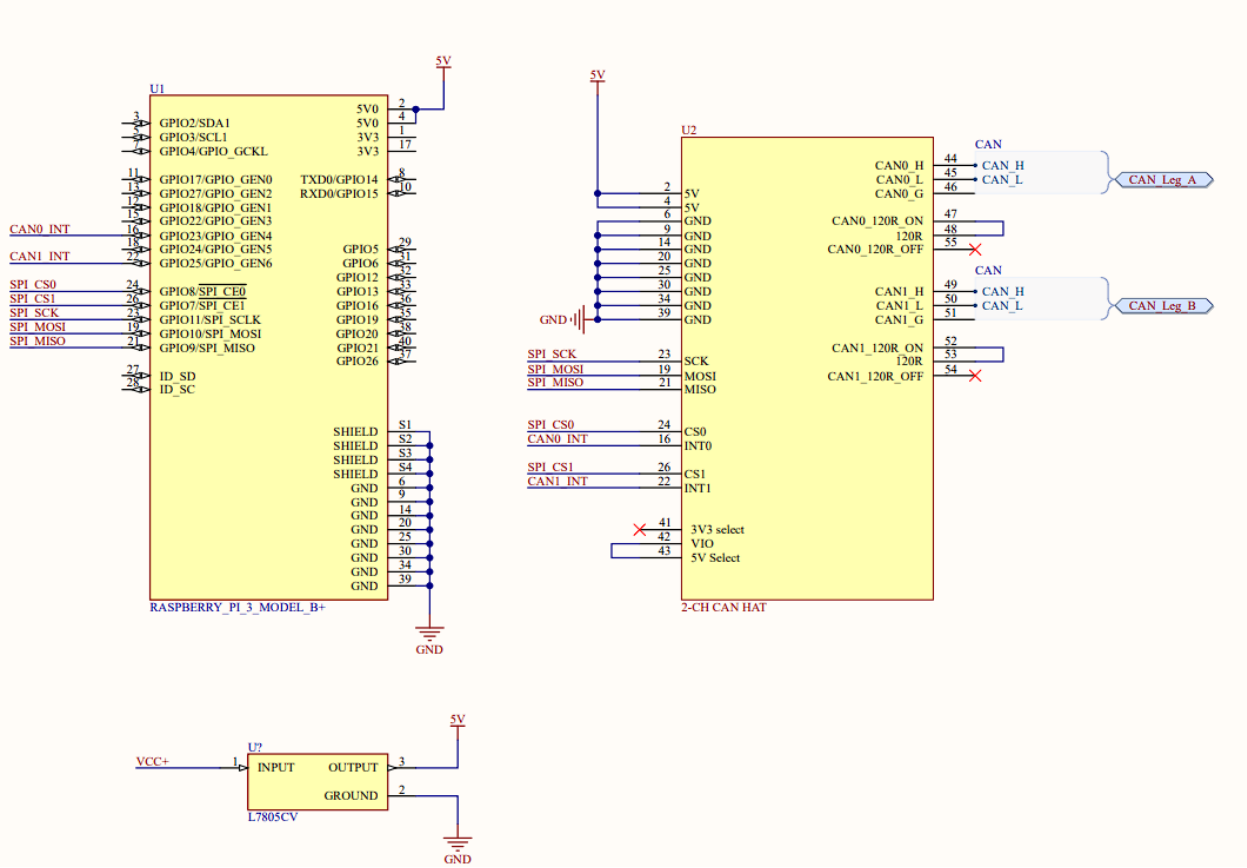
Values based on Bagheri, et. al. and "PLA Printing Specifications and Properties"

