ISS MicroMimic XRP

Educational ISS Model

Final Documentation

Fall 2025

Table of Contents

ISS Micro Mimic XRP Overview

Background

The ISS MicroMimic was originally developed by Boeing? We made a smaller version….

Purpose

The purpose of the ISS MicroMimic is…

Semester Objectives

The primary goal for Fall 2025 was to prepare a working prototype of ISS MicroMimic that would be able to work with telemetry…

The mechanical team focused on…

The electrical team focused on integration between the mechanical aspects of the ISS MicroMimic and the web GUI, ensuring smooth operation of the solar arrays and mapping ISS position to the AgXRP-inspired gantry system. The electrical team also formulated ideas about sensor and physical UI integrations that would allow for an elevated hands-on experience for space education in the classroom.

The computer science team…

Project Overview

Purpose

What is the project trying to achieve?

Semester Objectives

What is the project trying to achieve this semester?

Subsystems

Explain what the subsystems are for the project, including nicknames used to refer to them. E.g. nobody outside of Cup would know what we mean when we refer to a MechE “basekit”

Mechanical

ISS/MicroMimic Model

ISS Solar Panels & Servo Horn Assembly

The goal of this part of the project was to modify the existing model of the ISS’s solar panels to improve the motion of the solar panels using only 3D printed pieces. To allow for this, a servo horn assembly was designed to connect the servo motors and solar panels for movement. The movement of the solar panels play a crucial part in the recognizability and function of the ISS, and are a key feature of the potential curriculum and learning applications of the MicroMimic model.

Issues with previous design

The original design of the ISS did not allow for the motion of the solar panels using exclusively 3D printed pieces. A servo horn assembly had to be designed to attach onto the horn of the servo motors attached to the truss to connect them to the solar panels and allow for their motion.

Final Design Overview

Figure 1: Final design for servo horn assembly

The top of the servo horn assembly connects onto a standard XRP servo horn. The piece is designed to print such that the tightness of the attachment will hold the servo horn in place. To remove the servo horn from the attachment, there is a hole in the back of the attachment where a pencil may be inserted to push out the servo horn.

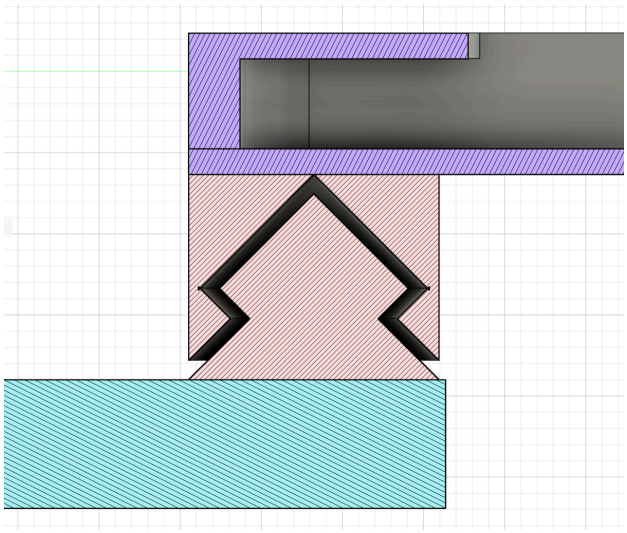


Figure 2: Cross-section of servo horn assembly, with cross-section of joint highlighted in red

This attachment has a joint at either end which connects it to two arms below, with one arm longer than the other. The joint connecting them is a cylindrical, print-in-place joint with cross-section shown above. The joint is very smooth, and encounters minimal friction. By having the joint print in place, there is no risk of the assembly coming apart at either joint. Because of the joint, each of the two arms is then able to spin freely. Each of the arms connects to a solar

panel, allowing the torque from the servo motors to be transmitted to two solar panels, moving in parallel.

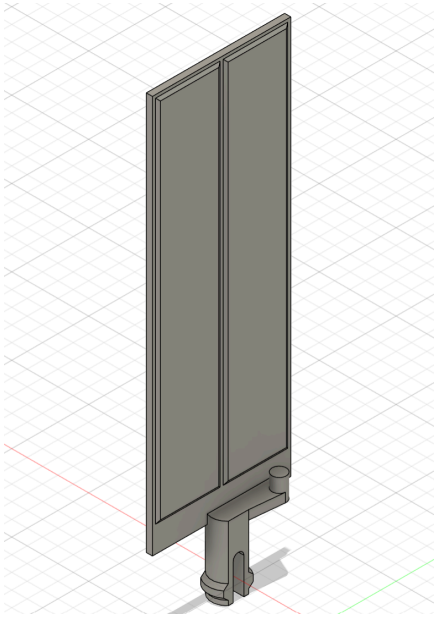


Figure 3: Final design for solar panel

The solar panels were adjusted only slightly. The prongs of the attachment between the solar panels and the truss were increased in length to match the new height of the servo horn assembly's arms. A cylinder pointing out from the base of the solar panel was also added to connect to the arms of the new servo horn assembly. Lastly, decorative elements were placed onto the solar panels to indicate the presence of actual solar panels.

Previous Iterations

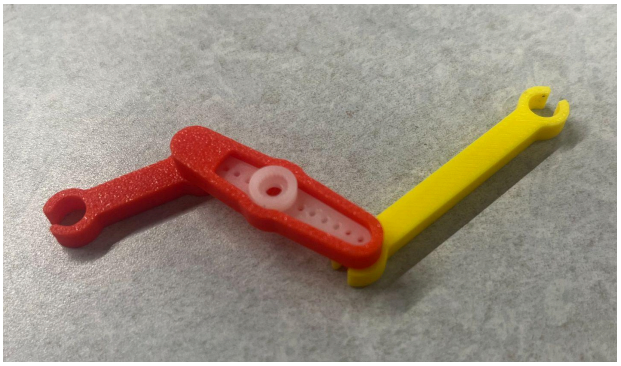


Figure 4: First iteration of servo horn assembly

An initial version of the servo horn assembly was made as three separate pieces, with a short arm, a long arm, and a horn attachment that could connect to form the full attachment. While this made for some smooth joints, it was found that depending on the specifications of the 3D printer on which this assembly was printed, sometimes the pieces would attach too loosely, or not all. Even when at best conditions, the pieces of the servo horn assembly would still tend to separate from each other when the servo motor would turn. Lastly, this design seemed to be overcomplicating what could ideally be accomplished with one piece.

Figure 5: Second iteration of servo horn assembly

A second iteration was made in which the arms of the assembly were printed directly onto the horn attachment as a closed joint. This was useful because it utilized print-in-place techniques to have the whole assembly print as one, and by having the joint closed at the bottom the pieces were no longer able to fall apart from each other. However, the piece encountered significant issues in friction due in part to the low print quality of this design and ultimately had to be redesigned to make the joint connection smoother and print more consistently.

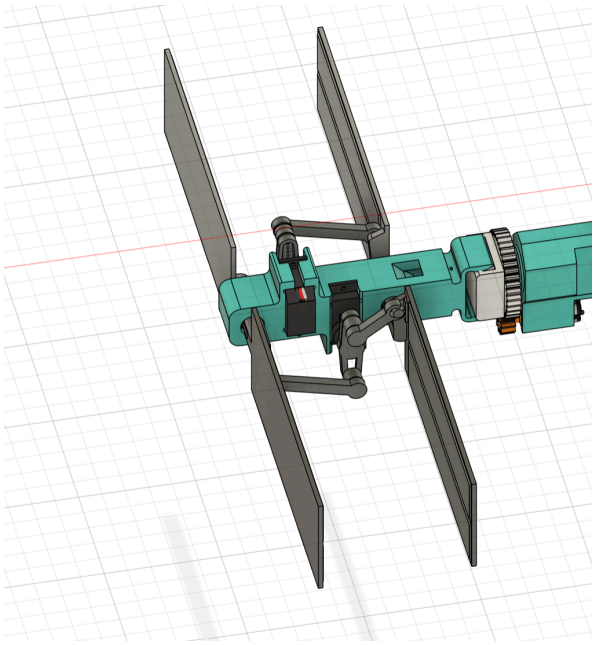


Figure 6: Full assembly

Fabrication Plan

Designs were made using Fusion 360, then printed via Bambu Studio onto Bambu printers.

Fabrications

Fabrication took some work to refine, especially with the exact dimensions of the print-in-place joint on the servo horn assembly, but the final design prints well and functions as intended. The servo horn assembly prints with a cylindrical custom support holding up the horn attachment, which may be removed after printing. The solar panel is printed laying down, with the normal (auto) support option selected in Bambu Studio while printing. For a full ISS, eight wings and four servo horn assemblies should be printed. Once printed, each servo horn assembly is connected to two solar panels by sliding the end of each arm into the cylinder near the bottom of each solar panel. The servo horn assembly is then attached to the horn of a servo motor on the outer truss, and the prongs on the bottom of the solar panels are then popped into the holes on the same side of the outer truss as the selected servo motor.

