FINAL LAB REPORT 102B

Group: 19

Sameer Khan, Sukhpal Ghotra, Jorge Pool, Elijah Priwer

Department of Mechanical Engineering, University of California at Berkeley

Due: December 14th, 2023

Opportunity

In the modern world, digital devices are rapidly replacing the traditional pen and paper for art, note-taking, and many other types of writing in the physical medium. However, there remain many cases when physical drawing remains crucial. The options for translating digital to physical drawing remains limited largely to printers, but these cannot operate in real-time. Therefore, we wished to create a device that could translate drawings from a digital to physical medium both synchronously and asynchronously. We believe that this has applications for the elderly, in education, and in art.

High-Level Strategy

Developed a device capable of transcribing user input onto paper. The device contains 3 degrees of freedom with seamless X and Y movement for writing anywhere on the paper. A retractable pen attached to an arm capable of being adjusted in the z-axis based on force sensor readings. An automatic paper dispenser triggered by user button commands.

The overall frame of the device largely mimics that of a 3D printer but in 2 dimensions. Dual stepper motors use belts to pull an arm along the 2 dimensions in movements that seamlessly combine motion along both axes. This arm can move in an additional 3rd axis (the Z axis) that allows for the pen to be moved onto and off of the paper. This serves an additional purpose as it allows for the modulation of pressure applied to the pen and therefore the line thickness of the drawing. A pressure sensor located in the arm allows for this pressure to be modulated and controlled, but in order for this pressure sensor to sense the change in pressure on the pen, the pen holder is located on an elevator with a spring attached to a piece of rubber pushing onto the pressure sensor. This means that as the arm is moved down, the pressure on the pen tip increases and is transmitted upwards through the spring to the pressure sensor and controlled.

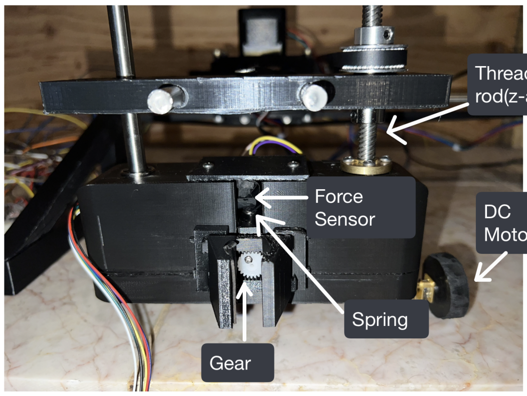
The pen holder is made up of 2 clamping jaws moved by a gear. The gear is modulated via a shaft by a servo motor located on the other end of the elevator in order to balance out the weight distribution on the elevator. We also located a pen ejector on the arm, made up of a dc motor and a rubber wheel. When we want the paper to be ejected, the rubber wheel is lowered onto it and spun, removing the paper from the drawing area.

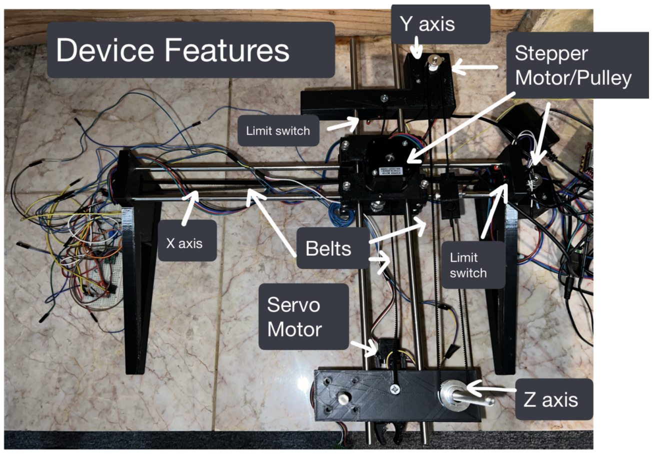
Limit switches located on the x and y axes allow for us to zero out the location of the arm, while a central housing unit controls the movements via a series of linear bearings.

Desired vs Achieved Functionality

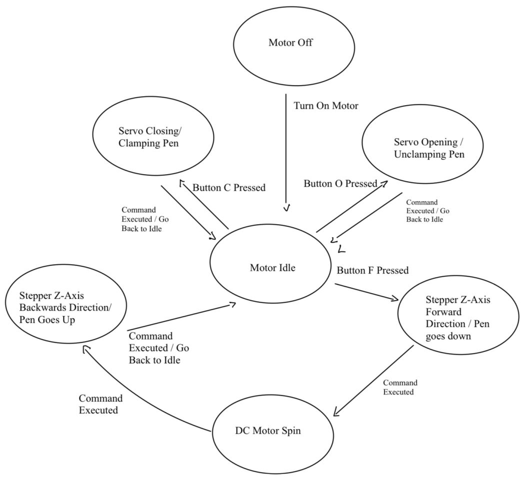
We initially wanted the device to have a number of functions we did not eventually realize, including a method for changing the pen type automatically and automated pairing between a tablet drawing and the device. These were largely due to time constraints on our part. We also wanted to achieve greater writing stability, which is not entirely quantifiable but includes wobble and positional drift of the pen in respect to the paper. This was in part due to the type of materials we used.

