

DIY Motion Simulator

Michael Rechten - 2025

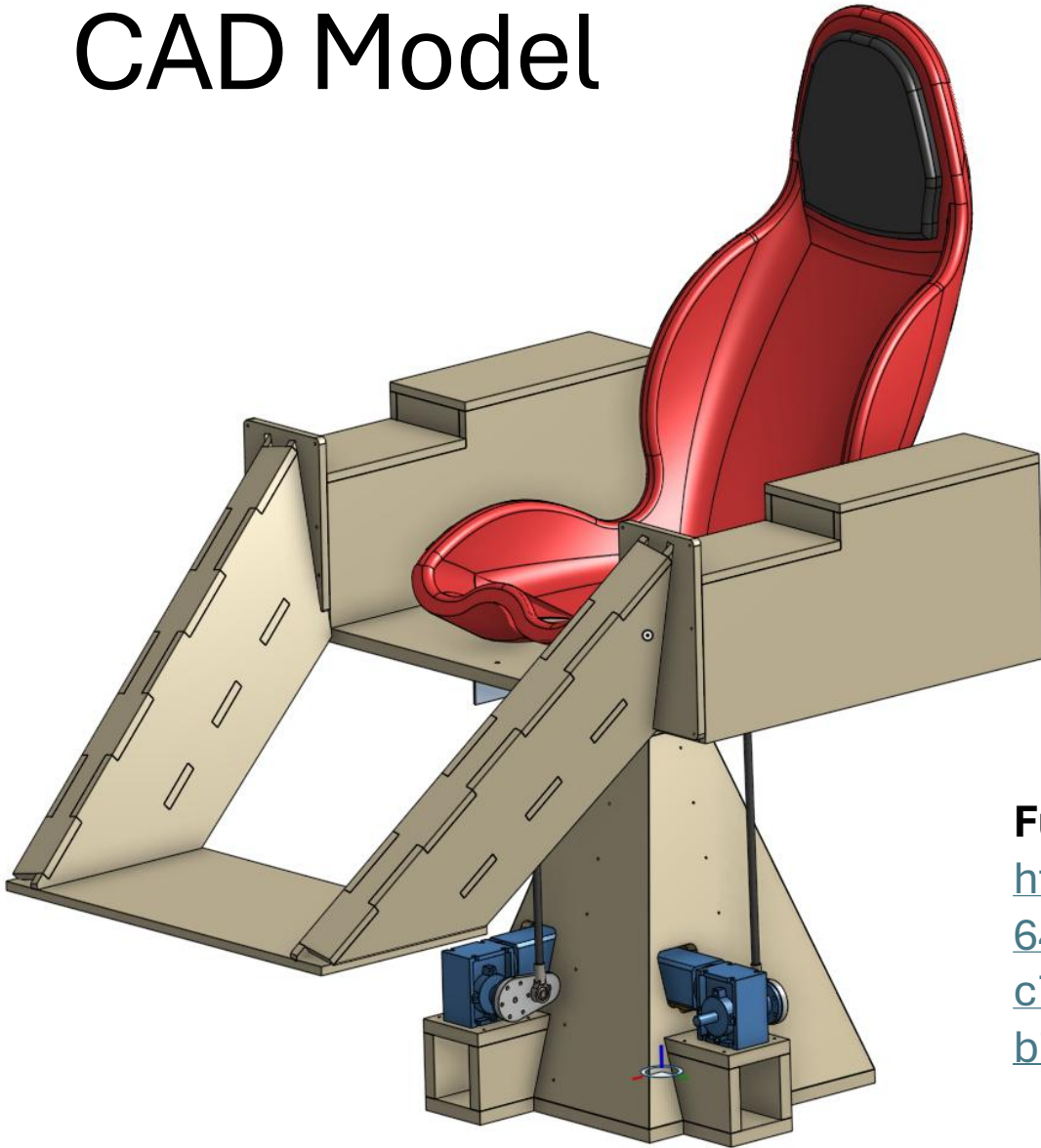
Project Overview

- This project is a great combination of woodworking, and mechatronics which allows you to build a motion simulator capable of flying both RC planes as well as simulator games at a very low price compared to commercial options.
- THIS IS A DIY PROJECT. I have documented my build in this document as well as released the CAD and Code associated to make replicating this build as easy as possible. However, part availability in your local area or different applications may warrant changes to the design for your use.

Disclaimer: Due to the unique nature and variability of each individual's project, I am unable to provide personalized assistance tailored to specific builds. The information and guidance provided are general in nature and may not address the specific requirements or conditions of your project.

I am not responsible or liable for any issues, damages, or outcomes that occur during or as a result of your build. It is your responsibility to ensure that all necessary precautions, safety measures, and appropriate practices are followed. Please consult a qualified professional if you require personalized advice or support for your project.

CAD Model



Full CAD:

<https://cad.onshape.com/documents/14465a5308115755a641bf91/w/473302792caf9fe5237c9eca/e/42b5f99c290e18c70798f158?renderMode=0&uiState=67888ffc05482c0eb0b74d11>

Tools Required

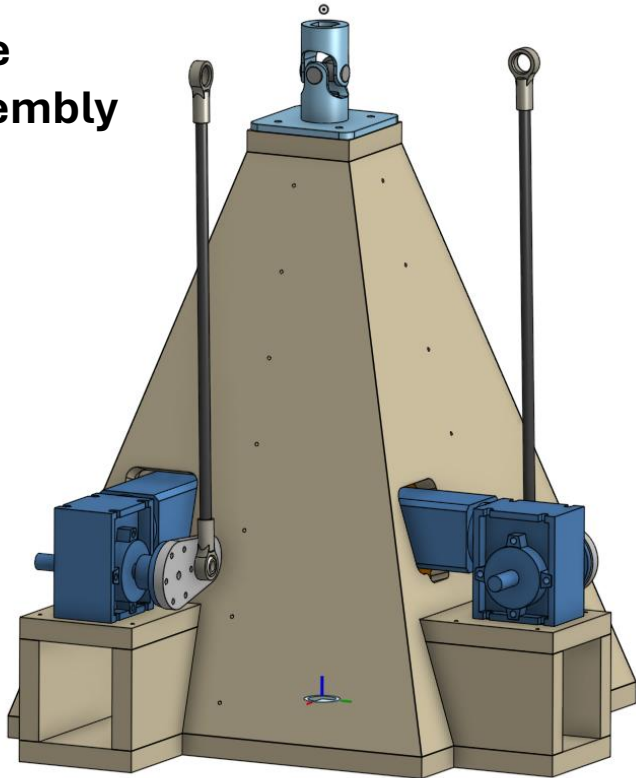
- A large portion of this project involves cutting shapes out of 0.5” (12.5mm) plywood and 0.75” (19mm) plywood. Many different saw types including circular saw, jigsaw, bandsaw can create the cuts needed in the project. A CNC router can also be an ideal choice if available.
- Tools required:
 - Some type of saw (circular saw, jig saw, table saw, band saw or CNC router with 4’x2’ or greater cut area)
 - Power drill, impact driver
 - Angle grinder or hack saw
 - Soldering iron
 - 3D printer

Mechanical Assembly

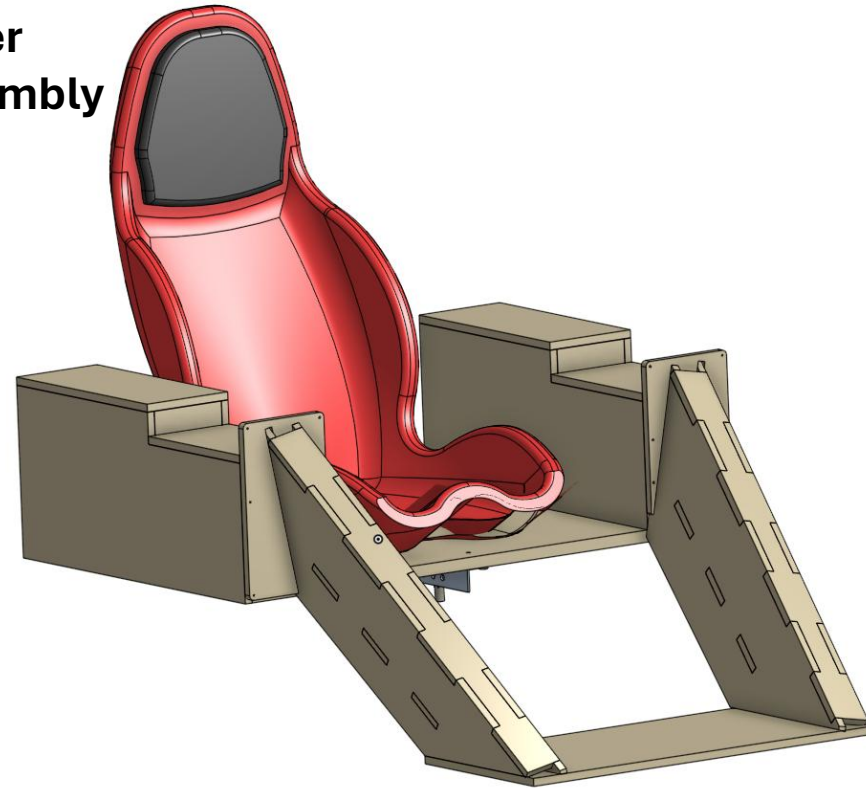
Parts Overview

- This chair breaks down into two main parts for easier transport. These parts are built separately and then joined together using universal joint.

**Base
Assembly**

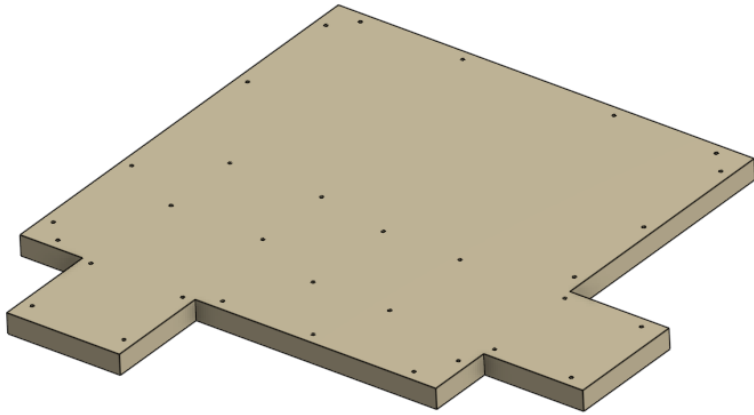


**Upper
Assembly**

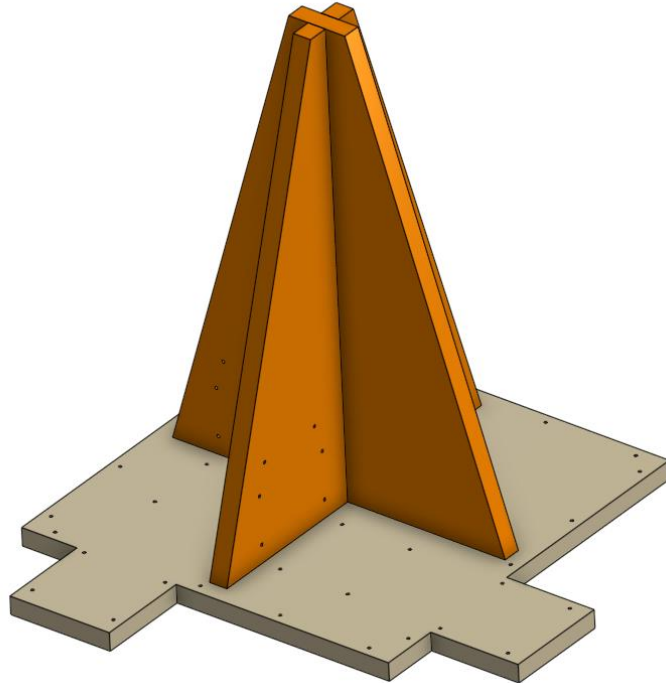


Base Assembly Structure

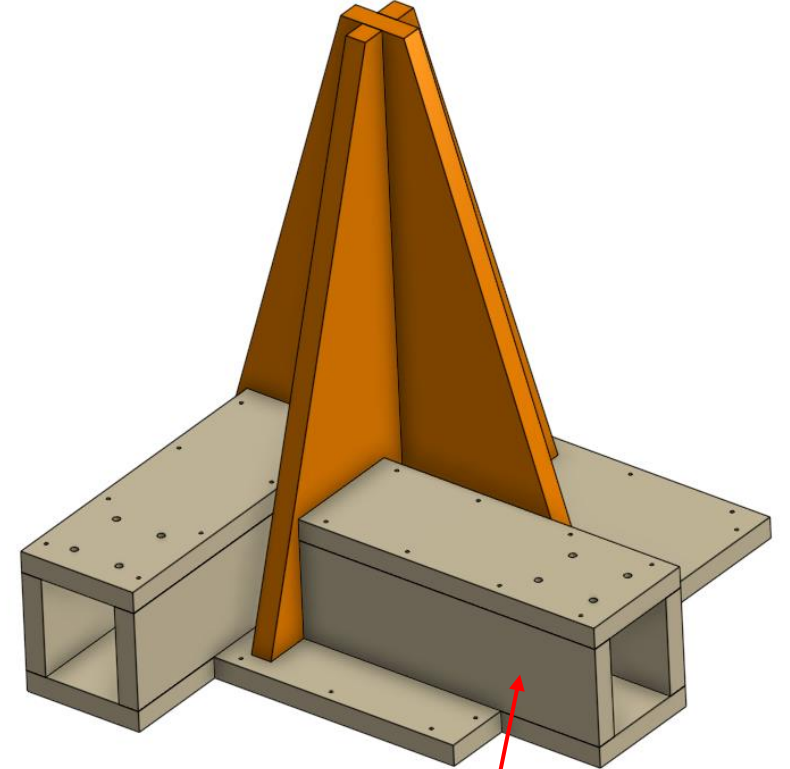
1. Start with base plate



2. Attach middle support structure using slot it interlock. Attach to base using screws and glue



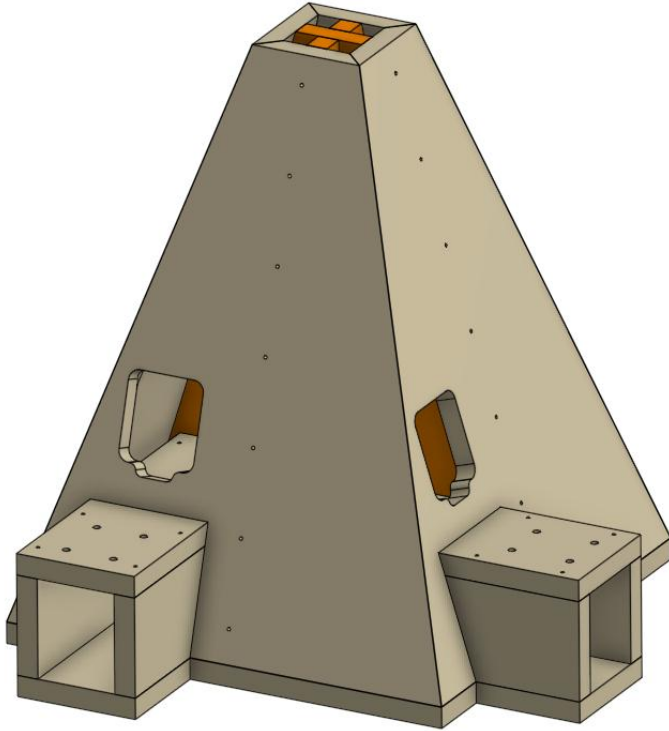
3. Attach motor mounts to the base plates and internal structure using screws and glue



