ME 102b Final Project Fall 2020, Nicolas Williams
Robotic Arm Prototype With Hand Sanitizer Demo

Description of the project:

For this project, I designed a protype robotic arm that can pick up object and drop in. I chose this project to further my understanding of kinematics, and mechatronics. To demonstrate the functionality of the robotic arm, I had it pick up a bottle of hand sanitizer, pour it in my hand, and then return the bottle to the same location it picked it up from.

Demonstration Video: https://youtu.be/6_NfBw743nw

Figure 1: Robotic Arm Rotation Points

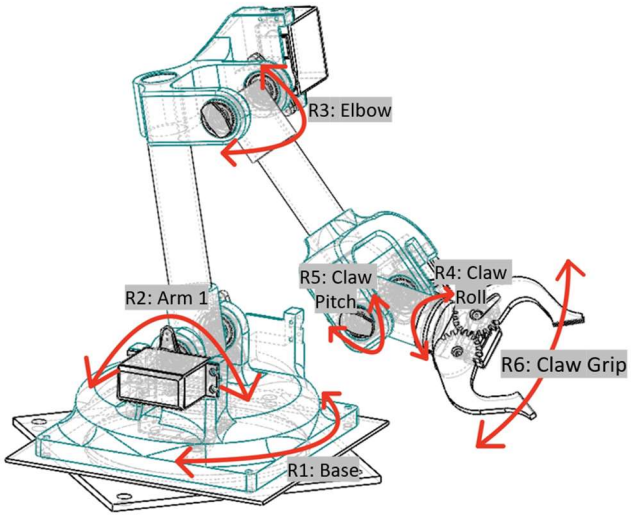


Figure 2: Robotic Arm Dimensions

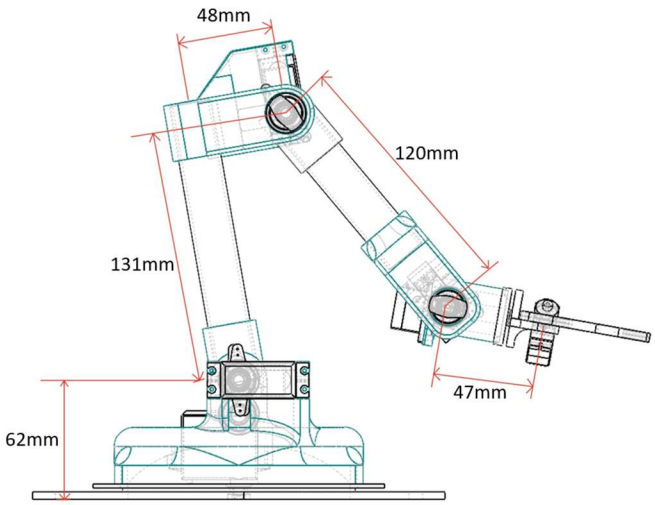


Figure 3: Isometric CAD Assembly

