# MAE 311 Term Project


## Group 1: Flex Sensor

Austin Click

Darnisha Crane

Walter Dietzler

Jacqueline Pepper

Hunter Phillips

November 28, 2017

# 1. Approval of Submission

I, Austin Click, approve this project report for submission. Information contained within is, to the best of my knowledge, complete and true, and in accordance with the process used throughout the completion of this project.

Austin Click_____

I, Darnisha Crane, approve this project report for submission. Information contained within is, to the best of my knowledge, complete and true, and in accordance with the process used throughout the completion of this project.

Darnisha Crane_____

I, Walter Dietzler, approve this project report for submission. Information contained within is, to the best of my knowledge, complete and true, and in accordance with the process used throughout the completion of this project.

Walter Dietzler_____

I, Jacqueline Pepper, approve this project report for submission. Information contained within is, to the best of my knowledge, complete and true, and in accordance with the process used throughout the completion of this project.

Jacqueline Pepper _____

I, Hunter Phillips, approve this project report for submission. Information contained within is, to the best of my knowledge, complete and true, and in accordance with the process used throughout the completion of this project.

Hunter Phillips_____

# 2. Abstract

To test the accuracy and implementation of flex sensor technology for robotic industry, this project focuses on measurement of angle of a robotic arm using a FlexPoint sensor. The sensor is an inexpensive solution to measuring a robotic arm angle, which previously was a difficult and expensive process. This device allows the students to measure an angle accurate to ±6.36%. Angle measurement of a robotic arm has previously been acquired in research using a potentiometer. This method requires multiple mechanical connections that have potential to fail, as well providing a force against the stepper motor. In contrast, the Flex sensor is a single device that does not impede the movement of the robotic arm. The Flex sensor works by reading a voltage that changes with the bend of the sensor. The voltage is converted to resistance and finally to angle in Matlab. The angle is then displayed live time in Matlab on a compass plot and the data is output to excel to allow easy visibility of the data for users.

# 3. Device Production

## 3.1. Concept Development

The project was originally conceptualized as a proof of concept for a new robotic joint measurement mechanism. The objective was to measure the angle of the rotational axis with a new, supposedly higher accuracy, bend sensor. This approach would hopefully lower costs for robotic design, thus enabling quicker turnarounds for cutting edge designs. It was decided that the goal was to provide these measurements in real-time with a controllable test situation to simulate realistic environments.

To ensure reliably and repeatability it was decided that the robotic joint needed to be locked down on a test bed with high precision components to move the assembly. A realistic looking robotic joint with proper extending lever arms was decided against since this was not the point of the project. Being that the purpose was purely for an angle measurement, the superficial accuracy of replicating the extensions would prove no additional value to this particular project. This resulted in a pointer arm being decided on due to its ability to provide higher accuracy references against the mounting plate.

## 3.2. Electrical Schematic

The electrical design was created in Eagle to ensure proper connections were made. An Electrical Rule Check (ERC) was then applied to the circuit to ensure no faults and missing connections. The resulting schematic, **Figure 1**, was created with the important circuit divisions highlighted for further explanation of testing below.

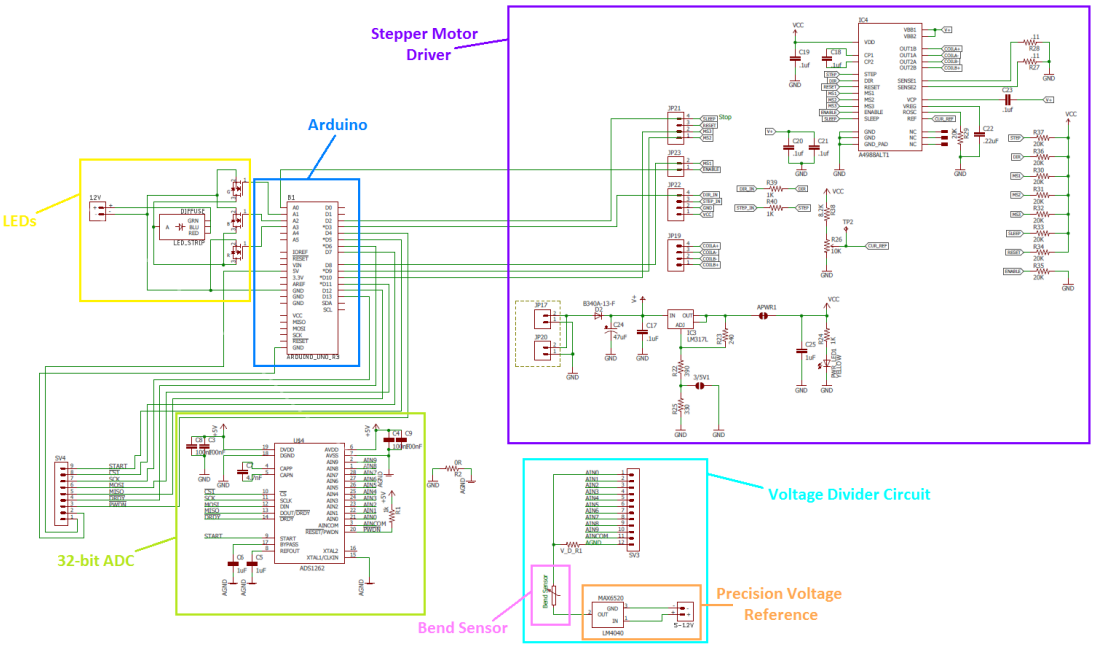



**Figure 1: Circuit Schematic**

The Arduino, stepper motor driver, and 32-bit ADC were already pre-fabricated boards, thus they did not need further electrical testing. The voltage divider circuit that contains the bend sensor and precision voltage reference needed validation through circuit simulation. The circuit was loaded into LTspice with the bend sensor being simulated with a variable potentiometer. The varying voltage that was to be fed into the 32-bit ADC was verified in simulation. The LEDs circuit was then tested to verify the MOSFETs provided proper switching for real time measurements. It was noted that the typical larger TO-220 MOSFET packages extra power dissipation was not needed for the amperage draw of the 12 V LEDs. These were exchanged for the smaller form factor SOT-23 package which enables a cleaner physical circuit design.

## 3.3. Programming Overview

The coding flow was that the Arduino would feed data into Matlab that would visually display the information and output the appropriate values to Excel. This logical design can be seen below in **Figure 2**.

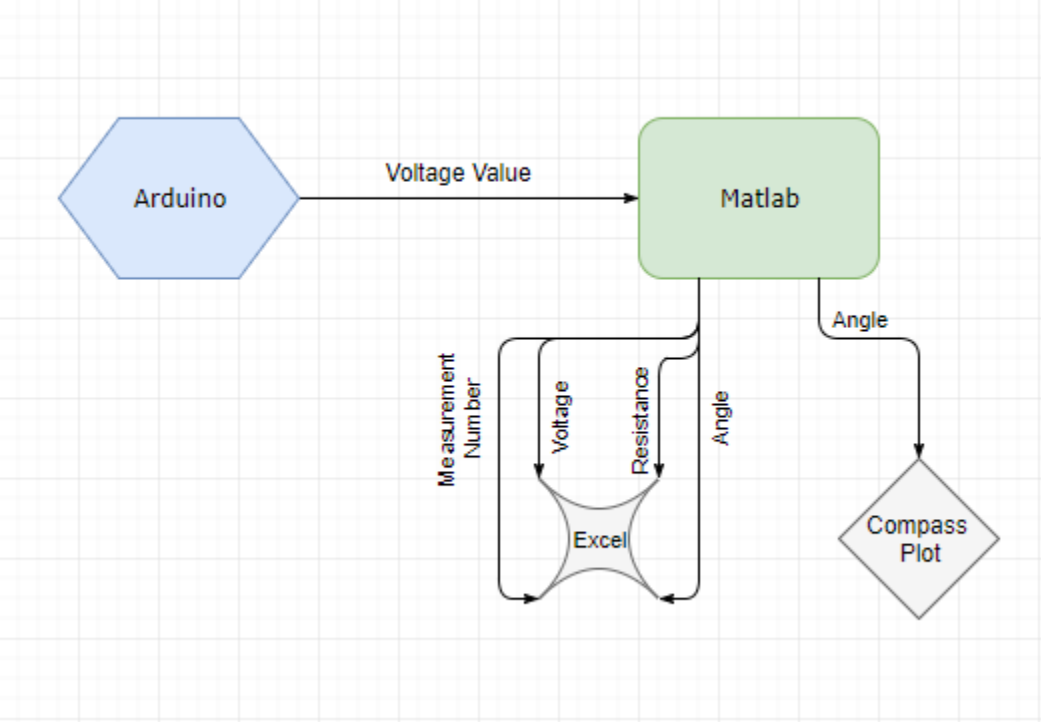



**Figure 2: Basic Programming Schematic**

## 3.4. Arduino Programming Analysis

The Arduino code consists mainly of the necessary functions to process the incoming 32-bit ADC byte chain. The code uses a custom ads1262.cpp and ads1262.h library to parse the appropriate ADC count [1]. SPI and SoftwareSerial were also needed for the Arduino to receive the output of the 32-bit ADC. Once the libraries were chosen the code seen below in **Figure 3**.

```
#include <SPI.h>
#include <SoftwareSerial.h>
#include <math.h>
#include <ads1262.h>

#define PGA 1                    // Programmable Gain = 1
#define VREF 5.00                // Internal reference of 2.048V
#define VFSR VREF/PGA
#define FSR (((long int)1<<23)-1)
ads1262 PC_ADS1262;                    // class
float volt_V=0;
float volt_mV=0;
volatile int i;
volatile char SPI_RX_Buff[10];
volatile long ads1262_rx_Data[10];
volatile static int SPI_RX_Buff_Count = 0;
volatile char *SPI_RX_Buff_Ptr;
volatile int Responsebyte = false;
volatile signed long sads1262Count = 0;
volatile signed long uads1262Count = 0;
double resolution;
bool direct_step = HIGH;

void setup()
{
  // initalize the  data ready and chip select pins:
  pinMode(ADS1262_DRDY_PIN, INPUT);                  // data ready input line
  pinMode(ADS1262_CS_PIN, OUTPUT);                   // chip enable output line
  pinMode(ADS1262_START_PIN, OUTPUT);                // start
  pinMode(ADS1262_PWDN_PIN, OUTPUT);                 // Power down output

  Serial.begin(1200);
  // initalize ADS1292 slave
  PC_ADS1262.ads1262_Init();                         // initialise ads1262
 }

void loop()
{
 volatile int i,data; // init data for 32-bit adc
 if((digitalRead(ADS1262_DRDY_PIN)) == LOW)               // monitor Data ready(DRDY pin)
  {
    SPI_RX_Buff_Ptr = PC_ADS1262.ads1262_Read_Data();      // read 6 bytes conversion register
    Responsebyte = true ;
  }
  if(Responsebyte == true)
  {
    for(i = 0; i <5; i++)
    {
      SPI_RX_Buff[SPI_RX_Buff_Count++] = *(SPI_RX_Buff_Ptr + i);
    }
    Responsebyte = false;
  }

  if(SPI_RX_Buff_Count >= 5)
  {
    ads1262_rx_Data[0]= (unsigned char)SPI_RX_Buff[1];  // read 4 bytes adc count
    ads1262_rx_Data[1]= (unsigned char)SPI_RX_Buff[2];
    ads1262_rx_Data[2]= (unsigned char)SPI_RX_Buff[3];
    ads1262_rx_Data[3]= (unsigned char)SPI_RX_Buff[4];

     uads1262Count = (signed long) (((unsigned long)ads1262_rx_Data[0]<<24)|((unsigned long)ads1262_rx_Data[1]<<16)|(ads1262_rx_Data[2]<<8)|ads1262_rx_Data[3]);
     sads1262Count = (signed long) (uads1262Count);        // get signed value
     resolution = (double)((double)VREF/pow(2,31));        // resolution= Vref/(2^n-1) , Vref=2.5, n=no of bits
     volt_V      = (resolution)*(float)sads1262Count;      // voltage = (resolution)(adc count)

     serialFloatPrint(volt_V); // call function to print ADC reading as a parsed float
     Serial.println();

   }
  SPI_RX_Buff_Count = 0;
}
```

**Figure 3: Arduino Code**

Since the device communicates over the SPI interface the first section of variable initialization requires a lot of parsing and timing setup. The void setup() enables the starting sequence for the 32-bit ADC over SPI and creates the Serial connection with the computer. It was discovered that the Matlab portion of the code was unable to process data coming in faster than a 1200 baud rate, thus the typical baud rate of 9600 was not used to enable real-time graphing. The void loop() initially checks if new data is available on the SPI line and if it is, it immediately starts consuming the bytes into an array. Once the byte count has been concatenated into an array they are shifted into a long data type. The resolution of the ADC is then calculated with respect to our VREF coming from the Arduino of 5.00 V. The corresponding voltage is then calculated using the determined resolution and prepared for Serial output to Matlab.

```
void serialFloatPrint(float f) {
  byte * b = (byte *) &f;
  Serial.print("f:");                                  // print delimiter
  for(int i=0; i<4; i++) {

    byte b1 = (b[i] >> 4) & 0x0f;                     // split the float chain into a 4 byte partioned chain
    byte b2 = (b[i] & 0x0f);                          // prepends the upper nibble of a byte

    char c1 = (b1 < 10) ? ('0' + b1) : 'A' + b1 - 10;     // encode the byte process in a standard IEEE HEX Representation
    char c2 = (b2 < 10) ? ('0' + b2) : 'A' + b2 - 10;     // prevents incorrect matlab readings due to delimeter tainting

    Serial.print(c1);                                 // prints the encoded hex to the serial line
    Serial.print(c2);
  }
}
```

**Figure 4: Serial Print Float**

This is where a roadblock was encountered when outputting a high precision float measurement. Arduino only supports integer Serial output and this was an unacceptable form of data writing. To solve this issue a serial float printing function was written using the IEEE standard as seen in **Figure 4**. Using a standard delimited 4-byte partitioned chain with the top byte being shifted to the upper nibble an array was formed of a separated float chain. These array elements were then encoded using the standard IEEE HEX representation with proper protection against delimiter tainting across the serial output chain. These converted hex value char chains are then outputted across the serial line to be read in by Matlab.

## 3.5. Matlab Programming Analysis

The Matlab program starts off with setting up the Excel variable workspace as seen in **Figure 5**.

```
% Setup Excel X Server
Excel = actxserver ('Excel.Application');
File='C:\Users\robolux\Desktop\311robot\demo\test1.xls';
if ~exist(File,'file')
ExcelWorkbook = Excel.workbooks.Add;
ExcelWorkbook.SaveAs(File,1);
ExcelWorkbook.Close(false);
end
invoke(Excel.Workbooks,'Open',File); % open the invoked Excel workbook
```

**Figure 5: Setting up Excel X Server**

The problem with the xlswrite function built into Matlab is that it starts a new writing session every time it is called. This causes a ~700 millisecond delay in the main loop which critically delays the real-time aspect of the sensor measurements. By setting up the X server of Excel the function does not have to continually invoke the session as it stays permanently open until the end of the program.

The initialization section of the script seen below in **Figure 6** showcases the variable declarations and purging sequences.

```
mean_angle=[];  % init mean_angle
trip = 2;       % dummy var
w=0;            % need w for assigning positions in arrays during while loop
data2 = [];     % init data 2 to null matrix
value = [];     % initi value to null matrix
figure('units','normalized','outerposition',[0 0 1 1])  % setup the figure
comp = compass(1,1);                        % compass graph
arduino=serial('COM4','BaudRate',1200);     % open up the com port for the arduino
fopen(arduino);                             % at a baud rate of 1200

while(trip~=102)            % the first bytes before 102 of information need
    trip=fread(arduino,1);  % to be read and assigned to a dummy variable
end                         % they contain unreadable characters / symbols

data = 102;                                 % Read one more byte chain
data = [data; fread(arduino,11,'char')];    % and dump the data into a dummy var
                                            % it should be valid, however ensures that proper flushing occurs

mapc = @(x) 4.2619958863*(10^(-10))*x^3 - 2.1440571377*(10^(-5))*x^2 + 3.3771916419*(10^(-1))*x - 1.5811722057*(10^(3));
```

**Figure 6: Matlab Initialization**

The main declarations serve the purpose of setting the variables and figure states before the loop. The serial line is opened with a baud rate of 1200 to correspond to the Arduino script. The following while loop is used to send all data on the serial line before 102 to a dummy variable. This ensures that the byte reading sequence is correct and starting from the 0 position, the delimiter (102), since Matlab can subscribe at any part of the byte chain. An initial set of 11 chars is read in to ensure that readable data is being

read in. If this passes the test the looping sequence begins, as seen in **Figure 7**, and the initialization is over.

```
while(1)            % infinite loop
    delete(comp)    % every iteration need to delete the old arrow from compass plot -> for speed
    w=w+1;          % iterate for array assignment advancement
    data2 = fread(arduino,12,'char');   % get data from serial com
    data_char = char(data2);            % ASCII is converted from raw data chain
    data_converted = [data_char(9) data_char(10) data_char(7) data_char(8) data_char(5) data_char(6) data_char(3) data_char(4)];
    value = hexsingle2num(data_converted); % data comes in backwards and needs to be rearranged and converted
                                           % from single precision IEEE hexadecimal string to number
    flexR = v2r(value);                    % value is in form of voltage, need to convert it to a resistance
    map_value = mapc(flexR);    % map the 0 - 90 degree angles to the straight and bent corresponding resistance values using cal curve

    data_point = sprintf('A%s',num2str(w));
    v_col = sprintf('B%s',num2str(w));
    r_col = sprintf('C%s',num2str(w));
    a_col = sprintf('D%s',num2str(w));
    xlswrite1('test1.xls',w,1,data_point)       % record data point number
    xlswrite1('test1.xls',value,1,v_col)        % record voltage
    xlswrite1('test1.xls',flexR,1,r_col)        % record resistance
    xlswrite1('test1.xls',map_value,1,a_col)    % record angle

    mean_angle = [mean_angle, map_value];                                   % get the mean angle values stored from the measurements
    [x_c,y_c] = pol2cart(map_value*(pi/180),1);                             % convert angle value from polar to cartesian cord.
    comp = compass(x_c,y_c);                                                % create compass polt with angle
    ylim([0 1])                                                             % only show top half of plot
    if mod(w,250)==0                                                        % every ~100 measurements ask user for input
        fprintf('Mean Angle Measurement : %.2f\n\n',mean(mean_angle))       % Show user the mean of the ~250 measurements
        mean_angle = [];
        user_dec=input('\n\nEnter 1 to increment the position of the stepper motor\nEnter 2 to take data points at position\nEnter 3 to end testing : ');
                                                                            % ask user to enter value for choice
        if user_dec == 3 % if they pick 3
            break       % exit loop and shutdown processes
        end
        if user_dec == 1
            fprintf(arduino,'%s',char(user_dec));
            pause(0.5)
            fprintf(arduino,'%s',char(user_dec));
        end
        fclose(arduino);
        fopen(arduino);
    end
    drawnow update % force the figure to update even if OPENGL denies it due to timing
end
```

**Figure 7: Main Matlab Loop**

The testing loop begins with a few housekeeping items to ensure each looping iteration starts with a proper clean process. Then the 12-byte data chain is read into Matlab with fread. The ASCII sequence is then converted from the data chain. For some undetermined reason the data does not come in linearly and needs to be rearranged into a custom sequence. This array is then converted from the single precision IEEE hexadecimal string to the final voltage value that was sent from the Arduino. This voltage value is then converted into a resistance value using custom values in the v2r function as seen below in **Figure 8.**

```
function flexR = v2r(volt_V)

R_DIV = 46880.0;     % Measured resistance of 47k resistor
VCC = 4.096;         % Measured voltage of precision 4.096 voltage reference

flexR = R_DIV * (VCC / volt_V - 1.0);

end
```

**Figure 8: v2r Function**

The resistance value is then used to calculate the resulting angle using the calibration curve equation. The data values: data number, voltage, resistance, and angle are then

formatted and recorded into the Excel file using xlswrite1 (a modified version of xlswrite). Since there appears to be noise in the value measurements a mean angle is then calculated to display in the command prompt once the set number of data points has been taken. The angle value is then converted to Cartesian coordinates which the compass graphing function takes in as its arguments. The function ylim is used to limit the viewing frame to the upper half of a circle since the measurements taken only range from ~0 to ~90 degrees. The following if statement determines when to stop taking measurements using the modulus of the iteration variable, w. It was set to 250 measurements as this was determined to be a good data set for each angle being measured. If it was triggered it displays the mean angle measurement of the 250 points and asks the user what they would like to do: take more data points at the angle position, advance the stepper motor, or end testing. The last thing in the while loop, drawnow update, forces the figure to update even if Matlab thinks it is a waste of computing time.

Once the loop has been ended by the user, the Arduino serial line is closed, and the Excel X Session is closed and saved as shown in **Figure 9**.

```
81 -    fclose(arduino); % close arduino
82
83      % close Excel and save work
84 -    invoke(Excel.ActiveWorkbook,'Save');
85 -    Excel.Quit
86 -    Excel.delete
87 -    clear Excel
```

**Figure 9: Closing Arguments**

## 3.6. Code Testing

The code was tested extensively using a compile and debug approach typical in software / hardware implementation design. This was a relatively fluid process and enabled improvement iterations to be fast. All code was stored in a GitHub repository so that it could be worked on even when not in physical contact of the testing apparatus. This also made it easier to work on the project as a group since all the files were stored in the cloud and synced locally. The code was tested for varying periods of time to ensure that it could handle extremely long testing periods without slowing down or breaking. During the calibration process the code experienced over a 700 user inputs and 70000+ measurements thus validating its robustness to real world use.

## 3.7. CAD Modeling and Machining

*a. Machining Metal Protractor Plate*

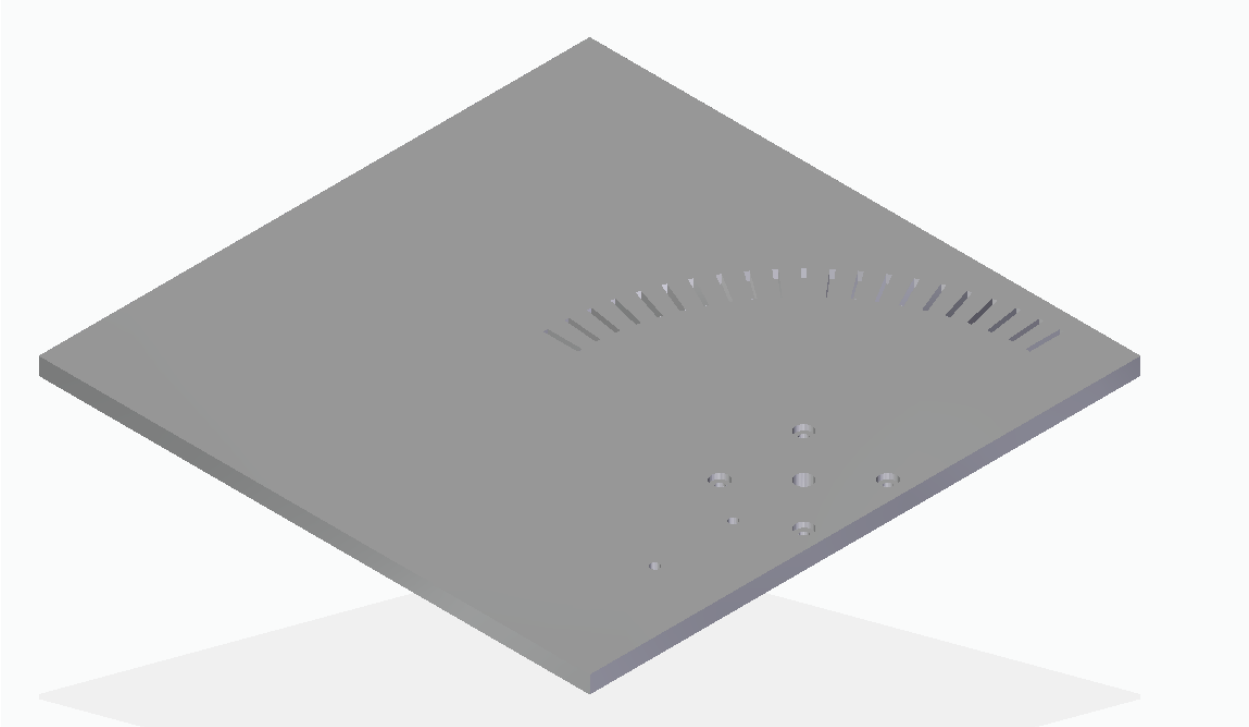


**Figure 10: CAD of fully machined plate**

The protractor plate shown in **Figure 10** was machined out of MIC6 aluminum, due to its machinable properties. The material was chosen due to its high dimensional stability. There were a total of 5 different processes used in the machining of the plate. The processes implemented were:

Process 1:

The first process was a hole process used to create center-drilled holes. The tool used was a #3 center drill, as it would provide a hole small enough to center both a 1/8” and a 1/4" drill bit so that they would not walk.

Feeds and Speeds:

- Spindle Speed: 6000 RPM
- Plunge Rate: 15 IPM

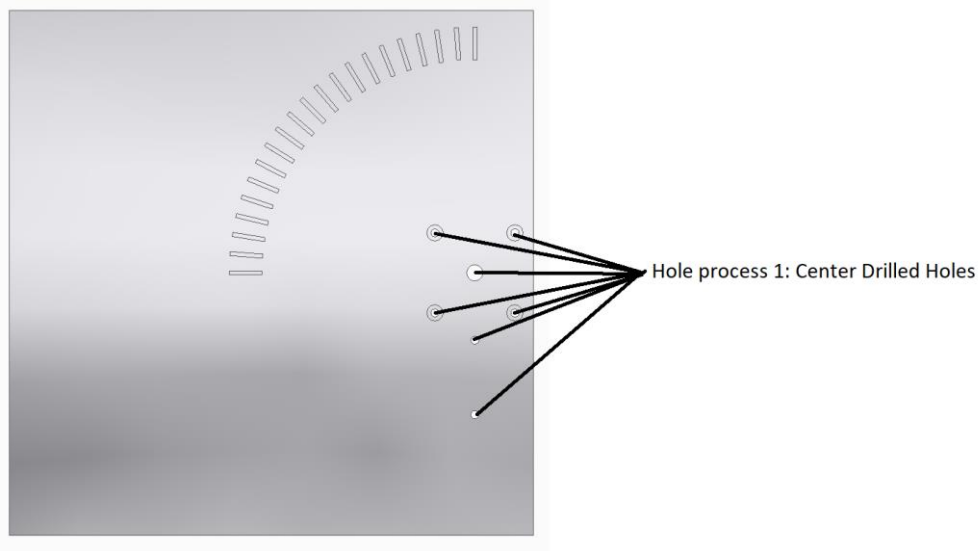



**Figure 11: Center Drilled Holes**

Process 2

The second process was a hole process to drill the six 1/8” holes for the screws. Every screw being used on the project (M3s and #4) had a shank size slightly less than 1/8”, so all the holes where screws would be mounted were drilled to be 1/8”. The 4 holes in a square pattern Marked in black, **Figure 11** were to fit the M3 screws being used to mount the stepper motor. The two holes in a line Marked in blue, **Figure 12** were used for the #4 screws that were used to mount the backplate which held the sensor.

