Principles of Engineering

Mini-Project 2 - DIY 3D Scanner

Repository Link: `https://github.com/toluooshy/PIE-Mini-Project-2`

Section 1
Walter Villa & Tolulope Oshinowo

Due: October 1, 2021

Abstract

The following report aims to depict the mechanical, computational, and collaborative work done to build and program a rudimentary 3D-scanner using a pan/tilt mechanism. The purpose in completing this project was to introduce us to **sensors** and **actuators**.

- By using sensors, we were able to transduce a physical quantity in the world, such as force, pressure, etc), into a more easily, measurable, quantifiable metric like voltage. In this mini-project we used an infra-red sensor to transduce distance into voltage. This metric will be used to help determine the density of local matter when creating the 3D render.
- By using actuators, we were able to transform one form of energy, such as electric potential, into mechanical work. In our mini-project we used two hobby servo motors to operate the panning and tilting mechanism for the overall 3D scanner.

In this mini-project we were tasked with building a 3D-scanner that could effectively visualize three-dimensional data. Once we built the scanner, we were tasked with scanning a letter of well-known geometry, and then visualizing the output from the scanner on a software medium of our choice. For this task we chose to go with scanning the letter "S" as can be seen in **Figure 0.0**.

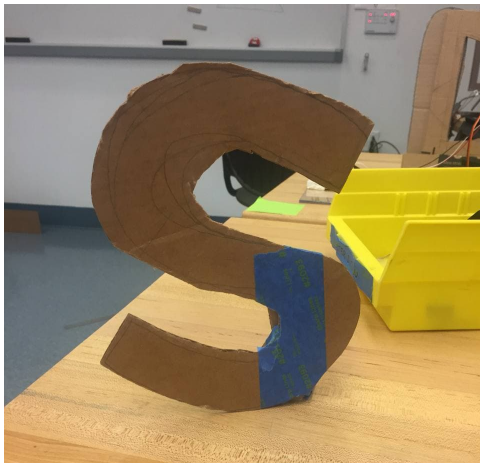


Figure 0.0 - Photo of the letter "S".

Materials

- Arduino UNO
- USB cable
- 1X Sharp GP2Y0A02YK0F IR distance sensor
- 1X 3-pin Molex connector with red, black, and white wire tails
- 2X Hobbyking HK15138 servo motors

Testing The Sensor

To test the sensor we first hooked it up to an Arduino. With the help of the distance sensor manual we were able to find out the recommended range for the sensor (20cm to 150cm), and its wiring configurations. This made the testing process much simpler rather that finding trying to find our way in the dark. Once the sensor was correctly connected we used the `AnalogInOutSerial` example script to see if the sensor readings made sense. First we hooked up the input voltage and ground wires from the sensor to the Arduino, and then plugged the signal/output voltage wire from the sensor into the Analog 0 (A0) port on the Arduino. With this done we verified and uploaded the `AnalogInOutSerial` script to the Arduino and opened up the Serial Monitor to see a flow of values coming in. This was a good indicator as it showed that the sensor was indeed working, and from experimenting with moving objects closer to and further from the receiver of the sensor we realized that there was a correlation between an increased voltage and the proximity of matter.

Calibrating The Sensor

With the confidence of knowing that our IR sensor worked we then moved onto calibrating it to get a more holistic understanding of the relationship between how close an object is and the derived voltage reading. We began by creating a stationary mount for the sensor and then marking several distances away from the sensor with tape ranging from 20cm to 150cm. We then used a notebook as a calibration object and listed the sensor's output voltage readings at each marked distance. This yielded the calibration table below in **Figure 1.0** that confirmed our prior hypotheses about the correlation between output/signal voltage and distance.

Calibration Table for Sharp GP2Y0A02YK0F IR Distance Sensor

Distance	[1023] - Signal	[255] - Output
20 cm	476	120
50 cm	240	59
75 cm	155	38
125 cm	90	22
150 cm	77	19

Figure 1.0 - Calibration data table.

