Final Report

Senior Design Group 4

The Tensegrity Tumbleweed

Steven Hyland - smh2265

Breanna Johnson - blj2127

Leonardo Costa - la2700

Luke Gile - lg2966

Columbia University

Department of Mechanical Engineering

Due May 10, 2019

Contents

1 Executive Summary

In approaching the unique design challenges that the tensegrity robot, or Tumbleweed, posed, we as a team came up with several different approaches and spent the last two semesters refining, editing, and testing different methods, materials, and techniques. To better understand the complexity surrounding the tensegrity robot, we immersed ourselves in literature from Berkeley, NASA, and even the advice of our own home institution professors. We developed an initial plan and began designing and testing. We began with simple straw and rubber band models, scaled up to PVC pipe and nylon, and finally focus on analyzing our Mark III design made from aluminum struts and Kevlar cable actuation with DC motors and an elastic lattice for symmetric tension. This report will lay out the engineering analysis performed by the team, the concept design generation and iterations through the design process and overall final structure. We used MATLAB, NTRT (NASA Tensegrity software), SolidWorks and other tools to provide us with information regarding our design and validation for the choices we have made during this project. The results also revealed aspects of our robot that could be improved in order to achieve our final goal. Specifically, our goal statement is to build a robot that is lightweight and durable which can traverse short distances using a tensegrity structure and wireless actuation.

2 Introduction

Tensegrity, or tensional integrity, is a structural principle that utilizes isolated components in compression that exist inside a net of continuous tension such that the compressed members do not touch one another. Typically, soft members are used as cables and hard members are used at struts. This structural principle allows for extreme flexibility and ability to maintain high impacts. Although tensegrity structures began as artistic exhibitions, this principle is crucial to the advancement of space exploration. Tensegrity structures are very resilient, economic, mass efficient and light weight. There exists no critical point of load concentration due to distributed force allowing for the ability to form into a compact area and withstand high impacts. These are just a few of the many beneficial characteristics of these structures.

Tensegrity structures vary drastically in size and number of compressed strut members. Our design is composed of six struts which allows for advanced locomotion and six areas for stability and actuation with high load bearing capacity. To ensure constant and equal tension to each strut, we utilized a rubber lattice of known spring constant and size. Our actuated tensegrity robot aims to move via tacking or rolling, and withstand high impacts when dropped. Specifically, we aim to drop the tensegrity from a height of 6ft and have it roll 10 feet using actuation to shift its center of mass. The multidisciplinary nature of this project allows for necessary advancement in understanding the complexities of space exploration.

3 Background Information

For the purposes of our project we are concerned with tensegrity and robotic applications in order to create an actuated tensegrity sphere of six struts capable of moving and avoiding obstacles. We found that the NASA Super Ball Bot is an actuated tensegrity robot that has been under research since 2013 with an initial Titan mission plan [1]. Moreover, research on the current Berkeley Emergent Space Tensegrity (B.E.S.T) Laboratory revealed alternative designs that use an outside mesh of stretched rubber-like material in addition to the cables that connect the struts. Some of their research shows progress in steep angle climbing motions and a most recent Laika quadruplet robot with a tensegrity trunk [2]. Lastly, the Creative Machines Lab at Columbia University also worked on tensegrity robots. Professor Hod Lipson was analyzing the possibility of achieving motion in multiple degrees of freedom by using one actuator, due to the mechanically redundant design [3]. Further investigation of all previously mentioned projects led to research regarding what design and material compositions were the most important for building the robot, what types of motors were used, and efficient motion algorithms to allow for quick robot movement. Figure 1 below shows NASA's Super Ball robot design.

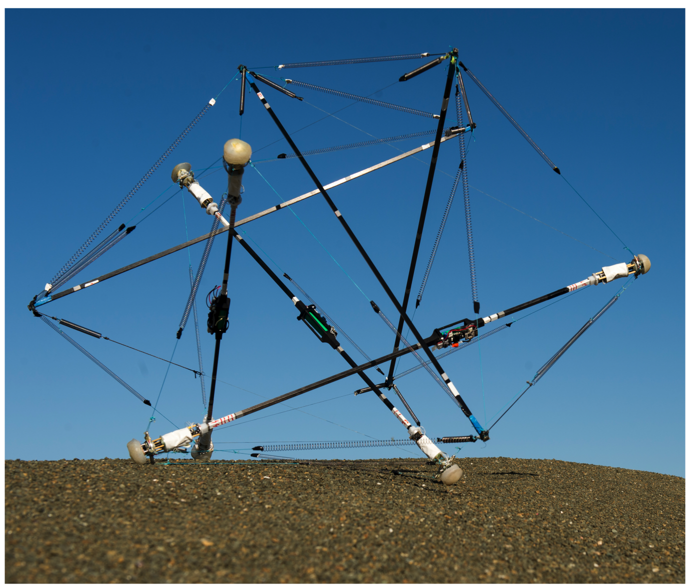


Figure 1: SuperBall Bot

Figure 2, below, shows the BEST lab's tensegirty robot that is capable of complex uphill movement. It also shows the rubber lattice technique that allows for symmetric compression throughout.

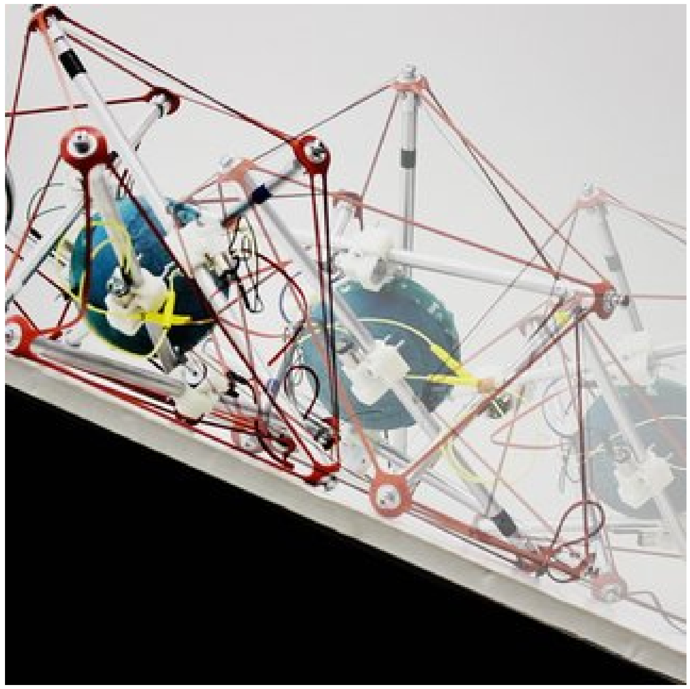


Figure 2: BEST Robot with rubber lattice

4 Design Details

4.1 Concept Generation

Different design approaches were considered when deciding upon a final robotic system. Originally, our design consisted of hollow PVC pipes with hand tied nylon cables to induce tension. This design revealed many issues with symmetry and equal tension to each strut but served primarily as a proof of concept, indicating what areas we would have to focus on going forward. The two foot long PVC pipes also revealed issues with overall scale and feasibility of actuation. For this setup, we created 3-D printed end caps to house batteries, an Arduino board, a servo motor, and attached spool. Due to the larger size of our strut/cable system, the end caps had to also be large enough to house a larger motor and corresponding batteries. After building the entire tensegrity structure with one strut fitted with an end cap, we realized that our design needed to be more compact. In addition, we also realized that to facilitate programming of micro-controllers, it would be helpful to have a wireless connection to the board, as well as boards that were easy to communicate among themselves.

Our second design consisted of smaller aluminum rods for struts and a laser cut rubber lattice [4] to ensure equal compression on each strut instead of nylon cables. This design also includes a new concept for the 3-D printed caps that house the actuation materials. The new compact cap design is now fitted to live in the middle of the aluminum strut. The caps have slots for two 3.7V batteries connected in series, a particle Photon board and a small geared (150:1) DC motor. The particle photon board is significantly smaller than the Arduino Uno and Wifi capable, which makes it easier to program, control and connect in real time. We also changed the motor from our preliminary design to reduce the overall weight and abide by torque requirements. As a trade-off, the speed has decreased in order to achieve the necessary size, weight, and power. An overview of the calculations for torque, rpm and power requirements is outlined in the upcoming sections.

Throughout the project, there were many iterations of the midcap part, since that is the most vital component in our project. This part houses the batteries, circuits, motors and serves as the main structure of our struts. Figure 3 below shows the different designs considered and the progression of our midcaps when designing the tensegrity sphere.

