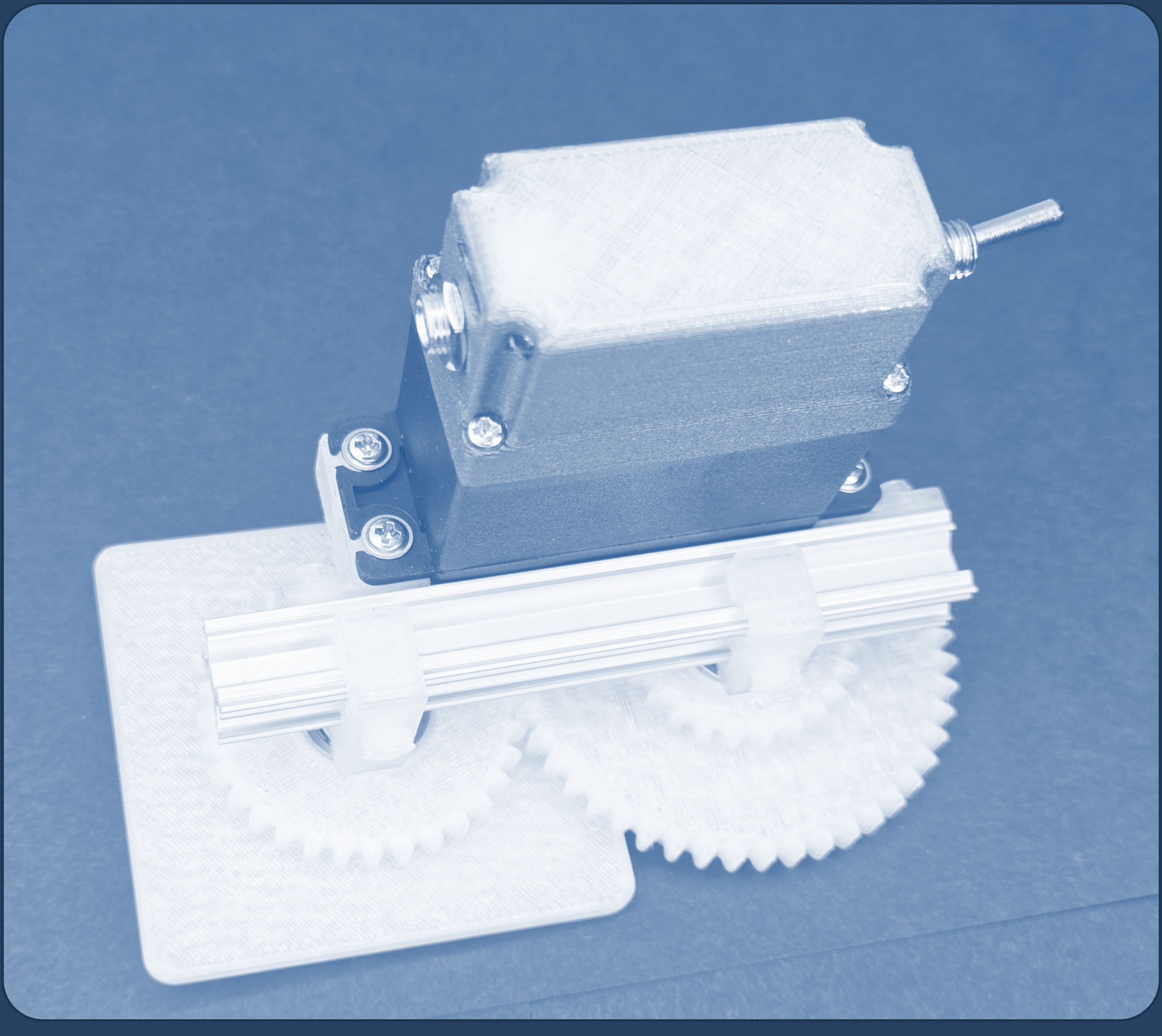
GEAR SYSTEMS: STEADY DISPLAY KIT



SMARTSERVO PROJECT

GEAR SYSTEMS: STEADY DISPLAY KIT

Version 1.0 | Published: June 17, 2025 | Author: Judson Wagner, Wagner Labs LLC

Commercial Use & Smart Servo Requirement

For commercial use, bulk educational licensing, or use outside of traditional educational settings, please contact Wagner Labs LLC.

Hardware Requirement: This guide requires **Smart Servo devices** to complete the projects and activities described. Smart Servos are available through the Smart Servo Store and authorized distributors.

About the Smart Servo Project

The Smart Servo Project empowers inclusive innovation by providing accessible tools for creating assistive technologies and engaging STEM education. Our mission is to bridge technology and compassion through community-driven maker education.

Support our mission by purchasing Smart Servos and sharing our resources with your educational community.

