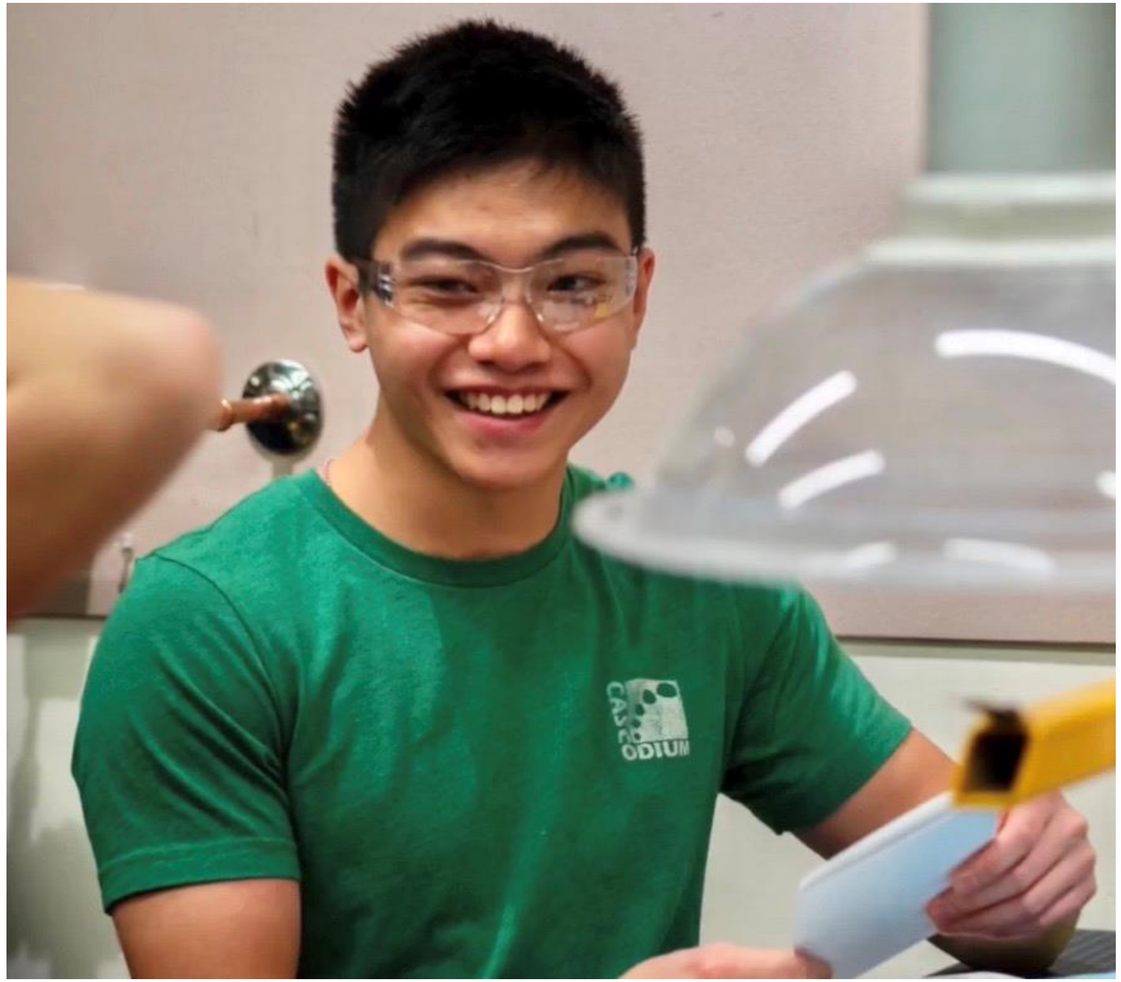
Michael Lu

EECS & MechE @ MIT ‘23

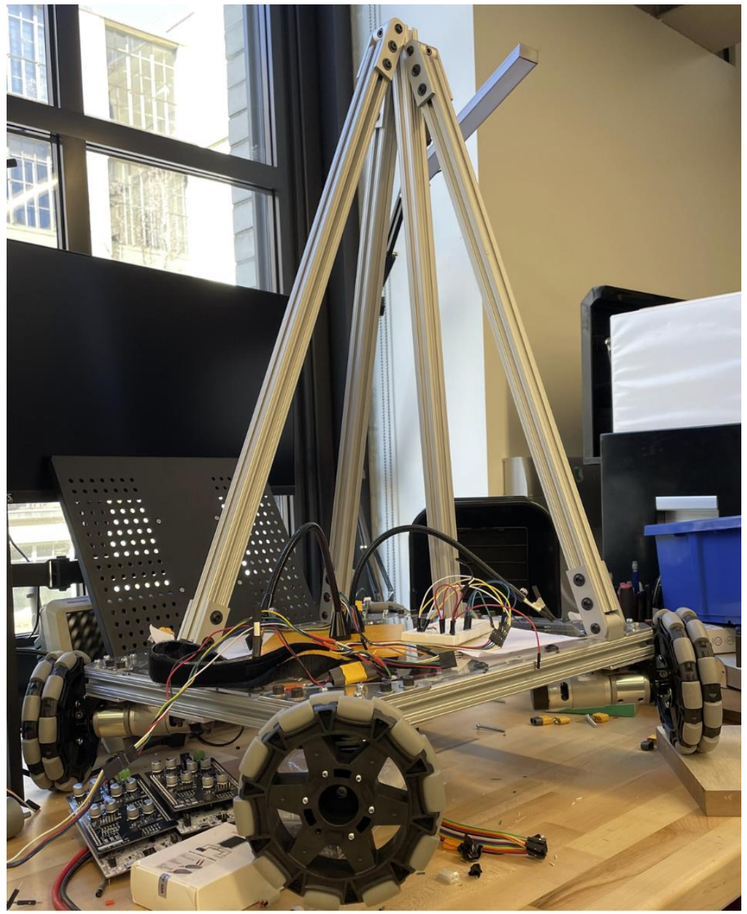
github.com/michaellu2019 mlu0708@mit.edu linkedin.com/in/michael-lee-lu



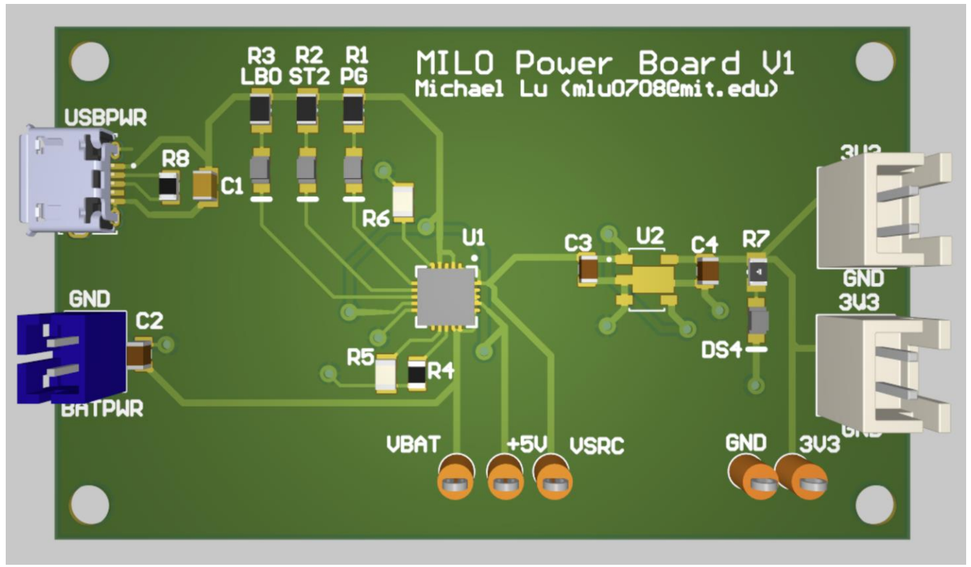
Hi, I'm Michael!

Just as painting is the creative outlet of an artist, engineering is my way to translate the imaginative ideas of my mind into the inventive work of my hand.

Upcoming Projects...



Omnidirectional perception robot



MicroUSB/Battery Power Board PCB

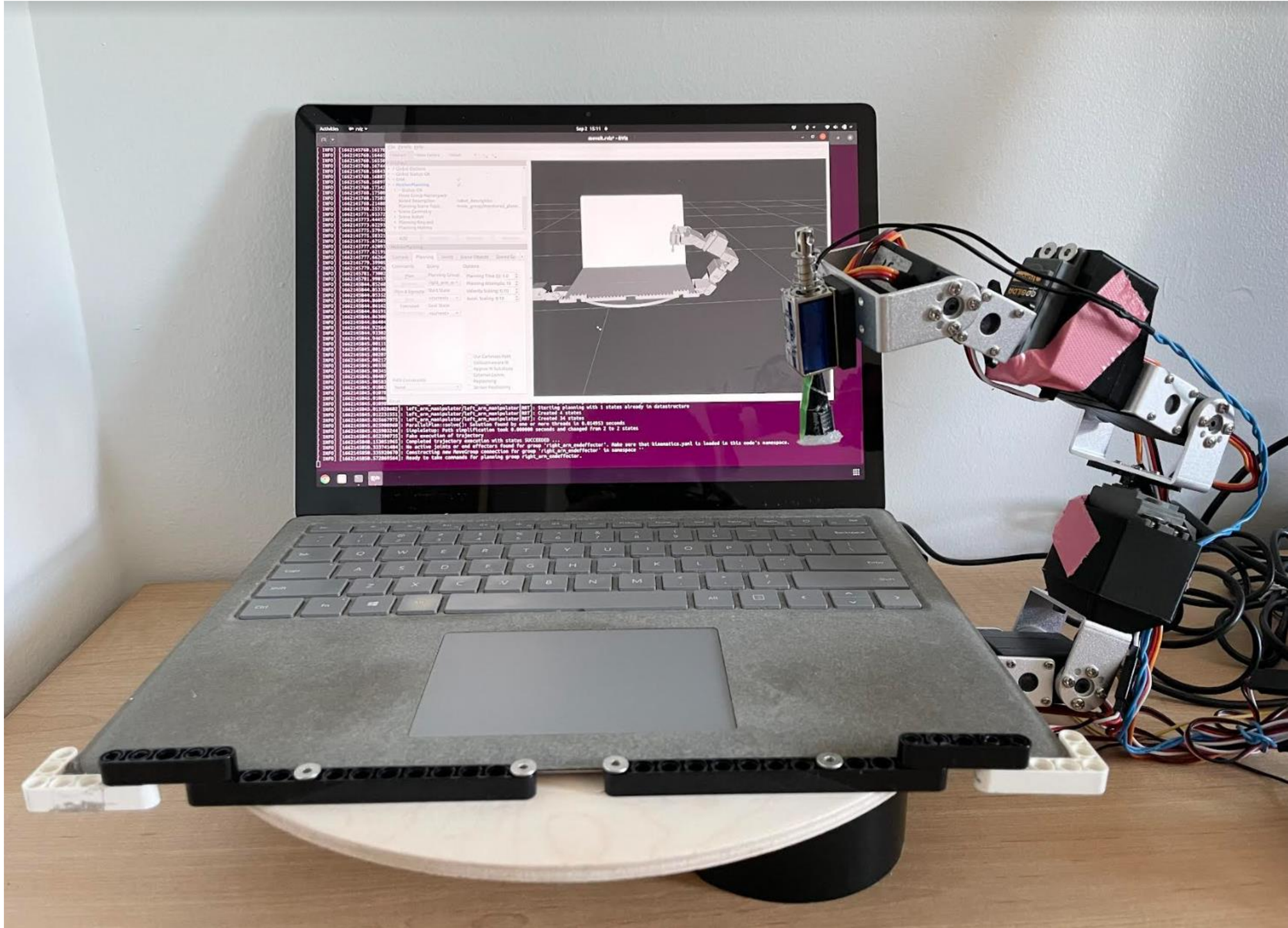
Injection-Molded Yo-yo

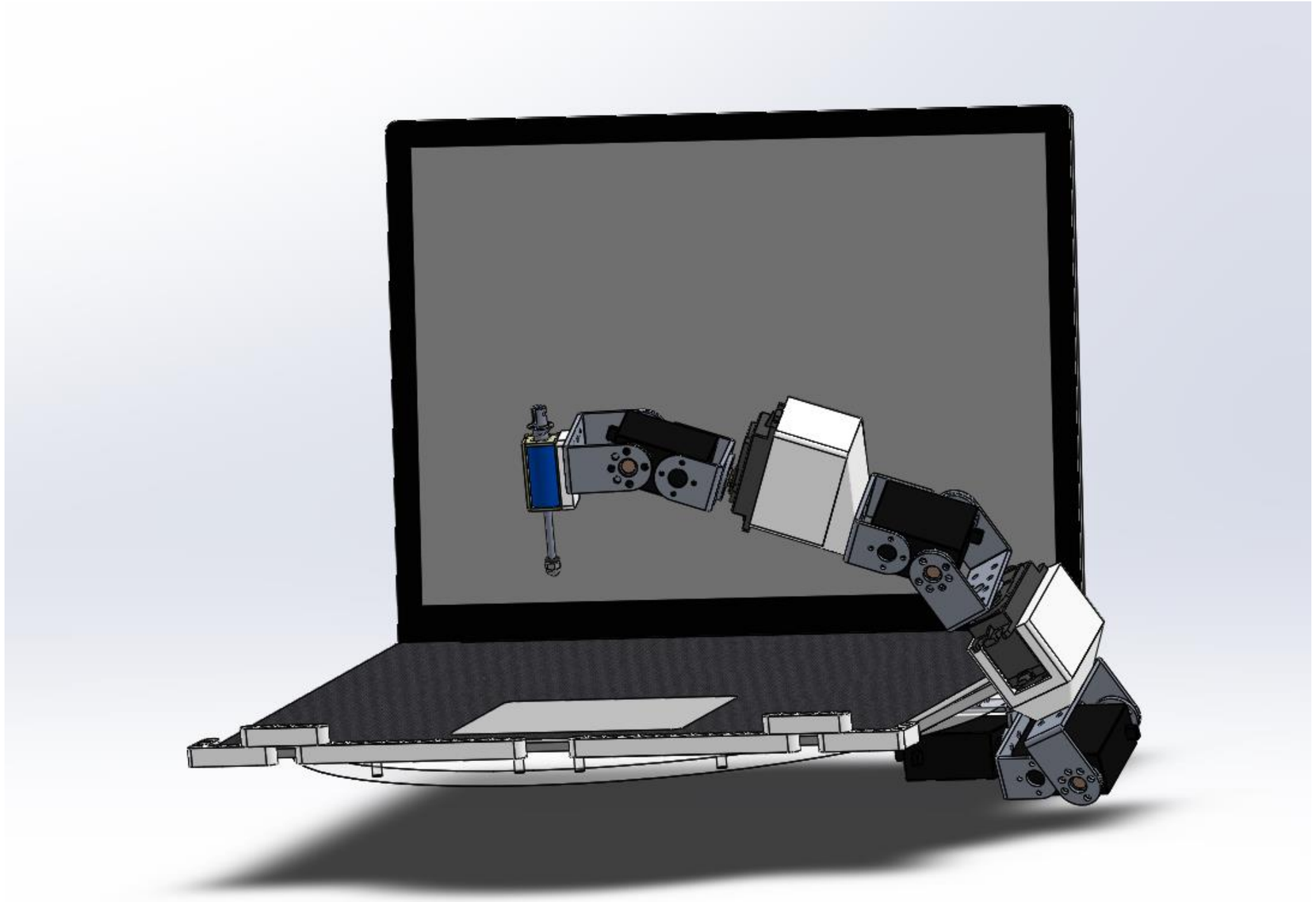
Laptop Bot

Personal Project
July – August 2022

A 6DOF robot arm that types on a laptop's keyboard. The robot arm takes commands from a laptop running ROS to move around and trigger an electrical solenoid at the end effector to press keyboard keys.

Code:
https://github.com/michaellu2019/laptop



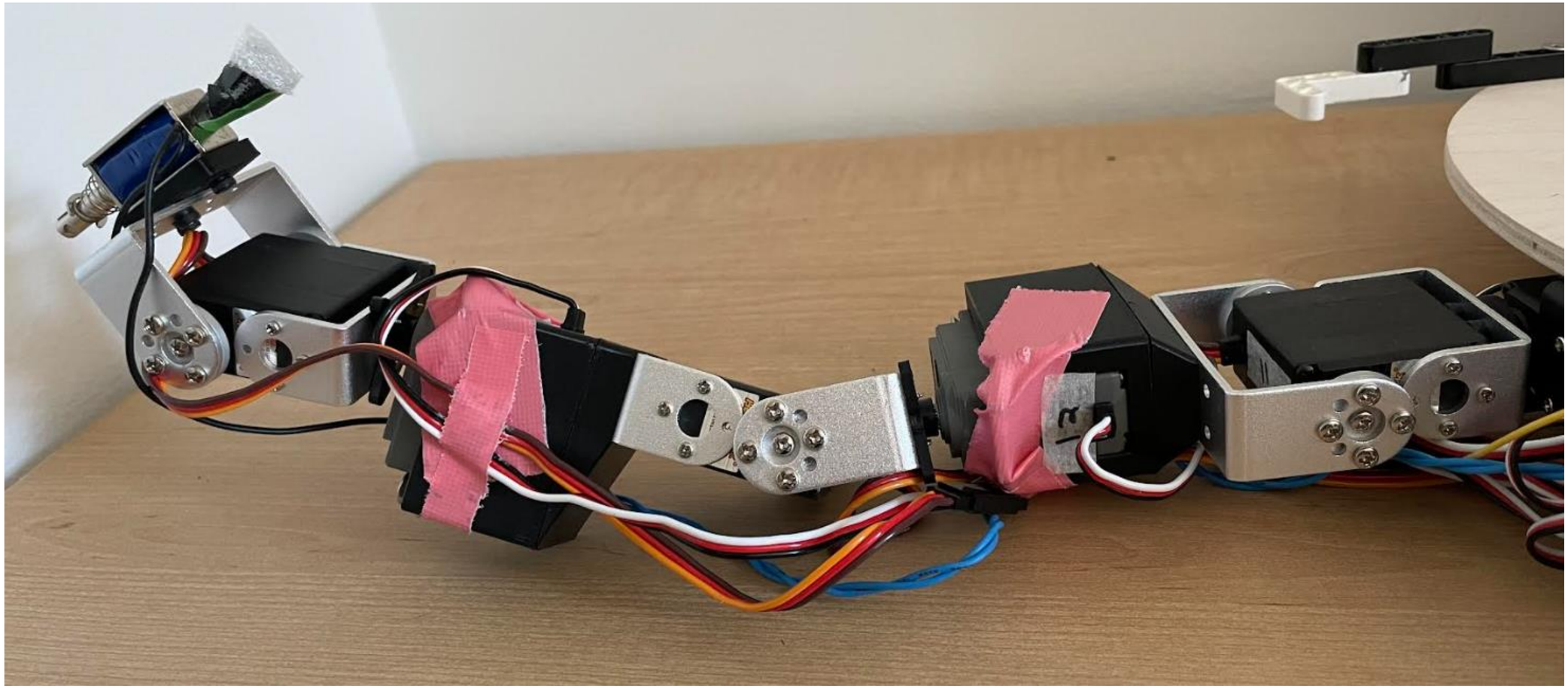


3D Modeling

I modeled the entirety of laptop bot in Solidworks, allowing me to test out the workspace of the 6DOF robot arm based on different assemblies to ensure all keyboard keys could be reached. The Solidworks model was exported to a URDF file that ROS used for inverse kinematics calculations. I also designed custom 3D-printed components to mount the arm's servo motors and the end effector's solenoid.

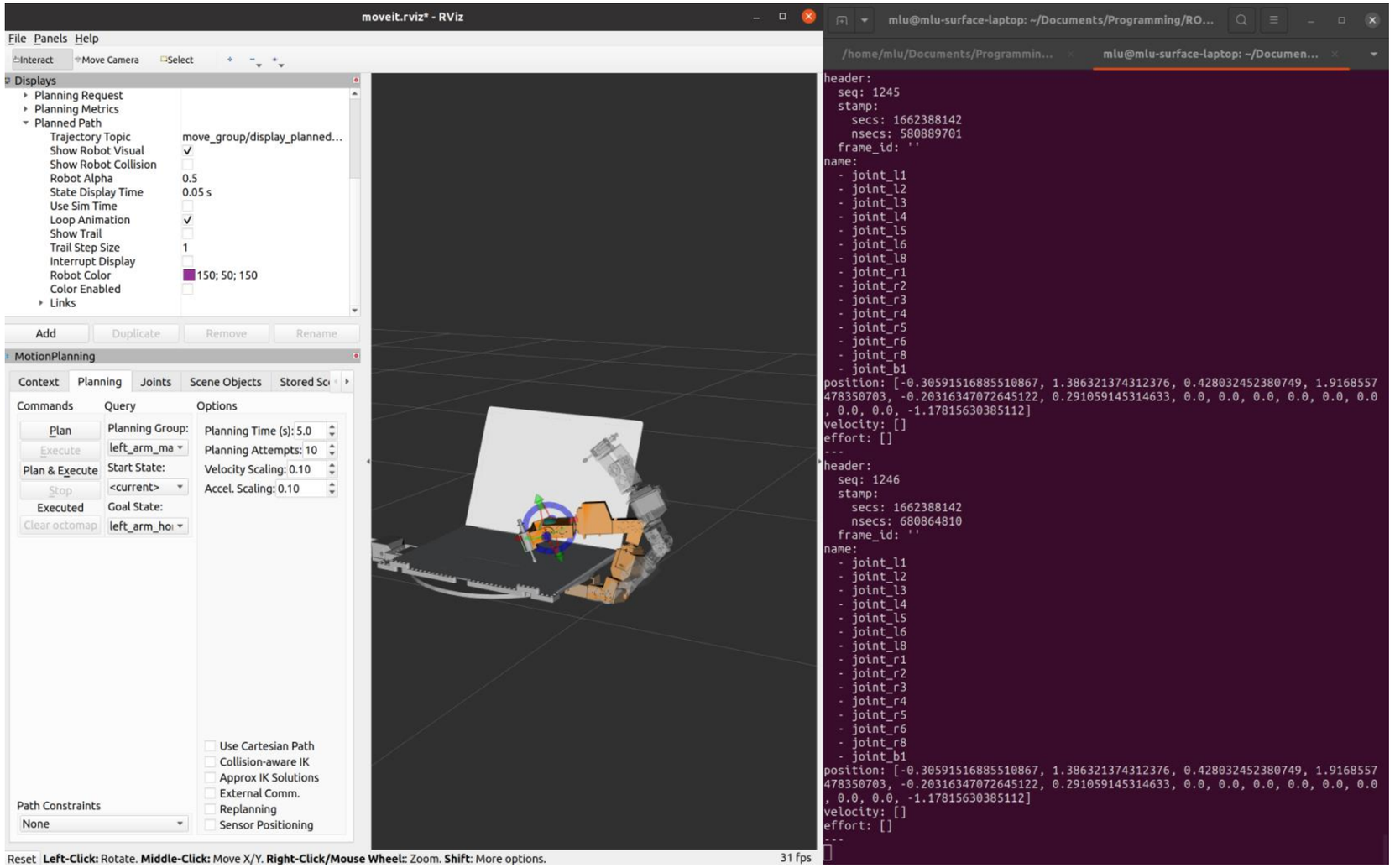
Prototyping

Servo motors and the end effector of the robot arm were mounted on custom 3D-printed components, allowing me to configure the arm's workspace by setting the desired lengths for the arm linkages. The arm's end effector was a DEVMO JF-0530B electronic solenoid that could be extended and retracted to press keyboard keys.



Prototyping

The robot arm was assembled with four HobbyPark servos and two goBilda servos, which were controlled by and Arduino Mega and an Arduino Mega sensor shield. To minimize bouncing of the arm caused by the extended end effector solenoid hitting a keyboard key, I padded the end of the solenoid with foam.