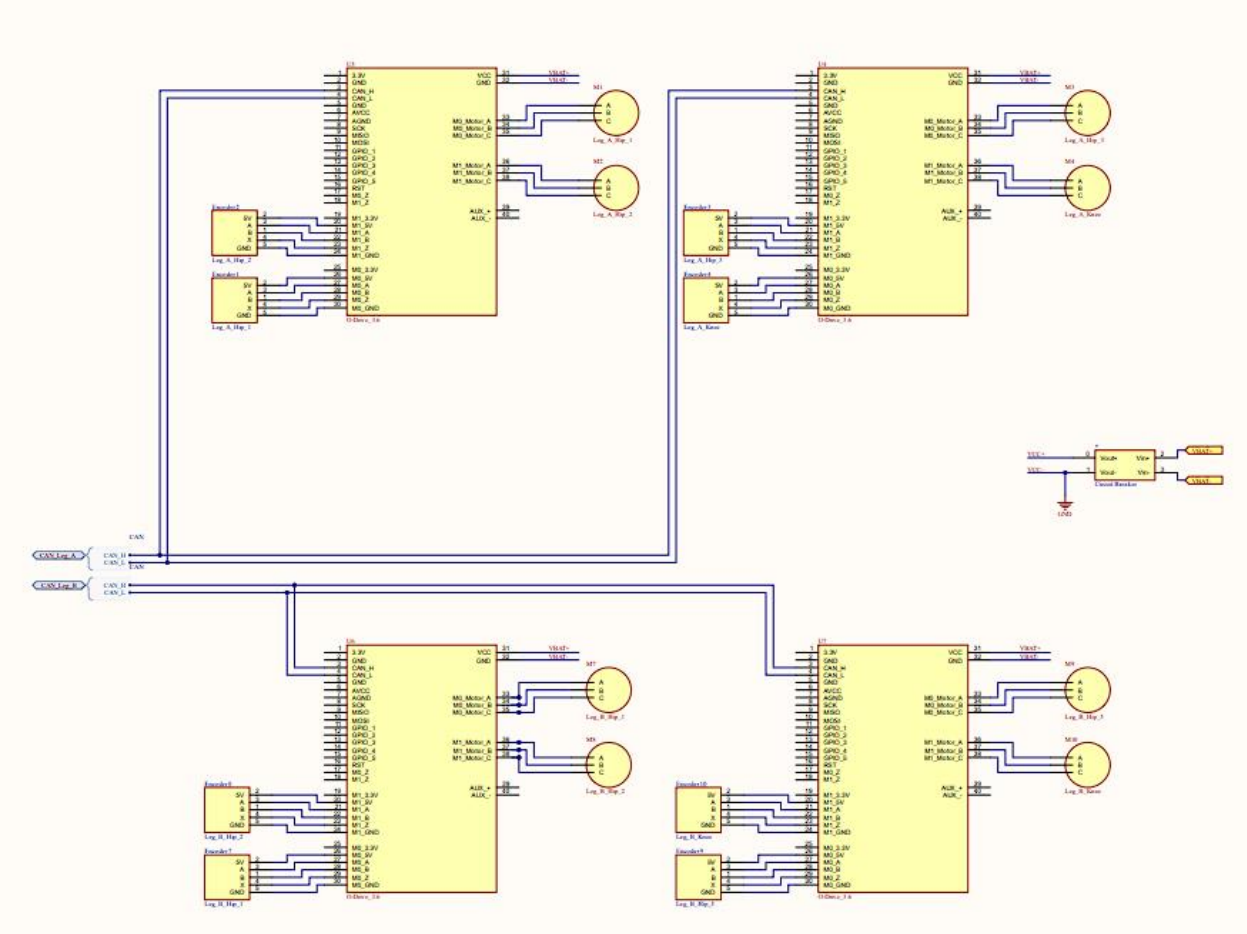
24.1.7 Appendix G: Plasma Cutting Photo

Figure 24.20: Brendyn Lining Stock into Plasma Cutter

24.2 Controls Team Appendices

24.2.1 Appendix A. Electrical Schematics of Control System





24.2.2 ODrive Configuration Code

```
odrv0.axis0.motor.config.current_lim = 60
odrv0.axis1.motor.config.current_lim = 60
odrv0.axis0.controller.config.vel_limit = 56
odrv0.axis1.controller.config.vel_limit = 56
odrv0.axis0.motor.config.calibration_current = 20
odrv0.axis1.motor.config.calibration_current = 20
odrv0.config.enable_brake_resistor = True
odrv0.axis0.motor.config.pole_pairs = 7
odrv0.axis1.motor.config.pole_pairs = 7
odrv0.axis0.motor.config.torque_constant = 8.27/190
odrv0.axis1.motor.config.torque_constant = 8.27/190
odrv0.axis0.motor.config.motor_type = MOTOR_TYPE_HIGH_CURRENT
odrv0.axis1.motor.config.motor_type = MOTOR_TYPE_HIGH_CURRENT
odrv0.axis0.encoder.config.cpr = 4096
odrv0.axis1.encoder.config.cpr = 4096
odrv0.axis0.config.can.node_id = 0 #These ID's will need to be changed for each axis
odrv0.axis1.config.can.node_id = 1
odrv0.can.config.baud_rate = 250000
odrv0.save_configuration()
```

24.2.3 MATLAB Inverse Kinematics Code

```
%% Robot Config
l0 = 155/2; %mm
l1 = 0;
l2 = 167;
l3 = 0;
l4 = 530;
l5 = 410;
l6 = 20;

L1 = Revolute('d',0,'a',0,'alpha',pi/2,'offset',0);
L2 = Revolute('d',0,'a',0,'alpha',-pi/2,'offset',-pi/2);
L3 = Revolute('d',0,'a',l4,'alpha',0,'offset',0);
L4 = Revolute('d',0,'a',l5,'alpha',0,'offset',0);
L5 = Revolute('d',0,'a',0,'alpha',pi/2,'offset',0);
L6 = Revolute('d',0,'a',l6,'alpha',0,'offset',0);
robot = SerialLink([L1,L2,L3,L4,L5,L6], 'name','huronleg');
```

```
%% Trajectory
distance = 500;
steps = 10;
%%
```

```
x = [ zeros(1,100)   zeros(1,100)             ,linspace(0,200,100)
y = [ zeros(1,100)   zeros(1,100)             , zeros(1,100)
z = [zeros(1,100)    linspace(1000,800,100)    ,  ones(1,100) * 800

% Orientation = [0 1 0; 0 0 -1; -1 0 0];

Orientation = [0 0 1 ; 0 1 0 ; -1 0 0 ];
```

```
%%
        th4 = atan2(-sqrt(1-D^2),D);
        th5 = atan2((sqrt(px+l6)^2+py^2),-pz) - atan2(sin(th4)*l4,cos(th4)*l4+l5)
        th6 = atan2((-px-l6),py) ;
        if (cos(th4+th5)*l4+cos(th5)*l5)<0
            th6=th6+pi
        end

        th2 = atan2(  sqrt(1-(sin(th6)*ax+cos(th6)*ay)^2),sin(th6)*ax+cos(th6)*ay

        th1 = atan2(-sin(th6)*sx-cos(th6)*sy, -sin(th6)*nx-cos(th6)*ny) ;
        if sin(th2)<0
            th1=th1+pi;
        end
        th345=atan2(az, cos(th6)*ax-sin(th6)*ay);
        if (sin(th2)<0)
            th345=th345+pi;
        end


        th3 = th345 - th4 - th5  -pi ;

        q = [th1;th2;th3;th4;th5;th6];
```


```
%%
        th4 = atan2(-sqrt(1-D^2),D);
        th5 = atan2((sqrt(px+l6)^2+py^2),-pz) - atan2(sin(th4)*l4,cos(th4)*l4+l5)
        th6 = atan2((-px-l6),py) ;
        if (cos(th4+th5)*l4+cos(th5)*l5)<0
            th6=th6+pi
        end

        th2 = atan2(  sqrt(1-(sin(th6)*ax+cos(th6)*ay)^2),sin(th6)*ax+cos(th6)*ay

        th1 = atan2(-sin(th6)*sx-cos(th6)*sy, -sin(th6)*nx-cos(th6)*ny) ;
        if sin(th2)<0
            th1=th1+pi;
        end
        th345=atan2(az, cos(th6)*ax-sin(th6)*ay);
        if (sin(th2)<0)
            th345=th345+pi;
        end


        th3 = th345 - th4 - th5  -pi ;

        q = [th1;th2;th3;th4;th5;th6];
```


24.3 Sensor Team Appendices

24.3.1 Appendix A. MATLAB Code

```
% plot the testing results
x = linspace(5,100,100); % weight in lbs
y = 842.84*log(x)-624.1; % voltage in mV
plot(y)
xlabel('Weight (lbs)')
ylabel('Voltage (mV)')
title('Weight vs. Voltage Relationship of Force Sensing Resistor')
legend('voltage = 843*ln(weight)-624')
```

```
% plot the predicted weight
x1 = linspace(0,3200,1000); % voltage (mV)
y1 = exp((x1+624)/842); % weight (lbs)
plot(x1, y1)
xlabel('Voltage (mV)')
ylabel('Predicted Weight (lbs)')
title('Voltage vs. Predicted Weight on Force Sensing Resistor')
```