Feeds and Speeds:

- Spindle Speed: 3000 RPM
- Plunge Rate: 15 IPM

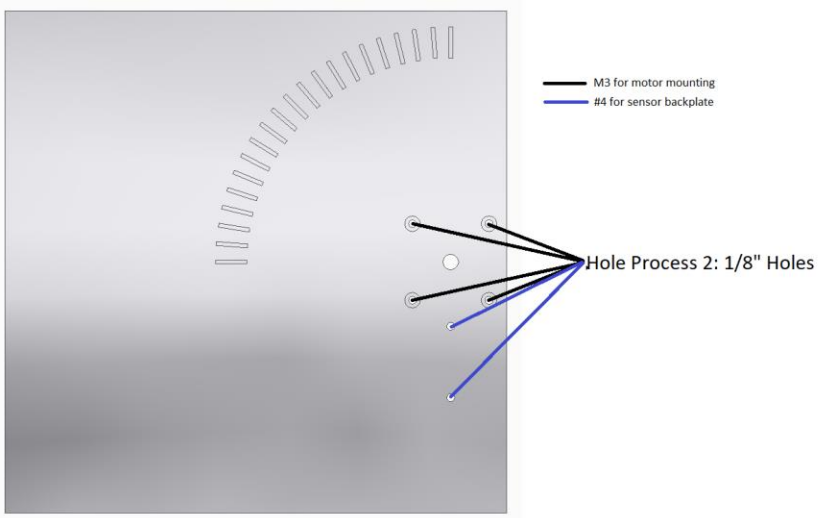



**Figure 12: 1/8” Screw Mounting Holes**

Process 3

The third process included drilling a 1/4" hole to make room for the shaft of the stepper motor – shown in **Figure 13**. The stepper motor shaft was 5mm, or ~0.197in, in diameter. The 1/4" hole gave enough room to guarantee the motor rotated cleanly.

Feeds and Speeds

- Spindle Speed: 3000 RPM
- Plunge Rate: 15 IPM

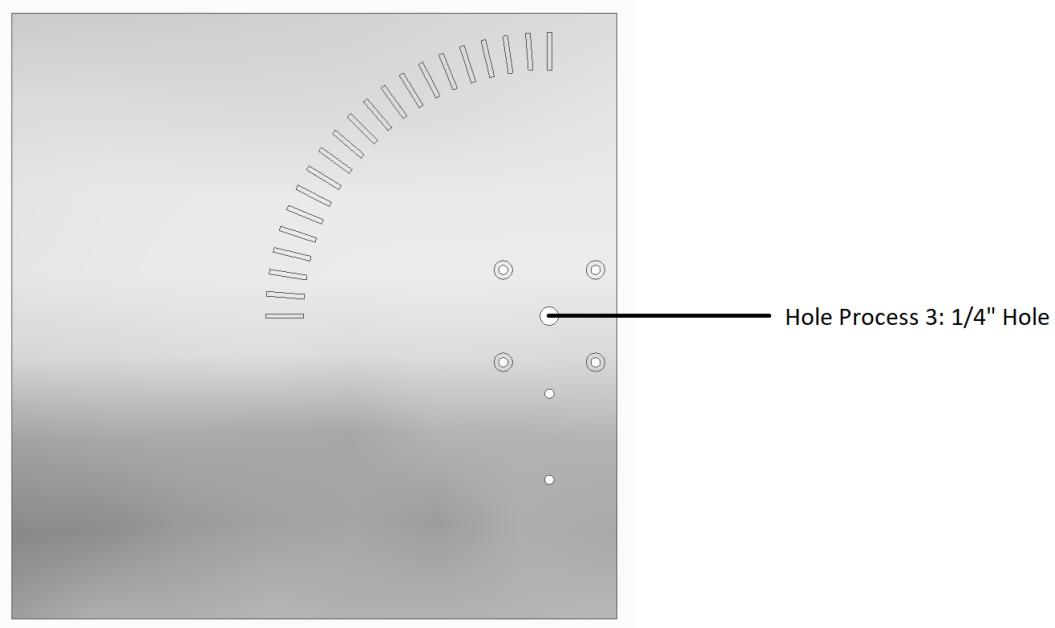



**Figure 13: 1/4" Hole**

Process 4

The fourth process was a pocketing process used to pocket out the spots in the metal where the screw heads used for mounting the motor would be recessed. A 3/16” end mill was used for this bit, as it was slightly smaller than the pocket size, so it made the machine circle and create a clean cut. The bolt heads were recessed to allow rotation of the protractor pointer. The pocket is 1/8” deep, to accommodate the 2.5mm (~0.098in) screw head. **Figure 14** shows the locations of the holes on the plate in relation to the other geometric entities.

Feeds and Speeds

- Spindle Speed: 6500 RPM
- Feed Rate: 30 IPM

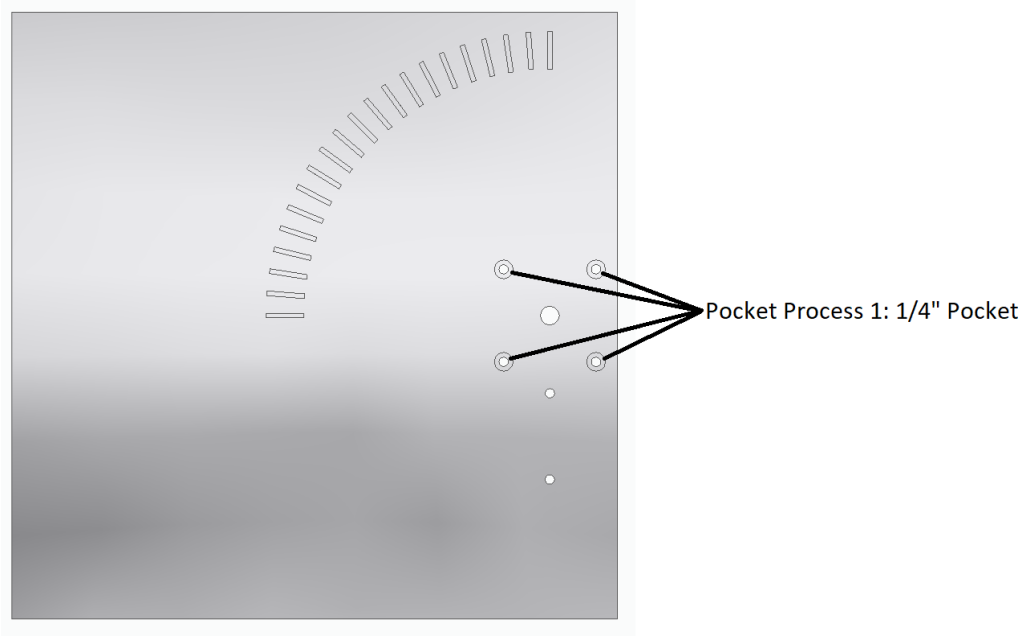



**Figure 14: Pocketed Screw Head Recess**

Process 5:

The final process was to create the grooves that indicate angle. This was done using a surfacing process using an engraving bit. The grooves had to be shallow, approximately 0.01” deep, to not break the bit during machining, due to the strength of the material. The grooves are highlighted in **Figure 15**.

Feeds and Speeds:

- Spindle Speed: 6500 RPM
- 30 IPM

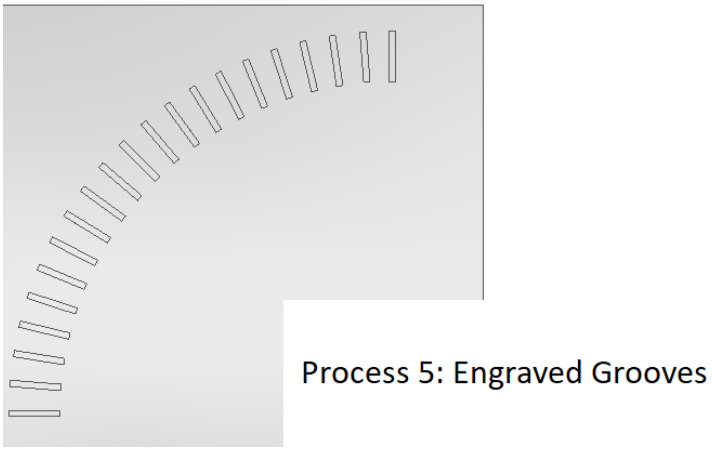



**Figure 15: Engraved Grooves**

*b. Pointer and Sensor Mount*

The pointer and mount for the sensor had to be made by additive manufacturing, as traditional manufacturing methods would have been ineffective. Both the pointer and the sensor mount were printed out of PLA. ABS was another potential material, but printing ABS is more expensive and time consuming.

For the pointer **Figure 16**, the printer did a 10% fill grid. This means that of all the volume being used by the pointer, 10% of it was PLA, the rest was air. 10% is a relatively low fill, so the piece was lighter and able to be printed much faster than if a higher fill was used. Because there was no load of any significance being exhibited on any part of the pointer, it made sense for it to be a low fill, so that it could be printed faster.

For the Sensor Mount **Figure 17**, the fill had to be slightly higher, due to the fact that there was a point of worry **Figure 18**. This point was very thin, < 1/16", and it was also the point at which the sensor bent around. The worry was that the bending senor may flex the thin piece point out of being straight, thus throwing off measurements. Using a 25% fill, the point was much sturdier and resistant to flexing. It took a little longer to print, but it helped prevent erroneous data at large bends.

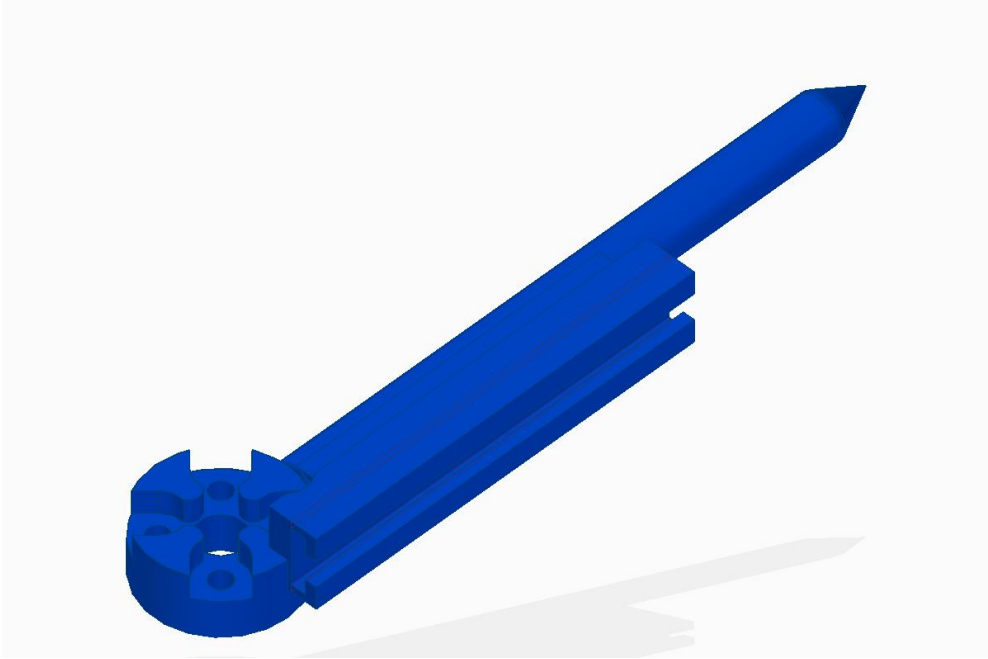


**Figure 16: Pointer**

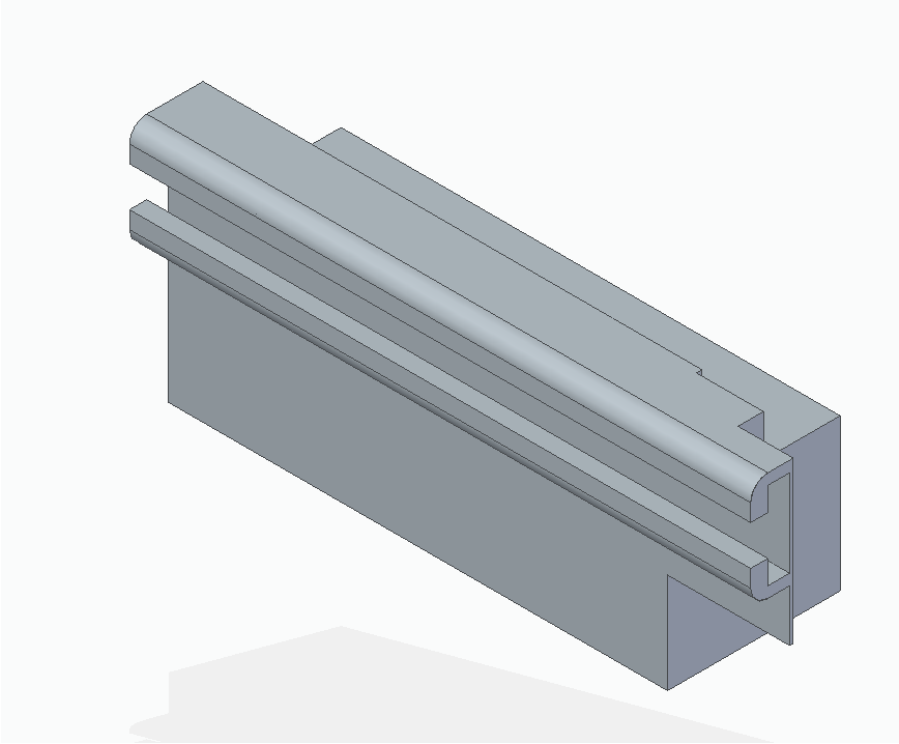


**Figure 17: Sensor Mount**

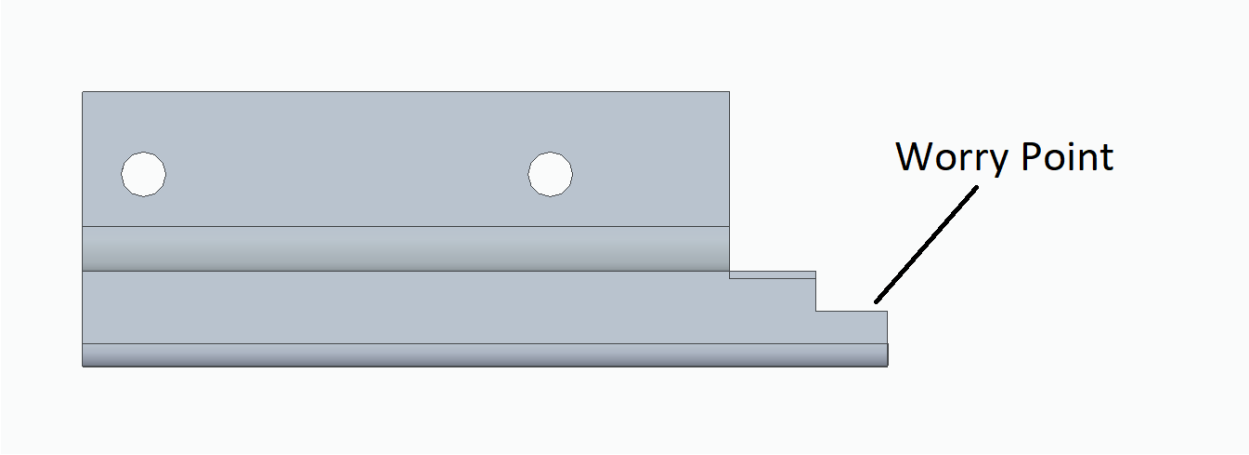



**Figure 18: Top view of Sensor Mount with Worry Point**

*c. Polycarbonate sides*

The polycarbonate sides were cut to size on a vertical bandsaw. The polycarbonate sides were attached with double sided tape. This was acceptable since the sides were not under any load. Without a load, the tape was strong enough to hold it together.

## 3.8. Assembly and Wiring

The assembly process started with connecting the aluminum top plate to the four plexiglass sides. After test drilling holes into the corner of the plexiglass it was determined that cracking would most likely occur. This led to the use of 3M double sided clear adhesive tape that is typically used to hold windows in their frame. This provided over 100 theoretical pounds of adhesion between the aluminum and plexiglass providing a robust structural enclosure to contain the testing equipment. The resulting completed enclosure can be seen below in **Figure 19**.

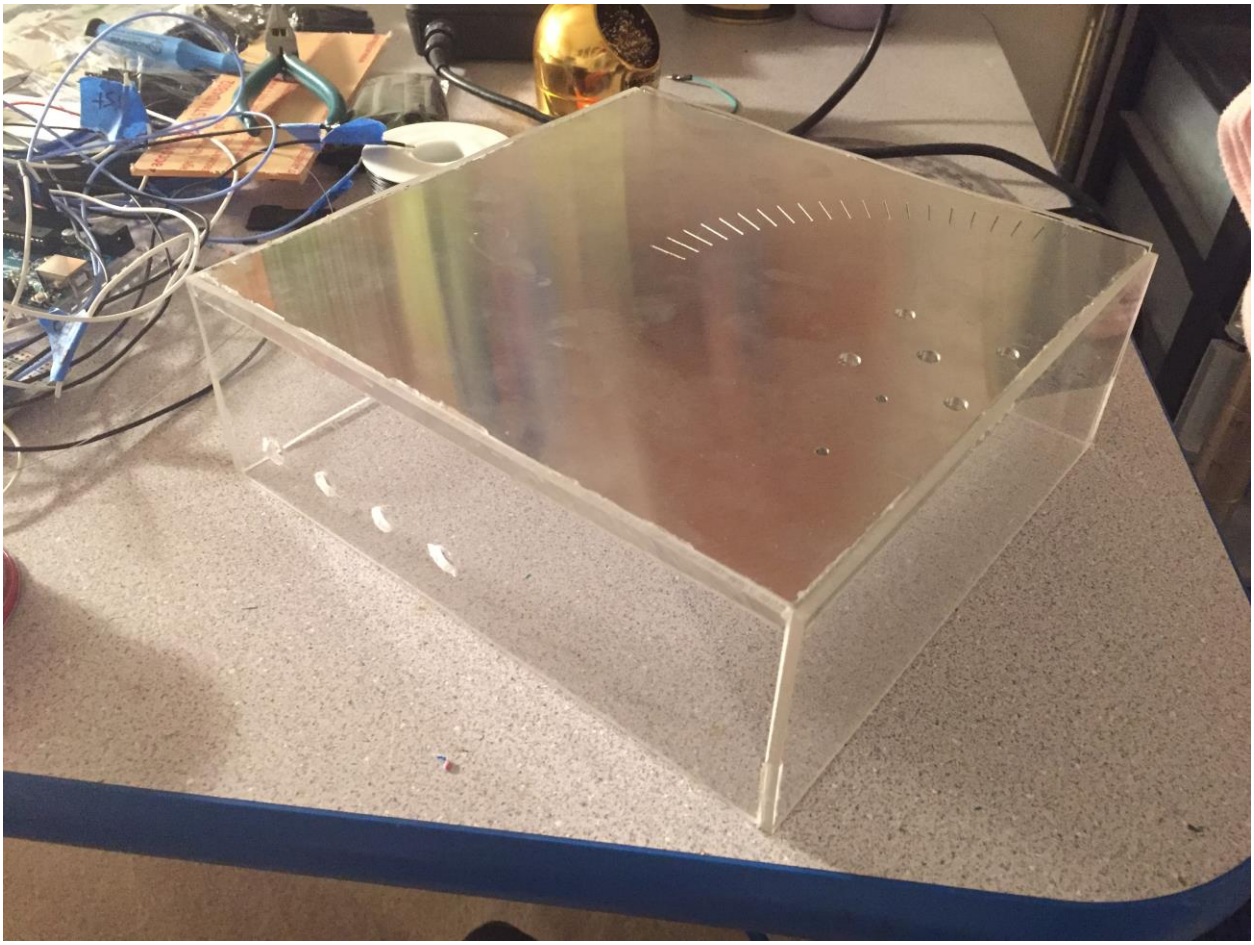


**Figure 19: Assembled Enclosure**

The wiring had been utilized on a breadboard during initial testing as seen in **Figure 20**.

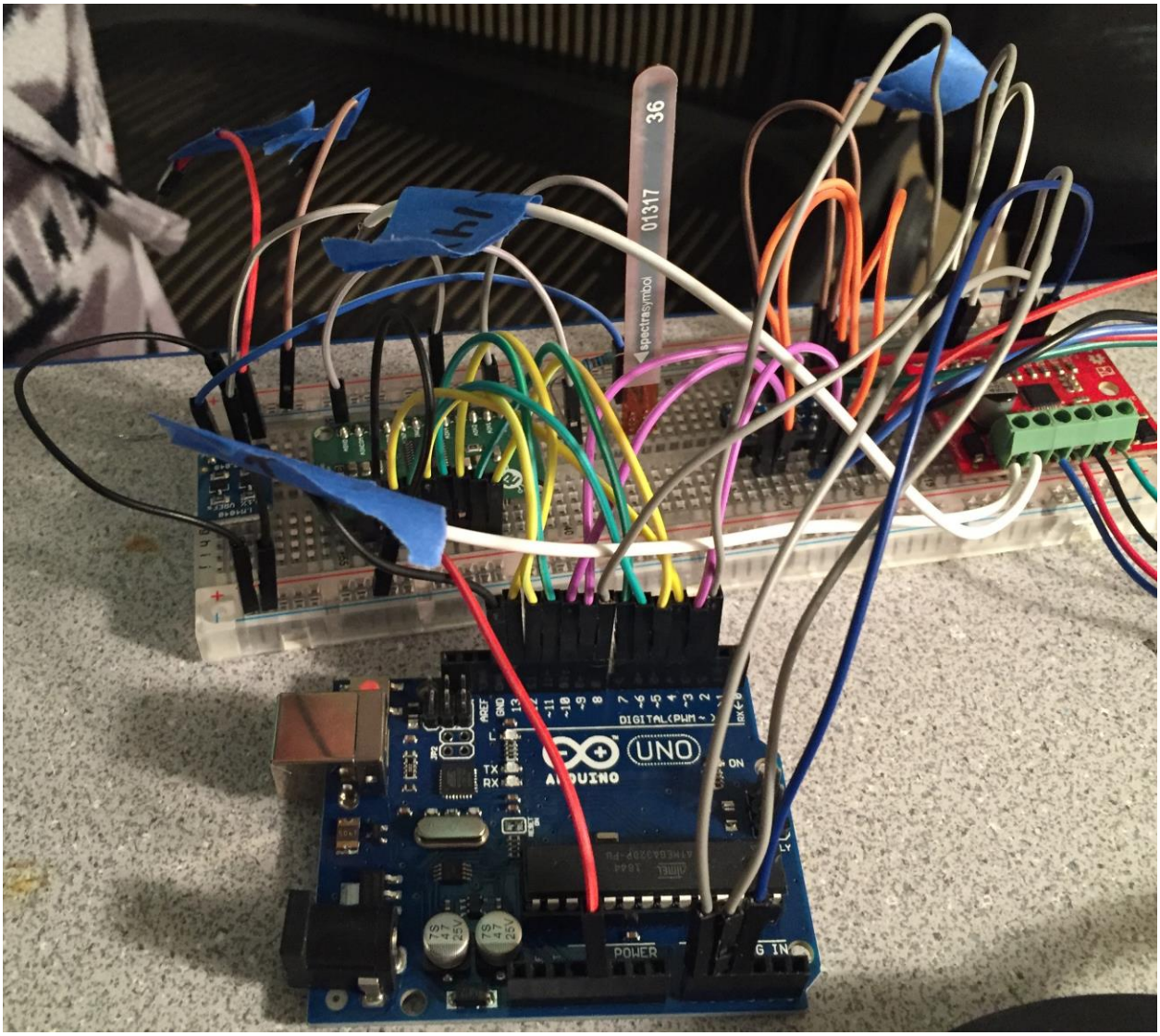



**Figure 20: Prototype Breadboard Wiring**

This needed to be converted to a more permanent solution once it had been verified that it worked. This came through the use of a solderless breadboard, which all the components were transferred to, populating it as seen in **Figure 21**.

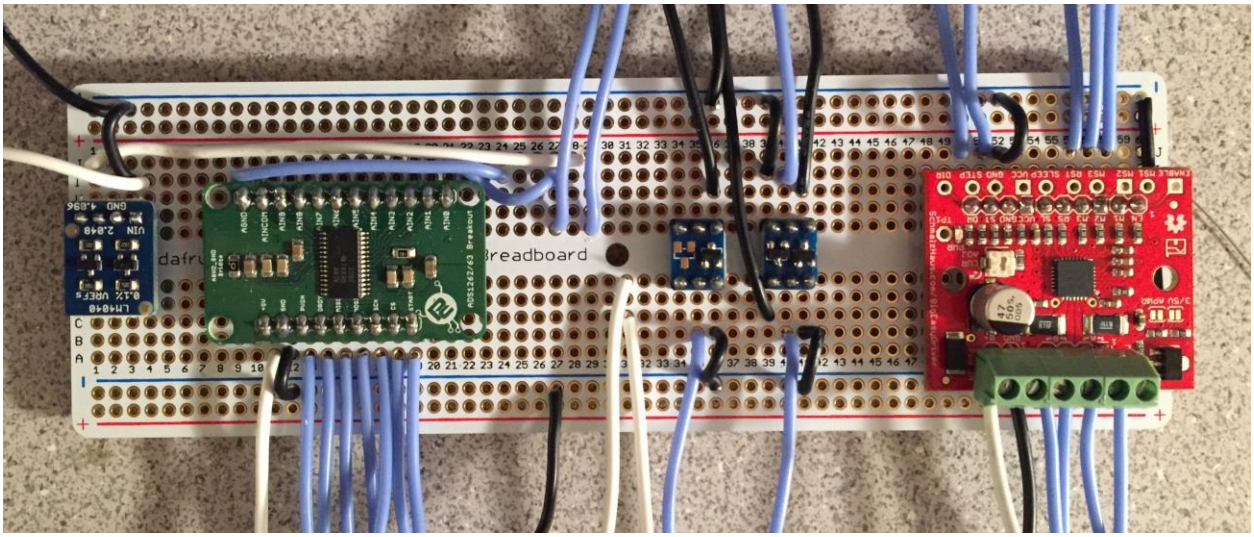


**Figure 21: Solderless Breadboard Wiring**

The power jacks were then attached to the side of the enclosure and the remaining power connections were soldered together. Once all of the connections were completed, the solderless breadboard and Arduino were stuck to the bottom side of the machined plate with 3M double sided pads. They were used to ensure that no shorts occurred between the pins on the lower side of the electronics, thus providing an electrically nonconductive insulating layer. The stepper motor was then mounted in place and the wires were screwed into the terminal connectors on the stepper motor driver. This completed the inside of the enclosure that can be seen below in **Figure 22**, on the next page.

