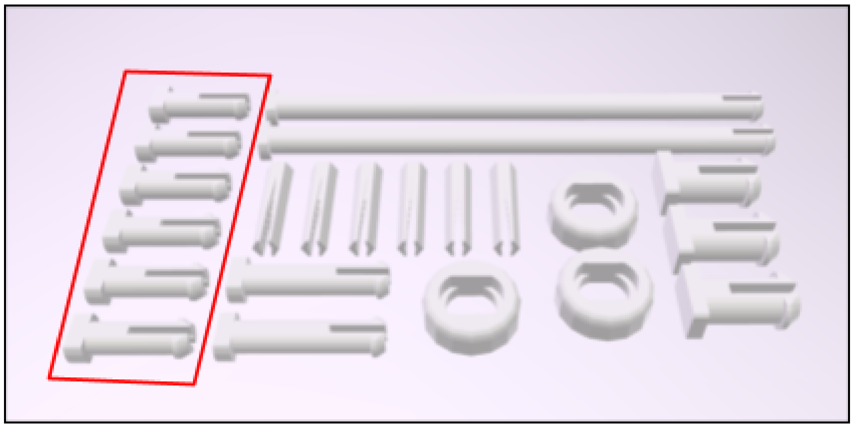
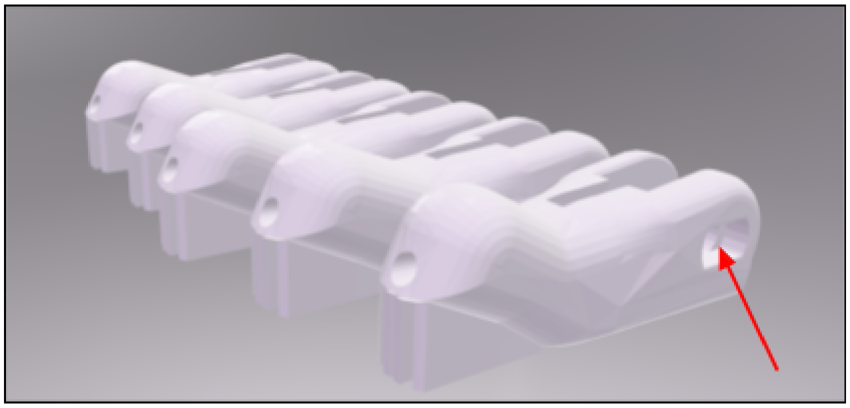
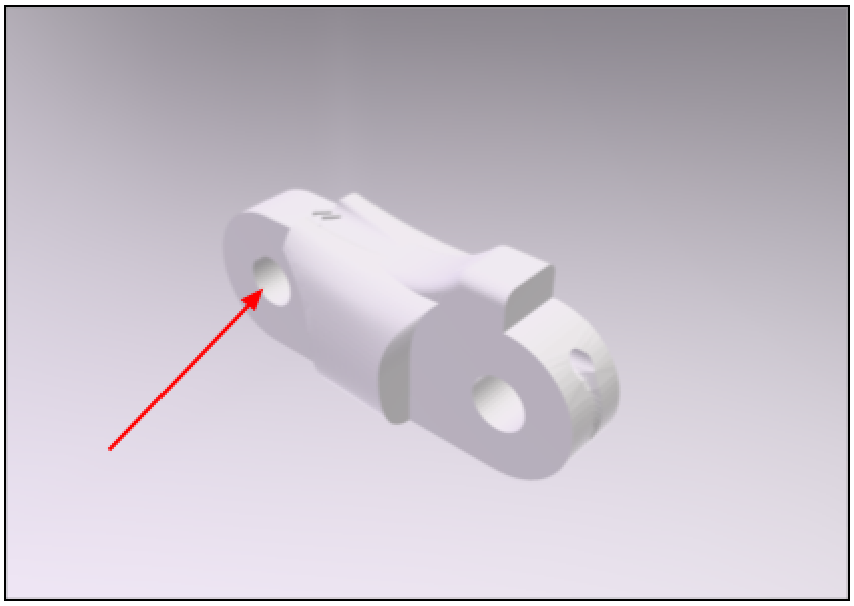
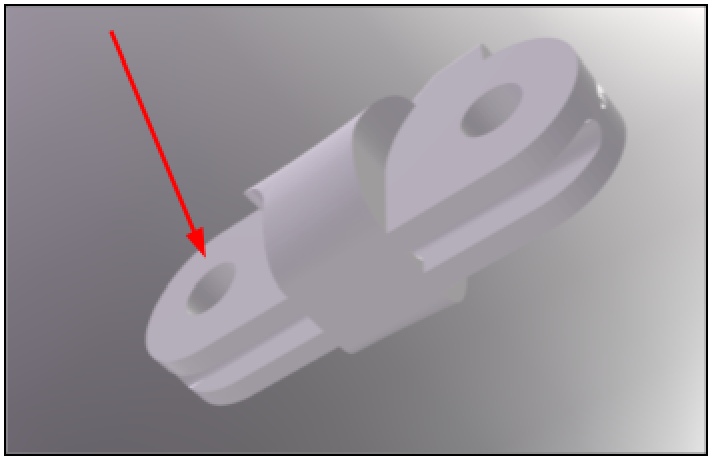
Assembly Documentation for EMG Signal-guided Open Source Prosthetic Hand

Using the pins circled in the image, connect the fingertips to the middle part of the finger where arrows are shown.