Figure 3: Midcap Design Progression

From left to right, our design changed to fit the different components in a tight and organized manner. Our first set of designs had only space for the Particle Photon board, without accounting for the H-bridge and voltage regulators - something we realized were necessary upon testing the first prototype. As the project progressed, we also discovered the necessity to increase our filament density, making our 3D printed part more resistant to impact. The dark green part shown above had 20% infill, while the light blue had 80%, which survived the preliminary drop tests. Following that, our circuitry and battery packages needed a larger space for accommodation. The remaining designs aimed to find the perfect balance for the geometry of the Tumbleweed, as well as an aesthetically pleasing design that could be used for our demonstration. The final midcap design included a deep slot for the batteries to be stacked on top of each other, a large slot for the wire-wrapped circuit, which will be introduced in the upcoming sections, and a through slot to fit the motor with an attached bracket. It also included a hole that connected the battery and the circuitry slot for wire management, which was important for easiness of assembly and compactness of design. A close up of our final midcap rendering and assembly are shown below as Figures 4 and 5, respectively.

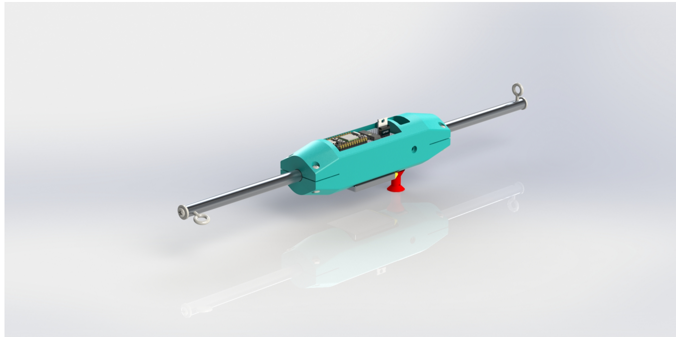


Figure 4: Final Midcap Solidworks Rendering

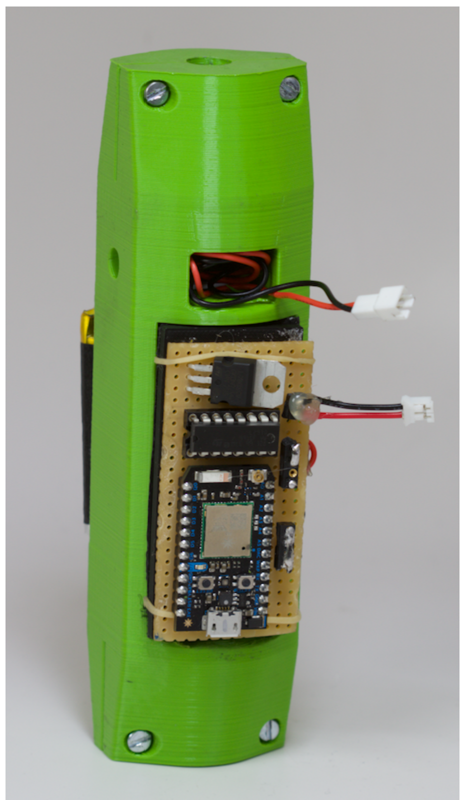


Figure 5: Final Midcap Assembled

4.2 Design Overview

A CAD design was created to illustrate and analyze the structure of our project. As mentioned before, our robot is constructed of six struts. Each strut is built with two 5.5 inch aluminum rods attached by the ends with a 3D printed motor holder in the middle. On the holder, there are three slots (one through-all for the motor, and two larger slots for the micro controller and battery pack). The motor has an attached spool, to which we have Kevlar string connected for actuation. A final addition to our design is the addition of eye hooks at the outer edges of the aluminum rods to facilitate the Kevlar connections and act as guides.

All six struts are held together by an elastic neoprene rubber lattice that envelops the assembly. Our rubber lattice is a single piece with both triangular and circular cutouts. The circular cutouts are where we insert the aluminum rods, followed by a washer and screw to secure the lattice to the rods. Many width and length combinations were experimented upon to find which combination gave our robot enough support while allowing for enough flexibility upon impact. Below is an image of the finalized neoprene lattice after being laser cut to size, shown as Figure 6.

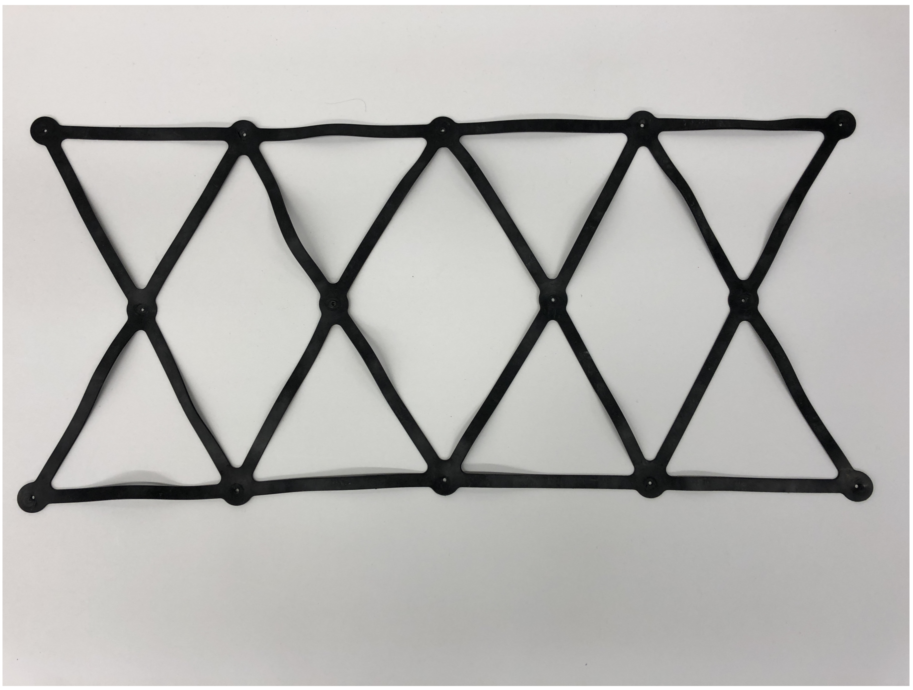


Figure 6: Final Neoprene Lattice

Additional parts include the spool, which is attached to the motor shaft and has a through all hole for the Kevlar string to be attached. We also have mounting screws on each strut that secures the aluminum rods to the motor holder part. Figure 7 below shows the full assembly.

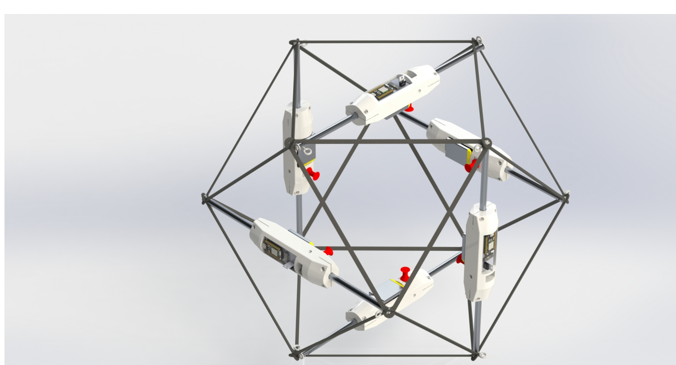


Figure 7: Tensegrity Six-Strut Assembly

The CAD drawings with their respective dimensions can be found in the Appendix under Figures 19 and 20. A sub-assembly of each strut can also be found in the Appendix as Figure 21.

4.3 Electronics

For our electronic components, we used the Particle Photon board, a Wi-fi enabled device that made it easier to connect each micro controller without any wires running along the length of the strut. These boards are also know to be lightweight and they have a user friendly interface, which was beneficial for the programming and controlling of our prototype. In addition to the board, we also needed to control the direction of motion of the DC motor. Multiple options were considered,among different motor shields and motor drivers, but finally we opted for creating our own circuit, in order to constraint size and functionality to what we required. To do that, we used a L293D H-bridge that was wired directly to the micro controller. However, the Photon board only has an output voltage pin of 3.3V, unlike other micro controllers such as the Arduino Uno and a Raspberry Pi - which both have a 5V output logic pin. Therefore, an additional voltage regulator component was used to reduce the voltage from the battery pack to 5V to be used by the logic gates of the H-bridge. Figure 25, in the Appendix, shows the photon schematic and Figure 8, below, shows the schematic for motor control.

