BruinBot:
UCLA's Companion Robot

ASME at UCLA: X1 Robotics

Technical Design Report

0 Table of Contents

1 Introduction

1.1 About X1 Robotics

Founded in 2015, X1 Robotics is one of the University of California, Los Angeles (UCLA) Engineering's student-led project teams with the mission of promoting real world engineering and problem solving skills through the development of challenging robotic systems. Falling under the umbrella organization of the American Society of Mechanical Engineers (ASME) UCLA Student Branch, what sets us apart is our passion and creativity; every year marks the start of a new project decided upon by the diverse team of student engineers, and a new cycle of research, development, and iteration. X1 is proud of the collaborative work environment it fosters and will continue to excel in the field of robotics as it attempts complex, challenging projects pushing UCLA students to their limits.

1.2 The 2019-2020 Team

During the 2019-2020 school year, X1 Robotics' resources were divided between two projects: the Guardian V2 and BruinBot. The Guardian was a continuation of the 2018-2019 project, while BruinBot was a new concept pitched in 2019. Overall, the club utilized 2 project leads, 6 subteam leads, 2 subteam mentors, and 54 total members distributed between the Guardian and BruinBot. Applications were sent out at the beginning of the year, and the team was assembled based on prior knowledge, diversity of experience, and willingness to commit through completion. A full list of the members responsible for completion of BruinBot is as follows:

John Tabakian X1 Project Lead *3rd Year Mechanical Engineering Major*	**Akaash Venkat** X1 Project Lead *4th Year Computer Science Major*

Rebecca Celsi Chassis Subteam Lead *2nd Year Mechanical Engineering Major*	**Junwei He** Chassis Subteam Member *3rd Year Mechanical Engineering Major*
Karina Ting Chassis Subteam Member *2nd Year Mechanical Engineering Major*	**Michael Mitroshin** Chassis Subteam Member *2nd Year Mechanical Engineering Major*
Nicholas Godshall Chassis Subteam Member *3rd Year Mechanical Engineering Major*	**Cole Rodgers** Payload Subteam Lead *3rd Year Mechanical Engineering Major*
Bethany Chang Payload Subteam Member *3rd Year Mechanical Engineering Major*	**Conor Mc Gartoll** Payload Subteam Member *2nd Year Computer Science Major*

Justeen Hipolito Payload Subteam Member *4th Year Mechanical Engineering Major*	**Kevin McDougal** Payload Subteam Member *3rd Mechanical Engineering Major*
Kyle McKim Payload Subteam Member *3rd Mechanical Engineering Major*	**Kenneth Vuong** Drivetrain Subteam Lead *2nd Year Mechanical Engineering Major*
Jeanine Lee Drivetrain Subteam Member *3rd Year Mechanical Engineering Major*	**Marshall Qian** Drivetrain Subteam Member *2nd Year Mechanical Engineering Major*
Michelle Kim Drivetrain Subteam Member *2nd Year Mechanical Engineering Major*	**Mihir Sharma** Drivetrain Subteam Member *4th Year Aerospace Engineering Major*
Zak Benjamin Rubio Drivetrain Subteam Member *2nd Year Mechanical Engineering Major*	**Jingbin Huang** Controls Subteam Lead *4th Year Electrical Engineering Major*
Jacob Sayono Controls Subteam Member *3rd Year Mechanical Engineering Major*	**James King** Controls Subteam Member *2nd Year Computer Science and Engineering Major*
Malcolm Francisco Controls Subteam Member *2nd Year Electrical Engineering Major*	**Michael Buck** Controls Subteam Member *2nd Year Electrical Engineering Major*
Olivia Loh Controls Subteam Member *2nd Year Computer Engineering Major*	**Anirudh Balasubramaniam** Vision/AI Subteam Lead *3rd Year Computer Science and Engineering Major*
Denny Tsai Vision/AI Subteam Member *3rd Year Computer Engineering Major*	**Dhruv Singhania** Vision/AI Subteam Member *2nd Year Computer Science Major*
Nikhil Kumar Vision/AI Subteam Member *2nd Year Computer Science Major*	**Frank Xing** UI/UX Subteam Lead *2nd Year Computer Science Major*

1.3 The 2020-2021 Team

During the 2020-2021 school year, the BruinBot project continued developments with the primary intent of refining existing designs and preparing for manufacturing once campus restrictions due to COVID-19 were relaxed. The team this year was much smaller, since much of the remaining progress could only be completed in-person. A full list of the team responsible for the continued development of BruinBot is as follows:

Rebecca Celsi X1 Project Lead *3rd Year Mechanical Engineering Major*	**Kenneth Vuong** X1 Project Lead *3rd Year Mechanical Engineering Major*

Zachary Yuan Mechanical Subteam Lead *3rd Year Mechanical Engineering Major*	**Sonika Sethi** Mechanical Subteam Member *3rd Year Mechanical Engineering Major*
Anthony Chung Mechanical Subteam Member *2nd Year Mechanical Engineering Major*	**Nikhil Kumar** Software Subteam Lead *3rd Year Computer Science Major*
Ryan Daly Software Subteam Member *2nd Year Computer Science Major*	

1.4 The 2021-2022 Team

In the 2021-2022 year, campus activities were resumed and work on BruinBot continued with a heightened pace and intensity with the goal of completely finalizing, building, and testing the robot within a year. Despite the efforts made in previous years, a huge amount of work remained and a full team of students within Mechanical Engineering, Electrical Engineering, and Computer Science was brought on to finalize the project. The full list of students involved during 2021-2022 is as follows:

Rebecca Celsi X1 Project Lead *4th Year Mechanical Engineering Major*	**Matt Lacaire** X1 Project Lead *3rd Year Mechanical Engineering Major*

Zachary Yuan Mechanical Subteam Lead *4th Year Mechanical Engineering Major*	**Sonika Sethi** Mechanical Subteam Member *4th Year Mechanical Engineering Major*
Anthony Chung Mechanical Subteam Member *3rd Year Mechanical Engineering Major*	**Karina Ting** Mechanical Subteam Member *4th Year Mechanical Engineering Major*
Katusch Strich Mechanical Subteam Member *3rd Year Mechanical Engineering Major*	**Vivian To** Mechanical Subteam Member *4th Year Aerospace Engineering Major*
Alexander Pak Mechanical Subteam Member *3rd Year Mechanical Engineering Major*	**Anand Janev** Mechanical Subteam Member *3rd Year Mechanical Engineering Major*
Daniel Farzannekou Mechanical Subteam Member	**Jaeinn Lee** Mechanical Subteam Member

4th Year Mechanical Engineering Major	*4th Year Mechanical Engineering Major*
Lorenzo Ontiveros Mechanical Subteam Member *4th Year Mechanical Engineering Major*	**Dylan Del Rosario** Electrical Subteam Lead *4th Year Electrical Engineering Major*
Avi Gerber Electrical Subteam Member *1st Year Mechanical Engineering Major*	**Sohun Patel** Electrical Subteam Member *3rd Year Mechanical Engineering Major*
Lawrence Liu Electrical Subteam Member *1st Year Electrical Engineering Major*	**Nikhil Kumar** Software Subteam Lead *4th Year Computer Science Major*
Jeremy Lin Software Subteam Member *Master Mechanical Engineering Major*	**Ryan Daly** Software Subteam Member *3rd Year Computer Science Major*
Brian Burrous Software Subteam Member *3rd Year Mechanical Engineering Major*	**Ky Heon** Software Subteam Member *3rd Year Mechanical Engineering Major*

1.5 Design Objectives and History

BruinBot was inspired by a combination of package delivery robots and interactive robots, such as those found in amusement parks or medical settings. BruinBot is intended to be UCLA's very own friendly robotic companion, spreading cheer and goodwill by traveling across campus and interacting with students.

The original design objectives were laid out in 2019 and underwent several modifications as the project progressed. The original objectives were as follows:

> "BruinBot is intended to travel around UCLA campus and interact with students in various ways. A drivetrain system will allow BruinBot to navigate variable terrain conditions, while sensors and GPS will be employed to allow the robot to autonomously traverse certain campus pathways. A payload system will be interfaced through touchscreen and demonstrates the main interactive aspect of BruinBot, allowing users to receive various snacks and gifts from BruinBot. Additionally, facial and voice detection will be employed to accomplish more personalized interactions. Finally, a robust chassis is necessary to withstand possible external forces and protect internal components."

The explicit design objectives of BruinBot were given as:

1. **Successfully navigate specified areas of UCLA campus (including Bruinwalk) using cameras, GPS, and other sensors.**
2. **Autonomously detect people and initiate interactions.**
3. **Interact with users through both touchscreen and vocal commands.**
4. **Understand and reply to certain voice commands, such as: "Who are you?", "How is your day?", or "Tell me a joke."**
5. **Dispense various items, such as candy bars or flyers, using a payload system and mechanical arm.**

Over the span of the project, several of these features were modified long before they were incorporated. A few major changes include:

1. **Navigation Changes**
    - The navigation feature was almost entirely scrapped due to the inherent difficulty of integrating fully autonomous robotic navigation. Using GPS alone would not be sufficient to navigate the robot safely through a complex area such as campus. In reality, full autonomous navigation would require an integration of GPS, computer vision, LiDAR or other distance sensing, and software such as SLAM (Simultaneous Localization And Mapping). This task was determined to be too complex for our team's capabilities.
    - Instead of pursuing autonomous campus navigation, we decided instead to modify our goal so that BruinBot could display enough movement to appear 'lifelike' without actually attempting to navigate a mapped space. We decided on instead implementing a "wander" mode where BruinBot would perform a series of random turns and movements while waiting to initiate a student interaction. This way, BruinBot can still appear to have full mobility, but will stay within a designated small area. The robot will still use a LiDAR system as a failsafe to prevent collision with objects in its space, in case it comes up against a wall, drop, or other obstacle.
    - The "wander" mode and student interactions are still fully autonomous, and the robot is capable of operating in this state indefinitely without outside control. However, for ease of transportation, the robot can also be controlled remotely by a person via the web dashboard. This allows a supervisor to "drive" the robot from place to place, and also access other features such as emotions, touchscreen functions, and payload dispensing for debugging purposes.
2. **Payload Changes**
    - The payload system underwent many changes, both in its physical design as well as modifications to the system's overall goals. Initially, the payload system was intended to include a dispensing system for small, rectangular objects (like granola bars or candy bars) as well as a paper dispenser for flyers or business cards. Both of the robot's arms were intended as functional parts of the payload system, with additional features like high-fiving or fist-bumping discussed in early 2019. Eventually, the paper dispensing system was scrapped to focus the team's efforts on creating a functional solid-object

dispensing system. Likewise, the right arm of the robot was designed to have no functionality, merely mirroring the left arm's appearance for symmetry.

3. **Future work (features yet to be fully implemented)**
    - Other than the items discussed above, the robot in its finished state will have all of the goals set forth in the original 2019 project definition. However, the robot's current state is not completely finished, and many features remain to be finalized. More details on Future Work are given in Section 5, "Remaining and Future Work".

The following sections detail the work that has been completed in the design and successful integration of multiple hardware and software subsystems.

2 Physical System Design

2.1 Chassis Design

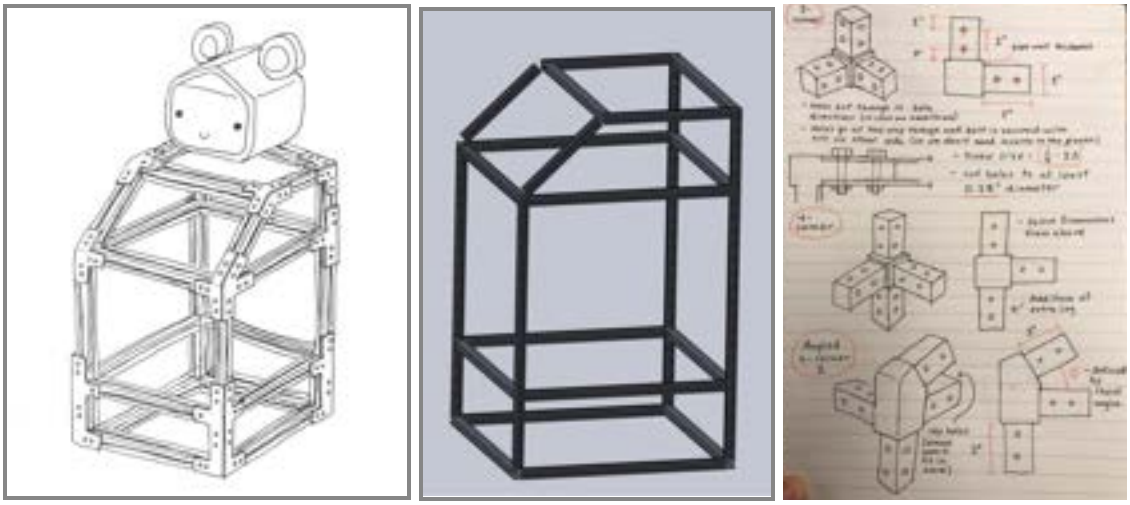


Figure 2.1.1: Initial concepts and sketches of Chassis Design

The chassis served as the main structural framework for the bot, supporting the weight of all elements as well as providing sturdy points of attachment for internal systems.
For this reason our main goals in the design were:

- Structural strength
- Manufacturability
- Ease of assembly and disassembly
- The ability for modular and changeable attachments

Since the subteams worked independently, we could not always know exactly where internal systems would need attachments. Thus we wanted to make the chassis bars out of a material that could be easily machined to add more holes, mounting plates, and screws. The original proposed chassis structure was to be built out of 80/20 profile aluminum beams, connected with commercially available connectors. Due to

concerns about the project budget and robot weight, this idea was scrapped in favor of using square PVC pipes and custom 3d printed connectors, a more cost-effective solution.

Once the main structural scheme was developed, the chassis subteam worked on the implementation of a number of physical and electrical components, which could be broken down into the following categories:

- Physical chassis:
    - Weight-bearing supports
    - Secure connections between supports
    - Hardware for assembly (bolts and nuts to secure bars to connectors)
    - Removable acrylic siding to contain internal components
    - Moveable head with two degrees of freedom
- Electrical elements:
    - Mounting and setup of interactive touchscreen panel
    - 2 independent motors to control head rotation
    - Implementation of LED panel for facial expressions
    - Mounting and setup of speaker, microphone, and camera

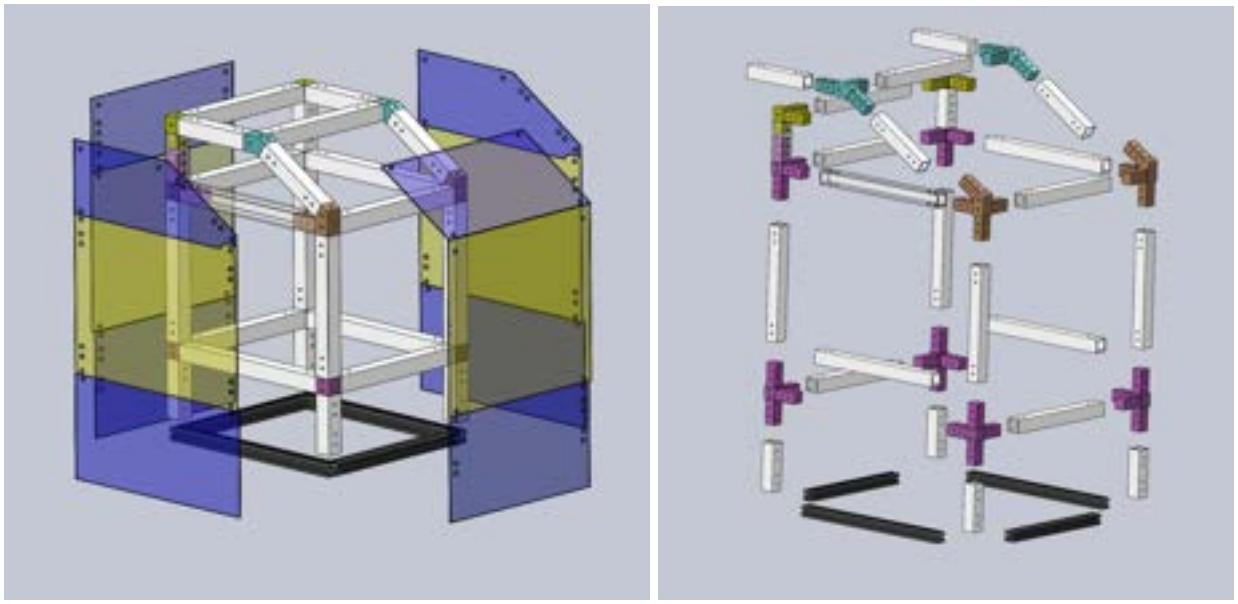


Figure 2.1.2: Base chassis design showing arrangement of acrylic panels and exploded view.

The majority of the chassis was made from 1.25” square-profile PVC pipes, while some minor parts were made from 80/20 aluminum bars. We 3D printed connectors to piece together the various geometries in the frame of our chassis and secured them using bolts. We also 3D printed the screen mount that would go on the front panel of our chassis. Finally, we added blue and yellow acrylic plates to the sides of the chassis in order to provide protection to the internal components, keeping the visual scheme consistent with the UCLA school colors.

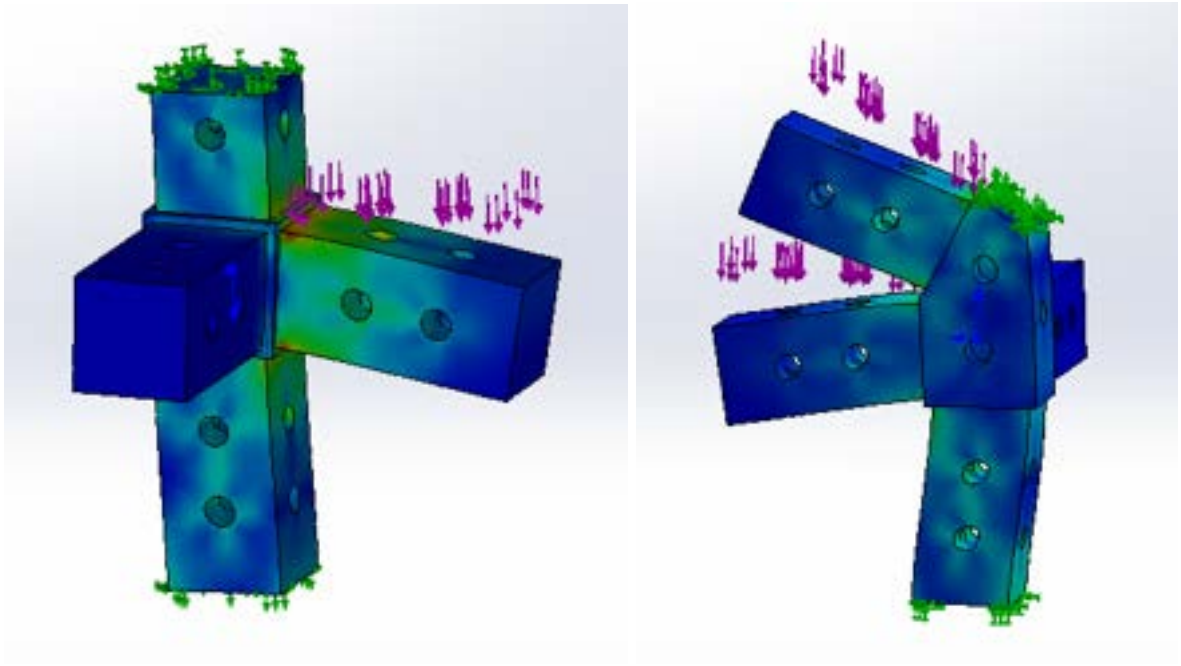


Figure 2.1.3: FEA analysis performed on two of the internal connectors, applying static downward loads.

The above images show FEA analysis of two different connectors when loads are applied. However, FEA is not guaranteed to be accurate because the parts are printed in stratified layers of plastic, so they are more susceptible to failure due to loads in certain directions. For this reason the quantitative values of the loads are not shown (as they cannot account for the material's variable strength) Still, the analysis is useful in showing where stress is concentrated when external loads are applied.

3D-printing the connectors allowed for customization to the exact geometry needed. This was especially important for the angled connectors, as we were able to design them so that the chest screen would have the desired angle for easy user interaction. Another advantage of 3D printing the connectors was it required no additional cost. The main disadvantage of 3D printing is that, as mentioned previously, the layers make the parts more susceptible to failure due to loads in certain directions. Specifically, the connectors are more prone to shear due to moment parallel to the printed layers and are weak in tension perpendicular to printed layers.