Discovering The Calibration Function

Our calibration function was a result of finding the most suitable line of best fit for our signal output and voltage data reading from the serial monitor. We began by taping off known distances, using the Infrared Sensor's data sheet to give us upper and lower bounds in terms of how far or close the sensor could read values. Once we collected the serial output and the voltage reading along with it, we put our data into our data entry and visualization platform of choice at this stage, Google Sheets, and we calculated the line of best fit for the points collected. We tried various different regressions, and the one that fit best was a quadratic regression with the equation:

$$f(x) = 590e^{-0.0152(x)} \text{ for the Signal Output, where our } R^2 \text{ value was 0.959, and}$$

$$f(x) = 149e^{-0.0155(x)} \text{ for the voltage output, where the } R^2 \text{ value was 0.956.}$$

Seeing that the correlation coefficient was around 95%, we made the decision to use this as our calibration curve. The following graph in **Figure 2.0** shows our calibration plot with their respective functions.

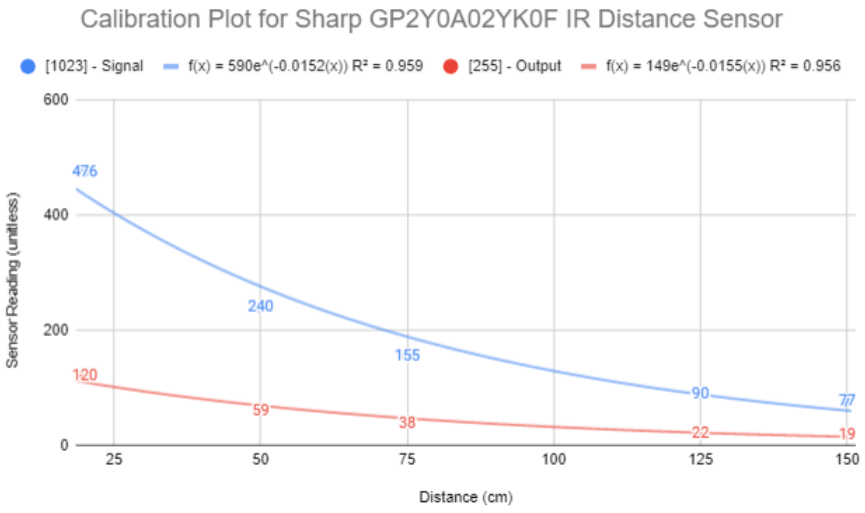


Figure 2.0 - Graph of calibration data.

No calibration function can truly be trusted without verification. To verify that our derived equations for the sensor were indeed valid abstractions of real distance, we carried over our theoretical plot into MATLAB and ran a plot of it along with some newly recorded empirical plots. Fortunately the empirical points we found were still within the neighborhood of the theoretical points, meaning that our equation was valid. Below in **Figure 2.1** is the MATLAB code we used to generate our error plot, and then the error plot itself in **Figure 2.2**.

```
distance = linspace(20,150,130)';

signal = 590*exp(-0.0152.*distance);

plot(distance,signal);
hold on;
%%% OBTAINED VALUES %%%
plot(30,362,'k.');
plot(60,227,'k.');
plot(100,135,'k.');
plot(140,84,'k.');
%%%%%%%%%%%%%%%%%%%%%%%%
title('Error Plot for Sharp GP2Y0A02YK0F IR Distance Sensor');
legend('Ideal Signal Regression (Theoretical Value)','Empirical Value')
xlabel('Distance (cm)');
ylabel('[1023] - Signal (unitless)');
hold off;
```

Figure 2.1 - MATLAB visualization code for error plot.

