Ball Balancer

by aaedmusa

Several months prior to this project I created this ball balancer. It's able to balance a 1.5" 3D printed ball on an 8" x 8" platform. I decided to revisit this project to make an even better ball balancer. This iteration is quieter, less bulky, easier to build, and more precise. Not only can this version balance a ball but it can also move the ball around in different patterns. In this instructable, I'll walk you through the step-by-step process of making a ball-balancing robot.

My Website

Github

Inspiration for this project

Supplies:

Tools

- needle nose pliers
- wire cutters
- wire strippers
- 2.5mm allen wrench
- soldering iron
- exacto knife
- hot glue gun

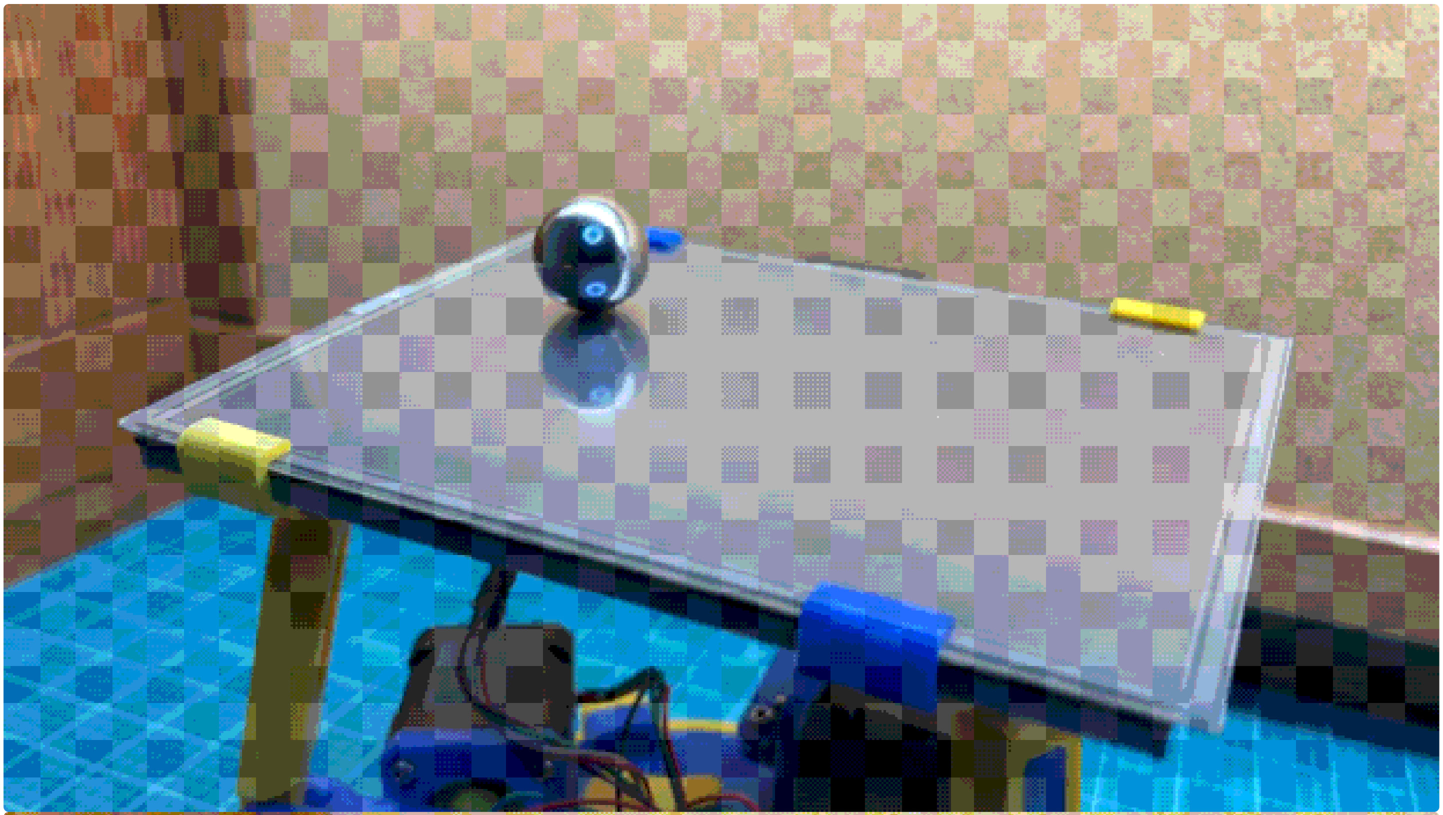
Electronics

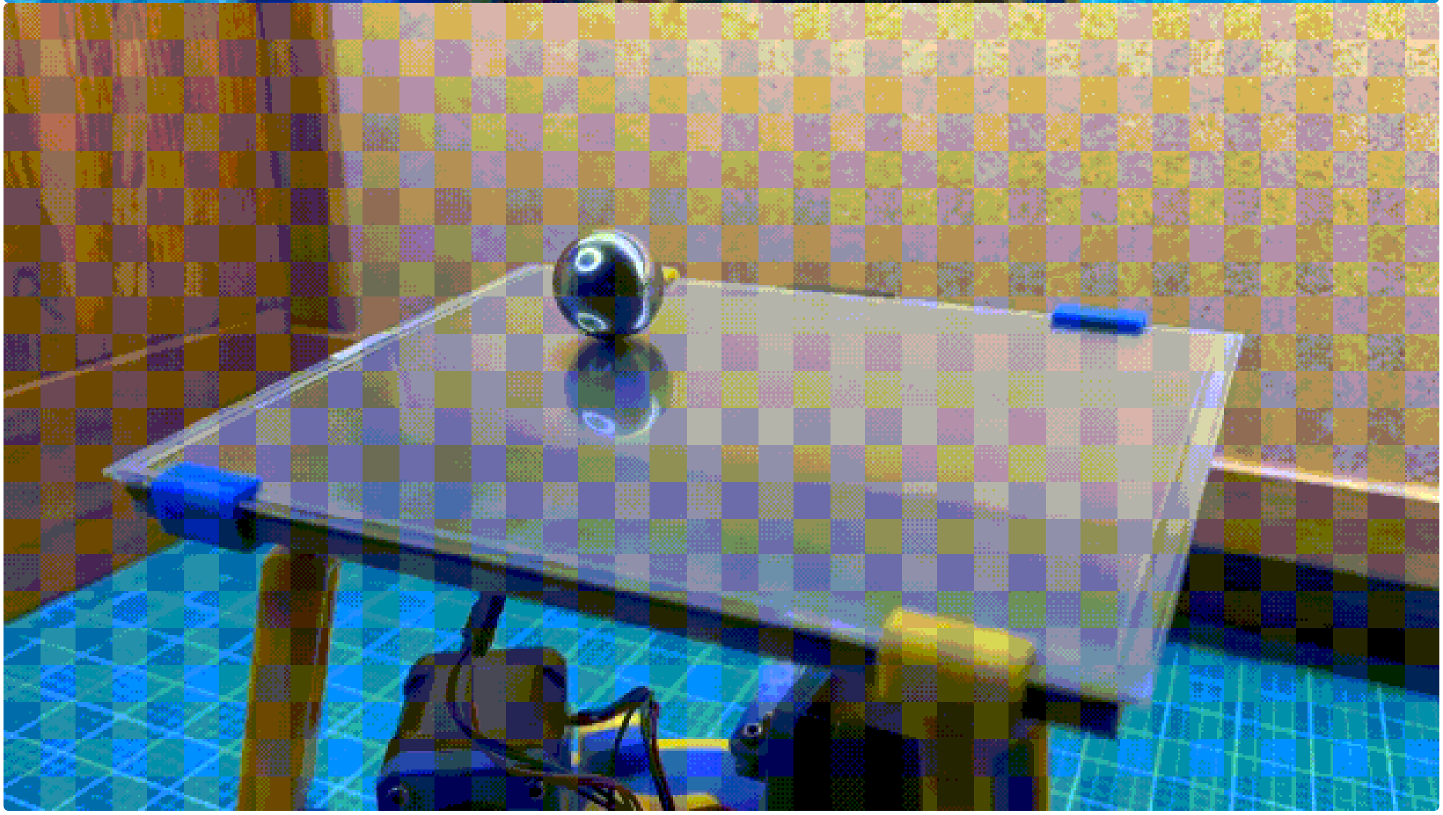
- Teensy 4.1 Microcontroller
- Nema 17 59 Ncm Stepper Motors (Bipolar)
- TMC2208 Stepper Motor Drivers
- Mini Protoboard + Screw Terminals
- 100uF Capacitor 35V
- Male and Female Header Pins
- 30V Bench Power Supply (or any 24V power supply)
- 22 AWG Wire
- 5V Regulator

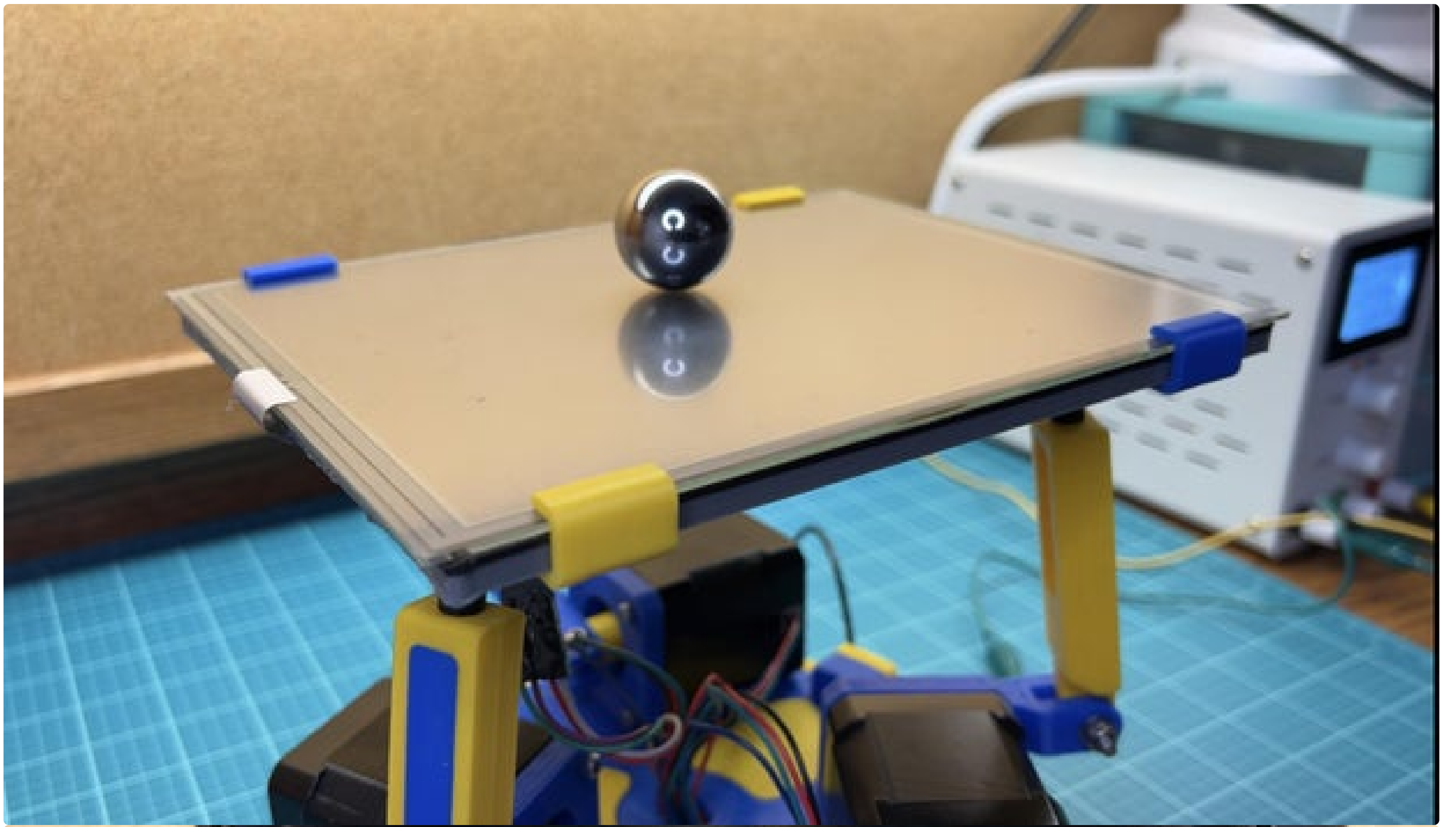
General Parts

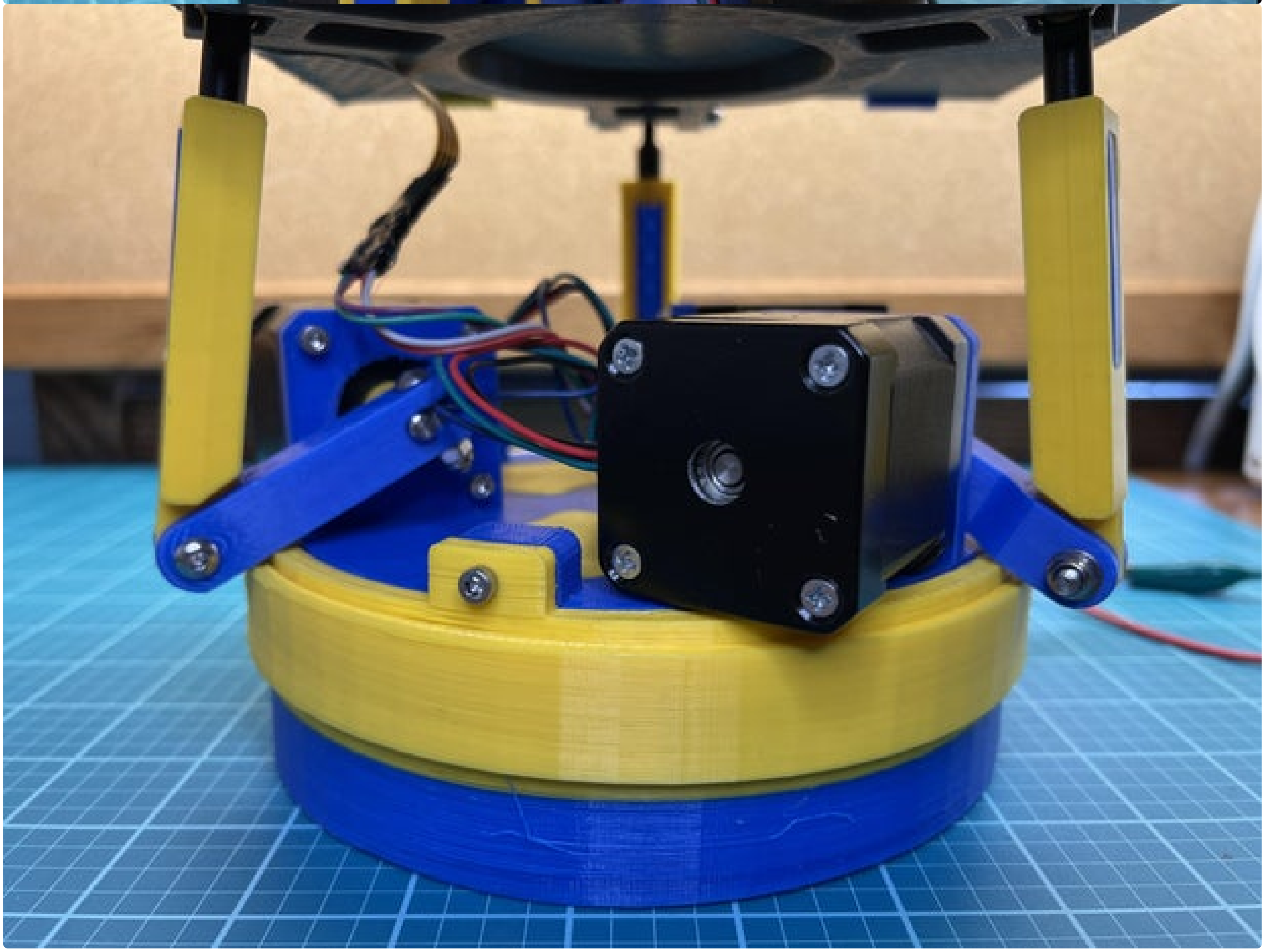
- 8.4" 4 Wire Resistive Touch Panel
- 1" Steel Bearing Ball
- 22mm long m3 tie rod

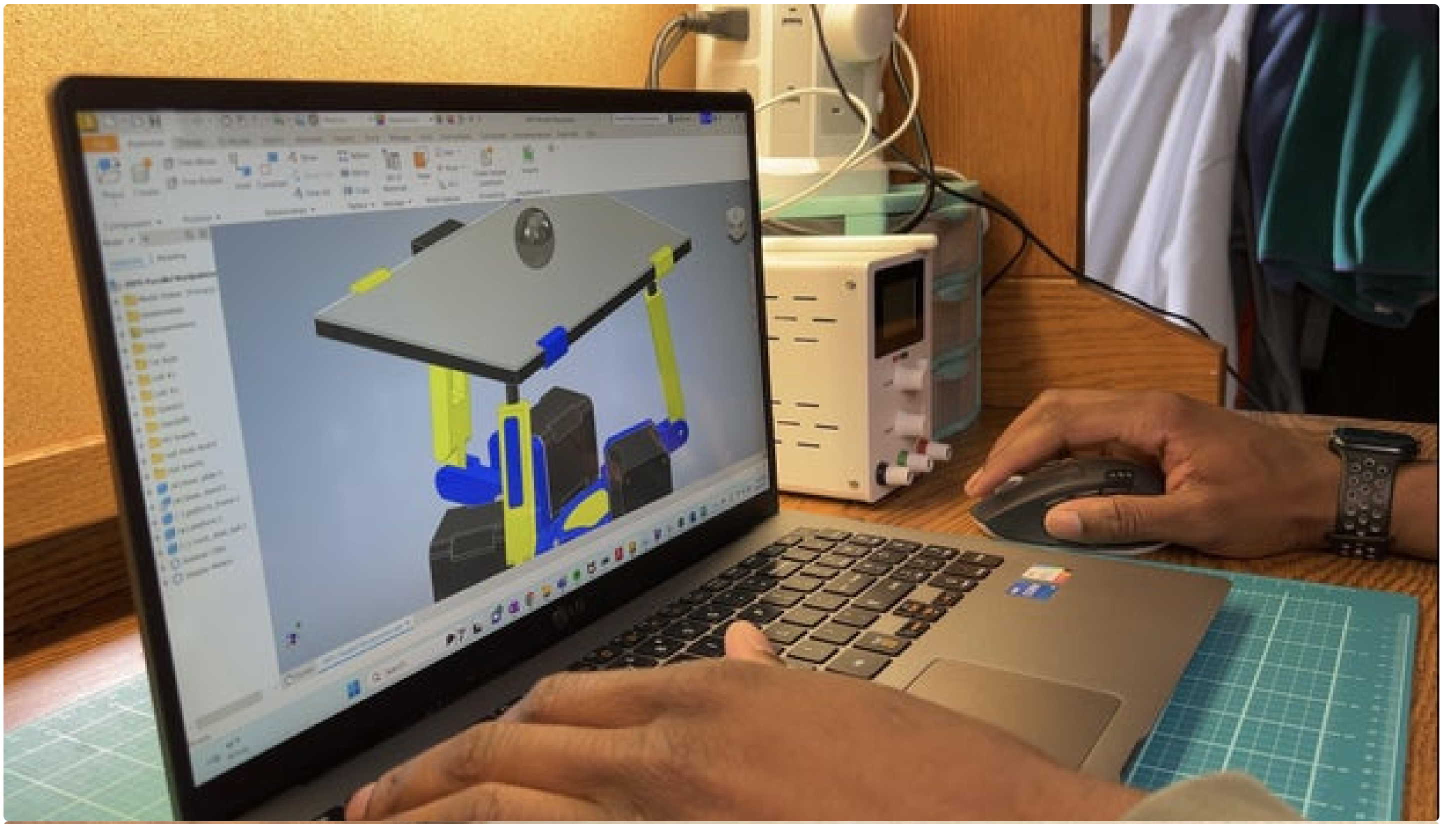
- m3 x 6mm threaded inserts
- M3 x 5mm Standoffs
- M3 x 5mm Screws
- M3 x 8mm Screws
- M3 x 10mm Screws
- M3 x 35mm Screws
- M3 Nylon Locknuts
- M4 x 20mm Screws
- M4 x 25mm Screws
- M4 Nylon Locknuts

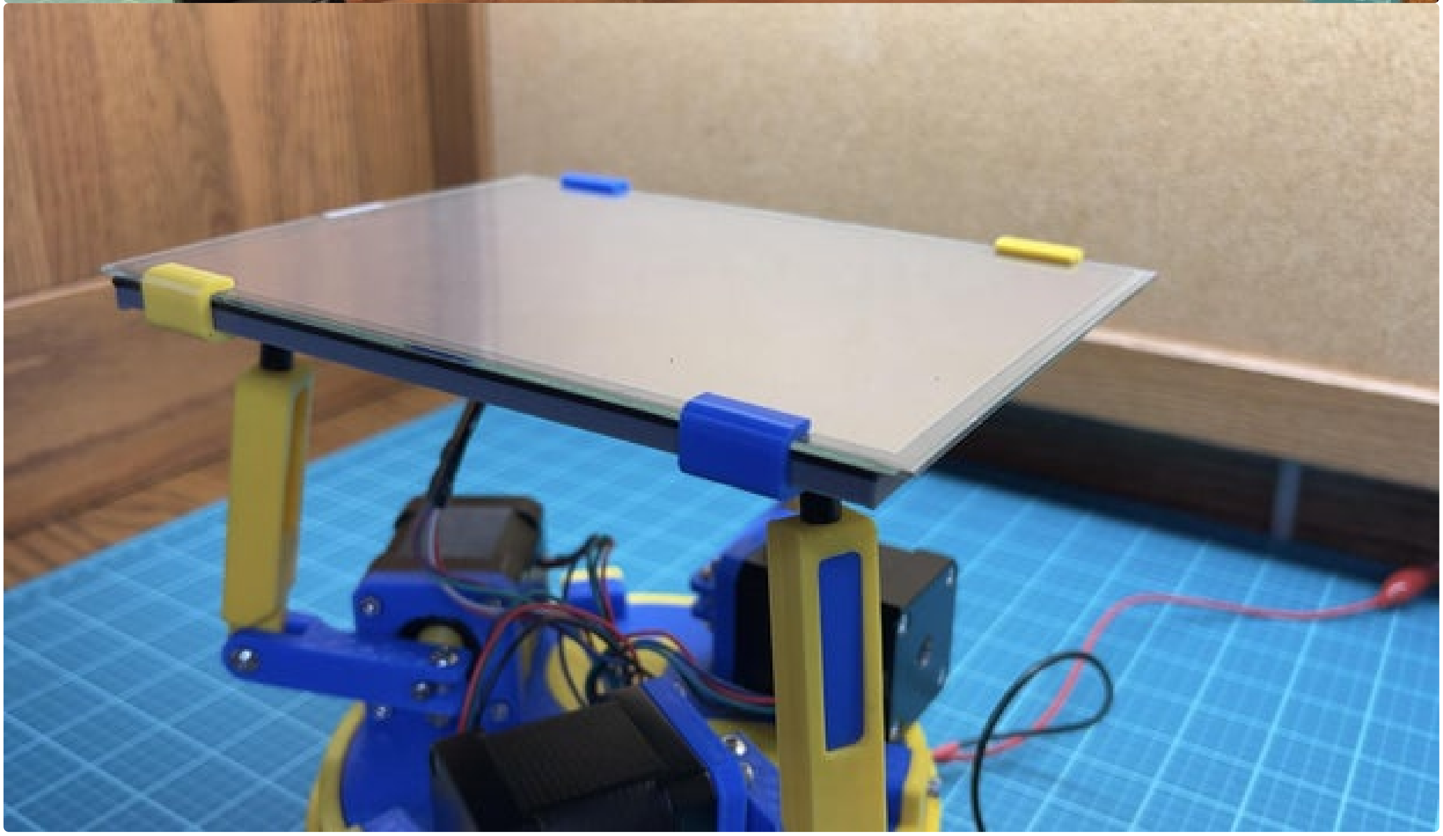


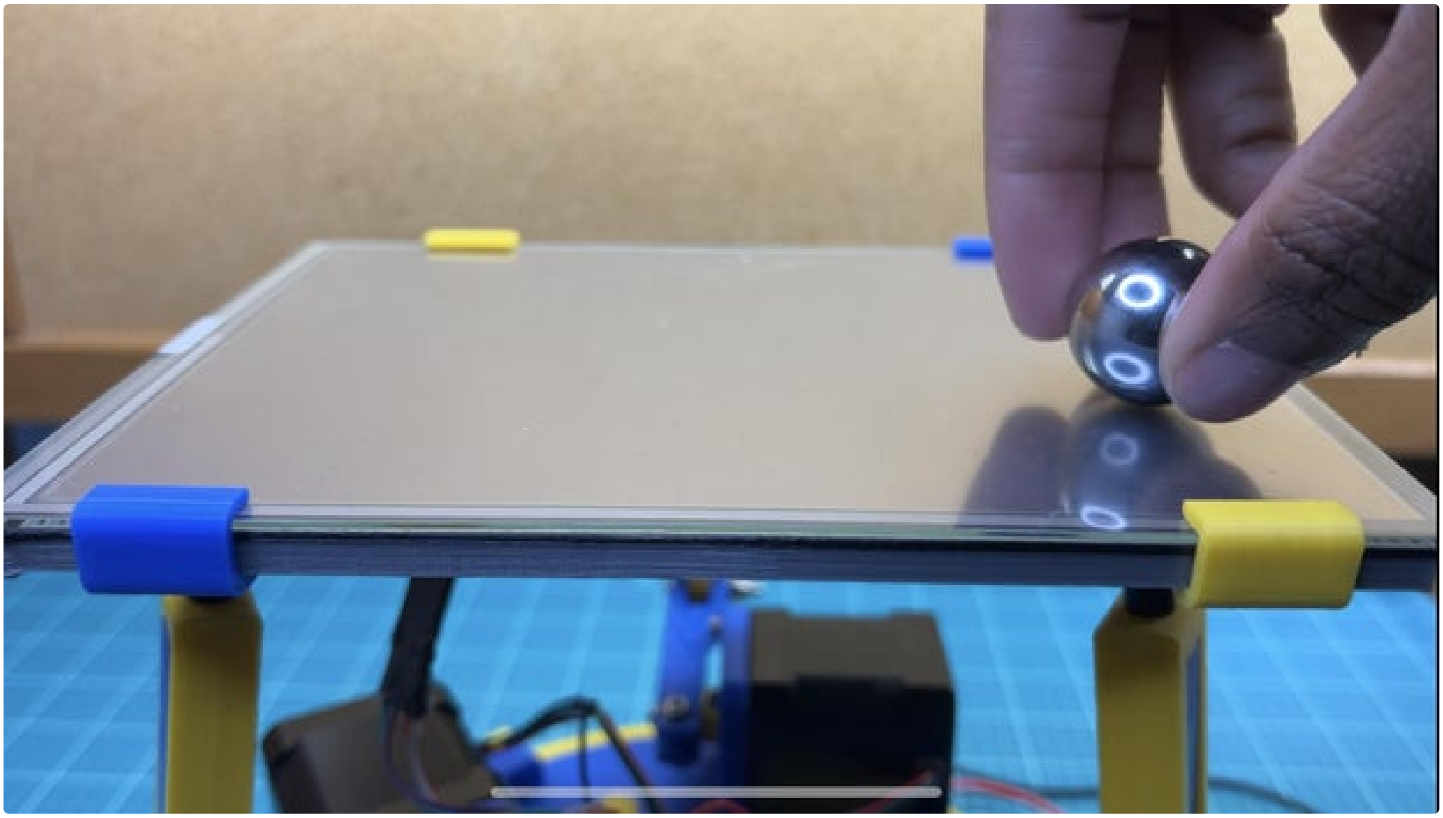












https://youtu.be/v4F-cGDGiEw

Step 1: 3D Print the Parts

Start by printing the parts. There are a total of 9 different parts which go as follows:

