TABLE OF CONTENTS

PAGE	TITLE	DATE
2	Design Brief	10/21/2025
3	Arm Design	10/21/2025
4	Flow chart	10/22/2025
5	Finger Drawing Design	10/30/2025
6	1# Circuit code design	11/4/2025
7	2# Circuit code design	11/6/2025
8	1# Circuit Design	11/6/2025
9	1# On shape Design	11/7/2025
10	Onshape basic hand design	11/13/2025
11	Current Design Flaws	12/2/2025
12	Potential Features	12/2/2025
13	Potential Features	12/2/2025
14	Arduino Code V1	12/3/2025
15	Arduino Code V1	12/3/2025
16	Arduino Code V2	12/3/2025
17	Arduino Code V2	12/3/2025
18	Arduino Code V2	12/3/2025
19	How We Used the Engineering Design Process	12/3/2025
20	How We Used the Engineering Design Process	12/3/2025
21		

TABLE OF CONTENTS

PAGE	TITLE	DATE
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		

DESIGNED BY:______________________________ DATE:____________________

WITNESSED BY:______________________________ DATE:____________________

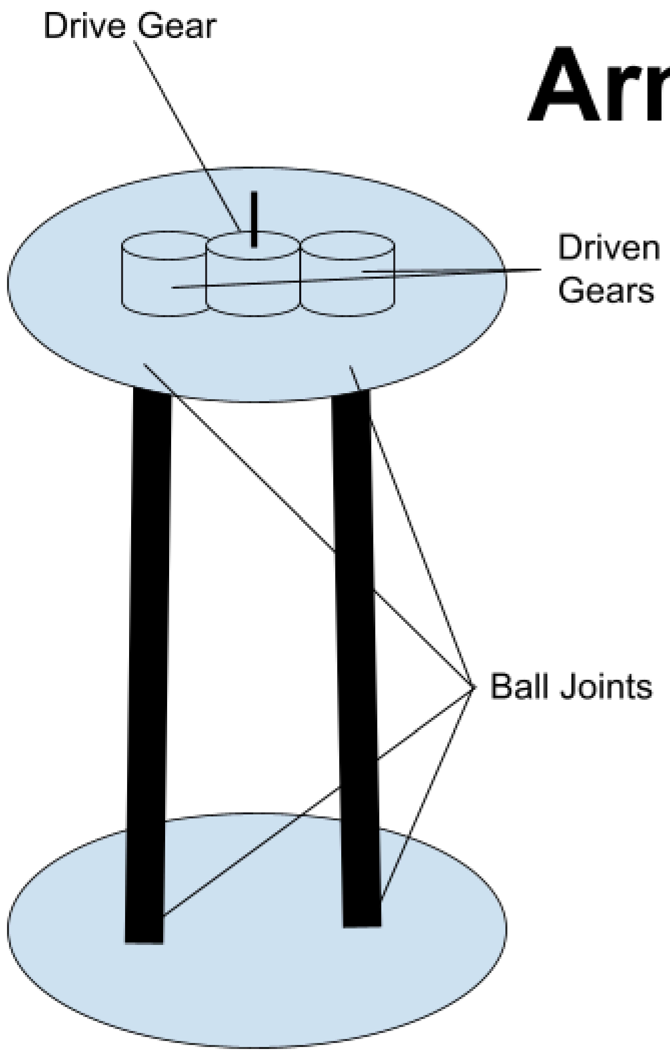
Design Brief

Client:	I.M. Terrell STEM Department Mascot Committee
Designer:	Conner Houser, Frankie Reale, Gabriel Arntz, Caroline King
Problem Statement:	Design and build an animatronic mascot that responds to stimuli and uses servo motors for movement and LEDs for lighting effects, controlled by an Arduino microcontroller. The animatronic should demonstrate at least two distinct motions (such as head turning and nodding) and include lighting effects (such as blinking eyes or accent lights). The design must be safe, functional, and fit within the size limits of the classroom 3D printer. The design will be lightweight, functional, and use no more than 100 grams of filament for 3D-printed parts. The project must follow the Engineering Design Process and be completed by the end of the semester.
Design Statement:	Our design we want to build a hand/arm that is capable to close and open and rotate its wrist.
Constraints:	- 2 motors - 200 watts - 1 Arduino Uno - All grounds must be connected - No exposed wiring - Must fit within 200mm x 200mm x 200mm - <100g of PLA filament - Must have LED lighting - Must perform 2 smooth, repeatable movements - Code must be commented and compile without errors
Deliverables:	- Team notebook - Finished animatronic

DESIGNED BY:________________________________ DATE:________________

WITNESSED BY:________________________________ DATE:________________

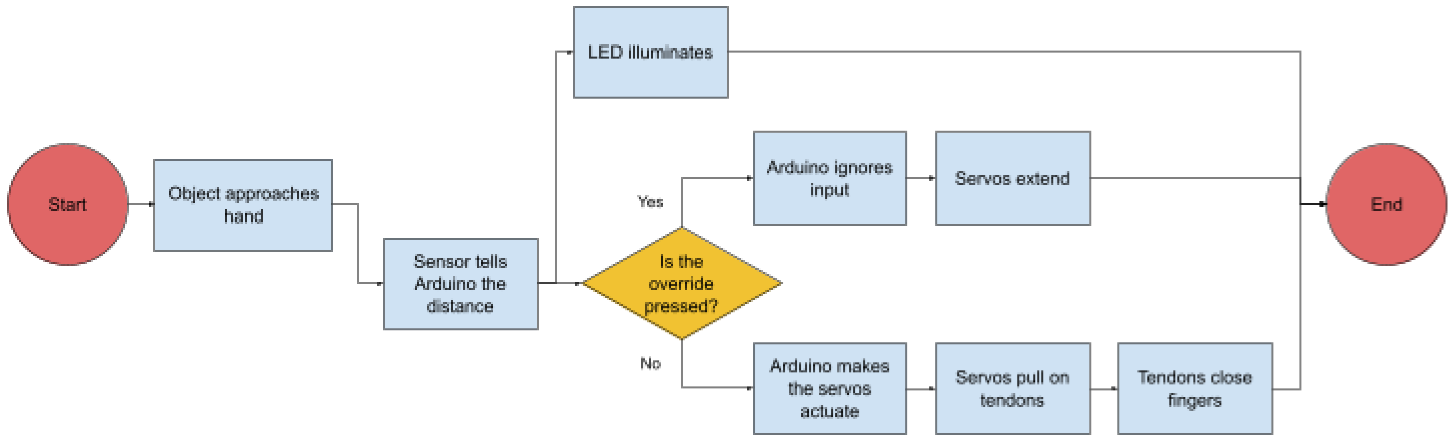
Arm Design



DESIGNED BY:	DATE:
WITNESSED BY:	DATE:

