SKOOTR Assembly Guide

Adam Hung, adamhung@umich.edu

CAD and Code:
https://www.dropbox.com/scl/fo/8pi8wrw467ds00p9o7x7i/h?rlkey=7f2q9ez5t7mopjb8u792ng1bj&dl=0

Parts Required

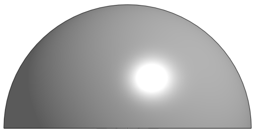
Off the shelf parts:
Alternative brands can be used for most parts, but if using different servos the servo mounts will likely need to be redesigned.

Bill of Materials:
https://docs.google.com/spreadsheets/d/1xyCOFo6V5J4kJTt5ITV_mHOmfF7aZpa3BmCuyWA0sX0/edit?usp=sharing

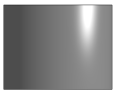
Printed Parts:
All parts should be printed in PLA with a 0.12 mm or smaller layer height, and oriented as shown. Use a functional infill pattern such as "Cubic" in Ultimaker Cura. No brims or rafts are necessary, and support should be used as indicated.

Ball:

Hemisphere (x2), hemisphere_connector
- **10% infill**
- **Use supports for hemisphere**

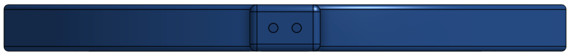


Hemisphere

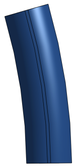


hemisphere_connector

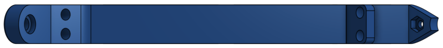
Frame:
Ringconstraint (x3), ringconstraint_connector (x3), triconstraint (x3), servo_hub_bottom, electronics_platform, lid, big_washer (x6, or as many as needed, see frame assembly)
- **20% infill**
- **No support**



Ringconstraint

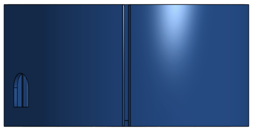


ringconstraint_connector

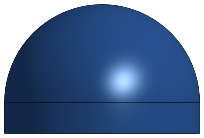


triconstraint

servo_hub_bottom



Electronics_platform

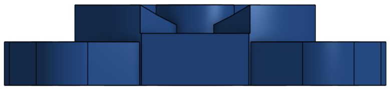


lid

big_washer

Triconstraint_hub

- **20% infill**
- **Use support, touching build plate only**

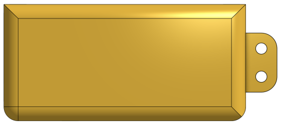


Triconstraint_hub

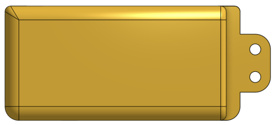
Legs:

Servo_mount_60kg (x3), servo_mount_slotted_60kg (x3), link1 (x3), link2p1 (x3), link2p2 (x3), pusher_servo_mount (x3), sliding_cylinder (x3), link2p3 (x3)

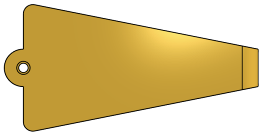
- **20% infill**
- **Use support, touching build plate only**



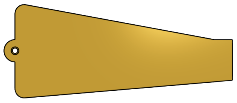
Servo_mount_60kg



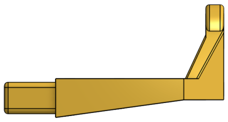
servo_mount_slotted_60kg



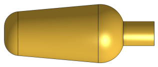
link1



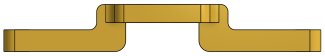
Link2p1



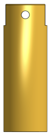
link2p2



link2p3



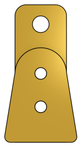
pusher_servo_mount



sliding_cylinder

Horn_sleeve (x3), actuator_arm (x3)

- **20% infill**
- **No support**



Horn_sleeve

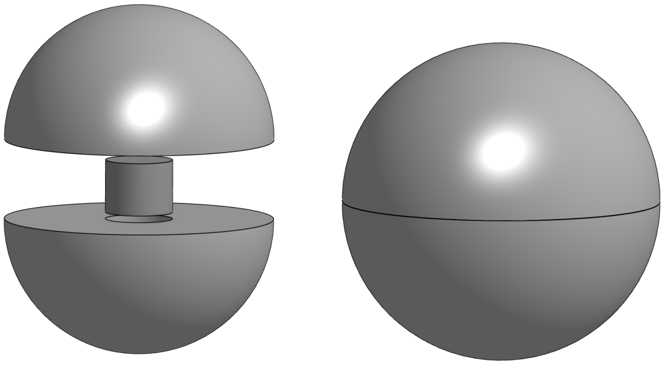
actuator_arm

Assembly:

Begin by removing all the support structures from the printed parts, then follow the steps below in order with the help of the images shown (**fasteners not shown in 3d renderings).

