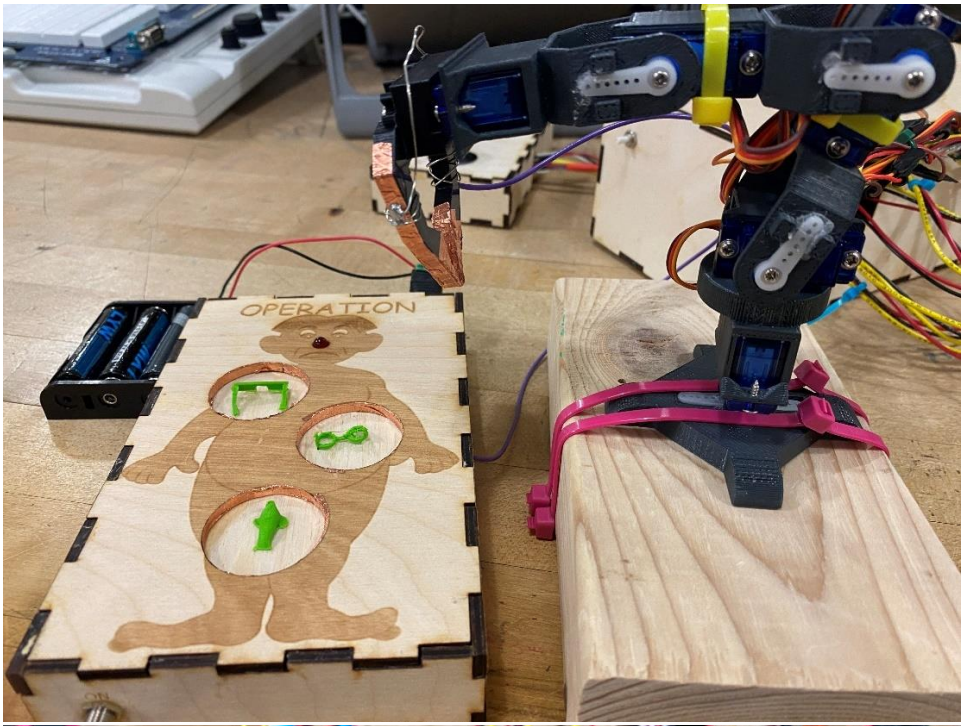
Christopher Mack II (GTID: 903580800)
Jesse Dill (GTID: 903590055)
CS 3651
April 28, 2022