Flow Chart

Ideas:
- In-arm servos linked to finger joints by elastic tendons
  - Bi-directional (one for each direction) tendons?
- Aluminum extrusion forearm frame
- Zip-tied servos in arm frame
- Palm-mounted distance sensor to trigger grip motion
- Override button for grip return
- Fingertip grips:
  - Electrical tape
  - 3d printed texture
- Axes of rotation:
  - Pulley-controlled wrist pitch
  - Tendon-controlled knuckles
  - Thumb metacarpal rotation
  - Spring return joints
  - Some amount of finger yawing, not servo-controlled
- Perfboard instead of breadboard



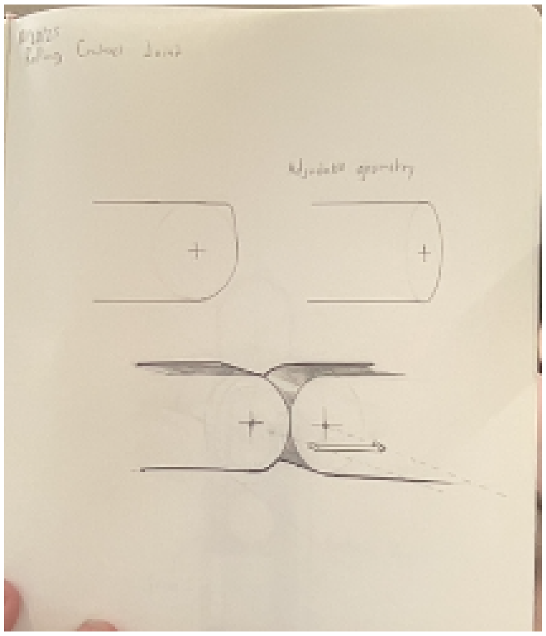
DESIGNED BY:______________________ DATE:______________

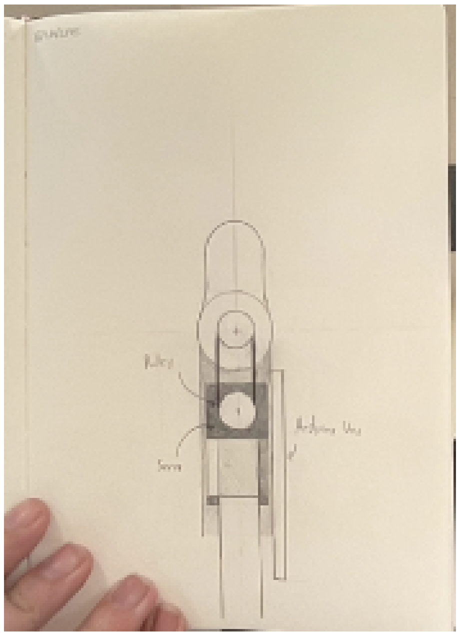
WITNESSED BY:______________________ DATE:______________

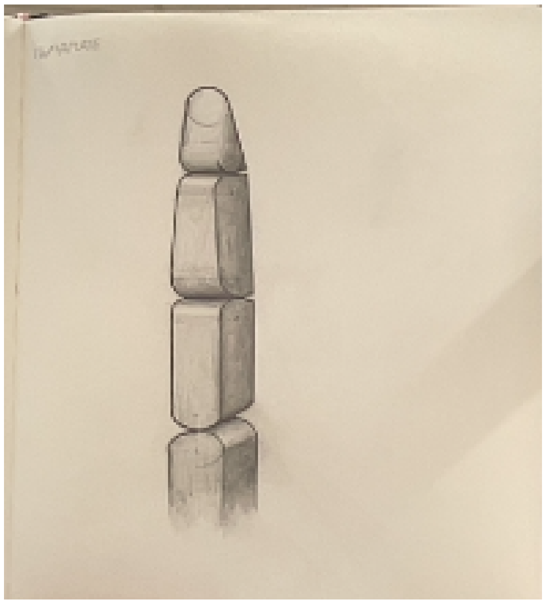
DESIGNED BY:____ DATE:____________________

WITNESSED BY:____ DATE:____________________

Finger Drawing Design







DESIGNED BY:____ DATE:____________________

WITNESSED BY:____ DATE:____________________

1# Circuit Code Design

```
#include <Servo.h>

Servo servo_9;

int cm = 0;

long readUltrasonicDistance(int triggerPin, int echoPin)
{
 pinMode(triggerPin, OUTPUT);
 digitalWrite(triggerPin, LOW);
 delayMicroseconds(2);
 digitalWrite(triggerPin, HIGH);
 delayMicroseconds(10);
 digitalWrite(triggerPin, LOW);
 pinMode(echoPin, INPUT);
 return pulseIn(echoPin, HIGH);
}

void setup()
{
 Serial.begin(9600);
 servo_9.attach(9, 500, 2500);
 pinMode(8, OUTPUT);
 pinMode(7, OUTPUT);
 pinMode(6, INPUT);
}

float timer = 0;

float lastCm = 0;

void loop()
{
 cm = 0.5 * readUltrasonicDistance(7, 4);
 lastCm = (lastCm - cm) / 10; // Interpolate position
 cm = abs(cm-lastCm) > 30 ? cm : lastCm; // Eliminate jittering
 servo_9.write(cm);
 Serial.println(cm);
 /*
 int angle = (sin(timer) + 1) * 90;
 servo_9.write(angle);
 digitalWrite(8, cm<50 ? HIGH : LOW);
 timer+=0.1;
 Serial.println(angle);
 */
  delay(25);
}
```

