Squishy Robotics: Developing Mobile Tensegrity Robots To Assist Emergency Responders

Eliana Abbas

Abhishek Bhagwat

Shawn Marshall-Spitzbart

Rebecca Schwartz

Aditya Vipin Thomas

From Left to Right: Abhishek, Eliana, Aditya, Shawn, Rebecca

Executive Summary

Squishy Robotics is developing rapidly deployable robots for disaster rescue, remote monitoring, and space exploration. The robots' flexible structure of rods and pulleys, called a tensegrity, makes them well suited for operating in rough, unpredictable environments. Squishy Robotics' technology merges robotics, mobile sensing, machine learning, big data fusion, and smart Internet of Things (IoT) to create robust information gathering devices. Many groups, particularly fire departments and the military, have expressed interest in these technologies and their applications.

Our team focused on mobile tensegrity robot development: creating a robust, reliable robot that can move across difficult terrain in an emergency scenario. Using Squishy Robotics' current mobile robot prototype, MR2, as a starting point, we completed multiple projects that involved developing a new mobile robot design, enhancing hardware reliability, improving the accuracy of electronics, and implementing low-level autonomous control. Key accomplishments include:

- Developing a calibration function that maps the electric current sensed from the robot's printed circuit board (PCB) to an external sensor, enabling us to obtain accurate, real-time estimates for the electric current.
- Designing a centrally actuated mobile robot prototype, named Flounder, and achieving autonomous successive punctuated motion while minimizing hardware complexity.
- Implementing a working simulation of the centrally actuated mobile robot to lay the groundwork for future advanced autonomous control with Model Predictive Control (MPC).

Additionally, individual members have participated in other projects involving refining the graphical user interface of MR2, building a state of charge estimator to allow for efficient battery usage, and assisting the stationary robot team with various design projects. This cross-over between different teams and projects has enabled greater individual productivity while decreasing the overlap and cohesion between each sub-project.

In future projects, the improvements to the electronic systems for MR2 will be adapted and implemented for use within the centrally actuated mobile robot. Further next steps include incorporating design knowledge from the stationary robot to improve the robustness of the mobile robot designs, and testing the mobile robots in a wider variety of response scenarios.

Section 1: The Role of Technology in Disaster Situations: How Do Squishy Robots Impact the Industry of Emergency Response?

The Industry of Squishy Robotics

Emergency situations occur daily across the world - from large events like earthquakes and wildfires, to smaller, localized events such as chemical spills and flash floods. To paint a picture: imagine that a major crash involving multiple cars and a large tank truck has just occurred on the interstate. Bystanders report that the truck is now spewing copious amounts of white gas, making it difficult to get a strong visual of the scene from above. Without knowing what chemicals the gas contains, emergency responders suit up in hazmat suits - slowing their response time and risking extreme dehydration to personally assess the situation. A delayed response in an emergency situation, or lack of knowledge about the situation the responders are about to enter, can cost lives. Situations like this that require HazMat responses occur over 1,150 times a day in the U.S. alone (NFPA Fire Experience Survey).

The outcome of the above scenario can be improved drastically with the help of Squishy Robotics, a company created at start-up incubator Berkeley SkyDeck. Squishy Robotics builds tensegrity robots that can be flown in via drone - reaching the disaster area extremely quickly. The robots can then be accurately dropped from heights exceeding 500 feet, surviving the fall and immediately gathering chemical, visual, and other sensor-related data to report back to the emergency responders. These rapidly deployable, mobile sensing robots have the potential to reshape the industries of emergency management and disaster response.

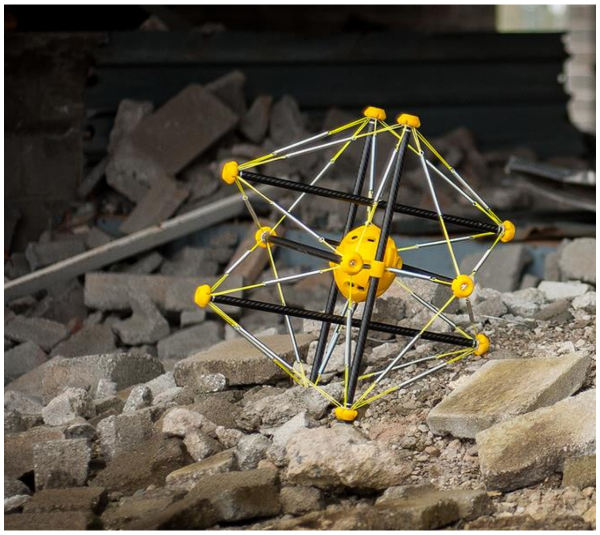


Figure 1: Squishy Robotics' Tensegrity Robot

The fields of emergency management and disaster response have shown increasing interest in adapting new technologies to better fit their industries. Advances in technology enable them to better gather and distribute information, which ultimately leads to increasing the emergency responders' ability to save lives (Holdeman). Technology enables more efficient rescue missions, improves our ability to manage emergencies, and helps to analyze and track the environment to predict disasters. With increased information - and improved ways to share and understand it - disaster response can be transformed to better reach and react to emergency situations.

As illustrated in the tank truck disaster scenario referenced earlier, Squishy Robotics' target market in the area of disaster response is the HazMat and CBRNE (Chemical, biological, radiological, nuclear, and explosive) response markets (Squishy Robotics). The problems faced in this specific area are threefold: first, emergency responders can arrive at a disaster site prior to it being recognized as a HazMat situation, thus exposing themselves to dangerous chemicals without the proper protective gear. Firefighters in particular are often at risk from carbon monoxide, but chlorine, argon, and sulfuric acid leaks are all relatively common due to their use in manufacturing. Second, suiting up in HazMat gear drastically increases the time it takes for responders to safely enter a disaster zone. Third, HazMat suits are extremely hot and uncomfortable - especially in summer or in warmer climates - and the responders using them are at high risk of dehydration. Even with the protection of their suits, emergency responders have occasionally suffered from exposure to toxic substances from gas leaks, oil spills, and chemical burns (Hudson). The ability to know the types and levels of chemical gasses in a disaster zone before entering it is invaluable to these responders.

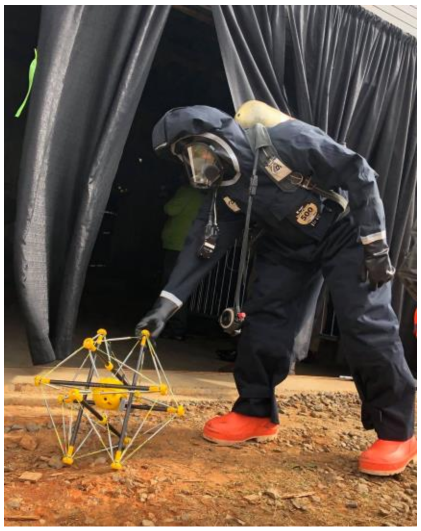


Figure 2: First Responder Training with a Squishy Robot

Another stakeholder highly invested in the technology behind the robots are Fire Departments. Squishy Robotics is partnered with some of the largest Fire Departments in the United States, including the County of Los Angeles Fire Department and the Houston Fire Department. Squishy Robotics' CEO, Alice Agogino, was hosted for an interview on a Smart Firefighting podcast. She explained how the data gathered by the tensegrity robots can be communicated to waiting firefighters via multiple methods - demonstrating how the product can work in the remote locations that firefighters often find themselves in (Gaughan). Firefighters are insistent about their need for receiving information early, preferably before arriving on the scene of a disaster in person. Additionally, they want the robots to include their standard four gas sensors - hydrogen sulfide, carbon monoxide, oxygen, and a combustible gas indicator

(Bonstell). Other sensors they have expressed interest in include thermal and infrared (Gaughan). With these accommodations in mind, the robots can currently be equipped with visual, audio, chemical, biological, radiological, and GPS sensors (Squishy Robotics). This data is transmitted wirelessly to human responders well outside of the disaster zone - preserving their safety.

While its primary market falls in the area of emergency response, Squishy Robotics also has the capability to reshape remote monitoring and commercial space exploration. In fact, the technology was initially developed as a robotic planetary lander / rover combination - with the idea that a tensegrity could safely land on planets with low atmosphere without a parachute, and then roll away to begin exploring (Squishy Robotics). The U.S. military has also shown interest, and has provided funding via multiple grants to assist with the development of the technology. Squishy Robotics participated in the Regional and National I-Corps programs to better understand this potential market.

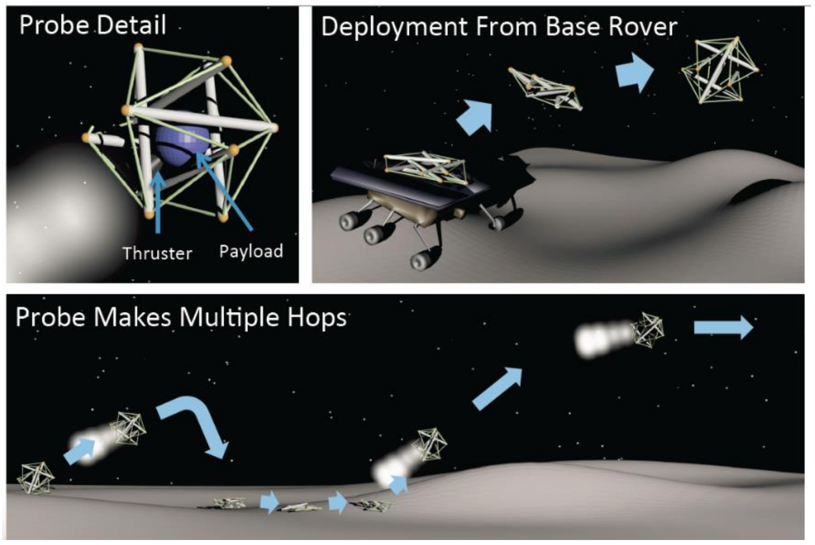


Figure 3: Initial Space Probe Concept

Today, Squishy Robotics is in the testing phase. The company is working with the L.A Fire Department to incorporate the tensegrity robots into its regular firefighter training program, as well as to field test the robustness of the robots' physical structure and communication ability (Gaughan). Additional work has been done with Menlo Park, one of the leaders in using drones to assist emergency responders. As of mid-2019, the goal of Squishy Robotics was to deploy enough tensegrity robots in the field to gather the data needed to use machine learning algorithms for data analytics. This will give emergency responders access to hazard recognition in real time - potentially leading to market opportunities such as wildfire condition monitoring and prediction.

Technical Background & Overview

The term “tensegrity” comes from the combination of the words tensile and integrity, and was coined by Buckminster Fuller. A tensegrity is a specific structure based on a network of continuously tensioned cables that hold together a system of compressed rods (Chen, “Soft Spherical Robot Design”, 1). When in equilibrium, the individual elements in the tensegrity structure only experience single axis loading - which reduces the number of potential failure modes. The lack of rigidity in the structure means that a tensegrity robot is a “soft robot” - a robot that can experience huge amounts of deformation without snapping, before returning to its original orientation. The flexible structure of a soft robot is often better suited for handling uneven terrain and unpredictable environments than their rigid body counterparts (Chen. “Modular Rod-Centered”, 1). Squishy Robotics’ tensegrity robots are designed to survive impact at terminal velocity. Upon impact, the structure transfers kinetic energy to potential energy via the deformation of the elastic elements of the structure, the tensioned cables (Chen, “Soft Spherical Robot Design”, 3). In testing, the robots have survived drops of over 600 feet.

While an infinite number of tensegrity structures exist, Squishy Robotics’ design uses six compressive elements - the rods - and 24 tensioned elements - the cables (Coldeway). The structure is robust enough to withstand the force of hitting the ground at terminal velocity, while compliant enough to bounce and deform around small obstacles and rough terrain.

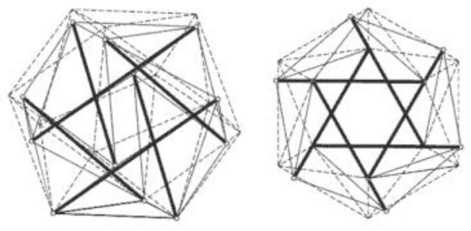


Figure 4: Basic Tensegrity Structure for a Squishy Robot

In the case of a mobile tensegrity structure, individual cables need to be pulled or slackened to deform the original shape of the robot. This changes the center of gravity of the robot, causing it to roll in the desired direction.

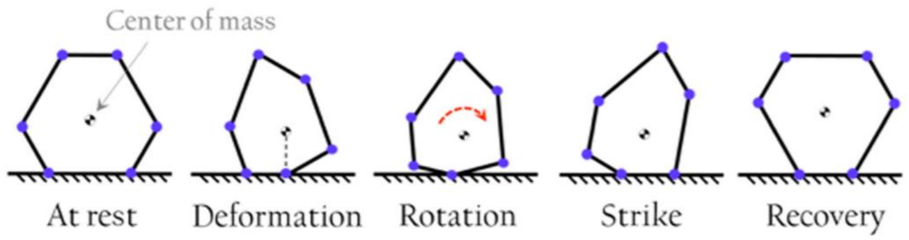


Figure 5: Conceptual Diagram of the Stages of Punctuated Rolling

Current mobile tensegrity robot designs rely on individually controlling each tensioned cable of the outer structure via motors housed in the compressed rods. While the designs have seen mixed success in terms of forward motion, they currently lack the robustness to withstand the drops that the stationary tensegrity robots have survived.