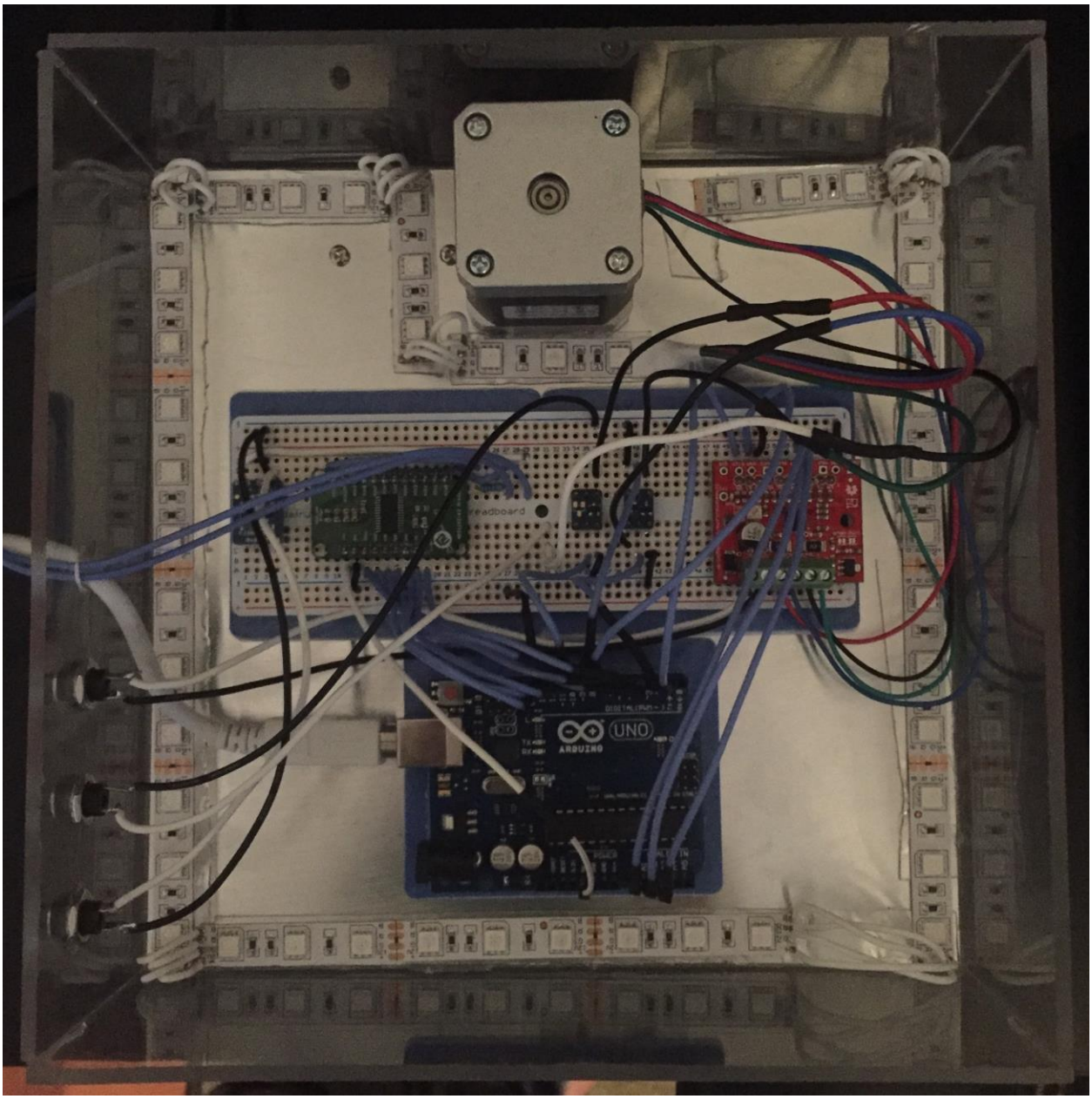



**Figure 22: Inside the finished Enclosure**

The upper side of the aluminum plate was then populated with the 3D printed components. An aluminum hub was screwed into the 3D printed part and mounted on the stepper motor shaft by turning the set screw. The other half of the 3D printed assembly that holds the flex sensor was secured using screws to the aluminum plate. An attempt was then made to slide the flex sensor and it was discovered the plastic had slightly expanded. An old rusted throttle cable was used as an abrasive surface inside the channel to widen it slightly. After this the flex sensor was successfully slid into the 3D printed assembly and locked into place perfectly. The completed upper portion of the assembly can be seen on the next page in **Figure 23**.

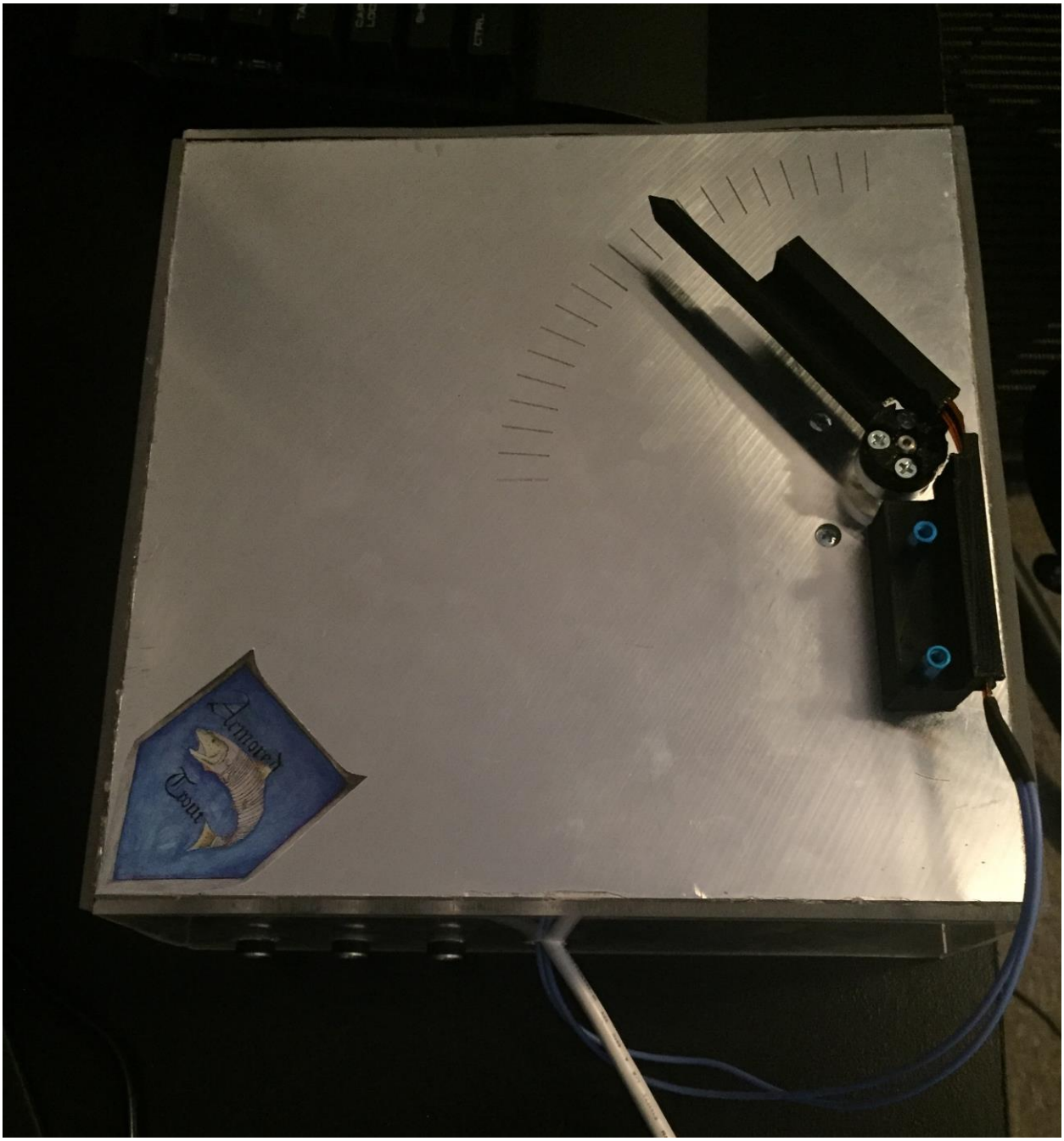


**Figure 23: Finished Enclosure**

## 3.9. Testing Setup

Once assembly was completed, tests were completed to ensure the device functioned as expected. A large offset was initially observed with the measurements; however this was later fixed with the calibration. Wide sweep testing of the stepper motor showed that it needed fine tuning to work properly with the stepper driver being loaded at a perpendicular angle. During testing three different power supplies had to be plugged in: 5V for the voltage reference, 12 V for the LED strip, and 14 V for the stepper motor. This was in addition to the USB cable providing data communication to the computer. Although, once the enclosure was powered and the Matlab script was run, data would consistently flow into the script and its logic flow. Once testing was completed and all necessary scripts were terminated, the enclosure could be disconnected and stored until further testing was desired.

# 4. Theory of Operation

## 4.1. Flex Sensor

*a. Background*

The device reads measurements through the Flexpoint bend sensor. The Flexpoint bend sensor we are using is a single layer one-directional bend sensor. This means that the sensor can only take data when bent in one direction. This sensor is a coated substrate that changes electrical conductivity as it is bent. The single layer design reduces risk of sensor damage and interference due to dirt, heat, liquid, and pressure effects [2]. These sensors are durable and used in variety of applications such as the measurement of mechanical movement, air and water flow, vibration and can be used as a durable switch in harsh environments [3]. Flexpoint makes a bi-directional sensor which is being applied to technology in gloves and the bio-medical field due to its durability and ease of bending [4].

*b. Mounting*

Mounting the digital sensor in a perfect fit was essential in ensuring slippage did not occur. Pressure mounts through the use of screws were considered but there was concern for unreliable resistance measurements [5]. It was decided the channel would be made slightly too small and softly ground out afterwards until the sensor fit perfectly. This worked quite well and the sensor slid as expected on the arm portion and was locked in by friction on the secured end of the mechanism.

*c. Measurement Type*

The measurement of angles tends to have a myriad of options available to the user for sensors. Instead of using a linear movement with a rotational potentiometer which is typically used on robotic joints a change in voltage is much easier to parse with one application mounting [6]. The change in resistance of the sensor has just recently been brought to such fine detail. This translates to a more accurate and precise angle being obtained by the sensor. The angle is critical for fast moving control of the robotic leg. This high accuracy angle measurement translates to multiple engineering fields if properly sampled at a 32-bit rate.

*d. Resolution and Sensitivity*

It was expected that miniscule twitching from the stepper motor would provide a slight vibration thus skewing the measurements in either direction [7]. It was also discovered that since the tiny surface mount die resistors on the polyimide base layer were being constantly moved ever so slightly that the resulting change was quite large. Combined with the 4294967296 theoretical levels translating to voltage increments of 9.5367432e-10 Volts it can be seen the high sample rate with floating point precision translated to a wide swing of measurements. Although when the average of the measurements was taken it could be seen that they tend to center their origin around the desired angle, thus with DSP could be filtered properly.

## 4.2. Voltage Divider

A voltage divider circuit was used in order to reduce the output voltage and have an accurate reference voltage to use for the conversion calculations. The varying resistance of the flex sensor can be described by the voltage divider equation as seen below in **Equation 1**.

$$V_{out} = V_{in}\left(\frac{R_2}{R_1+R_2}\right) \quad \textbf{(1)}$$

Where $V_{in}$ is input voltage, $V_{out}$ is output voltage, and $R_1$ and $R_2$ are the resistance values of the acting resistors in the circuit. The typical wiring schematic for a voltage divider circuit can be seen below in **Figure 24**.


**Figure 24: Voltage Divider**

In the case of this project, $R_1$ is a varying resistance due to the fact it is represented by the flex sensor. $R_2$ and $V_{in}$ are known pre-measured values that do not change, thus enabling the equation to be solved for R1 to know how much the flex sensor has changed position.

## 4.3. Stepper Motor Driver

The stepper motor has input pins that receive current from a supply. Stepper motors are driven by energizing the phases in a patterned sequence to make a step occur between coils. Using the generated waveform to command a sequence of steps enables precise control over the test bed environment. By varying the phase angle between coils the stepper motor driver is able to enable microstepping mode [8]. Microstepping provides the ability to move the stepper motor at finer steps than listed by the manufacturer through the use of phase current angles. Through the myriad of features available, the stepper motor driver is able to effectively provide reliable control over the reference for the system under test.

## 4.4. ADC Sampling

An ADC converts the analog signal to digital periodically. Thus, “sampling” the input at a specified period and rate. This allows an output of discrete time and discrete amplitude signal, rather than continuous time and continuous output signal. For the circuit in this project, a 32-bit ADC was used to sample the varying voltage of the voltage divider circuit [9]. This analog voltage signal is then converted to a digital signal and fed to the host microcontroller over SPI.

# 5. Tasks and Costs

The original Gantt chart, detailed in **Figure 25**, outlines which team member is responsible specific aspects of the project. Hunter Phillips was the team member responsible for the bend sensor wiring and programming. The wiring was done on a bread-board until everything was finalized and was then transferred onto a solder board. Hunter wrote the Arduino and Matlab scripts that took data from the Arduino and transferred the data into readable output in Matlab that could be written to Excel. This included a live time graph of the angle being read and an output to excel after a set number of data points.

In the original plan, Jacqueline's main duty was a kinematic analysis and animation of the arm. After discussion with the team this was later deemed unnecessary because the arm was not a four-bar linkage as usually is analyzed in kinematics and a simple protractor could be an accurate method of testing our accuracy, as well as comparing to the known stepper motor value. Because of this change, Jacqueline took more of a role in uncertainty, sensor research, and testing. For uncertainty this included planning out methods that could be used ahead of time, assisting with the code for the live time uncertainty, doing Chauvenet's method on the data and creating plots and graphs with the uncertainty from the data [10].

Darnisha Crane's role as specified in the proposal was to perform uncertainty and calibration. Since Jacqueline took more of a role in uncertainty, Darnisha focused more on testing and calibration. This included extensive tests of the stepper motor and bend sensor as well as taking data and calibrating the device using the method discussed in the calibration section.

Austin Click was responsible for the CAD design, construction and collaborating with Darnisha on calibration. Austin worked with Walter on CAD design and assembly of the device. Austin and Walter designed the mechanical components of the robotic arm and housing for the electrical components. He wrote the document presented during the project update that detailed the method of calibration that we would use.

Walter Deitzler was the machinist for the team. In addition to working with Austin on the CAD model, Walter 3-D printed the robotic arm as well as machining the aluminum plate with the engraved marks every .9 degrees.

The report was split up similarly to the physical tasks. Hunter and Walter were responsible for the Device Production Section which details the mechanical assembly as well as circuitry and code. Hunter and Jacqueline co-wrote theory of operation. Jacqueline was responsible for the Tasks and Costs section as well as collaborating on the Calibration and Uncertainty section. Darnisha assisted on the Calibration and Uncertainty section with Austin as well as writing the lessons learned section. Austin was the compiler of the report. All team members met together to review and revise the report before submission.

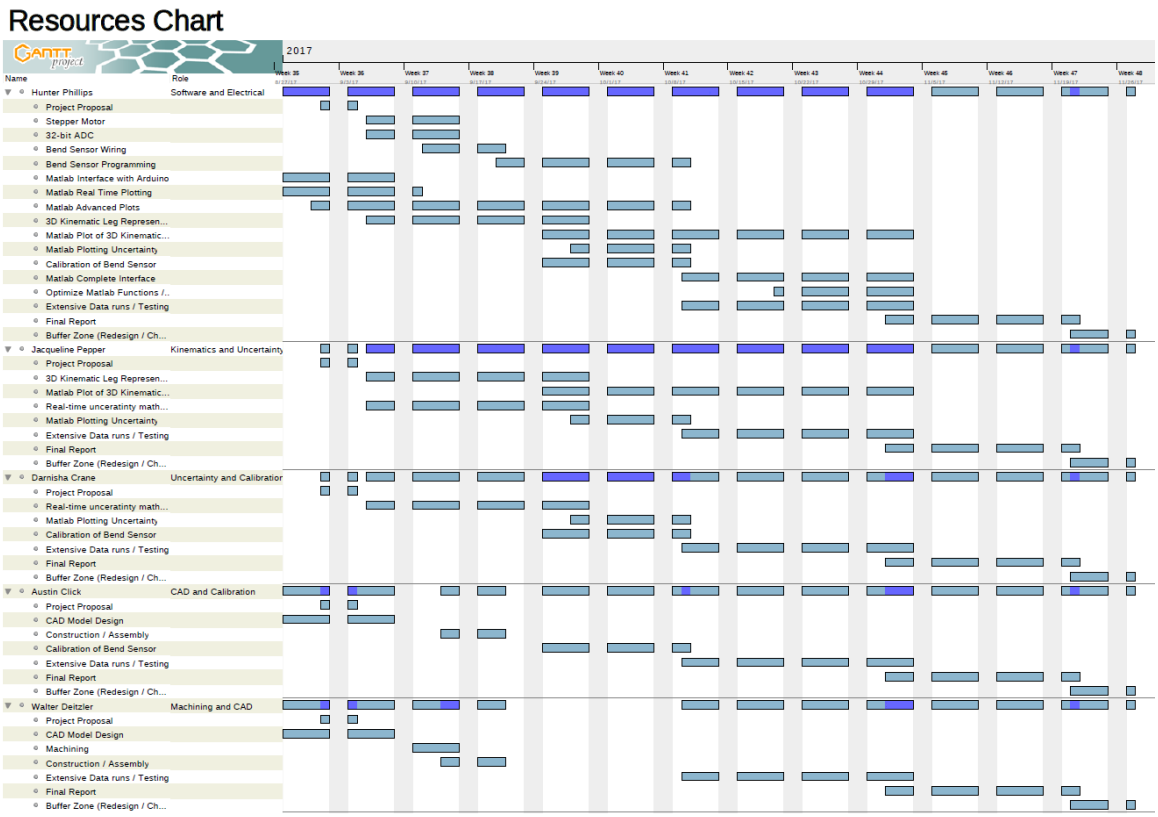



**Figure 25: Time Management and Project Schedule Chart**

The original Gantt chart timeline was disrupted due to a delay machining the device. The arm needed to be 3-D printed and the aluminum plate needed to be engraved on the CNC machine. Both of these require approval to do in the UAH machine shop. The aluminum ended up being machined in the UAH machine shop, but the arm was 3-D printed in the space hardware lab.

The below bill of materials, **Table 1**, lists all materials used for the project. The “Big Easy Driver” from Sparkfun is the driver motor for the stepper motor purchased. The Arduino Uno was purchased from Sparkfun as well. The 32-bit ADC was purchased from Protocentral. The Precision Voltage Reference, Perma-Proto Half-sized Breadboard, and SOT-23 Breakout Set were all purchased from Adafruit. The plexiglass sheets used belonged to a team member. The aluminum sheets used to create the housing were purchased from McMaster Carr. The bill of materials listed in the proposal did not change throughout the project.

## Table 1: Bill of Materials

| Part Name | Price | Link |
|---|---|---|
| Big Easy Driver | $19.99 | https://www.sparkfun.com/products/12859 |
| Stepper Motor (400 steps / rev) | $16.95 | https://www.sparkfun.com/products/10846 |
| Arduino Uno R3 | $24.95 | https://www.sparkfun.com/products/11021 |
| ProtoCentral 32-bit ADC | $50.94 | https://www.protocentral.com/analog-adc-boards/1005-protocentral-ads1262-32-bit-precision-adc-breakout-board-0642078949630.html |
| Precision LM4040 Voltage Reference | $7.50 | https://www.adafruit.com/product/2200 |
| RGB LED Strip - 1 meter | $9.95 | https://www.sparkfun.com/products/12021 |
| FlexPoint Bend Sensor | $12.20 | http://www.flexpoint.com/product/bend-sensor/ |
| Adafruit Perma-Proto Half-sized Breadboard | $4.50 | https://www.adafruit.com/product/1609 |
| SOT-23 Breakout Set | $4.95 | https://www.adafruit.com/product/1230 |
| 1/2" x 12" x 12" Plexiglass Sheet | | |
| 1/8" x 12" x 24" Plexiglass Sheet | | |
| 1/4" x 8" x 8" Aluminum Sheet - 86825K724 | $22.32 | https://www.mcmaster.com/#standard-aluminum-sheets/=195n9so |
| 1" x 1" x 18" Aluminum 6061 - 6546K21 | | https://www.mcmaster.com/#standard-aluminum-hollow-tubing/=195naau |

# 6. Calibration

The process of calibrating the bend sensor directly impacts the accuracy of the measurements recorded. To generate a curve that the MATLAB script could follow and generate points from, a defined method of calibration had to be set forth. Calibration went as follows.

1. A MATLAB script to increment the stepper motor by a predefined accurate amount had to be written. This was done by Hunter Phillips and Austin Click.
2. The exact steps of the motor were calculated to determine the degree increment per step. The data sheet for the motor states there are 400 steps per revolution. **Equation 2** shows this calculation.

$$increment = \frac{360}{steps} = \frac{360}{400} = .9\,\frac{degrees}{step} \quad \textbf{(2)}$$

3. When this was done, an exact increment was known. Because the bend sensor changes resistance with each increment, the Arduino reads a different resistance at each increment. This resistance is then associated with a specific degree increment.
4. Resistance values for each increment were recorded from MATLAB directly to a CSV file. For the testing increment of 90 degrees, 100 points were recorded per step. To generate a valid, accurate curve, this was repeated enough times to generate 5 quality data sets.
5. Each CSV was then imported into Microsoft Excel to generate a mean and reject outliers, if there were any. Using Excel's standard deviation function, average function, and Chauvenet's Criterion, data was reduced [10].
6. The mean and standard deviation for each increment are given in the Appendix, and the full calibration document is given in the Online Appendix.
7. The mean after rejecting outliers for each point was used to plot the curve that was necessary for MATLAB functionality. Angles could also be extrapolated through the two nearest points, as the high resolution of the curve gives the highest possible accuracy available through the Arduino.
8. The mode of each data set was also considered as the calibration was done. For example, if a specific value had many points that were the same, but one outlier, the mode was taken instead of the mean, as this was thought to be a more solid comparison (and could even be the mean without outliers). This method was design as a backup plan and was not used.

The calibration curve is shown in **Figure 26**. Also plotted is the line of best fit, and the $R^2$ value. Individual data sets can be found in the Appendix, Section 10.2., figures

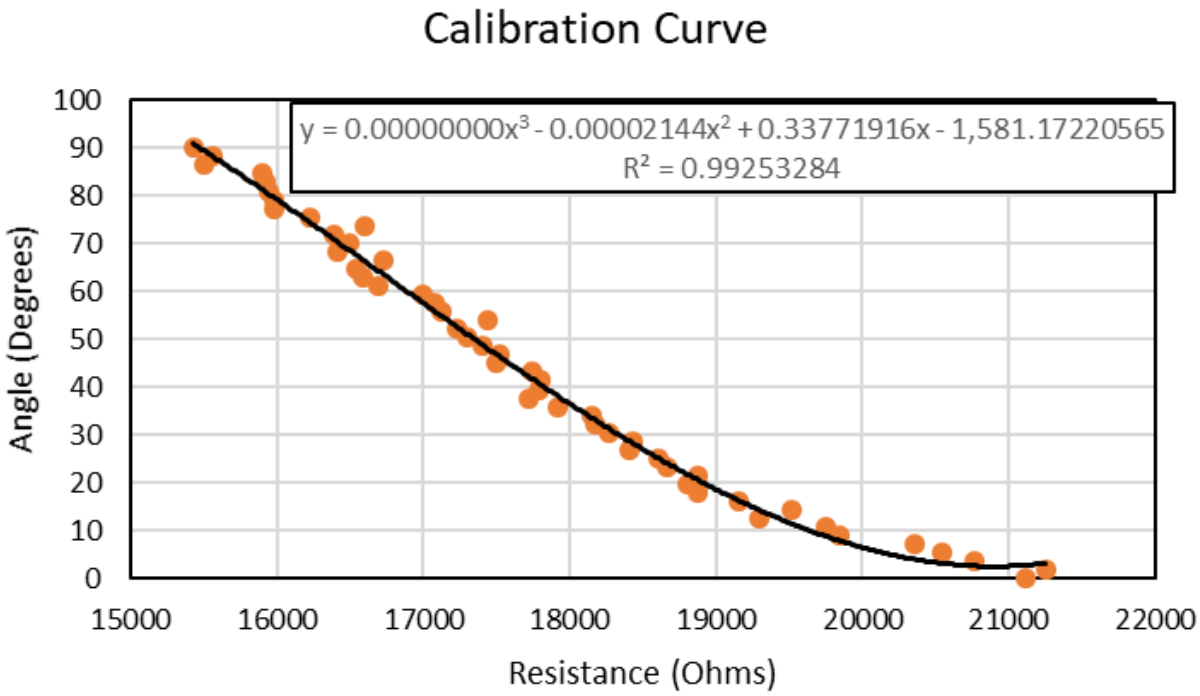



**Figure 26: Calibration Curve**

# 7. Uncertainty Analysis

Once the device had been calibrated measurements were taken every 10 degrees and compared to a known stepper motor angle. At each angle, 100 data points were taken. In order to evaluate whether Chauvenet's should be applied, all the data was plotted.

