Robotic arm
simulating human muscle kinematics

ECE-478
Dmitrii Fotin

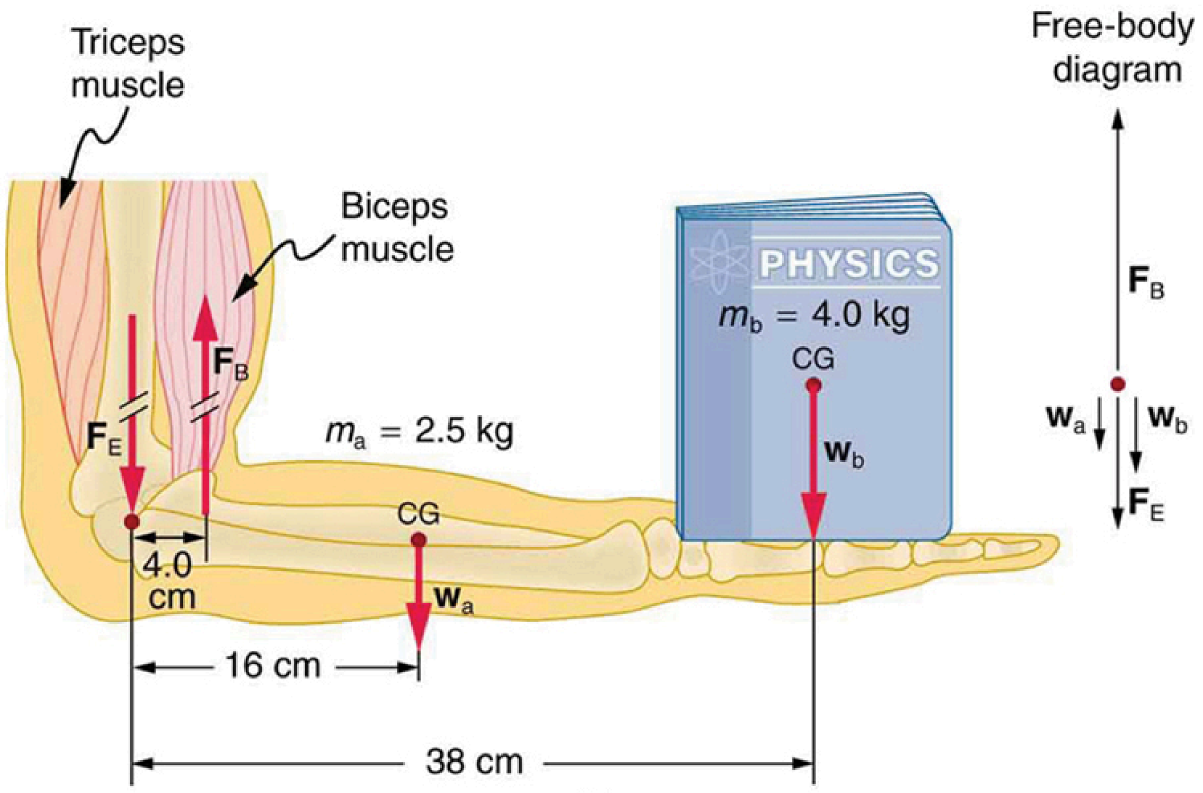
Introduction

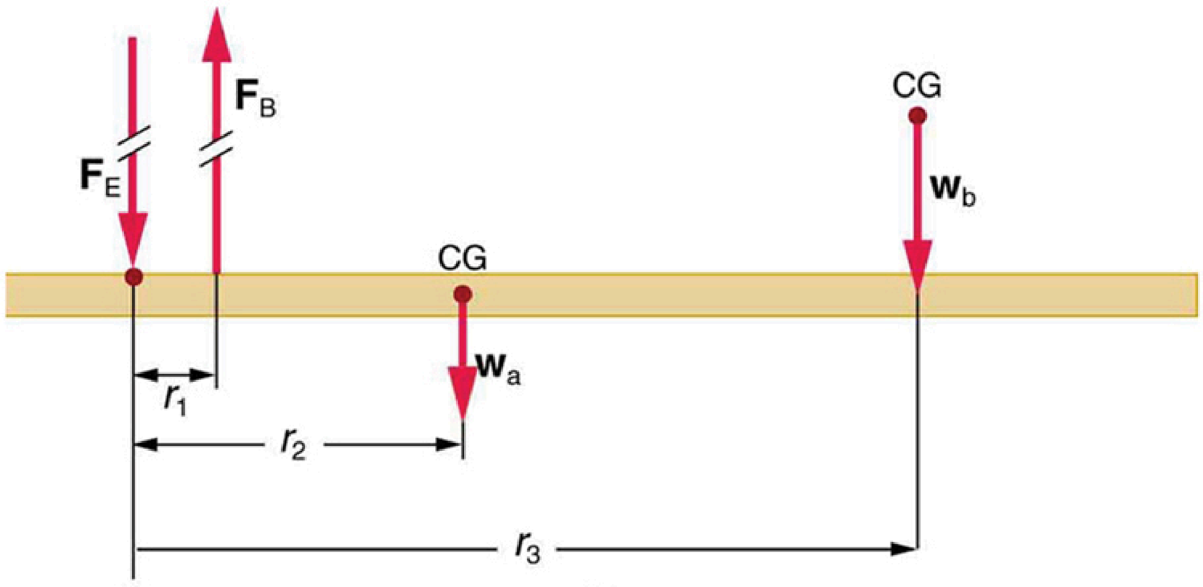
Robotic arms with servo motors installed in joints require increasingly stronger servos to handle heavier loads. The proposed solution is to have strings/ribbons attached to the actuated limbs which allows installing multiple weaker servo motors acting in the same force vector outside of the joint to achieve the same or better performance.

Design

Force diagram, calculations and modelling

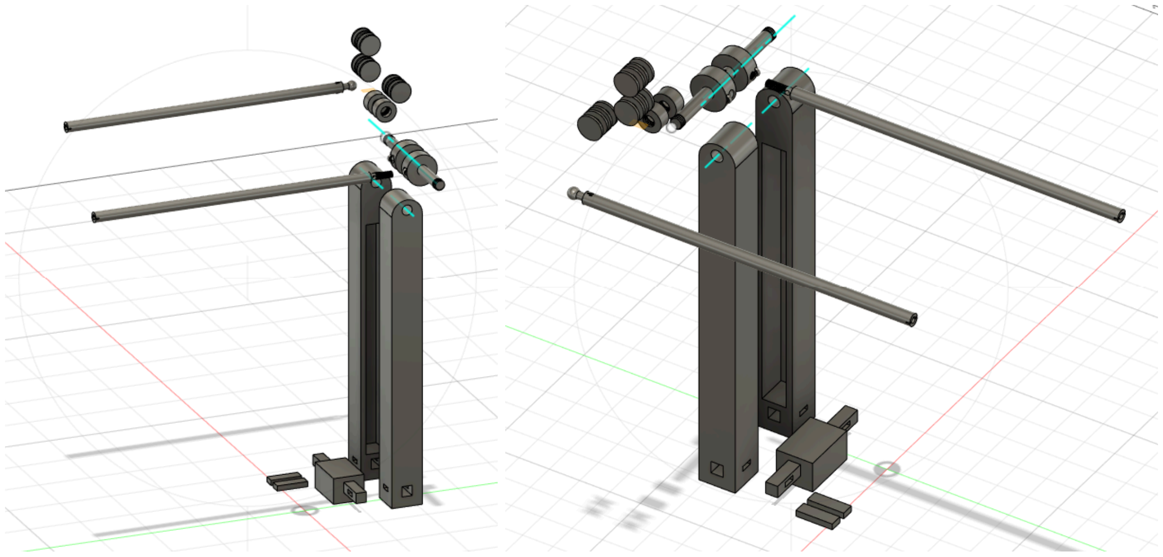
The robotic arm design is to have an elbow and a forearm of 1ft length each, the arm should be able to manage loads of up to 1 lb. There are to be two degrees of freedom: one in the elbow joint and one for wrist rotation around its axis.
The force diagram in Images 1 and 2 below shows the torques in equilibrium with the pulley attached perpendicularly to the forearm to maximize the upward torque that counterweights the sum of torques pulling the body down. Pulling the load up and down would require higher force than the sum of the counteracting forces.
Attaching the ribbons as close to the joint as possible results in a higher amount of torque needed but enables quick movements. Additionally, servo motors have small rotating elements, therefore designing rotating elements in the joints with the same or smaller radius than that of the servo motor would result in the optimal torque transfer.