2.2 Head Design

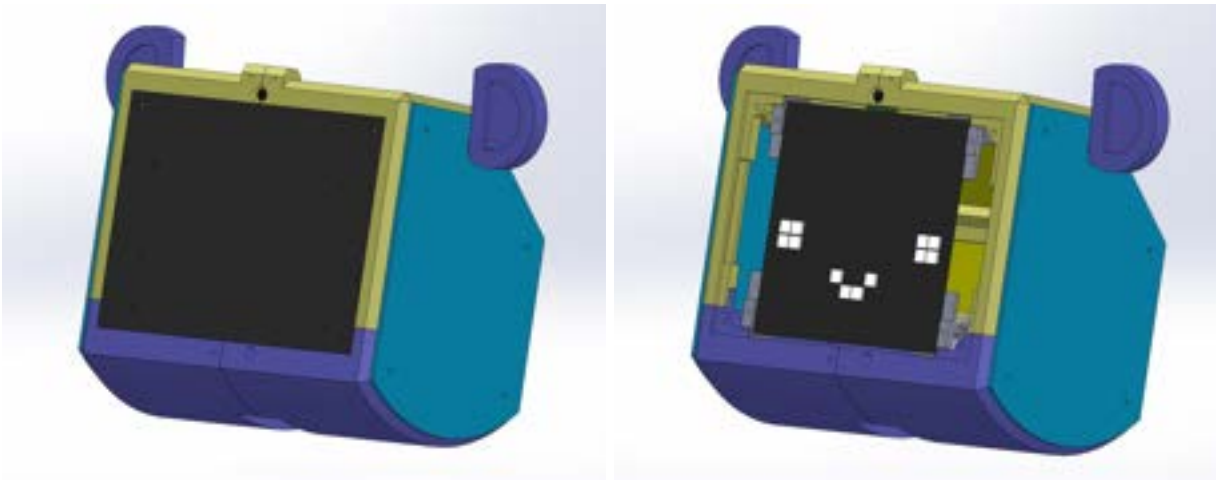


Figure 2.2.1: Exterior head design contains a pseudo-opaque "Black and LED Acrylic" panel (left), which appears opaque when unlit but allows light from a concealed LED panel (right) to shine through.

The design of the head and face were crucial to the overall goal of the BruinBot project, namely, to be a friendly, approachable entity that would provoke interactions from students. The aesthetic appeal of this part of the robot was therefore a driving factor in its design. A simple, blocky profile was chosen, (reminiscent of an old-school computer monitor) and the facial features are displayed on a low-resolution 16x16 LED screen. The LED screen is capable of displaying facial expressions using up to 256 pixels and different RGB colors. (More info on the facial expression control is written in section 3.5). The LED matrix is hidden behind a panel of specialized pseudo-opaque acrylic, known as "Black and LED Acrylic," which appears opaque when unlit but allows LED light and color to pass through. Additional standard acrylic plates were attached to the sides and back of the head to obscure the internal structure and create a smooth look.

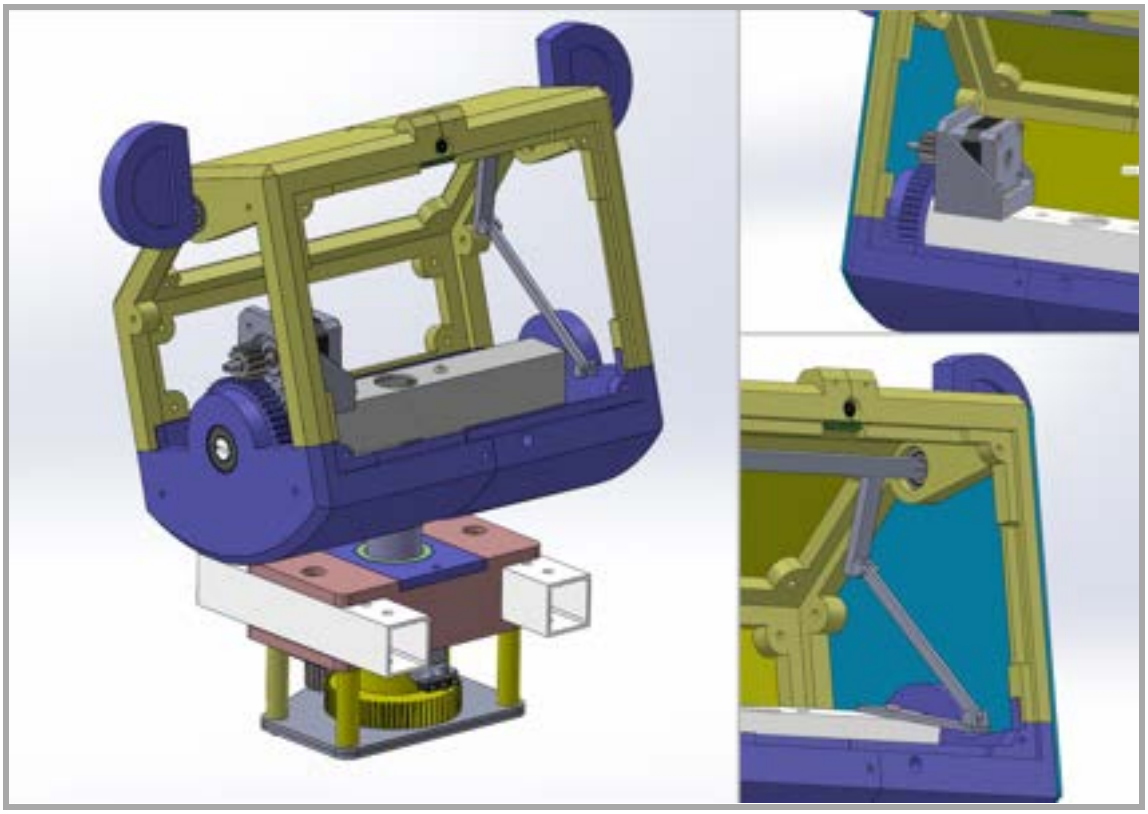


Figure 2.2.2: Structure of head and neck (left) with close-up views of head nod mechanism (top right) and four-bar linkage ear tilt mechanism (bottom right).

The head consists of a 3d-printed shell anchored to a support bar made of the same PVC pipe as the chassis, which is attached to the chassis via a rigid PVC neck. The head has two degrees of freedom in rotation (both side-to-side rotation and up-down rotation) driven by NEMA17 stepper motors. These motions, which enhance the lifelike nature of the robot's expression, also serve to give the robot's camera more freedom to move and focus in on areas of interest. Each motor is paired with a limit switch, mounted to set the motor's zero position upon startup. Moving ears were also added to provide an extra element of life and visual appeal. The ear's motion is linked to the nodding motion by using a variant of a four-bar linkage mechanism to provide a small rotation as the head moves up and down. Close-ups of the nodding motor and ear mechanism can be seen in Figure 2.2.2 above.

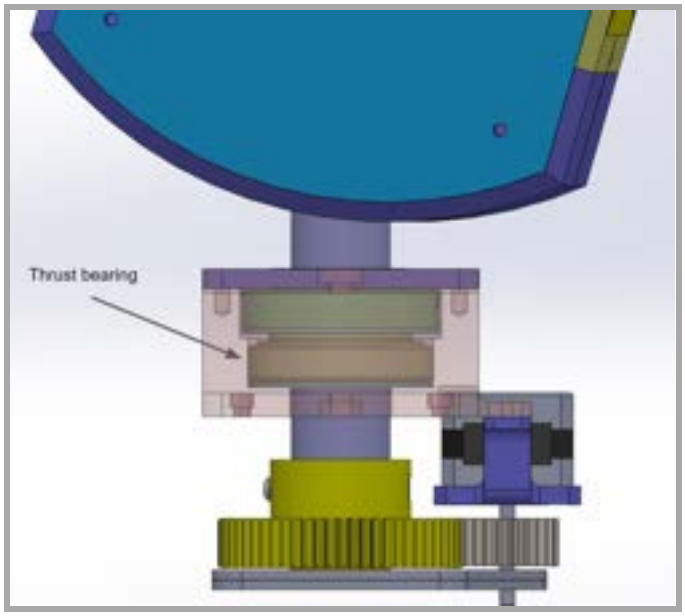


Figure 2.2.3: Head-turning mechanism with built-in thrust bearing and gear system

The head-turning mechanism is located down inside the body of the robot, with the powering stepper motor mounted to the stationary chassis. A series of 3D printed gears rotates the neck (a length of cylindrical PVC pipe) back and forth. Since the head system is significantly heavy, a large conical thrust bearing was incorporated to support the weight of the head and facilitate smooth rotation of the neck.

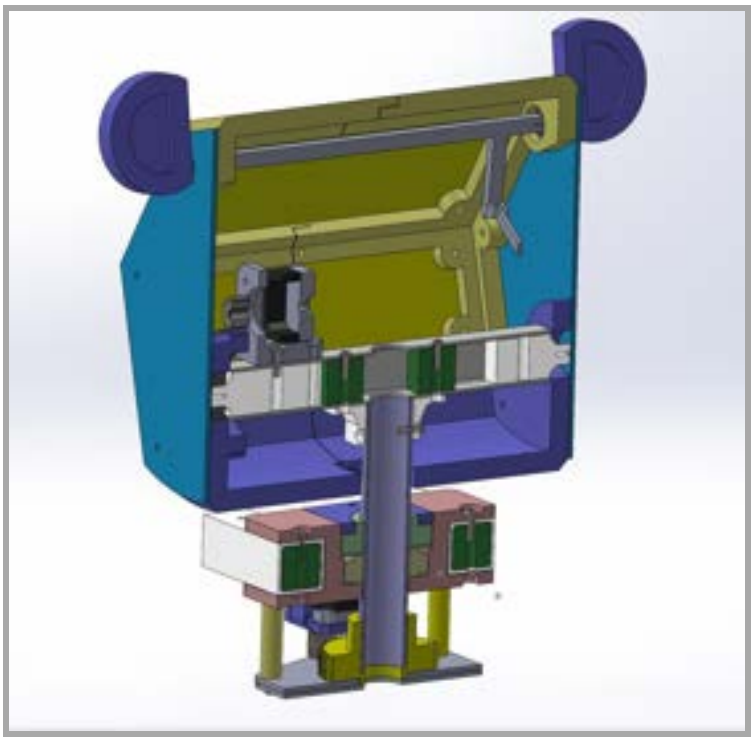


Figure 2.2.4: Cutaway view showing internal structure of head and neck including the neck's function as an electrical cable conduit.

Since the head houses one stepper motor, a limit switch, the microphone system, and the camera, it was important that the hollow inside of the PVC neck not be obstructed so it could be used as a conduit for cables connecting the head's electronics to the main controllers housed inside the body. This conduit can be seen in Figure 2.2.4 above, creating an unbroken connection between the head and body. It is important to note that since no slip rings are incorporated in the design, the neck should not be rotated continuously in one direction, as it will tangle and damage the internal cables.

2.3 Payload System Design

The payload system is intended to allow BruinBot to physically interact with users by dispensing various items, such as snacks and flyers, using a dispensing mechanism and robotic arm. The subsystem consists of an item storage compartment, dispensing mechanism, and single-joint mechanical arm. During the 2020-2021 school year, this subsystem underwent significant design revisions from the previous year. Due to lack of documentation, this previous design will be briefly touched on, with the newer revision described in greater detail.

Dispensing System

The previous design of the dispensing system was complex and involved many moving parts. It had a storage compartment consisting of four columns where the granola bars/snacks could be stacked. Behind this storage compartment was a sliding gantry that navigated to each compartment, pushing each snack out using a rack and pinion mechanism.

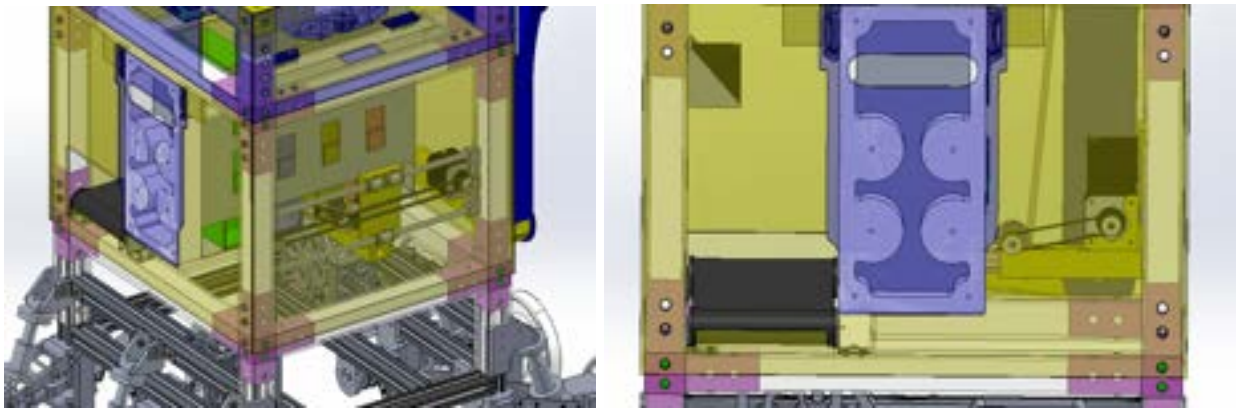


Figure 2.3.1: Dispensing system (old). This design utilized a 3D-printer-style sliding gantry and rack and pinion mechanism to dispense items.

Based on feedback received during the 2020-2021 school year, we decided to simplify the design to require fewer degrees of freedom and less failure points. To this end, we opted for a vending machine-style design, consisting of four rotating springs actuated by continuous servos. In this design, we reduce the degrees of freedom for dispensing from two to one, ideally reducing the number of failure modes.

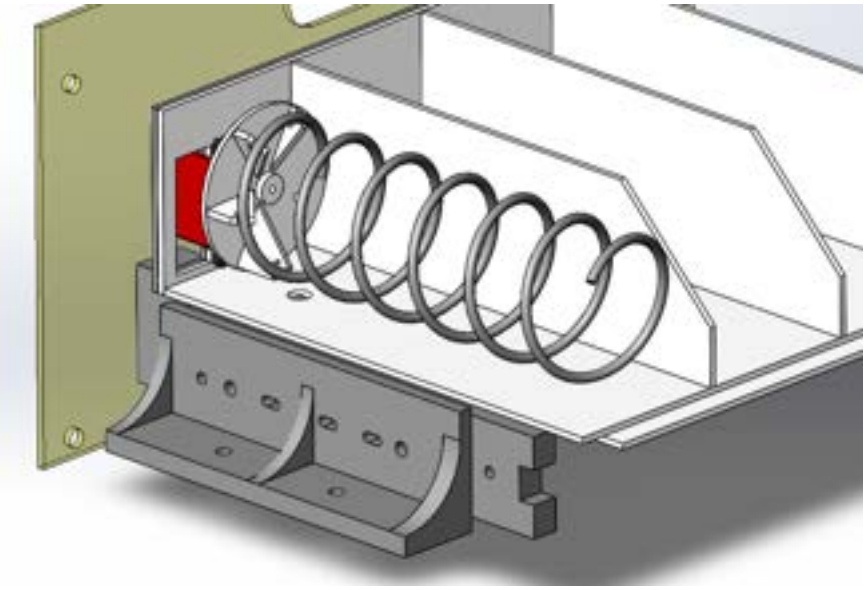


Figure 2.3.2: Dispensing system (new). This design utilizes a rotating spring to dispense items, similar to a vending machine. Note that only one spring assembly (out of four) is shown.

The dispensing carriage itself was designed to be manufactured using laser-cut acrylic held together using a “lock-and-key” design that constrains the assembly while also allowing for easy assembly and disassembly.

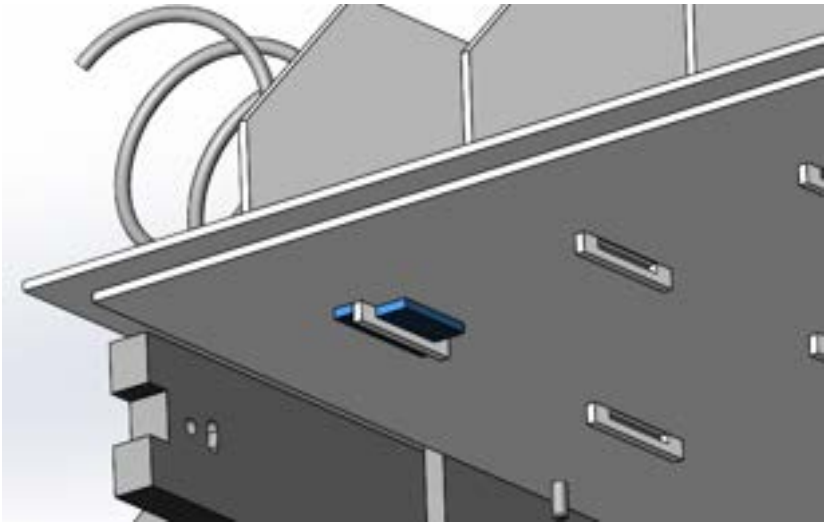


Figure 2.3.3: Dispensing carriage design using lock-and-key concept. Slots placed at varying points on the acrylic allow for insertion of “locks” that can be held together with small “key” pieces (shown in blue). This allows for easy disassembly without requiring the usage of adhesives or additional fasteners.

To attach the vending machine springs to the servo, a custom 3D-printed design was used that had various attachment points which matched the pitch of the spring, allowing for zip ties to hold the spring firmly in place.

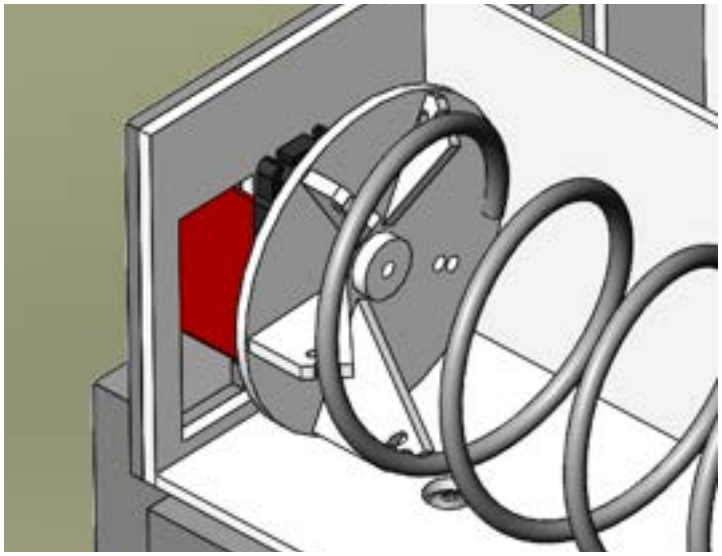


Figure 2.3.4: 3D printed adapter to attach servo horn to spring.

A final requirement for the dispensing system was that the entire assembly needed to be retractable to allow for convenient refilling of snacks. To achieve this, we used a pair of drawer slides that are commonly found in hardware stores. By fixing one side of the drawer slide to the chassis and the other side to the dispensing assembly, the entire assembly is easily slid in and out of the robot. To allow for ease of assembly and dimensional accuracy, the drawer slide brackets were designed such that two sides of the bracket could be set flush with the PVC chassis to align the mounting holes properly.

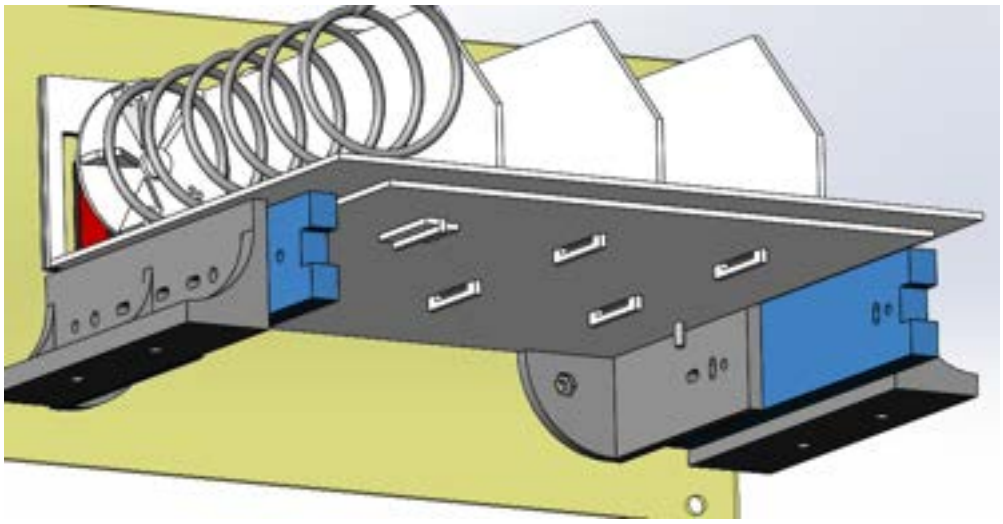


Figure 2.3.5: Drawer slides (shown in blue) to allow for convenient restocking of items. Note that the CAD has been simplified for dimensional purposes since this is an off-the-shelf part.

Conveyor System

The previous design of the conveyor system was driven by a stepper motor and belt-and-pulley system. To simplify this design, we opted to use a standard DC motor to directly drive the conveyor drum.

Because this system only requires continuous rotation without precise control, we believe a DC motor to be better suited for this application.

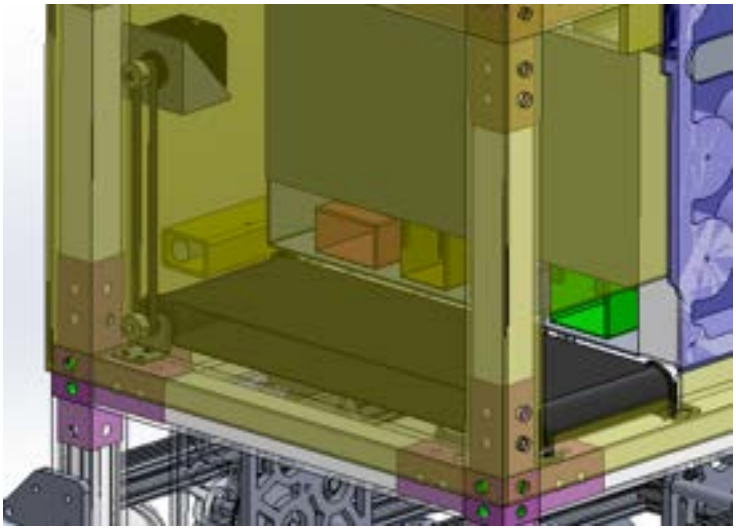


Figure 2.3.6: Conveyor system (old).

