Robot Metabolism: How to Build a Truss Link

Spring 2025

Creative Machines Lab
Columbia University, New York, NY 10027, USA

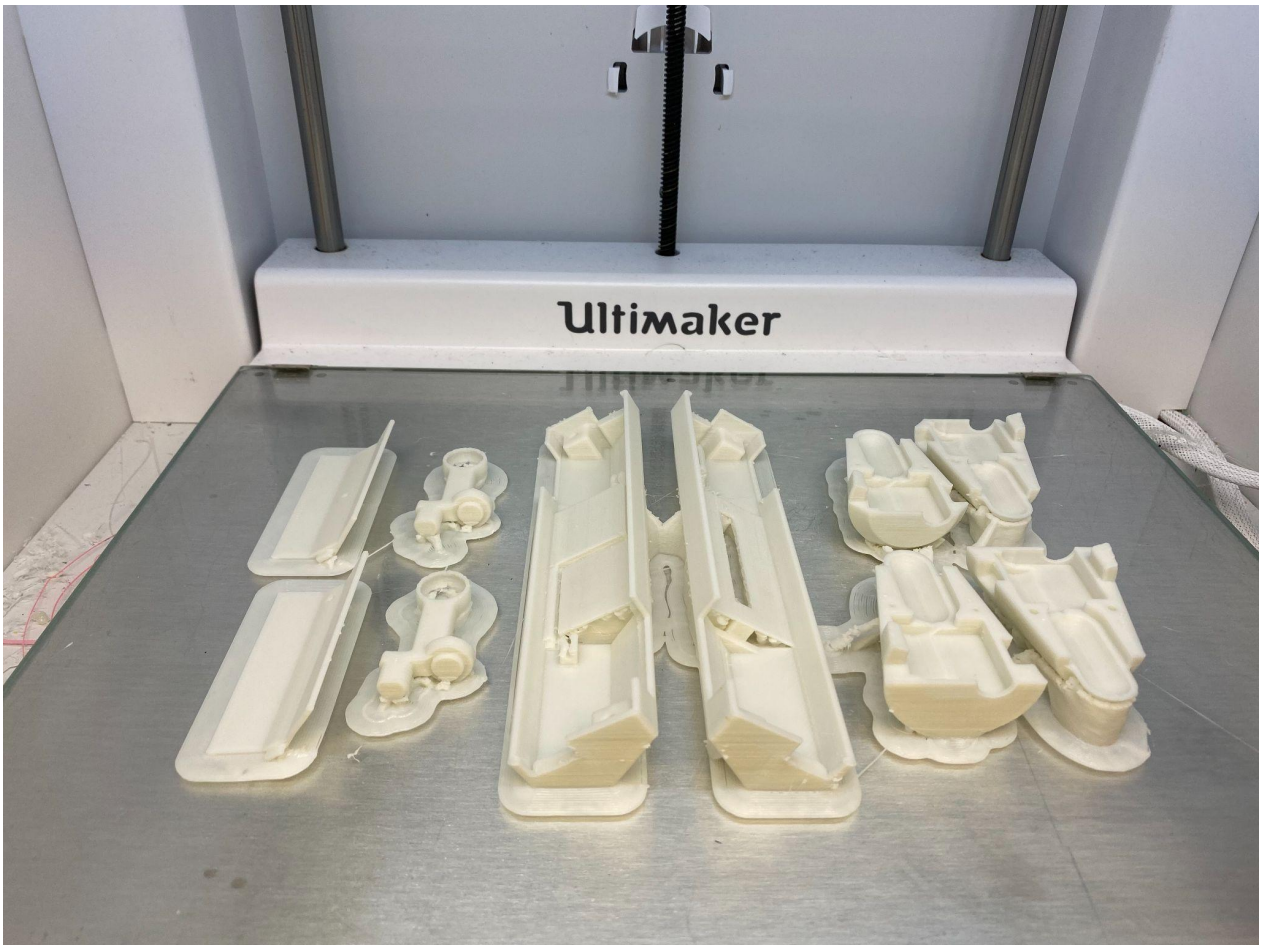
Prepared by:
Philipe Wyder, Jiahao Wu, Robert Kasumi, Matthew Modi, Quinn Booth

Version 2.5
Last Updated: 05/21/2025

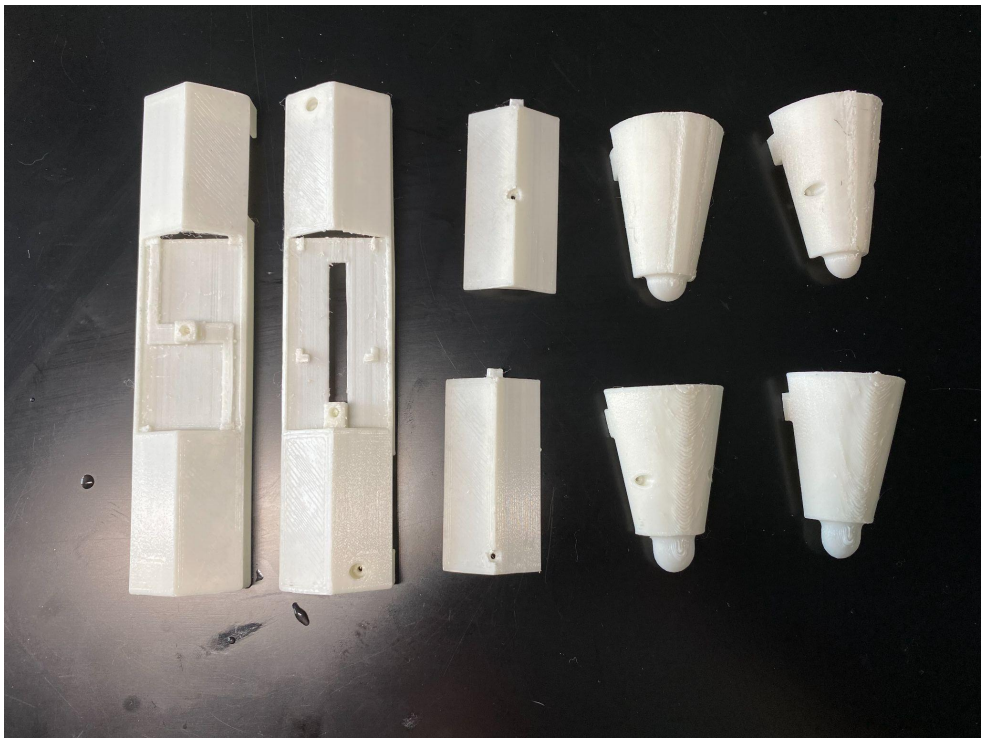
Table of Contents

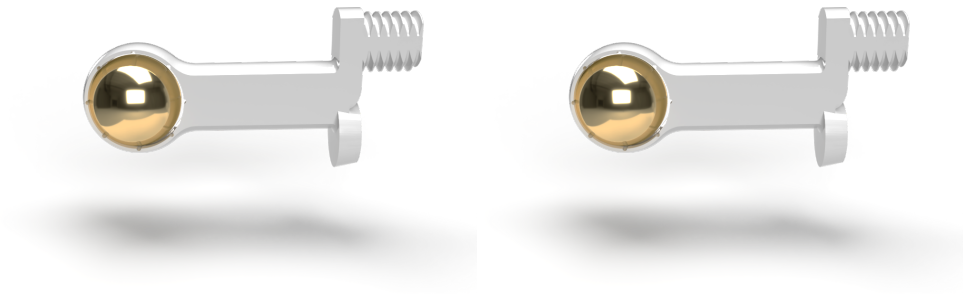
3D Printing Components

1. Download CAD files for the shell and connectors from below link:
   3D Files can be found in the 3D_files folder.
2. Use Glow-in-the-Dark filament 3mm with Ultimaker S5
3. Adjust Ultimaker S5 to following setting by using cura:
   a. Use Core CC 0.6
   b. Set quality layer height as 0.2
   c. Set printing temperature,initial printing temperature and final temperature as 215 C,
   d. Change Flow as 120%
   e. Change print speed and initial layer speed to 40 mm/s
   f. Change retraction distance to 5mm/s
   g. Turn Z Hop When Retracted on
   h. Use tree support with Gyroid shape and limit degree as 45 degree
   i. Use Brim for build plate adhesion



4. Use sandpaper to smoothen out any uneven print surfaces





Build Preparation

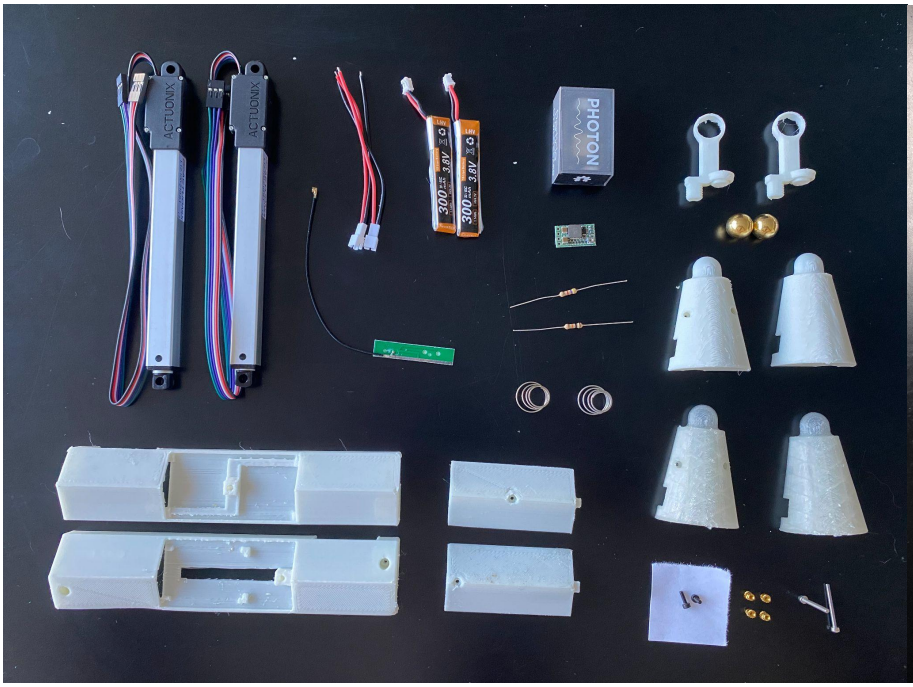
Partlist

Truss Link Body

1. 3D printed Body-Shell (Top & Bottom) and 2 x Cover
2. 2 x Actuonix L-12I
3. Particle Photon
4. 2.4GHz Mini Flexible WiFi Antenna with uFL Connector - 100mm
5. Drok Voltage Regulator
6. 4.7 kOhm Resistor
7. 10 kOhm Resistor
8. 2 x JST-PH 2.0 Female harness
9. 2 M2x20 Stainless Steel flat head screws
10. 2 M2x8 Carbon steel flat head screws
11. 4 x M2 heat set insert

