Tablet 1DOF

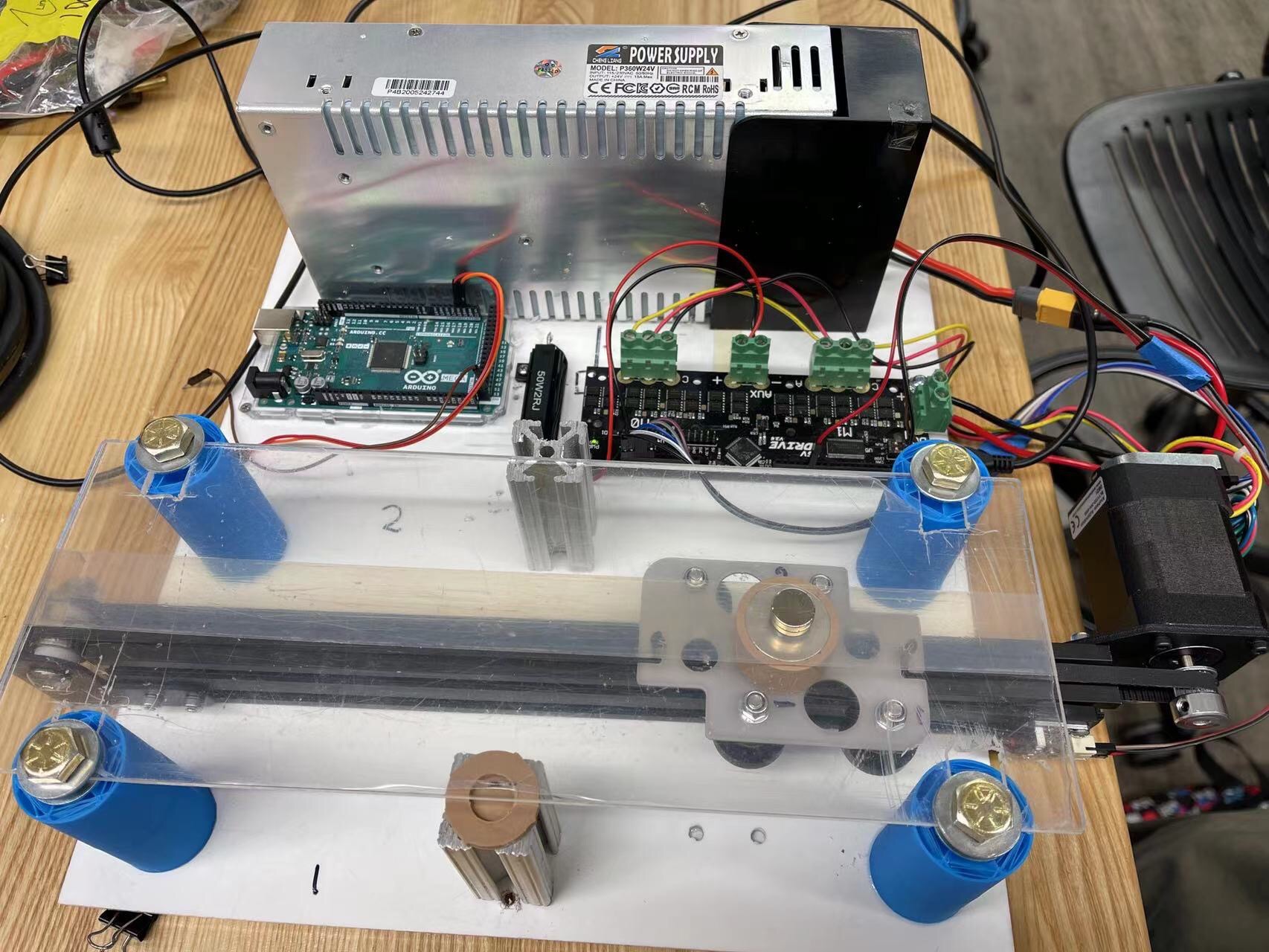
Part 4: Controls and Documentation

Lio Liang, Yujui Chen, Cam Mastoras

1. Stiff-wall prototype:

Compared to the previous prototype, we add the tablet on top of the linear rail and tape with PTFE Teflon tape which reduces the friction between the tablet surface and the magnet. In our design, if we want to increase the stiffness of the magnet which creates the virtual wall to prevent it going off, it will increase the friction of the free motion plate and vice versa. Therefore, we have to choose the benefit we want from stiffness or free motion and we can brainstorm other methods to reduce the friction.