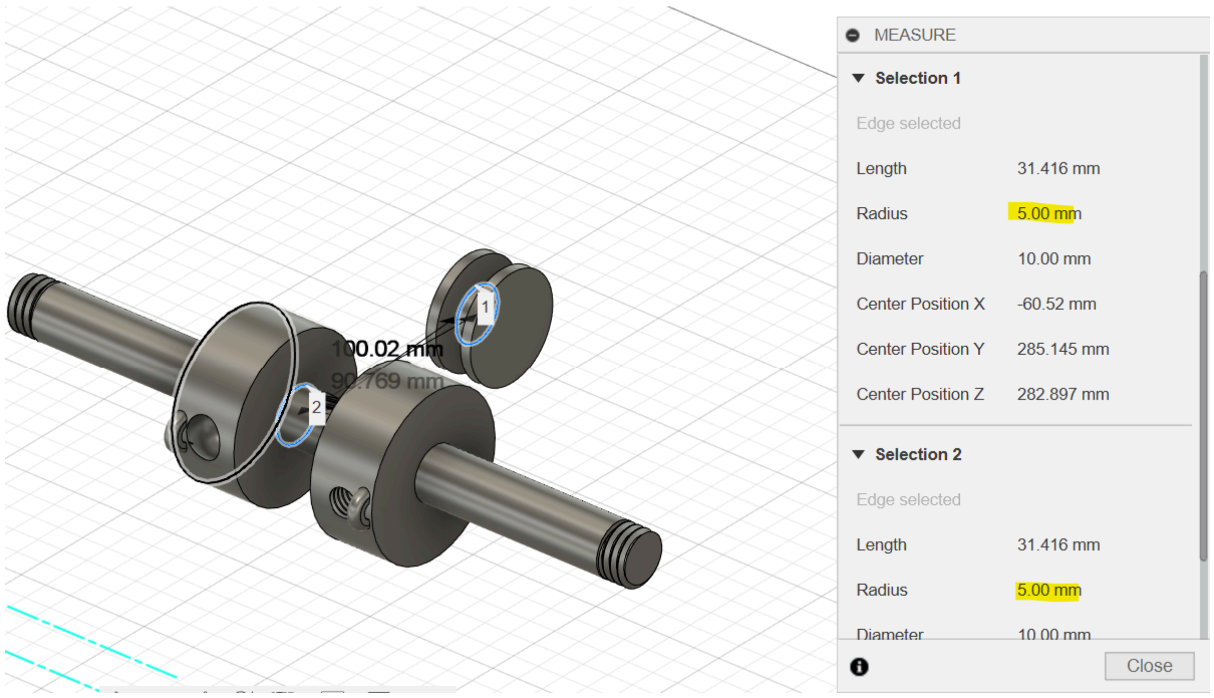


Images 1 and 2: Example torque diagram

The design of the robotic arm shown in Images 3 and 4 below has a rotating element in the joint and the rotating elements attached to the servo motors of the same radius to ensure optimal torque transfer and quick rotation of the forearm.



