Course: ENG ME 360

Assignment: 2.5 degrees of freedom system

Name: Yury Luzhkov

Date: 8th May 2022

Goal of this exercise:

The goal of this assignment was to fully design a 2.5 degrees of freedom mechanism.

Constraints:

We had all the freedom in the world. Our only constraint was that the cost of our parts (excluding parts that could be created or acquired using university resources) did not have to exceed 120 $ in cost.

Design decisions:

Our idea was to create a pointless machine. This funny mechanism is very popular on social media. The name of it stands for itself. It basically involves a button and a mechanism that turns off the button when user actuates the button. We needed to design 2.5 DOF system so we decided to use multiple buttons.

Our mechanism would consist of wooden board, 2 linear stages, 1 end effector stage, and a table for buttons. Please find table below as bill of materials tables for the mechanism.

№ (M)	**№ of items:**	**Major Part Name:**
1	1	Wooden board
2	2	Linear stages
3	1	End effector stage
4	1	Buttons table

Table 1: Bill of materials for major parts

Tables 2-4 summarize the minor parts for the assembly of each of the major parts:

Linear Stage:			
№	**Minor Part Name:**	**Part Property/info: (If needed)**	**№ of Items per assembly:**
1	Motor	NEMA 17	1
2	Metal Bar	8020 Extrusion Length 15 cm (Figure 1)	1
3	Linear Stage Connection	(Figure 2)	1
4	Linear Stage Stand	(Figure 3)	1
5	Linear Stage Slider	(Figure 4)	1
6	Linear Stage Stand 2	(Figure 5)	1
7	Linear Stage Stand 3	(Figure 6)	1
8	¼ inch - 20 Screws	-	14
9	¼ inch - 20 T-nuts	-	13
10	GT-2 Belt	-	1
11	M3 Screws	-	4

Table 2: Summary of parts for the Linear stage.

End Effector:			
№	**Minor Part Name:**	**Part Property/info: (If needed)**	**№ of Items per assembly:**
1	Motor	NEMA 17	1
2	Metal Bar	8020 Extrusion Length 15 cm (Figure 1)	1
3	End Effector Connection_1	(Figure 7)	1
4	End Effector Connection_2	(Figure 8)	1
5	End Effector Slider	(Figure 9)	1
6	End Effector Actuator	(Figure 10)	1
7	M5 Screws	-	4
8	Servo	-	1
9	¼ inch - 20 Screws	-	10
10	¼ inch - 20 T-nuts	-	6
11	GT-2 Belt	-	1
12	M3 Screws	-	4

Table 3: Summary of parts for the End Effector stage.

Buttons Table:			
№	**Minor Part Name:**	**Part Property/info: (If needed)**	**№ of Items per assembly:**
1	Buttons	90013L 12PCS Round Toggle LED Switch (Figure 11)	4
2	The Table itself	(Figure 12)	1
3	¼ inch - 20 Screws	-	4

Table 4: Summary of parts for the Buttons Table.

Wooden board would have ¼ inch holes in it to fix both linear stages and table for buttons to it. The places for the holes were done after overall assembly of the mechanism using hand drill in the classroom.

Figures 1-12 below illustrate the parts Solidworks models. (All of those parts were printed on Boston Universities' FDA 3-D printers.)