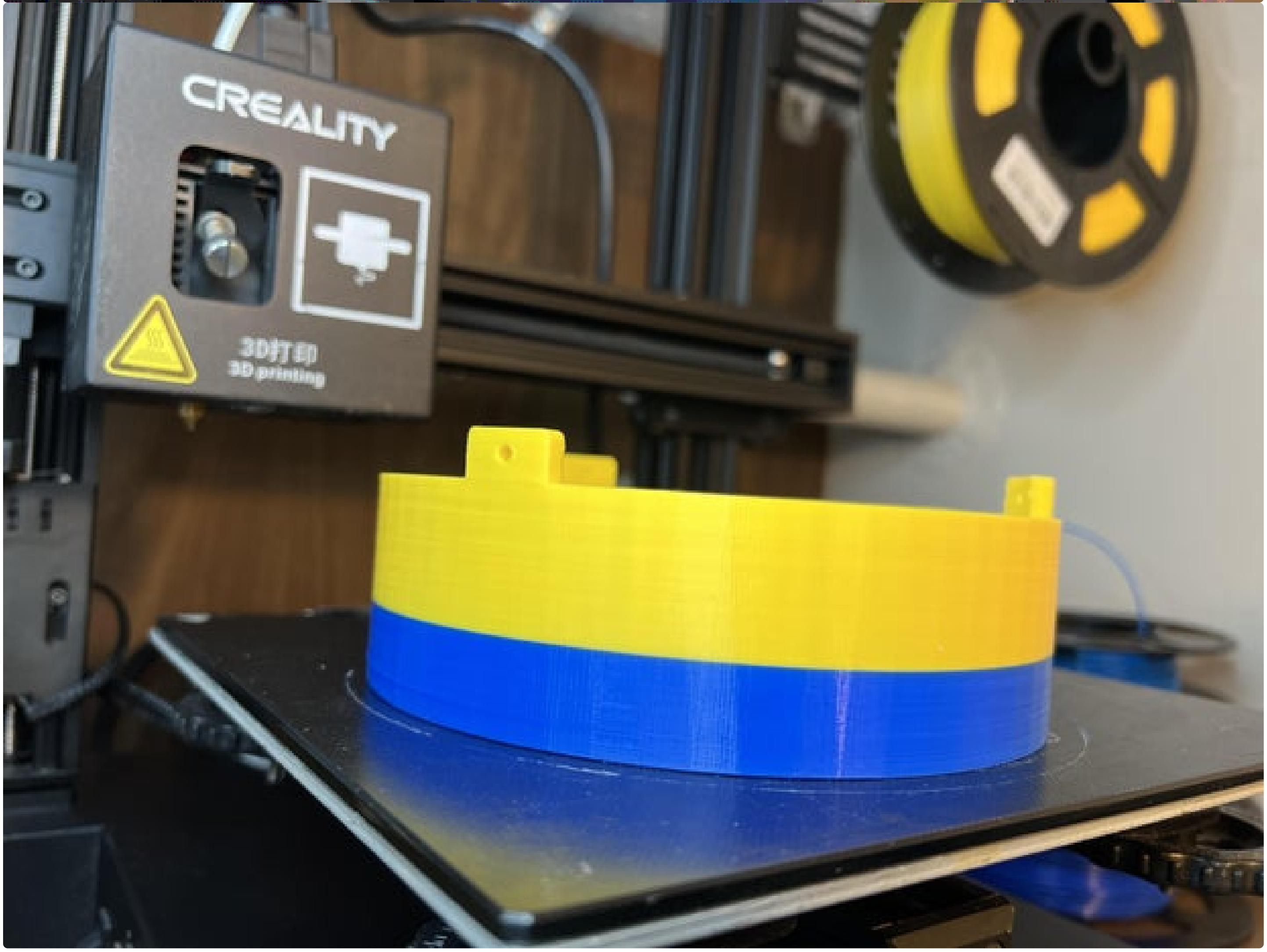
- base_stand
- base_plate
- spacer (x3)
- link_#1 (x3)
- link_#2 (x3)
- platform_frame
- retainer_clip (x4)
- slot_insert_1 (x3 optional)
- slot_insert_2 (x3 optional)

I printed all of these parts in PLA filament. I'd recommend using the following printer settings:

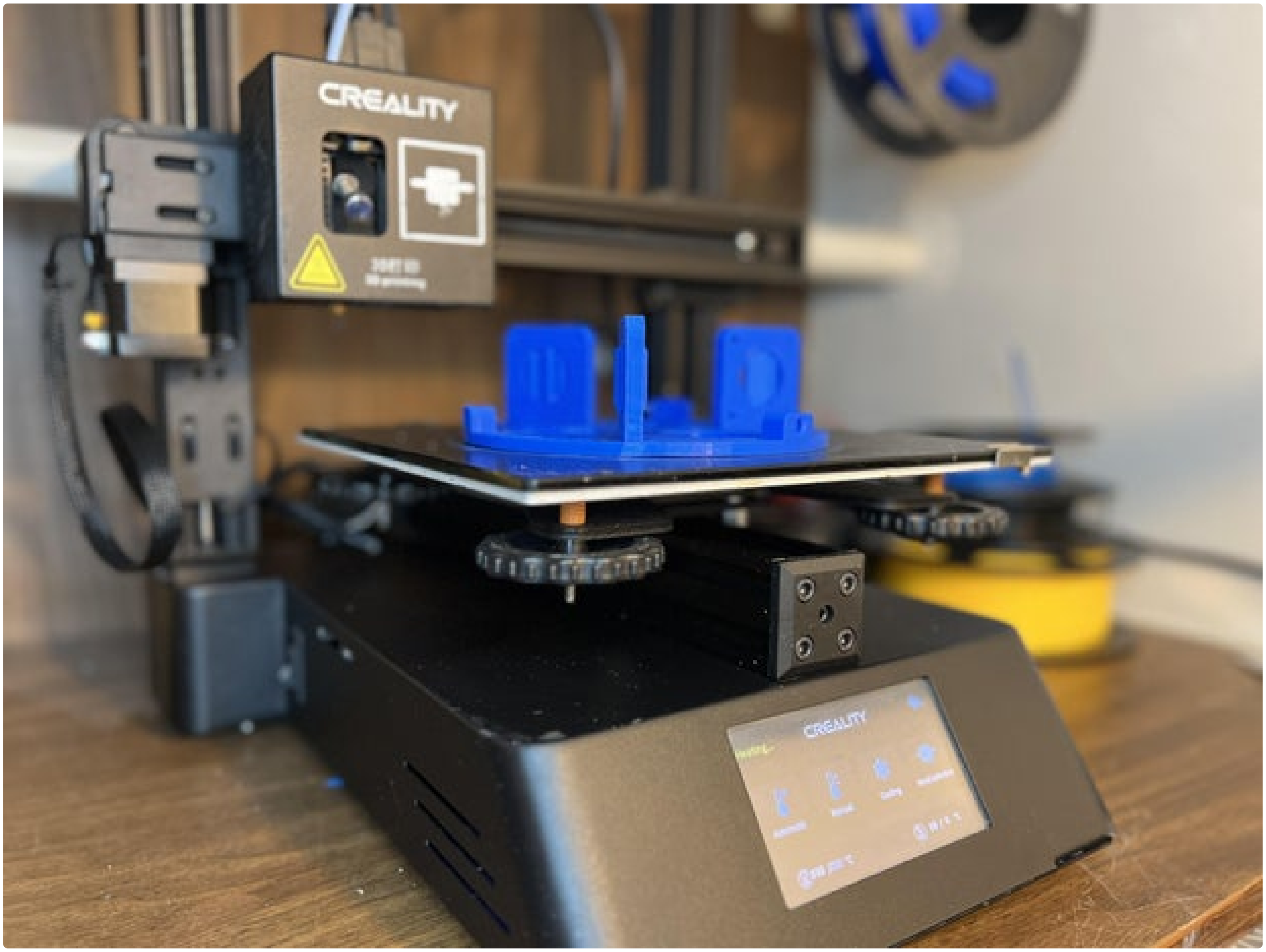
- 20% infill
- Auto-generated supports
- PLA (210° extruder and 45° bed)

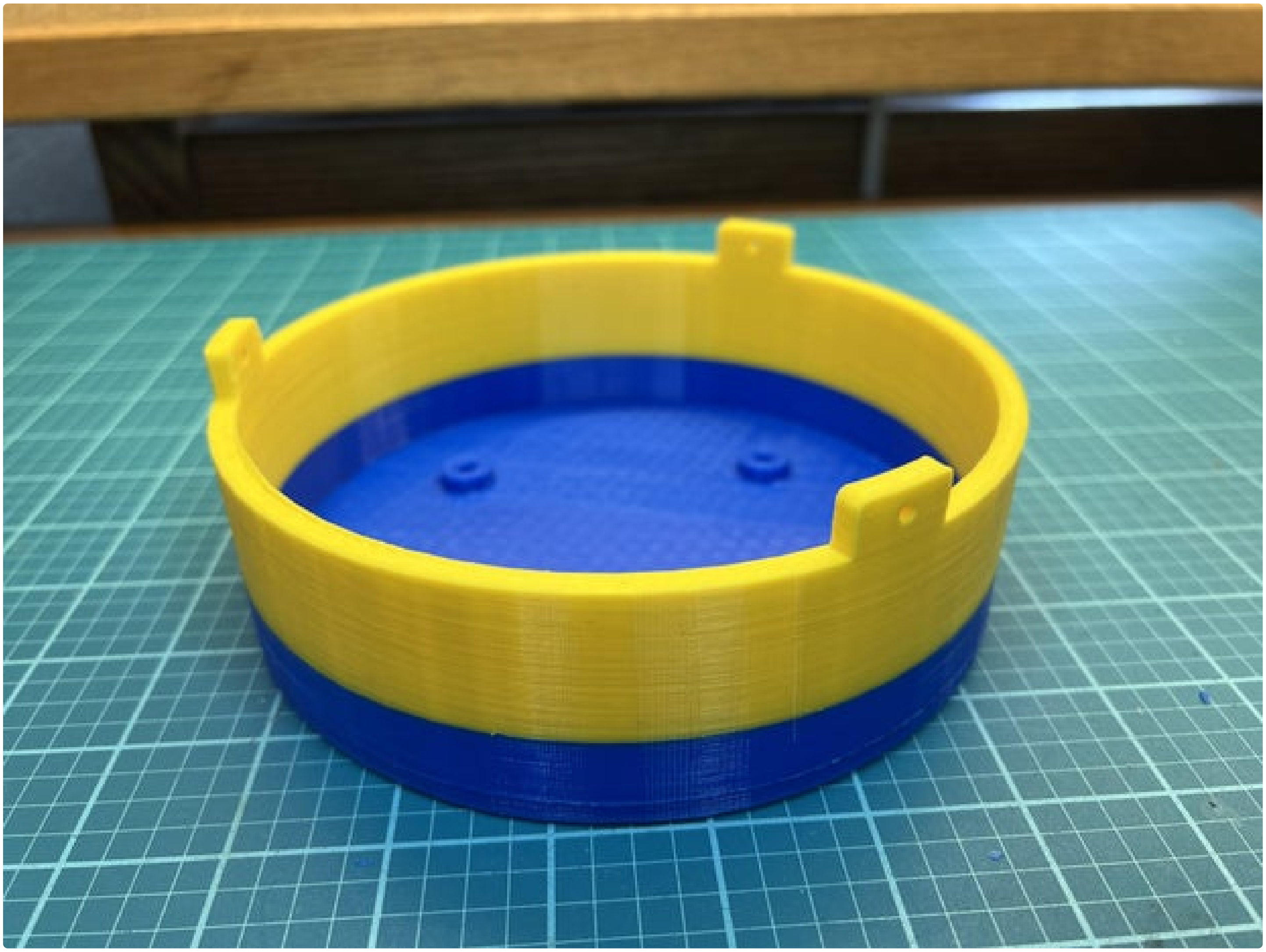
The filament usage is about 310g. It took a cumulative 25 hours to print everything.