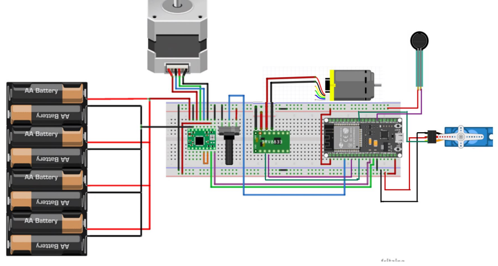
Device Diagram

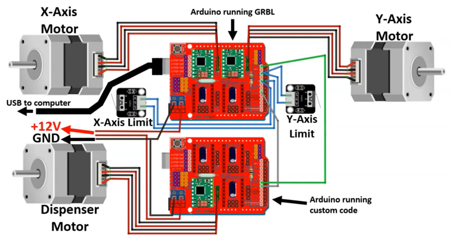




Circuit Diagram and State Diagrams


Motor Off
Turn On Motor
Servo Closing/ Clamping Pen
Servo Opening / Unclamping Pen
Button C Pressed
Button O Pressed
Command Executed / Go Back to Idle
Command Executed / Go Back to Idle
Motor Idle
Button F Pressed
Stepper Z-Axis Backwards Direction/ Pen Goes Up
Stepper Z-Axis Forward Direction / Pen goes down
Command Executed / Go Back to Idle
Command Executed
Command Executed
DC Motor Spin





Function-Critical Decision

Material: We decided to use 3D-printed PLA for most of the device's custom-manufactured parts, as we believed it would give acceptable tolerances for what we needed while being both cheap and easy to manufacture.

DC Motor Location: We originally planned to use the DC motor for modulating the claw on the arm of the device. However, we decided to use a servo motor instead, as it allows for a more precise position and can more easily resist being moved. We instead located the DC motor as a paper-ejector, placing it on the arm after confirming via below calculations that it would not cause the device to tip over.

Force Sensor Location: We decided to place the force sensor at the end of a shaft with a spring connected to an elevator with the claws and servo attached. This allows for a variable and slowly increasing amount of pressure to be applied to the force sensor as the z axis moves the arm downward, and therefore allows for the control of the pressure exerted on the pen. We related the amount of space required for the elevator to the quantity of force via the linear spring constant equation $F = kx$ whereby F is force, k is the spring constant, and x is the distance.

Function-Critical Calculations

Pulley System Torque:
We needed to decide how strong our stepper motors should be to reliably and smoothly move our housing unit along both axes. Our required torque was calculated as follows:

$\tau = r * w =$ **0.16 Nm**

From this we decided to use stepper motors with a maximum torque of $0.45\ Nm$, a factor of safety of ~3.

Values for Pulley System Torque Calculations		
Symbol	Meaning	Value
τ	Torque Required	0.16 Nm
r	Pulley System Radius	16 mm
w	Housing Weight	9.81 N

Equilibrium Calculations:
We needed to ensure that our device did not tip over when fully extended. We therefore performed 2 calculations for the torque in either direction.

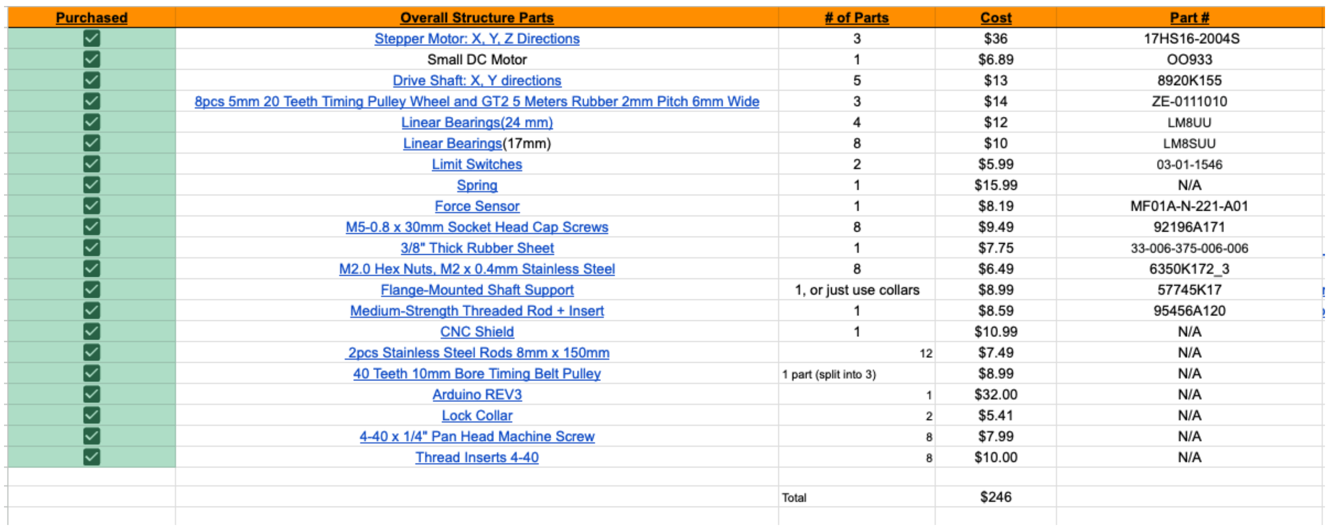
Torque Clockwise
$\tau = (r_{housing} w_{housing}) + 2(r_{stepper} w_{stepper}) + (r_{rod1} + r_{rod2}) w_{rod} + (r_{plate} w_{plate}) =$ **2.2 Nm**