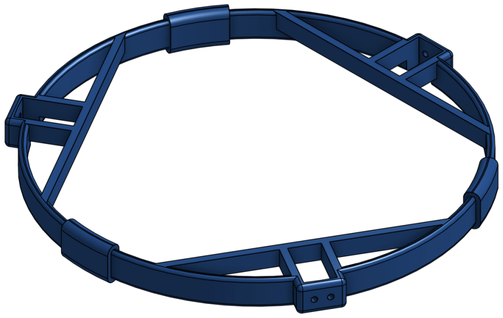
Center ball:

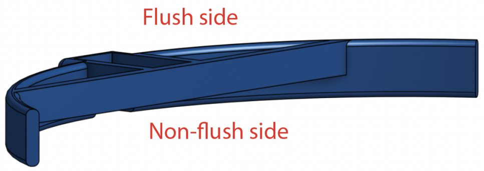
Sand both hemispheres until layer lines are no longer visible. Sand the hemisphere_connector until it fits into the hemispheres. Also lightly sand the flat faces of the hemispheres for better adhesion. Then apply super glue to the flat faces, and compress the three pieces together until dry.



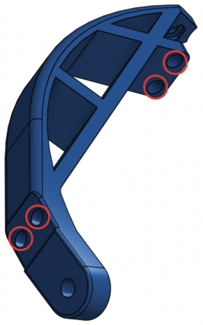
Frame:

Connect the three ring_constraints with the three ring_constraint_connectors, sanding for a snug fit where necessary. Apply super glue in the case of a loose or sliding fit. Ensure all ring_constraints are oriented with the flush side on top, as shown.

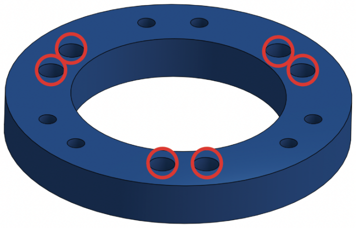


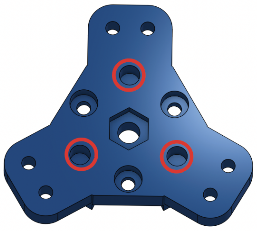


Next, install four m4 brass inserts into the four holes in each triconstraint (circled in red). This can be done with an insert machine, or a simple soldering iron (a heat insert tip soldering iron attachment makes things easier, but isn't necessary). See this video if you are not familiar with the technique: https://www.youtube.com/watch?v=hwq15qH-4x4.

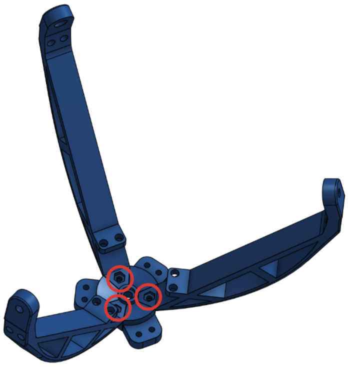


Also install six m4 inserts into the bigger holes in servo_hub_bottom, and three more into the triconstraint_hub.

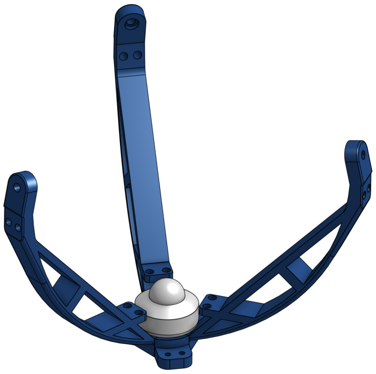


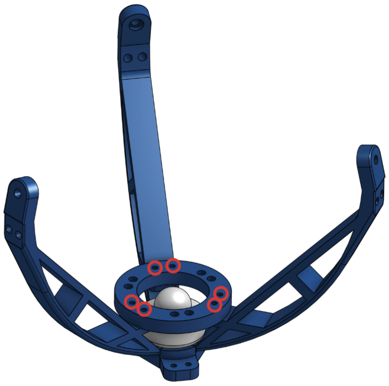


Now, insert one m4 nut into the hexagonal notch in each triconstraint, and connect the triconstraint_hub to the three triconstraints using three 10mm m4 bolts.