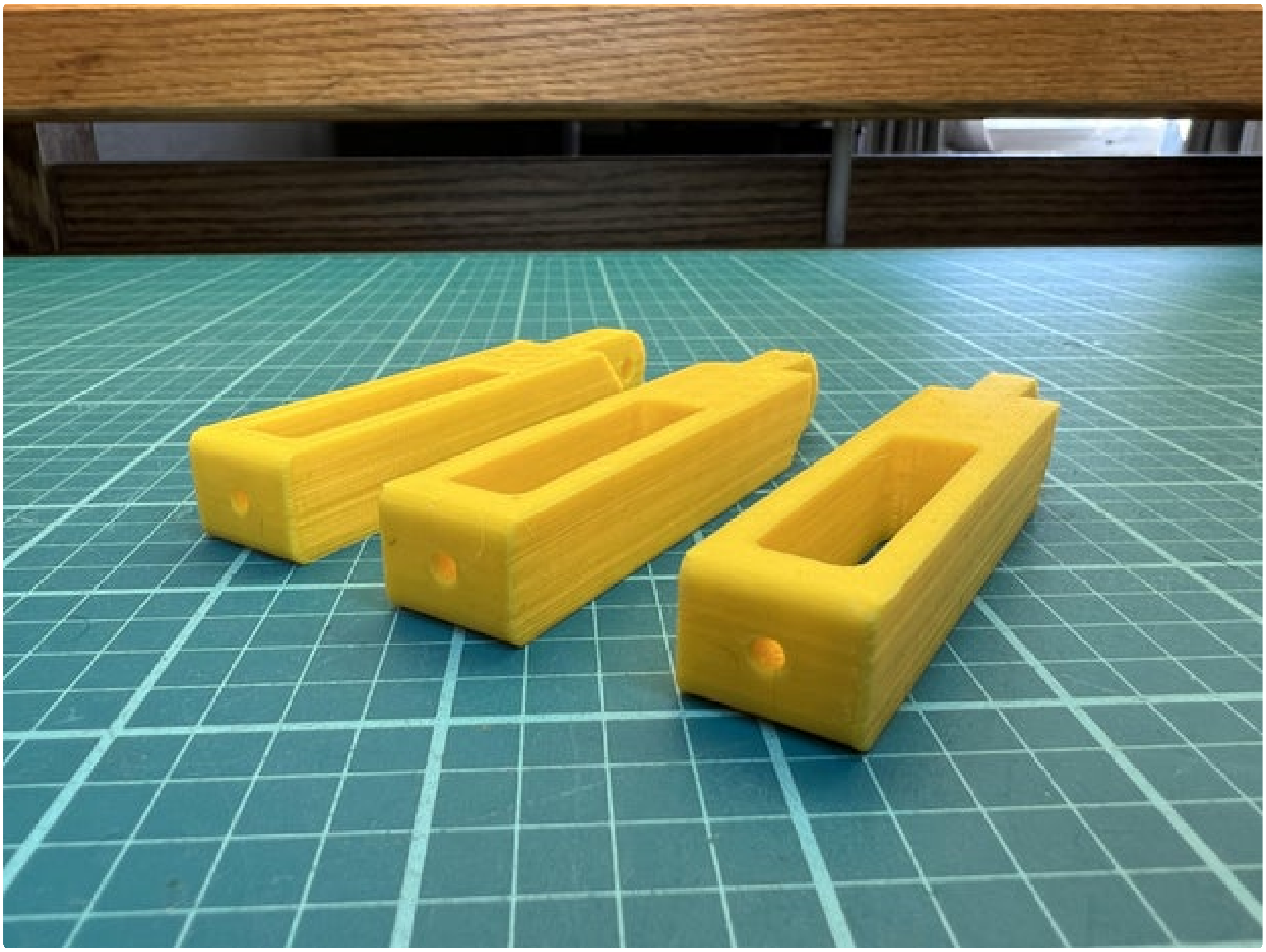
3D打印
3D printing

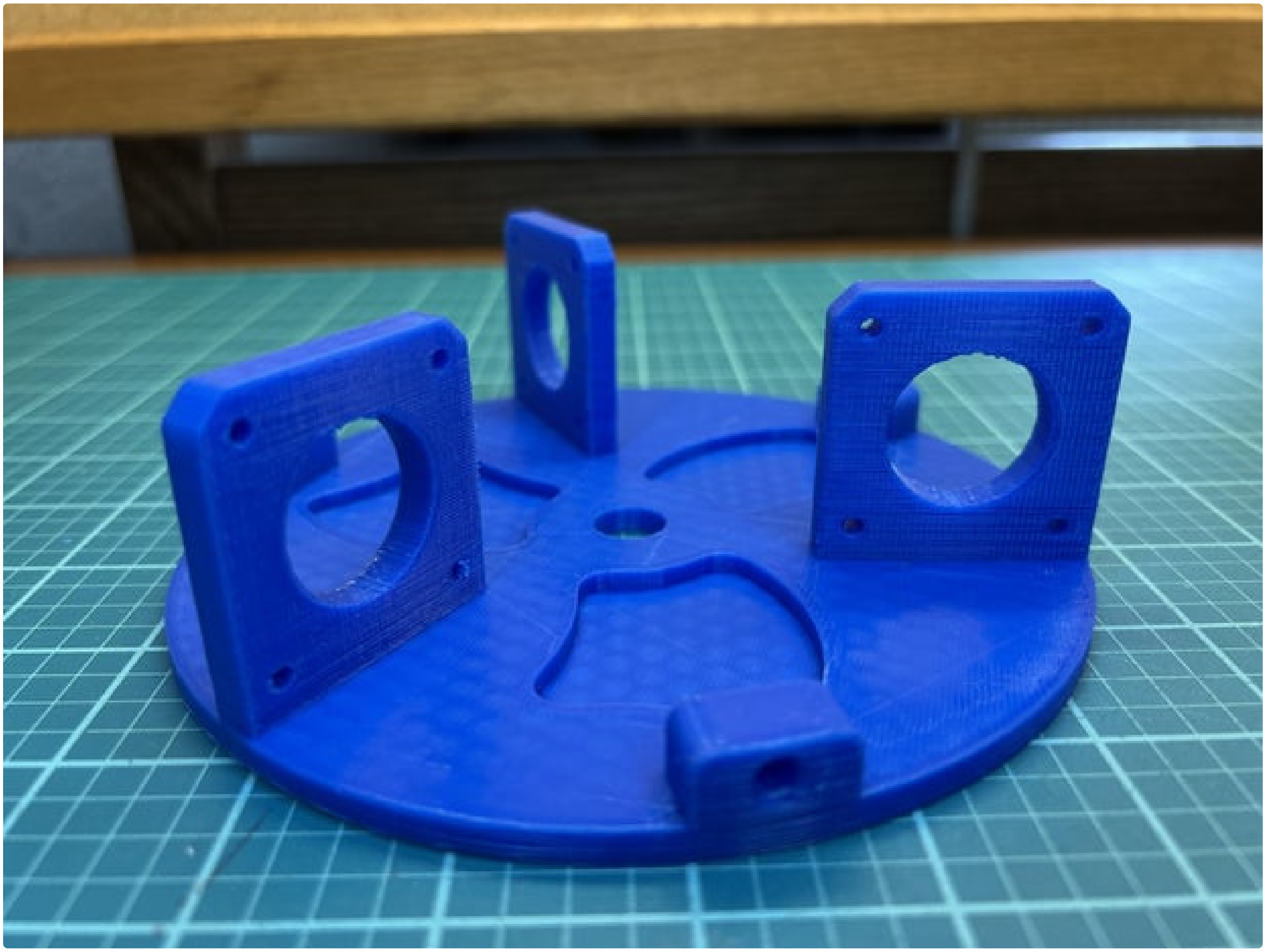


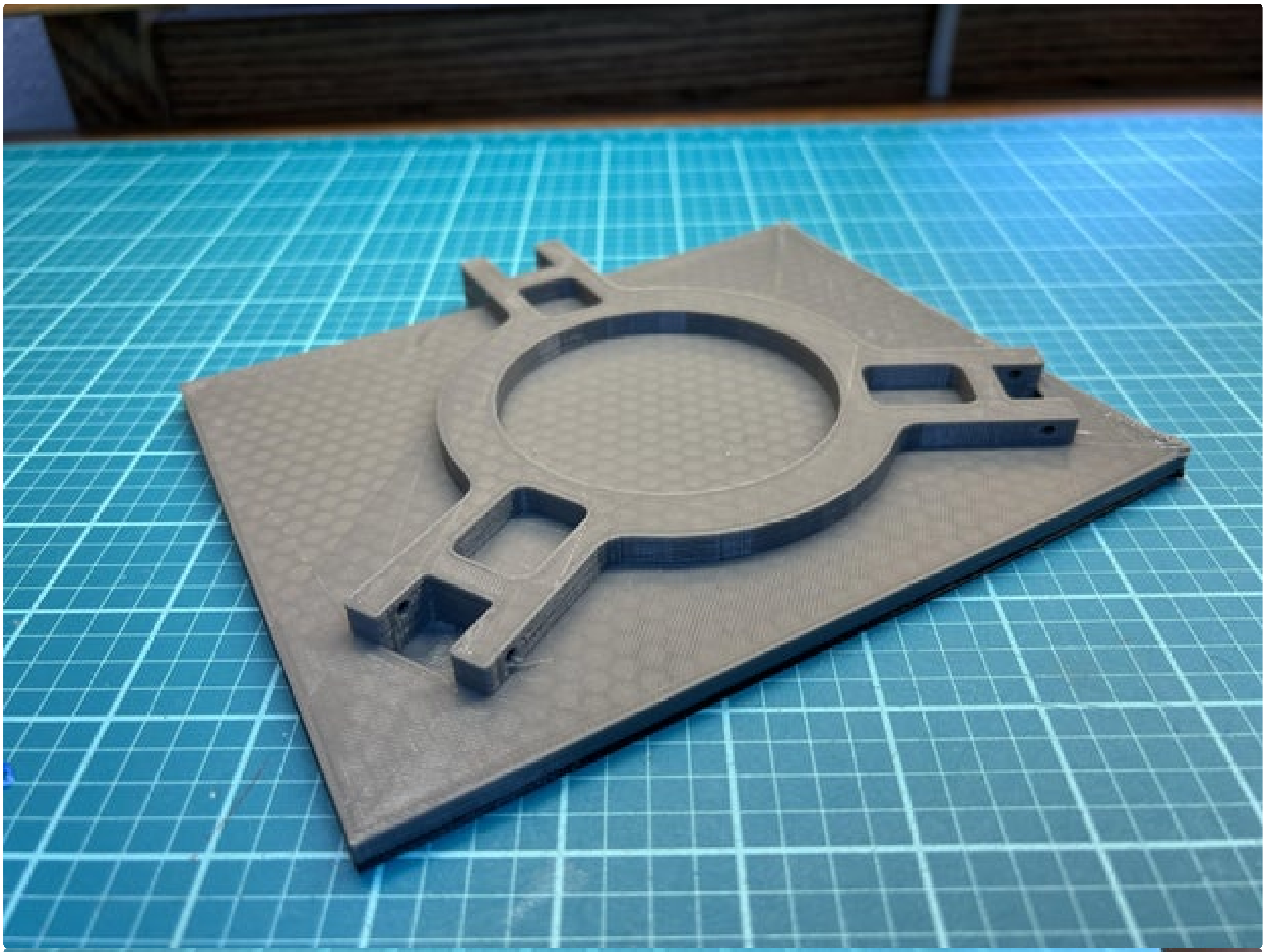
CREALITY
3D打印
3D printing

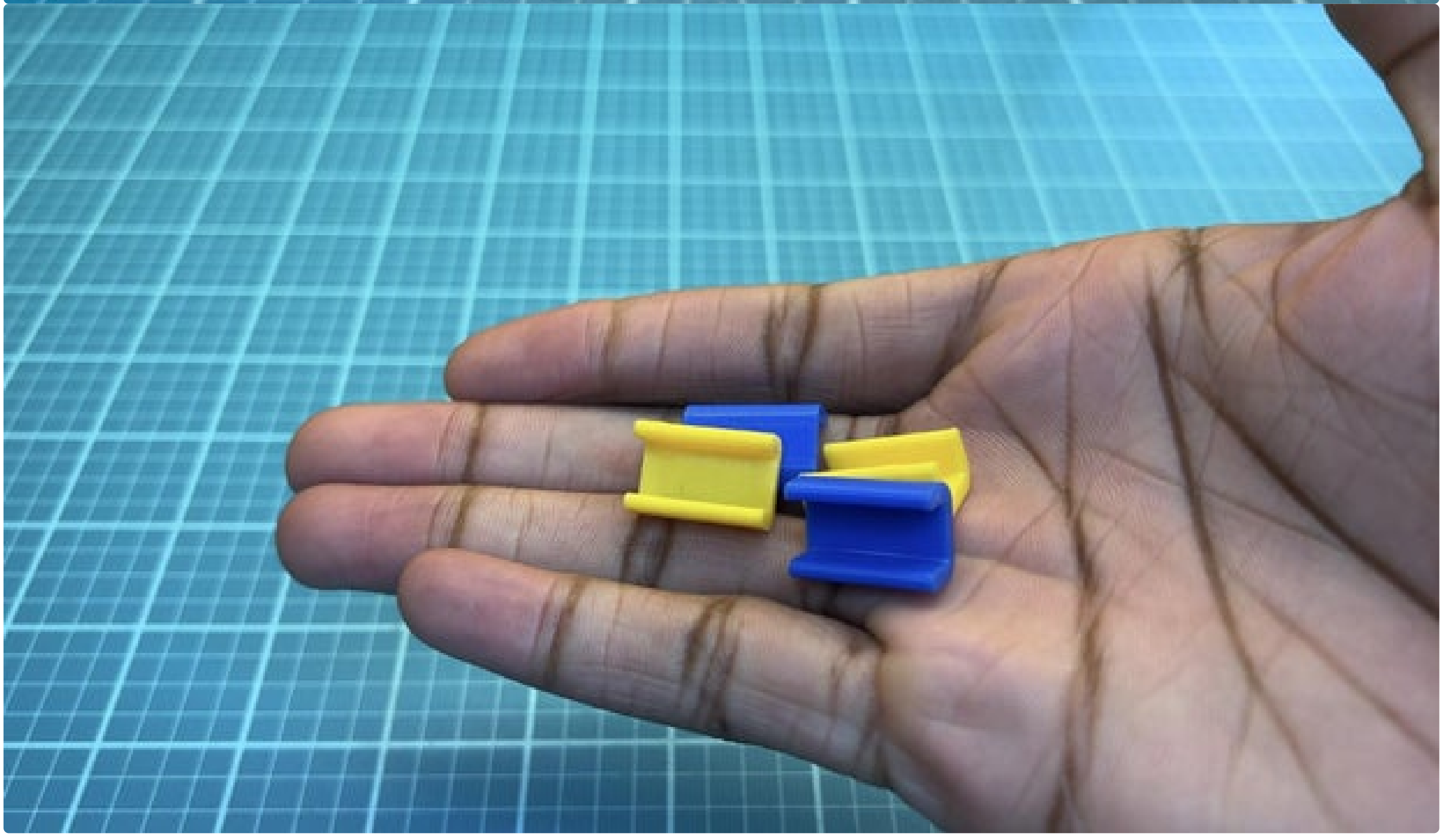

CREALITY
CREALITY











https://www.instructables.comhttps://a360.co/3V4uP7W

●●●

Download

https://www.instructables.com/FXX/HW29/LGJF3P02/FXXHW29LGJF3P02.stl

View in 3D Download

https://www.instructables.com/FJT/P1EJ/LGJF3P03/FJTP1EJLGJF3P03.stl

View in 3D Download

https://www.instructables.com/FPY/7IYH/LGJF3P04/FPY7IYHLGJF3P04.stl

View in 3D Download

https://www.instructables.com/FA9/SZUC/LGJF3P05/FA9SZUCLGJF3P05.stl

View in 3D Download

https://www.instructables.com/F3Q/SOPI/LGJF3P06/F3QSOPILGJF3P06.stl

View in 3D Download

https://www.instructables.com/FLM/CQTF/LGJF3P07/FLMCQTFLGJF3P07.stl

View in 3D Download

https://www.instructables.com/FBA/LTL4/LGJF3P08/FBALTL4LGJF3P08.stl

Download

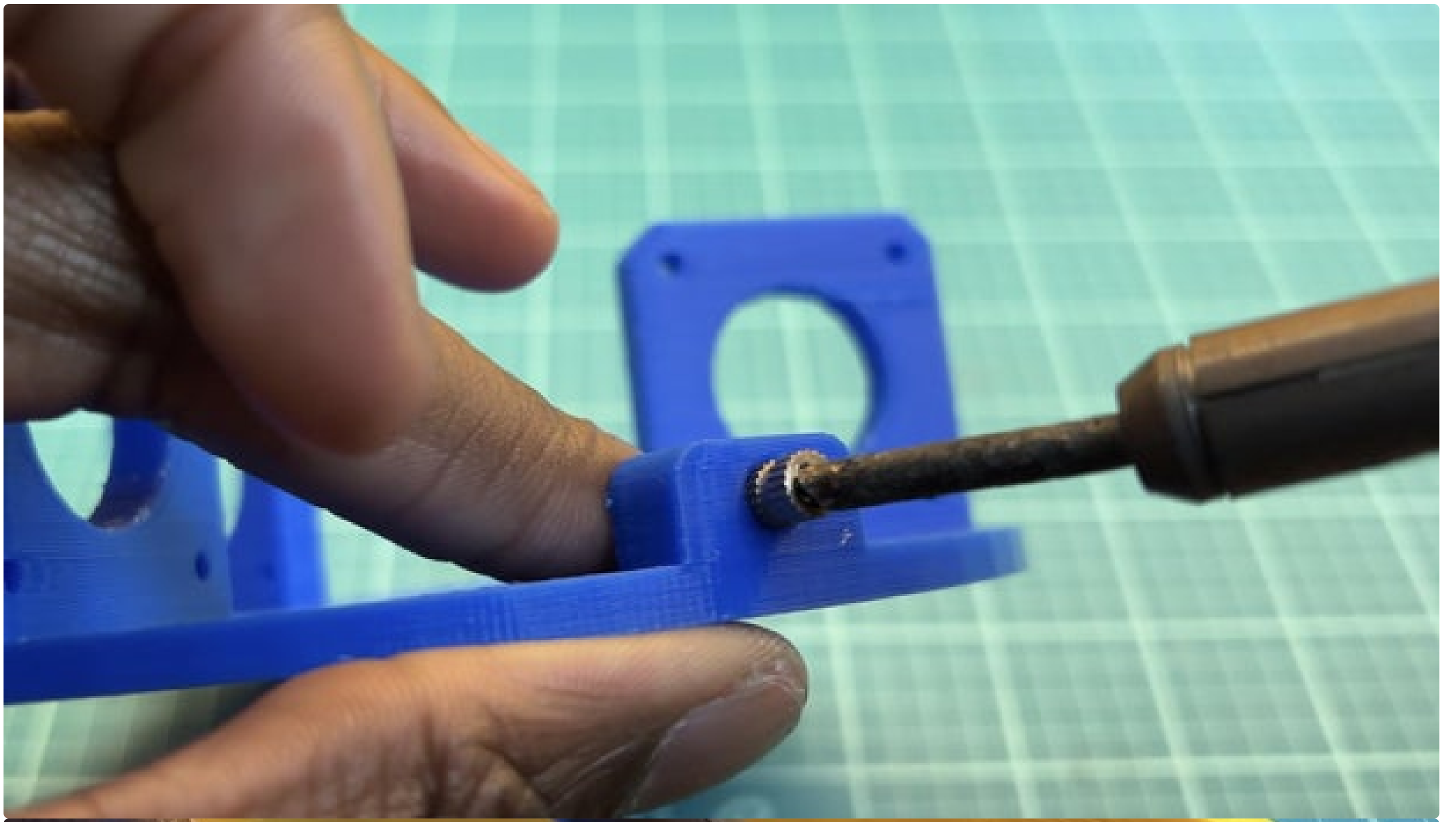
https://www.instructables.com/FS6/HFEL/LGJF3P09/FS6HFELLGJF3P09.stl

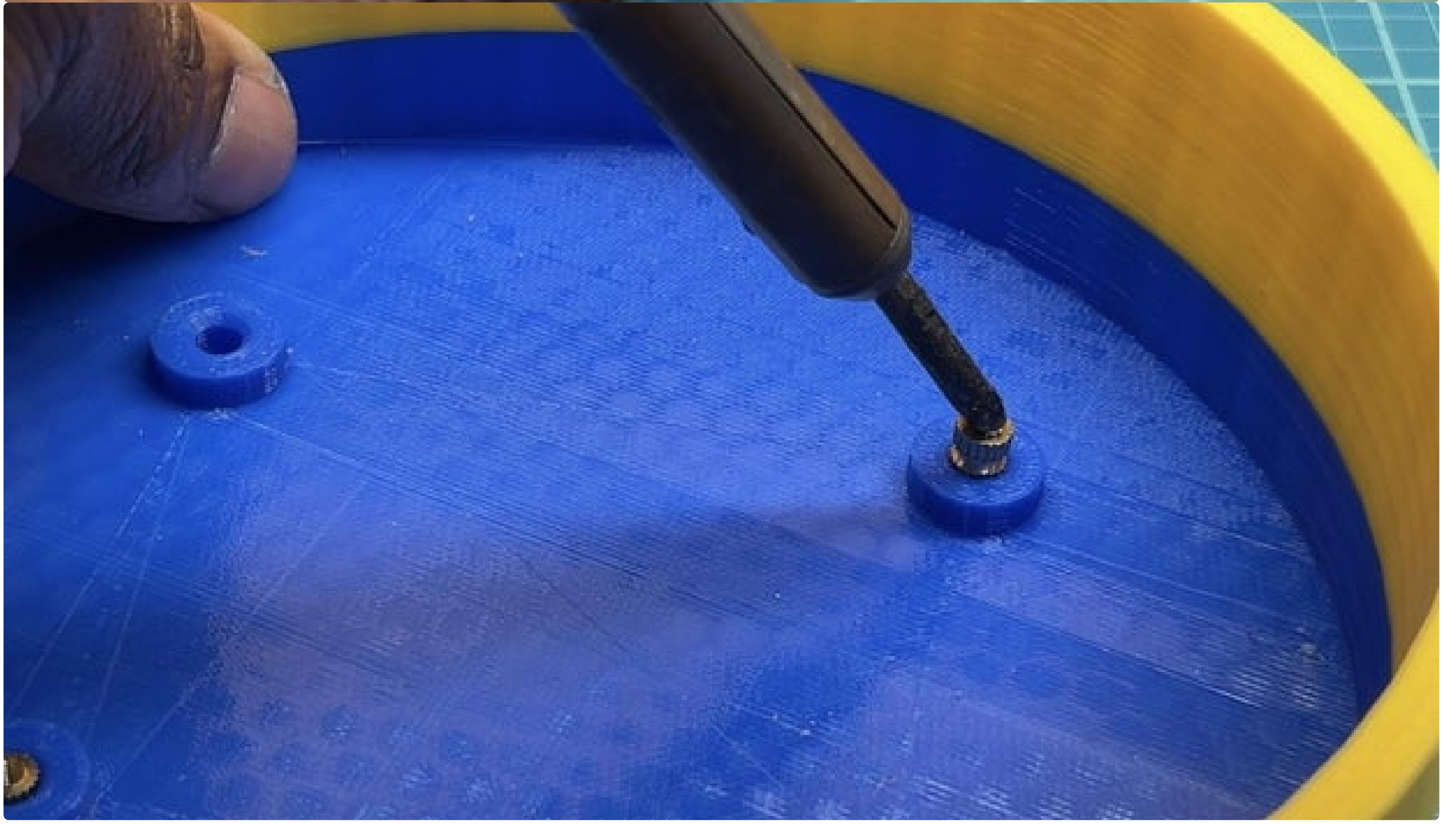
View in 3D Download

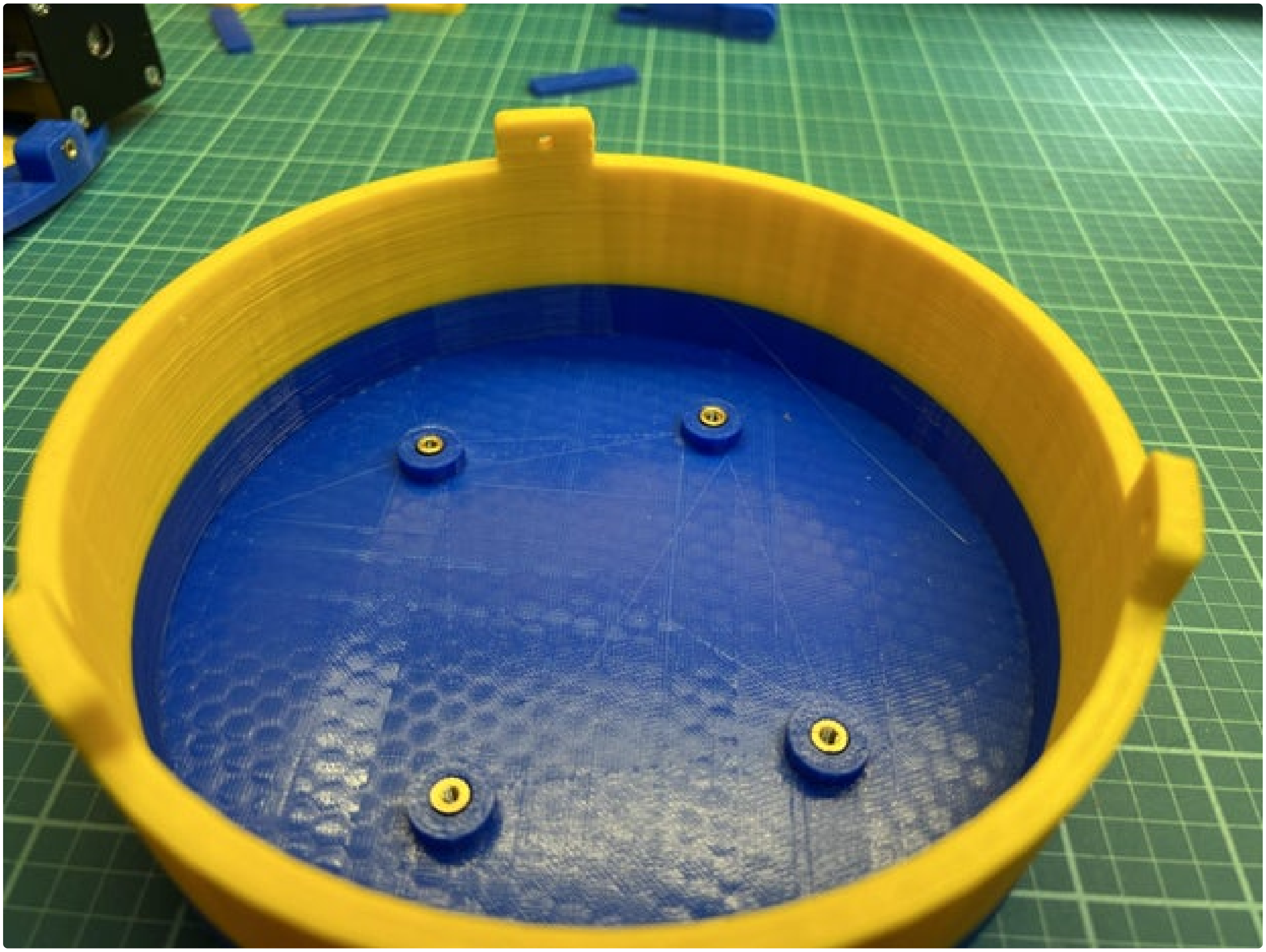
https://www.instructables.com/F83/934S/LGJF3P0A/F83934SLGJF3P0A.stl

Step 2: Add the Heated Inserts

Use a hot soldering iron to press four heated inserts into the base stand and three heated inserts into the base plate.







Step 3: Create the Circuit

Follow the schematic to build the circuit. I designated one protoboard for the teensy microcontroller and the other protoboard for the 3 stepper motor drivers. I'd recommend using header pins to allow the teensy and motor drivers to be removable.

Power

- Set your bench power supply to 24V or use an alternative 24V power supply.

5V Regulator

- The teensy 4.1 is powered by 5V. The regulator steps down the 24V from the power supply to 5V.
- Connect the positive and negative terminals of your power supply to the IN+ and GND pins respectively. The VO+ pin outputs 5V once you adjust the voltage on the regulator. Adjust this voltage by turning the potentiometer and measuring the voltage with a voltmeter.

Teensy 4.1 Microcontroller

- The drivers and the resistive touchpad connect to the teensy 4.1 microcontroller.

- You must cut the 5V trace on the teensy. This trace connects the USB 5V to the teensy 5V. We will power the teensy with the bench power supply using the 5V regulator. Refer to the images to see the location of this trace.

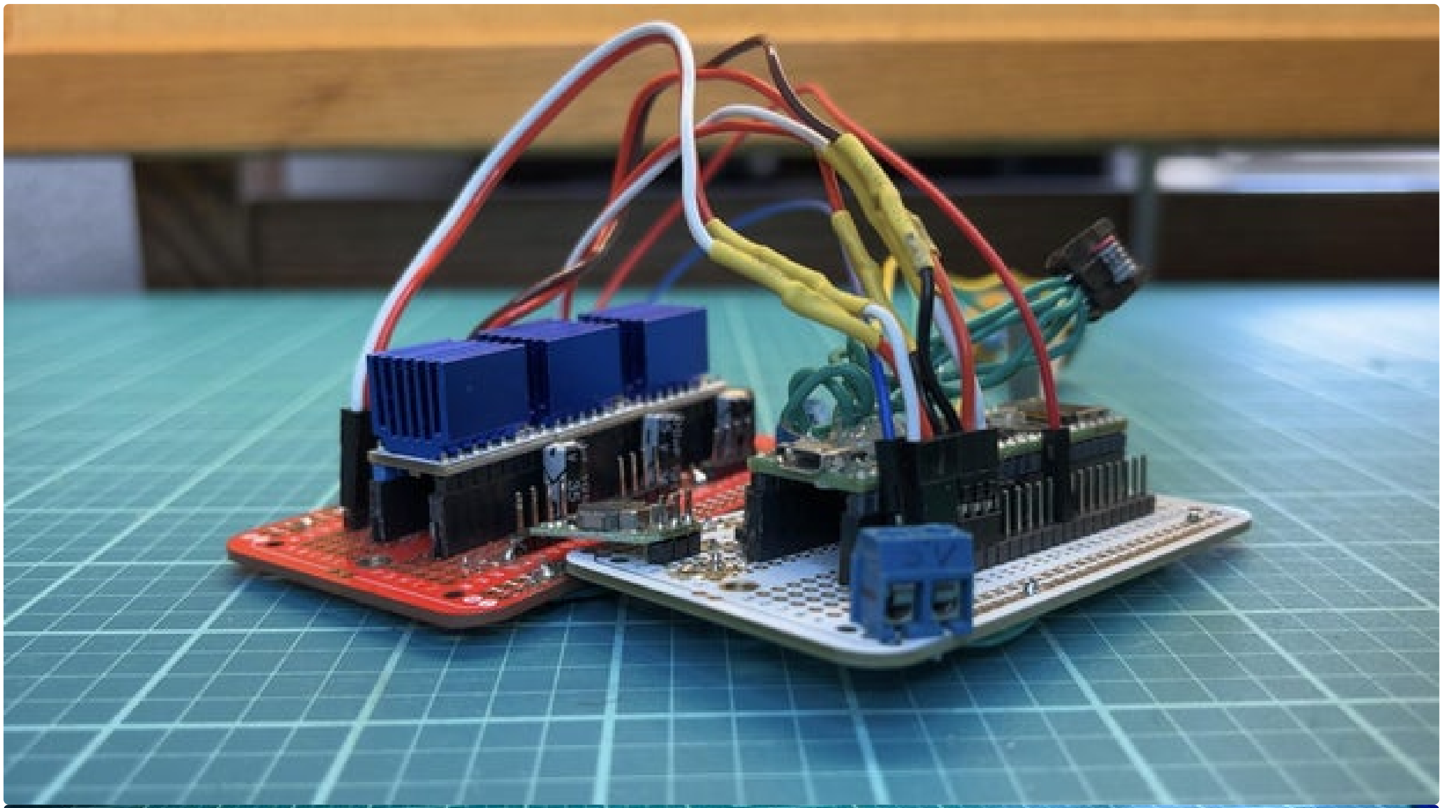
TMC2208 Stepper Motor Drivers

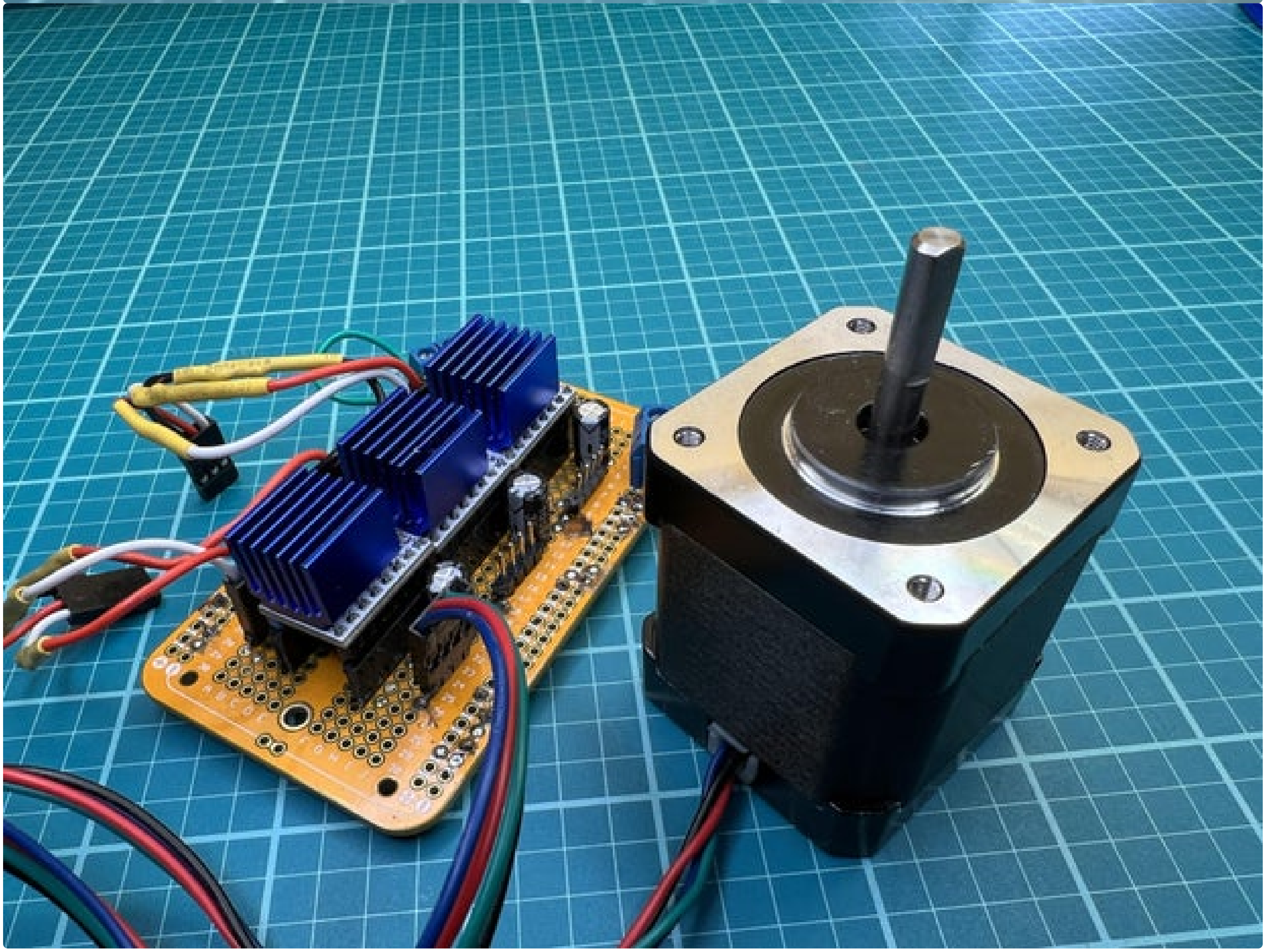
- The resistive touchpad has 4 pins that connect to the teensy analog pins A0, A1, A2, and A3. Refer to the images to see which touchpad pin connects to which analog pin.

