Lab Report 2

Anya Jensen and Brandon Zhang

October 6, 2018

Abstract

Create a sensor that can detect 3D objects and plot them in space.

1 Introduction

In this report, we will demonstrate how we created a sensor that can detect points in space, and programmed it with an Arduino to plot these points. Throughout this lab, you will find the basic circuit setup, explanations of our thought processes, and findings.

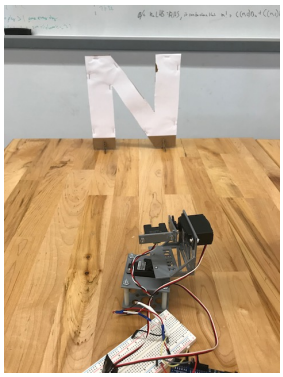
2 Set Up

2.1 Materials

- Arduino Uno
- 2 Servos
- 1 IR distance sensor

2.2 Circuit Setup

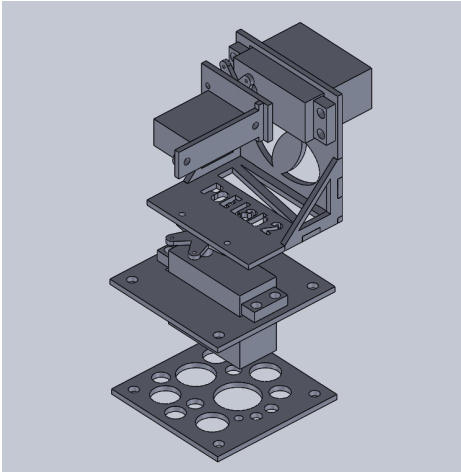
We used a solderless breadboard with connected to 5V power and ground on the Arduino Uno. We connected the servos and IR distance sensor to different pins on the Arduino, so we can send and receive data.



3 Process and Results

3.1 Overall Process

We began by figuring out what exactly we wanted our sensor to do. We needed to collect data by using a sensor mounted on 2 servos. To begin, one of us worked on CADing the mount to hold all of the parts in, while the other began experimenting with the servos to get a preferred motion. The model we made to mount everything to can be seen below.



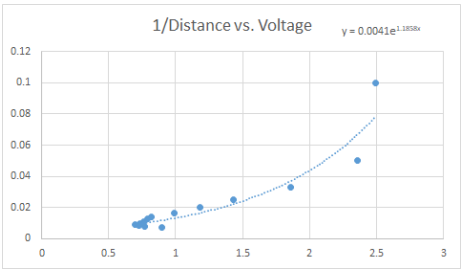
A CAD model of our sensor mount

We knew we needed to plot the data coming from the servo, which the Arduino IDE does not have capabilities for, so we decided to go with MATLAB. This is because MATLAB has Arduino capabilities, and we are already comfortable plotting in it.

We 3D printed and then built out sensor, then wrote a MATLAB script that would run the servos and collect data, then plot that data in 3D.

3.2 Calibration and Error

We started by calibrating the sensor, so we can get accurate distance readings from the sensor. We did this by collecting voltage given by the sensor and comparing it to known distances and finding the best line fit of this plot. Unfortunately, the sensor started giving strange readings around 100 cm. You can see that the line appears to double back on itself right at the beginning. You can see the data compared to the best line fit plot below:

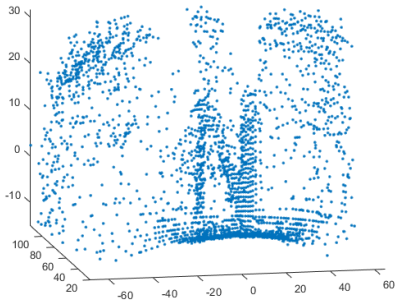


Our data points and the line of best fit. Note the line doubling back on the left of the plot

As you can see in the plot, the line of best fit is $y = 0.0041e^{1.858x}$.