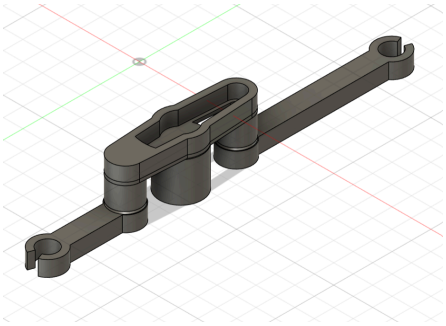


Figure 7: Servo horn assembly with custom support

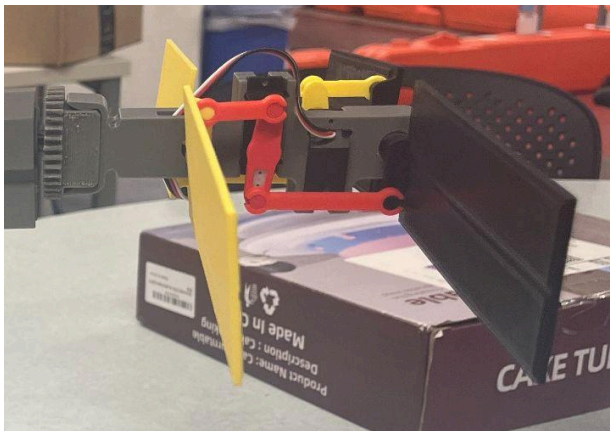


Figure 8: Full printed assembly, just using a previous iteration servo horn assembly

Next Steps

Future work on the movement of the ISS' solar panels will need to focus on the connection between the solar panels and the servo horn assembly. While the joints within the servo horn assembly itself have been made very smooth, there is still some friction found in the joint between the servo horn assembly and the solar panels themselves. Additionally, with recent requests to make the model appear even more realistic, aesthetic changes may need to be made to the solar panels to more accurately capture the dimensions and details of the solar panels on the actual ISS.

ISS Outer Truss & Interior Wiring

The outer truss of the ISS is what connects the main XRP board to the solar panels and holds the servo motors that power the movement of the solar panels. This piece serves a critical role in holding together the outer edges of the ISS and transferring communication between the board and the servo motors. The truss must also support the weight of the solar panels and servo motors while attaching only horizontally to the central truss.

Issues with previous design

The original design for the outer truss of the ISS did not account for the management of the wires connecting the two servo motors on either side of the ISS to the XRP board in the center. Without some form of wire management and with the rotation of the outer edges themselves, the

wires would get tangled up and obstruct the function of the servo motors and servo horn assembly.

Final Design Overview

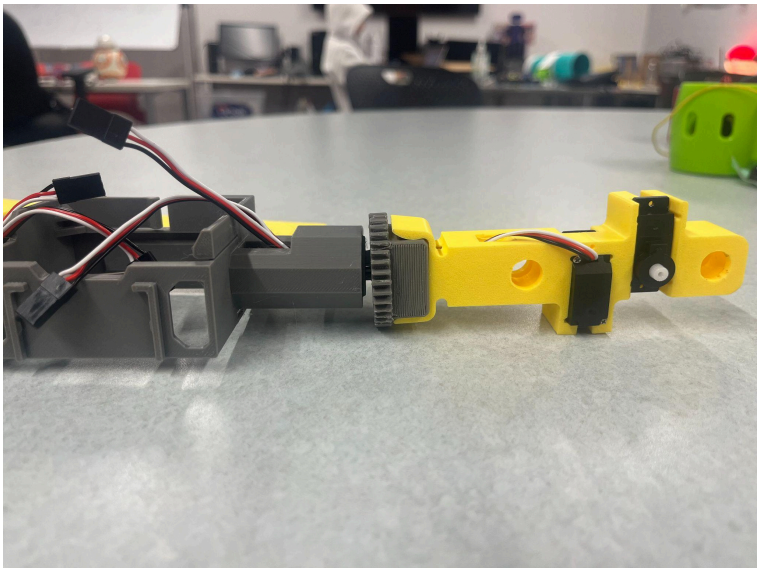
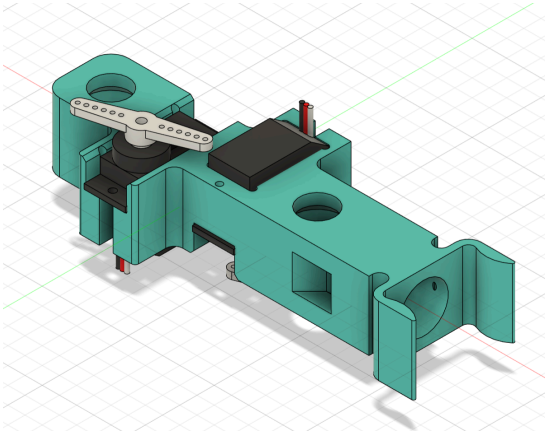
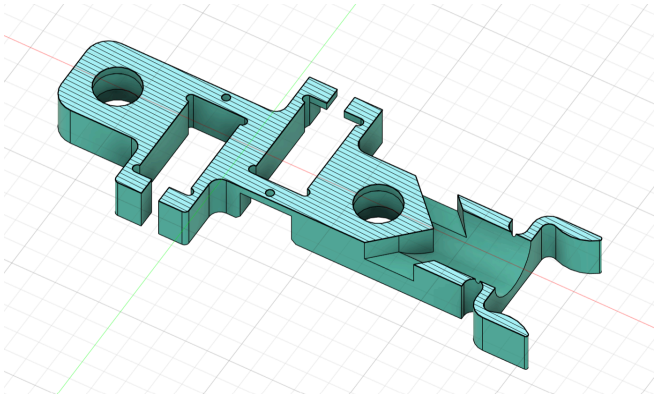


Figure 9: Fully printed final assembly

To solve the problem of wire management, the outer truss and all pieces between it and the XRP board were redesigned to support interior wiring through each of the pieces' centers. This way, no matter how the outer truss spins the wires remain in place. The wires from the servo motors enter the outer truss, go through the outer truss into a gear, through the gear into the core, and come out of the core near the XRP board.





Figures 10, 11: Final design of outer truss (left), cross-section view of final design of outer truss (right)

The outer truss had a tunnel added that connects each side of the truss near the start of a servo motor's wire to the center of the piece horizontally. The wires from each of the servo motors go through the tunnel and meet up, with the tunnel then leading towards the gear. The tunnel is very wide to make feeding the wire through as simple as possible.

The outer truss connects to a gear, which was adjusted to have an opening through its center as well. The tunnel in the gear is necessarily smaller, because of the placement of the prongs to connect the gear to the core. However, the tunnel is still comfortably large enough for the two wires to fit and for the prongs to bend enough for the piece to connect to the core. The wires coming from the outer truss are fed through the hole in the center of the gear, through to the core.

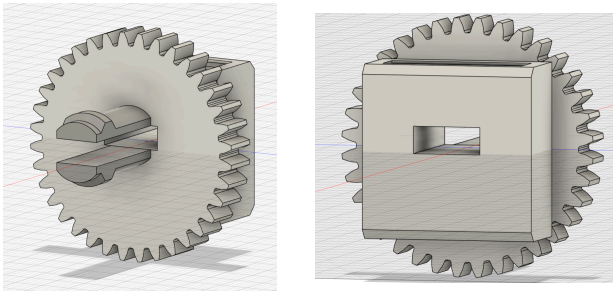


Figure 12, 13: Final design for gear, supporting interior wiring

The core had a tunnel added on each side, leading from the hole in which the gear's prongs connect to an opening on the front of the gear near the XRP board. The hole in the core is again larger to make feeding the wire through as simple as possible.

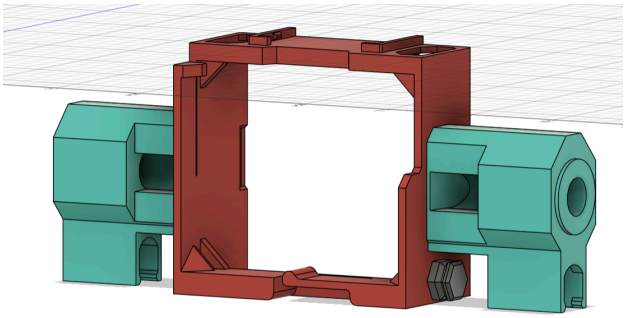


Figure 14: Final design for core, supporting interior wiring

Thus, the wire goes from the servo motor, through the outer truss, through the gear, into the core to reemerge near the XRP board.

Previous Iterations

Before it became clear that interior wiring would be needed to properly manage the wires of the servo motors, an option for exterior wiring was designed in which a handle was added to each side of the outer truss through which the wires could be fed. This way, the wires would be held tightly to the edge of the outer truss and not interfere with the servo horn assembly and movement of the solar panels. Unfortunately, the wires being present were still an eyesore, and due to changing design requirements, an interior wiring option had to be made to hide the wires completely.

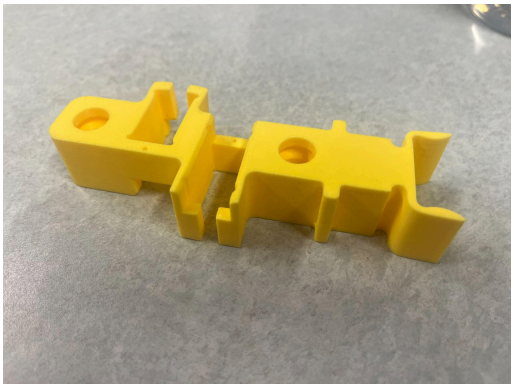


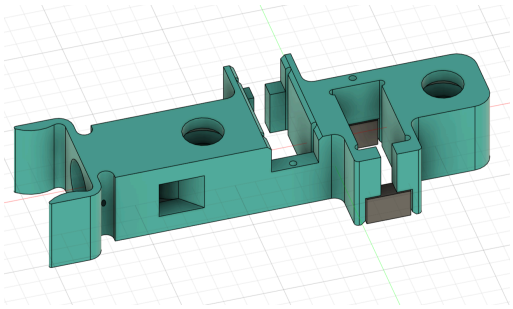
Figure 15: Previous iteration of outer truss with handles for exterior wiring

Fabrication Plan

Designs were made using Fusion 360, then printed via Bambu Studio onto Bambu printers.