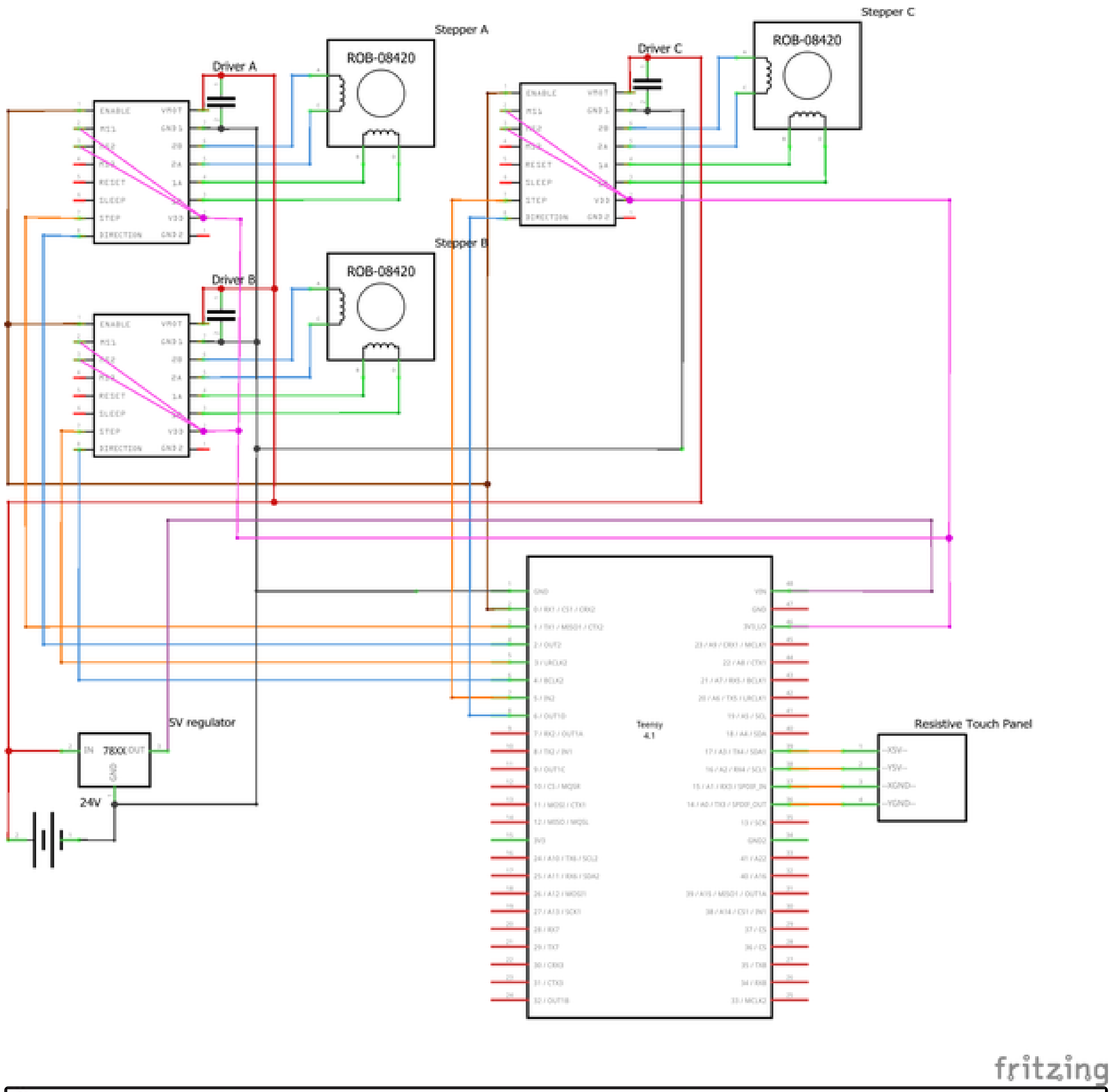
TMC2208 Stepper Motor Drivers

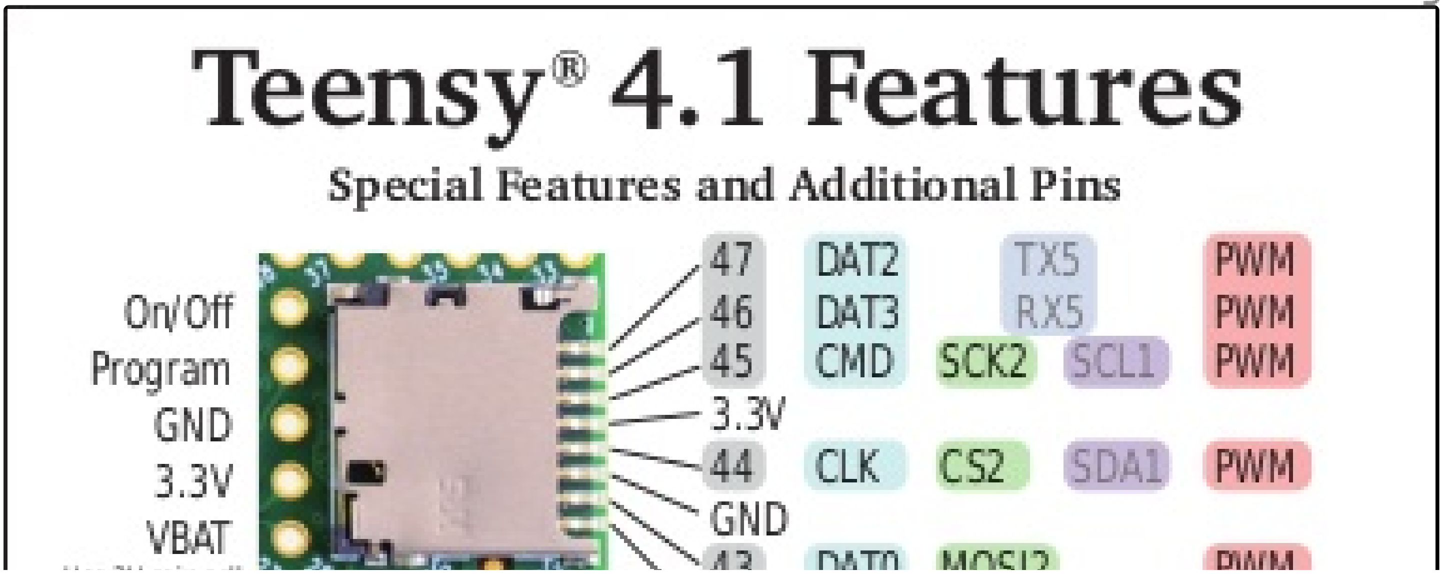
- Each stepper motor connects to a driver and each driver connects to the teensy.
- The VIO pin on each driver connects to the positive terminal of the power supply.
- The GND pin on each driver connects to the negative terminal of the power supply.
- You will need to add a 100 microfarad capacitor in parallel to the VIO and GND pins on each driver. This is done to prevent voltage spikes from damaging the drivers.
- This project uses 16 microstepping meaning that the steppers have 3200 steps/rev or a step angle of 0.1125°. In order to set the steppers to 16 microstepping, connect the MS1 and MS2 pins on all drivers to the 3.3V pin on the teensy. This is the highest microstepping setting on the driver. Contrasting this with the regular 200 steps/rev or a step angle of 1.8° you'll see that the stepper motors turn very precisely with 16 microstepping.
- The VM pin on each driver connects to the 3.3V pin on the teensy in order to power each driver's logic board.
- Teensy Pin Assignments:
- pin 0 connects to the ENA pin on each driver (we want to be able to enable and disable all the stepper motors at the same time)
- pins 1 and 2 connect to the STEP and DIR pins on Driver A
- pins 3 and 4 connect to the STEP and DIR pins on Driver B
- pins 5 and 6 connect to the STEP and DIR pins on Driver C
- pins A0, A1, A2, and A3 connect to the touchpad

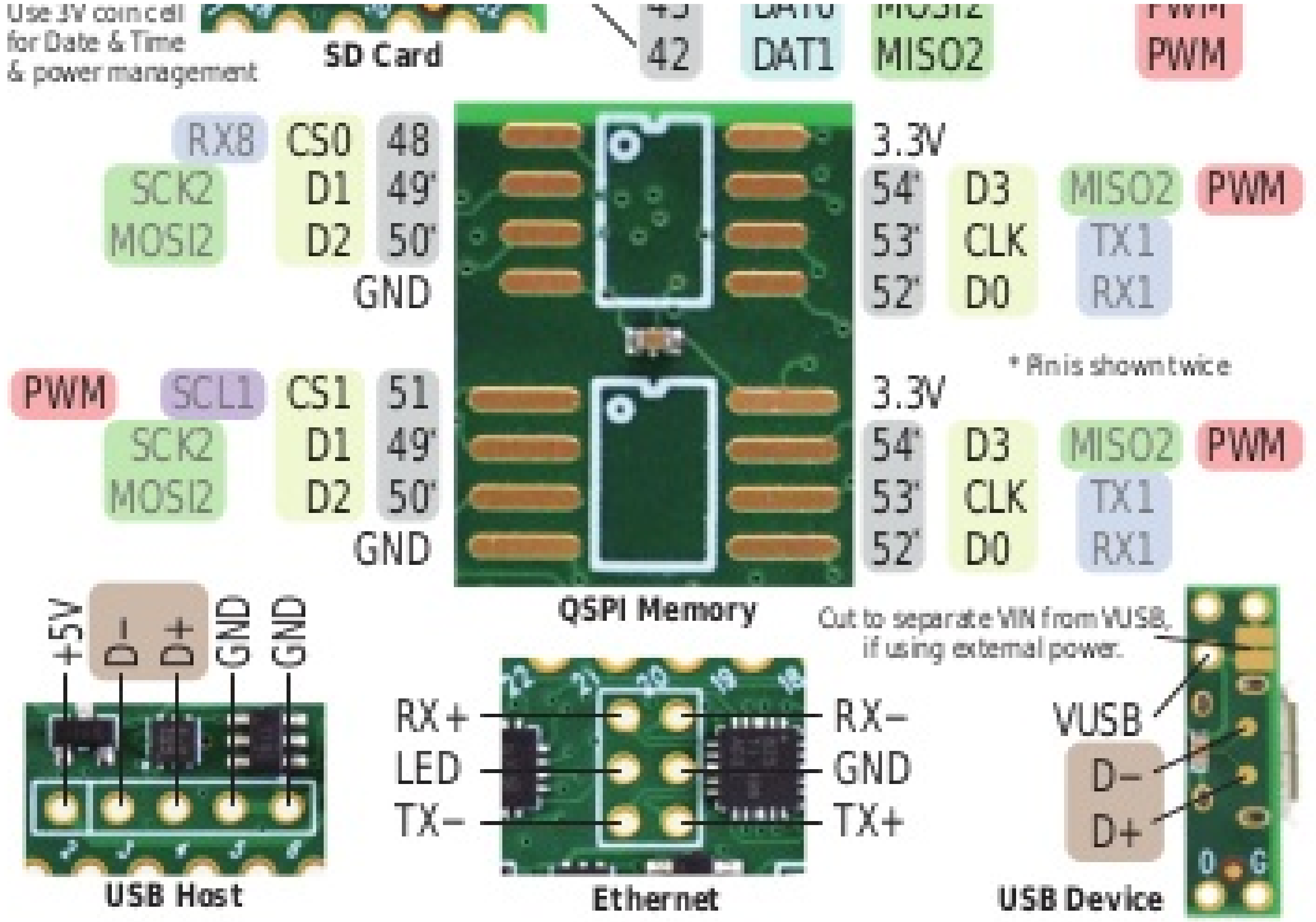
****Make sure that you have a common ground i.e. all ground pins on all components should be connected****






Stepper A
ROB-08420
Driver A
Stepper B
ROB-08420
Driver B
Stepper C
ROB-08420
Driver C
ENABLE
MS1
MS2
MS3
RESET
SLEEP
STEP
DIRECTION
VMOT
GND1
2B
2A
1A
1B
VDD
GND2
Teensy 4.1
GND
VIN
3V3
5V regulator
IN
78XX
OUT
GND
24V
Resistive Touch Panel
XSV
YSV
XGND
YGND
fritzing


Teensy® 4.1 Features
Special Features and Additional Pins
On/Off
Program
GND
3.3V
VBAT
47
46
45
3.3V
44
GND
43
DAT2
DAT3
CMD
CLK
DAT0
TX5
RX5
SCK2
CS2
MOSI2
SCL1
SDA1
PWM
PWM
PWM
PWM
PWM



For solutions to the most common issues and technical support, please visit:

www.pjrc.com/help

Teensy 4.1 pins are **not 5 volt tolerant**. Do not apply more than 3.3V to any pin, except VIN & VUSB.

Teensy 4.1 System Requirements:
PC computer with Windows 7, 8, 10, 11 or later
or Ubuntu Linux 14.04 or later
or Macintosh MacOS 10.10 or later
USB Micro-B Cable

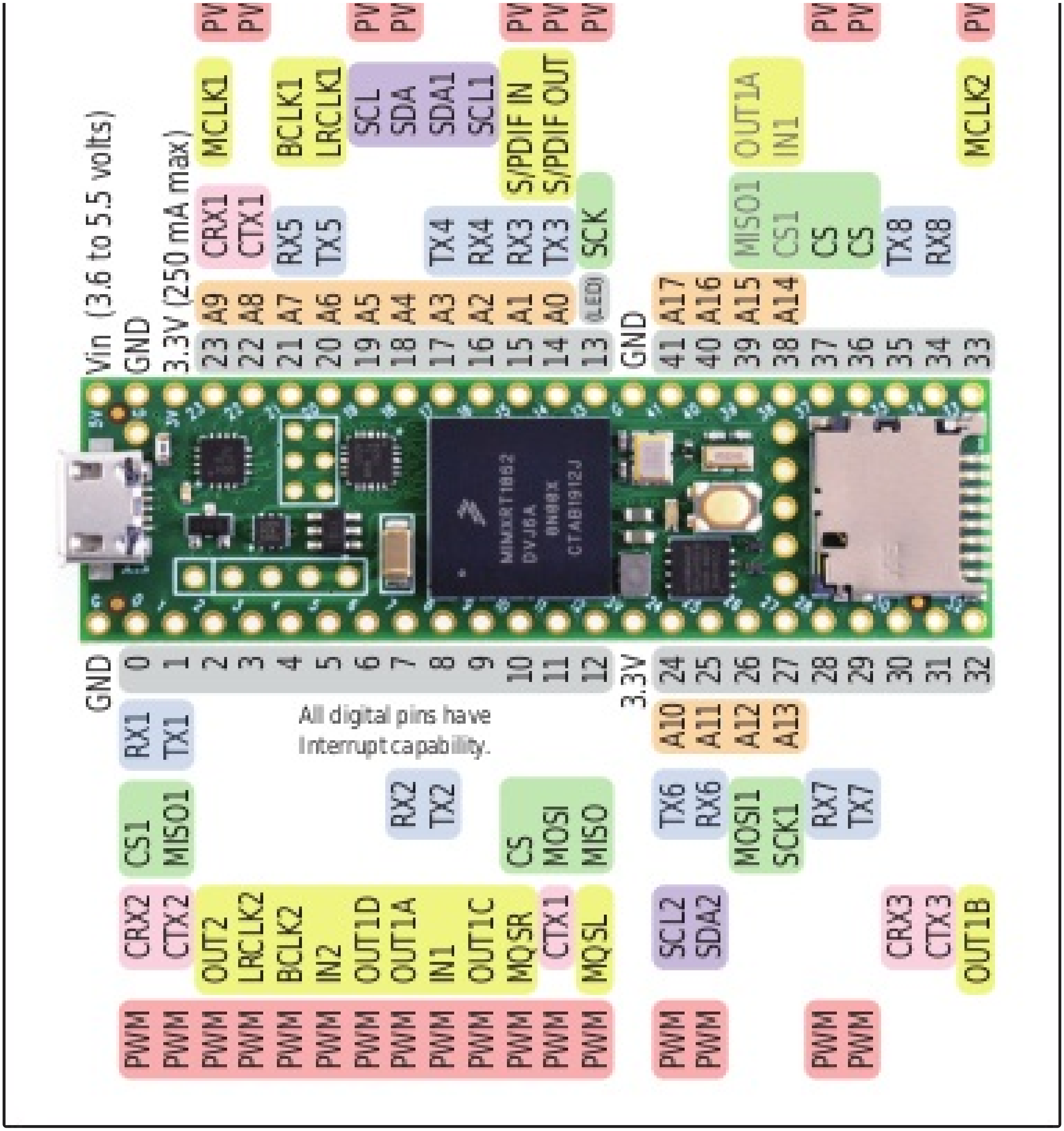
7 14833 87947 3

Welcome to Teensy® 4.1

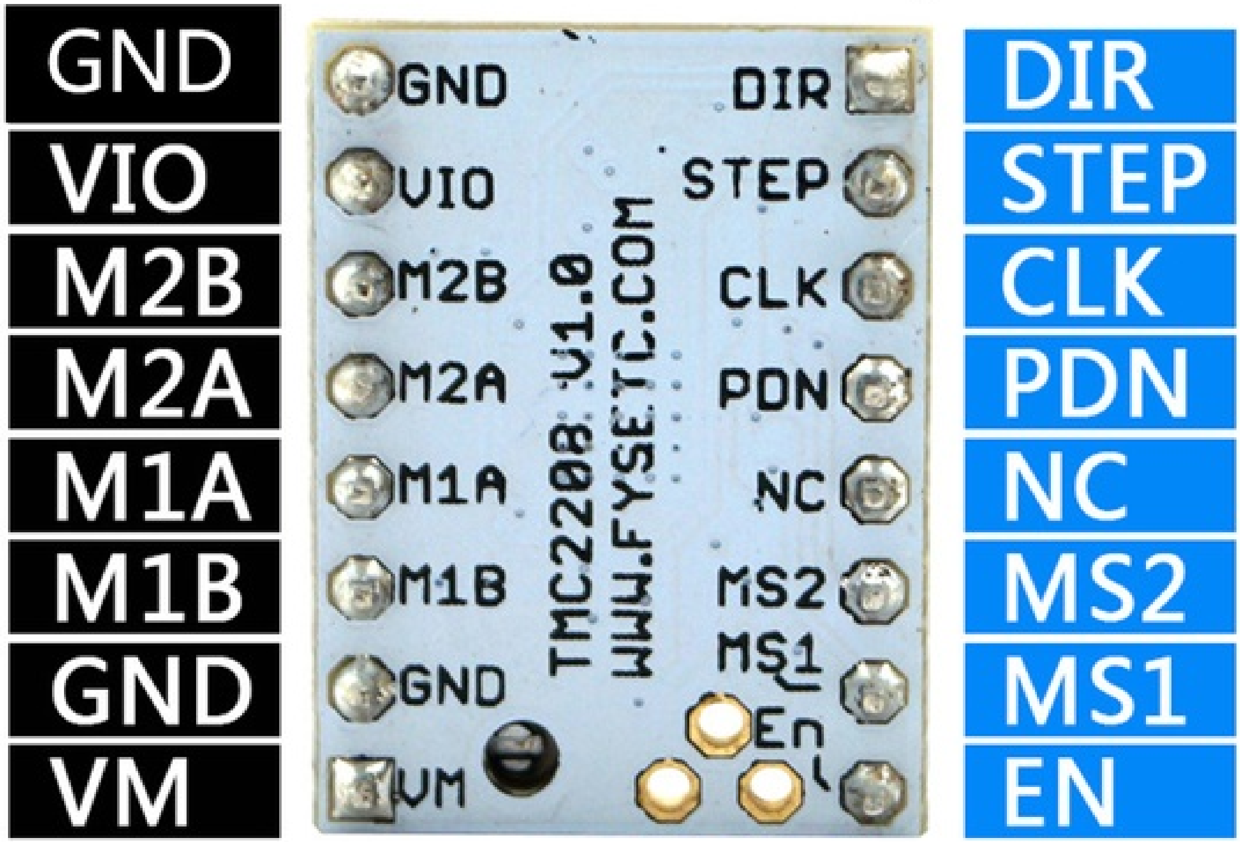
32 Bit Arduino-Compatible Microcontroller

To begin using Teensy, please visit the website & click Getting Started.

www.pjrc.com/teensy


Vin (3.6 to 5.5 volts)
GND
3.3V (250 mA max)
23 A9 MCLK1 PV
22 A8 CRX1 CTX1 PV
21 A7 RX5 BCLK1
20 A6 TX5 LRCLK1
19 A5 SCL PV
18 A4 SDA PV
17 A3 TX4 SDA1
16 A2 RX4 SCL1
15 A1 RX3 S/PDIF IN PV
14 A0 TX3 S/PDIF OUT PV
13 (LED) SCK PV
GND
41 A17
40 A16
39 A15 MISO1 OUT1A
38 A14 CS1 IN1
37 CS PV
36 CS PV
35 TX8
34 RX8
33 MCLK2 PV
GND
0 RX1 CS1 CRX2 PWM
1 TX1 MISO1 CTX2 PWM
2 OUT2 PWM
3 LRCLK2 PWM
4 BCLK2 PWM
5 IN2 PWM
6 OUT1D PWM
7 RX2 OUT1A PWM
8 TX2 IN1 PWM
9 OUT1C PWM
10 CS MQSR PWM
11 MOSI CTX1 PWM
12 MISO MQSL PWM
3.3V
24 A10 TX6 SCL2 PWM
25 A11 RX6 SDA2 PWM
26 A12 MOSI1
27 A13 SCK1
28 RX7 PWM
29 TX7 PWM
30 CRX3
31 CTX3
32 OUT1B
All digital pins have Interrupt capability.