The new conveyor system uses a series of 3D printed drums. On the driven side, the 3D printed axle sits in a drilled hole in the PVC chassis. On the non-driven side, the drum spins between two 3D printed brackets.

We soon discovered that using a 3D printed axle for the drums introduced significant plastic-on-plastic frictional issues and also resulted in a very brittle design. To remedy this, the non-driven barrel was redesigned to seat two ball bearings held in place by two bolts which also act as axles. This significantly improved frictional issues and also removed concerns regarding the axles breaking off.

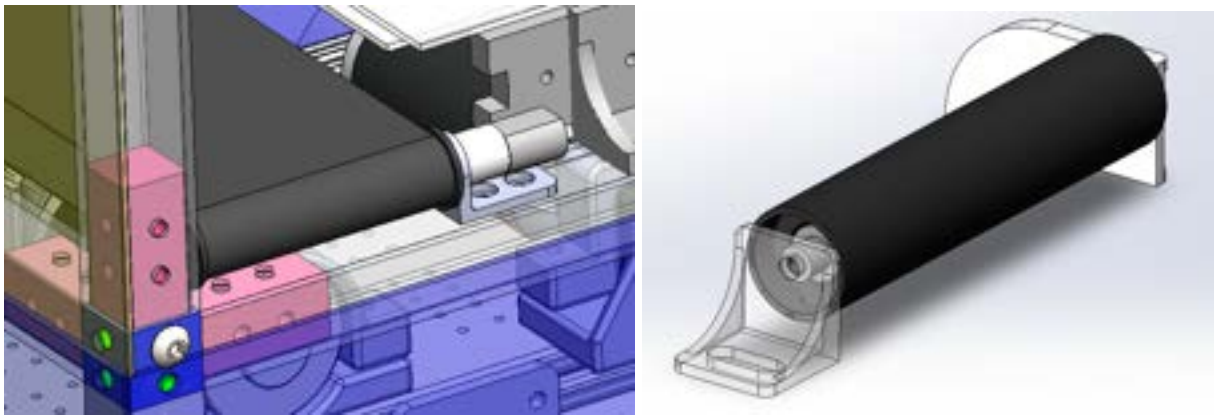


Figure 2.3.7: Conveyor system (new). Driven side (left) and non-driven side (right). The non-driven side utilizes a set of ball bearings to reduce friction and improve alignment.

The material of the conveyor belt itself was chosen through some experimentation (due to budget concerns, actual conveyor belt material was largely unattainable), and the results of this are documented

in more detail in the results section. The final material chosen was a leather fabric obtained from Hobby Lobby that is commonly used for furniture upholstery.

Left Arm Assembly (Functional)
The left arm is intended as the delivery method between the conveyor system and the user. It is intended to raise the dispensed item to a more accessible level for the user to receive.

The previous design actuated the arm using a 775pro DC motor, Versa Planetary gear kit, and an external optical shaft encoder.

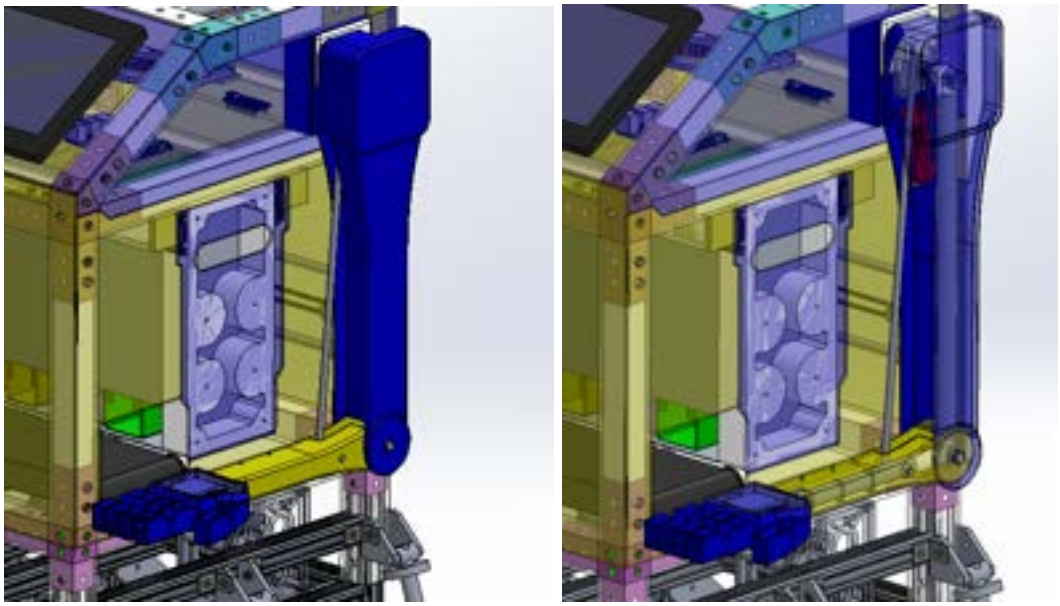


Figure 2.3.8: Arm assembly (old).

Based on feedback received during the 2020-2021 school year, we decided to replace these components with a single Pololu metal gearmotor with built-in gearbox and encoder. This decision was made in an effort to simplify mechanical design and assembly, as well as simplify future controls. As development continues, we are planning on utilizing PD control to actuate the arm in a smooth and friendly manner. Additionally, a limit switch was incorporated into the shoulder assembly to “home” the position of the motor on startup, since our encoder does not utilize an indexing position.

Note that although many aspects of the design involved this DC motor and limit switch design, through experimentation we determined that the DC motor was unable to provide sufficient torque to lift the arm. Thus, the motor was replaced with a high-torque servo. Further details of this are described in the results section.

The functional requirements of this arm are to raise the dispensed item from the conveyor system to a more accessible level for the user. To achieve this, we utilize a parallel linkage mechanism to maintain BruinBot’s paw parallelity with the ground. The structural components of the arm consist of two aluminum rods with several 3D printed attachment components.

In terms of aesthetic improvements, we aimed to achieve a “friendlier” and more cohesive look compared to the old design. To do this, we incorporated elements from the head design, including rounder and convex shapes, and made efforts to conceal functional and structural components that may convey an “unfriendly” appearance. A circular shoulder piece was designed to echo the circular ear pieces.

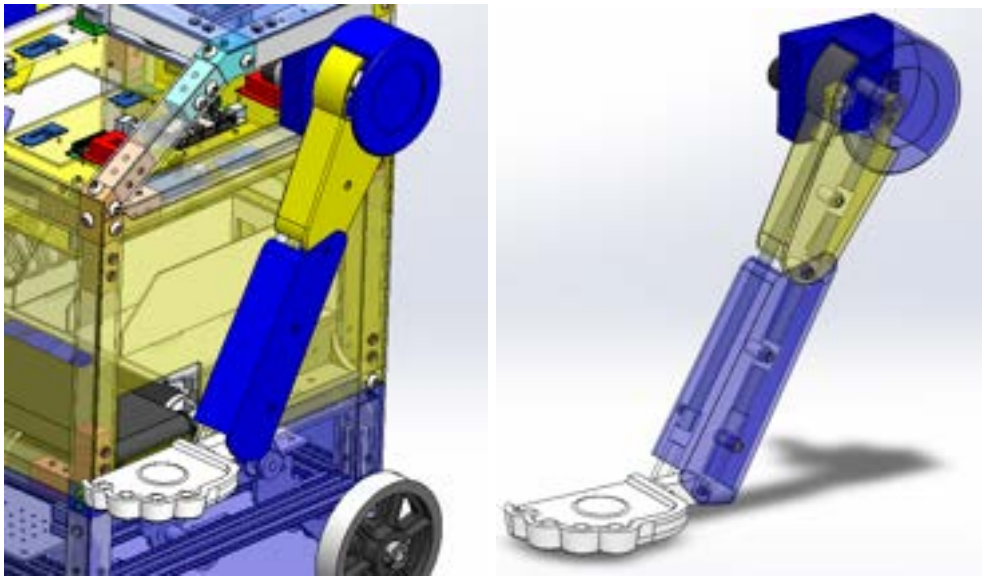


Figure 2.3.9: Arm assembly (new). The parallel linkage consisting of two aluminum bars can be seen on the right. The external sleeve attaches to this structure at various points.

For the moving arm, three connectors were designed and 3D printed to mount the outer sleeves to one of the metal shafts. This was done to increase the number of mounting points, improving stability of the sleeves. The additional parts were printed to avoid the need to drill through the metal shaft, and to allow the arm to be easily assembled and disassembled.

Right Arm Assembly (Non-Functional)

The right arm is intended to provide symmetry and a cohesive appearance for BruinBot while not serving a strict functional purpose. To achieve this, the left arm external sleeves and paw were mirrored for the right arm, while striping the internal components The aluminum rods were replaced with long mounting blocks that secured the arm sleeves together.

The paw has been redesigned at the wrist to be directly attached to the bottom sleeve at a specified position. For the shoulder joint, the two shafts were replaced with a block spacer that would attach the top arm sleeve to the mounting block, essentially attaching the right arm to the body.

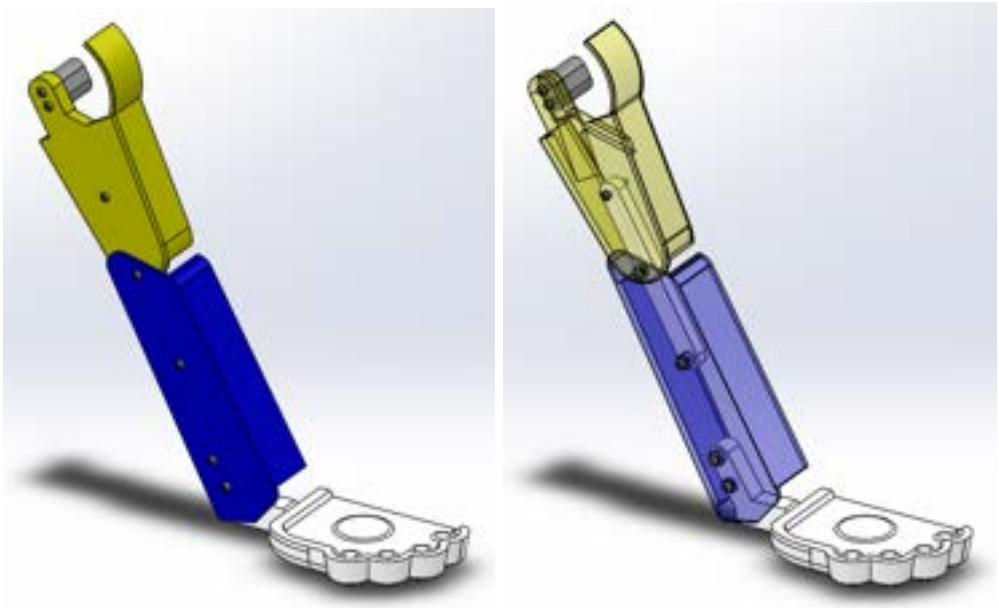


Figure 2.3.10: Right Arm Assembly. The external sleeves and paw are mirrored from the left arm. These are attached together and to the body using long blocks.

2.4 Drivetrain System Design

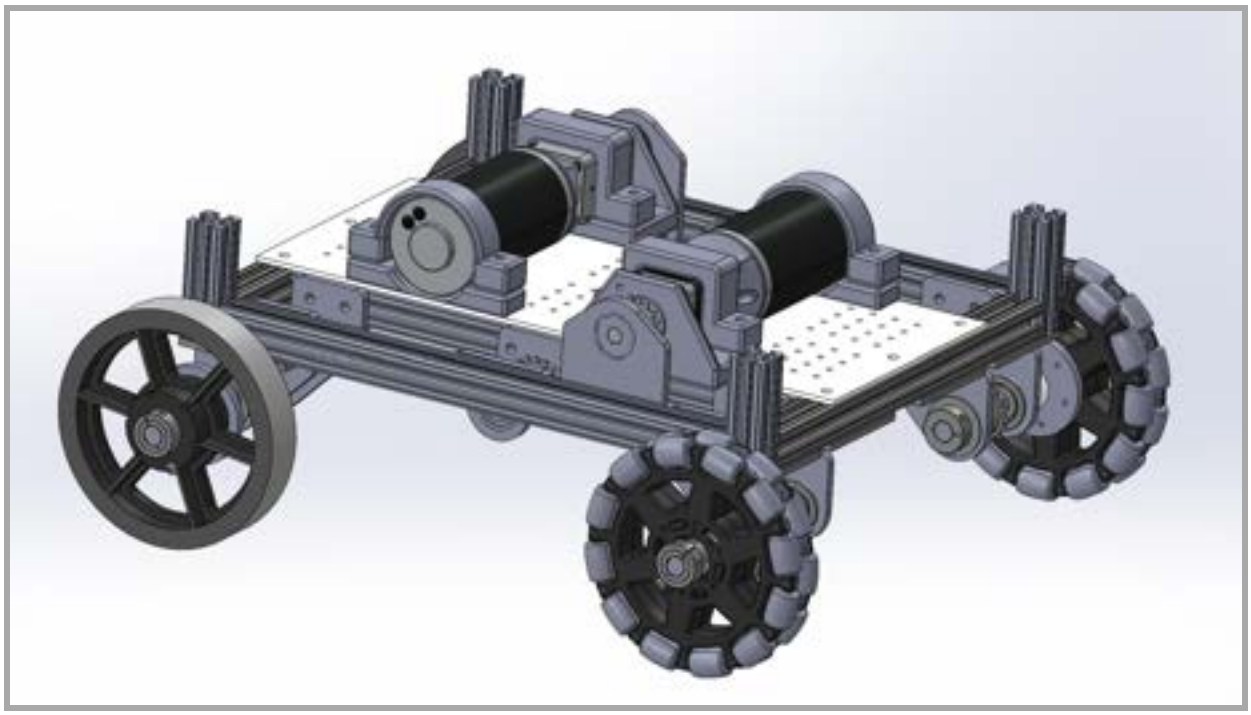


Figure 2.4.1: Final Drivetrain System 2021-2022

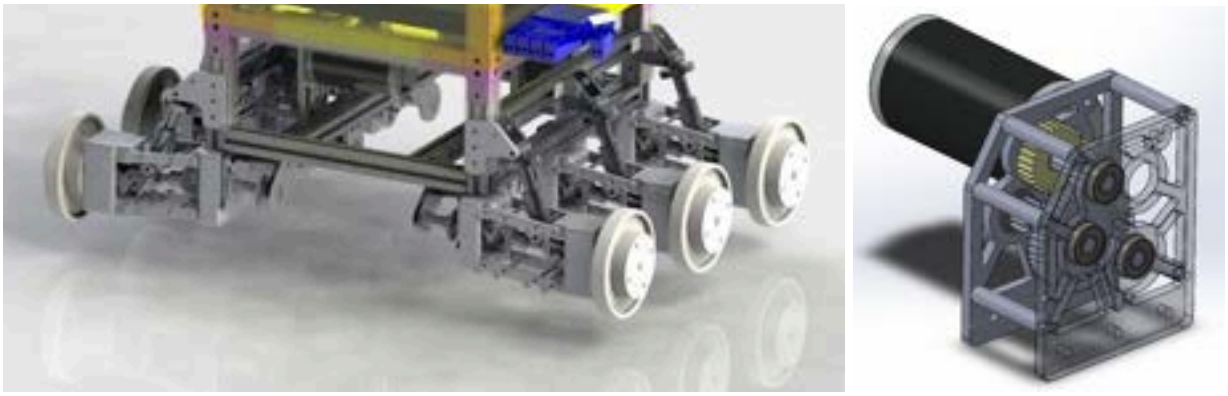


Figure 2.4.2: Original 2019-2020 Proposed Drivetrain System (left) and custom gearbox (right)

The drivetrain was heavily redesigned in 2021-2022 to simplify and strengthen the design. Previously, a six-wheeled West Coast Drive was used incorporating a suspension system, U-jointed flexible axles, and a custom gearbox. Due to budget constraints, many of these complex components were going to be 3D printed to save on the cost of purchasing metal components. This resulted in an overly complicated and unstable overall design that would have proposed a significant challenge in manufacturing, assembly, and stability.

In 2021, the drivetrain was completely redesigned to create a robust system that would be more reliable and simpler to manufacture and assemble. This was possible due to modifications to the tight budget that was enforced previously, as well as workarounds like using existing found parts (incurring no extra cost). The drivetrain system was changed to a four-wheel drive, the suspension and U-joints were removed, and the custom gearbox was replaced with a purchased VEX Robotics 30:1 gearbox capable of handling the high-torque application. The wheel axles are now attached to the center chassis via waterjet aluminum plates with inset bearings. Each wheel axle acts as a live shaft with a rigid connection to both the wheel and the driving sprocket. The wheels are driven via a chain drive, where a single loop of chain is connected to both wheels, an idler, and a driving motor sprocket. This system is duplicated on the left and right sides of the robot so they can be controlled independently producing forward, backward, and turning motions at variable radii.

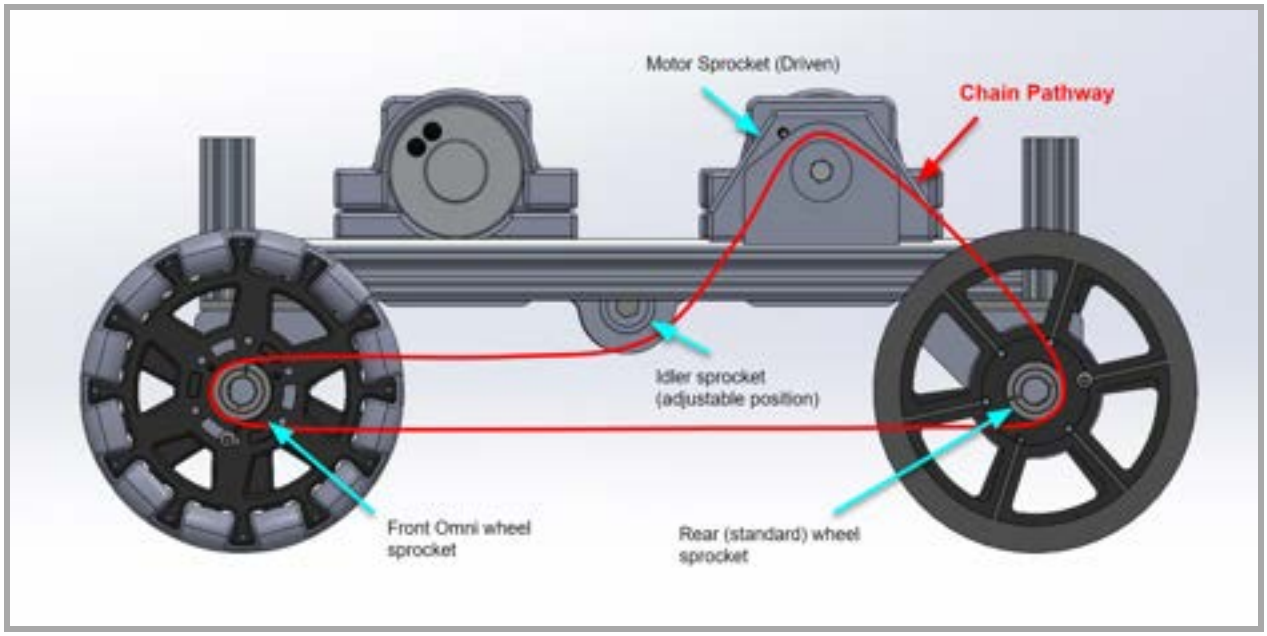


Figure 2.4.3: Pathway of chain over the 4 sprockets on each side of the robot

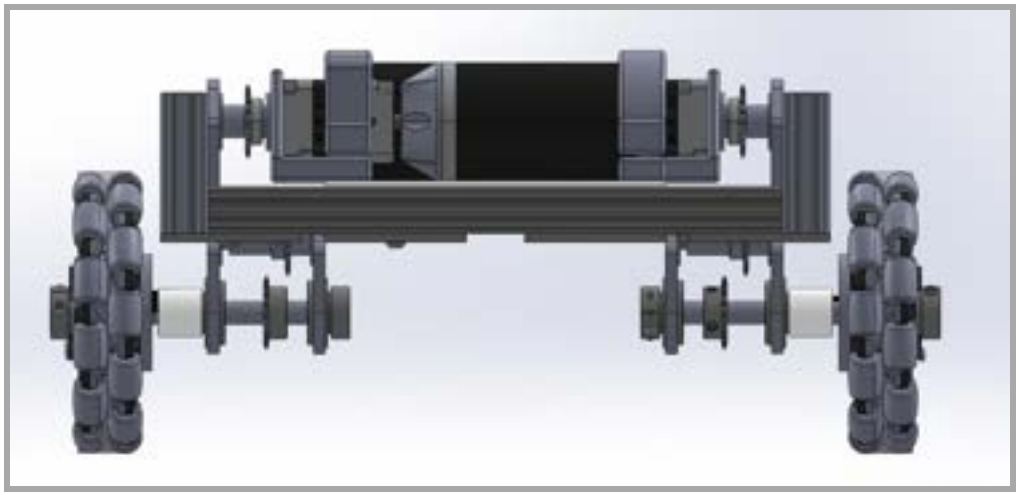


Figure 2.4.4: Front view of robot showing Omni wheels and attachment via aluminum plates

Another change made was to change the front wheels of the robot from static plastic and rubber wheels to 6-in Omni wheels from VEX Robotics. This was done to allow the robot to turn more easily while moving. The original design, with four standard rubber wheels, produced a huge amount of friction while turning as all four wheels were scraping against the ground. The rubber-wheel arrangement forced the center of rotation to be the center of the robot's footprint, but all four wheels were forced to travel along paths that were not parallel to the wheel's plane, resulting in scraping. This is illustrated in Figure 2.4.5 below.