3.3 Results

The following is the plot we got of data collected from the letter N created using cardboard and covered in white paper:



You can clearly see the N, but there is a fair amount of noise. We attempted to implement two filters. The first one was a real-time filter that checked if the measured distance of each point was off from the points on either side of it by a large amount. This would determine if the point was noise, or if it was simply the edge of the letter. If the point was noise, the distance would be changed to be the average of the surrounding points. However, this filter did not end up removing that much noise, so we removed it from the code. You can see the code below:

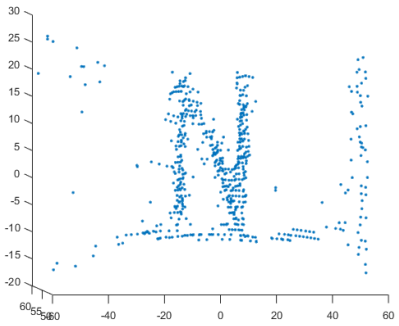
```
if (horzangle > 0.36 && horzangle < 0.59)
     if(abs(r(counter-1) - r(counter - 2)) > 10 && abs(r(counter-1) - r(counter)) > 10)
```

```
        r(counter - 1) = (r(counter - 2) + r(counter))/2;
    end
end
```

We also used a second "filter" that would only copy the points at a certain distance from the sensor, and then plot only those copied points. The hope was to tune the threshold so that we would only plot the N. The filter we implemented can be seen in this code:

```
for n = 1:length(r)
    if (50 < y(n) && y(n) < 60)
        xnew = [xnew, x(n)];
        ynew = [ynew, y(n)];
        znew = [znew, z(n)];
    end
end
```

You can see the new plot below:



While it worked, it was not optimal, because the variance in measured points on the N went up to around 5-6 cm, which made even the N itself too noisy to be plotted properly.

3.4 Obstacles

We had a few obstacles along the way for our project. One major one was that neither of us had experience coding Arduino in MATLAB. As such, we had to learn new syntax just to get the servos moving and the IR sensor to read data.

We also had the issue of having a fair amount of extraneous data points on our plot, which we remedied by creating a filter and cut down the scan range

Additional issues we ran into were the structure shaking when the servos moved too fast and the original two filters we used were removing too many points.

3.5 Code

Below you can see the code used to run this lab. All code was written in MATLAB:

```
a = arduino('COM3', 'Uno');
```

```
%Initialize horizontal and vertical servos
horz = servo(a, 'D8');
vert = servo(a, 'D9');

%Port for sensor
sensor = 'A0';

r = [];
theta = [];
phi = [];
x = [];
y = [];
z = [];

writePosition(horz, 0.25);
writePosition(vert, 0.4);
pause(0.05);
counter = 1;

%Have the vertical servo move between 0.4 and 0.6 radians
 for vertangle = 0.4:1/180:0.6
     %Have the horizontal servo move between .25 and .75 radians
    for horzangle = 0.25:1/180:0.75
        writePosition(horz, horzangle);


        %The current data point scaled using our calibration function
        rtemp = 1/(0.0041*exp(1.1858 * v));

        r = [r, rtemp];

        %The current angles of the servos
        theta = [theta, horzangle * pi];
        phi = [phi, vertangle * pi];

    end

    %Return horizontal servo slowly to initial position
    for horzangle = 0.75:-1/180:0.25
        writePosition(horz, horzangle);
        pause(0.05)
    end
    writePosition(vert, vertangle);
 end

%Find x, y, and z in 3D space using servo angles and the detected distance
for n = 1:length(r)
     x = [x, r(n)*cos(theta(n))*sin(phi(n))];
     y = [y, r(n)*sin(theta(n))*sin(phi(n))];
     z = [z, r(n)*cos(phi(n))];
```

```
end


plot3(x, y, z, 'LineStyle', 'none', 'Marker',  '.');
daspect([1 1 1])

clear all;
```