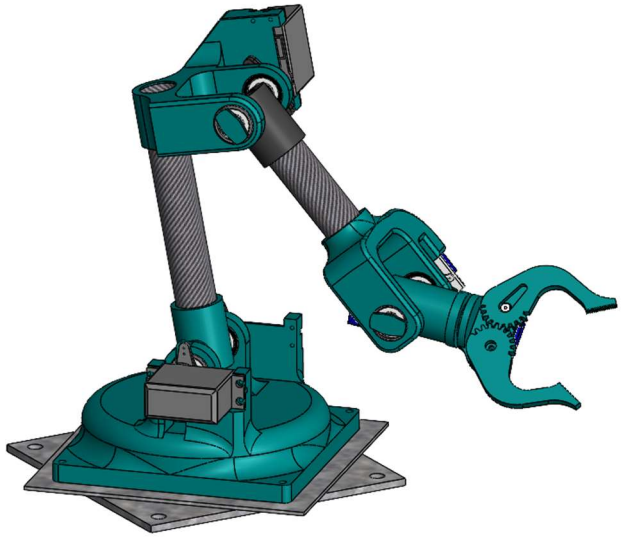
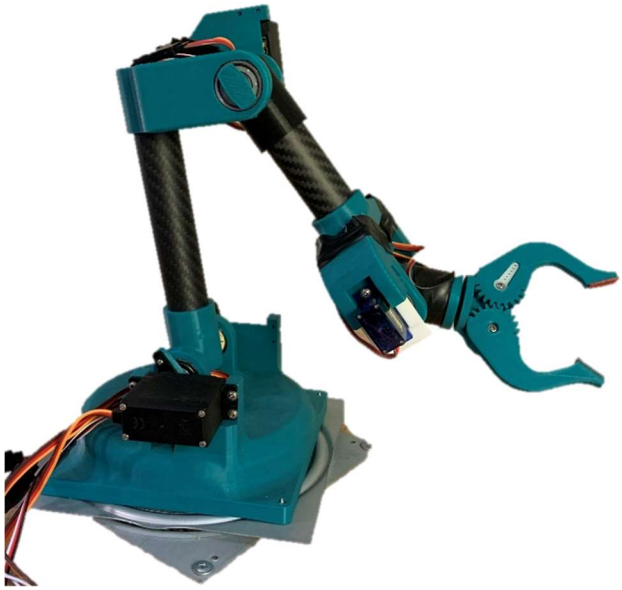


Figure 4: 3D Printed and Assembled Arm



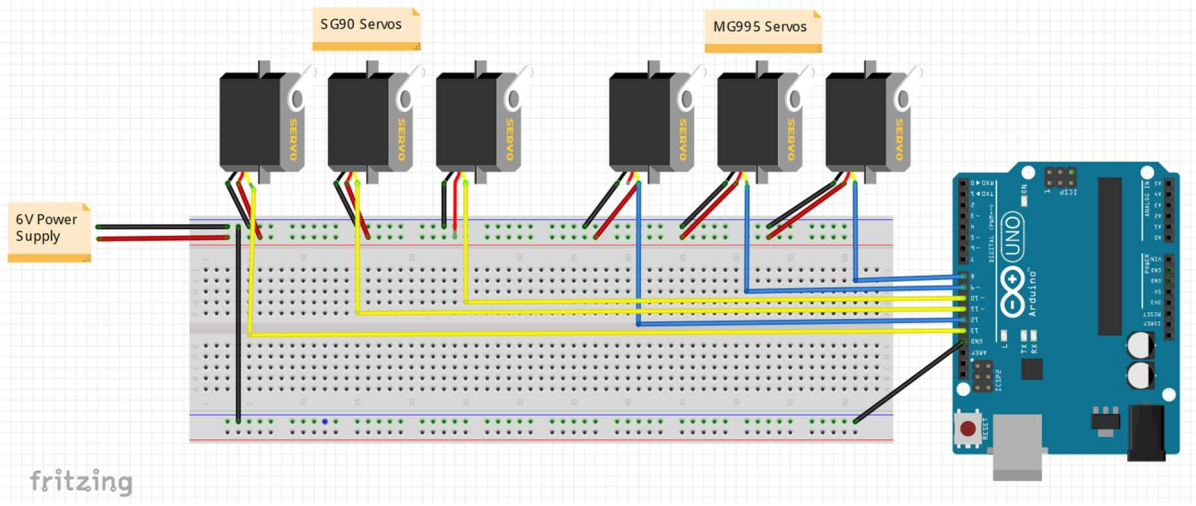
Motor Selection:

This project required a motor with precise location control. The lowest cost options available for this were hobbyist servo motors. I used two types of servos: SG90, and MG995. To control the motor, I used an Arduino UNO.

The SG90 servo is a light weight and low power servo. I used this servo on the end of the robotic arm (R4, R5, R6) since the light weigh servos reduced the moment at the other joints, and there are low torque requirements at these locations. The SG90 has a stall torque of approximately 2.2 kg/cm at 6.0v.