Torque Counterclockwise
$\tau = 2(r_{rod3} w_{rod}) + 2(r_{arm} w_{arm}) =$ **2.13 Nm**
We discovered that the assembly would be acceptably well balanced.

Values for Equilibrium Calculations		
Symbol	Meaning	Value
τ	Overall Torque applied by parts	2.2 Nm
$r_{housing}$	Housing Distance	0.08 m
$w_{housing}$	Housing Weight	9.81 N
$r_{stepper}$	Stepper Distance	0.08 m
$w_{housing}$	Stepper Weight	3.0411 N
$r_{rod,1}$	Rod 1 Distance	0.0910 m
$r_{rod,2}$	Rod 2 Distance	0.0502 m
w_{rod}	Rods Weight	4.091 N
r_{plate}	Plates Distance	0.08 m
w_{plate}	Plates Weight	7.955 N
r_{rod3}	CCW Distance	0.1024 N
r_{arm}	Arm Distance	0.205 m
w_{arm}	Arm Weight	6.58 N

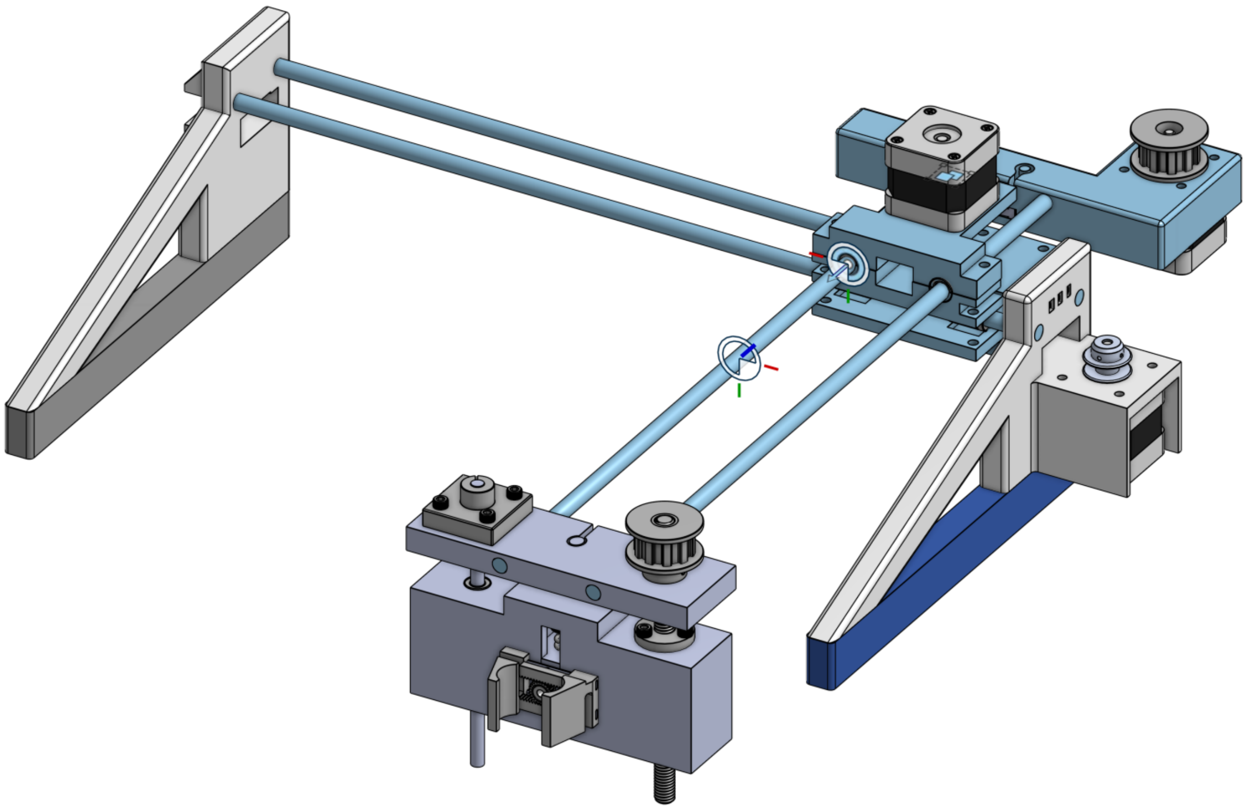
Reflection and Future Ideas
Reflecting on our overall project experience, we would emphasize the importance of taking into account proper tolerances when manufacturing the device. We largely used 3d printed material, which has a very low tolerance. This worked quite well for the larger parts, but did not work well for smaller, moving parts in our device. If we were to do it again, we would have switched to a finer material, like aluminum, which likely would have saved us a great deal of time and vastly improved the quality of our final product.