In terms of the electronics and software used on current iterations of mobile tensegrity robots, a nested controller is used to command the various motors (see Figure 6 below). The motion planner takes goal locations and the obstacle map as inputs, and then generates a feasible trajectory. This trajectory is then translated via the low-level controller into an optimal series of slackening and tensioning that the cables should undergo. The PID controller takes this series of instructions and computes the electric current required by the motors of the system to achieve the appropriate motion required. The downside to using this system is that PID controllers are inherently incapable of handling constraints on their outputs, meaning that they can demand larger values of electric current than the motors can physically handle (Wu). This results in burning out the motors - rendering them inoperable.

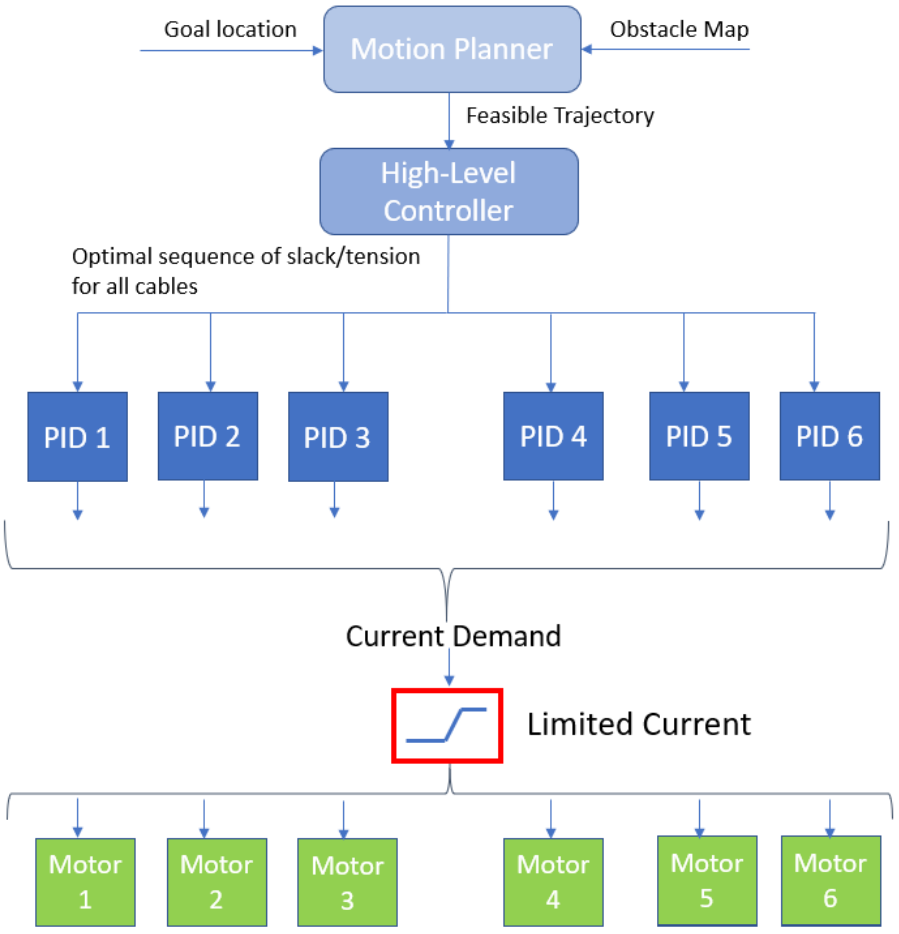


Figure 6: Control Architecture

The systems currently in development to overcome this issue utilize a finite state machine (FSM) based electric current limiting algorithm - which can override the output demands of the PID controller if it exceeds a given threshold. The success of this algorithm depends heavily on the accuracy of the electric current measurements it receives. Current mobile tensegrity robot systems gather these measurements from a piece of hardware called the motor driver - specifically an MC33926 motor driver. Readings from the MC33926 currently underestimate the value of the electric current - causing issues in successfully limiting the controller outputs as well as in other areas such as state of charge (SOC) estimations.

Measurements of the electric current gathered from the MC33926 motor drivers are internal to their circuit, meaning that there is no way to record them externally. This means that the inaccuracy cannot be directly corrected within the hardware of the circuit. Of the current propositions for a solution to this issue, the one implemented focuses on using an external electric current sensing circuit. This external circuit provides additional data to better interpret electric current readings, without requiring additional hardware to the physical robot.

In summary, the research completed by Squishy Robotics up until now has laid the groundwork for further developments in the field of mobile tensegrity robots. The following section of this report describes our team’s path in developing new designs and improving electronic systems used in these mobile robots.

Section 2: Technical Outcomes & Development in Mobile Tensegrity Robots for this Capstone Project

The majority of the work done for this project can be divided into two categories: improvements to the existing code, software, and interfaces for MR2 (Squishy Robotics' current mobile robot prototype), and development of a new, centrally actuated mobile robot design. Overall accomplishments include developing the hardware and software needed for accurate current sensing, developing a bed-of-nails tester for quality assurance, achieving successive punctuated motion with the centrally actuated robot both manually and autonomously, and creating simulations to assist with future high-level autonomous control. Additionally, individual team members have completed sub-projects in other areas, such as assisting the team working on improvements to the stationary tensegrity robots. These mechanical design projects include developing electronic enclosures for the robot transport drone and creating tools to improve the process of assembling and disassembling the robots.

1. Mobile Robot 2 (MR2) Improvements

The current mobile robot, MR2, moves by combining the actuation of multiple systems contained in the six rods forming its tensegrity structure. Each rod serves as a mini-robot, housing motors and electronic controllers that individually tension and slacken its attached cables. Movement in MR2 is induced through the combined actuation of these mini-robots. The electronic controller inserted in each rod, a PCB nicknamed "Slippy," controls four motors, four on-board motor drivers, one Teensy microcontroller, an inertial measurement unit (IMU), and a battery management system.

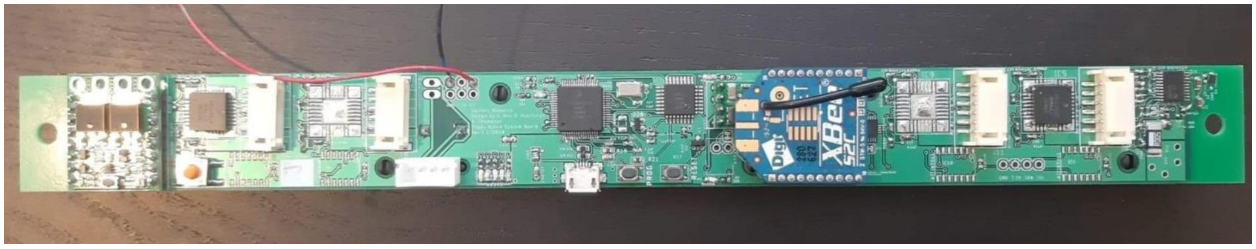


Figure 7: Slippy Board

Our team's original goals were to improve upon the electric current sensing performance of the Slippy board and to develop a quality assurance tool to help test the robustness of the robot's PCB electronic system. Later tasks added involved creating a user interface to visualize the sensor readings and developing a state of charge estimator for MR2.

1.1 Current Sensing Circuit Design for Improved Motor Driver Performance

Within the motor drivers used on MR2, there is a pin that indicates the electric current outputted to each individual motor. This reading is inaccurate, especially at mid-level electric current regimes. In stall situations with smaller motors, this has led to motor burnout. To fix this problem, our team prototyped a calibration circuit which measures this sensor inaccuracy and outputs a real-time correction function in software.

Our first steps involved initial research into the present electronic hardware on the Slippy board in order to understand the motor driver circuitry, the electrical connections of the microcontrollers, and the power ratings of the various components. The key goal to this research was to be able to measure the electric current drawn by the motor drivers at different load values, and then to feed the same values into the microcontroller for processing. As such, we first determined the best suitable electronic circuit to measure the input and output of the motor driver's electric current. Further research included studying and evaluating current sensing technologies using either linear integrated circuits or digital electronics solutions (Ziegler) (Regan).

We devised an external electronic circuit based on the electrical current sensor, IC ACS722. The sensor provides us with the analog output voltage proportional to the inputted electric current, which can then be processed by the microcontroller. Using a relay board, we were able to apply different current loads by changing the values of the resistors. These resistors are used to simulate the approximate draw of electric current from the motor in its locked state. When the motor rotation is locked, the electric current is limited by the resistance of the motor windings. The inductance of the motor windings themselves can be neglected in this case. The battery voltage of our system was 12V, and the maximum current draw for the motors was approximately 5.6A. Thus, by using Ohm's Law, we could calculate the resistance values needed. This enabled us to determine the response of the system at lower input electric currents - information necessary to properly develop a calibration system. The load values selected and tested are shown in Table 1 below, and the initial proof of concept circuit is shown in Figure 8.

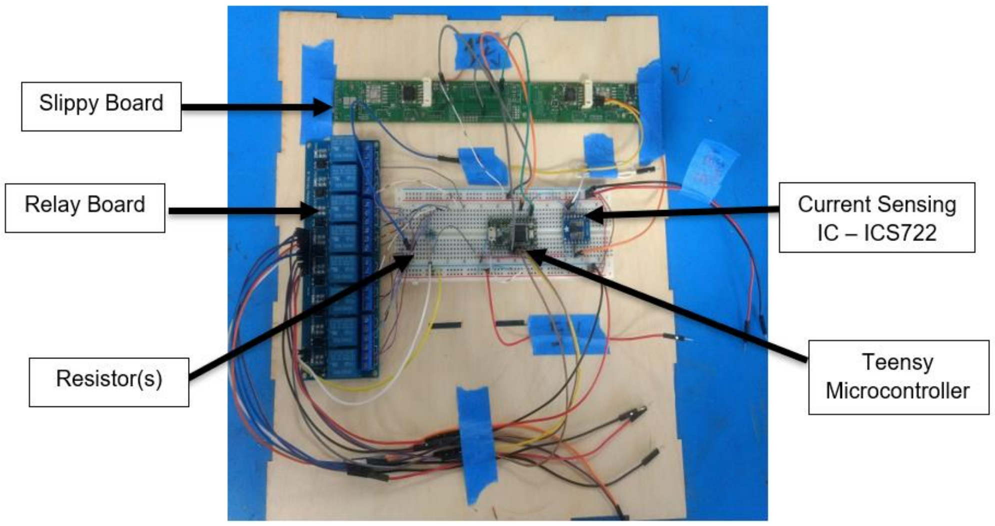


Figure 8: Proof of Concept for Current Calibration

Table 1: Load Values for Low Electric Current Response Testing

I (Amp)	R (ohms)	P(Watt)
0.1	120	1.2
0.3	40	3.6
0.5	24	6
0.7	17.14286	8.4
0.9	13.33333	10.8
1.1	10.90909	13.2
1.3	9.230769	15.6
1.5	8	18
1.75	6.857143	21
2	6	24
2.5	4.8	30
3	4	36
3.5	3.428571	42
4	3	48
4.5	2.666667	54
5	2.4	60

Multiple issues were discovered and troubleshooted during the development and testing of the above electrical system - specifically in terms of the hardware components. One key issue involved multiple required components, such as capacitors, not being initially soldered properly to the PCB. Additionally, it was discovered that the power draw was higher than initially predicted - necessitating the inclusion of power resistors.

In developing the final current calibration board design, electronics were transferred from the breadboard seen above to a one with robustly soldered connections. Eight different load values were selected and connected through a relay board to the Teensy microcontroller. Other features included PCB supports and clamping fixtures from the bed-of-nails tester (see Section 1.3), which were 3D printed and added to increase the physical robustness of the design (Mysore). The final electronic assembly is shown in Figure 9, and a complete circuit diagram of the current calibration assembly is shown in Figure 10.