DESIGNED BY:______________________________ DATE:______________

WITNESSED BY:______________________________ DATE:______________

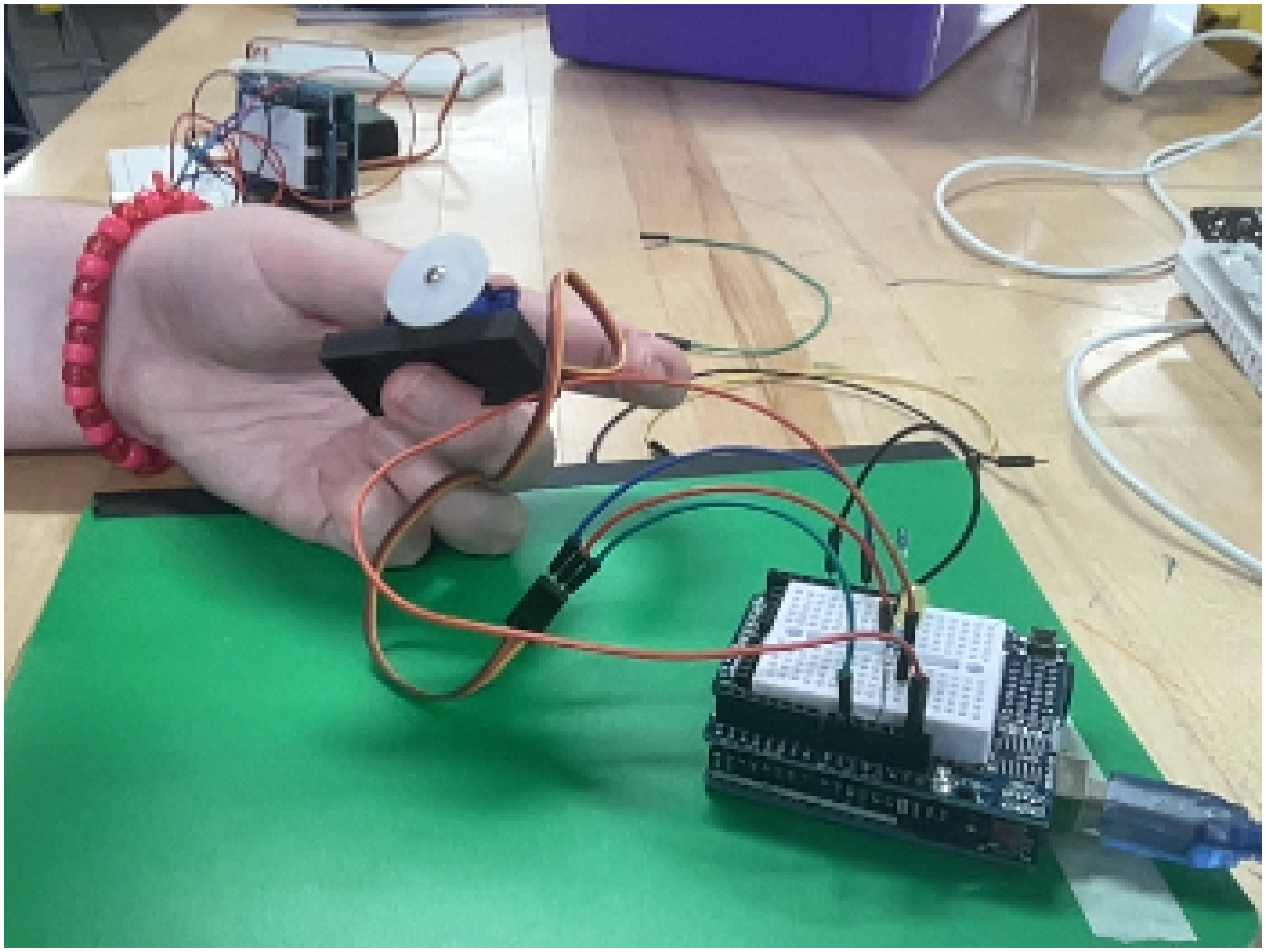
2# Circuit code design

```
#include <Servo.h>

Servo servo9;

void setup() {
 // put your setup code here, to run once:
 servo9.attach(9);
 Serial.begin(9600);
 pinMode(11, OUTPUT);
}

void readServoStress(int stepCount) {
 int stress;
 int stressValues[stepCount];
 for (int i = 0; i < stepCount; i++) {
   stressValues[i] = 1023 - analogRead(A0);
   stress += stressValues[i];
   delay(1);
 }
 stress /= stepCount;

 Serial.println(stress);
 return stress;
}

void loop() {
 // put your main code here, to run repeatedly:
 for (int i = 0; i < 180; i++) {
   servo9.write(i);
   readServoStress(1);
   delay(24);
 }
 servo9.write(0);
}
```

