Mini-Project 2 Lab Report

Section 4, Patrick, Moises and Melody

Narrative Description

With a well-rounded team of an ECE, an E with C, and a ME, we were able to get directly to work from the first day. Each of us took on an individual responsibility for the project whether it be setting up the servo/IR system, the mechanical design, or the program that would control the system. Still, we all kept in mind the overall goal which was creating a 3D scanner that would utilize a pan/tilt system to recognize a letter. Through our constant group communication we were able to work through the various issues we ran into. Examples include getting over the learning curve of Arduino-Matlab serial communication, not being able to get consistent readings from the IR sensor when calibrating distance, and a struggling tilt mechanism. Nevertheless, we were able to comfortably and accurately achieve the project goal and had a great teaming experience.

Circuit Diagram

Our 3D scanner utilizes an Arduino, one Sharp IR sensor to get distance readings, and two servo motors to control the pan-tilt movement (with connections shown in Fig 1).

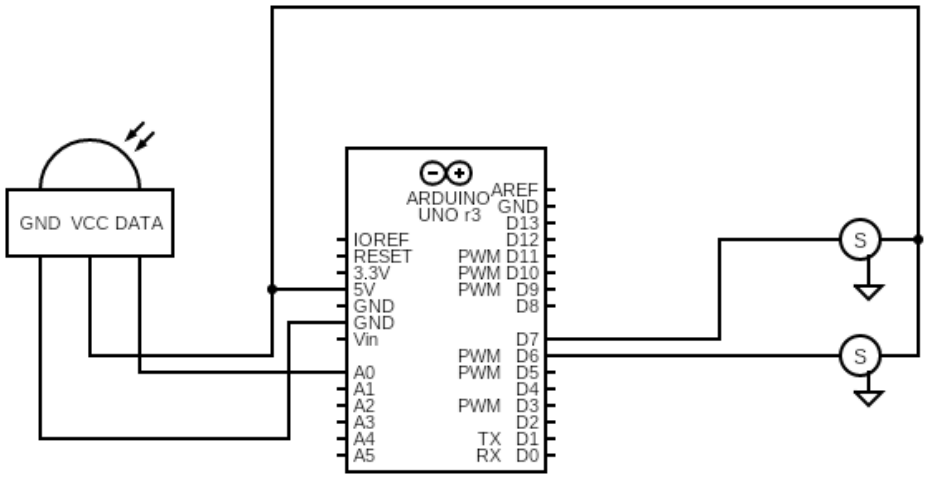

Fig 1. Circuit schematic of our 3D scanner

Pan/Tilt Mechanical Design

When thinking of the necessary components for the pan and tilt system, we realized that the possibilities were abundant. Knowing that the ultimate goal is to create a system that will pan in the horizontal axis and tilt in the vertical axis, we used a basic design as seen in #22527 on Thingiverse. The benefits to this simple design are numerous. First, its small form factor allowed for quick 3D prints, allowing us to iterate (which we

did). Secondly, it was created with almost the same dimensions for the servos as our components. Finally, the design was well suited for construction with hard assembly components such as screws as compared to glue, which in the prototyping phase makes it much easier to make changes.

In the first iteration, we found that the design was very constraining in the vertical/tilt axis. Moreover, we had underconstrained the servo, not fully screwing it down to the base plate and to the top component. This caused the top servo to move when powered on, meaning we could not get a consistent tilt. To fix this, we cut one of the vertical pieces from the top base plates because it was causing lots of rubbing (probably because of the print quality) which was creating an inconsistent tilt. This meant that one side was fully constrained using the base plate and the top piece and the other side was simply the top piece connected to the servo. This worked because there was no longer a frictional stress between the top and the base, allowing the tilt to freely move.

Sensor Testing and Calibration

For our infrared sensor, we made use of MATLAB, using a self driven program which takes in the actual distance and measured voltage received from the sensor from a variety of points. These points are taken in samples in order to minimize error. The resultant points are then graphed and a line of best fit is automatically generated. In this case, it resulted in an exponential graph with the distance as the dependent variable, as seen below in Figure 4. As a result we could work with real distances.