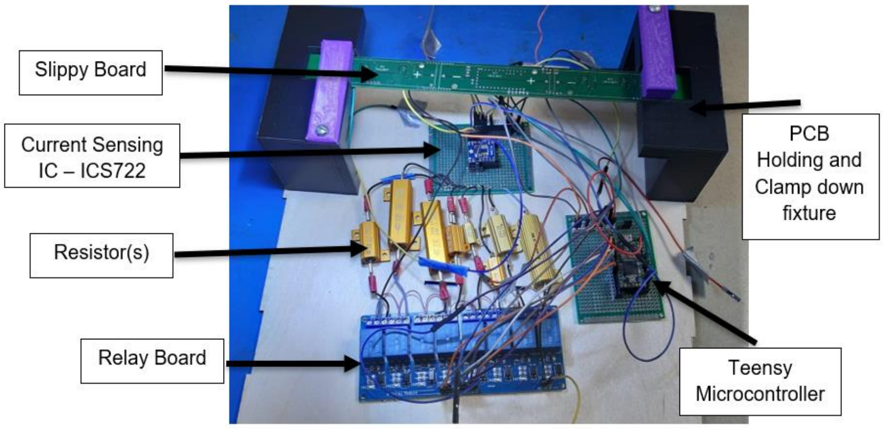

Slippy Board
Current Sensing IC – ICS722
Resistor(s)
Relay Board
PCB Holding and Clamp down fixture
Teensy Microcontroller

Figure 9: Final Current Calibration Board

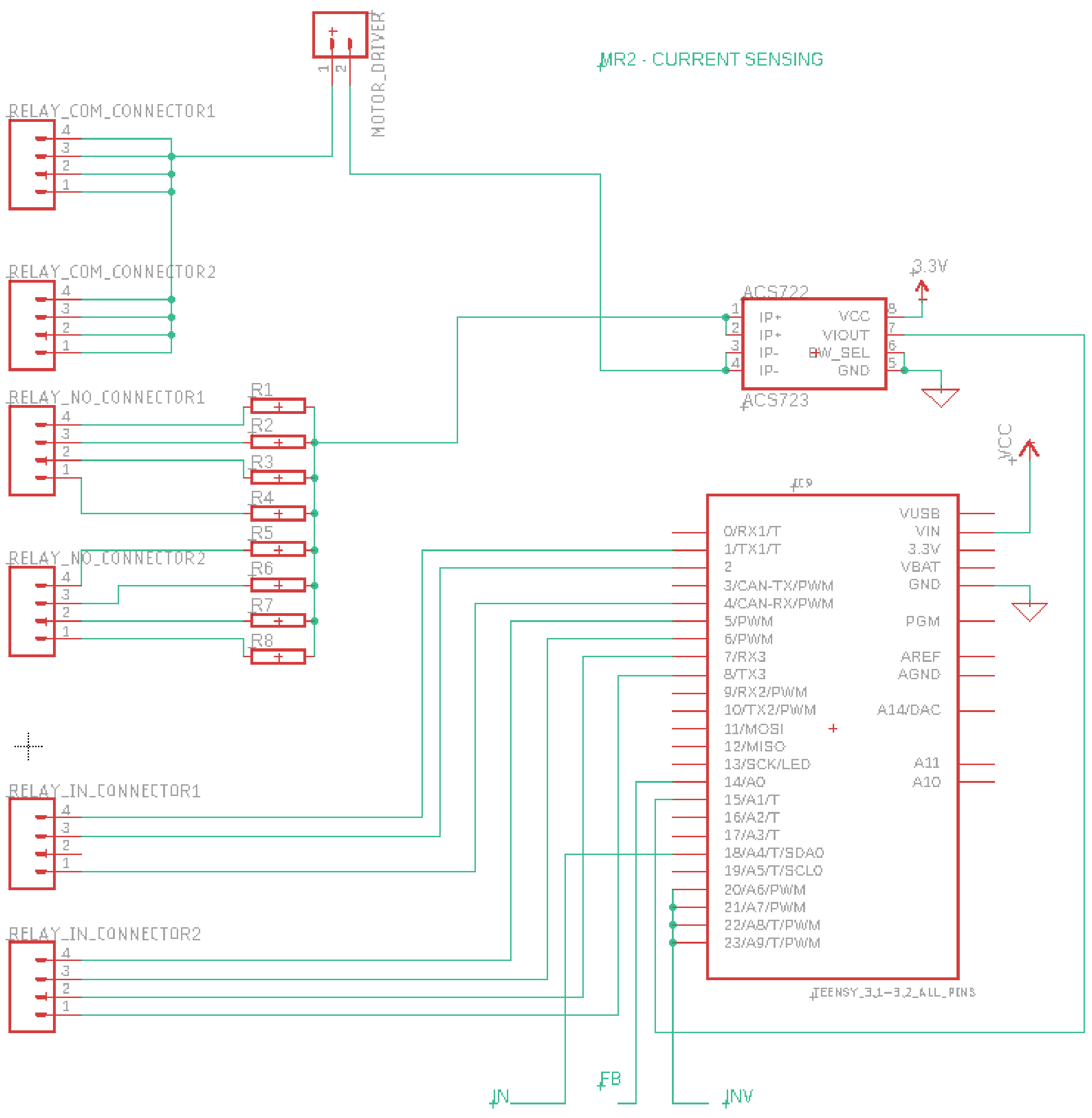


Figure 10: Final Current Calibration Electronic Assembly Circuit Diagram

1.2 Receiving Accurate Electric Current Measurements in Real Time

Now that we had the ability to measure the electric current properly for the Teensy microcontrollers, we had to develop an automatic calibration process for the microcontrollers currently in use on the mobile robots. To do this, we developed a software interface to go between the external electronic circuit and the robot's PCB. We developed a script using Arduino software, written in C++, to assign the correct pins for measurement to the microcontroller. From there, a Python script is used to import and then parse the serial data of the inaccurate electric current readings received from the microcontroller. To determine the true electric current, calibration constants are calculated from the known correct readings of the external circuit and the incorrect readings of the microcontroller using a least squares fit method. These calibration constants are then outputted from Python and inputted into a second script in Arduino, which loads them onto the EEPROM (electrically erasable programmable read-only memory) of the PCB's microcontroller. This entire process occurs during the set-up of the robot - once loaded, the constants remain in memory while the robot performs functions and uses them to output the correct electric current readings in real time. This allows for better electric current estimation and a more reliable electrical system.

The process described above is shown in a block diagram in Figure 11, and an example of determining the calibration constants is shown in Figure 12.

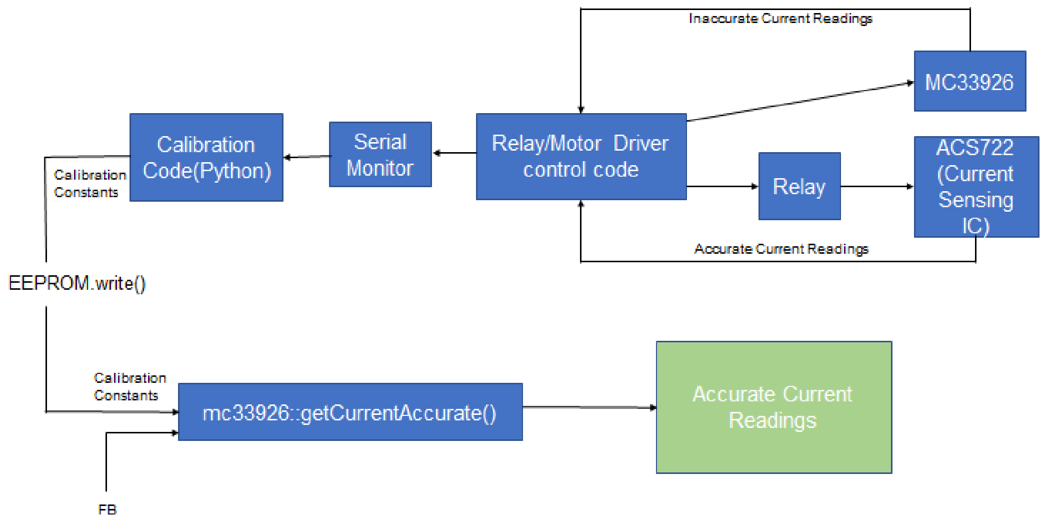

Inaccurate Current Readings
MC33926
Calibration Code(Python)
Serial Monitor
Relay/Motor Driver control code
Calibration Constants
Relay
ACS722 (Current Sensing IC)
Accurate Current Readings
EEPROM.write()
Calibration Constants
mc33926::getCurrentAccurate()
Accurate Current Readings
FB

Figure 11: Current Calibration Software Block Diagram

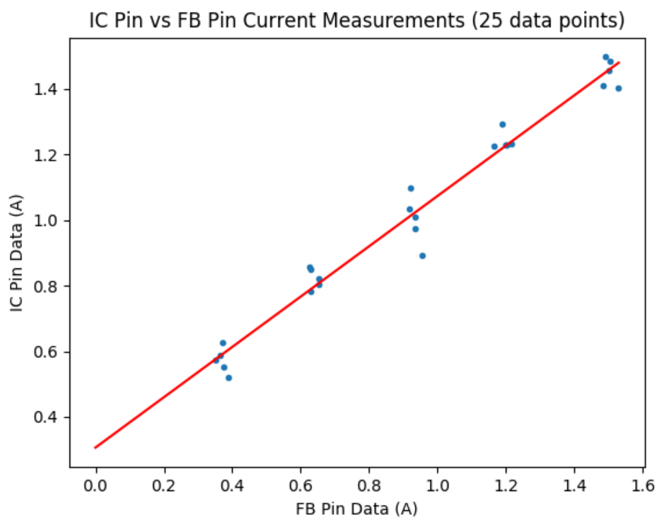


Figure 12: Example First Order Calibration Function for 25 Data Points

1.3 Bed-of-Nails Tester for Electrical Connectivity

To help ensure the quality of the PCBs, or Slippy boards, used on MR2 prior to their installation on the robot, we developed a new tool for testing the boards - called a bed-of-nails tester. A bed-of-nails tester is a quality assurance electronic device that checks for the continuity of desired pins on the electronic circuit board being tested. This is done by testing to see if a solid connection is actually formed between the conducting nails of the tool, and the pins on the PCB.

The first stage of developing the bed-of-nails involved designing and building a fixture to hold the PCB, as well as building an assembly to hold the conducting nails that would test the desired pins. Initially, the design for the bed-of-nails tester followed the industry convention - calling for a box-shaped design that allowed the PCB to rest along the top surface. The required wiring, conductive nails, and the electronic controller for the tester itself would be stored in the inner compartment of the box (see Figure 13 below).

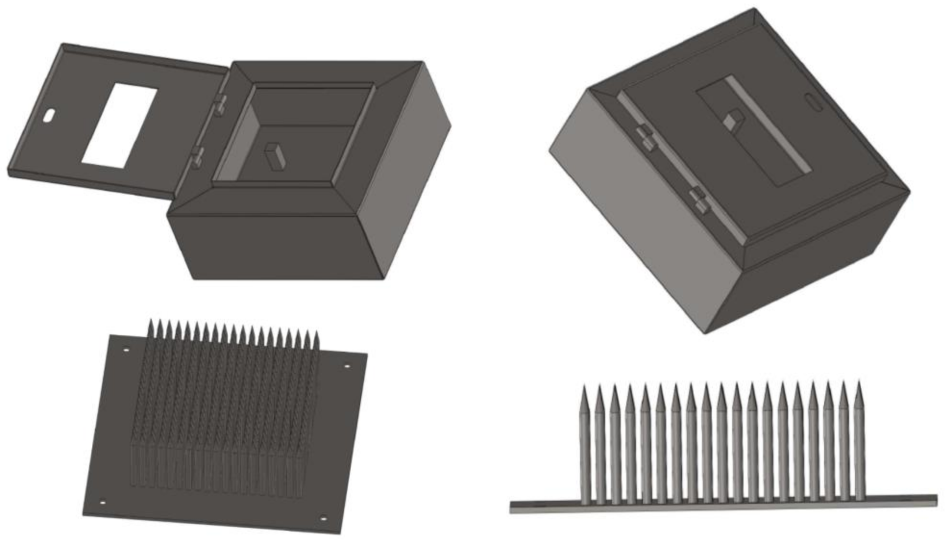


Figure 13: First Design of Bed-of-Nails tester

After creating this design, we realized that the Slippy board undergoes frequent updates - meaning that the bed-of-nails tester required a more open design to allow the user to freely access and adjust the positions of the underlying electronics and conduction nails. From this discovery, we decided to develop a basic structure to simply hold the PCB in place, allowing the user to work around it. Given that this deviated significantly from the industry standard for bed-of-nails testers, we first built a low-fidelity model to ensure the feasibility of the design (see Figure 14 below).

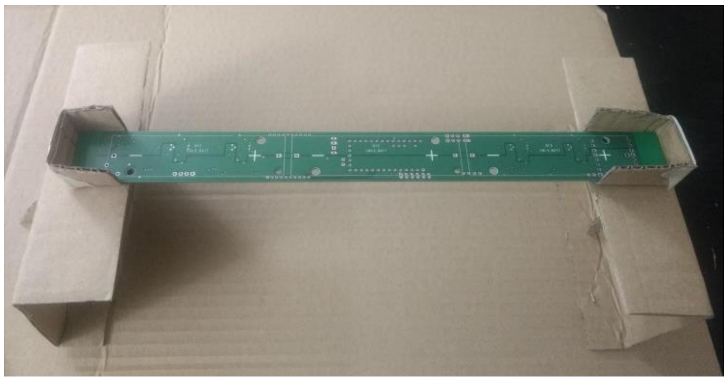


Figure 14: Low-Fidelity model for PCB hold fixture

By examining this prototype, we observed that if we simply allowed the PCB to rest on the slots and relied only on gravity to keep it in place, placing the system over the conductive nails pushed the board upwards and caused an unreliable point of contact between the PCB and the conductive nails of the tester. To compensate for this, we designed a mechanism to clamp the PCB to a support base. Final parts were then developed in CAD (see Figure 15) and 3D printed for use in the final current calibration board assembly, seen in Figure 9 and discussed in Section 1.1.