24.3.2 Appendix B: Reactive Balancing

```
int LBLfsrPin = A0;  // botom left on left foot FSR in at A0
int LBRfsrPin = A1;  // bottom right on left foot FSR in at A1
int LULfsrPin = A2;  // upper left on left foot FSR in at A2
int LURfsrPin = A3;  // upper right on left foot FSR in at A3

int RBLfsrPin = A4;  // bottom left on right foot FSR in at A4
int RBRfsrPin = A5;  // bottom right on right foot FSR in at A5
int RULfsrPin = A7;  // upper left on right foot FSR in at A6
int RURfsrPin = A6;  // upper right on right foot FSR in at A7

// variables for force matrix
const int rows = 2;                                  // 2 rows --> front and back
const int columns = 2;                               // 2 columns --> left and
right side of foot on each foot
float FL[rows][columns] = { { 0, 0 }, { 0, 0 } };  // initialize F to 0 for the
left foot
float FR[rows][columns] = { { 0, 0 }, { 0, 0 } };  // initialize F to 0 for the
right foot

// array for forces over threshold
float r[4] = { 0, 0, 0, 0 };  // proportion of maximum leg distance which robot
must move

// max distance each leg can move
float DMaxLeft = 210;
float DMaxRight = 210;

int Decision = 0;

void readFSR() {
  // read analog output of all FSRs and convert to lb reading
  // read analog output for all FSRs on left foot
  float LBLfsrReading1 = analogRead(LBLfsrPin);  // bottom left FSR
  float LBRfsrReading1 = analogRead(LBRfsrPin);  // bottom right FSR
  float LULfsrReading1 = analogRead(LULfsrPin);  // upper left FSR
  float LURfsrReading1 = analogRead(LURfsrPin);  // upper right FSR
```

```
// read analog output for all FSRs on right foot
float RBLfsrReading1 = analogRead(RBLfsrPin);  // bottom left FSR
float RBRfsrReading1 = analogRead(RBRfsrPin);  // bottom right FSR
float RULfsrReading1 = analogRead(RULfsrPin);  // upper left FSR
float RURfsrReading1 = analogRead(RURfsrPin);  // upper right FSR
/*
* analog voltages range from 1 to 1023 which maps to 0V to 5V (=5000mV)
* define F using voltages
* F =  {{UL,  UR},
*       {BL,  BR}}
*/
FL[0][0] = map(LULfsrReading1, 0, 1023, 0, 5000);  // upper left FSR
FL[0][1] = map(LURfsrReading1, 0, 1023, 0, 5000);  // upper right FSR
FL[1][0] = map(LBLfsrReading1, 0, 1023, 0, 5000);  // bottom left FSR
FL[1][1] = map(LBRfsrReading1, 0, 1023, 0, 5000);  // bottom right FSR

// convert left foot voltages to lbs
for (int i = 0; i < 2; i++) {    // loop through each row
  for (int j = 0; j < 2; j++) {  // loop through each column
    float voltage = FL[i][j];
    if (voltage < 44) {
      FL[i][j] = 0;
    } else if (voltage >= 44 && voltage < 1800) {
      FL[i][j] = 0.0128 * voltage - 0.5595;
    } else {
      FL[i][j] = 0.0194 * voltage - 21.93;
    }
  }
}

// convert right foot voltages to lbs
FR[0][0] = map(RULfsrReading1, 0, 1023, 0, 5000);  // upper left FSR
FR[0][1] = map(RURfsrReading1, 0, 1023, 0, 5000);  // upper right FSR
FR[1][0] = map(RBLfsrReading1, 0, 1023, 0, 5000);  // bottom left FSR
FR[1][1] = map(RBRfsrReading1, 0, 1023, 0, 5000);  // bottom right FSR

for (int i = 0; i < 2; i++) {    // loop through each row
  for (int j = 0; j < 2; j++) {  // loop through each column
    float voltage = FL[i][j];
    if (voltage < 44) {
      FR[i][j] = 0;
    } else if (voltage >= 44 && voltage < 1800) {
      FR[i][j] = 0.0128 * voltage - 0.5595;
    } else {
      FR[i][j] = 0.0194 * voltage - 21.93;
```

```
      }
    }
  }
}

struct maxForceInformation {
  // struct for max force of each foot
  float maxVal;     // max force
  int maxRowIndex;  // max force row
  int maxColIndex;  // max force column
};

struct maxForceInformation maxForce(float forceArray[2][2]) {
  // determine the maxForce location
  // pass in array of forces for left foot or right foot and return struct
containing information on the max force
  maxForceInformation MFI;
  MFI.maxVal = 0;       // maximum val initalized to 0
  MFI.maxRowIndex = 0;  // maximum val initalized to F[0,0]
  MFI.maxColIndex = 0;
  for (int i = 0; i < 2; i++) {             // loop through each row
    for (int j = 0; j < 2; j++) {           // loop through each column
      if (forceArray[i][j] > MFI.maxVal) {  // if F[i][j] is more than the maxVal
then save new maxVal and location
        MFI.maxVal = forceArray[i][j];
        MFI.maxRowIndex = i;
        MFI.maxColIndex = j;
      }
    }
  }
  return MFI;
}

float minForce(float forceArray[2][2]) {
  // determine the minimum force
  // pass in array of forces for left foot or right foot and return minimum force
  float minVal = 100;
  for (int i = 0; i < 2; i++) {         // loop through each row
    for (int j = 0; j < 2; j++) {       // loop through each column
      if (forceArray[i][j] < minVal) {  // if F[i][j] is more than the maxVal
then save new maxVal and location
        minVal = forceArray[i][j];
      }
    }
  }
```

```
  return minVal;
}

float getSum(float forceArray[2][2]) {
  // get the sum of the forces on one foot
  readFSR();
  float oneSideSum = 0;                // initalize sum to 0
  for (int i = 0; i < 2; i++) {        // loop through each row
    for (int j = 0; j < 2; j++) {      // loop through each column
      oneSideSum += forceArray[i][j];  // sum of all forces
    }
  }
  return oneSideSum;
}

float getOneSideAverage(float forceArray[2][2]) {
  // get average force on one foot
  // find sum of forces on foot
  float oneSideSum = getSum(forceArray);
  // calculate average
  float oneSideAverage;
  oneSideAverage = oneSideSum / 4;  // average force
  return oneSideAverage;
}

float getAverageForce(float forceArray[2][2], float forceArray2[2][2]) {
  // find average force between both feet
  // sum of forces for each foot
  float forceSum1 = getSum(forceArray);
  float forceSum2 = getSum(forceArray2);
  // find average force
  float averageForce = (forceSum1 + forceSum2) / 8;  // average force
  return averageForce;
}

float getFFSM(float forceArray[2][2], float forceArray2[2][2]) {
  // calculate FFSM for left foot