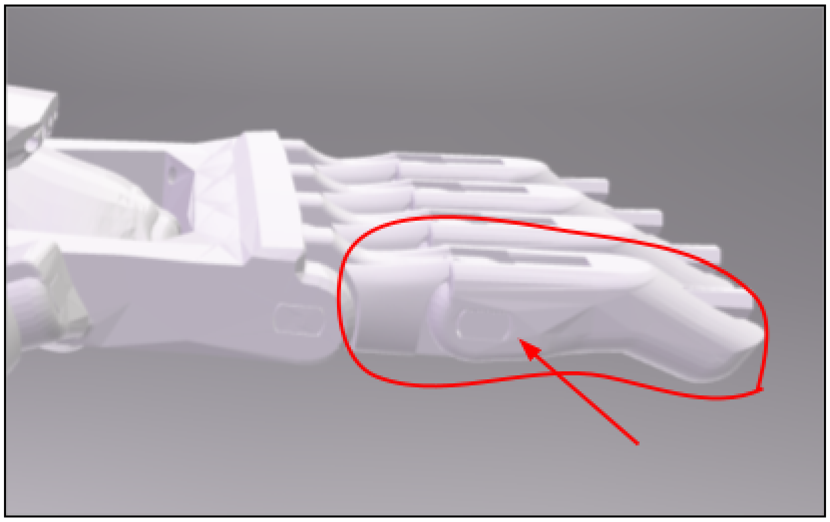


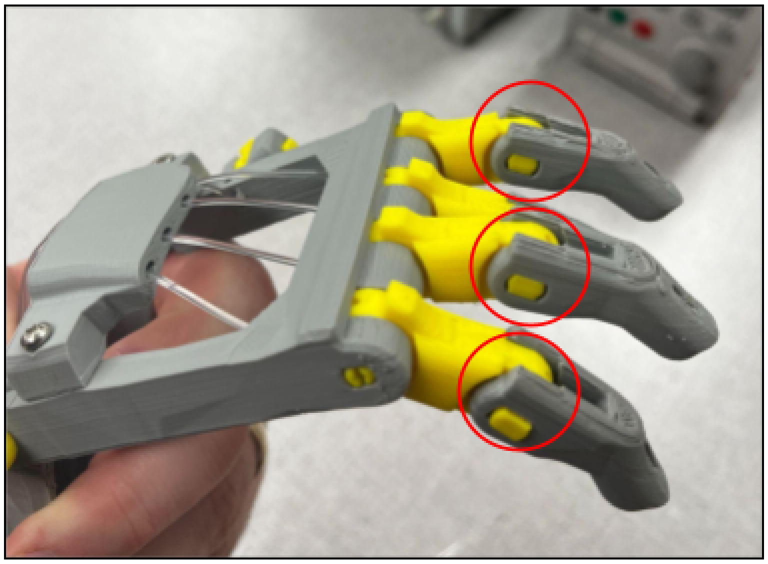






Notice below how it should look:

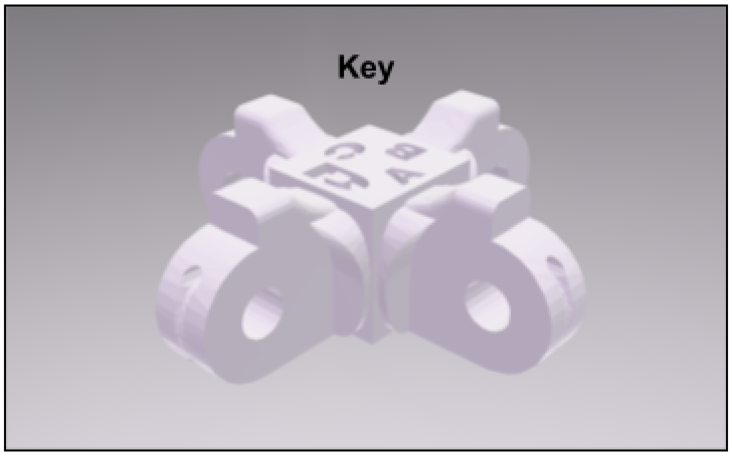


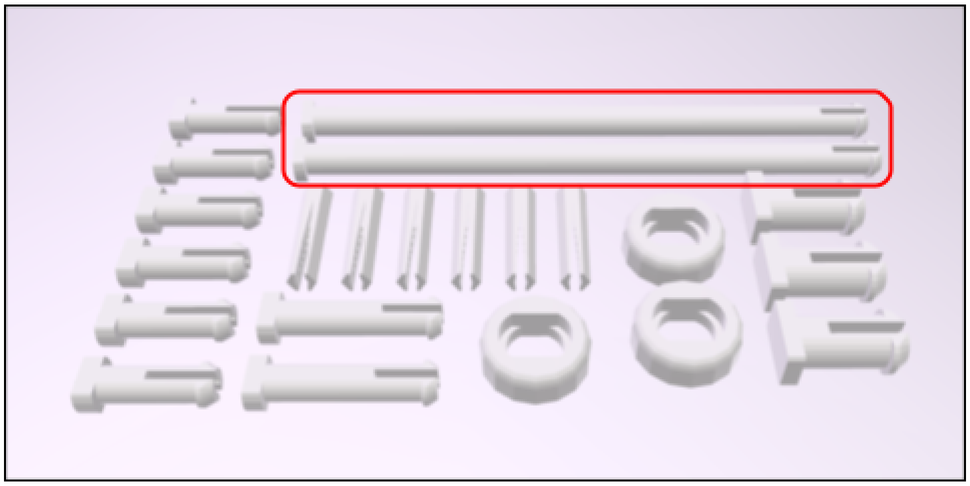


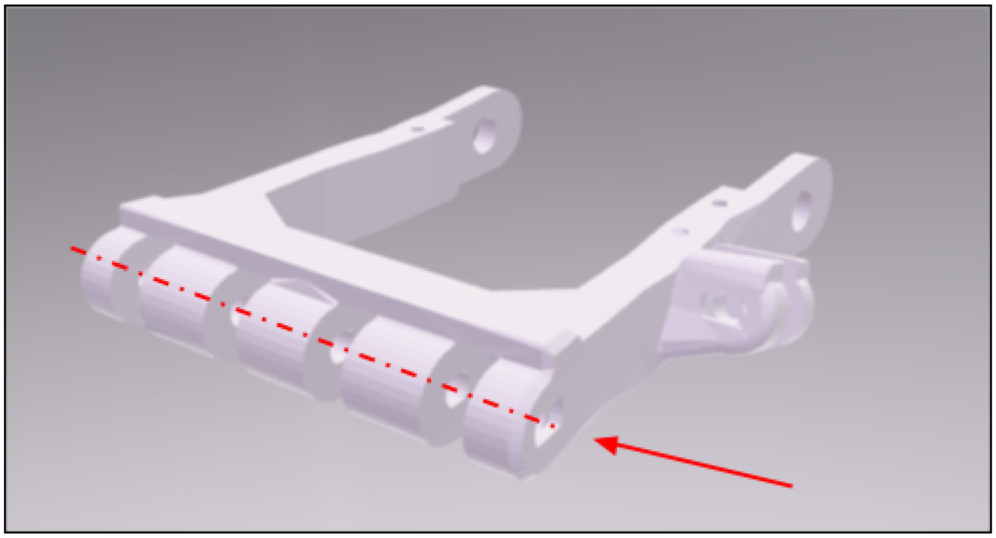
Once you connect all five fingertips to the middle portions of the finger, it's time to attach them to the frame of the hand.