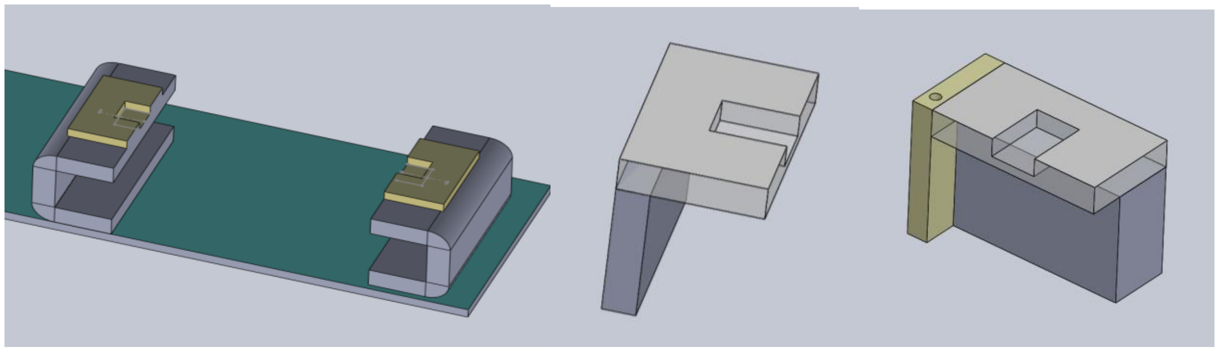


Figure 15: PCB Clamp Fixture Design Iterations

The second part of developing the bed-of-nails tester involved identifying which pins on the Slippy needed to be tested for proper conductivity. The three groups of pins determined to require testing are as follows:

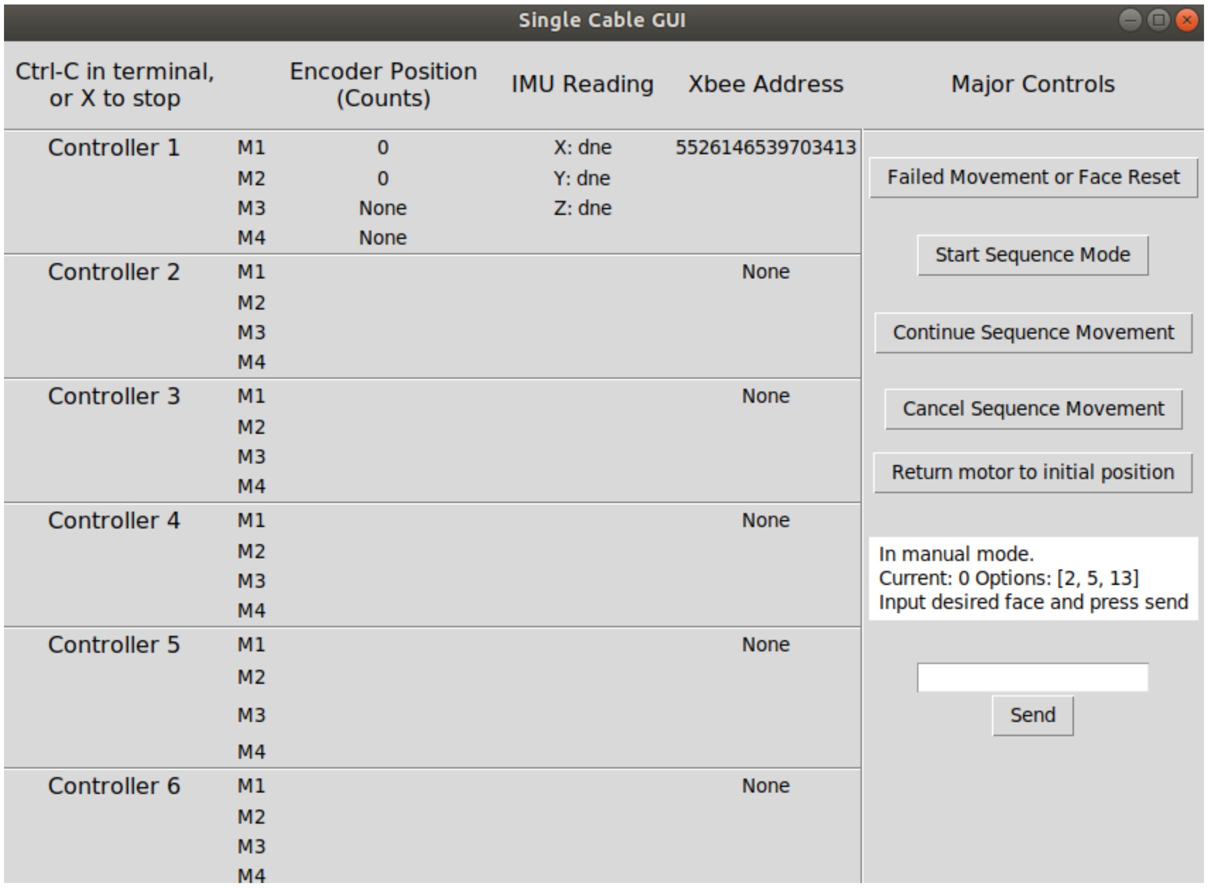
- Motor driver pins: out1 connector, out2 connector, FB1, IN1, INV.
- Teensy microcontroller pins: FB, In1, inv, analog inputs to relay (x8), VCC
- PCB pins: VPWR (x4), VCC, 3.3V, GND, EN

The layout of Slippy was then updated, and new boards were manufactured and assembled. The time required for the assembly process allowed our team to undertake new assignments to improve the use of MR2, as discussed in Sections 1.4 and 1.5.

1.4 Graphical User Interface For MR2's Electric Current Control Scheme

One issue in determining the robustness and accuracy of MR2's electronics is that Squishy Robotics did not have the infrastructure in place to reliably visualize sensor readings while running the robot's manual control code. To solve this problem, we helped create a new graphical user interface, or

GUI. Sensor readings are retrieved in real-time, and then relayed to the user via a tkinter window for user monitoring ("24.1. Tkinter - Python Interface to Tcl/Tk"). The GUI monitors and displays the desired outputs, including inertial measurement unit (IMU) readings, motor encoder readings, the specific type of robot being monitored, and the controller identification number. A design of this GUI is shown below in Figure 16.


Single Cable GUI
Failed Movement or Face Reset
Start Sequence Mode
Continue Sequence Movement
Cancel Sequence Movement
Return motor to initial position
Send

Ctrl-C in terminal, or X to stop		Encoder Position (Counts)	IMU Reading	Xbee Address
Controller 1	M1	0	X: dne	5526146539703413
	M2	0	Y: dne	
	M3	None	Z: dne	
	M4	None		
Controller 2	M1			None
	M2			
	M3			
	M4			
Controller 3	M1			None
	M2			
	M3			
	M4			
Controller 4	M1			None
	M2			
	M3			
	M4			
Controller 5	M1			None
	M2			
	M3			
	M4			
Controller 6	M1			None
	M2			
	M3			
	M4			

Major Controls

```
In manual mode.
Current: 0 Options: [2, 5, 13]
Input desired face and press send
```

Figure 16: Intermediate Design of MR2 Control GUI

1.5 State of Charge Estimation for MR2

The state of charge (SOC) of an electrical system is the percentage of usable charge remaining within the battery cell that provides power. As this percentage cannot be measured directly via sensors, we instead needed to determine its value from different measurements already provided by the existing sensors in the hardware of the system. To ensure computational tractability (i.e., to ensure a computer could handle the computations), we utilized a linear Kalman Filter based estimator, which is an observer that generates estimates by combining sensor measurement information with prior knowledge of the model dynamics (Kalman). We used an internal resistance model for the battery model in the estimation, with its parameters determined by MR2's Li+ion batteries (Plett). The initial model was implemented in MATLAB, and the proof of concept will later be translated into embedded code in order to work in tandem with the Battery Management System (BMS).

The benefit of implementing this system is to improve the battery usage efficiency of the mobile robot. By knowing the SOC, control algorithms can be implemented to prioritize performance over battery usage, and vice versa. For example, in the case where an estimated low SOC is reported - when the robot is low on charge - the algorithm can limit the power usage of its electronics while sacrificing performance in order to stay working for a longer period of time.

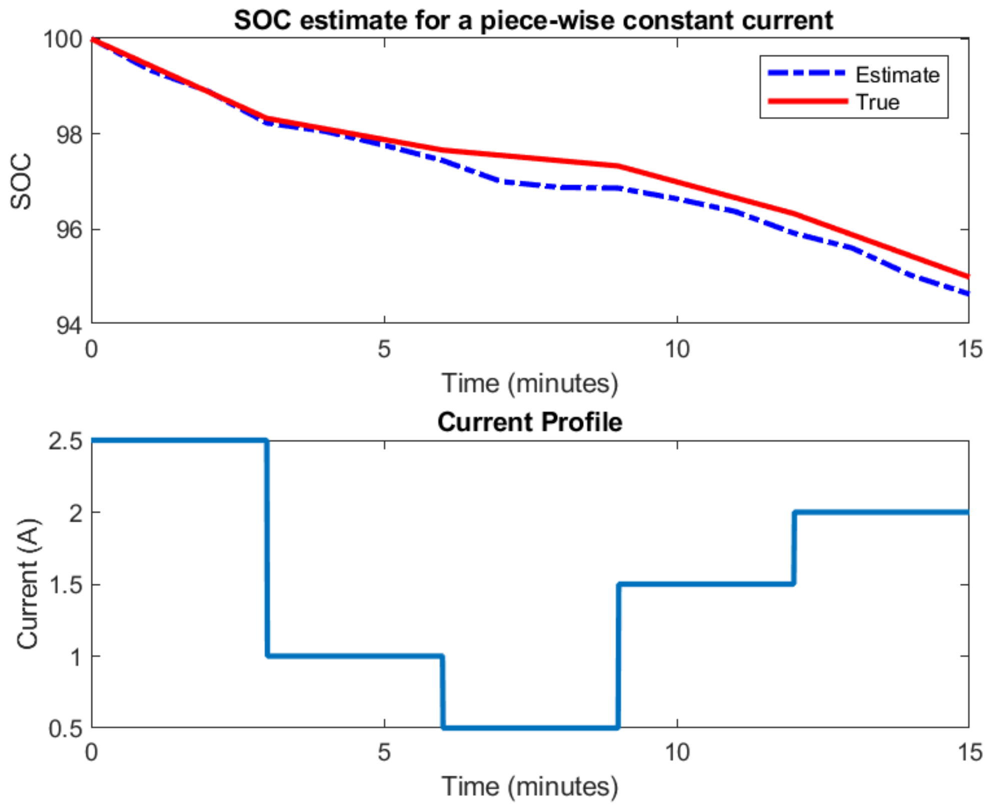


Figure 17: SOC Estimate for a Piecewise Constant Load

2. Centrally Actuated Mobile Robot Development

The current mobile robot design, MR2 (see Figure 18 below), moves by tensioning each individual pulley in the outer structure, resulting in a total of 24 motors and 6 electronic controllers housed in the compressed rods of the tensegrity structure ("Soft Spherical Tensegrity Robot Design", 2). The plurality of the controllers and actuators allows for each cord to be tensioned individually, resulting in highly controlled movement. From a systems engineering point of view, each rod acts as its own unique "mini-robot" - or a system of six different robots tied together by tensioned cables. This results in congested wireless communication, as well as a complex hardware system. Another unexplored design for potential controlled motion of a tensegrity robot uses a centrally actuated payload. In this design, motors housed in the central payload can tension pulleys attached to the external structure in order to shift the center of mass of the robot, causing a roll-like motion. The challenge set for our team was to determine the feasibility of this centrally actuated design, decrease the number of motors and controllers needed, and investigate integrating autonomous control into the system.

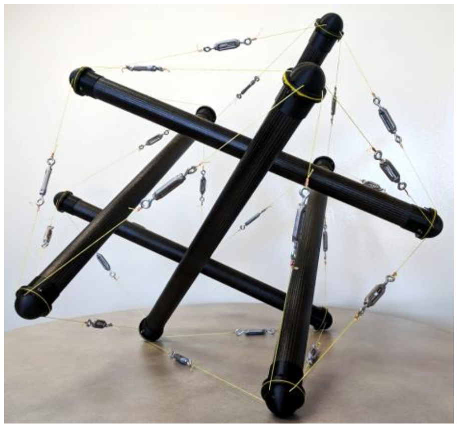


Figure 18: MR2, Current Mobile Robot Design

2.1 Feasibility of Motion using a Centrally Actuated Design

In our initial research and testing phase, we utilized one of Squishy Robotics' low fidelity education robots (see Figure 19). This gave us insight into the design of the basic stationary structure - specifically, the symmetry involved. This specific tensegrity has an external structure made up of 20 faces, 8 closed faces (where the external structure forms an equilateral triangle with three cables) and 12 open faces (where the external structure forms an isosceles triangle with two cables and one open side). Regardless of whether the face is open or closed, the length of the cable in its natural, untensioned position is the same. Additionally, by dropping the robot from various heights and by physically rolling it along the ground, we observed that it nearly always ended in a position where a closed face was against the ground. Once in this position, it took a relatively significant amount of force to turn the robot to the next face. In contrast, on the rare occasion the robot ended up supported by an open face, it took relatively little force to tip the robot onto one of its closed faces. As such, we determined that a "stable" position is when the robot rests upon a closed face, and an "unstable" position is when it rests upon an open face.

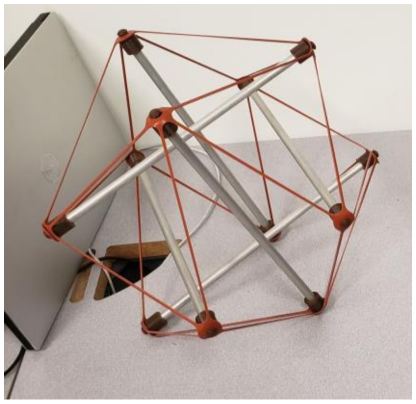


Figure 19: Squishy Educational Robot

From these design features, we theorized that if we could shift the center of mass of the robot while on a closed face enough to cause it to tilt to an open face, the momentum would allow it to continue falling until it reached the next stable position, or the nearest closed face. Thanks to the symmetry around an equilateral triangle, we also determined that the force needed to move from a closed face to an adjacent open face would be the exact same regardless of which of the three surrounding open faces we needed to move towards - and that once we generated enough motion for the robot to shift to that open face, it would continue to the next closed face. This action, termed “punctuated motion”, of moving between two semi-adjacent closed faces formed the basis of our first round of design. We estimated that punctuated motion would occur when the payload had shifted enough to move the center of mass beyond the edge of the equilateral triangle - thus exiting its stable state and causing the robot to tilt.