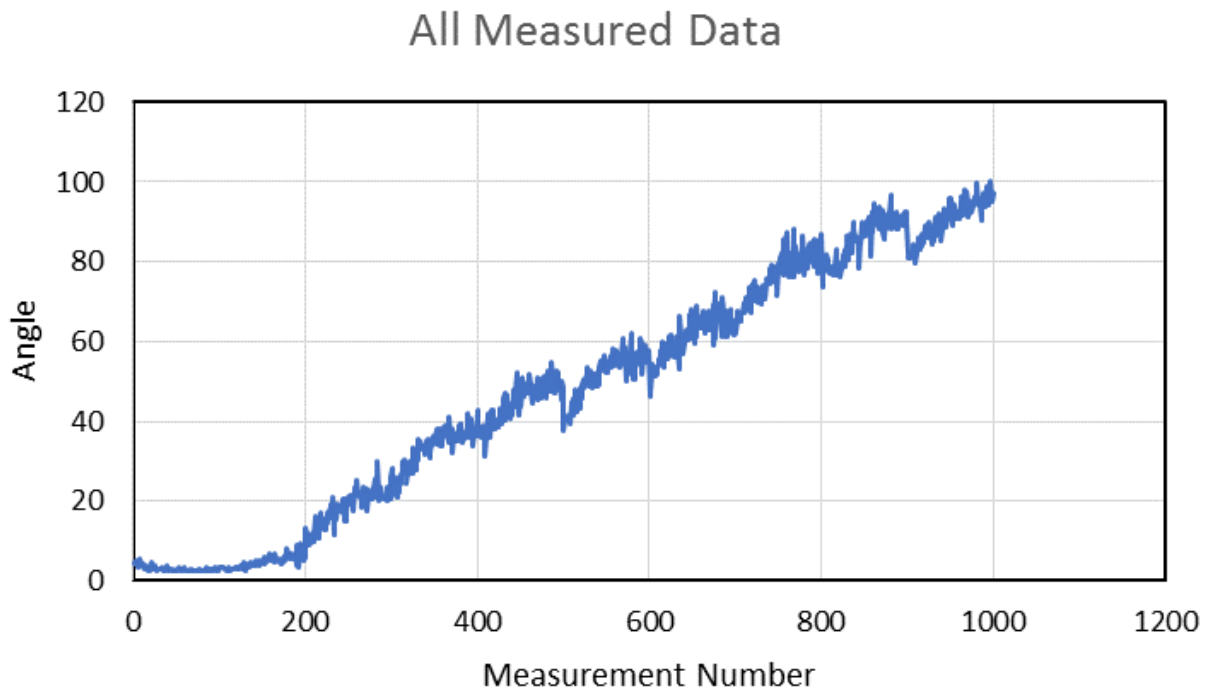



**Figure 27: Raw Data**

**Figure 27** displays the raw data from the first set of data taken after calibration. Having known the device to have random spikes in the past, and seeing that there could be some outlying data points, Chauvenet's criteria was used, but no outliers were found. Raw data can be found in **Appendix section 10.4.**

The mean and standard deviation were taken of each set of 100 measurements corresponding to a specific stepper motor, shown in **Table 2**, on the next page. Raw data that was used to create this table is included in **Appendix section 10.4.**

**Table 2: Angle and Error**

| Reference Angle | Mean Measured Angle | STDEV | Difference | %Error |
|---|---|---|---|---|
| 0 | 3.066103377 | 0.615854059 | 3.066103377 | - |
| 10 | 4.721091445 | 1.442561212 | 5.278908555 | 52.78909 |
| 20 | 18.51655611 | 4.402685349 | 1.483443887 | 7.417219 |
| 30 | 33.29856606 | 4.988654716 | 3.298566056 | 10.99522 |
| 40 | 44.88910136 | 4.948121998 | 4.889101359 | 12.22275 |
| 50 | 51.87002092 | 5.51185322 | 1.870020922 | 3.740042 |
| 60 | 61.26489725 | 5.316895737 | 1.264897248 | 2.108162 |
| 70 | 76.43552672 | 5.998677265 | 6.435526724 | 9.19361 |
| 80 | 85.90880394 | 5.335016187 | 5.908803939 | 7.386005 |
| 90 | 90.69518799 | 5.061896318 | 0.695187994 | 0.772431 |

This table was used to create several plots to visualize the accuracy of the measurements. **Figure 28,** below, shows the reference angle plot with a trendline, as well as the measured angle.

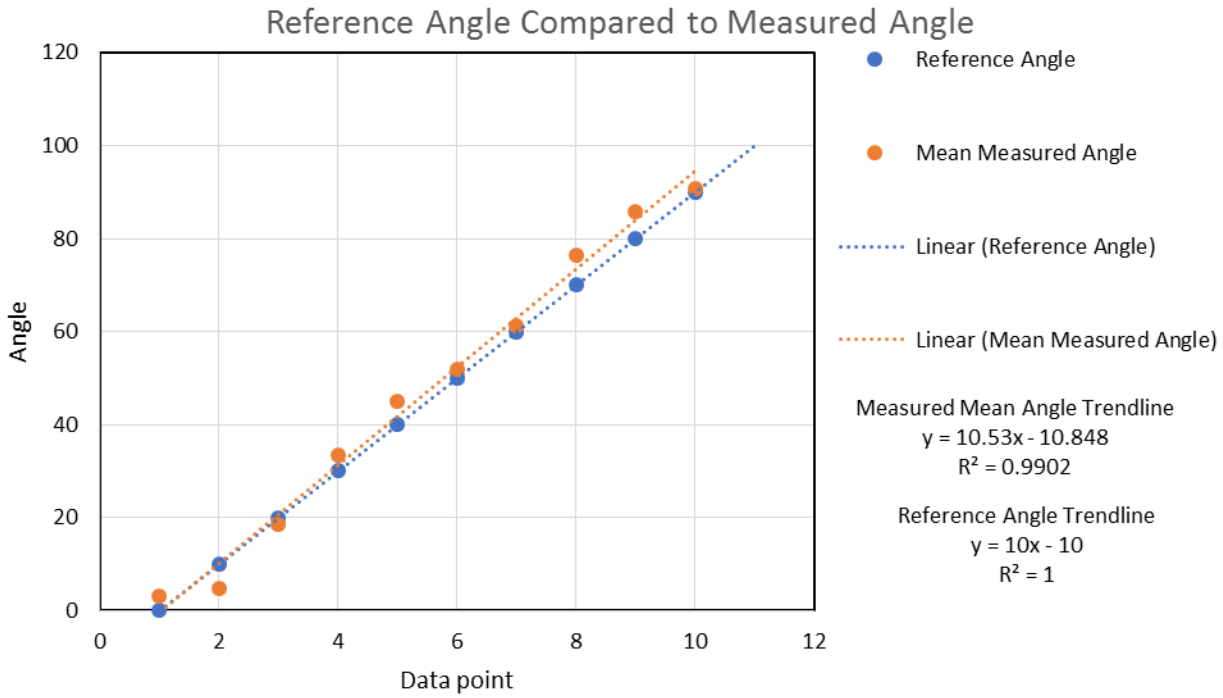



**Figure 28: Measured Angle vs Reference Angle**

**Figure 29** shows the percent error plotted against measurement number.


**Figure 29: Measurement Number vs Percent Error**

**Figure 30** similarly shows standard deviation.

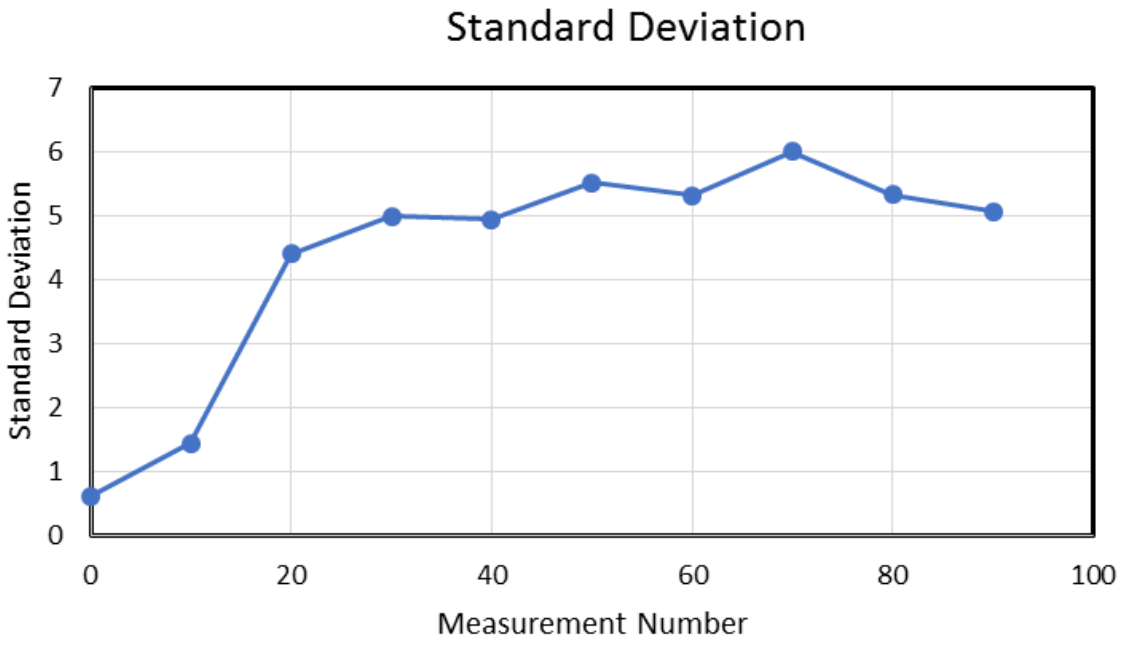



**Figure 30: Measurement Number vs Standard Deviation**

In order to show the accuracy of the measurements relative to the known stepper motor angle, the measured angle was plotted against the known angle. In an ideal scenario, this would be a linear relationship. It can be noted that compared to the trendline the relationship is almost linear. A perfectly accurate sensor would be perfectly linear with an intercept of zero and a slope of one (1). **Figure 31** shows the measured angle compared with the reference angle, and also includes the slope of reference versus measured angle.

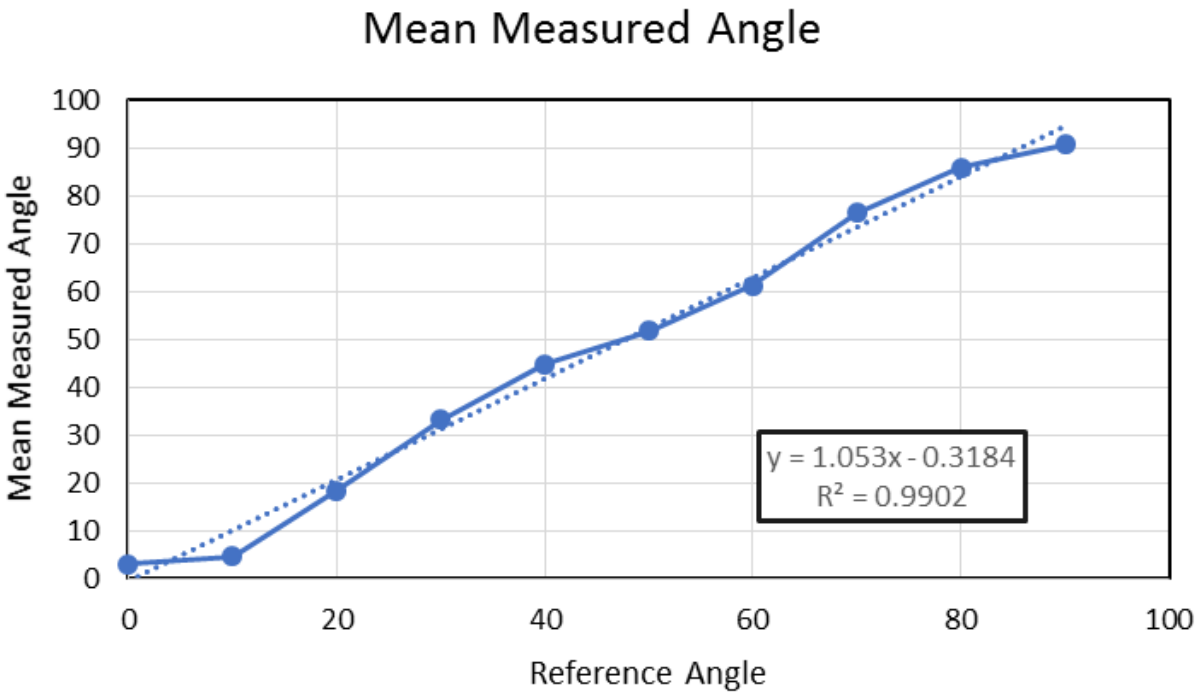



**Figure 31: Reference vs Measured Angle**

The factors to uncertainty for our device are the stepper motor, sensor, 32-bit ADC, and the voltage reference. The stepper motor has an uncertainty of .05 steps. The largest contributor to the uncertainty is the sensor with an uncertainty of 25%. The ADC has an uncertainty of 78 mV and the voltage reference has uncertainty of .1V. In order to find the total uncertainty, the square root of uncertainty of each component is divided by the value for that component squared as shown in **Equation 3**.

$$u = \sqrt{\left(\frac{u_{stepper}}{stepper}\right)^2 + \left(\frac{u_{sensor}}{sensor}\right)^2 + \left(\frac{u_{ADC}}{ADC}\right)^2 + \left(\frac{u_{volt}}{volt}\right)^2} \quad \textbf{(3)}$$

$u_{stepper}$ = uncertainty of stepper motor
$stepper$ = reference value for stepper motor
$u_{sensor}$ =uncertainty of flex sensor
$sensor$ = reference value for flex sensor
$u_{ADC}$ = uncertainty of 32-bit ADC
$ADC$ = reference value for 32-bit ADC
$u_{volt}$ = uncertainty of voltage reference
$volt$ = reference value for voltage reference

$$u = \sqrt{\left(\frac{.05}{.9}\right)^2 + \left(\frac{.25}{140000}\right)^2 + \left(\frac{.078}{4.096}\right)^2 + \left(\frac{.1}{4.096}\right)^2}$$

$$u = .063601$$

$$u = 6.36\%$$

After inputting the numbers, the total uncertainty comes to 6.36%, which is comparable 5% error in the data taken for uncertainty analysis.

# 8. Lessons Learned

## 8.1. Stepper Motor Inconsistent Step Advances

When the stepper motor was being tested, it appeared that it would not turn reliably unless a small amount of pressure was applied to the pointer arm. All the different micro-stepping modes were tried on the stepper motor driver board to alleviate the issue, although none worked. The possibility of not enough current being applied to the coils in the stepper motor was taken into consideration although was quickly passed on since the power supply is 3A which is almost double what the stepper motor requires for full output operation. A final solution was tried through loading more weight on the lever arm and it worked. After further research, it was discovered that sometimes not enough weight on a stepper motor cannot provide enough inertial energy with the coil being electrified and thus won't advance a position. Although this proved to be a disadvantage as it made the pointer arm slight bend downwards ever so slightly. This could cause a variance in the measurement readings from cross-directional bending occurs spiking the resistance. It was decided to just apply the small amount of pressure manually during the calibration and uncertainty analysis to ensure that all data taken as accurate. If the pointer arm was built in the future it would be equipped with more mass so that the stepper motor could function to its full potential.

## 8.2. Op-Amp Circuit Redesign

The use of a simple voltage divider circuit was the best way the project team knew how to measure a varying resistance into an ADC at the beginning of the semester. Throughout the labs it was hypothesized that if an op-amp was used to amplify the voltage signal a wider range of measurements could be obtained. This would be incredibly helpful as the ADC could have a wider range of voltages to sample. The Flexpoint Electrical Datasheet, found in the **Appendix**, actually lists an op-amp circuit as the recommended design, however at the beginning of the semester, the students were unable to fully understand how the circuit worked and thus avoided it by completing a simpler design.

## 8.3. ADS1262 32-Bit ADC

The high accuracy and precision provided through the use of a 32-bit ADC seemed promising at the time the project started. After programming with the included libraries, it was quickly realized that the company Protocentral, an Indian company, had minimal tech support and resources available for the user. It was a steep learning curve to fully understand the scope of their libraries and example test script. If the team could go back, they most likely would have chosen a much easier to use 16-bit ADC from a company with great resources such as Adafruit. Although the challenges associated with understanding the vaguely written C++ header files enabled the students to learn a lot about underlying code design. Once implemented and the correct values were being read across SPI by the Arduino, it made for a satisfying experience with the countless hours of work being spent on deciphering the code.

## 8.4. Linear Interpolation versus a Proper Calibration Curve

It was originally noted that in the datasheet, Flexpoint provides a curve of expected output across a certain bending radius, and it appeared to be a cubic curve across initial inspection. It was assumed that for initial testing a simple linear interpretation between a single obtained data point at 0 and 90 degrees would provide a measurement to a pretty fair degree of accuracy. The students assumed that although they needed to calibrate the device, it wouldn't provide a significant advantage over a simple solution. This assumption was wrong, and the calibrated cubic curve was significantly better across the broad spectrum. It is noted that around the weak point of the linear interpolation, 40 to 60 degrees, the curve exceled. This is a great example of a big lesson learned with a future impact in the student's careers.

## 8.5. Accuracy and Precision of Components

The team originally planned to go with the highest precision components available to ensure the best measurement was obtained. After the project update it was brought to the project team's attention that it is not all about precision since you are only referencing the accuracy of the components. This enabled the avoidance of purchasing unneeded components such as a high precision resistor for the voltage divider circuit. A few high precision components were still purchased such as the voltage reference, since it was incredibly reliable and consistent. Since the accuracy is less important as long as it is consistently at that value. To ensure the values were accurate, they were measured with a high quality Keysight Technologies U1241B. This showed that it is not always the most expensive or cutting-edge components that enable a successful project, it is how the team uses the components at their disposal to the highest potential.

## 8.6. Ground Loop Issues

When the circuit was first wired each component worked when plugged in individually through its voltage rail, although when the 12V and 14V sources were both plugged in an unexpected issue occurred. When the stepper motor was commanded to do a step the LEDs would get slightly brighter. This consistently happened and sometimes they would even cycle different colors depending on the stepping mode chosen for the stepper motor driver. After a lot of trial and frustrating error it was discovered that a small portion of the LED strip was touching the aluminum plate. A small bit of adhesive had contracted and allowed the contact of one of the strips ground pads to the surface. A multimeter was then used to figure out how the short was being tied to the stepper motor. It was discovered that the casing of the stepper motor was somehow internally connected to one of the coils in the stepper motor. Since the metal stepper motor housing was bolted with stainless steel bolts to the aluminum top plate it caused a connection between the LEDs and stepper motor. It is hypothesized that when the coil was electrified in the stepper motor that the short caused a small amount to travel to the LEDs and potential electrical energy was carried across the connection. Resulting in each and every time the stepper motor

was activated causing the potential electrical energy to increase, causing the brightness and colors to vary according to how the coils were being energized. It can sometimes be the smallest of errors in assembling that prove to provide countless hours of frustration, thus showing the team that attention to detail in assembly is critical to successful testing.

## 8.7. Digital Signal Processing (DSP)

It was noted that the measurements appeared to have a large bound around the origin when testing. This was not seen during initial research and the measurements in commercial products appeared to be fairly stable. After further investigation, it was assumed that the commercial products use some form of DSP to smooth out the oscillating measurements. This is something the team did not have the necessary amount of time to properly implement, however would greatly improve the measurement precision and accuracy. Although with more time a live time DSP algorithm could be used to filter out the wide bounding measurements and obtain the mean origin point. If this could be successful completed the live-time readings could be significantly more impactful in robotic applications. This taught the students the importance of filtering data and understanding the nuances of your measurement and how to improve the reading with filtering techniques.

## 8.8. Graphing Real-Time Updating in Matlab

When measurement data is being fed into Matlab it is critical that the graph visually shows the result as fast as possible. When the functionality was first being explored, the team could not figure out why the graph in Matlab would not update. Delays were added in the loop until a 1 second delay was the only way to get Matlab to plot the measurement. To bypass this issue the animatedline functionality was discovered to use different graphics libraries, however ended up being unusable since the team desired a compass type graph. After many attempts, it was discovered that OpenGL can force Matlab to update graphics using the *drawnow update* command. This was not very well document, however made it possible for the team to real-time plot data in a visually appealing compass plot.

## 8.9. 3D Kinematic Arm

In the original proposal, a 3D kinematic arm rendering in Matlab was explored. Initial testing was completed on the feasibility and it was determined an unnecessary portion of the experiment and that the team's efforts would be better utilized in other areas. This was due to the fact that Matlab does not have a simple interface to render an optimized 3D design. Especially with the conditions the team was placing the graphics processing unit (GPU) under already with the live-time plotting. It was noted that in the future the use of Python and libraries such as matplotlib and numpy combined with Blender would provide beautiful graphics for this kind of design. Preliminary testing also showed that one of the team member's computers, Hunter Phillips, would be able to handle the graphics load with its Nvidia Quadro M4000 graphics card. Although this was not enough to

necessitate the hours being put into it, since the desktop tower could not be transported to UAH for the demo.

## 8.10. Live-Time Uncertainty

The original proposal also projected the idea of a live-time uncertainty system to show the user the confidence in their measurements being obtained. This would be implemented in a two-prong approach. The user would be able to see a live-time graph with an uncertainty estimation, approximate error, and theoretical error plot. The functionality was actually built in as seen below in **Figure 32**.

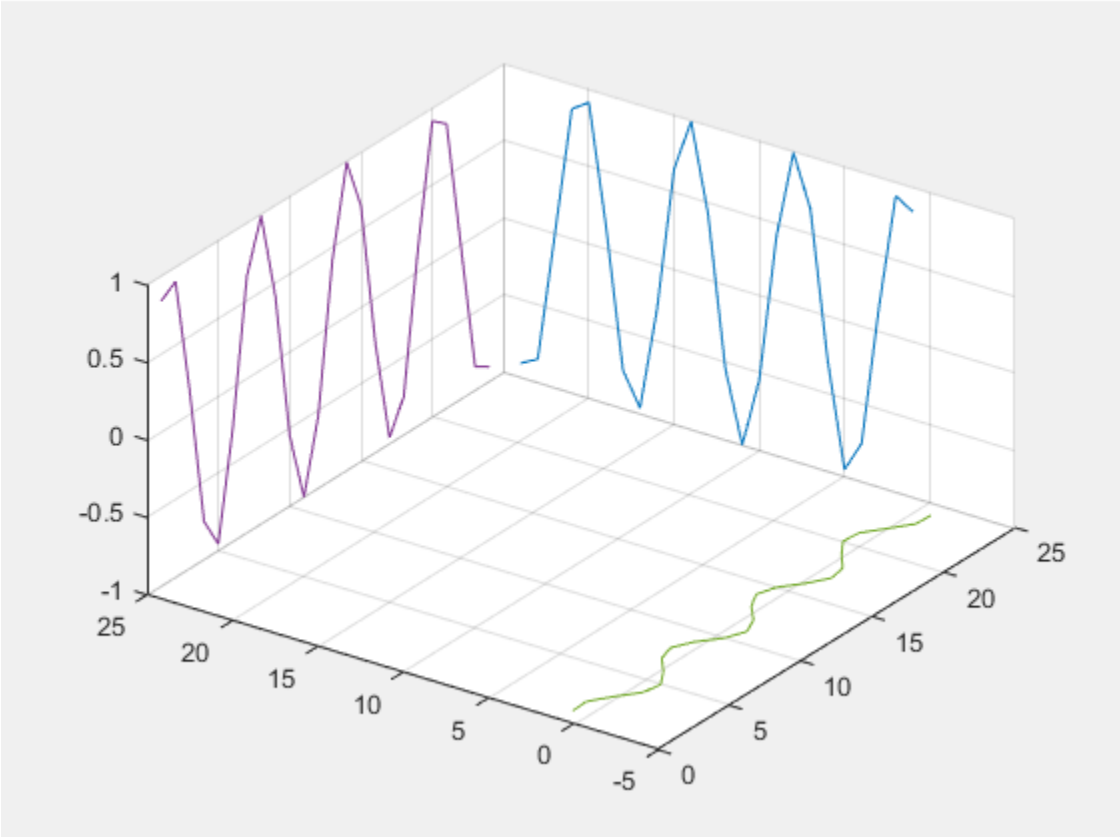



**Figure 32: Live Time Uncertainty Plot**

The code was never implemented due to the live-time uncertainty code never being reached due to timeline restrictions. Although it required a significant amount of time due to the fact that all 3-axis of the 3D plot had to be filled in real-time ensuring that they were all updated at the same time. This was obtained through the use of axis shifting techniques and array indexing of graphical elements. Purging of old elements in the array was also necessary to ensure that lag wasn't introduced due to the sheer number of points needing to be plotted in the 3D graph.

The LED portion of the uncertainty analysis also suffered the same fate of not having an uncertainty measurements to display in real-time due to time constraints preventing

the code from being written. Although the gradient fading function from green to red was written and can be seen in action below in **Figure 33**.