Contact Information:
Judson Wagner | Wagner Labs LLC
Email: **Judson@WagnerLabs.net**
Website: **WagnerLabs.net/SmartServo**
Smart Servo Store: **WagnerLabs..Store**

Client: Dr. Elena Rodriguez, Age 67

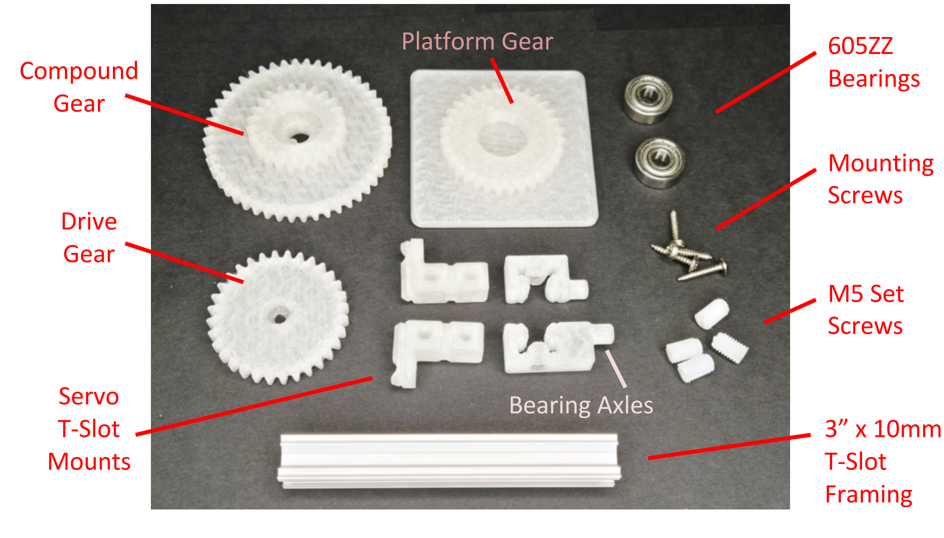
About Me: I'm a retired art history professor who loves collecting small sculptures and pottery. I have rheumatoid arthritis in my hands and wrists, which makes it painful to pick up and rotate objects to examine them from different angles.

My Challenge: I have a beautiful collection of ceramic pieces and small sculptures that I enjoy studying, but I can no longer comfortably handle them to view from all sides. The twisting motion required to rotate objects causes significant pain in my joints.

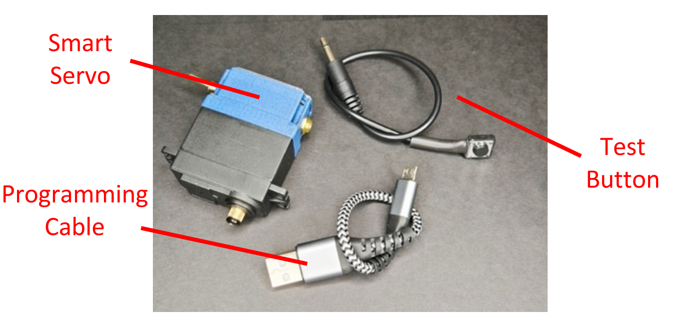
Technical Need: A slow, stable rotating platform that can display objects at various angles, allowing me to appreciate my collection without having to physically manipulate the pieces.

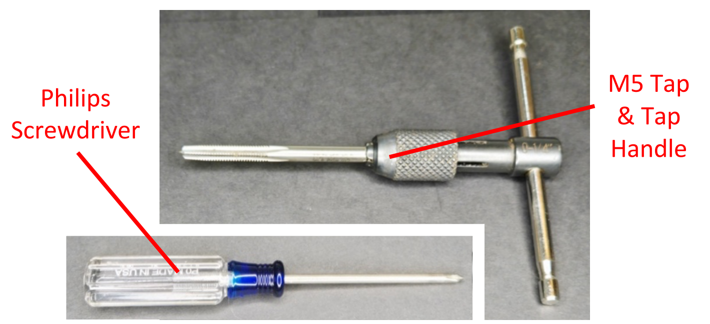
Let's investigate our kit and see if we can get started on something that can assist Dr. Rodriquez.

STEP 1: Lay out all the components that are new in this kit.

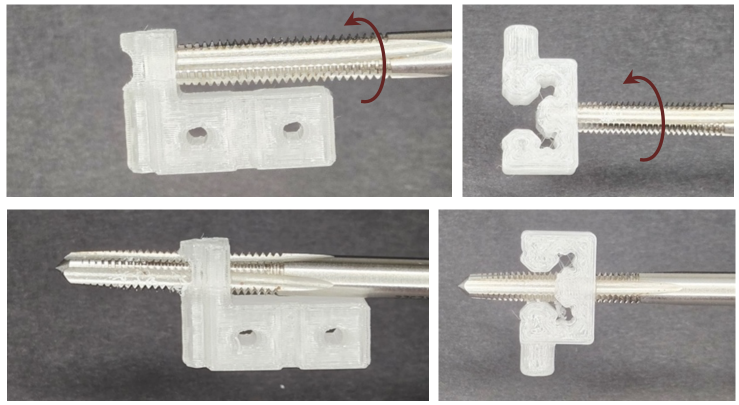


STEP 2: Make sure you have these items from your previous kits.