Programming

To manipulate the robot arm to type on the laptop's keyboard, I used ROS MoveIt's inverse kinematic solver to determine servo motor joint angles from cartesian coordinates of the arm end effector. These joint angles were published to a ROS subscriber in the Arduino Mega driver code that controlled the servo motor positions.

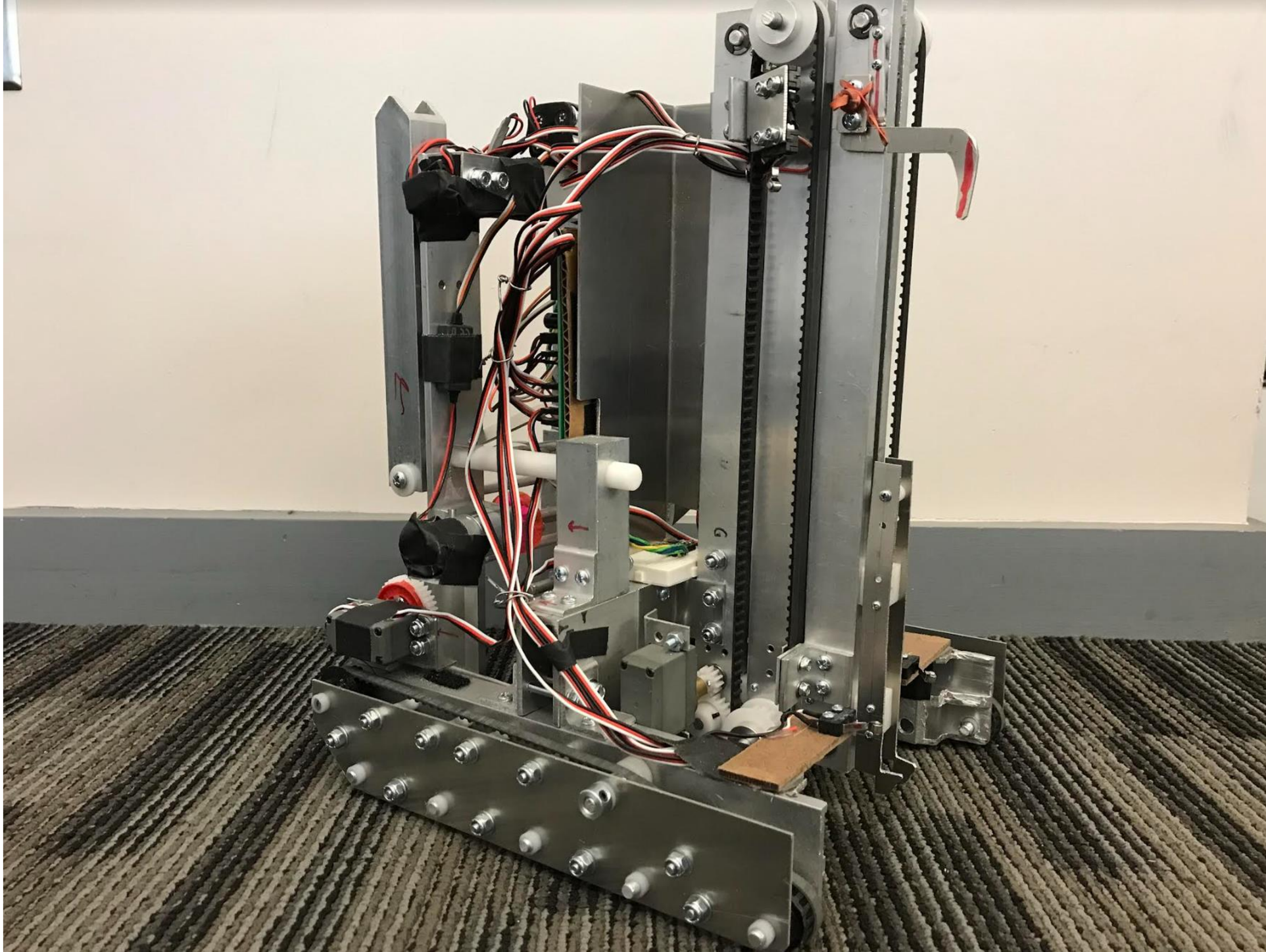

Lol hi

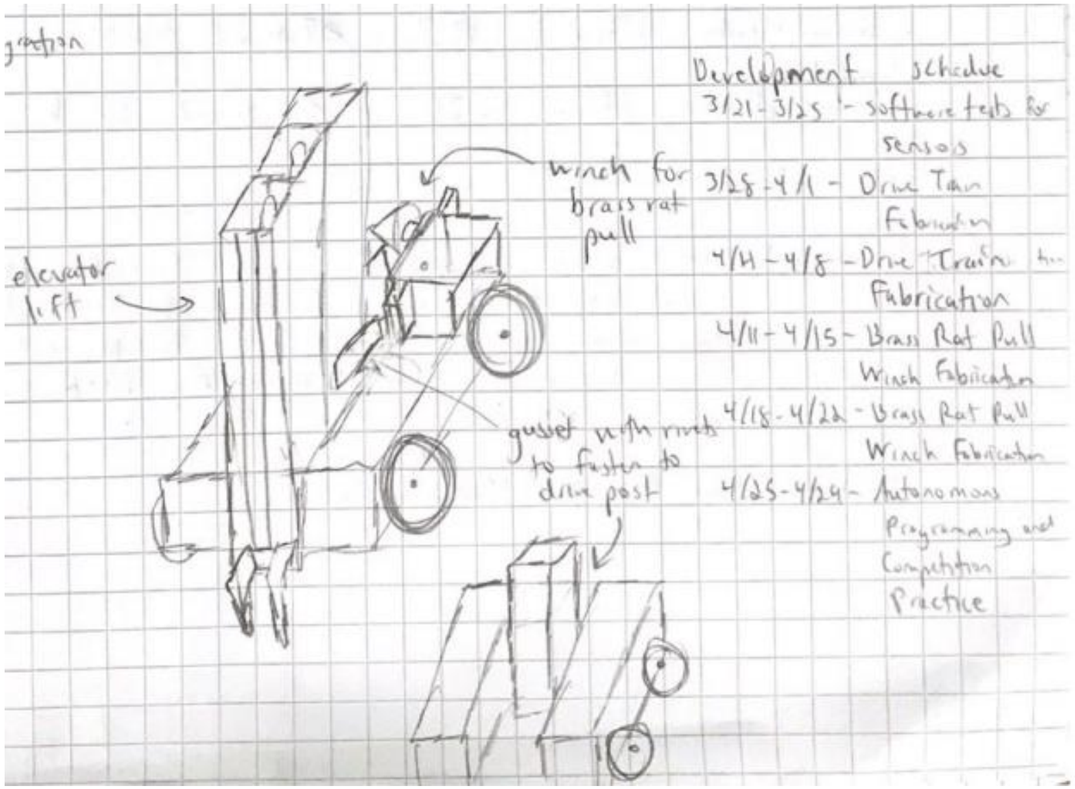
2.007 Robot

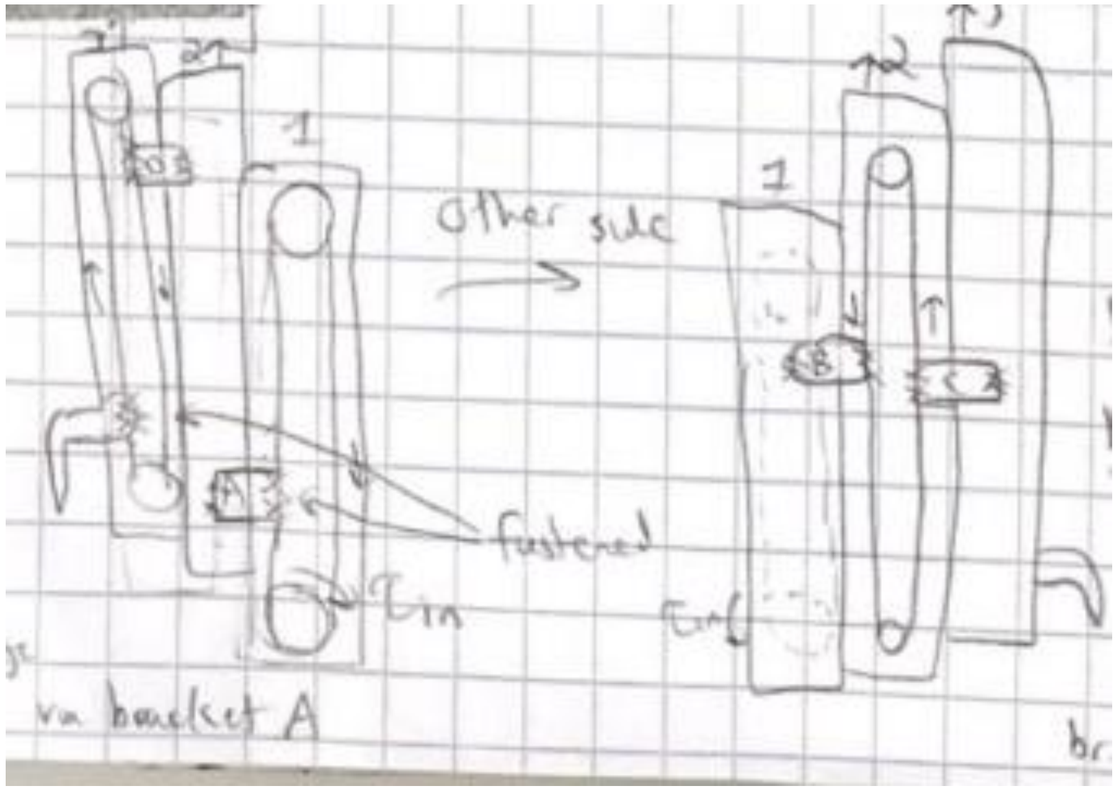
2.007 – Design and Manufacturing I
February – May 2022

A robot for the final competition of 2.007, MIT's design and manufacturing class. The aluminum robot featured a belt-driven cascade lift, detachable winch, and tank chassis to lift and pull heavy items on the competition field.

Code: https://github.com/michaellu2019/big-daddy





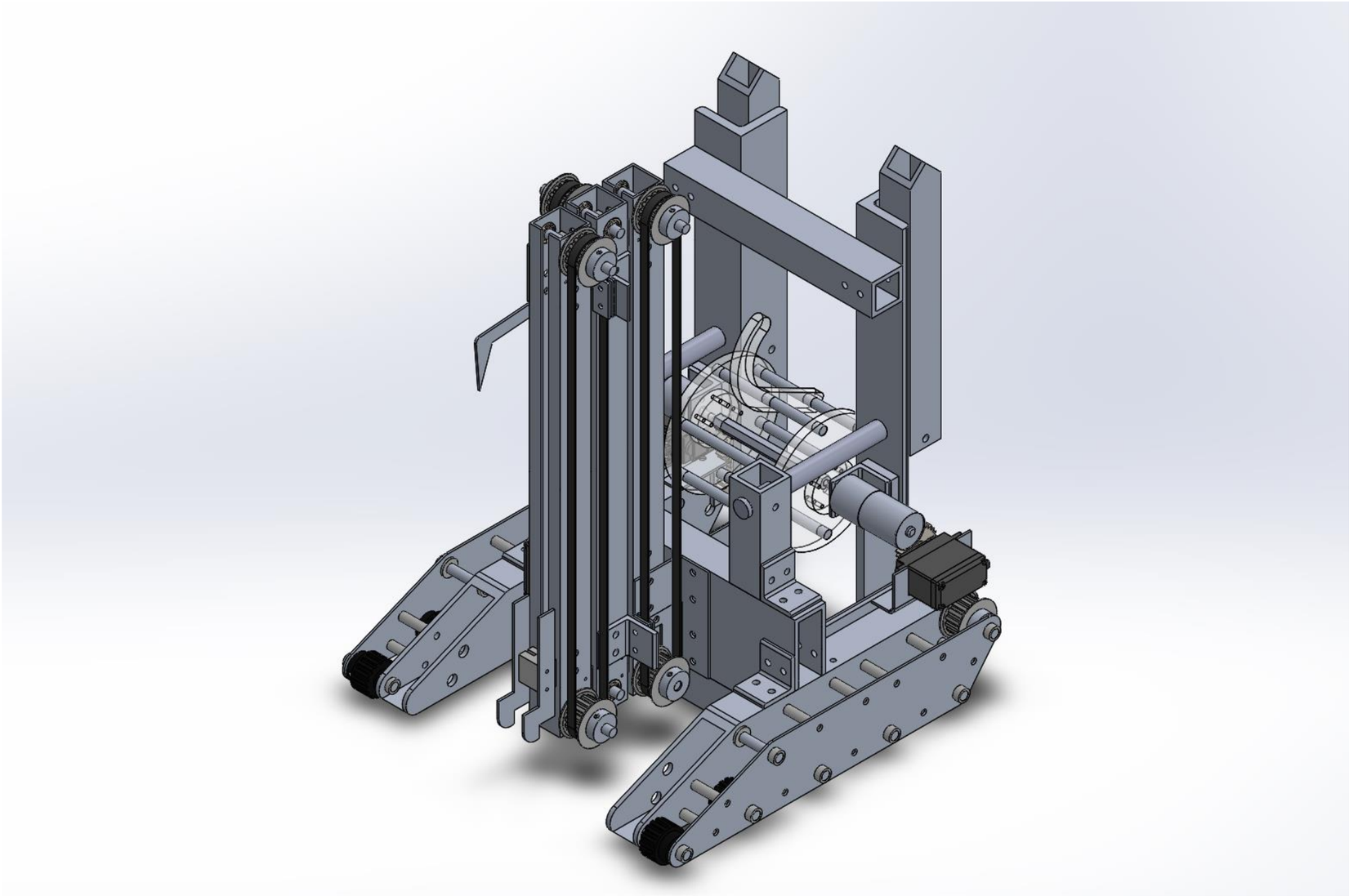


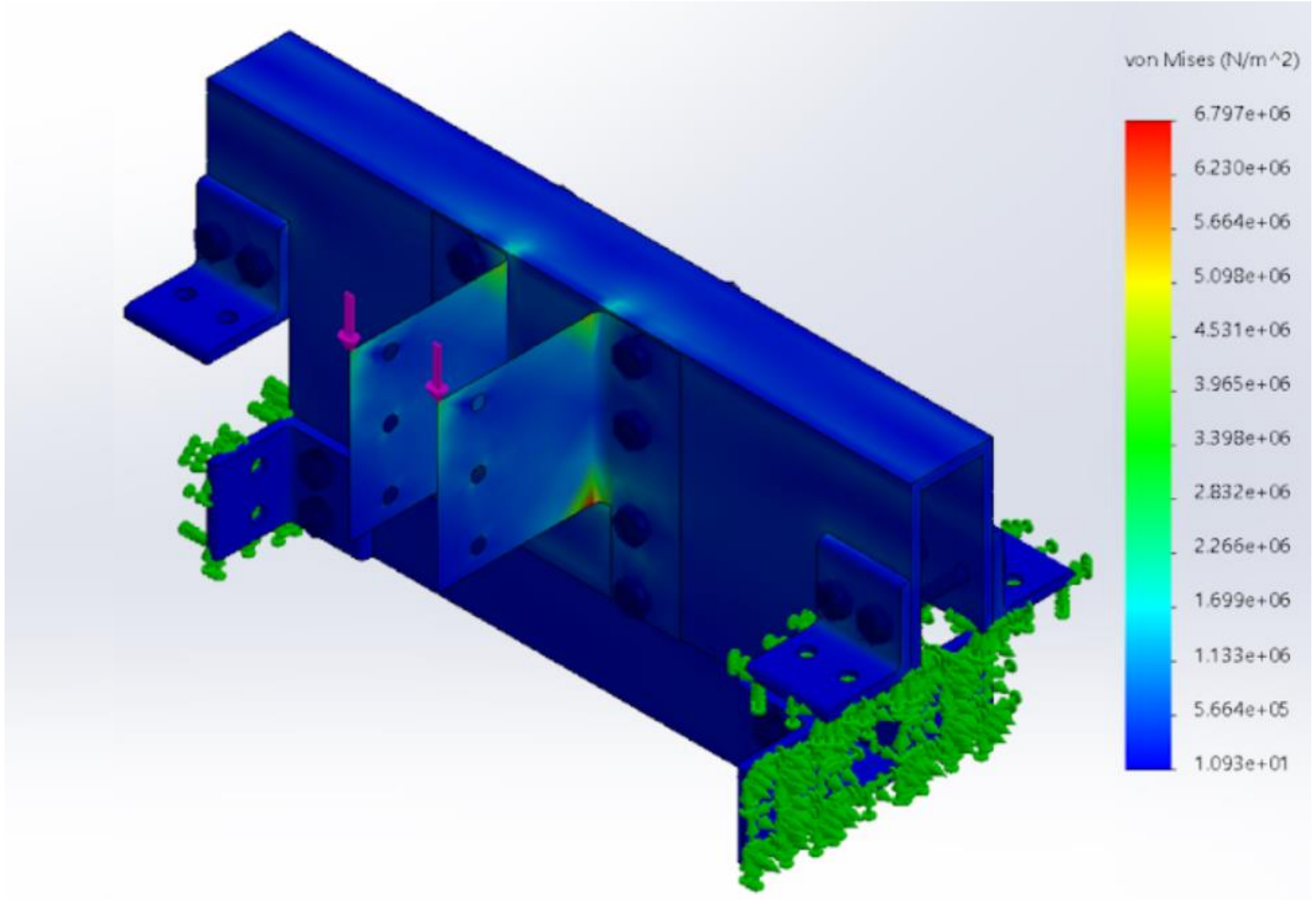
Ideation and Sketching

The final competition for the class rewarded robots that could lift and pull heavy game objects over long distances, so the key mechanisms for my robot were a grippy tank chassis, a powerful cascade lift, and high torque winch.

3D Modeling

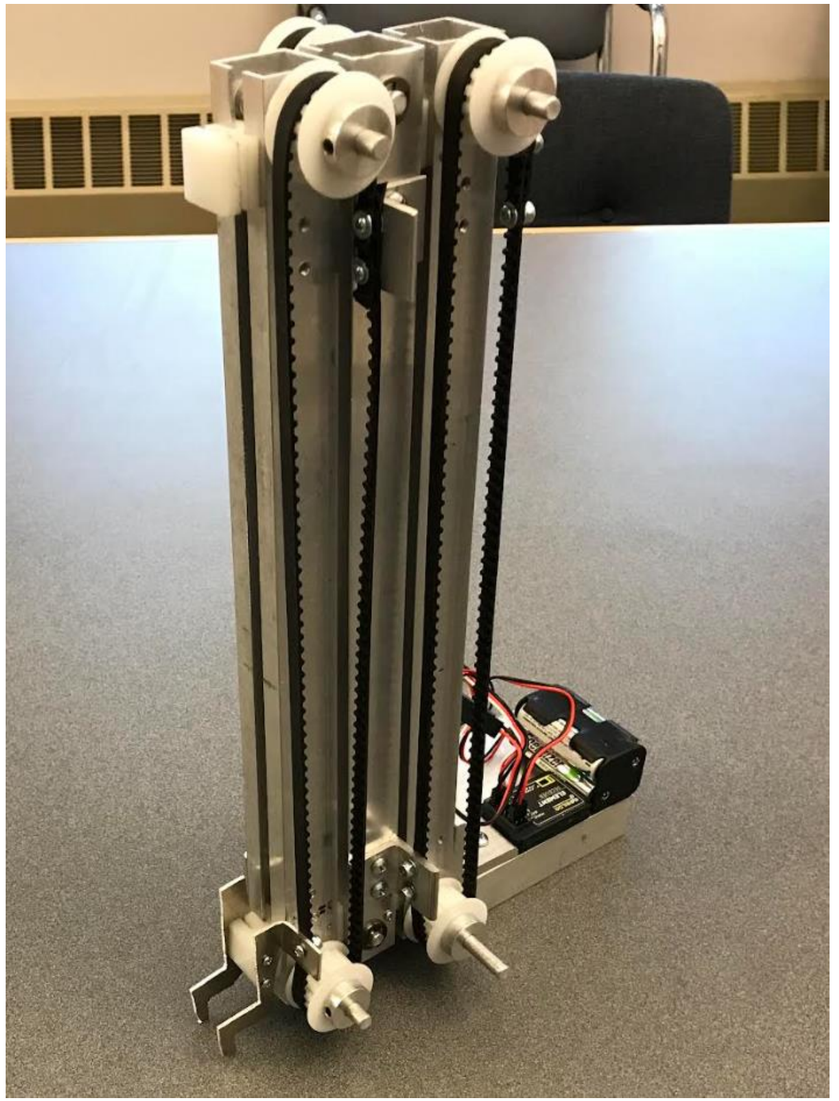
I used Solidworks to model the entire competition robot, ensuring that it would be within the 12”x12”x16” competition size limit and 10lbs weight limit. Assembling the entire robot in CAD also helped me figure out the internal spacing of mechanisms to ensure that they would not interfere with each other.





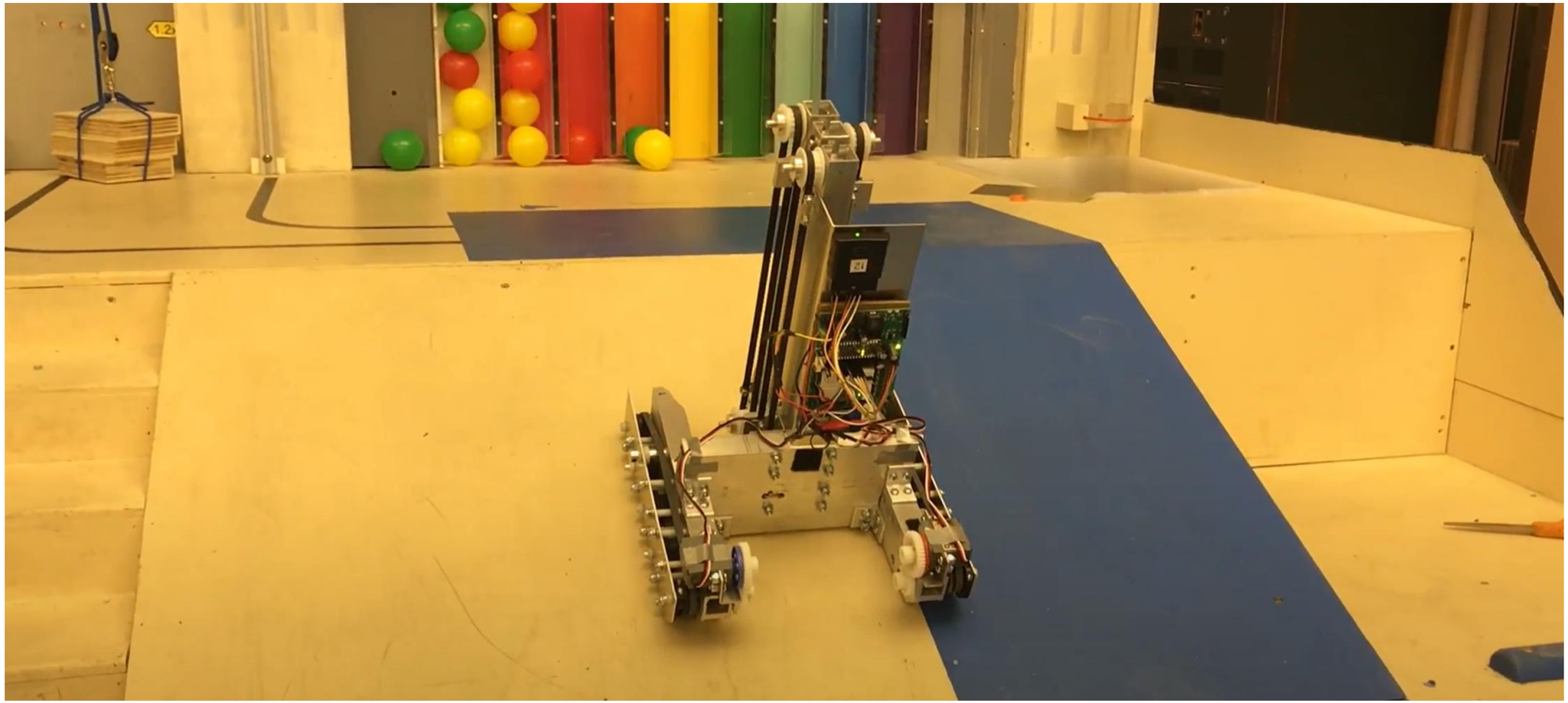
3D Modeling

Using Solidworks also allowed me to perform stress analysis on load-bearing components of the robot's chassis as well as analysis of the robot's mass properties to determine its center of gravity. These simulation tools were useful in anticipating any performance issues with the robot assembly.