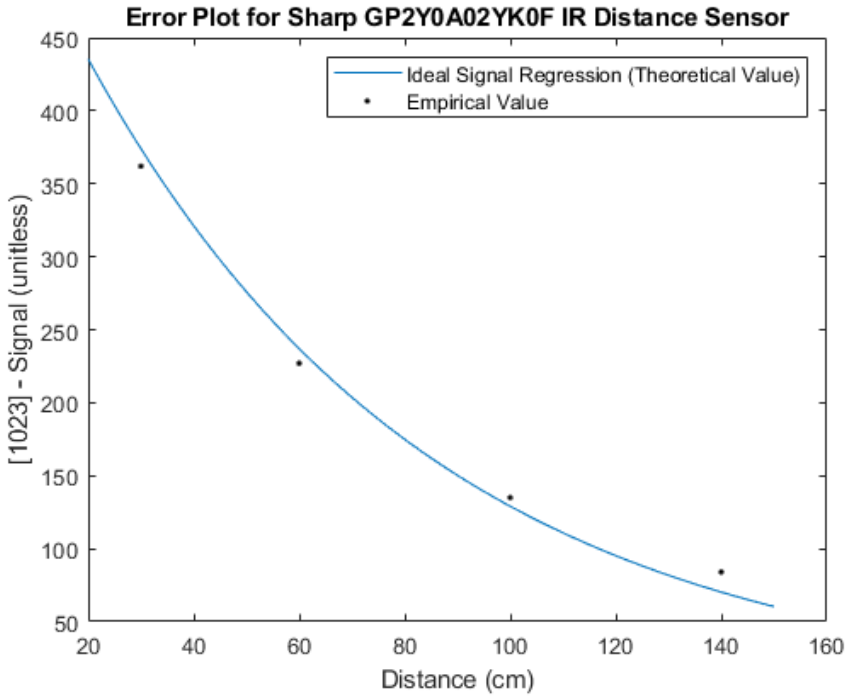


Figure 2.2 - Error plot based on distances not included in the original calibration routine.

Writing The Scanner Script

The process of coming up with the Arduino code script to move the IR sensor with the servo ended up being more intuitive that we had initially thought. Thanks to the `Sweep` example script and the aforementioned `AnalogInOutSerial` example script, we were able to adopt concepts from both and come up with our own code. Our code as shown in **Figure 3.0** would relay the sensor's voltage data to alongside the current angle position data from the servo, telling us at which angle(s) close solid matter is being detected.

```
void loop() {
  while(pos1 <= 125) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degrees
    servo1.write(pos1);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
    while(pos2 <= 115) {
      servo2.write(pos2);
      readsensor();
      delay(15);
      pos2 += 1;
    }
    while (pos2 >= 55) {
      servo2.write(pos2);
      readsensor();
      delay(15);
      pos2 -= 1;
    }
    pos1 += 5;
  }
```

(a) Snippet from code handling servo rotational logic.

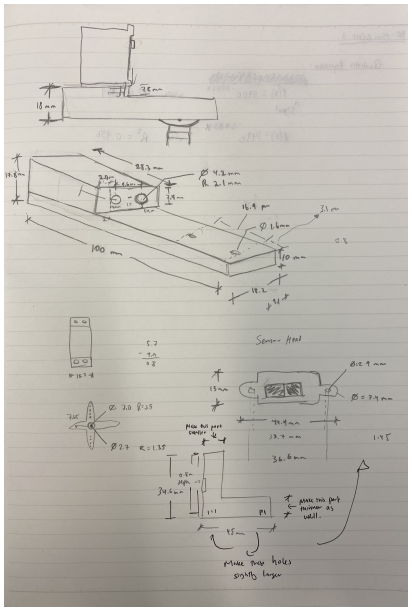
```
void readsensor() {
  // read the analog in value:
  sensorValue = analogRead(analogInPin);
  // map it to the range of the analog out:
  outputValue = map(sensorValue, 0, 1023, 0, 255);

  Serial.print(pos1); Serial.print(",");
  Serial.print(pos2); Serial.print(",");
  Serial.print(sensorValue); Serial.print(",");
  Serial.print(outputValue); Serial.print(",");
  Serial.println("");

  // wait 2 milliseconds before the next loop for the analog-to-digital
  // converter to settle after the last reading:
  delay(2);
}
```

(b) Snippet from code handling sensor data flow. updates.