In Squishy Robotics’ stationary tensegrity robot design, the central payload houses cameras and electronic sensors. It is attached to the external structure by a network of 12 cables, each cable leading from one of six different points on the payload - placed at the top and bottom and spaced evenly about its equator - and ending at one of the 12 rod ends.

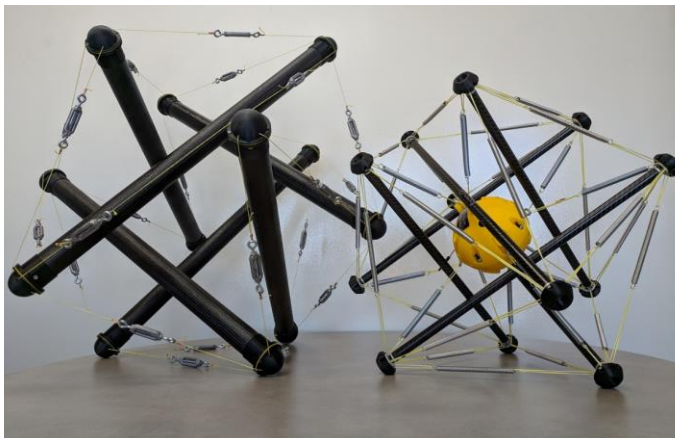


Figure 20: MR2 (Left) and Traditional Stationary Robot (Right)

We determined that the most straightforward method of testing punctuated motion would be to attach each of these internal cables to their own motor. This allowed us to reduce the number of necessary motors from 24 to 12, as well as to house all the motors in the same central location. With individual control over each internal cable, we were able to move the central payload anywhere within the structure of the tensegrity. As at this stage we were more concerned with determining feasibility than integrating the entire electronic sensor system used in the stationary robot into our mobile robot, we designed a basic structure intended to only support the motors used. Key features included positioning the motors such that they were at the same points they would occupy if they were placed on the outside of the current stationary payload design, and additional structure to guide the cables in order to avoid tangling.

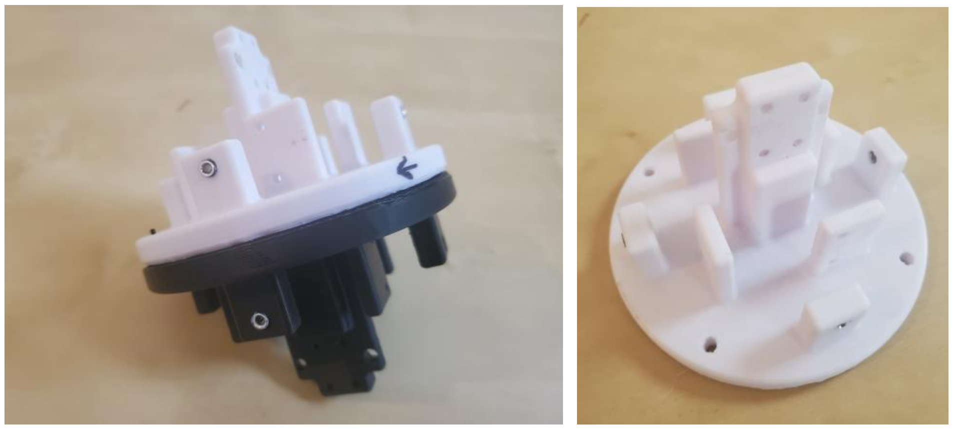


Figure 21: 12 Motor Payload Structure

Another design decision was choosing to use bungee cords to create the external tensioned structure, rather than the spring and pulley system found on the stationary robot. This was done to significantly decrease the spring constant of the system, making it more flexible and thus easier to test movement. Given how the bungee cords were thicker than the pulley system in place, we also had to design new temporary end caps for the rods to properly support the structure in place of the bolt and washer system used on the stationary robot. Another team working on the stationary robot had developed a new design using 3D printed inserts to minimize weight, and we were able to adapt that design to fit the size of our rods.

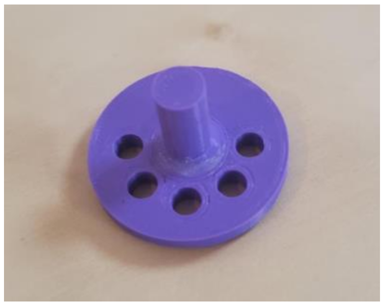


Figure 22: End Cap Design, Version 1

The last major design change we had to make involved the rods themselves. Basic simulations done by Squishy Robotics prior to this project showed that the center payload would need to move so far from its original position to achieve motion that it would collide with the standard compressed rods. As such, curved rods would be needed to enable the payload to move far enough to generate motion without causing a collision. To simplify the design, we elected to utilize bent rods - with the bend angle being at the joint where the two rods met (i.e., for a straight rod, the bend angle is 180 degrees). The bend angle had two major constraints: it had to be wide enough to enable payload movement without collision, and it had to be narrow enough such that it did not extend beyond the frame of the external structure so as to avoid colliding with the ground during rotation. For the former constraint, we estimated that the payload would need to move from its central position a minimum of 8 inches to move the center of mass over the edge of the equilateral triangle. This gave us a maximum angle of approximately 140 degrees. The latter constraint was determined by the geometry of the stationary robot, giving us a minimum angle of 126.9 degrees. The first angle connectors used were 3D printed at a bend angle of 120 degrees (the maximum angle we could print and maintain robustness of design), but later designs used pipe fittings to achieve a 135 degree bend angle.

For the electronics of the system, the 12 motor configuration required two control boards, each composed of a custom PCB, a Teensy microcontroller, six motor drivers, two voltage regulators, an XBee radio, a power switch, six ports for brushed DC motors, and an inertial measurement unit (IMU) (see Figure 23). The two boards were controlled through the same user interface via Python, run in Ubuntu. The two control boards were housed externally rather than on the central payload to provide us better access to the electronics, as can be seen in Figure 25 below.

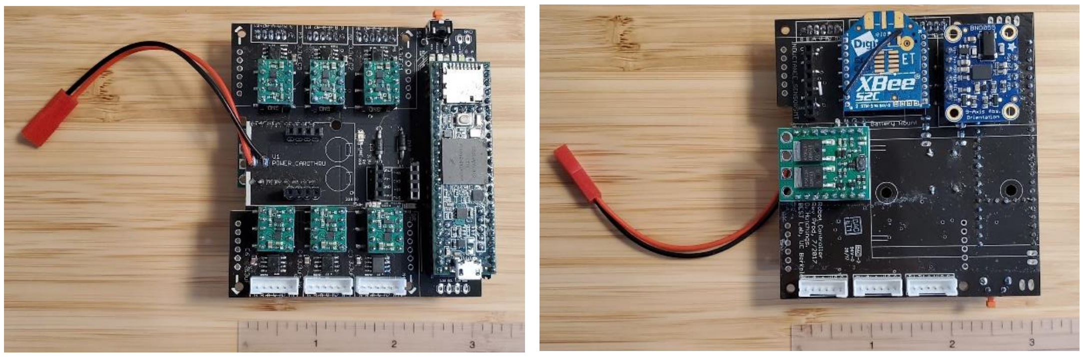


Figure 23: Front (left) and Back (right) of a Control Board

The motors used were Micro Metal Gearmotor High Power 6V brushed DC motors, with an extended motor shaft and a gear ratio of 250:1. Each individual motor can receive 6 V and spin up to 120 RPM at 70 mA of electric current when no physical load is applied, and has a stall torque of 3.4 kg-cm at 1.6 A.. Each motor was driven by a DRV883 Single Brushed DC Motor Driver Carrier, which can send up to 1.7 A each motor, enough to max out the power output of the motor.

The Teensy microcontrollers ran via Arduino code, and manual control of the motors came from the aforementioned user interface in Python. The code works by establishing radio communication via the Xbee, before defining the limiting parameters of PID and electric current limiting specific to the control board identified. This allowed us to detect the number of boards successfully communicating with the controller, and to simultaneously actuate motors from both boards. Specifically, the user interface allowed us to input the desired absolute value of each encoder individually, causing the motors to spin to reach that value once the command was initialized.

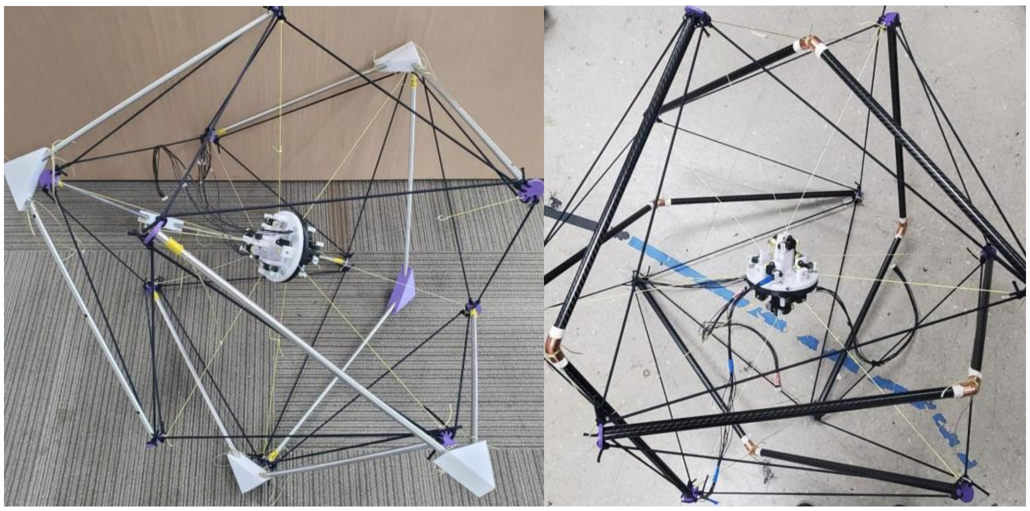


Figure 24: Flounder, 12 Motor Configuration

120° Bend Angle (Left) vs. 135° Bend Angle (Right)

The fully assembled robot, named Flounder, achieved motion by controlling four motors at a time. The two motors closest to the closed face we were trying to reach were spun so as to tension their attached cords, and the two motors in the opposite direction were spun so as to loosen their attached cords. This enabled us to shift the center of mass significantly without over-tensioning the system and snapping or bending the rods. We were able to successfully achieve punctuated motion (see the series of images in Figure 25 below).

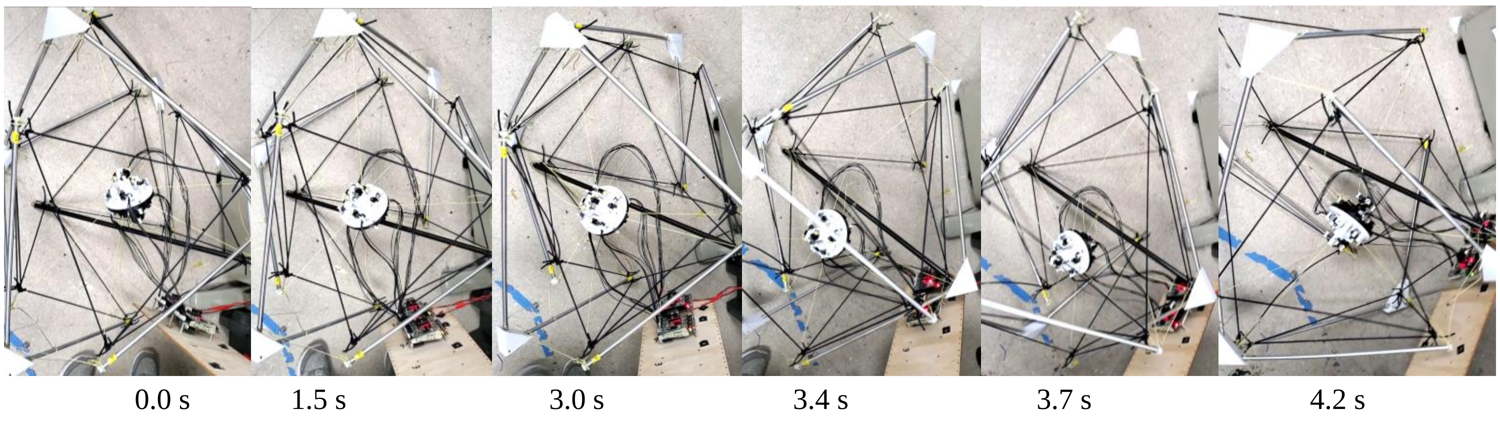


Figure 25: Punctuated Motion, 12 Motor Configuration

The 12 motor design iteration had repeated hardware issues - specifically with the electric current limiting of the motors. This was solved by using a power supply and limiting the voltage properly, which increased the difficulty of testing motion due to being tethered to the power supply. Punctuated motion was eventually achieved after multiple attempts - but was never achieved reliably. Issues started occurring when attempting to return the payload to a centered position in its new orientation - the robot had warped so dramatically that the unconstrained bent rods were able to spin, resulting in the rods getting caught over adjacent end caps. Constraining the rods' motion by tethering them to the endpoints of adjacent rods helped improve the issue, but significant motion still occurred as the robot warped. Another issue that occured was that to achieve punctuated motion, the payload had to move from the center to only an inch shy of the rod ends of the attached cables - farther than initially expected. The spools used to control the cables had too small a diameter to hold the amount of string wound in this motion - resulting in significant tangling issues during the tensioning and the loosening of the cables.

While reliable punctuated motion was not achievable with the electric current limitations, successful travel between two semi-adjacent closed faces and the subsequent recentering of the payload was achieved. Given the symmetry of the system, we hypothesized that punctuated motion should be possible from any closed face to any of its semi-adjacent closed faces. From here, we devoted our energy to simplifying the system in order to increase the robustness, with the end goal of stringing multiple occurrences of punctuated motion together to achieve successive motion.