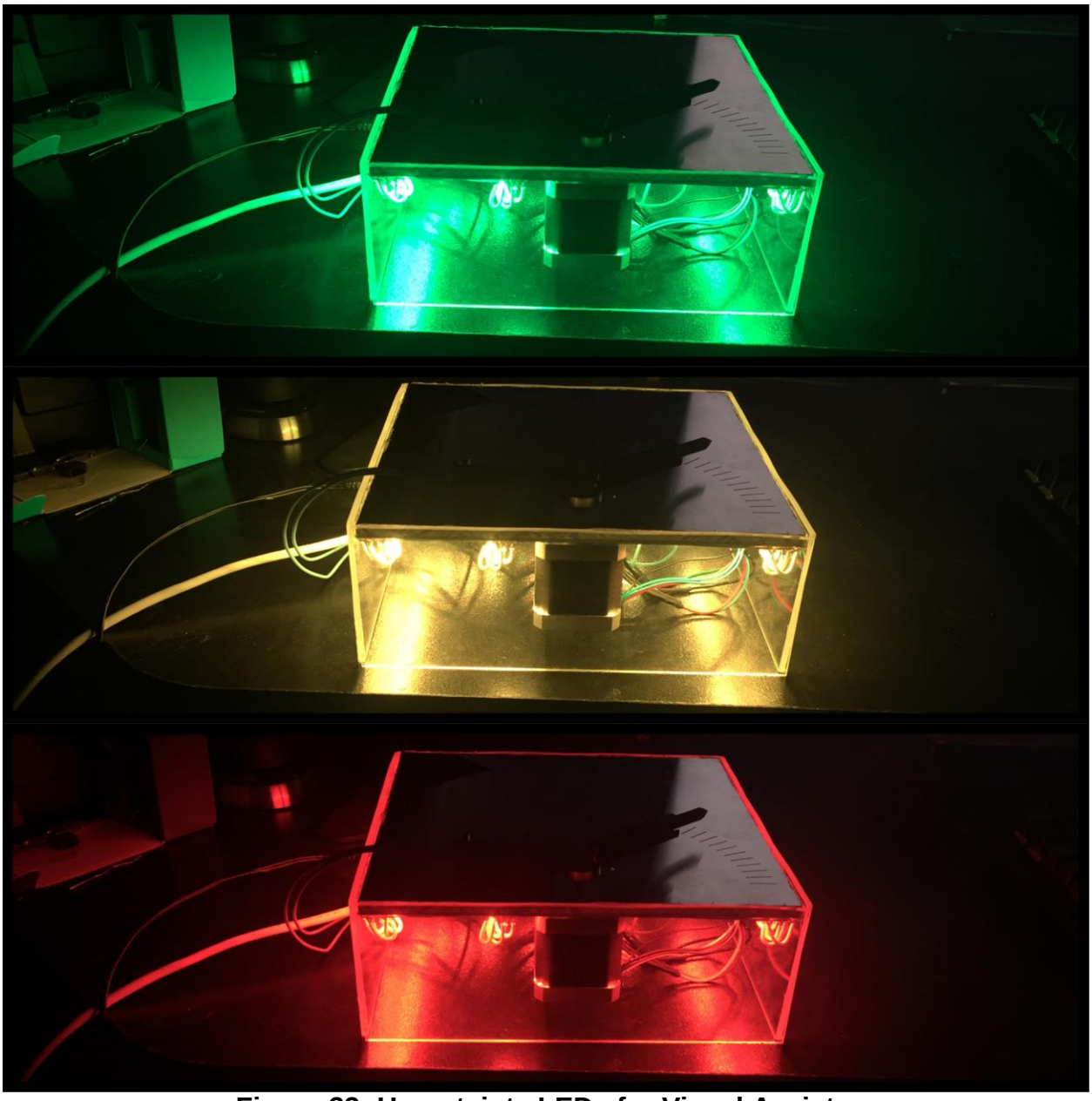


**Figure 33: Uncertainty LEDs for Visual Assistance**

The original intention of giving the user subliminal visual feedback through the use of LEDs gives the user the ability to notice changes without having to look at the graphs. Understanding the significance of the uncertainty values in the graphs could be hard to visualize, but the use of LEDS provides a simpler approach, since the transition from green to red signifies the measurement is losing confidence in the uncertainty and something might be going wrong with the system.

## 8.11. Demo Computer Validation

All of the testing on the robotic joint system was completed at Hunter Phillips house on his desktop computer with Windows 10. It was assumed that some initial debugging and testing would have to occur for the demo so the team arrived early on demo day to ensure it would work on another computer. This proved to be a smart decision as the rest of the day was spent debugging tons of issues. Trying to use the 311 labs computers proved to be near impossible since they would not compile the Arduino libraries due to permission errors. Along with a lot of other admin issues on the computers it was decided to move to a couple of team members laptops. Both laptops were missing necessary coding background packages and directory functionality. It also appeared that certain COM functionality was broken with the Arduino as every time the Arduino was unplugged and replugged in, it would no longer transmit data across the TX line. This was later discovered to be issues with the Windows binaries with Hunter's desktop computer having serial libraries already pre-installed from other projects and Arduino automatically detecting them and working. After all these issues were faced it was decided to attempt a rewrite of the code to work on an Apple Macintosh (Mac) computer. It was known that the Excel writing functions would not work on a Mac computer, so this issue was tackled first. With comprehensive research, it was determined that a new xlswrite function needed to be used and the following java libraries added as seen in **Figure 34**.

```
% Setup Excel X Server Apache Libraries
delete(instrfind)
javaaddpath('poi_library/poi-3.8-20120326.jar');
javaaddpath('poi_library/poi-ooxml-3.8-20120326.jar');
javaaddpath('poi_library/poi-ooxml-schemas-3.8-20120326.jar');
javaaddpath('poi_library/xmlbeans-2.3.0.jar');
javaaddpath('poi_library/dom4j-1.6.1.jar');
javaaddpath('poi_library/stax-api-1.0.1.jar');
```

**Figure 34: Modifications for Excel Code**

After the programming rewrite had been completed the only downside that was noted was the lack of a dedicated graphics unit in the Mac causing 100-200 millisecond delays in the graphing. This huge challenge reminded the team that testing across a large system of computers before demo day is important as different hardware can cause anomalies to appear hindering a smooth testing day.

## 8.12. Transmitting Floats Across Serial

When the 32-bit ADC was purchased the thought of getting the high precision float measurements to Matlab did not seem to cross the minds of the team. It was quickly realized that the test was not trivial since Arduino did not support this Serial writing functionality. The thought to parse the float into its representative 4-byte chain came after remembering that the IEEE standard can send these shifted bytes as chars across the line encoded. It took forever to realize that a delimiter was needed to prepend the chain for Matlab to understand the following data being parsed. On the Matlab side of code a variety of functions were called to decode the chain and concatenate it back into the original float value. An unusual issue arose, that the byte chain appeared backwards and shifted 2 bits over. To figure out this issue, a known float value was sent across, its hex value being known was then used to figure out where the char values were and how to rearrange them properly. Once it had been completed for one value, the decoding process worked for all incoming encoded float values. This was a huge roadblock in the project progress as it had not been expected initially. Although it taught the team a lot about bit and byte level programming and how to secure data across serial lines between different electronic systems.

# 9. References


1. Bhat Venkatesh, Protocentral_ads1262 (2016), GitHub repository, https://github.com/Protocentral/ProtoCentral_ads1262

2. "Flexpoint Awarded Another Bend Sensor Patent." Sensors. Questex LLC, 12 Nov. 2007. Web. 27 Nov. 2017.

3. Giovanni Saggio, Antonino Lagati, and Giancarlo Orengo, “Shaping Resistive Bend Sensors to Enhance Readout Linearity,” ISRN Electronics, vol. 2012, Article ID 359759, 7 pages, 2012.

4. Mower, Clark. "Flexpoint Sensor Systems Announces Launch of the "one of a Kind" Bend Sensor® Glove Kit." Cision. PR Newswire, 1 Nov. 2016. Web. 27 Nov. 2017.

5. Yen, Charles Low Khai, and Christian Kasztelan. "Recommendations for Screw Tightening Torque for IGBT Discrete Devices." Infineon. N.p., Dec. 2013. Web. 20 Nov. 2017.

6. Quan, Wei, Hua Wang, and Datong Liu. "A Multifunctional Joint Angle Sensor with Measurement Adaptability." Sensors 13 (2013): 15274-5289. Web. 24 Nov. 2017.

7. "How to Prevent Step Losses with Stepper Motors." MICROMO. Faulhaber Group, n.d. Web. 24 Nov. 2017.

8. "DMOS Microstepping Driver with Translator And Overcurrent Protection." Allegro MicroSystems. N.p., 2013. Web. 24 Nov. 2017.

9. Oliveira, Maurício De. "Sampling: DAC and ADC Conversion." University of California San Diego. N.p., n.d. Web. 24 Nov. 2017.

10. Armentrout, D., “MAE 311L Lab 3: Uncertainty Analysis,” Lab Manual, MAE Dept., Univ. Alabama in Huntsville, 2015.

# 10. Appendix

## 10.1. Bend Sensor Mechanical Data Sheets


**Bend Sensor® Technology**
**Mechanical Application**
**Design Guide**




# Bend Sensor® Mechanical Application Guide

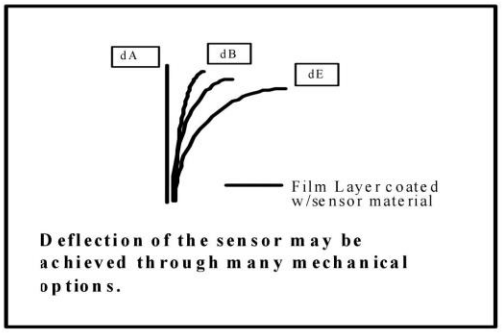
## Introduction

This Application Guide is designed to help mechanical designers build interfaces that result in the successful integration of Bend Sensor® components into products. Flexpoint Sensor Systems (Flexpoint) has successfully developed and marketed products incorporating Bend Sensor® technology. As described below the Bend Sensor® device is typically a film – the Bend Sensor® film is put into a system and essentially "goes along for the ride". The film is attached to a system and measures the movement of the system. Although the sensor itself is capable of being cycled millions of times, the system it is applied to must also be robust. Below is a description of the sensor, and many of the considerations that should be understood in the design of the system.

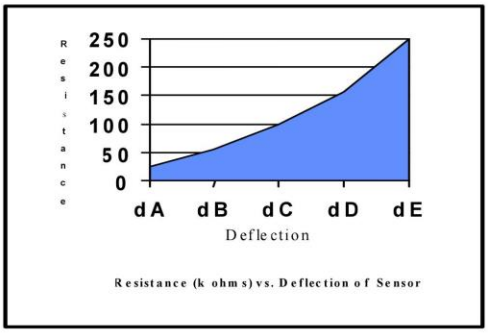
## Description of the Bend Sensor® Device

The Bend Sensor® potentiometer is a product consisting of a plastic substrate coated with a resistive material that changes in electrical conductivity as it is bent. Electronic systems can connect to the sensor and measure with fine detail the amount of bending or movement that occurs. Depending on the configuration and integration of the sensor resistance changes of greater than 10 times can be realized. Refer to the Bend Sensor® Electronic Design Guide (http://www.flexpoint.com/wp-content/uploads/2015/05/electronicDesignGuide.pdf) for a description of electronic interfaces.

An example application is one where a sensor is attached to a hinge. As the door is opened, among other things one can measure how far the door is opened and how fast it is moving. The sensor is lightweight, small, easily packaged and very reliable. The breadth of the application for the Bend Sensor® device is limited only by the customer's imagination.



Deflection of the sensor may be achieved through many mechanical options.



Resistance (k ohms) vs. Deflection of Sensor

The rate of change as displayed in the graph is dependent on two factors. The first is the bend radius, the smaller the radius the greater the change. Secondly is the angular deflection. For a given radius the more the sensor is bent around it the greater the change.

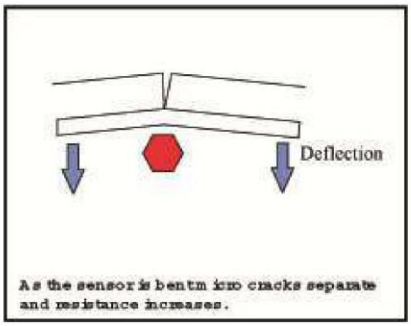
## How the Bend Sensor® Potentiometer Works

Flexpoint's patented Bend Sensor® device consists of a single layer, thin, (0.005" typ.) flexible piece of material that is coated with a proprietary carbon/polymer based ink. This type of resistive element is commonly used to make thick film resistors, resistor networks, slide potentiometers and transducers. Flexpoint's proprietary inks are primarily printed on thin plastic films.

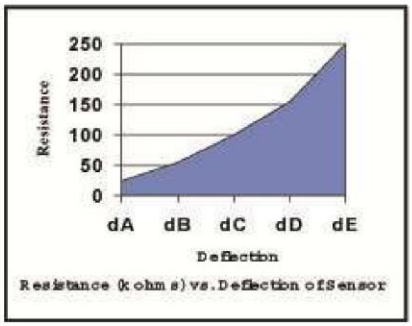
Flexpoint uses a coating that when printed on a plastic film such as polyimide forms a bond that is very strong. The ink is very hard and brittle. During processing micro cracks are introduced into the carbon/polymer coating, when the coated film is bent, the ink's many micro cracks, which upon movement open and close to the specific bending of the material. The ink maintains its integrity in shape and continues to have a very strong bond to the substrate. One of the most unusual characteristics of this ink occurs when the material is bent over a given radius repeatedly. It returns to its original value within the same slope or curve of resistance proportional to displacement.
The sensors are easily customized as the coatings can be applied in many patterns to adjust the resistance and can be screen printed as required by the application. The tremendous reliability, consistency and repeatability of the Bend Sensor® device make it highly desirable to many industries. Other technologies such as force sensors do not display and deliver the reliability and versatility of the Bend Sensor® potentiometer.

As mentioned the Bend Sensor® is a single layer construction. As such dirt, dust and other particulates that cause issue with multiple layer sensors do not affect the Bend Sensor®. The same is true for many liquids including gasoline, diesel fuel, motor oil, antifreeze, alcohol and water to name a few.
Since the coating can be placed on several types of substrates, the sensor may be designed to act as its own spring allowing the sensor to return to its original position.

The proprietary inks are custom manufactured by Flexpoint and can be easily adjusted to the specific resistance requirements of the customer. Flexpoint can design the method of actuation or bending to fit the measurement needs. New inks and products are constantly being developed.



As the sensor is bentm icro cracks separate and resistance increases.



Resistance (kohms) vs. Deflection of Sensor

## 10.2. Bend Sensor Electrical Data Sheets


# Bend Sensor® Technology
# Electronic Interface
# Design Guide

## Introduction

This Design Guide is designed to help electrical designers build interfaces that result in the successful integration of Bend Sensor® components into products. Flexpoint Inc. has successfully developed and marketed products incorporating Bend Sensor® products. Most successful Bend Sensor® interfaces start with the same building blocks. This Design Guide comprises a collection of circuits that effectively empower the designer to modify and build customized circuits that complement their product.

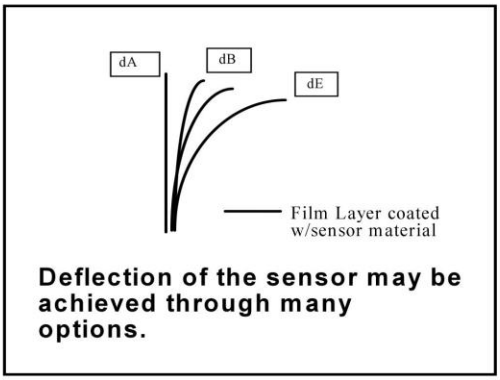
## Bend Sensor® Potentiometer

In the late 1980's, a new product was being developed to be a thin (< 0.005" typ.) light weight potentiometer which, when integrated into a glove could provide analog feedback showing finger movement. After substantial development efforts a new product utilizing a bend sensitive ink was created. This new product known as the Bend Sensor® potentiometer was integrated for use in a consumer product known as the Nintendo Power Glove. Many companies have since recognized the simplicity and reliability of the sensor making it an ideal solution which very effectively addresses many issues and challenges associated with standard potentiometers.

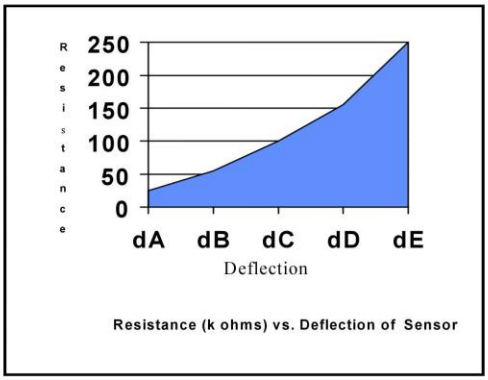
## Description of the Bend Sensor® Potentiometer

The Bend Sensor® potentiometer is a product consisting of a coated substrate such as plastic that changes in electrical conductivity as it is bent. Electronic systems can connect to the sensor and measure with fine detail the amount of bending or movement that occurs. A movement of only one inch can yield over 200,000 data points.

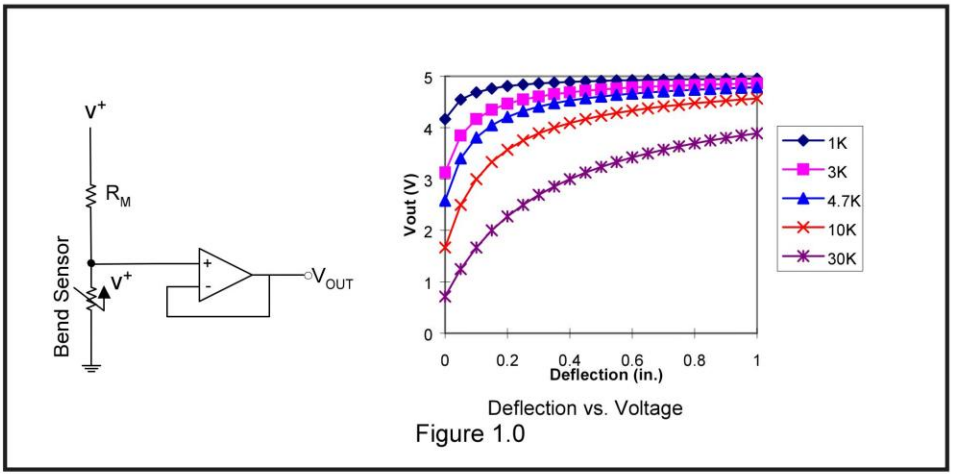
An example application is one where a sensor is attached to a door. As the door is opened one can measure how far the door has opened and how fast it is moving. The sensor is light weight, small, easily packaged and very reliable. The breadth of the applications for the Bend Sensor® is limited only by the customer's imagination.




**Deflection of the sensor may be achieved through many options.**




**Resistance (k ohms) vs. Deflection of Sensor**



Deflection vs. Voltage

Figure 1.0

**Bend Sensor® Voltage Divider:**

For a simple deflection-to-voltage conversion, the Bend Sensor® device is tied to a resistor $R_M$ in a voltage divider configuration. The output is described by the equation:

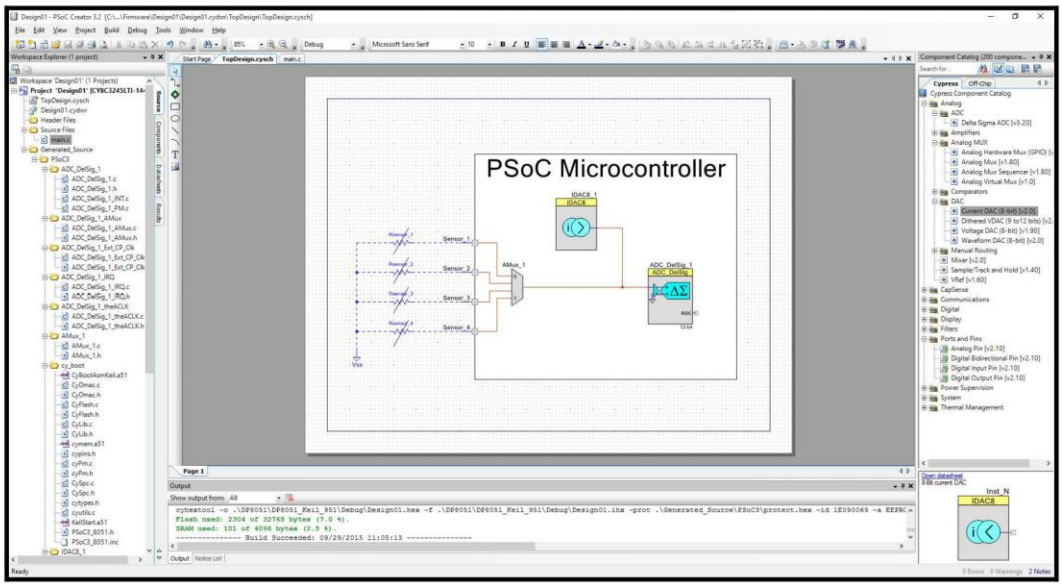
$$V_{OUT} = (V+) / [1+ R_M / R_{Bend\ Sensor®}]$$

In the shown configuration, the output voltage increases with increasing deflection. If $R_{Bend\ Sensor®}$ and $R_M$ are swapped, the output swing will decrease with increasing deflection. These two output forms are mirror images about the line $V_{OUT} = (V+) / 2$.

The measuring resistor, $R_M$, is chosen to maximize the desired deflection sensitivity range and to limit current. Suggested op-amps for single sided supply designs are LM358 and LM324. FET input devices such as LF355 and TL082 are also good. The low bias currents of these op-amps reduce the error due to the source impedance of the voltage divider.

A family of DEFLECTION vs. $V_{OUT}$ curves is shown in Figure 1.0 for a standard Bend Sensor® device in a voltage divider configuration with various $R_M$ resistors. A (V+) of +5V was used for these examples.

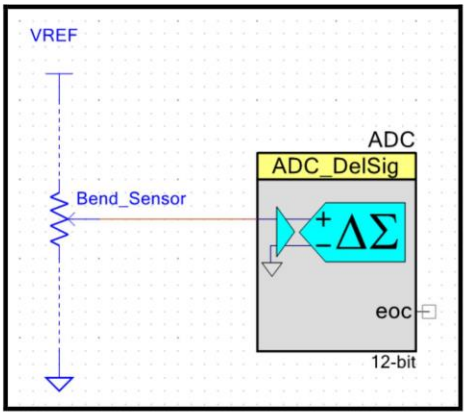
**Bend Sensor® Measurement using a Cypress PSoC Microcontroller:**



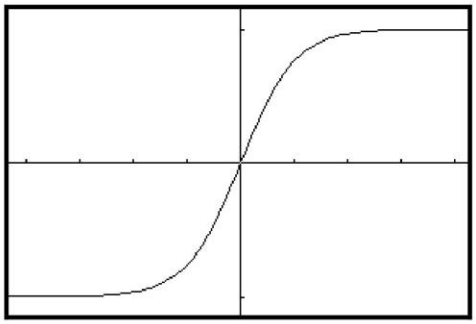

Cypress Semiconductor has a unique microcontroller that has a current DAC. When used in the circuit above the current source sinks current through the Bend Sensor® and generates a voltage across it which can be measured by the ADC. The current DAC also acts like a variable gain amplifier in the circuit which can be adjusted to make sure that the voltage across the Bend Sensor® is always within the range of the ADC. The Mux allows each Bend Sensor® to be measured separately. The PSoC can eliminate the need for any external components to measure the Bend Sensor®.

**Bi-directional Bend Sensors®:**

Flexpoint also makes Bi-directional Bend Sensors®. When bent in one direction one sensor increases in resistance while the other decreases in resistance. This allows for deflection to be measured in either direction. Bi-directional Bend Sensors® share one common pin. Therefore they can be measured individually using many of the above methods as long as the common pin can be tied to VREF or ground. The Bi-directional Bend Sensor® can also be used in a voltage divider circuit using each side as the top and bottom resistor. This looks similar to a potentiometer schematic:



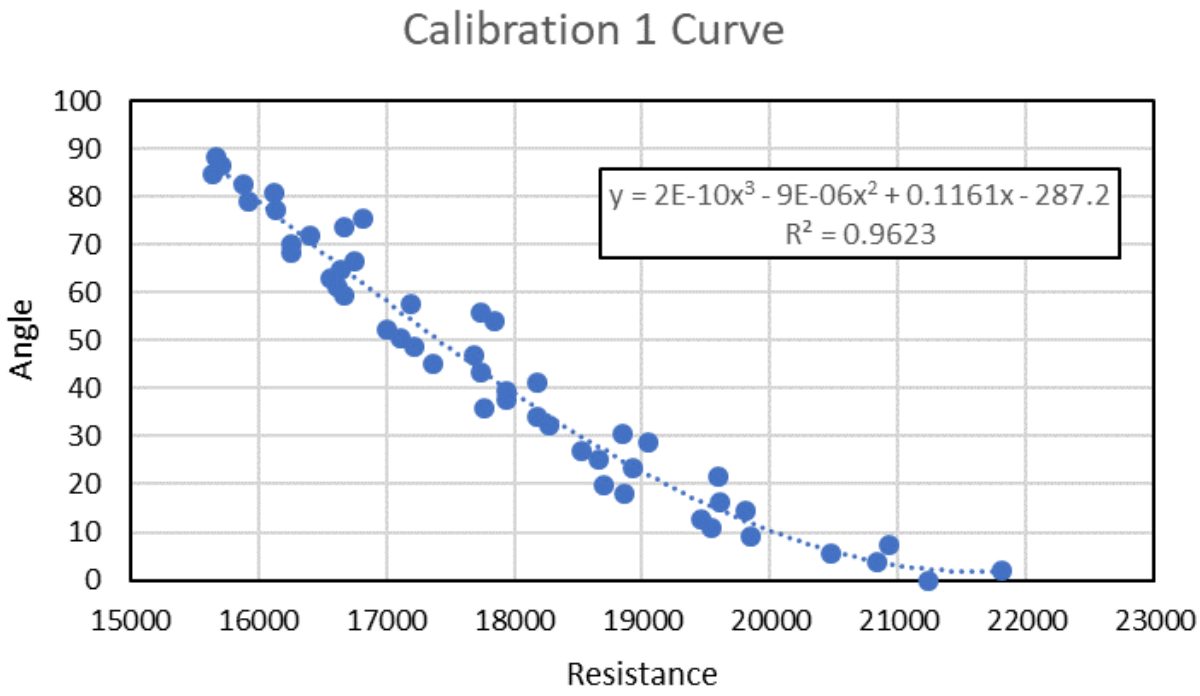

In this configuration the voltage that the ADC will measure will increase or decrease depending on the direction the Bend Sensor® is bent. Because each individual Bend Sensor® cannot go to zero ohms of resistance the response curve looks similar to a hyperbolic tangent function:



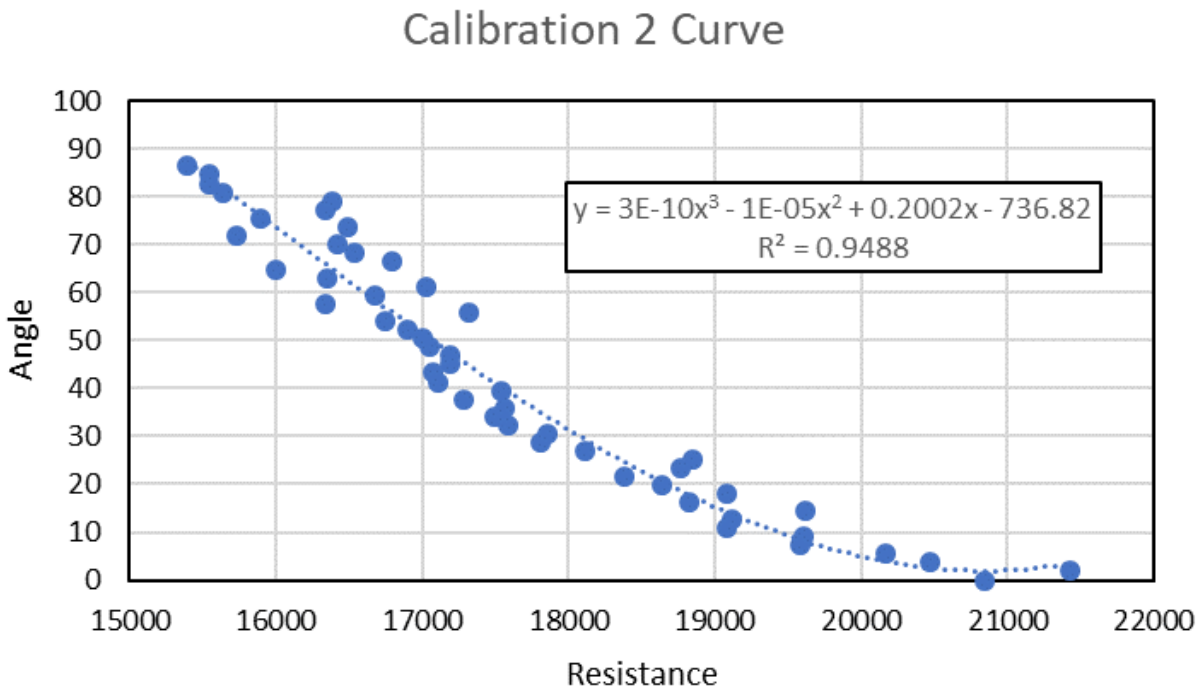
There is however a linear region near the center of the curve (VREF / 2).