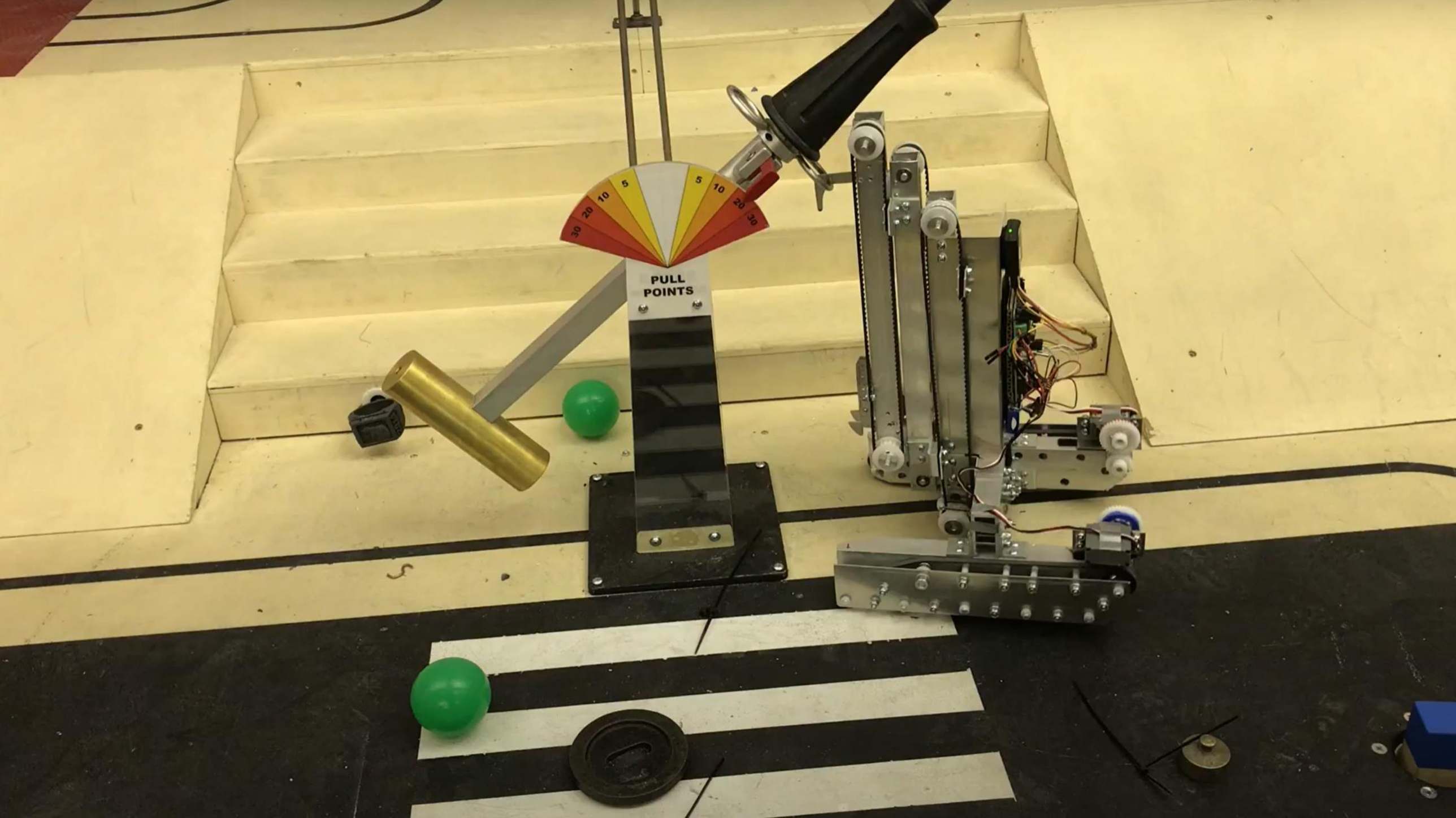
Prototyping

The cascade lift was assembled from three 6063-T6 aluminum box extrusion pieces. I milled channels into each extrusion to insert Delrin sliders and added a series of pulleys, brackets, and 6mm timing belt loops so that one goBilda 25-2 torque servo motor could move all three stages simultaneously, allowing it to lift a 17N weight in 4 seconds. The robot also had a detachable 3” winch that used two LDO 217:1 DC motors to pull a 12N weight for 30 seconds.



Programming

To mitigate any imperfections in the robot's tank chassis assembly, I used an Arduino Nano to process MPU6050 accelerometer data so that a PID controller would adjust the PWM signal to the robot's left and right drive motors, helping the robot drive straight and make precise turns.

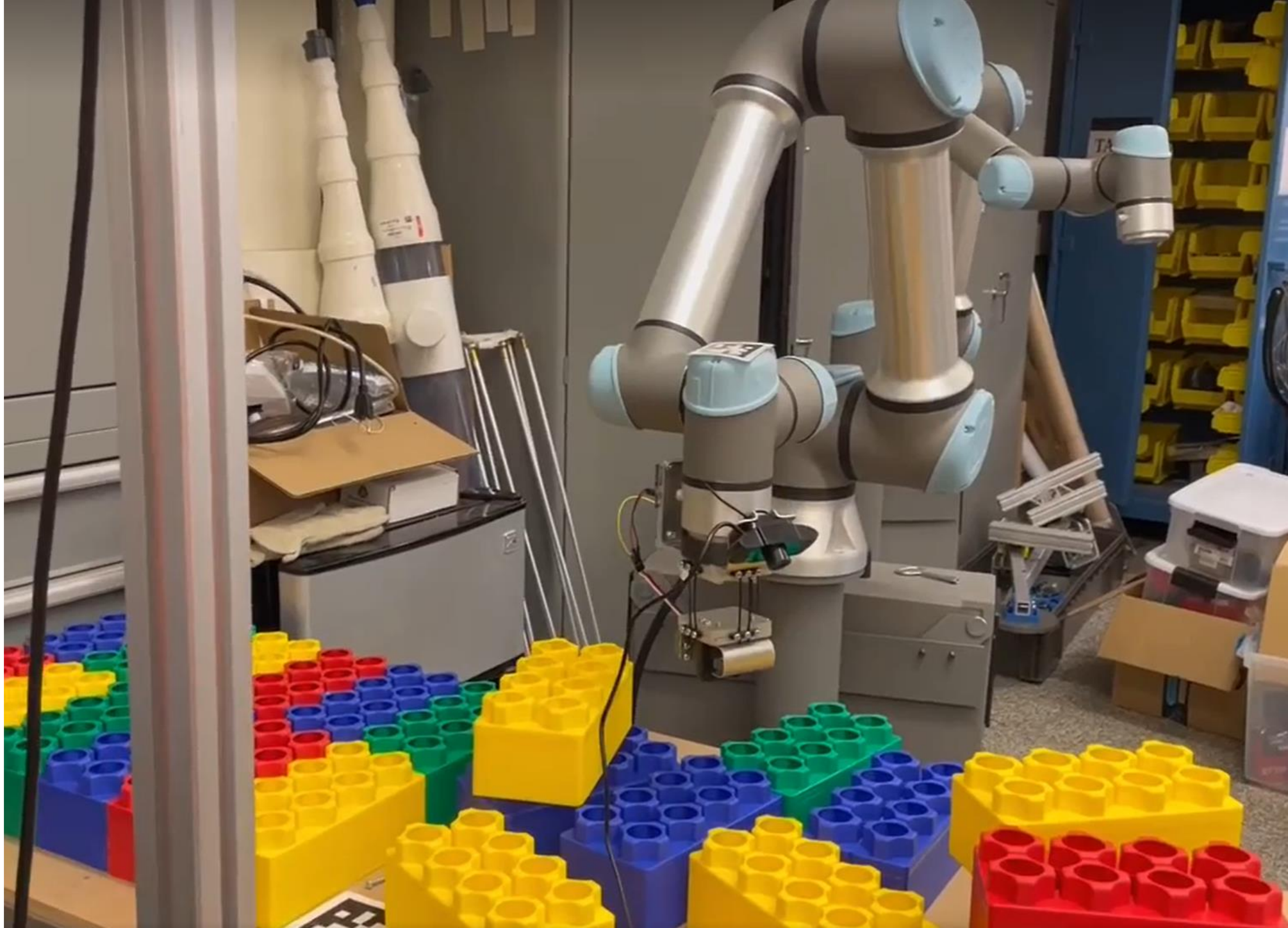

30
20
10
5
5
10
20
30
PULL
POINTS

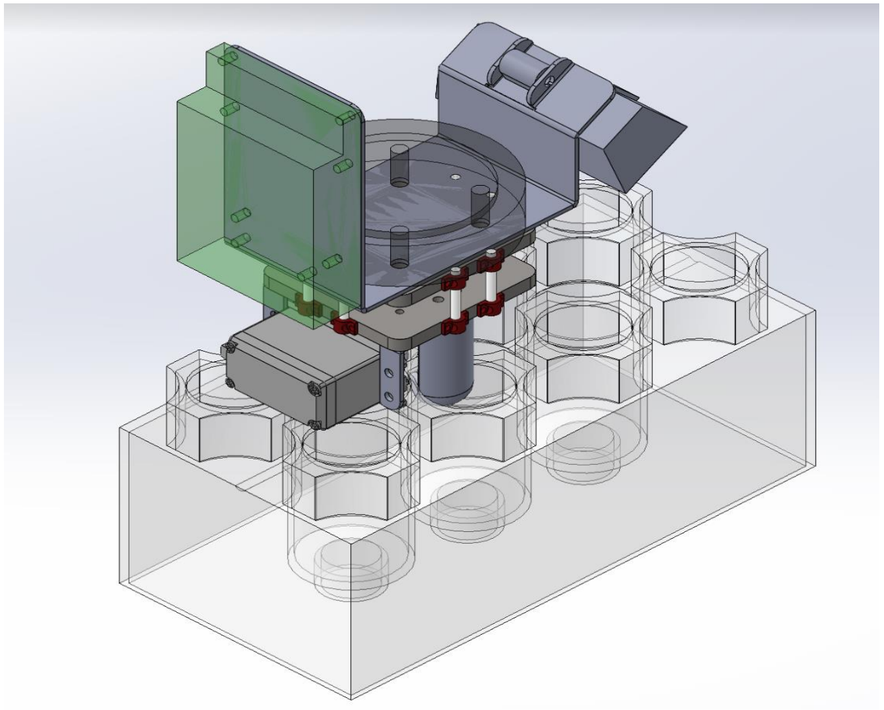
Lego Building Bot

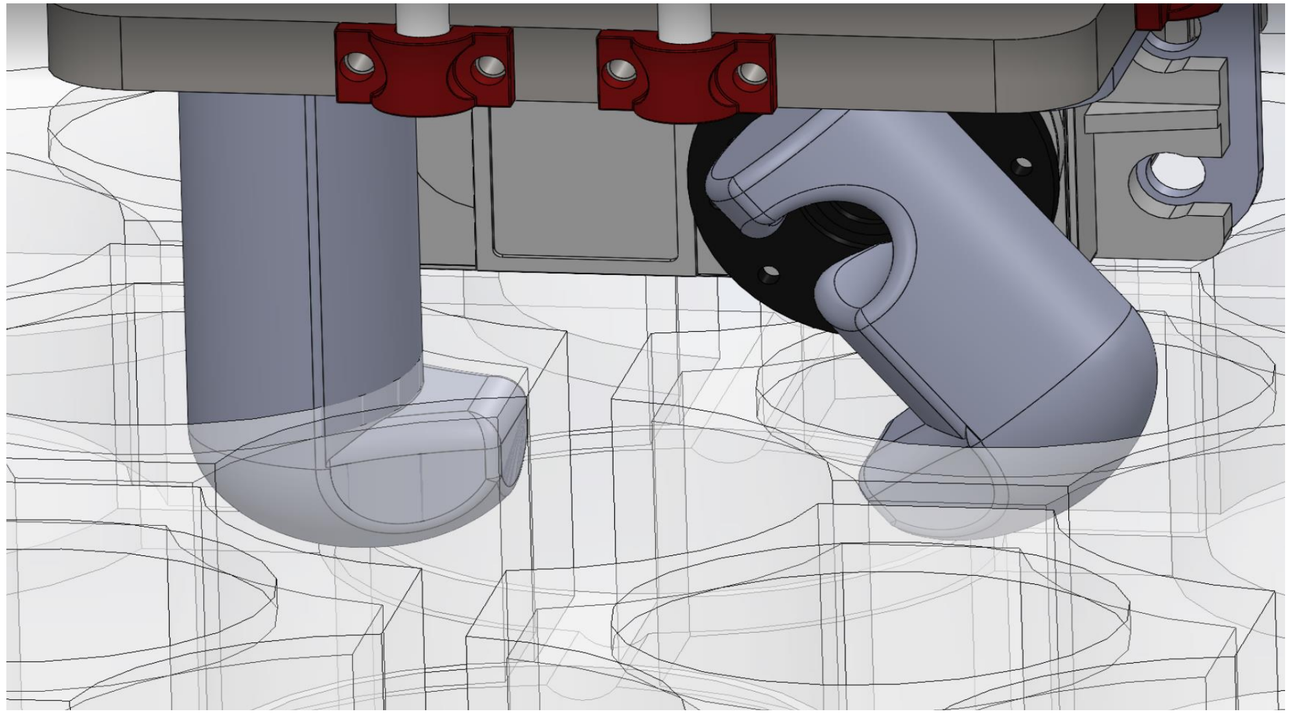
2.12 – Introduction to Robotics
April – May 2022

A ROS-controlled 6DOF UR5 robot arm that leverages OpenCV computer vision and a compliant gripper to assemble giant Lego bricks.

Computer Vision Code:
https://github.com/michaellu2019/2.12-final-project
Manipulation Code:
https://github.com/qbowers/arm_movement

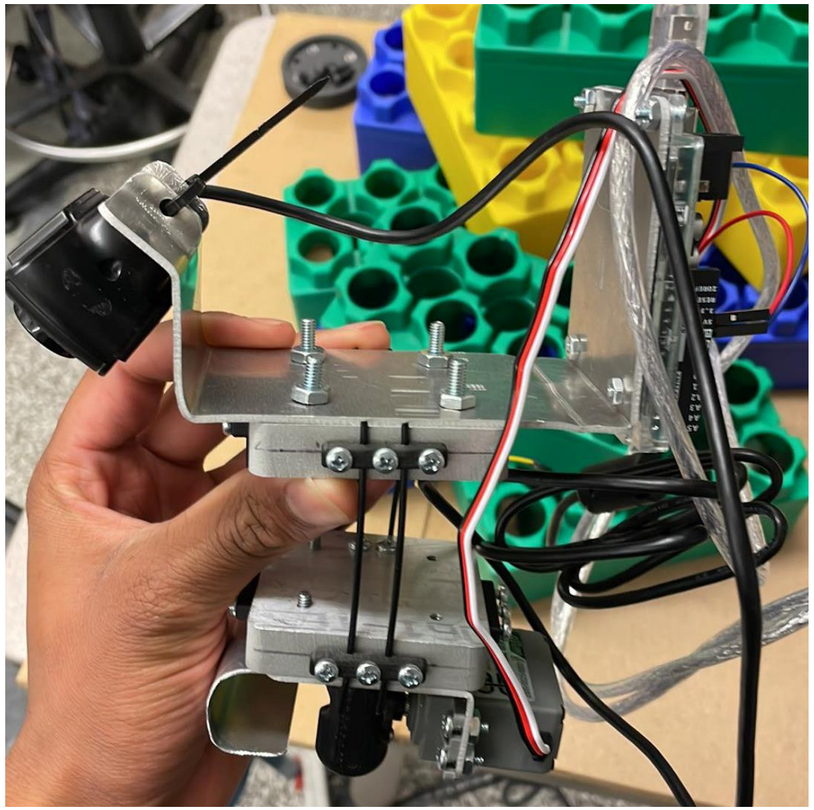






3D Modeling

Modeling the gripper and Lego brick helped with visualizing how much tolerance there would be in the end effector claws when grabbing the Lego brick hole edges for different claw shapes and sizes.

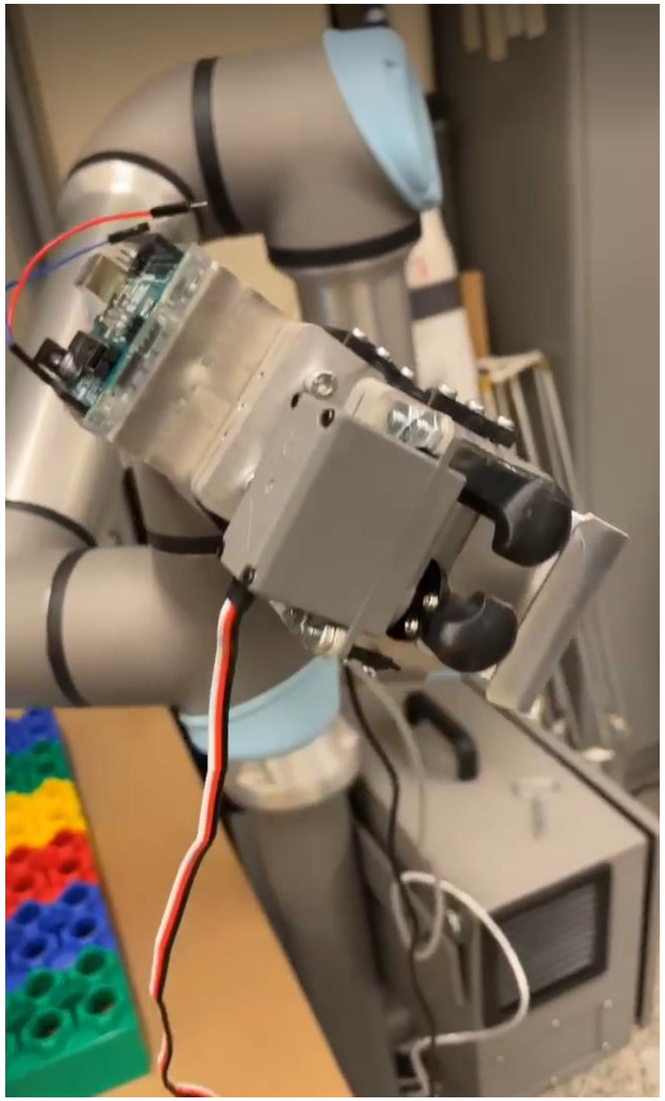


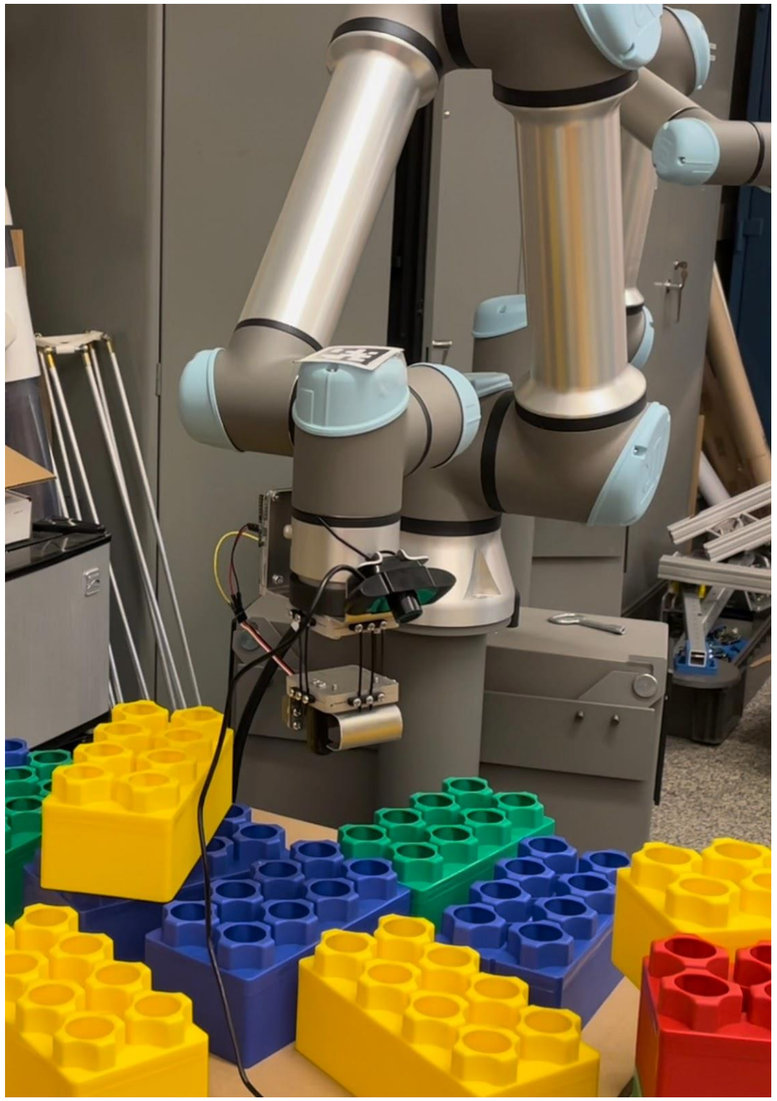
Prototyping

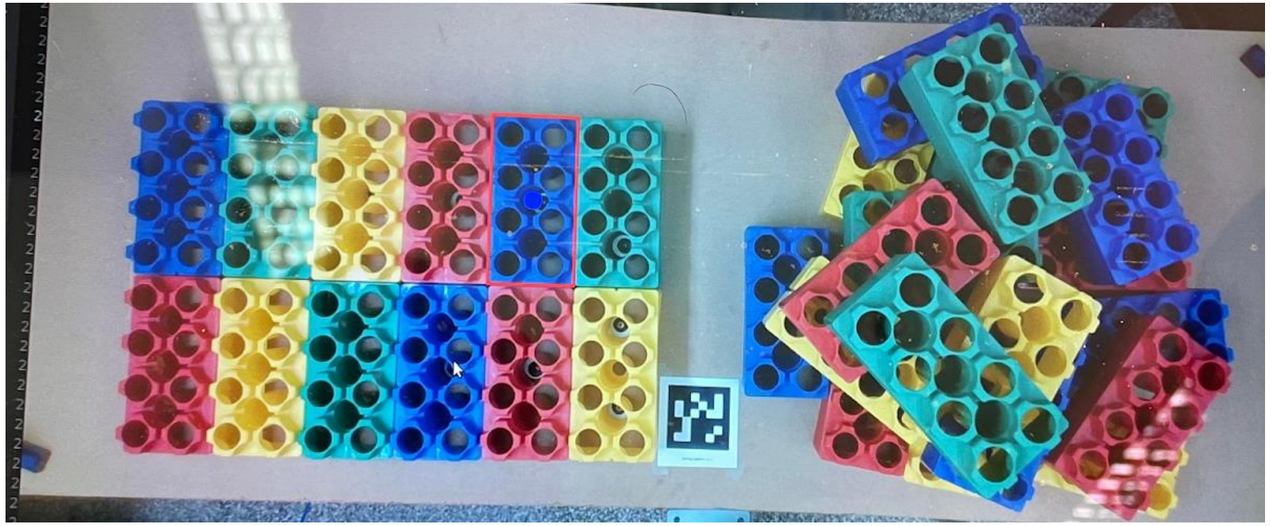
The gripper and UR5 robot arm were connected with six 1.75mm PETG 3D-printer filament strands, adding compliance to the gripper so it could shift itself into the holes of a Lego brick if it was slightly misaligned.