Using the long pins, circled in the image below, connect the fingers to each spot along the frame. (Adjust as needed to ensure pin slides all the way through)
You can use the *Key* to determine which spot is assigned to each finger, or just test out which spot produces the smoothest movement for each finger)

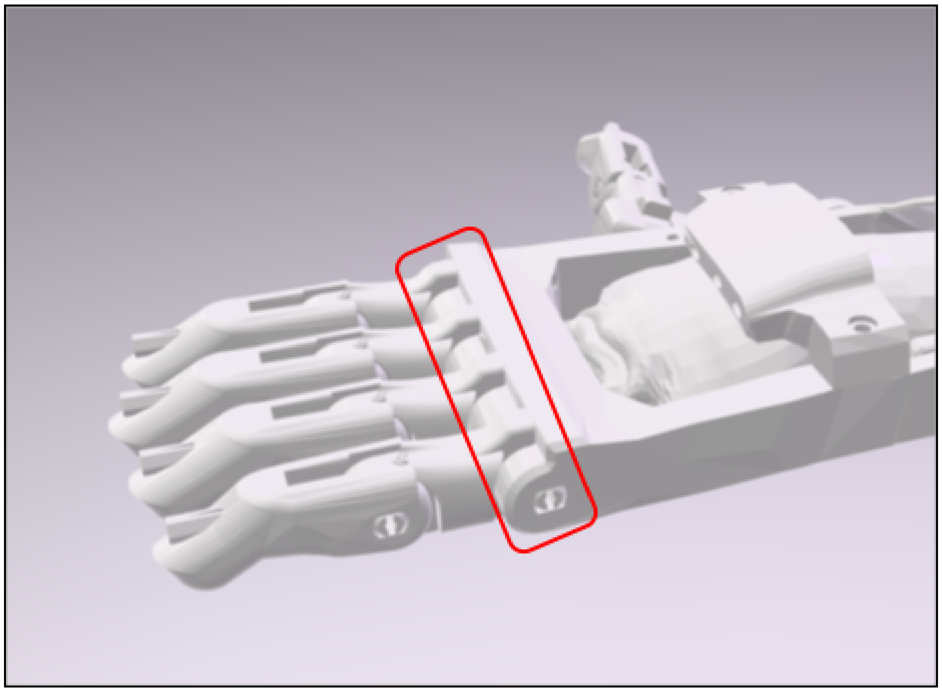
Note: ***Sanding may be needed***

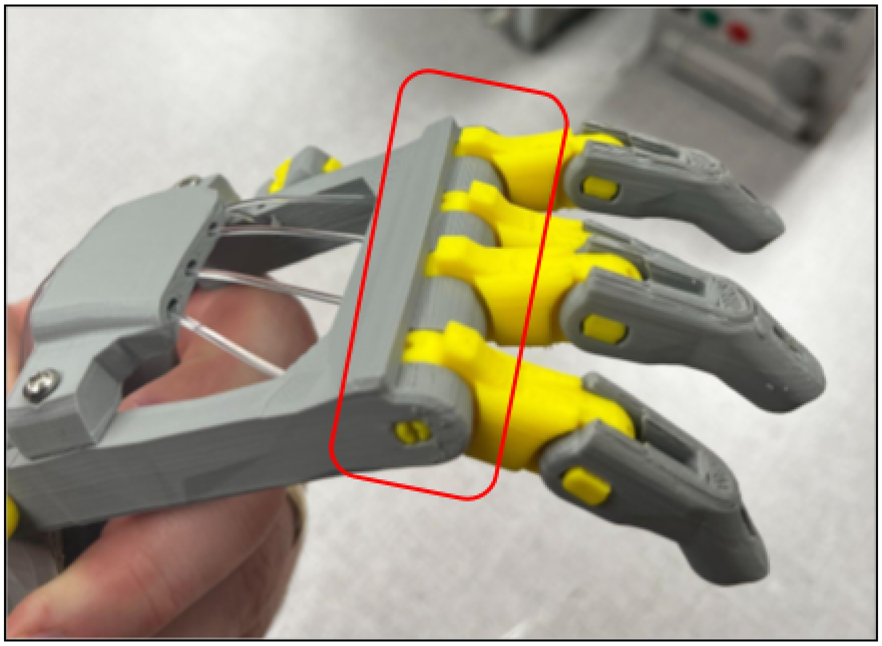






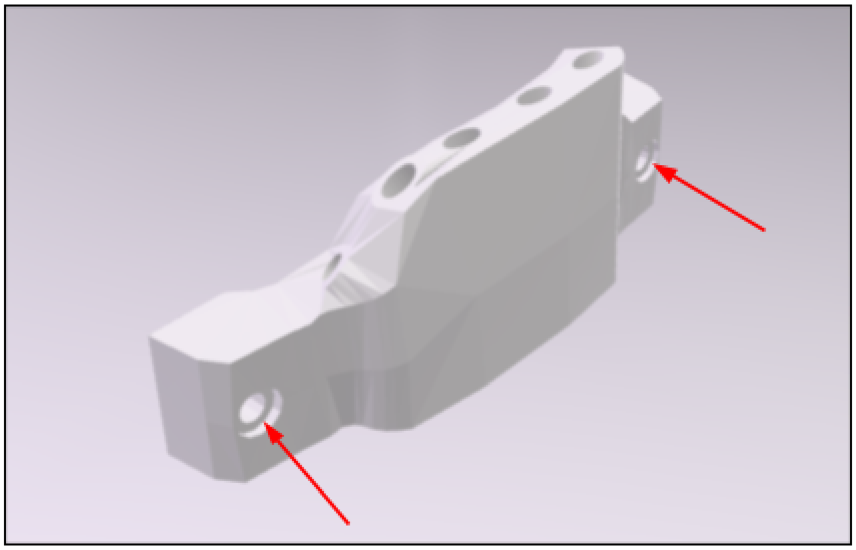
Notice below how it should look:

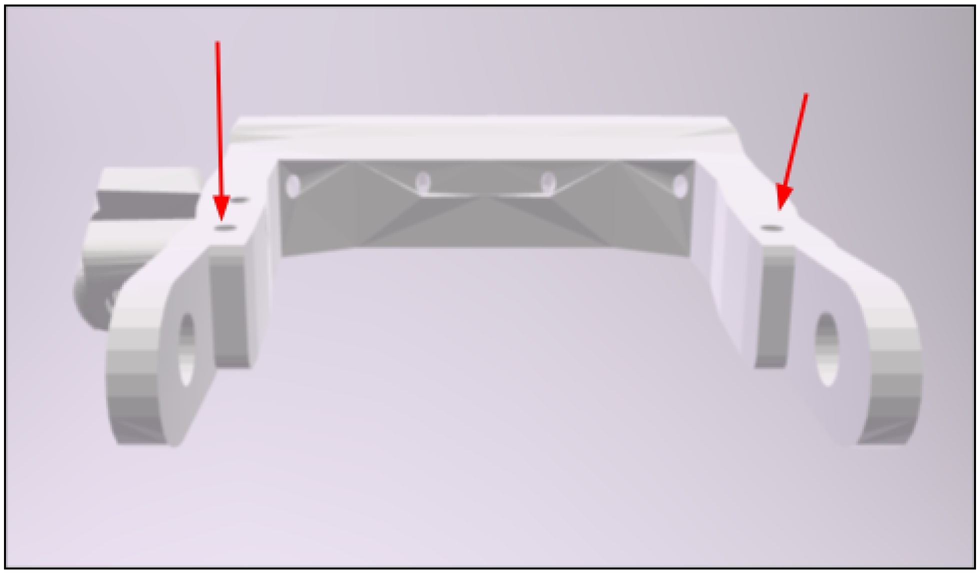


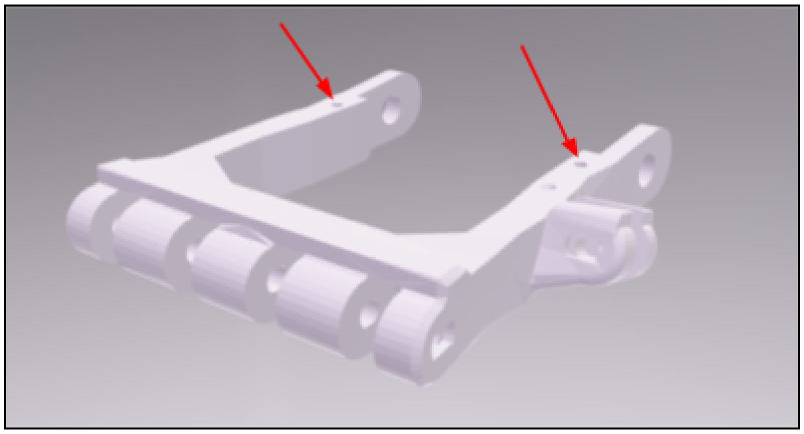


The next step is to attach the cable guide to the frame of the hand.

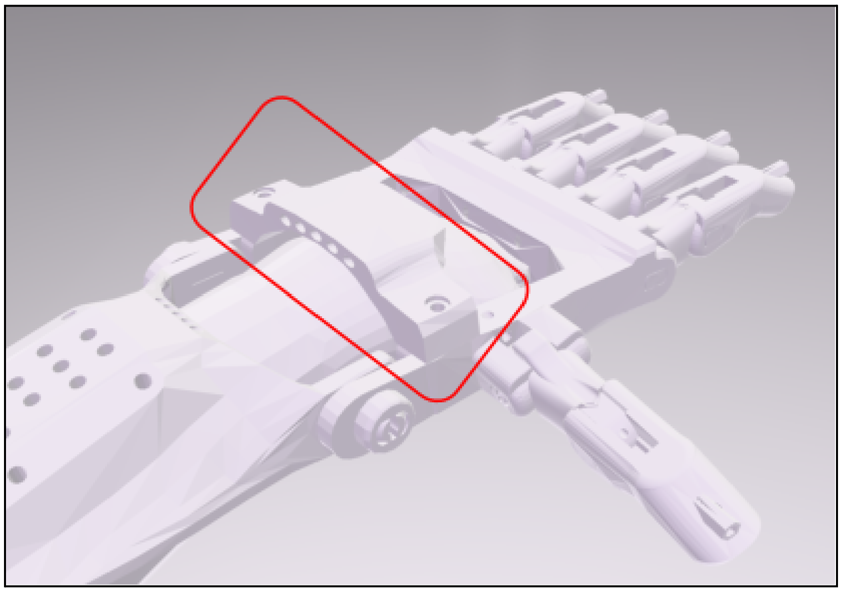
Using small screws, attach the cable guide to the frame at the locations indicated by the arrows.