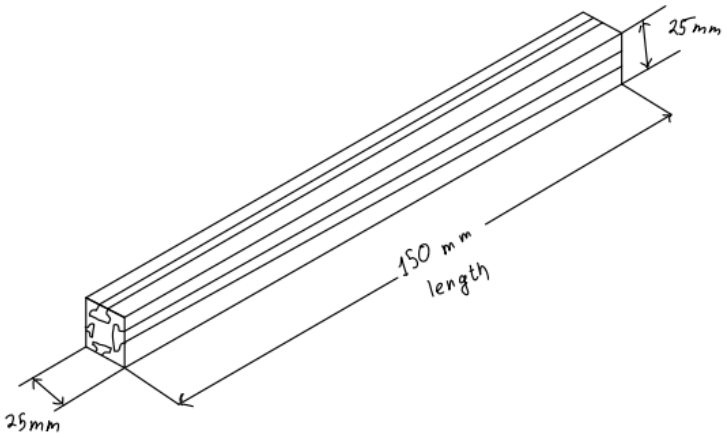


Figure 1: Metal piece Sketch

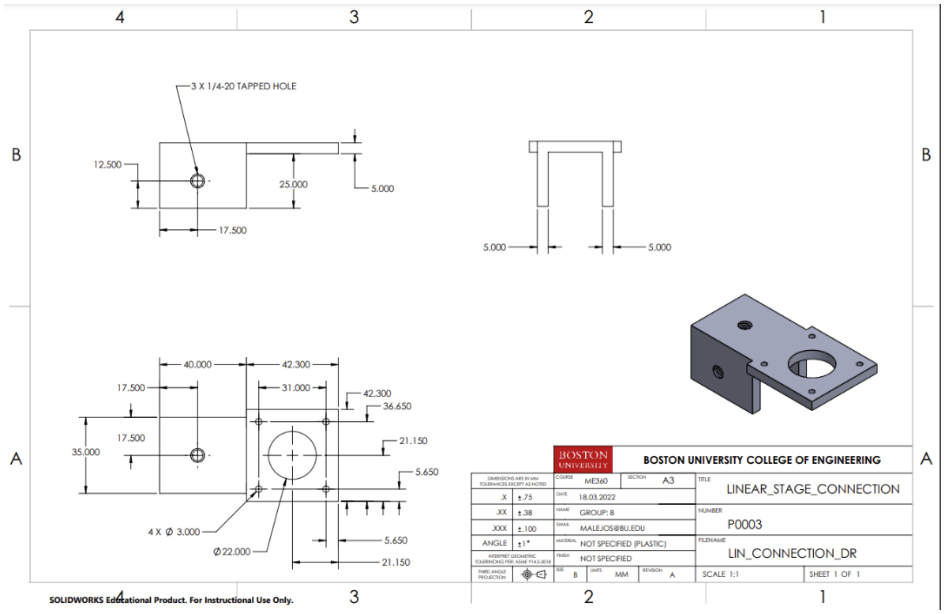


Figure 2: Linear Stage Connection

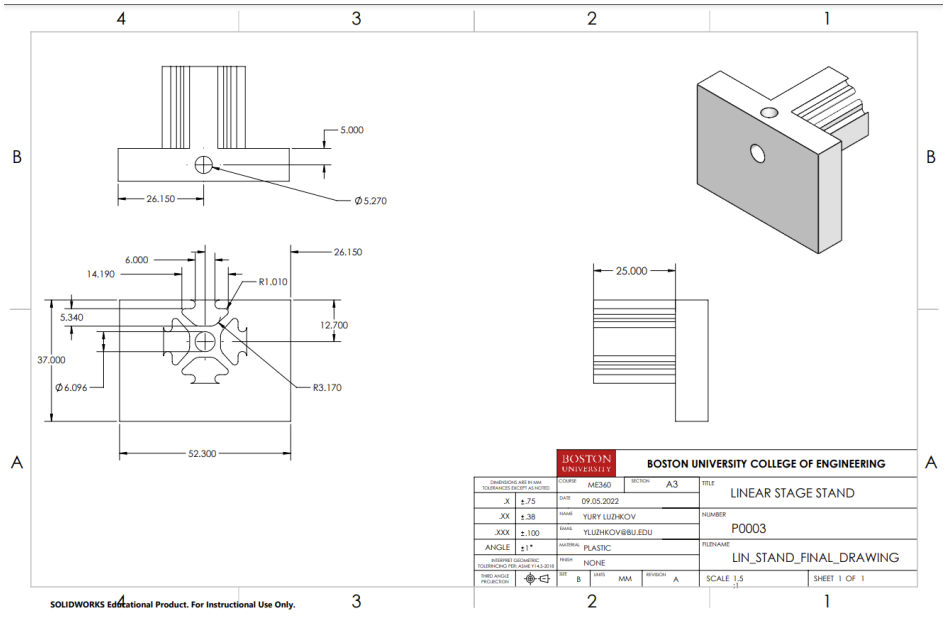


Figure 3: Linear Stage Stand

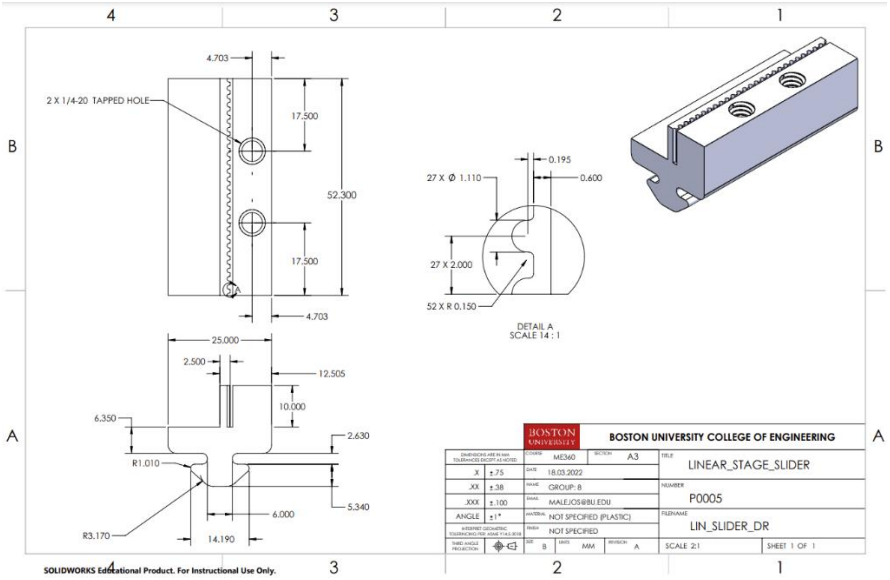


Figure 4: Linear Stage Slider

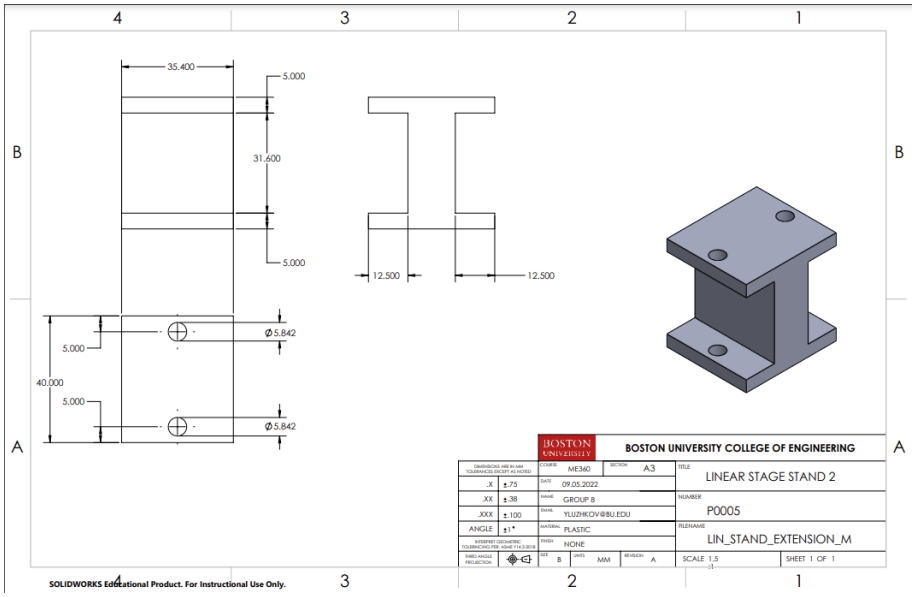


Figure 5: Linear Stage Stand 2

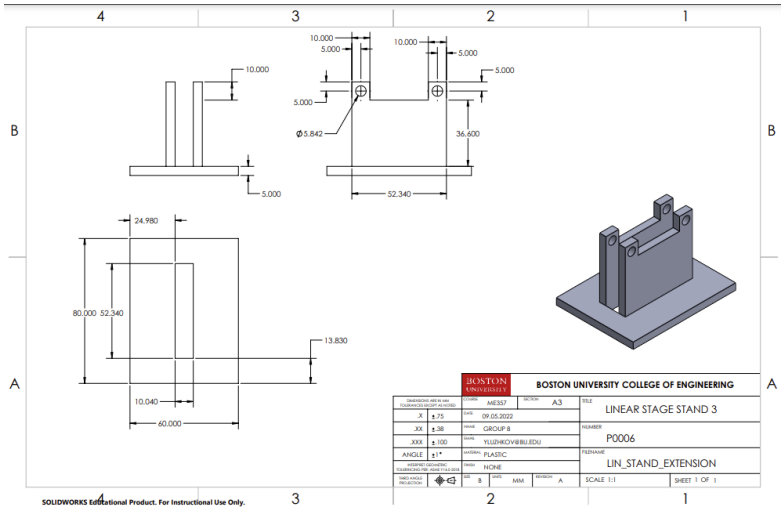


