Principles of Integrated Engineering (PIE)

Mini-Project 2 - Section 2: Group 16

Ian Walsh, Mark Belanger, Sparsh Gupta

Franklin W. Olin College of Engineering

October 15, 2023

Introduction

We created a 3D scanner using an infrared distance sensor. The sensor captures a rang of orientation angles when mounted on two hobby servo motors. The hobby servo motors implement a pan/tilt mechanism to capture the angles of orientation and the infrared distance sensor determines the distance of the sensor from a surface. This can that be plotted in 3D to reveal a plot of the object being scanned.

Implementation and Functionality

Hardware in Use

- Sharp GP2Y0A02YK0F IR distance sensor
- Hobbyking HK15138 servo motors
- Arduino UNO R3

Sensor Calibration

Our method for testing and calibrating the IR sensor was moving the sensor a known distance away from the wall and measuring the output voltage. We did this with nine different distances ranging from 25cm to 60cm and these distances were measured with a ruler. We started with distances greater than 20cm because the datasheet for the IR distance sensor indicates the consistent changes in voltage after about 20cm and this makes it possible to create a transfer function.

Software Implementation

The software part involved writing code for Arduino that would process the analog inputs obtained from the infrared sensors and servo motors to useful pan angle, tilt angle, and distance data and output that to the Serial port. The Arduino code utilized the inbuilt servo.h library to rotate the servo motors and take a scan. Then, the python code was used to read the angles and distance data from the serial port. The data was converted from spherical coordinates to Cartesian (x,y,z) coordinates using standard trigonometry transfer functions. Finally, these Cartesian coordinates were used to visualize a 3D scatter plot.

Calibration Function

$$distance = 286 - 43 * \ln(voltage)$$

This equation states that the distance of an object from the sensor in centimeters is proportional to the natural log of the output voltage (V). This logarithmic relationship is consistent with the data given in the datasheet for the sensor.