Truss Link Connector

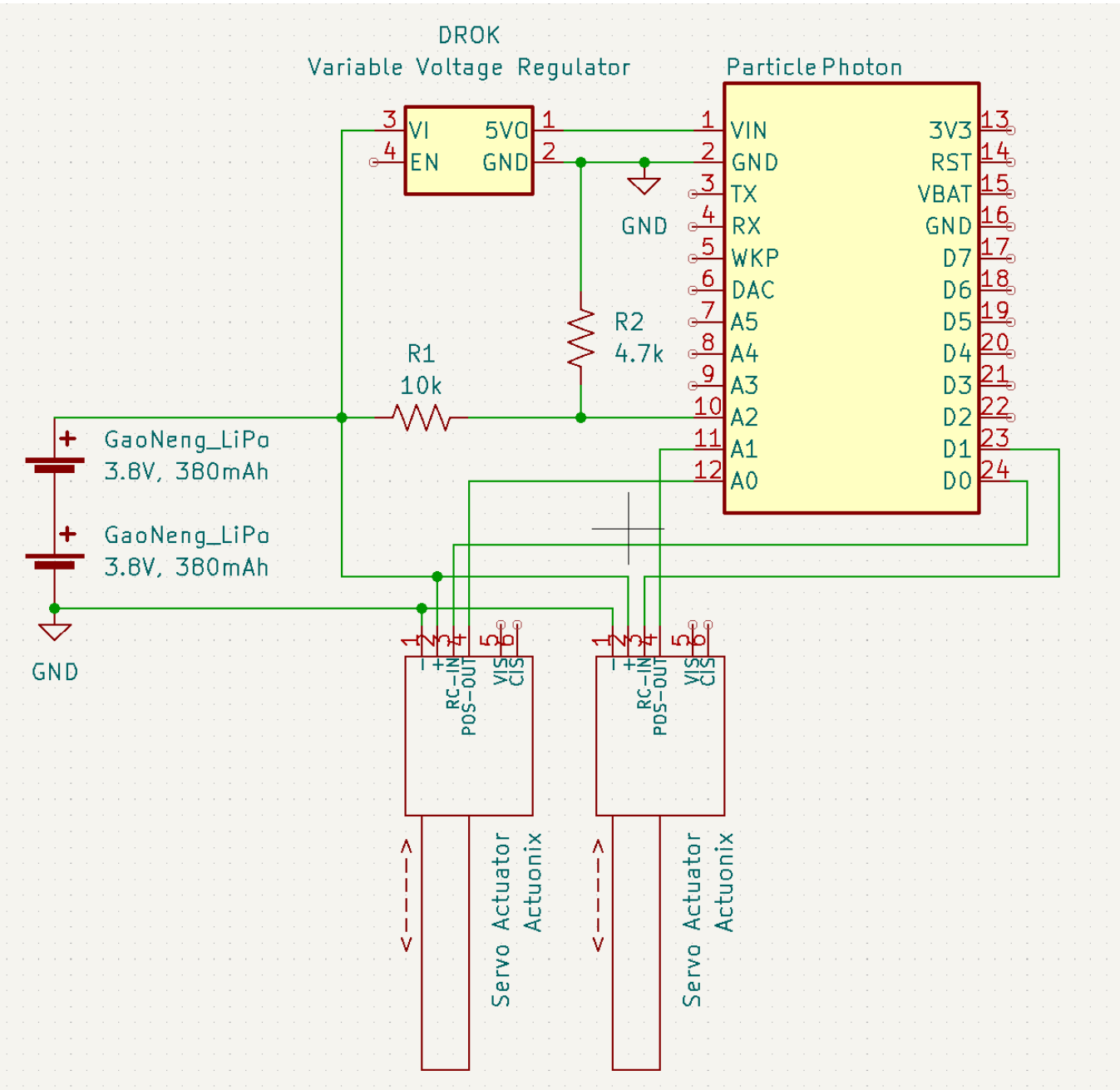
1. 2 x 3D printed Connector-Shell (Top & Bottom) and Magnet Holder
2. Confined-Space Conical Compression Spring 0.75” L, 0.6” x 0.375” OD
3. Neodymium magnet sphere 1/2” diameter
4. 2 M2x8 Carbon steel flat head screws
5. 2 x M2 heat set insert



Support Material:

1. Electrical tape
2. Heat Shrink tube
3. AWG 26 Wires

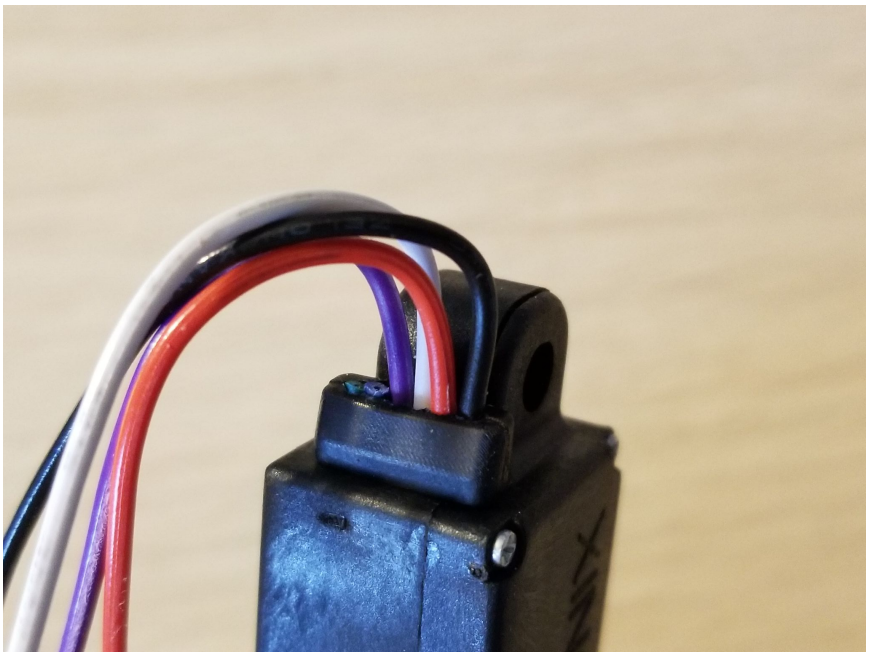
Circuit Diagram

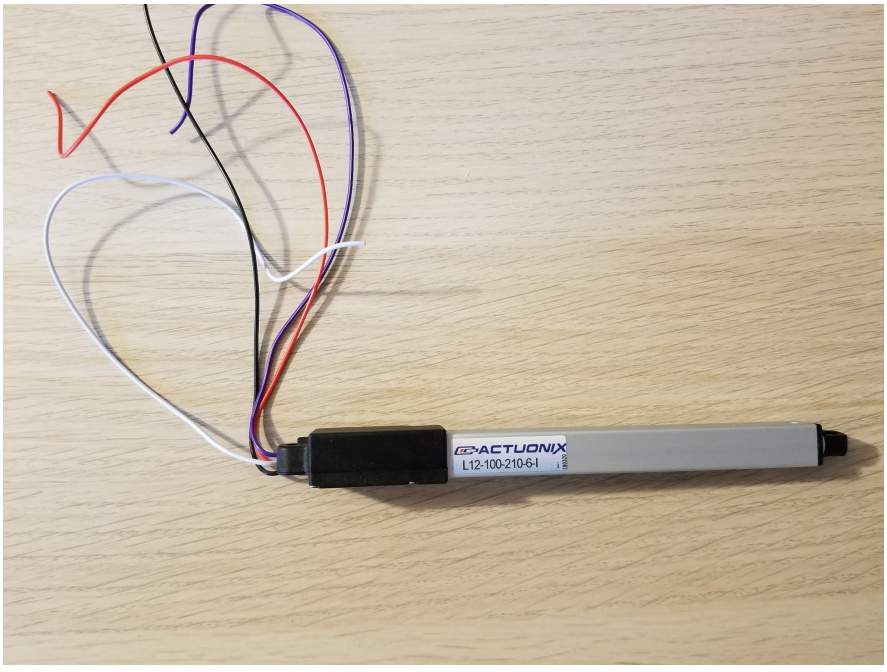


Getting the Components Together

Actuators

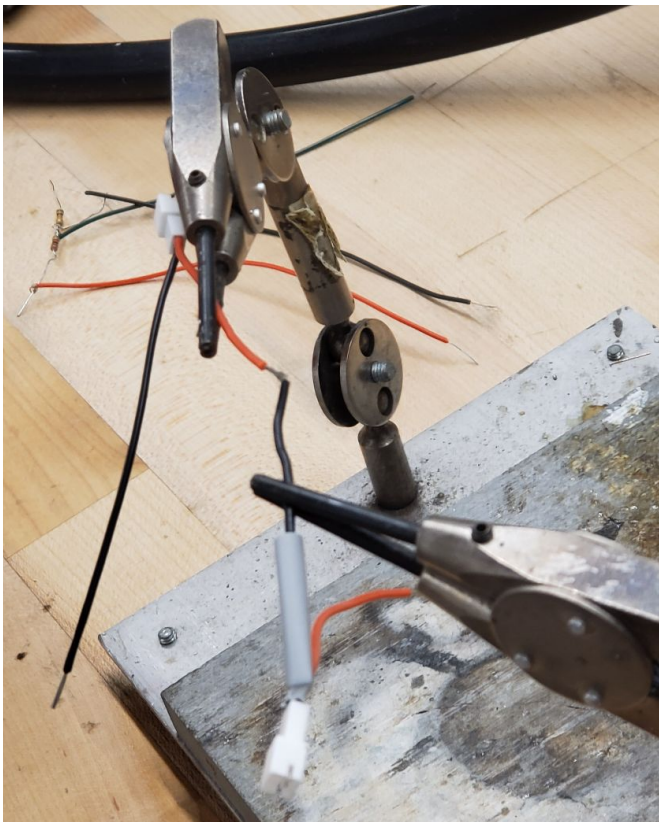
1. Cut green and blue wires flush with the actuator housing and save them for future use. For the rest of this assembly, we will only be using these wires.
2. Separate the remaining 4 wires (black, red, white, purple).
3. Cut the red and black wires in half for future use.

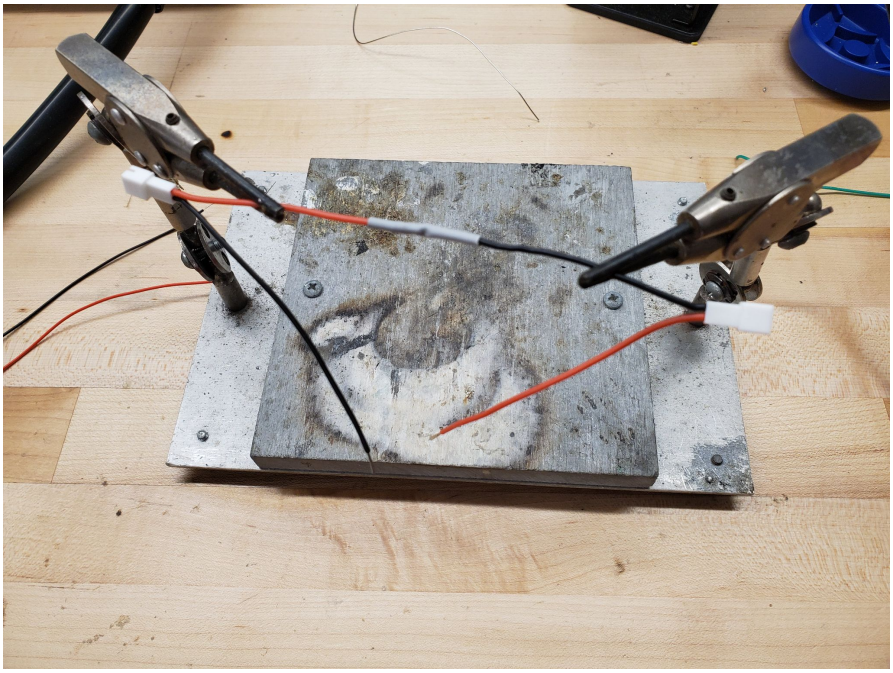




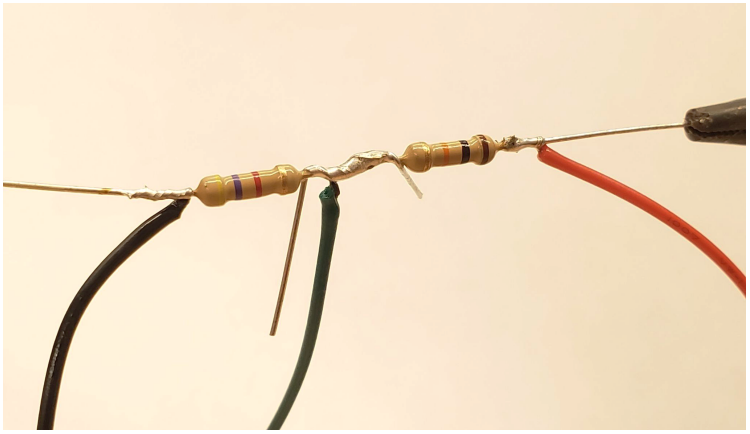
Battery Harness

1. Cut the red wire from one JST-PH 2.0 Female harness and the black wire of the other JST-PH 2.0 Female harness to 5cm in length.
2. Put heat shrink tubing around one end of the wires to be soldered.
3. Solder the shortened red wire from one JST-PH 2.0 Female harness to the shortened black wire of the other JST-PH 2.0 Female harness.
4. Heat the tubing so it conforms to the wires.





1. Cut green wire as 100mm, red wire as 100mm and black 50mm (tolerance: +-5mm).
2. Solder the resistors and wires as per the circuit diagram (red wire to 10 kOhm, black wire to 4.7 kOhm and green in the middle).
3. Arrange the wires such that the green and black wires are on one side and the red wire is on the other.

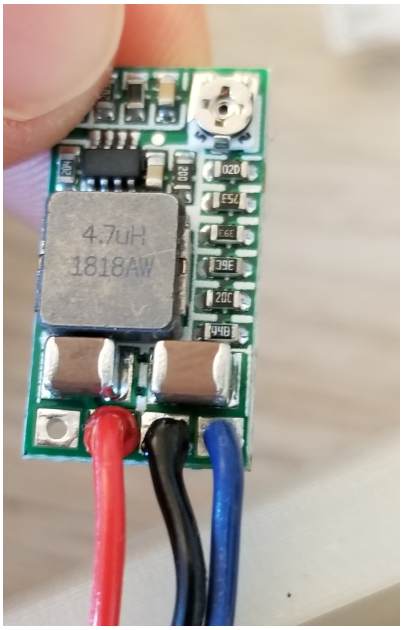


4. Clip all excess wire and sharp corners to prevent them from puncturing the heat shrink.

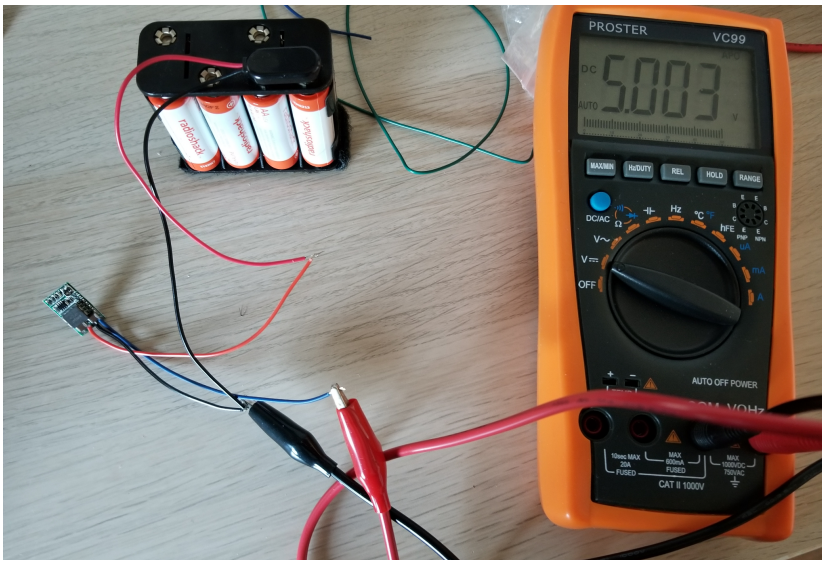
5. Pull heat shrink tubing over the soldered resistors.

6. Heat the tubing so it conforms to the wires and resistors.

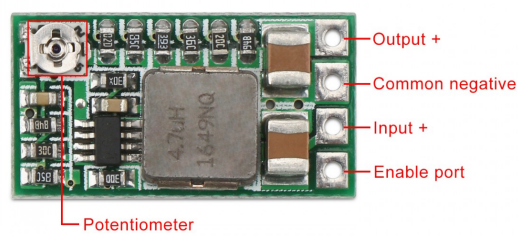
1. Red Wire 45mm, Black wire 110 mm and Blue wire 135mm (tolerance: +-5mm)
2. Solder the blue wire to the VO+ port.
3. Solder the black wire to the GND port.
4. Solder the red wire to the IN+ port.