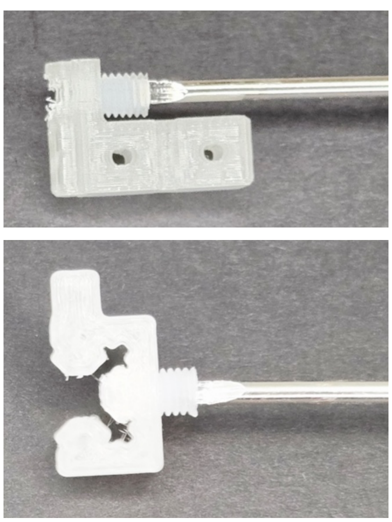


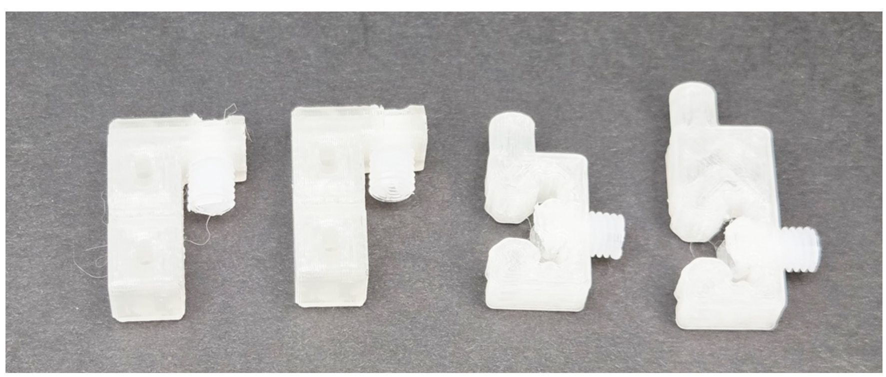


STEP 3: Prepare your parts. Use you M5 Tap to carve threads into the Servo T-Slot Mounts and Bearing Axles. Remember to carve slowly with the tap perpendicular to the hole. Make sure to keep carving until the tap turns smoothly and back the tap out completely. Blow away any remaining loose material.

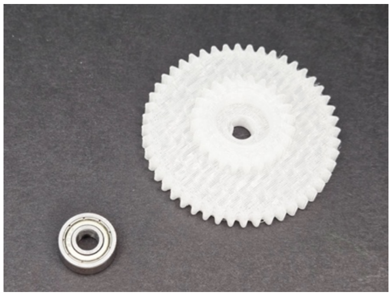


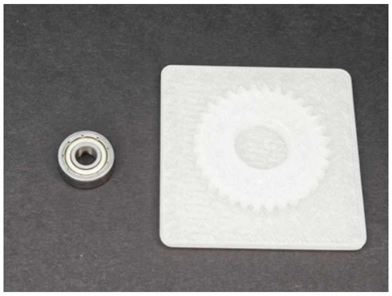
STEP 4: Insert your set screws. Make sure these are not hard to put in. Though they are small, they should be turn-able by hand. If not, gently re-tap to remove any remaining material. Back the screws out nearly all the way and set these parts aside.



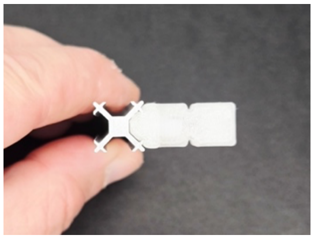


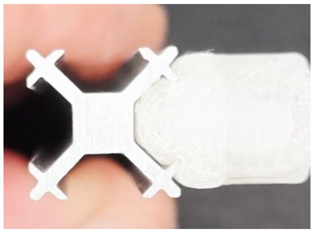
STEP 5: Press fit your bearings. Find your two bearings, your platform gear, and your compound gear. The bearings need to be pushed in perpendicular to the 10mm holes so that they are almost completely flush. This may require a lot of hand strength but if you need assistance, use flat surfaces such as a table and a book versus pliers to ensure that the bearing goes in straight. The ideal tool is an arbor press if you have access to one.