2.2 Design Iteration, Improvements, and Continued Testing

When testing punctuated motion in the 12 motor design iteration, we found it easiest to keep motion controlled and symmetrical when the motors were run in pairs. This meant that the two motors controlling the cables in the direction we wanted to go were tensioned simultaneously, while the two in the direction we were leaving were loosened simultaneously. The remaining 8 motors were left untouched. Due to motion being possible by running the motors in pairs, we determined we could further decrease weight and system complexity by halving the number of motors. Each of the 6 motors left would control motion in one direction by tensioning or loosening two cables simultaneously.

The first step to moving towards a 6 motor configuration was redesigning the payload structure. Like before, we decided to use a basic structure housing the motors placed at points that represented the diameter of the stationary robot's payload - although we continued to include guides for the cables.

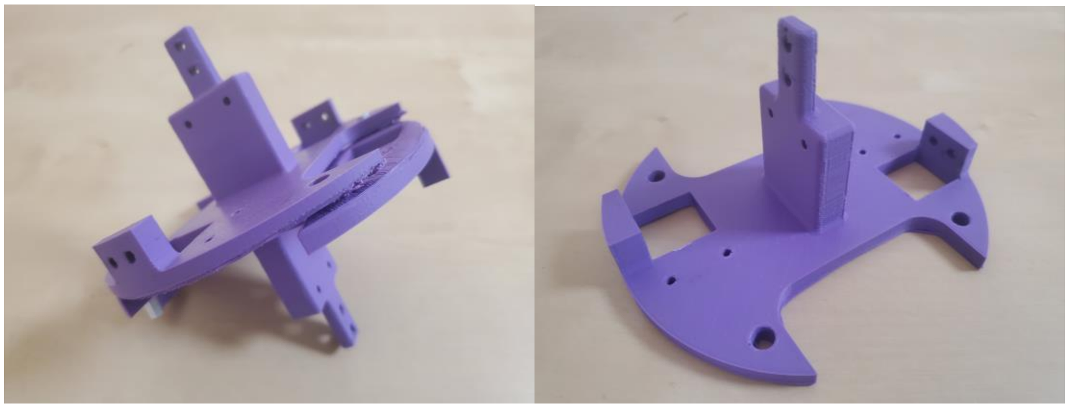


Figure 26: 6 Motor Payload Structure

Additionally, controlling two cables with a single motor meant that we had to redesign the spools used to guide the cables in order to prevent tangling. To keep the two cables separated, a double flanged spool was designed and then machined. We also took the opportunity to increase the diameter of the spool's flanges, so as to avoid cable slippage in extreme positions.

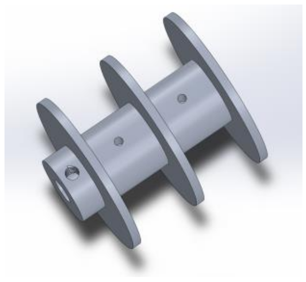


Figure 27: Double Flanged Spool CAD

We also tackled the issue of the unconstrained bent rods by redesigning the end caps. The first change made was increasing the diameter of the insert in order to have a much tighter press-fit into the rod itself. The second included switching material from PLA to Ninjaflex, a more flexible plastic with less slippage due to increased friction.

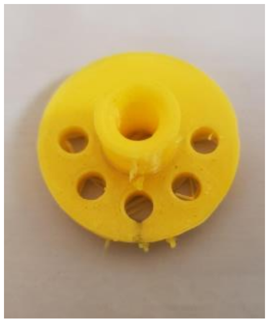


Figure 28: End Cap Design, Version 2

In moving from a 12 motor to 6 motor system, we estimated that the holding torque requirements of each motor had to be doubled. To keep consistency, the same family of motors were selected from for the 6 motor system. We chose the Micro Metal Gearmotor High Power 6V brushed DC motor, this time with a 1000:1 gear ratio, because of its ability to supply over twice the holding torque than the previously used motors. With a stall torque at 2.5 times the torque of the previous motor at 1.6 A, the existing motor drivers would remain sufficient, as would most components of the physical design as the two motors were of similar size and dimensions. The only downside to the selected motor was that its higher gear ratio would have a slower performance - at 31 RPM without a physical load. That said, the benefits of a 6 motor configuration still vastly outweighed the downsides as it only required one controller board. To maintain ease of access, the control board was still housed externally rather than on the payload itself.

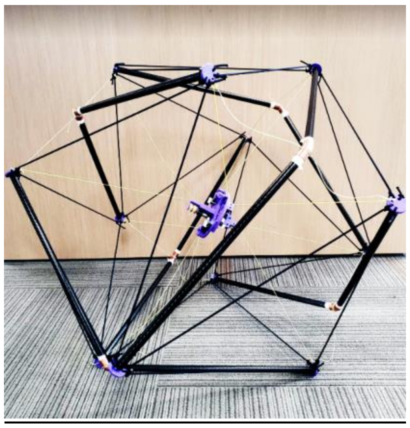


Figure 29: Flounder, 6 Motor Configuration (Updated End Caps not Pictured)

We saw immediate improvements when testing the 6 motor configuration. The change in end cap design drastically decreased the jamming issues found in prior tests when the bent rods swung out of place. While the initial stringing of the cables resulted in multiple windings and unwindings of the spools, once the positions were finalized, we encountered no tangling issues. The increase in gear ratio did result in slower movement of the central payload as expected, but punctuated motion was still achieved.

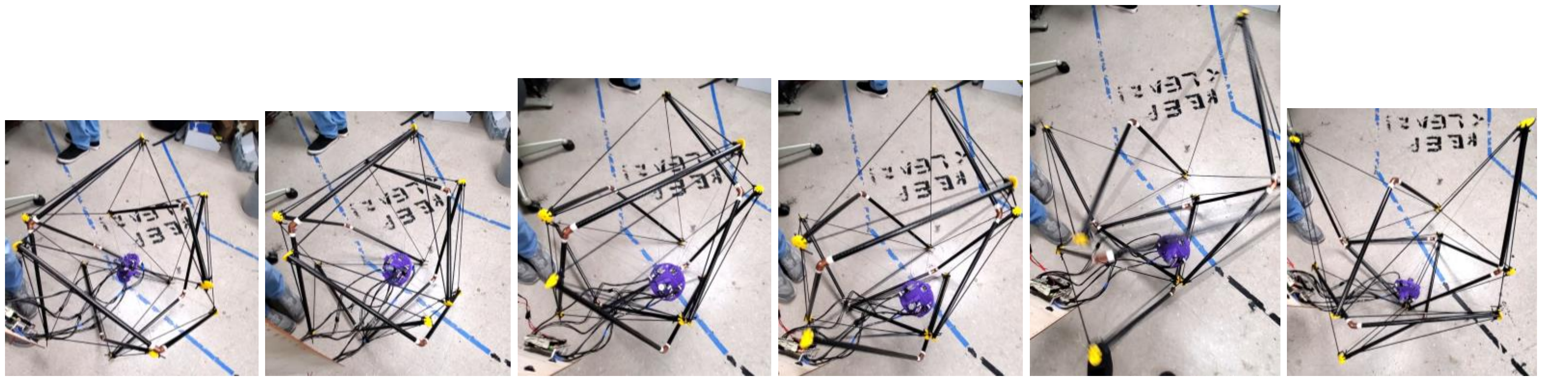


Figure 30: Punctuated Motion, 6 Motor Configuration

While both achieving punctuated motion and recentering the payload successfully occurred with increased reliability, several issues and areas for improvement were noted. First, if the central payload did not move far enough to cause motion on its initial command, successive incremental steps in the direction of motion did not move it to the next face - even when it had travelled the distance necessary if it had travelled the complete distance in a single motion. The momentum inputted by the dynamically moving system is thus vital: if the payload does not move far enough to generate motion, it must be recentered in the same orientation before reattempting. To ensure this position was reached in future tests, we ran multiple tests in a single direction at varying encoder counts to determine the optimal distance the motor needed to travel. Upon testing the motors, we found that ~10,000 encoder counts ≅ 1 inch of distance travelled by the payload. With cables of an average of 14 inches, this resulted in a working number of 120,000 encoder counts. Once determined, complete punctuated motion was achieved on 50% of the attempts, and partial punctuated motion (reaching only the adjacent open face rather than the next closed face) was achieved on 16.7% of the attempts. No motion was achieved for the remaining 33.3% attempts.

In terms of the success of recentering the payload after achieving motion, the only times jamming due to the bent rods overlapping with adjacent rod ends occurred was when Flounder did not complete punctuated motion by itself, and was instead moved manually into its next orientation and commanded to recenter. Each time that Flounder achieved punctuated motion without manual interference, it recentered itself correctly.

In finding this optimal distance travelled, we also realized that since the weight of the payload had decreased significantly from the 12 motor configuration to the 6 motor configuration, the payload had to travel further to reliably achieve punctuated motion. Adding weights to the payload appeared to make it more likely to achieve motion, but no concrete conclusions were drawn as to how much weight was needed. This insight is of high importance because decreasing weight of the robot is a goal across projects - the lighter the robot is, the easier it can travel via drone. With a centrally actuated robot, the weight of the payload must be enough to shift the center of mass fully over the edge of the equilateral triangle forming the base of the stable position. As such, there is a minimum weight requirement to the central payload to allow for this motion. As the 6 motor configuration is significantly lighter than the current stationary robot's fully developed payload while still achieving semi-reliable punctuated motion, this is not a current concern. That said, using the geometry of the robot to determine the necessary weight of the payload needed for punctuated motion in relation to the weight of the structure is an area for future research.

The last major issue noticed during testing was slippage of the motor gears. With relative frequency, the motor tensioning the cables would not move as far as the motor loosening them - or vice versa - even when the inputted encoder counts were the same. Initially, we assumed that slippage of the gears accounted for all inequalities of motion, as we could on occasion hear the gear teeth skipping. Further discussion around the electronics involved revealed that missing encoder ticks was a known issue for these control boards when motors moved too fast due to how PID was implemented. Adjustments were made to the code containing the PID control for the motors, after which the distance travelled was tested in the same direction five times in a row. The payload appeared to move the same distance for the first four attempts, while gear teeth skipping was responsible for the shortened distance travelled on the fifth.

At the end of this design iteration, we moved away from using the power supply and returned to battery powered operations to better manage the limitations put on Flounder's movement due to tethering. We then tested successive motion by attempting to chain three rounds of punctuated motion together. The first attempt completed a successful first round punctuated motion, but failed on the second round. The second attempt completed successfully punctuated motion twice in a row, but failed on the third round. Observers noted that even without using a power supply, the tethering to the external control boards prevented at least one, if not multiple, of the failed attempts from achieving motion. As our final goal was to implement low level autonomous control, we determined that our final design challenge would be to incorporate all the needed electronics into a single enclosed housing.

2.3 Implementing Low-Level Autonomous Control

To prepare Flounder for autonomous control, we first wanted to increase the reliability of the design by removing the chance of failure due to human error in tethering. The first step of this involved designing an enclosed version of the payload that could house the motors, control board, and battery. We again decided to keep the electronics as simplified as possible and did not include the sensors found in the stationary payload, even as we adopted its elliptical enclosed design.

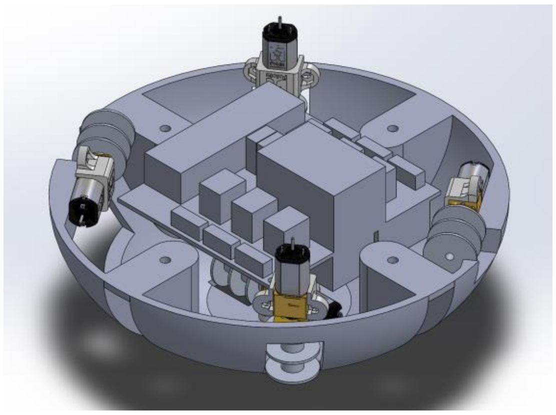


Figure 31: Enclosed 6 Motor Payload CAD

The payload design now included additional features to secure the control board, although it was determined that since this prototype did not require the robustness of the stationary design for the current testing, the battery could fit loosely within the open space of the payload. We also decided to include the motors inside the housing, rather than following the original plan of the earlier designs that assumed the motors would be placed outside the housing. This was so that we could include some semblance of guides for the cables to prevent tangling.

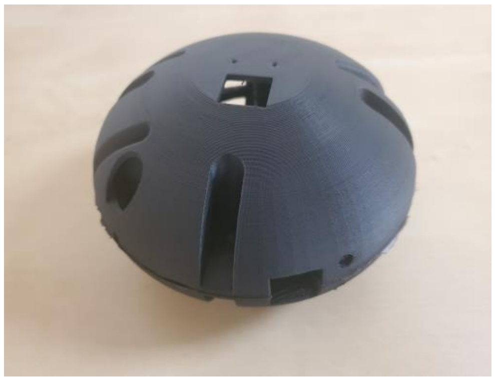


Figure 32: Enclosed 6 Motor Payload Structure

We also took the opportunity to further iterate and improve on our end cap design. The earlier designs used a flat disk that occasionally caused excess friction depending on the angle at which it collided with the ground. Instead, we decided to curve the design to allow for a smoother surface that Flounder could more easily pivot upon while moving. Doing this enabled us to encase the rods within the end cap, rather than using an insert into the rods, resulting in an even more stable orientation of the rods themselves and further decreasing the chance of jamming during motion.