Images 3 and 4: 3D model of the proposed robotic arm design



MEASURE

Selection 1

Edge selected

Length	31.416 mm
Radius	5.00 mm
Diameter	10.00 mm
Center Position X	-60.52 mm
Center Position Y	285.145 mm
Center Position Z	282.897 mm

Selection 2

Edge selected

Length	31.416 mm
Radius	5.00 mm
Diameter	10.00 mm

Image 5: Servo motor and joint rotating element radius

For equilibrium, the goal is to balance the sum of the upward torques against the sum of the downward torques.

$$r_{arm} m_{arm} g + r_{load} m_{load} g = r_{pulley} F_{pulley}$$

Equation 1: Equilibrium torque system

In order to pull the arm and the load upwards at a constant speed, the right side of the equation has to be bigger. To let the arm and the load go down at a constant speed, the left side of the equation has to be bigger.
Assuming the mass of the arm and the max mass of the load is 0.5 kg for each, the 30 cm length of the forearm and the 90° angle between the pulley and the forearm, the torque required for equilibrium is calculated below.

$$0.15\, m \cdot 0.5\, kg \cdot 9.8\, m/s + 0.3\, m \cdot 0.5\, kg \cdot 9.8\, m/s = 2.21\, N \cdot m$$

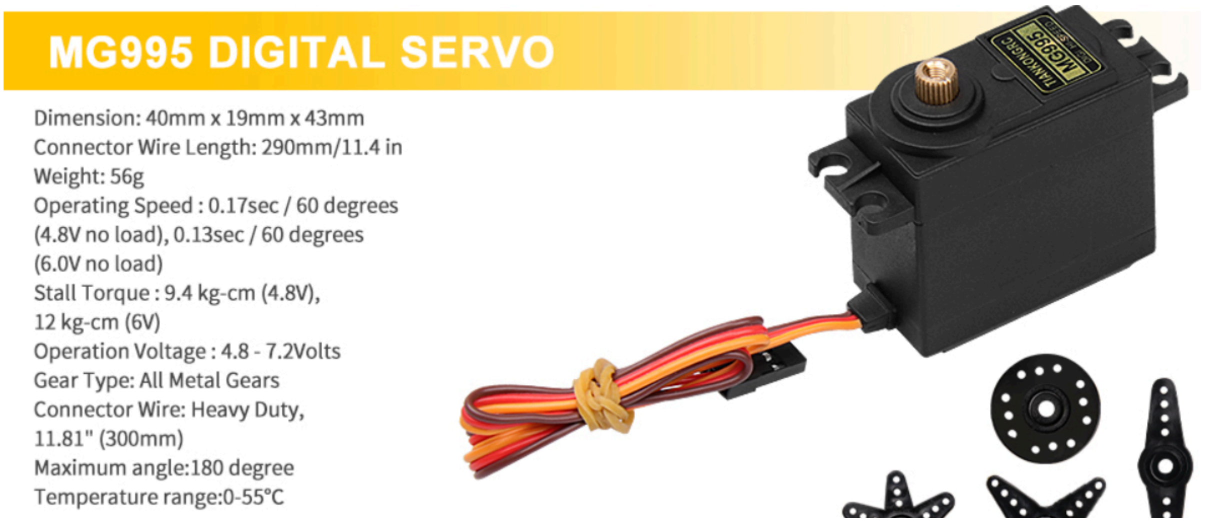
Equation 2: Required torque for equilibrium between shoulder and forearm with defined project parameters

For the forearm, only one "bone" needs to be pulled, which is expected to reduce the mass to 0.25 kg. The pulleys will be attached at the base of the moving component to get the angle as close to 90° as possible. For the purposes of the initial equations, the angle is assumed to be 90°.
Depending on the load shape, more kinematics come into play. Assuming a spherical object that would be rotated around its axis while resting on the static component of the forearm, the mass of the load can be eliminated.

$$0.15\, m \cdot 0.25\, kg \cdot 9.8\, m/s = 0.37\, N \cdot m$$

Provided a default range of 180° for most servos, it is important to design for the same range of motion between the servos and the component that would rotate the forearm at 1:≤1 ratio, i.e. the radius of the rotating element attached to each servo is greater than or equal to the

radius of the component that would rotate the forearm. That way, if the servo rotates 90°, the forearm should also rotate 90° or more.
After researching available servo options, the choice landed on a 180° servo with torque of 12kg/cm at 6V, [MG995 supplied by Beffkkip](#).