Fabrications

Fabrication was pretty straightforward for this project, with the only hiccups being figuring out support specifications. The outer truss piece is printed on its side, with two rectangular prisms acting as custom supports which can be easily removed after printing. The gear piece is printed with the flat surface of the prongs deliberately parallel to the printer bed. This piece should be printed using the tree (manual) support option in Bambu Studio, with care taken to ensure no supports are printed inside the wiring tunnel. The core should be printed as described in its section below, with care taken to avoid any supports from printing inside the wiring tunnel.



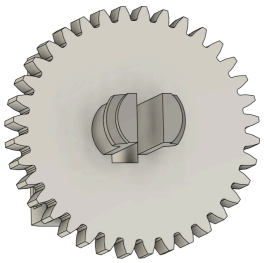


Figure 16, 17: Final outer truss design in correct printing orientation with custom supports in grey (left), Final gear design in correct printing orientation (right)

Next Steps

Future work on the outer truss and associated assembly will need to focus on print quality. Specifically, the designing of custom supports for the gear and core would improve the print quality and reliability of those pieces. Other than that, some changes to the outer truss and associated assembly could still be made to make it appear more distinctively like the ISS, or as has been recently suggested, to design a shell that snaps around the outer assemblies that more closely matches the aesthetics and design of the actual ISS.

ISS Mount

As the ISS is a stationary model, an ISS mount was necessary for securing the ISS onto an XRP for mobility as well as integration with the standard XRP chassis. This mount has 6 features:

1. Modular (for ease of assembly)
2. Attaches onto the standard XRP chassis without needing to modify the chassis
3. Fully 3D printed
4. Includes one degree of freedom to rotate the ISS
5. Center of mass is at the center of the XRP chassis when mounted
6. Includes a center piece to elevate the ISS so that the solar panels do not interfere with the XRP chassis as the ISS moves.

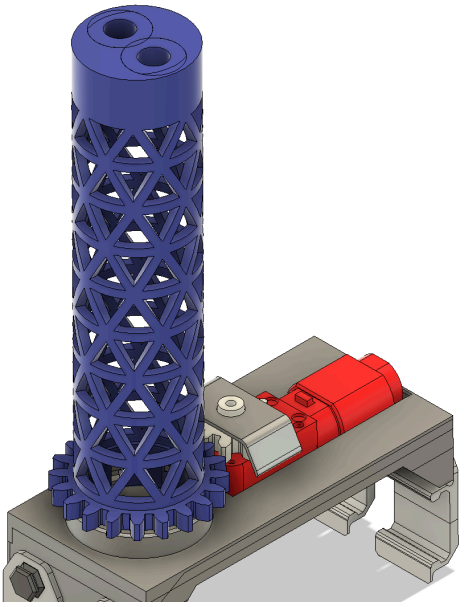


Figure X: Overview of ISS mount assembly

Design Overview

The base of the mount has a rectangular platform laying over an arch, which was designed to leave room for the XRP board and wires that will be directly underneath the mount. The 45 degree slant of the arch reduces the need for supports to print the arch and rectangular profile. Along with the base is a circular snap fit interface for the driven gear and overhang fastening mechanism that holds the driving gear and motor in place and ensures the gear stays rotating in one axis. Furthermore, the two sides of the base each have two hex bolts, where the XRP rail clips can be secured and used to clip the mount onto an XRP chassis. The red motor also tightly presses into the base, with an opening at the bottom for wires.

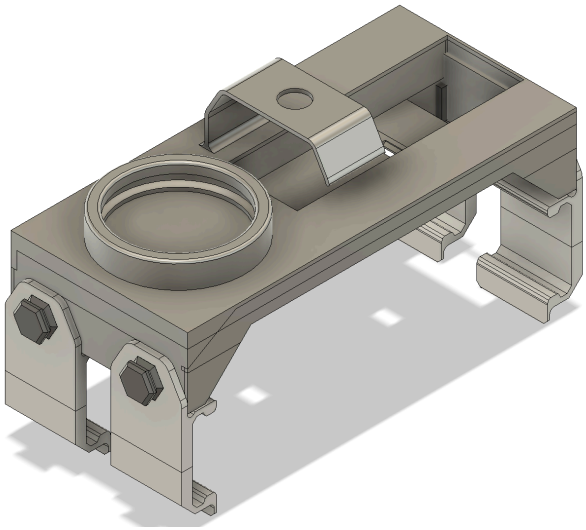


Figure X: ISS mount base with rail clips

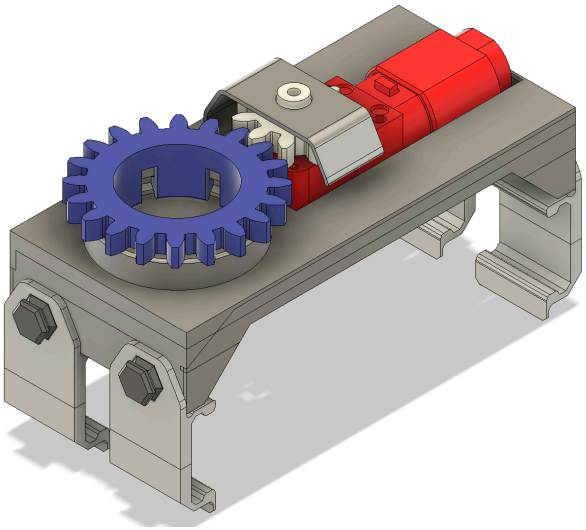


Figure X: ISS mount base with rail clips, motor, and gear train

As can be seen in the image on the right, a 1:2 gear train secures onto the base without any bolts. The driven gear (blue) is attached onto the base and enabled to rotate in one axis via snap fit (refer to the cross section below). The snap fit design was chosen to enable easy attachment, removal, and replacement of any part of the ISS structure.

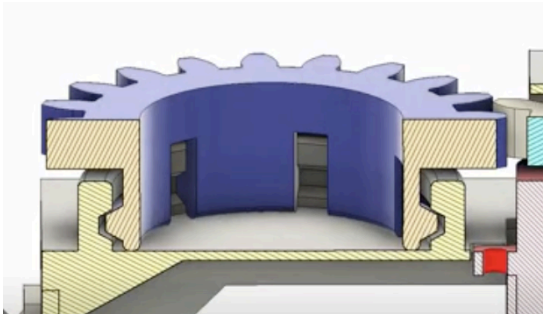


Figure X: Cross section of the driven gear

The elevation pillar has a hollow, truss-like exterior design. The triangular and hexagonal design is intended to increase the pillar's resistance to stress in multiple directions, optimize for 3D printing, and enhance aesthetics. There are two holes with snap fit interfaces at the top of the pillar for a tightly yet modular attachment to the ISS model's middle piece.

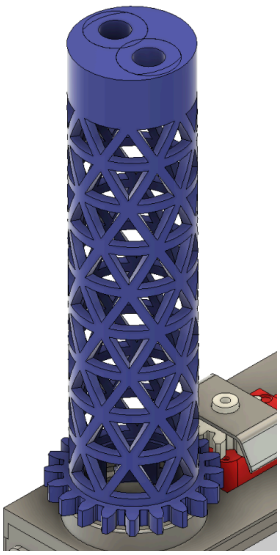


Figure X: Elevation pillar

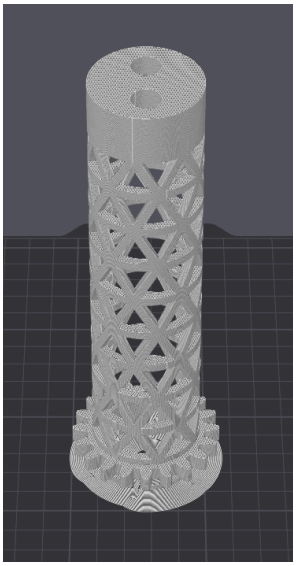


Figure X: Elevation pillar in the Bambu Studio slicer with manual supports at the gear teeth only.

Fabrication Plan

Initial designs were sketched by hand, then modeled and assembled in Fusion 360, and finally printed on Bambu Lab 3D printers.

Fabrications

The base can be printed with minimal tree supports (only at the motor-fastener's overhang). The driven gear and pillar structure was printed as one piece, which required manual tree supports to reduce the filament used, and it was printed with the pillar vertically standing and the snap fit teeth facing down. It was printed in this orientation because printing the supports are much easier to remove on this end than printing with the snap fit holes facing down. Similarly, printing the small driving gear also required a bit of support at the teeth.

Mount Wire Clip

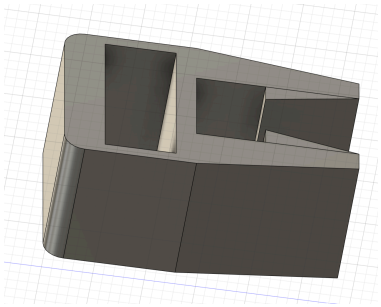


Figure : Final design for wire clip

To support the management of any wires that may be run up and down the mount, a clip that is able to attach to the lattice structure of the mount was designed. The clip easily attaches and detaches onto the lattice structure but stays comfortably on during normal operation. Any wires that may need to run up and down the mount may then be fed through the rectangular hole on the left side of the clip as shown above. The hole is designed to be large enough to comfortably fit thin wires as included in the XRP kit while being small enough to hold the wires close to the mount. It is recommended to print and use two or three, spread out across the height of the mount for best results. This was designed in Fusion 360 and printed in Bambu Studio on Bambu printers. Future work along these lines would include making a clip for larger or different types of wires that may need to run up and down the mount, or considering adapting the mount's design to support interior wiring through the center of the mount itself.