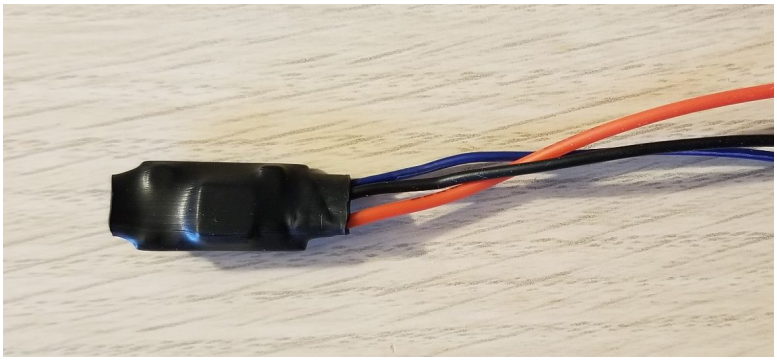
5. Connect the red wire of the combined JST-PH 2.0 Female harness to the red wire of the Voltage Regulator
6. Connect the black wire of the combined JST-PH 2.0 Female harness to the black wire of the Voltage Regulator
7. Connect the black prong of the Multimeter to the black wire of the Voltage Regulator
8. Connect the red prong of the Multimeter to the blue wire of the Voltage Regulator
9. Connect the batteries to the female harness to apply between 7-9V DC
10. Turn on Multimeter and set the read setting to DC Voltage



11. Adjust Voltage Regulator potentiometer with Philips screwdriver to make the output to 5V



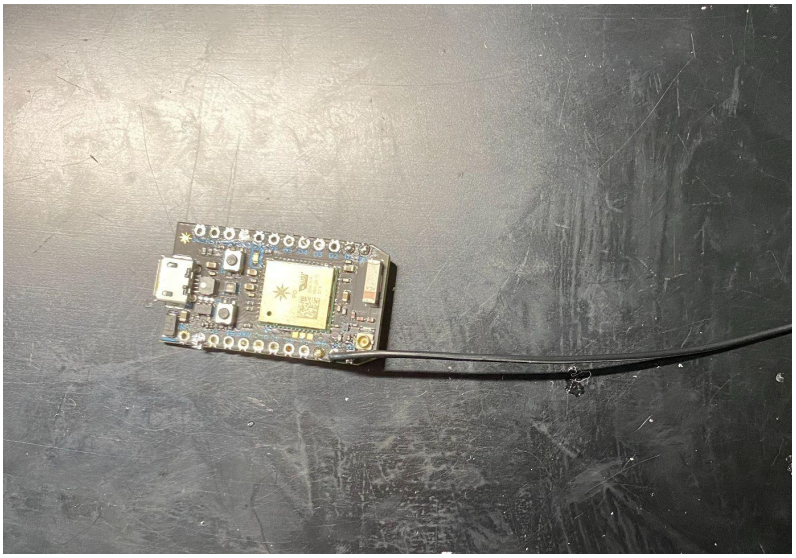
12. Pull heat shrink tubing over the voltage regulator.
13. Heat the tubing so it conforms around the voltage regulator.



Particle Photon

Note: This step can be deferred until later in the assembly process to better help gauge the wire lengths

1. Solder the black wire (100mm) on the GND port.



Truss Link Body Assembly

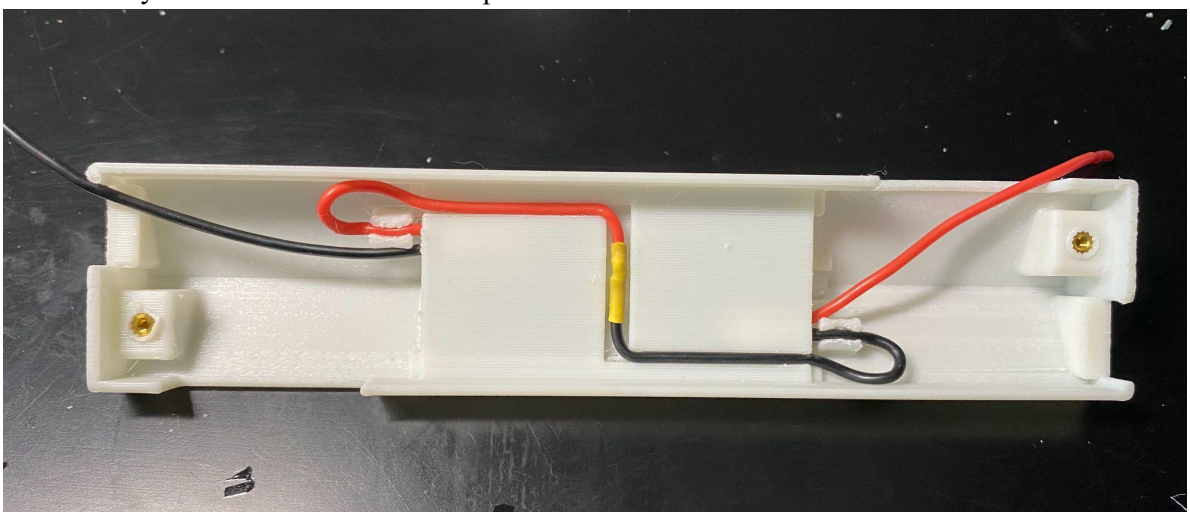
Heat Set Insert into shell:

1. Increase electrical soldering gun temperature to 300°C
2. Use a heat gun, press the heat insert into the body shell, double check if any 3d material is left inside of the heat insert, you will not be able to thread screw in if there is too much material inside of the nut.

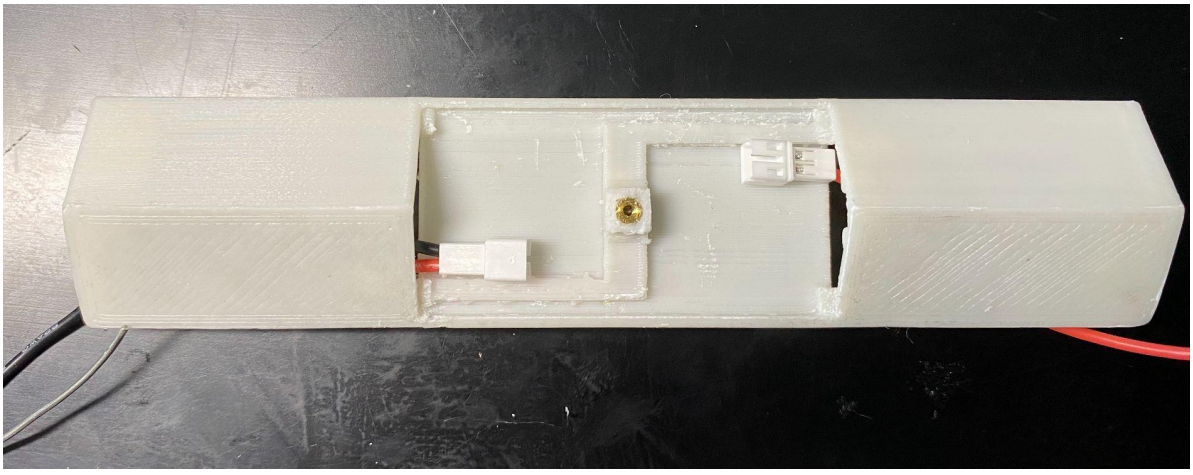
Below is a support video to let you learn how to install heat insert into 3d print part.
https://www.youtube.com/watch?v=G-UF4tv3Hvc&ab_channel=CNCKitchen\

Place battery wire into body shell:

1. Place battery wire into wire channel and push the wire into wire hook



2. Place battery and test if there is enough space for the battery, otherwise, adjust the wire location.



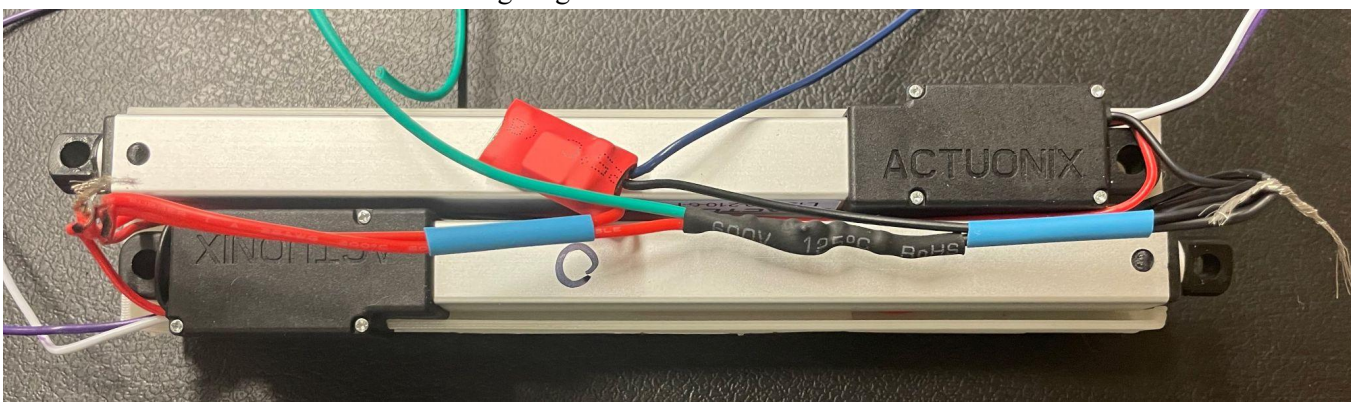
Actuator into shell

1. Before placing the actuator into a 3D printed shell (battery side), Stick electrical tape to cover the gap between two actuators.
2. Pull red and black battery wire in the end of the shell and make sure the two wires do not interference with the actuators.