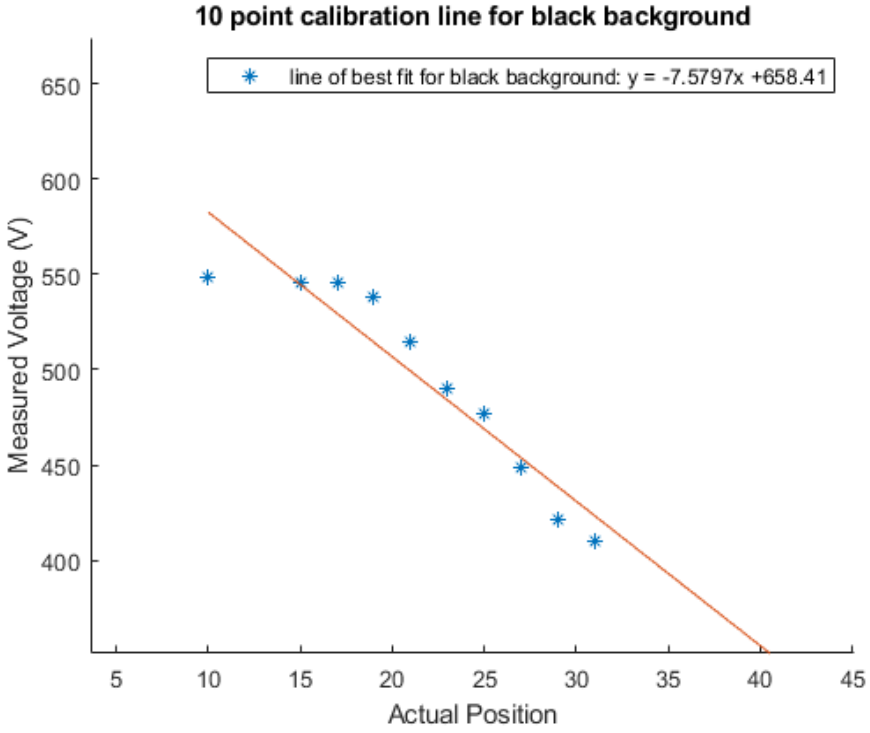

Fig 2. Calibration plot showing analog voltage reading vs. actual distance

Distance Error

As for our actual distances reported from real life measurements, they weren't too far off from the predicted values our sensor returned. The error was generally minimal, as shown from our error plot shown in Figure 2B below. Our ideal line-fit seemed to generally keep in trend with our actual data values.