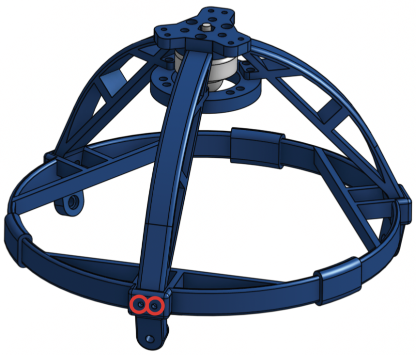


Insert a 1/4"-20 thread size nut into the back of the triconstraint_hub, and tightly screw a spherical bearing into it as shown. Now bolt servo_hub_bottom onto the three triconstraints, with the bolts facing down in the orientation shown (bolt heads where the red circles are). Use six 16mm m4 bolts.

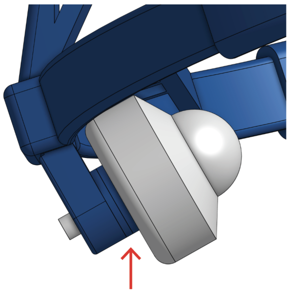


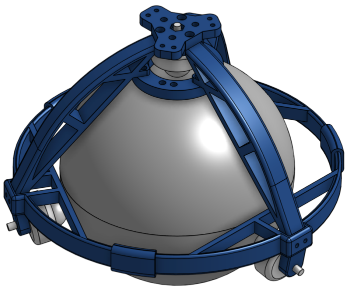


Take the triconstraint assembly and insert it into the ring_constraint assembly, and bolt them together using six 16mm m4 bolts. Make sure the crossbeams of the ring_constraint assembly are oriented so that they are flush on the top (as discussed above).

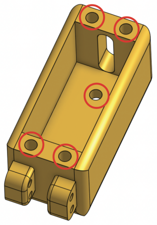


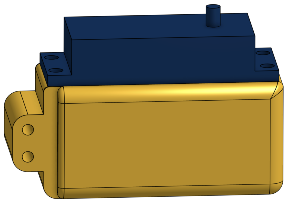
Now screw three more spherical bearings onto the bottoms of each triconstraint, securing with 1/4"-20 thread size nuts (nuts not shown in image). Add big_washers between the bearings and the triconstraints until when inserted, the center ball snaps snugly into the rest of the frame, but still spins freely. Around two bearings on each triconstraint worked best for me. After achieving a good fit, you can remove the ball until the end of the assembly to make things more stable (although all of my images show it).

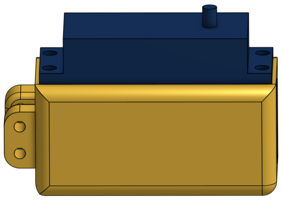




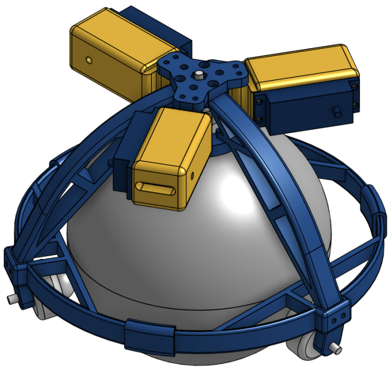
Before moving on we need to insert the 60kg servos into their mounts. Start by installing one m4 insert into the back of each 60kg servo mount (both slotted and non-slotted), and four m4 inserts into the holes on the top. Then push in the servos, with the servo cables fitting into the slots in the sides. Bolt them in with four 8mm m4 bolts and m4 washers.





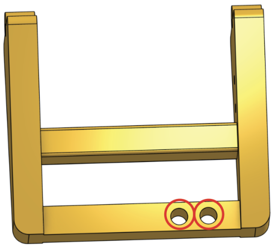


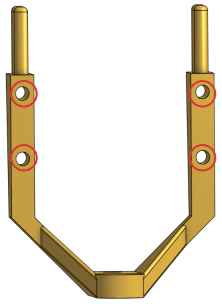
Now we can take the non-slotted servo+mounts and slide them into place between the triconstraint_hub and the servo_hub_bottom, so that the servos face clockwise from a top view. Using six 50mm m4 bolts, bolt the three servo mounts into place all the way from the top of the triconstraint_hub through the bottom of servo_hub_bottom.