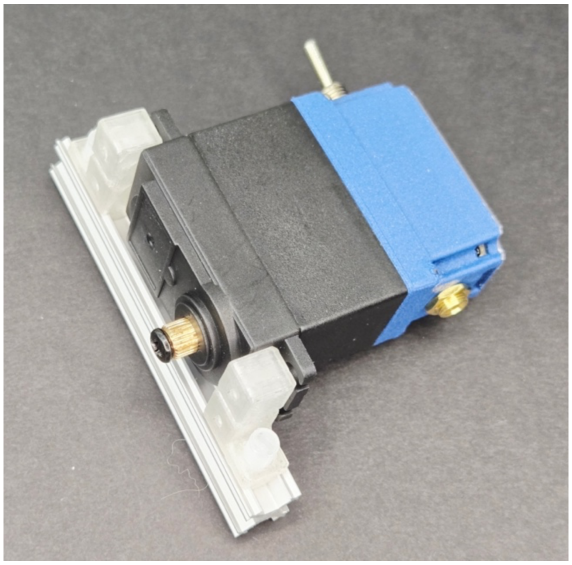


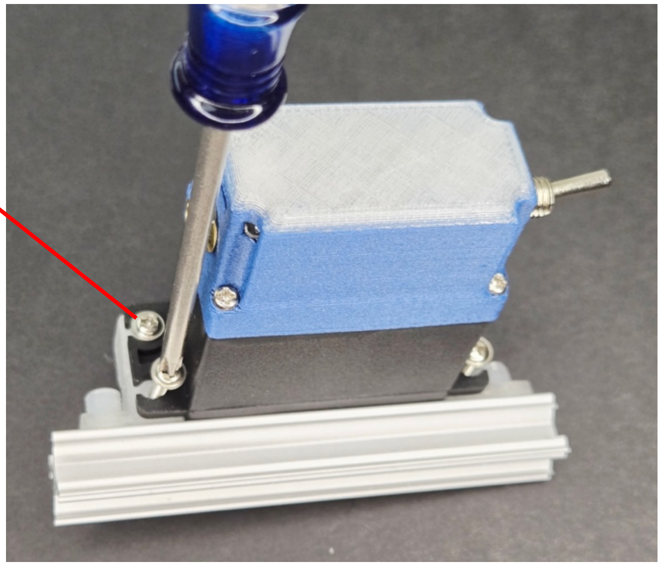
STEP 6: Add your Servo T-Slot Mounts. It might take a little wiggling, but the plastic pieces should slide into the T-Slot Frame. Double check that your set screws aren't in too far. Make sure the two mounts are oriented as shown but don't tighten set screws yet.



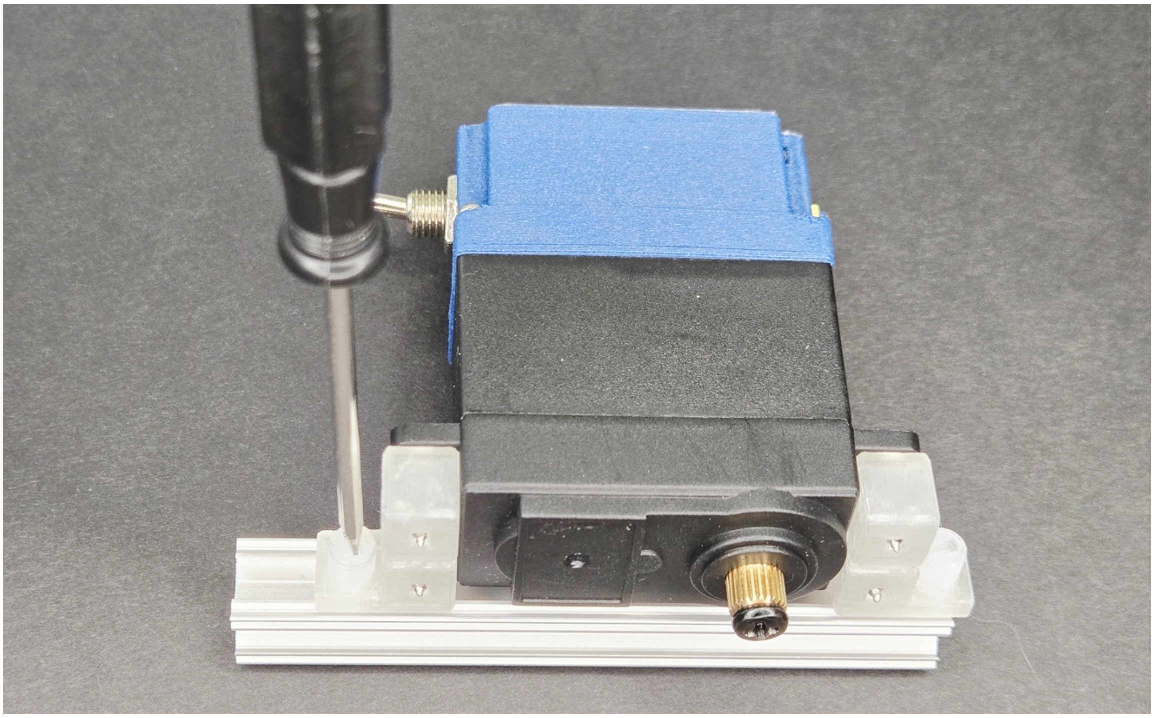


STEP 7: Position your Smart Servo as shown and slide the Servo T-slot Mounts inwards. Flip these over, place your mounting screws over the smaller holes in the mounts, and screw down tight using your Philips screwdriver.