BOM

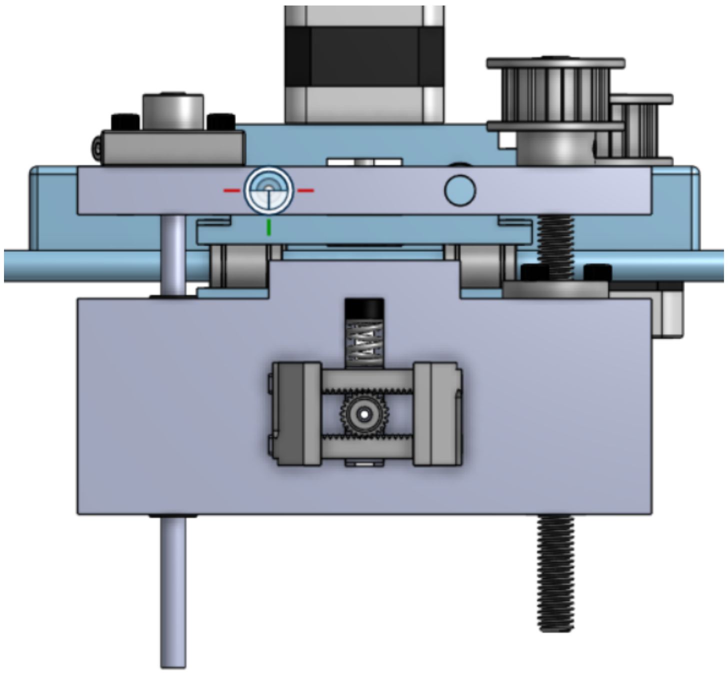


Purchased	Overall Structure Parts	# of Parts	Cost	Part #
✓	Stepper Motor: X, Y, Z Directions	3	$36	17HS16-2004S
✓	Small DC Motor	1	$6.89	OO933
✓	Drive Shaft: X, Y directions	5	$13	8920K155
✓	8pcs 5mm 20 Teeth Timing Pulley Wheel and GT2 5 Meters Rubber 2mm Pitch 6mm Wide	3	$14	ZE-0111010
✓	Linear Bearings(24 mm)	4	$12	LM8UU
✓	Linear Bearings(17mm)	8	$10	LM8SUU
✓	Limit Switches	2	$5.99	03-01-1546
✓	Spring	1	$15.99	N/A
✓	Force Sensor	1	$8.19	MF01A-N-221-A01
✓	M5-0.8 x 30mm Socket Head Cap Screws	8	$9.49	92196A171
✓	3/8" Thick Rubber Sheet	1	$7.75	33-006-375-006-006
✓	M2.0 Hex Nuts, M2 x 0.4mm Stainless Steel	8	$6.49	6350K172_3
✓	Flange-Mounted Shaft Support	1, or just use collars	$8.99	57745K17
✓	Medium-Strength Threaded Rod + Insert	1	$8.59	95456A120
✓	CNC Shield	1	$10.99	N/A
✓	2pcs Stainless Steel Rods 8mm x 150mm	12	$7.49	N/A
✓	40 Teeth 10mm Bore Timing Belt Pulley	1 part (split into 3)	$8.99	N/A
✓	Arduino REV3	1	$32.00	N/A
✓	Lock Collar	2	$5.41	N/A
✓	4-40 x 1/4" Pan Head Machine Screw	8	$7.99	N/A
✓	Thread Inserts 4-40	8	$10.00	N/A
		Total	$246	