Docking

The purpose of this subsystem is to enable one XRP robot to physically "dock" with another XRP robot in a controlled and repeatable way. The subsystem is designed as an engaging, hands-on challenge for learners, particularly when operating the XRP remotely. By requiring precise alignment and control, the docking mechanism encourages users to develop better driving accuracy, spatial awareness, and understanding of sensor-based interaction. The primary objective for the semester was to design, iterate, and fabricate a functional docking subsystem that is robust, easy to integrate with the XRP platform, and suitable for educational use.

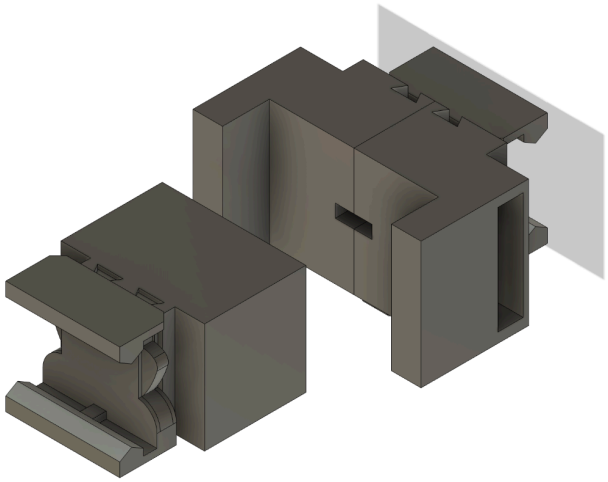


Figure X: Docking Mechanism

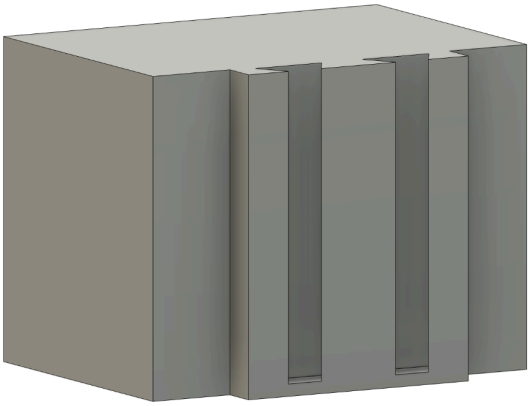


Figure X: Sliding section

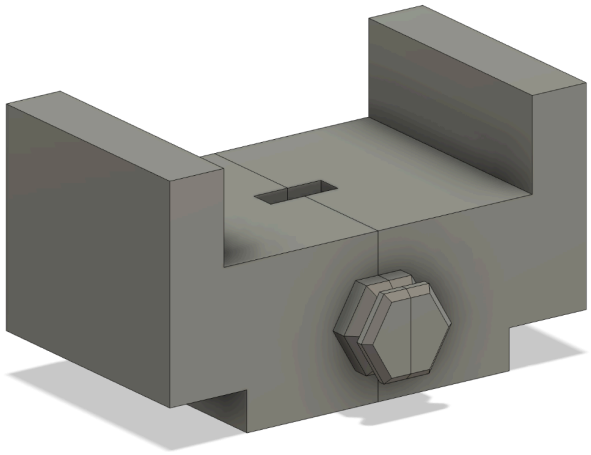


Figure X: Receiving component assembled from two parts

Design Overview

The final design consists of two complementary components, one mounted on each XRP robot. One XRP carries a protruding “in” piece, while the other XRP carries a receiving component that the protruding piece docks into. The receiving component houses a limit switch, which senses when the incoming piece has been fully inserted, providing clear confirmation that docking has occurred. This receiving component is composed of two separable parts that, with a snap fit assembly, securely mount and retain the limit switch. The subsystem mounts to the XRP using a backpack-style clip that attaches directly to the XRP rail, ensuring easy installation and compatibility with the existing platform. A sliding mechanism connects the backpack clip to the rest of the docking assembly, allowing for improved alignment and smoother engagement between the two XRP units. This two-part design promotes reliable docking while maintaining simplicity and durability for repeated educational use. Additionally, April Tag mounts were integrated into MicroMimic's core design for the docking system, utilizing a press-fit mechanism that enables quick swapping between different April Tag configurations.

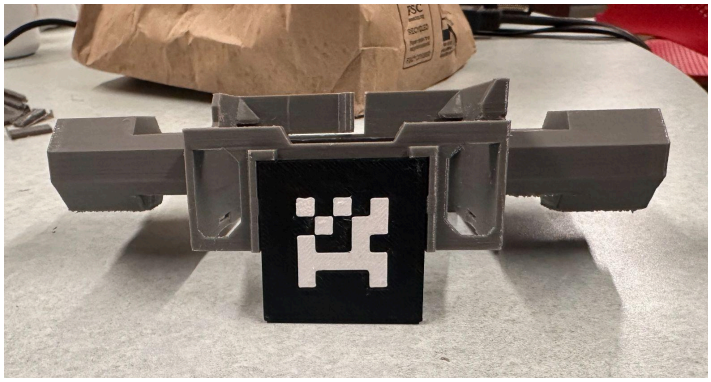


Figure X: April Tag mounted on core of MicroMimic

Previous Iterations

Earlier design iterations were significantly smaller and proved difficult to fabricate reliably. The limit switch used was approximately 2 mm in size, which made it challenging to design a functional and secure mounting feature within the receiving component. The reduced scale of the overall part led to print failures, fragile components, and difficult assembly. Additionally, aesthetic concerns arose, as the backpack clips had to maintain a minimum size to properly fit onto the XRP rail, resulting in an unbalanced appearance. These limitations informed later design decisions, leading to a revised approach for stabilizing the limit switch through the use of a snap-fit feature. In future iterations, the design could be further optimized to accommodate a wider range of printers, or a larger limit switch could be used to allow for a more consistent and reliable printable mounting system

Fabrication Plan

The components were modeled in Autodesk Fusion 360 and fabricated using Bambu Lab 3D printers. All parts were 3D printed with the exception of the limit switch, which was integrated during assembly.

Fabrication Results

Overall, fabrication went well, and the piece functioned as intended. It can print with minimal supports in six total pieces. There is room for improvement, however, in both the overall design efficiency and filament usage. Future iterations could reduce material consumption and further refine the geometry to improve strength-to-weight ratio and print time, as well as account for a larger limit switch.

Core

The purpose of this subsystem is to serve as a structural component that securely holds essential hardware while connecting the outer solar panels to the central rotational axis. This part acts as the mechanical interface between the solar array and the axis, ensuring proper alignment, stability, and load transfer during operation. Over the course of the semester, the primary objective was to redesign this subsystem to improve functionality, integration, and manufacturability while maintaining compatibility with existing project components.

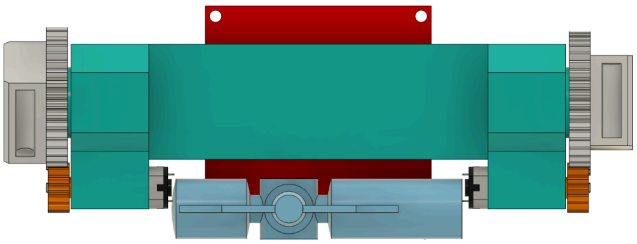


Figure X: Previous Design

Issues with the Previous Design

The previous design had several limitations that motivated a redesign. Most notably, there was no dedicated space to house the battery pack or the XRP board, which made system integration difficult and cluttered. Additionally, the design did not provide a clear or reliable method for connecting the subsystem to the axis, limiting structural robustness and ease of assembly. These shortcomings reduced both the usability and scalability of the system.

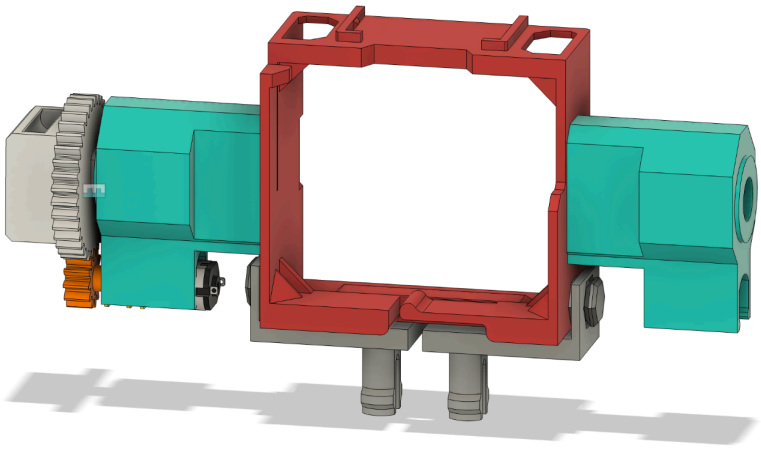


Figure X: New Design

New Design Overview

The new design addresses the shortcomings of the previous version while prioritizing compatibility and simplicity. It uses the same parts and interfaces as earlier projects to promote connectivity and modularity. Specifically, the connection points between the core and the axis are identical to the connection interfaces used between the core and the solar arrays, ensuring consistency across the system and simplifying assembly. Additionally, the portion of the core that houses the XRP board and battery pack is reused from other XRP projects, maintaining continuity and reducing the need for redesign. The geometry was refined to reduce the need for additional support material during fabrication, resulting in a cleaner and more efficient structure. The updated design also consolidates the two previously separate components used to mount the XRP board and battery pack into a single integrated mounting feature.