Dimension: 40mm x 19mm x 43mm
Connector Wire Length: 290mm/11.4 in
Weight: 56g
Operating Speed : 0.17sec / 60 degrees (4.8V no load), 0.13sec / 60 degrees (6.0V no load)
Stall Torque : 9.4 kg-cm (4.8V), 12 kg-cm (6V)
Operation Voltage : 4.8 - 7.2Volts
Gear Type: All Metal Gears
Connector Wire: Heavy Duty, 11.81" (300mm)
Maximum angle:180 degree
Temperature range:0-55°C

Image 6: Servo selected for the project

The 12 kg·cm torque at 6V can be converted to 1.17 N·m. Given the the 2.21 N·m, at least two servos will be required to rotate the forearm up and down. The design will first be tested with 2 servos for each bicep and tricep function and may be increased to 3 servos for each to increase the speed of rotation if needed.

For the forearm configuration, given the 0.37 N·m required torque, one servo on each side to pull the mobile "bone" is enough.

Circuit and code

The arm is operated with a potentiometer for each component. All servos can operate concurrently. The speed of rotation is the same throughout the movement.
The microprocessor used for the arm is Huzzah Feather ESP32 shown in image 7 below.

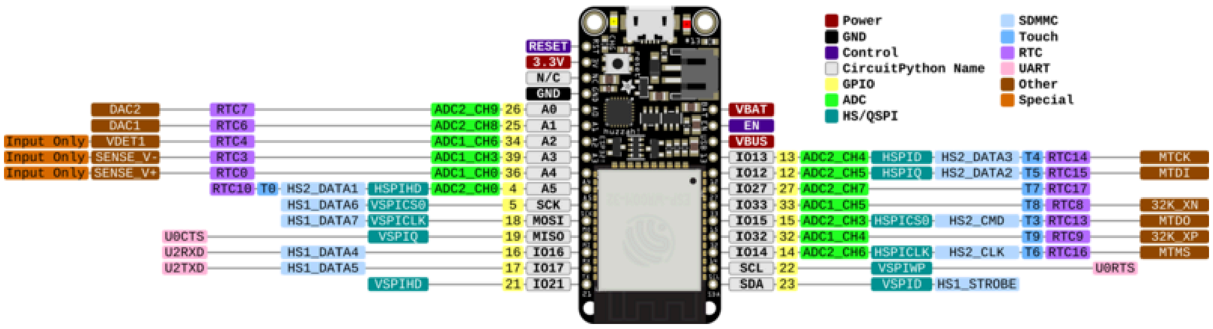
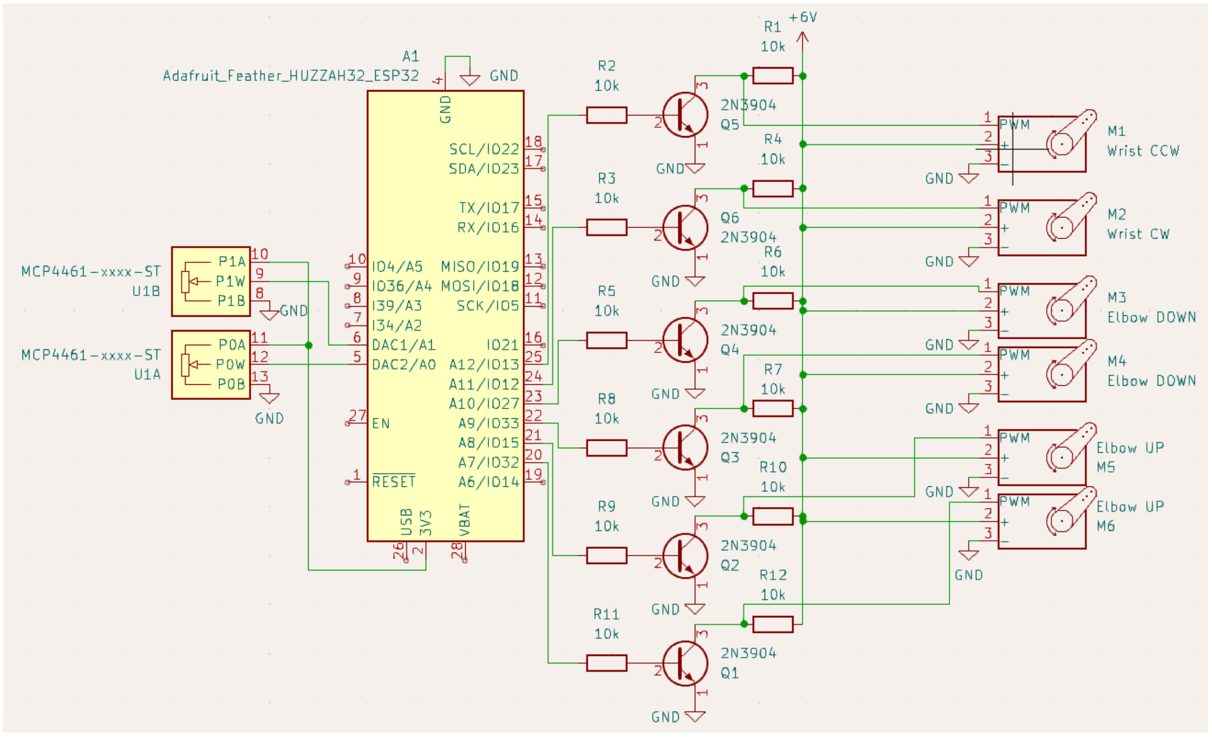