## 10.3. Calibration Data Sets 1-5 and Mean Values for Each Reading

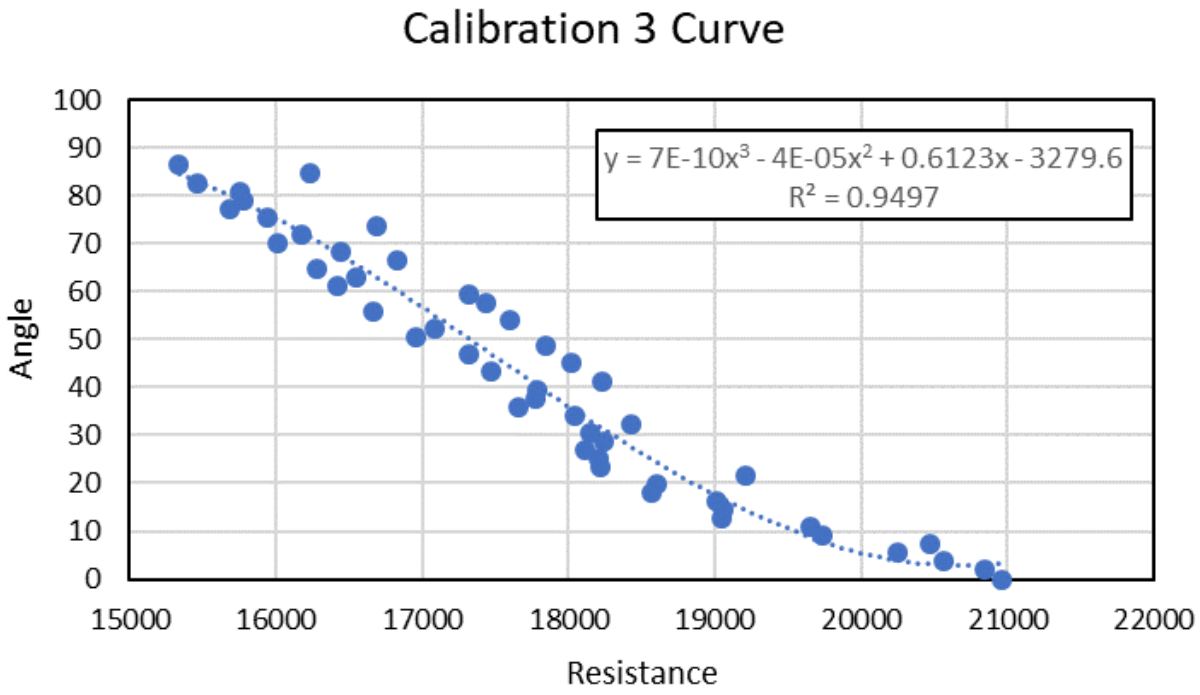
*a. Data Set 1*



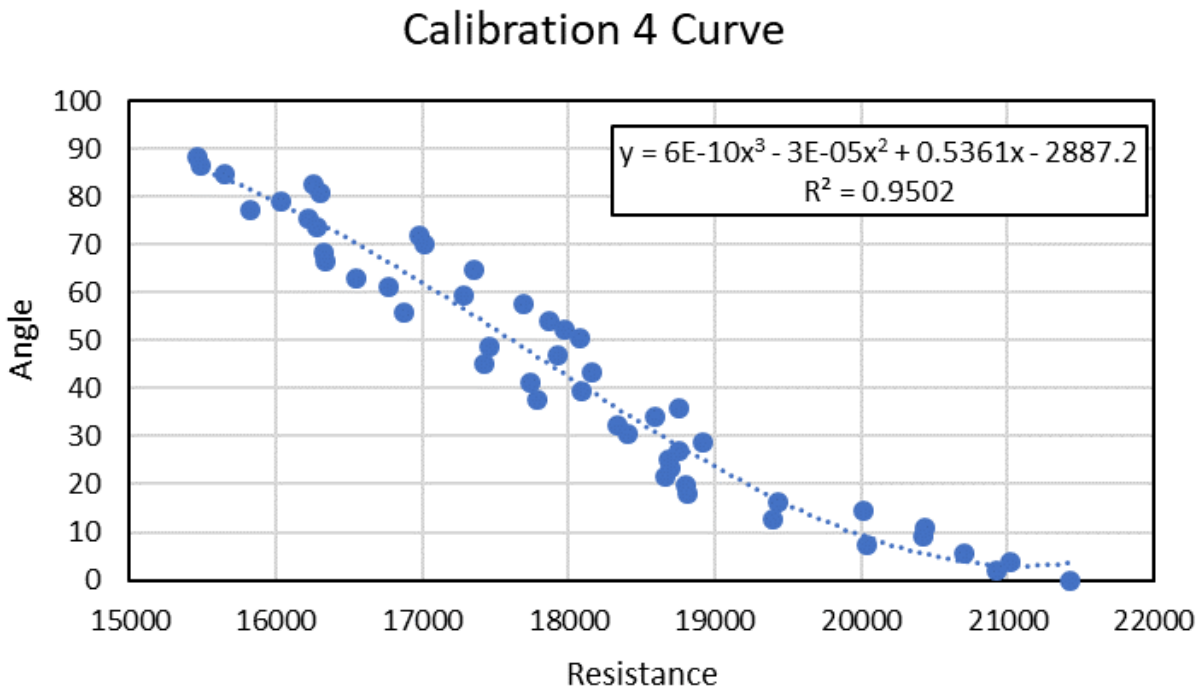
*b. Data Set 2*



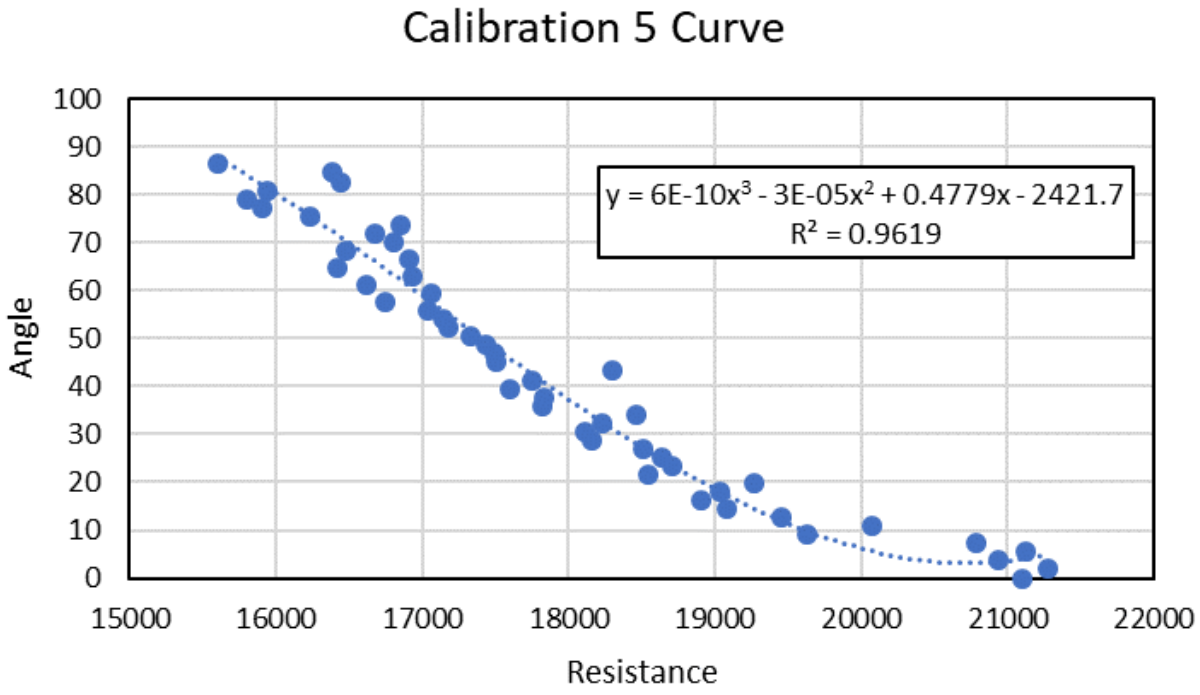

*c. Data Set 3*



*d. Data Set 4*




*e. Data Set 5*



## 10.4. Raw Uncertainty Data

| Measurement Number | Voltage | Resistance | Angle | Z |
|---|---|---|---|---|
| 1 | 2.808063 | 21501.8269 | 4.6295 | 0.555298 |
| 2 | 2.810202 | 21449.7871 | 4.2904 | 0.562502 |
| 3 | 2.807467 | 21516.3449 | 4.7299 | 0.553165 |
| 4 | 2.806065 | 21550.5213 | 4.9764 | 0.547927 |
| 5 | 2.813244 | 21375.9015 | 3.8647 | 0.571548 |
| 6 | 2.818159 | 21256.8485 | 3.3135 | 0.58326 |
| 7 | 2.802963 | 21626.2481 | 5.5737 | 0.535238 |
| 8 | 2.811714 | 21413.0299 | 4.0705 | 0.567175 |
| 9 | 2.811731 | 21412.6129 | 4.0681 | 0.567226 |
| 10 | 2.808046 | 21502.2391 | 4.6323 | 0.555239 |
| 11 | 2.809336 | 21470.8429 | 4.4237 | 0.559671 |
| 12 | 2.817666 | 21268.7657 | 3.3612 | 0.582244 |
| 13 | 2.818811 | 21241.0922 | 3.2528 | 0.584549 |
| 14 | 2.820031 | 21211.6163 | 3.1469 | 0.586798 |
| 15 | 2.819204 | 21231.5982 | 3.2176 | 0.585296 |
| 16 | 2.814527 | 21344.7888 | 3.7047 | 0.574948 |
| 17 | 2.818515 | 21248.2433 | 3.28 | 0.583971 |
| 18 | 2.828068 | 21018.1066 | 2.6939 | 0.596422 |
| 19 | 2.829589 | 20981.6064 | 2.6547 | 0.597257 |
| 20 | 2.820939 | 21189.7015 | 3.0746 | 0.588334 |
| 21 | 2.818588 | 21246.4857 | 3.2732 | 0.584114 |
| 22 | 2.808859 | 21482.4405 | 4.4994 | 0.558063 |
| 23 | 2.81993 | 21214.0515 | 3.1553 | 0.58662 |
| 24 | 2.815188 | 21328.7492 | 3.6266 | 0.576606 |
| 25 | 2.813983 | 21357.9566 | 3.771 | 0.573538 |
| 26 | 2.820078 | 21210.4879 | 3.1431 | 0.58688 |
| 27 | 2.822526 | 21151.442 | 2.9613 | 0.590741 |
| 28 | 2.826477 | 21056.3308 | 2.7506 | 0.595219 |
| 29 | 2.821425 | 21177.973 | 3.0381 | 0.589109 |
| 30 | 2.823148 | 21136.4524 | 2.9215 | 0.591588 |
| 31 | 2.822899 | 21142.4498 | 2.9371 | 0.591256 |
| 32 | 2.821547 | 21175.0343 | 3.0292 | 0.589298 |
| 33 | 2.821923 | 21165.9783 | 3.0024 | 0.589868 |
| 34 | 2.817446 | 21274.0886 | 3.3831 | 0.58178 |
| 35 | 2.829592 | 20981.5321 | 2.6546 | 0.597258 |

| | | | | |
|---|---|---|---|---|
| 36 | 2.822832 | 21144.0584 | 2.9414 | 0.591165 |
| 37 | 2.829082 | 20993.7764 | 2.6662 | 0.597012 |
| 38 | 2.824342 | 21107.6867 | 2.8519 | 0.593066 |
| 39 | 2.812375 | 21396.9929 | 3.9796 | 0.569106 |
| 40 | 2.821209 | 21183.1781 | 3.0541 | 0.58877 |
| 41 | 2.83198 | 20924.3266 | 2.6219 | 0.597952 |
| 42 | 2.825285 | 21084.99 | 2.8035 | 0.594093 |
| 43 | 2.82564 | 21076.4453 | 2.7868 | 0.594449 |
| 44 | 2.825286 | 21084.967 | 2.8035 | 0.594094 |
| 45 | 2.835056 | 20850.7465 | 2.631 | 0.59776 |
| 46 | 2.822955 | 21141.0939 | 2.9335 | 0.591332 |
| 47 | 2.832023 | 20923.2877 | 2.6217 | 0.597958 |
| 48 | 2.832427 | 20913.6138 | 2.6197 | 0.598 |
| 49 | 2.826377 | 21058.732 | 2.7547 | 0.595132 |
| 50 | 2.83778 | 20785.7444 | 2.686 | 0.596591 |
| 51 | 2.824819 | 21096.216 | 2.8268 | 0.5936 |
| 52 | 2.827619 | 21028.8926 | 2.7083 | 0.596117 |
| 53 | 2.819852 | 21215.9341 | 3.1618 | 0.586482 |
| 54 | 2.818537 | 21247.7189 | 3.278 | 0.584014 |
| 55 | 2.836157 | 20824.453 | 2.648 | 0.597399 |
| 56 | 2.831086 | 20945.7395 | 2.6301 | 0.597779 |
| 57 | 2.838921 | 20758.5356 | 2.722 | 0.595827 |
| 58 | 2.828239 | 21014.0084 | 2.6888 | 0.596531 |
| 59 | 2.821748 | 21170.1811 | 3.0148 | 0.589606 |
| 60 | 2.815276 | 21326.635 | 3.6166 | 0.57682 |
| 61 | 2.832926 | 20901.6836 | 2.6186 | 0.598022 |
| 62 | 2.833835 | 20879.9393 | 2.6206 | 0.597981 |
| 63 | 2.837021 | 20803.8446 | 2.6663 | 0.59701 |
| 64 | 2.83867 | 20764.5176 | 2.7134 | 0.596008 |
| 65 | 2.827302 | 21036.4974 | 2.7192 | 0.595886 |
| 66 | 2.82778 | 21025.0163 | 2.703 | 0.59623 |
| 67 | 2.825671 | 21075.6999 | 2.7854 | 0.594479 |
| 68 | 2.837478 | 20792.9481 | 2.6778 | 0.596766 |
| 69 | 2.838683 | 20764.2222 | 2.7138 | 0.596 |
| 70 | 2.833112 | 20897.2344 | 2.6186 | 0.598022 |
| 71 | 2.832921 | 20901.8034 | 2.6186 | 0.598022 |
| 72 | 2.834088 | 20873.897 | 2.622 | 0.597951 |
| 73 | 2.832247 | 20917.9225 | 2.6204 | 0.597984 |
| 74 | 2.834482 | 20864.4765 | 2.625 | 0.597888 |
| 75 | 2.829361 | 20987.079 | 2.6596 | 0.597151 |

| | | | | |
|---|---|---|---|---|
| 76 | 2.834058 | 20874.6152 | 2.6218 | 0.597955 |
| 77 | 2.824048 | 21114.7754 | 2.8682 | 0.59272 |
| 78 | 2.831193 | 20943.1578 | 2.6288 | 0.597806 |
| 79 | 2.831335 | 20939.7654 | 2.6273 | 0.597838 |
| 80 | 2.835833 | 20832.2057 | 2.6422 | 0.597521 |
| 81 | 2.83132 | 20940.1366 | 2.6275 | 0.597835 |
| 82 | 2.839885 | 20735.5888 | 2.7582 | 0.595057 |
| 83 | 2.836943 | 20805.7047 | 2.6645 | 0.597049 |
| 84 | 2.83725 | 20798.3732 | 2.6719 | 0.596891 |
| 85 | 2.83292 | 20901.8148 | 2.6186 | 0.598022 |
| 86 | 2.819096 | 21234.2076 | 3.2272 | 0.585093 |
| 87 | 2.838561 | 20767.1141 | 2.7098 | 0.596085 |
| 88 | 2.830723 | 20954.4227 | 2.6348 | 0.59768 |
| 89 | 2.829396 | 20986.2555 | 2.6589 | 0.597167 |
| 90 | 2.842621 | 20670.5071 | 2.8898 | 0.592261 |
| 91 | 2.840764 | 20714.6714 | 2.7958 | 0.594257 |
| 92 | 2.831514 | 20935.4768 | 2.6256 | 0.597875 |
| 93 | 2.829363 | 20987.0275 | 2.6596 | 0.597152 |
| 94 | 2.836874 | 20807.3486 | 2.6629 | 0.597082 |
| 95 | 2.850202 | 20490.83 | 3.4705 | 0.579924 |
| 96 | 2.840989 | 20709.3051 | 2.8062 | 0.594037 |
| 97 | 2.84006 | 20731.4111 | 2.7653 | 0.594905 |
| 98 | 2.833941 | 20877.4026 | 2.6211 | 0.597969 |
| 99 | 2.85129 | 20465.1135 | 3.5791 | 0.577617 |
| 100 | 2.835441 | 20841.5659 | 2.6361 | 0.597651 |
| 101 | 2.851444 | 20461.4759 | 3.5949 | 0.577279 |
| 102 | 2.850675 | 20479.6622 | 3.5169 | 0.578938 |
| 103 | 2.851561 | 20458.717 | 3.607 | 0.577022 |
| 104 | 2.850497 | 20483.8595 | 3.4993 | 0.579312 |
| 105 | 2.847792 | 20547.8536 | 3.2522 | 0.584561 |
| 106 | 2.84721 | 20561.6248 | 3.2042 | 0.585581 |
| 107 | 2.844058 | 20636.3662 | 2.9758 | 0.590434 |
| 108 | 2.839511 | 20744.4965 | 2.7435 | 0.59537 |
| 109 | 2.835226 | 20846.7027 | 2.6331 | 0.597714 |
| 110 | 2.849609 | 20504.8429 | 3.4139 | 0.581125 |
| 111 | 2.839393 | 20747.2902 | 2.739 | 0.595464 |
| 112 | 2.841901 | 20687.6104 | 2.8511 | 0.593083 |
| 113 | 2.838897 | 20759.1264 | 2.7211 | 0.595845 |
| 114 | 2.846464 | 20579.3002 | 3.1452 | 0.586834 |
| 115 | 2.842818 | 20665.8106 | 2.901 | 0.592024 |

| | | | | |
|---|---|---|---|---|
| 116 | 2.845423 | 20603.973 | 3.0681 | 0.588473 |
| 117 | 2.846337 | 20582.3176 | 3.1355 | 0.587041 |
| 118 | 2.851156 | 20468.2953 | 3.5653 | 0.577909 |
| 119 | 2.849974 | 20496.2293 | 3.4485 | 0.580391 |
| 120 | 2.849509 | 20507.2222 | 3.4045 | 0.581325 |
| 121 | 2.842456 | 20674.4223 | 2.8807 | 0.592454 |
| 122 | 2.8441 | 20635.3701 | 2.9785 | 0.590377 |
| 123 | 2.844534 | 20625.0652 | 3.0068 | 0.589775 |
| 124 | 2.850967 | 20472.7447 | 3.5462 | 0.578315 |
| 125 | 2.853181 | 20420.4873 | 3.7823 | 0.573298 |
| 126 | 2.854237 | 20395.6 | 3.9038 | 0.570718 |
| 127 | 2.842636 | 20670.1559 | 2.8906 | 0.592243 |
| 128 | 2.845483 | 20602.5651 | 3.0723 | 0.588383 |
| 129 | 2.859686 | 20267.4081 | 4.6194 | 0.555513 |
| 130 | 2.839938 | 20734.3286 | 2.7603 | 0.595012 |
| 131 | 2.848939 | 20520.7004 | 3.3522 | 0.582436 |
| 132 | 2.847521 | 20554.2614 | 3.2296 | 0.585041 |
| 133 | 2.857158 | 20326.8131 | 4.2692 | 0.562954 |
| 134 | 2.851961 | 20449.2765 | 3.649 | 0.57613 |
| 135 | 2.848346 | 20534.72 | 3.2997 | 0.583552 |
| 136 | 2.857685 | 20314.4214 | 4.3396 | 0.561458 |
| 137 | 2.86088 | 20239.3671 | 4.7958 | 0.551766 |
| 138 | 2.858422 | 20297.0919 | 4.4404 | 0.559316 |
| 139 | 2.853801 | 20405.8574 | 3.853 | 0.571796 |
| 140 | 2.857046 | 20329.4378 | 4.2544 | 0.563267 |
| 141 | 2.85464 | 20386.0817 | 3.9517 | 0.569699 |
| 142 | 2.85414 | 20397.8873 | 3.8924 | 0.57096 |
| 143 | 2.862504 | 20201.2965 | 5.0463 | 0.546443 |
| 144 | 2.85607 | 20352.424 | 4.128 | 0.565953 |
| 145 | 2.861064 | 20235.0603 | 4.8235 | 0.551178 |
| 146 | 2.862812 | 20194.0898 | 5.0952 | 0.545404 |
| 147 | 2.852912 | 20426.8372 | 3.7523 | 0.573937 |
| 148 | 2.862095 | 20210.8911 | 4.982 | 0.54781 |
| 149 | 2.861654 | 20221.2265 | 4.9135 | 0.549263 |
| 150 | 2.861715 | 20219.8009 | 4.9229 | 0.549064 |
| 151 | 2.86052 | 20247.8312 | 4.7418 | 0.552913 |
| 152 | 2.85743 | 20320.4204 | 4.3053 | 0.562186 |
| 153 | 2.86901 | 20049.1783 | 6.173 | 0.522504 |
| 154 | 2.866007 | 20119.2983 | 5.629 | 0.534063 |
| 155 | 2.860662 | 20244.48 | 4.7631 | 0.55246 |

| | | | | |
|---|---|---|---|---|
| 156 | 2.865451 | 20132.3151 | 5.5326 | 0.53611 |
| 157 | 2.864957 | 20143.87 | 5.4483 | 0.537902 |
| 158 | 2.872719 | 19962.7746 | 6.9 | 0.507058 |
| 159 | 2.866816 | 20100.3926 | 5.7716 | 0.531034 |
| 160 | 2.863663 | 20174.1536 | 5.2327 | 0.542482 |
| 161 | 2.864142 | 20162.9399 | 5.3116 | 0.540806 |
| 162 | 2.869697 | 20033.1639 | 6.3031 | 0.51974 |
| 163 | 2.865801 | 20124.1254 | 5.5931 | 0.534826 |
| 164 | 2.861357 | 20228.193 | 4.868 | 0.550232 |
| 165 | 2.873183 | 19951.9586 | 6.9954 | 0.505032 |
| 166 | 2.86204 | 20212.171 | 4.9734 | 0.547991 |
| 167 | 2.863997 | 20166.3165 | 5.2877 | 0.541313 |
| 168 | 2.863174 | 20185.589 | 5.1534 | 0.544167 |
| 169 | 2.865921 | 20121.3105 | 5.614 | 0.534381 |
| 170 | 2.862273 | 20206.7054 | 5.0099 | 0.547216 |
| 171 | 2.859296 | 20276.5625 | 4.5634 | 0.556703 |
| 172 | 2.858484 | 20295.6351 | 4.449 | 0.559133 |
| 173 | 2.866254 | 20113.5246 | 5.6722 | 0.533144 |
| 174 | 2.860166 | 20256.1295 | 4.6895 | 0.554024 |
| 175 | 2.86057 | 20246.6507 | 4.7493 | 0.552754 |
| 176 | 2.869249 | 20043.6113 | 6.218 | 0.521548 |
| 177 | 2.863033 | 20188.912 | 5.1306 | 0.544653 |
| 178 | 2.870608 | 20011.9232 | 6.4789 | 0.516005 |
| 179 | 2.879036 | 19816.1036 | 8.2728 | 0.47789 |
| 180 | 2.866872 | 20099.1003 | 5.7814 | 0.530824 |
| 181 | 2.875453 | 19899.1991 | 7.474 | 0.494863 |
| 182 | 2.871818 | 19983.7287 | 6.7181 | 0.510924 |
| 183 | 2.866337 | 20111.5854 | 5.6868 | 0.532835 |
| 184 | 2.871613 | 19988.5085 | 6.677 | 0.511795 |
| 185 | 2.869037 | 20048.5554 | 6.178 | 0.522397 |
| 186 | 2.868889 | 20052.0095 | 6.1503 | 0.522988 |
| 187 | 2.865178 | 20138.6888 | 5.486 | 0.537102 |
| 188 | 2.870088 | 20024.0368 | 6.3782 | 0.518145 |
| 189 | 2.872099 | 19977.1958 | 6.7744 | 0.509727 |
| 190 | 2.855347 | 20369.4283 | 4.0377 | 0.567873 |
| 191 | 2.881467 | 19759.8252 | 8.8445 | 0.465744 |
| 192 | 2.860364 | 20251.4849 | 4.7187 | 0.553403 |
| 193 | 2.84914 | 20515.9344 | 3.3705 | 0.582047 |
| 194 | 2.88307 | 19722.7868 | 9.234 | 0.457468 |
| 195 | 2.871679 | 19986.9707 | 6.6902 | 0.511515 |