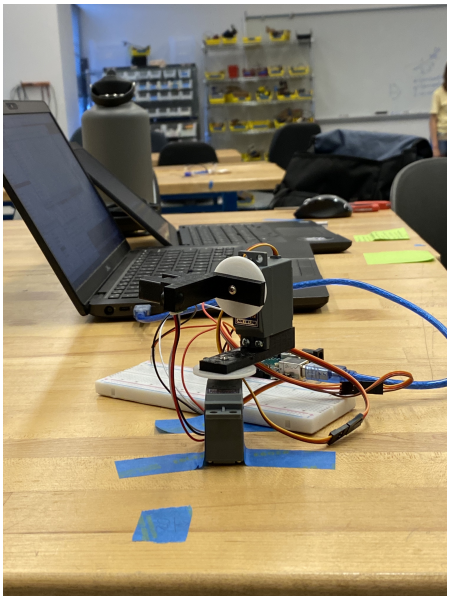
Figure 3.0 - The Arduino code behind the scanner.

Mechanical Design

When it came to the mechanical design of the 3D scanner, our objective was to create a minimalist, practical scanner. We began to sketch ideas on paper to see how both servos would be able to work in conjunction with the Infrared Sensors. We concluded that we needed a flat support piece to be able to rotate the overall direction that the Infrared Red sensor in the horizontal, and included in this flat piece would be a support for the other servo motor that would help the IR sensor rotate in the vertical direction. To mount the sensor onto the vertical-rotating servo, we created an L-shaped attachment that would be able to attach to a the Infrared Sensor's screw mounts. Our final design came out as the following in **Figure 4.0**.



(a) Our design sketches for our 3D pan tilt mechanism.

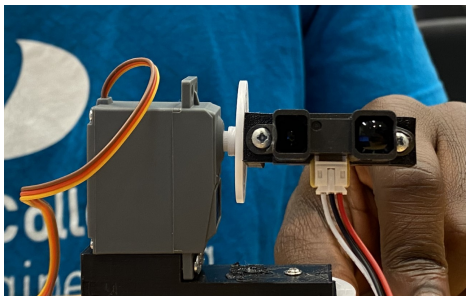


(b) Final working design for our 3D scanner.

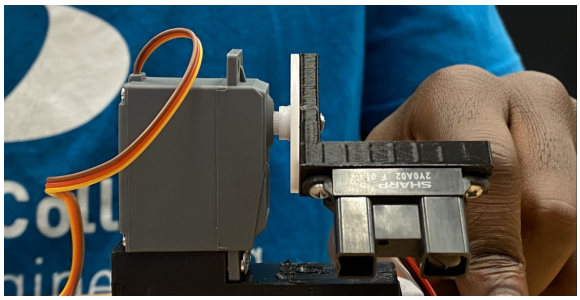
Figure 4.0 - From designs to the finished product.

Setup With One Servo

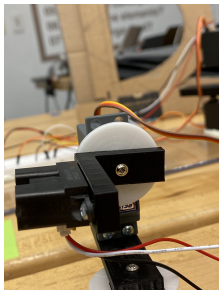
Before scanning the entire letter across two degrees of freedom, we first had to get a working model across one degree of freedom. This meant determining how to mount the IR sensor on one servo and sweeping along one axis. We chose the vertical axis for this stage and went with a setup that is shown in **Figure 5.0**.



(a) Servo mounted scanner in level position.



(b) Servo mounted scanner in rotated position.



(c) Profile View

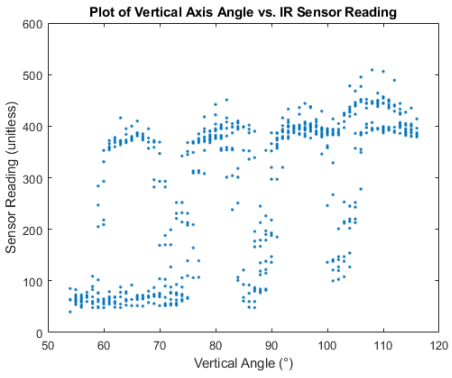
Figure 5.0 - Images of the one servo setup.

With an operational servo mechanism guided by the Arduino code, we now had a "tilt" scanner. We decided to test out it's viability with our letter "S" by scanning straight down the middle of our cutout letter. If our workflow was correct we would get an output of points that would oscillate between highs and lows due to how the letter "S" looks (top of the letter, empty space, middle of the letter, empty space, bottom of the letter). Below in **Figure 5.1** is an excerpt from the MATLAB code that we wrote which allowed us to visualize the data.