  float F1 = forceArray[0][0];
  float F2 = forceArray[0][1];
  float F3 = forceArray[1][0];
  float F4 = forceArray[1][1];
  float F5 = forceArray2[0][0];
  float F6 = forceArray2[0][1];
  float F7 = forceArray2[1][0];
  float F8 = forceArray2[1][1];
```

```
  // get average force
  float ave = getAverageForce(forceArray, forceArray2);
  float FFSM = F1 / ave * F2 / ave * F3 / ave * F4 / ave * F5 / ave * F6 / ave *
F7 / ave * F8 / ave;
  return FFSM;
}

float calculateTheta(struct maxForceInformation MFI, float forceArray[2][2]) {
  // define angle of external perturbation based on one foot
  float adjacent1Left;
  float adjacent2Left;
  if (MFI.maxRowIndex == MFI.maxColIndex) {
    adjacent1Left = forceArray[1][0];
    adjacent2Left = forceArray[0][1];
  } else {
    adjacent1Left = forceArray[0][0];
    adjacent2Left = forceArray[1][1];
  }

  // calculate eta, phi, and theta
  float eta = (adjacent1Left - adjacent2Left) / (adjacent1Left + adjacent2Left -
(2 * MFI.maxVal));
  float phi = 90;
  float theta = eta / 2 * phi;
  return theta;
}

float getReferenceAngle(struct maxForceInformation MFI, float theta) {
  float referenceAngle = 0;
  if (MFI.maxRowIndex == 0 && MFI.maxColIndex == 0) {
    referenceAngle = 315;
  } else if (MFI.maxRowIndex == 0 && MFI.maxColIndex == 1) {
    referenceAngle = 45;
  } else if (MFI.maxRowIndex == 1 && MFI.maxColIndex == 0) {
    referenceAngle = 225;
  } else if (MFI.maxRowIndex == 1 && MFI.maxColIndex == 1) {
    referenceAngle = 135;
  } else {
    referenceAngle = 0;
  }
  float angle = referenceAngle + theta;
  return angle;
}
```

```
float getAngle(struct maxForceInformation MFI, float forceArray[2][2], struct
maxForceInformation MFI2, float forceArray2[2][2]) {
  // define angle of external perturbation based on left foot
  float theta1 = calculateTheta(MFI, forceArray);
  float angle1 = getReferenceAngle(MFI, theta1);

  // define angle of external perturbation based on right foot
  float theta2 = calculateTheta(MFI2, forceArray2);
  float angle2 = getReferenceAngle(MFI2, theta2);

  // angle of external perturbation for whole system
  float aveAngle;
  aveAngle = (angle1 + angle2) / 2;
  return aveAngle;
}

float oppositeAngle(float angle) {
  float oppAngle;
  if (0 <= angle < 180) {
    oppAngle = angle + 180;
  } else {
    oppAngle = angle - 180;
  }
  return oppAngle;
}

float distanceToMove(float forceArray[2][2], float threshold, float fmax, float
fmin, float Dmax) {
  // calculate ri for each force above the threshold
  int k = 0;
  float r[4] = { 0, 0, 0, 0 };
  for (int i = 0; i < 2; i++) {    // loop through each row
    for (int j = 0; j < 2; j++) {  // loop through each column
      if (forceArray[i][j] > threshold) {
        r[k] = (forceArray[i][j] - fmin) / (fmax - fmin);  // calculate ri
        k += 1;                                            // num of FSRs over
threshold
      }
    }
  }

  // average ri to find Ri for the foot
  float rSum = 0;
  for (int i = 0; i < k; i++) {
    rSum += r[i];
```

```
  }
  float R = rSum / k;
  if (R > 1 || isnan(R)) {
    R = 1;
  }

  // distance to move
  float distance;
  return distance = R * Dmax;
}

int makeDecision(float leftAve, float rightAve, float FFSM, float angle) {
  if ((leftAve == 0) || rightAve == 0) {
    // Serial.print("One foot is off the ground, ");
    Decision = 0;            // cannot use FFSM
  } else if (FFSM >= 0.7) {  //robot is balanced
    // Serial.print("Robot is stable, ");
    Decision = 1;  // foot is stable
  }
  else if (FFSM < 0.7) {  //NOT STABLE
    // Serial.print("take a step in direction: ");
    if (angle >= 180 && angle < 360) {
      // move left leg
      Decision = 2;
    } else if (angle < 180) {
      // move right leg
      Decision = 3;
    }
  } else {
    Serial.print("error");
  }
  return Decision;
}

// set up
void setup(void) {
  Serial.begin(9600);  // We'll send debugging information via the Serial monitor
}
void loop(void) {
  // read forces on feet
  readFSR();

  // find maximum force for left foot
  maxForceInformation leftFootMaxForce;
  leftFootMaxForce = maxForce(FL);
```

```
  // find maximum force for right foot
  maxForceInformation rightFootMaxForce;
  rightFootMaxForce = maxForce(FR);

  // find minimum force for left foot
  float leftFootMinForce = minForce(FL);

  // find minimum force for right foot
  float rightFootMinForce = minForce(FR);

  // find average force for left foot
  float leftAve = getOneSideAverage(FL);

  // find average force for right foot
  float rightAve = getOneSideAverage(FR);

  // total FFSM
  float FFSM = getFFSM(FL, FR);

  // angle robot is leaning
  float angle = getAngle(leftFootMaxForce, FL, rightFootMaxForce, FR);

  // get parameters for calculating the distance to move
  float forceThreshold = (getSum(FL) + getSum(FR)) / 4;  // minimum stability
requirement
  float fmax = (leftFootMaxForce.maxVal > rightFootMaxForce.maxVal) ?
leftFootMaxForce.maxVal : rightFootMaxForce.maxVal;
  float fmin = (leftFootMinForce < rightFootMinForce) ? leftFootMinForce :
rightFootMinForce;

  // make reactive decision
  int decision = makeDecision(leftAve, rightAve, FFSM, angle);

  Serial.print(millis());
  Serial.print(",");
  Serial.print(decision);
  Serial.print(",");
  if (decision == 0) {
    // Serial.println("Robot is in the air, no FSRs are touching the ground.");
    Serial.println(0);
  } else if (decision == 1) {
    // Serial.println("Robot is stable");
  } else if (decision == 2) {
    // Serial.print("Move LEFT foot this distance: ");
```

```
    float distanceLeft = distanceToMove(FL, forceThreshold, fmax, fmin,
DMaxLeft);
    Serial.println(distanceLeft);
  } else if (decision == 3) {
    // Serial.print("Move RIGHT foot this distance: ");
    float distanceRight = distanceToMove(FR, forceThreshold, fmax, fmin,
DMaxRight);
    Serial.println(distanceRight);
  } else {
    Serial.println("Error");
  }
}  // last parentheses
```