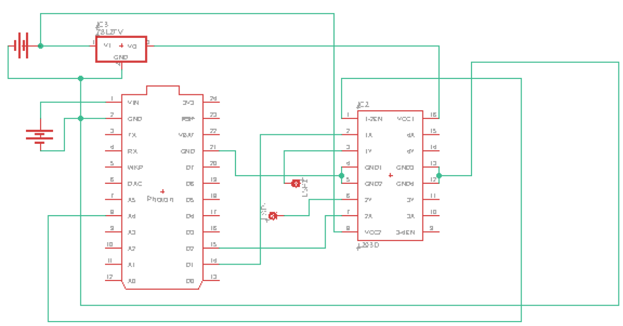


Figure 8: Motion Control Circuit

At first, we tried to design a PCB (printed circuit board) that could integrate all of those components together. The first design constituted of a double sided board. However, after a few manufacturing tests, it was concluded that double sided PCB's would be hard to manufacture due to lack of precision machining tools. A single sided PCB was then designed as an alternative trying to minimize the number of operations required to make our circuits. However, the same problem persisted. Due to small circuit components and space constraints, the house-made PCB's would be a problematic part of manufacturing. Lastly, wire wrapping on perforation boards were considered, and upon preliminary testing, the connections seemed strong enough for the purposes of our project. It also allowed the group to maneuver components around, decreasing the space required for the circuits, but still maintaining some organization. Hot glue was used to secure wires and components to the board, while velcro strips were used to connect the circuit assembly to the midcap. Figures 23 and 24, found in the Appendix show the PCB designs in EAGLE. Figure 9 below shows a closeup of the circuitry attached to the midcap.

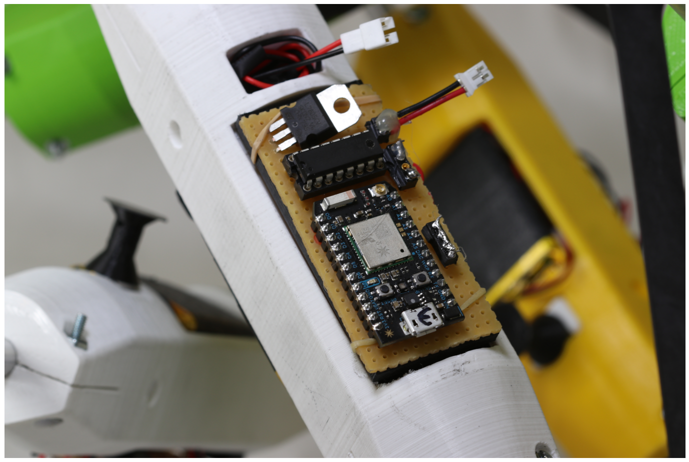


Figure 9: Wire-wrapped Circuit in Midcap

For our batteries, our original circuit consisted of two 3.7 Li-po batteries in series that would power both the motor and the micro controller, but upon preliminary testing, it would be better to separate the batteries for each component. Hence, the group decided to use 3.7 batteries to power the micro controller, while 9V batteries were used to power the motor. One important remark is that for easiness of assembly, JST connections were used to connect the circuits to the batteries. That way, all the part components can sit in the midcap disconnected until motion is required. There was one JST connection for each battery and one used for the motor (See Figure 9).

5 Engineering Analysis

The applicable methods we used to differentiate between competing designs includes quantitative measures of weight, strength, and power while staying under the allocated budget. We began our analysis by performing hands-on calculations for sizing of the motors and analyzing the effects of the torque output on our assembly. We performed bending stress analysis to validate the spool design, motor specs and material selection. The full calculations, as well as Figure 22 with motor specifications can be found in the Appendix. With these calculations and information, we were able to determine a model for the motor we expected to use and add that to our cost analysis and parts list.

Using MATLAB, we modeled the tensegrity sphere by plotting elements and connecting nodes for visualization. Furthermore, we performed form-finding calculations, which helped define and conceptualize the geometry of our robot [6]. Below, Figures 10 and 11 show the relationship between strut height and horizontal distance from the center of mass. This relationship reveals how our structure form can be obtained through mathematical derivations and analytical solutions. With a ratio of approximately 0.86, we were able to define the necessary dimensions for future prototyping. Because our current ratio is close to 1, which is larger than what the derivations suggest, the rubber lattice acts as a pre-tensioned spring, holding the structure together and avoiding sudden collapses.

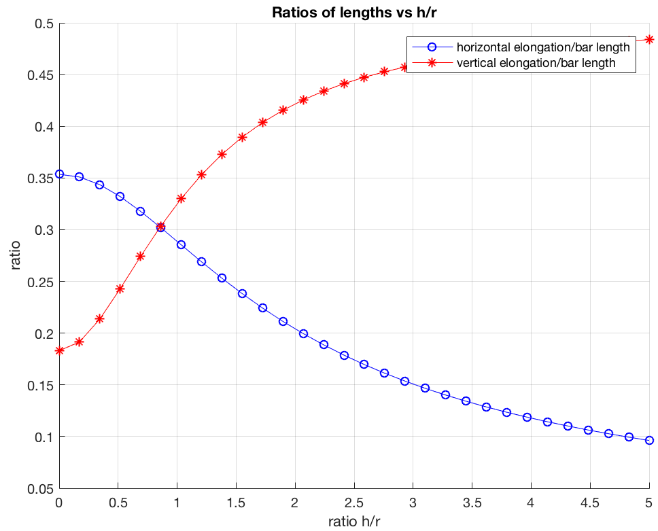


Figure 10: Ratio of Strut Height and Horizontal Distance to Center of Mass

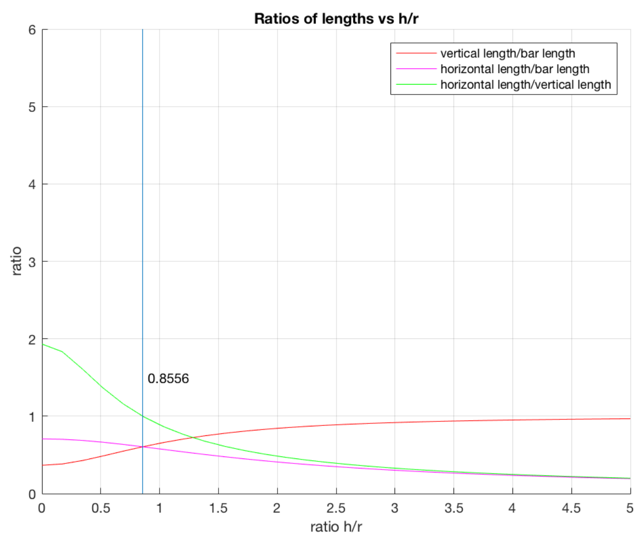


Figure 11: Form Finding Data from Complete Compression to Equilibrium State

In addition, we also performed programming simulations to develop some insight about the motion of our robot. NASA has a library called NASA Tensegrity Robot Toolkit (NTRT) that allowed us to model the struts and cables in a 3D environment and control the movement by applying component stimuli through mouse movements. We found this to be a useful tool in designing this robot, since NASA's literature is very organized and structured, which made it easy for us to adapt to our design. The image bellow is a screenshot of the interactive model that we accessed. In Figure 12, the yellow and pink tubes refer to the struts, while the red lines refer to the cables that hold the structure under tension. In the case for our design, the cables are actually the rubber lattice as previously mentioned.