On the other joints (R1, R2, R3) I used an MG995 servo motor because it has much higher torque (11kg/cm at 6.0V). In the a medium extended position (see Figure 1 and 2) the arm can carry a mass of 0.505 kg (see Appendix 1 for MATLAB calculations). The robot was designed to carry an object of 0.3kg with Arm 1 vertical, and the elbow joint at a 90 degree angle. At this position, the max weigh it can hold it 0.3699kg which makes the robot have a FOS of 1.23 at this limitation.

Figure 5: Fritzing Circuit Schematic



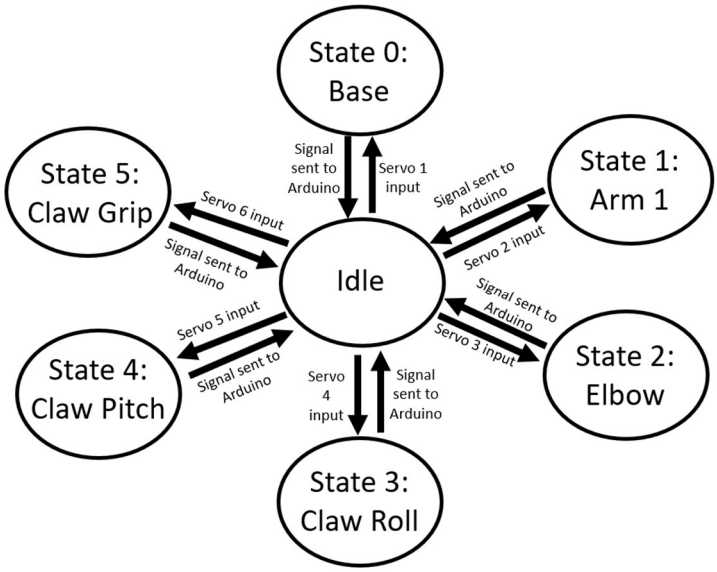
Finite State Machine:

The robotic arm operates off of a finite state machine with 7 states. There is one state for each servo, and then an idle state. When the robot is turned on it enters the idle state. From there, the operator uses the Serial Monitor to input commands in the form of state : angle.

An example input would be “0:45” to set the base servo to 45 degrees. A limitation of this design is that only one servo can move at a time. However, this limitation ensures that adequate power is delivered to the servo. If each servo moves at one, the robot would not be able to lift objects with the mass it was design for (designed for 0.3kg or 11oz).

In the diagram, the first state is State 0. Indexing on the FSD starts at 0 because indexing on the Arduino programming language starts at zero and it is more organized when the states correspond to the inputs.

Figure 6: Finite State Diagram



Mechanical Transmissions:

The transmission system was designed to be low cost, and strong. Most of the parts were 3D printed with the exception of the ball bearings. I used standard 22mm skateboard ball bearings due to their low cost and high strength.

The R1 servo (rotates the base) is slip fit into the base and then screwed into place. The servo arm is screwed into the table / platform. A lazy Susan bearing turn table supports the base which is what supports the weight of the machine. The lay Susan bearing was chosen for its low cost, however, it has some slop which makes the Arm 1 have ~3 degrees of error when passing through 90 degrees.

At the joints R2 (Arm 1 rotation), R3 (elbow rotation), and R5 (claw pitch) the servo motor is directly connected to 3D printed shafts which are pressure fit into the 3D printed arm assembly. There is a shaft on both sides of the arm assembly and each shaft is supported by ball bearings. The ball bearings are force fit into the plastic housing. On one side of the ball bearing there is a internal plastic ridge in the housing, and on the other side there is the shaft which is connected the servo. The servo is screwed into the housing and this prevents the bearing of translating axially.