| | | | | |
|---|---|---|---|---|
| 196 | 2.869447 | 20038.9793 | 6.2556 | 0.520749 |
| 197 | 2.869979 | 20026.5768 | 6.3572 | 0.518591 |
| 198 | 2.870292 | 20019.2798 | 6.4176 | 0.517308 |
| 199 | 2.864321 | 20158.7378 | 5.3415 | 0.540172 |
| 200 | 2.867282 | 20089.5153 | 5.855 | 0.529261 |
| 201 | 2.897402 | 19393.3238 | 13.14 | 0.374487 |
| 202 | 2.892897 | 19496.5433 | 11.833 | 0.402241 |
| 203 | 2.890705 | 19546.8766 | 11.223 | 0.415206 |
| 204 | 2.885697 | 19662.1572 | 9.8939 | 0.443448 |
| 205 | 2.881861 | 19750.7174 | 8.9393 | 0.463729 |
| 206 | 2.883738 | 19707.352 | 9.3994 | 0.453955 |
| 207 | 2.888342 | 19601.2043 | 10.585 | 0.428771 |
| 208 | 2.894114 | 19468.634 | 12.179 | 0.39489 |
| 209 | 2.893644 | 19479.4034 | 12.045 | 0.39774 |
| 210 | 2.885718 | 19661.6624 | 9.8994 | 0.443331 |
| 211 | 2.89129 | 19533.4236 | 11.385 | 0.411778 |
| 212 | 2.906759 | 19179.9927 | 16.063 | 0.312378 |
| 213 | 2.888641 | 19594.3344 | 10.664 | 0.42708 |
| 214 | 2.897438 | 19392.5058 | 13.15 | 0.374261 |
| 215 | 2.905959 | 19198.1818 | 15.802 | 0.317913 |
| 216 | 2.888294 | 19602.3238 | 10.572 | 0.429045 |
| 217 | 2.896141 | 19422.1905 | 12.767 | 0.382404 |
| 218 | 2.90987 | 19109.3747 | 17.094 | 0.290479 |
| 219 | 2.901951 | 19289.4409 | 14.527 | 0.345021 |
| 220 | 2.906167 | 19193.4602 | 15.87 | 0.316481 |
| 221 | 2.897871 | 19382.5932 | 13.28 | 0.371513 |
| 222 | 2.904618 | 19228.6965 | 15.37 | 0.327102 |
| 223 | 2.897009 | 19402.3232 | 13.023 | 0.376968 |
| 224 | 2.897092 | 19400.425 | 13.047 | 0.376446 |
| 225 | 2.905104 | 19217.6177 | 15.526 | 0.32378 |
| 226 | 2.901473 | 19300.3425 | 14.378 | 0.348183 |
| 227 | 2.909827 | 19110.3479 | 17.079 | 0.290785 |
| 228 | 2.911162 | 19080.072 | 17.53 | 0.281205 |
| 229 | 2.90687 | 19177.4625 | 16.099 | 0.311604 |
| 230 | 2.916298 | 18963.9124 | 19.31 | 0.243393 |
| 231 | 2.905524 | 19208.0827 | 15.661 | 0.320908 |
| 232 | 2.919905 | 18882.5683 | 20.601 | 0.215951 |
| 233 | 2.920895 | 18860.2863 | 20.962 | 0.208301 |
| 234 | 2.891437 | 19530.0502 | 11.425 | 0.410914 |
| 235 | 2.910453 | 19096.1523 | 17.29 | 0.286308 |

| | | | | |
|---|---|---|---|---|
| 236 | 2.915744 | 18976.4195 | 19.114 | 0.247543 |
| 237 | 2.904797 | 19224.6161 | 15.427 | 0.32588 |
| 238 | 2.91623 | 18965.4574 | 19.286 | 0.243907 |
| 239 | 2.911822 | 19065.1389 | 17.755 | 0.276437 |
| 240 | 2.914834 | 18996.986 | 18.795 | 0.254327 |
| 241 | 2.913411 | 19029.1597 | 18.301 | 0.264836 |
| 242 | 2.911177 | 19079.7371 | 17.535 | 0.281098 |
| 243 | 2.909585 | 19115.8202 | 16.998 | 0.292504 |
| 244 | 2.920178 | 18876.4313 | 20.7 | 0.21385 |
| 245 | 2.920142 | 18877.2474 | 20.687 | 0.21413 |
| 246 | 2.903644 | 19250.8653 | 15.059 | 0.333699 |
| 247 | 2.9031 | 19263.2643 | 14.887 | 0.33736 |
| 248 | 2.911028 | 19083.1243 | 17.484 | 0.282176 |
| 249 | 2.920505 | 18869.0663 | 20.819 | 0.211322 |
| 250 | 2.918039 | 18924.6239 | 19.929 | 0.230236 |
| 251 | 2.916122 | 18967.8961 | 19.248 | 0.244717 |
| 252 | 2.92202 | 18834.9789 | 21.374 | 0.199543 |
| 253 | 2.922357 | 18827.3927 | 21.498 | 0.196903 |
| 254 | 2.918029 | 18924.8604 | 19.925 | 0.230315 |
| 255 | 2.91397 | 19016.5193 | 18.494 | 0.260722 |
| 256 | 2.911153 | 19080.2881 | 17.527 | 0.281274 |
| 257 | 2.928051 | 18699.6229 | 23.635 | 0.151502 |
| 258 | 2.926937 | 18724.5749 | 23.211 | 0.160506 |
| 259 | 2.927505 | 18711.8588 | 23.427 | 0.155926 |
| 260 | 2.932564 | 18598.7001 | 25.38 | 0.114424 |
| 261 | 2.922029 | 18834.7698 | 21.377 | 0.19947 |
| 262 | 2.922311 | 18828.4434 | 21.481 | 0.197269 |
| 263 | 2.921416 | 18848.5742 | 21.152 | 0.204256 |
| 264 | 2.924724 | 18774.2201 | 22.377 | 0.178224 |
| 265 | 2.921538 | 18845.8118 | 21.197 | 0.2033 |
| 266 | 2.913705 | 19022.5045 | 18.402 | 0.262672 |
| 267 | 2.92167 | 18842.8512 | 21.245 | 0.202275 |
| 268 | 2.927327 | 18715.8387 | 23.359 | 0.157361 |
| 269 | 2.920518 | 18868.7765 | 20.824 | 0.211222 |
| 270 | 2.92236 | 18827.3391 | 21.499 | 0.196885 |
| 271 | 2.927031 | 18722.4694 | 23.247 | 0.159749 |
| 272 | 2.911458 | 19073.385 | 17.63 | 0.279073 |
| 273 | 2.922945 | 18814.1919 | 21.715 | 0.192295 |
| 274 | 2.92589 | 18748.0592 | 22.815 | 0.168921 |
| 275 | 2.92265 | 18820.8211 | 21.606 | 0.194612 |

| | | | | |
|---|---|---|---|---|
| 276 | 2.921067 | 18856.4229 | 21.024 | 0.206968 |
| 277 | 2.92003 | 18879.7708 | 20.646 | 0.214994 |
| 278 | 2.924465 | 18780.0329 | 22.28 | 0.180282 |
| 279 | 2.927702 | 18707.4467 | 23.502 | 0.154332 |
| 280 | 2.927691 | 18707.6924 | 23.497 | 0.154421 |
| 281 | 2.933422 | 18579.5413 | 25.717 | 0.107269 |
| 282 | 2.920885 | 18860.5116 | 20.958 | 0.208378 |
| 283 | 2.931331 | 18626.2392 | 24.899 | 0.124645 |
| 284 | 2.944015 | 18344.0096 | 29.99 | 0.016478 |
| 285 | 2.923076 | 18811.2449 | 21.763 | 0.191264 |
| 286 | 2.918997 | 18903.0333 | 20.273 | 0.222928 |
| 287 | 2.92566 | 18753.2095 | 22.728 | 0.170758 |
| 288 | 2.928227 | 18695.6823 | 23.702 | 0.150074 |
| 289 | 2.920465 | 18869.9627 | 20.805 | 0.21163 |
| 290 | 2.920902 | 18860.1253 | 20.964 | 0.208245 |
| 291 | 2.91962 | 18889.0072 | 20.498 | 0.218152 |
| 292 | 2.920066 | 18878.944 | 20.66 | 0.214711 |
| 293 | 2.921562 | 18845.2755 | 21.206 | 0.203115 |
| 294 | 2.91962 | 18889.0072 | 20.498 | 0.218152 |
| 295 | 2.918266 | 18919.5111 | 20.01 | 0.22851 |
| 296 | 2.928055 | 18699.5374 | 23.636 | 0.151471 |
| 297 | 2.921479 | 18847.1528 | 21.175 | 0.203765 |
| 298 | 2.91942 | 18893.5082 | 20.426 | 0.219687 |
| 299 | 2.920208 | 18875.7549 | 20.711 | 0.213618 |
| 300 | 2.934273 | 18560.5532 | 26.052 | 0.100143 |
| 301 | 2.940038 | 18432.2344 | 28.361 | 0.051081 |
| 302 | 2.934158 | 18563.1322 | 26.006 | 0.101113 |
| 303 | 2.928631 | 18686.6336 | 23.857 | 0.146789 |
| 304 | 2.934125 | 18563.8714 | 25.993 | 0.101391 |
| 305 | 2.927057 | 18721.887 | 23.257 | 0.15954 |
| 306 | 2.922 | 18835.4186 | 21.367 | 0.199695 |
| 307 | 2.92128 | 18851.6319 | 21.102 | 0.205314 |
| 308 | 2.929902 | 18658.185 | 24.345 | 0.136404 |
| 309 | 2.924302 | 18783.7053 | 22.219 | 0.181579 |
| 310 | 2.93423 | 18561.5263 | 26.035 | 0.100509 |
| 311 | 2.930396 | 18647.1473 | 24.536 | 0.132353 |
| 312 | 2.935727 | 18528.1501 | 26.628 | 0.0879 |
| 313 | 2.938082 | 18475.7257 | 27.571 | 0.067882 |
| 314 | 2.943971 | 18344.9868 | 29.972 | 0.016865 |
| 315 | 2.938133 | 18474.5855 | 27.591 | 0.067444 |

| | | | | |
|---|---|---|---|---|
| 316 | 2.944576 | 18331.5938 | 30.222 | 0.011554 |
| 317 | 2.935956 | 18523.0403 | 26.72 | 0.085961 |
| 318 | 2.929505 | 18667.0712 | 24.192 | 0.139657 |
| 319 | 2.938052 | 18476.3887 | 27.559 | 0.068137 |
| 320 | 2.935315 | 18537.3251 | 26.465 | 0.091377 |
| 321 | 2.944209 | 18339.7261 | 30.07 | 0.014781 |
| 322 | 2.937644 | 18485.459 | 27.395 | 0.071618 |
| 323 | 2.943211 | 18361.8416 | 29.658 | 0.023527 |
| 324 | 2.936996 | 18499.8813 | 27.135 | 0.077138 |
| 325 | 2.936723 | 18505.9695 | 27.025 | 0.079462 |
| 326 | 2.952178 | 18163.6555 | 33.415 | 0.056291 |
| 327 | 2.944243 | 18338.9656 | 30.084 | 0.014479 |
| 328 | 2.94017 | 18429.3162 | 28.415 | 0.049947 |
| 329 | 2.938809 | 18459.5594 | 27.864 | 0.061657 |
| 330 | 2.943512 | 18355.1515 | 29.782 | 0.020885 |
| 331 | 2.945037 | 18321.3889 | 30.413 | 0.007497 |
| 332 | 2.957574 | 18044.9984 | 35.732 | 0.105531 |
| 333 | 2.9512 | 18185.2155 | 32.999 | 0.047456 |
| 334 | 2.955643 | 18087.4145 | 34.899 | 0.087813 |
| 335 | 2.952228 | 18162.5734 | 33.436 | 0.056736 |
| 336 | 2.953742 | 18129.225 | 34.082 | 0.070473 |
| 337 | 2.951684 | 18174.5625 | 33.204 | 0.051817 |
| 338 | 2.951221 | 18184.7529 | 33.008 | 0.047645 |
| 339 | 2.952212 | 18162.9149 | 33.429 | 0.056595 |
| 340 | 2.948135 | 18252.857 | 31.706 | 0.019972 |
| 341 | 2.955088 | 18099.6223 | 34.66 | 0.082737 |
| 342 | 2.956297 | 18073.0322 | 35.181 | 0.093806 |
| 343 | 2.957243 | 18052.2689 | 35.589 | 0.102485 |
| 344 | 2.954343 | 18116.0043 | 34.34 | 0.075942 |
| 345 | 2.946355 | 18292.2174 | 30.961 | 0.004148 |
| 346 | 2.949153 | 18230.3837 | 32.133 | 0.029064 |
| 347 | 2.956908 | 18059.6301 | 35.444 | 0.099405 |
| 348 | 2.953146 | 18142.3461 | 33.828 | 0.065058 |
| 349 | 2.959482 | 18003.1289 | 36.561 | 0.123144 |
| 350 | 2.957438 | 18047.9765 | 35.674 | 0.104283 |
| 351 | 2.956225 | 18074.6142 | 35.15 | 0.093146 |
| 352 | 2.957387 | 18049.1019 | 35.652 | 0.103811 |
| 353 | 2.963145 | 17922.9303 | 38.165 | 0.157204 |
| 354 | 2.954328 | 18116.3347 | 34.333 | 0.075805 |
| 355 | 2.961212 | 17965.2288 | 37.317 | 0.139188 |

| | | | | |
|---|---|---|---|---|
| 356 | 2.963108 | 17923.7437 | 38.148 | 0.156857 |
| 357 | 2.962725 | 17932.1189 | 37.98 | 0.153281 |
| 358 | 2.952833 | 18149.2447 | 33.694 | 0.062216 |
| 359 | 2.953196 | 18141.2437 | 33.849 | 0.065513 |
| 360 | 2.955904 | 18081.687 | 35.011 | 0.090198 |
| 361 | 2.960623 | 17978.1428 | 37.059 | 0.13371 |
| 362 | 2.964248 | 17898.8135 | 38.65 | 0.167527 |
| 363 | 2.964612 | 17890.8636 | 38.811 | 0.170938 |
| 364 | 2.964108 | 17901.8773 | 38.589 | 0.166214 |
| 365 | 2.959254 | 18008.1315 | 36.462 | 0.121033 |
| 366 | 2.965243 | 17877.0784 | 39.09 | 0.176861 |
| 367 | 2.954654 | 18109.1547 | 34.473 | 0.078781 |
| 368 | 2.970231 | 17768.3308 | 41.307 | 0.223977 |
| 369 | 2.9584 | 18026.8683 | 36.091 | 0.113143 |
| 370 | 2.962226 | 17943.0422 | 37.761 | 0.148624 |
| 371 | 2.962973 | 17926.6847 | 38.089 | 0.155601 |
| 372 | 2.949563 | 18221.3419 | 32.306 | 0.032733 |
| 373 | 2.961261 | 17964.1638 | 37.338 | 0.13964 |
| 374 | 2.956115 | 18077.0397 | 35.102 | 0.092135 |
| 375 | 2.958801 | 18018.0764 | 36.265 | 0.116842 |
| 376 | 2.956125 | 18076.8249 | 35.106 | 0.092224 |
| 377 | 2.957322 | 18050.5205 | 35.624 | 0.103217 |
| 378 | 2.959729 | 17997.7141 | 36.669 | 0.12543 |
| 379 | 2.964715 | 17888.603 | 38.857 | 0.171908 |
| 380 | 2.96588 | 17863.1794 | 39.371 | 0.182845 |
| 381 | 2.956807 | 18061.8294 | 35.401 | 0.098485 |
| 382 | 2.955405 | 18092.6503 | 34.796 | 0.085634 |
| 383 | 2.957186 | 18053.5253 | 35.564 | 0.101959 |
| 384 | 2.956764 | 18062.7719 | 35.382 | 0.098091 |
| 385 | 2.961078 | 17968.1736 | 37.258 | 0.137938 |
| 386 | 2.958294 | 18029.1857 | 36.045 | 0.112169 |
| 387 | 2.959431 | 18004.2632 | 36.539 | 0.122665 |
| 388 | 2.965682 | 17867.5046 | 39.284 | 0.180981 |
| 389 | 2.971531 | 17740.0409 | 41.889 | 0.236342 |
| 390 | 2.958791 | 18018.2856 | 36.261 | 0.116754 |
| 391 | 2.958603 | 18022.4119 | 36.179 | 0.115017 |
| 392 | 2.968965 | 17795.8977 | 40.742 | 0.21197 |
| 393 | 2.968033 | 17816.2114 | 40.327 | 0.203149 |
| 394 | 2.952783 | 18150.3368 | 33.673 | 0.061767 |
| 395 | 2.954463 | 18113.3608 | 34.391 | 0.077037 |

| | | | | |
|---|---|---|---|---|
| 396 | 2.961433 | 17960.3946 | 37.413 | 0.141241 |
| 397 | 2.962177 | 17944.1066 | 37.739 | 0.14817 |
| 398 | 2.9592 | 18009.3183 | 36.439 | 0.120533 |
| 399 | 2.961395 | 17961.2298 | 37.397 | 0.140886 |
| 400 | 2.973343 | 17700.6712 | 42.703 | 0.25362 |
| 401 | 2.95903 | 18013.0513 | 36.364 | 0.118959 |
| 402 | 2.963246 | 17920.7144 | 38.209 | 0.158151 |
| 403 | 2.962773 | 17931.0653 | 38.001 | 0.153731 |
| 404 | 2.958117 | 18033.078 | 35.968 | 0.110533 |
| 405 | 2.964086 | 17902.3566 | 38.579 | 0.166008 |
| 406 | 2.96092 | 17971.6251 | 37.189 | 0.136474 |
| 407 | 2.964688 | 17889.2124 | 38.844 | 0.171647 |
| 408 | 2.958398 | 18026.905 | 36.09 | 0.113128 |
| 409 | 2.946758 | 18283.2955 | 31.129 | 0.007724 |
| 410 | 2.958858 | 18016.8109 | 36.29 | 0.117375 |
| 411 | 2.959294 | 18007.2585 | 36.479 | 0.121401 |
| 412 | 2.971833 | 17733.488 | 42.024 | 0.239212 |
| 413 | 2.961762 | 17953.1916 | 37.557 | 0.144303 |
| 414 | 2.965268 | 17876.5369 | 39.101 | 0.177094 |
| 415 | 2.958577 | 18022.982 | 36.168 | 0.114777 |
| 416 | 2.967654 | 17824.4601 | 40.159 | 0.199574 |
| 417 | 2.973433 | 17698.719 | 42.743 | 0.254479 |
| 418 | 2.973786 | 17691.0563 | 42.902 | 0.257852 |
| 419 | 2.965734 | 17866.3647 | 39.307 | 0.181472 |
| 420 | 2.966992 | 17838.9146 | 39.864 | 0.193318 |
| 421 | 2.963409 | 17917.1483 | 38.281 | 0.159676 |
| 422 | 2.966623 | 17846.9557 | 39.701 | 0.189844 |
| 423 | 2.964504 | 17893.2286 | 38.763 | 0.169923 |
| 424 | 2.964375 | 17896.0313 | 38.707 | 0.16872 |
| 425 | 2.968712 | 17801.4191 | 40.629 | 0.20957 |
| 426 | 2.967761 | 17822.1365 | 40.206 | 0.20058 |
| 427 | 2.974882 | 17667.267 | 43.395 | 0.268341 |
| 428 | 2.96581 | 17864.6991 | 39.341 | 0.18219 |
| 429 | 2.968526 | 17805.4555 | 40.547 | 0.207817 |
| 430 | 2.969551 | 17783.1341 | 41.004 | 0.217524 |
| 431 | 2.980592 | 17543.6042 | 45.982 | 0.323299 |
| 432 | 2.982837 | 17495.1175 | 47.005 | 0.345028 |
| 433 | 2.967669 | 17824.1482 | 40.165 | 0.199709 |
| 434 | 2.982132 | 17510.3313 | 46.683 | 0.3382 |
| 435 | 2.974653 | 17672.2283 | 43.292 | 0.266151 |

| | | | | |
|---|---|---|---|---|
| 436 | 2.977808 | 17603.8454 | 44.718 | 0.29644 |
| 437 | 2.977091 | 17619.3586 | 44.394 | 0.28955 |
| 438 | 2.969132 | 17792.2519 | 40.817 | 0.213555 |
| 439 | 2.97114 | 17748.5502 | 41.714 | 0.232618 |
| 440 | 2.970659 | 17759.007 | 41.499 | 0.228048 |
| 441 | 2.970616 | 17759.9461 | 41.48 | 0.227637 |
| 442 | 2.985055 | 17447.2793 | 48.018 | 0.366558 |
| 443 | 2.975221 | 17659.9014 | 43.549 | 0.271595 |
| 444 | 2.976856 | 17624.452 | 44.287 | 0.28729 |
| 445 | 2.984173 | 17466.3001 | 47.615 | 0.357987 |
| 446 | 2.994343 | 17247.7493 | 52.284 | 0.457199 |
| 447 | 2.9917 | 17304.41 | 51.067 | 0.431335 |
| 448 | 2.987948 | 17385.0019 | 49.343 | 0.394712 |
| 449 | 2.970725 | 17757.5753 | 41.528 | 0.228673 |
| 450 | 2.985093 | 17446.4624 | 48.035 | 0.366926 |
| 451 | 2.975932 | 17644.4928 | 43.869 | 0.278409 |
| 452 | 2.984005 | 17469.9144 | 47.538 | 0.35636 |
| 453 | 2.991264 | 17313.7515 | 50.867 | 0.42708 |
| 454 | 2.981061 | 17533.4693 | 46.195 | 0.327833 |
| 455 | 2.988259 | 17378.3209 | 49.486 | 0.39774 |
| 456 | 2.984779 | 17453.2346 | 47.892 | 0.363873 |
| 457 | 2.986042 | 17426.0104 | 48.47 | 0.376157 |
| 458 | 2.98346 | 17481.67 | 47.289 | 0.351071 |
| 459 | 2.98389 | 17472.4081 | 47.485 | 0.355238 |
| 460 | 2.983914 | 17471.8734 | 47.497 | 0.355478 |
| 461 | 2.993054 | 17275.3645 | 51.69 | 0.444583 |
| 462 | 2.987282 | 17399.3274 | 49.038 | 0.388223 |
| 463 | 2.984254 | 17464.542 | 47.652 | 0.358779 |
| 464 | 2.977273 | 17615.4176 | 44.476 | 0.291299 |
| 465 | 2.980974 | 17535.3548 | 46.156 | 0.326989 |
| 466 | 2.984354 | 17462.3881 | 47.698 | 0.359749 |
| 467 | 2.98686 | 17408.4142 | 48.844 | 0.384112 |
| 468 | 2.986762 | 17410.5132 | 48.8 | 0.383162 |
| 469 | 2.981678 | 17520.1499 | 46.476 | 0.333798 |
| 470 | 2.991063 | 17318.0804 | 50.774 | 0.425109 |
| 471 | 2.979234 | 17572.9602 | 45.365 | 0.31019 |
| 472 | 2.981399 | 17526.1651 | 46.349 | 0.331103 |
| 473 | 2.981064 | 17533.4023 | 46.197 | 0.327863 |
| 474 | 2.980148 | 17553.1958 | 45.78 | 0.319012 |
| 475 | 2.986363 | 17419.1053 | 48.617 | 0.379278 |

| | | | | |
|---|---|---|---|---|
| 476 | 2.991828 | 17301.6634 | 51.126 | 0.432587 |
| 477 | 2.985894 | 17429.2145 | 48.402 | 0.37471 |
| 478 | 2.981834 | 17516.7668 | 46.548 | 0.335314 |
| 479 | 2.992852 | 17279.6984 | 51.597 | 0.442605 |
| 480 | 2.980151 | 17553.134 | 45.782 | 0.319039 |
| 481 | 2.985947 | 17428.0591 | 48.426 | 0.375232 |
| 482 | 2.987373 | 17397.3625 | 49.08 | 0.389113 |
| 483 | 2.994771 | 17238.5751 | 52.482 | 0.461395 |
| 484 | 2.984636 | 17456.3232 | 47.826 | 0.362481 |
| 485 | 2.983712 | 17476.2442 | 47.404 | 0.353512 |
| 486 | 2.983574 | 17479.2064 | 47.341 | 0.352179 |
| 487 | 2.999964 | 17127.5897 | 54.878 | 0.512312 |
| 488 | 2.989591 | 17349.6899 | 50.097 | 0.410734 |
| 489 | 2.984327 | 17462.9741 | 47.685 | 0.359485 |
| 490 | 2.983287 | 17485.3991 | 47.21 | 0.349395 |
| 491 | 2.986059 | 17425.6612 | 48.477 | 0.376315 |
| 492 | 2.990142 | 17337.8442 | 50.351 | 0.416117 |
| 493 | 2.994948 | 17234.803 | 52.563 | 0.463121 |
| 494 | 2.994317 | 17248.3161 | 52.272 | 0.45694 |
| 495 | 2.991072 | 17317.8809 | 50.778 | 0.4252 |
| 496 | 2.98405 | 17468.9427 | 47.559 | 0.356797 |
| 497 | 2.988357 | 17376.1984 | 49.531 | 0.398702 |
| 498 | 2.979535 | 17566.4619 | 45.502 | 0.313089 |
| 499 | 2.989632 | 17348.8037 | 50.116 | 0.411136 |
| 500 | 2.985801 | 17431.2223 | 48.359 | 0.373804 |
| 501 | 2.961863 | 17950.9788 | 37.602 | 0.145245 |
| 502 | 2.968664 | 17802.4632 | 40.608 | 0.209116 |
| 503 | 2.969613 | 17781.7791 | 41.031 | 0.218114 |
| 504 | 2.968247 | 17811.5501 | 40.422 | 0.205171 |
| 505 | 2.970163 | 17769.8046 | 41.277 | 0.223334 |
| 506 | 2.97092 | 17753.3425 | 41.615 | 0.230523 |
| 507 | 2.967595 | 17825.7597 | 40.132 | 0.199011 |
| 508 | 2.965831 | 17864.2463 | 39.35 | 0.182385 |
| 509 | 2.978796 | 17582.4523 | 45.166 | 0.30596 |
| 510 | 2.975359 | 17656.9121 | 43.611 | 0.272916 |
| 511 | 2.970729 | 17757.5026 | 41.53 | 0.228705 |
| 512 | 2.979881 | 17558.98 | 45.659 | 0.316428 |
| 513 | 2.973618 | 17694.7062 | 42.826 | 0.256245 |
| 514 | 2.984965 | 17449.2317 | 47.977 | 0.365677 |
| 515 | 2.972362 | 17721.9771 | 42.262 | 0.24426 |