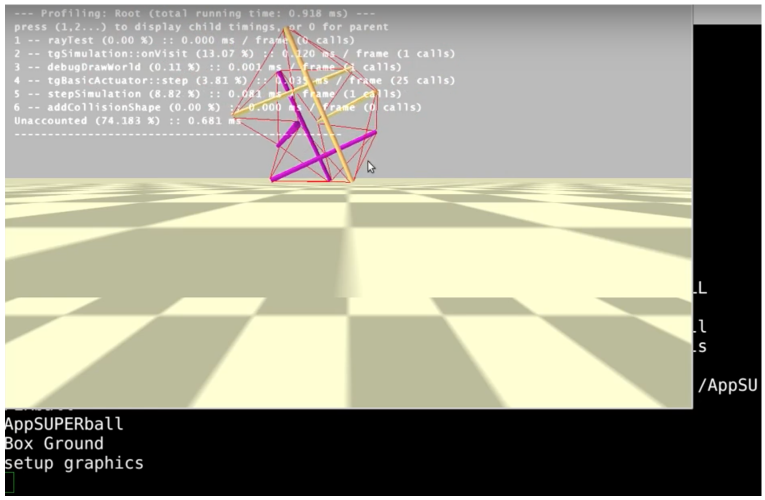


Figure 12: NTRT Simulation model

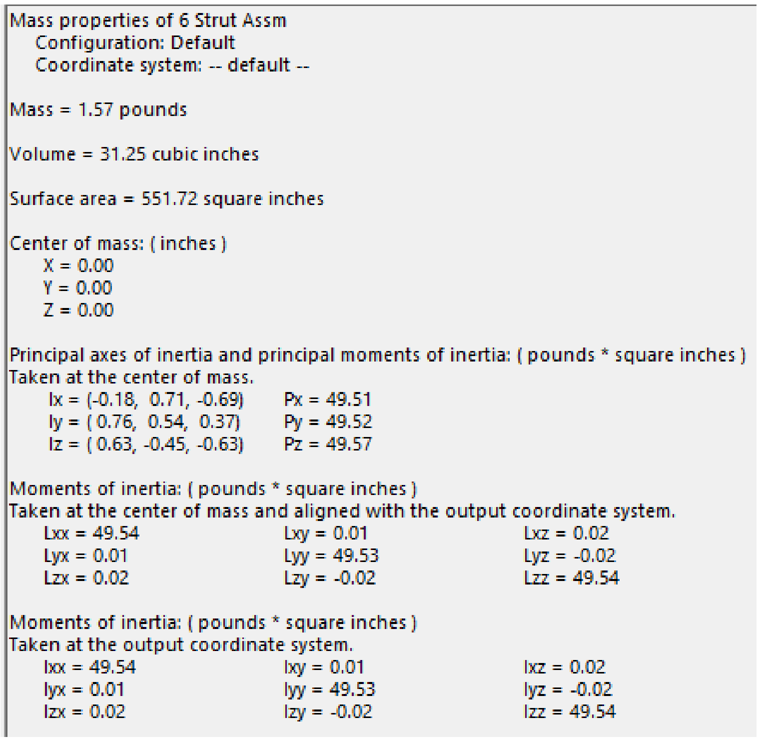
For the rubber lattice, there were also some geometry calculations that were done to come up with a value for the length and thickness of each equilateral triangle that makes the structure of the lattice. Based on previous research and material data, the team estimates that the design is viable and operational. Figure 13 shows a table from the Berkeley BEST lab illustrating the different spring constant to width relations for 60A Durometer Neoprene rubber. The outlined stress analysis is similar to the studies conducted in Fabrication and Analysis of Tensegrity Based Prism structure [5]. We were able to use this research to determine exactly the size and gauge of rubber that would work best for our purposes.

Width (mm)	Spring Constant (N/m)	±Error (N/m)
6.35	986	24.52
7.94	1472	35.56
9.53	2104	55.96
11.11	2364	56.50
12.70	2812	67.75
14.29	2973	68.22

Figure 13: Width to Spring Constant Relations for 60A Durometer Rubber [5]

5.1 Experiments & Test Results

Now that the final design for our second prototype is finalized, the next steps are to run experiments on the different structural components of our robot. This includes tensile tests on the aluminum rods and rubber lattice, as well as motion tests and circuitry/battery tests. Alongside this, we also extracted mass and volume parameters for our second prototype from SolidWorks. Below, figure 14, reveals the volume and mass specifications for our system.



Mass properties of 6 Strut Assm
Configuration: Default
Coordinate system: -- default --

Mass = 1.57 pounds

Volume = 31.25 cubic inches

Surface area = 551.72 square inches

Center of mass: (inches)
X = 0.00
Y = 0.00
Z = 0.00

Principal axes of inertia and principal moments of inertia: (pounds * square inches)
Taken at the center of mass.
Ix = (-0.18, 0.71, -0.69) Px = 49.51
Iy = (0.76, 0.54, 0.37) Py = 49.52
Iz = (0.63, -0.45, -0.63) Pz = 49.57

Moments of inertia: (pounds * square inches)
Taken at the center of mass and aligned with the output coordinate system.

Lxx = 49.54	Lxy = 0.01	Lxz = 0.02
Lyx = 0.01	Lyy = 49.53	Lyz = -0.02
Lzx = 0.02	Lzy = -0.02	Lzz = 49.54

Moments of inertia: (pounds * square inches)
Taken at the output coordinate system.

Ixx = 49.54	Ixy = 0.01	Ixz = 0.02
Iyx = 0.01	Iyy = 49.53	Iyz = -0.02
Izx = 0.02	Izy = -0.02	Izz = 49.54

Figure 14: Mass and Volume Data for Assembly

Moving forward with the project, we assembled the components in a modular fashion, which allowed for easy replacement in case of local failures. The six strut robot was assembled and motion coded were tested to determine motion sequences and further areas of improvement. Quantitatively, the results of our experiment helped us determine the precise thickness of our elastic lattice, power requirements and motor requirements. This is primarily where the benefit of testing was most apparent as the aluminum and 3D printed parts will be so overbuilt that the testing is more for completion than for evaluation. Aluminum is the obvious choice in this case over PVC given the weight savings and ease manufacturability as well as strength.

5.2 Actuation Method and Sequence

Actuation is achieved by reeling in cables, which are tied around a spool-motor assembly. When the cables are pulled together, two struts come together and the center of mass shifts, causing the robot to tumble from its original base stance to another base stance. The cable is then reeled out, releasing the tension on the assembly. That process is repeated cyclically, which projects the robot in a forward trajectory. Figure 15 and 16 show the spool reeling the Kevlar string and the struts coming together as a result.

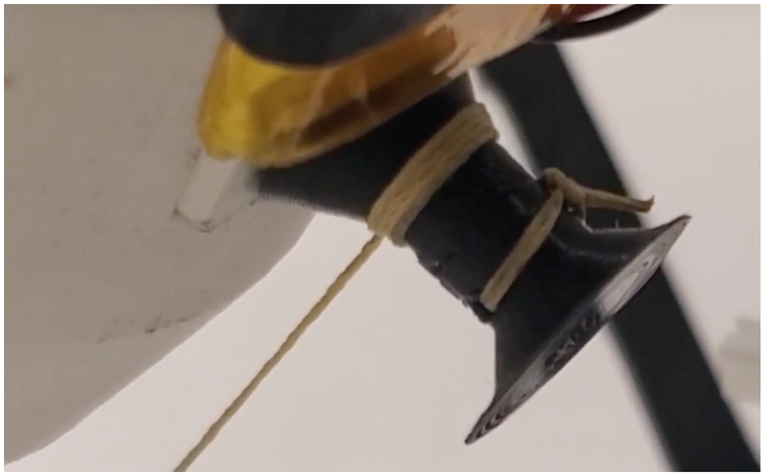


Figure 15: Spool Reeling In Cables

Figure 16: Strut Ends Coming Together

After preliminary actuation testing, the team noticed that the most effective way of shifting the center of mass and moving the robot was to pull on the strings at the base triangle. The base triangle is defined by three ends that touch the ground. The rubber lattice contains eight closed equilateral triangles, which means there are eight possible bases. In order to automate motion, the base and the relative motor necessary to move the assembly - at that respective base - were hard-coded in a motion sequence. Because there are a total of six motors in our project, there were also six possible positions for actuation. Each position was configured to follow a sequence from one to six, in which motor one moves from position one to position two, motor two moves from position two to position three and so forth.

One of the features of the micro-controller selected for this project is the Wi-Fi capabilities. In addition to being wireless, the Photon board also allows for a publishing and subscribing scheme. An off-board controller, publishes commands to each Particle Photon board, which is associated to a motor. Each Photon has its own topic, where they receive commands, such as "f" for forward motion, "b" for backward motion, "s" for stop or "move" for a sequence of forward, followed by backwards motion dictated by a timer. The controller is also able to set the timer for all of the motors, which allows for faster iteration for testing purposes. All the publishing commands are pushed to a database within the Particle website. It has displays real-time events and the message that has been sent. Figures 17, below, shows the Particle Console that holds the commands published. All code blocks can be found in the Appendix as Figures 26, 27, 28, 29, 30, 31, and 32.

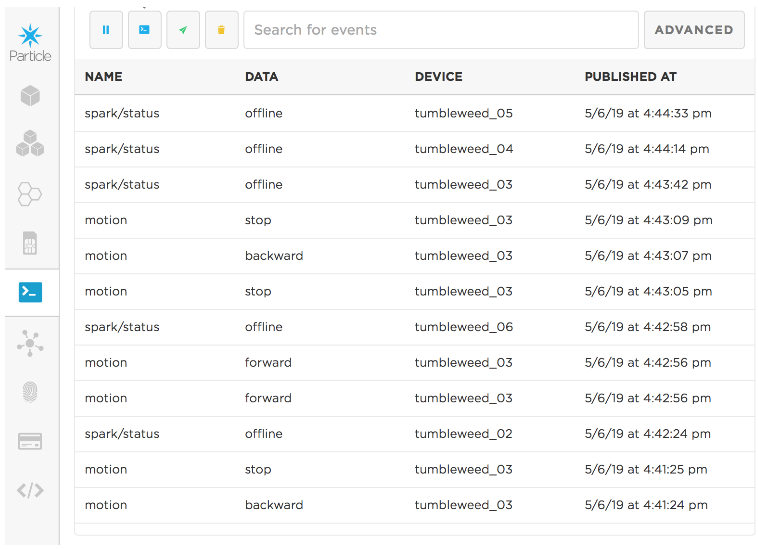


Figure 17: Particle Database Interface

6 Future Steps

Although the main goals of the project were achieved in our final design, there were many more layers of implementation that could be considered to upgrade the current design. This section will go over some of the next steps the group would take in this project. The addition of a feedback system could help move the control algorithm from an open loop control to a completely closed loop control, in which no controller would be required for moving the robot. Furthermore, different actuation commands could be investigated to determine more optimal motion sequences, making the overall motion faster. In terms of design, due to space limitations, our design goal was to make the robot as compact as possible, but larger models - closer to the SuperBall Bot dimensions - could also be a good alternative for better packaging, organization and addition of new features. Lastly, other materials can be used to test how the robot would respond to different properties.