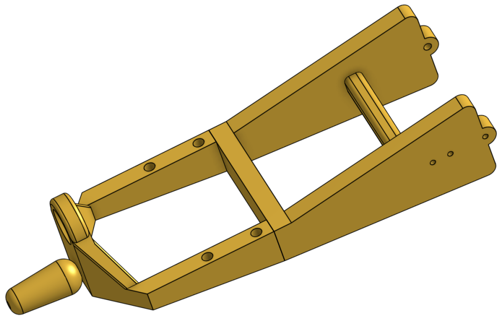


Legs (x3):

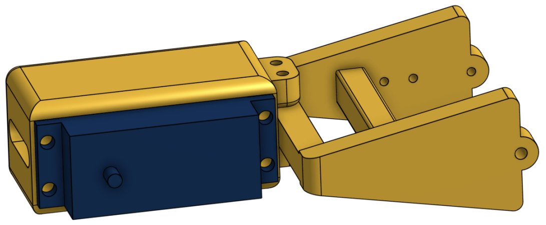
Install the inserts for link1 (two m4s), and link2p2 (four m4s). You can press link2p1, link2p2, and link2p3 together at this point, sanding and gluing if necessary.



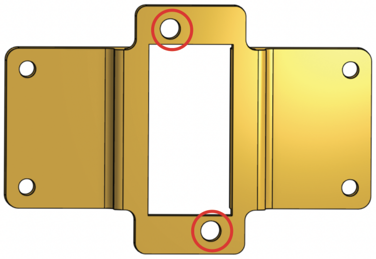


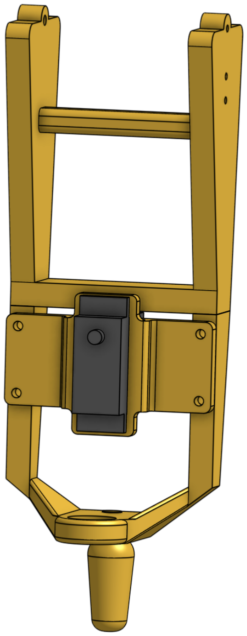


Bolt a link1 to a slotted 60kg servo mount using two 30mm m4 bolts.



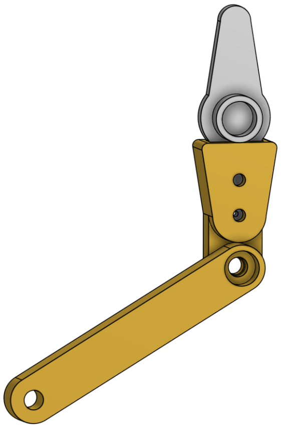
Now, install two m3 inserts into a pusher_servo_mount, and fasten a 20kg servo to the mount using two 6mm m3 bolts and m3 washers. Using four 12mm m4 bolts, bolt the servo and mount to link2p2, in the orientation shown.

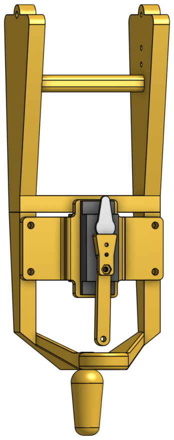




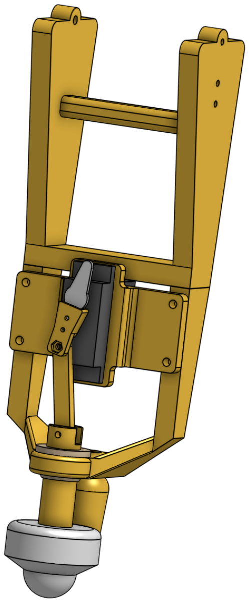
Use an m3 nut and 6mm m3 bolt to attach a horn_sleeve to an actuator_arm, tightening enough that it will stay in place but not so much that the joint can't rotate.