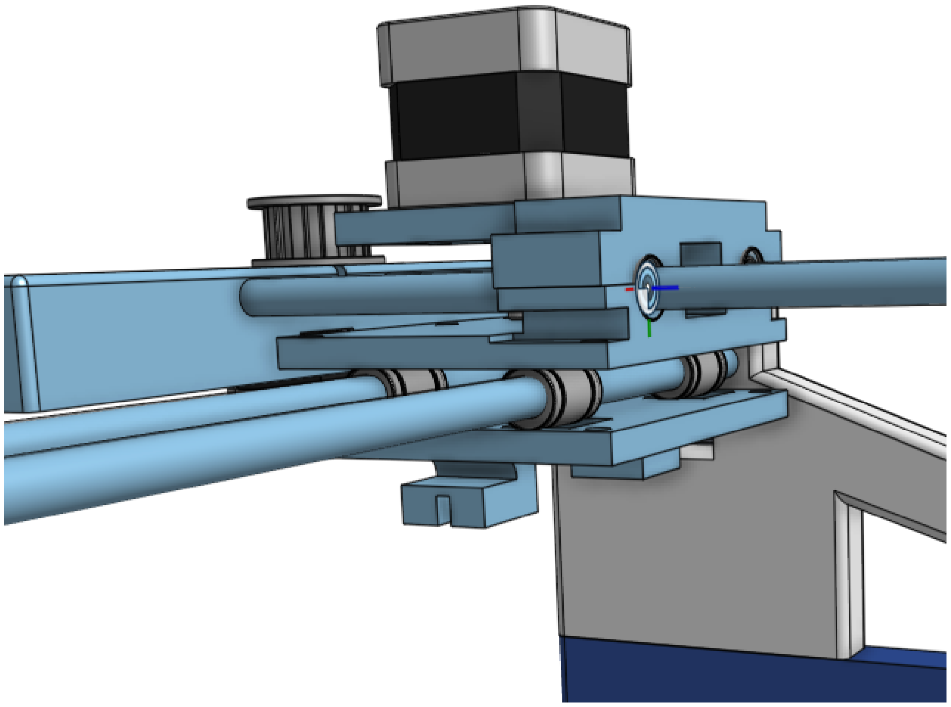
FULL CAD

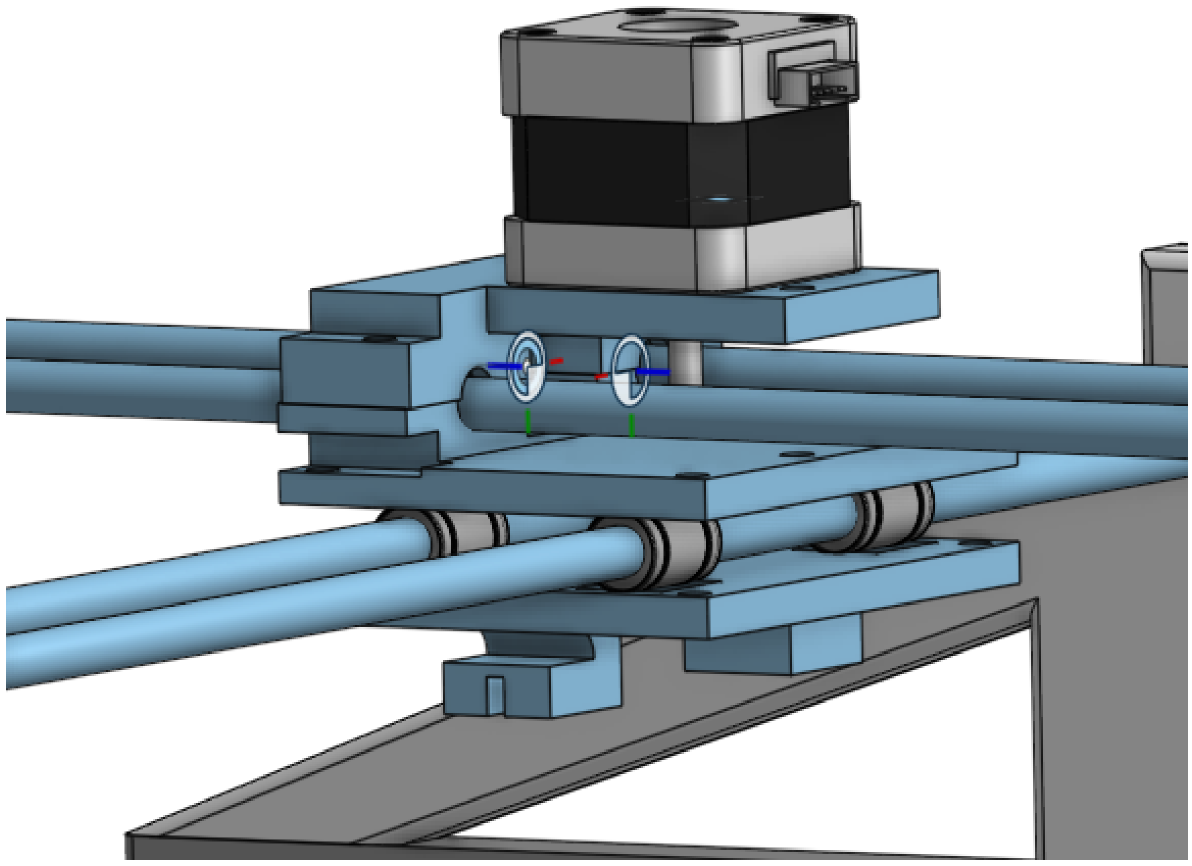