6.1 Codes and Standards

The standards we researched were fairly nebulous given that Tensegrity for space travel is still a novel concept. Although there are no codified standards, we imagine that there are suggested guidelines for weight to power ratios so as to maximize mobility. In terms of weight, if the Falcon Heavy rocket were to take a batch of tensegrity robots to the moon of Titan, it could carry 3,370kg of payload. This means that about 150 tensegrity robots could be transported with housing and deployment technology.

In addition, the Team followed standards for ASME - American Society of Mechanical Engineers - in terms of units and measurements used in this project. All additional parts, such as screws and nuts, follow the standard imperial guidelines. The produced engineering drawings also follow ASME drawing rules and use common notations for indicating features in each part. In terms of electronics, the team followed the conventional standards for circuit nomenclature and symbols. The batteries used were also standard values suggested by IEC - International Electrotechnical Commission. The necessary safety measures were also included in the circuits during the its manufacturing.

7 Conclusion

Space exploration technologies are an important topic of discussion for engineering innovation. New ideas and designs are required to adapt to the conditions of different mission. Tensegrity has shown potential as a lightweight, durable and cheap option for space missions. Our project aimed to validated the usefulness of tensegrity structures as a means for space exploration. Our design displayed many interesting features of this technology and successfully achieved the desired goals for the project.

After manufacturing and testing three tensegrity designs, our group finalized the robotic structure by choosing the most effective option, able of performing all of the desired features. The final design is able to move 3 feet per cycle, at a speed of around 2 feet per minute. It also sustains impacts of around 3 to 4 feet without failures. Lastly, the robot is able to fully compress for easy locomotion and storage. In summary, This design allows for easy actuation and capabilities to withstand high impact loads when dropped. Another useful feature of the final design is the modular capacity of parts which makes it simple for replacing parts. The link to the website, which contains more information on the project and all the pictures taken is listed here: https://tensegritytumbleweed.wordpress.com/

7.1 Acknowledgments

We would like to thank Dr. Josh Browne and all of the Senior Design teaching assistants for their guidance and mentoring throughout this two semester process. We would also like to thank the various Columbia department of mechanical engineering faculty for their help and aid during both the research and implementation phases. This project would not have been possible without the continued support from members of the undergraduate laboratory, Bill Miller, Bob Stark and Andrei Shylo.