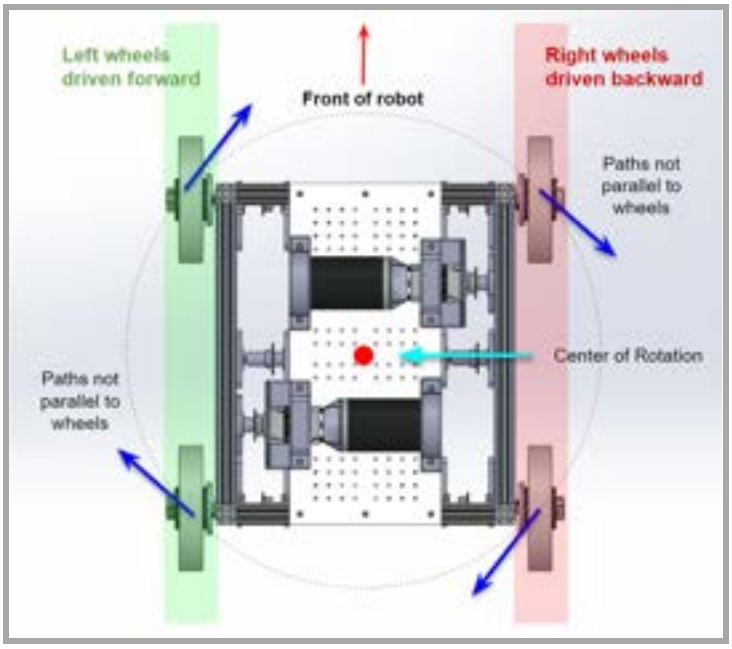


Figure 2.4.5: Original problematic arrangement with all wheels scraping during a turn

The addition of Omni wheels corrected this behavior as the structure of Omni wheels allows smooth non-parallel movement. The lower resistance of the Omni wheels shifted the center of rotation to the back of the robot, allowing both rubber wheels to turn around a center point directly between them. This allows both rubber wheels to move only in their own planes with very little friction, and the Omni wheels move in an arc at the front of the robot.

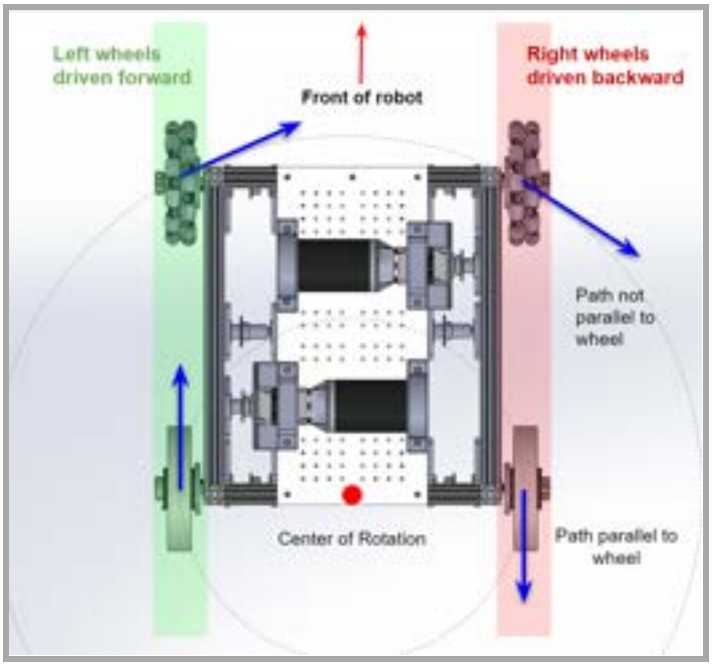


Figure 2.4.6: Adjusted turning pattern with addition of Omni wheels

As the robot no longer has active suspension, it is not able to navigate rough terrain and is assumed to only operate on a relatively flat concrete surface. This is acceptable as the robot is not expected to navigate terrain other than the paved courtyards and paths of UCLA campus or building interiors. Thus, the main goal of the powertrain system is to be powerful enough to move the robot at a reasonable speed across campus, while deftly controlling the motion at low speeds so it is not alarming to students. Two high-power 12V CIM Motors are used to drive the right and left sides of the robot independently. The motors are geared down at a 30:1 ratio (from top speed 5000 RPM to 166.67 RPM) and power is transmitted via chain and sprockets to both wheels on each side.

Incorporating the weight of the robot and the size of the wheels (6 in. diameter) the robot has a top speed of 4.6 ft/s (Drivetrain Calculation Spreadsheet). The motor can handle a maximum of 60A current continuously, so the motor controller used (Talon SR DC Motor Controller) is also able to handle 60A with spikes of up to 100A. The entire drivetrain system is rated for 60A, including the Talons, motors, battery wires, and power switch. During normal use, the motor would be driven at far below these top speeds and would draw far less than max current.

2.5 Electrical System Design

Bruinbot uses a system of 4 Arduinos and 2 Raspberry Pi's. It contains a 25.9V battery for the drivetrain, and two 11.1V batteries for the rest of the motors, microcontrollers, and other electronics. It also contains 4 buck converters which converts the 12V from the batteries to 5V to make it more manageable for the Arduinos and other components that should be run off of 5V power. The entire wiring diagram can be found in the GrabCAD and if viewed in Adobe Illustrator, various layers of the drawing can be toggled on/off making it clearer. The diagram is shown in the photo below:

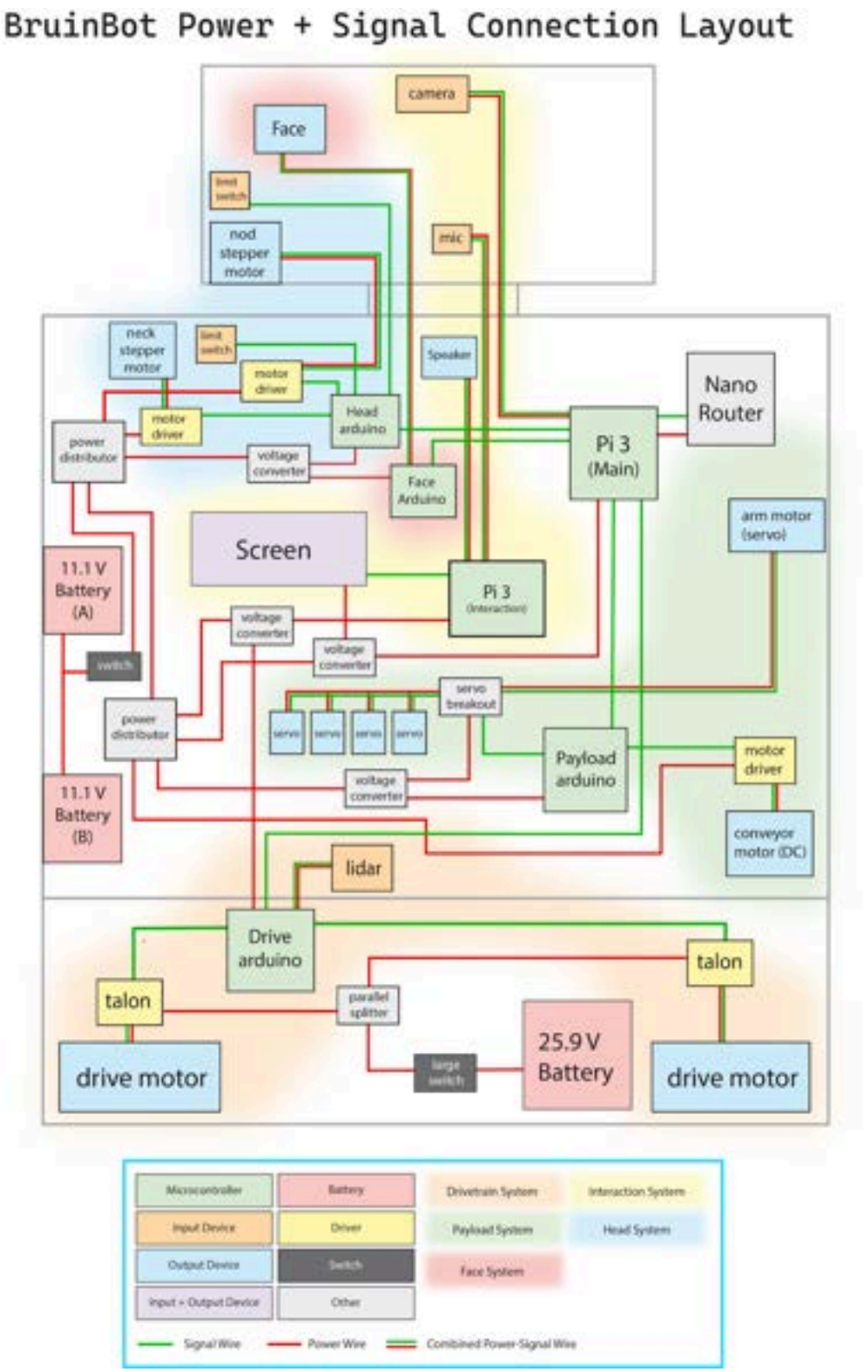


Figure 2.5.1: A simplified view of the connections between all subsystems within BruinBot, including both power and signal distribution.

As shown in the diagram, the red wires represent power wires, and the green wires represent signal wires. The drive motors are connected to the Talon SRX motor driver, which are connected to the battery and arduino as shown in the diagram below.

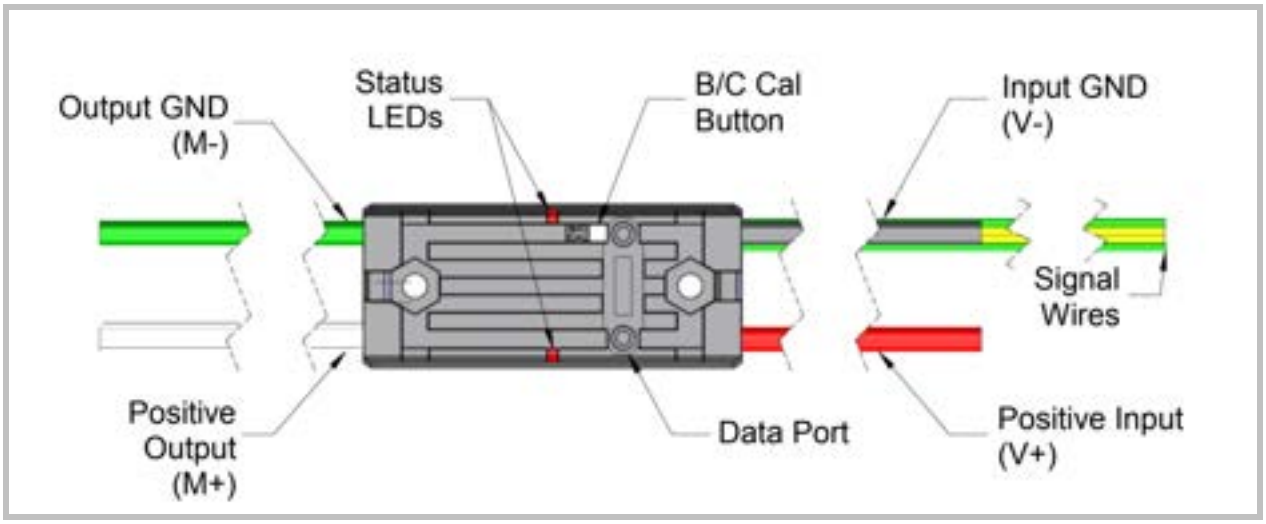


Figure 2.5.2: Talon Motor Controller Wiring

A splitter runs from the 25.9V battery to power both of the drive motors, as well as the conveyor motor (which is DC motor). The conveyor motor is controlled by a L298N Dual H-Bridge Motor Controller, and is wired as shown in the following figure (arduino digital pins are not correct):

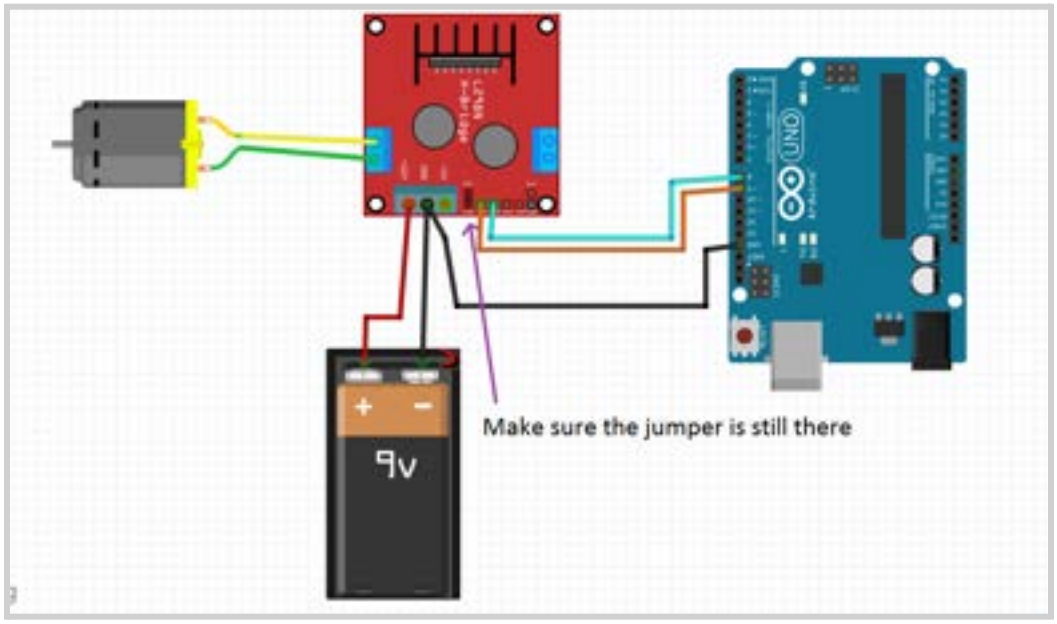
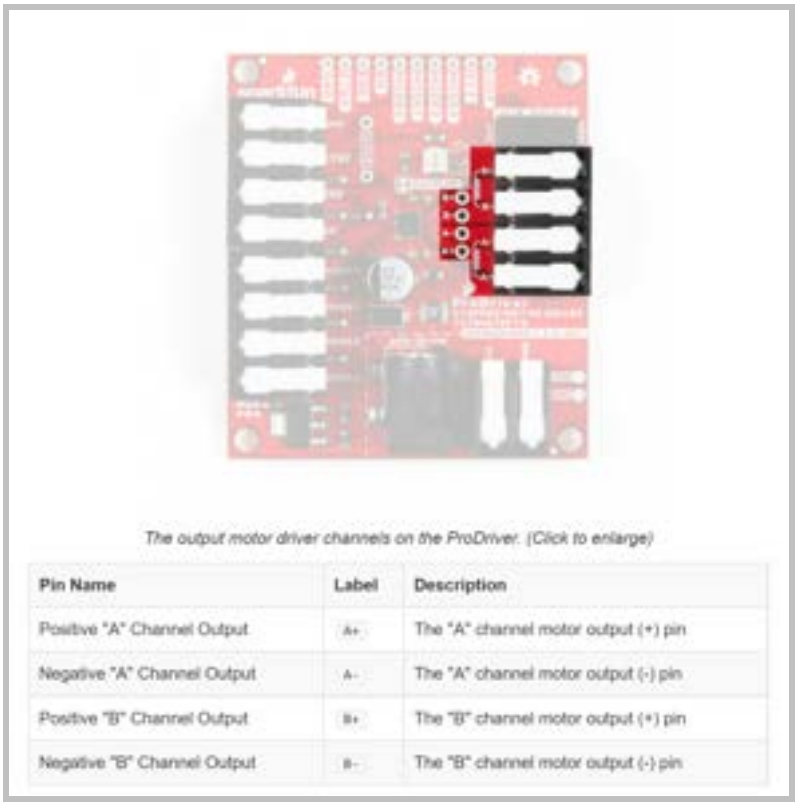


Figure 2.5.3: L298N Wiring Diagram (taken from this website)

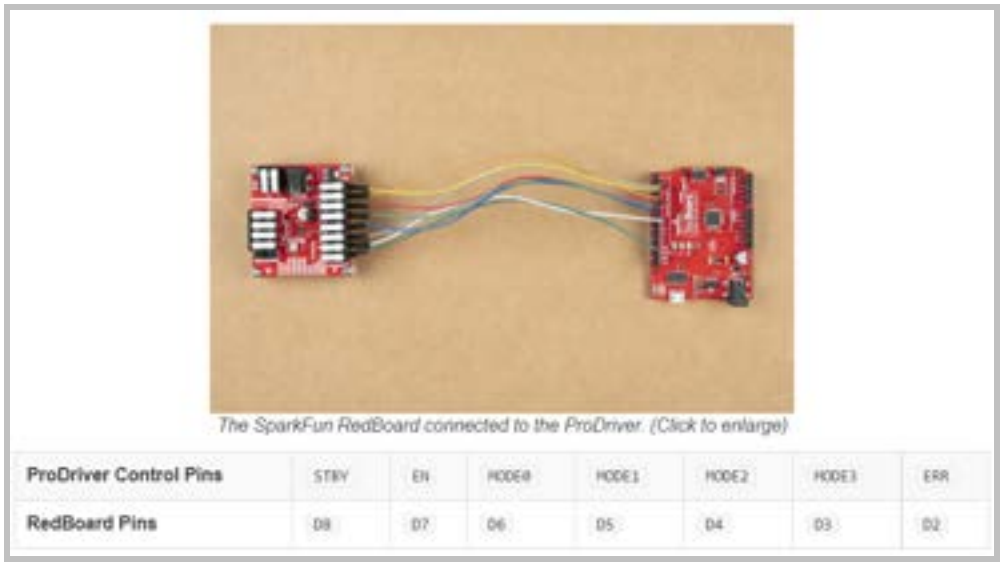
The last set of motor drivers are the TC78H670FTG Stepper Motor Drivers. They are connected to both the neck stepper motor and the nod stepper motor. It is connected as shown in the figures below:



The output motor driver channels on the ProDriver. (Click to enlarge)

Pin Name	Label	Description
Positive "A" Channel Output	A+	The "A" channel motor output (+) pin
Negative "A" Channel Output	A-	The "A" channel motor output (-) pin
Positive "B" Channel Output	B+	The "B" channel motor output (+) pin
Negative "B" Channel Output	B-	The "B" channel motor output (-) pin

Figure 2.5.4: TC78H670FTG to Stepper Motor Connection (source)



The SparkFun RedBoard connected to the ProDriver. (Click to enlarge)

ProDriver Control Pins	STBY	EN	MODE0	MODE1	MODE2	MODE3	ERR
RedBoard Pins	D8	D7	D6	D5	D4	D3	D2

Figure 2.5.5: TC78H670FTG to Arduino (source)

For more in depth information about the wiring from the arduino to the motor driver look at the table with descriptions at this link,

The electronics are physically placed on 2 trays attached with standoffs to the top of the frame as shown in the photo below:

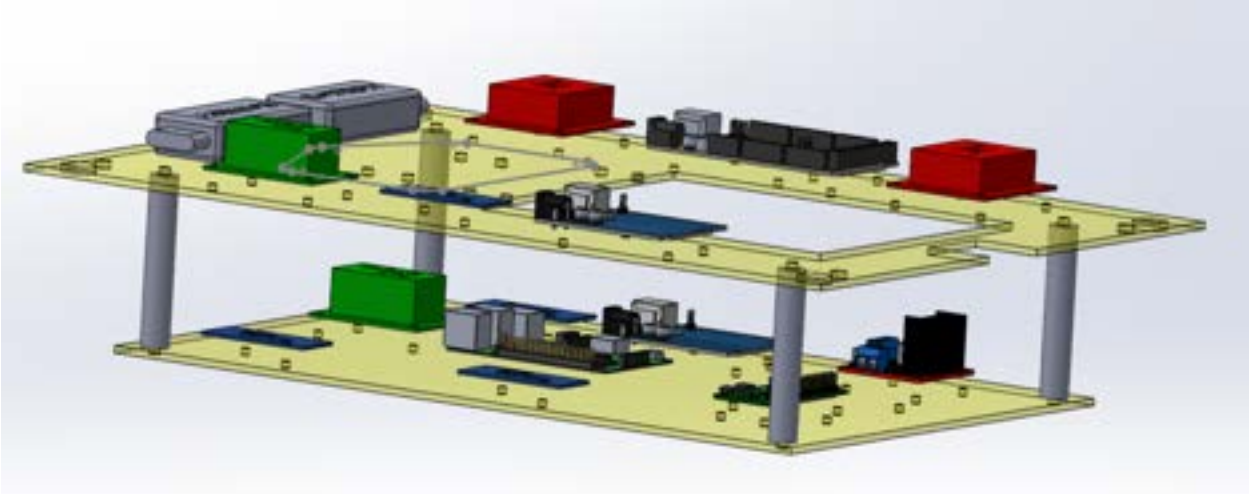


Figure 2.5.6: CAD of electronics tray, including both levels separated by standoffs.

3 Software Architecture

3.1 Distributed Controls

The code for the entire project is visible in the following Github **repository**.

BruinBot uses a distributed controls paradigm. This means the sensor array and control loops for different subsystems are running on different microcontrollers. A Raspberry Pi 3 is used as the central control unit that facilitates the information exchange between different subsystems. Subsystems that require real-time feedback control will use an Arduino Nano as the microcontroller since the real-time OS is much more suitable for encoders running at high sampling rates. The general sensor array also runs on an Arduino Nano because of I2C clock issues on the Raspberry Pi. While this system architecture adds more hardware to manage, the flexibility will make coordination and integration easier. For example, if we want to change the control law drivetrain motors, then only the code on the drivetrain microcontroller needs to change; if we want to change the encoder sampling rate for the arm motors, this will not affect the timing on the central processor that has to coordinate all other subsystems.