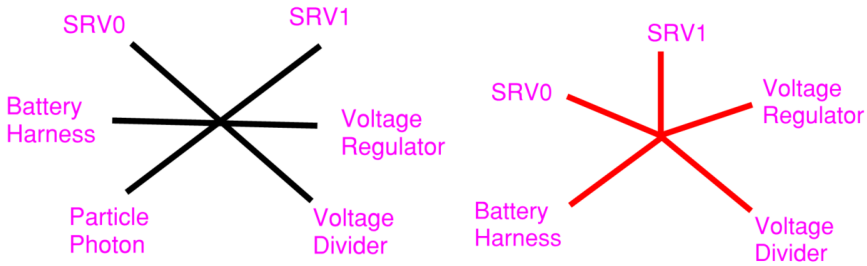
3. Cut Servos' wire following below table: (tolerance: +-5mm)

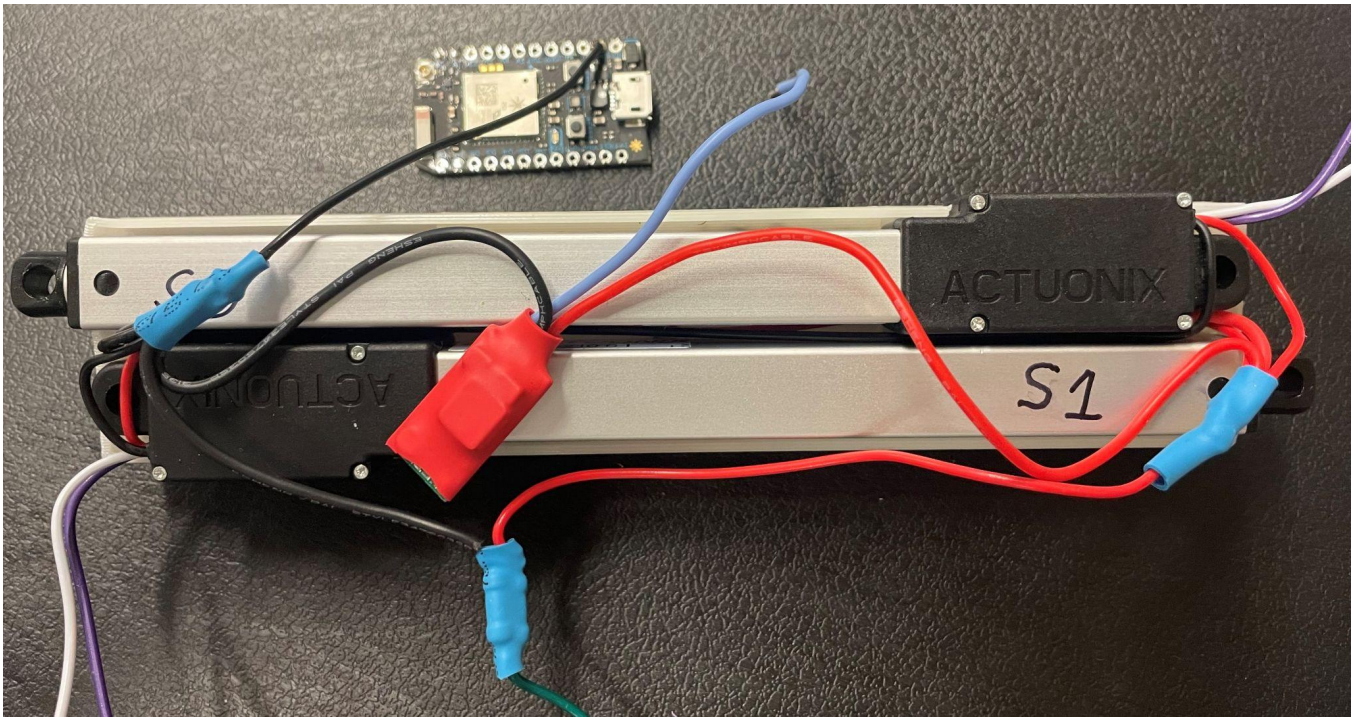
	Black Wire	Red Wire	White Wire	Purple Wire
Servo 0	160mm	65mm	140mm	140mm
Servo 1	65mm	160mm	100mm	100mm

4. Push Servo 0 black wire and Servo 1 red wire into the mid-gap between two actuators. We will place all back (ground) wires on the Servo actuator 1 side and all red (positive) wires on the Servo actuator 0 side.
5. Place voltage regulator on the end of Servo 1 and place resistances on the end of Servo 0. Use Electrical Tape to hold VR and R on the actuators.
6. Make sure all red (Pos) wires and all-black (GND) wires stay together.
7. Place the blue wire from the voltage regulator on servo 1 side.

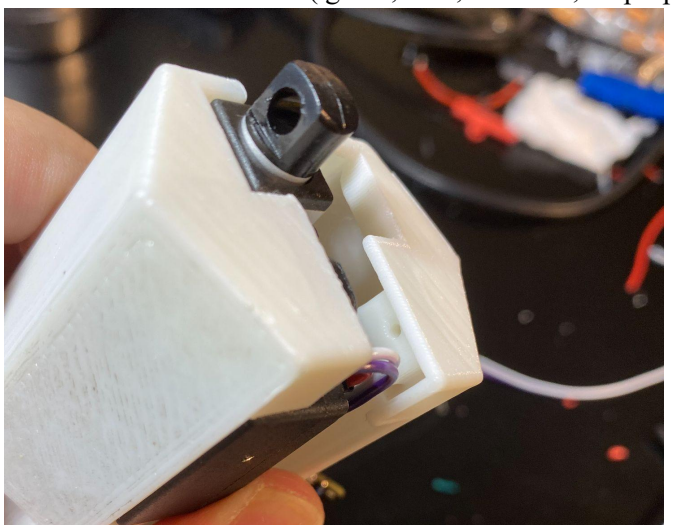


8. Solder the remaining loose black and red wires as per the Circuit Diagram. Use the below images as reference.
    a. Remember to place heat shrink around one of the wires before soldering. Once soldered, heat the tubing so it conforms to the wires.



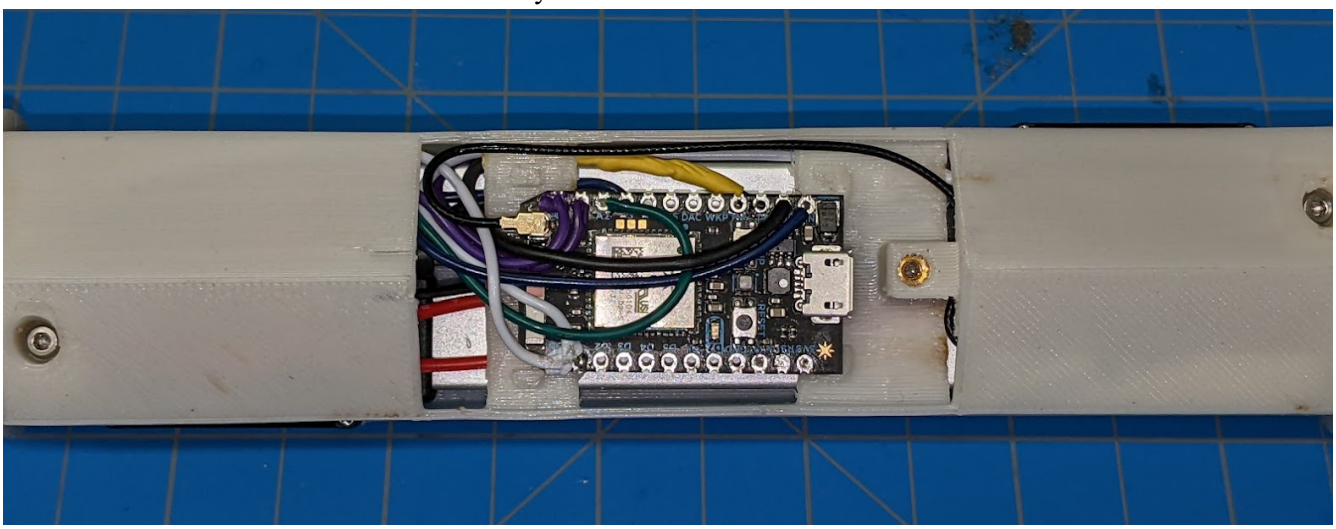


9. Place the bushing into the mount hold of the actuator. And make sure all of the wires not above or close to the hole, it will cause interference when you close the shells.
10. Close the photon side body shell, make sure the actuator wires (black, red, white, purple) do not interfere with the body. And pull all the rest of the wire (green, blue, 2x white, 2x purple) out of the body shell.



Photon Placement

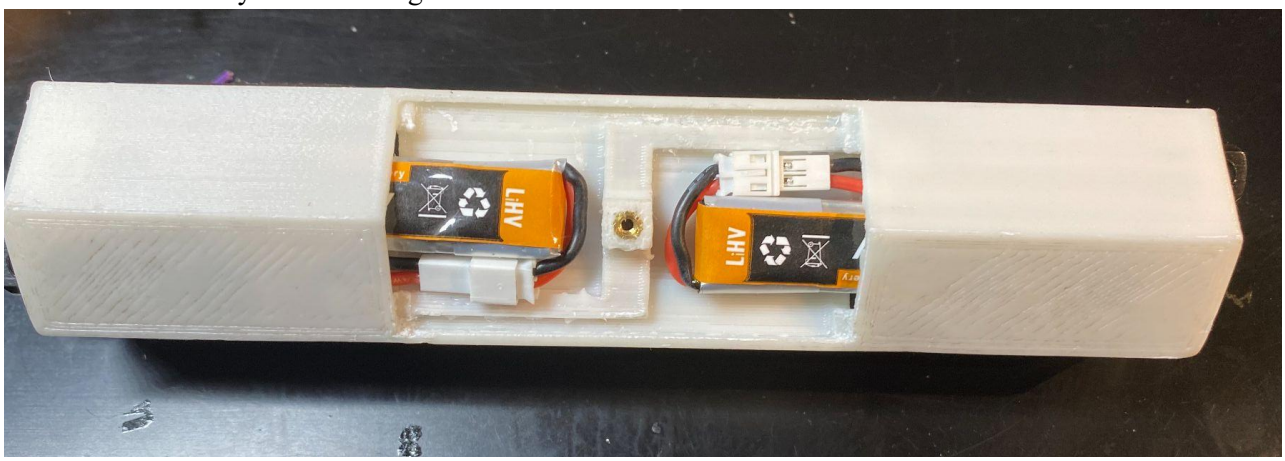
1. Solder the wires to the Photon as per the Circuit Diagram.
    a. Solder the White and purple wire from SRV0 to the D0 and A0 ports respectively.
    b. Solder the White and purple wire from SRV1 to the D1 and A1 ports respectively.
    c. Solder the Blue wire from the Voltage Regulator to the VMN port.
    d. Solder the Green wire from Voltage Divider to A2 Port.
2. Push all extra wires inside of the body shell.