8 Appendix

8.1 CAD and Hardware Info:

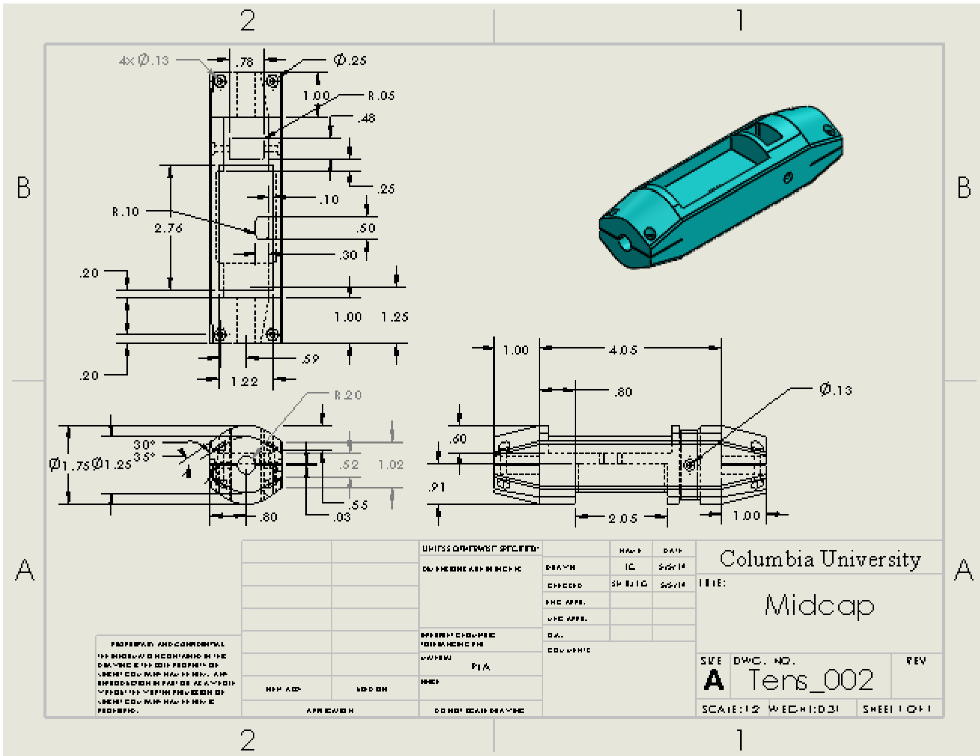


Figure 18: Midcap Drawing

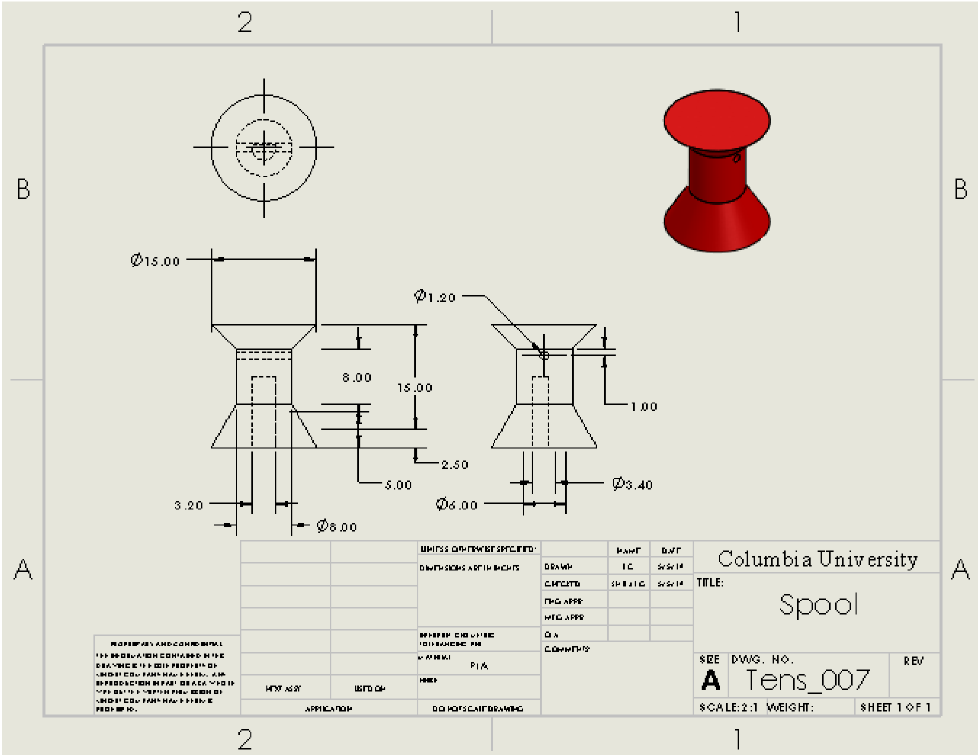


Figure 19: Spool Drawing

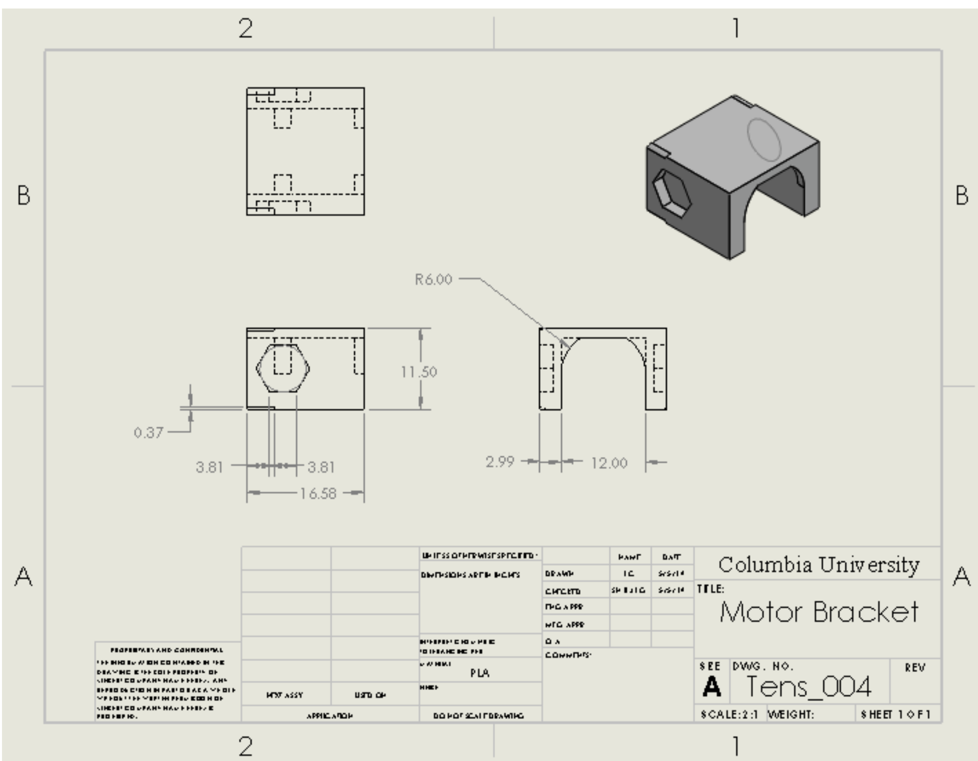


Figure 20: Motor Holder Drawing

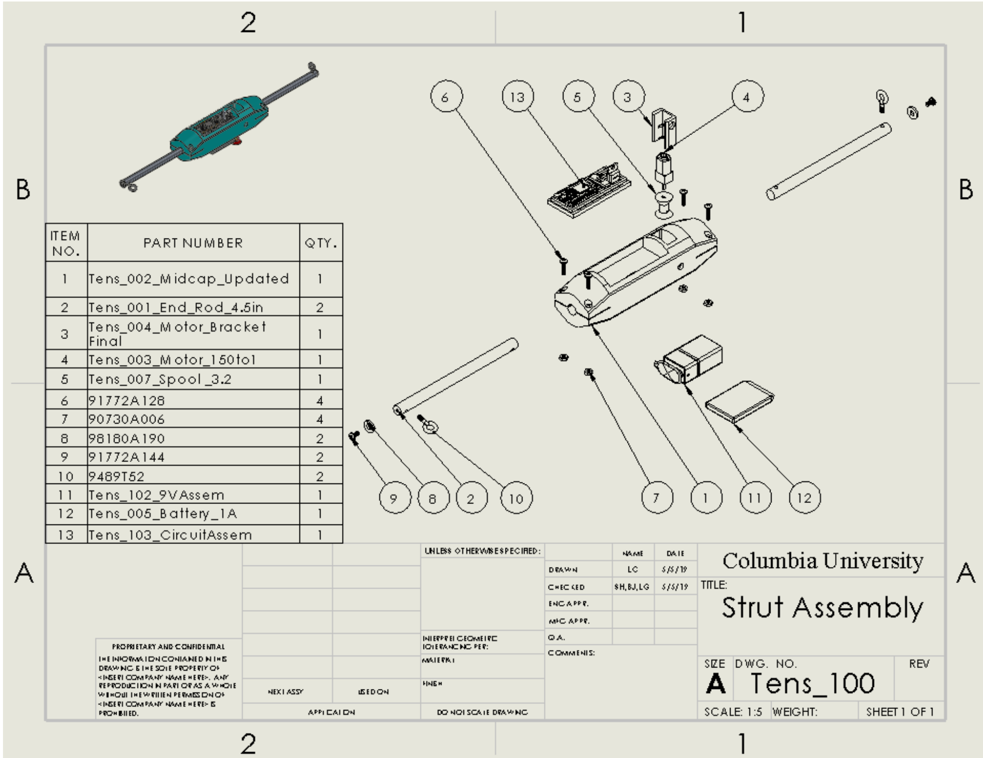


Figure 21: Midcap Strut Assembly

Hands On Calculations

	Value	Unit
Motor Current	1.200	Amps
Motor Voltage	9	Volts
Motor Resistance	7.5	Ohms
Photon Current	0.1	Amps
Battery Capacity	600.00	mAh
Constant	9.5488	
Speed	131	RPM
Motor Power	10.8	Watts
Torque	0.79	Nm
Spool Radius	0.002	m
Spool Length	0.01	m
Spool Mass	0.00053	KG
Force	393	N
Max bending moment	3.93	Nm
Sigma	7.41	Mpa
Safety Factor	3	
Sigma_max	22.24	MPa
Battery Life	39.56	min
Max allowable	48	Mpa

Equations Used

Electrical Equations:

V (Volts) = R (Ω) * I (A)

P (W) = I (A) * V (Volts)

$$Battery\ Life\ (min) = \frac{Capacity(mAh) * 60}{I(mA) * 0.7(disturbance)}$$

Mechanical Equations:

T (Nm) = F (N) * d (m)

M(Nm) = F (N) * l (m)

$$\sigma = -\frac{M\ (Nm) * y(m)}{I_c\ (kgm^2)}$$

Variables	
V = Voltage	T = Torque
R = Resistance	F = Force
I = Current	d = Moment arm
P = Power	l - length
M = Bending moment	y = Centroid
σ = Bending Stress	Ic = Moment of Inertia

Even with the safety factor of 3, the max possible pressure is 2 times smaller than the max allowable

Legend	
Given Specifications	(green)
Calculated Values	(orange)
Constants	(blue)
Cutoff Values	(pink)

Figure 22: Calculations

8.2 Electronics and Actuation Code

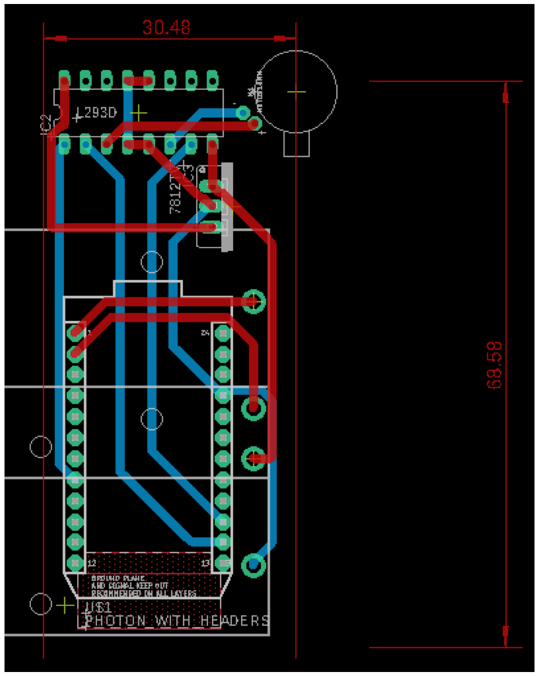


Figure 23: Double Sided PCB design

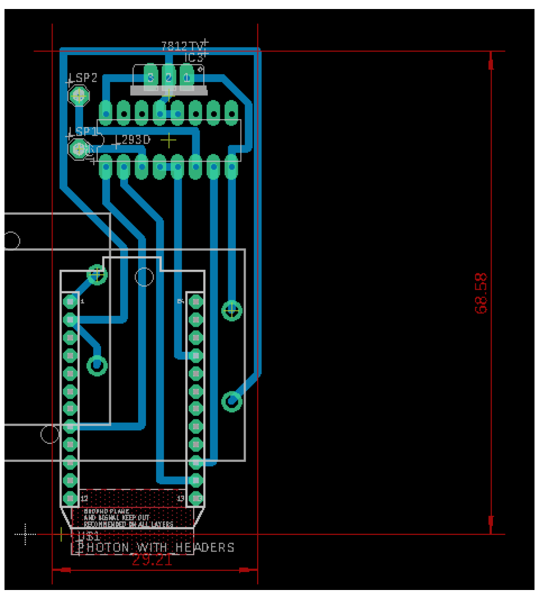


Figure 24: Single Sided PCB design

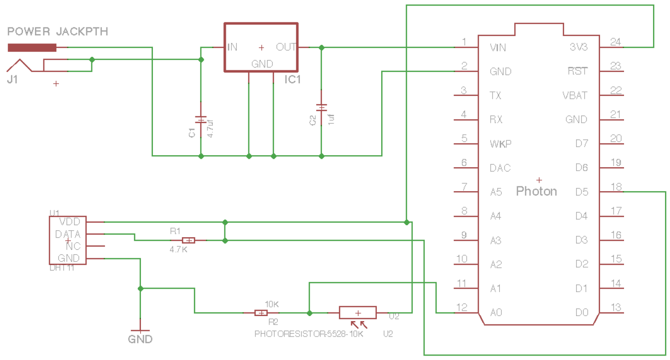
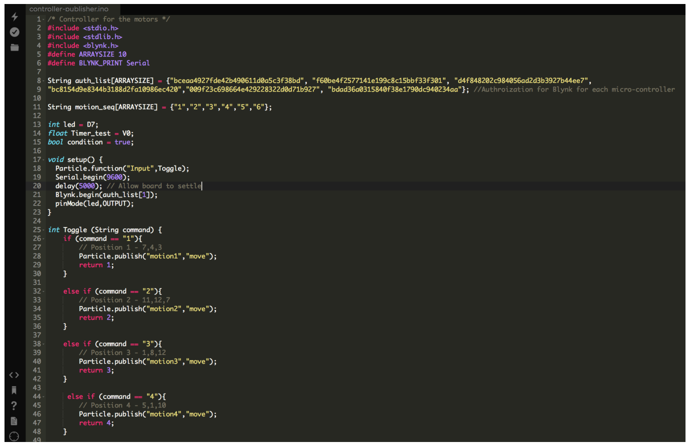