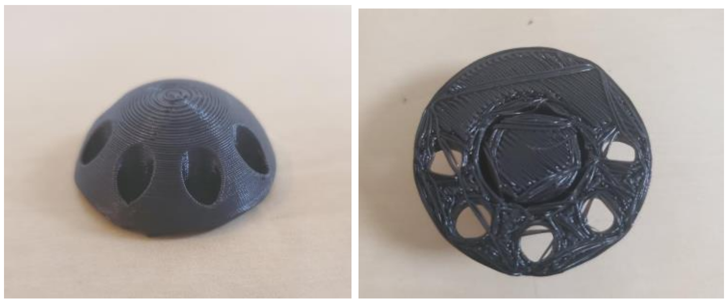
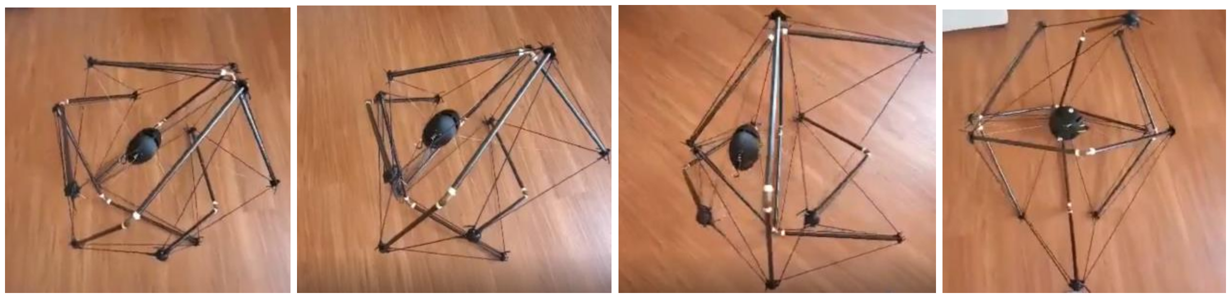


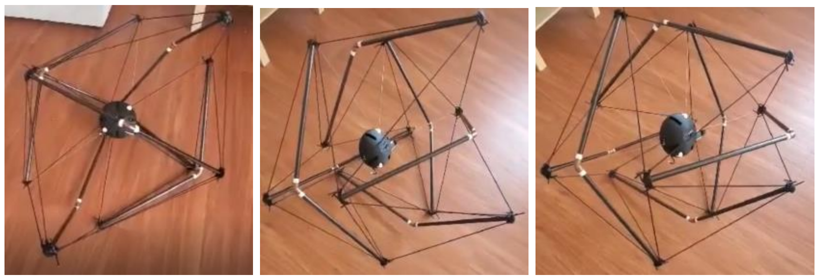
Figure 33: End Cap Design, Version 3

Prior to testing and implementing the autonomous controller, we first tested this new iteration of Flounder to ensure successive motion was still possible. While the initial punctuated motion took a few attempts to achieve as we needed to test if the required encoder counts had changed at all with the new payload design, it achieved successive motion on its first try - with no issues in either the initial motion, or in re-centering the payload. While the motion below does show that Flounder overshot its first step and ended on the open face beyond the closed face it was aiming for, re-centering it returned it to the desired position.



2.0 s 7.0 s 11.0 s 13.0 s

Figure 34: Stage 1 Successive Motion, Enclosed 6 Motor Configuration



1.0 s 7.0 s 15.0 s

Figure 35: Recentering during Successive Motion, Enclosed 6 Motor Configuration

After testing, it was noted that the elliptical shape of the stationary robot's payload was due to the offset of the hooks used to connect the pulleys to the payload - and that the actual connection points formed a sphere. As such, a second iteration of the enclosed payload was designed to create a more spherical structure.

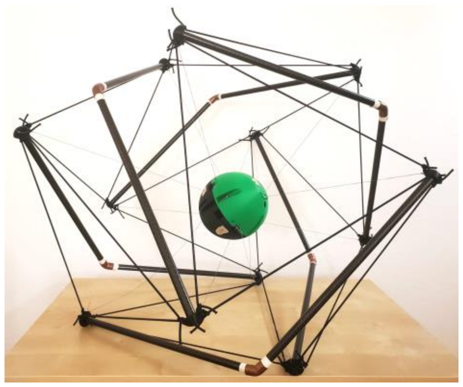


Figure 36: Flounder, Enclosed 6 Motor Configuration

Moving to a more spherical structure also enabled us to achieve greater symmetry with the installation of the motors.

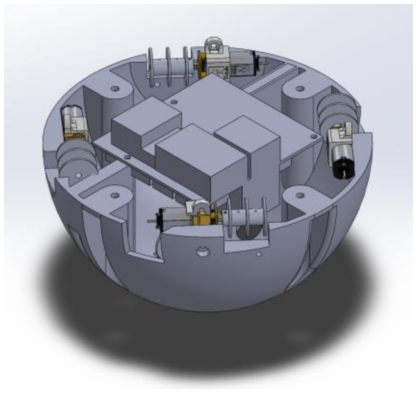


Figure 37: Enclosed 6 Motor Payload CAD, Spherical Version

A third iteration including more user-friendly features, such as access to the control board and improved battery manipulations and improved mounting for the electronics, is currently in the design and testing stages, and can be seen in-use on the robot in Figure 39.

In terms of implementing the low-level autonomous control, we decided that the first step would be to have the robot determine its current position in terms of the closed face it rested upon, and then to automatically determine the best way to navigate to any other selected closed face. The first step was accomplished via an inertial measurement unit, or IMU - specifically, an Adafruit BNO055 Absolute Orientation Sensor. This IMU is able to output its absolute orientation and accompanying gravity vector, which enabled us to map the different closed faces of the robot to their current position and determine which face the robot is currently resting on (Ahmad).

To then move the robot to a user selected face, we started with code written for earlier iterations of the rod-actuated mobile robot design, like MR2. To determine the most optimal path to reach the desired face, we implemented Dijkstra's Shortest Path First algorithm (Dijkstra) (Keen). Specifically, this meant determining the shortest path between any two faces for all eight of the closed faces on Flounder to minimize the number of required instances of punctuated motion.

To set up Dijkstra's algorithm, we defined nodes and edges - with each node representing a closed face (denoted using an alphabetic letter) and each edge (shown with blue line) in Figure 37 below. This mapping represents the physical adjacency between faces and shows the potential paths to get between any two faces. Figure 38 shows the physical mapping of these faces on Flounder.

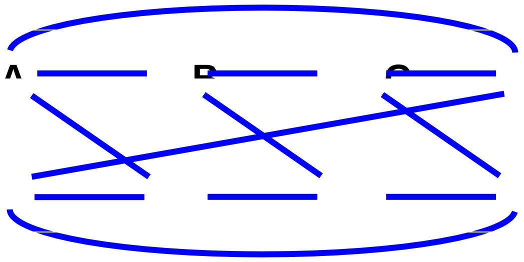


Figure 38: Dijkstra Mapping of Flounder

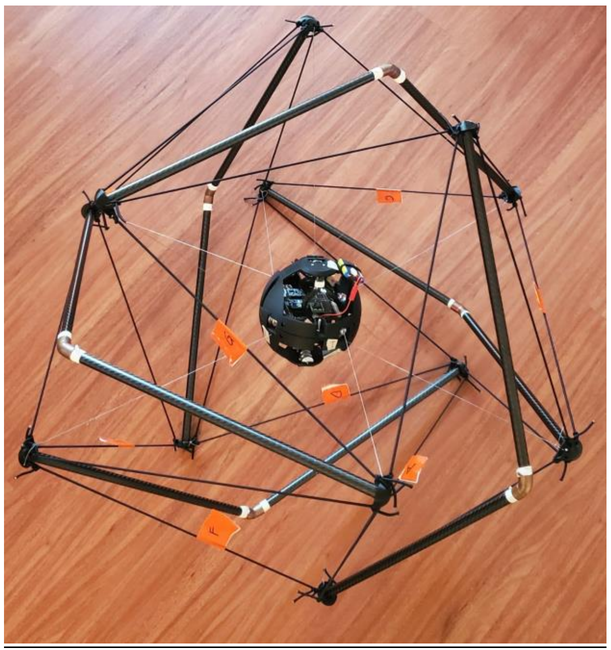


Figure 39: Physical Mapping of Faces for Flounder

In implementing Dijkstra's algorithm, edges between adjacent faces are given a weight of 1, representing that only 1 instance of punctuated motion is needed to travel from one face to the other (note that this implementation of Dijkstra is bidirectional). The cost of each path is determined by the total weight between the two nodes. For example, the cost of going from A to B is 1, while the cost to go from A to C is 2, with the optimal path defined by Dijkstra as A → B → C. The greatest possible weight of the most optimal path between the furthest two nodes, for example from A to H, is 3, with the optimal path defined as A → B → C → H. That said, there are multiple paths of the same weight that can be used to get between these two nodes - which represents the different directional movements in the ground plane. This principle leads to the idea of directional continuous motion, which will likely be implemented in future iterations of Flounder.

When implemented on the actual robot, Dijkstra defines the optimal path with assistance from the IMU to ensure the robot is actually travelling along the intended path. For example, if Flounder is currently resting on face A, the user interface informs the user that Flounder is resting on face A. The user may declare a desired face for Flounder to go to, for example, face C. Dijkstra's algorithm runs and determines the next face, B, and the robot actuates to move to that position. Once the new position is reached, the IMU performs a check to ensure the robot has actually reached face B and, once it has correctly reached that position, actuate again to follow Dijikstra's algorithm from face B to face C. If the IMU detects that a different face than intended has been reached, Dijkstra's algorithm will be rerun to determine the optimal path to go to reach the desired face. Once the IMU determines that Flounder is on the desired face, all motor actuation ceases.

Although outside of the scope of this project, the next steps to improve Flounder's autonomous motion is to implement autonomous directional control algorithmically. Motors would be actuated in the same way as they currently are to induce punctuated motion, except that visual sensors or other external environmental information would be involved to determine the desired physical direction of movement. After this is accomplished, a more advanced controller such as a Model Predictive Controller might be used to further optimize Flounder's maneuverability (Borrelli).

2.4 Optimization of Autonomous Control Using Model Predictive Control

Our team has achieved a specific face-to-face version of autonomous control, but the idea can be extended further to more practical applications, including directionality based control. More generalized autonomous control requires a greater range of sensors, such as cameras, to physically interpret and manipulate the entire system. To achieve this, we can use higher level controls based on minimizing a cost function, while constraining states and inputs. An example of one such higher level controller is a Model Predictive Controller, or MPC. Unlike the algorithmic approach used for controlling face-to-face motion, MPC has the potential to allow Flounder to travel greater distances with minimized energy losses.

Simulating a centrally actuated mobile robot in MATLAB shows that MPC can help Flounder achieve smoother optimal control at a communication frequency of 100 Hz. Once this frequency was determined, the simulation parameters were then adjusted to best match the motion type achieved by Flounders 6 motor configuration. Parameters for the simulation are shown in Table 2, and the actual deformation needed for forward motion by the robot structure is shown in Figure 40, using a 10 second simulation to travel forward a distance of approximately 1.0 meter.

Table 2: Parameters of Central Payload Mobile Robot Simulation

Parameter	Value or Method
Controller	MPC
Simulation Timestep	0.01 s
Total Simulated Timesteps	1000 steps
Controller Prediction Horizon	10 steps
Incline	3 degrees
Simulated Robot Apx. Diameter	0.5 m

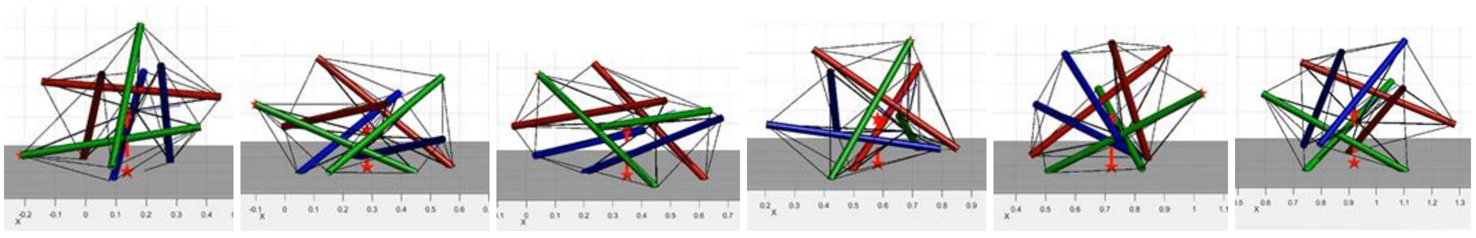


Figure 40: Centrally Actuated Robot Simulation with 0.01 s Timestep

Traveling at an average velocity of about 0.11 m/s, Figure 41 shows the progression of the robot's center of mass (COM) from a bird's eye view - represented by the solid blue line. The red star shows the starting point of the robot, and the dotted triangles show how the robot's external structure interacted with the plane along which it traversed.