24.3.3 Appendix C: ZMP Research Notes

Balance Control of Humanoid Robot for HuroSot

Citation: Bum-Joo Lee, Yong-Duk Kim, Jong-Hwan Kim, "BALANCE CONTROL OF HUMANOID ROBOT FOR HUROSOT", *IFAC Proceedings Volumes*, Volume 38, Issue 1, 2005, pp. 215-220, ISSN 1474-6670, ISBN 9783902661753, https://doi.org/10.3182/20050703-6-CZ-1902.02088.

The paper, *Balance Control of Humanoid Robot for HuroSot*, focuses on balancing control of a humanoid robot and uses the upper body motion and swinging two arms to balance. The upper body is modeled as an inverted pendulum for an on-line compensation. As the force sensitive resistors attached to the soles of the feet obtain ZMP error, the upper body is supposed to compensate for this. The upper body also helps with overall balance. A gait is generated in off-line and it is compensated in on-line control. The off-line gait is generated using a spline method. Since the gait satisfied the ZMP criterion, roll and pitch moments of the robot are zeros. The yawing moment is canceled out so the robot can slip along with a vertical axis. Therefore, the on-line balance control is needed for stable walking. To derive an on-line compensation based on ZMP its upper body is modeled as an inverted pendulum. The ZMP is obtained from the FSRs attached to the sole.

The first phase is gait generation. Off-line gait is generated to satisfy the ZMP condition using a cubic spline while the upper body is not moving. To do this, points are selected from the

walking condition such as a walking period, a step size, a maximum foot height, etc. Then a whole trajectory is made using a cubic spline interpolation.

The second phase is yawing moment cancellation. A general ZMP equation is shown below.

inmi(ri-rp)*ri..=mi(ri-rp)*G+T
mi is the mass of ith link
ri is a vector between the origin and a COM of ith link
rp is a vector from the origin to the ZMP
G is a gravity vector
T is a torque vector applied to the ZMP

From the above ZMP equation, the following can be used.

$$X_{ZMP} = \frac{\sum_i^n m_i(\ddot{z}_i + g)x_i - \sum_i^n m_i z_i \ddot{x}_i}{\sum_i^n m_i(\ddot{z}_i + g)}$$
$$Y_{ZMP} = \frac{\sum_i^n m_i(\ddot{z}_i + g)y_i - \sum_i^n m_i z_i \ddot{y}_i}{\sum_i^n m_i(\ddot{z}_i + g)}$$

A yawing moment equation is also obtained.

$$T_Z = \sum_i^n m_i(x_i - x_{ZMP})\ddot{y}_i - (y_i - y_{ZMP})\ddot{x}_i$$

The team then used the following parameters for gait generation during their experiments: Total step time, Supporting time, Rising time, Swing time, Landing time, Step size, and Hip height.

Computational Efficient Balance Control for a Lightweight Biped Robot with Sensor Based ZMP Estimation

Citation: M. Folgheraiter et al., "Computational Efficient Balance Control for a Lightweight Biped Robot with Sensor Based ZMP Estimation," 2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids), 2018, pp. 232-237, doi: 10.1109/HUMANOIDS.2018.8625016.

- The paper proposes an algorithm for balance control of a bipedal robot. A LIP model of the robot is combined with the ZMP calculation to derive a joint space control action based on a PD controller.

- Three main categories of balancing algorithms: (1) kinematic / dynamic model of robot used to calculate ZMP, (2) estimates ZMP from real-time measurements from IMUs and F/T sensors, (3) combination of (1) and (2) involving data fusion.
- ZMP represents point on the floor where horizontal components of net moment is 0
- Dynamic model algorithms are held back due to model inaccuracies and intense computational resource requirements
    - Some researchers used Linear Inverted Pendulum (LIP) model in combo with ZMP calcs to find "stabilizing trajectory of Center of Mass" (CoM)
- ZMP for human gait moves outside the sole when on one leg then transitions to outside the other when walking
- Authors relied on simulation when testing out control algorithms before using the real thing
    - V-REP for robot dynamics and visualization, MatLab for control algorithms

ZMP calculation used (y-direction (lateral)):

Finally, the position of the ZMP along the y-axis can be found as:

$$y_{zmp} = \frac{l^2}{g}\ddot{\theta} + \frac{cl^2}{mg}\dot{\theta} + \frac{kl^2}{mg}\theta + l\theta. \qquad (7)$$

PID control algorithm:

$$\theta_c(t) = k_p e(t) + k_i \int_0^t e(\tau)d\tau + k_d \dot{e}(t) \qquad (8)$$

where $e(t) = -y_{zmp}$.

Theta(c) is the correctional angle adjustment, which is added to the current angle to calculate desired angle in the control algorithm.

- Calculating the ZMP in this way is very computationally intensive, the researchers instead used force sensors on the feet to estimate Center of Pressure

Y CoP and X CoP Calcs using four force sensors in corners of feet:

$$y_{CoP} = \frac{\sum_{n=1}^{4}(F_i * y_i)}{\sum_{n=1}^{4} F_i},\ x_{CoP} = \frac{\sum_{n=1}^{4}(F_i * x_i)}{\sum_{n=1}^{4} F_i}. \quad (9)$$

Just average right and left measurements to find everall CoP

- In real robot, balance control implemented in C on Raspberry Pi 3, which ended up damping oscillations by 60% when an external force was applied

Dynamic Balance Control for Biped Robot Walking Using Sensor Fusion, Kalman Filter, and Fuzzy Logic

Citation: T. -H. S. Li, Y. -T. Su, S. -H. Liu, J. -J. Hu and C. -C. Chen, "Dynamic Balance Control for Biped Robot Walking Using Sensor Fusion, Kalman Filter, and Fuzzy Logic," in IEEE Transactions on Industrial Electronics, vol. 59, no. 11, pp. 4394-4408, Nov. 2012, doi: 10.1109/TIE.2011.2175671.

- Fundamental research for biped robots walking ability by using ZMP criterion.
    - To derive ZMP measurements we can use
        - Measure derived from position and acceleration
        - Force sensors of soles
            - Using geometry of force sensor positions on soles during touching ground we can find ZMP
- A dynamic balance control is made up of a Kalman filter and fuzzy motion controller
- Popular choice for ZMP position is set in the middle of the supporting foot sole, for a more natural walk, the ZMP trajectory is on a more piecewise continuous function with respect of time