The software architecture coordinates communication between several subsystems which serve as inputs and outputs for the robot: the Main Microprocessor, Interaction System, Head System, Face System, Payload System, and Drivetrain System. Each of these systems is controlled by one microcontroller, either a Raspberry Pi 3 or an Arduino Uno or Mega. The most complicated systems are the Interaction System, which simultaneously coordinates the voice and touchscreen behavior, and the Main Microprocessor, which handles the vision/camera behavior as well as coordinating communication between all other devices. The Web Dashboard system allows users of the robot to directly control the drivetrain motors for movement, head motors to turn the robot's head, payload motor to dispense snacks, and the LED face to display emotions all with the click of a button. Together all of these systems help bring Bruinbot to life and allow users to interact seamlessly with the robot in an intuitive manner.

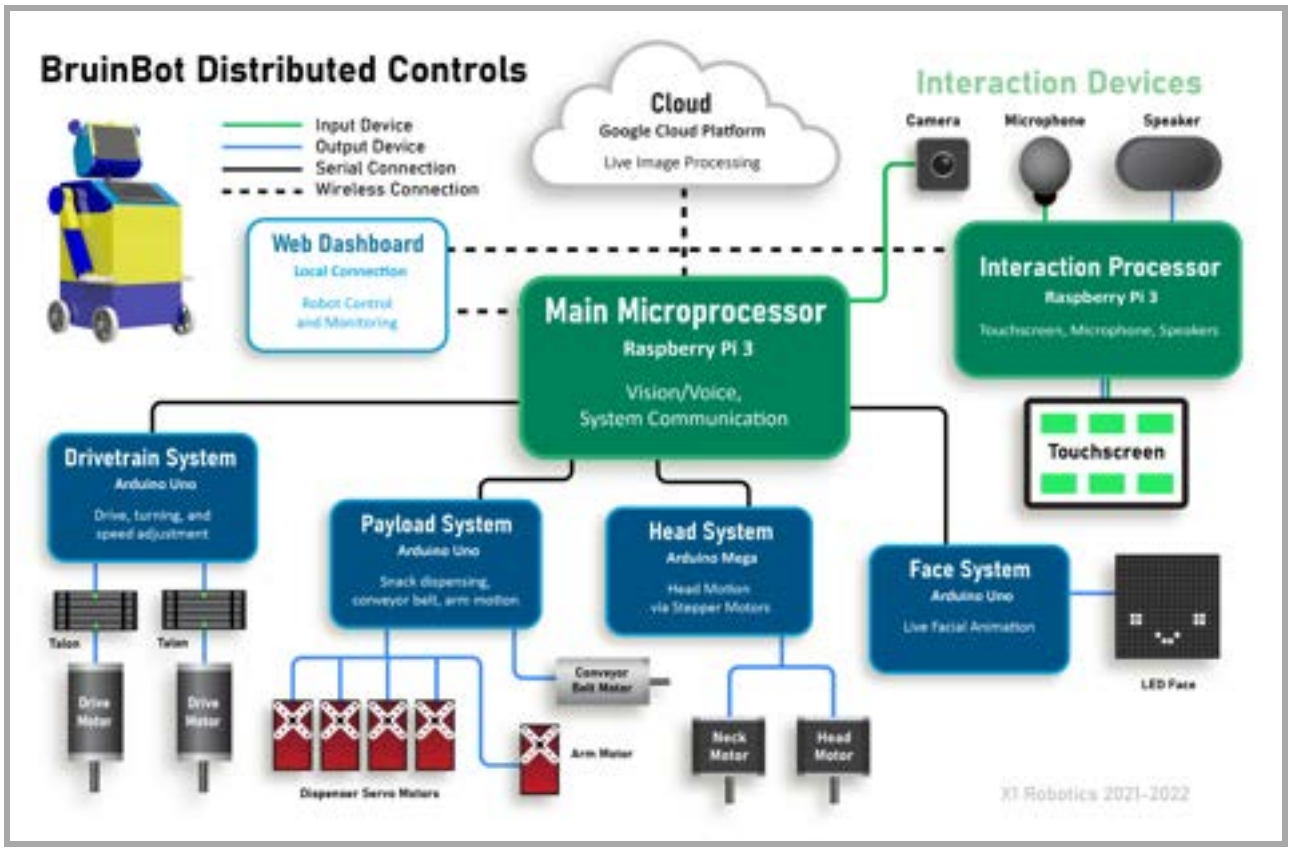


Figure 3.1.1: System architecture

With this distributed system architecture, information exchange between different components is key. Our design utilizes USB serial communication between subsystems. There are several reasons for this design choice. First, since all subsystems will be sitting in the same chassis, the physical link will be over a short distance, which means wireless channels are not required. Second, serial communication is very simple in software, and implementations in Python for the Raspberry Pi and in C++ for the Arduino are readily available. Finally, instead of using the two-wire serial connection, which can be hard to maintain, standard USB ports are much easier to use physically.

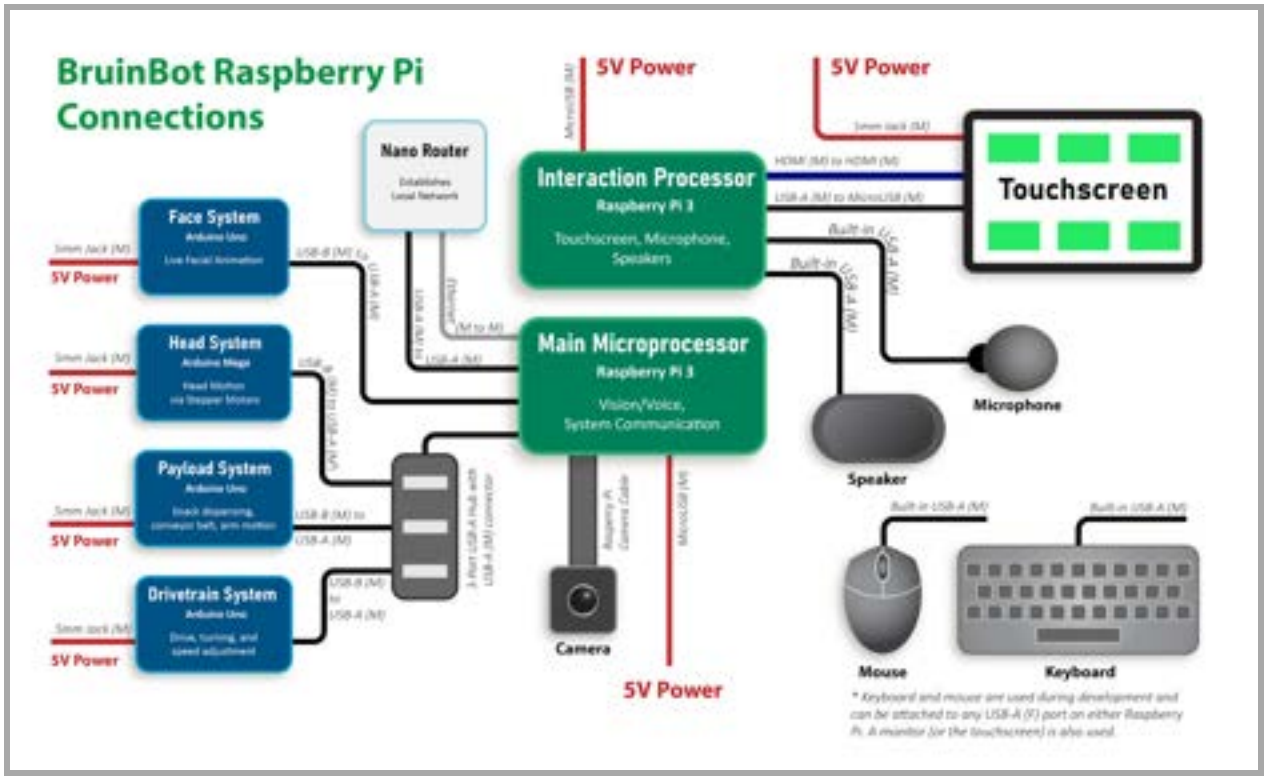


Figure 3.1.2: Detailed diagram of connections for each Raspberry Pi System.
See Also: Distributed Power and Controls Diagram.

Subsystem Communication

When the main script starts on the Pi side, a handshake sequence is initiated between the Pi and all the Arduinos. In this sequence, the Pi will send a message to each Arduino. Each Arduino will respond with its identity, i.e. which subsystem it is controlling. After this sequence, we can confirm that all the communication links are properly set up before continuing operation. We also ensure that the Pi can send the commands through the right serial port.

During regular operation of the robot, many actions can trigger serial commands being sent to the Arduino boards. The Main Microprocessor system is responsible for handling and sending commands to ensure they are formatted properly. The Main Microprocessor, however, can receive instructions to send a command from multiple sources. It can trigger these actions as part of the execution of a python script, or receive commands at any point from the Web Dashboard. Alternately, during development, the Serial Monitor on the Arduino boards can be accessed directly by plugging in a laptop computer and typing in commands by hand.

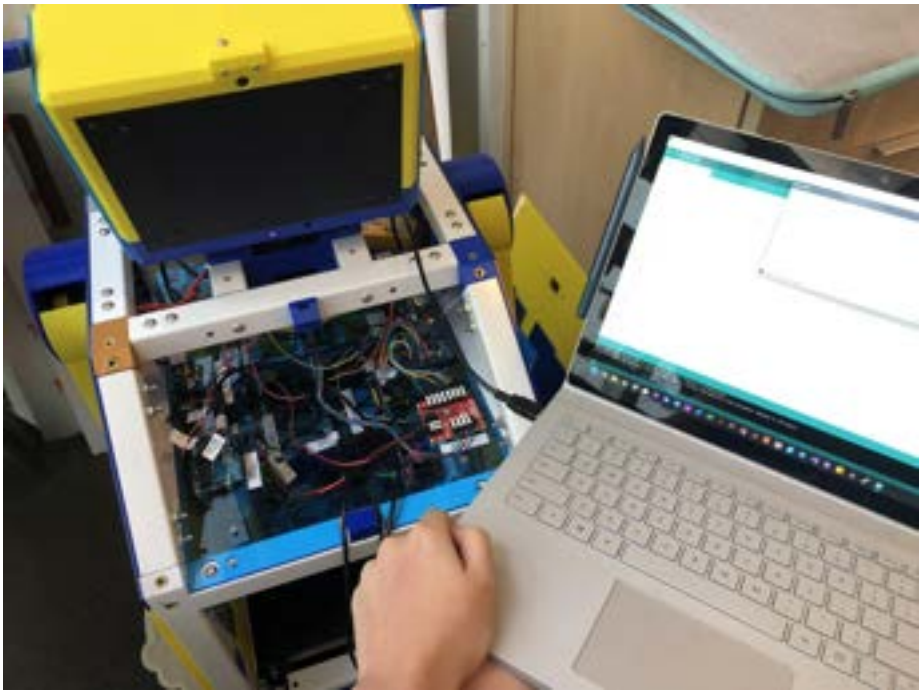


Figure 3.1.3: Overriding the Pi-Arduino connection by plugging the arduino's signal cable into a laptop for manual input of serial commands.

3.2 Robot Behavior Algorithm

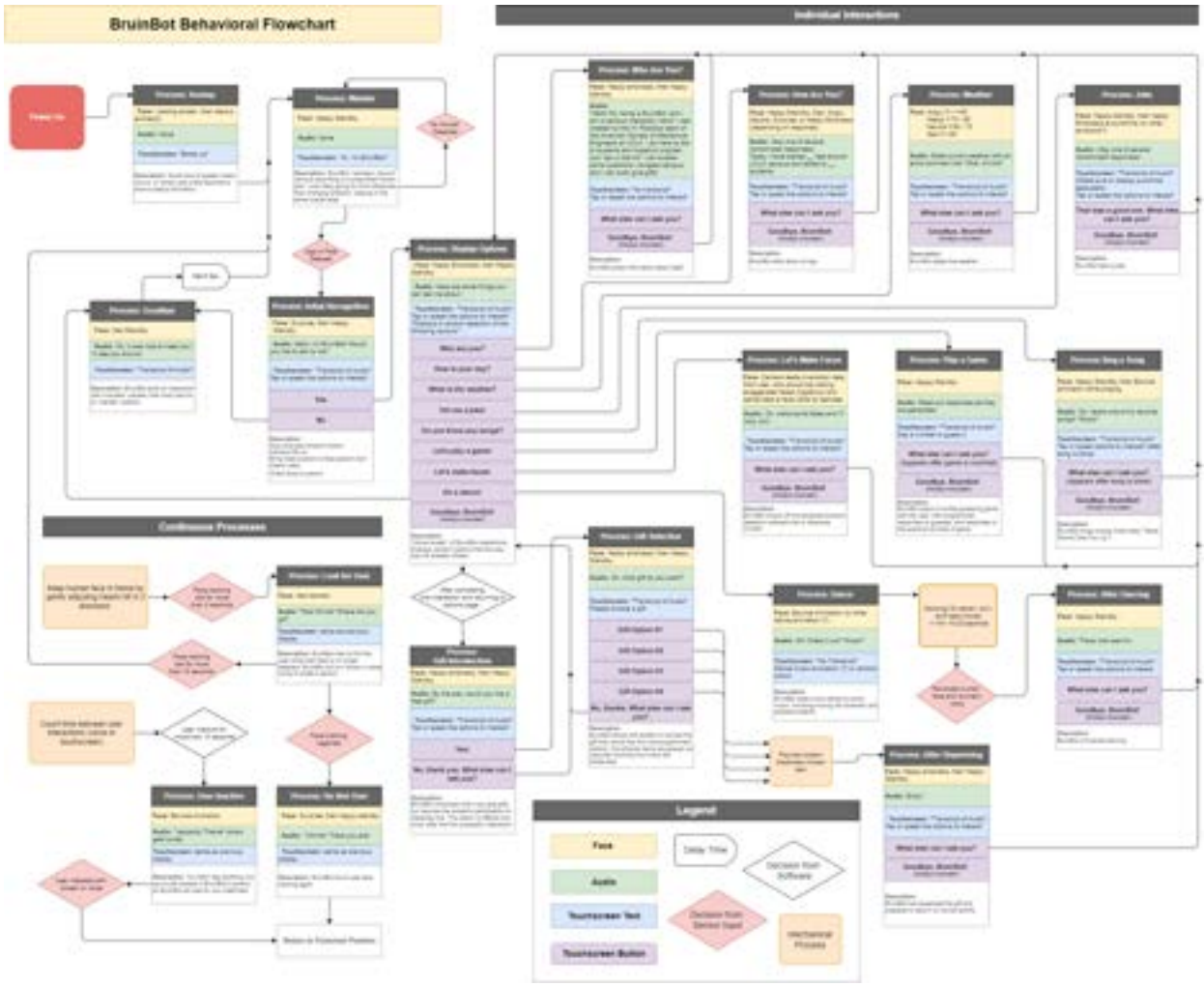


Figure 3.2.1: Complete BruinBot Behavioral Flowchart

The behavioral flowchart (Full version here) outlines the steps the robot takes from the power on state to the continuous interaction state where it is constantly interacting with the user. The primary step the robot takes is first listening to the user's input via the microphone. The voice program then translates the input into sentences that the voice code analyzes. Using a keyword system, the program finds keywords in the sentence that are mapped to different prompts. For example, if the user asks a question with the word “joke”, the program will recognize that the question most likely is related to the prompt “Tell me a joke”. After the program has determined which prompt is asked, it will give the appropriate response using our Text-to-Speech engine and any API’s associated with it. The program then will send the corresponding emotion to the face emotion code and change the LED expression of the robot based on the question asked. This then interacts with the touchscreen display to have a transcript of what the robot outputs for better accessibility. The exact flow of each subprocess action is outlined in the flowchart.

3.3 Vision and Voice Systems

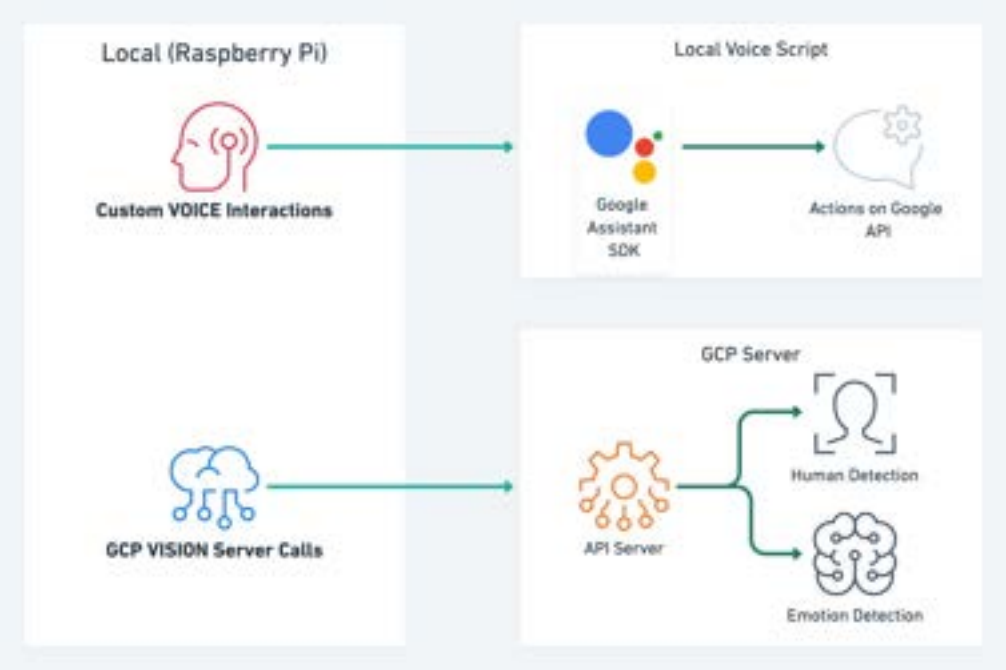


Figure 3.3.1: Architecture of vision and voice systems

Vision:

The vision subsystem uses the attached camera to detect humans and their emotions. The script takes in live video feed and detects humans face and legs; once it detects a face, it will try to detect emotions. The software utilizes Tensorflow (Keras API) and OpenCV with Python. This information will allow the BruinBot to know if it should approach a person. The vision subsystem works locally on a computer; It is able to detect human faces and output emotions if a face is found. It uses the biggest face in the frame to use for emotion detection.

Examples of the emotion detection can be seen in the following images:

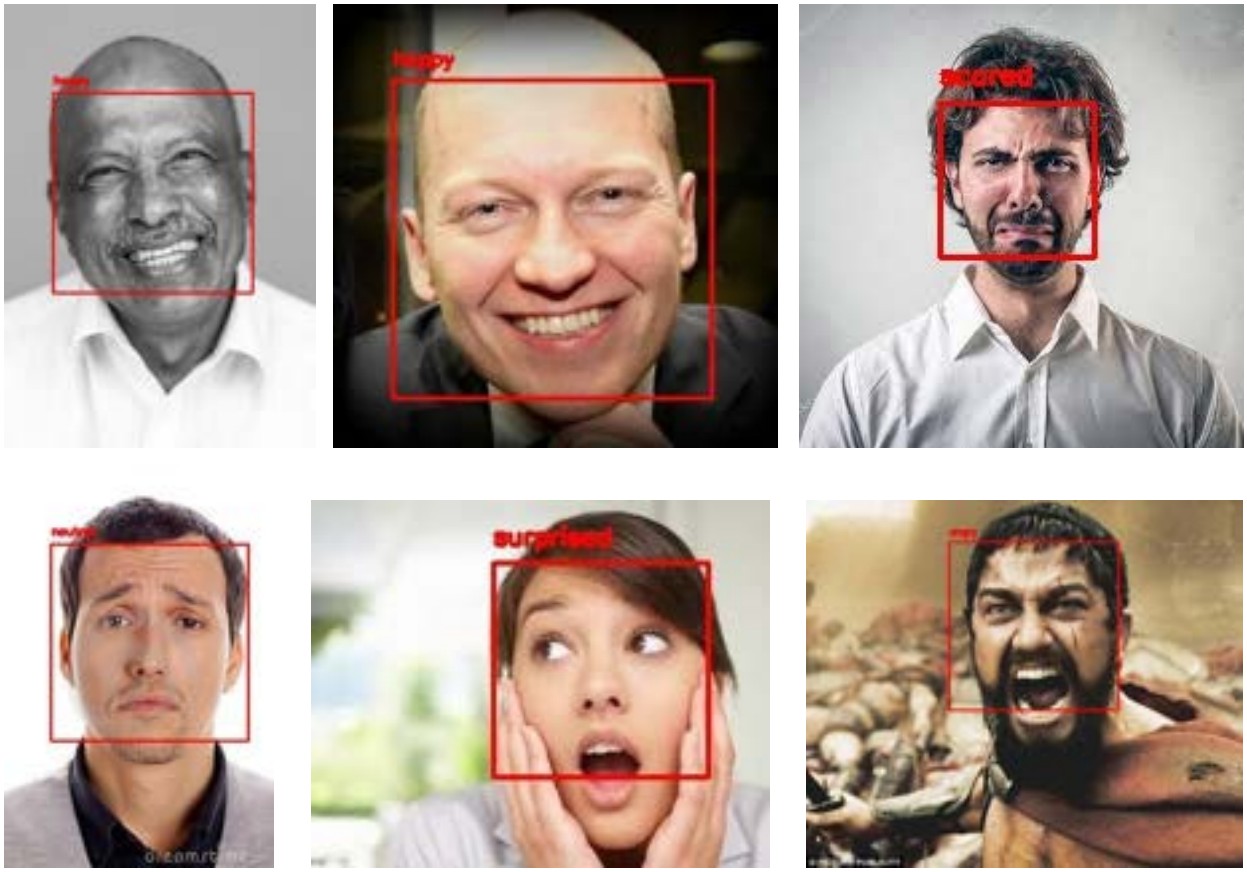


Figure 3.3.2: Examples of emotion detection on stock images using our detection algorithm.