DESIGNED BY:____ DATE:__________________

WITNESSED BY:_______________________________________ DATE:__________________

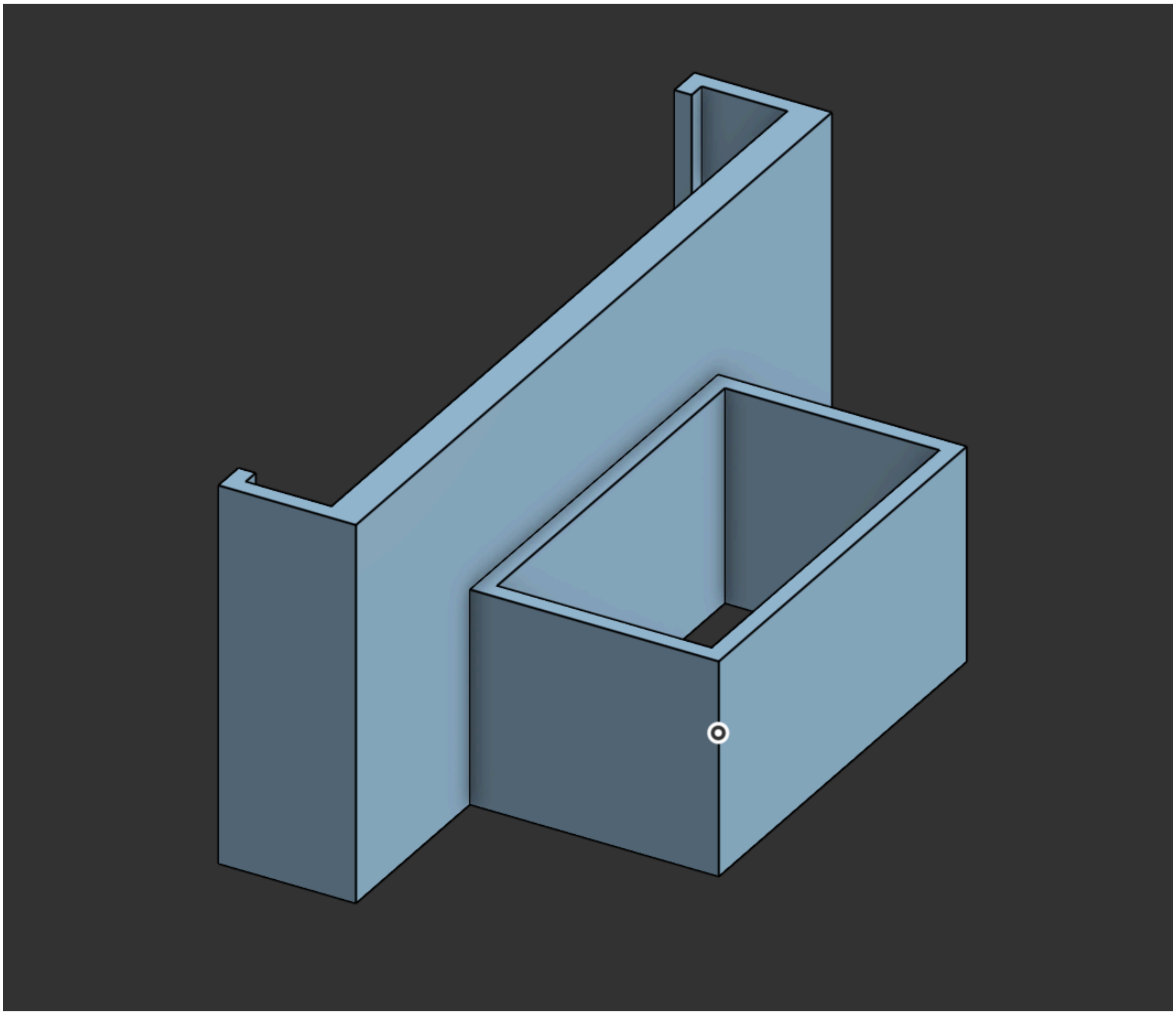
1# Circuit design



DESIGNED BY:____ DATE:____________________

WITNESSED BY:____ DATE:____________________

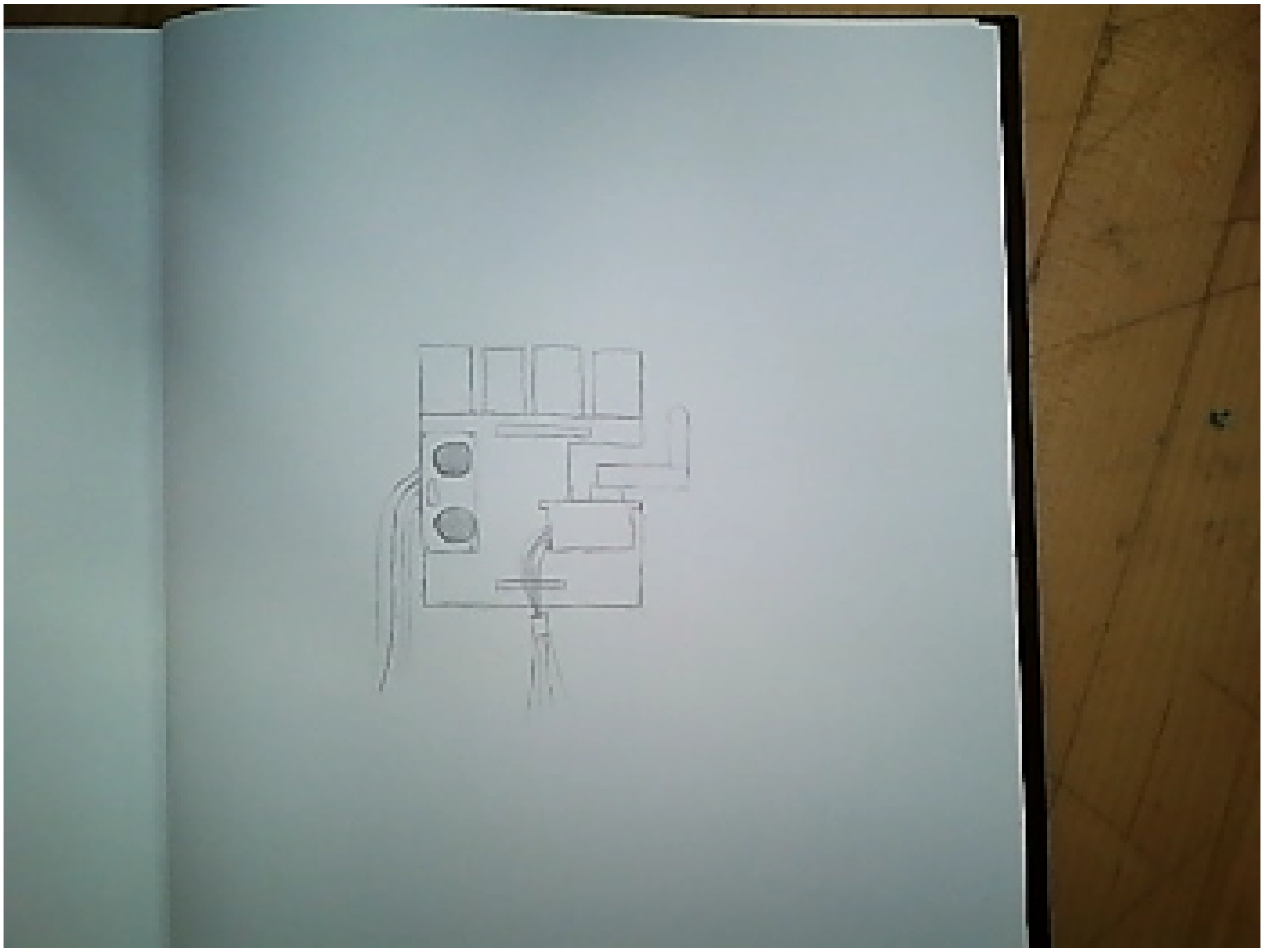
1# On shape Design



DESIGNED BY:____ DATE:________________________

WITNESSED BY:____ DATE:________________________

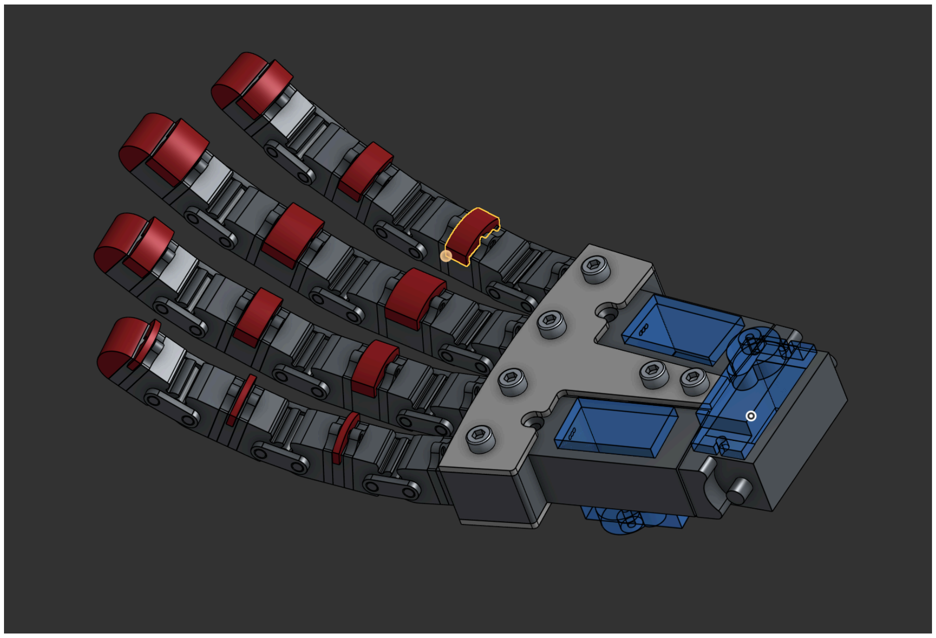
Hand Drawing Design