* Picture:



* Demo video link:

<https://drive.google.com/file/d/1b-5rsob17bxKr4VI0O48QQaTTQc6v787/view?usp=sharing>

1. Block diagram:



The block diagram has been updated to include more detail about how the virtual environment is constructed, and also more detail on the interaction between the two magnets. Basically, the user provides a position via their magnet. They give that position to the gantry magnet through the “spring” of the lateral component of the magnetic interaction, and receive some force. The encoder takes the position of the gantry magnet, and passes it to the impedance controller through the ODrive. The impedance controller looks at a map of the virtual environment to decide what force to apply to the user. It then passes the desired current/Torque output to the ODrive and thus to the motor. The motor drives the gantry to provide force on the gantry magnet, which modulated through the magnet, is felt by the user.

1. Stiff-wall interaction:

Code Overview  
Our control system for the 1DOF prototype demo was a simple impedance control algorithm that modeled a virtual wall as a spring. When the user passed the virtual wall, an artibrarily set position threshold, the robot would provide a torque pushing the user back out of the wall from the direction they came. The more they push, the greater the force will be, up to a limit that we prescribed for safety reasons

1. Comments of source code:

* Python script

**import odrive**

**from odrive.enums import \***

**# import the odrive Python API**

import time

# using time.sleep() method to block the code waiting for process

import csv

# output the data to csv file for further analysis

print("finding an odrive...")

my\_drive = odrive.find\_any()

# finding the device

print("starting calibration...")

**my\_drive.axis0.requested\_state = AXIS\_STATE\_FULL\_CALIBRATION\_SEQUENCE**

**# start calibration with axis state request**

while my\_drive.axis0.current\_state != AXIS\_STATE\_IDLE:

time.sleep(0.1)

# waiting for calibration

**my\_drive.axis0.controller.config.input\_mode = INPUT\_MODE\_PASSTHROUGH**

**my\_drive.axis0.controller.config.control\_mode = CONTROL\_MODE\_TORQUE\_CONTROL**

**my\_drive.axis0.requested\_state = AXIS\_STATE\_CLOSED\_LOOP\_CONTROL**

**# Set the closed loop control with torque input**

my\_drive.axis0.controller.input\_torque = 0.016

while my\_drive.get\_adc\_voltage(5) > 0.3:

time.sleep(0.1)

# moving to the leftside (switch on the leftside) to calibrate the absolute position

my\_drive.axis0.controller.input\_torque = 0

# when hit the switch reset the torque

**ref = my\_drive.axis0.encoder.pos\_estimate**

**# recode the left end point position as reference coordinates**

**my\_drive.axis0.controller.config.control\_mode = CONTROL\_MODE\_POSITION\_CONTROL**

**my\_drive.axis0.controller.input\_pos = ref - 2**

**# use position control to move to the midpoint of rail**

**my\_drive.axis0.controller.config.control\_mode =**

**CONTROL\_MODE\_TORQUE\_CONTROL**

**my\_drive.axis0.controller.input\_torque = 0**

**# back to torque control and reset request value as zero**

torque = 0

with open("test.csv", "w") as f:

writer = csv.writer(f)

while True:

# loop

if my\_drive.axis0.encoder.pos\_estimate < ref - 3:

torque = -0.3 \* (my\_drive.axis0.encoder.pos\_estimate + 3 - ref)

elif my\_drive.axis0.encoder.pos\_estimate > ref - 1:

torque = -0.3 \* (my\_drive.axis0.encoder.pos\_estimate + 1 - ref)

else:

torque = 0

**# calculate the torque with two walls**

if torque < -0.04:

torque = -0.04

if torque > 0.04:

torque = 0.04

**# avoid to big torque damage device**

my\_drive.axis0.controller.input\_torque = torque

# print(my\_drive.axis0.encoder.pos\_estimate)

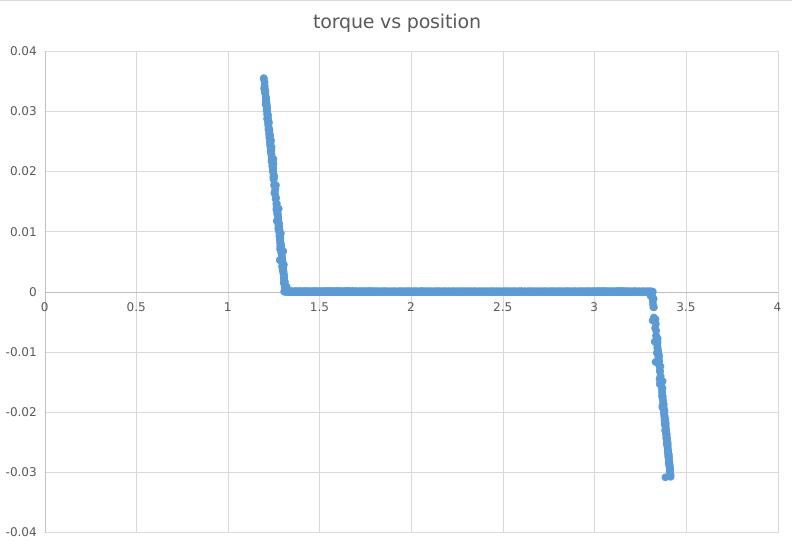
print(my\_drive.axis0.controller.input\_torque)

writer.writerow([my\_drive.axis0.encoder.pos\_estimate,

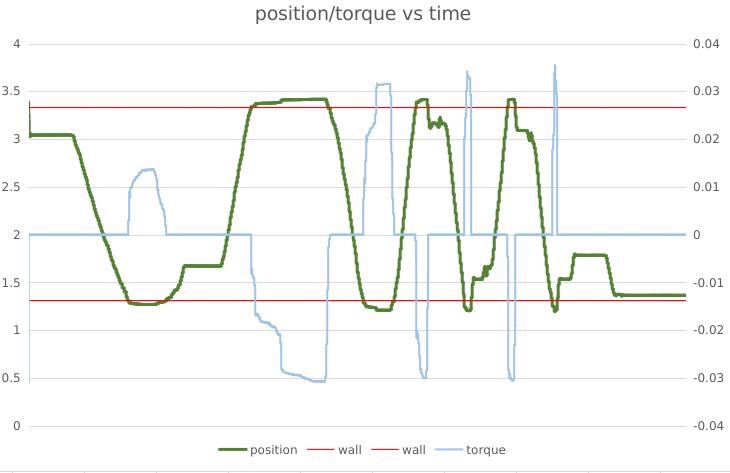
my\_drive.axis0.controller.input\_torque])

# output data for debug and further analysis

* Output of the position:
  + Torque(y-axis) vs position(x-axis)



* + Torque/position (y-axis) vs time (x-axis)



* Limit switch:

We insert the signal pin at GPIO 5 and utilize the function get voltage which is my\_drive.get\_adc\_voltage(5). While the voltage is equal to zero, it means the switch is attached. On the other hand, the voltage is 3.3V, the switch is open. Therefore, we set a threshold voltage equal to 0.3V and if it is smaller than the value the switch is on and vice versa.

while my\_drive.get\_adc\_voltage(5) > 0.3:

time.sleep(0.1)

# moving to the leftside (switch on the leftside) to calibrate the absolute position

1. Key findings

We discovered a couple things that will be really important going forward, with the most important one being that friction is one of the biggest obstacles in the way of achieving our original design goals. We found that there was really no way to win by picking the perfect magnet size. Too small, and free motion feels fine but virtual walls don’t feel like very much, and too big and while the walls are now stiff, the friction in free motion is pretty terrible. Thus, going forward we now truly understand that we will need a more intentional solution to combat problems of friction

1. Future Improvements

We have a number of improvements to the control system that we are planning to make in the coming weeks.

* Include a damper along with a spring in the virtual environment
* Add in anticogging
* More testing and datacollecting, especially in regards to our design goals of stiffness and inertia
* Experiment with different virtual formations