TMC2208
Motor Drive Pin Descriptions

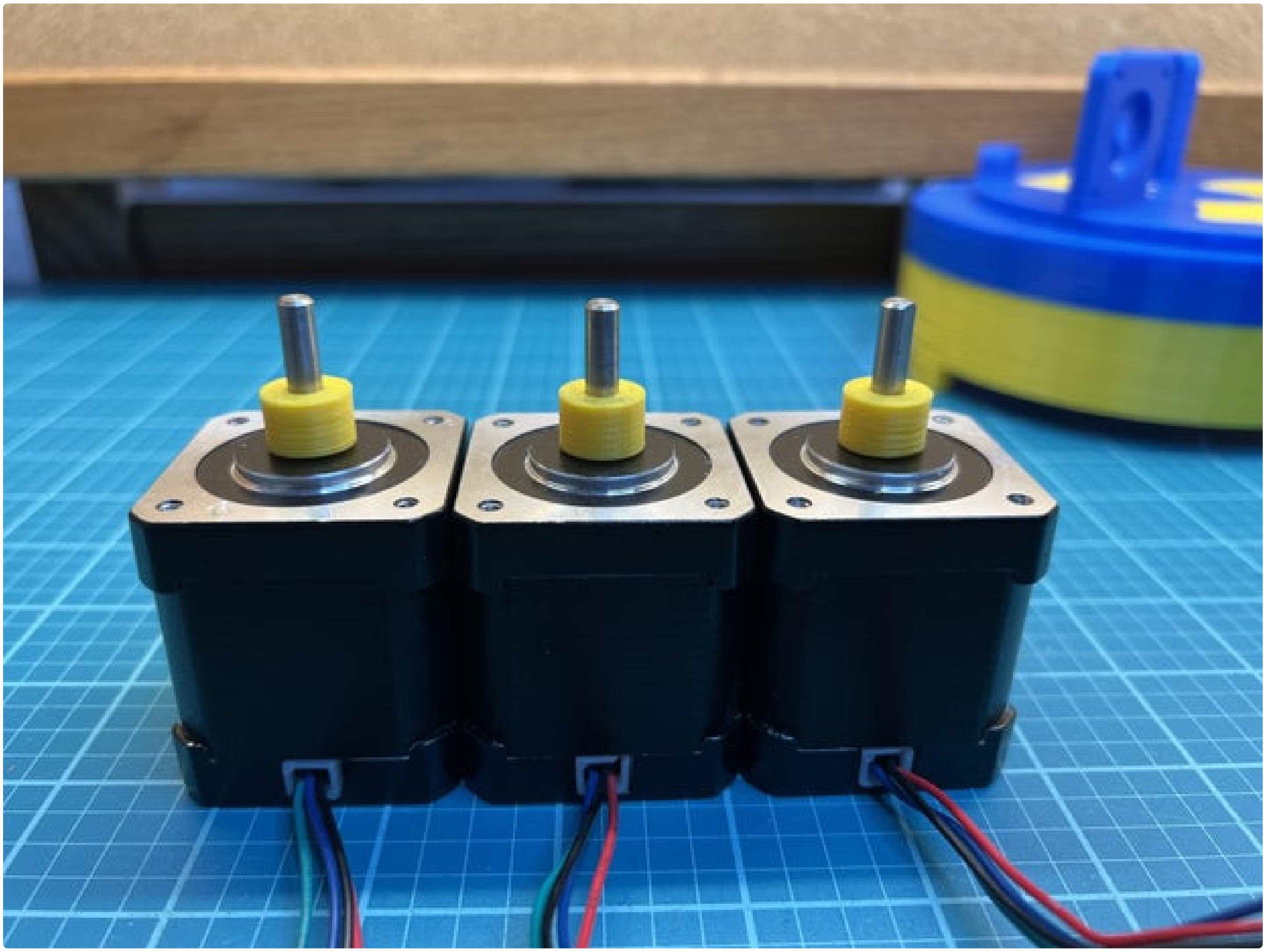


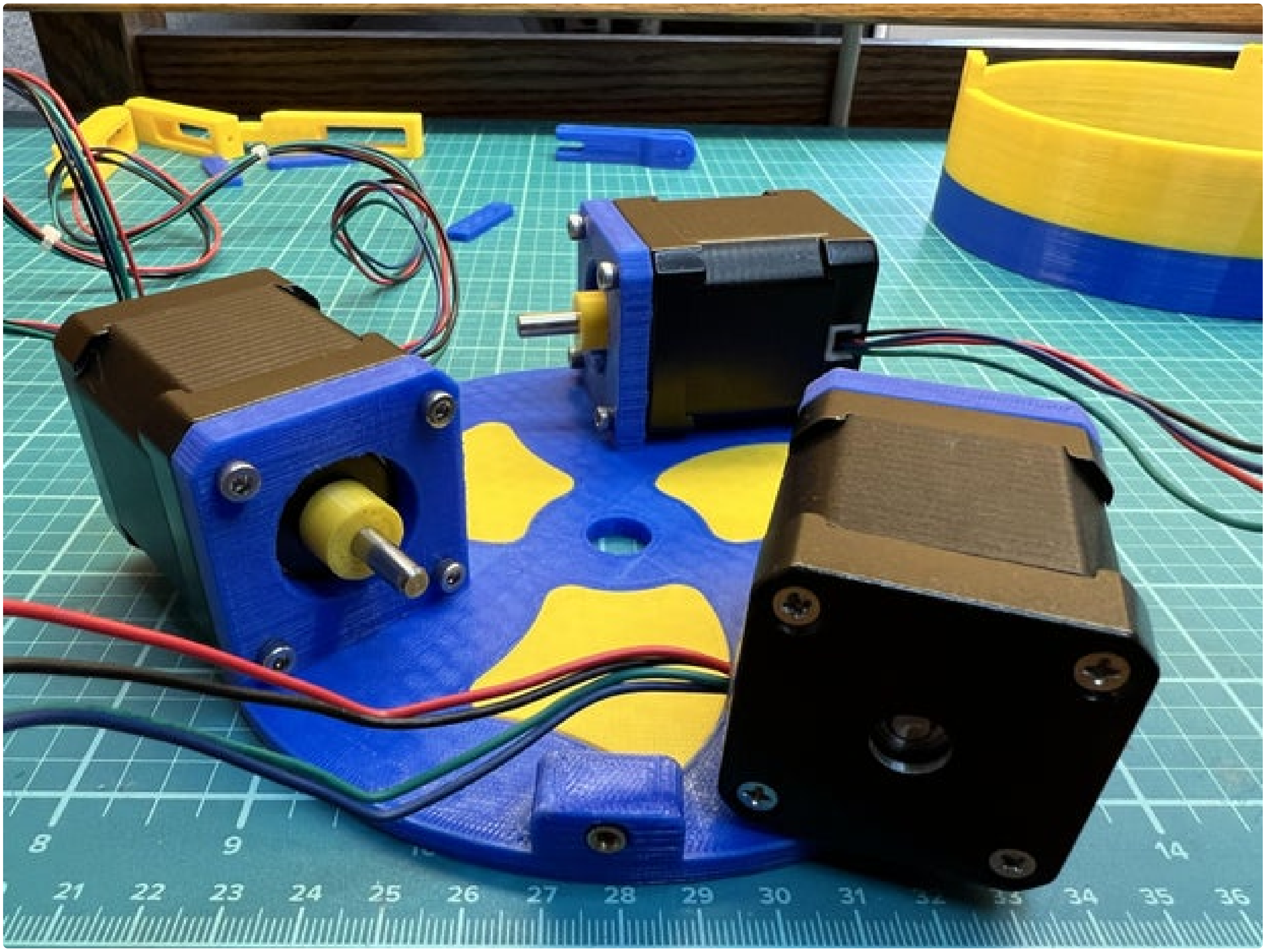
Download

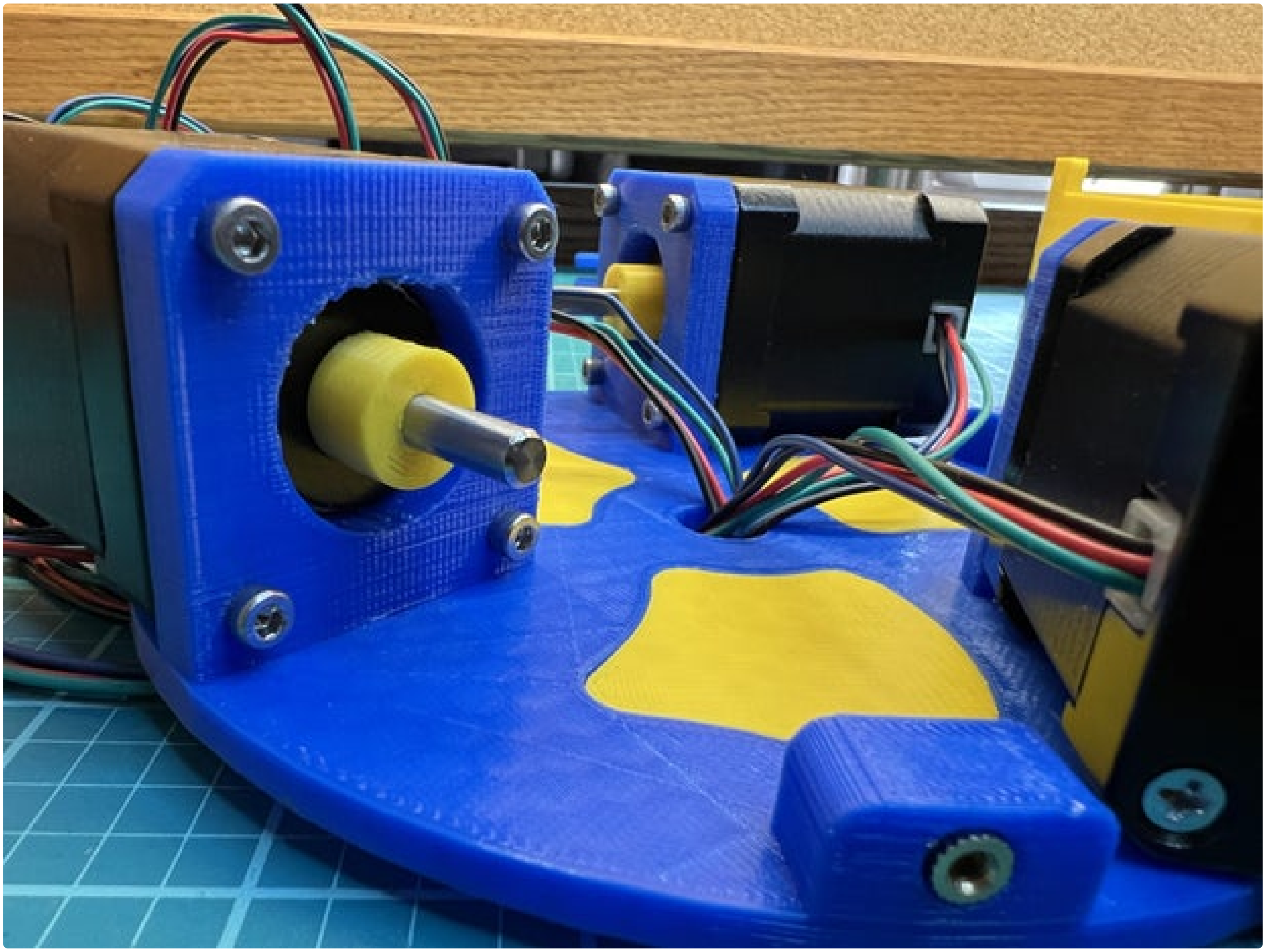
https://www.instructables.com/FYH/V0OT/LGKUKCRE/FYHV0OTLGKUKCRE.pdf

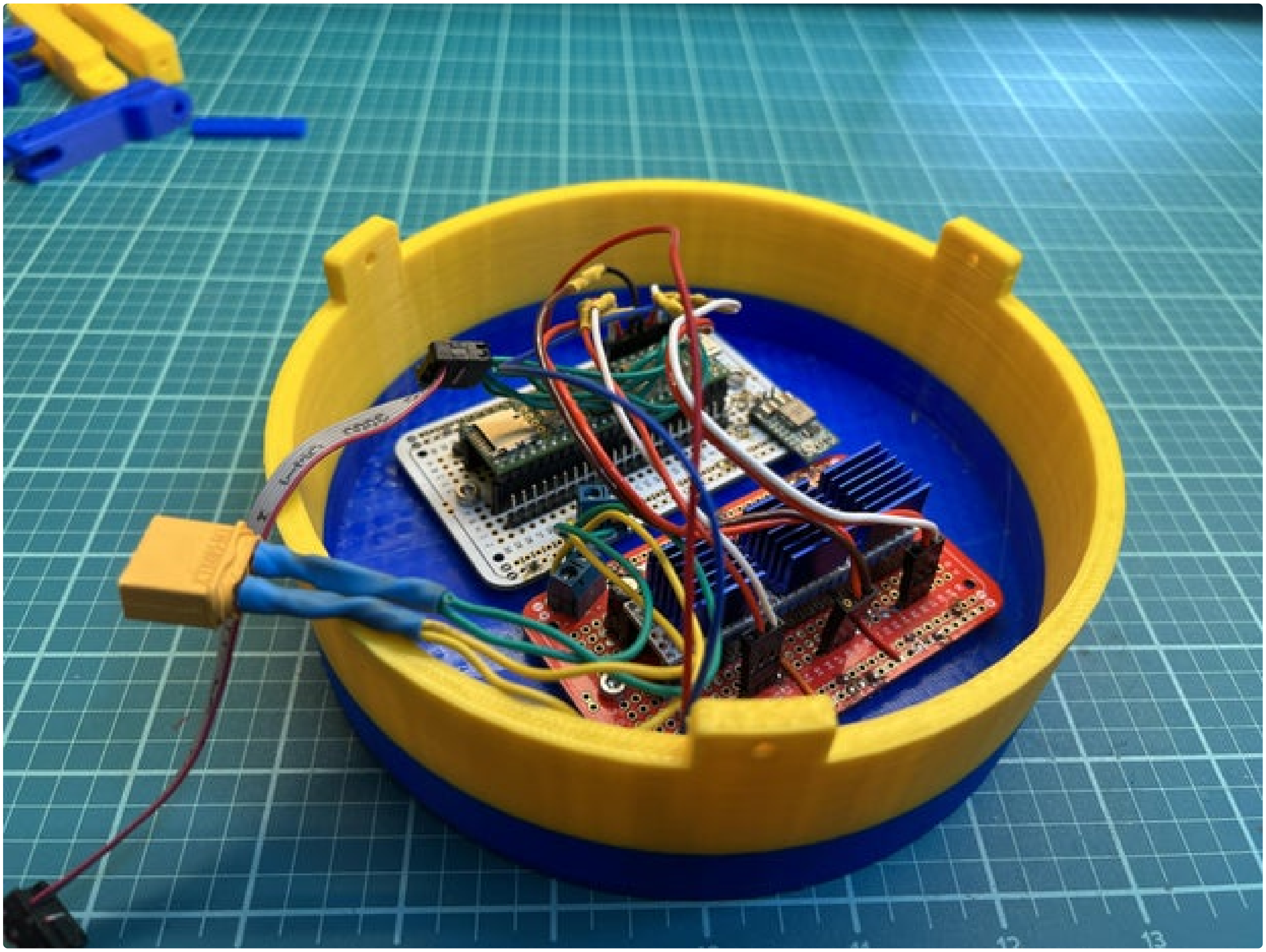
Step 4: Assemble the Base

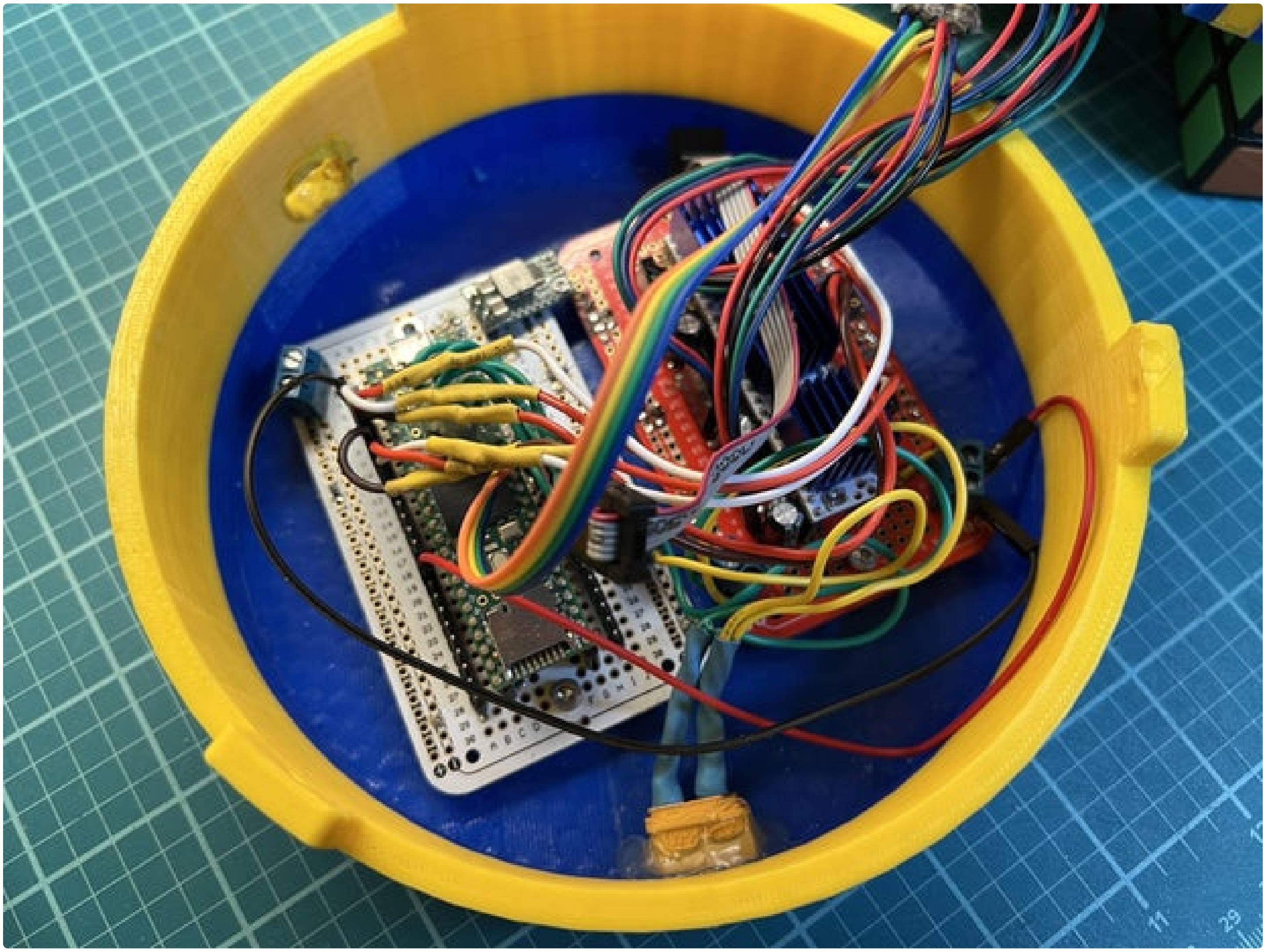
- Start by adding a spacer on each stepper motor.
- Screw each stepper motor onto the base plate using **M3 x 10mm screws (x12)**
- Mount the protoboards with the electronics onto the base stand using **M3 x 5mm screws (x4)**
- Connect stepper motors A, B, and C to drivers A, B, and C by connecting each stepper motor's pins to pins M1B, M1A, M2A, and M2B on each driver. Remember which stepper motors were assigned A, B, and C. This is important for step #6.
- Screw each link_ #1 onto each stepper motor using an **M4 x 20mm screw** and an **M4 locknut**.