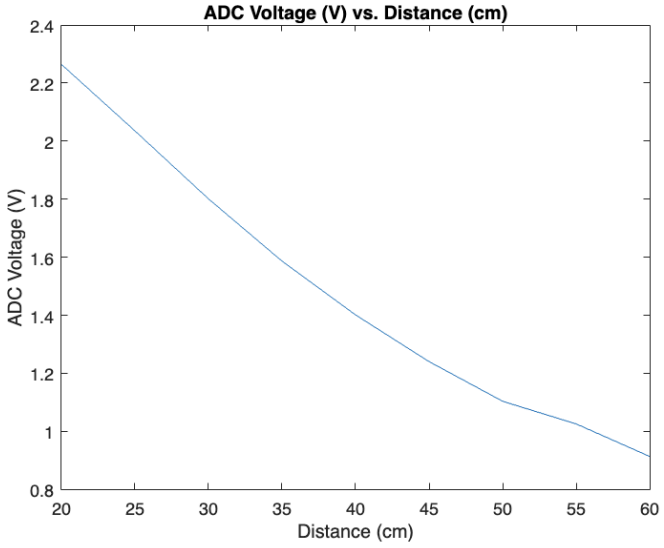

Figure 1: Calibration Function Plot

Error Validation

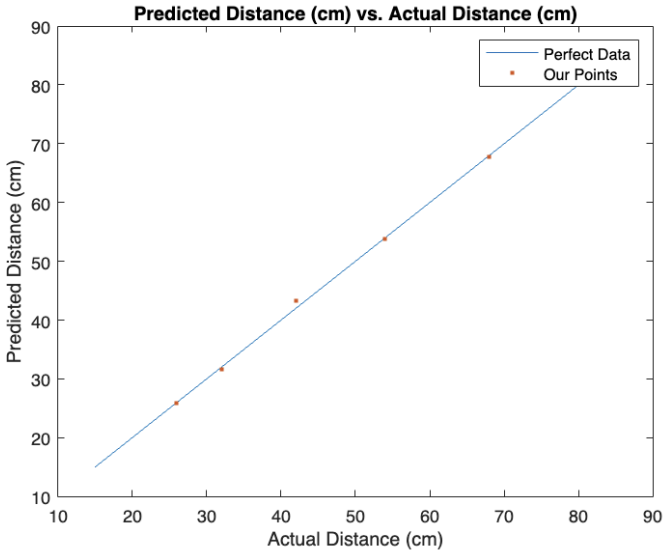

Figure 2: Error Validation

Plot of 1 servo scan

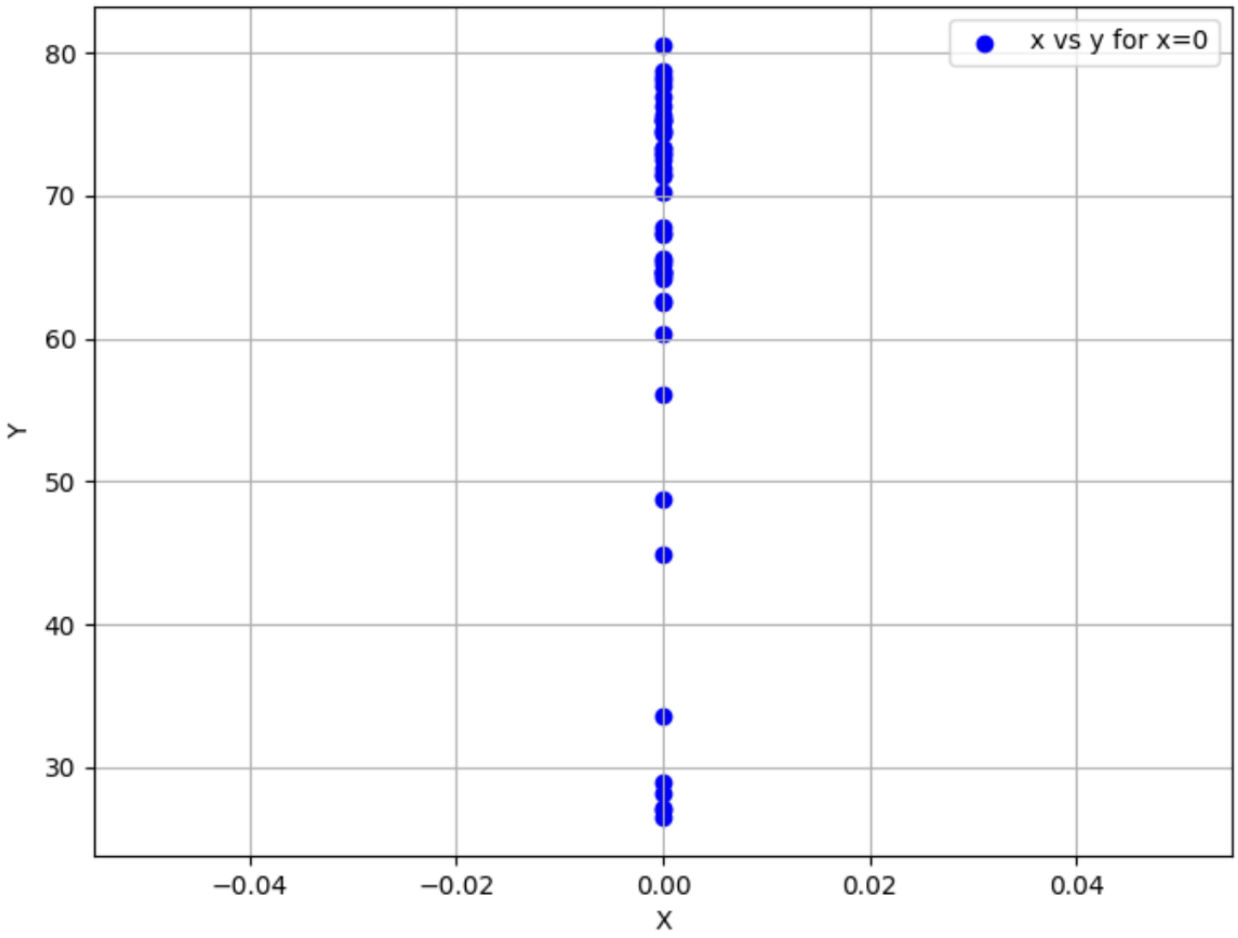

Figure 3: 1 Servo Scan

Plot of 2 servo scan with the letter 'W'