ARM CAD

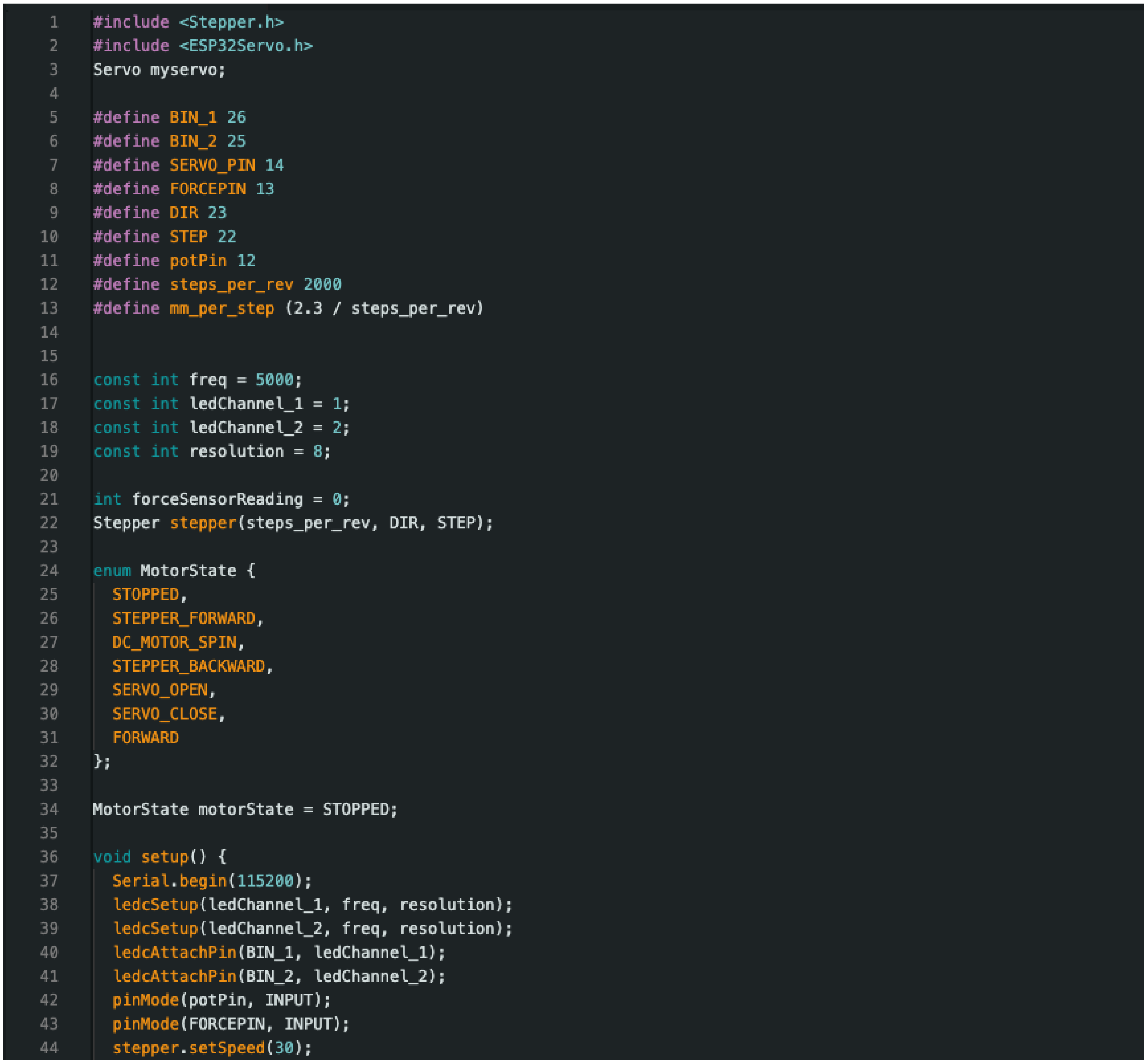


Housing CAD (x and y-axis movement)