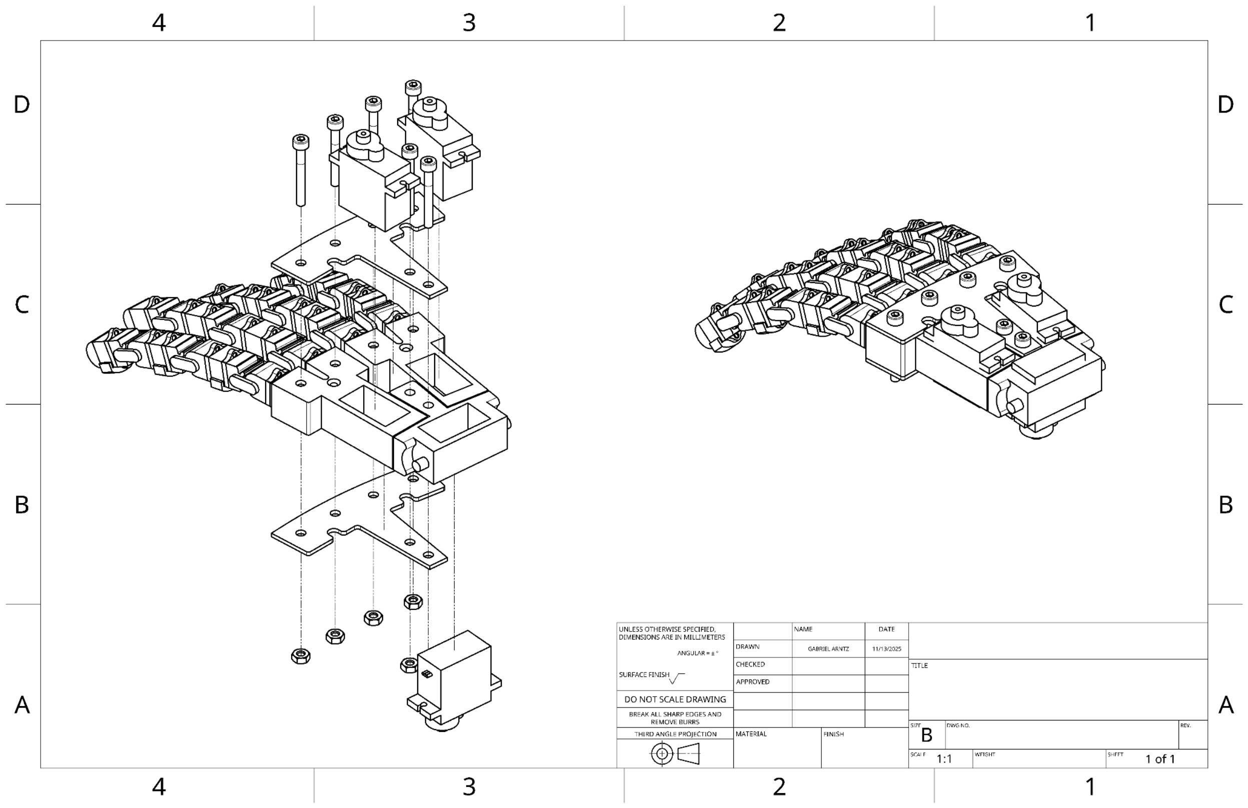


DESIGNED BY:______________________________ DATE:______________

WITNESSED BY:______________________________ DATE:______________

Onshape Basic Hand Design





UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN MILLIMETERS ANGULAR = ± ° SURFACE FINISH		NAME	DATE
	DRAWN	GABRIEL ARNTZ	11/13/2025
	CHECKED		
	APPROVED		
DO NOT SCALE DRAWING			
BREAK ALL SHARP EDGES AND REMOVE BURRS			
THIRD ANGLE PROJECTION	MATERIAL	FINISH	