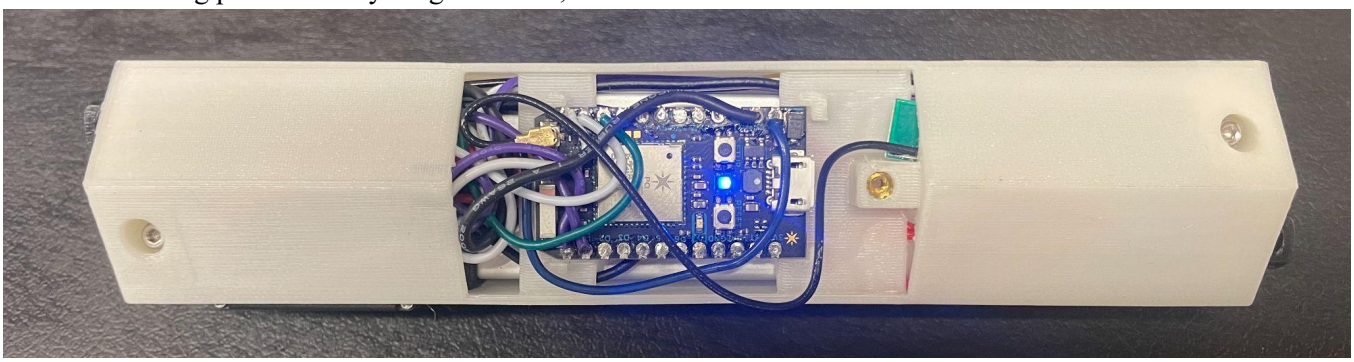
3. Connect WiFi Antenna to the Particle Photon (be careful not to damage the connector) and push it to the inside of the Truss Link body shell.

Body finishing

1. Secure the body shell by using M2X 20 mm stainless steel screw.
2. Place battery like below figure



3. Connector battery to see robot power on. If the indicator light is not on, double check your wires and soldering point. If everything is correct, switch batteries.



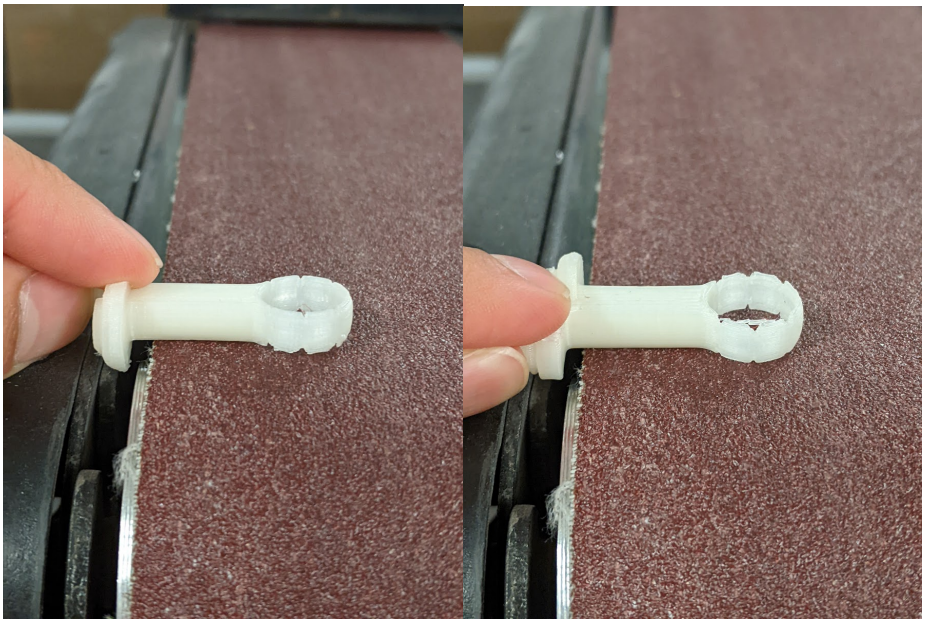
4. Secure the inspect window by using M2X8 mm carbon steel screw.

Now, you have a Truss Link body ready, let’s assemble connectors!

Truss Link Connector Assembly

Magnet support bar assembly

1. Sand each face of magnet bar to smooth it out and prevent interference:

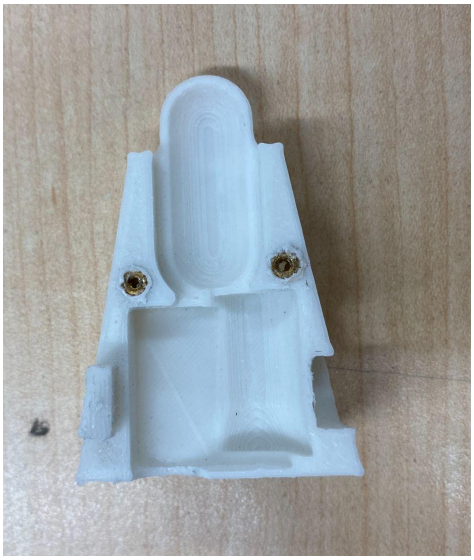


2. Briefly heat round end of magnet holder to soften plastic slightly. Heat it just enough that the plastic doesn't shatter when the magnet is snapped in, not such that it is malleable.

3. Snap magnet into magnet holder.

Connector shell finishing

1. Press heat insert into connector using soldering iron.



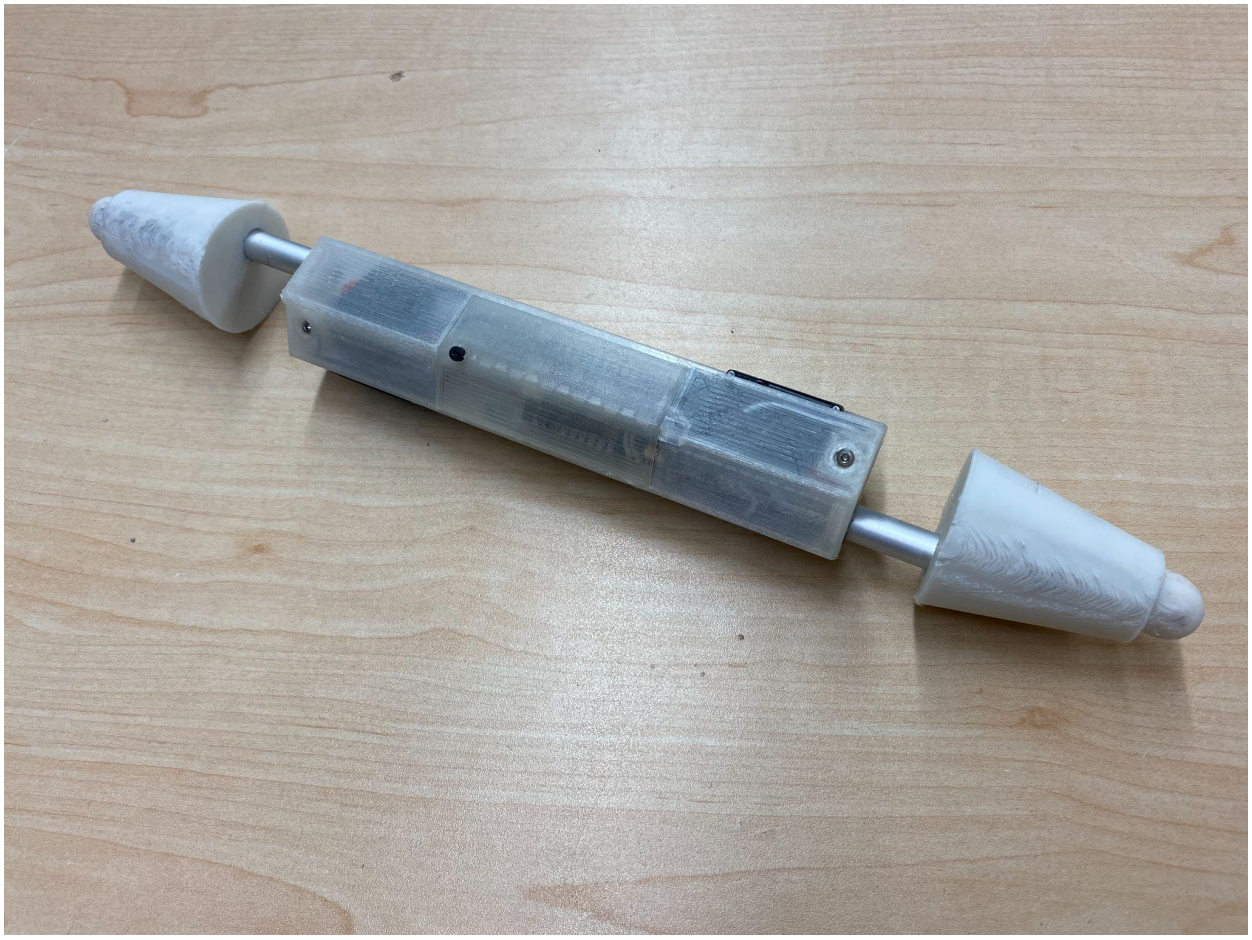
2. Drill out connecter shell hole with ⅜” bit. (not necessary)

1. Extend each servo 1-2” (this can be done using a script).
2. Remove plastic end caps, taking care not to unscrew the shaft from servo body.
3. Screw the magnet holder into place and center it with the robot body.