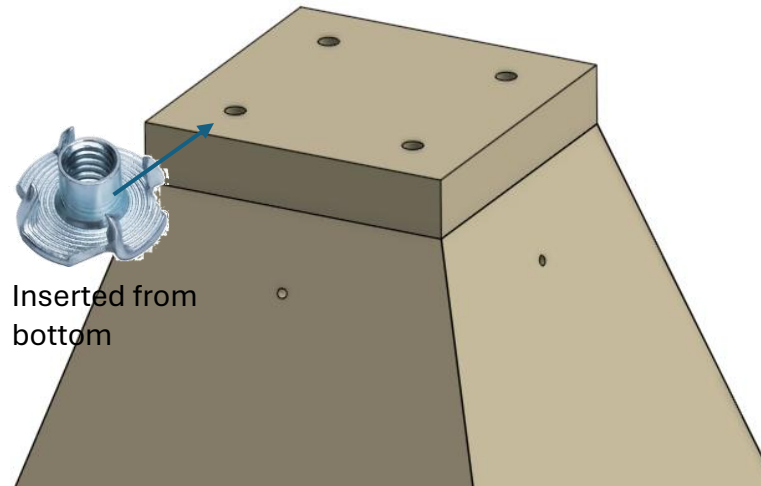
Attach this one first

Base Assembly Structure

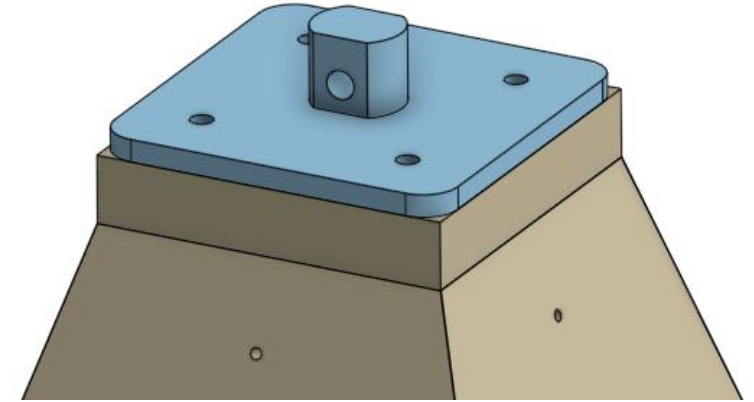
4. Attach outer panels. Perform multiple dry fits since this piece has multiple compound angles. Attach using screws and glue



5. Attach top plate. **Ensure that the holes for the joint mounting plate are already drilled and 0.25"x20 threaded inserts are already installed.** Attach with screws and glue

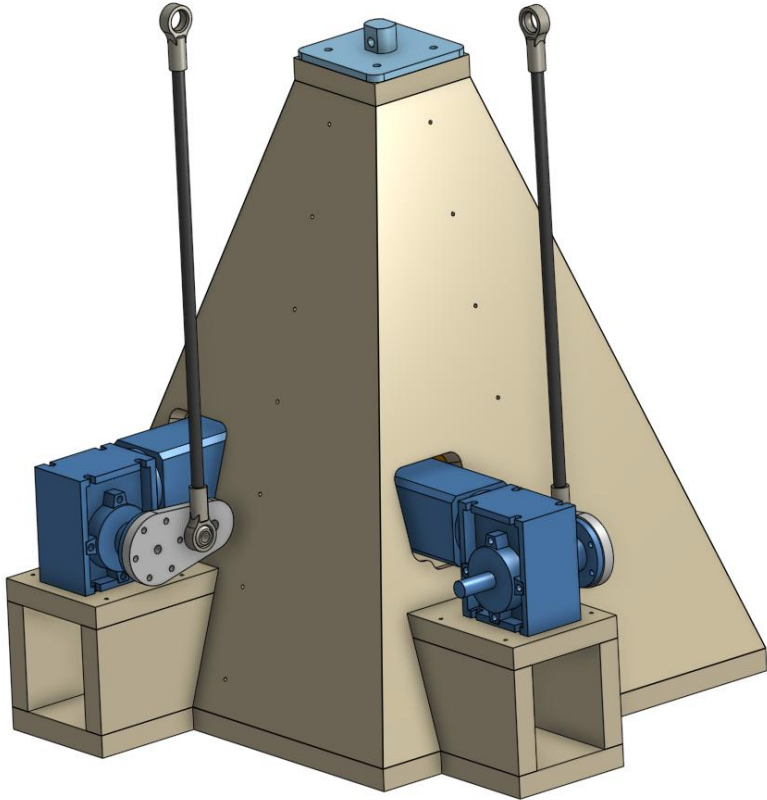


6. Attach joint mounting plate using 0.25"x20 bolts. Joint mounting plate should ideally be machined from aluminum or steel