TITLE		
SIZE B	DWG NO.	REV.
SCALE 1:1	WEIGHT	SHEET 1 of 1

DESIGNED BY:____ DATE:____________________

WITNESSED BY:____ DATE:____________________

Current Design Flaws:
Servo horns collide when moving. Potential fixes:
- Stagger movement so one finger moves at a time
- Replace return tendon with a spring to allow one-sided servo horns to be used
- Replace servo horns with a winding drum (could be imbedded in the palm to reduce size, needs extra work to be done)

Servo wires are messily routed and loose. Potential fixes:
- Use a zip tie imbedded in the wrist to route wires
- Add a printed loop to the palm to route wires through
- Add holes to the palm to route wires along the back of the hand
- Add recessed channels to the hand for the wires to sit in

New Ideas:
Replace outer tendon with elastic return springs
- Fixes uneven motion
- Creates more tension, increasing strength
- Allows one-sided servo horns or coiling drums to be used, decreasing space and removing the collision problem

Replace posts on the finger joints with screws
- Decreases innacuracy due to improper assembly
- Increases modularity, making it easier to replace parts

Remove thumb for now
- Currently unnecessary, could simplify design and allow more time for other parts to be developed
- Can be added later on

Potential Functions:
Gripping
- Detect servo load to close fingers when force is applied to the fingers
- Detect servo stall and call `servo.disconnect()` to prevent damage

Rock Paper Scissors
- Press a button to activate
- LED blinks for ~2 seconds
- Hand moves to random position (hand open, hand closed, scissors position)

Potential Future Features
Thumb
- Allows complex gripping
- Would require more servos and electronics

Custom PCB

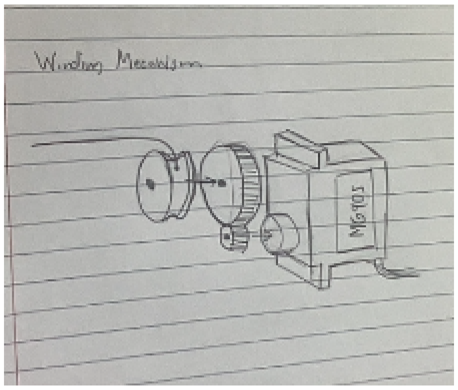
DESIGNED BY:________________________ DATE:________________

WITNESSED BY:________________________ DATE:________________

- Much more reliable than a breadboard
- Easier to detect and prevent damage

Replace servos with MG90S servos with a gearbox and winding drum

- Metal gears in the MG90S provide more torque and help prevent damage
- Use a gear increaser to speed up winding and a drum to wind the tendon rather than using levers which are less durable due to the decreased mechanical advantage of the system
- Since they would be mounted facing forward and have smaller rotating pieces, one could be used for each finger
- Imbedding the winding drums in the palm like this could be safer and more durable because they are less exposed to outside collisions
- The drums could be sandwiched on both sides by the hand part, providing more strength



Touch sensing using LEDs

- Imbed colored LEDs in the finger segments/tips
- Use a photoresistor with a color filter to detect reflected light from the LED (more light = closer object)
- Potentially control the system with pulsed cycles to distinguish between finger sensors
- Could be controlled by daisy-chained PCBs in the finger segments to have sensing in all segments, could also be implemented in the palm (might be better to use another system for that)
- The LED could be infrared if a filter and photoresitor can be sourced that can detect and filter IR light

Wrist actuation

- Could be pulley-controlled
- Roll could be in the arm, yaw and pitch could be imbedded in the wrist section
- Could be controlled by stepper motors or larger servos (would need a driver board for stepper motors)

Arm housing

- 3D printed housing in the arm to cover electronics
- Could be extended to the hand to allow for covering sensitive components
- A flexible TPU sheath could be implemented in the wrist for wire routing

Use multiple 9V batteries in parallel and a buck converter to step down voltage

- Increases max current supply, meaning servos can safely draw more current
- Using the batteries in parallel increases capacity, making for a significantly longer battery life

DESIGNED BY:______________________ DATE:______________

WITNESSED BY:______________________ DATE:______________

- For example, a buck converter rated for 5A would be able to supply significantly more power than using 4x AA batteries, which have lower capacity and max current (1-2 Amps). SG90 servos draw around 200mA at nominal use, and at a stall can draw upwards of 1A. Assuming we are use five servos, this should be plenty of power for the servos.

DESIGNED BY:____ DATE:____________________

WITNESSED BY:_______________________________________ DATE:____________________

Arduino Code V1

```
#include <Servo.h>

Servo index_servo;
Servo ring_servo;

const int OPEN_POSITION = 180;
const int CLOSED_POSITION = 0;
const int COLLISION_BUFFER = 300;
const int CYCLE_DELAY = 500;

void setup() {
  pinMode(2, INPUT_PULLUP); // use internal pull-up
  pinMode(3, INPUT_PULLUP); // use internal pull-up
  pinMode(4, INPUT_PULLUP); // use internal pull-up
  pinMode(5, INPUT_PULLUP); // use internal pull-up
  pinMode(LED_BUILTIN, OUTPUT);

  index_servo.attach(6);
  ring_servo.attach(7);

  index_servo.write(OPEN_POSITION);
  ring_servo.write(OPEN_POSITION);
}

void move_to(Servo &servo, int target) {
  servo.write(target);
}

void loop() {

  // If both are pressed, move index first so they don't collide
  if (digitalRead(2) == LOW && digitalRead(3) == LOW) {
    index_servo.write(CLOSED_POSITION);
    delay(COLLISION_BUFFER);
```