4. Apply heat glue on the gap between thread and actuator shaft.
   Note: be careful, it’s very HOT!

5. Lubricate inside of the connector body.

6. Use graphite to lubricate the magnet ball.
7. Place spring and screw connecter together.

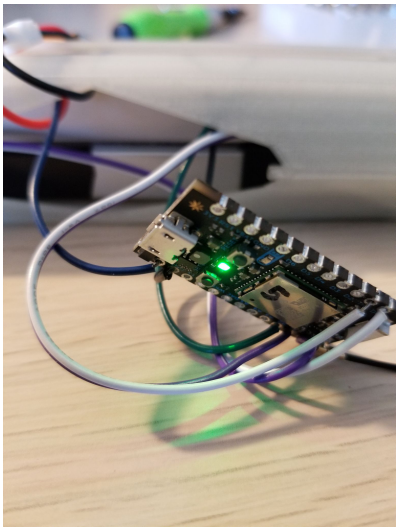


Now, you have a completed Truss Link, Congrats!
Let’s connect it to wifi and do some tests!

Testing

Test Wiring and Components

1. Plug in batteries
2. Photon LED should flash

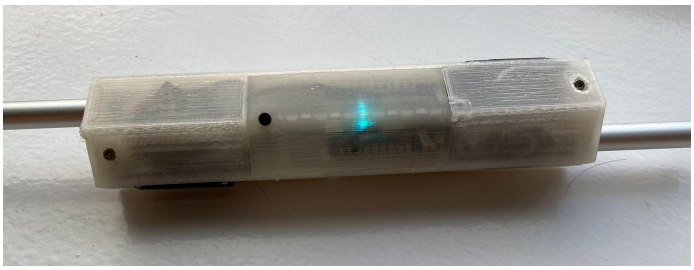


Connect to Wifi

1. Each Photon must be setup to connect to the wifi SSID “team_20-mobile” with password “robots1234" (since the firmware is programmed as such
2. Follow this setup to connect your photon to the wifi https://docs.particle.io/quickstart/photon/
3. Name your photon as an increment of the highest photon id using the format “P#” where “#” represent the number of the photon.

Test Links

1. Navigate to Particle Web IDE: https://build.particle.io/build/new
2. On the bottom left-hand ribbon, choose “Devices”
3. Select the number of the Photon you are working with (should be written on the device)
    a. If not written then select “Add New Device” and set up the Photon as a new device
4. Select the photon “to flash”
5. Navigate to the Code section
6. Select the testlinks-hardwareteam.ino firmware: https://build.particle.io/build/606ddbdbde63ee0017f5c0d3
7. On the top left, flash firmware to the Particle
8. Both actuators should extend one at a time



Flashing & Running

Once built and tested, the Truss Links can be flashed with our proprietary firmware and connected to our proprietary server software to control the system.

Installing the Server Software

Clone TrussLinkServer repository and follow the instructions in the README.md file to install all dependencies.

Installing the Truss Link Firmware

Assuming you have cloned the TrussLinkFirmware repository

Ensure proper configuration

1. Make sure before flashing the Truss Link Firmware that
    a. In RMProtocol.h #define SERVER_IP {192,168,0,15} is set to the ip address of the computer that you are running the server script on while connected to the *team_20-mobile* wifi network.
    b. That all links participating in the experiment are registered in utility.h under *ids*
2. Flash the firmware to all links participating in the experiment, by uncommenting all participating links in flash.sh, and calling *sh flash.sh* from the particle CLI.
3.

Test Wiring and Components by running the below firmware script.

TestLinks-HardwareTeam.ino

```
/*********
 * Modular Robot Control
 * Simple Inch Worm Gate
 * *******/
int move_wait_time = 140;

Servo srv0;
Servo srv1;

int srv0_sens = 0;
int srv1_sens = 0;
int voltage_raw = 0;
int srv0_pos = 0;
int srv1_pos = 0;

int MIN_NON_DETATCH_VAL = 60;
// temporarytest variables
int pos;

void setup() {
  // attaches the servo on the A4,A5 pin to a servo object
  srv0.attach( D0 );
  srv1.attach( D1 );
  srv0.write(MIN_NON_DETATCH_VAL);
  srv1.write(MIN_NON_DETATCH_VAL);
  delay(8000);

  // readout of position
  srv0_sens = analogRead( A0 ); // attach position readout of SRV0
  srv1_sens = analogRead( A1 ); // attach position readout of SRV1

  // readout voltage value
  voltage_raw = analogRead( A2 ); // readout raw voltage value

  Particle.variable(  "srv0_sens" , &srv0_sens , INT );
  Particle.variable(  "srv1_sens" , &srv1_sens , INT );
  Particle.variable(  "voltage_raw" , &voltage_raw , INT );

}

void read_analog_inputs(){
      srv0_sens = analogRead(A0);  // read the analogPin
      srv1_sens = analogRead(A1);  // read the analogPin
      voltage_raw = analogRead( A2 ); // readout raw voltage value
}

void loop() {
```

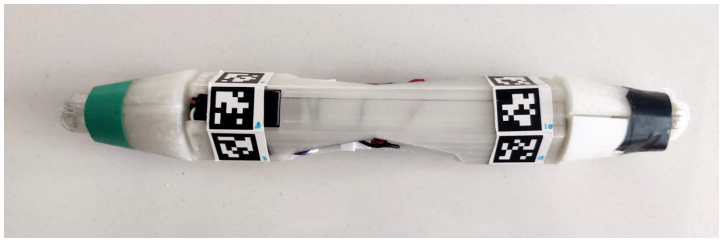
```
  // expand SRV0
  for(pos = MIN_NON_DETATCH_VAL; pos < 180; pos += 1)  // goes from 0 degrees to 180 degrees
  {                                  // in steps of 1 degree
    srv0.write(pos);              // tell servo to go to position in variable 'pos'
    read_analog_inputs();
    delay(move_wait_time);                    // waits 15ms for the servo to reach the position
  }
  // expand SRV1 and contract SRV0
  for(pos = MIN_NON_DETATCH_VAL; pos < 180; pos += 1)  // goes from 0 degrees to 180 degrees
  {                                  // in steps of 1 degree
    if ((180-pos) > MIN_NON_DETATCH_VAL){
      srv0.write(180-pos);              // tell servo to go to position in variable 'pos'
    }
    srv1.write(pos);              // tell servo to go to position in variable 'pos'
    read_analog_inputs();
    delay(move_wait_time);                    // waits 15ms for the servo to reach the position
  }
  // Contract SRV1
  for(pos = 180; pos>=MIN_NON_DETATCH_VAL; pos-=1)     // goes from 180 degrees to 0 degrees
  {
    srv1.write(pos);              // tell servo to go to position in variable 'pos'
    read_analog_inputs();
    delay(move_wait_time);                    // waits 15ms for the servo to reach the position
  }

  //contract links
  srv0.write(0);
  srv1.write(0);
  delay(15000);                    //give user 15 seconds to unplug battery
}
```

April Tag Attachment (experimental)

Introduction

One of the capabilities of the physical modular robot platform is being able to determine the position and orientation of each link in a certain environment. This is accomplished through the use of April Tags. April Tags are fiducial markers used in augmented reality, robotics, and camera calibration that can be detected by software to determine the 3D position, orientation, and identity of the tags relative to the camera being used.

In earlier versions of the modular robot platform, a set of April Tags would be individually cut out and attached on all six faces of the link housing, forming a ring. This was done on both ends of the link, as seen below:

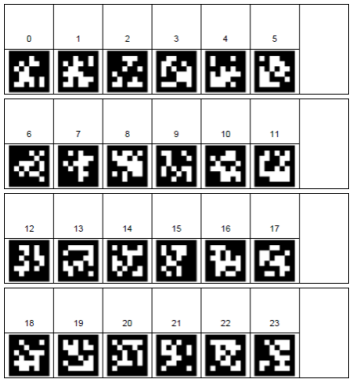


April Tag attachment on previous link design

As one could imagine, cutting each April Tag and individually attaching them is a very tedious process prone to a variety of errors. Firstly, a link assembler using this method to attach April Tags to the link could potentially adhere tags to the link in the wrong order. Links with April Tags attached out of order cannot be tracked by the **camera detection system**. Additionally, individually attaching tags may result in a ring of tags where each tag is slightly staggered when they should be in line with one another. Both of these issues are solved with the atagmain.py script.

atagmain.py Description

The atagmain.py script uses the reportlab package to generate PDFs of strips of April Tags that correspond to a specific link. Each strip consists of a series of seven boxes. Six of these boxes contain images of April Tags and their respective tag number. The last box of the strip is empty, and is used to improve adhesion between the tag and the link.



Example of PDF generated by atagmain.py

The script first checks that you included the path to a folder containing the list of April Tags and that one or more links were selected to generate tags. The list of these links is iterated through, and two sets of tags are generated for each link. For example, if links 0 and 1 are entered when running the atagmain.py script, the PDF generated will contain 4 April Tag strips, the first two for link 0 and the second two for link 1.

The corresponding number of each April Tag is printed above its respective tag image. For example, if links 0 is entered when running the atagmain.py script, the first strip of April Tags will be labeled 0-5, and the tags on the next strip will be labeled 6-11.