Robotic Operation



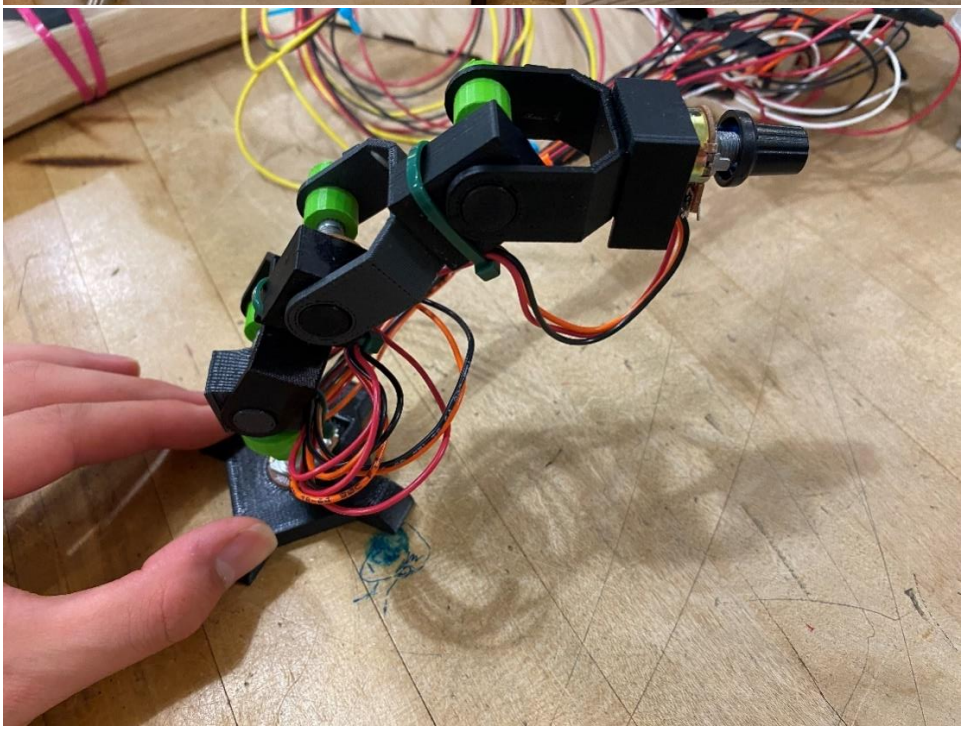


Table of Contents

Introduction

Project Overview

Initially, the project was meant to only be a robot arm that is able to be controlled by another arm of the same form factor. However, we also built an operation-style game board that was used as an application for the arm. The purpose of this project was to create an alternative method of controlling a robotic arm similar to those in industrial cranes or claw machines found in arcades. In lieu of a traditional joystick setup, the user can use the controller arm for the controlled arm to mirror its actions, resulting in a finer control than what some joysticks can provide. Additionally, it can be more entertaining to use in games than the traditional joystick. Though the primary focus is the arm-shaped controller, we also implemented a traditional joystick controller that can function with the robot arm as a backup.

Technical Description

List of Parts Used

- 5 SG90 servos
- 5 small 10kΩ potentiometers
- 1 LM7805 voltage regulator
- 2 dual-axis 10kΩ potentiometer analog joysticks
- 1 large 10kΩ potentiometer
- 1 Arduino Mega 2560
- 1 protoboard
- 1 red LED
- 1 buzzer
- 1 switch
- 1 button
- 2 male barrel jack connectors
- 1 9VDC 3A adapter
- 3 AA batteries
- copper tape
- wood, wood glue, and screws
- 3D printed materials
- 1 1µF capacitor, 1 10µF capacitor, and 1 330Ω resistor

Servos

The main arm, i.e., the one being controlled, used five SG90 micro servos that were contained within a 3D-printed housing. One servo connected to a 5V DC source drew ~300mA of current during testing under a heavier load, meaning that the five servos cannot be powered directly through the microcontroller used for this project, an Arduino Mega 2560. Instead, the system used an LM7805 voltage regulator to convert 9V DC from an external power supply to 5V DC for the servos to use, while the microcontroller remained connected directly to the 9V DC to ensure that the servos did not drop the microcontroller below its desired operating voltage. In order to use the servos, we utilized the Arduino Servo.h library that uses timers to generate the appropriate PWM signals that controlled the servos' positions from 0 degrees to 180 degrees.

Potentiometers

The controller arm used five 10kΩ potentiometers to provide the microcontroller information on desired position. The microcontroller used a function to map a specified input voltage range to a position to which the associated servo can move. In this case, the range is 0.75V to 4.4V when the potentiometers are provided 5V through the microcontroller, which mapped a reading of 0.75V to the 180 position and a reading of 4.4V to the 0 position. Intermediate values were automatically calculated based on the voltage range; however, values outside of the voltage range were ignored due to the 180-degree limitation of the SG90 servos, whereas the potentiometers can turn 270 degrees. The gripper was mapped to different values to perform its own full range of motion due to risk of damaging the servo for going too far in one direction: 0V for 20 degrees and 5V for 180 degrees.

Joystick Controller

The joystick controller used two analog joysticks to control the position of the robot arm and one potentiometer to control the servo which opens and closes the gripper. The gripper works with the same mapping mentioned previously, but the functions of the joysticks differ. The joysticks were dual-axis potentiometers that provide the microcontroller with a reading every 25 milliseconds. Values between 2.35V and 2.65V kept its respective servo from moving. Values from 2.35V to 0V moved the servos by 1 to 3 degrees every cycle until 180 degrees is reached, and values from 2.65V to 5V moved the servos by 1 to 3 degrees every cycle until 0 degrees is reached. A user can press a button on the system to select which of the two controller modes to use.

Operation Board

The operation board used a red LED and resistor that were connected in parallel with a buzzer, which were all connected in series with copper tape placed around the edges of entrance holes. The positive sides of the LED and buzzer were connected to the end of an on/off switch that was connected in series with the positive terminal of a 4.5V battery source, and the negative terminal of the battery source is connected to the copper tape on the gripper of the robot arm via a wire. When the system was provided with a power source and turned on and when the gripper touched the edges of the entrance holes, the LED was lit, and the buzzer beeped due to completing the circuit.

Lessons and Future Work

What have we learned from this experience?

Listed below are lessons that we have learned and the challenges associated with them while completing this project:

- **Thoroughly test parts as early as possible to find alternatives and find what works and what does not work.**
    - Although we had received all parts that we would need for the project fairly early, late in the project, we were experiencing major issues with controlling the servos via the main potentiometers, despite at least one potentiometer functioning during early testing. We tested a total of 25 potentiometers at three different values (1kΩ, 10kΩ, and 50kΩ) of similar models and sizes purchased from different providers, but none worked as expected at all times. We determined late in the project that those model potentiometers may have had manufacturing issues that caused the wiper to only read the value connected to the left pin rather than act as a voltage divider for both the left and right pins, but larger model

potentiometers did not give the same issues, such as the one used for the joystick controller.