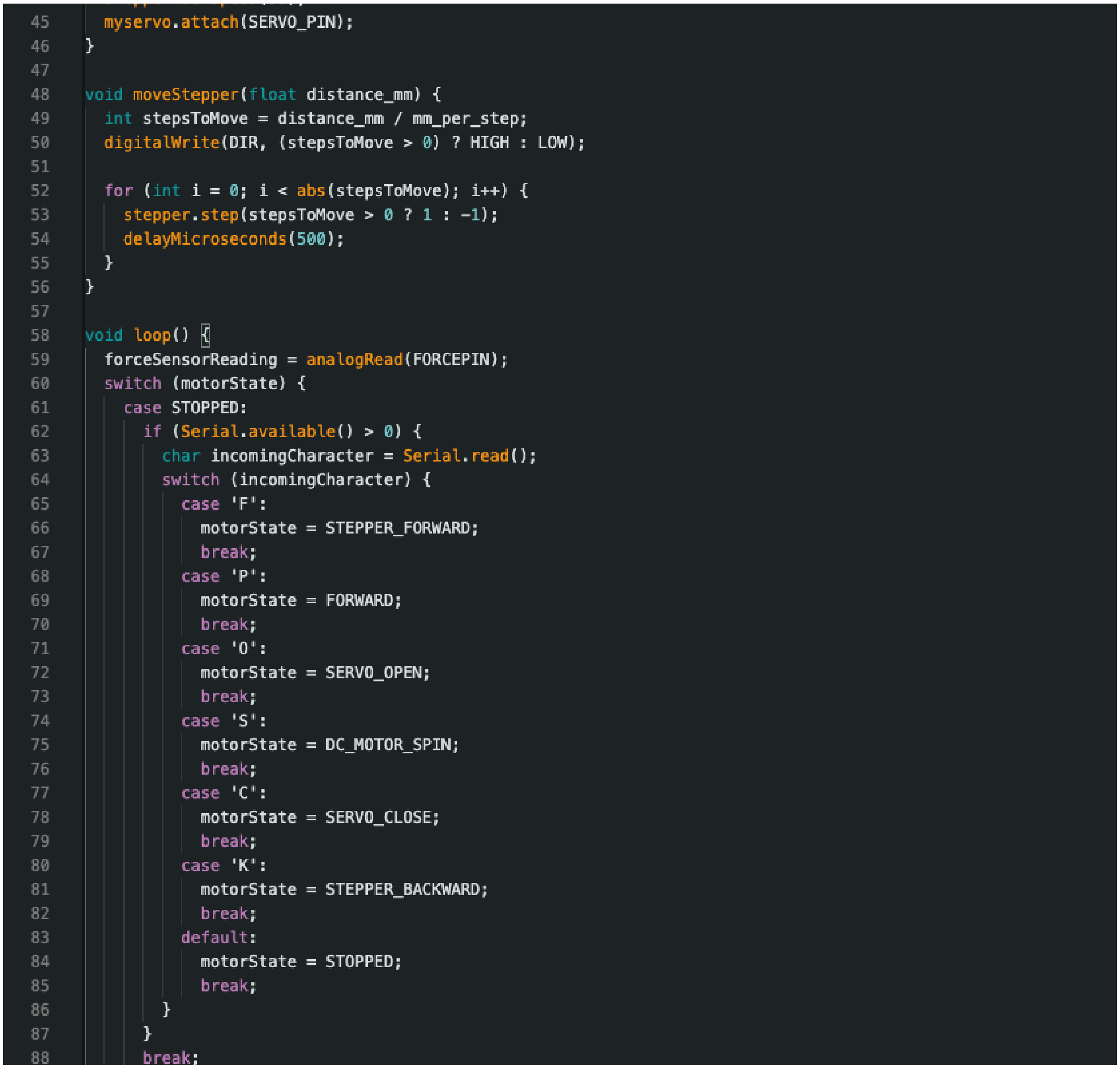




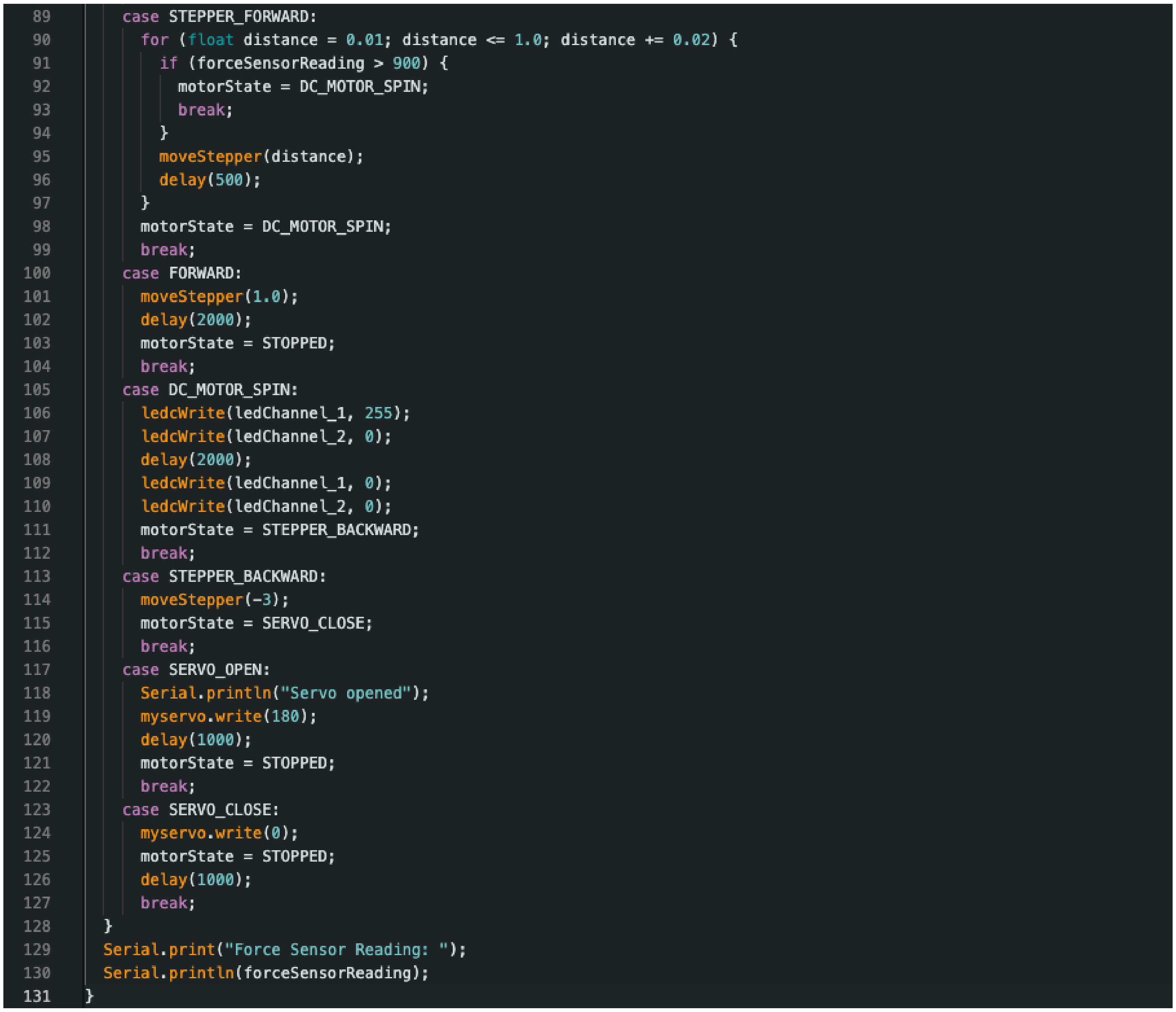
CODE:



```
#include <Stepper.h>
#include <ESP32Servo.h>
Servo myservo;

#define BIN_1 26
#define BIN_2 25
#define SERVO_PIN 14
#define FORCEPIN 13
#define DIR 23
#define STEP 22
#define potPin 12
#define steps_per_rev 2000
#define mm_per_step (2.3 / steps_per_rev)


const int freq = 5000;
const int ledChannel_1 = 1;
const int ledChannel_2 = 2;
const int resolution = 8;

int forceSensorReading = 0;
Stepper stepper(steps_per_rev, DIR, STEP);

enum MotorState {
  STOPPED,
  STEPPER_FORWARD,
  DC_MOTOR_SPIN,
  STEPPER_BACKWARD,
  SERVO_OPEN,
  SERVO_CLOSE,
  FORWARD
};

MotorState motorState = STOPPED;

void setup() {
  Serial.begin(115200);
  ledcSetup(ledChannel_1, freq, resolution);
  ledcSetup(ledChannel_2, freq, resolution);
  ledcAttachPin(BIN_1, ledChannel_1);
  ledcAttachPin(BIN_2, ledChannel_2);
  pinMode(potPin, INPUT);
  pinMode(FORCEPIN, INPUT);
  stepper.setSpeed(30);
```



```
  myservo.attach(SERVO_PIN);
}

void moveStepper(float distance_mm) {
  int stepsToMove = distance_mm / mm_per_step;
  digitalWrite(DIR, (stepsToMove > 0) ? HIGH : LOW);

  for (int i = 0; i < abs(stepsToMove); i++) {
    stepper.step(stepsToMove > 0 ? 1 : -1);
    delayMicroseconds(500);
  }
}

void loop() {
  forceSensorReading = analogRead(FORCEPIN);
  switch (motorState) {
    case STOPPED:
      if (Serial.available() > 0) {
        char incomingCharacter = Serial.read();
        switch (incomingCharacter) {
          case 'F':
            motorState = STEPPER_FORWARD;
            break;