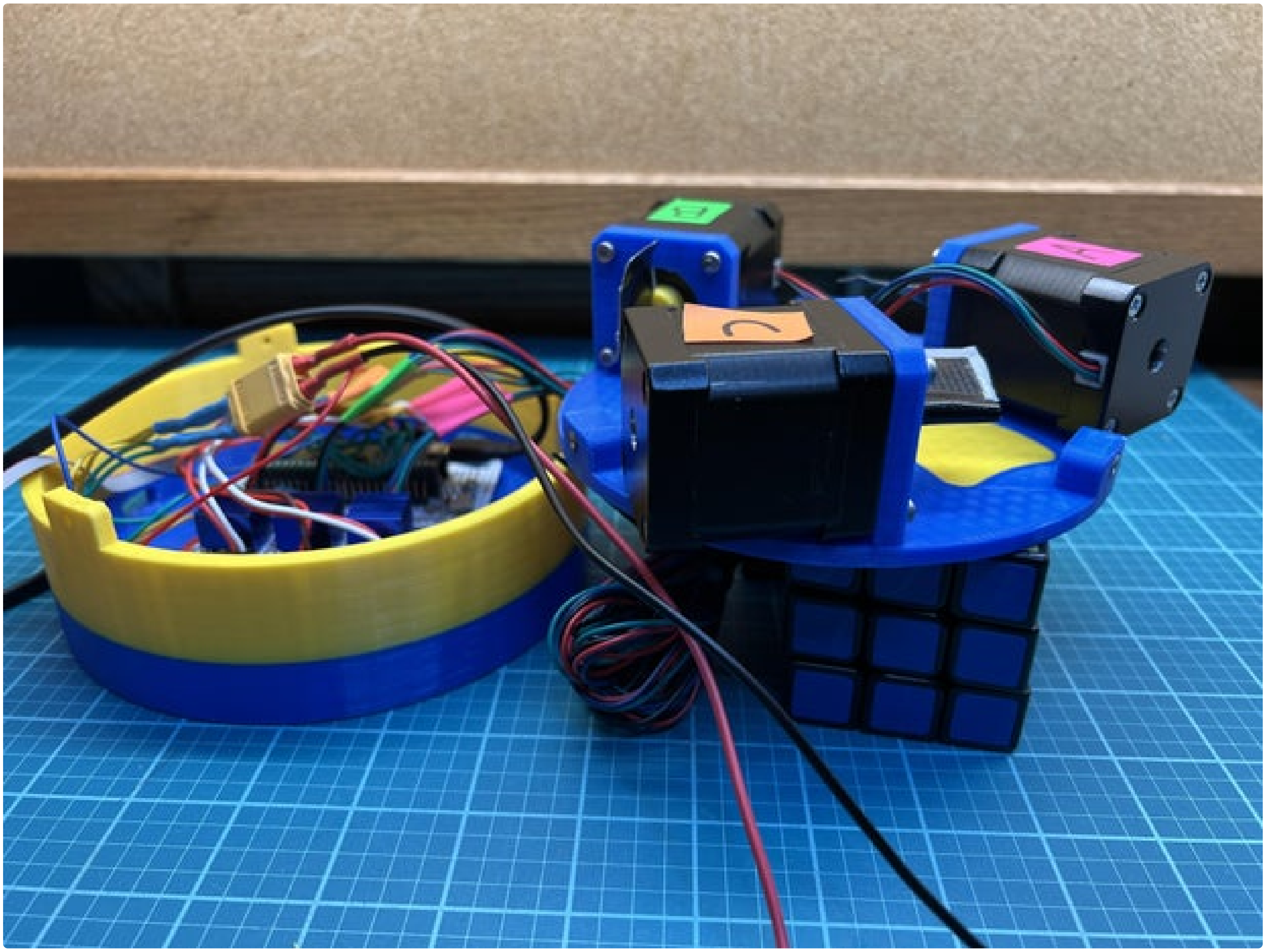


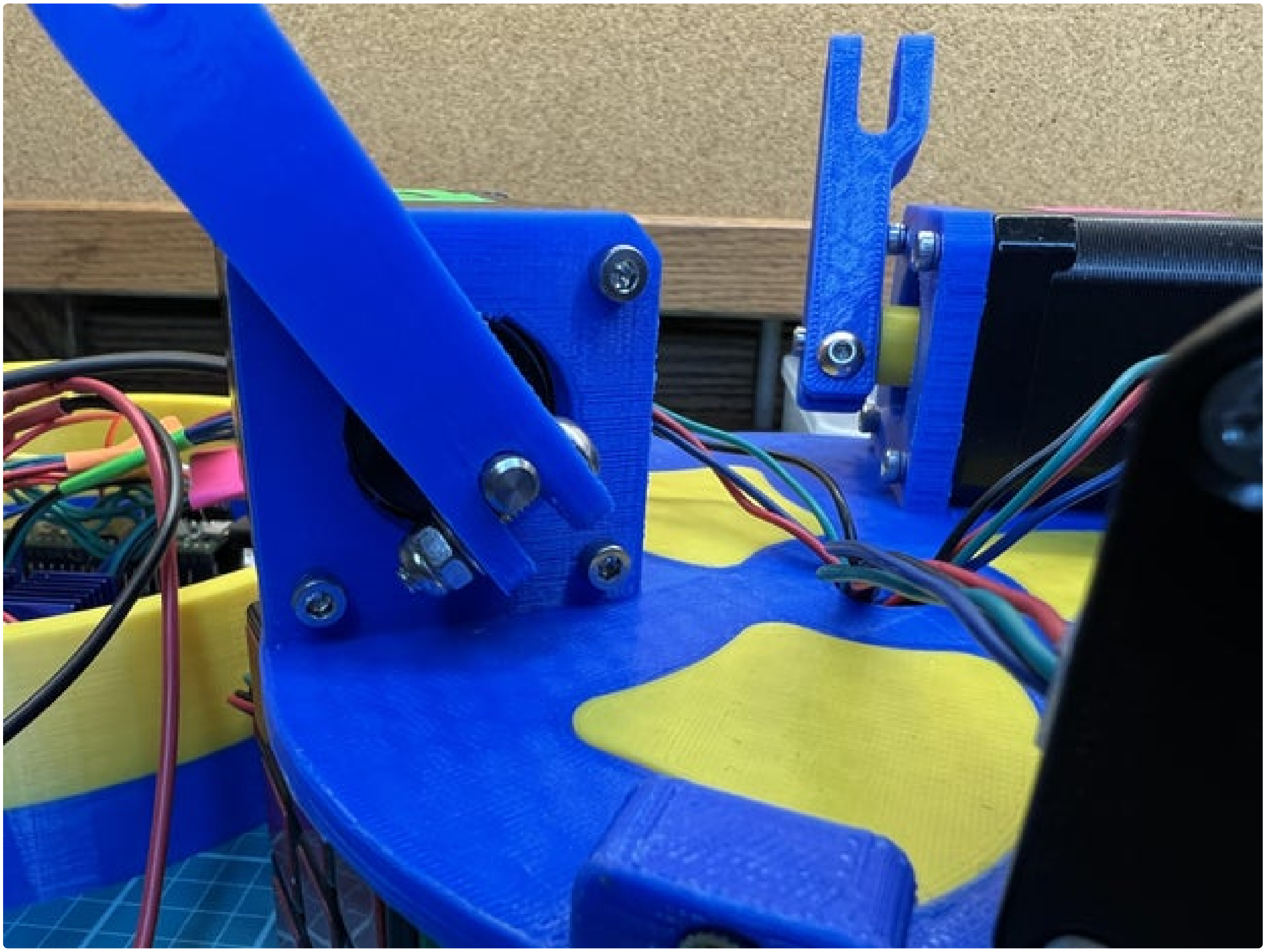


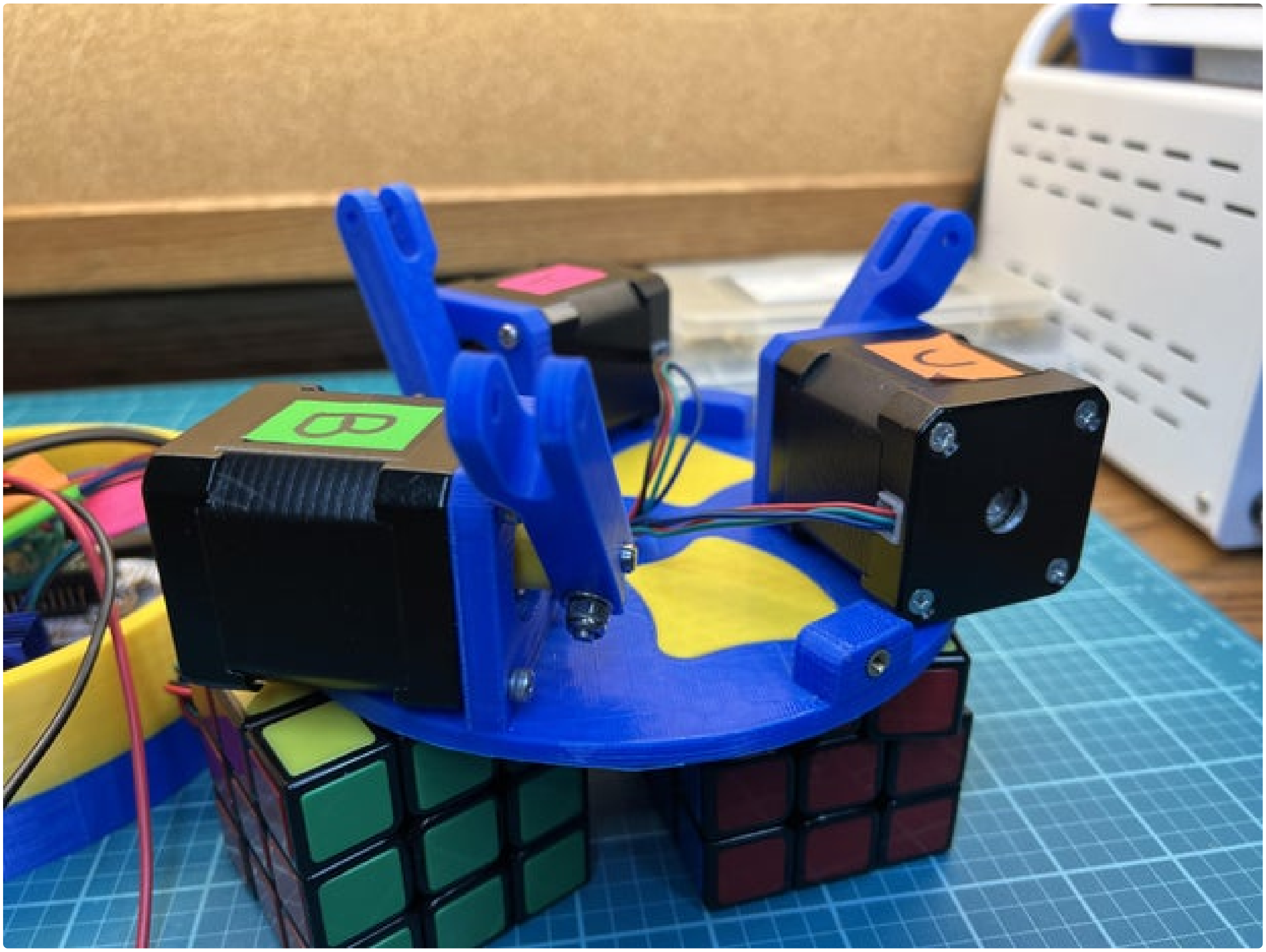






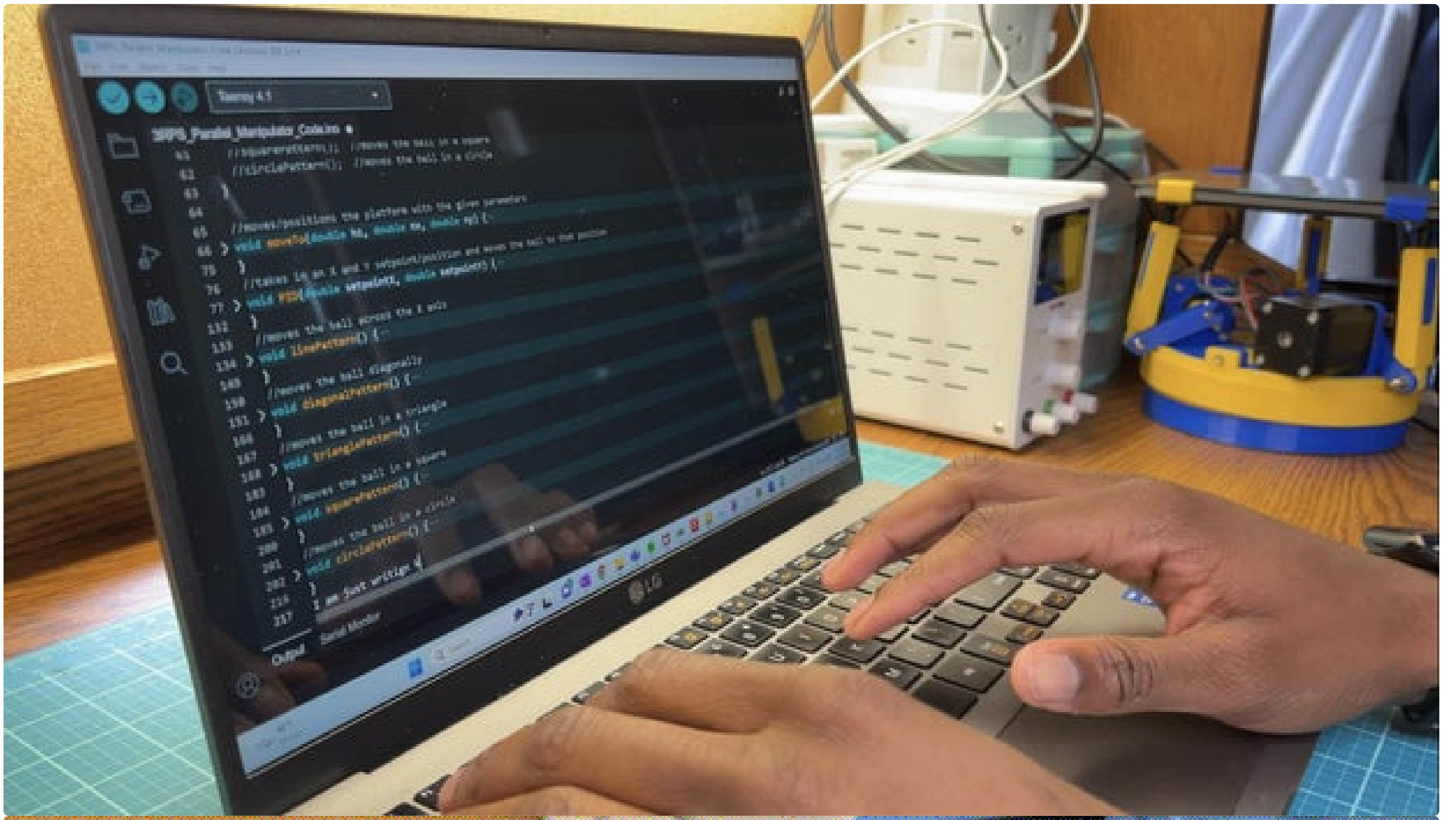

C





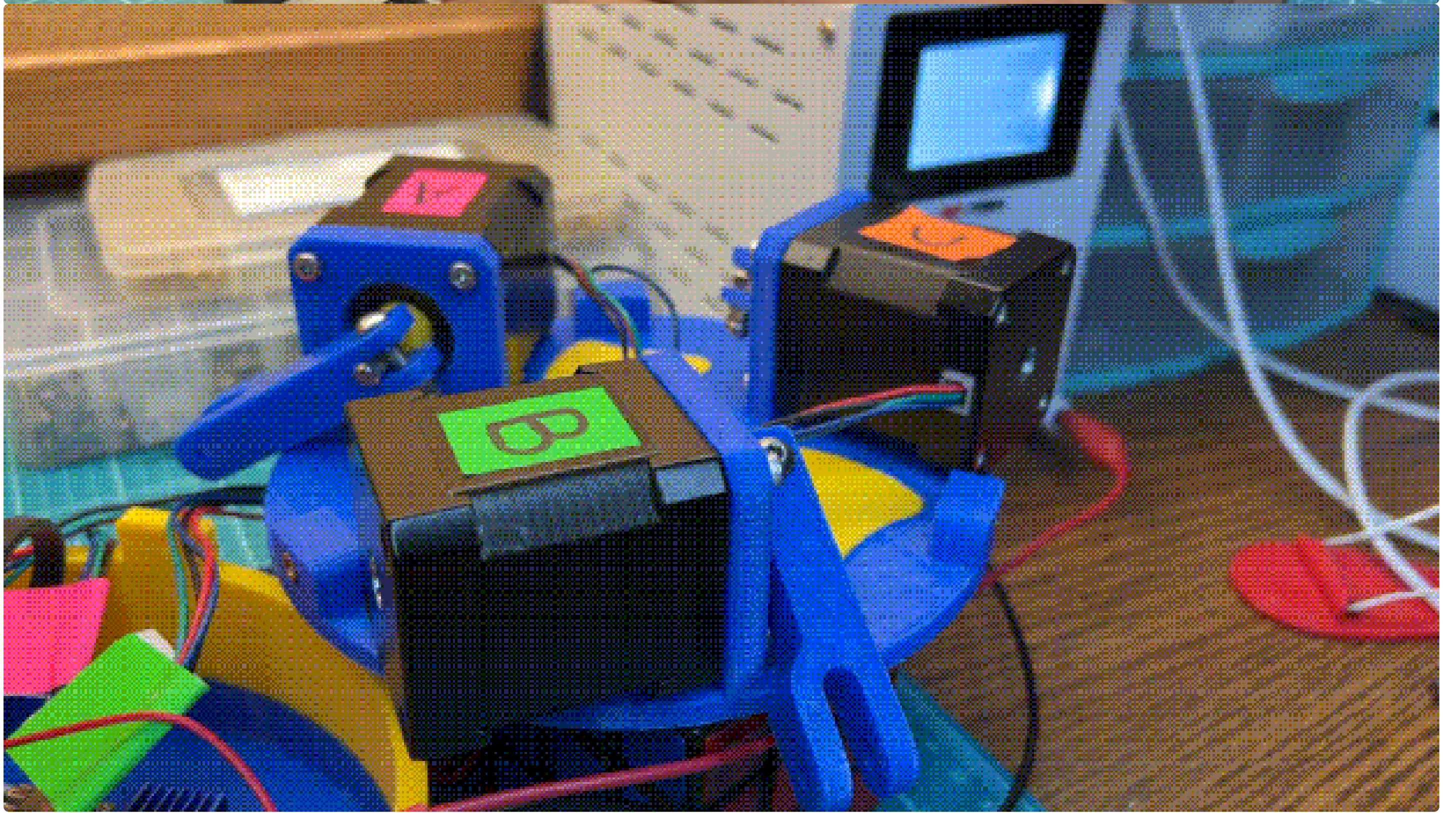
Step 5: Upload the Sketch

- First download these three Arduino libraries
- **AccelStepper**
- **InverseKinematics**
- **Adafruit_TouchScreen-master**
- Confirm that the stepper motors move in the right direction by uploading the **Stepper_Test** Arduino sketch to the teensy. All of the stepper motors should move inward. If any moves outward flip its connection to its driver.
- Next, Upload the **Ball Balancing** sketch to the teensy. This is the code that allows the robot to balance the ball.



```
//squarePattern(); //moves the ball in a square
//circlePattern(); //moves the ball in a circle
}

//moves/positions the platform with the given parameters
void moveTo(double hz, double nx, double ny) {…
}
//takes in an X and Y setpoint/position and moves the ball to that position
void PID(double setpointX, double setpointY) {…
}
//moves the ball across the X axis
void linePattern() {…
}
//moves the ball diagonally
void diagonalPattern() {…
}
//moves the ball in a triangle
void trianglePattern() {…
}
//moves the ball in a square
void squarePattern() {…
}
//moves the ball in a circle
void circlePattern() {…
}
I am just writing
```



Download

https://www.instructables.com/FZ6/KXB8/LGWA3BBC/FZ6KXB8LGWA3BBC.ino

Download

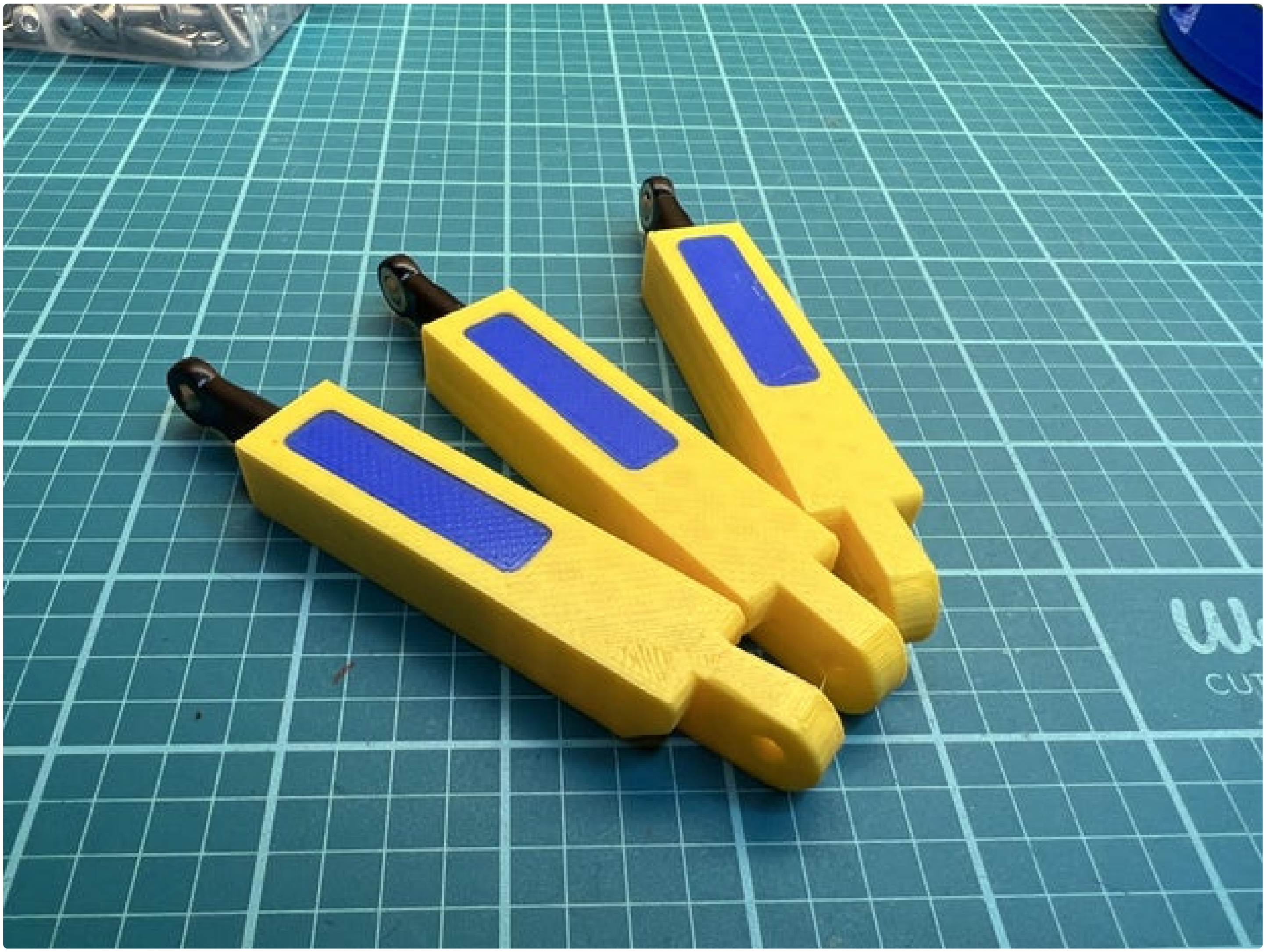
https://www.instructables.com/FS6/157R/LGWA3BBD/FS6157RLGWA3BBD.ino

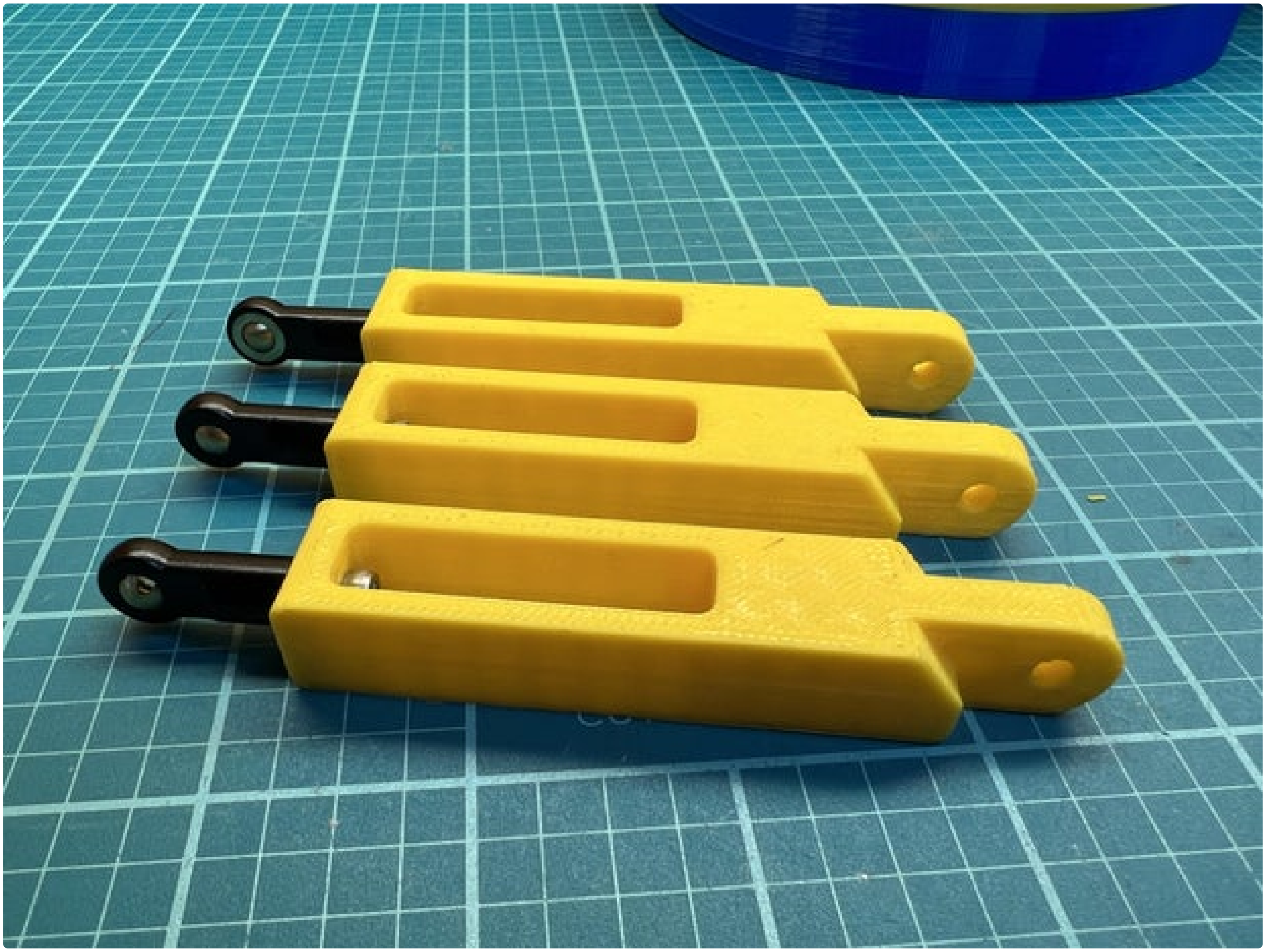
Download

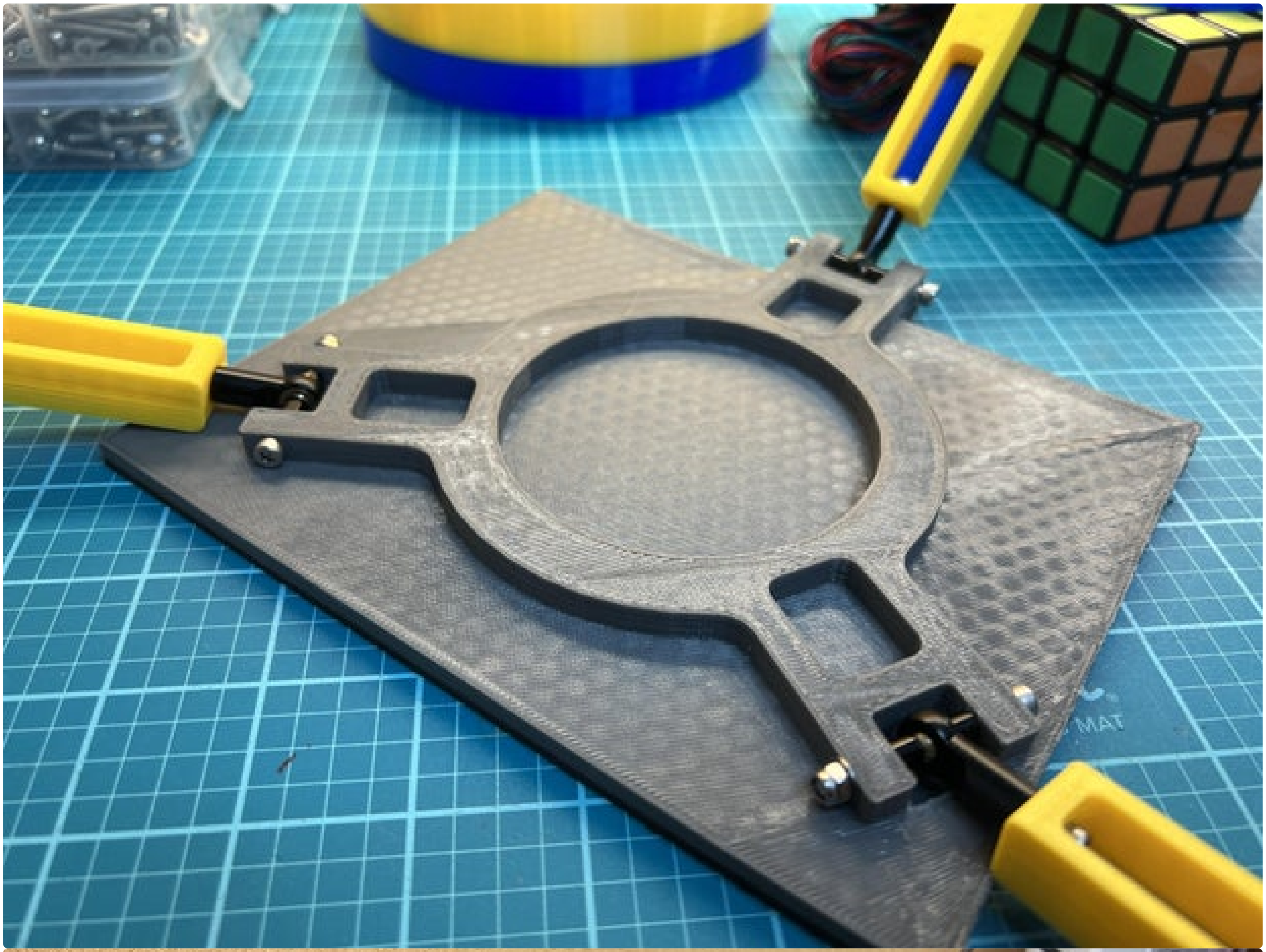
https://www.instructables.com/FCE/RFLY/LGWA3BBE/FCERFLYLGWA3BBE.ino