Voice:

The voice subsystem uses the attached microphone and speaker to interact with the user.. It uses python's SpeechRecognition library for recognizing a user's speech and python's pyttsx3 text to speech library to speak out the robots responses. These are used along with the touchscreen GUI(**3.4**) to create a conversational user interface that can be run with a command to interact with a user.

The voice subsystem also works successfully locally on a computer. It is able to listen for a set of questions or prompts and respond to them correctly. The prompts currently supported are:

- What is the weather ?
    - Gives the temperature in Los Angeles based on pytemperature python library
- Do a dance
    - Currently just says that it is doing a dance
    - Implementation of a physical dance is left as an exercise to the reader
- Sing a song
- Play Music
    - Plays a random song from a list of youtube urls in voice.py
    - Youtube url is converted to audio using python-vlc library
- How old are you

    - Gives a random age from 0-186 as bruinbot's age
- Tell me a fun fact
    - Gives a random fun fact from ucla taken from a list of fun facts in voice.py
- Who are you?
    - Gives an introduction of who bruinbot is
- Tell me a joke.
    - Tells a random joke from axju-jokes python library
- Let's play a game
    - Plays a game where a user has to guess a random number between 1 and 10
- Goodbye Bruinbot
    - Bruinbot says goodbye to user and ends the interaction

The voice code is currently stored in the BruinBotVoice github in voice.py.

3.4 Touchscreen GUI

We have developed a simple GUI app in Python using the kivy library. The app has six on screen buttons, and pressing each button prints a different message to the console. This emulates the core functionalities for the BruinBot GUI app, which is to let the user select a particular payload, then the payload system dispenses the product. For the final integration, the message will be sent to the central control unit, which then relays the information to the payload subsystem for actuation.

Figure 3.4.1: Mockup of touchscreen GUI (left) and final functional Kivy interface (right)

3.5 Facial Expression Control

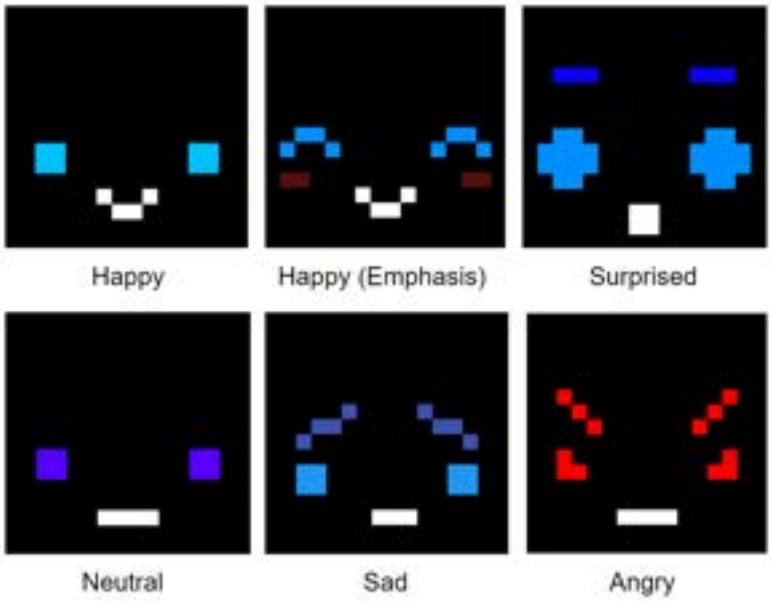


Figure 3.5.1: BruinBot's range of emotions

BruinBot's emotions are displayed to the user via a 16x16 LED panel grid hidden within the head system. The low resolution of this system proved to be a huge limitation, but also aided in keeping the robot's expressions simple and appealing. The code for the face is run off an Arduino Uno Board, which is responsible for displaying every pixel and frame of the custom animations as well as handling interruptions that may be received from the Main Microprocessor at any time. To facilitate this, each emotion is written as a series of different frames, where the emotion checks for interruptions at specific points within its runtime. Frame functions consist mainly of commands to turn on and off individual pixels. The smooth transitions between animations are created by setting the "happy" emotion as the default display, so each animation must transition in and out of "happy" while changing between emotions. If the emotion code receives an interruption (sent as a Serial command from the Raspberry Pi via USB), it will set a new emotion "target" and then trigger transitions until the new "target" emotion is reached.

3.6 Web Dashboard

The Web Dashboard is a website used to monitor and control the robot. The **`/logs`** page provides real time status about the robot. The console displays error logs and other important messages. Below the console, the robot speed, direction, and status.

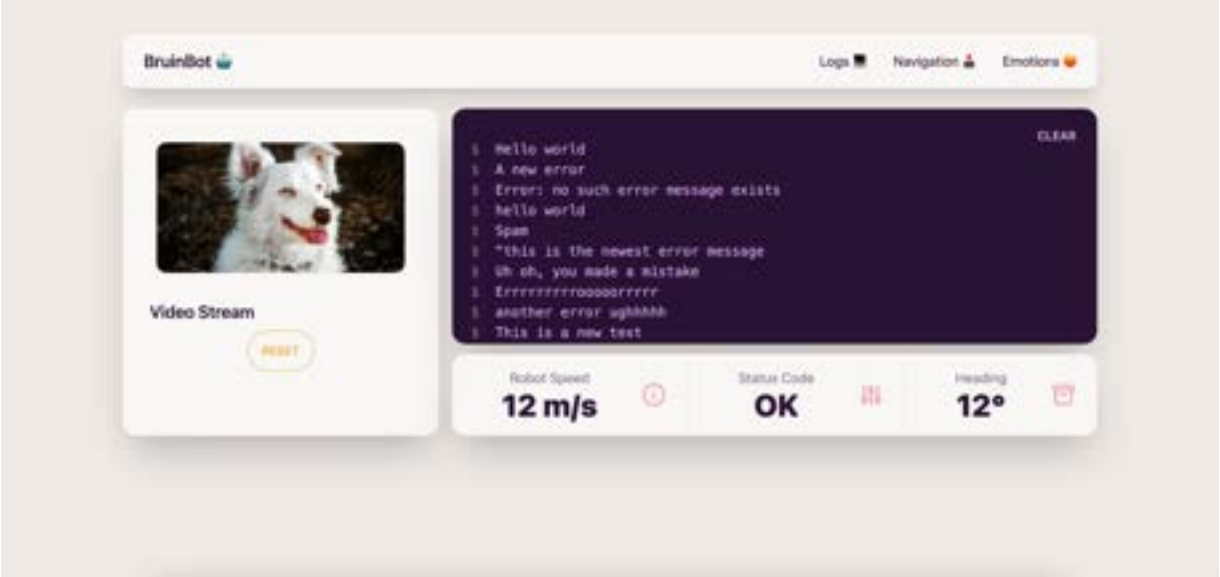


Figure 3.6.1: Home page of the web dashboard showing logs of activity and video feed.

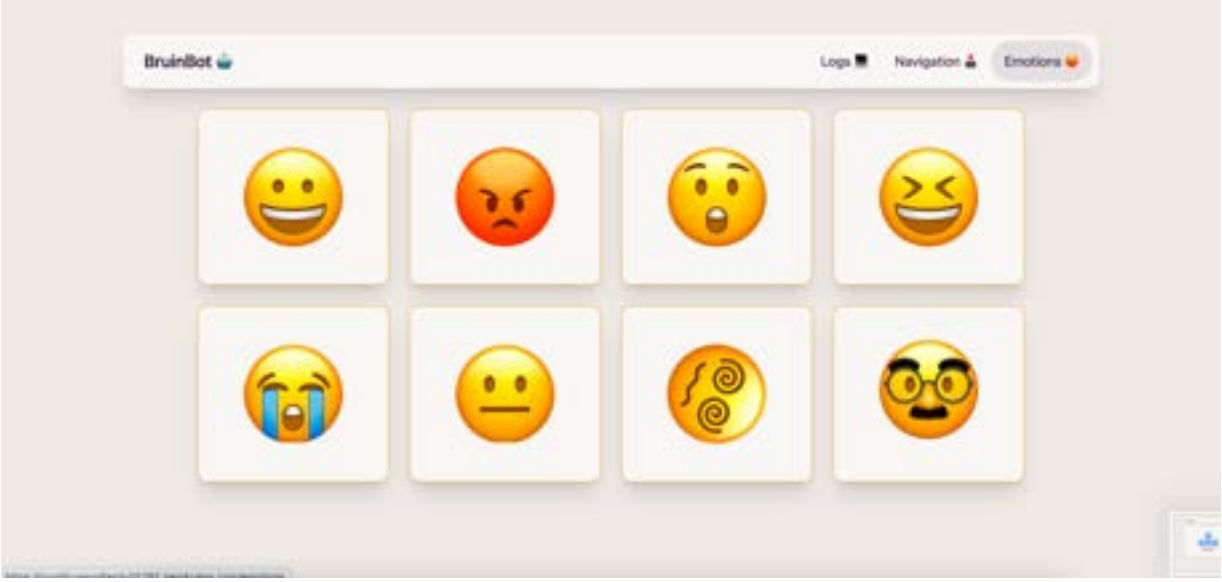


Figure 3.6.2: The Emotions page of the web dashboard controls displayed facial expressions.

The **`/emotions`** page is used to control the facial expression of the robot. The robot has an LED panel used to display various faces. Clicking each button will initiate a transition from the current expression to the expression represented by the emotion.

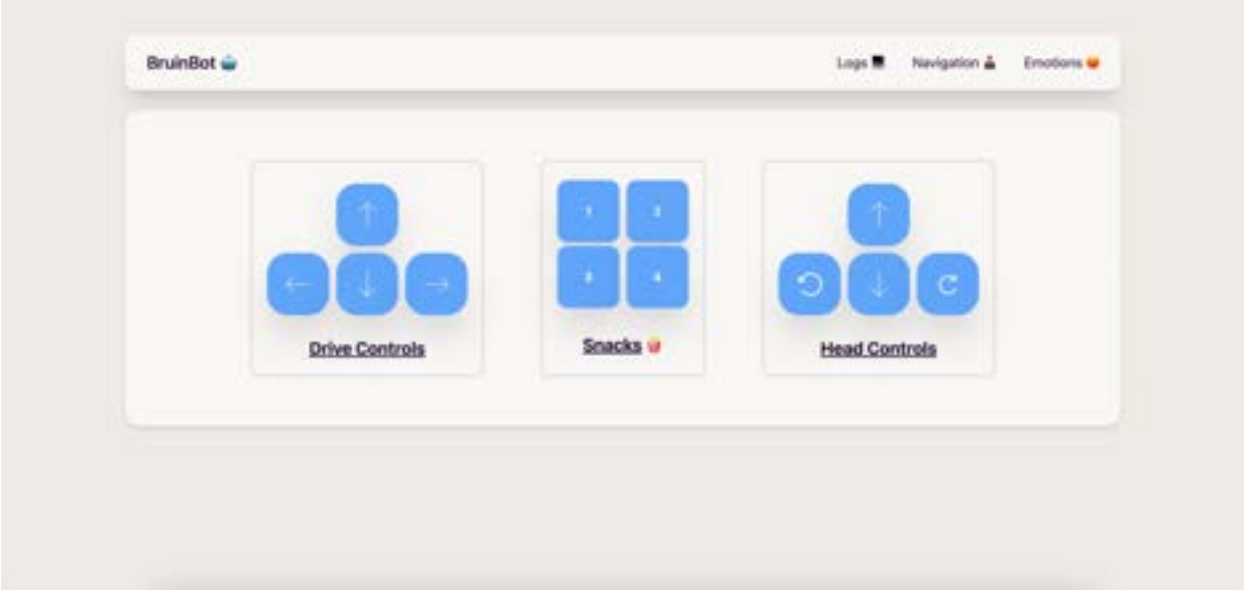


Figure 3.6.3: The Navigation page of the web dashboard contains controls for mechanical systems.

The **`/navigation`** page is used to control the mechanical elements of the robot. *Drive Controls* is used to drive the robot. *Head controls* is used to rotate the robot's head left, right, up, and down. The robot has four built in snack dispensers which operate similar to a vending machine. The user can select any of these slots to dispense from using the *snacks* panel.

The Web Dashboard is built using flask—a python backend framework. The frontend is styled with the help of the Tailwind CSS framework. The dashboard is hosted on the Heroku PAAS platform.

The Web Dashboard communicates with the robot in two ways. The first way is through the Firebase database. Error logs, settings, and more are added to the database via the robot's onboard Raspberry Pi. The Web Dashboard reads from this database and updates accordingly.

Robot control is accomplished in a different manner—via HTTP communication over a local Wifi Network. In order to utilize the control features of the dashboard, the user must connect to the robot's onboard wifi network. The dashboard will then begin to send requests to an HTTP server running on the Raspberry Pi.

The code for the Web Dashboard can be found in this Github repo.

4 Results of Physical Assembly

4.1 Chassis Assembly

Figure 4.1.1: Initial Construction of the top section of the chassis (completed in 2020)

The only physical assembly completed in BruinBot's first year was the top section of the chassis, prepared as a proof of concept for the design review. This section was finished with the plan to soon construct the rest of the chassis, along with the rest of the robot's systems. This plan was interrupted by campus shutdowns due to the COVID-19 pandemic, and manufacturing work was not resumed until Fall 2021. At that point, the remaining bars were cut, and the framework was fully created to build the other systems.

The chassis construction was relatively straightforward. One issue encountered was that the specific assembly order required for the chassis bars made it difficult to quickly take apart the chassis as intended. Since the internal connectors engage in all directions, the chassis can only be taken apart by pulling off multiple bars at once by pulling them in one direction. It is impossible to remove (or add) a single bar

without disassembling an entire section. However, this difficulty also provides the system's strength by bracing the structure in all directions.

Another issue encountered was in the interface between the acrylic panels and the chassis frame. Since the panels were laser cut, they had perfect dimensions taken from the CAD. The chassis frame, however, was hand-manufactured and suffered from human error in the measurements. This produced many places where the holes in the acrylic did not perfectly line up with the holes in the bars, and extra adjustments or hole enlargements were required.

Figure 4.1.2: Complete PVC Chassis Structure (left), and chassis with the later addition of concealing acrylic panels (right)

4.2 Head Assembly

The head was mostly designed at the end of the 2019-2020 school year, and some work was done on prototyping over the 2020-2021 school year while most work was remote. The prototype was printed on a member's personal printer, to demonstrate the fit and size of components. Eventually, the CAD was modified enough so that a new print was required. (The most important new addition was a properly sized mount for the raspberry pi camera board). The neck assembly was also heavily modified over the 2020-2021 year, and built in Fall 2021.

Figure 4.2.1: Prototype head structure contrasted with final head print (left). Final head with installed LED panel and face plate (right).

Figure 4.2.2: Neck gearing assembly seen from below (left), and head and neck assembled together and attached to the chassis frame (right)

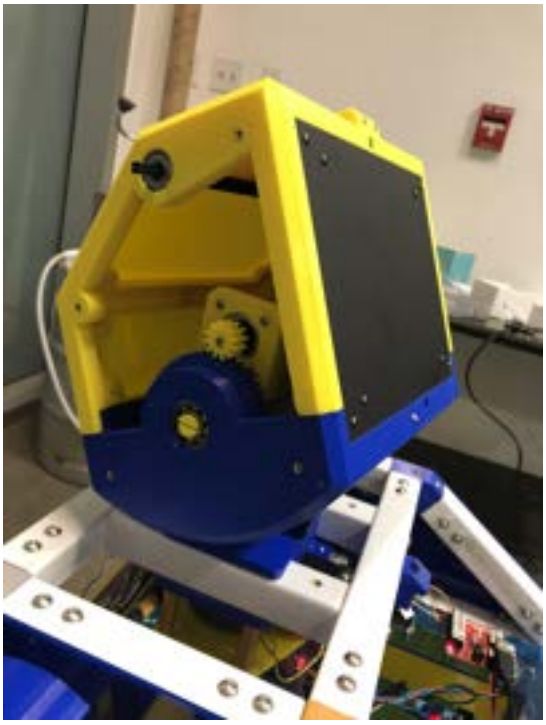


Figure 4.2.3: Head with side panel removed, demonstrating the interface between the nodding motor and gears built into the side of the head.

4.3 Payload Assembly

Dispensing System

When prototyping the dispensing assembly, we ran into several issues during preliminary testing. Firstly, when mounting the servos, we neglected to account for the space required for the wires, which required a slight redesign of the acrylic mounting plate. Additionally, our initial servo selection (blue 9g plastic servos) did not provide sufficient torque function as expected. While the servo provided enough torque to rotate the spring itself, we did not account for friction between the vending coil and the acrylic, which the servo was unable to overcome. To remedy this, we switched to a high-torque continuous servo which is now used in the final design.

We also ran into some issues regarding the acrylic panels, as we determined that the actual thickness of the panels was not exactly 1/8th inch, but closer to 0.118”. Additionally, the kerf of the laser cutter shifted the dimensions of our parts by approximately 0.007”. These two factors meant that our initial prototypes fit together very loosely and were not sufficiently sturdy. This was remedied with some simple dimensional adjustments to account for manufacturing error.

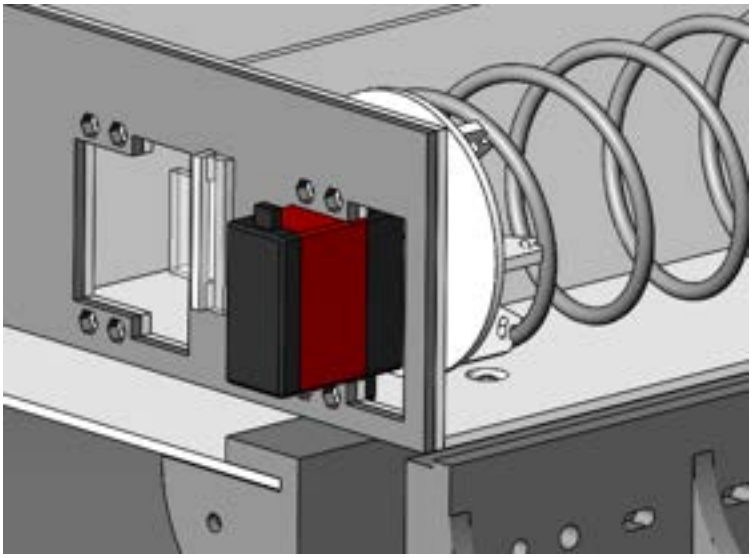


Figure 4.3.1: High torque servo mounted into acrylic dispensing system.

Conveyor System

When attempting to assemble the driven drum part of conveyor belt assembly, we ran into a design flaw that created a circular dependency for the order of assembly. The motor mount needed to be bolted to the chassis before the motor was attached, but because the drum slotted into both the chassis and the motor, the drum could not be attached without disassembling the chassis. To remedy this, the motor mount for the conveyor belt motor was redesigned and printed with slots for nuts on its upper face, so that the mount could be fixed to the chassis using bolts coming from the bottom. This way, the motor can be attached to the mount, then the drum could be slotted into the chassis and the motor, and finally the mount can be bolted to the chassis.

Experimentation with multiple materials was necessary to choose the conveyor belt material. It needed to be stiff enough to support the weight of the snacks, while flexible enough to roll smoothly over the drums. We originally purchased 1/16th inch neoprene sheets, which proved far too inflexible. We created a make-shift belt out of a thick adhesive vinyl electrical tape with a trash bag backing to test the efficacy of the rest of our conveyor belt assembly. We then tested elastic workout band material, as well as a thin sheet of inelastic plastic, making us realize it was important to use a stretchy material so that it could be more easily tensioned. We eventually settled on a stretchy faux leather fabric that was also stiff enough to support the dispensed items.

Figure 4.3.2: Conveyer belt prototypes. From left to right then top to bottom: neoprene, tape and trash bag, thin plastic sheet, workout band, faux leather.

The conveyor belt material was held together using duct tape during testing for simplicity. The ends of the final belt were sewed together with a zigzag stitch on a sewing machine, which permanently formed the belt into a loop without any loose material. To avoid slippage, we wanted to increase the friction between the 3D printed drums and the conveyor belt. We added strips of self-fusing silicone rubber electrical tape to the drums; this tape is slip-resistant, adding grip.

One issue we encountered is that the conveyor belt shifts towards one side of the drums as the belt is run. We believe this was because the drums were not perfectly parallel or because the belt has a different radius on either side. In order to fix the shift of the conveyor belt, we added walls on the outer ends of the drums so that the belt could not slip off. We needed to increase the wall height from its initial size because the belt was able to fold up and over the earlier, smaller wall. Additionally, the non-driven drum mounts have slots so that the belt can be tensioned. However, because they are fixed with only one bolt, they are able to rotate, making the tensioning and orientation of the drum imperfect. Given more revisions, we would change the mounts to constrain that rotational degree of freedom.

Figure 4.3.3: Entirety of dispensing and conveyor systems fully assembled. An additional yellow acrylic plate was mounted to catch any stray items that may fall out (on the bot's right hand side).

Left Arm Assembly (Functional)

Initial prototyping of the left arm revealed several design issues that needed to be remedied. Firstly, in order to attach various 3D printed components to the aluminum bars, we initially planned on relying on press-fit connections. However, we soon realized that these were unreliable due to 3D printed variability and gradual deformation of plastic over time which degraded the performance of the press-fits.