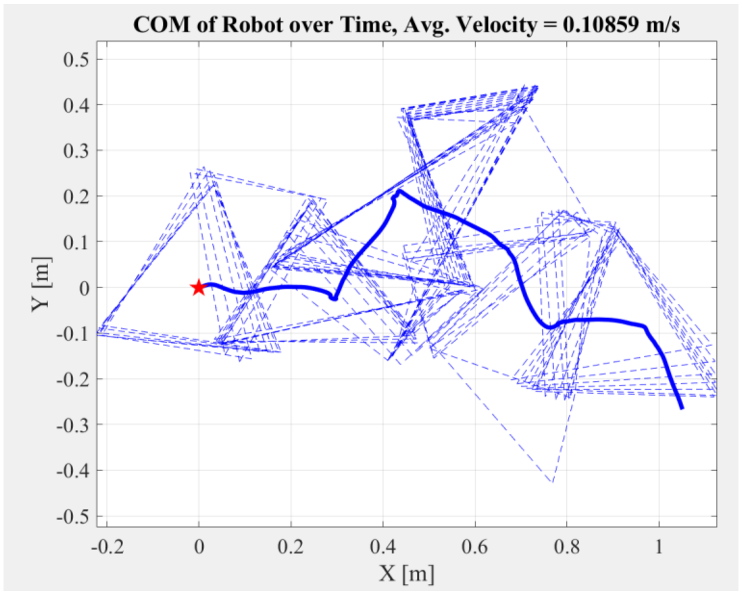


Figure 41: Progression of a Mobile Robot's Center of Mass during a 10s Simulation

Presently, physical motion on Flounder is implemented via a system of tensioning and loosening paired cables to induce motion. Only one pair of motors, and thus, only four cables, are actuated at a time. Using MPC, we would have the possibility of actuating multiple motor pairs simultaneously in order to optimize maneuverability. Figure 42 below shows how cables are tensioned and loosened throughout the 10 second simulation. The top graph shows how individual cables acted, and the bottom graph shows the cumulative trend of tensioning and loosening of cables over time.

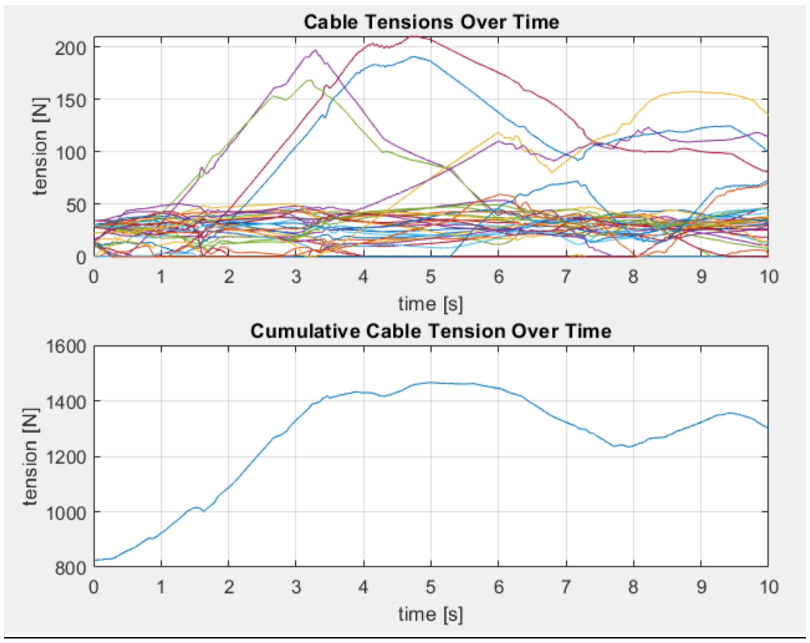


Figure 42: Individual and Cumulative Cable Tension Over Time

3. Squishy Stationary Robot Capstone Team Support

After the successful completion of the current calibration and bed-of-nails tester for MR2 improvements, the team leveraged its mechanical design skills to support the teams working on projects with Squishy Robotics' stationary tensegrity robots. Specific goals included expediting the process of developing physical prototypes for different tasks.

3.1 Electronics Enclosure for Robot Transport Drone

Squishy Robotics deploys drones to transport its tensegrity robots to disaster locations, as well as to drop the robots successfully into the disaster zone upon arrival. One key area of importance that has guided development of drone accessories is the accuracy of drop targeting of the robots. The current system requires a mountable camera, Raspberry Pi controller board, and a battery for the system. We were tasked with helping to design an enclosure that could securely hold the Raspberry Pi and battery to the transport drone.

Multiple locations for this enclosure were considered, including the arms of the drone used to carry the robot, the top surface, and the lower rods that support the main body. With considerations such as the minimum additional weight necessary, the safety of the electronics, and the robustness of the enclosure, we decided to utilize the pre-existing wooden board that connected with the supporting rods of the drone (see Figure 43 below).

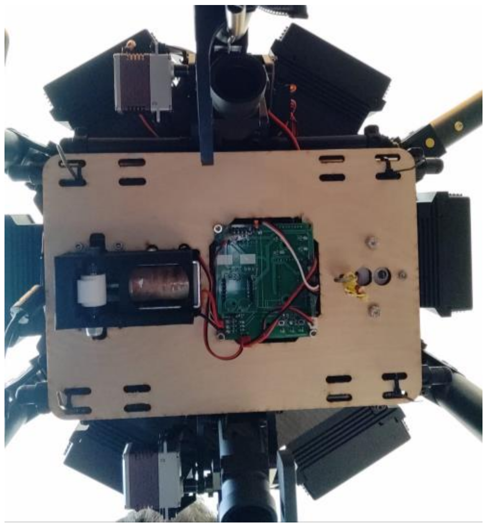


Figure 43: Board used to mount the enclosure

The figure above shows the mounting placement of the electronics to the drone. From this placement, we designed and 3D printed an enclosure to protect the electronics from the elements (see Figure 44 below).

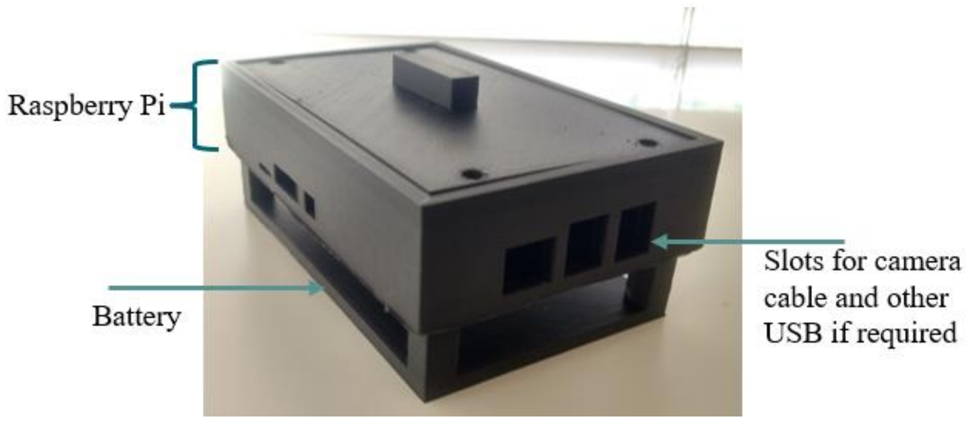


Figure 44: Raspberry Pi Enclosure

3.2 Assistive Tool for Assembling and Disassembling the Stationary Robot

In the current iteration of the stationary tensegrity robot, the cables in the structure are highly tensioned springs - attached to the end caps of the tensioned carbon fiber rods (see Figure 45 below).

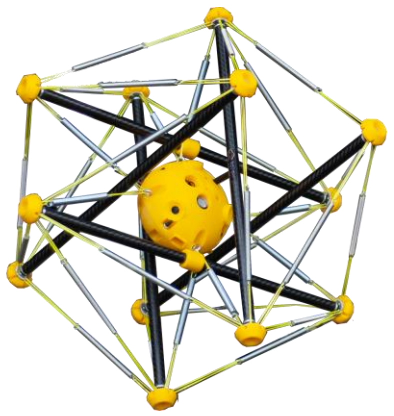


Figure 45: Stationary Robot

The high tension of the structure results in increased difficulty in the assembly and disassembly process of the robot - requiring significant strength to properly insert or remove the rods. As not every person assigned to build robots was capable of achieving this consistently, we were tasked with designing and building an assistive tool that could reduce the force required to insert or remove the rods. The first prototype developed by the team was a handle-like tool (see Figure 46 below) that could be placed between the rod and the end cap. It provided a better grip to increase the leverage of force applied to the end cap to allow the user to remove the rod with their free hand.

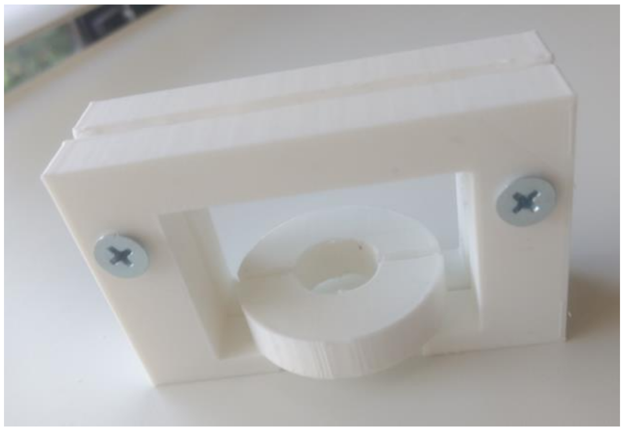


Figure 46: First Design for Robot Assembly/Disassembly Assistive Tool

This tool eased the process slightly - but failed to meet the initial goals of reducing the total force required by 50%. The team is currently in the brainstorming phases to see if development of a different mechanism is possible, with potential options including lever action, gears, and a pump action jack. Given how the final design requirements request a tool that is small enough to easily use, store, and carry, the team is currently pursuing replicating an enhanced plier design by exploring custom parts that could be attached to existing pliers.

Works Cited

"24.1. Tkinter - Python Interface to Tcl/Tk." *Python 2.7.18rc1 Documentation*. Python. N.d. Web. docs.python.org/2/library/tkinter.htm. Accessed 10 Apr. 2020.

Ahmad, Norhafizan, et al. "Reviews on various inertial measurement unit (IMU) sensor applications." *International Journal of Signal Processing Systems* 1.2 (2013): 256-262.

Bonstell, Nick. "Four-Gas Monitoring for the Fire Service". *FireHouse*. 1 May 2017. Web. https://www.firehouse.com/rescue/article/12319881/fourgas-monitoring-for-the-fire-service. Accessed 12 Apr. 2020.

Borrelli, Francesco, et al. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.

Coldeway, Devin. "How Squishy Robotics Created a Robot that can be Safely Dropped out of a Helicopter". *Tech Crunch*. 18 Apr. 2019. Web. https://techcrunch.com/2019/04/18/how-squishy-robotics-created-an-robot-that-can-be-safely-dropped-out-of-a-helicopter/. Accessed 12 Apr. 2020.

Chen, Lee-Huang, et al. "Modular Rod-centered, Distributed Actuation And Control Architecture For Spherical Tensegrity Robots" patent WO 2017/083534 Al, November 2016

Chen, Lee-Huang, et al. "Soft Spherical Tensegrity Robot Design Using Rod-Centered Actuation and Control." *Journal of Mechanisms and Robotics*, vol. 9, no. 2, 2017, doi:10.1115/1.4036014.

Dijkstra, Edsger W. "A note on two problems in connexion with graphs." *Numerische mathematik* 1.1 (1959): 269-271.

Gaughan, Ben, and Alice Agogino. "Episode 54: Squishy Robotics Shape Shifting Robots." *Smart Firefighting*, Smart Fire Fighting Community, 14 May 2019, www.smartfirefighting.com/episode-54-squishy-robotics-shape-shifting-robots/. Accessed 4 Mar. 2020.

Holdeman, Eric. "Technology Plays an Increasing Role in Emergency Management". *Emergency Management*. Government Technology (GT). 26 June 2014. https://www.govtech.com/em/training/Technology-Increasing-Role-Emergency-Management.html. Accessed 4 Mar. 2020.

Hudson, Greg. "What is a Squishy Robot - and How can it Save Lives?" *Changing America*, 2 Jan 2020. https://thehill.com/changing-america/resilience/natural-disasters/478787-what-is-a-squishy-robot-and-how-can-they-save. Accessed 4 Mar. 2020.

Kalman, R. E. "A New Approach to Linear Filtering and Prediction Problems." *Journal of Basic Engineering*, vol. 82, no. 1, Jan. 1960, pp. 35–45. *ASME Digital Collection*, doi:10.1115/1.3662552.

Keen, Ben. "Implementing Djikstra's Shortest Path Algorithm with Python." *Implementing Djikstra's Shortest Path Algorithm With Python*, 11 Jan. 2017, benalexkeen.com/implementing-djikstras-shortest-path-algorithm-with-python/. Accessed 16 April 2020.

Mysore, Gurudatt D., et al. "A Microcontroller-Based Bed-of-Nails Test Fixture to Program and Test Small Printed Circuit Boards." *Proceedings of the IEEE SoutheastCon 2006*. IEEE, 2006.

NFPA Fire Experience Survey. “Fire Department Calls”. *National Fire Protection Association.* Nov. 2019. https://www.nfpa.org/News-and-Research/Data-research-and-tools/Emergency-Responders/Fire-department-calls. Accessed 21 Apr. 2020.

Plett. Gregory L. “ Equivalent Circuit Cell Model Simulation - Coursera”, https://bit.ly/2y7iFVv Accessed 30 Mar. 2020

Regan, Tim, et al. “Current Sense Circuit Collection - Making Sense of Current”, December 2005

Squishy Robotics. “The Technology”. *About Squishy Robotics*, Squishy Robotics. n.d. Web. https://squishy-robotics.com/about.html. Accessed 12 Apr 2020.

Wu, Hang, et al. "PID controllers: Design and tuning methods." *2014 9th IEEE Conference on Industrial Electronics and Applications*. IEEE, 2014.

Ziegler, S. et al. "Current Sensing Techniques: A Review," in *IEEE Sensors Journal*, vol. 9, no. 4, pp. 354-376, April 2009.