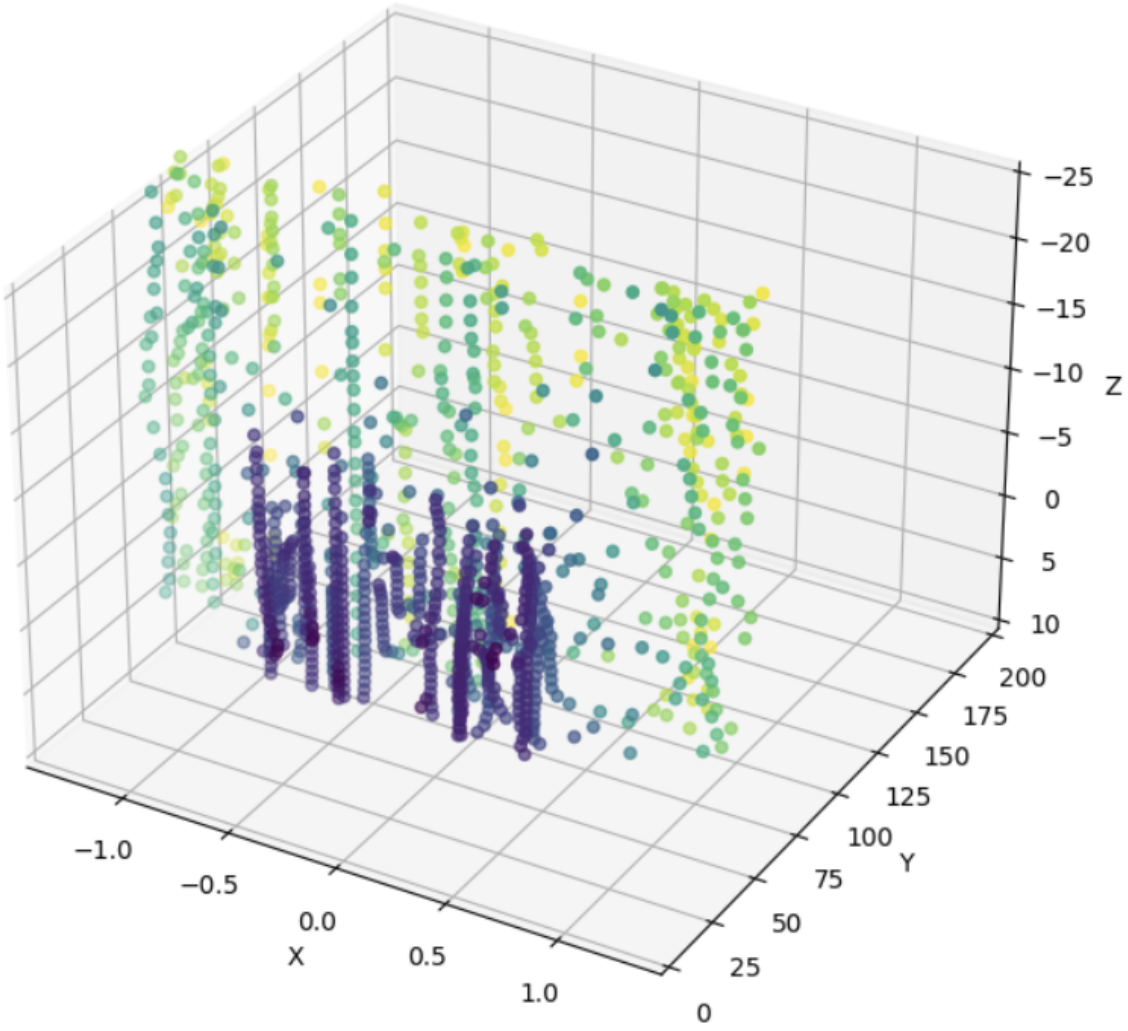

Figure 4: 2 Servo Scan with letter 'W'