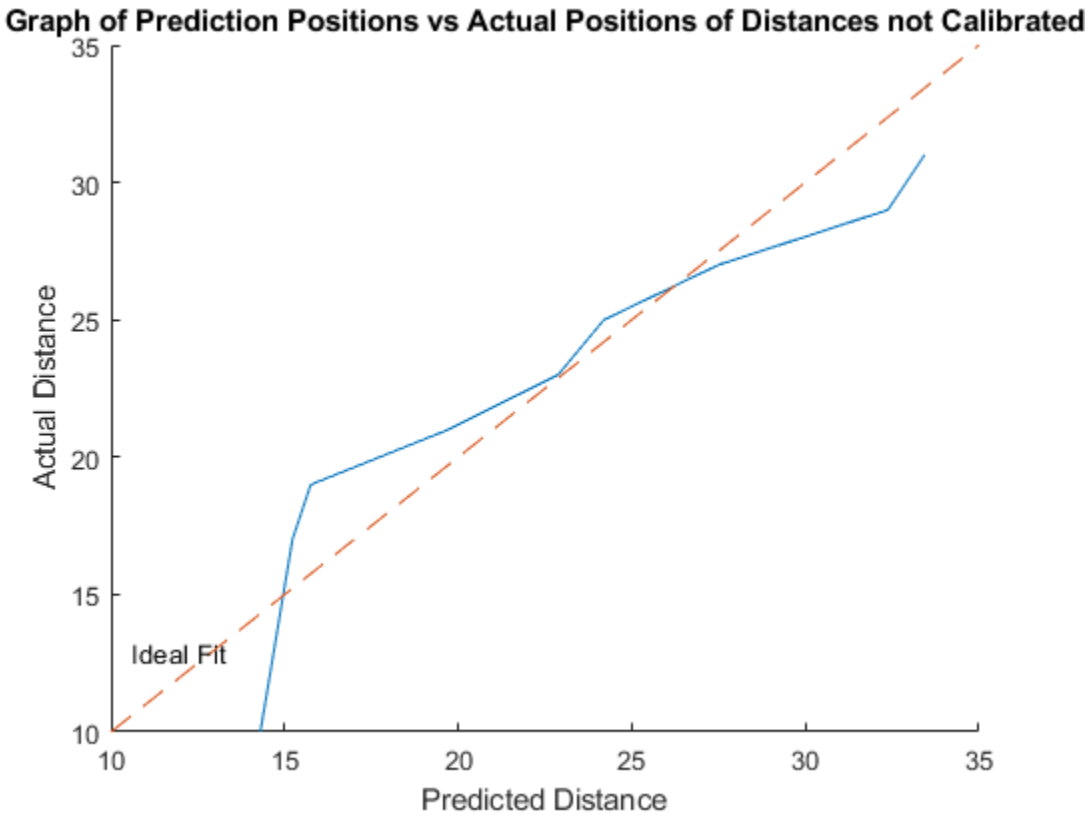

Figure 2B: Error Plot of Actual vs Predicted Distance

[Source Code]

To program our 3D scanner, we have an Arduino program that controls the pan and tilt servos to sweep the IR sensor across the letter and receive distance readings. Through a serial port, this program passes the information to a Matlab program that handles polar-to-cartesian conversion and graphing of the 2D, top-down representation of the scanned letter.

Firstly, our Arduino program sets up the pin connections for our hardware and declares variables such as the max angles of our pan and tilt servos.

```
#include <Servo.h>

const int irPin =  A0;
const int maxPanAngle = 50;
const int maxTiltAngle = 50;
Servo panServo;
Servo tiltServo;

// setup function to initialize hardware and software
void setup()
{
  // start the serial port
  long baudRate = 9600;
  Serial.begin(baudRate);

  pinMode(irPin, INPUT);
  tiltServo.attach(6);
  panServo.attach(7);
}
```

Fig 3. Arduino code block: declaration of global variables and setup function

For a single cross-sectional sweep of the letter, our main loop first commands the tilt servo to be at a constant angle while the pan servo sweeps through different angles incrementally. At each interval angle, a voltage reading is taken from the IR sensor. This voltage reading (stored in the variable ir_voltage) is then passed through our calibration equation to convert voltage into distance. After sending distance and pan angle data to the serial port for our Matlab program to read from, we have a delay that prevents the serial connection from being overrun.

For the full sweep (Fig 4), our loop function is slightly different. Instead of sweeping through different pan servo angles and a constant tilt servo angle, we sweep through intervals of pan and tilt angles between 0 and the desired max angle. At each interval angle, we command the pan and tilt servos to go that angle position and then take in a voltage reading from the IR sensor. Again, this voltage reading is converted into distance and sent to the serial port, along with tilt angle and pan angle data. Looping through these intervals effectively sweeps the sensor across the letter.

```
21  void loop()
22  {
23    float panInterval, tiltInterval;
24
25    // loop: read voltage at each angle, then send it from the Arduino to the Matlab program
26    for (tiltInterval = 10; tiltInterval <= maxTiltAngle; tiltInterval=tiltInterval + 1.5) {
27      for (panInterval = 10; panInterval <= maxPanAngle; panInterval=panInterval + 1.5) {
28
29        tiltServo.write(tiltInterval); //write tilt limit at 50
30        panServo.write(panInterval);
31
32        // get voltage reading from ir sensor
33        float ir_voltage = analogRead(irPin);
34
35      float distance = (ir_voltage - 658.41)/-7.5797; // convert voltage to distance with calibration
36
37        // transmit one line of text to Matlab with 3 numeric values
38        Serial.print(distance);     Serial.print(",");
39        Serial.print(panInterval);     Serial.print(",");
40        Serial.println(tiltInterval + 20);
41
42        // delay after sending data so the serial connection is not over run
43        delay(500);
44      }
45    }
46  }
47
```