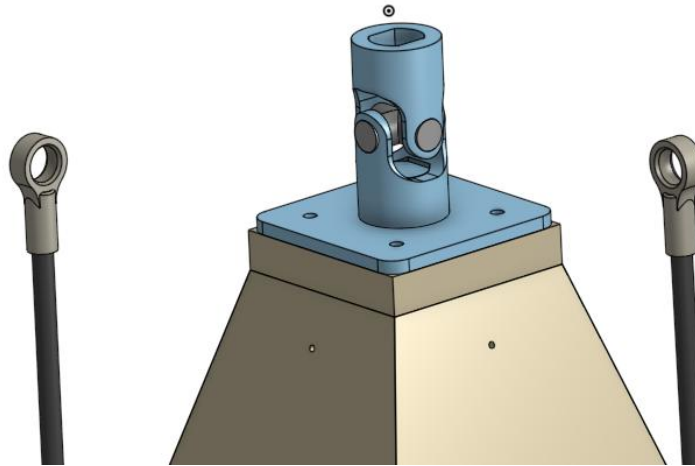


Base Assembly Structure

7. Attach motors, motor arms, threaded rod, heim joints

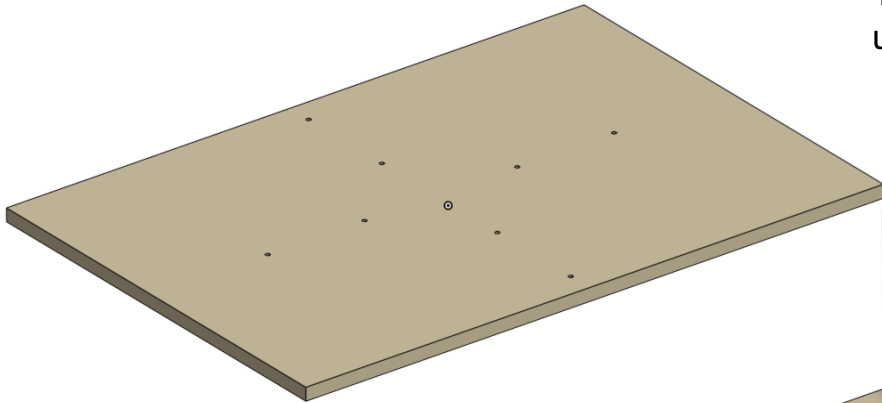


8. Attach universal joint

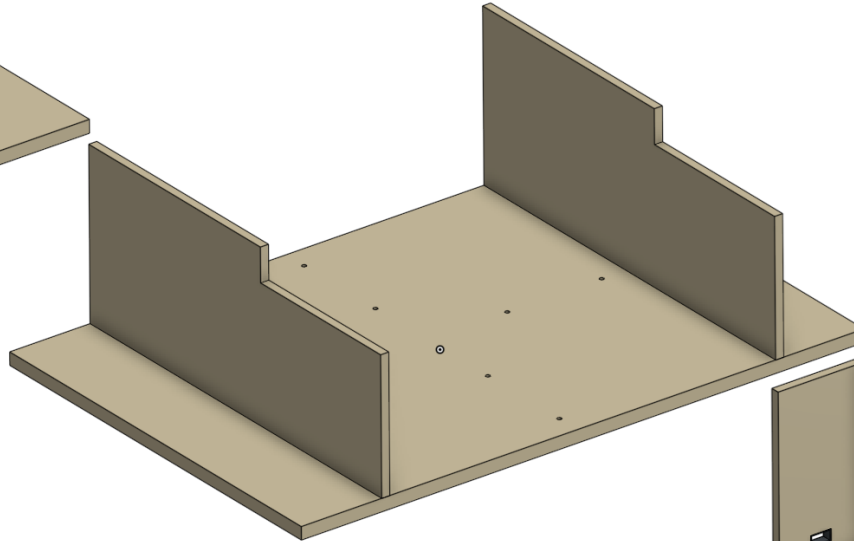


Chair Assembly Structure

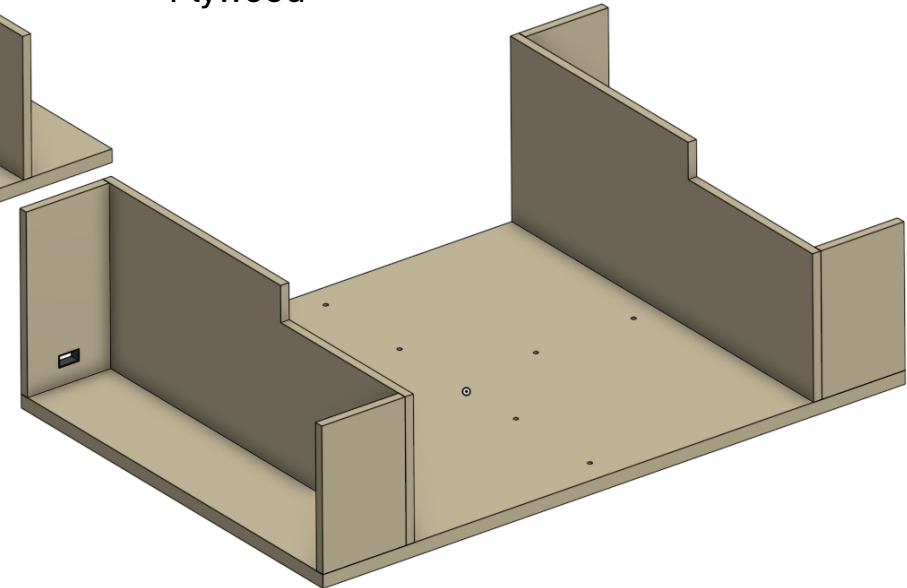
9. Start with baseplate. 0.75" Plywood



10. Add internal side panels and attach using screws and glue. 0.5" Plywood

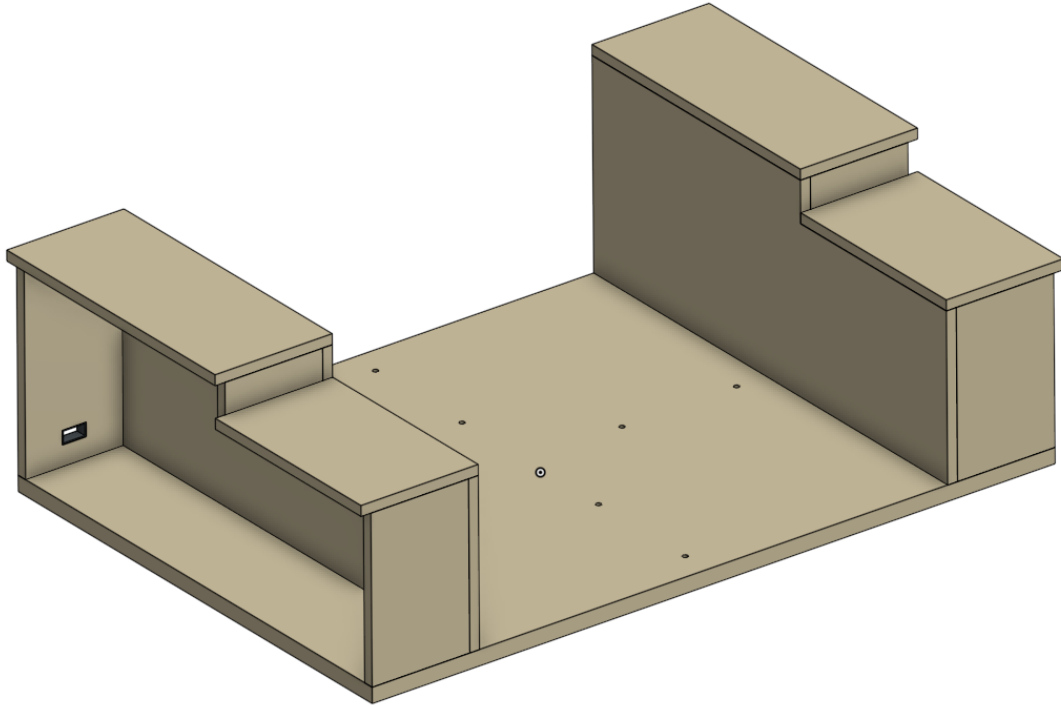


11. Add front and back panels and attach using screws and glue. 0.5" Plywood

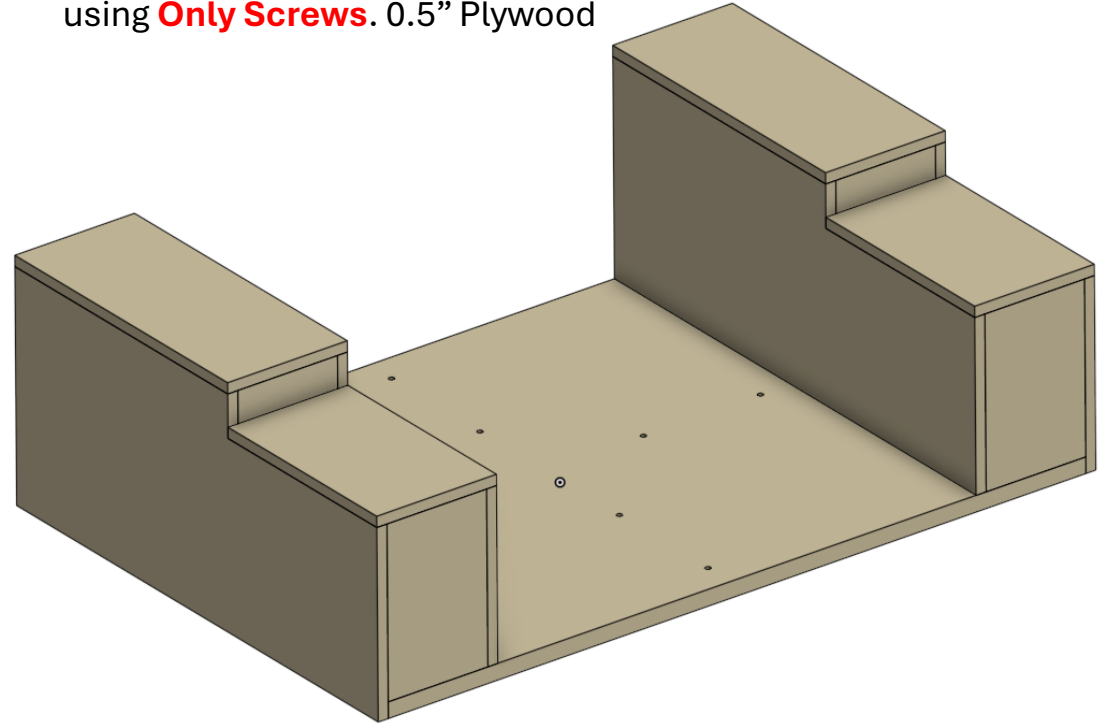


Chair Assembly Structure

12. Add top panels and attach using screws and glue. 0.5" Plywood

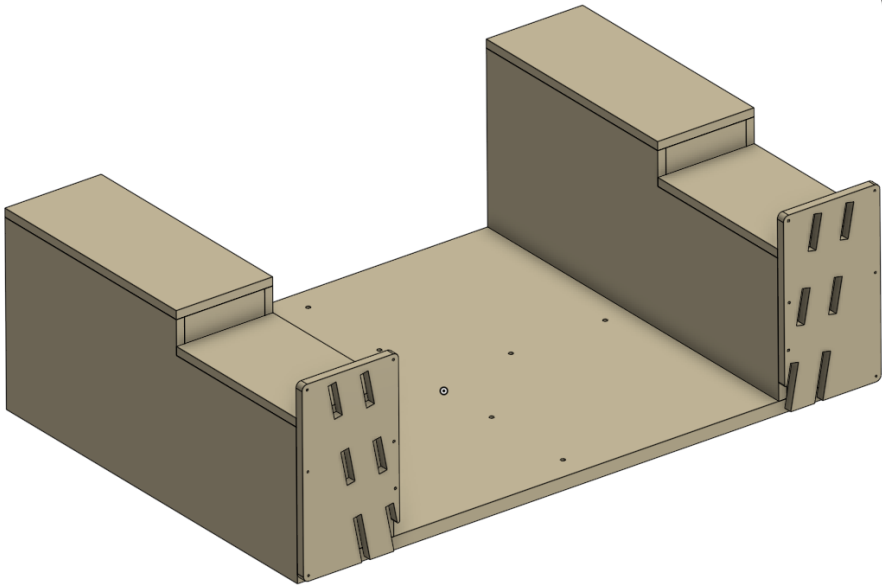


13. Add outer side panels and attach using **Only Screws**. 0.5" Plywood

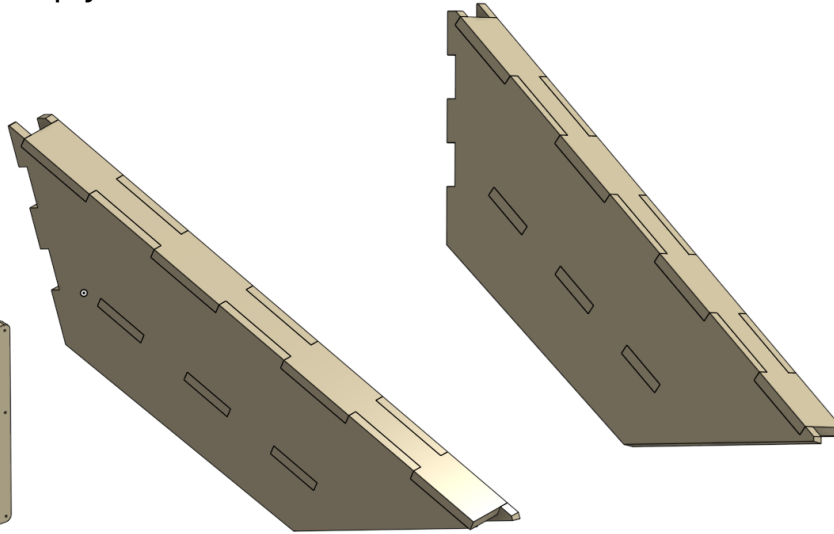


Chair Assembly Structure

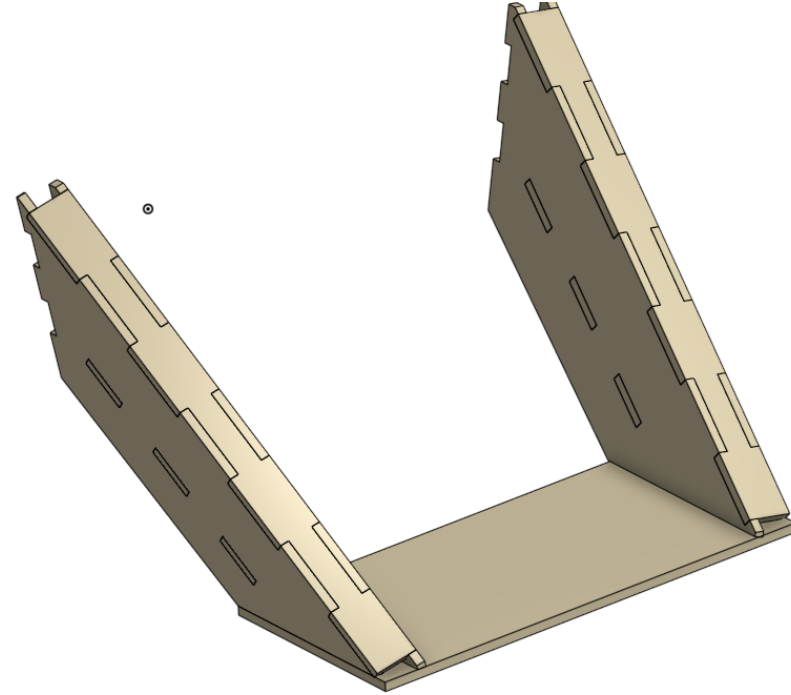
14. Add foot support attachment plates to assembly using **Only Screws**