Image of setup

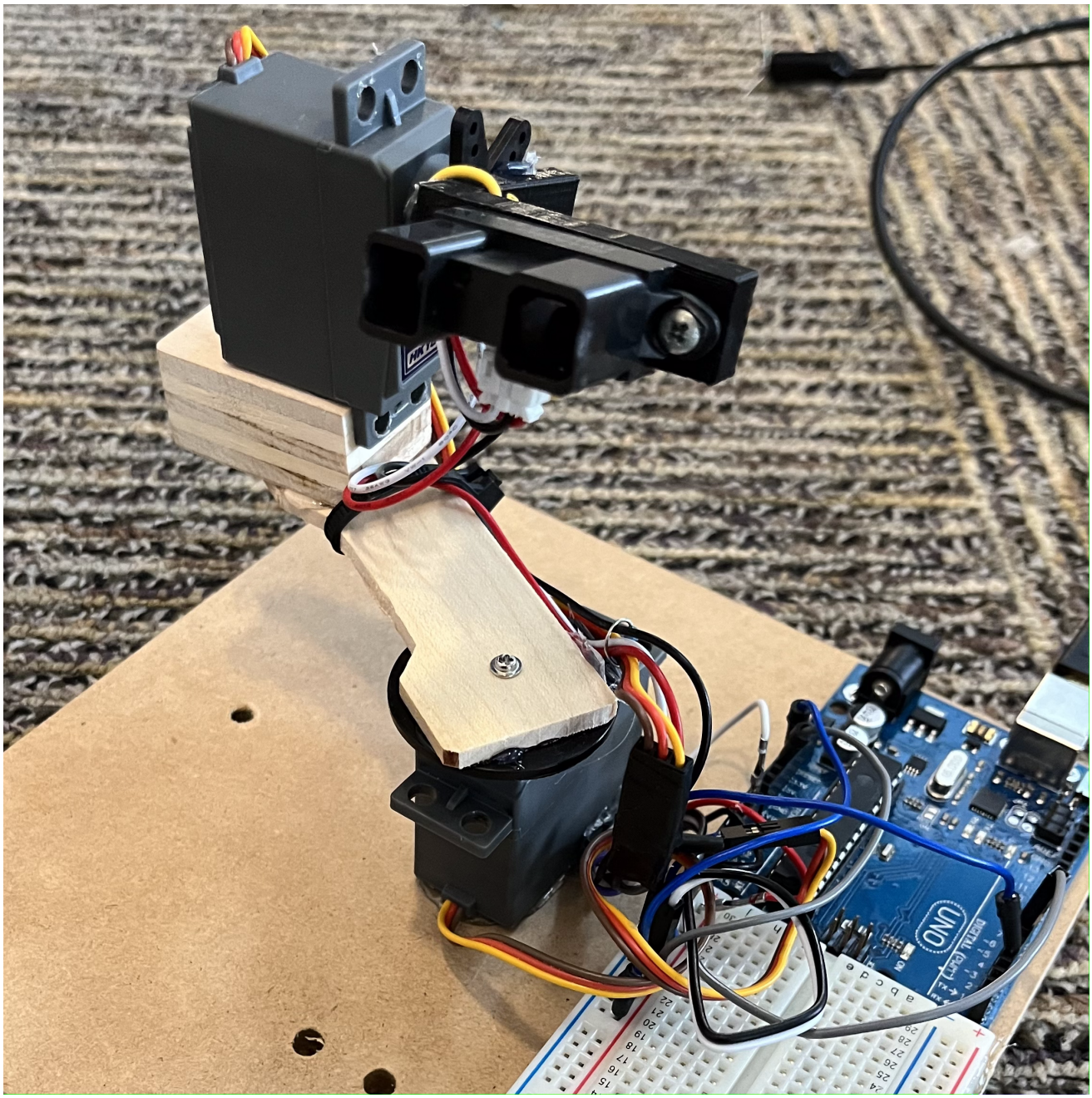

Figure 5: Image of setup for one and two servo scan

Mechanical Design

The bottom servo, the pan servo, is mounted to the base made out of MDF. An arm, cut from two pieces of plywood, sanded, and secured using screws, is attached to the pan servo so that a tilt servo can be mounted to it. Attached to the tilt servo is the IR sensor. The tilt servo is positioned on the arm so that the IR sensor is centered over the axis of rotation of the pan motor. The end effector was made from PLA and secured to the hand servo using a screw. The IR distance sensor was also attached using screws. The breadboard and Arduino were secured to the MDF base plate with hot glue. Wires were secured along the arm using a combination of zip ties and glue. Wire routing was chosen to have the least flexure/stretch from servo hand/wrist movement.