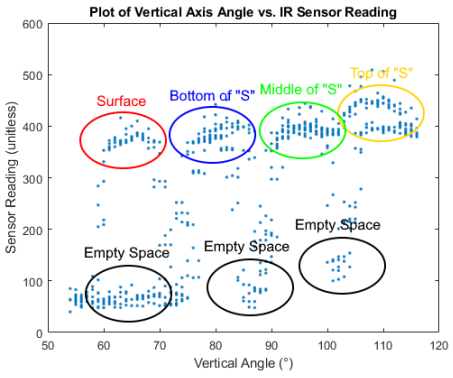
```
tiltScan = csvread('tiltdata.csv');

angle = tiltScan(:,1);
sensor1 = tiltScan(:,2);

plot(angle,sensor1,'.');
title('Plot of Vertical Axis Angle vs. IR Sensor Reading');
xlabel('Vertical Angle (°)');
ylabel('Sensor Reading (unitless)');
```

Figure 5.1 - MATLAB visualization code for one servo.

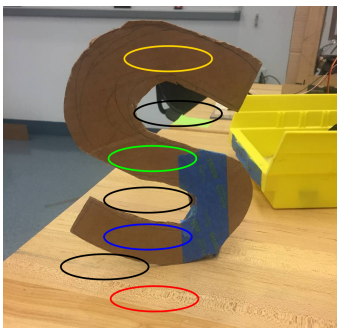
Upon running the MATLAB code we ended up receiving positive results. In **Figure 5.2**, we show the results of our setup with one servo collecting data in the vertical direction, straight down the middle of the cardboard cutout letter.



(a) Original plot.



(b) Annotated plot.

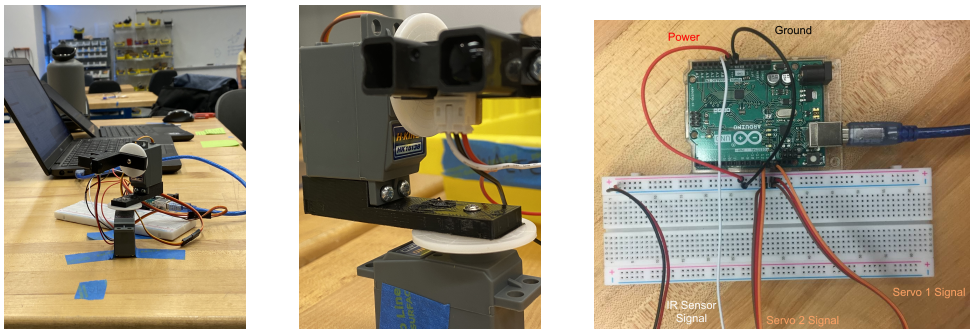


(c) Annotations mapped to "S".

Figure 5.2 - Results from the one servo scan.

Setup With 2 Servos

With our setup effectively working along one degree of freedom, we decided to add in the second degree with a second servo sweeping along the horizontal axis, positioned under the first servo. With this setup we would essentially have our pan-tilt scanner ready for use, and in **Figure 6.0** one can see how we ended up building out this stage of the mini-project.



(a) 3D Scanner (b) 3D Scanner Closeup (c) 3D Scanner Wiring

Figure 6.0 - Images of the two servo setup.

Additionally, below in **Figure 6.1** is an abstracted circuit diagram for our pan-tilt 3D scanner.