- **3D printing can take massive amounts of time due to a variety of factors.**
    - Modelling 3D prints can take a lot of time in itself, but combined with misprints caused by problems with the 3D printer; sizing errors; a lack of resources, e.g., printers available to us already in use, being out of the appropriate filament, etc.; and waiting for prints to finish in general, a lot of time was dedicated to 3D printing early on in the project, distracting from other tasks.
- **PCB design can save a lot of trouble regarding wire management and routing when many wires are involved.**
    - Late in the project, wiring was a primary pain point due to not realizing exactly how we would want to wire our project until we began to wire it. We believe that creating a schematic earlier in the project and fabricating a PCB alongside this schematic could have eased this problem.

In addition to learning lessons during the completion of this project, we have greatly enhanced our physical prototyping skills. We have become more proficient in the use of 3D CAD programs and using 3D printers, SVG editors for laser cutting, and using power tools.

What are our next steps?

Though the project was an overall success, there are things that we can do to enhance the project. The following are possible future improvements:

- Add a transistor switch to the operation board connected to a microcontroller that keeps track of how much time is spent attempting to grab pieces to alert the player when their time has run out.
- Use different model, more reliable potentiometers, along with using bigger servos to apply to other situations that may require more torque than an operation game, or instead of using potentiometers to control servos, use stepper motors and high count to turn ratio rotary encoders for much higher accuracy compared to the current setup.
- Remodel the gripper so that it is facing forward rather than downwards for easier control.
- Add a feature where the microcontroller can record and replay movements that were made over a short period of time.
- Add ability to control “wirelessly”.