Figure 7: Simplified CAD of Base Design (R1 and R2)

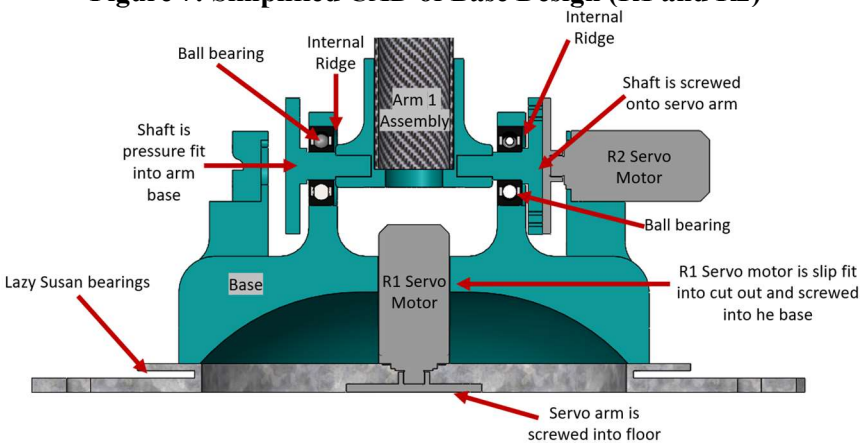
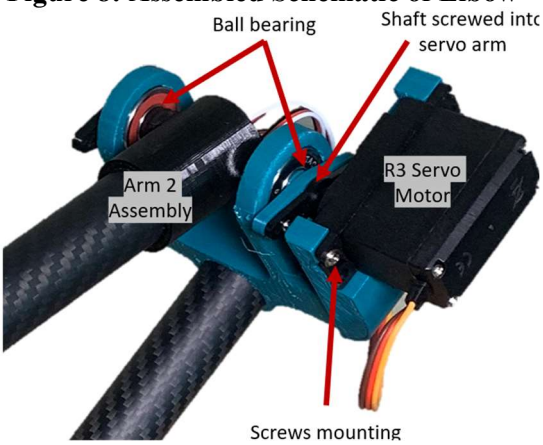


Figure 8: Assembled Schematic of Elbow



To connect the main joints (base to elbow and elbow to claw) I used a carbon fiber rod for due to it's high stiffness to weight ratio. The hollow carbon fiber rods also provides a location to route the servo wires.

At the claw, I used a similar design as the base and elbow to mount the servos. However, in this case the smaller SG90 servos are used. The claw's rolling motion is controlled by the R4 servo, which is directly connected with a 3D printed shaft.

For the opening and closing of the claw, I designed my own gear meshing to have no backlash. One of the claw arms is directly connected to the R6 servo. When the R6 servo rotates, the gear meshing causes both arms to open and close at the same rate (See Figure 9 and 10 for closed and open images).

Figure 9: CAD of Claw Design

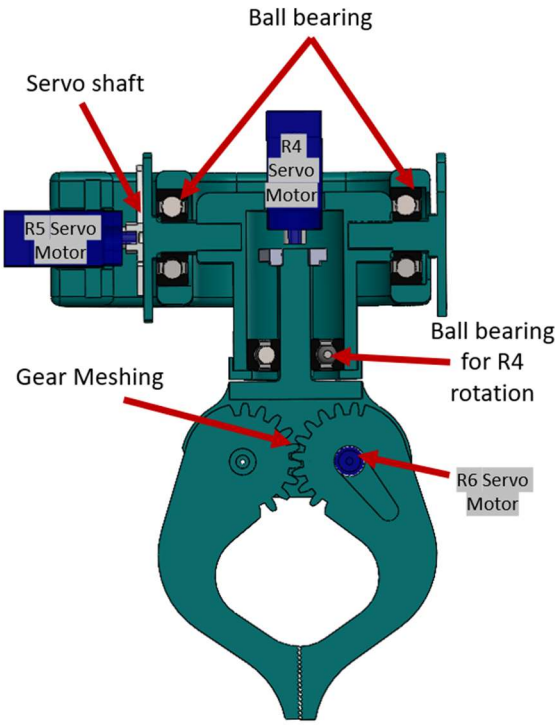


Figure 10: Assembled Claw

Results and Future Work:

After assembly of the robotic arm, the mechanical portions needed to be verified to ensure operation. Using a parametric SolidWorks assembly of the robotic arm, I determined the servo angles that would allow the robot to pick up, pour, and return a bottle of hand sanitizer.