15. Assemble foot support side structures as shown in CAD assembly.
0.5" plywood

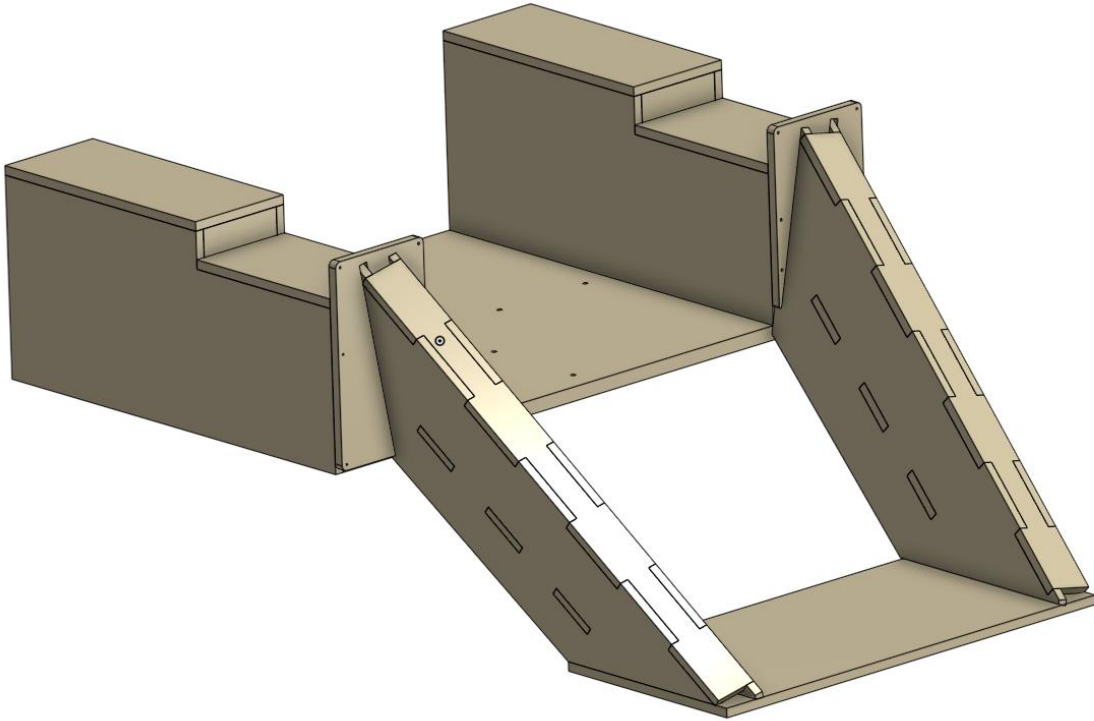


16. Add foot support plate. May need to be customized for your specific controls

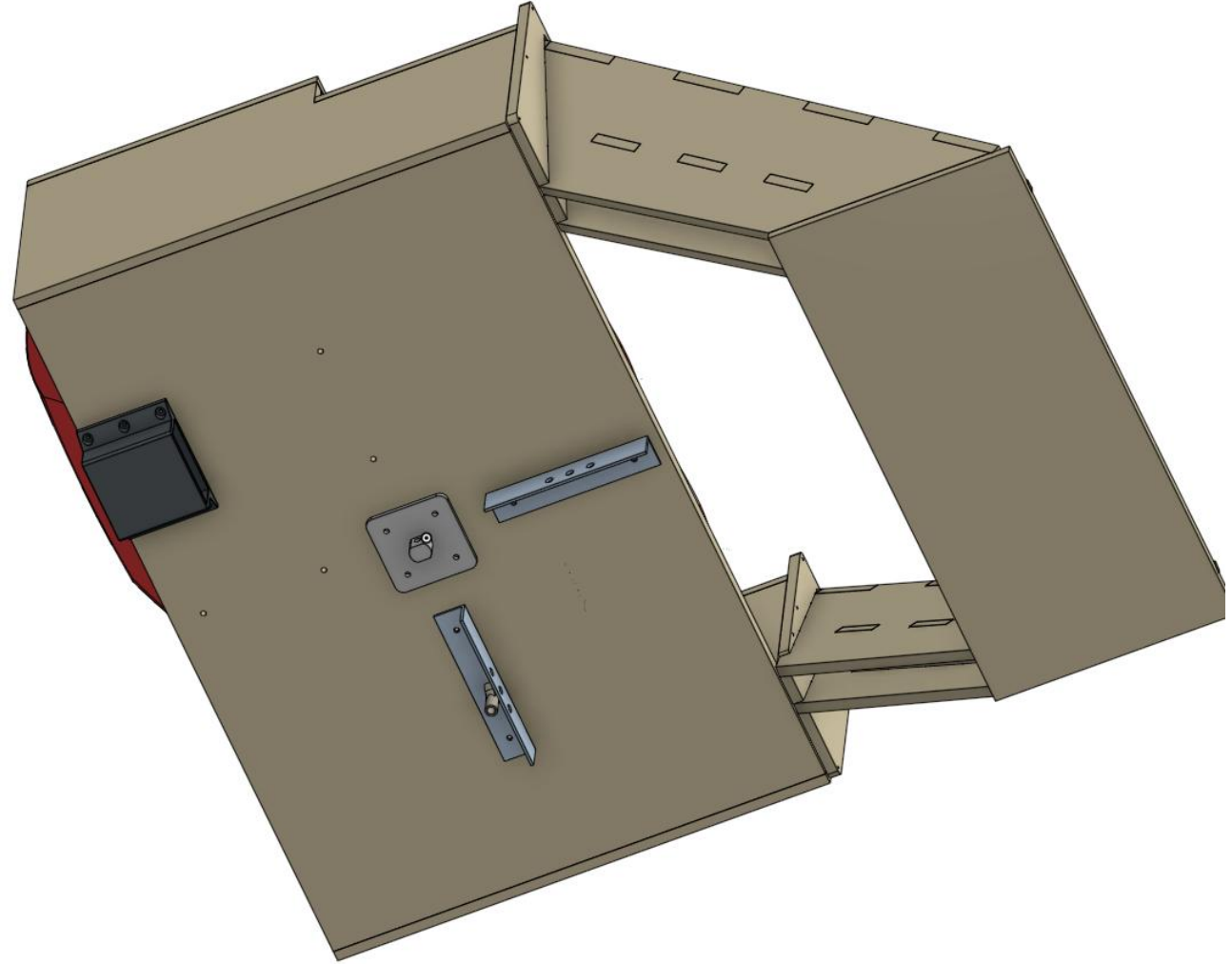


Chair Assembly Structure

17. Attach to foot support attachment place using glue

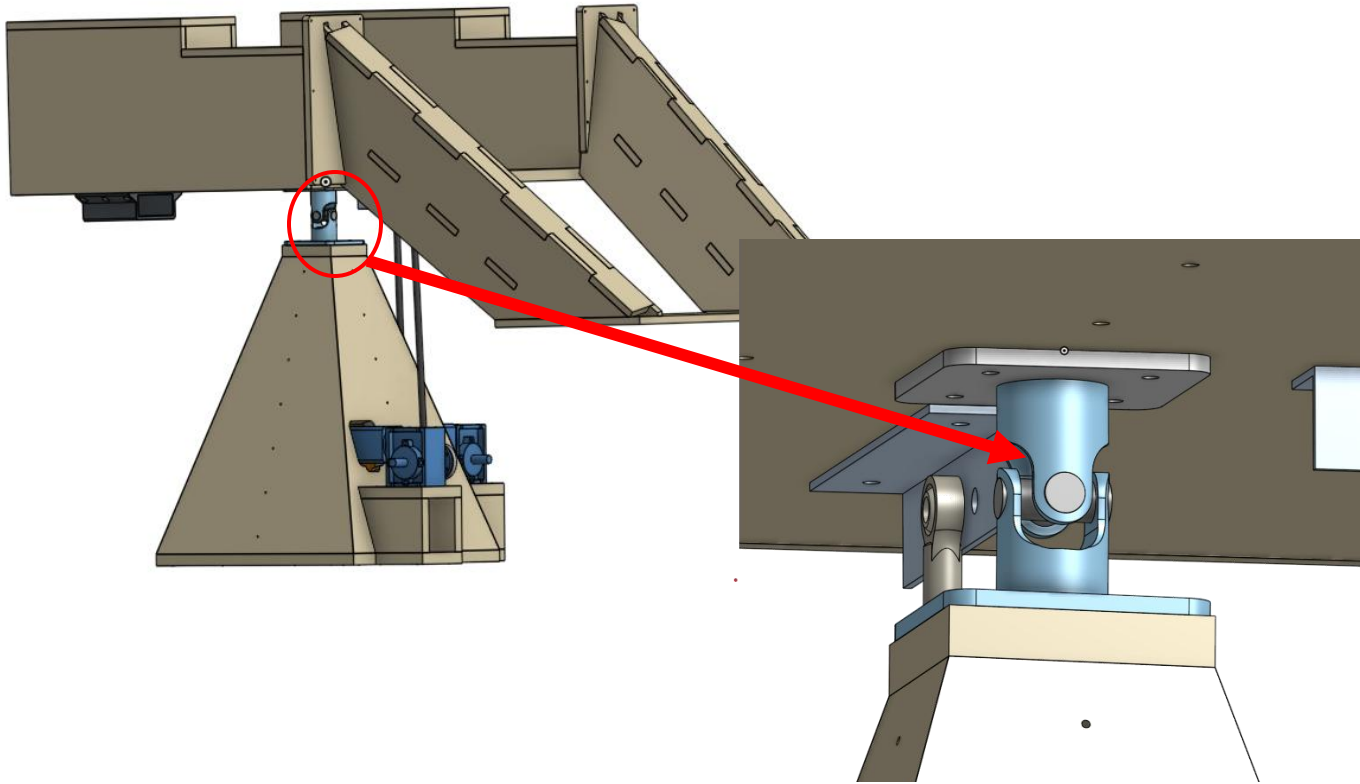


18. Add bottom auxiliary parts: Angle brackets and universal joint mounts

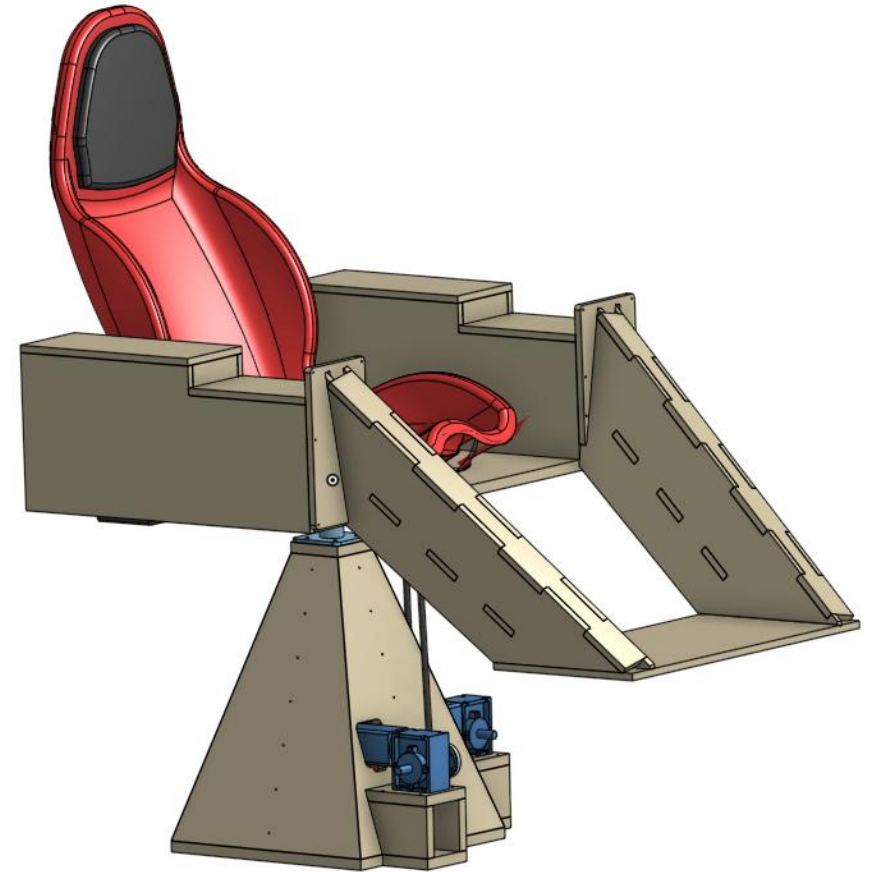


Base / Chair Attachment

19. Attach upper and lower assemblies using universal joint

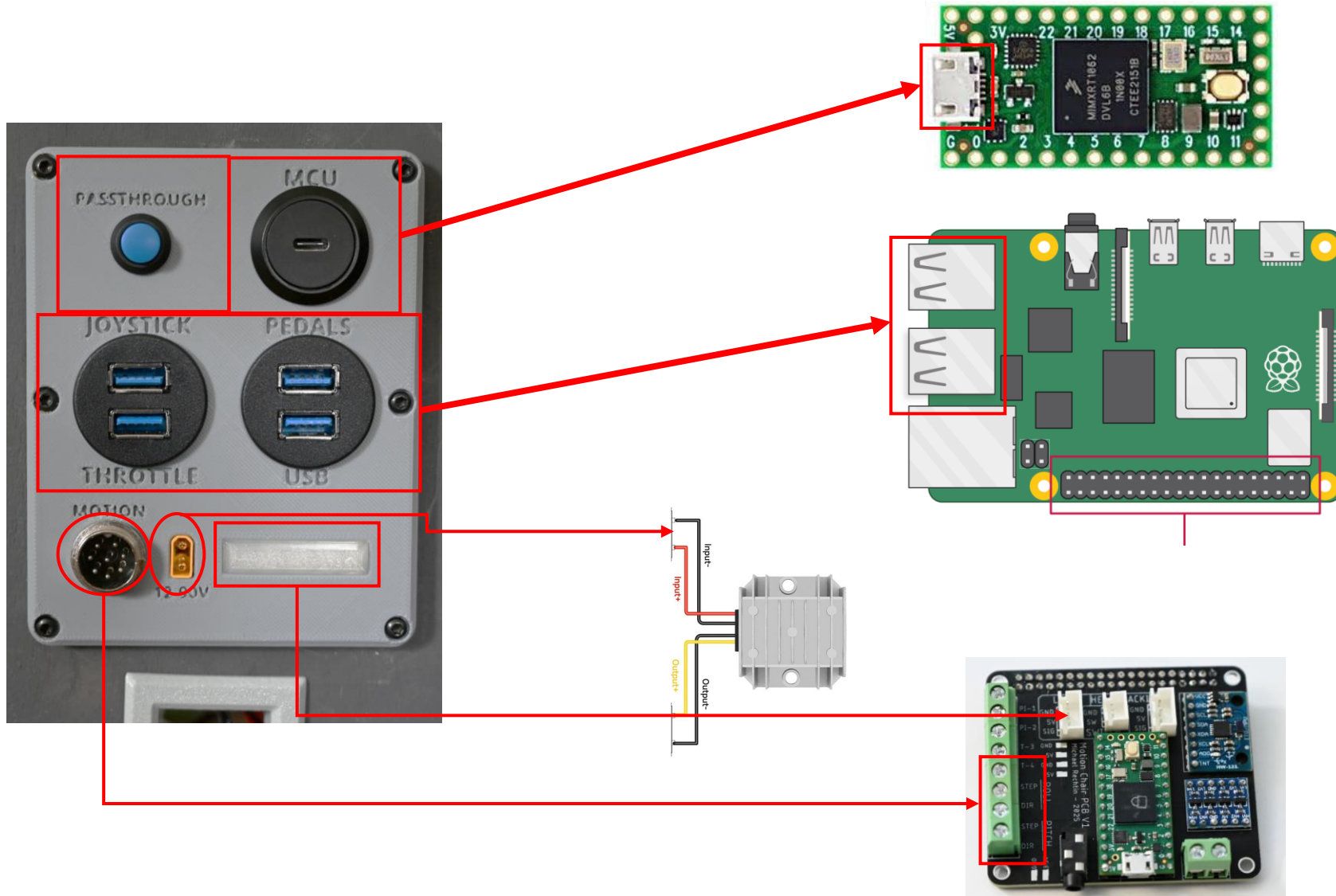


20. Add chair



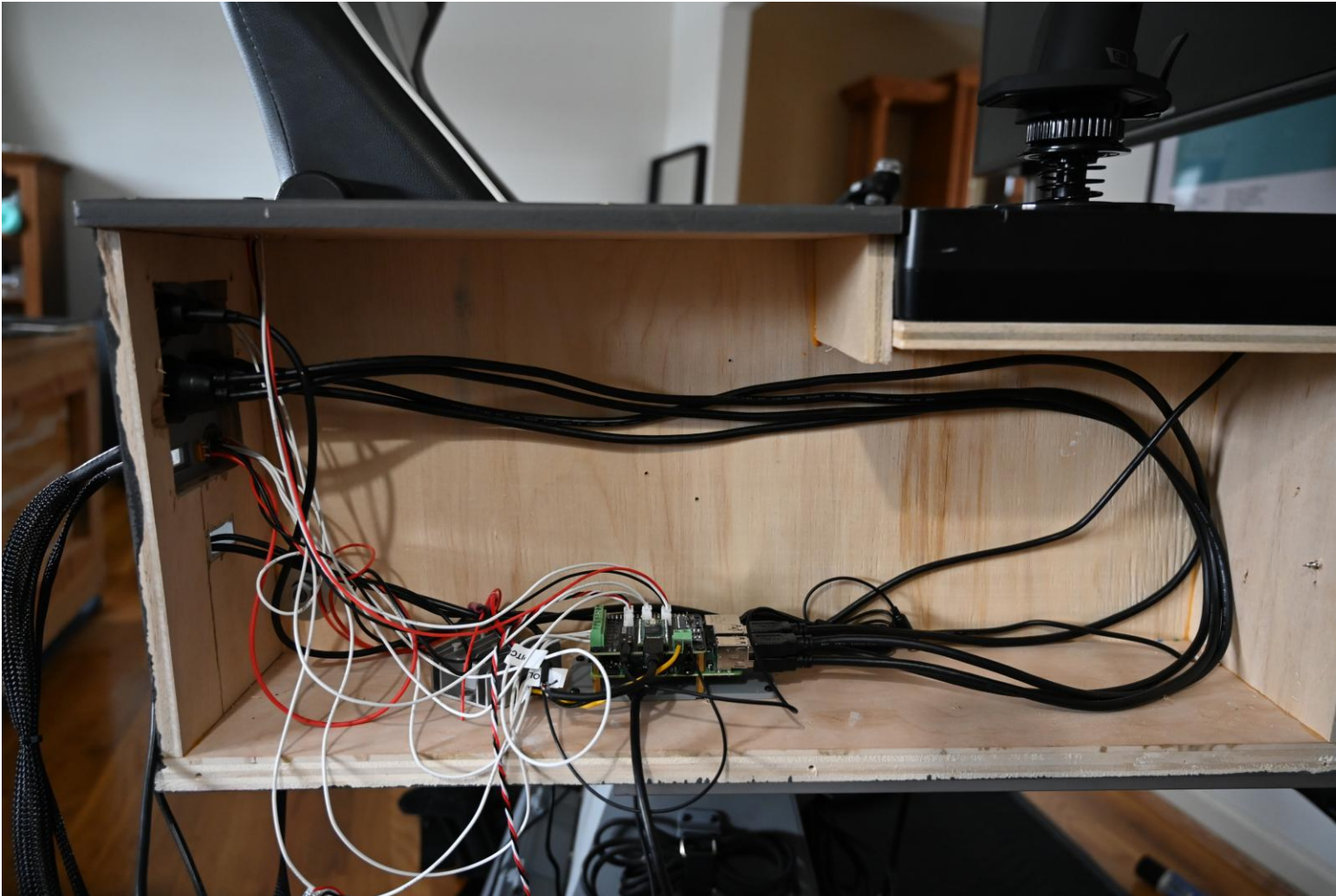
Electronics

Back Panel Wiring



