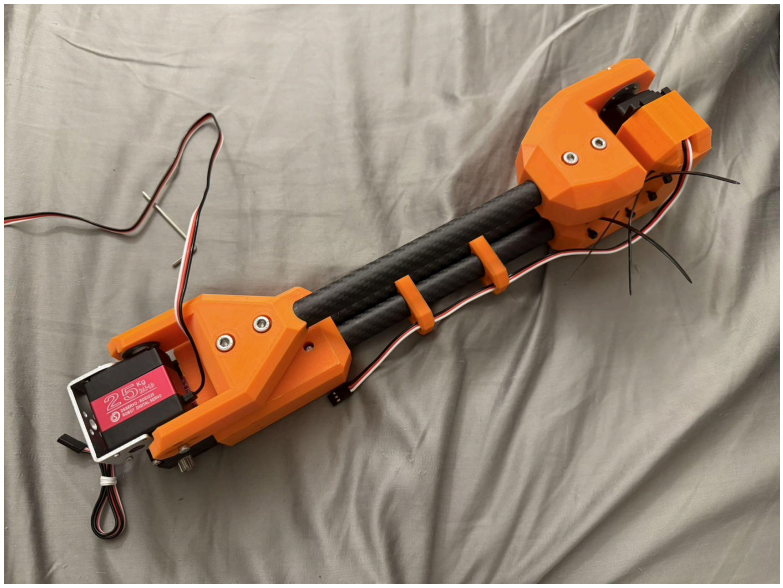
Printing Guide:

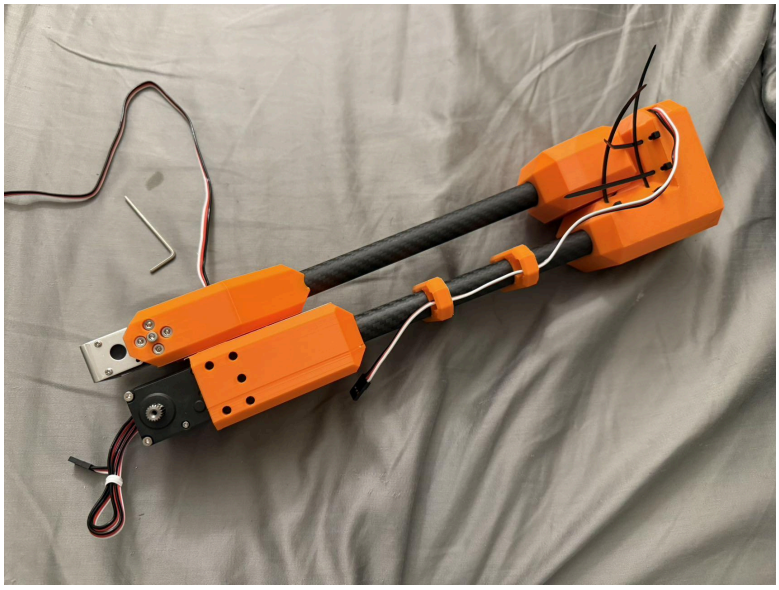
Print Bases with 40% infill, 4 walls side, top and bottom.
The middle part with 25% infill
Upper part with 20% infill

Material Needed:
M5, M3, M2.5 Bolts
M5, M3 Heat Inserts
Carbon Fiber Tube 16mm OD
BOM: Upload later