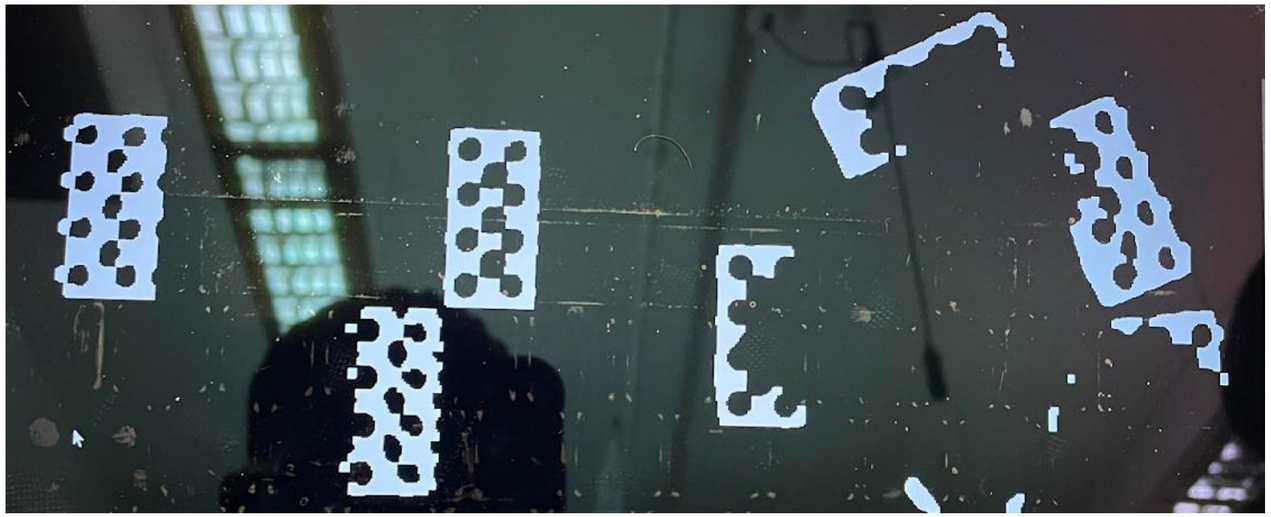
Programming

The UR5 robot arm was controlled by pressing laptop keyboard arrow keys, which would change the cartesian coordinates of the UR5 end effector. These coordinates would be passed into ROS MoveIt, which would return the UR5 joint angles and a planned path to move to those angles. The gripper was controlled by an Arduino Uno that would read USB serial commands from the laptop.









Programming

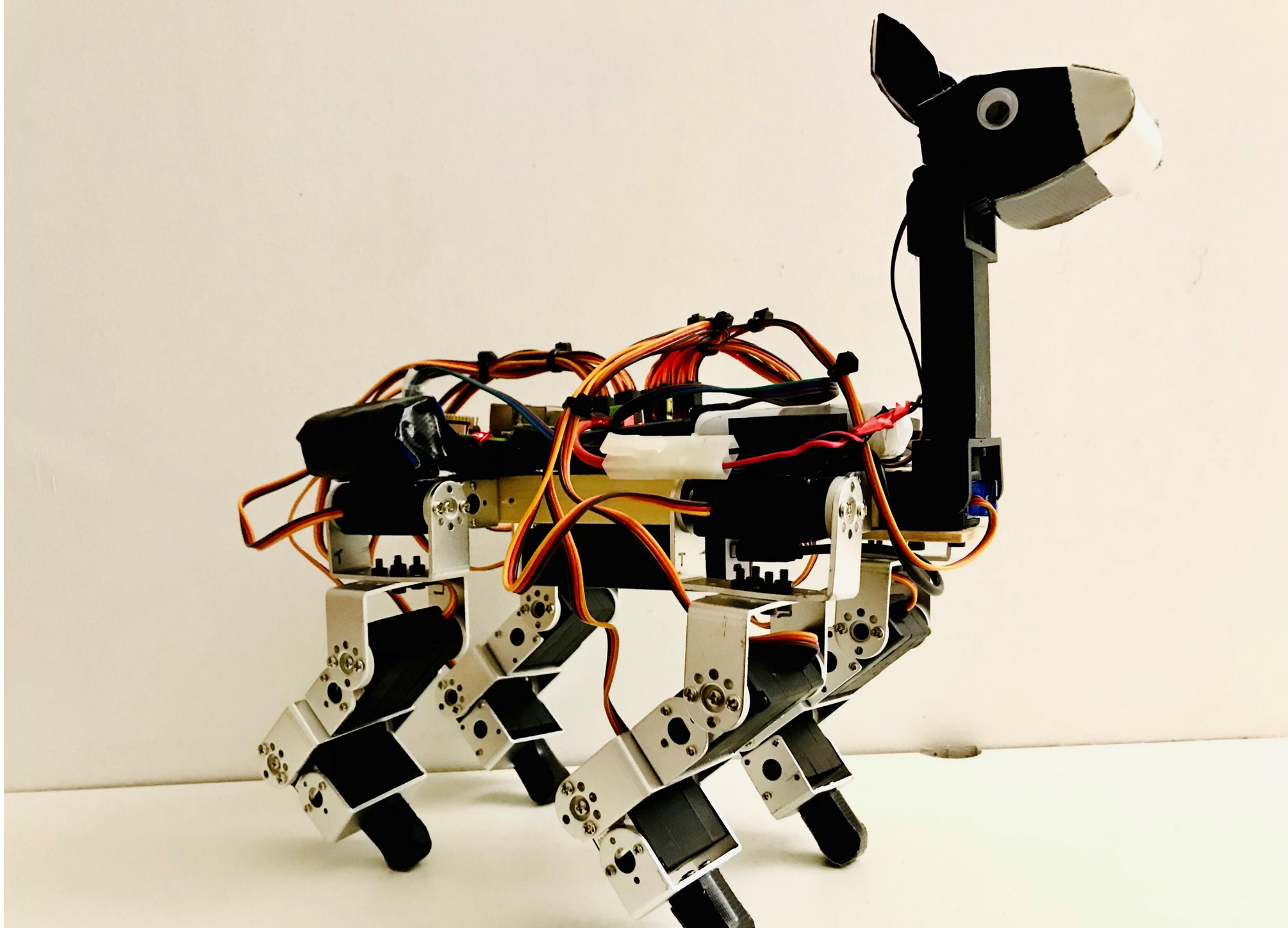
OpenCV algorithms were run on the UR5 camera feed to detect Lego bricks. Hue segmentation was used to isolate bricks of certain colors, followed by brick layer filtering based on Intel RealSense depth camera data, and finally contour detection with Hough transforms to draw bounding boxes around bricks. Erosion and dilation were used in between steps to remove noise from the image, improving brick detection performance.

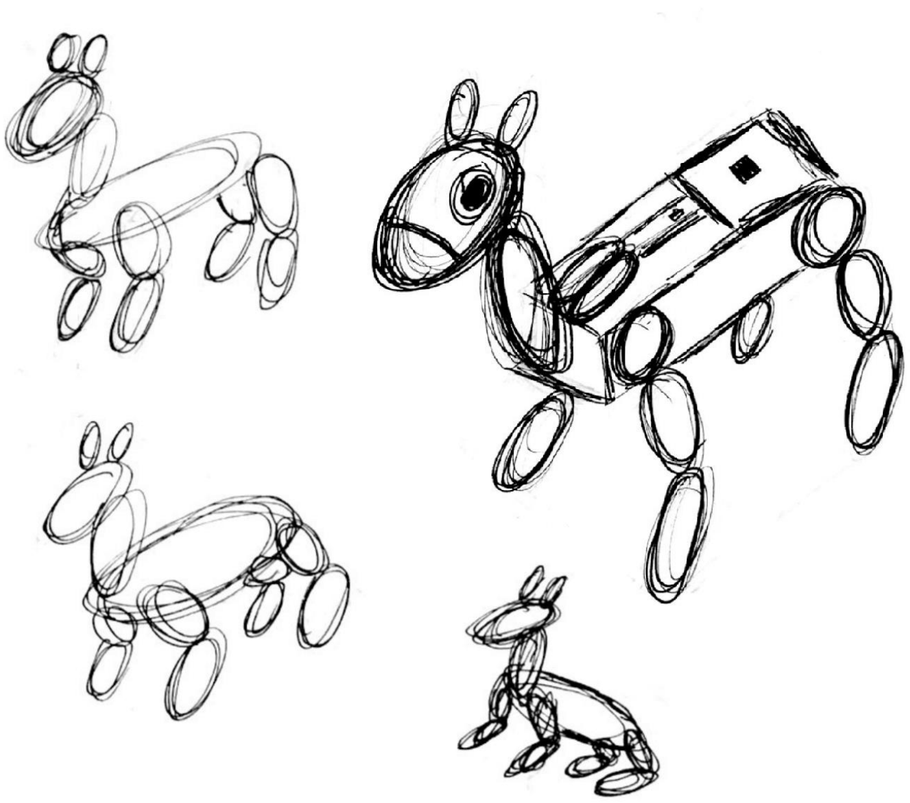
Robot Donkey

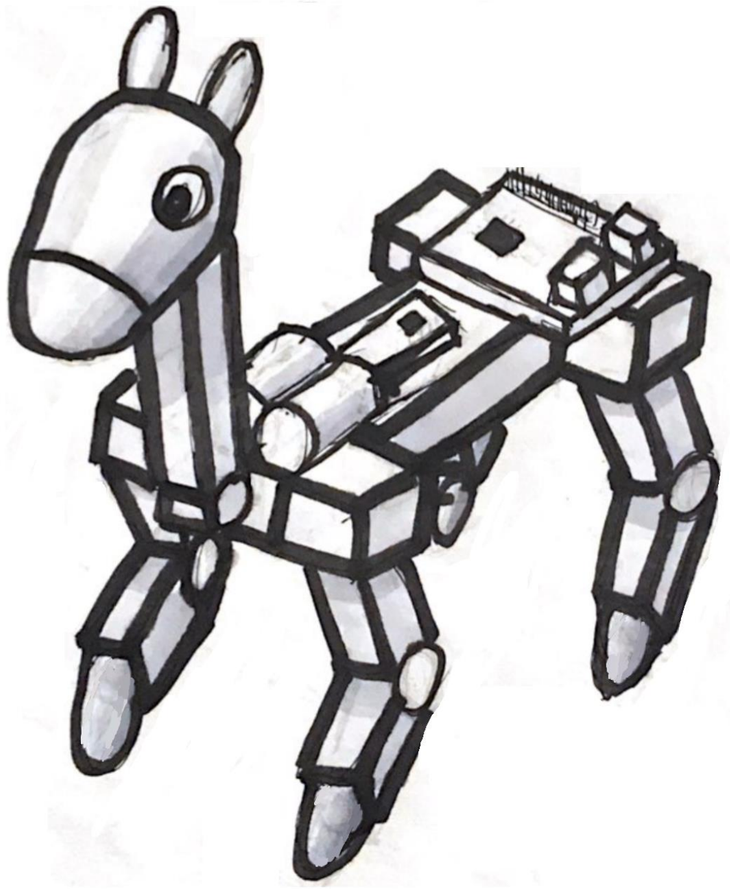
Personal Project
January – June 2021

A WiFi-controlled robotic donkey that walks using inverse kinematics. The robot has four 3-degrees-of-freedom (DOF) legs, a 2-DOF neck/head, and two mini speakers for audio output.

Demo:
https://youtu.be/aikBnfc7GT0
Code:
https://github.com/michaellu2019/dog

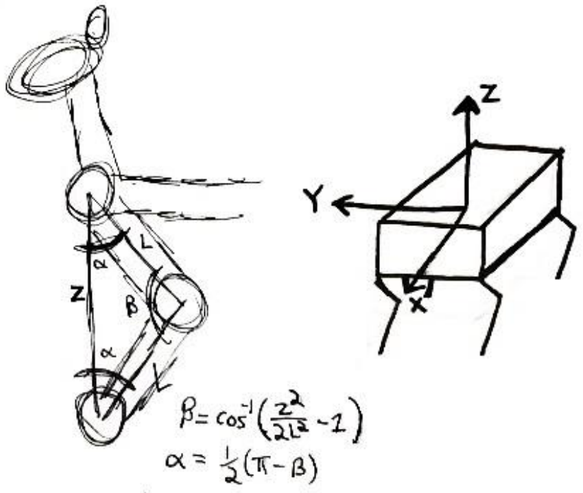






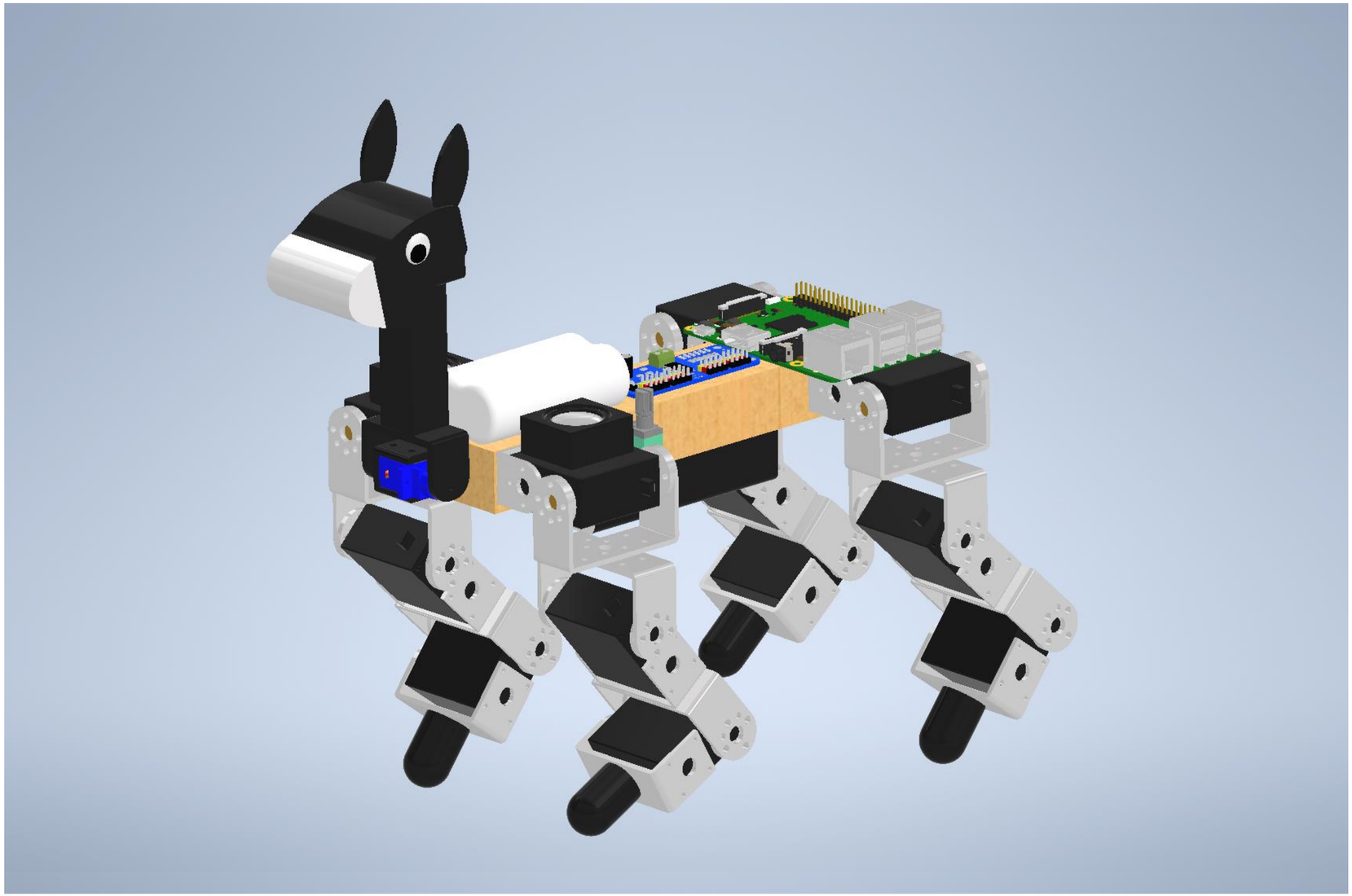
Ideation and Sketching

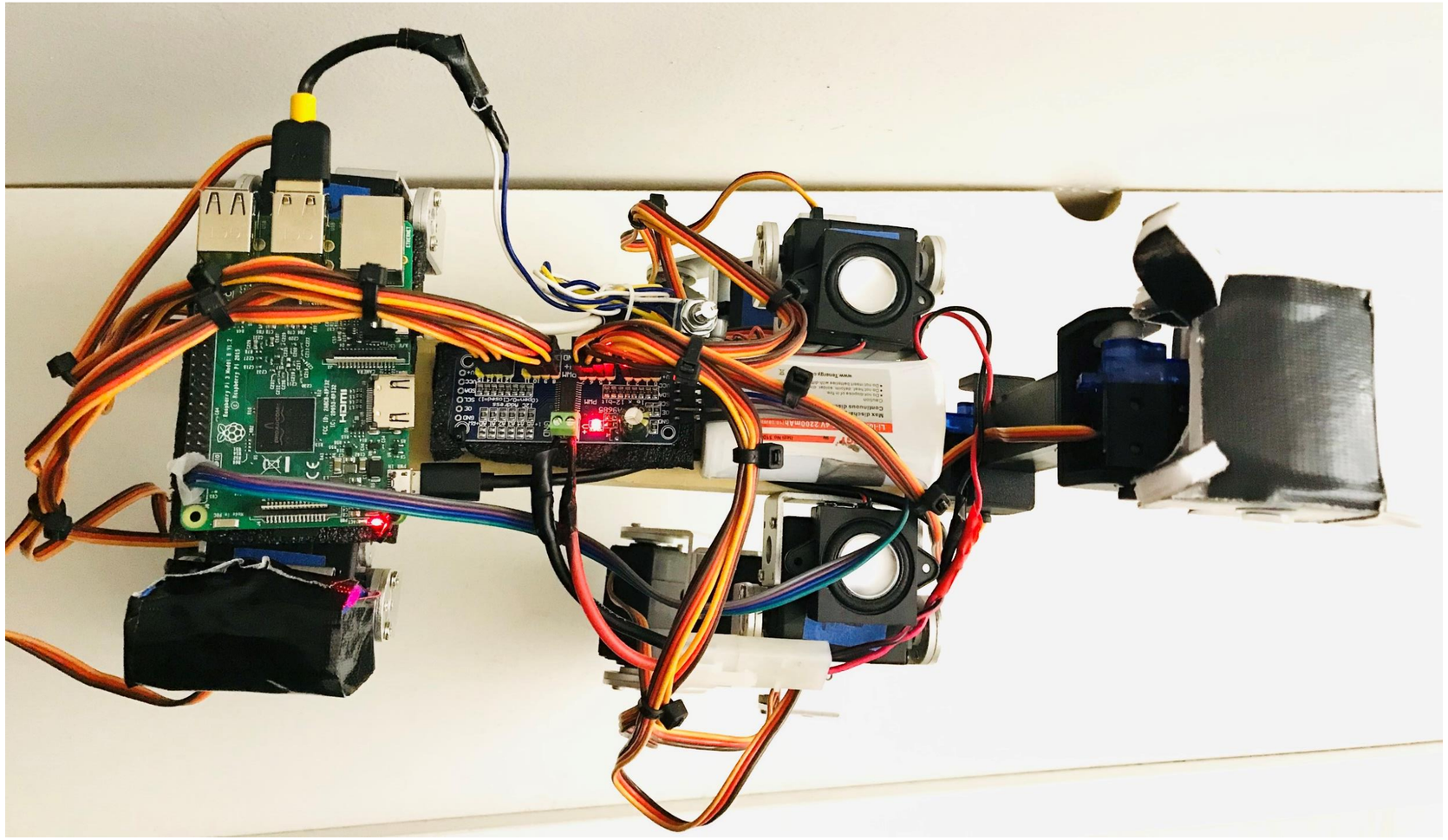
To build a 12-DOF quadruped and manipulate its orientation and position, I had to figure out how to pass in a 3D position and translate it into a set of angles for the servos to rotate to. Inverse kinematics would provide an easy way to calculate these values.



3D Modeling

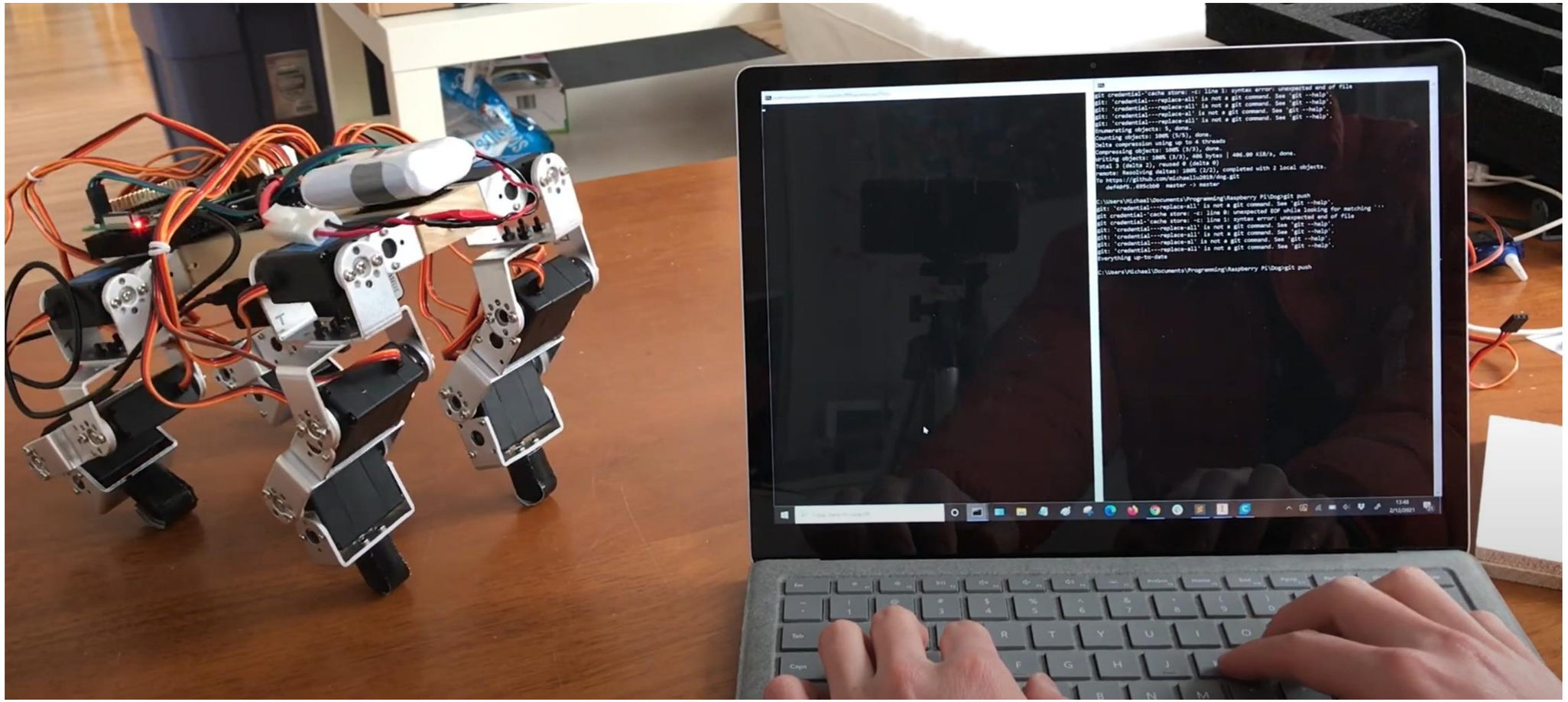
Robot Donkey's HobbyPark servos came with multiple aluminum mounting brackets, which made it easy to build four 3-DOF legs around a main wooden body. For the robot's feet, I used Autodesk Inventor to design four 3D-printed ball-point hooves that could be fastened to the servo brackets and were covered with rubber tips for more traction with the ground.