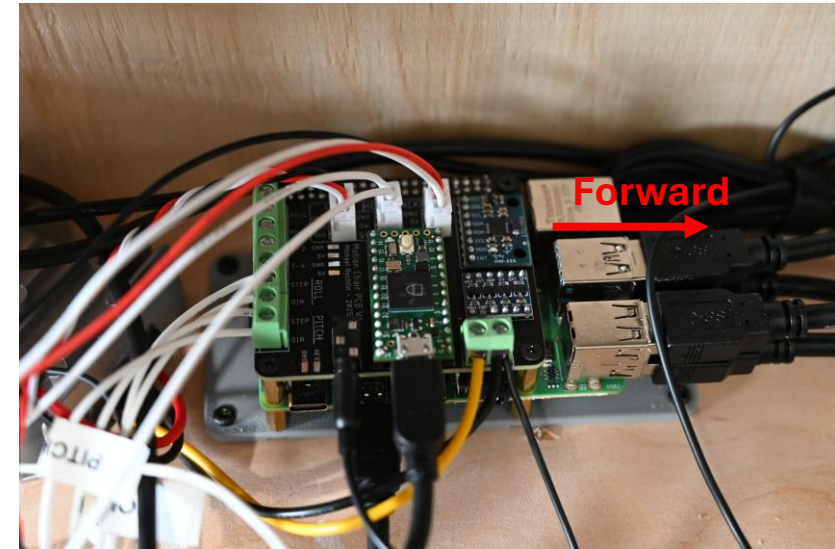
Side Compartment Wiring

- All wiring is housed in the right-side armrest. Wire management is optional!



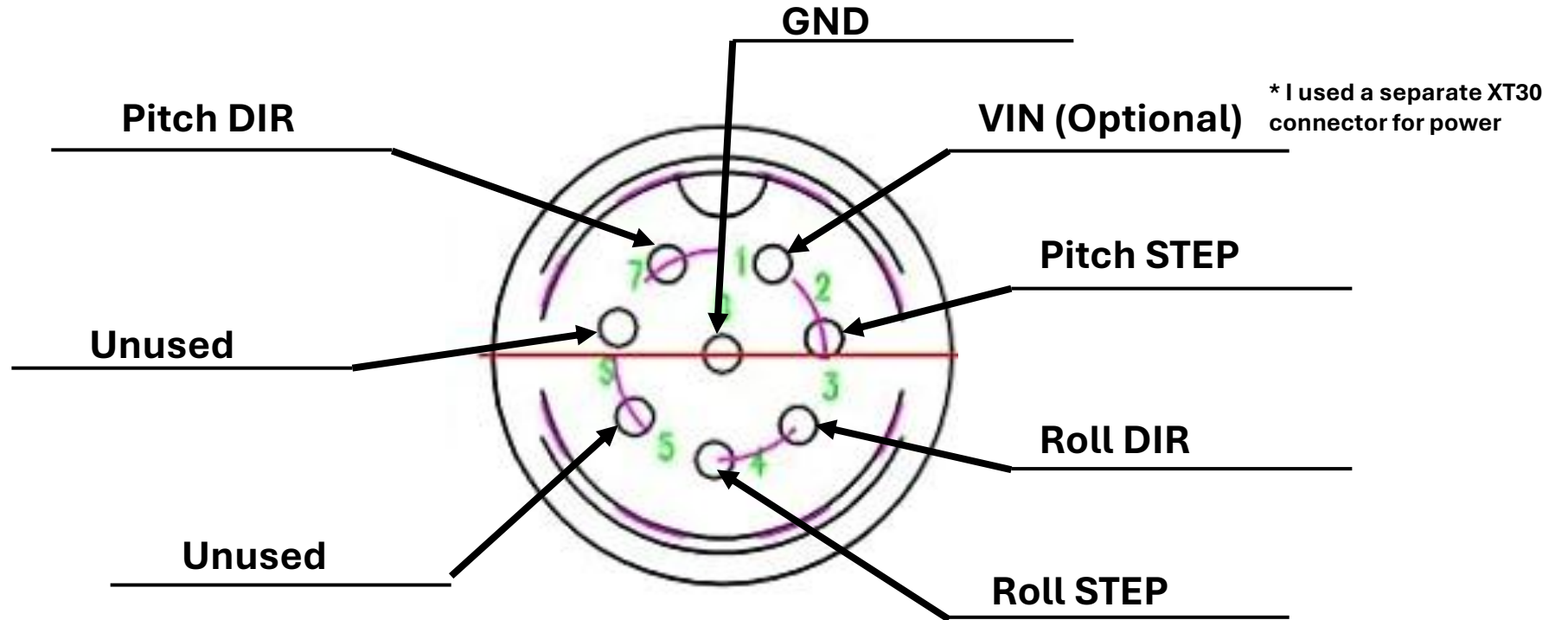
IMPORTANT

MPU 6050 must be in the orientation shown below for IMU based chair calibration.

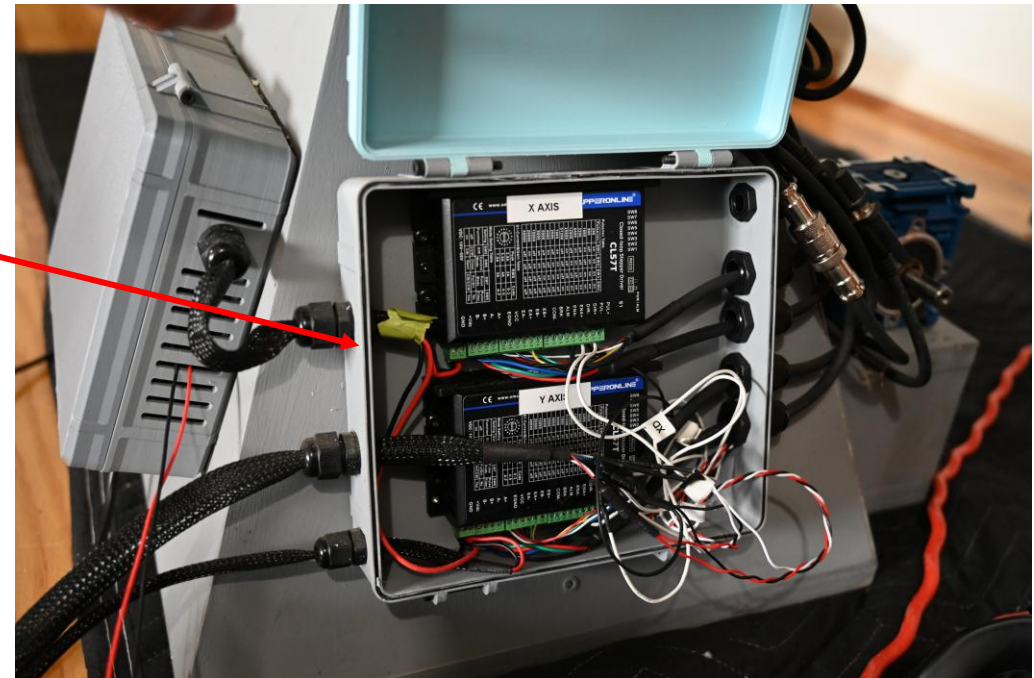
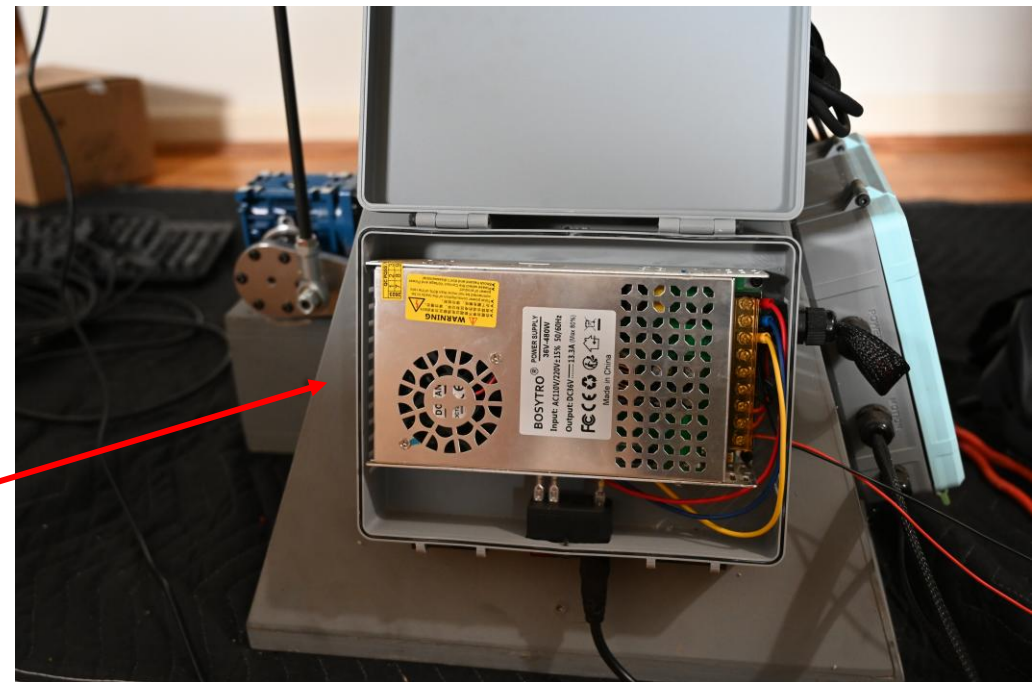


Back Panel Connector Pinout

- Adding connectors to the harness from the base to the chair is helpful if the chair needs to be disassembled for moving.
- Pinout can be customized but **MUST** be consistent between the base and chair to create proper connections



Motion Base Wiring

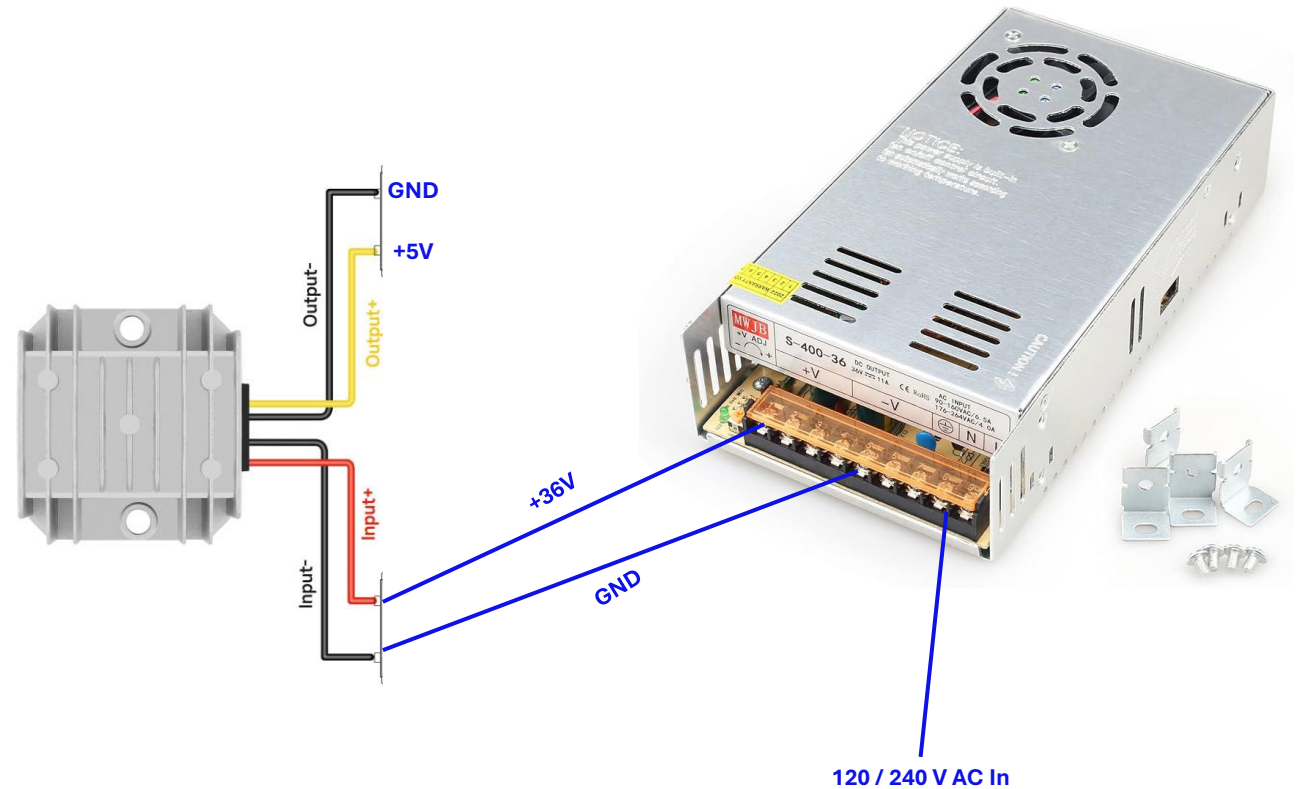
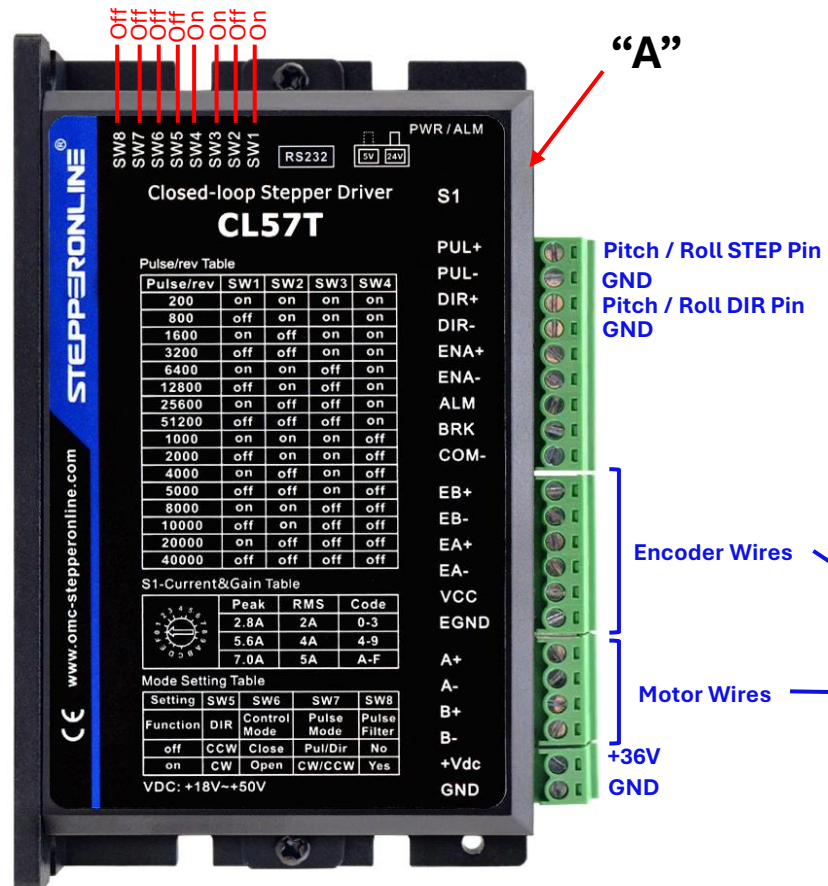