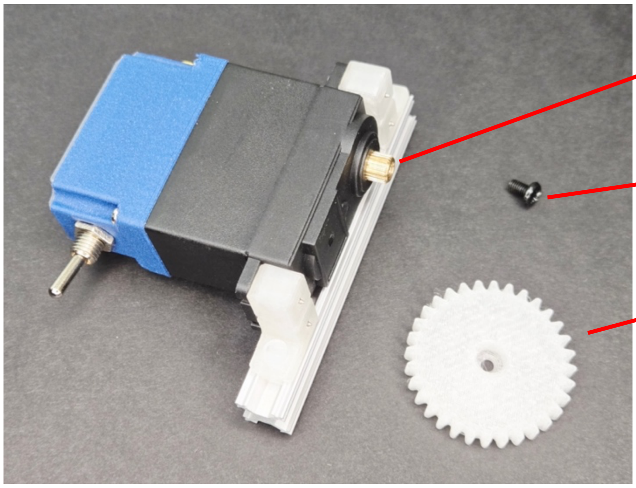


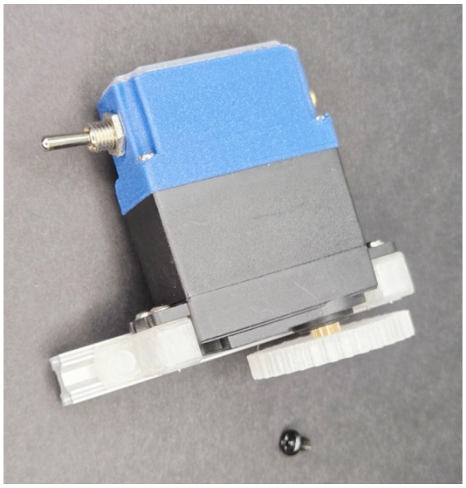


STEP 8: Tighten your set screws using you 3mm slotted screwdriver taking care not to over tighten. Check that the Smart Servo is held firmly in place against the T-Slot Frame.

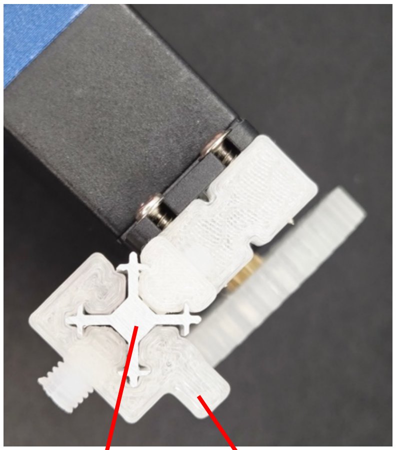


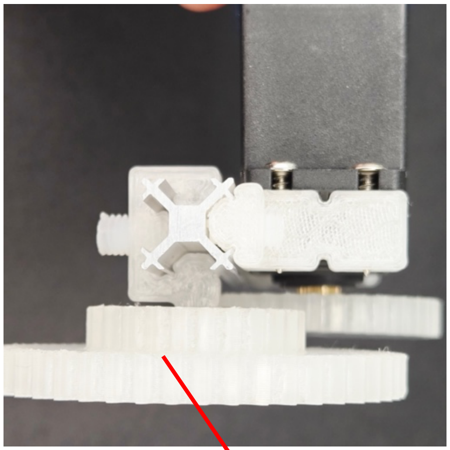
STEP 9: Remove the 3m Spline Screw from the Smart Servo and Press the Drive Gear into the Smart Servo Spline. Store the M3 Spline Screw in a safe place.