          case 'P':
            motorState = FORWARD;
            break;
          case 'O':
            motorState = SERVO_OPEN;
            break;
          case 'S':
            motorState = DC_MOTOR_SPIN;
            break;
          case 'C':
            motorState = SERVO_CLOSE;
            break;
          case 'K':
            motorState = STEPPER_BACKWARD;
            break;
          default:
            motorState = STOPPED;
            break;
        }
      }
      break;
```



```
    case STEPPER_FORWARD:
      for (float distance = 0.01; distance <= 1.0; distance += 0.02) {
        if (forceSensorReading > 900) {
          motorState = DC_MOTOR_SPIN;
          break;
        }
        moveStepper(distance);
        delay(500);
      }
      motorState = DC_MOTOR_SPIN;
      break;
    case FORWARD:
      moveStepper(1.0);
      delay(2000);
      motorState = STOPPED;
      break;
    case DC_MOTOR_SPIN:
      ledcWrite(ledChannel_1, 255);
      ledcWrite(ledChannel_2, 0);
      delay(2000);
      ledcWrite(ledChannel_1, 0);
      ledcWrite(ledChannel_2, 0);
      motorState = STEPPER_BACKWARD;
      break;
    case STEPPER_BACKWARD:
      moveStepper(-3);
      motorState = SERVO_CLOSE;
      break;
    case SERVO_OPEN:
      Serial.println("Servo opened");
      myservo.write(180);
      delay(1000);
      motorState = STOPPED;
      break;
    case SERVO_CLOSE:
      myservo.write(0);
      motorState = STOPPED;
      delay(1000);
      break;
  }
  Serial.print("Force Sensor Reading: ");
  Serial.println(forceSensorReading);
}
```