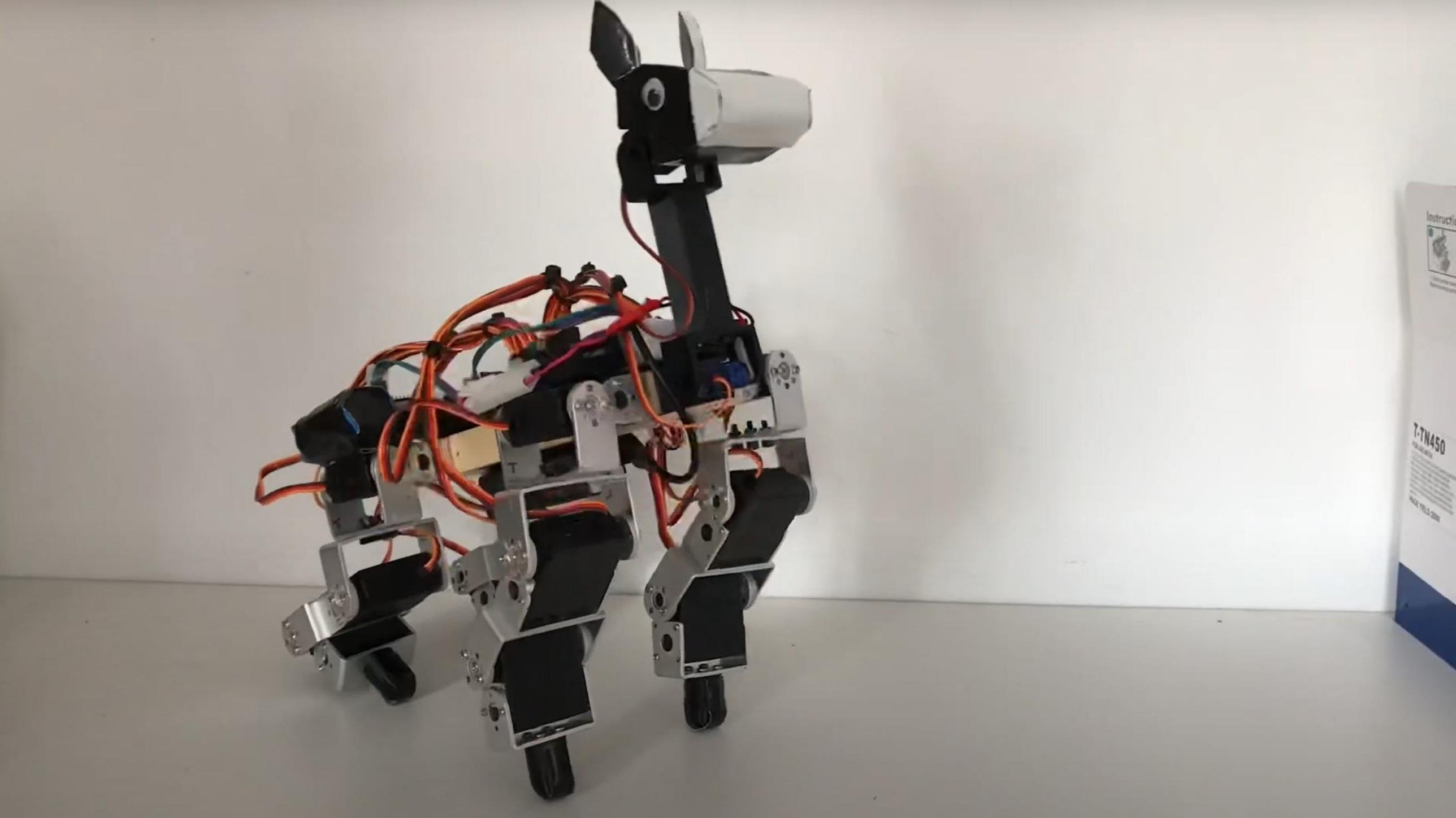
Prototyping

I built Robot Donkey with an Adafruit PCA9695 servo driver, 12 HobbyPark servos for the robot legs, 2 SG90 servos for the head and neck, a Super Mini PAM8403 audio amplifier and 2 MakerHawk speakers for sound, a Raspberry Pi to control all these components, and a USB power bank and 7.4V LiPo battery for power.



Programming

Robot Donkey was controlled by a Raspberry Pi Python script that rotated the leg joints based on inverse kinematics to move the robot to precise positions and orientations. Once the Raspberry Pi connected to the Internet after powering up, I could SSH into the board and control the robot from the command line on my laptop over WiFi.

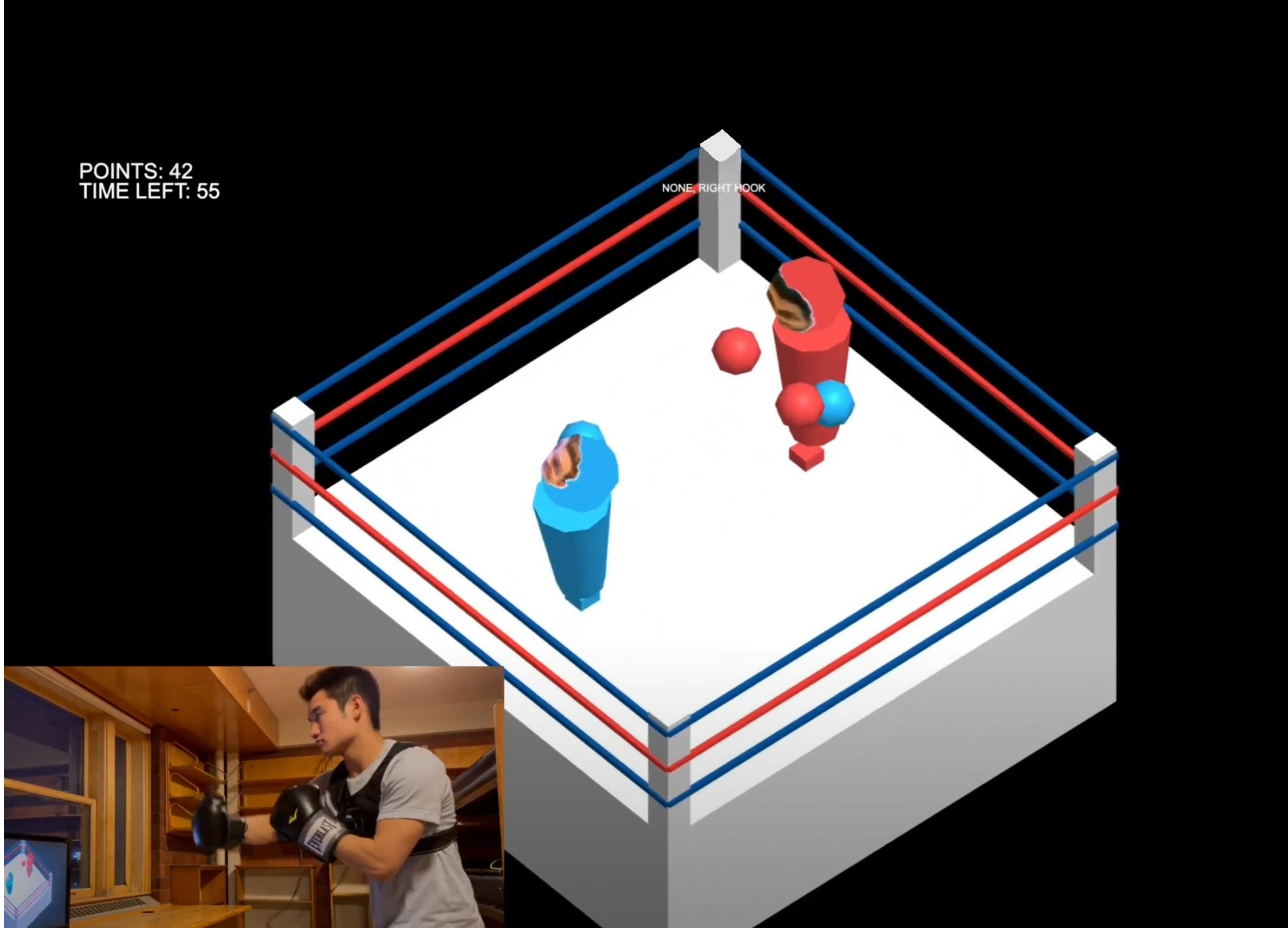

T-TN450

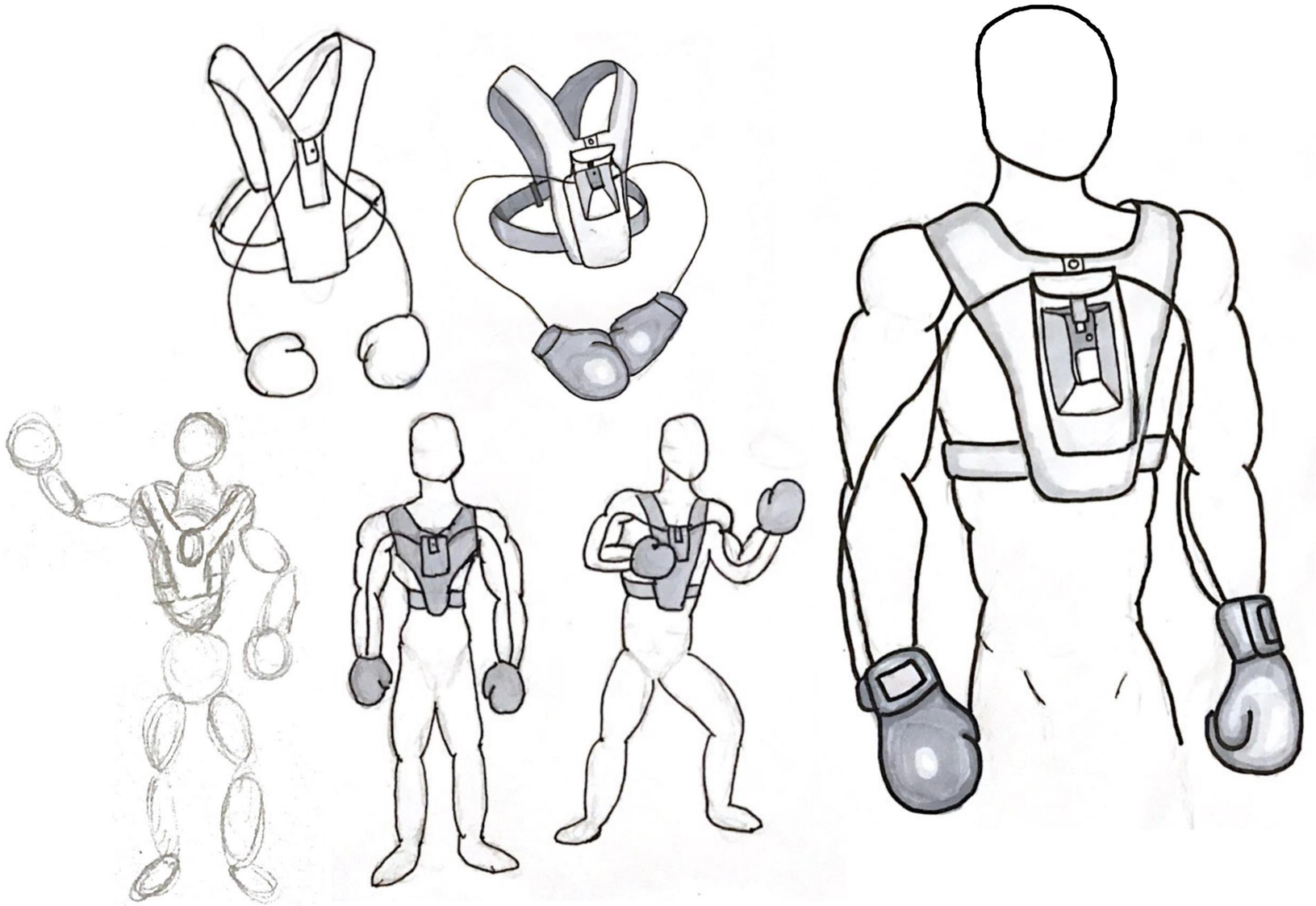
IoT Boxing

6.08 – Embedded Systems
April – May 2021

An IoT version of the iconic Wii Sports Boxing game that uses an ESP32 WiFi MCU and accelerometer to send real time punch motion data over WebSockets to an online 3D boxing game.

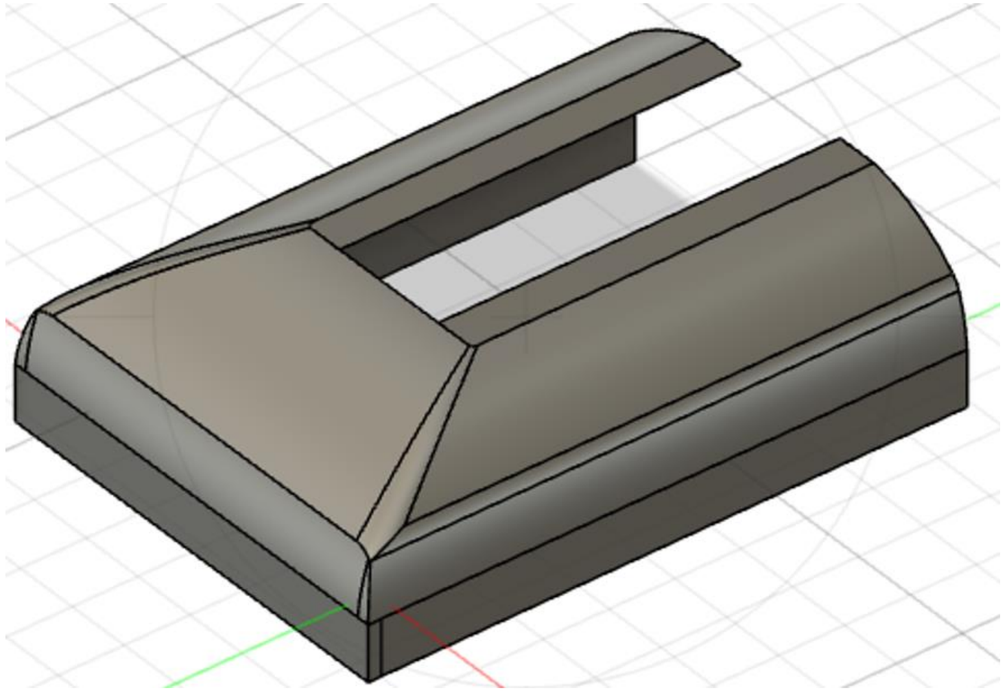
Demo:
https://youtu.be/XnNphWLp-24
Code:
https://github.com/michaellu2019/ragdoll-boxing

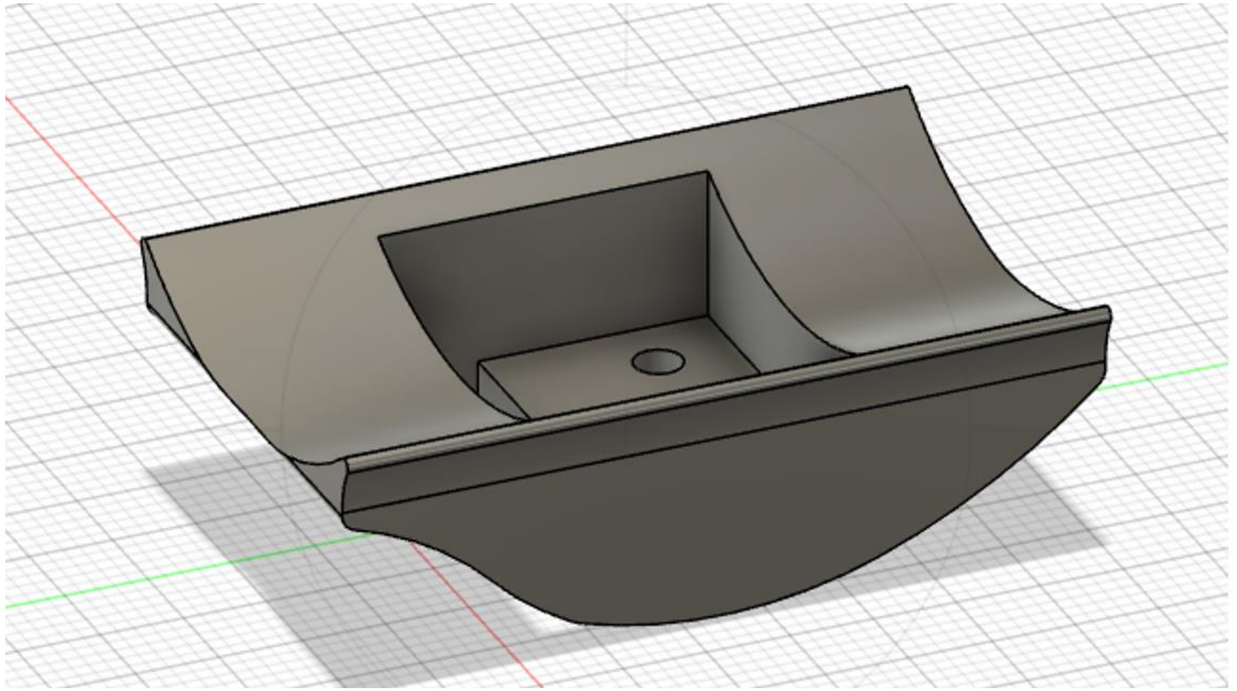




Ideation and Sketching

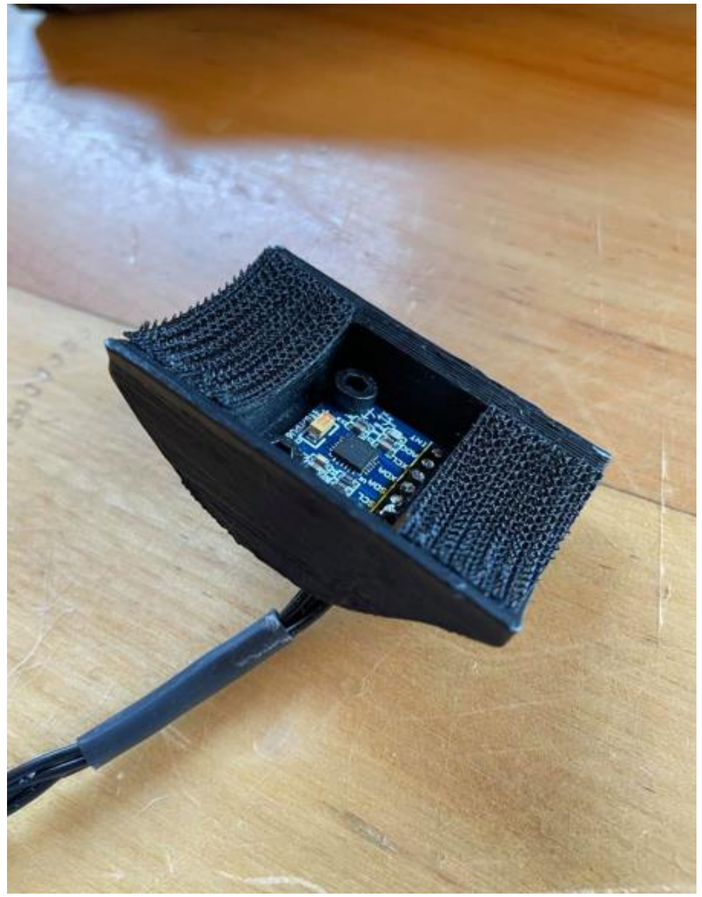
To give the player the most immersive boxing experience, I decided to embed the electronics inside a pair of boxing gloves and a sports vest. The boxing gloves would hold the MPU6050 accelerometers to detect punch movements in real time, and the main electronics, such as the ESP32 and TNTOR Ultra Thin Power Bank, would be mounted inside the sports vest.





3D Modeling

In order to give the wearable game accessories a polished look, the electronics had to blend in with the boxing gloves and sports vest, so custom mounts for the MPU6050 and ESP32 were designed in Autodesk Fusion360 to be 3D printed.

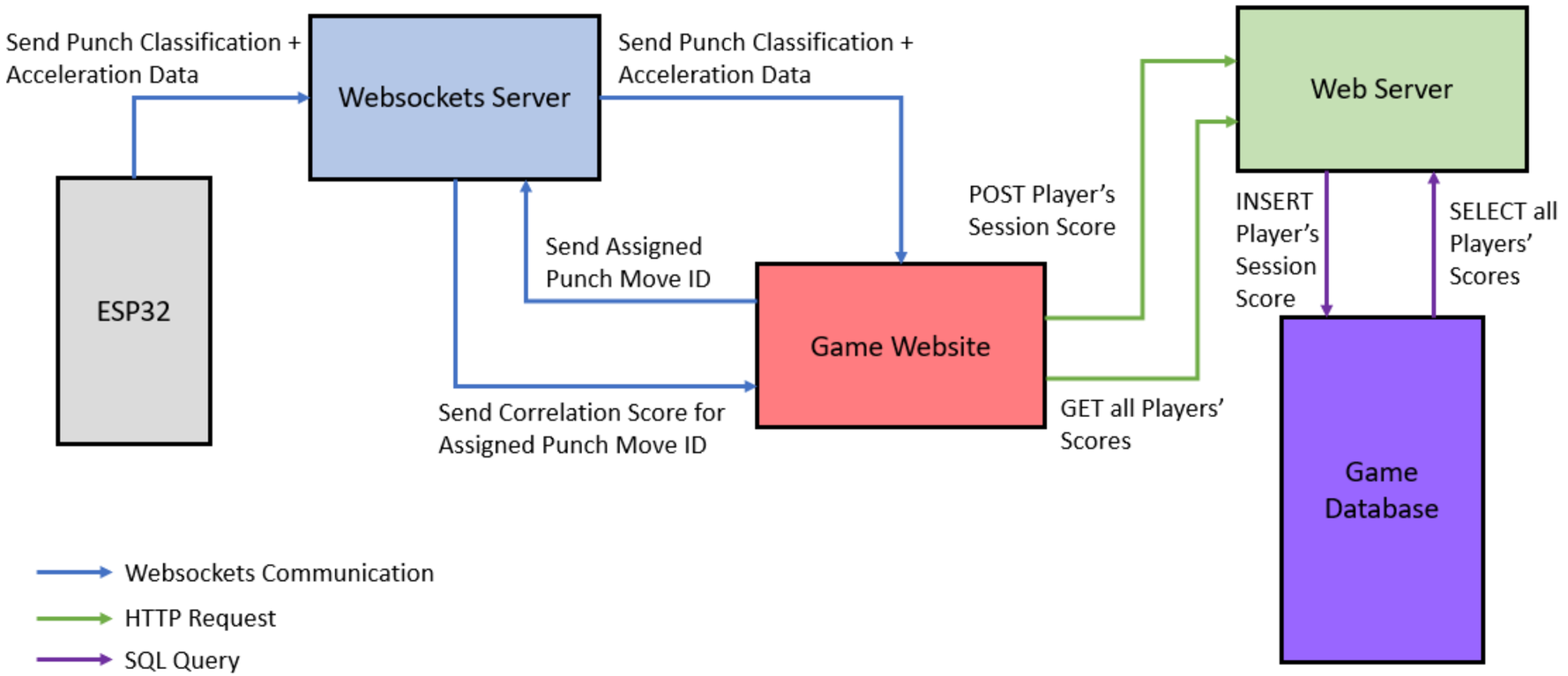


Prototyping

I mounted the 3D-printed electronics housings to the wearable game accessories using Velcro, which would be relatively inconspicuous. In addition, using Velcro would allow the electronics to be easily removed in case they needed to be replaced or modified.