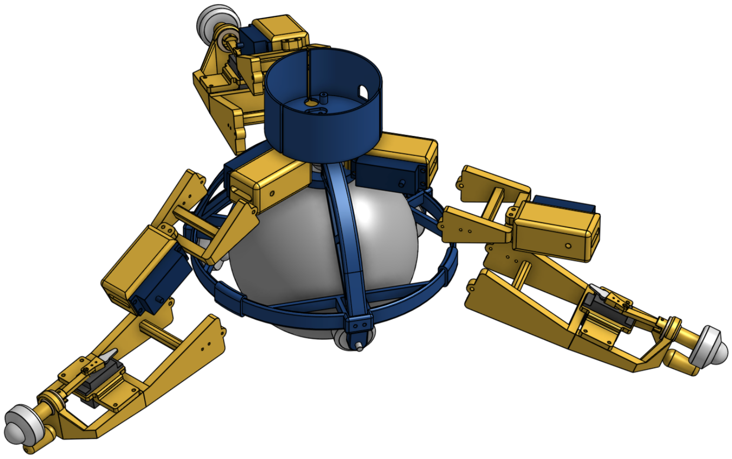
Now use the self-tapping screws and servo horn that come with the 20kg servo to screw a servo horn into the horn_sleeve, and temporarily fix this assembly to the servo as shown.





Screw a spherical bearing tightly into a sliding_cylinder (glue can be used if this threading gets stripped), and hammer a brass bushing into link2p2 so that the wider part of the bushing is on top. Now slide the sliding_cylinder into the bushing, and bolt the sliding_cylinder to the actuator arm with a 16mm m3 bolt and nut (there is a hexagonal space for the nut on the sliding_cylinder).

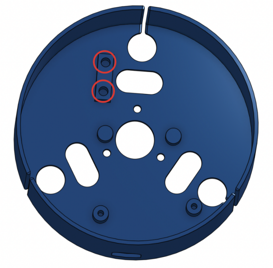




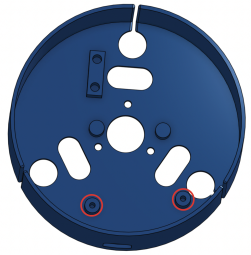
At this point, remove the servo horns from the 20kg servos for now. To continue with assembly we now need to get everything wired together.

Electronics:

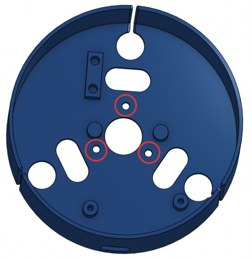
Begin by installing two m3 inserts into the two holes in the small rectangular extrusion at the bottom of the electronics_platform. Secure the MPU-6050 to these holes using two 6mm m3 bolts.



Using two 8mm m3 bolts, bolt the arduino to the electronics_platform using the holes in the two small cylindrical extrusions, and allowing the other side of the arduino to rest on the solid cylindrical extrusions.



Using three 6mm m4 bolts, secure the electronics_platform to the triconstraint_hub through the three radial holes that align between the two parts.

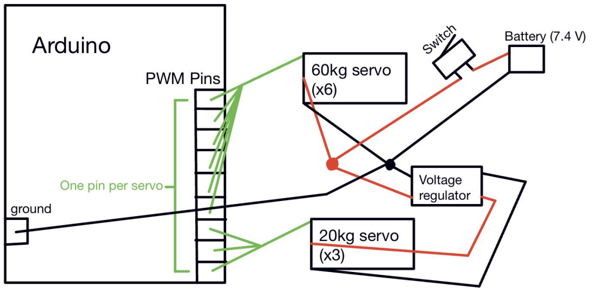


Each servo has three pins corresponding to power, ground, and signal (PWM). Using jumper wires, wire all of the 60kg servo power pins together in parallel (I just used a breadboard), and separately connect the 20kg servo power pins together in parallel. Now connect all of the 60kg as well as the 20kg servo ground pins together in parallel. Now wire the battery through a power switch to connect to the 60kg power line and the shared ground, and since the 20kg servos require a 6V power supply, connect a voltage regulator set to 6V to take the 60kg power line (7.4 V nominally) as input and the 20kg power line (6V) as output. Now connect jumper wires to every servo control pin, but don't connect the other ends to anything yet.