Appendices

Appendix A: 3D Printed Arms

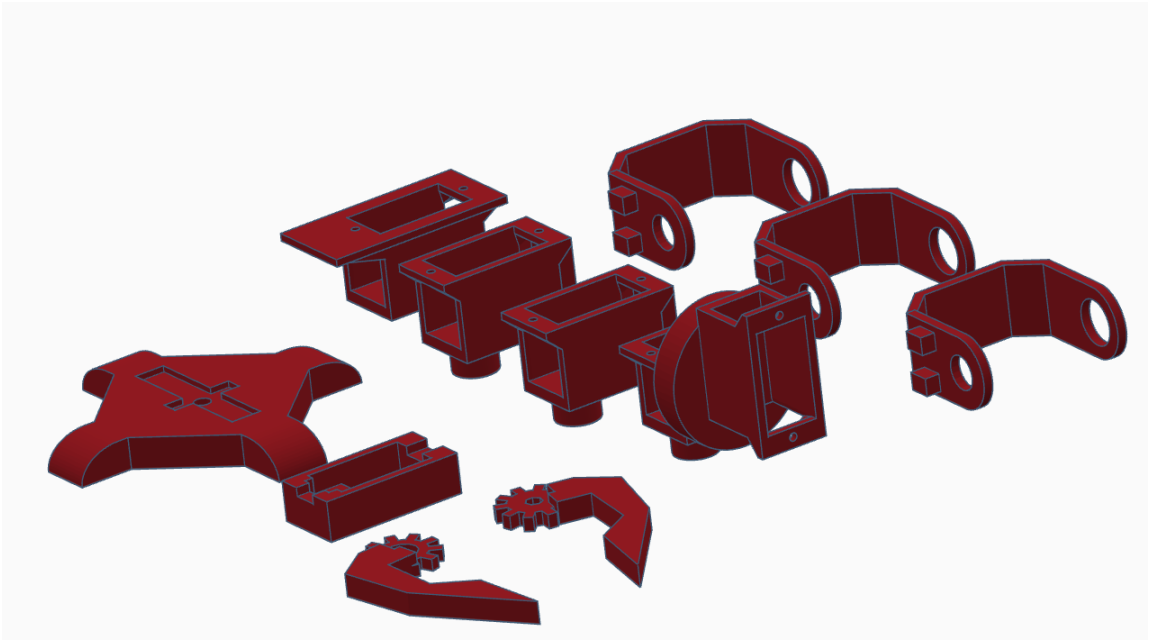


Figure A.1: Unassembled servo arm

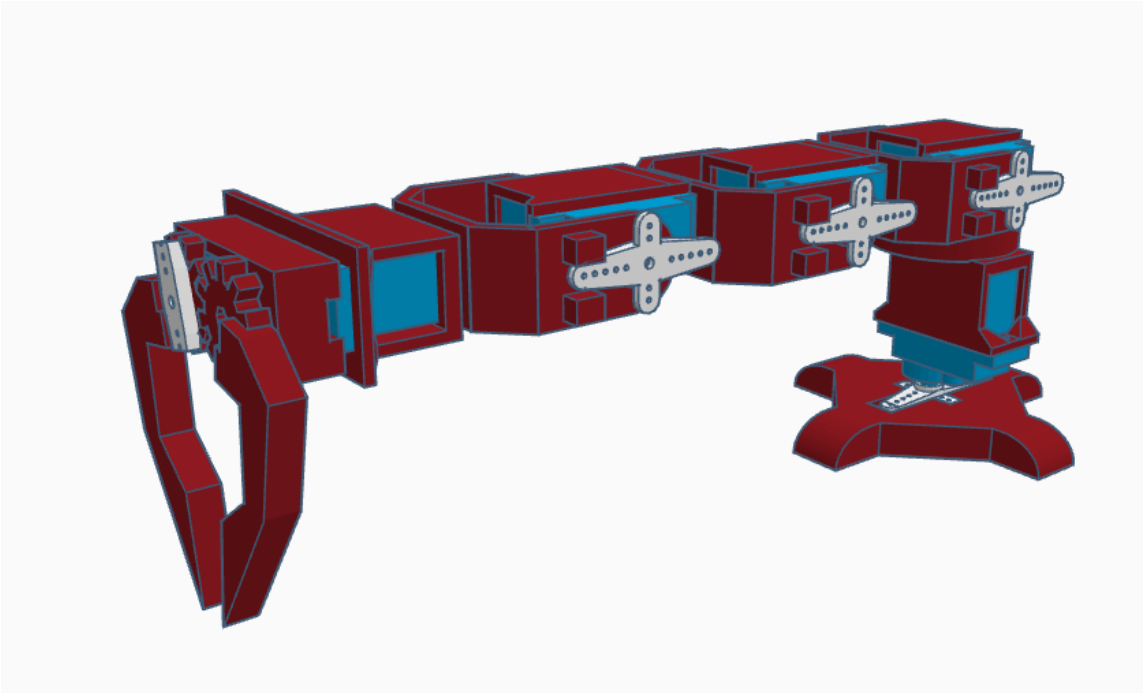


Figure A.2: Assembled servo arm

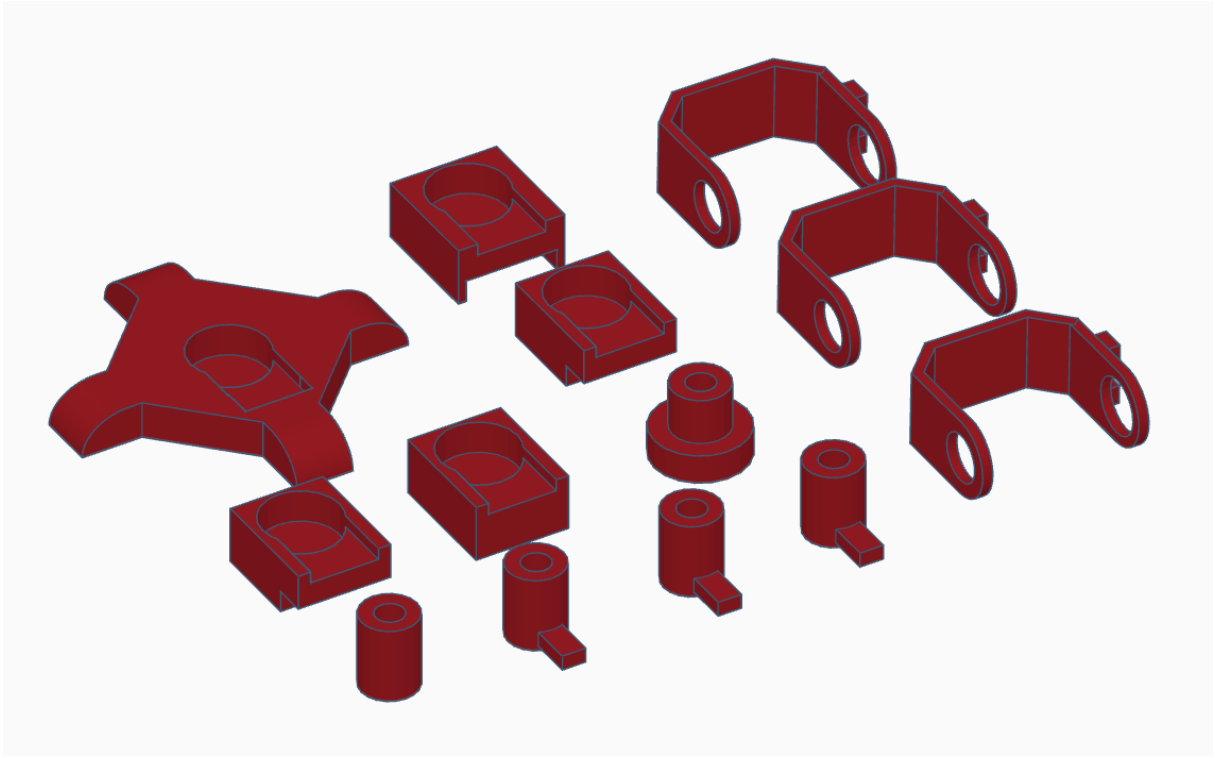


Figure A.3: Unassembled potentiometer arm

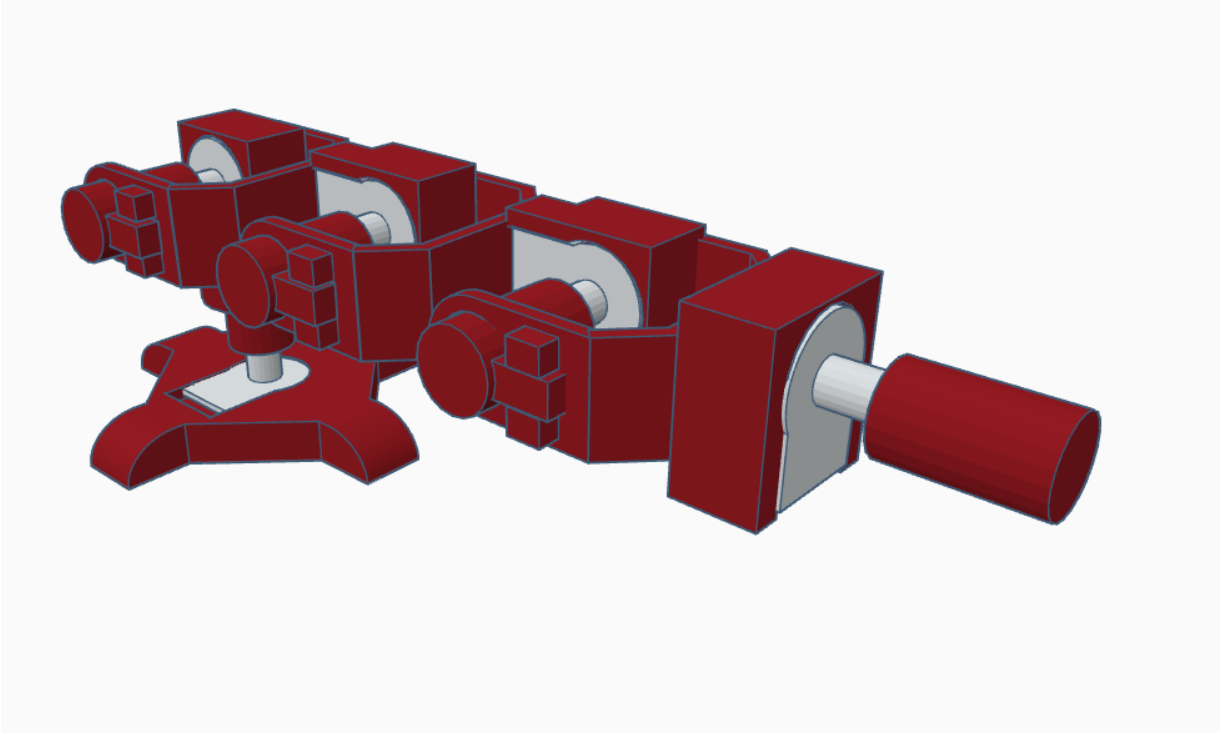


Figure A.4: Assembled potentiometer arm

Appendix B: Laser Cut Joystick Controller

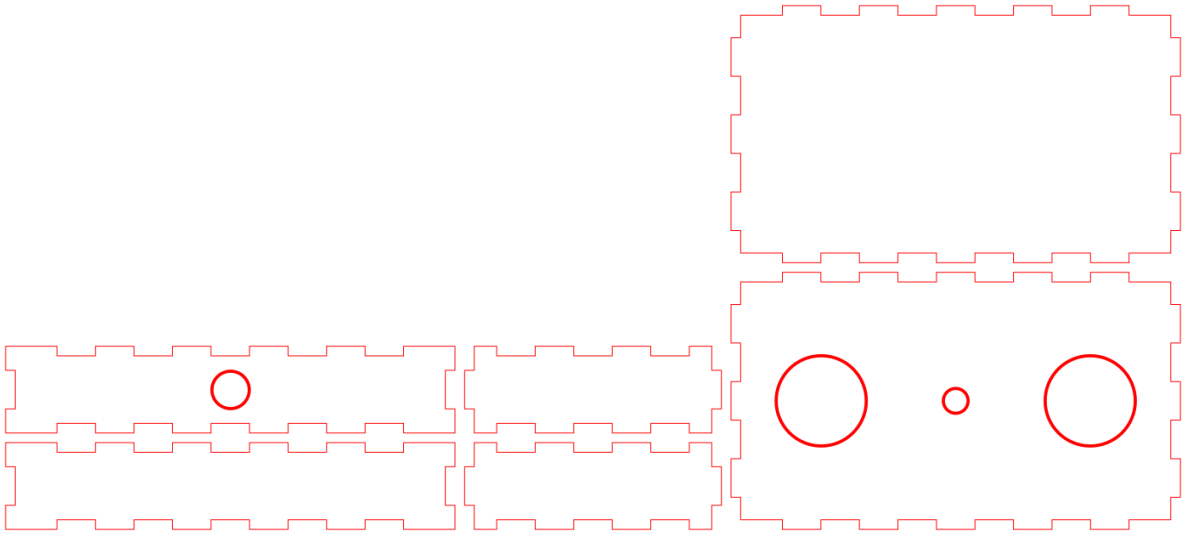


Figure B.1: Laser cut joystick controller layout

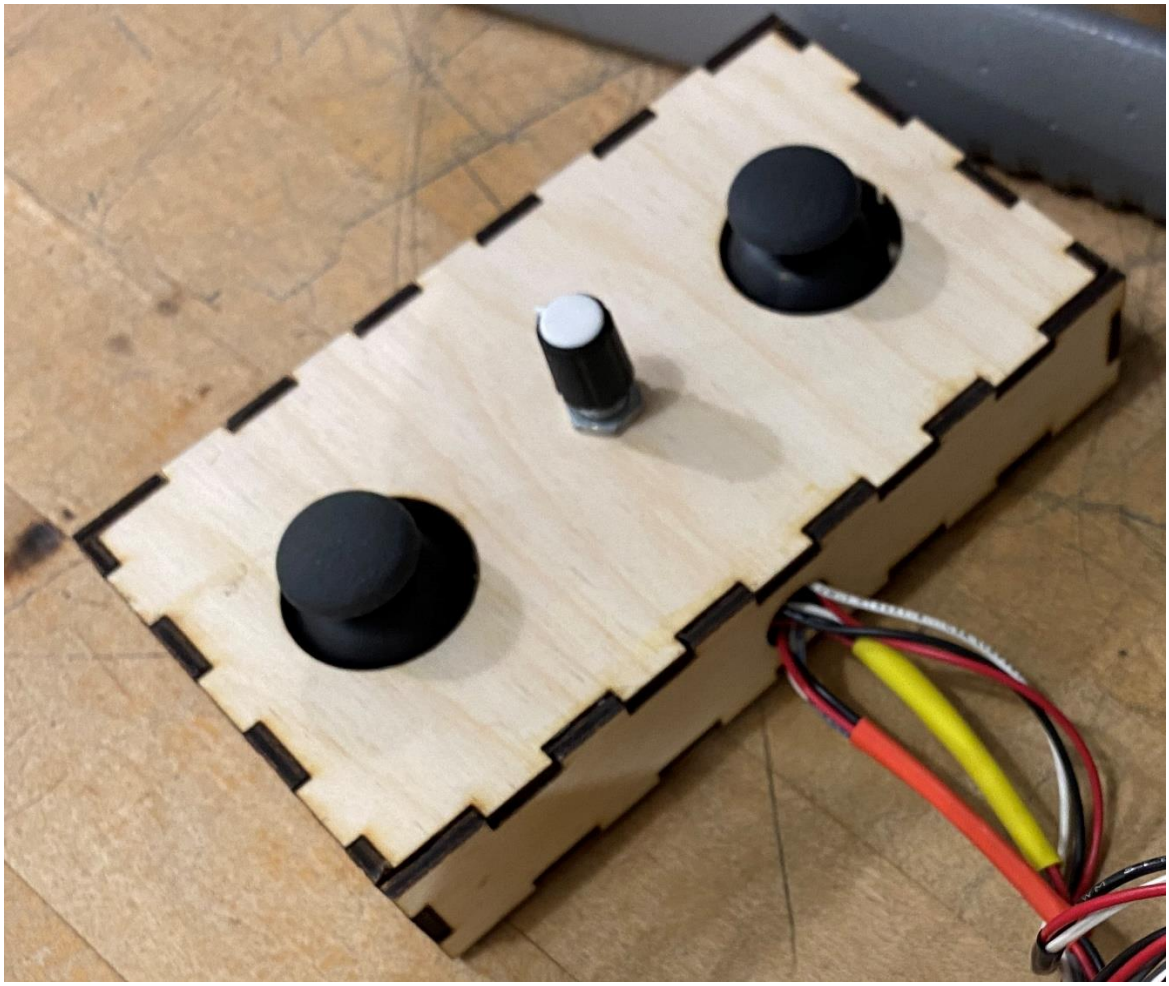


Figure B.2: Assembled laser cut joystick controller