$$
\begin{aligned}
p_x &= \frac{1}{\sum_{i=1}^{8} f_i}[(f_2+f_4)l_{\text{width}} + (f_5+f_7)(l_{\text{width}}+d_x) + (f_6+f_8)(2l_{\text{width}}+d_x)] \\
&= \frac{(f_2+f_4)l_{\text{width}} + \sum_{i=5}^{8} f_i(l_{\text{width}}+d_x) + (f_6+f_8)l_{\text{width}}}{\sum_{i=1}^{4} f_i + \sum_{i=5}^{8} f_i} \\
&= \frac{(f_2+f_4)l_{\text{width}} + \sum_{i=5}^{8} f_i(l_{\text{width}}+d_x) + (f_6+f_8)l_{\text{width}}}{\sum_{i=1}^{4} f_i + \sum_{i=5}^{8} f_i} + \frac{\left(\frac{\sum_{i=5}^{8} f_i}{\sum_{i=1}^{4} f_i}(f_2+f_4)l_{\text{width}} - \frac{\sum_{i=5}^{8} f_i}{\sum_{i=1}^{4} f_i}(f_2+f_4)l_{\text{width}}\right)}{\sum_{i=1}^{4} f_i + \sum_{i=5}^{8} f_i} \\
&= \frac{\frac{\sum_{i=1}^{4} f_i + \sum_{i=5}^{8} f_i}{\sum_{i=1}^{4} f_i}(f_2+f_4)l_{\text{width}} + \sum_{i=5}^{8} f_i(l_{\text{width}}+d_x) + (f_6+f_8)l_{\text{width}}}{\sum_{i=1}^{4} f_i + \sum_{i=5}^{8} f_i} \\
&= \frac{1}{\sum_{i=1}^{4} f_i}(f_2+f_4)l_{\text{width}} + \frac{\sum_{i=5}^{8} f_i}{\left(\sum_{i=1}^{4} f_i + \sum_{i=5}^{8} f_i\right)}(l_{\text{width}}+d_x) - \frac{\sum_{i=5}^{8} f_i}{\left(\sum_{i=1}^{4} f_i + \sum_{i=5}^{8} f_i\right)}\left(\frac{(f_2+f_4)l_{\text{width}}}{\sum_{i=1}^{4} f_i} + \frac{(f_6+f_8)l_{\text{width}}}{\sum_{i=5}^{8} f_i}\right) \\
&= p_{\text{left\_}x} + \frac{\sum_{i=5}^{8} f_i}{\sum_{i=1}^{8} f_i}(l_{\text{width}} - p_{\text{left\_}x} + d_x + p_{\text{right\_}x}) \qquad (20)
\end{aligned}
$$

Equation to find X-directional ZMP

$$
\begin{aligned}
p_y &= \frac{(f_1+f_2)l_{\text{length}} + (f_5+f_6)(l_{\text{length}}+d_y) + (f_7+f_8)d_y}{\sum_{i=1}^{8} f_i} = \frac{(f_1+f_2)l_{\text{length}} + (f_5+f_6)l_{\text{length}} + \sum_{i=5}^{8} f_i d_y}{\sum_{i=1}^{8} f_i} \\
&= \frac{\frac{\left(\sum_{i=1}^{4} f_i + \sum_{i=5}^{8} f_i\right)}{\sum_{i=1}^{4} f_i}(f_1+f_2)l_{\text{length}} + (f_5+f_6)l_{\text{length}} + \sum_{i=5}^{8} f_i d_y}{\left(\sum_{i=1}^{4} f_i + \sum_{i=5}^{8} f_i\right)} - \frac{\frac{\sum_{i=5}^{8} f_i}{\sum_{i=1}^{4} f_i}(f_1+f_2)l_{\text{length}}}{\left(\sum_{i=1}^{4} f_i + \sum_{i=5}^{8} f_i\right)} \\
&= \frac{(f_1+f_2)l_{\text{length}}}{\sum_{i=1}^{4} f_i} + \frac{\sum_{i=5}^{8} f_i}{\sum_{i=1}^{4} f_i + \sum_{i=5}^{8} f_i}\left(\frac{1}{\sum_{i=5}^{8} f_i}(f_5+f_6)l_{\text{length}}\right) + \frac{\sum_{i=5}^{8} f_i}{\sum_{i=1}^{4} f_i + \sum_{i=5}^{8} f_i}\left(\sum_{i=5}^{8} f_i d_y - \frac{(f_1+f_2)l_{\text{length}}}{\sum_{i=1}^{4} f_i}\right) \\
&= p_{\text{left\_}y} + \frac{\sum_{i=5}^{8} f_i}{\sum_{i=1}^{8} f_i}(p_{\text{right\_}y} + d_y - p_{\text{left\_}y}) \qquad (21)
\end{aligned}
$$

Equation to find Y-directional ZMP

This paper was on the usage of four force sensors on the corners of the soles of the feet to calculate the zmp which would be used to create a desired walking path. The first step was to place the force sensors in the corners of a rectangular foot. The next step would be to create the zmp which would be dynamically moving

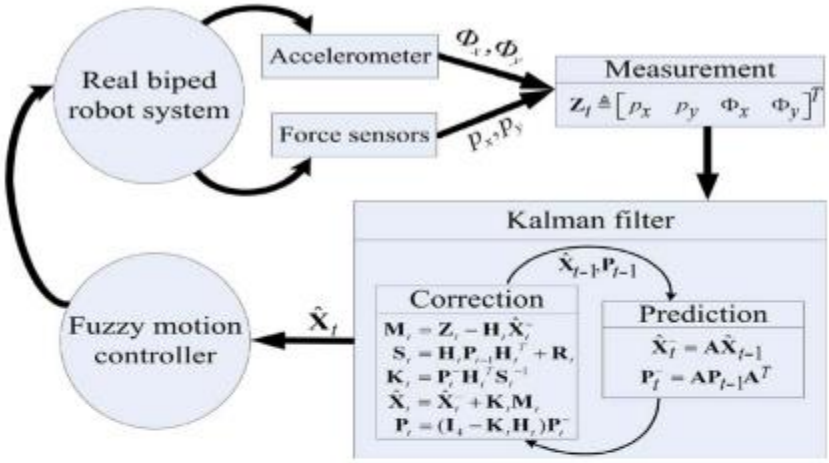


Fig. 7. Block diagram of presented KF.

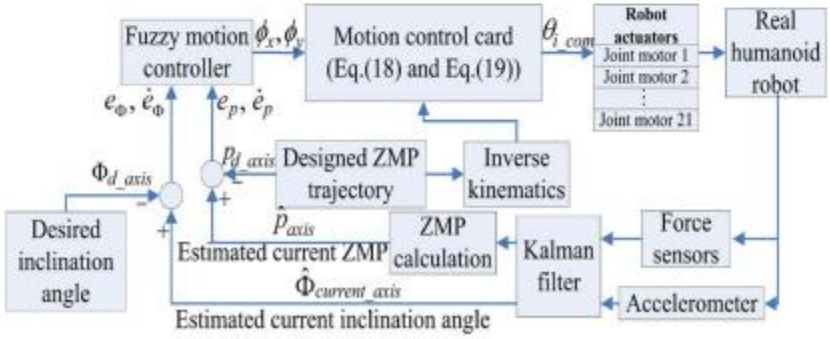


Fig. 8. Illustration of DBC.

Humanoid Robot Posture-Balance Control

Citation: H. -I. Lin and X. -A. Nguyen, "Humanoid robot posture-balance control," *2016 55th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, 2016, pp. 160-165, doi: 10.1109/SICE.2016.7749249.