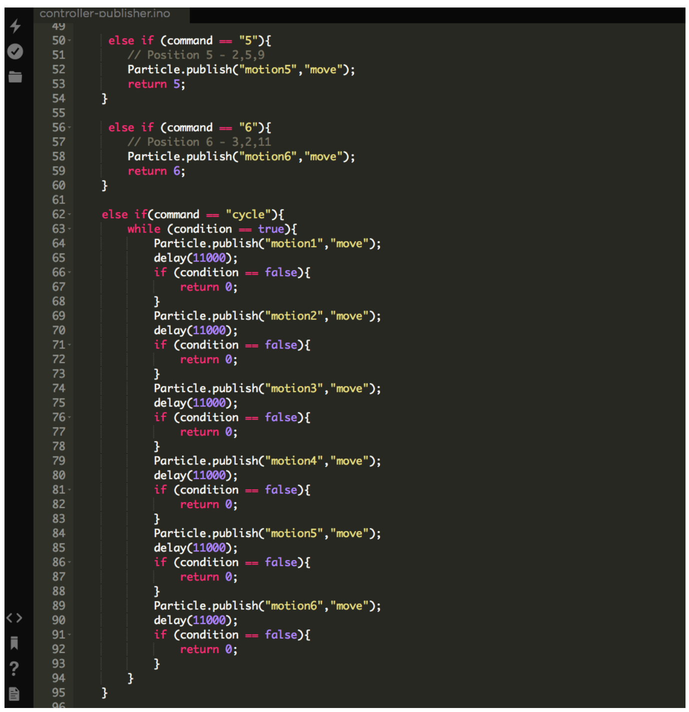
Figure 25: Photon Schematic

8.3 Controller Actuation Code



```
/* Controller for the motors */
#include <stdio.h>
#include <stdlib.h>
#include <blynk.h>
#define ARRAYSIZE 10
#define BLYNK_PRINT Serial

String auth_list[ARRAYSIZE] = {"bceaa4927fde42b490611d0a5c3f38bd", "f60be4f2577141e199c8c15bbf33f301", "d4f848202c984056ad2d3b3927b44ee7",
"bc8154d9e8344b3188d2fa10986ec420","009f23c698664e429228322d0d71b927", "bdad36a0315840f38e1790dc940234aa"}; //Authroization for Blynk for each micro-controller

String motion_seq[ARRAYSIZE] = {"1","2","3","4","5","6"};

int led = D7;
float Timer_test = V0;
bool condition = true;

void setup() {
  Particle.function("Input",Toggle);
  Serial.begin(9600);
  delay(5000); // Allow board to settle
  Blynk.begin(auth_list[1]);
  pinMode(led,OUTPUT);
}

int Toggle (String command) {
    if (command == "1"){
        // Position 1 - 7,4,3
        Particle.publish("motion1","move");
        return 1;
    }

    else if (command == "2"){
        // Position 2 - 11,12,7
        Particle.publish("motion2","move");
        return 2;
    }

    else if (command == "3"){
        // Position 3 - 1,8,12
        Particle.publish("motion3","move");
        return 3;
    }

     else if (command == "4"){
        // Position 4 - 5,1,10
        Particle.publish("motion4","move");
        return 4;
    }
```

Figure 26: Publisher Code: Part 1



```
    else if (command == "5"){
        // Position 5 - 2,5,9
        Particle.publish("motion5","move");
        return 5;
    }

    else if (command == "6"){
        // Position 6 - 3,2,11
        Particle.publish("motion6","move");
        return 6;
    }

    else if(command == "cycle"){
        while (condition == true){
            Particle.publish("motion1","move");
            delay(11000);
            if (condition == false){
                return 0;
            }
            Particle.publish("motion2","move");
            delay(11000);
            if (condition == false){
                return 0;
            }
            Particle.publish("motion3","move");
            delay(11000);
            if (condition == false){
                return 0;
            }
            Particle.publish("motion4","move");
            delay(11000);
            if (condition == false){
                return 0;
            }
            Particle.publish("motion5","move");
            delay(11000);
            if (condition == false){
                return 0;
            }
            Particle.publish("motion6","move");
            delay(11000);
            if (condition == false){
                return 0;
            }
        }
    }
```

Figure 27: Publisher Code: Part 2

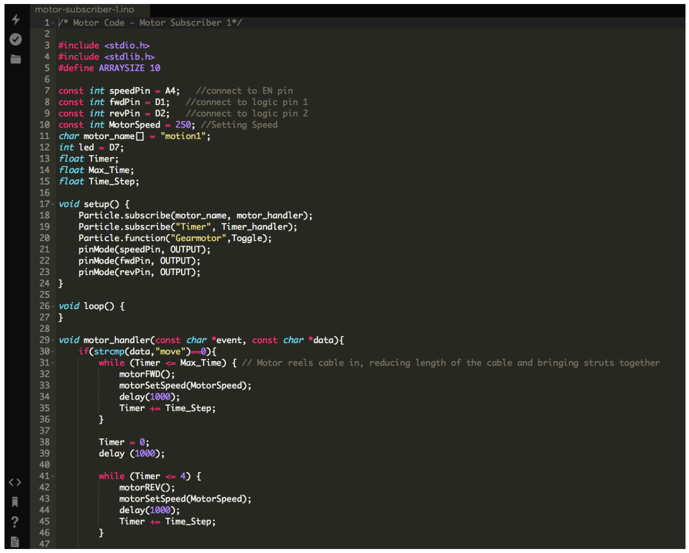
controller-publisher.ino

```
    else if(command == "s"){
        condition = false;
        Particle.publish("motion1","stop");
        delay(500);
        Particle.publish("motion2","stop");
        delay(500);
        Particle.publish("motion3","stop");
        delay(500);
        Particle.publish("motion4","stop");
        delay(500);
        Particle.publish("motion5","stop");
        delay(500);
        Particle.publish("motion6","stop");
        delay(500);
        return 0;
    }

    else if (command == "t7"){
        Particle.publish("Timer", "7");
        return 7;
    }

    else if (command == "t8"){
        Particle.publish("Timer", "8");
        return 8;
    }

    else if (command == "t9"){
        Particle.publish("Timer", "9");
        return 9;
    }

    else if (command == "t10"){
        Particle.publish("Timer", "10");
        return 10;
    }

}

//Led Test
BLYNK_WRITE(D7)
{
  digitalWrite(led, HIGH);
  digitalWrite(led, LOW);
}

void loop() {
    Blynk.run();
    //Particle.publish("Controller", "On");
}
```

Ready.

Figure 28: Publisher Code: Part 3

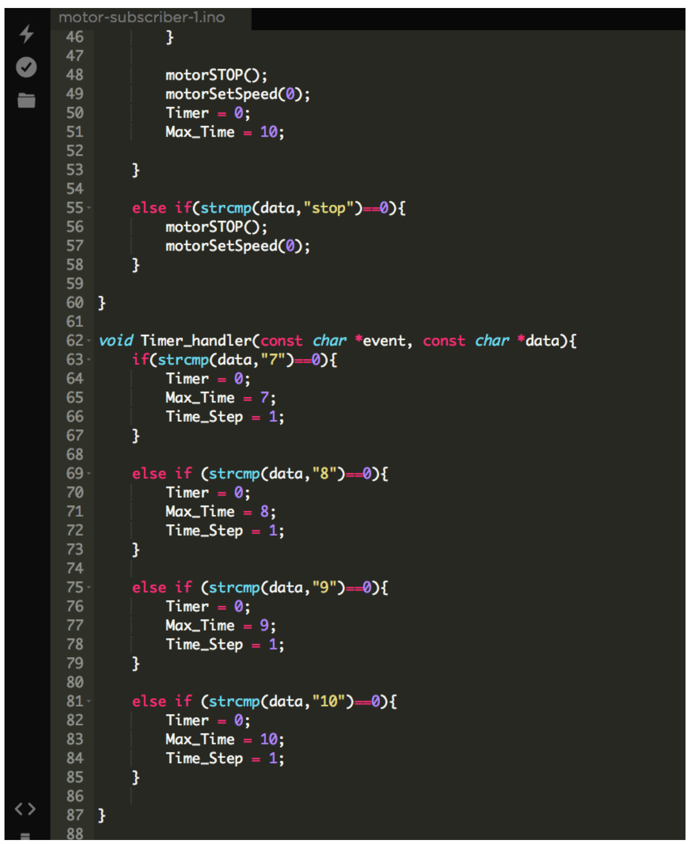
8.4 Motor Subscriber Actuation Code:



motor-subscriber-1.ino

```
/* Motor Code - Motor Subscriber 1*/

#include <stdio.h>
#include <stdlib.h>
#define ARRAYSIZE 10

const int speedPin = A4;   //connect to EN pin
const int fwdPin = D1;   //connect to logic pin 1
const int revPin = D2;   //connect to logic pin 2
const int MotorSpeed = 250; //Setting Speed
char motor_name[] = "motion1";
int led = D7;
float Timer;
float Max_Time;
float Time_Step;

void setup() {
    Particle.subscribe(motor_name, motor_handler);
    Particle.subscribe("Timer", Timer_handler);
    Particle.function("Gearmotor",Toggle);
    pinMode(speedPin, OUTPUT);
    pinMode(fwdPin, OUTPUT);
    pinMode(revPin, OUTPUT);
}

void loop() {
}

void motor_handler(const char *event, const char *data){
    if(strcmp(data,"move")==0){
        while (Timer <= Max_Time) { // Motor reels cable in, reducing length of the cable and bringing struts together
            motorFWD();
            motorSetSpeed(MotorSpeed);
            delay(1000);
            Timer += Time_Step;
        }

        Timer = 0;
        delay (1000);

        while (Timer <= 4) {
            motorREV();
            motorSetSpeed(MotorSpeed);
            delay(1000);
            Timer += Time_Step;
        }

```

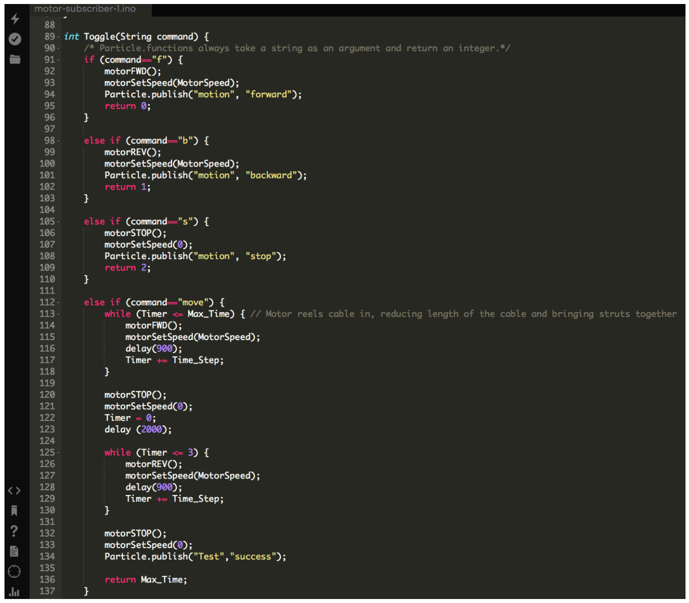
Figure 29: Subscriber Code: Part 1



motor-subscriber-1.ino

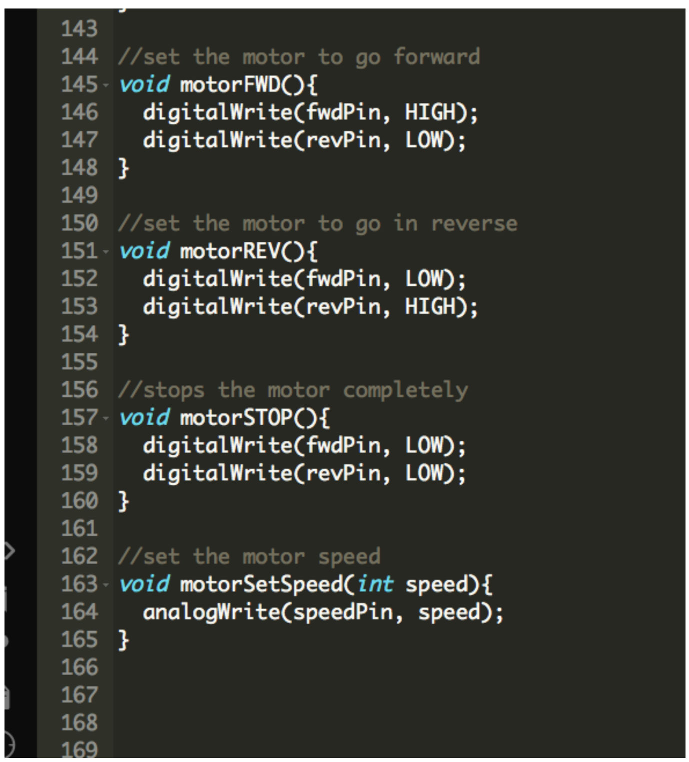
```
        }

        motorSTOP();
        motorSetSpeed(0);
        Timer = 0;
        Max_Time = 10;

    }

    else if(strcmp(data,"stop")==0){
        motorSTOP();
        motorSetSpeed(0);
    }

}

void Timer_handler(const char *event, const char *data){
    if(strcmp(data,"7")==0){
        Timer = 0;
        Max_Time = 7;
        Time_Step = 1;
    }

    else if (strcmp(data,"8")==0){
        Timer = 0;
        Max_Time = 8;
        Time_Step = 1;
    }

    else if (strcmp(data,"9")==0){
        Timer = 0;
        Max_Time = 9;
        Time_Step = 1;
    }

    else if (strcmp(data,"10")==0){
        Timer = 0;
        Max_Time = 10;
        Time_Step = 1;
    }

}

```

Figure 30: Subscriber Code: Part 2



```
motor-subscriber-1.ino

int Toggle(String command) {
    /* Particle.functions always take a string as an argument and return an integer.*/
    if (command=="f") {
        motorFWD();
        motorSetSpeed(MotorSpeed);
        Particle.publish("motion", "forward");
        return 0;
    }

    else if (command=="b") {
        motorREV();
        motorSetSpeed(MotorSpeed);
        Particle.publish("motion", "backward");
        return 1;
    }

    else if (command=="s") {
        motorSTOP();
        motorSetSpeed(0);
        Particle.publish("motion", "stop");
        return 2;
    }

    else if (command=="move") {
        while (Timer <= Max_Time) { // Motor reels cable in, reducing length of the cable and bringing struts together
            motorFWD();
            motorSetSpeed(MotorSpeed);
            delay(900);
            Timer += Time_Step;
        }

        motorSTOP();
        motorSetSpeed(0);
        Timer = 0;
        delay (2000);

        while (Timer <= 3) {
            motorREV();
            motorSetSpeed(MotorSpeed);
            delay(900);
            Timer += Time_Step;
        }

        motorSTOP();
        motorSetSpeed(0);
        Particle.publish("Test","success");

        return Max_Time;
    }
```

Figure 31: Subscriber Code: Part 3



```
//set the motor to go forward
void motorFWD(){
  digitalWrite(fwdPin, HIGH);
  digitalWrite(revPin, LOW);
}

//set the motor to go in reverse
void motorREV(){
  digitalWrite(fwdPin, LOW);
  digitalWrite(revPin, HIGH);
}

//stops the motor completely
void motorSTOP(){
  digitalWrite(fwdPin, LOW);
  digitalWrite(revPin, LOW);
}

//set the motor speed
void motorSetSpeed(int speed){
  analogWrite(speedPin, speed);
}
```

Figure 32: Subscriber Code: Part 4

9 References

1. Agogino, A. , (2013). Final Report Super Ball Bot - Structures for Planetary Landing and Exploration. Prepared for NASA

2. The BEST Lab at UC Berkeley – Berkeley [Emergent Space Tensegrities — Energy and Sustainable Technologies — Expert Systems Technologies] Lab

3. Rieffel, J., Stuk, R., Valero Cuevas F., Lipson, H. (2007) "Locomotion of a Tensegrity Robot via Dynamically Coupled Modules". Proceedings of the International Conference on Morphological Computation, Venice Italy, March 2007.

4. Chen, L.,(2017). Modular Elastic Lattice Platform for Rapid Prototyping of Tensegrity Robots. Volume 5B: 41st Mechanisms and Robotics Conference. doi:10.1115/detc2017-68264

5. Yadav, S. (2015), Fabrication and Analysis of Tensegrity Based Prism structure

6. Shenk, M.(2015, Statically Balanced Tensegrity Mechanisms.

7. Paul C., Lipson H., Valero Cuevas F. J. (2005) , "Gait Production in a Tensegrity Based Robot" Proceedings of 12th International Conference on Advanced Robotics (ICAR), Seattle, Washington, USA, July 18th-20th, pp 216-222.