To remedy this, we attempted a "shaft collar" style securing method where a bolt could clamp the plastic down to the aluminum bar. This design worked well for our purposes and allowed for easy assembly and disassembly. The only issue we ran into with this design was that the compliance required resulted in a relatively fragile part, which sometimes broke by accident. Eventually, the parts proved fragile enough that we opted to drill holes directly through the aluminum bars and use them to fasten the collars. This caused some issues with alignment (since it is ideal for the 3D-printed components to be as parallel as possible). In hindsight, it would have been preferable to utilize a square bar instead of a circular bar, which would make alignment much simpler.

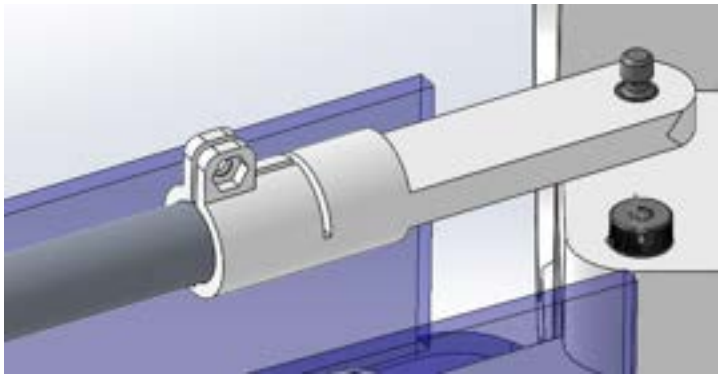


Figure 4.3.4: Shaft-collar style attachment design to secure plastic parts to aluminum bars. This design proved too fragile to use, so holes were drilled directly through the aluminum bars and secured with bolts.

The biggest challenge we ran into regarding the left arm assembly was providing sufficient torque to lift the arm. Our initial motor selection was a 150:1 geared DC motor with built-in encoder in order to achieve smooth PD-controlled motion. However, we soon discovered that this motor did not provide sufficient torque to lift the arm, despite the listed stall torque as being well above the expected torque that the motor would be subject to under the weight of the arm.

We speculate a couple of reasons for this behavior. First, we neglected a warning provided by Pololu (the motor supplier) in which they state that listed stall torques are "theoretical extrapolations", and motors will likely fail at a much earlier point.

> **Note:** The listed stall torques and currents are theoretical extrapolations; units will typically stall well before these points as the motors heat up. Stalling or overloading gearmotors can greatly decrease their lifetimes and even result in immediate damage. The recommended upper limit for continuously applied loads is 10 kg-cm (150 oz-in), and the recommended upper limit for instantaneous torque is 25 kg-cm (350 oz-in). Stalls can also result in rapid (potentially on the order of seconds) thermal damage to the motor windings and brushes; a general recommendation for brushed DC motor operation is 25% or less of the stall current.

Figure 4.3.5: DC motor stall torque disclaimer; a possible reason for why the purchased motor failed to lift the arm. Source: Pololu (https://www.pololu.com/product/2828).

Another reason the motor could have failed is because we mistakenly used bolts that were too long in two of the mounting holes for the motor. While we did not realize this at the time, the motor's mounting holes enter directly into the gearbox (we have no idea why). Thus, if long bolts are used, they will screw directly into the gears and potentially damage the motor's internals.

> **Warning**: Do not screw too far into the mounting holes as the screws can hit the gears. We recommend screwing no more than 3mm (0.12") into the screw hole.

Figure 4.3.6: Another motor disclaimer. Source: Pololu.

Unfortunately, these disclaimers were not spotted by any members until after failures had already occurred. In summary, the original selected DC motor was replaced by a high-torque servo that was obtained in the ASME lab. It was determined that this high-torque servo provided sufficient torque and was used in the final product.

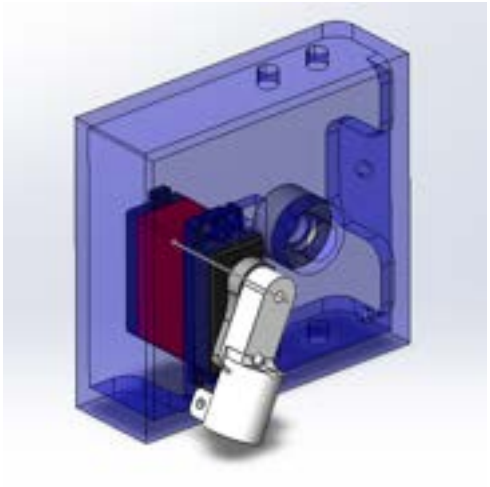


Figure 4.3.7: Adjusted shoulder motor mounting block to accommodate high-torque servo.

Right Arm Assembly (Non-Functional)

The right arm was 3D printed with PLA filament and assembled with minimal difficulty. However, we ran into various issues with the attachment of the right arm to the mounting block. The spacer that was attached to the upper sleeve on one side and the mounting block on the other was extremely difficult to line up and attached securely. The 3D printed piece had intended for metal inserts to be added to all four holes, however this process was prone to human error as it was difficult to align all four holes as intended. This caused the arm to swing and sway as BruinBot was driving.

A second iteration of this spacer utilized two holes that went through the entire block. Two long screws would run from the mounting block to the outer sleeve, where a nut would keep the screw secured in place. However, this did not keep the arm from continuing to swing when driving BruinBot.

Our final solution was to have another point of contact for the right arm by attaching a magnet to the paw. This magnet would ensure the right arm would be stable during drive and stay in place to mirror the height that the left arm would be resting at. Even this solution, however, would cause the arm to sway if it ever detached from the magnet (which happened somewhat frequently). More work is needed to permanently attach the paw to the robot's side.

Figure 4.3.8: Left and right arms fully assembled. Their aesthetic appearance from the outside is nearly identical, however their internal structures differ significantly.

Another challenge that we faced while assembling the right arm was the assembly order that was implied from the design. The aforementioned spacer, which attached the right arm to the mounting block, had to run through the shoulder piece. Therefore, the arm could only be mounted at the same time, or after the shoulder was mounted. This shoulder obstructed the view and location of the holes for the screws. It also made it impossible to reach the outward facing side of the arm to hold it in place when assembling or to hold the nuts in place for the second iteration of the spacer.

4.4 Drivetrain Assembly

The drivetrain system underwent by far the most changes of any of BruinBot's systems between it initial conception and eventual manufacturing. Fortunately, none of the original 2019-2020 design had ever been physically built, so the redesign could start from a 'clean slate' of design possibilities. The eventual drivetrain assembly consisted of a simple frame of 80/20 aluminum extrusion framing, with bolted-on waterjet aluminum plates supporting the wheels on live axles. As discussed in the Physical System Design section, a change was also eventually made between 4 rubber wheels and an Omni Wheel system for easier turning.

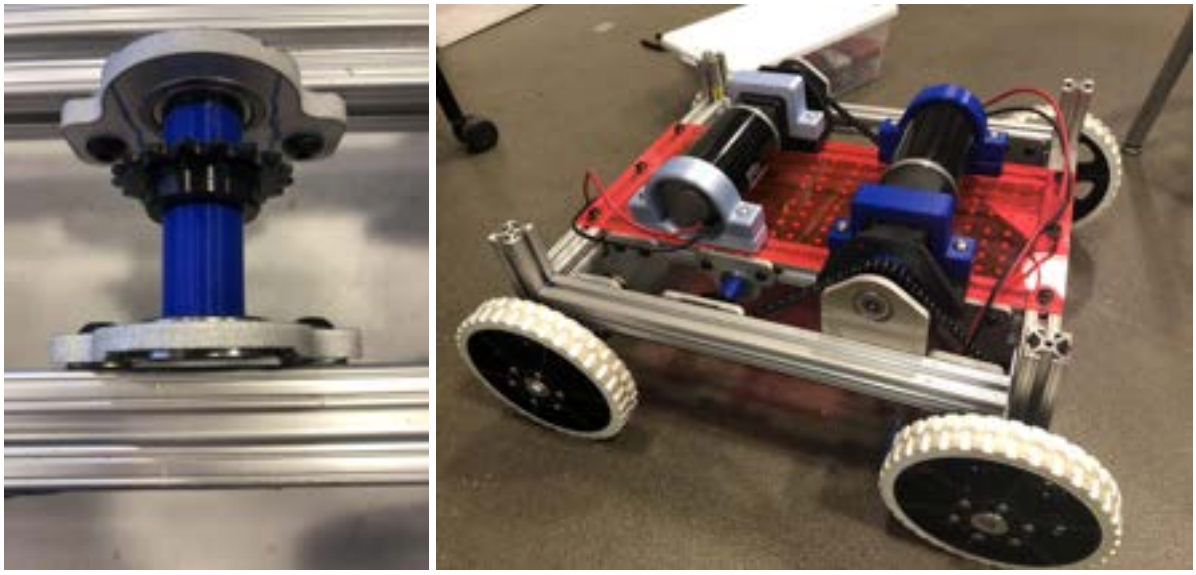


Figure 4.4.1: Closeup of aluminum plate mounting for the idler sprocket (left). Final Drivetrain assembly (before swap to front Omni Wheels) (right).

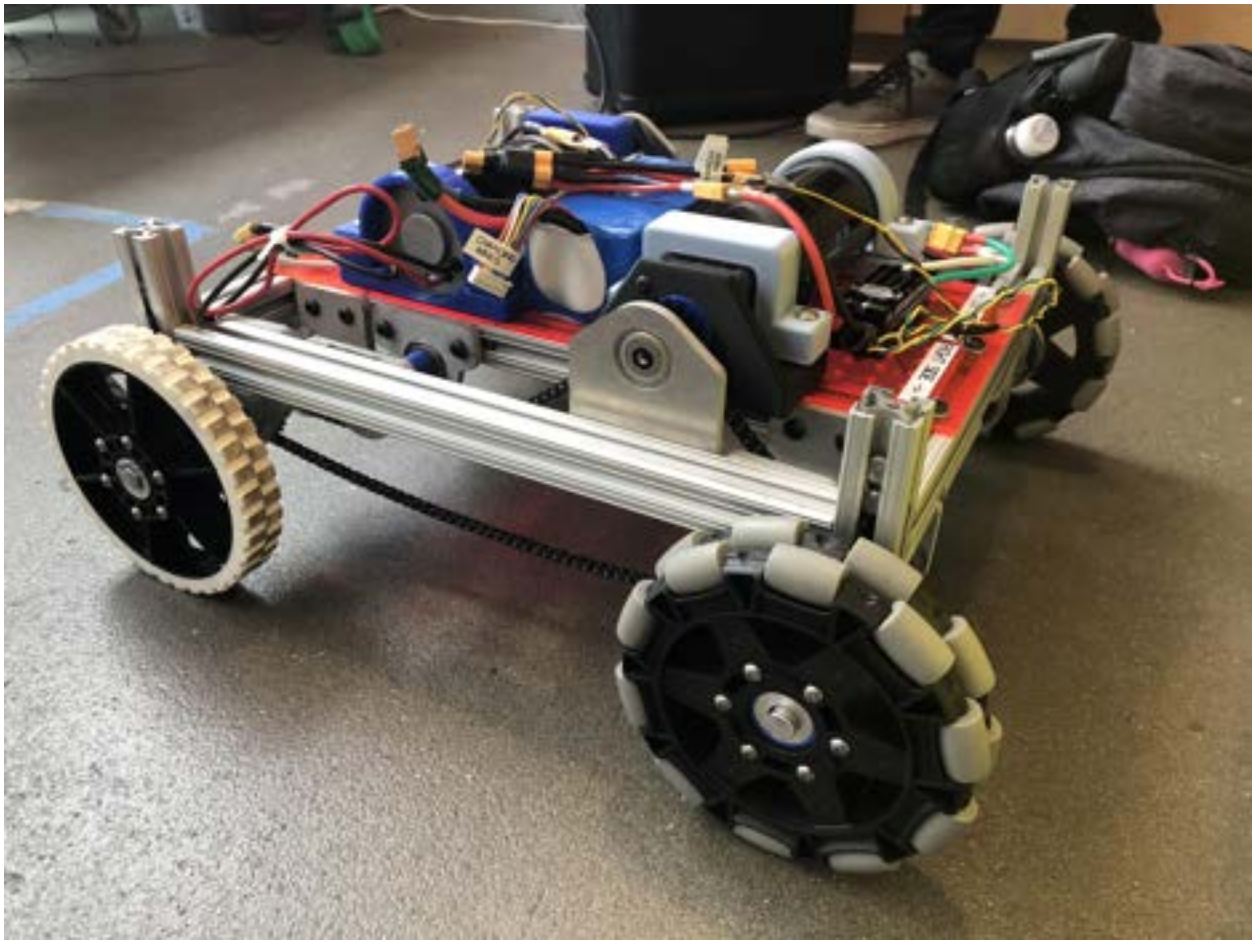


Figure 4.4.2: Final drivetrain carriage with omni wheels and mounted electronics.

4.5 Electrical Systems

The physical assembly and mounting of the electrical systems was almost an afterthought when compared to the rest of the physical assembly of the robot. A lot of work was put in on designing a functional electrical system in Fall 2021 and Winter 2022, because no work had previously been invested in it. Many changes were made as the realities of certain systems became more clear (and many components were swapped out due to unexplained errors or issues). The actual mounting of the electronics thus could not be completed until significant testing of all components had occurred, to ensure nothing else would need to be swapped out.

Initially, the division between the top "chest" area on the robot and the lower payload area was planned to house a single horizontal plate where electronics would be mounted. In practice however, this was not enough space to house all needed components. The neck mounting protruded downward into the space reserved for electronics, and there was simply not enough surface area to house everything with sufficient clearances. Thus, a two-level approach was devised, with a lower level plate suspended inside the robot on standoffs. The upper level houses the batteries, head arduino, and face arduino systems, while the lower level houses both Raspberry Pis and the Payload Arduino system.

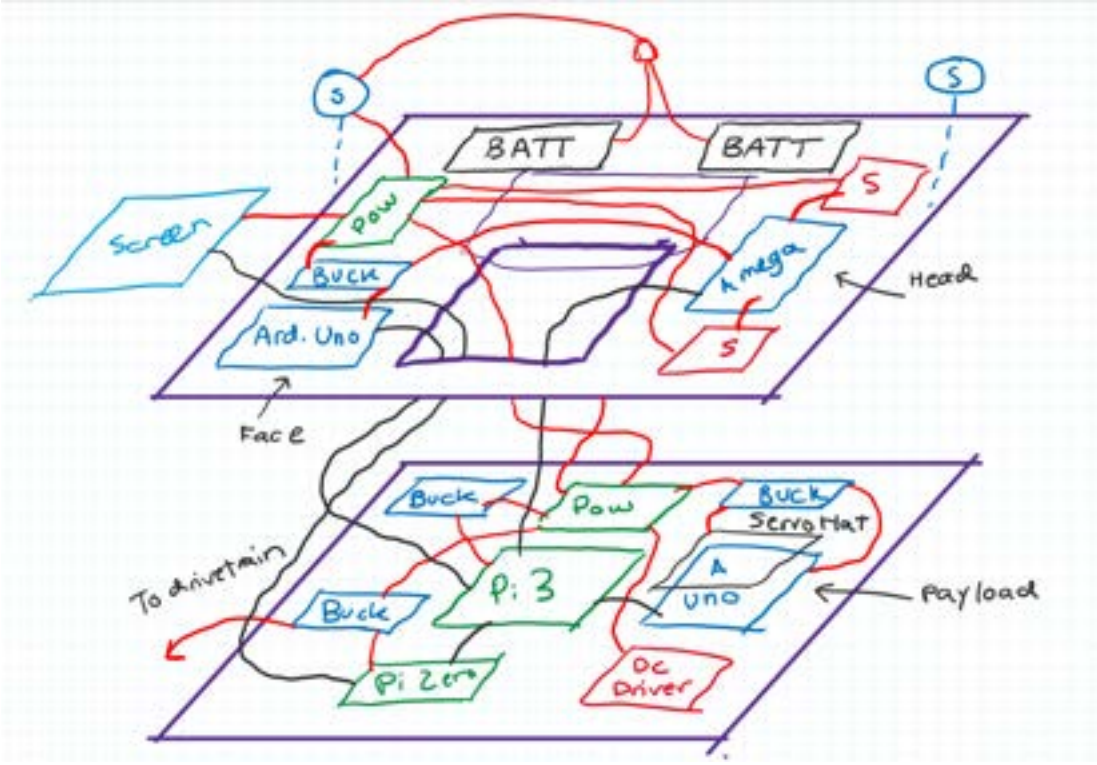


Figure 4.5.1: Diagram showing the distribution of boards and wires between the top and lower electronics tray. Black wires represent signal, and red wires represent power/ground. (A few connections may be outdated).

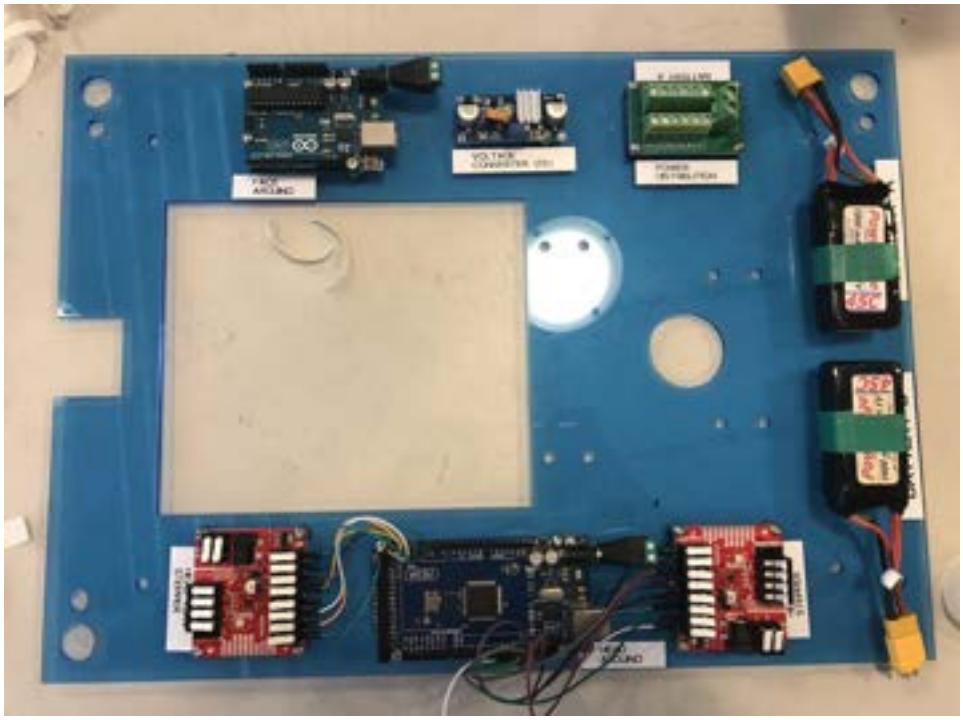


Figure 4.5.2: Top electronics plate before wiring. The large rectangular hole allows access for wires to the lower plate, while the circular hole is placed directly underneath the neck so wires can continue down through both levels.

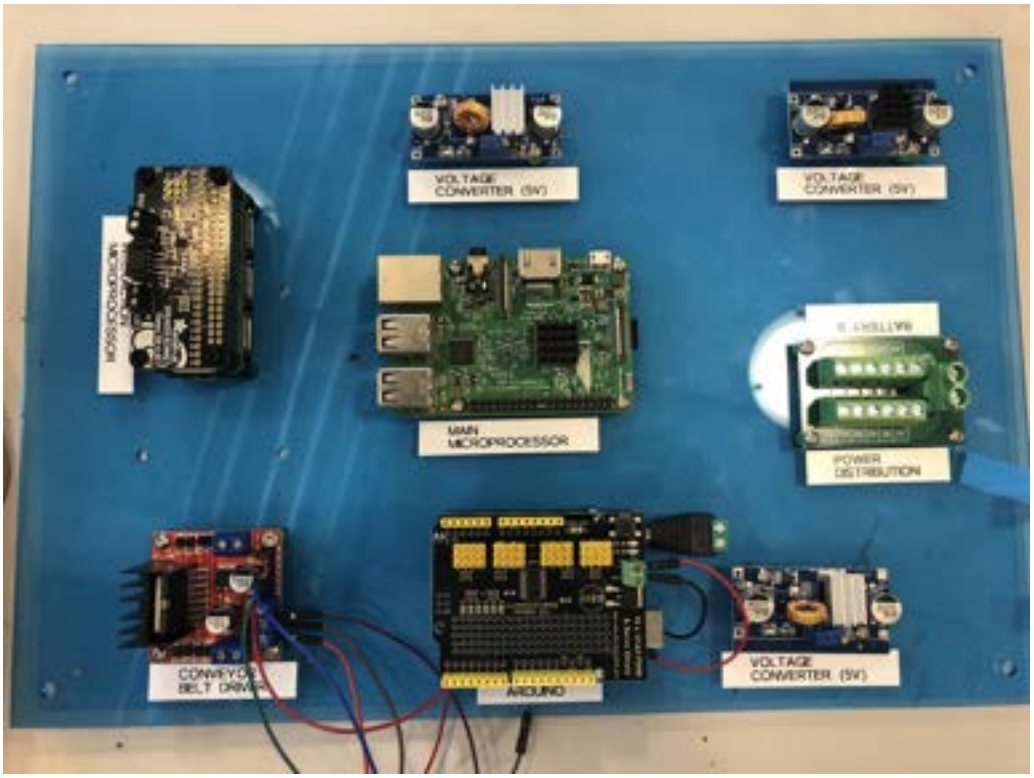


Figure 4.5.3: Lower electronics plate before wiring. The Interaction Processor is shown here as a Pi Zero, though it has since been changed to a Pi 3.