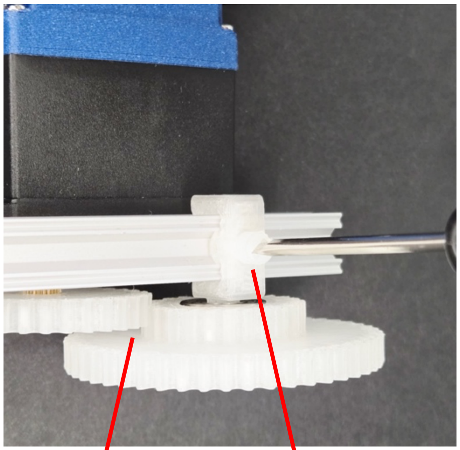




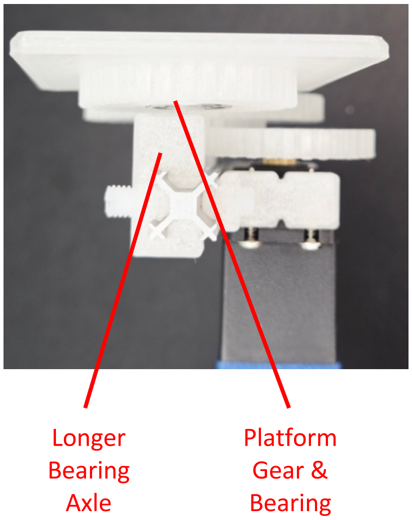
STEP 10: Slide the shorter Bearing Axle onto the T-Slot Frame. Press the Bearing, found in the Compound Gear, all the way onto the short Bearing Axle as shown. Finally, slide the Bearing Axle across the T-Slot Frame until the gear teeth on the Compound Gear mesh tightly with the teeth on the Drive Gear as shown. Use your screwdriver to secure the set screw taking care not to over tighten.

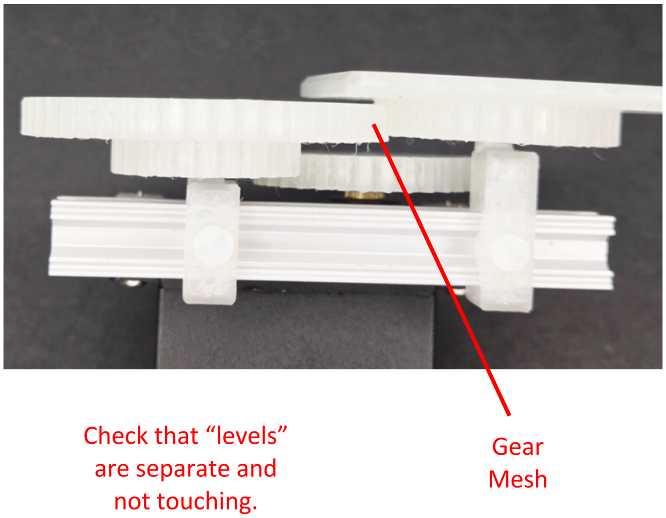


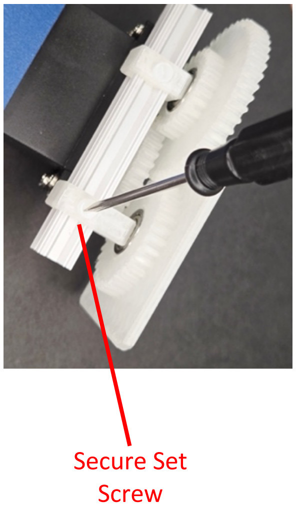




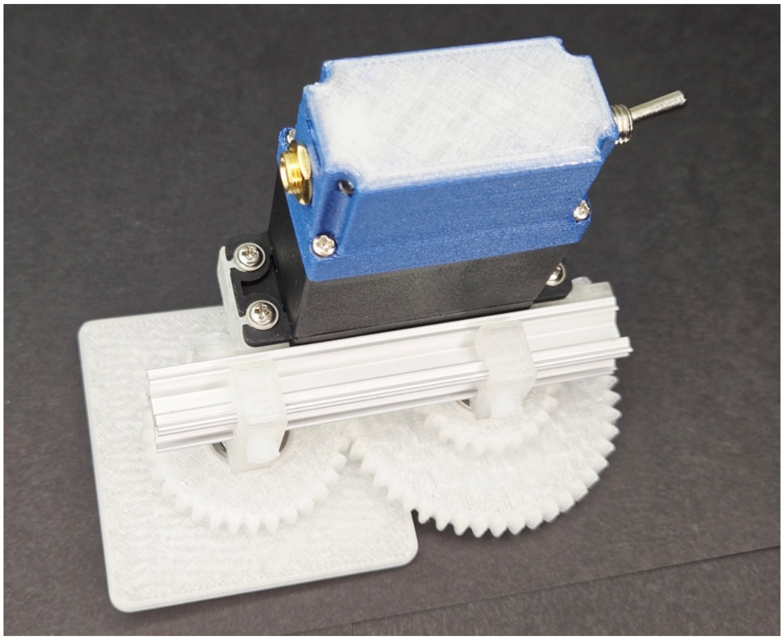
STEP 11: Repeat Step 10 with the longer Bearing Axle and Platform Gear. Make sure to slide the Bearing Axle onto the opposite end of the T-Slot Frame. When you press the Bearing that is in the Platform Gear onto the Bearing Axle, the gear is in the same plane as the larger side of the Compound Gear. Make sure the teeth mesh and lock it down with the set screw. (But don't over-tighten)







STEP 12: Give your mechanism a quick check. None of the gears should be able to move, no gear should be touching more than one other gear, and the set screws should be snug enough to keep anything from sliding on the T-Slot Rail.



STEP 13: Connect your testing button and USB power to the Smart Servo. When you press the button and run Frank's Code, you should notice that the platform rotates around much farther than the original "180 degrees". This is good, but you'll probably also notice that it rotates way to fast. To slow this down, we're going to add the following code. Feel free to Copy & Paste. We've highlighted the part that you can tweak to alter the speed of rotation.