Power
GND
Control
CircuitPython Name
GPIO
ADC
HS/QSPI
SDMMC
Touch
RTC
UART
Other
Special
RESET
3.3V
N/C
GND
DAC2 RTC7 ADC2_CH9 26 A0
DAC1 RTC6 ADC2_CH8 25 A1
Input Only VDET1 RTC4 ADC1_CH6 34 A2
Input Only SENSE_V- RTC3 ADC1_CH3 39 A3
Input Only SENSE_V+ RTC0 ADC1_CH0 36 A4
RTC10 T0 HS2_DATA1 HSPIHD ADC2_CH0 4 A5
HS1_DATA6 VSPICS0 5 SCK
HS1_DATA7 VSPICLK 18 MOSI
U0CTS VSPIQ 19 MISO
U2RXD HS1_DATA4 16 IO16
U2TXD HS1_DATA5 17 IO17
VSPIHD 21 IO21
VBAT
EN
VBUS
IO13 13 ADC2_CH4 HSPID HS2_DATA3 T4 RTC14 MTCK
IO12 12 ADC2_CH5 HSPIQ HS2_DATA2 T5 RTC15 MTDI
IO27 27 ADC2_CH7 T7 RTC17
IO33 33 ADC1_CH5 T8 RTC8 32K_XN
IO15 15 ADC2_CH3 HSPICS0 HS2_CMD T3 RTC13 MTDO
IO32 32 ADC1_CH4 T9 RTC9 32K_XP
IO14 14 ADC2_CH6 HSPICLK HS2_CLK T6 RTC16 MTMS
SCL 22 VSPIWP U0RTS
SDA 23 VSPID HS1_STROBE

Image 7: [Huzzah Feather ESP32](#) microprocessor used to control the arm

Pins A0 and A1 are used for the potentiometer inputs and pins 13, 12, 27, 33, 15, 32 are used to send PWM signals to the servos.
As ESP32 operates at 3.3V, the voltage it provides is not sufficient to operate the servos selected, which require a minimum of 4.8V for a high signal. With that in mind, the circuit features voltage shifters using NPN transistors that enable a 6V high signal to the servo

motors whenever the transistors are disabled via the servo control pins. The waveform is therefore inverted.
The full final circuit is shown in Image 8 below.