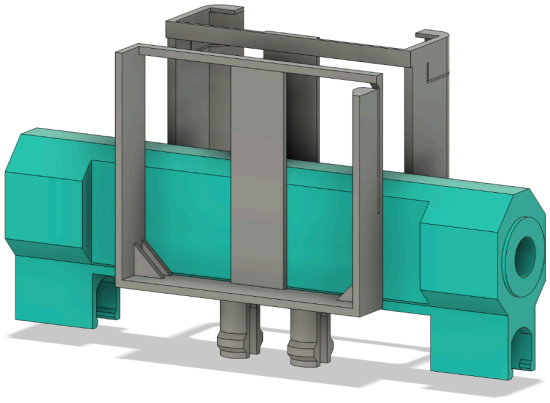


Figure X: One previous iteration

Previous Iterations

Several earlier iterations were explored before arriving at the final design. These initial versions generally required significantly more structural support and were less visually clean and compact. One of the earliest designs included two separate components to hold the XRP board and the battery pack, which introduced additional complexity, required extensive support material, and made printing more difficult and less reliable. Additionally, the features used to attach the core of the subsystem to the axis were designed to print in an unfavorable orientation, which compromised the intended compliance of the design and reduced overall performance. Through iterative refinement, unnecessary material and complexity were eliminated, print orientations were improved, and the design evolved into a more streamlined, functional, and effective final solution.

Fabrication Plan

The component was 3D modeled on Fusion 360, then printed entirely on Bambu Lab Printers.

Fabrication Results

Fabrication was largely successful, and the part printed as intended. The core is printed with supports in one piece with the two gears on each side, and the connection points for the axis printing with minimal supports. However, there is room for improvement, particularly in the design of custom supports and overall ease of printing. Future iterations could further optimize print orientation and support structures to improve surface quality and reduce post-processing effort.

Electrical

The Electrical team is responsible for the development of systems that connect realtime ISS data to movements on the ISS mimic model, both virtually and physically. In addition, the team explored various additions that can be integrated with the system and add to the user experience, such as the Ham Radio.

Telemetry Data Integration

While CS was mostly responsible for the backend involving the connection between the Lightstream telemetry data server and the GUI, ECE was responsible for connecting the servos for the individual solar arrays and N20 motors for the alpha rotational assembly.

Servo Integration

Purpose

Map the beta gimbal assembly (BGA) and Solar Array Rotation Joint (SARJ) angles to the movement of individual solar arrays and the yaw of the solar array groupings, respectively.

Implementation

Using an adapted version of the PestoLink protocol, the GUI translates telemetry data into PestoLink packets, sending a specific motor angle to the backend and triggering a PWM movement signal for the servos mapped to BGA and feedback-controlled motor movement using N20 motors for the SARJ angles.

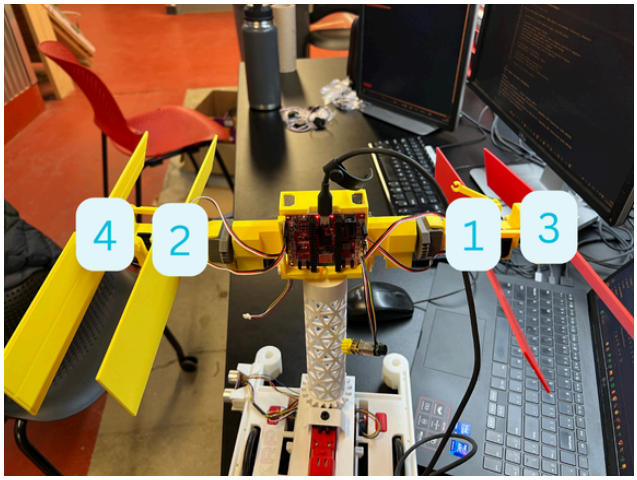


Figure X. BGA Servo Wiring Assembly

Ham Radio

Amateur radio, also known as Ham radio, consists of radio frequencies used for recreation and experimentation. Astronauts on the ISS utilize ham radio frequencies to communicate with people on earth, which would allow for a cool experience in connecting students with real people aboard the ISS. While a physical integration option using an FPGA was explored, online methods via the world wide web seems to be a more viable option.

SDR on FPGA

Purpose

This project aims to build a compact and simple Software Defined Radio (SDR) signal chain on an FPGA to demonstrate and learn how modern RF systems move radio functionality into reconfigurable digital processing. The goal is to digitize an input signal, translate it to a baseband using a numerically controlled oscillator (NCO) and digital mixing, isolate the desired channel through decimation and filtering via CIC/FIR and the perform simple demodulation (such as FM and AM) to recover usable audio or data. Implementing these blocks on a FPGA highlights the key advantages of hardware acceleration for SDR such as high throughput parallel processing, deterministic timing.

Design Overview

Although this work was not implemented this semester, this is a good candidate project for a future semester where the project will not be tied down to XRP. Software Defined Radio (SDR) is the industry standard for RF communications as it provides more control over the physical

layer in software (i.e. modulation, bandwidth, channel selection, filtering, synchronization, and even protocol framing) rather than locking these choices into fixed RF hardware. In a typical SDR, a wideband RF front end (tuner, mixers, ADC/DAC) brings a slice of spectrum into baseband, and the rest of the signal chain runs digitally: numerically controlled oscillators (NCOs), mixers, decimating/interpolating filters (CIC/FIR), demodulators, and packet/PHY logic.

Currently in the Lab we have the Basys3 artix7 FPGA with the following specifications:

- 33,280 logic cells in 5200 slices (each slice contains four 6-input LUTs and 8 flip-flops)
- 1,800 Kbits of fast block RAM
- Five clock management tiles, each with a phase-locked loop (PLL)
- 90 DSP slices
- Internal clock speeds exceeding 450 MHz
- On-chip analog-to-digital converter (XADC)
- Digilent USB-JTAG port for FPGA programming and communication
- Serial Flash
- USB-UART Bridge
- 12-bit VGA output
- USB HID Host for mice, keyboards and memory sticks
- 16 user switches
- 16 user LEDs
- 5 user pushbuttons
- 4-digit 7-segment display
- 4 Pmod ports: 3 Standard 12-pin Pmod ports, 1 dual purpose XADC signal / standard Pmod port

Utilizing this FPGA board (or possibly purchasing a better FPGA), we could potentially utilize this FPGA in order to accelerate very specific and deterministic tasks as a part of a variety of robotic projects outside of just SDR. A good reference for inspiration of guidance is the final project of Professor Hunter Adams ECE 5760: Advanced microcontrollers class (or as it is now called, Hardware Acceleration via FPGA).

Next Steps

The next steps are to first pick a simple goal to achieve and define a frequency plan, sample rate, bandwidth, and expected output so the requirements are clear. The Basys3 FPGA has a XADC but is not designed for high speed sampling that is necessary for true RF sampling. What can be done instead is to skip the analog to digital conversion of RF signals and work with a predefined raw digital encoding of a RF signal. The FPGA implementation should proceed incrementally: NCO, complex mixer, CIC decimator, FIR/channel filter, demodulator, and finally an output path such as UART streaming, PWM audio, or an external codec. This modular approach allows us to test each stage reliably. Finally, the design should be evaluated by comparing outputs to a golden model and reporting resource usage, achievable clock rate, throughput, and end-to-end latency, culminating in a demo that can tune/select a channel and produce intelligible audio/data.

CS

Graphical User Interface

3D "Digital Twin" Visualization

The Graphical User Interface includes a "Digital Twin" visualization built using React Three Fiber, which renders a real-time 3D model of the ISS directly in the browser. This module synchronizes the 3D mesh properties with live telemetry data, mapping specific data streams—such as the Solar Alpha Rotary Joint (SARJ) and Beta Gimbal Assembly (BGA) angles—to the rotation of the digital solar array components. To ensure performance during high-frequency updates, the application utilizes a global state management system via React Context, allowing the 3D scene to update smoothly at 60 FPS without triggering unnecessary re-renders of the surrounding UI elements.

Web Bluetooth Hardware Interface

To enable direct control of the XRP robot without requiring intermediate backend servers or native desktop applications, the frontend stack integrates the Web Bluetooth API. This architecture allows the Next.js application to establish a secure, low-latency connection directly to the robot's microcontroller via the browser. A custom communication protocol was developed to serialize 6-degree-of-freedom angle data and button states into compact byte arrays. These

packets are transmitted to the robot's UART service, effectively functioning as a localized ground control station that can operate entirely client-side.

Live Telemetry Data Pipeline

The application's data layer is managed through a TelemetryContext provider that interfaces with the Lightstreamer client to handle the asynchronous nature of space telemetry. The system subscribes to specific telemetry IDs (e.g., S0000003 for SARJ positions) and manages the connection lifecycle. Additionally, the pipeline includes logic to monitor signal integrity by calculating the timestamp delta between the client and the incoming data packets. This allows the interface to automatically detect and visually flag "Stale Signals" or "Signal Loss" events, ensuring that the physical robot operates only on valid, recent synchronization data.