Motion Base Wiring

CL57T Settings using Switches:

Current: "A" 7.0 Amp

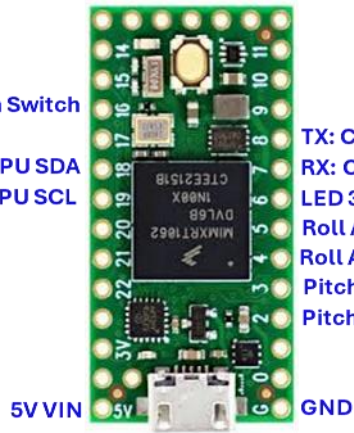
Pulse/Rev: 1600



Stepper Motor / Driver wiring details: <https://www.omc-stepperonline.com/closed-loop-stepper-driver-v4-1-0-8-0a-24-48vdc-for-nema-17-23-24-stepper-motor-cl57t-v41>

Wiring Pinouts

Passthrough Switch
SDA: Connect to MPU SDA
SCL: Connect to MPU SCL



TX: Connect to PI GPIO 15
RX: Connect to PI GPIO 14
LED 3.3V Signal
Roll Axis: DIR
Roll Axis: STEP
Pitch Axis: DIR
Pitch Axis: STEP

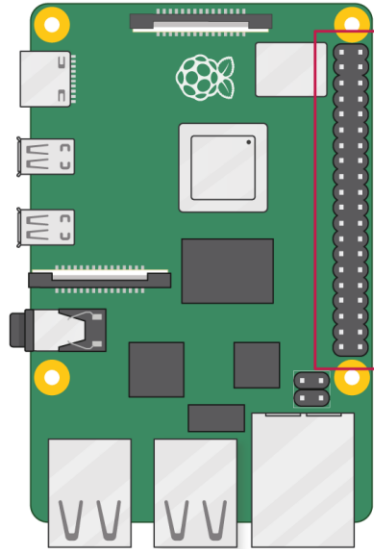
LED 3.3V Signal

3.3V
GND



LED 5V Signal

5V
GND



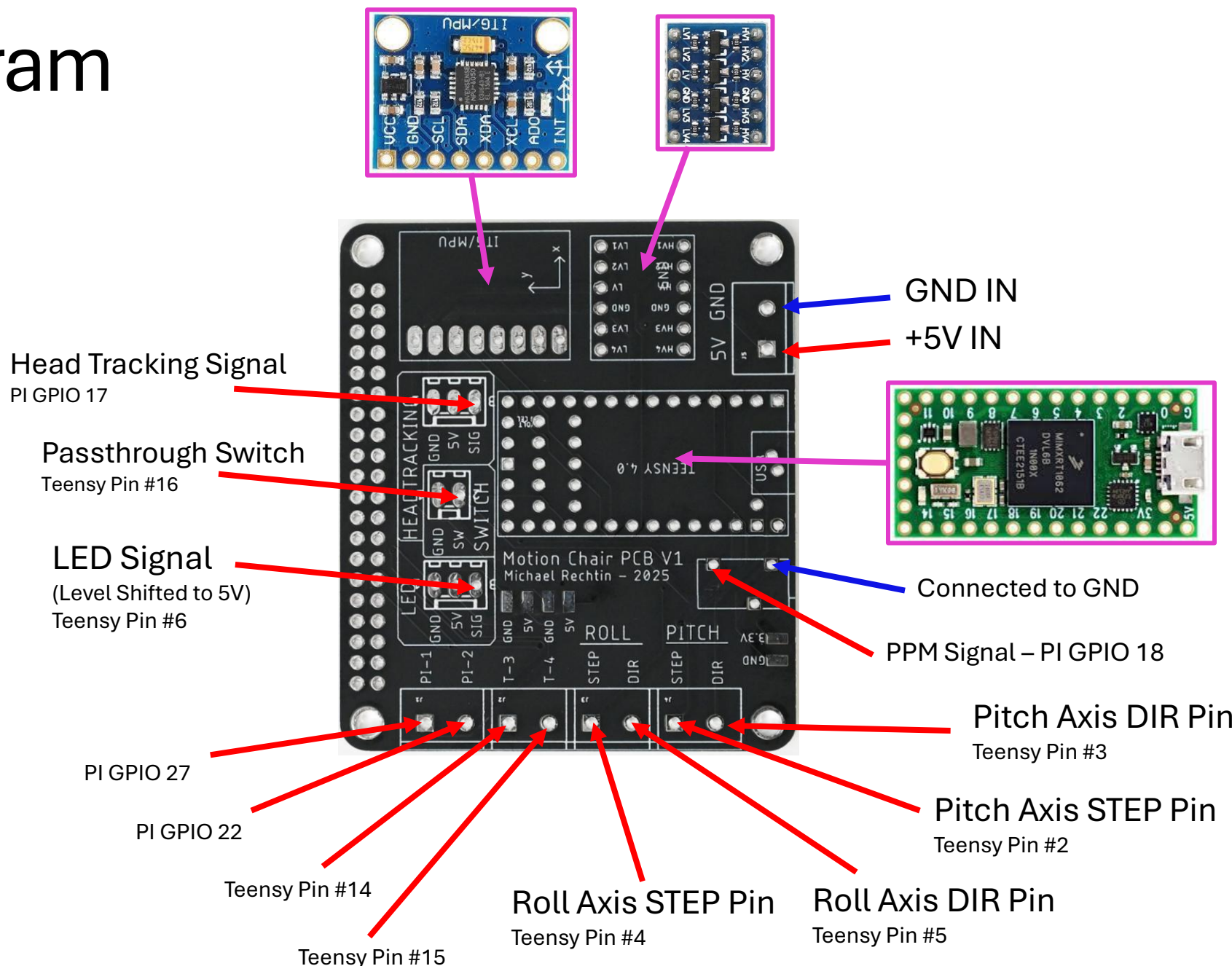
Headtracking PPM in

3.3V
GND
SCL: Connect to Teensy Pin 19
SDA: Connect to Teensy Pin 18



3V3 power	1	2	5V power
GPIO 2 (SDA)	3	4	5V power
GPIO 3 (SCL)	5	6	Ground
GPIO 4 (GPCLK0)	7	8	GPIO 14 (TXD) Connect to Teensy Pin 7
Ground	9	10	GPIO 15 (RXD) Connect to Teensy Pin 8
GPIO 17	11	12	GPIO 18 (PCM_CLK) RC PPM Out
GPIO 27	13	14	Ground
GPIO 22	15	16	GPIO 23
3V3 power	17	18	GPIO 24
GPIO 10 (MOSI)	19	20	Ground
GPIO 9 (MISO)	21	22	GPIO 25
GPIO 11 (SCLK)	23	24	GPIO 8 (CE0)
Ground	25	26	GPIO 7 (CE1)
GPIO 0 (ID_SD)	27	28	GPIO 1 (ID_SC)
GPIO 5	29	30	Ground
GPIO 6	31	32	GPIO 12 (PWM0)
GPIO 13 (PWM1)	33	34	Ground
GPIO 19 (PCM_FS)	35	36	GPIO 16
GPIO 26	37	38	GPIO 20 (PCM_DIN)
Ground	39	40	GPIO 21 (PCM_DOUT)

PCB Diagram



Flight Controls

- Most USB based controls should work but the ones listed below are options I have tested

Logitech G X56 H.O.T.A.S



<https://amzn.to/4g3T0MK>

Thrustmaster TFRP Rudder Pedals



<https://amzn.to/4hnNeq2>

Software Setup - Chair

All code available at: <https://github.com/MichaelRechtin/MotionSim>

Teensy 4.0 Code

C++ code running on the Teensy must be uploaded using the Arduino IDE. The teensy takes serial inputs from either the the Pi (passthrough mode) or an external computer to control the stepper motors for pitch / roll

IMU_Motion_Control.ino

```
1  #include <AccelStepper.h>
2  #include <Adafruit_NeoPixel.h>
3
4  // Define the stepper motor connection:
5  #define XDIR_PIN 3
6  #define XSTEP_PIN 2
7  #define YDIR_PIN 5
8  #define YSTEP_PIN 4
9  #define PASSTHROUGH_PIN 16
10 #define LED_PIN 6
11 #define NUM_LEDS 5
```

Adjustable parameters
depending on pinout. Default
should work with PCB

```
// Define the maximum speed and acceleration
#define MAX_SPEED 10000.0 // Steps per second
#define ACCELERATION 10000.0 // Steps per second^2
```

Adjustable motion parameters.
If motors skip steps these
values may need to be lowered

```
int x_zero=0;
int y_zero=-4;
```

Adjust the “Zero” position
that the chair calls home

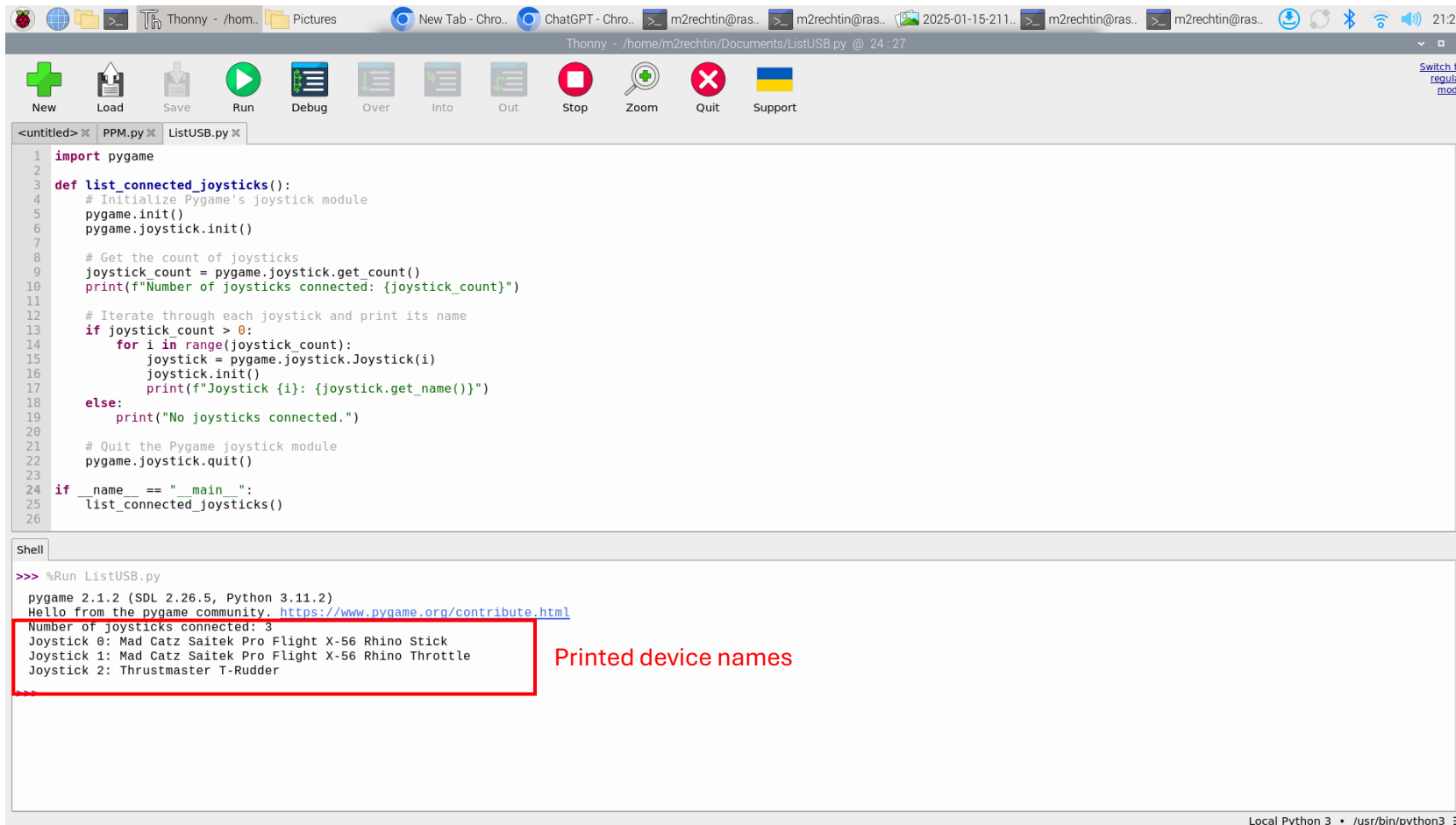
```
int x_pos=0;
int y_pos=0;
int steps_per_rev=3200;
```

Adjust is microstepping
differently

Raspberry Pi: Identify Flight Controls Names ListUSB.py

If you are using different USB controls you must first find the names of your USB devices.