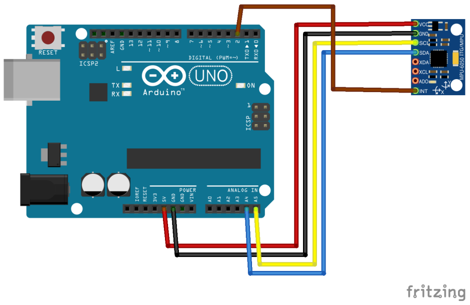
There are three radial slots in the electronics_platform: slide the cables from one leg one by one through one of the slots all the way to the wider opening at the bottom of the platform. Repeat for each leg.

Now connect each signal pin to an appropriate PWM-capable output pin on the Arduino (see code for the pins I used), and place the Arduino and battery inside the electronics_hub.

The power distribution for the servos should be something like this:

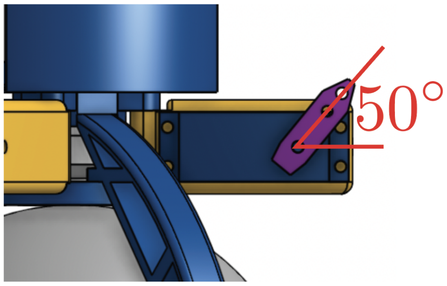


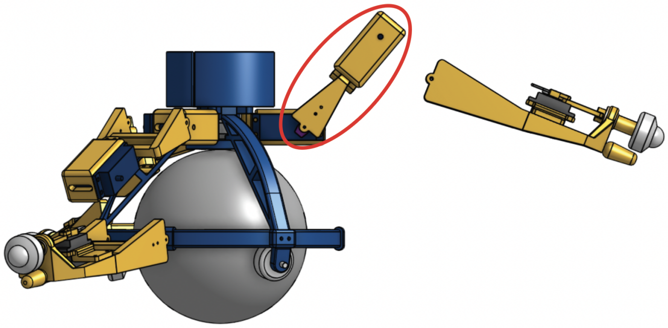
Also wire the MPU-6050 to the arduino as shown:



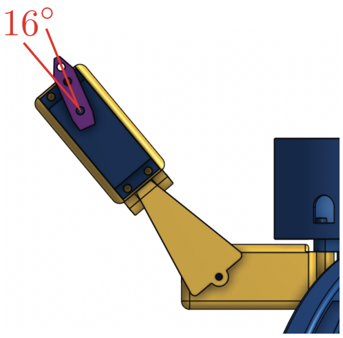
Now upload *SKOOTR_INIT.ino* to the Arduino. The 60kg servos have a limited range of motion, so we must strategically set their zero positions so that we can achieve all of the positions we want within this 180 degree range. The zero positions of the potentiometers inside the servos that we use to control their positions cannot be altered electronically, so we must set these zero positions where we want them as we are assembling the robot. To do this, we will power the servos and command them to go to their 0 positions, attach the links for each leg in roughly the same position, and then tune offset variables for each joint so that each leg will move to the exact position we want it to in our code later on.

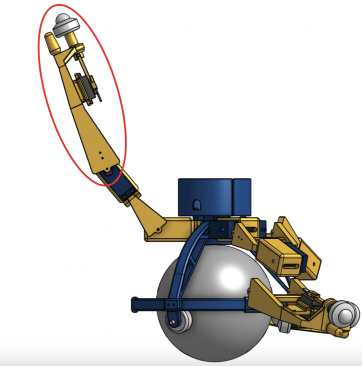
Begin with the upper joint of each leg. With the power turned on and the servos actuated, place the servo horn onto the joint pointing so that if the leg were to move any more clockwise, it would collide with the ring_constraint. This should be about a 50 degree angle up from being straight out. Now use three 16mm m3 bolts to bolt a link1 to each servo horn.



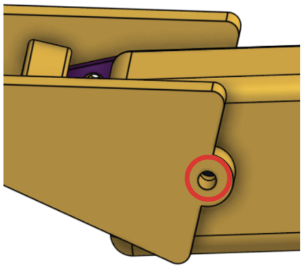


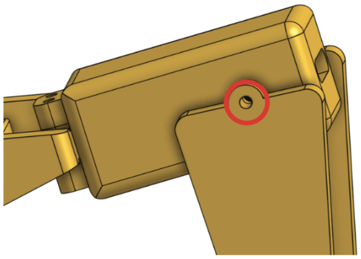
For the second joint, point the servo horns so that they point 15 to 20 degrees up from being straight out, and then bolt a link2p1 to each servo horn.