Prototyping

To minimize the electronics and weight in the boxing gloves, the MPU6050s in the boxing gloves were wired to the heavier components—the ESP32 and TNTOR Ultra Thin Power Bank—in the sports vest. This would allow the player to throw punches without feeling the added weight of the electronics.



Programming

The ESP32 ran C code to process the acceleration data from the player's punches and sent them to a Python WebSockets server, which forwarded it to the game website. Written in HTML, CSS, and JavaScript (using Three.js for graphics and Cannon.js for physics), the game website would control a 3D model of a boxer based on the received movements and would also save player scores in a backend database.

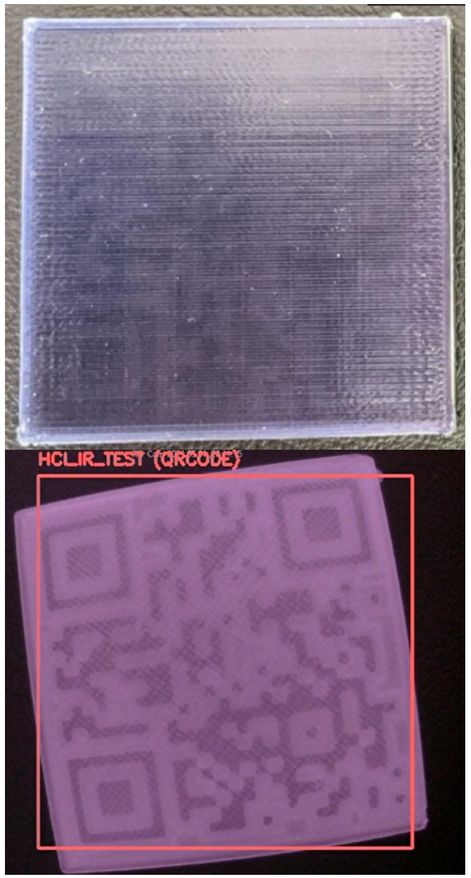
TIME LEFT: 106

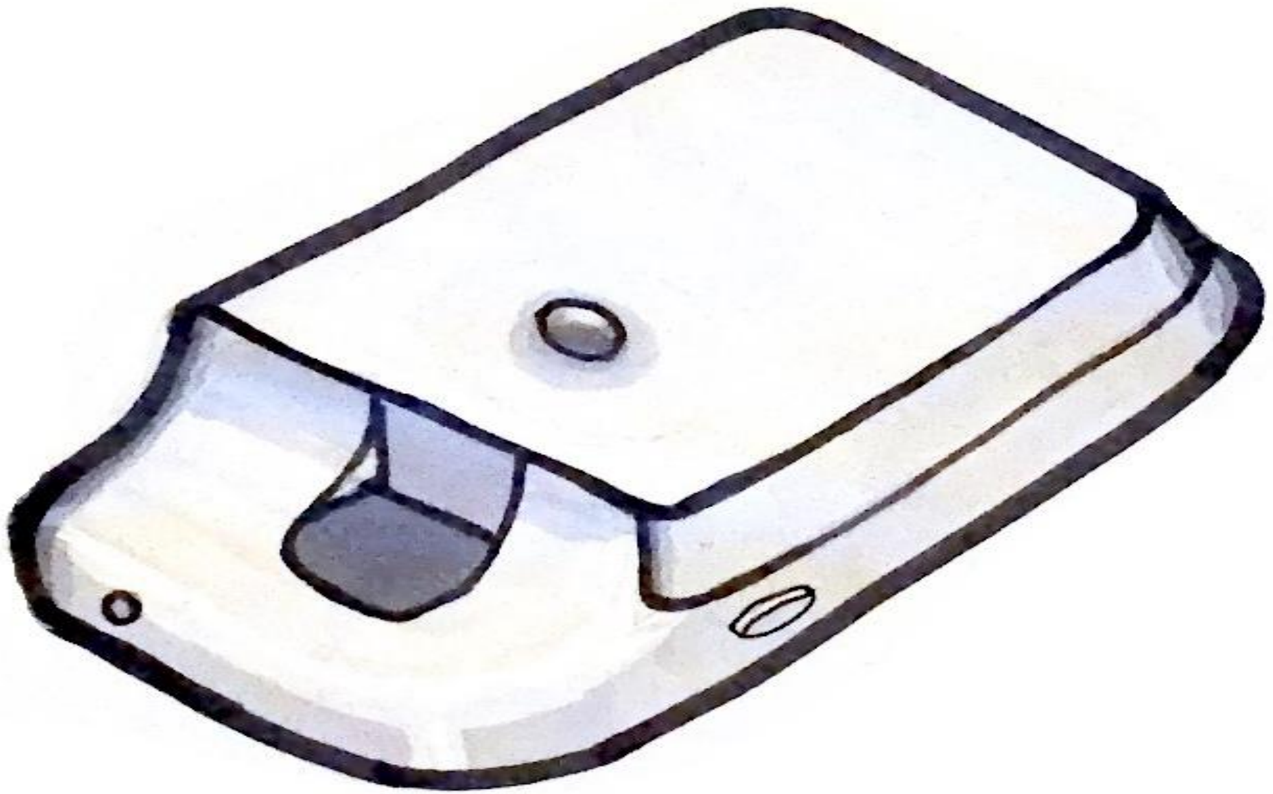
LEFT HOOK, NONE

IR Camera Module

MIT CSAIL HCIE Research Project
February – May 2021

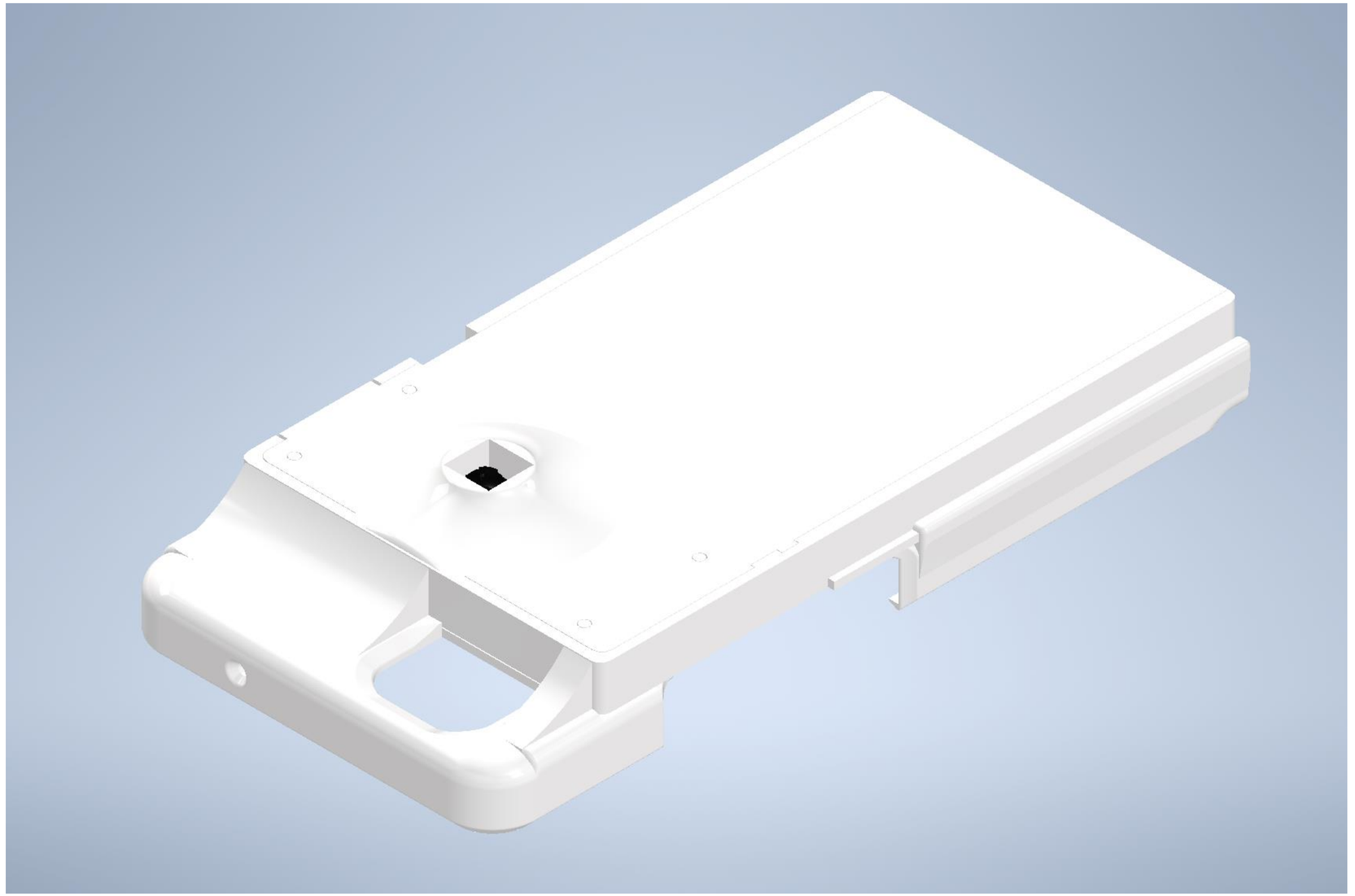
A 3D printed phone case with a Raspberry Pi Zero W infrared camera module that can read QR codes hidden behind infrared-transparent plastic.





Ideation and Sketching

With the ability to use infrared-transparent filament to 3D print hidden QR codes that could only be decoded with an IR camera, I wanted to design a lightweight, thin, removable phone case module that would house an IR camera that could read these hidden QR codes and stream them to my phone.

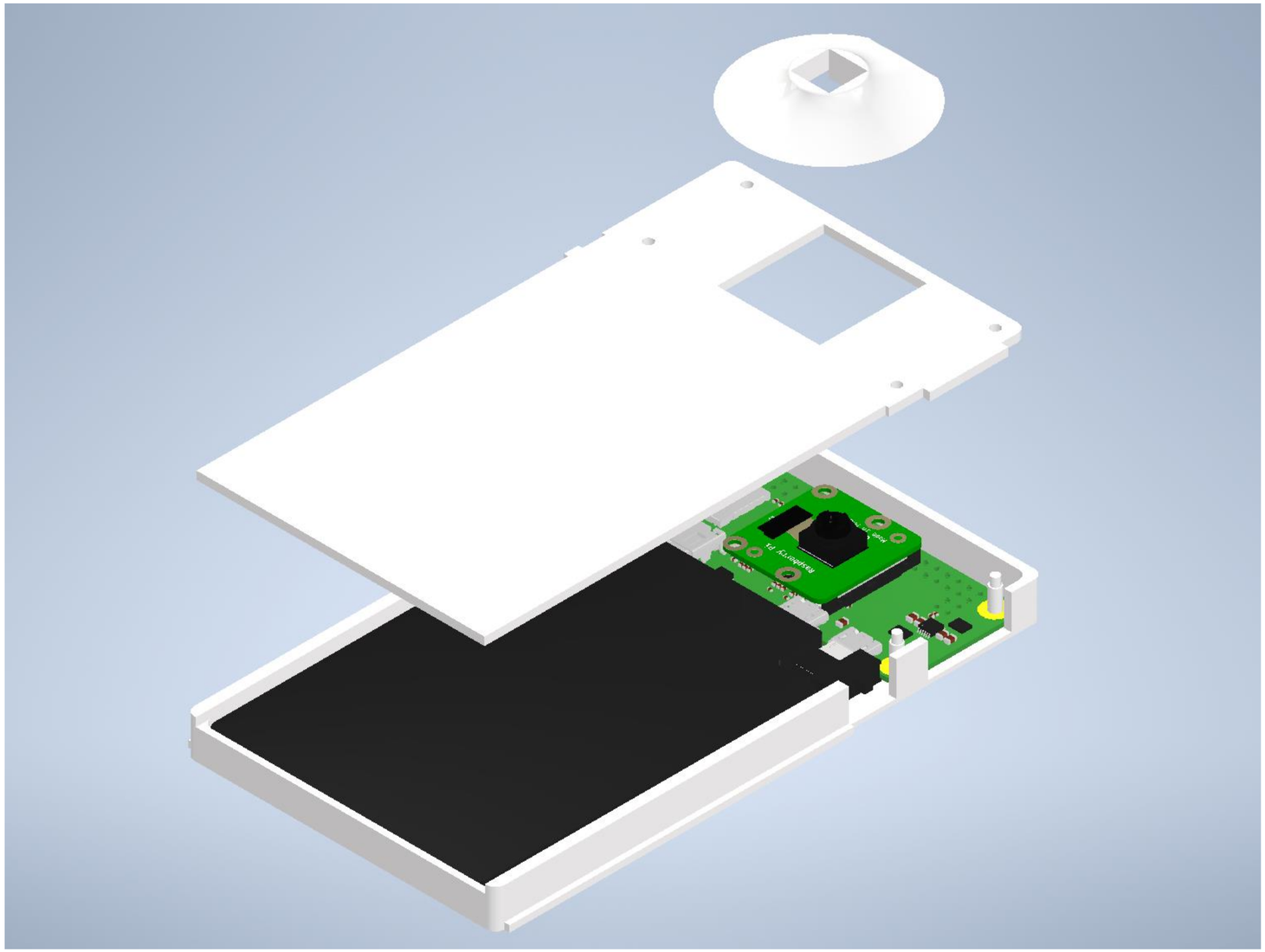


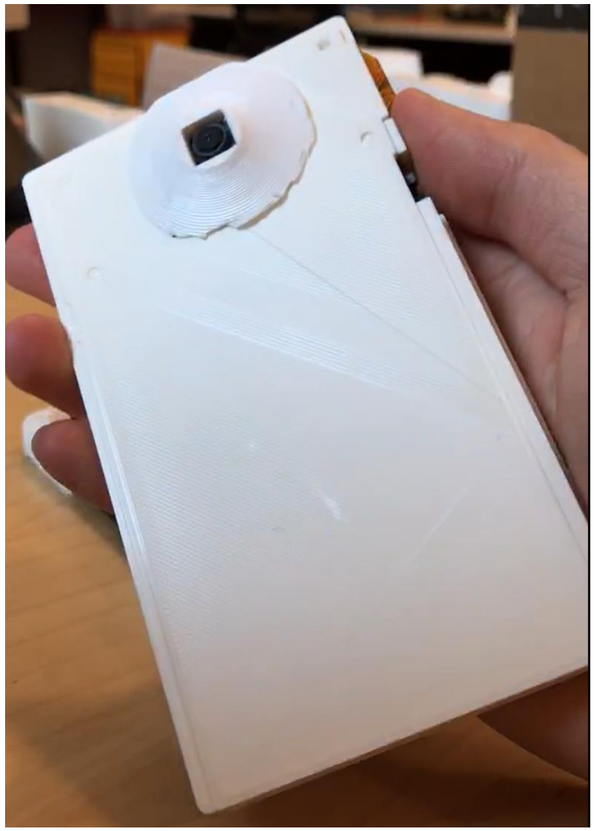
3D Modeling

The IR camera housing would be a removeable module that could slide into small grooves on a custom 3D-printed phone case. To maintain the slim form factor of a smartphone phone, I designed the IR camera module housing to be just a few millimeters thick.

3D Modeling

To keep the IR camera module as small and simple as possible, the main electronic components—Raspberry Pi Zero W, NoIR V2 Camera, and TNTOR Ultra Thin Power Bank—were press tightly together in a case with connecting pegs to secure the module cover, removing the need for any fasteners.



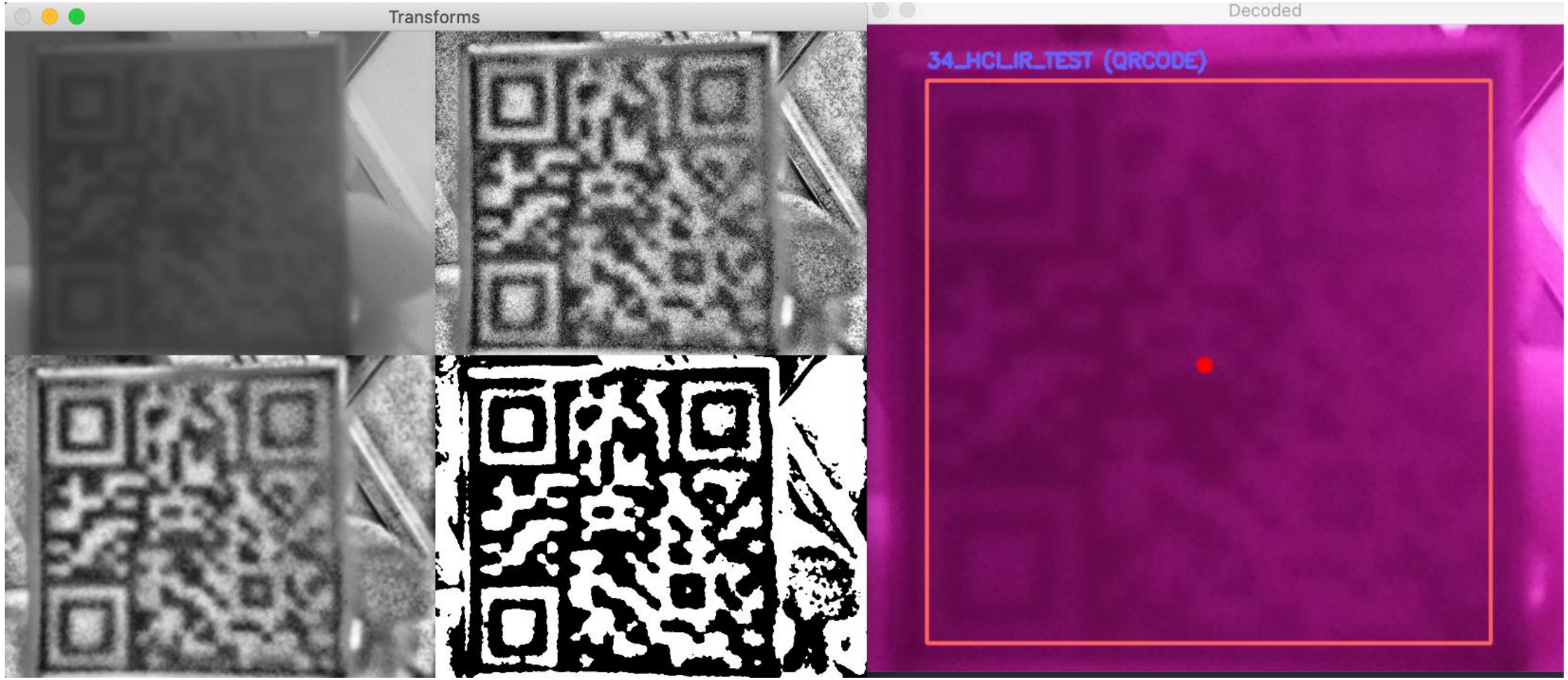


Prototyping

The IR camera module was printed with PLA to give it sufficient rigidity. To achieve a tight fit inside the module, I replaced the TNTOR Ultra Thin Power Bank charging cable with a compact Micro USB and switch assembly. The NoIR camera would fit into the square hole of the camera cap above the Raspberry Pi Zero W.

Prototyping

The IR camera module was able to easily slide into the custom phone case, which was 3D printed with TPU so that it was flexible enough to fit around the phone and camera module.



Programming

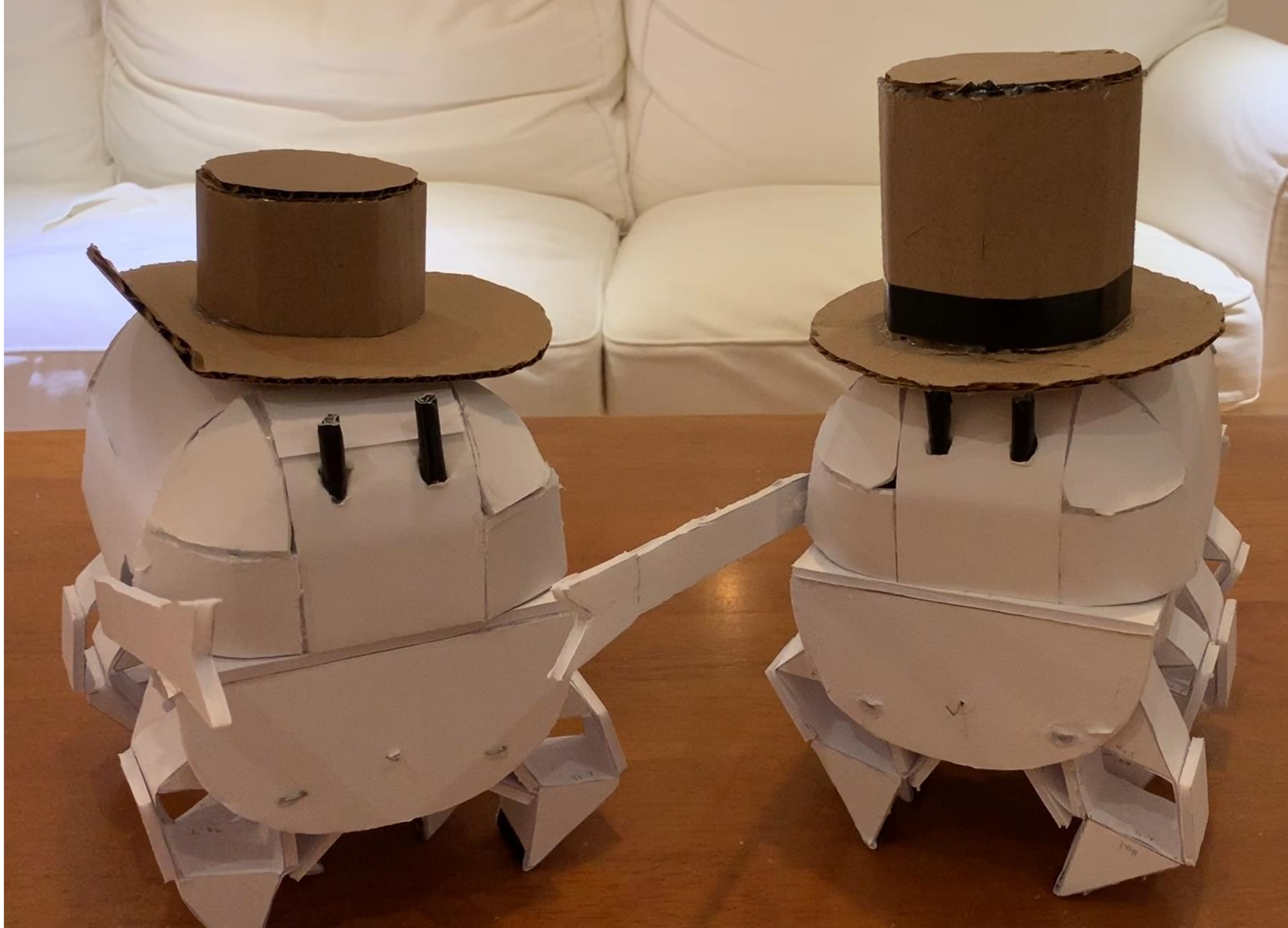
An OpenCV image processing script ran on the Raspbery Pi Zero W, which would take the IR camera output and apply several image transform techniques, including grayscale, Gaussian blur, and contrast limited adaptive histogram equalization (CLAHE), so that it was easier to read the QR code. The results would be streamed in real time to a smartphone app built with the cross-platform Python framework Kivy.