Figure 6: Linear Stage Stand 3

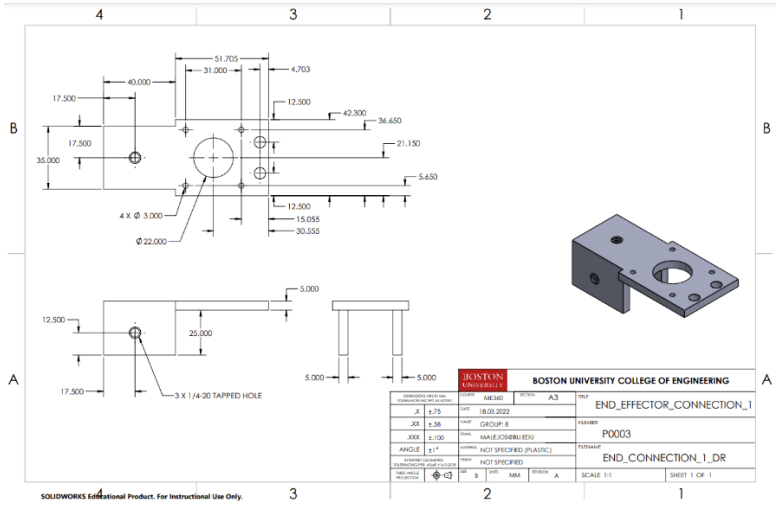


Figure 7: End Effector Connection_1

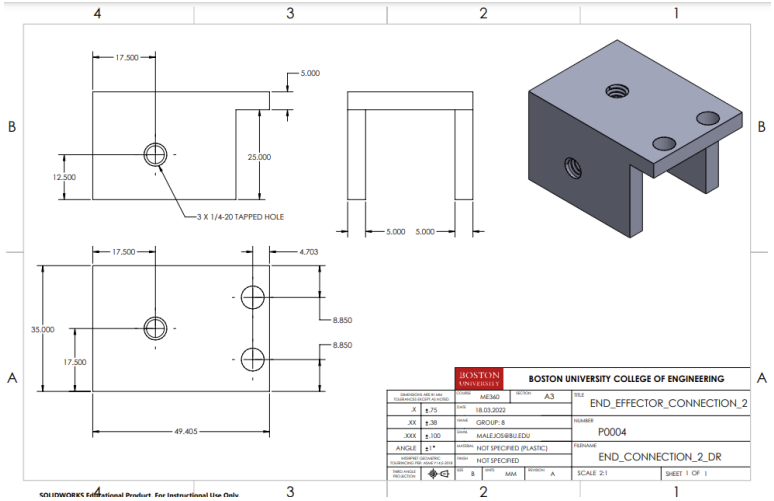


Figure 8: End Effector Connection_2

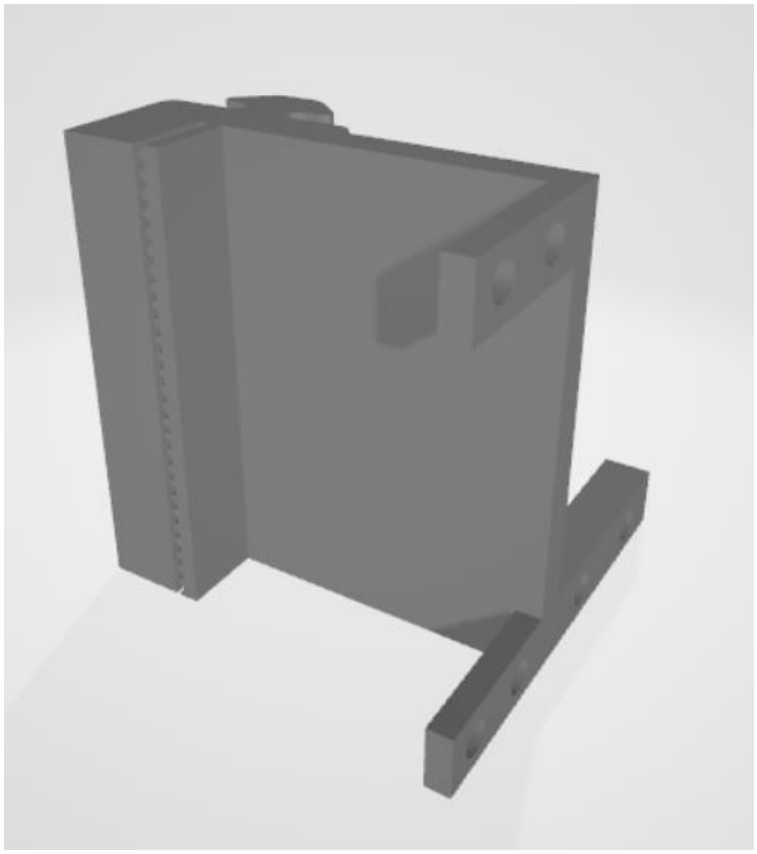


Figure 9: End Effector Slider

Figure 10: End Effector Actuator

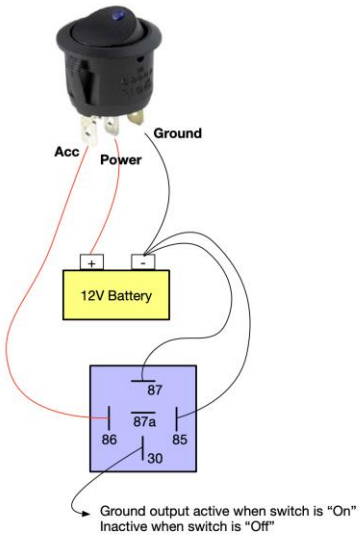


Figure 11: Button picture

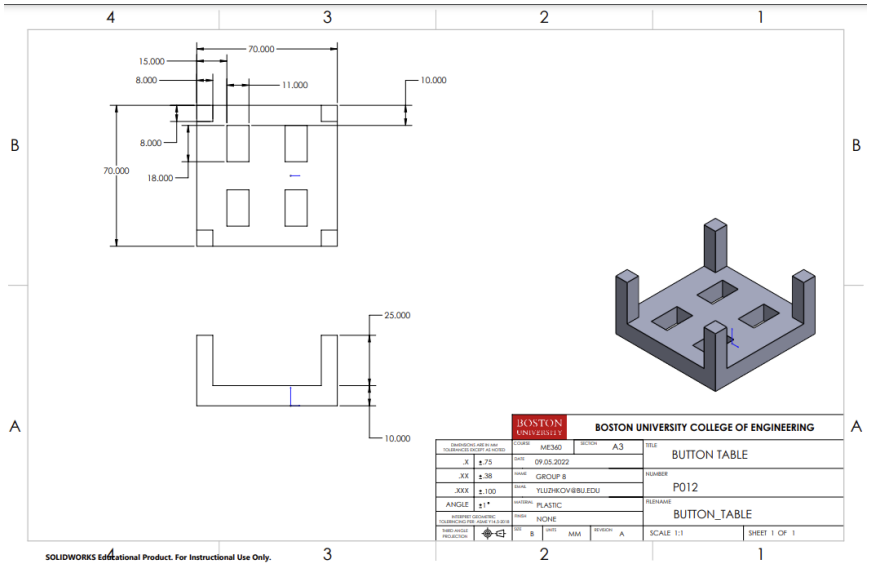