DESIGNED BY:______________________________ DATE:______________

WITNESSED BY:______________________________ DATE:______________

```
    ring_servo.write(CLOSED_POSITION);

  } else {
    // Open or close index and middle finger and light LED if the button is pressed
    if (digitalRead(2) == LOW) {
      move_to(index_servo, CLOSED_POSITION);
      digitalWrite(LED_BUILTIN, HIGH);
    } else {
      digitalWrite(LED_BUILTIN, LOW);
    }

    // Open or close ring and pinkie finger and light LED if the button is pressed
    if (digitalRead(3) == LOW) {
      move_to(ring_servo, CLOSED_POSITION);
      digitalWrite(LED_BUILTIN, HIGH);
    } else {
      digitalWrite(LED_BUILTIN, LOW);
    }
  }

  // Reset to open and stagger to avoid collisions
  if (digitalRead(2) == HIGH && digitalRead(3) == HIGH) {
    index_servo.write(OPEN_POSITION);
    delay(COLLISION_BUFFER);
    ring_servo.write(OPEN_POSITION);
  }

  // Allow a buffer to counteract servo collision if the user makes it close within a
certain timing
  delay(CYCLE_DELAY);
}
```

DESIGNED BY:______________________________ DATE:______________

WITNESSED BY:______________________________ DATE:______________

Arduino Code V2

```
#include <Servo.h>

const int OPEN = 180;
const int CLOSED = 50;
const int COLLISION_BUFFER = 0; // 200ms delay between movements to ensure servos
don't collide

Servo index_servo;
Servo ring_servo;

int grasp_positions[3] = {OPEN, 140, CLOSED};
int position = 0;

void setup() {
  // put your setup code here, to run once:
  pinMode(LED_BUILTIN, OUTPUT);

  pinMode(2, INPUT_PULLUP); // use internal pull-up
  pinMode(3, INPUT_PULLUP); // use internal pull-up
  pinMode(4, INPUT_PULLUP); // use internal pull-up

  index_servo.attach(6);
  ring_servo.attach(7);

  randomSeed(analogRead(0)); // Makes the randomness unpredictable

  //index_servo.write(OPEN);
  //ring_servo.write(OPEN);
}

void blink() {
  index_servo.write(CLOSED);
  delay(COLLISION_BUFFER);
  ring_servo.write(CLOSED);
```

DESIGNED BY:______________________________ DATE:______________

WITNESSED BY:______________________________ DATE:______________

```
  for (int i = 0; i < 3; i++) {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(250);
    digitalWrite(LED_BUILTIN, LOW);
    delay(250);
  }
}

void rps_position(char  pos) {
  index_servo.write(CLOSED);
  delay(COLLISION_BUFFER);
  ring_servo.write(CLOSED);

  if (pos == 's' || pos == 'p') {
    index_servo.write(OPEN); // Open the index and middle fingers for paper or
scissors, else leave closed for rock
  }
  delay(COLLISION_BUFFER);
  if (pos == "p") {
    ring_servo.write(OPEN); // Open the ring and pinkie fingers for paper, else leave
closed for rock or scissors
  }
}

void grasp(int amt) {
  amt = constrain(amt, CLOSED, OPEN);
  index_servo.write(amt);
  ring_servo.write(amt);
}

void loop() {
  // put your main code here, to run repeatedly:
  if (digitalRead(4) == LOW) {
    blink();
    char choices[] = {'r', 'p', 's'};
```

DESIGNED BY:______________________________ DATE:______________

WITNESSED BY:______________________________ DATE:______________

```
    char choice = choices[random(0, 3)];
    rps_position(choice);
  }

  if (digitalRead(2) == LOW) {
    position ++;
    position = position % 3;
    grasp(grasp_positions[position]);
    delay(500);
  }

  if (digitalRead(3) == LOW) {
    while (1) {
      bool escape = false;

      for (int i = CLOSED; i <= OPEN; i++) {
        grasp(i);
        delay(10);
        if (!(digitalRead(2) == HIGH && digitalRead(4) == HIGH)) {
          escape = true;
          break;
        }
      }
      for (int i = OPEN; i >= CLOSED; i--) {
        grasp(i);
        delay(10);
        if (!(digitalRead(2) == HIGH && digitalRead(4) == HIGH)) {
          escape = true;
          break;
        }
      }

      if (escape) break; // breaks out of the *while* loop
    }
  }
}
```

DESIGNED BY:______________________________ DATE:______________

WITNESSED BY:______________________________ DATE:______________

Planned Demo Functionality

- [x] Play Rock-Paper-Scissors with the user at the press of a button
- [ ] Have a grip mode where the hand automatically senses any objects in the hand and closes
- [ ] Have 3-4 indicator LEDs to show the current state (blinking for RPS mode, finger position indicators, load indicator)

DESIGNED BY:______________________________	DATE:_______________
WITNESSED BY:______________________________	DATE:_______________

How We Used the Engineering Design Process

Define the problem: Defined at the top of our journal. The project idea is listed at the top and the requirements are named and fulfilled.

Background Research: We looked to see how other people have done various types of animatronics. A big inspiration was Will Cogley. We also looked at many designs of robotic hands, and this came with a lot of studying the anatomy of the human hand. It was actually quite an interesting process, as we learned a bunch about the human hand. The thumb turned out to be extremely complicated with a unique joint that only appears in the thumb, and this is why we eventually decided to scrap the idea of a thumb alltogether.

Brainstorming and Idea Selection: We brainstormed several ideas including a bionic hand (which we eventually settled on), some sort of Boston Dymanics style robot dog, and a miniature industrial robot arm. We decided on the bionic hand because it was the coolest design and implemented biology in a fascinating way, and it can be used for prosthetics, a very demanding field.

Build a Prototype: We didn't exactly build prototypes of the entire robot at once, but there were many prototypes especially of the fingers as we designed the robot. We went through several designs of rolling contact joints for the fingers, as well as several palm designs as we discussed whether to include the thumb and how it would be achieved, as the thumb is very complex.

Test the Design: As we built prototypes for the finger joints and even full fingers, we made sure to test vigorously. As testing doesn't always go perfectly to plan, we did end up having a casualty in the form of a shorted servo, though our mechanical designs did mostly turn out well and worked as intended.