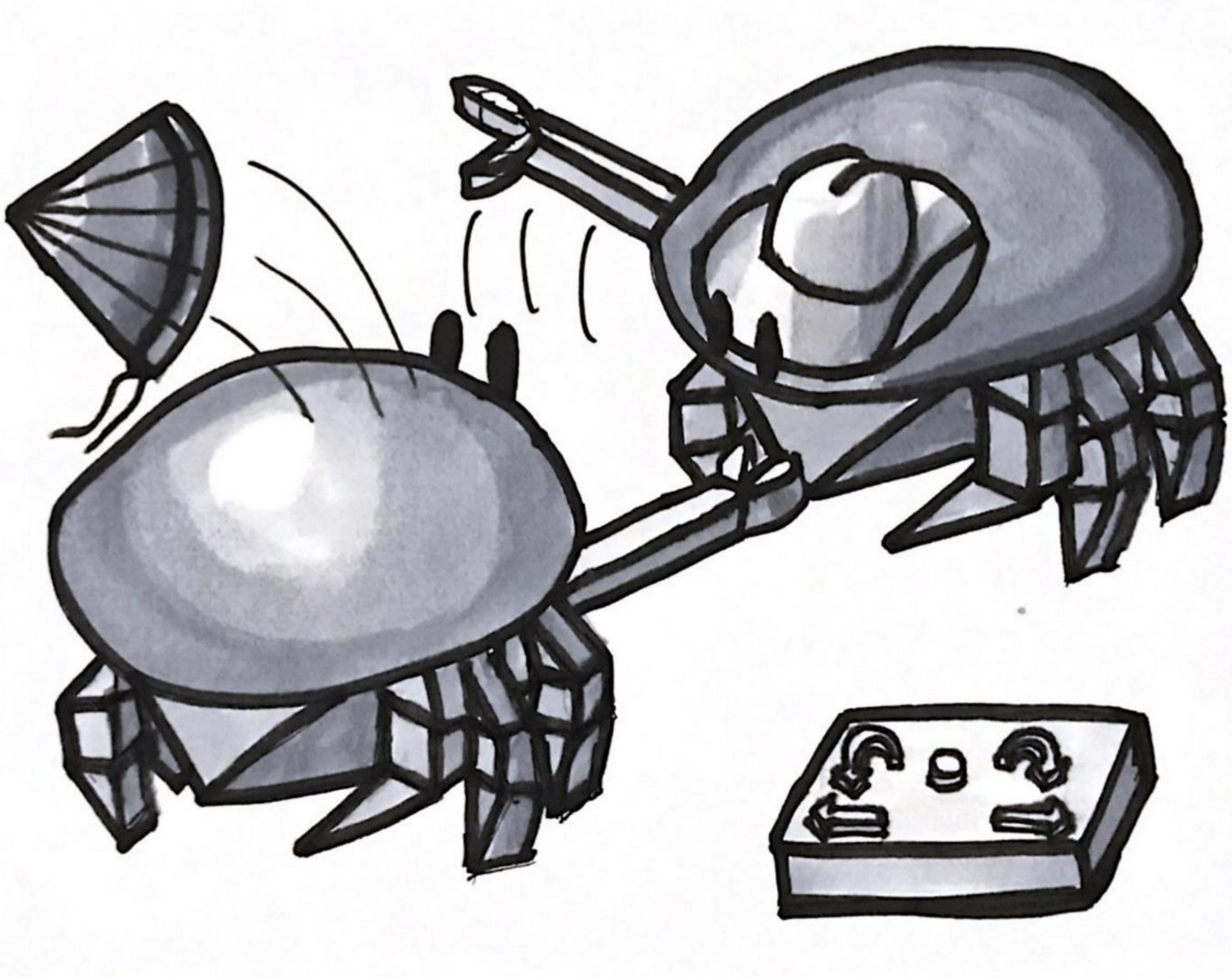
Kombat Krabs

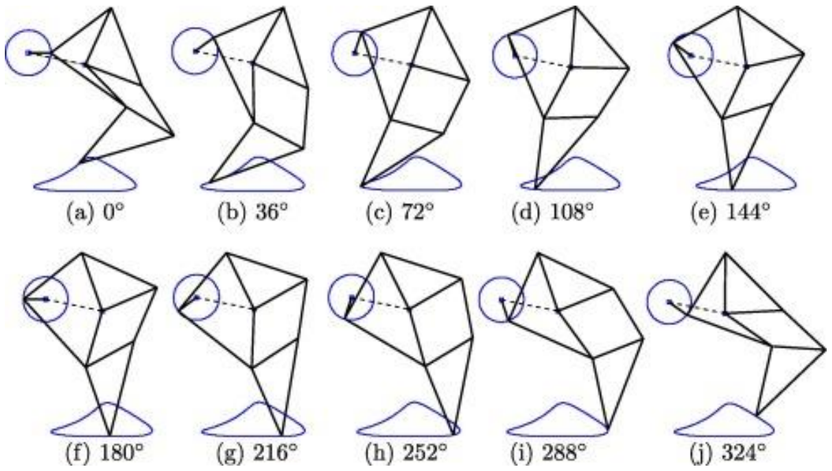
2.00B – Toy Product Design
April – May 2020

Remote-controlled toy robot crabs that walk using the famous leg linkage of kinetic sculptor Theo Jansen. These were built primarily out of foam core, paper, and cardboard.

Demo:
https://youtu.be/xg4couBU9fM
Code:
github.com/michaellu2019/kombat-krabs
CAD:
a360.co/3ePCxNx

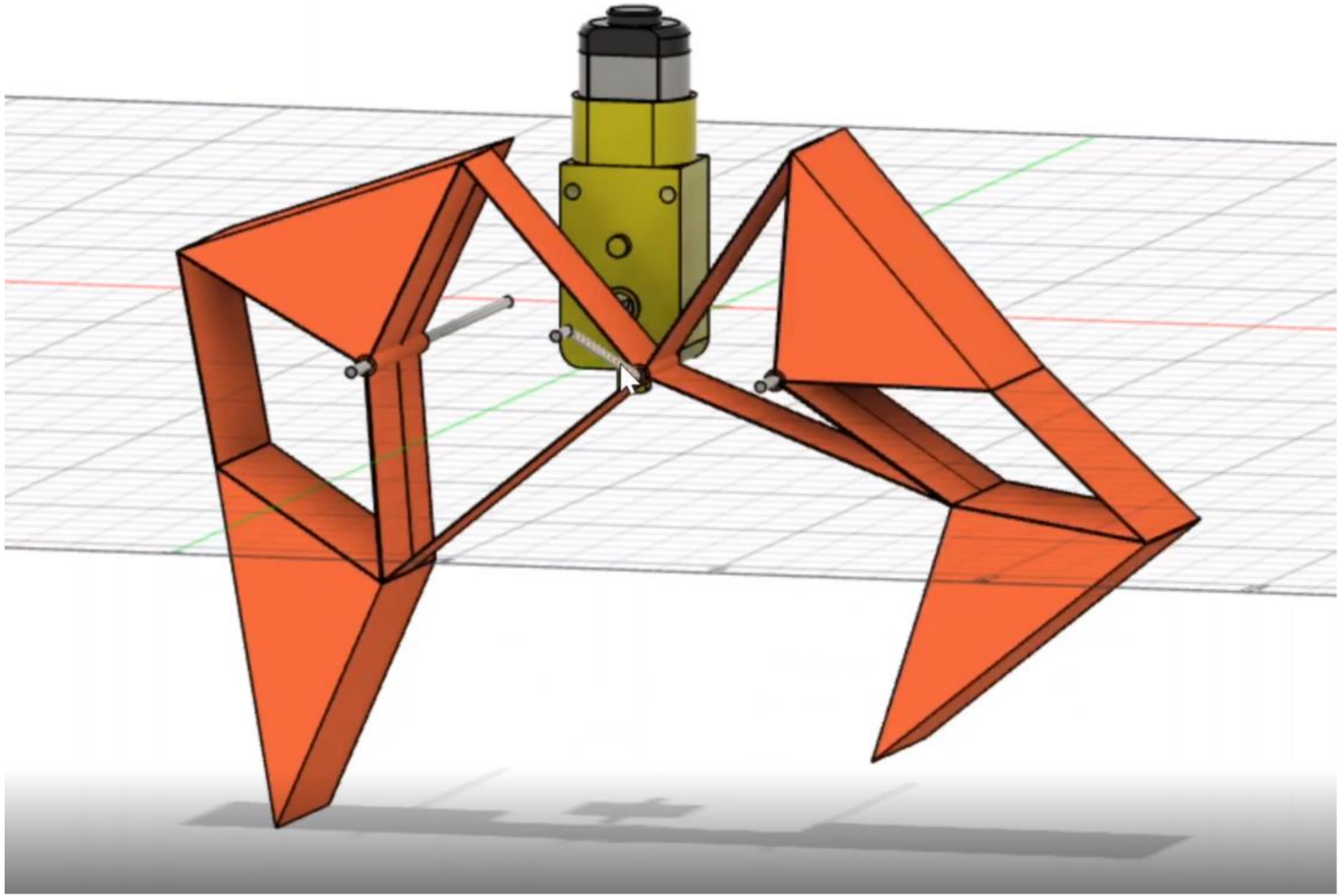






Ideation and Sketching

The biggest challenge of Kombat Krabs was designing a mechanism for the crabs to walk on eight legs. I decided to use Jansen's Linkage, a clever walking mechanism composed of several connected segments that would trace out a smooth walking curve when driven by a circular rotational shaft.

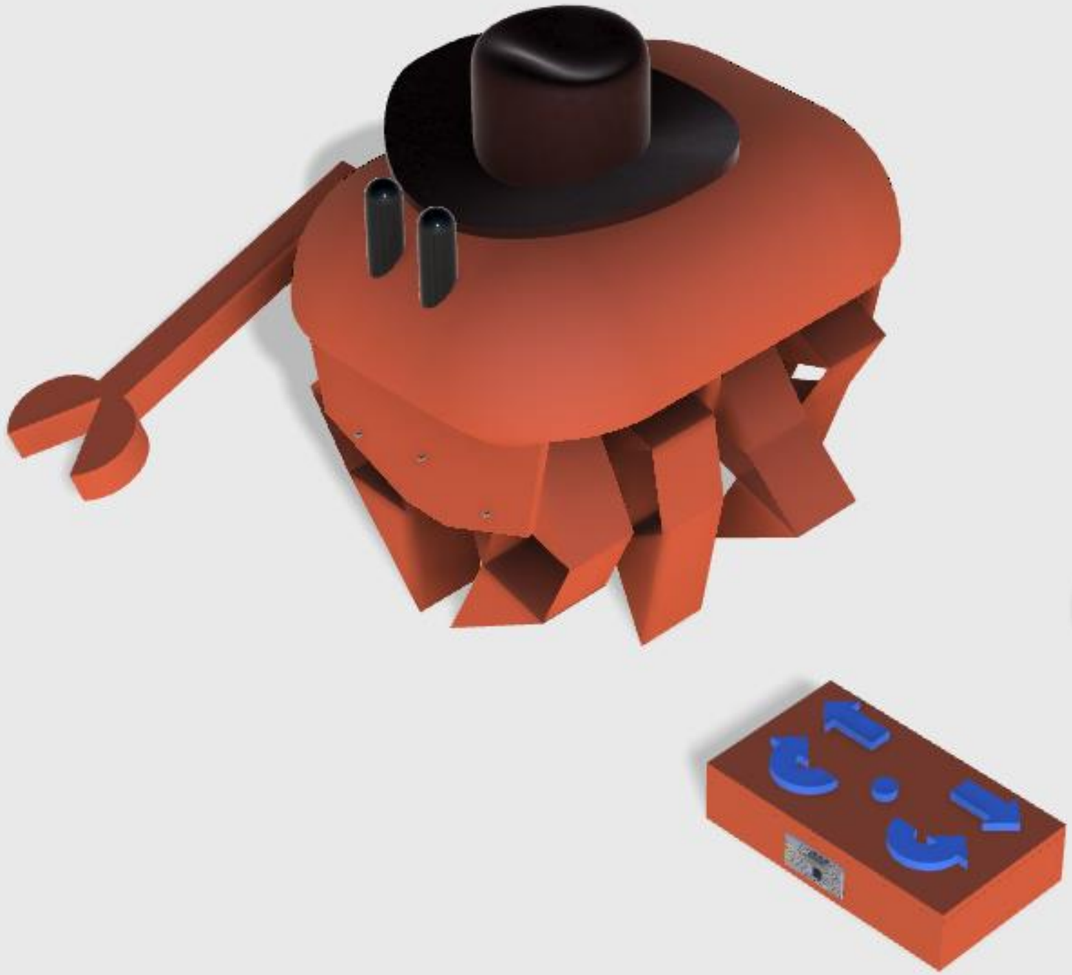


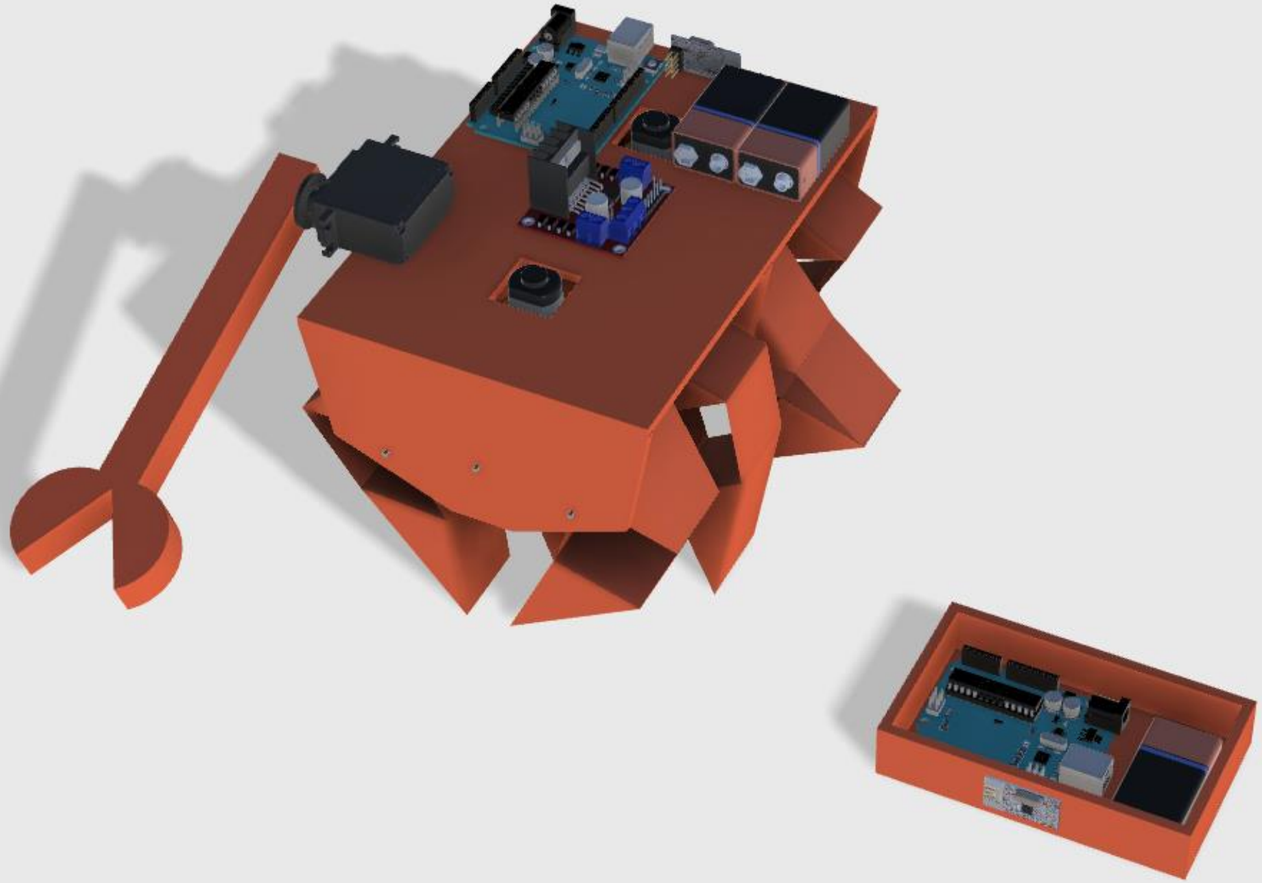
3D Modeling

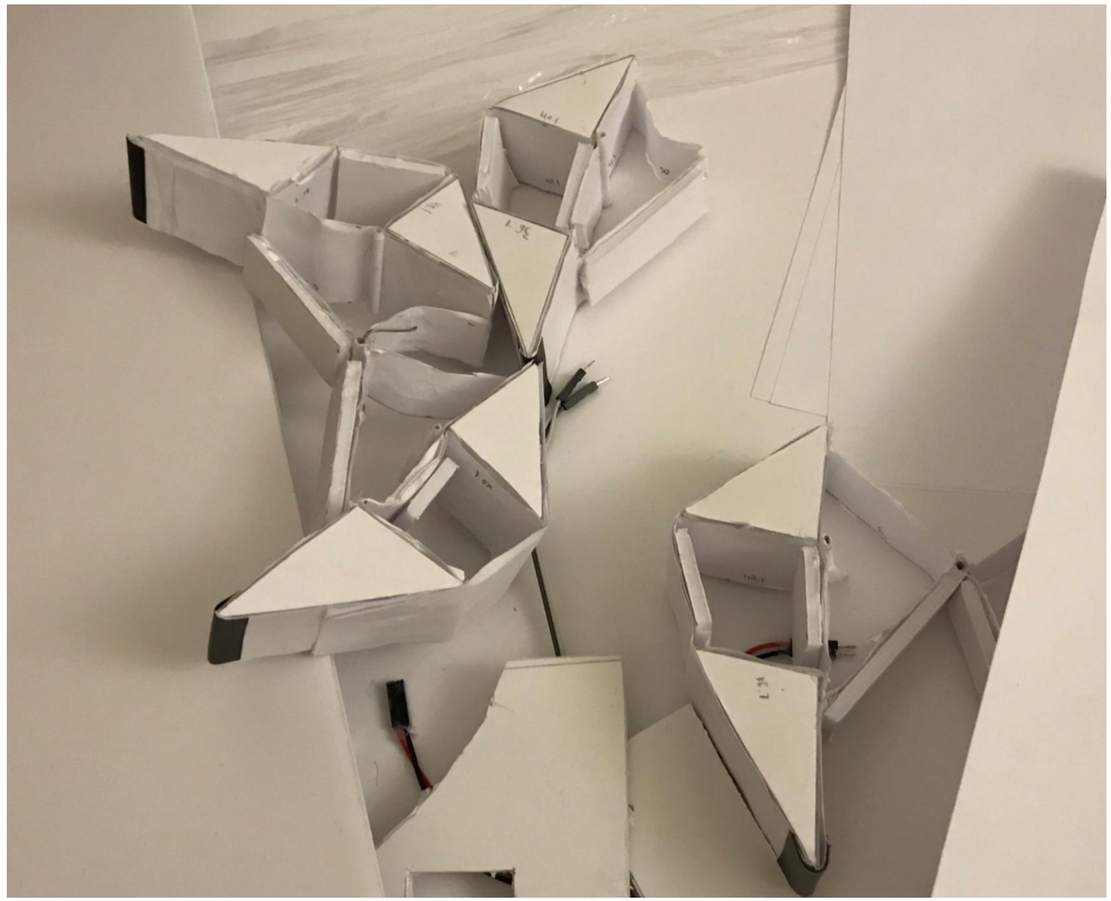
Using Jansen's Linkage, multiple legs could be driven by one motor. Thus, a crab with eight legs would have two motors, each controlling four legs. This would allow crabs to move straight and turn.

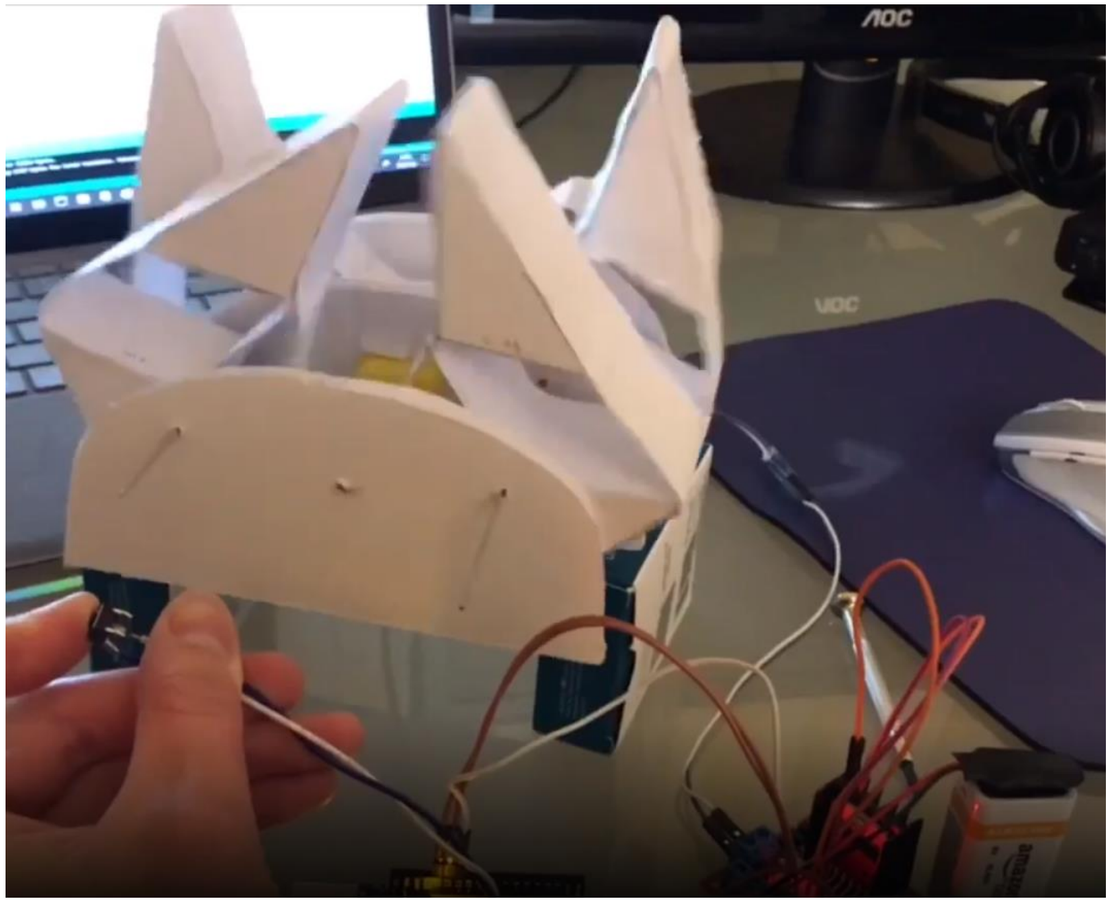
3D Modeling

The entire crab, everything from the leg mechanism to the crab shell to the electronics, was modeled in Autodesk Fusion360.