Circuit Diagram

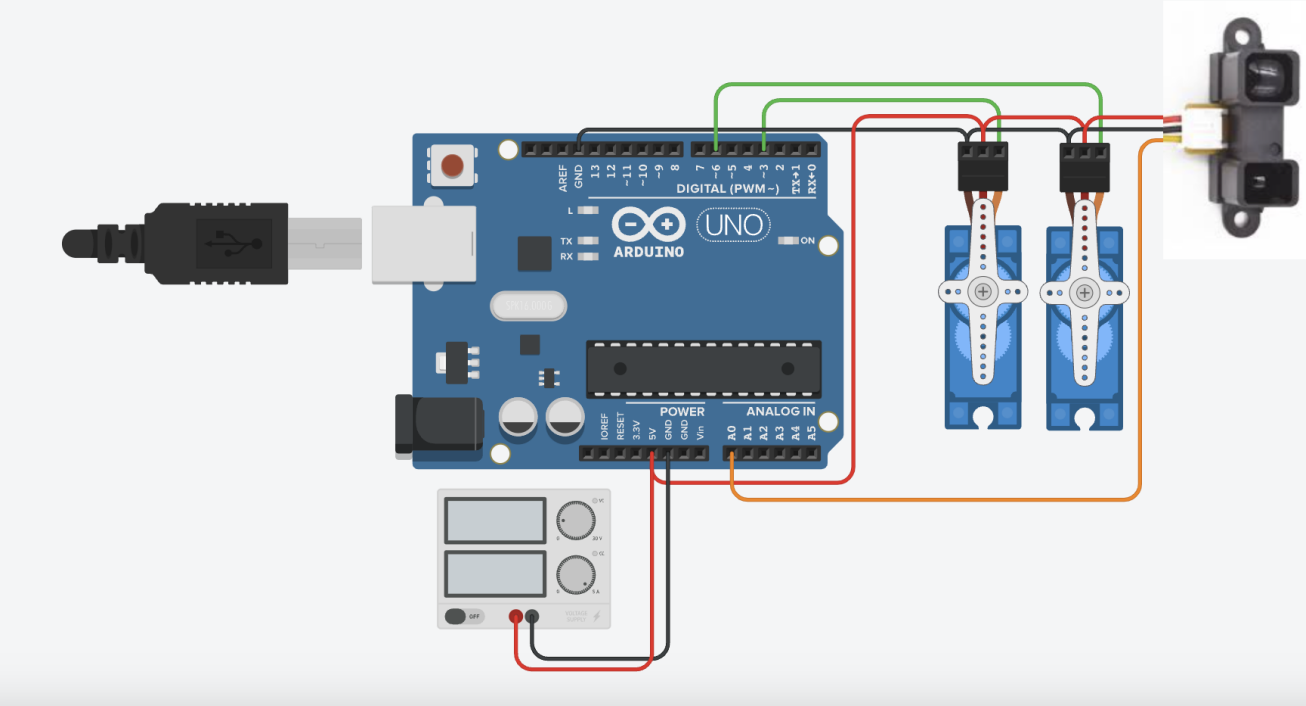

Figure 6: Circuit Diagram

Code

Arduino

```
#include <Servo.h>

Servo tilt_servo, pan_servo;  // creates servo object for tilt and pan servo

uint8_t PAN_PIN = 3; // Variable representing the digital pin connected to the pan servo
uint8_t TILT_PIN = 6; // Variable representing the digital pin connected to the tilt servo
int IR_SENSOR_PIN = 0; // variable representing the analog pin that the IR sensor is connected to

int pan = 0;    // variable representing the starting pan angle
int tilt = 75;   // variable representing the starting tilt andlge

int pan_max = 50;
int pan_min = 0;
int pan_change = 1;

int tilt_max = 120;
int tilt_min = 75;

uint16_t x, y, z; // variables used to calculating/averaging IR sensor output
double res = 0; // variable used to store average IR sensor output value

bool running = true; // bool variable for stopping the movement and data collection after a scan

double get_IR_data() {
// Function to get data fro the IR sensor. Three data points
// are taken at once and that the smallest is taken to account for
// the spike that happens in the sensors output at regular intervals
// 300 of these measurements happen and are then averaged. This average value is then fed into the transfer function we calculated for the sensor.
  for(int i = 0; i < 300; i++){
    x = analogRead(IR_SENSOR_PIN);            .
    y = analogRead(IR_SENSOR_PIN);
    z = analogRead(IR_SENSOR_PIN);

    res += (double)min(min(x, y), z);
```

```
    delayMicroseconds(48);
  }
  res /= 300;

  return (286 + (-43 * log((double)res)));
  // transfer function is distance = 286 + -43 * ln(sensor output)
  }

void send_IR_data(int pan, int tilt, double IR_data) {
// Function to send the IR sensor output over a serial connection.
   Converts the 8 bit values to angle values.
    Serial.print(pan); Serial.print(",");
    Serial.print(tilt); Serial.print(",");
    Serial.println(IR_data);
}

void setup() {
  Serial.begin(115200);
  tilt_servo.attach(TILT_PIN);  // Attaches the servo on the pin
     TILT_PIN to the servo object
  pan_servo.attach(PAN_PIN);    // Attaches the servo on pin PAN_PIN
      to the servo object
  pan_servo.write(pan);         // Moves the pan servo to the
     stating position
  delay(1000);
  tilt_servo.write(tilt);       // Moves the tilt servo to the
     starting position
  delay(1000);
}

void loop() {


  while (running) {
  // While the scanner has not finished panning all angles, the
     scanner changes pan angle
    for (pan = pan_min; pan <= pan_max; pan += pan_change) {
    // then takes distance measurements over the course of the min
       and mac tilt angle defined above.
    // The measured data is sent over serial connection after each
       tilt interval.
      for (tilt = tilt_min; tilt <= tilt_max; tilt += 1) {
      // Iterate through tilt angles and take data
        tilt_servo.write(tilt);
        delay(10);
        send_IR_data(pan, tilt, get_IR_data());
```

```
      }
      pan_servo.write(pan);                                    // Change pan angle
      delay(10);
    }
    running = false;                                           // Stop panning/tilting and taking data once all angles within range have been iterated through
  }
  Serial.println("end");
}
```