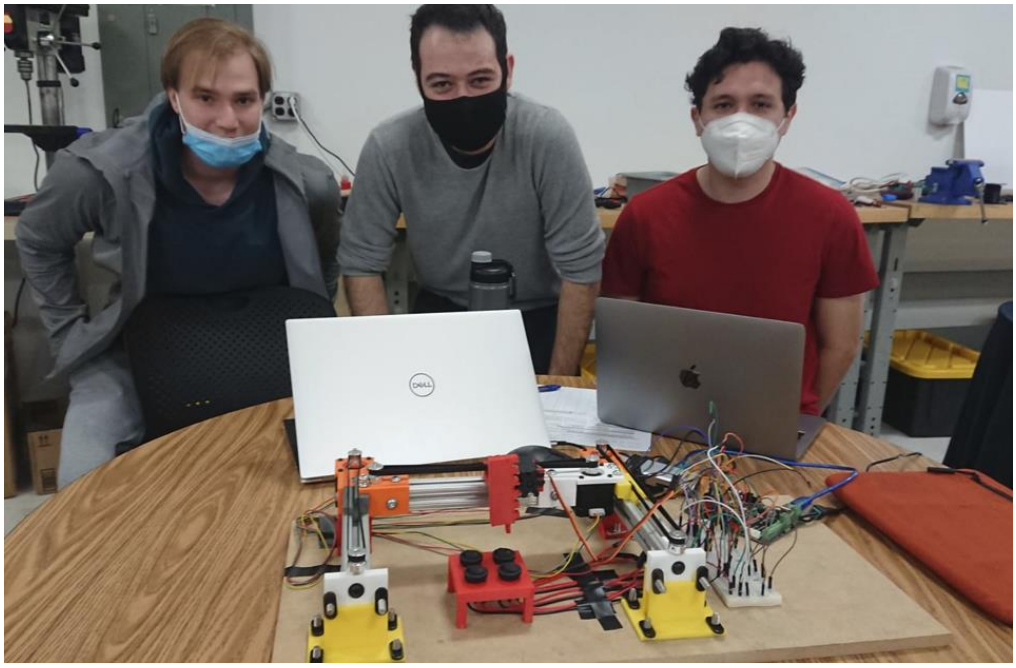


Figure 12: Button table model

Those parts together make the mechanism illustrated in the picture below:



Picture 1: Final Assembly

Button Circuit was designed in the following way. We have connected 4 digitals pins on the Arduino mega board (MKS Base V1.6). When the buttons were turned on, they outputted the voltage on the digital pins. If they were off no voltage passed on digital pins.

Coding:

We have used Arduino app to code our mechanism. We did not use G-code because we were unable to find how to write conditional statements in G-code.

```
#include <Servo.h>

// Buttons
int green_button = 37;
int orange_button = 17;
int blue_button = 23;
int red_button = 27;

// Servo
int servoPin = 2;
Servo Servo1;

// Motor X

const int X_step = 54;
const int X_dir = 55;
const int X_enable = 38;

// Motor Y

const int Y_step = 60;
const int Y_dir = 61;
const int Y_enable = 56;

// Motor Z

const int Z_step = 46;
const int Z_dir = 48;
const int Z_enable = 62;

int Motor_Speed = 150;
// Green Button
int Green_XY = 2650;
int Green_Z = 2800;
// Orange Button
int Orange_XY = 2650;
int Orange_Z = 450;
// Blue Button
int Blue_XY = 4800;
int Blue_Z = 450;
// Red Button
int Red_XY = 4800;
int Red_Z = 2800;
```