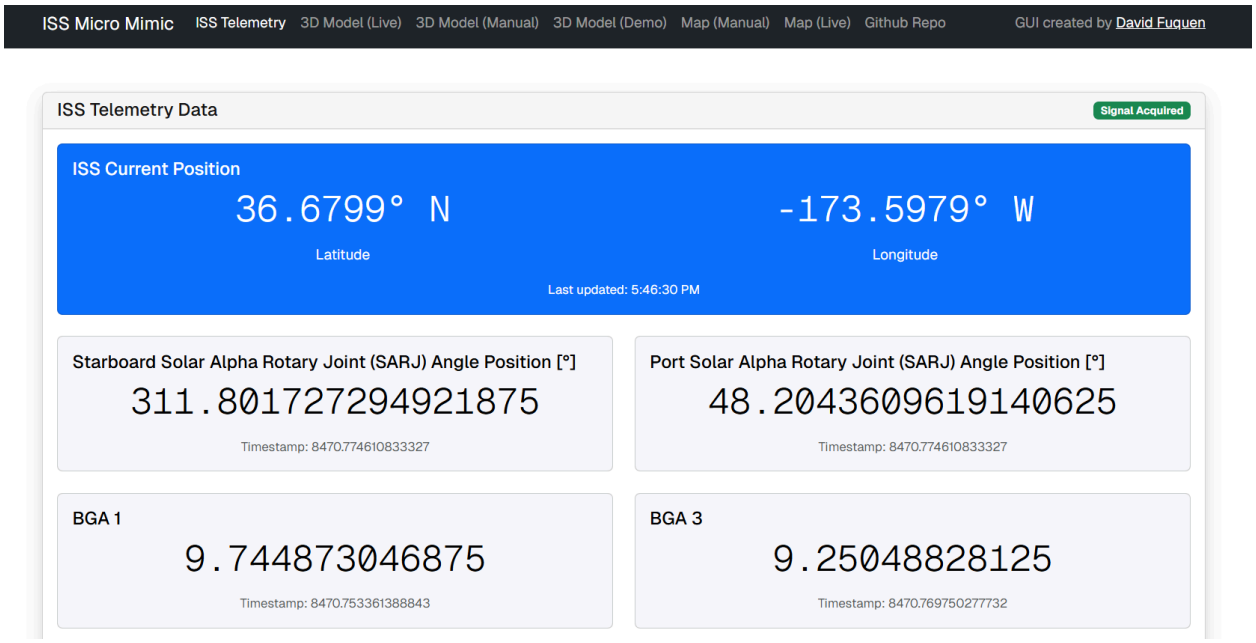


Figure 1: Dashboard displaying real-time telemetry data from the ISS

This dashboard serves as the primary software interface for monitoring incoming ISS telemetry, aggregating positional and joint-angle data into a single, real-time view. It demonstrates the end-to-end telemetry pipeline by streaming values into the UI and propagating them to downstream visualization and control components.

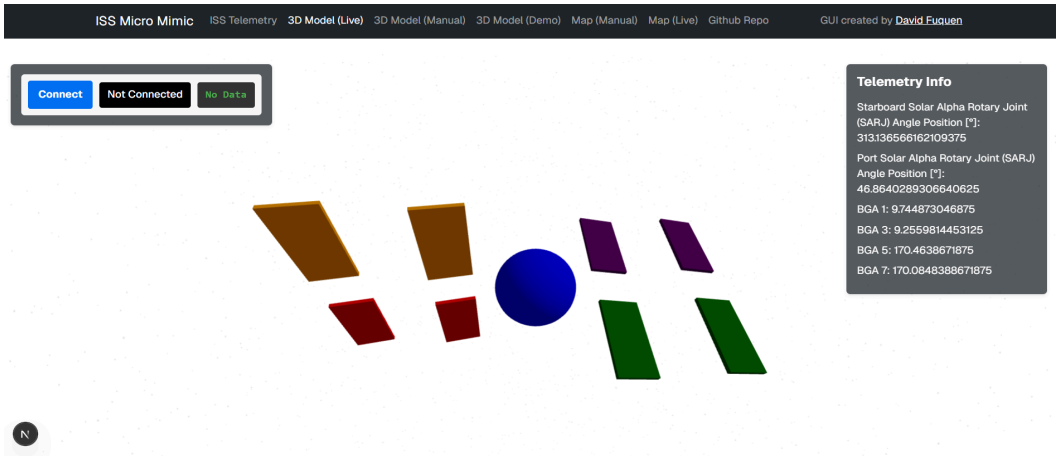


Figure 2: 3D model of the physical ISS mimic, synchronized with its current state

This figure shows the browser-based 3D "digital twin" of the ISS mimic, rendered using React Three Fiber and updated continuously from telemetry inputs. The model reflects the current physical configuration of the robot, enabling visual verification that software state, telemetry, and hardware motion remain synchronized.

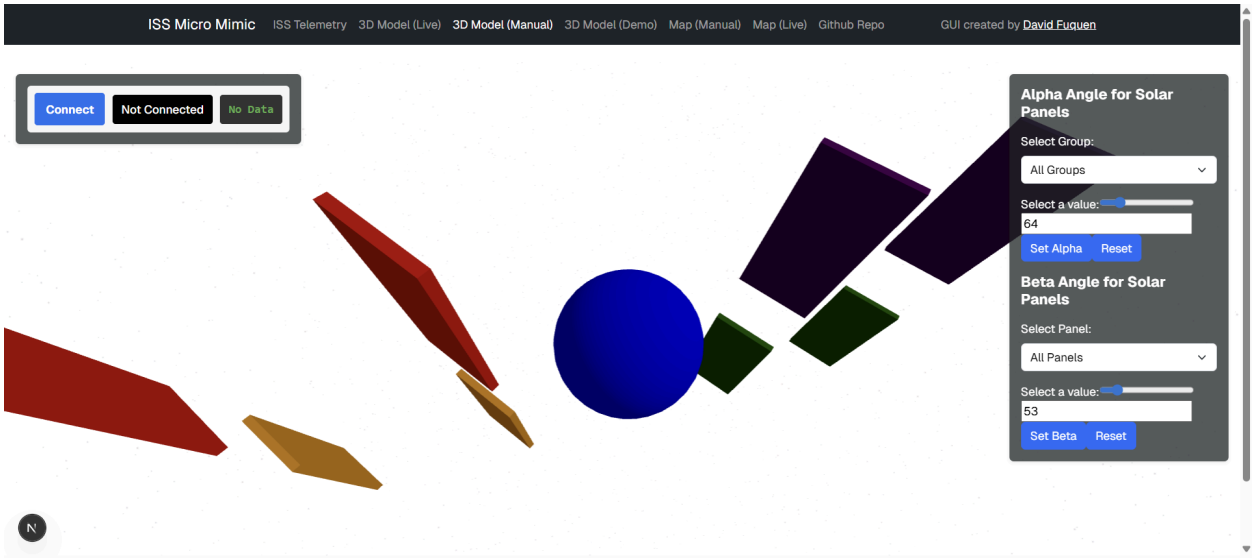


Figure 3: Manual Control Model of the ISS

The manual control interface allows operators to directly command individual or grouped joints by specifying Alpha (rotary) and Beta (gimbal) angles. User inputs are serialized into low-level command packets and transmitted over Web Bluetooth, enabling precise and deterministic hardware control without a backend server.

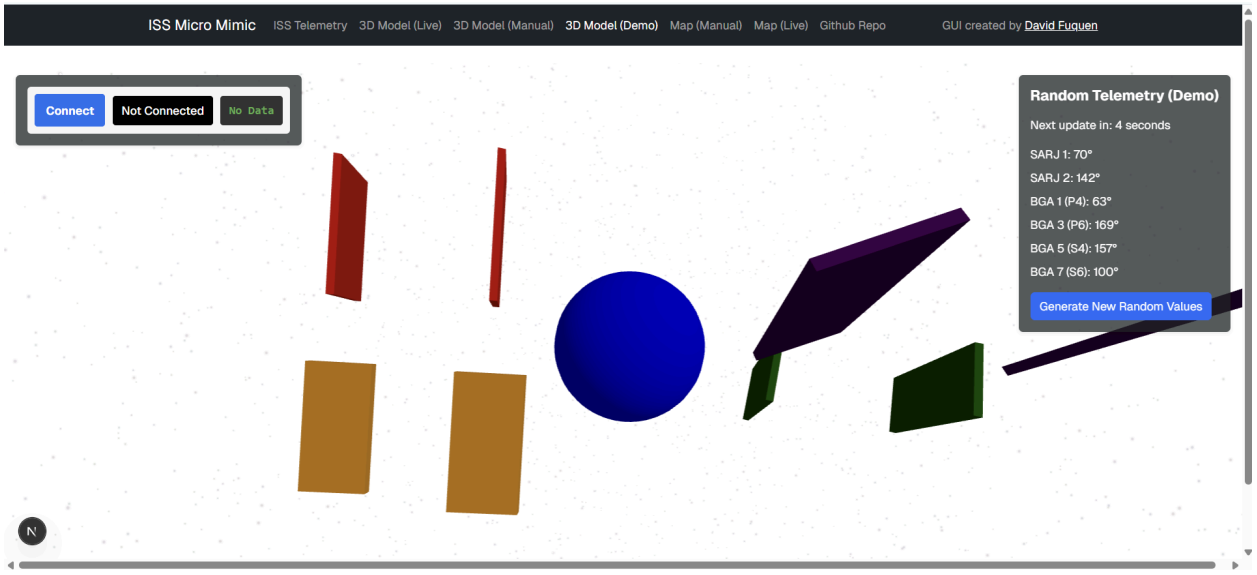


Figure 4: Model of ISS with random generated positions for wings.

This demo mode injects synthetic telemetry values into the system to simulate joint motion without requiring live hardware or external data sources. It is used to validate rendering logic, state updates, and communication stability during development and testing.