Fig 4. Arduino code block: void loop function of 2 servo sweep

Next, we have our Matlab program that reads information from the serial port and processes the data to compute a visual representation of the scanned letter. Firstly, our program sets the serial port to connect to and the baud rate. We close the serial port if it was previously open, then open it so that we can read data communicated from our Arduino program in our main loop.

```
clear all


arduinoComPort = 'COM5';  % set serial port
baudRate = 9600;   % set the baud rate

% open the serial port, close it first in case it was previously open
fclose(instrfind({'Port','Status'},{arduinoComPort,'open'}));
serialPort = serial(arduinoComPort, 'BAUD', baudRate);
fopen(serialPort);
fprintf(serialPort, '\n');

% initialize a timeout in case MATLAB cannot connect to the arduino
timeout = 0;

x_values = [];
y_values = [];
z_values = [];
loop_index = 1;
```

Fig 5. Matlab code block: serial port setup

For the single-line sweep, our Matlab program reads data sent from the Arduino in a loop. Distance and pan angle data are added to their respective arrays to later be plotted on a 2D graph. For the full sweep, we do the following in the loop: receive and store the data into individual variables for distance, pan angle and tilt angle, convert those polar coordinates into Cartesian coordinates (x, y, z), and load them into their respective arrays (Fig 6).

```
% main loop to read data from the Arduino
while timeout < 5
    % check if data was received
    while serialPort.BytesAvailable > 0
        timeout = 0;    % reset timeout
        % data received from Arduino, convert it into array of integers
        values = eval(strcat('[',fscanf(serialPort),']'));

        % store the integers in three variables
        distance = values(1);
        pan_angle = values(2);
        tilt_angle = values(3);

        % polar to cartesian conversion
        x_values(loop_index) = distance*sind(tilt_angle)*cosd(pan_angle);
        y_values(loop_index) = distance*sind(tilt_angle)*sind(pan_angle);
        z_values(loop_index) = distance*cosd(tilt_angle);
        loop_index = loop_index + 1;
    end
    pause(0.5);
    timeout = timeout + 1;
end

plot3(x_values,y_values,z_values, '.')
title("3D Scanner Result");
xlabel("x");
ylabel("y");
zlabel("z");
```

Fig 6. Matlab code block: main loop for 2 servo sweep plot

To convert between spherical and Cartesian coordinates, we referenced Oregon State's online guide [Coordinate Systems in Two and Three Dimensions]. The equations we used are:

$$x = p^*\sin\phi^*\cos\theta$$

$$y = p^*\sin\phi^*\sin\theta$$

$$z = p^*\cos\phi$$

where p is distance reading from the IR sensor, θ is the pan angle, and ϕ is the tilt angle.

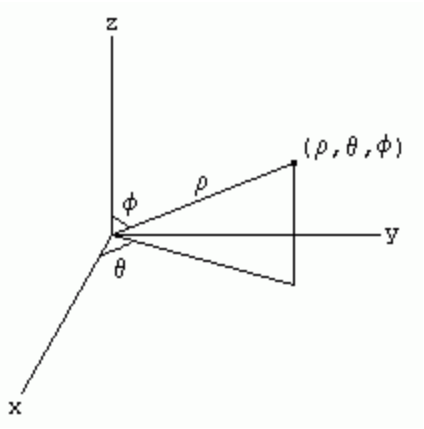

Fig 7. Relationship between polar and Cartesian coordinates

Finally, we plot the data as x, y, z points on a 3D graph.