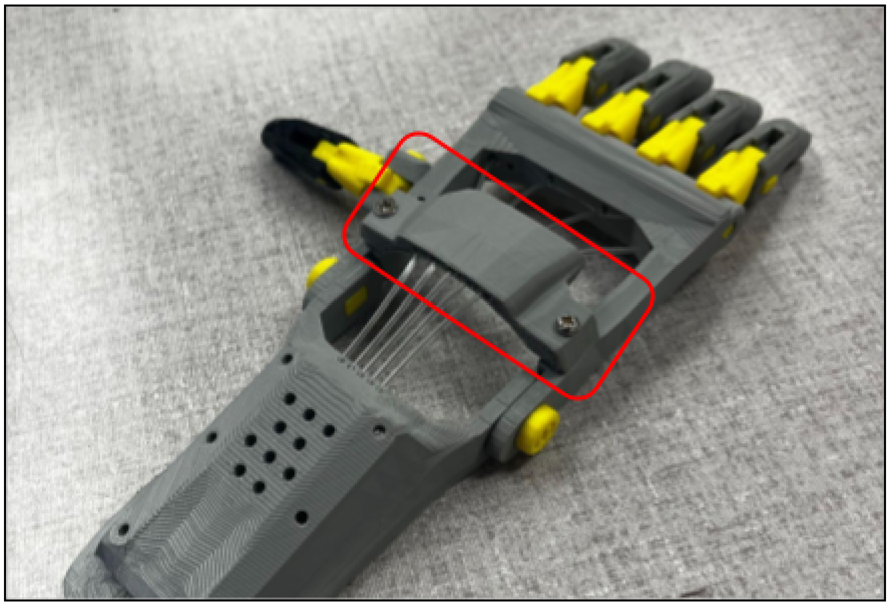






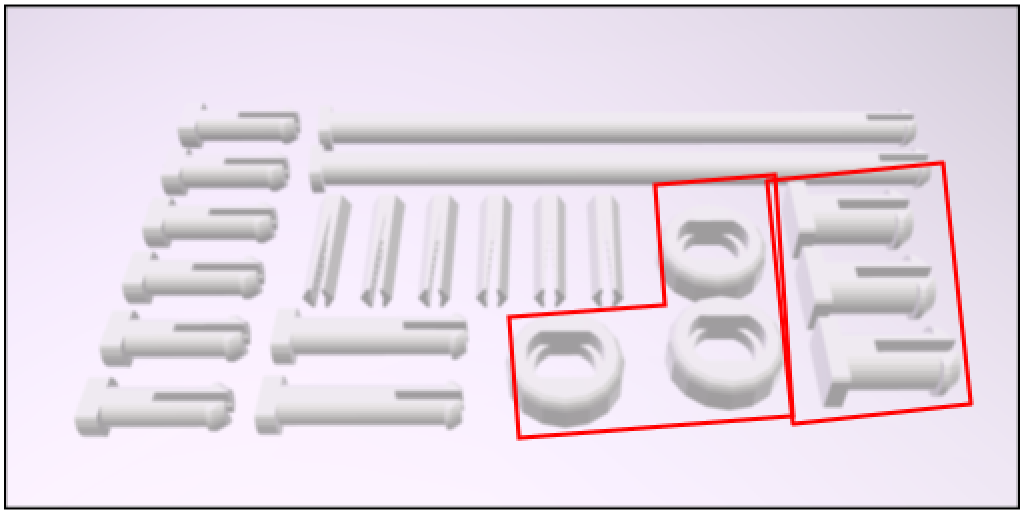
***Note:* Notice the orientation of the cable guide**

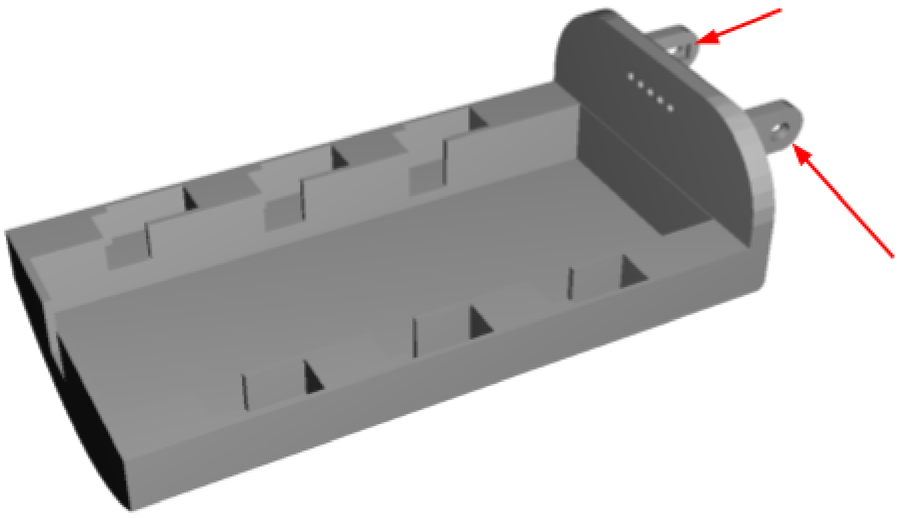




Now, the final step is attaching the housing for the motors and circuit to the hand.

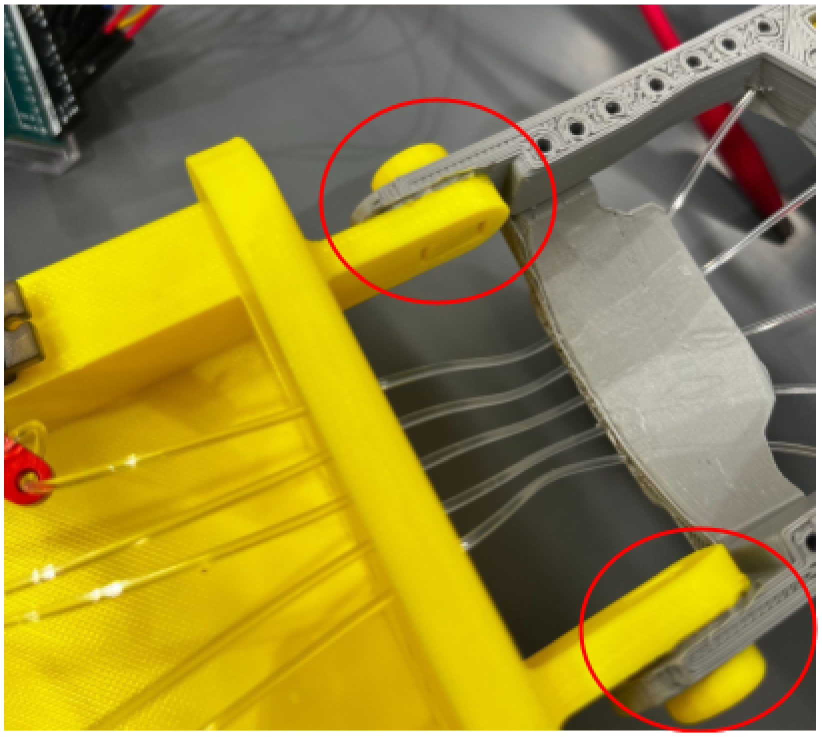
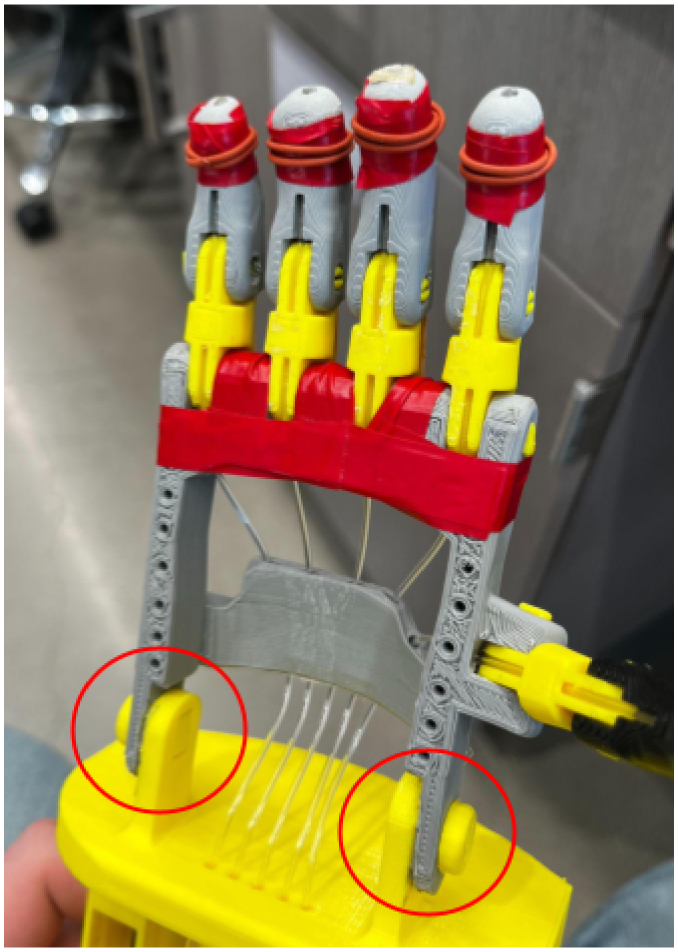
Using the pins and retaining rings boxed in below, connect the ends of the hand frame with the connection arms of the housing at the locations indicated by the arrows.