The paper, *Humanoid Robot Posture-Balance Control*, presents a posture-balance control approach for users to program humanoid postures. This study develops a system which helps to program robot motion by human demonstration and maintain robot balance as well. Using a Kinect camera, the team was able to acquire information of human demonstration. A Kinect camera is a RGB-D camera that can recognize people and capture their motion over time. Based on the joint positions of the human captured by the camera, they calculated all joint angles through a vector-based geometrical method. Once the joint angles were calculated, they then mapped the human skeleton to the robot model. The team used a Denavit and Hartenberg (D-H) model to select frames of reference from the camera to create a skeleton. Center of Gravity (COG) Jacobian was used to adapt robot posture. Using the equation below, and the D-H model the body mass center position can be obtained.

$$\dot{x}_G = \frac{\sum_{i=0}^{n-1} m_i \dot{r}_{Gi}}{\sum_{i=0}^{n-1} m_i} = \frac{\sum_{i=0}^{n-1} m_i J_{Gi} \dot{\theta}}{\sum_{i=0}^{n-1} m_i}.$$

The COG is calculated by projecting COM on the ground. This is shown in the equation below.

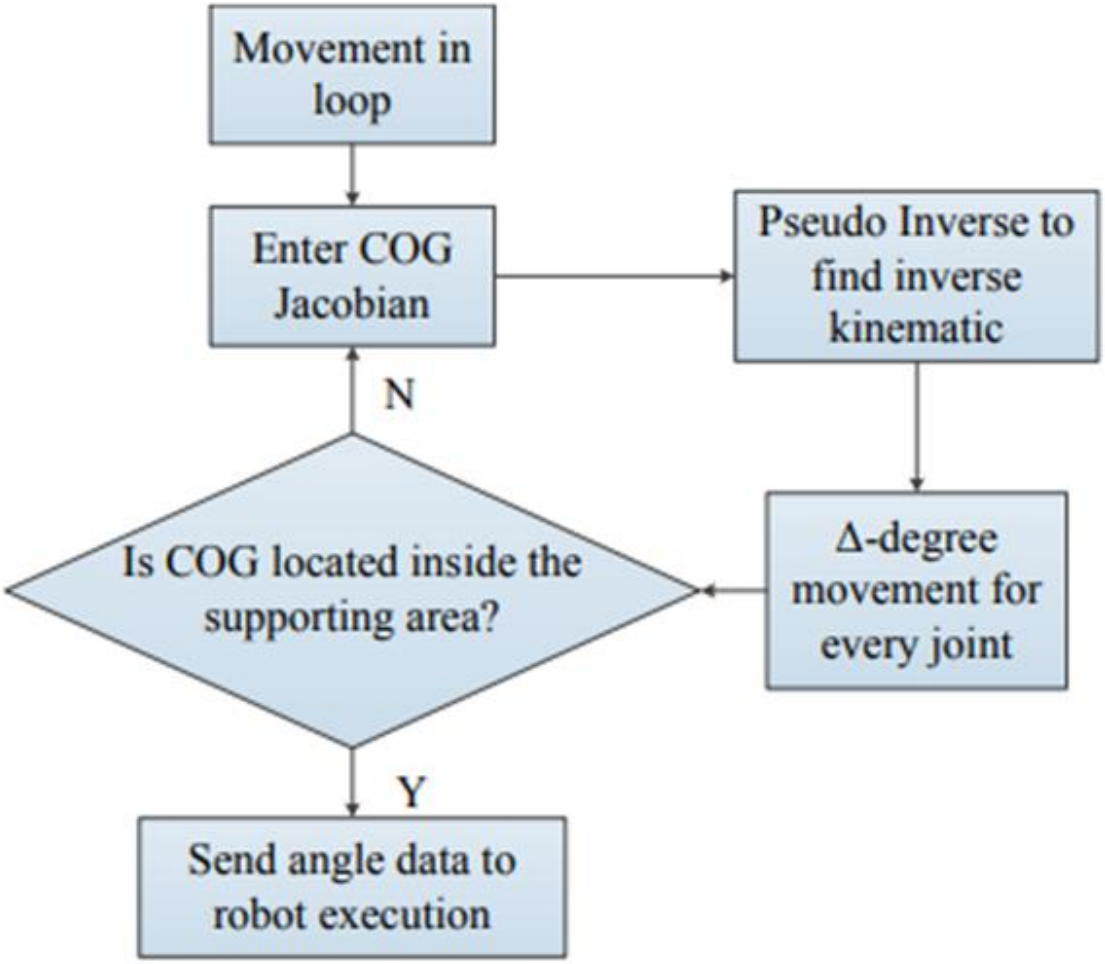
$$J_G = \frac{\partial r_{Gi}}{\partial \theta},$$

$$J_{G,i} = \left[\frac{\partial X_{COM,i}}{\partial \theta_1}, \frac{\partial X_{COM,i}}{\partial \theta_2}, \ldots, \frac{\partial X_{COM,i}}{\partial \theta_{22}} \right]$$

Where

$$\frac{\partial X_{COM,i}}{\partial \theta} = \frac{\partial \left(\left(\prod_{j=1}^{i} A_j * Rotz(\theta_i) X_{COM,i} \right) \right)}{\partial \theta}$$

To obtain the value of body motion of each joint angle, the group used inverse kinematics which is shown below.