1. Connect USB controls to the Raspberry Pi through the back panel or the USB ports directly.
2. Run ListUSB.py and it will print out the names of all detected devices



```
<untitled> PPM.py ListUSB.py
1 import pygame
2
3 def list_connected_joysticks():
4     # Initialize Pygame's joystick module
5     pygame.init()
6     pygame.joystick.init()
7
8     # Get the count of joysticks
9     joystick_count = pygame.joystick.get_count()
10    print(f"Number of joysticks connected: {joystick_count}")
11
12    # Iterate through each joystick and print its name
13    if joystick_count > 0:
14        for i in range(joystick_count):
15            joystick = pygame.joystick.Joystick(i)
16            joystick.init()
17            print(f"Joystick {i}: {joystick.get_name()}")
18    else:
19        print("No joysticks connected.")
20
21    # Quit the Pygame joystick module
22    pygame.joystick.quit()
23
24 if __name__ == "__main__":
25     list_connected_joysticks()
26
```

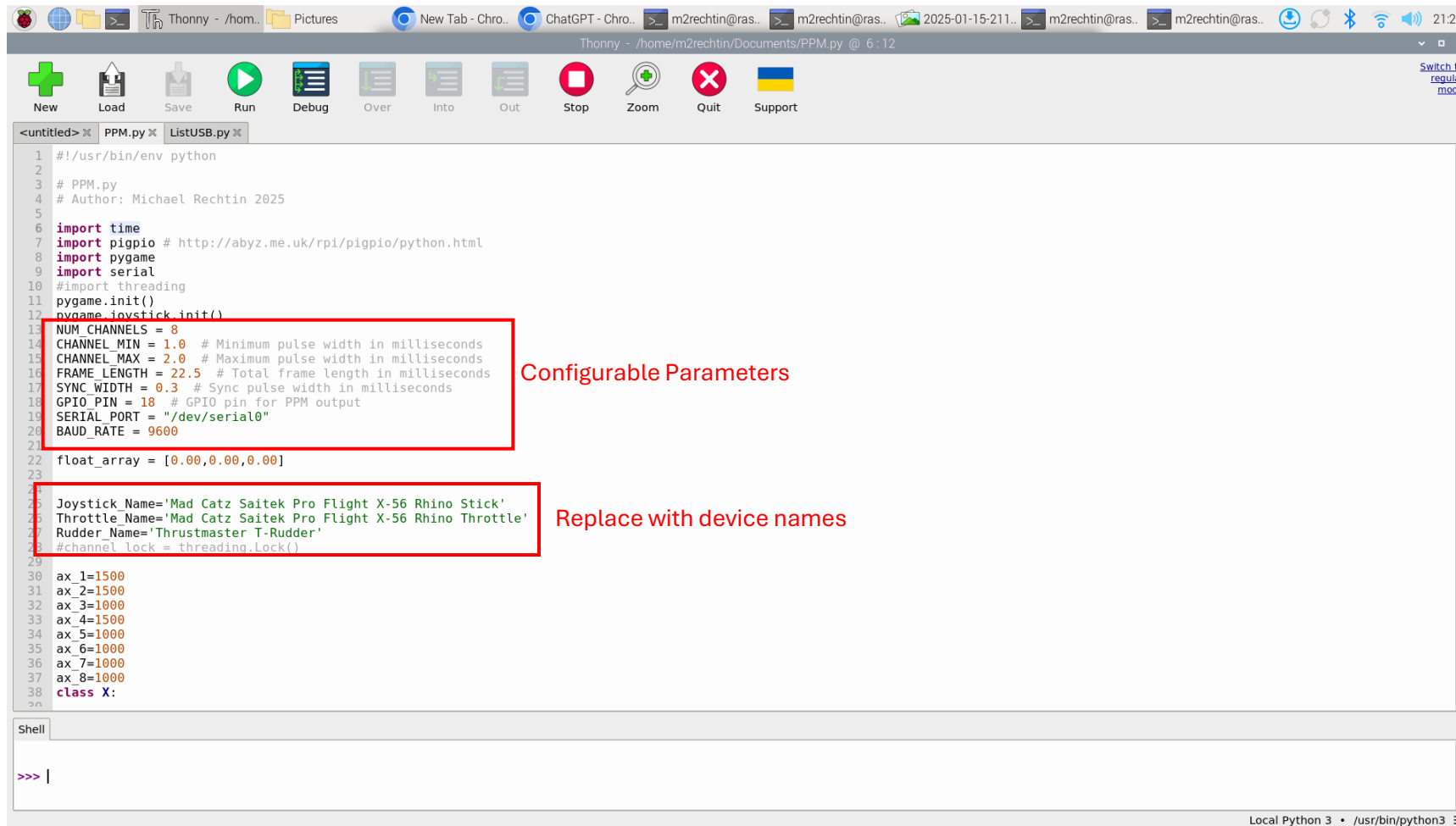
```
Shell
>>> %Run ListUSB.py
pygame 2.1.2 (SDL 2.26.5, Python 3.11.2)
Hello from the pygame community. https://www.pygame.org/contribute.html
Number of joysticks connected: 3
Joystick 0: Mad Catz Saitek Pro Flight X-56 Rhino Stick
Joystick 1: Mad Catz Saitek Pro Flight X-56 Rhino Throttle
Joystick 2: Thrustmaster T-Rudder
>>>
```

Printed device names

Local Python 3 • /usr/bin/python3

Raspberry Pi: Insert Flight Controls Names – PPM.py

1. Open PPM.py
2. Replace device names with the outputs from ListUSB.py
3. Optionally configure any changes needed for RC applications



```
1#!/usr/bin/env python
2
3# PPM.py
4# Author: Michael Rechten 2025
5
6import time
7import pigpio # http://abyz.me.uk/rpi/pigpio/python.html
8import pygame
9import serial
10import threading
11pygame.init()
12pygame.joystick.init()
13NUM_CHANNELS = 8
14CHANNEL_MIN = 1.0 # Minimum pulse width in milliseconds
15CHANNEL_MAX = 2.0 # Maximum pulse width in milliseconds
16FRAME_LENGTH = 22.5 # Total frame length in milliseconds
17SYNC_WIDTH = 0.3 # Sync pulse width in milliseconds
18GPIO_PIN = 18 # GPIO pin for PPM output
19SERIAL_PORT = "/dev/serial0"
20BAUD_RATE = 9600
21
22float_array = [0.00,0.00,0.00]
23
24Joystick_Name='Mad Catz Saitek Pro Flight X-56 Rhino Stick'
25Throttle_Name='Mad Catz Saitek Pro Flight X-56 Rhino Throttle'
26Rudder_Name='Thrustmaster T-Rudder'
27#channel lock = threading.Lock()
28
29
30ax_1=1500
31ax_2=1500
32ax_3=1000
33ax_4=1500
34ax_5=1000
35ax_6=1000
36ax_7=1000
37ax_8=1000
38class X:
```

Configurable Parameters

Replace with device names

Shell

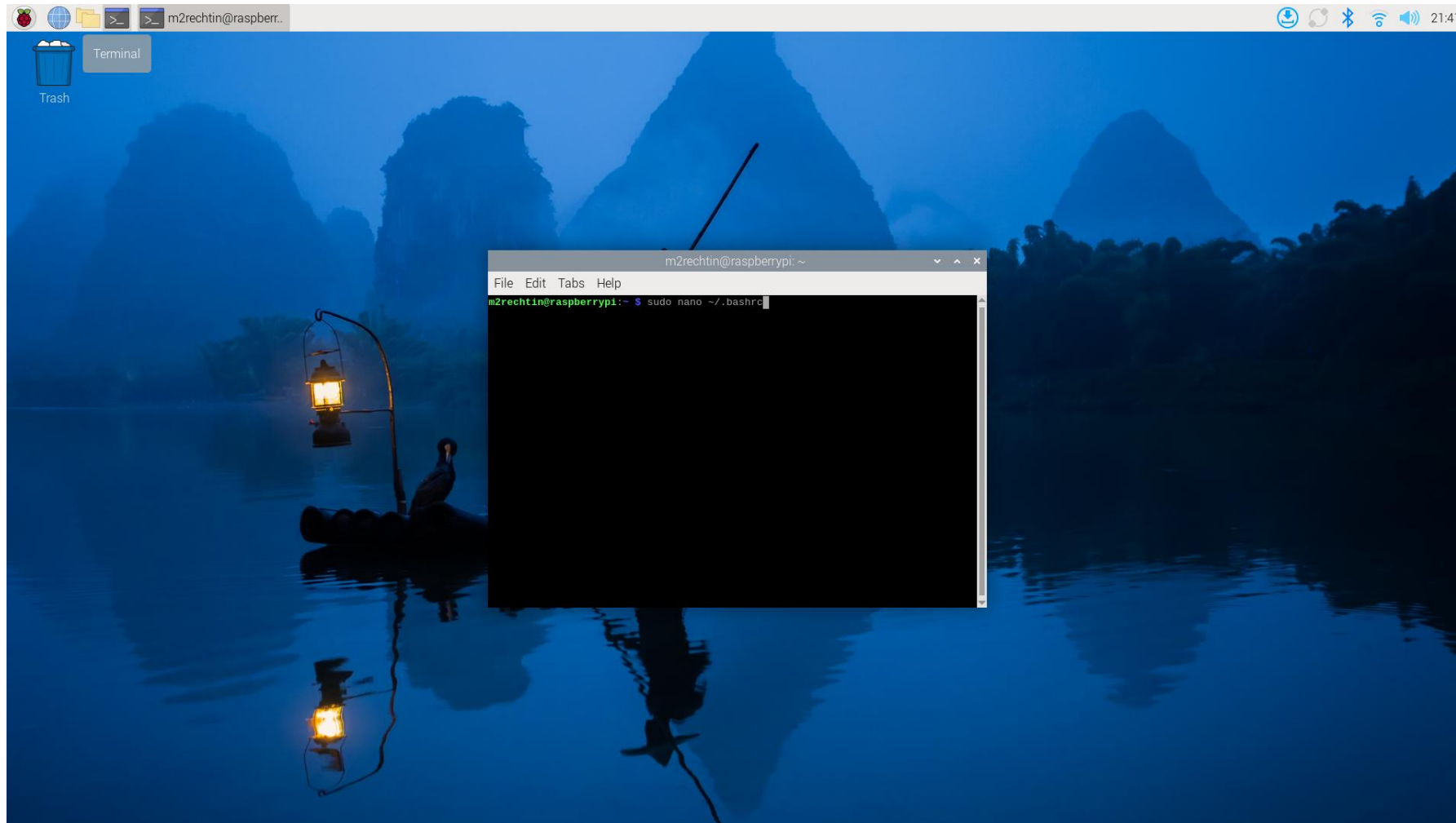
>>> |

Local Python 3 • /usr/bin/python3

Raspberry Pi: Auto-run script

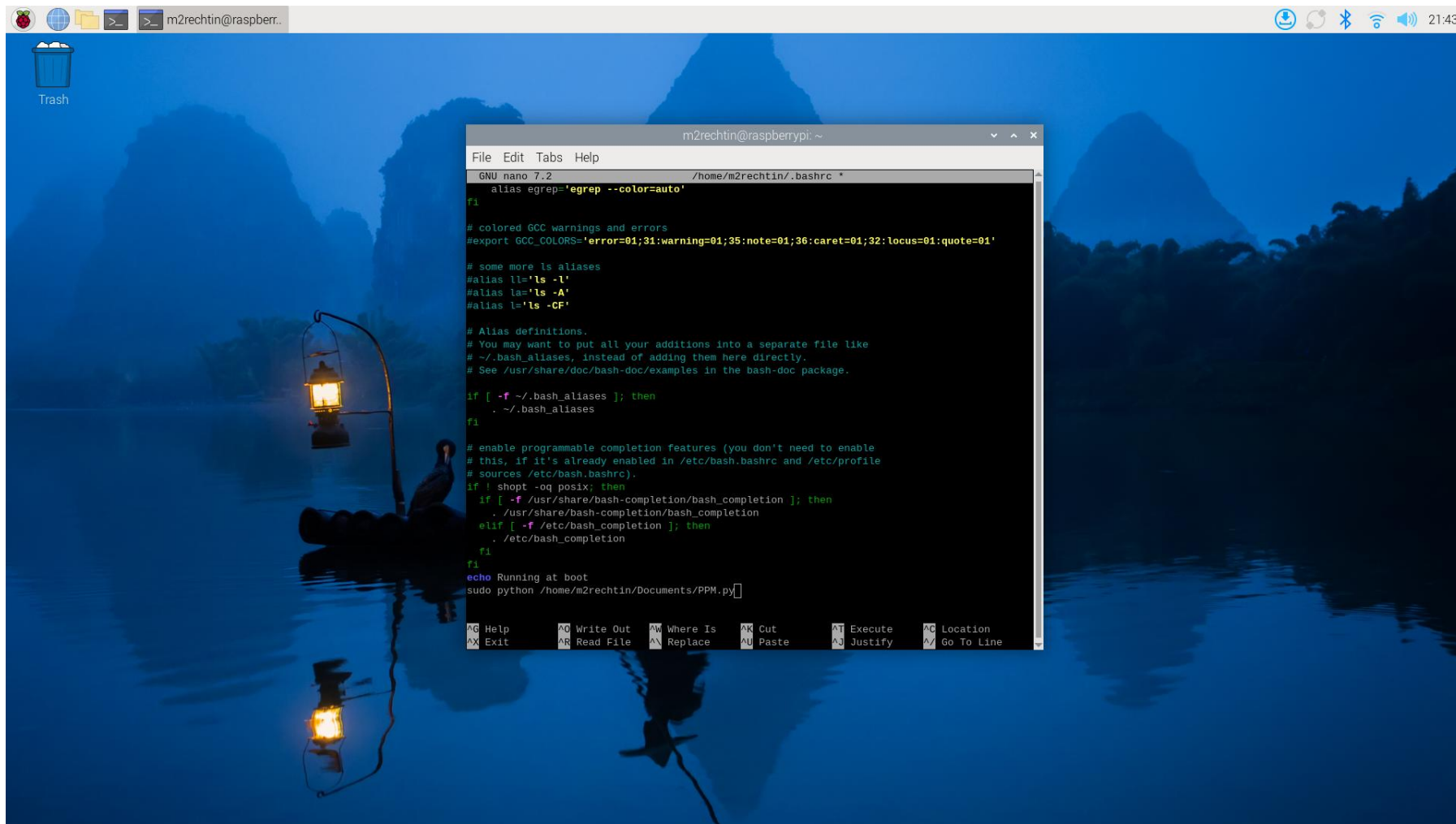
To operate properly, PPM.py must autorun when Pi is powered on.