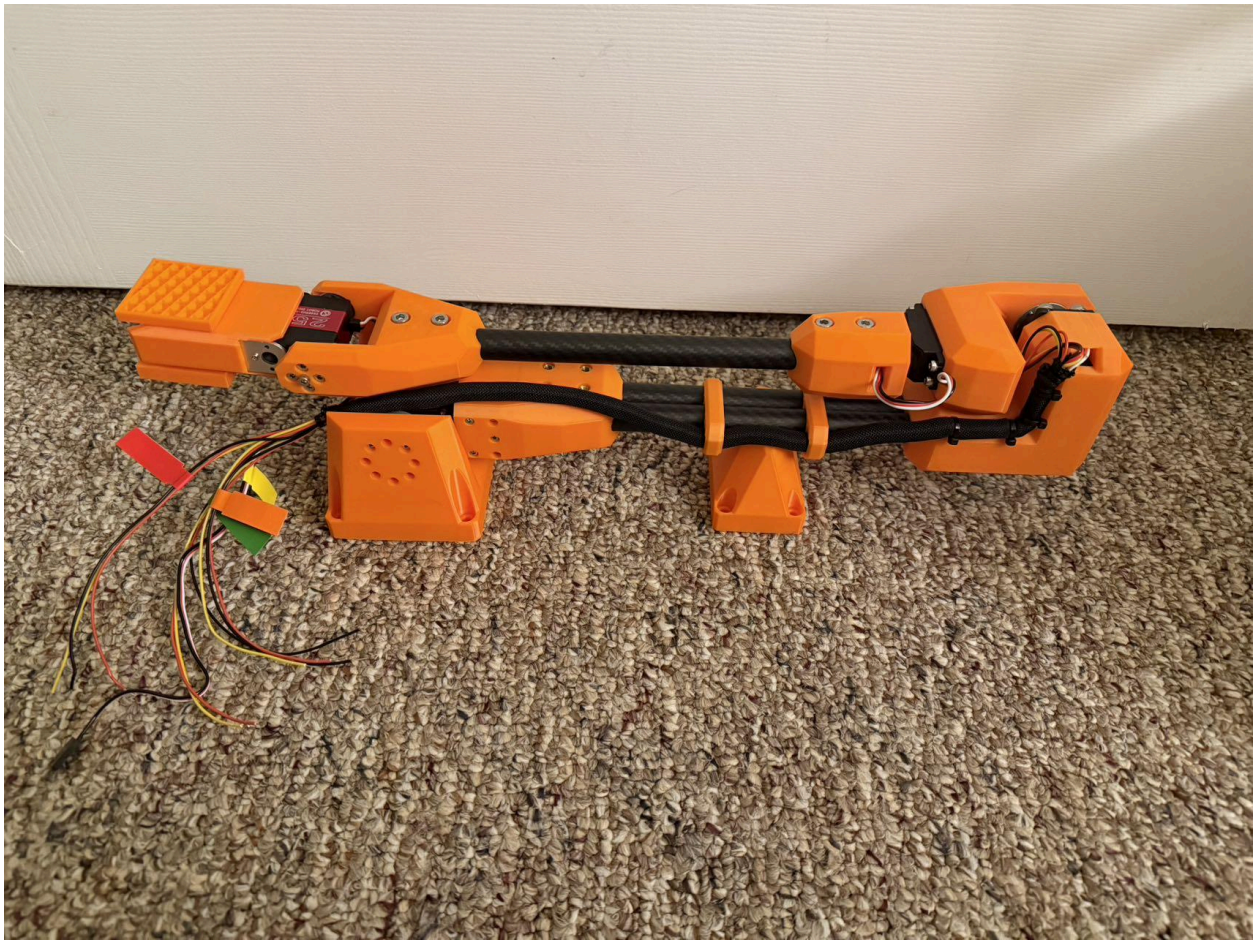
The following is the assembling of the robotic arm. This is what I did this semester.