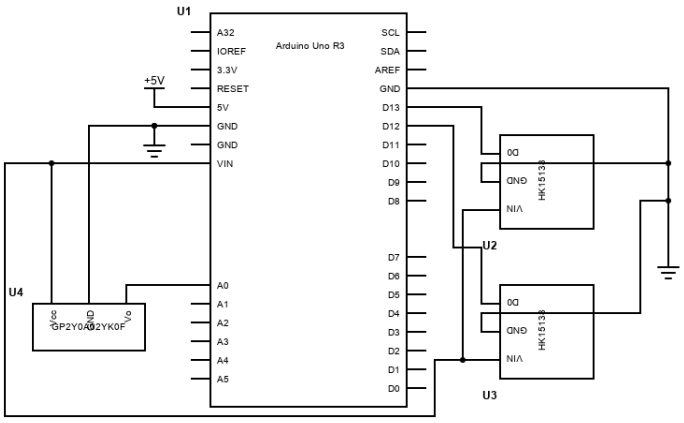


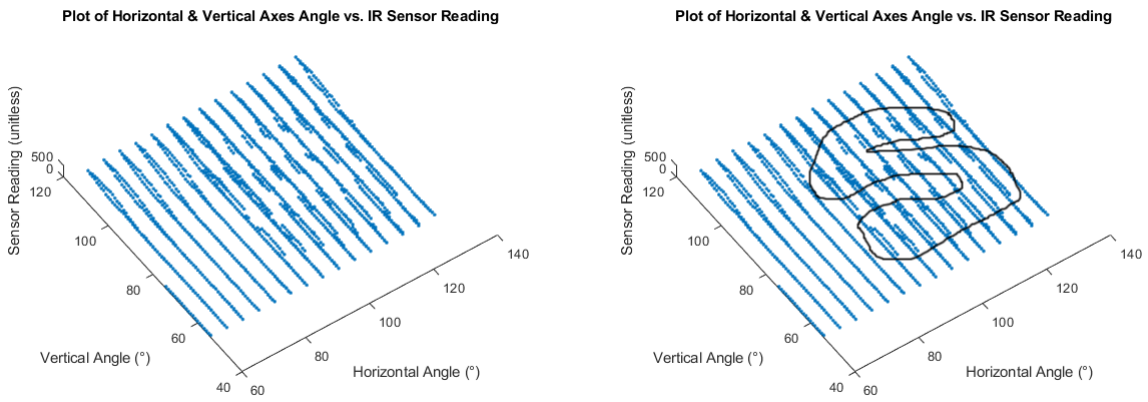
Figure 6.1 - 3D scanner schematic.

All in all we were able to scan the area surrounding the "S" and used the following MATLAB code to generate the visualizations for the 3D scan as shown in **Figure 6.2**.

```
pantiltScan = csvread('pantiltdata.csv');

angle1 = pantiltScan(:,1);
angle2 = pantiltScan(:,2);
sensor2 = pantiltScan(:,3);

plot3(angle1,angle2,sensor2,'.');
title('Plot of Horizontal & Vertical Axes Angle vs. IR Sensor Reading');
xlabel('Horizontal Angle (°)');
ylabel('Vertical Angle (°)');
zlabel('Sensor Reading (unitless)');
view([-35 87])
```

Figure 6.2 - MATLAB visualization code for two servos.

This code after being run ended up displaying the following plot in **Figure 6.3**.



(a) Original plot.

(b) Annotated plot.

Figure 6.3 - 3D scan of the cardboard cutout letter "S".

Conclusion

In conclusion, this mini-project served as a great opportunity to showcase our understanding of both mechanical and electrical concepts. From the CAD-ing and mount-building development processes to the math and logical reasoning that went into the calibration, sweeping, and visualization functions, we were both excited to figure out how to approach each challenge.

Walter: In this project, I took the lead on the Mechanical Design of this scanner. While it may not look like much, this scanner's meant a lot to me because I designed and printed it from scratch. I haven't touched Fusion360 since Design Nature, and it made me really content to be able to relearn and practice designing parts for this project. I have lost the fear of CAD-ing, and I truly want to do more of it in future projects. In addition to this, I was able to witness and learn how the data was collected and parsed into the visualizations that we were able to create.

Tolu: Thanks to this mini-project I was able to put my abilities of project planning to the test. I can remember at the beginning of this endeavor I was not sure about how to proceed, but as I began story boarding a workflow and breaking things into digestible milestones what had initially seemed like a daunting task became something I was eager to undertake. I additionally was able to sharpen my Arduino skills, and with the help of the teaching team, create something that I was proud of.

References

- Sharp GP2Y0A02YK0F IR Datasheet
  `https://canvas.olin.edu/courses/298/files/38407?wrap=1`
- Hobbyking HK15138 Servo Motor Info Page
  `https://hobbyking.com/en_us/hobbykingtm-hk15138-standard-analog-servo-4-3kg-0-17sec-38g.html`