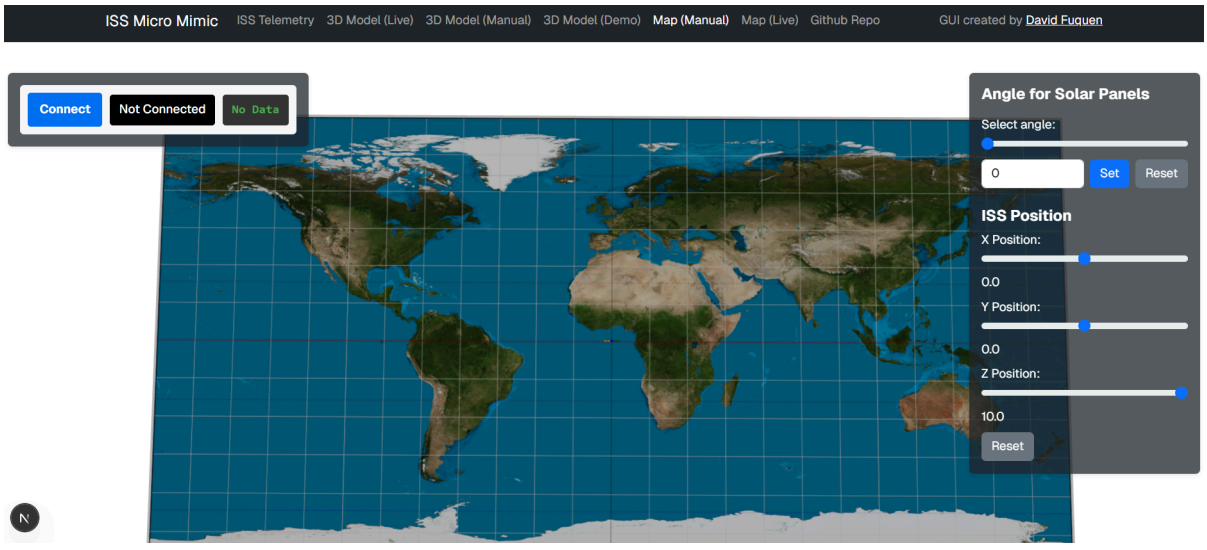


Figure 5: Planned map mode, which allows the user to visualize the position of the ISS above the Earth.

This planned visualization mode displays the ISS ground track over a global map using propagated orbital data. It provides spatial context for telemetry by linking orbital position with station orientation and subsystem states shown elsewhere in the interface.

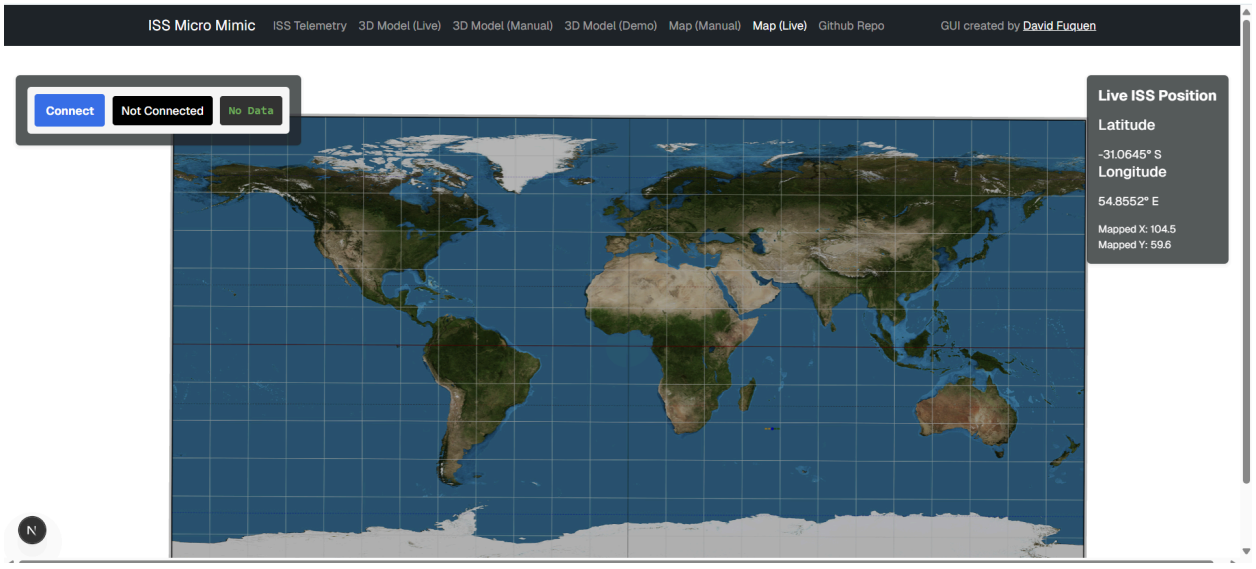


Figure 6: Live position of the ISS

This view presents the real-time geographic position of the ISS derived from live telemetry and orbital propagation. It demonstrates integration between external orbital data sources and the frontend visualization layer, completing the full software pipeline from data ingestion to user-facing insight.

Gantry System

The ISS Mimic's gantry system is the mechanism that lets the platform move in a controlled way across a 2D workspace (X-Y plane). This functionality lives in the XYMotion class, which turns motor encoder rotations into millimeters of motion, enforces workspace limits, and provides higher-lever commands like home, find_size, and move_to. At a high level, the gantry gives the ISS Mimic three core capabilities:

- **Coordinate-based motion**: You can command an (x, y) position in mm (move_to) instead of manually controlling motors.
- **Repeatable reference frame**: Homing establishes a known (0, 0) origin so movement is consistent across runs.

- **Safe bounded movement**: Once x_max and y_max are known, the system prevents motion outside the physical travel range.

Position tracking and coordinate transforms are handled by get_position(), xy_to_ab(x, y) and ab_to_xy(a, b). get_position() reads encoder positions from both motors, subtracts the stored zero_zero, converts turns to mm using turns_to_mm, and then converts motor-space distances (a, b) into Cartesian (x, y) coordinates. Actual motor execution happens in move_relative_ab, which is an asynchronous control loop. It computes target encoder positions (targetA/targetB) and then (1) repeatedly measures how far each motor has progressed, (2) computes a synchronization error using the fraction-done comparison and a proportional gain kp, (3) adjusts motor efforts in opposite directions to keep the motors in sync, and (4) slows down near the target using slow_down_distance and proximity_distance. The loop ends once the combined remaining error is small, and stop() sets both efforts to zero.

The expected workflow begins by calling home(), which uses the bang() helper to drive the gantry into its reference edges until motion stops, establishing a consistent origin by setting zero_zero and marking the system as homed. If workspace limits are not provided at initialization, find_size() is then called to determine x_max and y_max by moving to the far ends and reading the resulting position. After this calibration sequence, all motion commands pass safe_to_move() checks, remain within bounds, and execute.

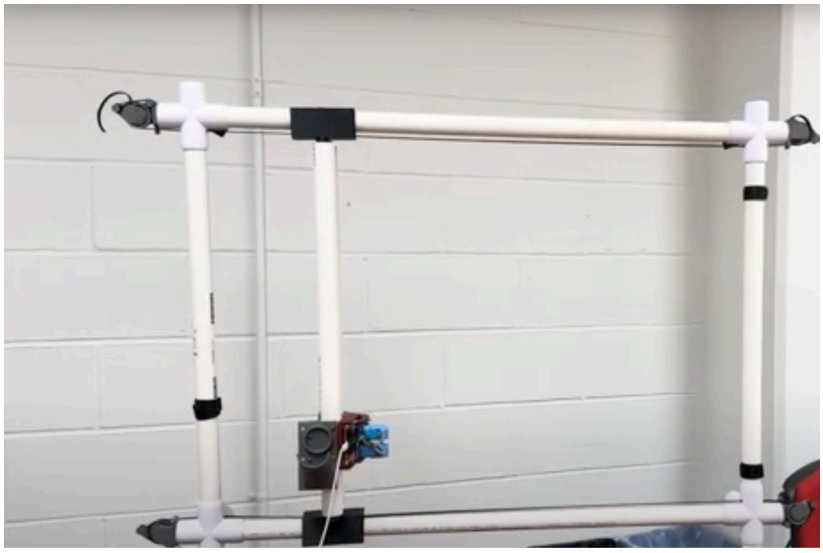


Figure X: Screenshot of the current gantry system, the XRP is connected to a laptop via USB

Telemetry Data

Solar Alpha Rotary Joint (SARJ) Angles

The ISS mini mimic collects real-time angular position data from two Solar Alpha Rotary Joints through Lightstreamer channels S0000003 (Starboard SARJ) and S0000004 (Port SARJ). These massive 10-foot diameter rotating mechanisms control the yaw motion of the ISS's solar arrays by rotating entire solar panel assemblies about a vertical axis relative to the station's body. The SARJ system allows the solar arrays to track the sun as the ISS orbits Earth every 90 minutes, maintaining optimal power generation throughout each orbit. In the physical miniature model, these angular values (ranging from 0-360 degrees) are transmitted via Bluetooth to servo motors that replicate this rotational movement, causing the grouped solar panel arrays on each side of the model to pivot in unison, mimicking the actual station's sun-tracking behavior in real-time as the data streams from NASA's telemetry servers.

Beta Gimbal Assembly (BGA) Angles

The system monitors four independent Beta Gimbal Assembly angles from Lightstreamer telemetry channels P4000007, P6000007, S4000007, and S6000007, representing the roll motion control for individual solar panel wings. Each BGA acts as a secondary rotational joint that rotates its corresponding solar panel about its own longitudinal axis, providing fine-tuned adjustment perpendicular to the SARJ's primary rotation. This two-axis articulation system (SARJ for yaw, BGA for roll) gives each solar array full hemispherical pointing capability to maintain perpendicular alignment with solar rays throughout the station's complex orbital dynamics and seasonal variations. The mini mimic receives these four angle values simultaneously and drives four separate servos, each controlling one of the model's solar panel wings, creating an accurate physical representation of the sophisticated gimbal choreography occurring 250 miles above Earth as the arrays continuously adjust to maximize solar incidence angle.