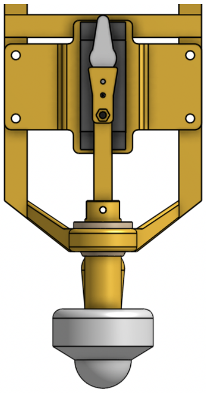


Also use 20mm m4 bolts to secure link1 and link2p1 to the holes in their respective servo_mounts (location of the holes are shown below). An optional bit of glue on the ends of these bolts will make sure leg movements don't loosen them over time.

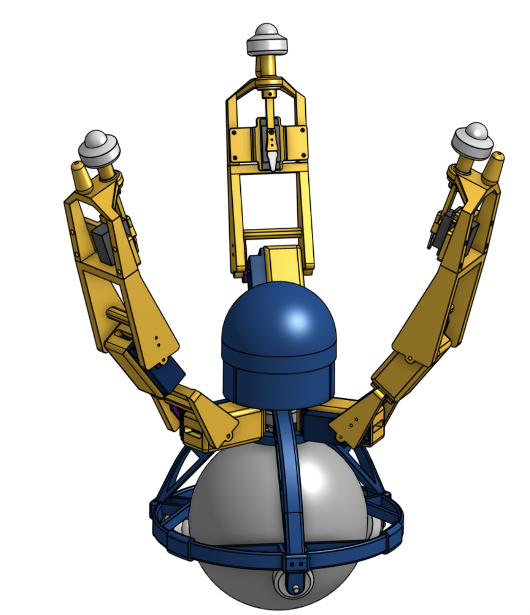




For the 20kg servo joints, (with power on and servos set to 0 position) just adjust the servo horns to point straight down, and then screw them into place.

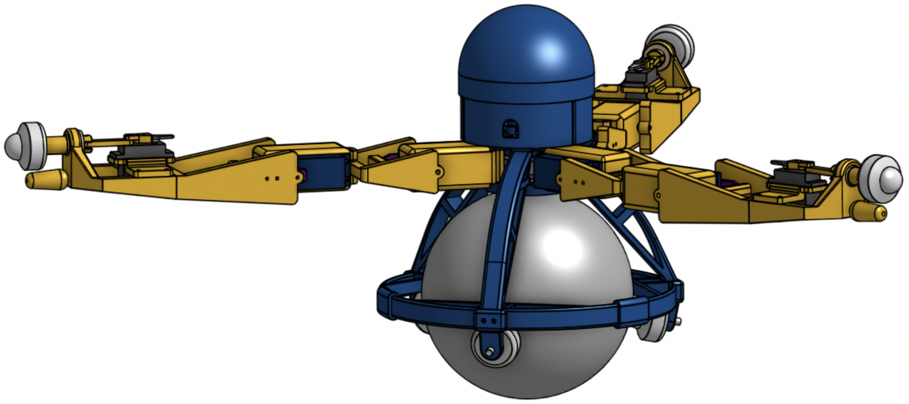


At this point, you can slide the lid onto the top of the robot, and powered on it should look something like this.



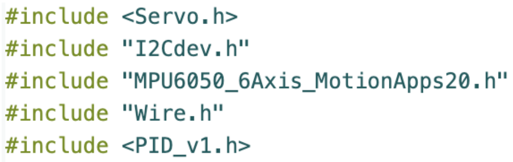
Lastly, we can finish tuning each joint position.

***IMPORTANT**:* Across all of my code, leg1 refers to the leg that is parallel and closest to the MPU-6050 (from a top view). Leg2 refers to leg1's counterclockwise neighbor, and leg3 refers to leg1's clockwise neighbor. I would recommend labeling the legs with some tape or something to avoid confusion.
Change the universaloffset1 and universaloffset2 variables until the legs point more or less straight out, and reupload. Now, change the individual legoffset1 and legoffset2 variables for each leg until each leg is perfectly straight, using an electronic angle finder if available for maximum precision. Taking the center ball out for this step makes it easier. Take note of all of these offset values for the next step.