During the testing, the mechanical transmissions functioned as designed, and the use of servos was effective. The robot was able to pick up and return the hand sanitizer in the same location with a position error less than 0.5mm.

Future work for this project includes refining the code to ensure the robot cannot run into it's self, and implementing an XYZ coordinate system to make inputting controls.

Appendix

Appendix 1: Calculations for Max Weight Arm Can Carry (MATLAB Script)

```
%% Max Weight
T=.11;%kg/m
theta=-15; %Arm 1 angle relative to horizon
phi=45; %Elbow angle relative to horizon
omega=-15; %Claw angle relative to vertical
L=.025; % location of object in claw. .025 for middle, .054 for tip
W_elbow=.227; %kg
W_claw=.220; %kg
weight=.1 %mass of claw

Moment_elbow=(.131*sind(theta)+.048)*W_elbow;
Moment_claw=(.131*sind(theta)+.048+.120*cosd(phi))*W_claw;
%
Moment_weight=(.131*sind(theta)+.048+.120*cosd(phi)+(.047+L)*cosd(omega))*we
ight

Max_Weight=(T-Moment_elbow-
Moment_claw)/(.131*sind(theta)+.048+.120*cosd(phi)+(.047+L)*cosd(omega))
```

Appendix 2: Rotation Points Description

Rotation Point	Name	Degrees of Rotation	Servo	State
R1	Base	180	MG995	0
R2	Arm 1	180	MG995	1
R3	Elbow	180	MG995	2
R4	Claw Roll	180	SG90	3
R5	Claw Pitch	180	SG90	4
R6	Claw Grip	100	SG90	5

Appendix 3: Arduino Code

```
#include <Servo.h>

Servo base; //servo1;
//Servo arm2;
Servo arm1; //servo2;
Servo elbow; //servo3;
Servo claw_roll; //servo4;
Servo claw_pitch; //servo5;
Servo claw_grip; //servo6;

int base_pin=8;
//int arm2_pin=7;
int arm1_pin=9;
int elbow_pin=12;
int claw_roll_pin=13;
int claw_pitch_pin=10;
int claw_grip_pin=11;
```

```
void setup() {
  Serial.begin(115200);
//  AttachServos();
  base.attach(base_pin);
  arm2.attach(arm2_pin);
  arm1.attach(arm1_pin);
  elbow.attach(elbow_pin);
  claw_roll.attach(claw_roll_pin);
  claw_pitch.attach(claw_pitch_pin);
  claw_grip.attach(claw_grip_pin);
}

void loop() {
}

void serialEvent() {
  int channel;
  int pos;

  channel = Serial.readStringUntil(':').toInt();
  //pos = Serial.readStringUntil('*').toInt();
  pos = Serial.readString().toInt();
//  servos[channel].write(pos);
  Serial.print(channel  );
  Serial.println(pos);

  switch(channel){
  case 0:
   //servos[0].write(pos);
   base.write(pos);
    Serial.println("Base");
    Serial.println(pos);
   break;
   case 1:
   arm1.write(pos);
   //arm2.write(180-pos);
    Serial.println("Arm 1");
    Serial.println(pos);
   break;
   case 2:
   elbow.write(pos);
    Serial.println("Elbow");
    Serial.println(pos);
   break;
   case 3:
   claw_roll.write(pos);
    Serial.println("Claw Roll");
    Serial.println(pos);
   break;
   case 4:
  claw_pitch.write(pos);
   Serial.println("Claw Pitch");
   Serial.println(pos);
   break;
   case 5:
   claw_grip.write(pos);
```

```
    Serial.println("Claw Grip");
    Serial.println(pos);
    break;
    default:
    break;
}
}
```