Running atagmain.py

The main functionality of atagmain.py is achieved through the ReportLab toolkit. This can be downloaded using the following command line statement:

```
pip install reportlab==3.2.0
```

Next, using the command line, navigate to a directory that the April Tag layout repository can be downloaded into. Download the repository into your current working directory using:

```
git clone https://github.com/GitWyd/RM_AprilTagLayoutGen.git
```

Finally, run the script using:

```
python atagmain.py -i <input file> -l <list of link numbers>
```

The flag -i is followed by the name of the input file to be read. The input file used to generate tags should always be tags36h11_big, as the scaling factors used in atagmain.py were based on the image dimensions in tags36h11_big. The flag -l is a list of links that April Tag strips will be generated for. For example, the command line statement

```
python atagmain.py -i tags36h11_big -l 0 1
```

Will produce four strips total, the first two for link zero, and the last two for link one.

April Tag Strip Attachment

In order to effectively convey how April Tag strips are attached to the link housing, a coordinate system will be used to ensure the April Tag strips are adhered properly.

Prior to placing April Tag strips on the link body, ensure the link is positioned as it is during assembly, such that servo zero extends to the left and servo one extends to the right. In this configuration, assume servo zero extends in the -z direction and servo one extends in the +z direction. Two strips will be adhered to the link housing, one on both right and left ends of the link.

Once a PDF of April Tag strips has been generated and printed out, each strip must be individually cut. Every set of two consecutive April Tag strips generated will be attached to one link, and each April Tag image is labeled with a unique number. The mapping between link number and corresponding April Tag numbers is shown below:

For each link l:

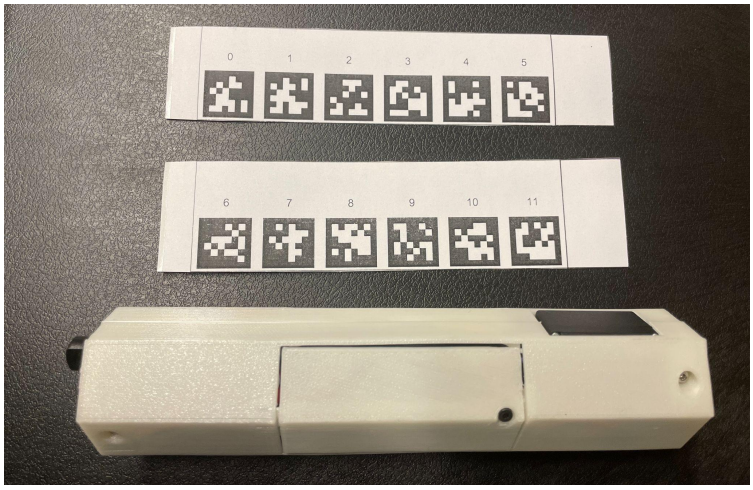
Strip 0 contains tag numbers: $12 \times l$ to $12 \times l + 6$

e.g. $12 \times l$	$12 \times l + 1$	$12 \times l + 2$	$12 \times l + 3$	$12 \times l + 4$	$12 \times l + 5$	*Extra Tag*

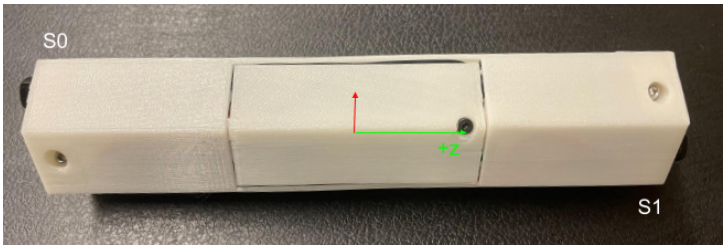
Strip 1 contains tag numbers: $12 \times l + 6$ to $12 \times l + 11$

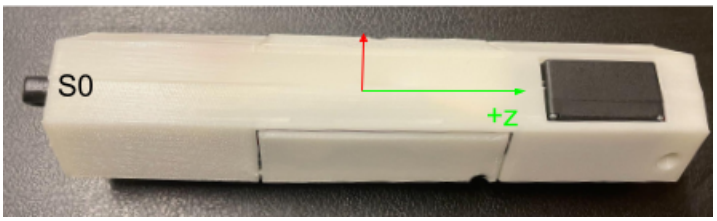
e.g. $12 \times l + 6$	$12 \times l + 7$	$12 \times l + 8$	$12 \times l + 9$	$12 \times l + 10$	$12 \times l + 11$	*Extra Tag*

With this mapping, it can be seen that one of the two strips contains a set of lower consecutive numbers while the other strip contains a set of higher consecutive numbers.

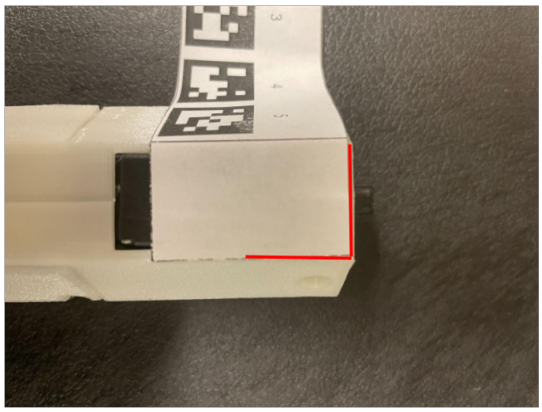


Prior to placing Apriltag strips on the link body, ensure the link is positioned as it is during assembly, such that servo zero extends to the left and servo one extends to the right. In this configuration, assume servo zero extends in the -z direction and servo one extends in the +z direction.

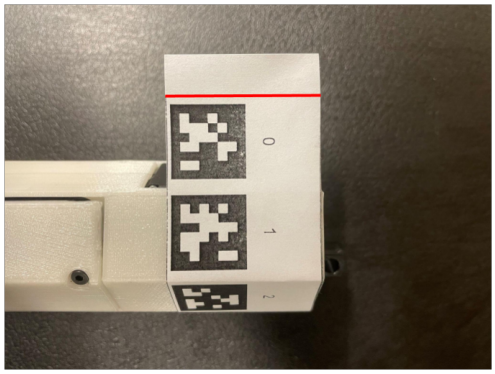




Two strips will be adhered to the link housing, one on both right and left ends of the link. The extra tag on the strip containing the set of lower consecutive numbers is placed over the body of servo zero and wrapped clockwise about the +z axis previously designated, ensuring that the strip is aligned with the edges of the link (indicated with red lines shown below).

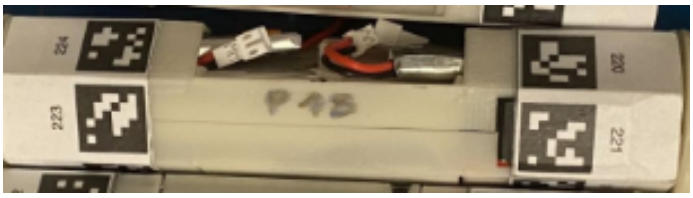


Continue wrapping the strip around all of the link, ensuring that each Apriltag image is centered over each face. Finally, cut any excess paper off the strip (indicated with red lines in the image below) and fold the last tag such that it overlaps the first tag that was placed.



Look towards the opposite side of the link where the body of servo 1 (black rectangle) is located. Rotate the link such that servo 1's body is facing upwards. Take the blank tag of the strip containing the set of larger numbers and place it on servo 1's body such that, when wrapped, the tags will face inwards. Ensure that the strip is aligned with the edges of the link (indicated with red lines in the image below).

Wrap the strip clockwise about the +z axis, again ensuring that each April Tag image is centered over each face and cut any excess paper off the strip. Then, protect both April Tag strips by wrapping them in a couple strips of tape each. The final result is shown below.



Bonus: Truss Link fully contract script for testing

```
/*********
 * Modular Robot Control
 * Simple Fully Contract Link
 * *******/
int move_wait_time = 140;

Servo srv0;
Servo srv1;

int srv0_sens = 0;
int srv1_sens = 0;
int voltage_raw = 0;
int srv0_pos = 0;
int srv1_pos = 0;
int color_counter = 0;
int MIN_NON_DETATCH_VAL = 60;
int FULL_CONTRACTED_VAL = 40;
// temporary test variables
int pos;

void setup() {
  // attaches the servo on the A4,A5 pin to a servo object
  srv0.attach( D0 );
  srv1.attach( D1 );
  srv0.write(MIN_NON_DETATCH_VAL);
  srv1.write(MIN_NON_DETATCH_VAL);
  delay(8000);


  // readout of position
  srv0_sens = analogRead( A0 ); // attach position readout of SRV1
  srv1_sens = analogRead( A1 ); // attach position readout of SRV2

  // readout voltage value
  voltage_raw = analogRead( A2 ); // readout raw voltage value

  Particle.variable(  "srv1_sens" , &srv0_sens , INT );
  Particle.variable(  "srv2_sens" , &srv1_sens , INT );
  Particle.variable(  "voltage_raw" , &voltage_raw , INT );

  RGB.control(true);

}
void read_analog_inputs(){
        srv0_sens = analogRead(A0);  // read the analogPin
        srv1_sens = analogRead(A1);  // read the analogPin
        voltage_raw = analogRead( A2 ); // readout raw voltage value
}

void loop() {
    // take control of the LED
    color_counter = (color_counter+1)%255;
    // red, green, blue, 0-255.
    // the following sets the RGB LED to white:
    RGB.color(255, 0, 0);          // tell servo to go to position in variable 'pos'
    srv0.write(FULL_CONTRACTED_VAL);
    srv1.write(FULL_CONTRACTED_VAL);
    delay(1000);
}
```