Drill the Carbon Fiber Tube using the provided mold for drilling



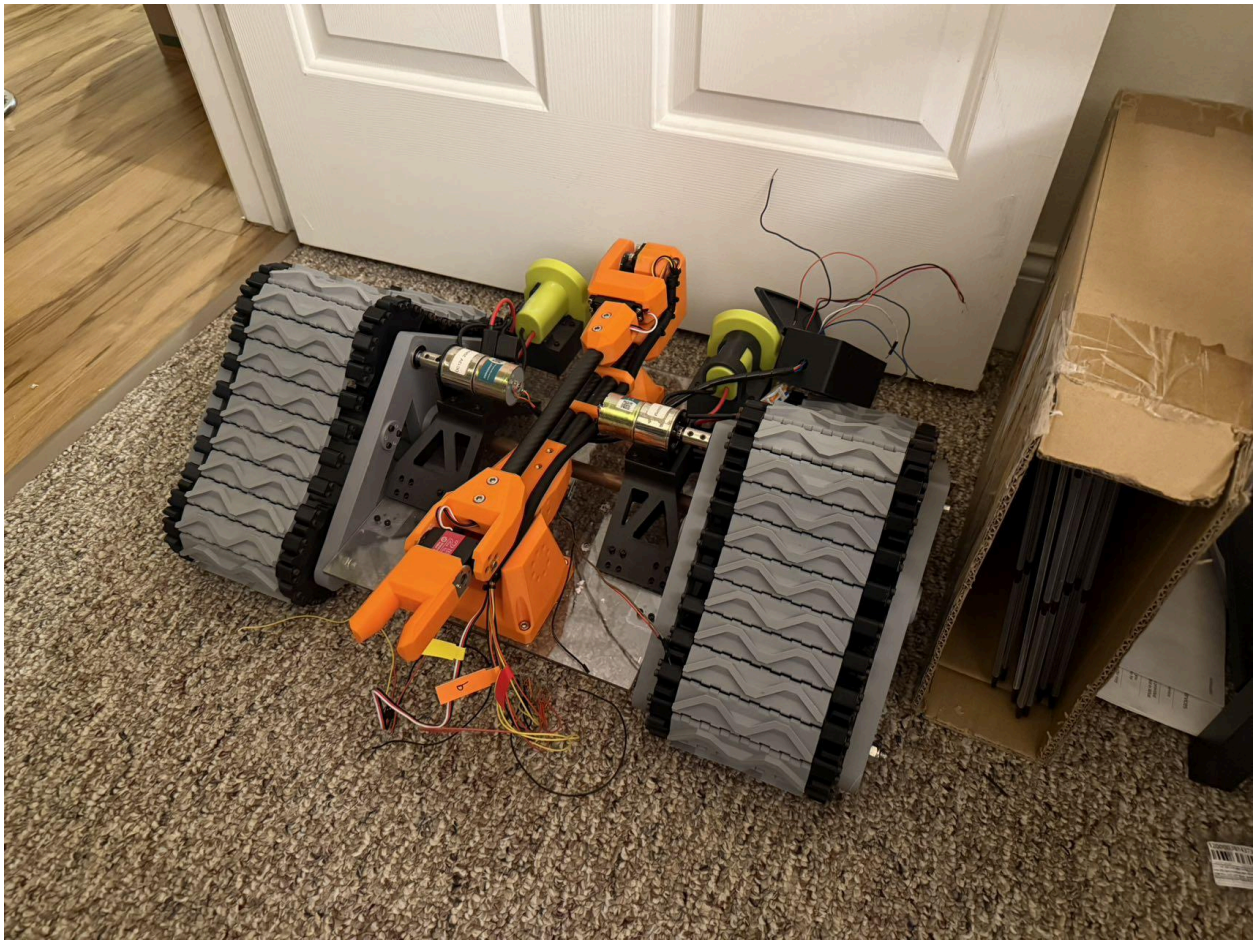


Assemble the printed parts with carbon tube with M5 bolts using the proper size.



All printed parts are bolted onto the arm and are functional.

I rewired and extended all wires. All soldered and heat shrink protected.



I assemble the arm onto the robot.