DESIGNED BY:______________________________ DATE:______________

WITNESSED BY:______________________________ DATE:______________

Refine the Design: Throughout the process, the design changed quite a bit. Initially it was planned that we would include a thumb, but that was cut due to time constraints. The CAD model changed about as much as the design itself, and was and still is being updated constantly.

The Final Design: The final design uses aluminum T-bar as bones in the arm, with a 3d printed housing for the electronics, and the hand is almost entirely printed. Two SG90 servos control the fingers in groups of two. The entire thing is assembled using M3 hardware, as that is what the team had access to. We decided on a modular design as opposed to a print in place mechanism because while it does make it harder to assemble, it makes it much easier to replace a single part, which happened a lot during the prototyping process.

DESIGNED BY:______________________________ DATE:________________

WITNESSED BY:______________________________ DATE:________________

Code:

```
#include <Servo.h>

const int OPEN = 180;
const int CLOSED = 50;

const int SENSE_THRESHOLD = 25;
const int STALL_THRESHOLD = SENSE_THRESHOLD * 2;

Servo index_servo;
Servo ring_servo;

int index_pos = OPEN;
int ring_pos = OPEN;

String mode = "boot";

// Sense buffer
const int BUFFER_SIZE = 50;
int sensorBuffer[BUFFER_SIZE] = {0};
int index = 0;

void addValue(int val) {
   sensorBuffer[index] = val;          // Overwrite the oldest value
   index = (index + 1) % BUFFER_SIZE;  // Wrap around
}

// Function to calculate the average of the buffer
float getAverage() {
   long sum = 0;  // Use long to avoid overflow
   for (int i = 0; i < BUFFER_SIZE; i++) {
       sum += sensorBuffer[i];
   }
   return sum / BUFFER_SIZE;  // Return the average
}

void setup() {
 // put your setup code here, to run once:
 Serial.begin(9600);

 // Set button pins
 pinMode(2, INPUT_PULLUP); // use internal pull-up
 pinMode(3, INPUT_PULLUP); // use internal pull-up
 pinMode(4, INPUT_PULLUP); // use internal pull-up
 pinMode(5, INPUT_PULLUP); // use internal pull-up
  // Set LED pins
 pinMode(8, OUTPUT);
 pinMode(9, OUTPUT);
 pinMode(10, OUTPUT);
  // Set analog read pins
 pinMode(A0, INPUT);
 pinMode(A1, INPUT);

 index_servo.attach(6);
```

DESIGNED BY:______________________________ DATE:______________

WITNESSED BY:______________________________ DATE:______________

```
 ring_servo.attach(7);
}

void rps_position(char pos) {
 // Update the servo positions
 index_servo.write(CLOSED);
 ring_servo.write(CLOSED);

 for (int i = 0; i < 3; i++) {
   digitalWrite(9, HIGH);
   delay(250);
   digitalWrite(9, LOW);
   delay(250);
 }

 if (pos == 's') {
       index_pos = OPEN;
       ring_pos = CLOSED;
 }
 if (pos == 'p') {
       index_pos = OPEN;
       ring_pos = OPEN;
 }
 if (pos == 'r') {
       index_pos = CLOSED;
       ring_pos = CLOSED;
 }


 // Update the servo positions
 index_servo.write(index_pos);
 ring_servo.write(ring_pos);
}

void grasp(int amt) {
 amt = constrain(amt, CLOSED, OPEN);
 index_pos = amt;
 ring_pos = amt;
}

void loop() {
 // Move when the first button is pressed
 if (digitalRead(2) == LOW) {
   mode = "positioning";
   index_pos -= 0.75;
   if (index_pos <= CLOSED) {
     index_pos = OPEN;
   }
   grasp(index_pos);
   delay(10);
   digitalWrite(9, HIGH);
 } else {
   digitalWrite(9, LOW);
 }

 // Run the Rock-Paper-Scissors function when the second button is pressed
```

DESIGNED BY:__________________________________ DATE:________________

WITNESSED BY:__________________________________ DATE:________________

```
if (digitalRead(3) == LOW) {
  mode = "rps";
  char choices[] = {'r', 'p', 's'};
  char choice = choices[random(0, 3)];
  rps_position(choice);
}

// Go into sensing mode when the third button is pressed
if (digitalRead(4) == LOW) {
  mode = "sensing";
}

void detach_stalled() {
    // Disconnect stalled servos to prevent further damage
    if (sense > STALL_THRESHOLD) {
      index_servo.detach();
      ring_servo.detach();
      while (true) {
        digitalWrite(9, HIGH); // Keep the blue LED on
        delay(10);
      }
    }
}

if (mode == "sensing") {
  digitalWrite(8, LOW);
  digitalWrite(10, LOW);
  addValue(analogRead(A0));
  addValue(analogRead(A1));
  int sense = getAverage();
  bool is_touching = false;
  while (index_pos > CLOSED) {
      index_pos -= 2;
      grasp(index_pos);
      digitalWrite(9, LOW);
      detach_stalled();
      if (sense > SENSE_THRESHOLD) {
        is_touching = true;
        digitalWrite(9, HIGH);
      }
      delay(10);
  }

  if (!is_touching) {
    digitalWrite(8, HIGH);
    digitalWrite(10, HIGH);
    delay(500);
    index_pos = OPEN; // Open hand if it’s not touching anything
  } else {
    while (digitalRead(2) == HIGH && digitalRead(3) == HIGH) {
      delay(100);
    }
  }


  delay(10);
```

DESIGNED BY:______________________________ DATE:______________

WITNESSED BY:______________________________ DATE:______________

```
  }

  // Illuminate the LEDs when closed
  digitalWrite(8, index_pos < (OPEN + CLOSED) / 2);
  digitalWrite(10, ring_pos < (OPEN + CLOSED) / 2);

  // Update the servo positions
  index_servo.write(index_pos);
  ring_servo.write(ring_pos);
}
```

DESIGNED BY:______________________________ DATE:______________

WITNESSED BY:_____________________________ DATE:______________