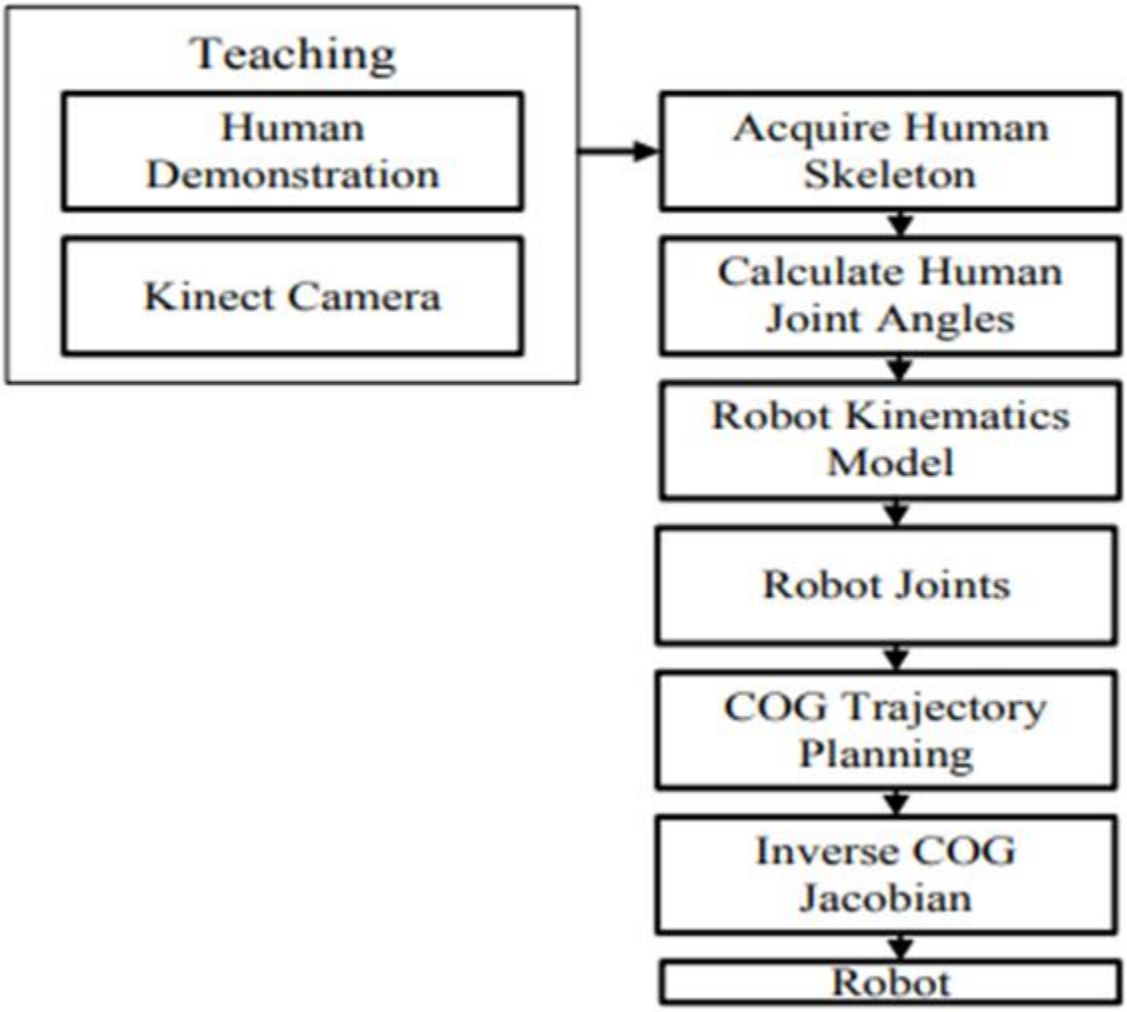


To keep the robot balanced, the robot COG should be kept within the supporting area of the foot. The closer the COG is to the boundary of the supporting area, the easier the robot falls down. There is no unilateral relationship between the balance ability and the distance (D) from the COG to the center of the supporting area so far. Therefore, they used a quality function to evaluate the degree of unbalance as shown below

$$B'\left(D(\theta), R\right) = \left(\frac{D(\theta)}{R}\right)^{k}$$

R is the distance from the balance center to the boundary of the supporting area. D(0) is the distance from the balance center to the COG. 0 is the vector of the joint angles. K is the exponent of the power function.

The steps of this development are summarized in the chart below.



Integral Control of Humanoid Balance

Citation: B. Stephens, "Integral control of humanoid balance," 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007, pp. 4020-4027, doi: 10.1109/IROS.2007.4399407.

- Two strategies to deal with balancing: (1) ankle strategy - ankle is fixed, inverted pendulum, (2) hip strategy - bending of hips results in repositioning of COM
    - Ankle - balance achieved by only torquing the ankles, Hip - balance achieved by accelerating CoM
- Unconstrained balance control (high computational complexity?)

- Linear Quadratic Regulator (LQR) counteracts torques at ankle, driving torque to zero (ie. stiff inverted pendulum)

$$\tau_{\mathrm{LQR}} = -K_{\mathrm{LQR}} \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} \tag{2}$$

- If ankle deflection is large, torque is also very large. So we also move horizontal position of CoM above ankles (like hip strategy). End up having this equation for torque of CoM:

$$\begin{aligned} \tau_{\mathrm{CM}} \quad = \quad & N - M J_x^+ \Big(\dot{J}_x \dot{\theta} + k_{\mathrm{CMp}} (x_{\mathrm{CM}} - x_{\mathrm{CM}}^d) \\ & + k_{\mathrm{CMd}} \dot{x}_{\mathrm{CM}} \Big), \end{aligned} \tag{4}$$

- Both controllers are used simultaneously
- To calculate center of pressure:

$$x_{\mathrm{CoP}} = \frac{\tau_{\mathrm{ankle}}}{F_{\mathrm{normal}}} = \frac{A\tau}{B + C\tau}, \tag{5}$$

- See paper for A, B, and C matrices definitions
- Reference torque calculated from adding LQR & CoM torques, constrained by keeping CoP inside foot
- Posture control was also implemented (for some reason I can't understand)
- In actual biped, there are three "breaking points" which the robot rotates: ankles to knee, knee to hip, hip to shoulder. Authors modeled system through double inverted pendulum (not controlling ankle to knee angle). They had a separate controller for CoM which compensated for this.

Zero-Moment Point - Thirty Fives Years of its Life

Citation: Vukobratovic, M., & Borovac, B. (2004). *Zero-moment point — thirty five years of its life*. International Journal of Humanoid Robotics. Retrieved December 16, 2022, from https://www.cs.cmu.edu/~cga/legs/vukobratovic.pdf

- The static equilibrium equations for the supporting foot

$$\mathbf{R} + \mathbf{F}_{\mathrm{A}} + \mathbf{m}_{\mathrm{s}}\mathbf{g} = 0, \tag{2}$$

$$\overrightarrow{\mathbf{OP}} \times \vec{\mathbf{R}} + \overrightarrow{\mathbf{OG}} \times \mathbf{m}_{\mathrm{s}}\mathbf{g} + \mathbf{M}_{\mathrm{A}} + M_{\mathrm{z}} + \overrightarrow{\mathbf{OA}} \times \mathbf{F}_{\mathrm{A}} = 0, \tag{3}$$

- Projecting the equations on the horizontal plane, gives

$$(\overrightarrow{\mathbf{OP}} \times \vec{\mathbf{R}})^{\mathrm{H}} + \overrightarrow{\mathbf{OG}} \times \mathbf{m}_{\mathrm{s}}\mathbf{g} + \mathbf{M}_{\mathrm{A}}^{\mathrm{H}} + (\overrightarrow{\mathbf{OA}} \times \mathbf{F}_{\mathrm{A}})^{\mathrm{H}} = 0. \tag{4}$$

- Determining ZMP position

- Step 1: Compute OP (radius vector of force acting point to origin) from equation and call it point P (computed ZMP position), still unknown if point will be in or outside foot
- Step 2: Compare the ZMP to the size of the foot, if outside the foot the force acting point is on the edge of the foot where the mechanism rotation about the foot toe will be initiated by the unbalanced moment (intensity of this is dependent on the