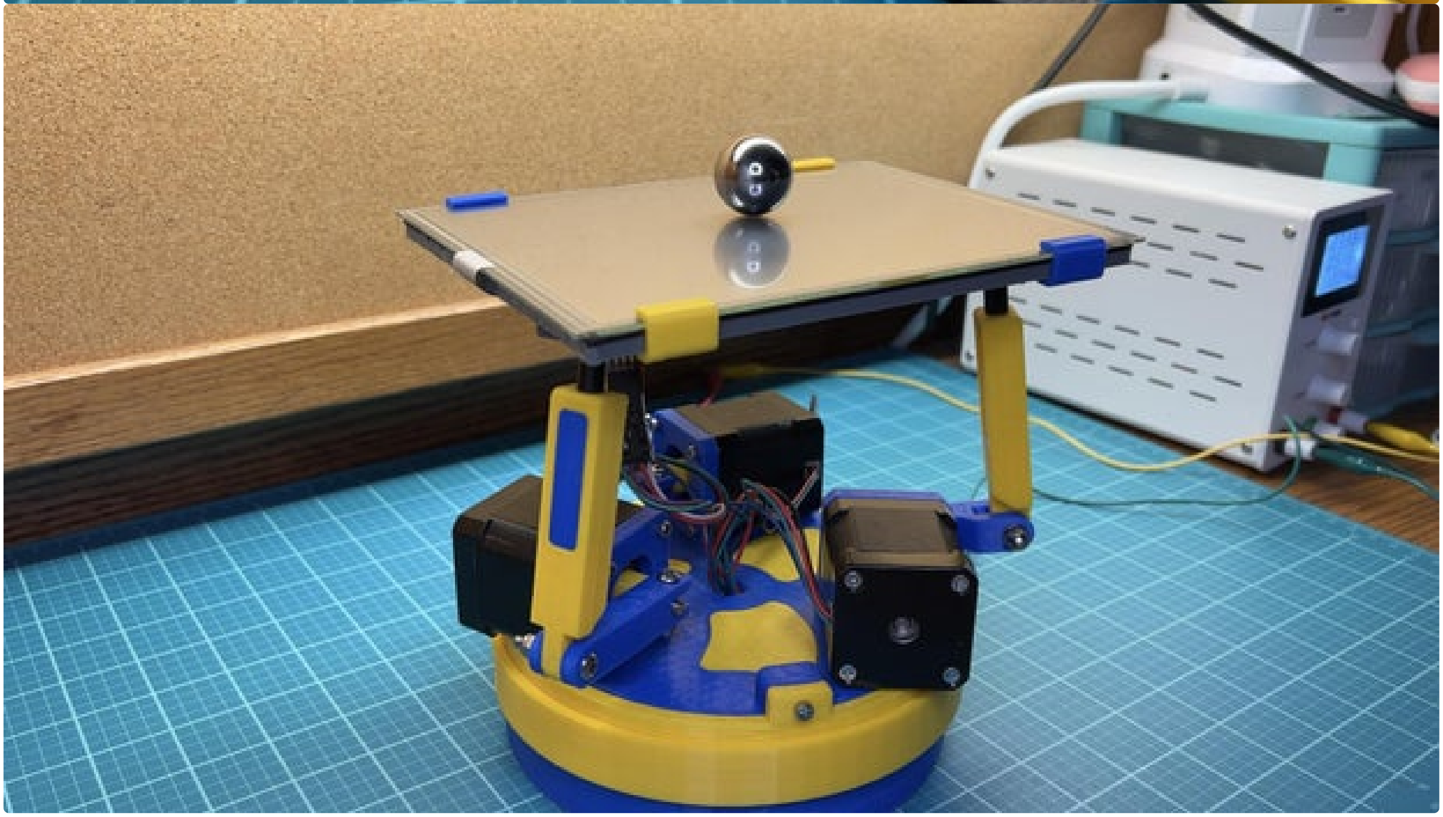
Step 6: Finish the Assembly

- On each link_#2, screw a tie rod onto the end using an **M3 x 8mm screw**.
- Screw each link_#2 to the platform frame (on the tie rod end) using an **M3 x 35mm screw**. You will also want to use **M3 x 5mm standoffs (x2)**to fill in the gap on either side of each tie rod.
- Screw the other end of each link_#2 to the end of each link_#1 using an **M4 x 25mm screw** and **M4 locknut**. While doing this ensure that Stepper A connects to the side of the platform frame that is labeled on image #3.
- Screw the base plate onto the base stand using **M3 x 8mm screws (x3)**.
- Using the four retainer clips, clip the resistive touchpad onto the platform frame.









Step 7: Operation

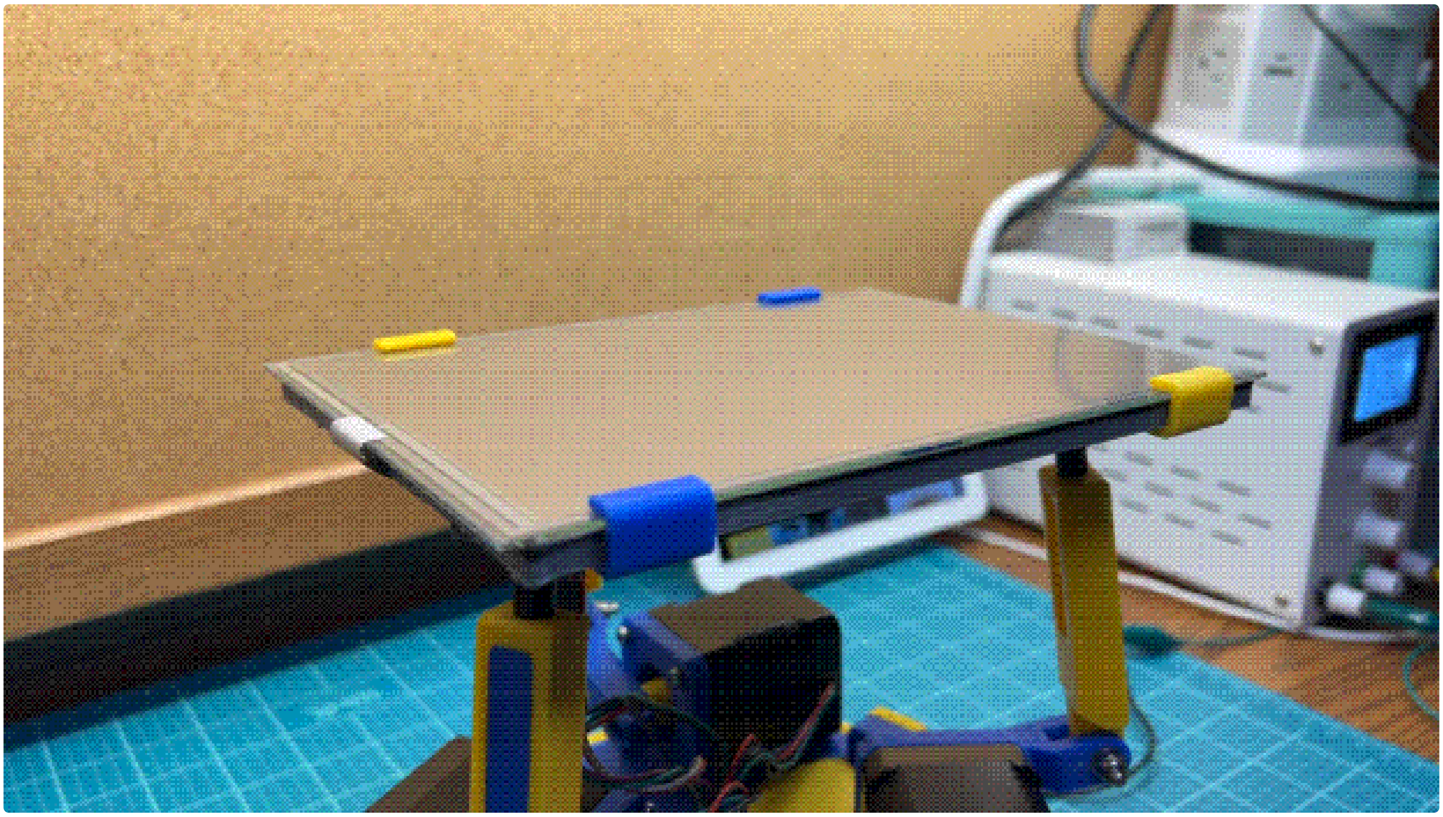
To get the robot up and running simply ensure that the platform starts at a fully down and flat position, turn on the robot, and place the steel ball on it. The platforms should now balance the ball.

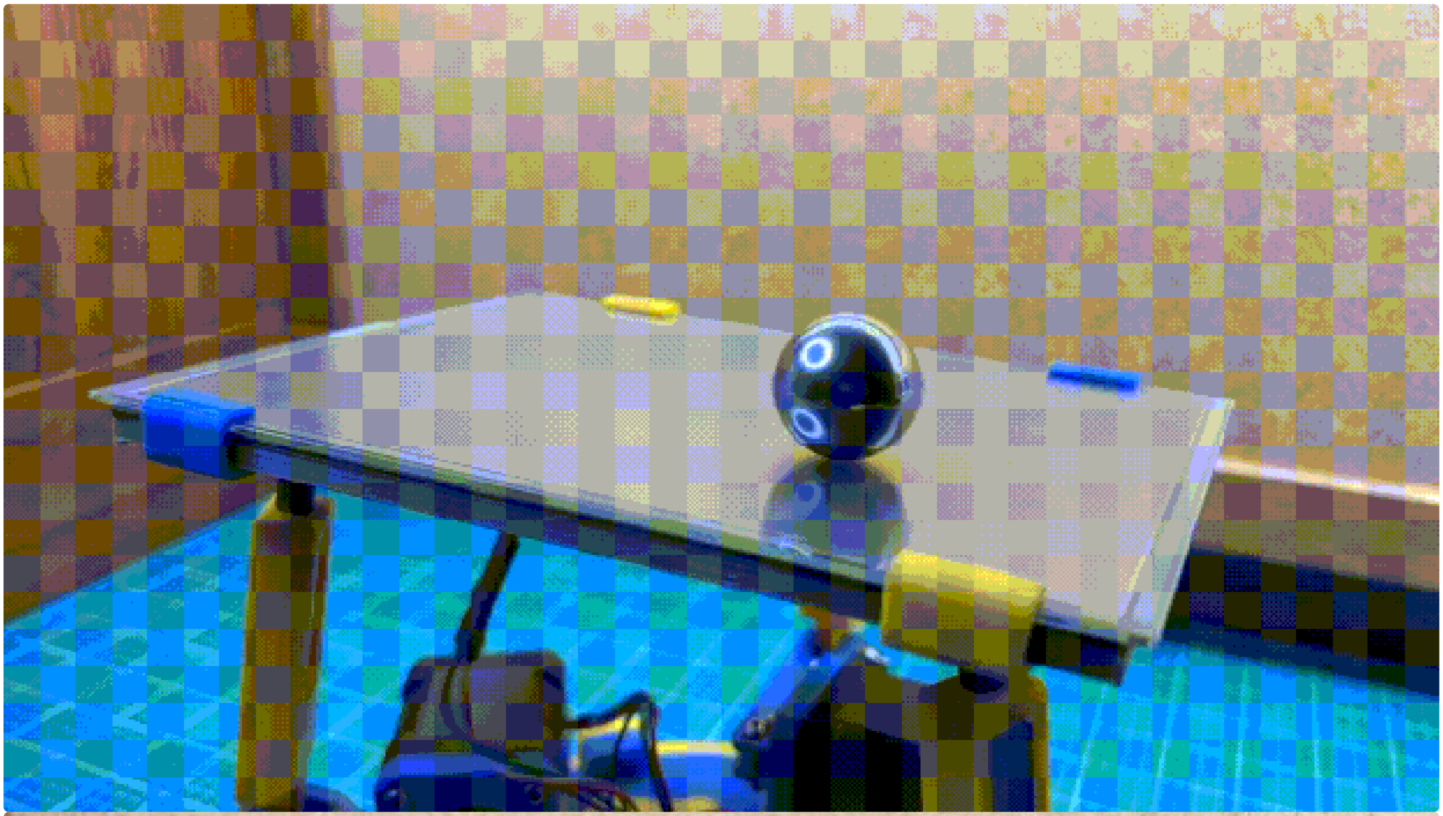
Patterns:

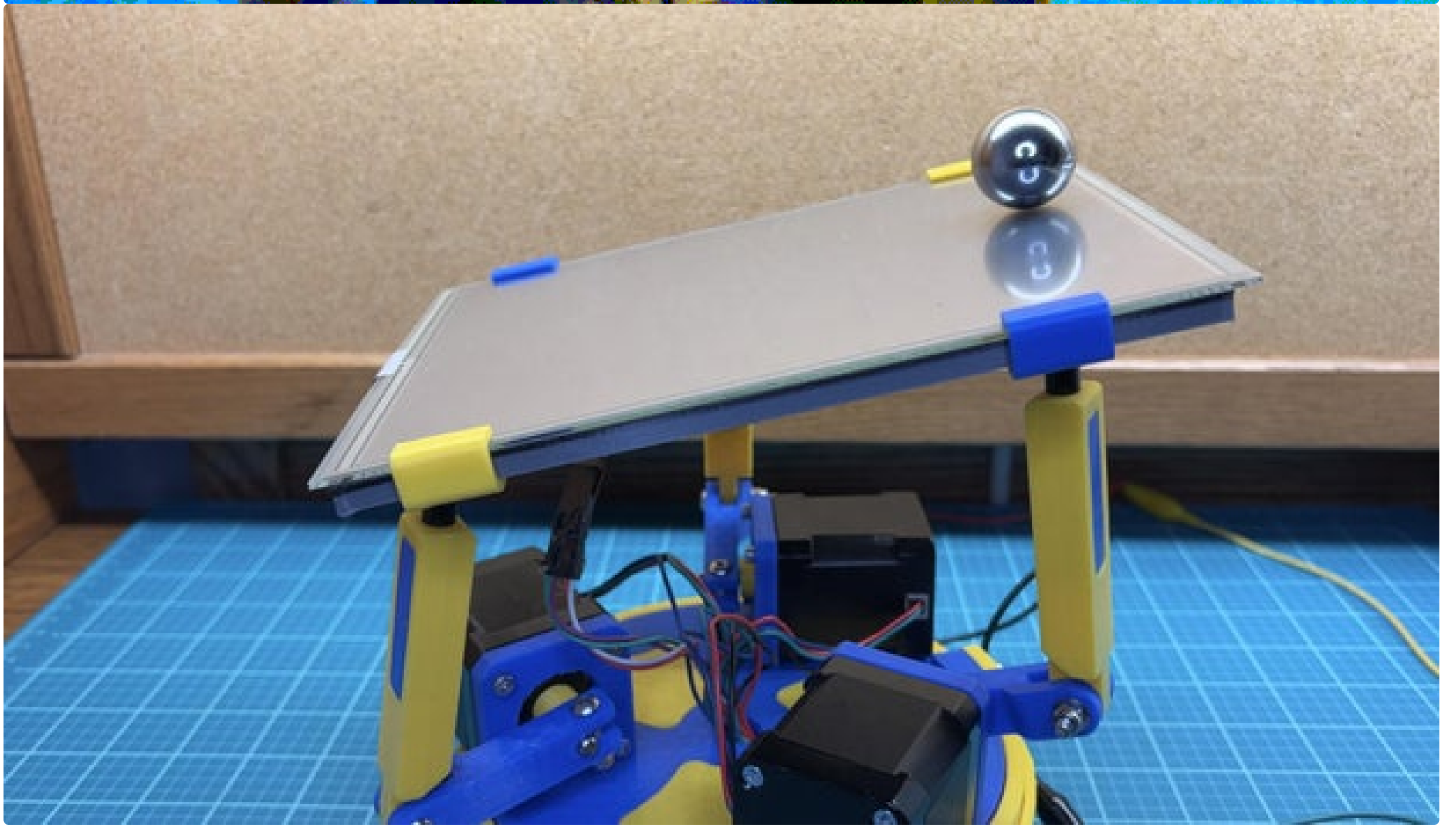
The initial uploaded program is set to balance the ball, but I have also programmed a couple of different patterns to move the ball around in. By uncommenting a different line in the code, you can make the platform move the ball around in a line, a triangle, a rectangle, a circle etc....

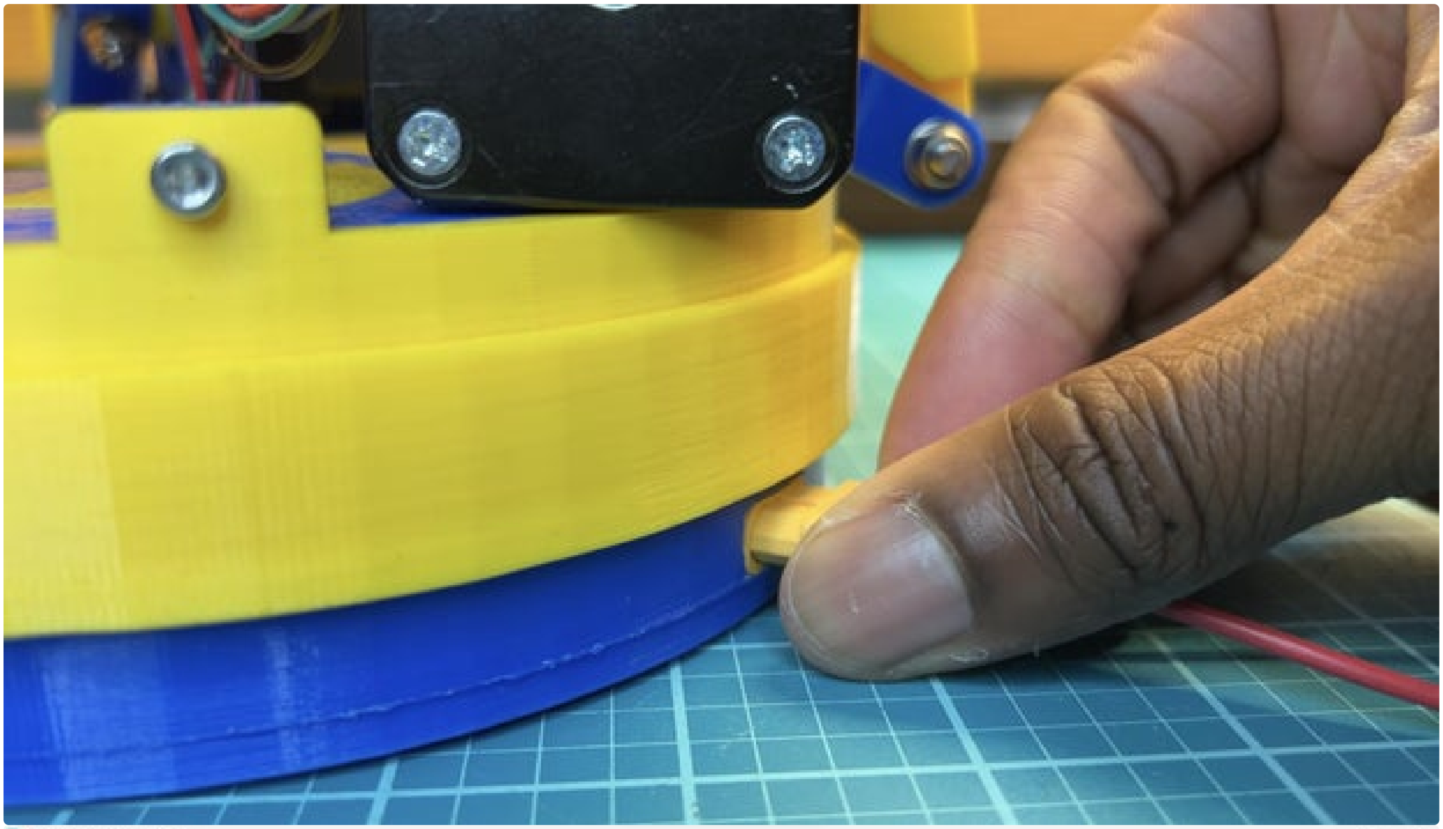
Changing the Offset Values

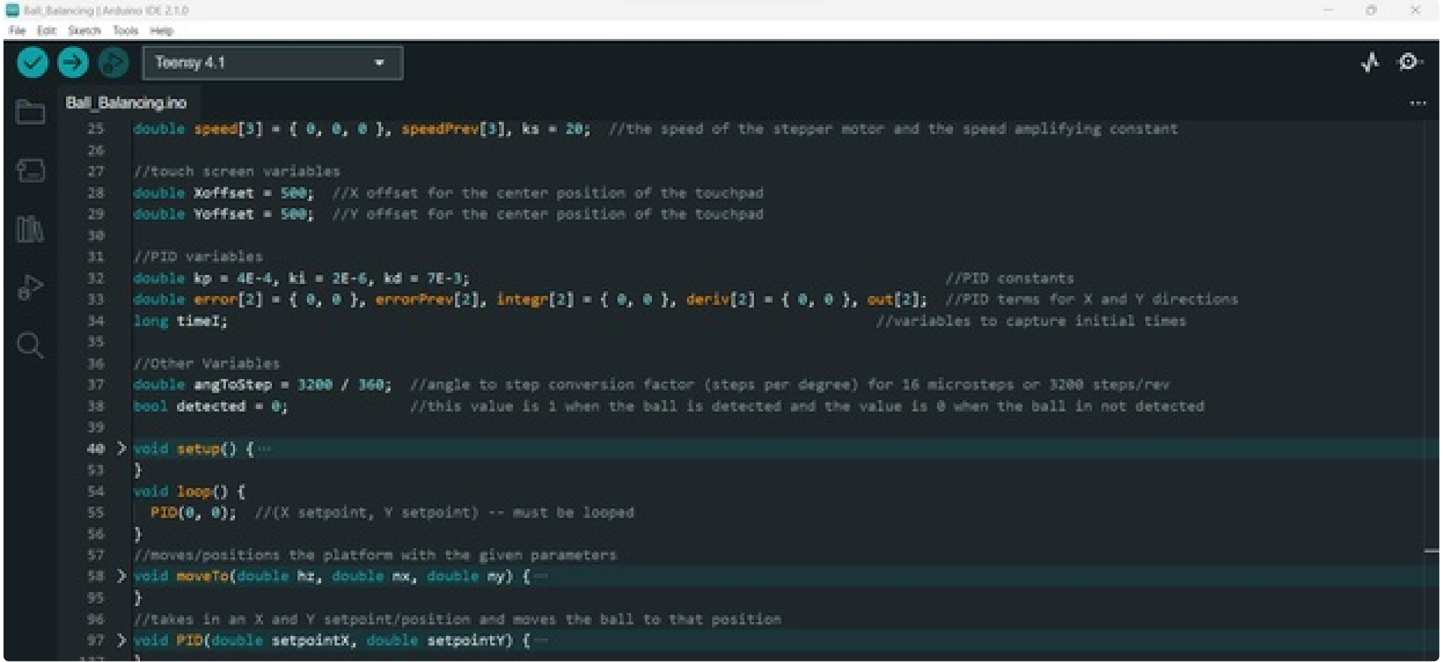
If you find that the ball doesn't move exactly to the center of the platform, you can change the X and Y offset values in the code. If you are feeling adventurous (and you know a bit about PID algorithms) you can also change the proportional gain (kp), integral gain (ki), and derivative gain (kd) values in the code to get sharper or softer reactions from the robot.