Appendix C: Laser Cut Operation Board and 3D Printed Game Pieces

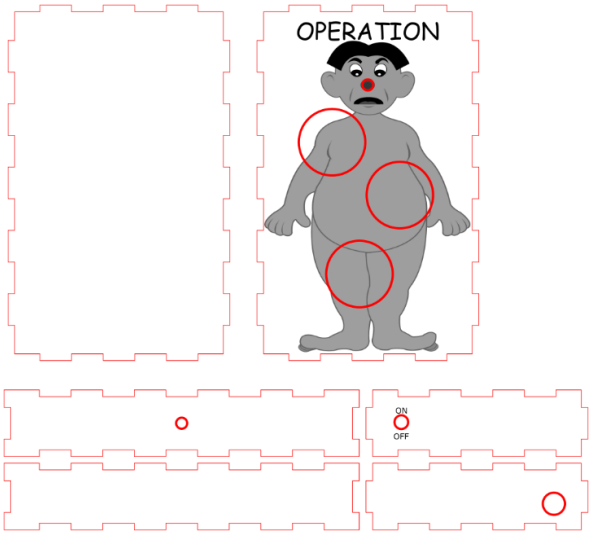


Figure C.1: Laser cut operation board layout

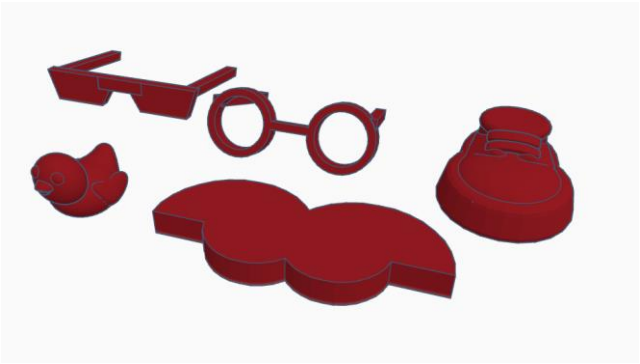


Figure C.2: 3D printed game board pieces

Figure C.3: Assembled laser cut operation board with 3D printed game pieces

Appendix D: Electronics Schematic

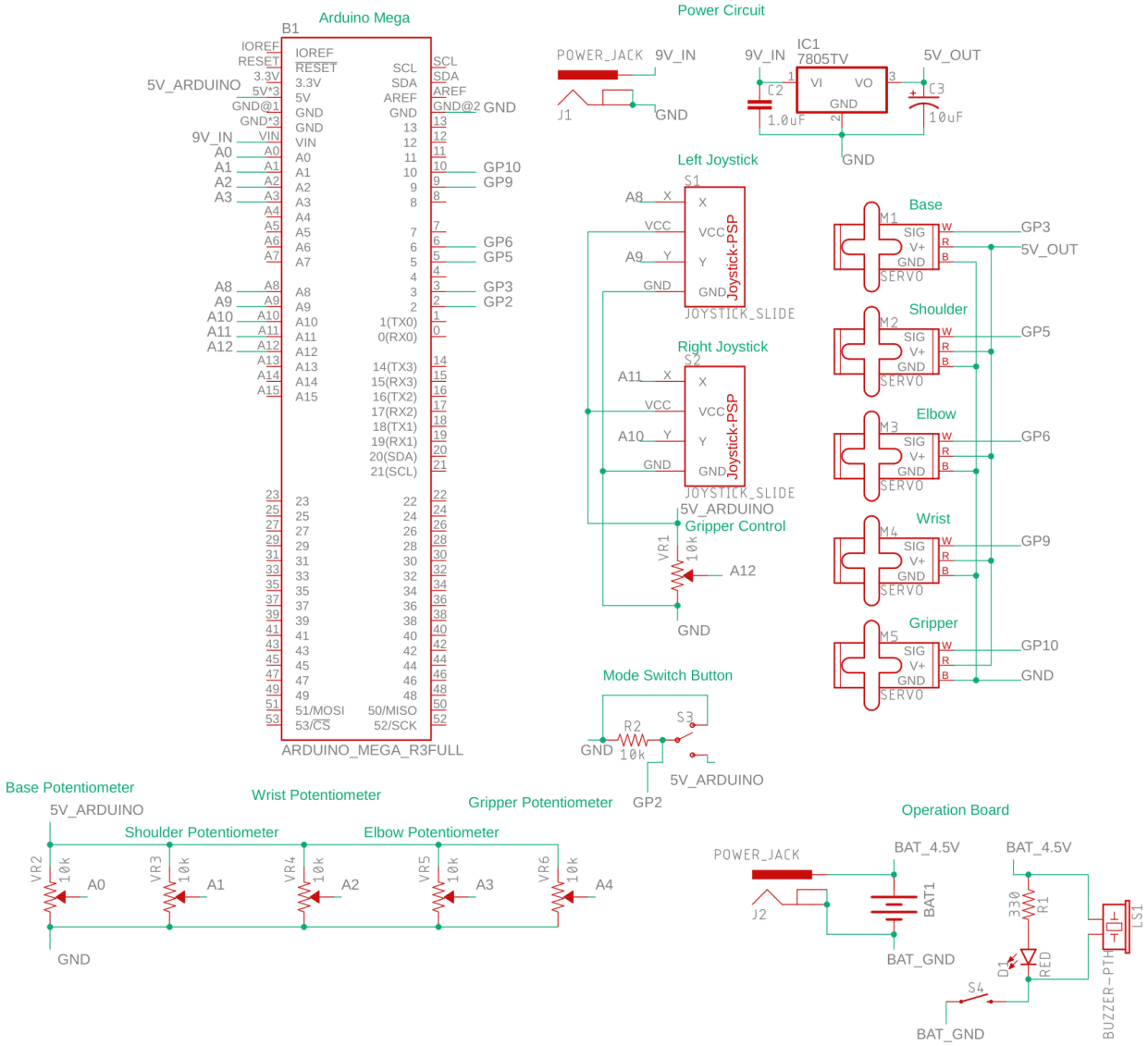


Figure D.1: System circuit diagram created in Eagle

Appendix E: Driver Code

```
#include "Servo.h"
Servo base;
Servo shoulder;
Servo elbow;
Servo wrist;
Servo hand;
int mode;

void setup() {
  mode = 1;
  pinMode(A0, INPUT);
  pinMode(A1, INPUT);