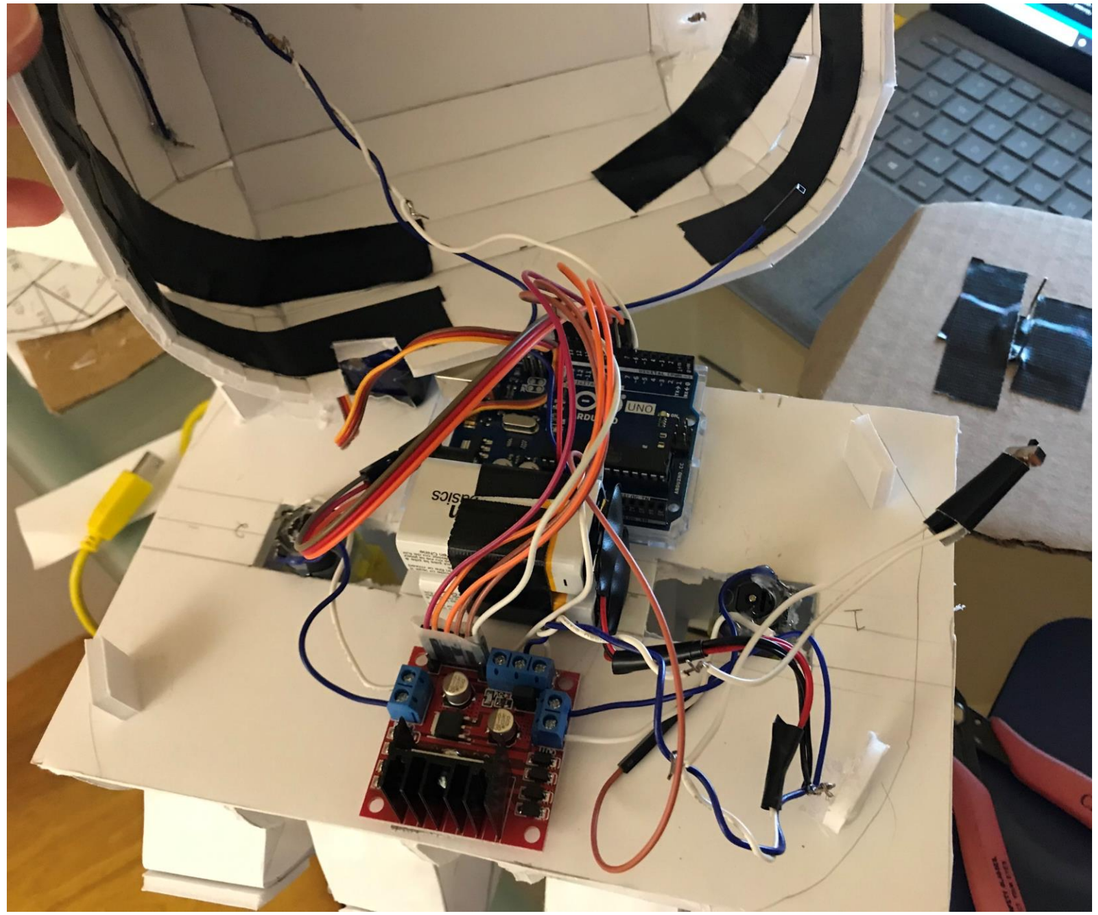


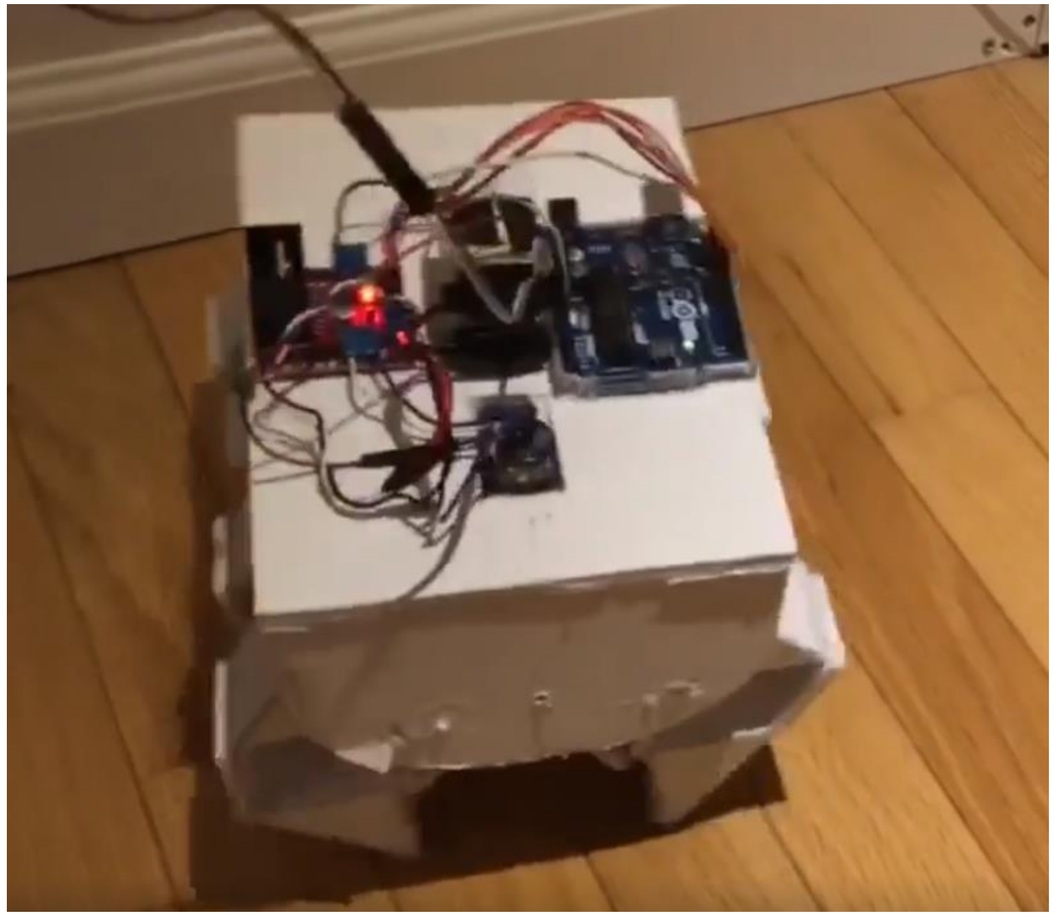




Prototyping

The biggest challenge of prototyping was optimizing the crab's walking curve by tweaking the dimensions of Jansen's linkage for the crab legs. Some linkage segments were shortened to reduce the size of the crab's walking curve, which was important so that the crab could maintain its balance, especially with a high center of gravity.





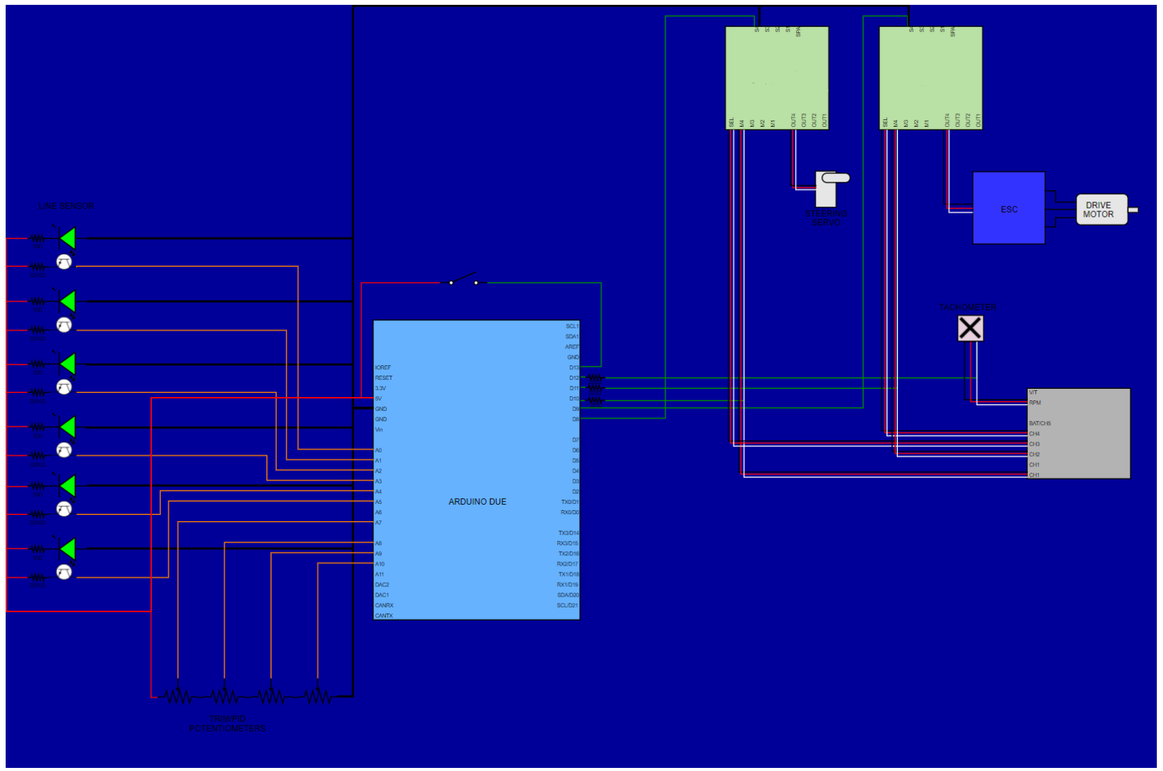
Prototyping

The main crab electronics included an Arduino Uno R3, L298N motor driver, NRF24L01 transceiver, SG90 micro servo motor, two Antrader dual-shaft gear motors, and two 9V batteries.

BeatBot

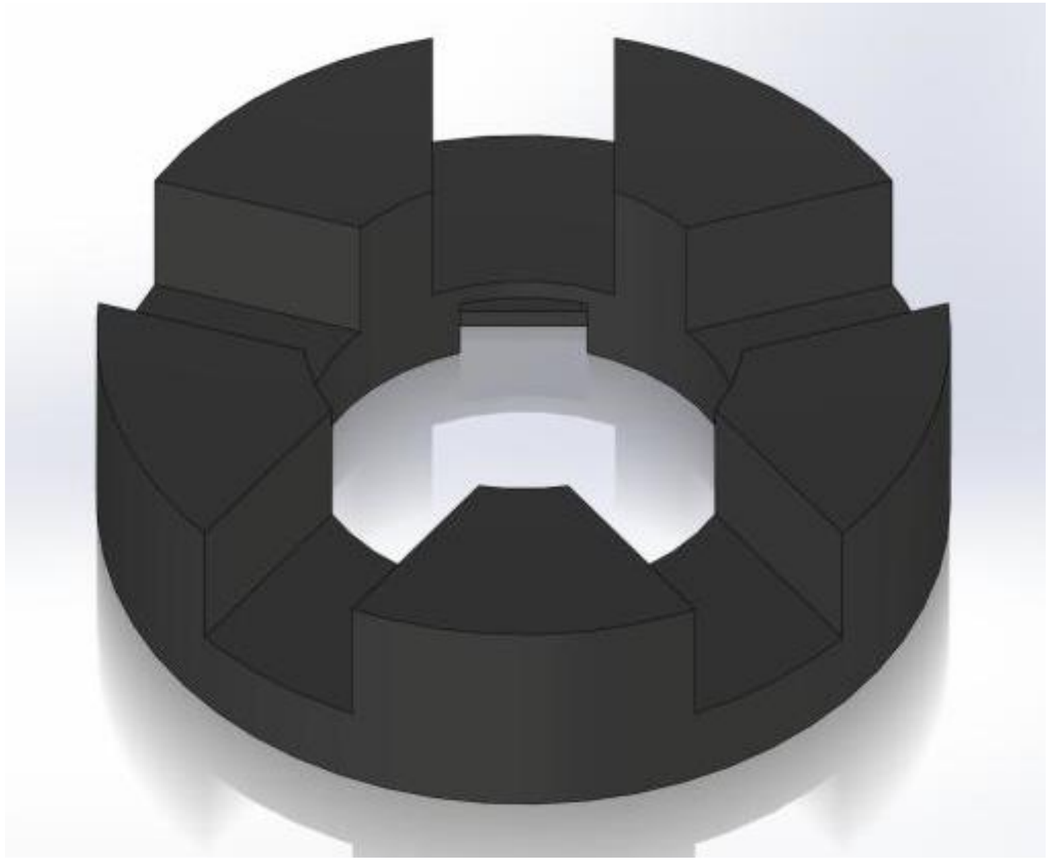
10XBeta
October 2018 – June 2019

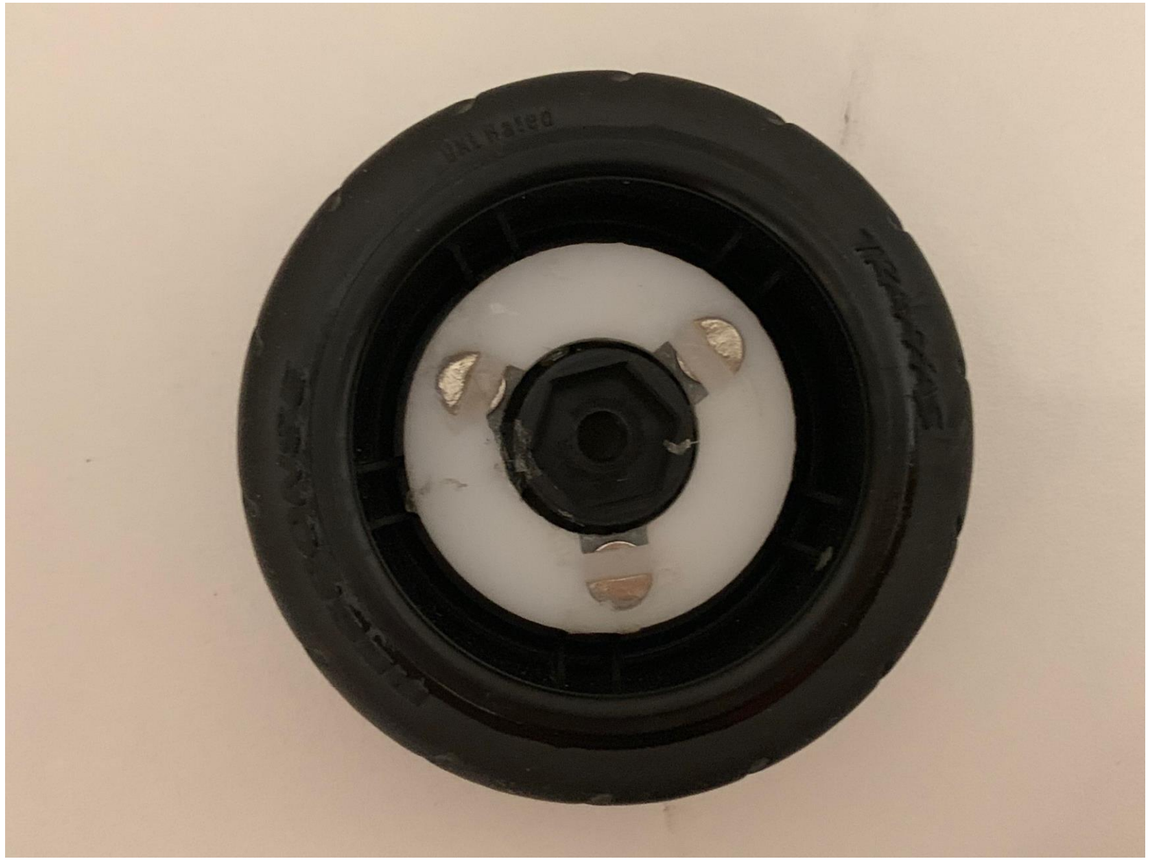
An autonomous line-following car that paces long distance runners along a running track. This was a prototype of the second version for the original PUMA BeatBot built by 10XBeta, JWT, and PUMA in 2016.



Ideation and Designing

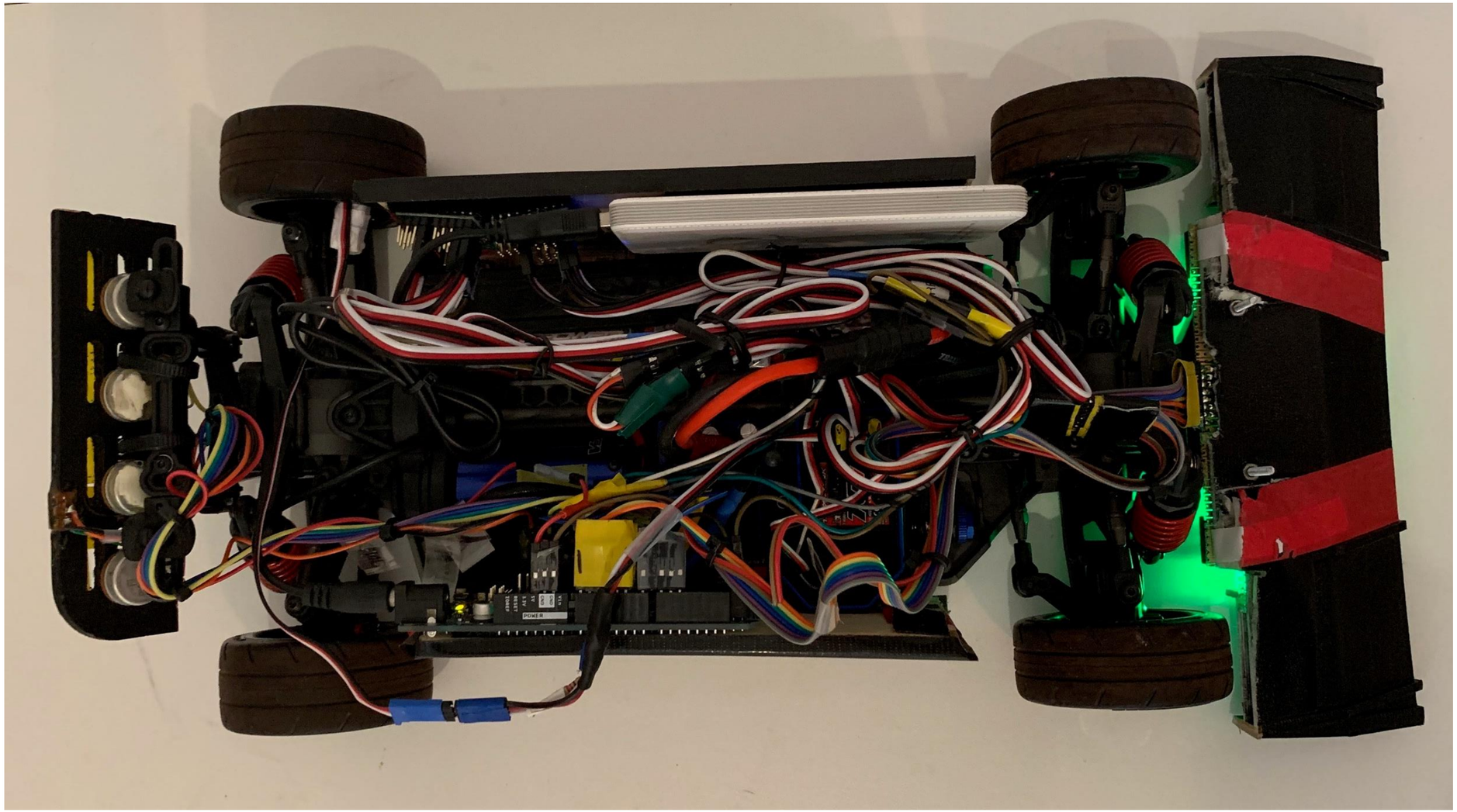
The previous BeatBot had been built on a large, heavy remote-controlled car chassis, which was limiting its maneuverability at high speeds. For the new BeatBot, I decided to go with a smaller and lighter chassis. The challenge thus became fitting the numerous electronic components on a smaller chassis.





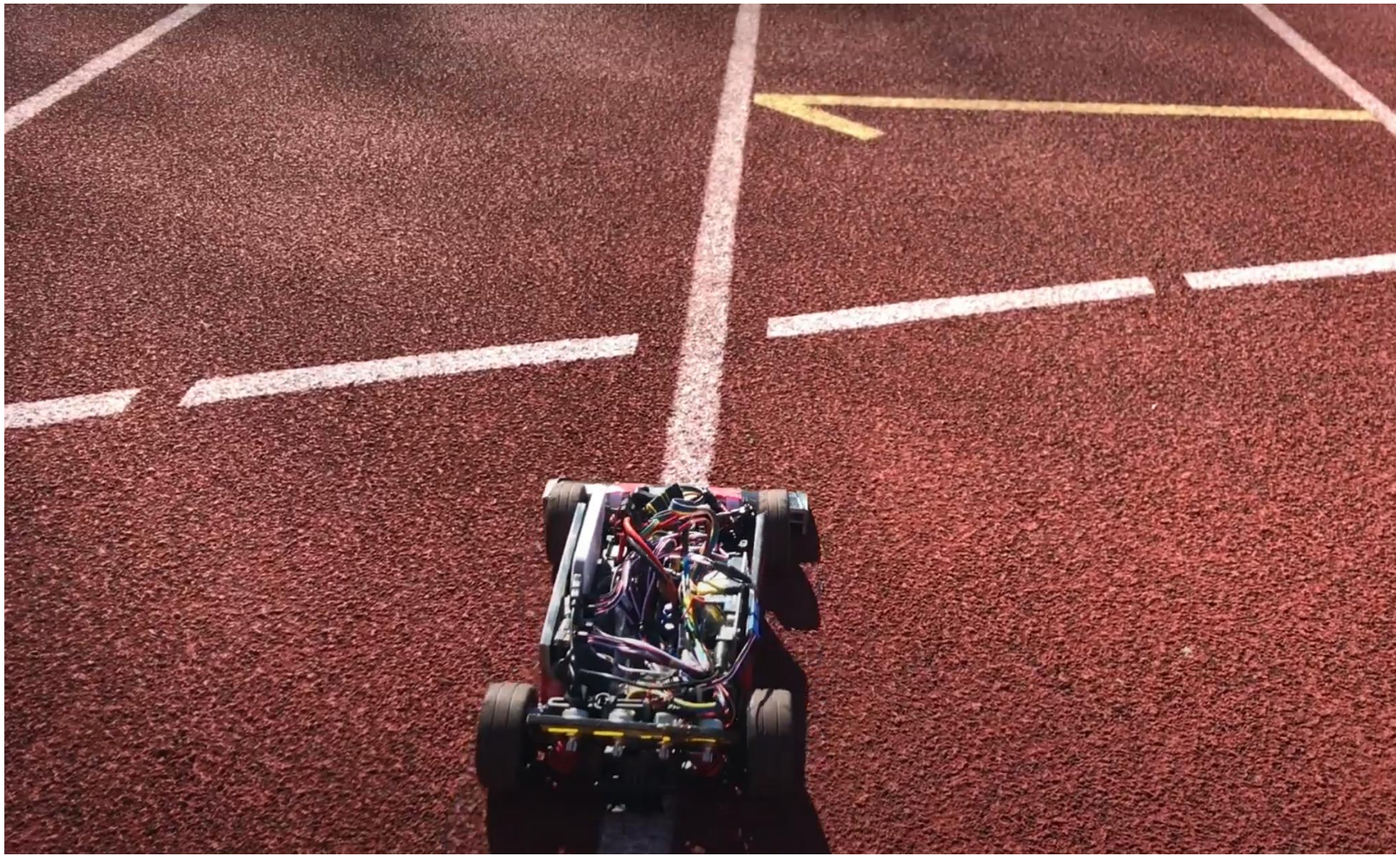
3D Modeling

Because a new chassis was being used, I had to redesign many sensor mounts in Solidworks. For instance, the tachometer magnets used to measure wheel rotations would be housed in a new custom SLA 3D-printed wheel mount that would clip onto the car wheel spokes.



Prototyping

In addition to changing the chassis, I switched many metal pieces of the old BeatBot to 3D-printed plastic pieces that were strong yet light. This change ultimately reduced the prototype weight by 34% and size by 40%.



Programming

The line-following feature of the BeatBot relied on a PID loop in its software. A large portion of the prototype development cycle was dedicated to retuning the PID values for the new car chassis.

Thank You

github.com/michaellu2019 mlu0708@mit.edu linkedin.com/in/michael-lee-lu