Results

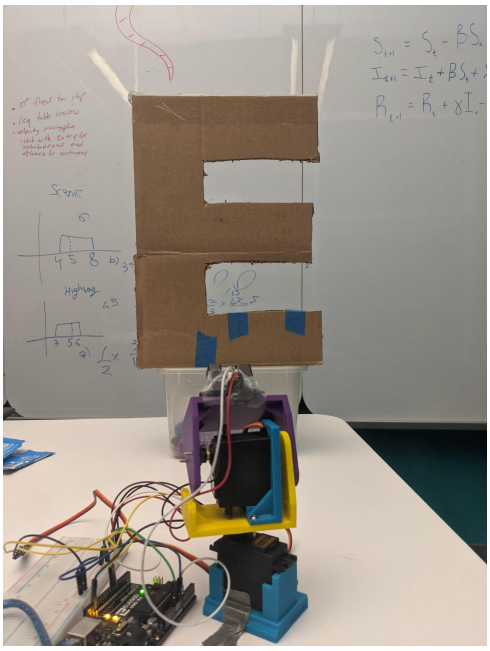

Fig 8. Setup for the 1 servo and 2 servo scan of our letter

Before doing an entire scan of our letter, we first scanned a horizontal line across it and plotted it to show the 2D, top-down representation of the letter along that line.

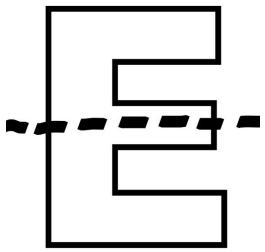

Fig 9. Horizontal sweep

In the single-line scan (shown in Fig.), as the pan angle increases, we can see the distance reading go from high to low to high. This shows the measured distance to the wall, then the measured distance to the letter face as the mechanism pans to face it directly, then back to the wall as it passes the other side of the letter.

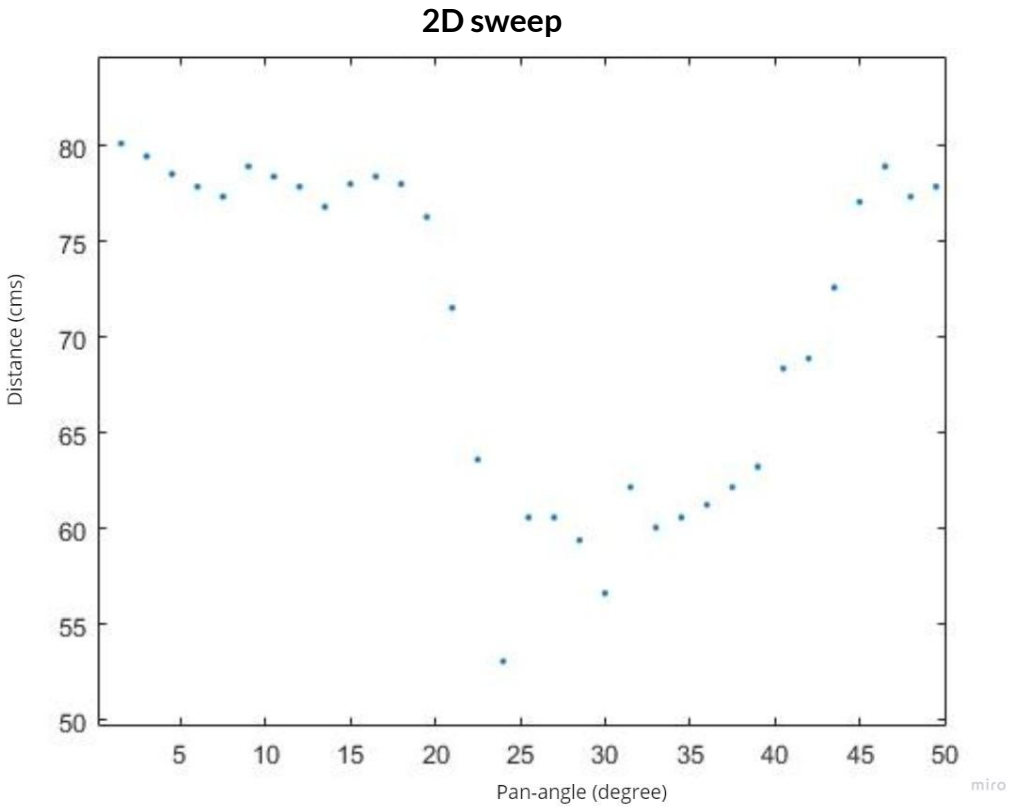

Fig 10. Graph of a horizontal sweep of the letter ‘E’

With this graph looking accurate to what we programmed the mechanism to scan, we moved on to program a full sweep of the entire letter. After tuning the pan and tilt angle ranges and testing out varying densities of sensor data, we got our pan/tilt mechanism to successfully sweep the face of a letter and output a graph to visualize it!