  pinMode(A2, INPUT);
  pinMode(A3, INPUT);
  pinMode(A4, INPUT);
  pinMode(A8, INPUT);
  pinMode(A9, INPUT);
  pinMode(A10, INPUT);
  pinMode(A11, INPUT);
```

```
  pinMode(A12, INPUT);
  base.attach(3);
  shoulder.attach(5);
  elbow.attach(6);
  wrist.attach(9);
  hand.attach(10);
  base.write(90);
  shoulder.write(90);
  elbow.write(90);
  wrist.write(90);
  hand.write(90);
  pinMode(2, INPUT);
}

void loop() {
  if (mode == 0) {
    mirror_servo(A0, base);
    mirror_servo(A1, shoulder);
    mirror_servo(A2, elbow);
    mirror_servo(A3, wrist);
    mirror_servo(A4, hand);
    if (digitalRead(2) == 1) {
      mode = 1;
      delay(500);
    }
  }

  if (mode == 1) {
    controller_servo(A8, base);
    controller_servo(A9, shoulder);
    controller_servo(A10, elbow);
    controller_servo(A11, wrist);
    controller_servo(A12, hand);
    if (digitalRead(2) == 1) {
      mode = 0;
      delay(500);
    }
    delay(25);
  }
}

void mirror_servo(int pin, Servo servo) {
  int output;
  if (pin == A4) {
    output = map(analogRead(pin), 0, 1023, 20, 180);
    servo.write(output);
    return;
  }

  if (analogRead(pin) < 150 || analogRead(pin) > 900) return;
  else {
    output = map(analogRead(pin), 150, 900, 180, 0);
    servo.write(output);
  }
}

void controller_servo(int pin, Servo servo) {
  int output;
  if (pin == A12) {
    output = map(analogRead(pin), 0, 1023, 20, 180);
    servo.write(output);
    return;
  }

  int pos = servo.read();
  int val = analogRead(pin);
  if (val > 480 && val < 544) {
    return;
  } else if (val <= 480) {
    output = map(val, 0, 480, 3, 1);
    if (pos + output > 180) servo.write(180);
    else servo.write(pos + output);
  } else if (val >= 544) {
    output = map(val, 544, 1023, 1, 3);
    if (pos - output < 0) servo.write(0);
    else servo.write(pos - output);
  }
}
```