1. Open terminal
2. Open bashrc file (using command below)



Raspberry Pi: Auto-run script

1. Add lines to bottom of file. As shown below
2. Ctrl+X and then Y to save and exit



Software Setup - PC

All code available at: <https://github.com/MichaelRechtin/MotionSim>

Overview

The chair receives movement commands from an external PC. These movements can come from a simulation game (MSFS 2020, MSFS 2024, WarThunder currently supported) or from an RC plane (Ardupilot)

1. Connect PC to chair using USB Cable
2. Run python script



Connect to PC

If using Chair for “Passthrough Mode” or flying RC planes, Connect USB joystick devices here

If using the chair for flying simulator games, connect USB joystick devices to the PC running the game

Microsoft Flight Sim 2020/2024

```
temp.py x MSFS_Connect.py x War_Thunder_Connect.py x Send_Data.py x
1  # -*- coding: utf-8 -*-
2  """
3  Created on Sat Oct 26 18:48:44 2024
4
5  @author: micha
6  """
7
8  from SimConnect import SimConnect, AircraftRequests
9  import time
10 from pprint import pprint
11 import serial
12 import serial.tools.list_ports
13 import math
14
15
16 port = "COM8" # update this to your Arduino's serial port
17 baud_rate = 9600
18 ser=serial.Serial(port, baud_rate, timeout=1)
19 time.sleep(2)
20
21 vector = [0, 0]
22
23 sm = SimConnect()
24 # Create a request object to pull data from MSFS
25 aq = AircraftRequests(sm, _time=10)
26
27 while True:
28     try:
29         # Retrieve pitch and roll data
30
31         pitch = aq.get("PLANE_PITCH_DEGREES")
32         roll = aq.get("PLANE_BANK_DEGREES")
33
34         # Display pitch and roll
35         #print(f"Pitch: {pitch:.2f} degrees, Roll: {roll:.2f} degrees")
36
37         # Pause for a short interval to reduce load
38         #time.sleep(0.1)
39         vector[0]=roll*(-180/math.pi)
40         vector[1]=pitch*(-180/math.pi)
41         vector_str = f"{vector[0]},{vector[1]}\n"
42         ser.write(vector_str.encode('utf-8'))
43         print(vector)
44
45
46 except KeyboardInterrupt:
47     ser.close()
48     print('Closing')
49     break
50
51 except:
52     next
```

Download packages
with Pip Installer

Replace with
correct COM port

SimConnect:

<https://pypi.org/project/SimConnect/>

<https://github.com/odwdinc/Python-SimConnect>

WarThunder

```
temp.py x  MSFS_Connect.py x  War_Thunder_Connect.py x  Send_Data.py x
1  #-*- coding: utf-8 -*-
2  """
3  Created on Wed Jul 31 00:17:30 2024
4
5  @author: micha
6  """
7
8  from WarThunder import telemetry
9  from WarThunder import mapinfo
10 from pprint import pprint
11 import serial
12 import serial.tools.list_ports
13 import time
14
15 port = "COM14" # update this to your Arduino's serial port
16 baud_rate = 9600
17 ser=serial.Serial(port, baud_rate, timeout=1)
18 time.sleep(2)
19
20 vector = [0, 0]
21
22 while True:
23     try:
24         telem = telemetry.TelemInterface()
25
26         while not telem.get_telemetry():
27             pass
28
29         dat=telem.basic_telemetry
30         pitch_ang=dat['pitch']
31         roll_ang=dat['roll']
32
33         vector[0]=roll_ang
34         vector[1]=pitch_ang
35         vector_str = f"{vector[0]},{vector[1]}\n"
36         ser.write(vector_str.encode('utf-8'))
37         print(vector)
38
39     except KeyboardInterrupt:
40         print('Closing')
41         break
42
43
```

Download packages
with Pip Installer

Replace with
correct COM port

WarThunder Python Package:

<https://pypi.org/project/WarThunder/2.0.4/>

Ardupilot RC Plane

```
temp.py x  MSFS_Connect.py x  War_Thunder_Connect.py x  Ardupilot_Connect.py x
1  # -*- coding: utf-8 -*-
2  """
3  Created on Tue Feb  6 16:56:49 2024
4
5  @author: micha
6  """
7
8  from pymavlink import mavutil
9  import struct
10 import math
11 import serial
12 import serial.tools.list_ports
13 ports = serial.tools.list_ports.comports()
14
15
16 import serial
17 import time
18 port = "COM5" # Update this to your Arduino's serial port
19 baud_rate = 9600
20 ser=serial.Serial(port, baud_rate, timeout=1)
21 time.sleep(2)
22
23 vector = [0, 0]
24
25 def float_to_bytes(f):
26     return struct.pack('f', f)
27
28 master = mavutil.mavlink_connection('udpin:localhost:14445')
29
30 timeout = 5 # maximum seconds to wait for a message
31 if not master.wait_heartbeat(timeout=timeout):
32     ser.close()
33     master.close()
34     raise TimeoutError(f"No heartbeat within {timeout=}A seconds!")
35
36 while True:
37     msg = master.recv_match(type='ATTITUDE', blocking=True).to_dict()
38     #roll_bytes = float_to_bytes(msg['roll']*(180/math.pi))
39     #pitch_bytes = float_to_bytes(msg['pitch']*(180/math.pi))
40     #heave_bytes = float_to_bytes(0)
41     #yaw_bytes = float_to_bytes(0)
42     #sway_bytes = float_to_bytes(0)
43     #surge_bytes = float_to_bytes(0)
44     #data_send = roll_bytes + pitch_bytes+heave_bytes+yaw_bytes+sway_bytes+surge_bytes
45     vector[0]=msg['roll']*(180/math.pi)
46     vector[1]=msg['pitch']*(180/math.pi)
47     vector_str = f"{vector[0]},{vector[1]}\n"
48     ser.write(vector_str.encode('utf-8'))
49     print(vector)
50
```

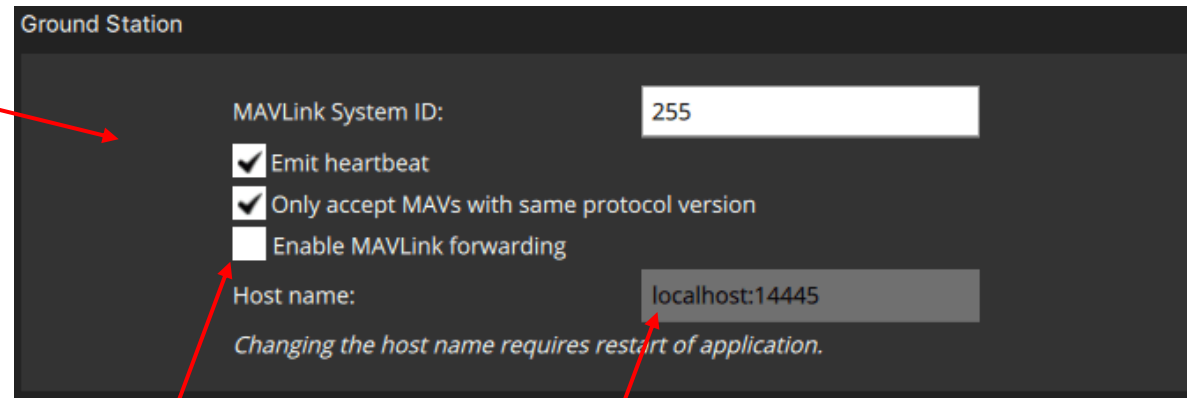
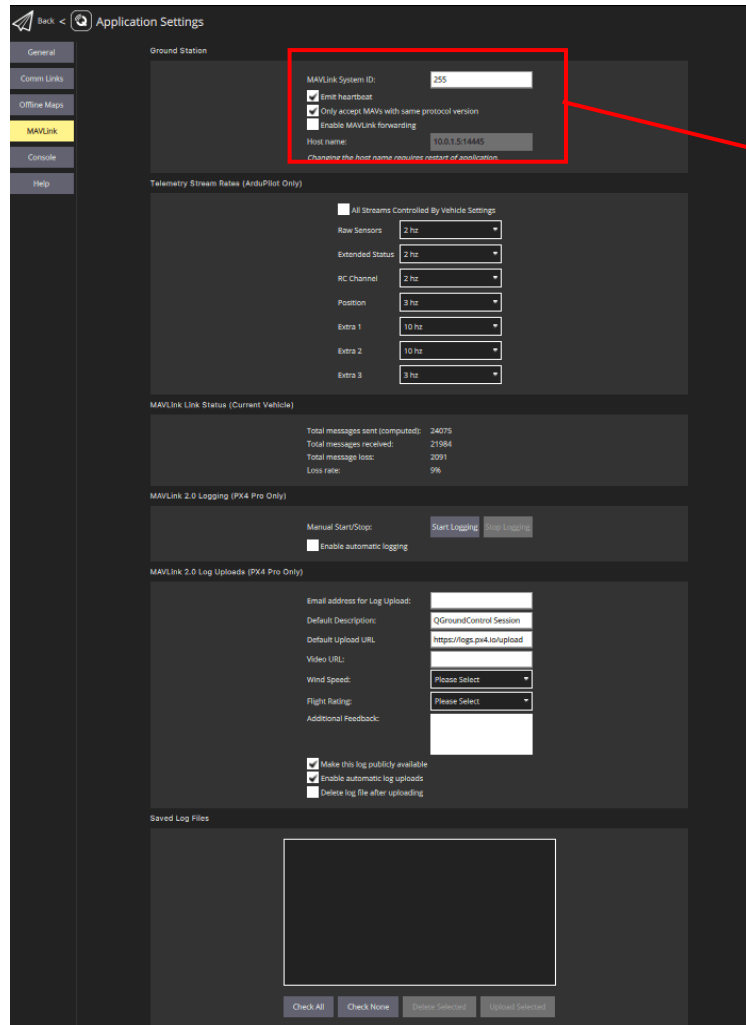
Download packages
with Pip Installer

Replace with
correct COM port

Replace with
correct UDP port.
See next Slide

Ardupilot RC Plane

- QGroundControl must be configured to forward the MAVLINK packets to a UDP port
- Full Documentation: https://docs.qgroundcontrol.com/Stable_V4.3/en/qgc-user-guide/settings_view/mavlink.html



Check This

Port Name

Buttkicker for Sim Games



Bill of Materials

Item	Qty	Link (May be Affiliate)	Notes
Electronics			
Raspberry Pi 4	1	https://amzn.to/4gDeqRJ	
Teensy 4.0	1	https://amzn.to/4h26auC	
MPU6050	1	https://amzn.to/3DHh3n9	
Level Shifter (Only if using LED)	1	https://amzn.to/3DGJHoi	
5V LED Strip	1	https://amzn.to/4j0shDi	
JST Connectors (Optional but makes wiring easier)	1	https://amzn.to/4j26Rpf	
Terminal Blocks (Optional but makes wiring easier)	1	https://amzn.to/4gACqVC	
Custom PCB (Limited Stock Available)	1		
USB Flight Stick / Throttle (Most should work)	1	https://amzn.to/3BJ2AXf	
USB Rudder Pedals (Most should work)	1	https://amzn.to/3BWKVLD	
20:1 NEMA 21 Gearbox	2	https://amzn.to/4iXhycP	
NEMA 21 Closed Loop Stepper Motor Kit	2	https://www.omc-stepperonline.com/ts-series-3-0-nm-424-92oz-in-1-axis-closed-loop-stepper-cnc-kit-nema-23-motor-driver-1-cl57t-s30a-v41	
NEMA 21 Motor Shaft Adapter	2	https://amzn.to/3PhDD8q	
Motor Shaft Hub	2	https://amzn.to/3W463Gx	Many versions exist of this hub. Steel or aluminum should would
Gearbox Double Shaft	2	https://amzn.to/40mkeJW	
Chair Structure			
3/4" Plywood		Local Hardware Store	
1/2" Plywood		Local Hardware Store	
Screws		Local Hardware Store	
Heim Joints		https://amzn.to/4jpR4Rm	3/8-24 Female Thread Right Hand
Connecting Rod		https://www.mcmaster.com/98842A031/	3/8 x 24 - 2 sections at least 400mm
Angle Iron		Local Hardware Store	
T Nuts		https://amzn.to/42f8U3v	
Pack of spacers for Heim Joints		https://amzn.to/4g4bUCX	1/2" OD x 3/8" ID 0.5" length
Pack of Bolts for Mounting Heim Joints:		https://amzn.to/3WrSz7X	
		https://amzn.to/4ar7t3P	
3/8 NyLock Nuts			
Other (Optional)			
Buttkicker		https://amzn.to/3Q2x4ah	
		https://www.amazon.com/gp/product/B07MR6YQC1/ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&psc=1	
Amp for Buttkicker			