I wrote this code for Arduino test the arm.

```
#include <Servo.h>

// Create Servo objects
Servo Servo1, Servo2, Servo3, Servo4;

// Pin assignments
int servoPin1 = 9, servoPin2 = 10, servoPin3 = 11, servoPin4 = 6;

// Current angles for each servo
int currentAngle1 = 0, currentAngle2 = 0, currentAngle3 = 0, currentAngle4
= 0;

void setup() {
  Serial.begin(9600);
```

```
  // Attach servos to their respective pins
  Servo1.attach(servoPin1);
  Servo2.attach(servoPin2);
  Servo3.attach(servoPin3);
  Servo4.attach(servoPin4);

  // Initialize all servos at 0 degrees
  Servo1.write(currentAngle1);
  Servo2.write(currentAngle2);
  Servo3.write(currentAngle3);
  Servo4.write(currentAngle4);

  delay(1000); // Allow servos to initialize
  Serial.println("Servo control initialized.");
}

// General function to move a servo to a specified angle at a reduced
speed
void moveServoToAngle(Servo &servo, int &currentAngle, int targetAngle,
int speed) {
  targetAngle = constrain(targetAngle*0.75, 0, 180);
  Serial.println("Moving servo to: " + String(targetAngle) + " degrees");

  if (currentAngle < targetAngle) {
    for (int angle = currentAngle; angle <= targetAngle; angle++) {
      servo.write(angle);
      delay(speed);
    }
  } else {
    for (int angle = currentAngle; angle >= targetAngle; angle--) {
      servo.write(angle);
      delay(speed);
    }
  }

  currentAngle = targetAngle;
}

void loop() {
```

```
  // Example movements for all servos
  moveServoToAngle(Servo3, currentAngle3, 90, 15); // Move Servo3 to 135
  delay(1000);

  moveServoToAngle(Servo4, currentAngle4, 90, 15); // Move Servo4 to 180
  delay(1000);

  moveServoToAngle(Servo1, currentAngle1, 0, 15);  // Move Servo1 to 90
  moveServoToAngle(Servo2, currentAngle2, 0, 15);  // Move Servo2 to 45

  moveServoToAngle(Servo3, currentAngle3, 45, 15); // Move Servo3 to 135
  delay(1000);

  moveServoToAngle(Servo4, currentAngle4, 45, 15); // Move Servo4 to 180

  delay(1000); // Wait for 1 second

  moveServoToAngle(Servo1, currentAngle1, 180, 15);   // Move Servo1 to 0
  moveServoToAngle(Servo2, currentAngle2, 180, 15);  // Move Servo2 to 90

  moveServoToAngle(Servo3, currentAngle3, 135, 15);  // Move Servo3 to 45
  delay(1000);

  moveServoToAngle(Servo4, currentAngle4, 135, 15);  // Move Servo4 to 90

  delay(1000); // Wait for 1 second
}
```

For raspberry pi:

```
import time
import pigpio

# GPIO pin assignments for servos
servo_pin1 = 17  # GPIO17 (Pin 11)
servo_pin2 = 27  # GPIO27 (Pin 13)
servo_pin3 = 22  # GPIO22 (Pin 15)
servo_pin4 = 6   # GPIO6 (Pin 31)

# Initialize pigpio
pi = pigpio.pi()
```

```
if not pi.connected:
    print("Failed to connect to pigpio daemon.")
    exit()

# Function to set servo angle
def move_servo(pi, pin, target_angle, speed):
    """
    Move the servo to the target angle at the given speed.

    :param pi: pigpio instance
    :param pin: GPIO pin connected to the servo
    :param target_angle: Target angle in degrees (0-180)
    :param speed: Delay between steps in seconds (smaller value = faster)
    """
    target_angle = max(0, min(180, target_angle))  # Constrain to 0-180 degrees
    pulse_width = int(500 + (target_angle / 180) * 2000)  # Map 0-180 to 500-2500 µs
    pi.set_servo_pulsewidth(pin, pulse_width)
    time.sleep(speed)  # Add delay to simulate slow movement

# Initialize all servos to 0 degrees
for pin in [servo_pin1, servo_pin2, servo_pin3, servo_pin4]:
    pi.set_servo_pulsewidth(pin, 500)  # 0 degrees (500 µs pulse width)
time.sleep(1)

print("Servo control initialized.")

# Example movements
try:
    while True:
        # Move servos to specific angles
        move_servo(pi, servo_pin1, 90, 0.03)  # Servo1 to 90 degrees
        move_servo(pi, servo_pin2, 45, 0.03)  # Servo2 to 45 degrees
        move_servo(pi, servo_pin3, 135, 0.03) # Servo3 to 135 degrees
        move_servo(pi, servo_pin4, 180, 0.03) # Servo4 to 180 degrees
        time.sleep(1)

        # Move servos back to initial positions
        move_servo(pi, servo_pin1, 0, 0.03)   # Servo1 to 0 degrees
        move_servo(pi, servo_pin2, 90, 0.03)  # Servo2 to 90 degrees
        move_servo(pi, servo_pin3, 45, 0.03)  # Servo3 to 45 degrees
        move_servo(pi, servo_pin4, 90, 0.03)  # Servo4 to 90 degrees
        time.sleep(1)

except KeyboardInterrupt:
```

```
    print("Exiting program...")

finally:
    # Turn off servos and cleanup
    for pin in [servo_pin1, servo_pin2, servo_pin3, servo_pin4]:
        pi.set_servo_pulsewidth(pin, 0)  # Turn off servo
    pi.stop()
    print("Cleanup complete.")
```