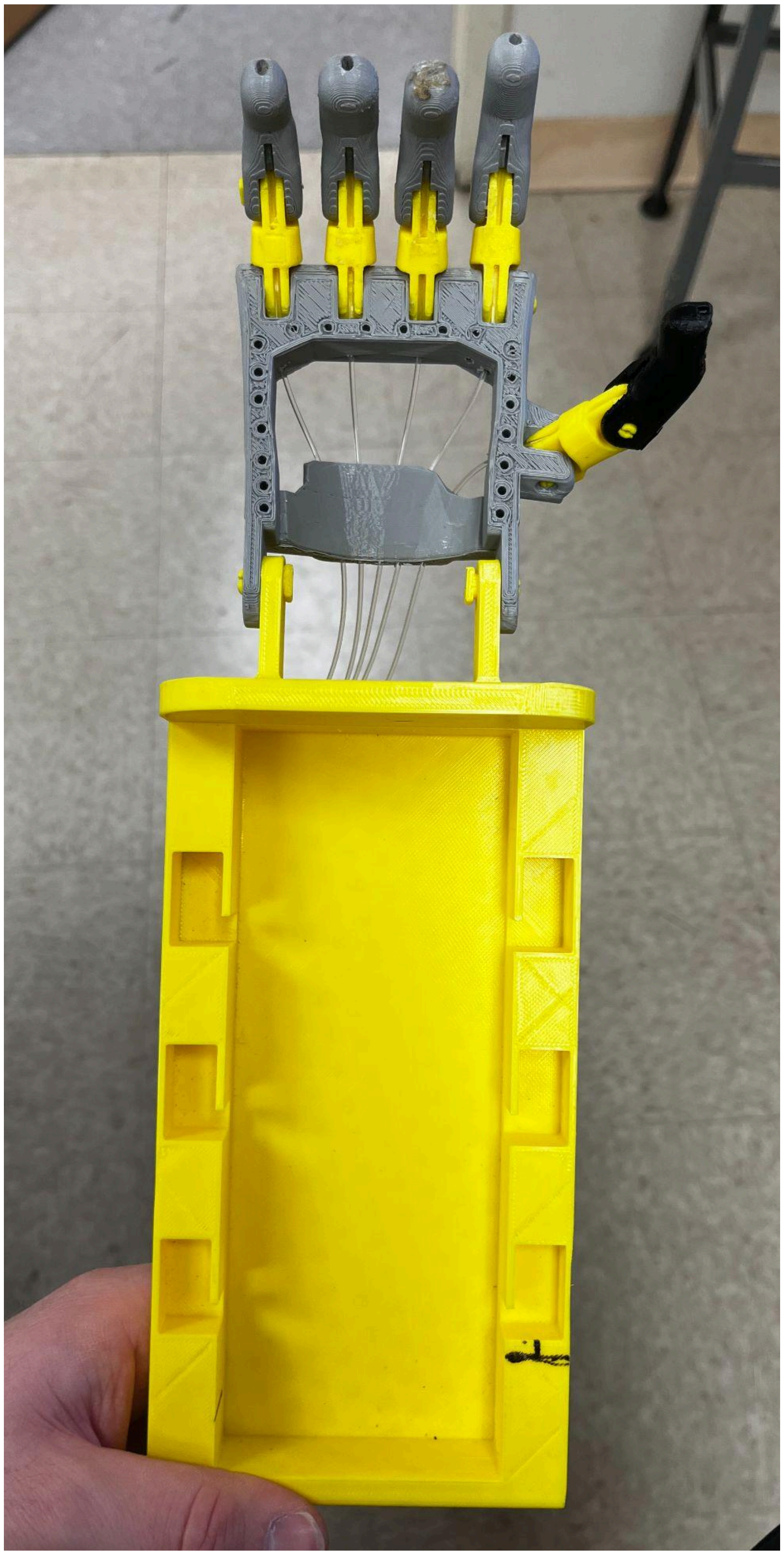


***Note:* The squared side of the pin goes on the inside of the housing connections**

See below what it should look like attached:

Your end result should look like this!🥳



⚡ Electrical & Coding Setup

To get started, first download the **Arduino IDE** from the official website:
https://www.arduino.cc/en/software

You'll also need to install the **Adafruit PWM Servo Driver Library**, which can be found here:
https://github.com/adafruit/Adafruit-PWM-Servo-Driver-Library

Once the software is installed, connect your Arduino to your computer via USB. You'll find two code templates below—one that includes keypad support, and one that does not. Upload the version that suits your setup using the Arduino IDE.