COPY & PASTE this code from: tinyurl.com/ss-code-gears

```
# Fixed code with slow servo movement
import time
import board
from digitalio import DigitalInOut, Direction, Pull
import pwmio
import servo

import adafruit_dotstar
pix = adafruit_dotstar.DotStar(board.APA102_SCK, board.APA102_MOSI, 1)

button = DigitalInOut(board.D2)
button.direction = Direction.INPUT
button.pull = Pull.UP

switch = DigitalInOut(board.D1)
switch.direction = Direction.INPUT
switch.pull = Pull.DOWN

pwm = pwmio.PWMOut(board.A2, duty_cycle=2 ** 15, frequency=50)
servo1 = servo.Servo(pwm)

# Initialize servo position
current_angle = 180  # Starting position
servo1.angle = current_angle

state = 0
last_button_state = button.value
debounce_time = 0

while True:
    pix[0] = (0, 100, 0)  # Green light when idle

    # Simple debounce for button
    if time.monotonic() - debounce_time < 0.05:
        time.sleep(0.01)
        continue
```

```
    if switch.value == 1 and button.value == 0 and state == 0:
        pix[0] = (0, 0, 100)  # Blue light during movement

        # Slow movement to 0 degrees
        target = 0
        delay = 0.05
        steps = 20  # Slowing down by dividing the motion into steps

        angle_step = (target - current_angle) / steps
        for i in range(steps):
            # Calculate intermediate position
            pos = int(current_angle + angle_step * (i + 1))
            servo1.angle = pos
            time.sleep(delay)

        current_angle = 0
        time.sleep(0.5)  # Short pause at end position

        # Slow movement to 180 degrees
        target = 180
        angle_step = (target - current_angle) / steps
        for i in range(steps):
            pos = int(current_angle + angle_step * (i + 1))
            servo1.angle = pos
            time.sleep(delay)

        current_angle = 180
        state = 1
        pix[0] = (0, 100, 0)  # Green light when done
        debounce_time = time.monotonic()

    elif switch.value == 1 and button.value == 1 and state == 1:
        # Maintain position at 180
        servo1.angle = 180
        current_angle = 180
        pix[0] = (0, 100, 0)
        state = 0

    elif switch.value == 0 and button.value == 0:
        # Maintain position at 180 with yellow light
        servo1.angle = 180
        current_angle = 180
        pix[0] = (125, 115, 3)  # Yellow light
        time.sleep(0.2)  # Reduced to make the system more responsive

    elif switch.value == 0 and button.value == 1:
```

```
        # Maintain position at 180 with red pulsing light
        servo1.angle = 180
        current_angle = 180
        # Pulse light more efficiently
        for i in range(10, 255, 5):  # Smaller step count is more memory 
efficient
            pix[0] = (i, 0, 0)
            time.sleep(0.01)
        for i in range(255, 10, -5):
            pix[0] = (i, 0, 0)
            time.sleep(0.01)

    # Short delay to prevent CPU overload
    time.sleep(0.01)
```

CONGRATULATIONS!