Ball_Balancing | Arduino IDE 2.1.0
File Edit Sketch Tools Help
Teensy 4.1
Ball_Balancing.ino

```
double speed[3] = { 0, 0, 0 }, speedPrev[3], ks = 20;  //the speed of the stepper motor and the speed amplifying constant

//touch screen variables
double Xoffset = 500;  //X offset for the center position of the touchpad
double Yoffset = 500;  //Y offset for the center position of the touchpad

//PID variables
double kp = 4E-4, ki = 2E-6, kd = 7E-3;                                                         //PID constants
double error[2] = { 0, 0 }, errorPrev[2], integr[2] = { 0, 0 }, deriv[2] = { 0, 0 }, out[2];  //PID terms for X and Y directions
long timeI;                                                                                 //variables to capture initial times

//Other Variables
double angToStep = 3200 / 360;  //angle to step conversion factor (steps per degree) for 16 microsteps or 3200 steps/rev
bool detected = 0;              //this value is 1 when the ball is detected and the value is 0 when the ball in not detected

void setup() { ⋯
}
void loop() {
  PID(0, 0);  //(X setpoint, Y setpoint) -- must be looped
}
//moves/positions the platform with the given parameters
void moveTo(double hz, double nx, double ny) { ⋯
}
//takes in an X and Y setpoint/position and moves the ball to that position
void PID(double setpointX, double setpointY) { ⋯
```

Download

https://www.instructables.com/FWL/3P3Z/LGWA3D5A/FWL3P3ZLGWA3D5A.ino

Step 8: How It Works

Design Approach

The robot is a 3RPS parallel manipulator. This is essentially a 3DOF platform with 3 legs each having a rotational joint and a spherical joint. Extending or contracting each leg will result in the platform being angled in a different way. This is done by each stepper motor. My biggest considerations when designing the balancer was to have sleekness and simplicity.

For sleekness, I ensured that the electronics and most of the wires were well hidden. For simplicity, I decided to use 3 motors (3DOF) rather than the 6 motors (6DOF) used in version 1. Weighing the cost-benefits, I also chose not to use position encoders with the stepper motors. Using encoders would only eliminate the need to have the machine start at a common position. Without them, you simply just have to push the platform down every time.

Inverse Kinematics

The machine is able to angle itself through the use of inverse kinematic equations. These equations calculate the position to move each stepper motor in order to achieve the desired platform orientation. I derived these equations myself mainly using vector calculus, trigonometry, and lots of patience. The equations are fully parameterized meaning that they would work with any 3RPS parallel manipulator of any shape or size.

Ball Balancing PID Algorithm

The ball-balancing aspect of the code uses a PID algorithm. The algorithm constantly finds the most effective platform orientation to balance the ball.

- **Error** - The error is defined as the distance from the ball to the setpoint. The setpoint in this case is the center of the platform.
- **Proportional Term (P) = kp*error ––** The proportional term is proportional to the error
- **Integral Term (I) = ki*∫error ––** The integral term is the accumulation of the error as time moves on. The integral term helps to fine-tune the position of the ball and ensure that the ball is exactly at the center.
- **Derivative Term (D) = kd*(d/dt)*error –** The derivative term is the derivative of the error which is essentially the speed of the ball. This term helps to slow the ball down.
- **Output = P + I + D ––**The output value is the sum of the proportional, integral, and derivative terms. This output is essentially the most effective orientation to orient the platform in order to keep the ball from falling. The robot is constantly moving the ball toward the center (P), fine-tuning the position of the ball (I), and slowing the ball down (D). Each term in the output is tuned with its respective constants (kp, ki, kd) also known as the proportional gain, integral gain, and derivative gain. One thing that I found during testing is that the robot can balance the ball with a PD algorithm just as well as it can with a PID algorithm. The integral term just gets the ball exactly to the center of the platform if it's a little off.

The control algorithm consists of 2 PID algorithms for the X and Y directions of the error.

Stepper Motor Control

Simply having a ball-balancing algorithm only got me so far with this project. During my initial testing, I found that I had very jittery motion. To solve this, I decided to make the speed and acceleration of each stepper motor proportional to the difference between the motor's current and target positions. In other words, when a motor is close to its target position its speed and acceleration are low, and when it's far away from its target position its speed and acceleration are high. In addition to this, I also decided to switch the drivers to 1/16th micro stepping which means that instead of having 200 steps/revolution they have 3200 steps/revolution which makes actuation much more smooth and more precise.

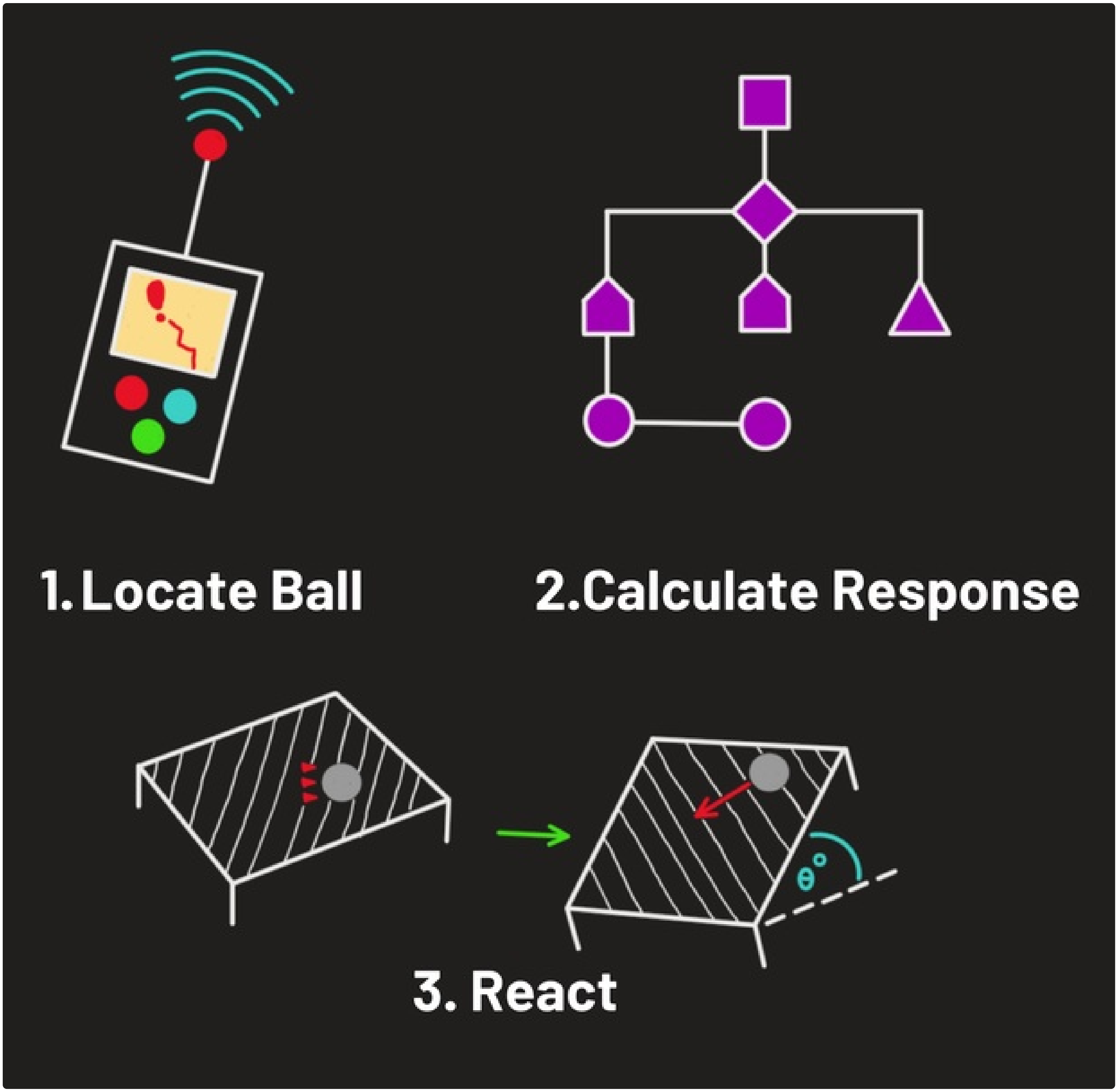
Synopsis of Events

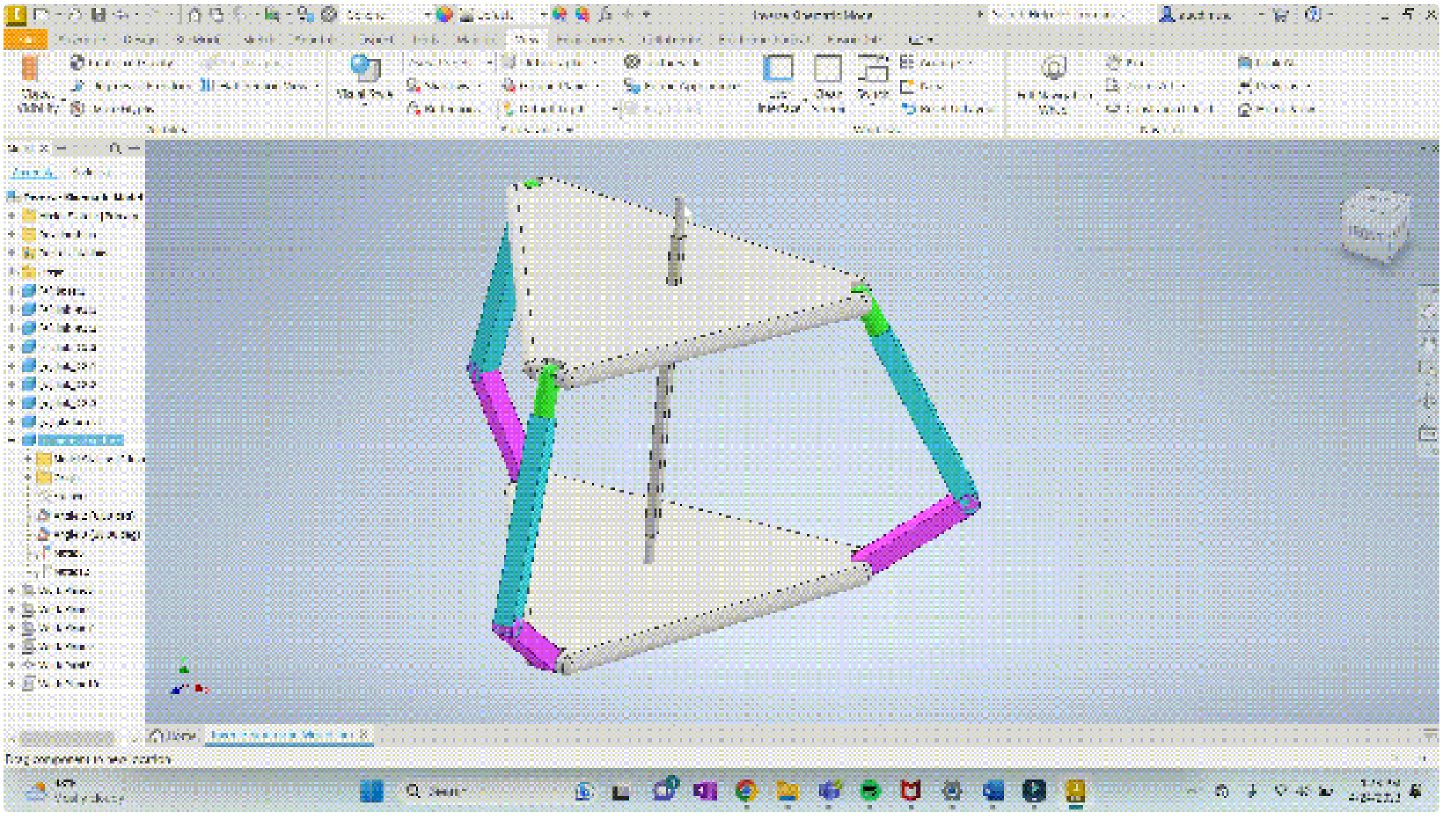
1. The touchpad reads the position of the ball, waits 20 milliseconds, and reads the position of the ball again.
2. The program then calculates the terms in the PID algorithm. The output (aka the projected orientation of the platform) is fed to the inverse kinematic equations. These equations derive the position to move each stepper motor in order to achieve the platform's projected orientation.
3. With the new stepper motor positions, the speed and acceleration of the stepper motors are calculated and set. The stepper motors are then moved to their target positions which put the platform in the orientation that best slows the ball down and moves it to the center. The process is then repeated.

Patterns

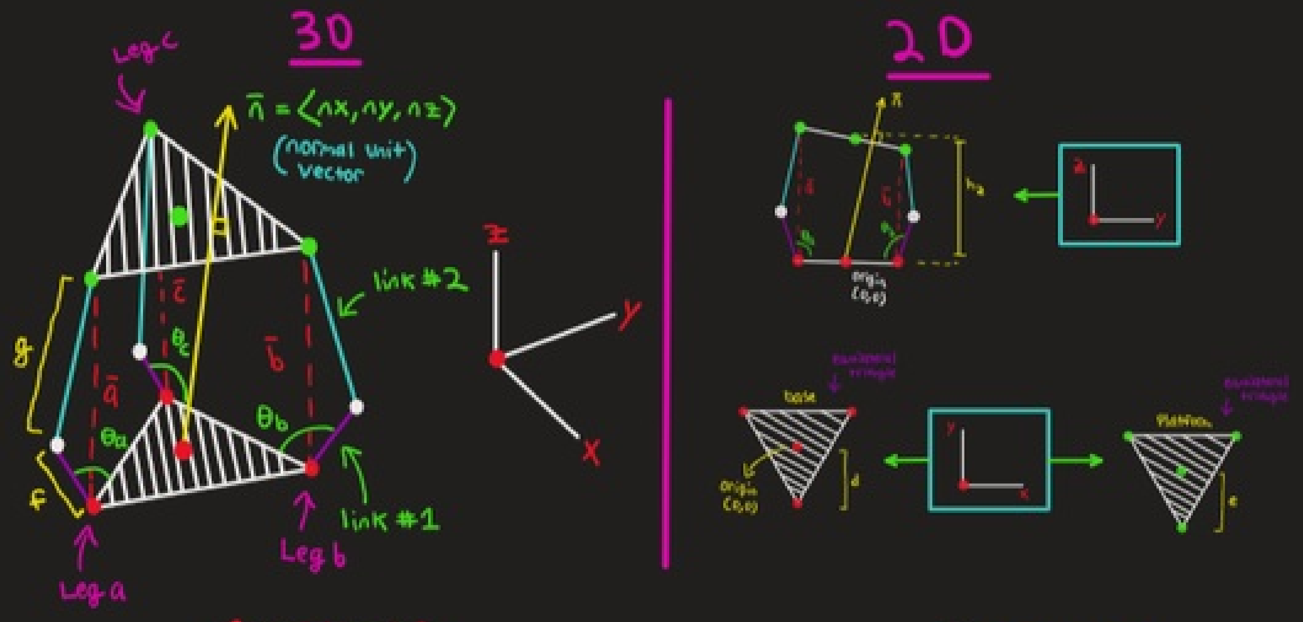
The balancer can also move the ball around in different patterns. Implementing this was fairly simple. The ball balancing algorithm moves the ball to the center of the platform. But if you change what the platform thinks is the center (ie change the setpoint) then you can move the ball to a different point on the platform. Constantly doing this creates motion and doing this with equations creates shapes. You can essentially create any shape that can be described in rectangular equations, parametric equations, or polar coordinates. Here are the shapes that I was able to create.

- line
- rectangle
- triangle
- circle
- ellipse
- sinusoidal functions
- zigzags
- figure 8


1. Locate Ball
2. Calculate Response
θ°
3. React


Drag component in new location

3RPS Parallel Manipulator Inverse Kinematics



Constants

d → distance from the center of the base to any of its corners

f → length of link #1

e → distance from the center of the platform to any of its corners

g → length of link #2

Parameters

h → Platform height

$\bar{n} = \langle nx, ny, nz \rangle$ → Unit Vector normal to the Platform Plane

Leg a

$$a_y = d + \left(\frac{e}{2}\right)\left(1 - \frac{nx^2 + 3nz^2 + 3nz}{nz + 1 - nx^2} + \frac{nx^4 - 3nx^2ny^2}{(nz+1)(nz+1-nx^2)}\right)$$

$$a_z = h_z + e\,ny$$

$$a_m = \sqrt{ay^2 + a_z^2}$$

$$\boxed{\theta_a = \cos^{-1}\left(\frac{ay}{a_m}\right) + \cos^{-1}\left(\frac{a_m^2 + f^2 - g^2}{2a_m f}\right)}$$

Leg b

$$bx = \frac{\sqrt{3}}{2}\left(e\left(1 - \frac{nx^2 + \sqrt{3}\,nx\,ny}{nz+1}\right) - d\right)$$

$$by = \frac{bx}{\sqrt{3}}$$

$$b_z = hz - \frac{e}{2}(\sqrt{3}\,nx + ny)$$

$$b_m = \sqrt{bx^2 + by^2 + b_z^2}$$

$$\boxed{\theta_b = \cos^{-1}\left(\frac{\sqrt{3}\,bx + by}{-2b_m}\right) + \cos^{-1}\left(\frac{b_m^2 + f^2 - g^2}{2b_m f}\right)}$$

Leg c

$$cx = \frac{\sqrt{3}}{2}\left(d - e\left(1 - \frac{nx^2 - \sqrt{3}\,nx\,ny}{nz+1}\right)\right)$$

$$cy = -\frac{cx}{\sqrt{3}}$$

$$c_z = hz + \frac{e}{2}(\sqrt{3}\,nx - ny)$$

$$c_m = \sqrt{c_x^2 + c_y^2 + c_z^2}$$

$$\boxed{\theta_c = \cos^{-1}\left(\frac{\sqrt{3}\,cx - cy}{2c_m}\right) + \cos^{-1}\left(\frac{c_m^2 + f^2 - g^2}{2c_m f}\right)}$$

I think there may be a typo on this line:
Screw the base plate onto the base stand using **M3 x 3mm screws (x3)**.
a 3mm won't be long enough for this connection (unless I missed something), I used an 8mm as that's what I had laying around

Thanks for catching that. It has been corrected now.

Did this overkilled robot with buds for our term project. We had a blast doing it and we definitely had the best project in the whole class.

Awesome!!!

Hi, I'm planning to do such a project. Can you please tell me why we need the disk that is sold together with the resistive panel? I just plan to buy everything separately and there is no disk there

The disc is not needed for this project.

I am studying electrical engineering with a focus on control engineering and wanted to put my knowledge to practical use. Thanks for the inspiration!
(I don't like it that colorful, so I decided to use black and dark blue PLA).

Awesome!

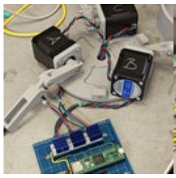
Hey, I am very impressed by your project. I am trying to build and adapt your project but for that I want to understand the kinematics of the system which is not going great. Do you maybe by chance have some files where the calculations and derivations are writen down? Kind regards a fellow engineering student

Sure thing, send me your email in a private message and I can send my derivations over to you.

Great project !!! I'm trying to build one.
But,,, it it possible to use Arduino Uno or Mega board instead of Teensy?

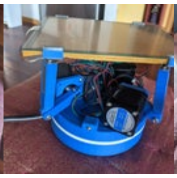
I have not tried this but I believe it should work

Great project! Thank you for such an amazing design :)
After putting it together, I realized that something seemed reversed and after looking closely, my touchscreen had the bent looking downwards compared to your images, so I had to rotate it 180 degrees and swap motors A and C.
Maybe a different model? or a glitch on their manufacturing?



Interesting, perhaps it's just inconsistent manufacturing. Anyways great job! Looks awesome!

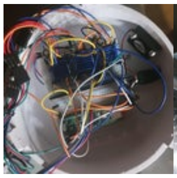
Thank you Aaed for giving me the confidence to tackle this project. Your instructions and format was easy to follow. I did struggle with the touch screen and had originally ordered the wrong size stepper motors. Building the circuit boards and printing up the parts was straight forward.



Built this drop off and catch plate as an added feature. Be interesting what other approaches can be used.

Can you share your notes? i want to see it to study.thinks

It's Great, but I would recommend modding the base with a mini pc fan. I had troubles with getting the recommend 5v reg to work so I went for the more conventional type and work out of the box!. I also used mini bread boards(2) as it is much easier than soldering each pin to the recommended board. Oh!. I almost forgot about the power supply I changed, I did not want to use a MASSIVE power supply that looked kinda ugly, So I used this! I had many things break because I was stupid sometimes, so I am going to tell you what to do, to not have to re-order parts. 1:Buy the teensy pre soldered. 2:when using the 5volt regulator I recommended use a heat sink, and capacitors(ask me I found that out). 3:When the motor controllers over heat, they just stop the server set up the fan close to the motor controllers. 4:pay attention to the direction of the capacitor(gets me every-time 4th of July happens on my desk).5:It works just fine printing at 10% infill.
If you got this far on my rant, thankyou for reading.



This is an amazing project... Thanks for sharing all the incredible work that went into this.

You got my attention a few month back with the first Ball Balancer project. Your version 2 is much slicker and easier to build. My granddaughters (10 &11) are going to build this a summer project. Great introduction to physics, math and engineering. We will share the results in a few months.

I couldn't find the 5 volt regulator in the Bill of Material. I would think any 7805 regulator will work, but what do you recommend?

By the way - I love your style.

Yes, that should work. I've now added the link to the voltage regulator that I used. Pretty much any 5V regulator should work. Good luck with the build!

Hi great project and design, I'm planing on building one, as per your intrusions.

Are you able to provide info on the touchpad you used ?

Thanks.

What kind of info? I’ve linked the touchpad that I used in the supplies section.

sorry, I couldn't see it on the list, but since found it,
thanks
looking forward to building this project

Waouh! You've done an excellent job! I really like this instructable, the way you designed and explain your project, the videos, and the final result. Especially when the ball 'dances', just by moving the 'center'. BRAVO !
Have you tried it in a moving environment, such as in a car, a sailing boat, or any other moving situation? I'm pretty sure it still works...

Thanks! And no, I haven’t tried it any of those situations.

Bravo!
Fantastic Instructable!
Excellent write up and video (and gifs!)
Your work is enviable! Thanks for sharing!
(But, but *why* is it entered in the Battery-powered contest? There is no battery mentioned. :-) [You should save this and enter it in a different contest.]

Well it doesn’t run on hopes an wishes 😂. It does use a power supply aka a big battery.

LOL. Power supply plugged into the wall outlet does not equal "battery-powered". Just wanted you to be able to compete. It's too good of a project not to win!

I see your point, good eye. I guess we’ll leave that one up to the judges.

LOL at codeline 217 in 4:23!
Nice job!

You caught that nice! 😂

You did it again! This time it is a real product. Comparing to the last one that looks now like a toy. I know, you spent all nights for this. I appreciate it.
I was born too early, I would love to things like this in your age.
The equations look really complex. Could you make some estimations and simplify them?
Your solution of controlling the acceleration too is nice, although I know that it added complexity to the PID.
GOODJOB!

Thanks! The equations could probably be simplified using some assumptions but I really wanted exacts equation for this project because I didn’t want any room for error with the code.

I am nowhere near that level of math.
I can't think about seeing that level of math from where I am.
I do however fully appreciate that you can 😁

You did an amazing bunch of thinking, designing, programming, and implementing. (And Mathing).

Great project. Seriously.

Now, mass produce them and I'll get one when they go on sale 😁.

Haha thanks! You'll be the first to be notified once these are mass produced lol.

My wife and I watched this video and were amazed!
All that expert engineering, a sense of rhythmic and humor to boot!
Now, go save the planet - it needs you!

Awesome!

I totally loved that video and the Eurythmics song was right on. Besides all the hard work that went into making this, it as well engineered. Gen 2 was leaps and bounds better including the "play time" stuff. I'm going to guess you literally "dreamed" this project.

You have a great career in front of you. Always have fun!

Thanks! This was definitely a fun one to make.