Parts required

Arduino Uno
https://shorturl.at/n6nXW

Adafuit PCA9685 motor driver
https://shorturl.at/SOTls

Mizuei MG90S servo motors (8pack) :
https://shorturl.at/HV0HP

Breadboard and wires:
https://shorturl.at/tiySS
https://shorturl.at/yC6Q7

Voltage Source:
https://shorturl.at/5CEET

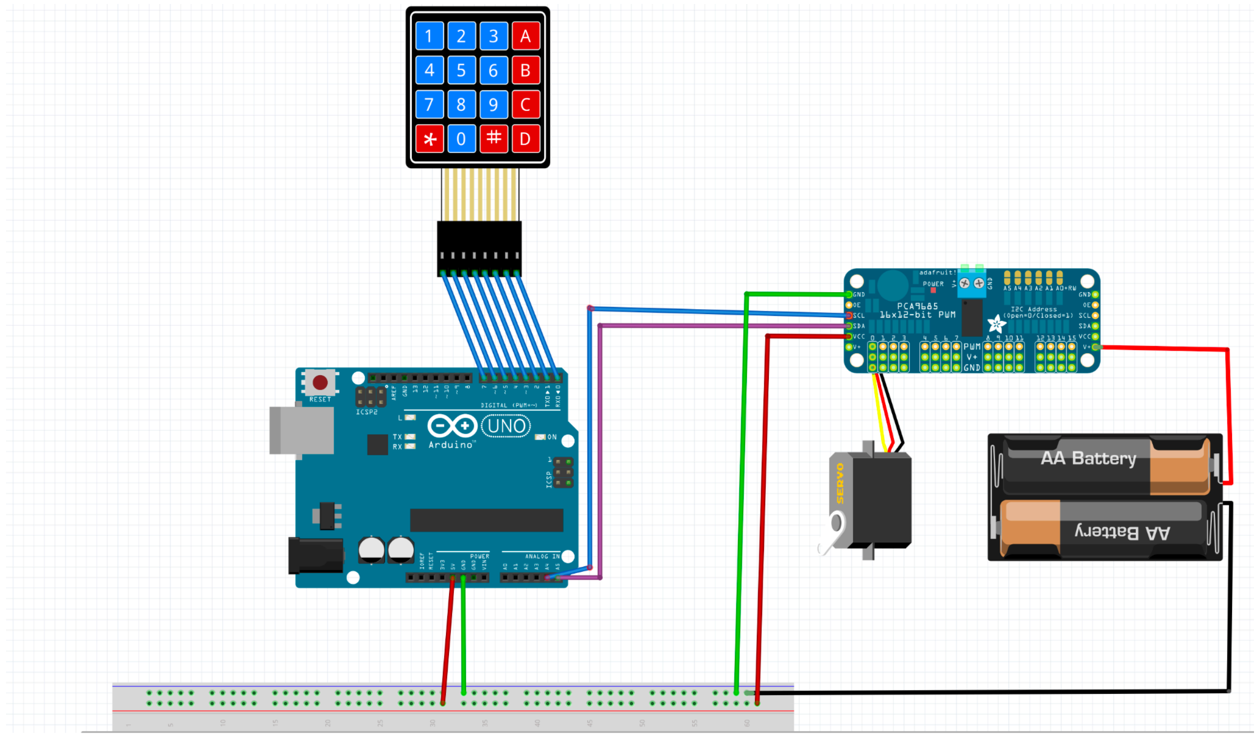
Keypad:
https://shorturl.at/zRaio

Fishing Line:

Servo Motor Heads:

Keypad Model Wiring Diagram

The wiring for the **keypad model** is shown below. If you're using the **non-keypad model**, the wiring will be exactly the same **except without the keypad**. Be sure to follow the correct diagram based on the version you're building.



Code for non-keypad model:

```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>

#define SERVOMIN 0       // Minimum pulse length count (out of 4096)
#define SERVOMAX 500     // Maximum pulse length count (out of 4096)
#define NUM_SERVOS 5     // Total number of servos

Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();
int p[NUM_SERVOS] = {350, 300, 250, 250, 250};  // Initial values for each
motor

void setup() {
  Serial.begin(9600);
  Serial.println("Servo Controller via Serial");
  Serial.println("Type: <motor_index> <pwm_value> (e.g., 2 300)");

  pwm.begin();
  pwm.setPWMFreq(50);  // 50 Hz for analog servos
  delay(10);

  // Initialize servos to initial positions
  for (int i = 0; i < NUM_SERVOS; i++) {
    pwm.setPWM(i, 0, p[i]);
  }
  Serial.println("Initial PWM values set.");
}

void loop() {
  if (Serial.available()) {
    String input = Serial.readStringUntil('\n');
    input.trim();

    int spaceIndex = input.indexOf(' ');
    if (spaceIndex == -1) {
      Serial.println("Invalid format. Use: <motor_index> <pwm_value>");
      return;
    }

    int motorIndex = input.substring(0, spaceIndex).toInt();
    int pwmValue = input.substring(spaceIndex + 1).toInt();

    if (motorIndex < 0 || motorIndex >= NUM_SERVOS) {
      Serial.println("Motor index out of range (0-4).");
      return;
    }

    if (pwmValue < SERVOMIN || pwmValue > SERVOMAX) {
      Serial.println("PWM value out of range (0-500).");
      return;
    }

    p[motorIndex] = pwmValue;
    pwm.setPWM(motorIndex, 0, pwmValue);
    Serial.print("Set motor ");
    Serial.print(motorIndex);
    Serial.print(" to PWM: ");
    Serial.println(pwmValue);
  }
}
```

Keypad Model Code:

```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>
#include <Keypad.h>
```

```
#define SERVOMIN 0  // Minimum pulse length count (out of 4096)
#define SERVOMAX 500  // Maximum pulse length count (out of 4096)
#define NUM_SERVOS 5  // Total number of servos
#define STEP_SIZE 25  // Smaller step size for smoother movement

// Setup for the PWM Servo Driver
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();
int p[NUM_SERVOS] = {350, 300, 250, 250, 250};  // Initial values for each
motor
int currentMotor = 0;  // Index for the currently active motor

// Keypad Setup
const byte ROWS = 4;  // Four rows
const byte COLS = 4;  // Four columns
char hexaKeys[ROWS][COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
byte rowPins[ROWS] = {9, 8, 7, 6};  // Connect to the row pinouts of the
keypad
byte colPins[COLS] = {5, 4, 3, 2};  // Connect to the column pinouts of
the keypad
Keypad keypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS,
COLS);

void setup() {
  Serial.begin(9600);
  Serial.println("Servo Test - Press '1' to rotate forward, '2' to rotate
backward, '0' to switch motor");
  Serial.println("Press '5' to move motors 2 & 3 forward, '6' to move them
backward");

  pwm.begin();
  pwm.setPWMFreq(50);  // Set frequency to 50Hz for servo motors
  delay(10);

  // Initialize PWM values for all motors to specified positions
  for (int i = 0; i < NUM_SERVOS; i++) {
    pwm.setPWM(i, 0, p[i]);  // Set initial position for each motor
  }
  Serial.println("Initial PWM values set.");
}

void loop() {
```

```
char key = keypad.getKey();

if (key) {
  Serial.print("Key Pressed: ");
  Serial.println(key);

  if (key == '1') {  // Rotate active motor forward
    if (p[currentMotor] + STEP_SIZE <= SERVOMAX) {
      p[currentMotor] += STEP_SIZE;
      pwm.setPWM(currentMotor, 0, p[currentMotor]);
      Serial.print("Motor ");
      Serial.print(currentMotor);
      Serial.print(" rotated forward, PWM: ");
      Serial.println(p[currentMotor]);
    }
  }
  else if (key == '2') {  // Rotate active motor backward
    if (p[currentMotor] - STEP_SIZE >= SERVOMIN) {
      p[currentMotor] -= STEP_SIZE;
      pwm.setPWM(currentMotor, 0, p[currentMotor]);
      Serial.print("Motor ");
      Serial.print(currentMotor);
      Serial.print(" rotated backward, PWM: ");
      Serial.println(p[currentMotor]);
    }
  }
  else if (key == '5') {  // Move motors 2 & 3 forward together
    if (p[2] + STEP_SIZE <= SERVOMAX && p[3] + STEP_SIZE <= SERVOMAX) {
      p[2] += STEP_SIZE;
      p[3] += STEP_SIZE;
      pwm.setPWM(2, 0, p[2]);
      pwm.setPWM(3, 0, p[3]);
      Serial.println("Motors 2 & 3 moved forward");
    }
  }
  else if (key == '6') {  // Move motors 2 & 3 backward together
    if (p[2] - STEP_SIZE >= SERVOMIN && p[3] - STEP_SIZE >= SERVOMIN) {
      p[2] -= STEP_SIZE;
      p[3] -= STEP_SIZE;
      pwm.setPWM(2, 0, p[2]);
      pwm.setPWM(3, 0, p[3]);
      Serial.println("Motors 2 & 3 moved backward");
    }
  }
  else if (key == '0') {  // Switch to the next motor
    currentMotor++;
    if (currentMotor >= NUM_SERVOS) {
        currentMotor = 0;
      }
      Serial.print("Switched to motor ");
      Serial.println(currentMotor);
    }
  }

  delay(100);
}
```