First, we defined important variables. Illustrated in the screen captures above. Those include pin numbers and positions of buttons (in number of steps of the motor)

```
void setup() {
  Serial.begin(9600);
  pinMode(green_button,INPUT);
  pinMode(orange_button,INPUT);
  pinMode(blue_button,INPUT);
  pinMode(red_button,INPUT);

  Servo1.attach(servoPin);

  pinMode(X_step,OUTPUT);
  pinMode(X_dir,OUTPUT);
  pinMode(X_enable,OUTPUT);

  pinMode(Y_step,OUTPUT);
  pinMode(Y_dir,OUTPUT);
  pinMode(Y_enable,OUTPUT);

  pinMode(Z_step,OUTPUT);
  pinMode(Z_dir,OUTPUT);
  pinMode(Z_enable,OUTPUT);


}
```

Our void loop (loop that runs only 1 time) is shown above. Here we define the pins to be input, or output pins.

```
void loop() {
  DashedLine();
  Servo1.write(0);
  delay(500);
                                                    //Green Button Perfect
  if (digitalRead(green_button) == HIGH){

    // Green X Y Forward
    digitalWrite(X_dir,LOW);
    digitalWrite(Y_dir,HIGH);
    for(int xy = 0; xy < Green_XY; xy++) {
      digitalWrite(X_step,HIGH);
      digitalWrite(Y_step,HIGH);
      delayMicroseconds(Motor_Speed);
      digitalWrite(Y_step,LOW);
      digitalWrite(X_step,LOW);
      delayMicroseconds(Motor_Speed);
    }
    delay(250);

    // Green Z Forward
    digitalWrite(Z_dir,LOW);
    for(int z = 0; z < Green_Z; z++){
      digitalWrite(Z_step,HIGH);
      delayMicroseconds(Motor_Speed);
      digitalWrite(Z_step,LOW);
      delayMicroseconds(Motor_Speed);
    }
    delay(250);

    // Green Servo Motor
    Servo1.write(0);
    delay(250);
    Servo1.write(30);
    delay(500);
    Servo1.write(0);
    delay(500);

    // Green Z Backwards
    digitalWrite(Z_dir,HIGH);
    for(int z = 0; z < Green_Z; z++){
      digitalWrite(Z_step,HIGH);
      delayMicroseconds(Motor_Speed);
      digitalWrite(Z_step,LOW);
      delayMicroseconds(Motor_Speed);
    }
    delay(250);

    // Green X Y Backward
    digitalWrite(X_dir,HIGH);
    digitalWrite(Y_dir,LOW);
    for(int xy = 0; xy < Green_XY; xy++) {
      digitalWrite(X_step,HIGH);
      digitalWrite(Y_step,HIGH);
      delayMicroseconds(Motor_Speed);
      digitalWrite(Y_step,LOW);
      digitalWrite(X_step,LOW);
      delayMicroseconds(Motor_Speed);
    }
  }
```

Finally, above find the screen capture of conditional code for Green Button (other button codes are very similar except they use their own position and pin values. The code above basically says:

If digital pin connected to green button has voltage,

then move X position to (green button) move Y position to (green button), actuate servo motor, and return to original position.

Modeling and optimizations:

Our design turned out to be successful. Talking mechanically, we had a little issue on one of the linear stages because metal excursion provided to us had a hole in the middle of it. As the linear slider passed above the hole some additional friction was experienced on it. In the beginning the mechanism experienced so much friction that the belt started sliding and the platform was not moving where it should. However, after many tries the platform fixed itself and started working perfectly. Speaking electrically everything worked but was done not in the most energy efficient way and code smart way.

Conclusions:

One sphere of improvement could be the code. We could improve the code. Our mechanism does not track the current position of the actuator thus cannot turn of multiple buttons at 1 go. After it turns 1 button it comes back to the origin and so on. Another area of improvement could be the circuit for the buttons. As you can see in picture 1, our circuit has a lot of jumper wires. I am shure there is a way to make circuit simpler. Speaking mechanically, we for shure did not take the most material saving path. There are many ways to decrease the amount of material used in the assembly of similar accomplishment

mechanism. But over all our project has accomplished all the expectations we had. It is a valid 2.5 degrees of freedom mechanism.