Setup:

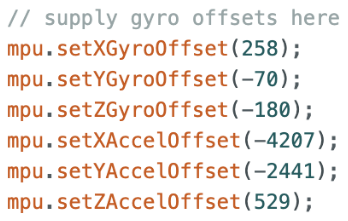
My code uses several common external libraries, which should be installed using the Arduino library manager (for help see https://support.arduino.cc/hc/en-us/articles/5145457742236-Add-libraries-to-Arduino-IDE).



```
#include <Servo.h>
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#include "Wire.h"
#include <PID_v1.h>
```

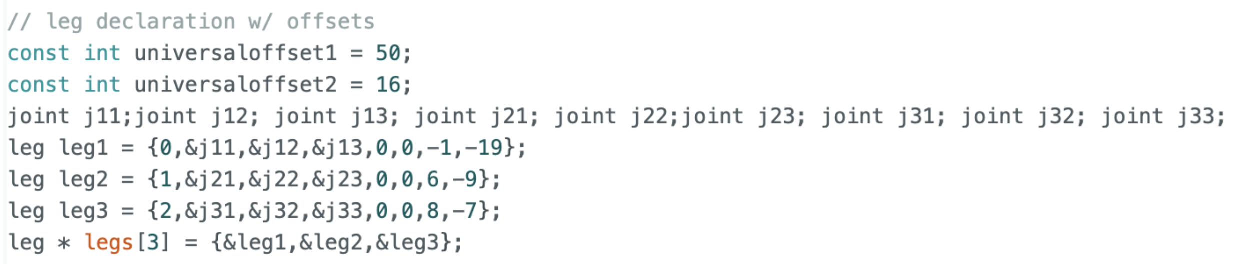
For the mpu6050 to work properly, it needs to be calibrated, so once you have installed the MPU6050 library, from Arduino open File -> Examples -> MPU6050 -> IMU_Zero and follow the directions there to obtain the necessary offset variables.

Now open SKOOTER_MAIN.ino, and paste these values into the gyro offset variables in the mpu setup section of the setup() function.



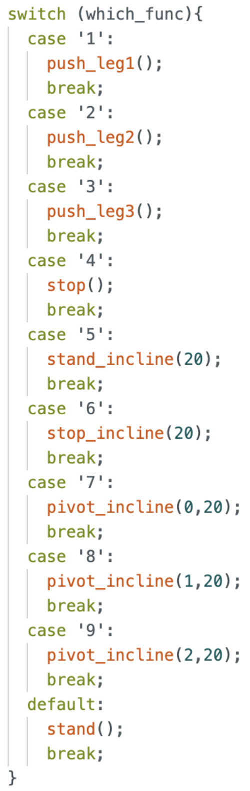
```
// supply gyro offsets here
mpu.setXGyroOffset(258);
mpu.setYGyroOffset(-70);
mpu.setZGyroOffset(-180);
mpu.setXAccelOffset(-4207);
mpu.setYAccelOffset(-2441);
mpu.setZAccelOffset(529);
```

Also insert your leg offsets from the previous step. For each leg, the second to last parameter of the leg declaration is the joint1 offset, and the last parameter is the joint2 offset.



```
// leg declaration w/ offsets
const int universaloffset1 = 50;
const int universaloffset2 = 16;
joint j11;joint j12; joint j13; joint j21; joint j22;joint j23; joint j31; joint j32; joint j33;
leg leg1 = {0,&j11,&j12,&j13,0,0,-1,-19};
leg leg2 = {1,&j21,&j22,&j23,0,0,6,-9};
leg leg3 = {2,&j31,&j32,&j33,0,0,8,-7};
leg * legs[3] = {&leg1,&leg2,&leg3};
```

That's it! Make sure the baud rate in your Serial Monitor is set to 115200, and if you upload the code and then power up the robot, you can run any of the movement functions by entering characters into the Serial Monitor. Here are all of your options (code from the loop() function): You can change the degree of incline you're working with by adjusting the parameters to the incline functions, and by default the robot will just stand.



```
switch (which_func){
  case '1':
    push_leg1();
    break;
  case '2':
    push_leg2();
    break;
  case '3':
    push_leg3();
    break;
  case '4':
    stop();
    break;
  case '5':
    stand_incline(20);
    break;
  case '6':
    stop_incline(20);
    break;
  case '7':
    pivot_incline(0,20);
    break;
  case '8':
    pivot_incline(1,20);
    break;
  case '9':
    pivot_incline(2,20);
    break;
  default:
    stand();
    break;
}
```