ISS Current Position Coordinates

The ISS coordinates are fetched through a Next.js API endpoint at app/api/iss-position/route.ts that first reads Two-Line Element (TLE) orbital data from utils/Iss_Tle.json, and if that data is older than 8 hours, it automatically refreshes it by fetching new TLE data from celestrak.org. Once the TLE data is available, the endpoint uses the satellite.js library to propagate the satellite orbit to the current time, converting the Earth-Centered Inertial (ECI) position to geodetic coordinates (latitude and longitude) using Greenwich Mean Sidereal Time (GMST). The contexts/IssPositionContext.tsx then wraps the application and calls this API endpoint every 10 seconds, providing real-time ISS position updates to all child components through React context, ensuring the entire application always has access to the most current ISS location data without each component needing to fetch it independently. The conversion scales the latitude and longitude values into normalized percentages (0-100) that are then transmitted in bytes 15-16 of the Bluetooth packet using protocol version 0x03. Upon reception, the XRP's backend (pestolink_adapted.py:411-418) scales these percentages to actual millimeter coordinates within the gantry's calibrated workspace (205.8mm x 103.6mm), and the dual-motor CoreXY mechanism (xy_motion.py) physically moves the model to the corresponding position. This creates a miniature orbital tracker where the model's physical location on the gantry platform represents the ISS's current position over Earth, allowing observers to see not only the solar panel orientations but also the station's ground track as it completes its 15.5 daily orbits.

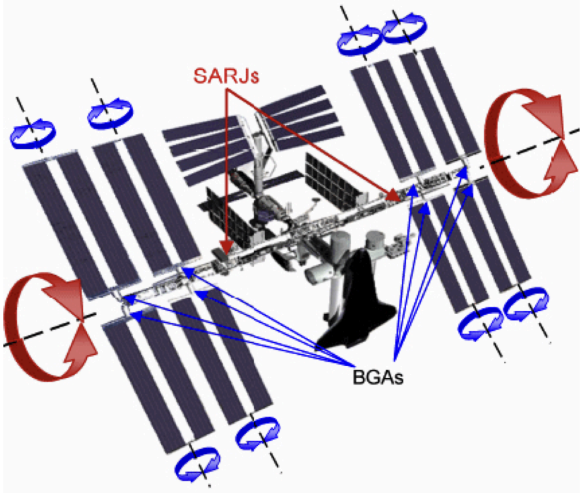


Figure X: direction of rotation of the SARJ angles and the BGA angles

Data Streaming Pipeline

The frontend application establishes a WebSocket connection to NASA's Lightstreamer server (push.lightstreamer.com) through the TelemetryContext (TelemetryContext.tsx:82). It subscribes to specific ISS telemetry channels including solar panel angles (S0000003, S0000004 for SARJ positions) and Beta Gimbal Assembly angles (P4000007, P6000007, S4000007, S6000007 for individual solar panel orientations). When Lightstreamer pushes real-time updates, the subscription listener (TelemetryContext.tsx:108) captures each data item containing a timestamp and value, then stores it in React state. This data is immediately used to render a 3D visualization of the ISS in the browser, with solar panels rotating in real-time to match the actual ISS configuration. The context also monitors signal quality by tracking timestamp deltas to determine if the data stream is stale or if signal acquisition of signal is lost.

When a user connects their XRP robot via Bluetooth, the BluetoothContext establishes a Web Bluetooth API connection using the PestoLink BLE service (BluetoothContext.tsx:76). The live view page (live_v3/page.tsx:48) monitors telemetry changes through a React useEffect hook that triggers whenever any solar panel angle updates. It packages the six angle values (four BGA angles and two SARJ angles) into a 26-byte data packet using the protocol version 0x03 format (robotPackets.ts:25), where each angle is encoded as a 16-bit little-endian integer (0-360 degrees). The packet also includes optional x-y coordinate data in bytes 15-16 that represent ISS position scaled to the gantry's physical dimensions. This packet is transmitted via BLE to the XRP robot at SERVICE_UUID "27df26c5-83f4-4964-bae0-d7b7cb0a1f54".

The XRP's MicroPython backend (main.py:82) runs a PestoLinkAgent that receives incoming BLE packets and parses them in the on_write method (pestolink_adapted.py:382). It extracts servo angles using get_angle() which decodes the 16-bit angle values from the byte array, and position coordinates using get_position() which scales the x-y values from percentage (0-100) to millimeters based on the gantry's maximum dimensions (205.8mm x 103.6mm). The async main loop (main.py:93) continuously checks for new data every 100ms and when position updates are

detected, it commands the XY_motion gantry system to move to the specified coordinates using dual-motor CoreXY kinematics with PID control for synchronized movement. Simultaneously, it updates four servo motors (servo_one through servo_four) to match the received BGA angles, with logic to handle the 0-360 degree range by mirroring angles greater than 180 degrees. This creates a physical miniature ISS model that mimics the real station's solar panel orientations and orbital position in real-time.

April Tag Detection System

The April Tag detection system is used to track multiple ISS MicroMimics that are docking. April tags are simply attached on top of the prototypes, and a camera will be displayed from above to track the April Tags. The camera will use a computer vision model to conveniently monitor the location of the prototypes, allowing for smoother coordination.

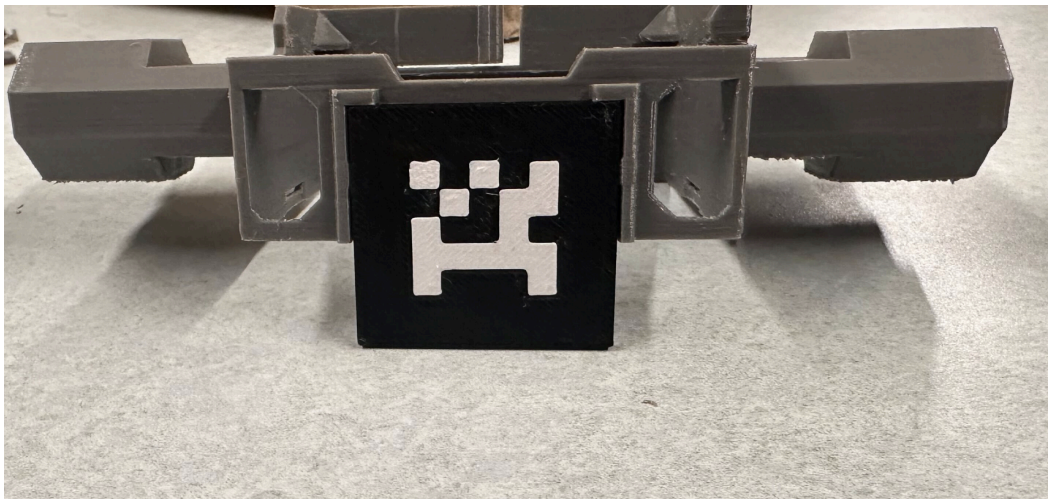


Figure X: first april tag for the ISS MicroMimic 3D printed by one of the members

Machine Learning Models

Machine learning models are trained to use certain telemetry data as inputs to output a prediction. For example, using voltage data of batteries, a model can predict whether certain batteries need to be charged. The purpose of the models are to get students more interested in

learning about the ISS and also sparking their interest in machine learning. To make the machine learning models more interactive, there will also be tunable hyperparameters (specifically learning rate and epochs) to allow students to slightly modify the model to optimize its outputs, making them more interactive.

Next Steps

The next steps are to mainly focus on combining the ISS micro mimic XRP with the gantry system. When both systems are combined, users can utilize the GUI to command the gantry system to move the ISS MicroMimic across the map and change the SARJ and/or BGA angles of the ISS MicroMimic. One very important feature when allowing users to control the translation and angles of the prototype is asynchronous functionality - the system should be able to receive commands while a current command is still being performed. For example, if the ISS MicroMimic were in the process of moving to a certain coordinate, the user should still be able to perform an emergency stop to interrupt the action midway without any malfunctions. Actions and commands cannot be limited to being performed sequentially.

Another functionality to focus on is the april tag detection system. The code for detecting multiple tags has been completed, so one of the members over break will set up the dependencies and code on his or her laptop to get it running successfully.

The last main functionality is to train at least one machine learning model over break that can accept input telemetry data and output a predicted telemetry statistic, such as predicted ISS velocity. Additionally, it would be better if this machine learning model can be incorporated onto a separate webpage on the GUI where users are also able to change hyperparameters to modify the results, making the model more interactive.

Systems

System Requirements

What are the system requirements?

Timelines

Put overall timelines here

Interfaces

Describe the overall system

Mechanical

What are the mechanical-mechanical integrations?

Electrical

CS

Mechanical-Electrical

Mechanical-CS

Electrical-CS

Notes for final submission:

You must submit all project documentations in a combined document, including MechE, ECE, CS, and Systems documentation. Meaning, for Minibot, there must be a single document, and for C1C0 there must be a single document. These must be converted into Microsoft Word documents, sent in a zipped folder titled “CCRT [Fall/Spring] [Year]” (for example, CCRT Fall 2010). This zipped file needs to be uploaded to Box. The Box links can be found here: https://it.cornell.edu/box. Then, the file in Box needs to be shared with David Schneider’s Cornell email.