Python

```
import serial
import matplotlib.pyplot as plt
import math
import csv
from mpl_toolkits.mplot3d import Axes3D

arduinoComPort = "/dev/cu.usbmodem12101"

# Set the baud rate
baudRate = 115200

# open the serial port
serialPort = serial.Serial(arduinoComPort, baudRate, timeout=1)

# Initialize spherical data lists
pan_angles = []
tilt_angles = []
distances = []

# main loop to read data from the Arduino, then display it
while True:
    lineOfData = serialPort.readline().decode()

    if "end\r\n" in lineOfData:
        break

    elif len(lineOfData) > 0:
        pan_angle, tilt_angle, distance = map(float, lineOfData.split(','))

        print(f"Pan angle: {pan_angle}, "
              f"Tilt angle: {tilt_angle}, "
              f"Distance: {distance}")

        # Append data to lists
        pan_angles.append(pan_angle)
```

```
        tilt_angles.append(tilt_angle)
        distances.append(distance)

# Lists for plotting cartesian coordinates
x = []
y = []
z = []

for i in range(len(distances)):
    x.append(distances[i] * math.sin(tilt_angles[i] * math.pi / 180) *
             math.sin(pan_angles[i]) * math.pi / 180)
    y.append(distances[i] * math.sin(tilt_angles[i] * math.pi / 180) *
             math.cos(pan_angles[i] * math.pi / 180))
    z.append(distances[i] * math.cos(tilt_angles[i] * math.pi / 180))

file_name = 'data.csv'

# Create a list of lists where each sublist contains the values for each
   column
data = list(zip(x, y, z))

# Write the data to a CSV file with three columns
with open(file_name, mode='w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(['x', 'y', 'z'])  # Write header row
    writer.writerows(data)

# Create a 3D plot
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.clear()
ax.scatter(x, y, z, c=y, cmap='viridis')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_xlim(-1.4, 1.4)
ax.set_ylim(0, 200)
ax.set_zlim(10, -25)
plt.show()
```

Reflection

Software: The software part was the major part of this project. The Arduino code had to be modified a lot after an initial draft to account for how we want to process the data obtained from the servo motors and the infrared sensor, and averaging over several readings to remove errors in data readings. Also, we struggled a little bit to convert the spherical data to Cartesian data in Python and using MATLAB would have been better for that as it has an in-built function to do that.

Electrical: The electrical system was simple enough (just two servo motors and the distance sensor plugged directly into the Arduino Uno) that we would not change anything that we designed. Unfortunately, there was a power issue. The servo motors seemed to be drawing slightly more than the 500mA limit of the Arduino, and as time passed, our servo motors started to seize. To remedy this issue, we connected an external power supply.

Mechanical: We emphasized material efficiency in our mechanical design and chose to use materials readily available to us. While this made our final mechanical design not look completely clean, it challenged us to use scrap supplies and minimize waste. After this project, we will be able to disassemble it's parts and they can be used for other projects in the future. We had an issue with our initial servos not functioning as they should and had to replace them for our setup to function.

Teaming: We functioned as an efficient team and divided responsibilities based on what areas we wanted to improve upon and already had experience with among mechanical, electrical, and software. Ian worked on the mechanical and electrical side. Sparsh worked on mostly software and a little bit of mechanical. Mark worked on electrical and Arduino software.