Programming the microprocessor is done with Arduino IDE. There is a need for the driver/ledc.h library to enable PWM signals for ESP32 microprocessors within the Arduino IDE.
The microprocessor is programmed to take in the inputs from the potentiometers, map them to the servo PWM range and output PWM to the transistors, which in turn route 6V to the servos or to ground.
The full code can be viewed below.

```
#include <Arduino.h>
#include <driver/ledc.h>

const int servoPins[] = {13, 12, 27, 33, 15, 32};  // GPIO pins for
servos
const int potPin1 = A0;  // Analog pin for potentiometer 1 (controls 4
servos)
const int potPin2 = A1;  // Analog pin for potentiometer 2 (controls 2
servos)

void setup() {
  for (int i = 0; i < 6; i++) {
    ledcSetup(i, 50, 16);  // Configure LEDC PWM for servo control
    ledcAttachPin(servoPins[i], i);  // Attach the servo pin to the LEDC
channel
  }
  Serial.begin(9600);
}

void loop() {
  int potValue1 = analogRead(potPin1);
  int potValue2 = analogRead(potPin2);

  // Control the first four servos with potentiometer 1
  for (int i = 0; i < 4; i++) {
    int servoAngle1 = map(potValue1, 0, 4095, 0, 180);
    int pwmValue1 = map(servoAngle1, 0, 180, 57000, 65000);
    ledcWrite(i, pwmValue1);
  }

  // Control the last two servos with potentiometer 2
  for (int i = 4; i < 6; i++) {
    int servoAngle2 = map(potValue2, 0, 4095, 0, 180);
    int pwmValue2 = map(servoAngle2, 0, 180, 57000, 65000);
    ledcWrite(i, pwmValue2);
  }
  Serial.print("Potentiometer 1: ");
  Serial.print(potValue1);
  Serial.print("\t Potentiometer 2: ");
  Serial.println(potValue2);

  delay(20);  // Adjust delay as needed for smoother motion
}
```

Results

The 3D model has a few faults that prevent it from functioning correctly:

- The rotating wrist component has no constraints in place to make it follow the expected trajectory, so depending on the arm position, it either rotates below or above the static wrist component. Adding constraints would ensure movement similar to the human wrist kinematics.
- The stress points in the joint and where strings/ribbons attach to the components need to be stronger to avoid snapping the components.
- The strings initially used in the design did not prove strong enough to manage the tension applied to them and were replaced with ribbons.
- The ribbon connections for the wrist need to be redesigned, as currently the motors that move the wrist also move the arm up and down.
- The ribbons need to be attached in a more fanned out pattern along the components to spread out pressure and to make sure the expected loads can be handled, as currently the arm is unable to lift the expected 1lb loads.
- The elbow rotation is currently constrained by the servo motors that get in the way as forearm approaches the shoulder. The arm needs to be redesigned to allow for a full elbow bend.

There were some code and circuit complications during construction as well:

- ESP32 3.3V were not enough to operate the servos, so voltage shifter circuitry had to be added.
- The servos have some jitter in the current design, adding capacitor and incorporating float values in the code can help minimize it.
- The initial goal to ease the movements in and out, as opposed to having constant rotation speed, needs to be implemented. More research needs to be performed to find solutions as the ledc library does not provide that functionality.
- Design a more stable power supply that can maintain a steady voltage and current for longer.

Conclusion

Designing and constructing the proposed concept presented multiple unique challenges and provided a great learning experience in the field of human arm kinematics, torque balance and transfer, and translating the concept into an actual 3D model. The initial prototype showed which areas need further research and improvement.
Overall, the constructed robotic arm meets the goals set up at the beginning of the design process and helps map a clearer path forward to refining the current prototype and design a more efficient version of a robotic arm.

Resources

1) (n.d.). *Forces and Torques in Muscles and Joints*. Lumenlearning.com. https://courses.lumenlearning.com/suny-physics/chapter/9-6-forces-and-torques-in-muscles-and-joints/
2) (n.d.). *MG995 Servo*. Amazon.com. https://www.amazon.com/dp/B0BYDFF79C?ref=ppx_yo2ov_dt_b_product_details&th=1
3) (n.d.). *Adafruit HUZZAH32 - ESP32 Feather Overview*. Learn.Adafruit.com. https://learn.adafruit.com/adafruit-huzzah32-esp32-feather
4) (n.d.). *LED Control (LEDC)*. Espressif-Docs.Readthedocs-Hosted.com. https://espressif-docs.readthedocs-hosted.com/projects/arduino-esp32/en/latest/api/ledc.html