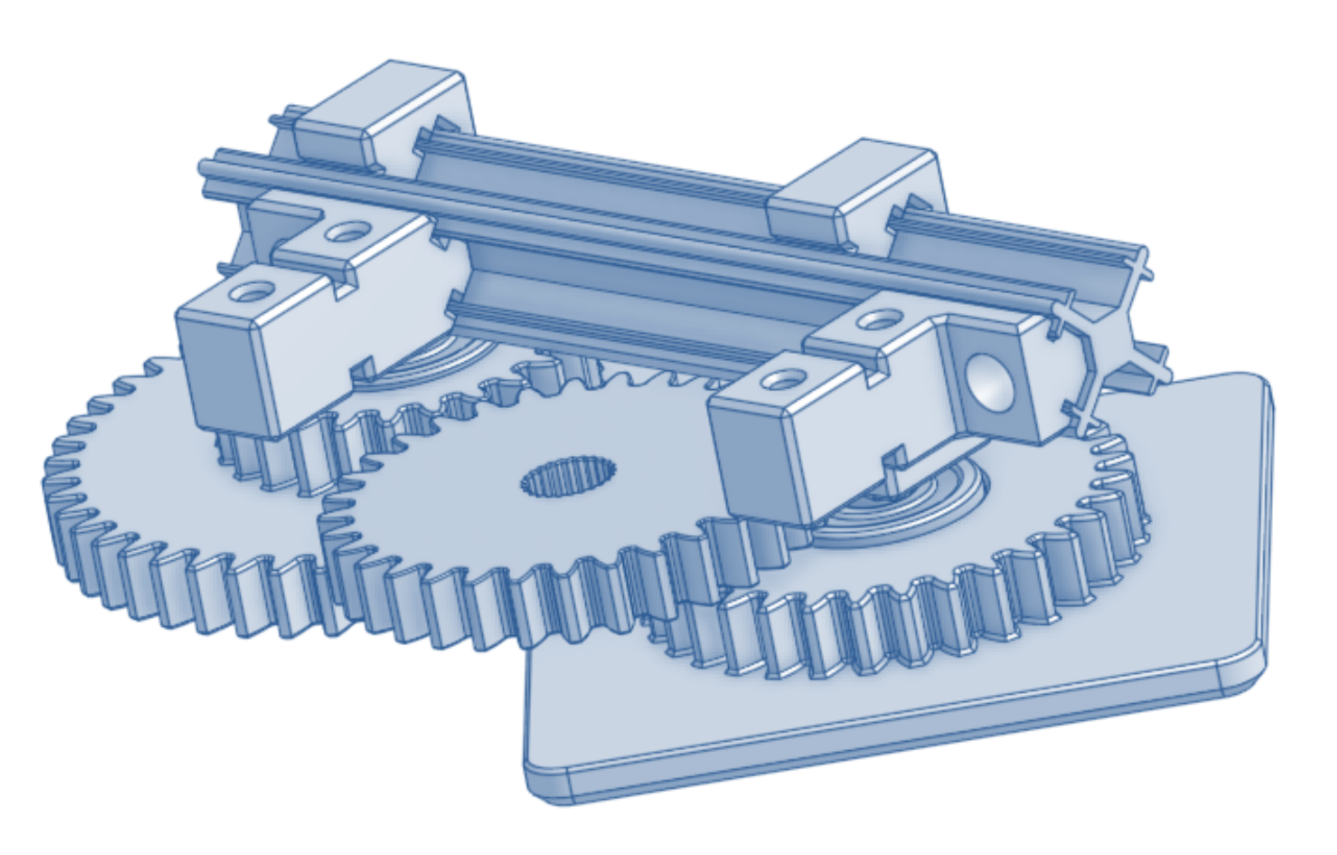
You just constructed a display system using gears. Next, we'll want to consider what needs to be done next to better assist Dr. Rodriquez. Read her profile again and see if she would appreciate something like this platform rotator. If so, reflect on what follow up questions you would ask her and what else she would need to use it.

REMINDER ABOUT CODING SNIPS

If you want to return your code to the original "factory setting", just copy and paste from here: tinyurl.com/SmartServoSnips

3D PRINTING FILES

If you're able to 3D Print, download the 3D parts used in this project here: tinyurl.com/SS-STL-GEARS



THE BIGGER PICTURE

UNDERSTANDING MECHANICAL ADVANTAGE IN YOUR GEAR SYSTEM

Trading Speed for Power: The Fundamental Trade-off

When you connected gears to slow down your Smart Servo's rotation, you discovered one of the most important principles in mechanical engineering: you can't get something for nothing. Your servo motor spins fast but with limited turning force (called torque). By connecting a small gear to a larger gear, you traded some of that speed for increased torque and finer control. The motor has to work longer to complete one full rotation of your platform, but now it can rotate heavier objects with smooth, stable motion.

This trade-off is called mechanical advantage, and it's everywhere in your daily life. When you shift a bicycle into a lower gear to climb a hill, you're trading speed for power - your legs spin faster, but each push moves you forward more slowly with more force. When you use a screwdriver with a fat handle, you're trading the speed of your rotating hand for the increased torque needed to drive the screw into wood.

The Math Behind Gear Ratios

The relationship between your gears follows a simple but powerful rule: the ratio of teeth determines exactly how much speed you lose and torque you gain. If your small gear has 10 teeth and connects to a large gear with 40 teeth, that's a 4:1 gear ratio. The large gear rotates 4 times slower but with 4 times more torque.

Gear ratios are why car transmissions exist - different gears let the engine operate efficiently whether you're accelerating from a stop (needing high torque) or cruising on the highway (needing high speed). Engineers calculate these ratios precisely to match the motor's capabilities to the task at hand.

Compound Gears: Multiplying the Effect

Your gear system likely uses compound gears - multiple gear pairs working together. This is where mechanical advantage gets really powerful. If you have a 4:1 reduction followed by another 4:1 reduction, you don't just add them (4+4=8). You multiply them (4×4=16)! Each stage compounds the effect of the previous one.

This compounding principle enables extreme gear reductions. Industrial robots use harmonic drives with gear ratios exceeding 100:1 to achieve precise positioning. Clock mechanisms use multiple compound gear stages to convert the relatively fast oscillation of a quartz crystal into the slow movement of hour and minute hands. Wind turbines use compound gearboxes to convert the slow rotation of massive blades into the fast spin needed by electrical generators.

The platform you built demonstrates the same engineering thinking behind all these applications: understanding the speed-torque trade-off and using gear ratios to precisely match a motor's capabilities to real human needs.