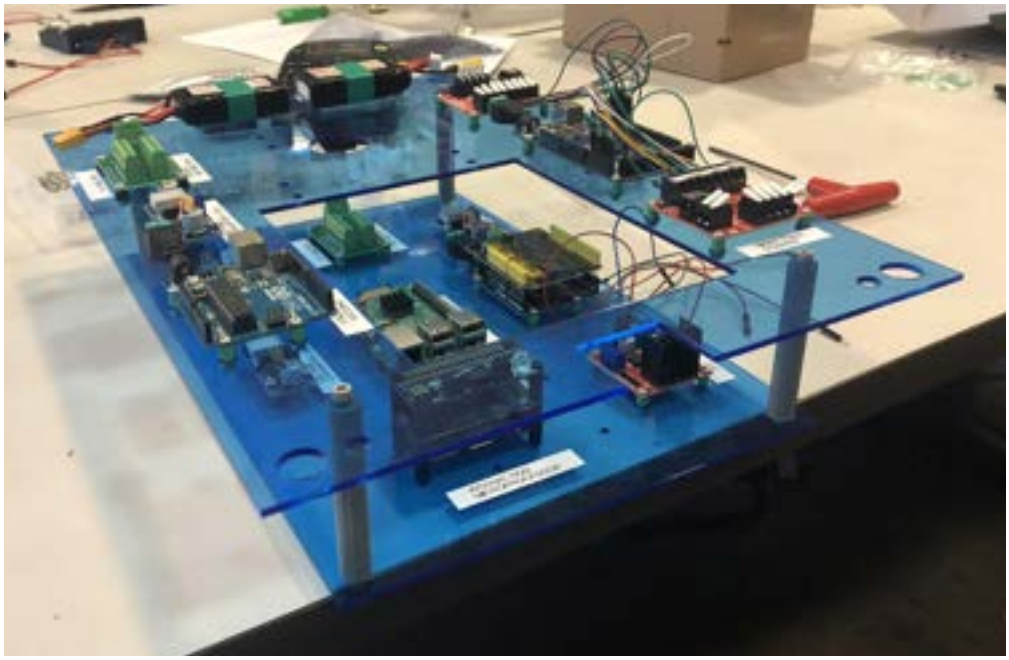


Figure 4.5.4: Plates assembled together with standoffs.

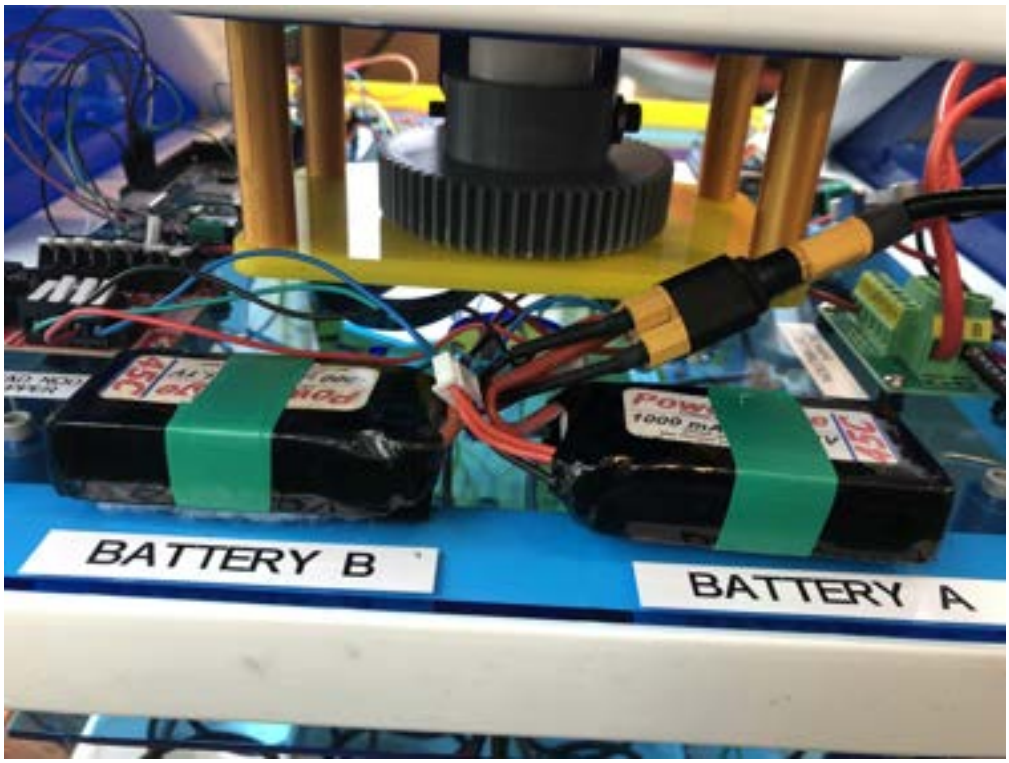


Figure 4.5.5: View of the inside of the top electronics plate when installed, showing the space limitation due to the neck mount.

As a final component of the electronics system, two switches were installed that are used to turn on the robot. The top switch, a simple toggle switch that glows yellow when turned on, controls power to all the

components in the two electronics plates. This switch is mounted on the right side of the robot, next to the right shoulder. (A second yellow switch is mounted on the left side, but currently is not wired to anything.) The drivetrain system, including the higher-voltage drivetrain battery, is controlled with a large circuit breaker switch mounted to the bottom rear of the robot. This switch was used because it was the only available switch able to handle upwards of 60A of current (the drivetrain system was designed with a 60A current maximum.)

Figure 4.5.6: Circuit breaker switch controlling the drivetrain system. Snap the black handle inwards to power on the drivetrain, and press the red button to power it off.

4.6 System Integration

The final mechanical assembly of BruinBot is almost entirely complete. From the outside, the robot looks exactly as it will in its fully-functional form, with a completed aesthetic exterior design and functional mechanical components (except for those detailed in the next section, “Remaining and Future Work.”). The main remaining tasks are in software, to fully integrate together all of the various functions and behaviors of the robot, though some work remains in mechanical and electrical tweaking.
GIFs of the robot moving and performing various actions can be found in this folder.

Figure 4.6.1: BruinBot's transformation from CAD model to finished robot.

Currently, BruinBot can:

1. Establish the correct voltage of power to (almost) all its subsystems with the flick of two switches.
2. Zero its head position upon startup in both degrees of freedom (turning and nodding) and receive a command from the main Raspberry Pi to adjust its head position.
3. Drive smoothly in any direction (forward backward, left, right) over long distances, controlled with a handheld remote control.
4. Create a network using a small router, which it uses to connect its two Raspberry Pi's to the locally hosted web dashboard.
5. Display facial expression animations (triggered by buttons on the web dashboard) and automatically display smooth transitions between animations.
6. Take input from the web dashboard to dispense one of 4 payload items from its internal channels, including dispensing via coils, running the conveyor belt, and lifting its arm to present the gift to the user.
7. Display the GUI app to the touchscreen panel, and use the touchscreen to trigger different actions and interactivity.
8. Use the speaker and microphone to take in audio input from the user and speak a response.

Remaining tasks, including the integration of most of these actions, are detailed below.

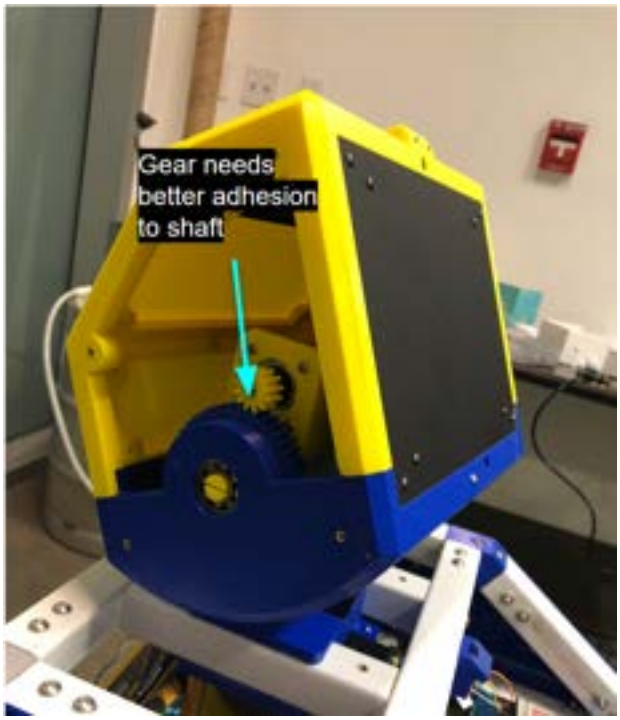
5 Remaining and Future Work

While a huge push was made in the 2021-2022 academic year to bring the robot to completion, unfortunately not all of our goals were reached and significant work remains to bring BruinBot into a fully-functional, autonomous, and user-friendly reality. Most of the remaining work is entirely within software integration, meaning that the robot can be fully completed without extensive work on any mechanical or electrical systems. However, there are some electromechanical issues that were not fully resolved at the end of the year, solely because we had completely run out of time before classes and campus activities adjourned for the summer.

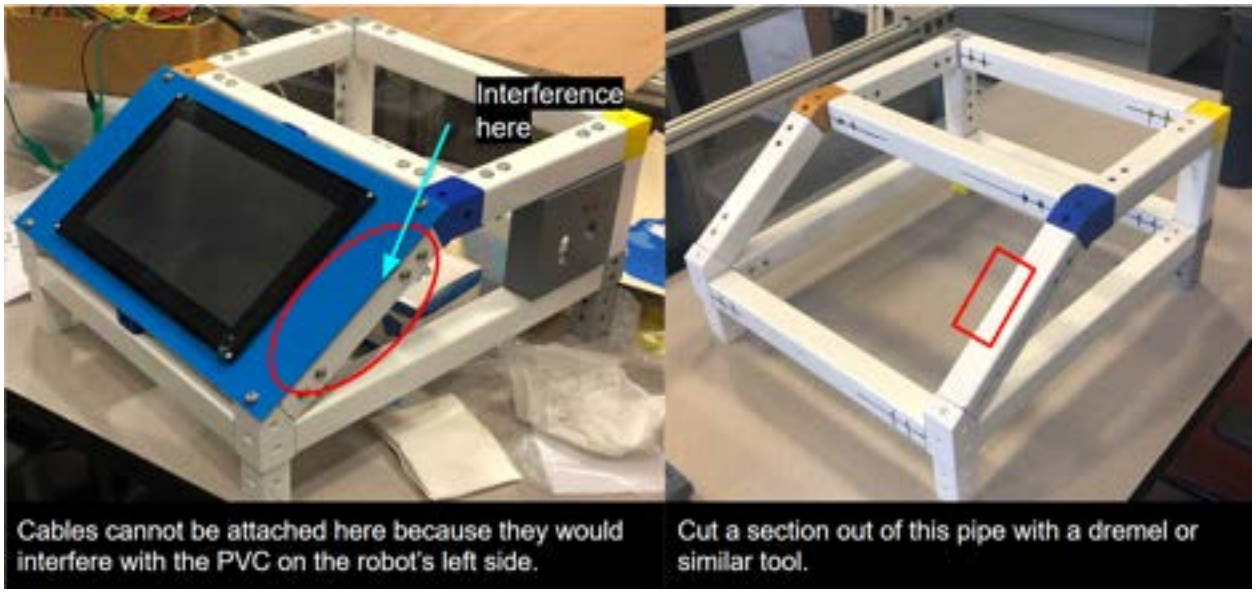
Here is an extensive list of remaining issues as well as what must be done to fix them.

5.1 Remaining Mechanical Tasks

1. Head and neck rotation issues
    a. The integration of two stepper motors for the head's rotation was not fully finished. Both motors were installed, and both limit switches were placed in the correct positions. In theory, the head can boot up, set its zero positions using the limit switches, and accept further motor control commands via a serial connection to the main Raspberry Pi. This code was written but could not be fully tested. The main mechanical issue is that the side-to-side head rotation is encountering a strange resistance, occasionally causing the motor to stall and exhibit a strange 'skipping' behavior where it is unable to turn further.
    b. We were not able to determine exactly what was causing this issue, so we are not sure what should be done to remedy this. Determining and removing the source of the resistance would hopefully be helpful to make the motor turn more smoothly.
2. Nodding gear friction
    a. The nodding motion of the head is created by a stepper motor mounted inside the head. The motor's axle is mounted to a small gear, which interfaces with a large partial gear inside the wall of the head structure. The small gear is attached to the motor axle with a tight friction fit as it is a smooth round shaft. Currently, if the head is in certain positions, the weight of the head provides too much torque and it overpowers the friction between these components, causing the motor to rotate without moving the head.
    b. This is only occasionally an issue, so it could probably be fixed with adding adhesives between the gear and shaft. Alternately the gear can be redesigned with a threaded insert and set screw so it can be tightened on the shaft.



3. Touchscreen panel cable allowances
    a. An oversight was made during the design of the chassis system where the touchscreen panel cannot be secured in place to the ‘chest’ of the robot because the protruding cables interfere with the PVC pipe running alongside the chest. The panel fits well when it is attached without the HDMI, USB, and Power cables connected, but with these cables in place (especially the bulky HDMI cable) it is impossible to fasten the plate correctly.
    b. A possible fix for this would be to reposition the mounting position of the ports on the acrylic backing of the touchscreen (accomplished by cutting additional holes) though this would make it difficult to access the ports to plug and unplug the cables. Another possible fix would be to remove a section of the PVC pipe, creating enough clearance for the cables



4. Right arm mounting
    a. The right arm of the robot does not move, and should be sturdily attached to the side of the robot. However it was never securely fastened as the shoulder mount turned out to be flimsy and allowed too much unwanted movement and rotation of the arm. The magnet at the side was not securely attached so the hand is still free to rattle as the robot moves.
    b. Ideally, some kind of bolt should be installed to hold the hand securely at the robot's side, or even a new shoulder mount should be created. The magnet solution is insufficient.

5.2 Remaining Electrical Tasks

Overall, the electrical systems are the most complete in the whole robot, with almost every connection and power distribution sorted out to not require editing. However there are a few small tasks that were not finished as they came right up against our deadline.

1. Wire the Drivetrain Arduino
    a. An important part of making the robot autonomous is not yet implemented, which is adding a separate Arduino Uno board to run the drivetrain system with PWM signals. There is wiring in place for this system, but no code written and no physical board installed. Currently, the 5V power intended for the arduino is instead wired directly into the receiver, so the robot can be driven with a remote control. Autonomous behavior requires that the robot is instead driven with signals from the main raspberry Pi. There is even a feature implemented on the website for driving the robot, but it can't be finished without the Arduino and PWM signal code.
    b. An arduino can be easily plugged into the 5V power wire that drops the the lower section of the robot in place of the receiver. Two of the GPIO pins should be used for the PWM signal cables of the Talon controllers, and drive code should be written to receive serial

commands from the Pi and respond by sending appropriate PWM signals to the motors to go forward, left, right, backwards, etc. Lastly, a USB-B (M) to USB-A (M) cable should be used to serially connect this arduino to the main pi. This is a relatively simple task that wasn't finished based only on time constraints.

2. LiDAR system
    a. The LiDAR system was intended as a failsafe for the robot's driving, to create an emergency stop if a nearby object is detected. Code was written to interpret and smooth the signals from the LiDAR unit, but it was never fully functional. It was also never integrated with drive code, since drive code was not written. However it is an important piece of making the robot's driving safe.
    b. The LiDAR unit is mounted in the front of the robot on an adjustable-position mount. It can point at the ground angled, or point straight ahead. The angled position was added to detect both ground-level obstacles as well as drops (a low distance reading indicates an obstacle, while a higher than normal distance reading indicates a drop.) However the readings from this system were not reliable. More work is needed on the code to provide a reliable reading, and the system should be physically wired to the Drive Arduino.
3. Wiring power to the touchscreen
    a. Initially, the wiring diagram assumed the touchscreen would be powered with a 12V power source, which could be drawn directly through the battery through one of the many parallel channels in the power distribution boards. However, when attempting to hook up this system we realised that the touchscreen should actually be powered from a 5V power source. 5V sources are more difficult to come by in our system, as they must be drawn through the four 5V Buck Converters. There was no available outlet from one of these buck converters at the time of this realization, so the touchscreen power was never fully established.
    b. Adding this source of power is a very simple task. The first option is to solder two additional wires onto a buck converter to split a new channel of 5V power and ground to the touchscreen. Some of the buck converters are already split, but there is at least one that currently only has a single channel.
    c. Alternately, since the touchscreen is a relatively high-power draw device, it should receive its own buck converter for an unsplit 5V power supply. Additional buck converters are available in the X1 supplies, and can be wired into the 12V power distribution boards. Make sure to adjust the output voltage to 5V with a screwdriver before hooking it up.
4. Fix the wiring to the nodding stepper motor
    a. The SparkFun ProDriver that controls the nodding stepper motor has been having issues where wires come loose spontaneously. This is due to the annoying clip terminals on this board, which may be faulty due to overuse. The power and ground wires especially have been coming loose and de-powering this board. You can tell if the motor is depowered if the 'nodding' motion offers no resistance when moved by hand when the robot is powered on. Even if the motor is not receiving signal, it should provide a resistance

b. The clip terminals should be very securely fastened, which will be easier to do if the wires are swapped for thicker ones. Alternatively, a more permanent solution can be sought where the terminals are soldered instead of clipped.

5.3 Remaining Software Tasks

The biggest section of remaining work is in the robot's software. Enough software has been completed that the robot can do individual actions with manual control such as dispensing the payload, generating different facial expressions, and using the touchscreen to trigger voice responses. However, these systems are not completely integrated with each other, and the robot is not at all user-friendly. To run any of BruinBot's scripts, a keyboard is required to type in commands, and this requires detaching the entire touchscreen from the robot and plugging in a keyboard to the internals. Many planned functionalities are also not completed at all.

1. Computer vision to trigger robot behavior
    a. Though the computer vision is individually functional to recognize human faces and detect emotions, there is nothing built into the software structure to actually use this system to influence the behaviors of the robot. Initial plans for the robot's autonomous behavior dictate that the robot should scan for human faces, and when one is detected, it should begin the interaction process. The emotion recognition is also used to play the "let's make faces" game available as one of the interaction options (which is not coded yet). More software integration is required to associate facial recognition with behavior.
2. Computer vision to guide head movement
    a. An important feature that will make BruinBot appear more lifelike and realistic is facial tracking. This feature uses a simple algorithm to calculate the position of a human face in the robot's vision relative to its center of vision. If a face is positioned to the left of the camera's center, for example, the coordinates of the drawn bounding box should indicate that the head should turn to the left until the face is centered. If the head is at its maximum angle, then the body of the robot should move instead; in this way BruinBot can track a person moving 360 degrees around it. This feature, while not critically necessary, would go a long way towards interactibility and would also utilize the head rotation mechanical system well. None of this code is written or integrated currently.
3. "Wander Mode" code behavior
    a. "Wander mode" is the robot's only mobile state when it is operating autonomously. When interacting with a human, it is mostly stationary, but when it is not currently interacting it will "search" for a human by executing a serious of short random movements and turns to appear lifelike and idle. The decision-making behind these movements is random, but also and informed by the feedback from the LiDAR system. When the LiDAR system detects an obstacle, it should access control code to stop the robot, choose a new direction to turn, and navigate away from the obstacle. If no obstacles are detected, the robot should still choose to change direction every so often so it doesn't move too far from its starting position.

4. Behavioral code and overall system integration
    a. Some of the behavioral code is integrated (especially the actions that are triggered by the touchscreen) but not everything is linked together. An end goal is to recreate the entirety of the Behavioral Flowchart in code, even the autonomous behaviors such as “wander” mode and reading sensor input from the camera, microphone, and LiDAR systems. This is still a challenging task, but will be valuable to complete as it will allow the robot to run its functions and demonstrate itself without extensive supervision and control from humans.
5. Running robot’s code upon startup
    a. As discussed in the introduction to this section, none of the code required to run BruinBot can be run without detaching the touchscreen from the front of the robot, attaching a keyboard to the main pi, and manually typing in commands to begin running various scripts. This is an extremely labor-intensive process, especially to someone unfamiliar with the software architecture of the project. To be truly user-friendly, the robot must be run without having to remove any panels or attach any additional devices. This means that scripts must be set to run upon the startup of the Pi or at very least can be triggered to run from the web dashboard instead of from accessing the Pi itself. The user should have full control over the robot via the web dashboard, so the user can toggle between autonomous/manual modes, trigger actions, and start or end scripts if needed.

5 Conclusions

Overall, the BruinBot project was a massive interdisciplinary undertaking for X1 Robotics and a pivotal moment in the development of the group. All members who contributed to the project over its 3-year lifespan developed critical engineering and decision-making skills, as well as gaining teamwork and leadership experience. The project spanned a huge range of skills and components: computer aided design, manufacturing, machining, engineering calculations, electrical calculations, wiring, coding, computer vision and AI development, user experience design, aesthetic design, and documentation. It was the most ambitious project attempted by this group by far, and taught a host of lessons about the realities of bringing such a complex robot to life. These lessons were especially valuable in the wake of the COVID-19 pandemic which completely removed the hands-on aspect of our engineering education for over a year of our lives. It was an immensely rewarding process to return to hands-on projects, especially for BruinBot, which had been placed on hold right before its manufacturing phase could begin. In conclusion, BruinBot has taught us a lot about our own capabilities as engineers and provided us with the victorious return to hands-on engineering that we needed.