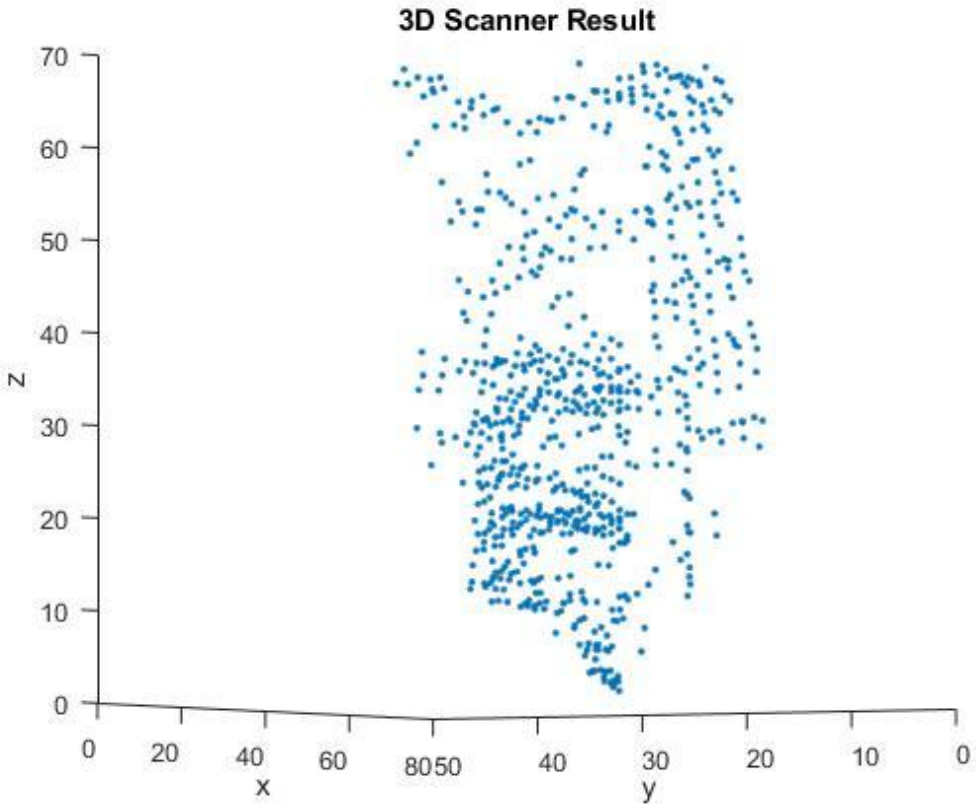

Fig 11 . 3D scan of the letter 'E'

Reflections

Moises: This project was interesting because it was the first time that I had to use my ME background in a group project. To my benefit, I felt very comfortable and confident having my team rely on me for those components. I believe that I had it under control for the entirety of the project. Nevertheless, there are definitely highs and lows to my work. I feel that highs include only having to make minimal changes to the design used and having everything work. Lows include having to drill holes because I didn't make them on my design and not having a place to put the IR sensor. I feel as though overall, I accomplished my job and did it well.

Melody: I find working with pan-tilt mechanisms really fun! I've written code for pan-tilt control before, but this is the first time I've gotten to work with a team on building it from scratch. I also appreciate getting to learn about serial communication between Arduino and Matlab.

Patrick: I find setting up the circuit very interesting, seeing how all the parts work together. Also programming two servos hand in hand is very interesting. What I found difficult was I guess understanding the best mechanics for assembling a pan-tilt

mechanism, but once that was done, it was very exciting. It was also tricky working on serial communication between Arduino & MATLAB, as I'm used to interfacing directly through MATLAB. I was very happy when we saw our letter visualize itself on our graph, and overall it was a fun project.