⚠️ Motor Position Warning & Setup

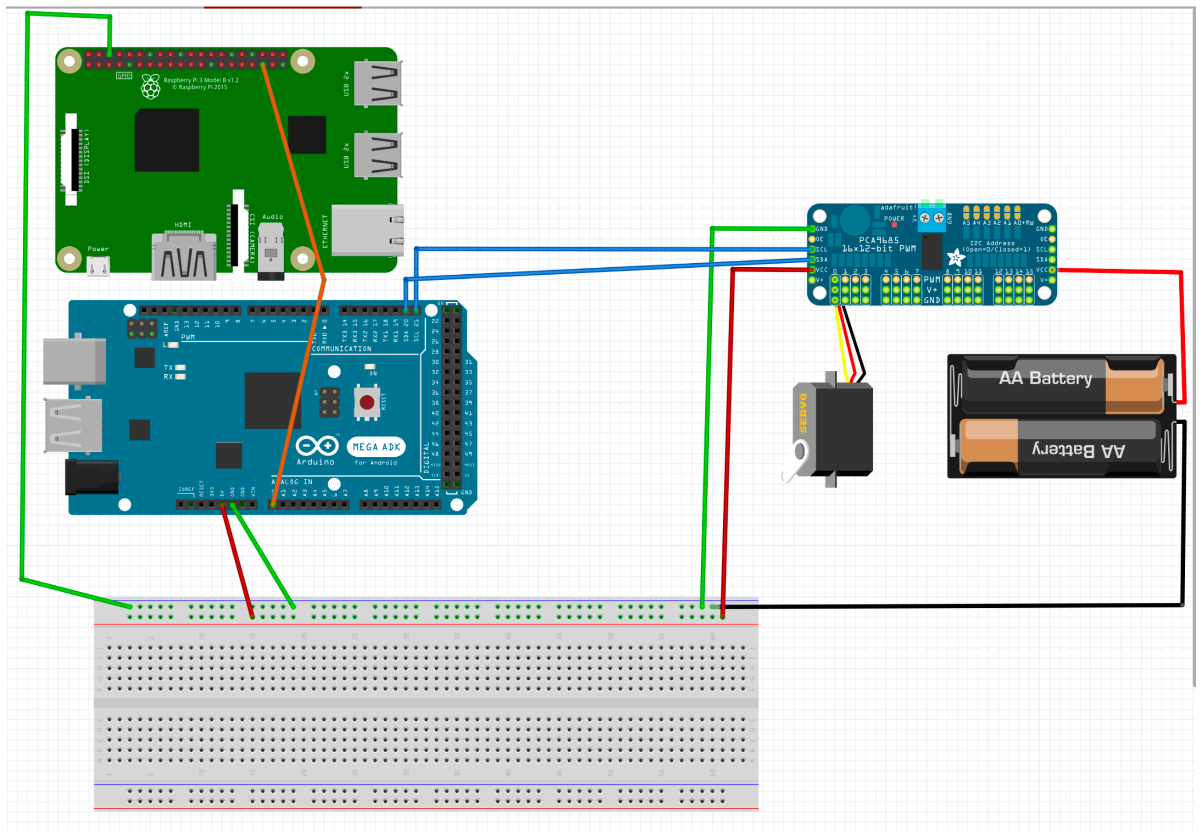
These motors have low torque, and forcing them to move too far in one direction can cause permanent damage. To prevent this, each motor’s starting position must be carefully set in the code.

Adjust the values in the line below to define the initial position for each motor:

```
int p[NUM_SERVOS] = {350, 300, 250, 250, 250};  // Initial values for each
```

Each number corresponds to a motor in order—**350** is for Motor 1, **300** for Motor 2, and so on. Tune these values as needed to ensure safe movement on startup.

Incorporating EMG signal using Raspberry Pi:



The Raspberry Pi is used to output direct PWM values to the Arduino to facilitate servo control. Pre-recorded EMG signals can be fed or read directly and processed into the Pi. Please review this part under the EMG section.

Pi code used to control Arduino (python):

```
import serial
import time

# Establish real serial connection
arduino = serial.Serial('/dev/ttyUSB0', 9600, timeout=1)  # Adjust with correct port

time.sleep(2)

def send_pwm(motor, value):
    """Send PWM command to the motor."""
    command = f"{motor} {value}\n"
    arduino.write(command.encode())
    time.sleep(0.1)

try:
    while True:
        motor = int(input("Enter motor index (0-4): "))
        if motor not in range(0, 5):
            print("Invalid motor index. Choose between 0-4.")
            continue

        pwm_value = int(input("Enter PWM value (0-500): "))
        if not (0 <= pwm_value <= 500):
            print("PWM value must be between 0 and 500.")
            continue

        send_pwm(motor, pwm_value)

except KeyboardInterrupt:
    print("\nExiting...")
    arduino.close()
```