| | | | | |
|---|---|---|---|---|
| 516 | 2.981888 | 17515.6083 | 46.572 | 0.335833 |
| 517 | 2.981458 | 17524.8981 | 46.376 | 0.33167 |
| 518 | 2.976276 | 17637.029 | 44.025 | 0.281714 |
| 519 | 2.974791 | 17669.2276 | 43.355 | 0.267476 |
| 520 | 2.986134 | 17424.0388 | 48.512 | 0.377048 |
| 521 | 2.98016 | 17552.9484 | 45.785 | 0.319122 |
| 522 | 2.987866 | 17386.7608 | 49.306 | 0.393915 |
| 523 | 2.990425 | 17331.772 | 50.481 | 0.418879 |
| 524 | 2.985948 | 17428.0386 | 48.427 | 0.375241 |
| 525 | 2.990366 | 17333.0365 | 50.454 | 0.418303 |
| 526 | 2.989412 | 17353.537 | 50.015 | 0.408986 |
| 527 | 2.993787 | 17259.6537 | 52.028 | 0.451758 |
| 528 | 2.996972 | 17191.5046 | 53.497 | 0.482955 |
| 529 | 2.994643 | 17241.3215 | 52.423 | 0.460139 |
| 530 | 2.986879 | 17407.9934 | 48.853 | 0.384302 |
| 531 | 2.994716 | 17239.7696 | 52.456 | 0.460848 |
| 532 | 2.988052 | 17382.7559 | 49.391 | 0.395729 |
| 533 | 2.985406 | 17439.7223 | 48.179 | 0.369967 |
| 534 | 2.993691 | 17261.7174 | 51.984 | 0.450815 |
| 535 | 2.99324 | 17271.3837 | 51.776 | 0.4464 |
| 536 | 2.986792 | 17409.8717 | 48.813 | 0.383452 |
| 537 | 2.990865 | 17322.3127 | 50.683 | 0.423183 |
| 538 | 2.992935 | 17277.9147 | 51.636 | 0.443419 |
| 539 | 2.994094 | 17253.0907 | 52.169 | 0.454757 |
| 540 | 2.986499 | 17416.1794 | 48.679 | 0.3806 |
| 541 | 2.9881 | 17381.7253 | 49.413 | 0.396197 |
| 542 | 2.997353 | 17183.3452 | 53.673 | 0.486698 |
| 543 | 3.000678 | 17112.3732 | 55.208 | 0.519313 |
| 544 | 2.99523 | 17228.7605 | 52.693 | 0.465886 |
| 545 | 2.999597 | 17135.4296 | 54.709 | 0.508706 |
| 546 | 3.000432 | 17117.6107 | 55.094 | 0.516903 |
| 547 | 2.999067 | 17146.7375 | 54.464 | 0.503508 |
| 548 | 3.003515 | 17051.9197 | 56.519 | 0.547169 |
| 549 | 2.994306 | 17248.5357 | 52.267 | 0.45684 |
| 550 | 2.99753 | 17179.5592 | 53.754 | 0.488435 |
| 551 | 2.993898 | 17257.2735 | 52.079 | 0.452846 |
| 552 | 2.997452 | 17181.2407 | 53.718 | 0.487664 |
| 553 | 3.001042 | 17104.605 | 55.376 | 0.522889 |
| 554 | 2.999767 | 17131.7918 | 54.787 | 0.510379 |
| 555 | 3.003857 | 17044.6431 | 56.677 | 0.550526 |

| | | | | |
|---|---|---|---|---|
| 556 | 2.996649 | 17198.4068 | 53.348 | 0.479791 |
| 557 | 3.005887 | 17001.4642 | 57.615 | 0.570462 |
| 558 | 3.007385 | 16969.6548 | 58.307 | 0.585164 |
| 559 | 2.999821 | 17130.642 | 54.812 | 0.510908 |
| 560 | 2.998401 | 17160.957 | 54.156 | 0.496975 |
| 561 | 3.006079 | 16997.3856 | 57.704 | 0.572346 |
| 562 | 2.99898 | 17148.5954 | 54.424 | 0.502654 |
| 563 | 3.0016 | 17092.7022 | 55.634 | 0.52837 |
| 564 | 3.004772 | 17025.1658 | 57.1 | 0.559516 |
| 565 | 3.003448 | 17053.3559 | 56.488 | 0.546507 |
| 566 | 2.997301 | 17184.4713 | 53.648 | 0.486181 |
| 567 | 2.999774 | 17131.6595 | 54.79 | 0.51044 |
| 568 | 3.003554 | 17051.0925 | 56.537 | 0.547551 |
| 569 | 3.013221 | 16845.9857 | 61.002 | 0.64241 |
| 570 | 3.005334 | 17013.2266 | 57.359 | 0.565029 |
| 571 | 2.999955 | 17127.783 | 54.874 | 0.512223 |
| 572 | 3.008086 | 16954.7663 | 58.631 | 0.592049 |
| 573 | 2.989719 | 17346.9393 | 50.156 | 0.411983 |
| 574 | 2.993484 | 17266.1568 | 51.888 | 0.448787 |
| 575 | 2.993222 | 17271.772 | 51.768 | 0.446223 |
| 576 | 3.003673 | 17048.5501 | 56.592 | 0.548724 |
| 577 | 2.997541 | 17179.335 | 53.759 | 0.488538 |
| 578 | 3.004333 | 17034.5023 | 56.897 | 0.555206 |
| 579 | 3.015717 | 16793.2419 | 62.152 | 0.666851 |
| 580 | 2.996976 | 17191.4077 | 53.499 | 0.483 |
| 581 | 2.99425 | 17249.7407 | 52.241 | 0.456289 |
| 582 | 2.990933 | 17320.8643 | 50.714 | 0.423842 |
| 583 | 2.990816 | 17323.3721 | 50.661 | 0.4227 |
| 584 | 2.99697 | 17191.5351 | 53.496 | 0.482941 |
| 585 | 2.998977 | 17148.6667 | 54.422 | 0.502622 |
| 586 | 3.00506 | 17019.0562 | 57.233 | 0.562337 |
| 587 | 3.003803 | 17045.7847 | 56.652 | 0.55 |
| 588 | 3.00508 | 17018.6304 | 57.242 | 0.562533 |
| 589 | 3.001957 | 17085.1014 | 55.799 | 0.531872 |
| 590 | 3.012613 | 16858.841 | 60.721 | 0.636455 |
| 591 | 3.010124 | 16911.5521 | 59.572 | 0.612045 |
| 592 | 2.993095 | 17274.4804 | 51.709 | 0.444986 |
| 593 | 3.004382 | 17033.4676 | 56.92 | 0.555684 |
| 594 | 3.006289 | 16992.9175 | 57.801 | 0.574411 |
| 595 | 3.006218 | 16994.4372 | 57.768 | 0.573709 |

| | | | | |
|---|---|---|---|---|
| 596 | 3.009397 | 16926.9614 | 59.237 | 0.604913 |
| 597 | 2.999945 | 17128.0017 | 54.869 | 0.512122 |
| 598 | 3.004148 | 17038.4385 | 56.812 | 0.553389 |
| 599 | 3.000057 | 17125.6211 | 54.921 | 0.513217 |
| 600 | 3.006064 | 16997.7098 | 57.697 | 0.572197 |
| 601 | 2.981611 | 17521.5763 | 46.446 | 0.333159 |
| 602 | 2.98436 | 17462.2596 | 47.7 | 0.359807 |
| 603 | 2.994279 | 17249.1229 | 52.255 | 0.456571 |
| 604 | 2.998662 | 17155.3866 | 54.277 | 0.499534 |
| 605 | 2.996774 | 17195.7252 | 53.405 | 0.48102 |
| 606 | 2.994113 | 17252.6668 | 52.178 | 0.454951 |
| 607 | 2.992576 | 17285.6228 | 51.47 | 0.439901 |
| 608 | 2.992276 | 17292.0442 | 51.332 | 0.436972 |
| 609 | 2.993645 | 17262.6931 | 51.963 | 0.450369 |
| 610 | 2.996713 | 17197.0303 | 53.377 | 0.480422 |
| 611 | 2.998379 | 17161.4407 | 54.146 | 0.496753 |
| 612 | 3.00174 | 17089.7145 | 55.699 | 0.529747 |
| 613 | 3.001366 | 17097.6976 | 55.526 | 0.52607 |
| 614 | 3.005914 | 17000.8916 | 57.628 | 0.570727 |
| 615 | 3.0041 | 17039.4683 | 56.789 | 0.552914 |
| 616 | 3.010924 | 16894.5999 | 59.942 | 0.619893 |
| 617 | 2.997306 | 17184.349 | 53.651 | 0.486238 |
| 618 | 3.00468 | 17027.1333 | 57.057 | 0.558607 |
| 619 | 3.001629 | 17092.0874 | 55.647 | 0.528654 |
| 620 | 3.010171 | 16910.5618 | 59.594 | 0.612504 |
| 621 | 3.007937 | 16957.9388 | 58.562 | 0.590582 |
| 622 | 3.001672 | 17091.1677 | 55.667 | 0.529077 |
| 623 | 3.003407 | 17054.2085 | 56.469 | 0.546113 |
| 624 | 3.013468 | 16840.7573 | 61.116 | 0.644833 |
| 625 | 3.014195 | 16825.3846 | 61.451 | 0.651956 |
| 626 | 3.003474 | 17052.7875 | 56.5 | 0.546769 |
| 627 | 3.003172 | 17059.2284 | 56.36 | 0.543798 |
| 628 | 3.008823 | 16939.1313 | 58.972 | 0.599282 |
| 629 | 3.002325 | 17077.2586 | 55.969 | 0.535486 |
| 630 | 3.002681 | 17069.6665 | 56.134 | 0.538985 |
| 631 | 3.013124 | 16848.0329 | 60.957 | 0.641462 |
| 632 | 3.014158 | 16826.1808 | 61.433 | 0.651587 |
| 633 | 3.003649 | 17049.0728 | 56.581 | 0.548483 |
| 634 | 3.005182 | 17016.4505 | 57.289 | 0.56354 |
| 635 | 2.99646 | 17202.4346 | 53.261 | 0.477944 |

| | | | | |
|---|---|---|---|---|
| 636 | 3.024888 | 16600.1992 | 66.361 | 0.756276 |
| 637 | 3.013985 | 16829.8243 | 61.354 | 0.649899 |
| 638 | 3.015628 | 16795.1146 | 62.111 | 0.665983 |
| 639 | 3.01096 | 16893.8374 | 59.958 | 0.620247 |
| 640 | 3.004148 | 17038.4588 | 56.811 | 0.55338 |
| 641 | 3.010325 | 16907.288 | 59.665 | 0.614019 |
| 642 | 3.017238 | 16761.1518 | 62.852 | 0.681724 |
| 643 | 3.015623 | 16795.2203 | 62.109 | 0.665934 |
| 644 | 3.013578 | 16838.4434 | 61.166 | 0.645905 |
| 645 | 3.012255 | 16866.4185 | 60.556 | 0.632945 |
| 646 | 3.012545 | 16860.2837 | 60.69 | 0.635787 |
| 647 | 3.025529 | 16586.7377 | 66.654 | 0.762503 |
| 648 | 3.014605 | 16816.7389 | 61.639 | 0.655962 |
| 649 | 3.028919 | 16515.7085 | 68.199 | 0.79533 |
| 650 | 3.021501 | 16671.3528 | 64.81 | 0.723334 |
| 651 | 3.017927 | 16746.6217 | 63.169 | 0.688458 |
| 652 | 3.010037 | 16913.3863 | 59.532 | 0.611196 |
| 653 | 3.021482 | 16671.764 | 64.801 | 0.723143 |
| 654 | 3.0135 | 16840.0868 | 61.13 | 0.645143 |
| 655 | 3.03036 | 16485.5675 | 68.854 | 0.809241 |
| 656 | 3.018692 | 16730.4956 | 63.521 | 0.695931 |
| 657 | 3.022608 | 16648.0882 | 65.317 | 0.734108 |
| 658 | 3.023877 | 16621.4209 | 65.899 | 0.746455 |
| 659 | 3.020501 | 16692.3963 | 64.352 | 0.713585 |
| 660 | 3.024528 | 16607.7603 | 66.196 | 0.752777 |
| 661 | 3.018111 | 16742.7314 | 63.254 | 0.690261 |
| 662 | 3.027491 | 16545.6235 | 67.549 | 0.781512 |
| 663 | 3.024508 | 16608.1807 | 66.187 | 0.752582 |
| 664 | 3.016035 | 16786.5374 | 62.298 | 0.669958 |
| 665 | 3.017548 | 16754.6049 | 62.995 | 0.684758 |
| 666 | 3.017047 | 16765.1801 | 62.764 | 0.679857 |
| 667 | 3.020457 | 16693.3196 | 64.331 | 0.713157 |
| 668 | 3.027813 | 16538.8712 | 67.695 | 0.784632 |
| 669 | 3.017735 | 16750.6633 | 63.081 | 0.686585 |
| 670 | 3.017167 | 16762.6403 | 62.819 | 0.681034 |
| 671 | 3.020057 | 16701.7311 | 64.148 | 0.70926 |
| 672 | 3.025288 | 16591.7995 | 66.544 | 0.760162 |
| 673 | 3.023218 | 16635.2626 | 65.597 | 0.740047 |
| 674 | 3.031449 | 16462.8024 | 69.348 | 0.819738 |
| 675 | 3.008964 | 16936.1478 | 59.037 | 0.600662 |

| | | | | |
|---|---|---|---|---|
| 676 | 3.038388 | 16318.133 | 72.477 | 0.886217 |
| 677 | 3.013188 | 16846.6916 | 60.986 | 0.642083 |
| 678 | 3.030329 | 16486.2106 | 68.84 | 0.808944 |
| 679 | 3.02809 | 16533.064 | 67.822 | 0.787315 |
| 680 | 3.023253 | 16634.5263 | 65.613 | 0.740388 |
| 681 | 3.028709 | 16520.1101 | 68.103 | 0.793298 |
| 682 | 3.024804 | 16601.9654 | 66.322 | 0.755458 |
| 683 | 3.032064 | 16449.9519 | 69.626 | 0.825659 |
| 684 | 3.013631 | 16837.3192 | 61.191 | 0.646426 |
| 685 | 3.035166 | 16385.2361 | 71.028 | 0.855435 |
| 686 | 3.026637 | 16563.5151 | 67.159 | 0.773242 |
| 687 | 3.016189 | 16783.2864 | 62.369 | 0.671465 |
| 688 | 3.021535 | 16670.6307 | 64.826 | 0.723668 |
| 689 | 3.013451 | 16841.1203 | 61.108 | 0.644665 |
| 690 | 3.027956 | 16535.8751 | 67.761 | 0.786016 |
| 691 | 3.013847 | 16832.7525 | 61.29 | 0.648542 |
| 692 | 3.024629 | 16605.6234 | 66.243 | 0.753766 |
| 693 | 3.018296 | 16738.8365 | 63.339 | 0.692066 |
| 694 | 3.028884 | 16516.4371 | 68.183 | 0.794994 |
| 695 | 3.023766 | 16623.7591 | 65.848 | 0.745372 |
| 696 | 3.01514 | 16805.4213 | 61.886 | 0.661207 |
| 697 | 3.022952 | 16640.8431 | 65.475 | 0.737463 |
| 698 | 3.017704 | 16751.3168 | 63.066 | 0.686282 |
| 699 | 3.014444 | 16820.1243 | 61.566 | 0.654393 |
| 700 | 3.015834 | 16790.7754 | 62.206 | 0.667994 |
| 701 | 3.016459 | 16777.5702 | 62.494 | 0.674114 |
| 702 | 3.023043 | 16638.9394 | 65.517 | 0.738344 |
| 703 | 3.028039 | 16534.1474 | 67.798 | 0.786814 |
| 704 | 3.025101 | 16595.7314 | 66.458 | 0.758343 |
| 705 | 3.023463 | 16630.1239 | 65.709 | 0.742426 |
| 706 | 3.021892 | 16663.1398 | 64.989 | 0.727138 |
| 707 | 3.027083 | 16554.1708 | 67.363 | 0.777562 |
| 708 | 3.026521 | 16565.9391 | 67.107 | 0.772122 |
| 709 | 3.026688 | 16562.4406 | 67.183 | 0.773739 |
| 710 | 3.032188 | 16447.3576 | 69.683 | 0.826854 |
| 711 | 3.026752 | 16561.1113 | 67.212 | 0.774354 |
| 712 | 3.035754 | 16372.9784 | 71.293 | 0.861066 |
| 713 | 3.028633 | 16521.7072 | 68.069 | 0.79256 |
| 714 | 3.029322 | 16507.2763 | 68.382 | 0.799223 |
| 715 | 3.033659 | 16416.6497 | 70.348 | 0.840992 |

| | | | | |
|---|---|---|---|---|
| 716 | 3.027985 | 16535.2659 | 67.774 | 0.786298 |
| 717 | 3.040555 | 16273.1069 | 73.446 | 0.906811 |
| 718 | 3.027006 | 16555.7747 | 67.328 | 0.77682 |
| 719 | 3.039605 | 16292.8469 | 73.021 | 0.897789 |
| 720 | 3.033552 | 16418.8883 | 70.3 | 0.839962 |
| 721 | 3.043362 | 16214.8557 | 74.696 | 0.933372 |
| 722 | 3.039708 | 16290.6915 | 73.068 | 0.898774 |
| 723 | 3.044539 | 16190.467 | 75.218 | 0.944463 |
| 724 | 3.036344 | 16360.6858 | 71.559 | 0.866709 |
| 725 | 3.033487 | 16420.2564 | 70.27 | 0.839332 |
| 726 | 3.032463 | 16441.6218 | 69.807 | 0.829496 |
| 727 | 3.040707 | 16269.9378 | 73.514 | 0.908258 |
| 728 | 3.037454 | 16337.5786 | 72.057 | 0.877307 |
| 729 | 3.037358 | 16339.5685 | 72.014 | 0.876395 |
| 730 | 3.031331 | 16465.2685 | 69.294 | 0.818601 |
| 731 | 3.039572 | 16293.5208 | 73.007 | 0.89748 |
| 732 | 3.035732 | 16373.4404 | 71.283 | 0.860853 |
| 733 | 3.04244 | 16233.9707 | 74.286 | 0.924667 |
| 734 | 3.040774 | 16268.5613 | 73.544 | 0.908887 |
| 735 | 3.034694 | 16395.0625 | 70.815 | 0.850919 |
| 736 | 3.043311 | 16215.9085 | 74.674 | 0.932892 |
| 737 | 3.045712 | 16166.1811 | 75.737 | 0.955488 |
| 738 | 3.04353 | 16211.3662 | 74.771 | 0.93496 |
| 739 | 3.043479 | 16212.4288 | 74.748 | 0.934476 |
| 740 | 3.043319 | 16215.7454 | 74.677 | 0.932967 |
| 741 | 3.052128 | 16033.6475 | 78.552 | 1.015296 |
| 742 | 3.04414 | 16198.7262 | 75.041 | 0.940709 |
| 743 | 3.053387 | 16007.7046 | 79.099 | 1.026925 |
| 744 | 3.047599 | 16127.1279 | 76.57 | 0.973177 |
| 745 | 3.048914 | 16099.9603 | 77.147 | 0.985451 |
| 746 | 3.046062 | 16158.9271 | 75.892 | 0.958778 |
| 747 | 3.052252 | 16031.0773 | 78.606 | 1.016449 |
| 748 | 3.040426 | 16275.7861 | 73.388 | 0.905587 |
| 749 | 3.036531 | 16356.7928 | 71.643 | 0.868495 |
| 750 | 3.051806 | 16040.2681 | 78.412 | 1.012324 |
| 751 | 3.054247 | 15989.9827 | 79.473 | 1.034853 |
| 752 | 3.055296 | 15968.4011 | 79.926 | 1.04449 |
| 753 | 3.0578 | 15916.9428 | 81.004 | 1.067385 |
| 754 | 3.060923 | 15852.8713 | 82.337 | 1.095719 |
| 755 | 3.053429 | 16006.8305 | 79.118 | 1.027317 |

| | | | | |
|---|---|---|---|---|
| 756 | 3.04822 | 16114.3049 | 76.842 | 0.978974 |
| 757 | 3.068905 | 15689.7025 | 85.688 | 1.166921 |
| 758 | 3.052999 | 16015.6802 | 78.931 | 1.023353 |
| 759 | 3.067527 | 15717.8211 | 85.116 | 1.154755 |
| 760 | 3.072675 | 15612.9307 | 87.241 | 1.199896 |
| 761 | 3.052366 | 16028.7285 | 78.656 | 1.017503 |
| 762 | 3.046755 | 16144.582 | 76.198 | 0.965278 |
| 763 | 3.051271 | 16051.3006 | 78.179 | 1.007368 |
| 764 | 3.051921 | 16037.9038 | 78.462 | 1.013386 |
| 765 | 3.060358 | 15864.459 | 82.097 | 1.090609 |
| 766 | 3.048926 | 16099.7141 | 77.153 | 0.985562 |
| 767 | 3.047061 | 16138.2649 | 76.332 | 0.968138 |
| 768 | 3.075572 | 15554.0608 | 88.419 | 1.224933 |
| 769 | 3.04653 | 16149.2479 | 76.098 | 0.963164 |
| 770 | 3.064634 | 15776.9126 | 83.906 | 1.129044 |
| 771 | 3.058572 | 15901.0876 | 81.335 | 1.074414 |
| 772 | 3.059925 | 15873.3273 | 81.912 | 1.086694 |
| 773 | 3.052447 | 16027.0578 | 78.691 | 1.018253 |
| 774 | 3.049349 | 16090.9835 | 77.338 | 0.989501 |
| 775 | 3.054081 | 15993.4085 | 79.4 | 1.033322 |
| 776 | 3.059265 | 15886.8595 | 81.631 | 1.080713 |
| 777 | 3.064295 | 15783.84 | 83.763 | 1.126017 |
| 778 | 3.070805 | 15650.9944 | 86.473 | 1.183592 |
| 779 | 3.056061 | 15952.6719 | 80.256 | 1.051501 |
| 780 | 3.048248 | 16113.7235 | 76.855 | 0.979236 |
| 781 | 3.062486 | 15820.8431 | 83 | 1.109807 |
| 782 | 3.053089 | 16013.8433 | 78.97 | 1.024176 |
| 783 | 3.061892 | 15833.0147 | 82.749 | 1.104459 |
| 784 | 3.051887 | 16038.6116 | 78.447 | 1.013068 |
| 785 | 3.066103 | 15746.8756 | 84.522 | 1.142138 |
| 786 | 3.062891 | 15812.5655 | 83.171 | 1.113439 |
| 787 | 3.055984 | 15954.2651 | 80.223 | 1.050791 |
| 788 | 3.066763 | 15733.4135 | 84.797 | 1.14799 |
| 789 | 3.065959 | 15749.8172 | 84.462 | 1.140858 |
| 790 | 3.058005 | 15912.7224 | 81.092 | 1.069257 |
| 791 | 3.052774 | 16020.3172 | 78.833 | 1.021275 |
| 792 | 3.068509 | 15697.7776 | 85.524 | 1.163432 |
| 793 | 3.05798 | 15913.2511 | 81.081 | 1.069022 |
| 794 | 3.064831 | 15772.8865 | 83.988 | 1.130802 |
| 795 | 3.060784 | 15855.7202 | 82.278 | 1.094463 |

| | | | | |
|---|---|---|---|---|
| 796 | 3.049191 | 16094.2283 | 77.269 | 0.988037 |
| 797 | 3.068425 | 15699.4891 | 85.489 | 1.162692 |
| 798 | 3.054642 | 15981.8517 | 79.644 | 1.038486 |
| 799 | 3.049353 | 16090.89 | 77.34 | 0.989543 |
| 800 | 3.072039 | 15625.8754 | 86.98 | 1.194361 |
| 801 | 3.040693 | 16270.2398 | 73.508 | 0.90812 |
| 802 | 3.051986 | 16036.562 | 78.49 | 1.013988 |
| 803 | 3.054573 | 15983.2795 | 79.613 | 1.037848 |
| 804 | 3.051042 | 16056.0412 | 78.079 | 1.005236 |
| 805 | 3.056636 | 15940.8606 | 80.503 | 1.056758 |
| 806 | 3.059943 | 15872.9557 | 81.92 | 1.086858 |
| 807 | 3.05642 | 15945.2856 | 80.411 | 1.054789 |
| 808 | 3.049867 | 16080.2866 | 77.565 | 0.994323 |
| 809 | 3.05016 | 16074.2284 | 77.693 | 0.997052 |
| 810 | 3.048316 | 16112.3095 | 76.885 | 0.979875 |
| 811 | 3.052699 | 16021.8647 | 78.801 | 1.020581 |
| 812 | 3.052104 | 16034.1439 | 78.542 | 1.015073 |
| 813 | 3.047656 | 16125.9548 | 76.595 | 0.973707 |
| 814 | 3.052998 | 16015.7195 | 78.93 | 1.023335 |
| 815 | 3.051658 | 16043.3306 | 78.347 | 1.010949 |
| 816 | 3.052938 | 16016.9425 | 78.905 | 1.022787 |
| 817 | 3.062348 | 15823.6793 | 82.942 | 1.108561 |
| 818 | 3.048078 | 16117.2318 | 76.78 | 0.977651 |
| 819 | 3.054417 | 15986.4837 | 79.546 | 1.036417 |
| 820 | 3.050959 | 16057.7478 | 78.042 | 1.004469 |
| 821 | 3.055119 | 15972.0551 | 79.849 | 1.04286 |
| 822 | 3.046903 | 16141.5342 | 76.263 | 0.966658 |
| 823 | 3.052849 | 16018.7895 | 78.866 | 1.02196 |
| 824 | 3.050319 | 16070.9562 | 77.763 | 0.998525 |
| 825 | 3.060003 | 15871.7334 | 81.946 | 1.087398 |
| 826 | 3.05692 | 15935.0203 | 80.626 | 1.059355 |
| 827 | 3.062399 | 15822.6346 | 82.963 | 1.10902 |
| 828 | 3.052307 | 16029.9569 | 78.63 | 1.016952 |
| 829 | 3.066185 | 15745.2151 | 84.556 | 1.14286 |
| 830 | 3.070353 | 15660.2056 | 86.287 | 1.179633 |
| 831 | 3.056704 | 15939.4543 | 80.533 | 1.057383 |
| 832 | 3.064918 | 15771.1076 | 84.025 | 1.131578 |
| 833 | 3.06787 | 15710.8158 | 85.259 | 1.15779 |
| 834 | 3.071679 | 15633.2062 | 86.832 | 1.191222 |
| 835 | 3.067765 | 15712.961 | 85.215 | 1.156861 |

| | | | | |
|---|---|---|---|---|
| 836 | 3.060563 | 15860.2457 | 82.184 | 1.092468 |
| 837 | 3.079074 | 15483.0678 | 89.826 | 1.25482 |
| 838 | 3.074476 | 15576.3178 | 87.975 | 1.215493 |
| 839 | 3.070264 | 15662.0074 | 86.25 | 1.178858 |
| 840 | 3.070469 | 15657.8261 | 86.335 | 1.180656 |
| 841 | 3.069506 | 15677.4504 | 85.937 | 1.172207 |
| 842 | 3.068143 | 15705.2419 | 85.372 | 1.160203 |
| 843 | 3.066504 | 15738.6857 | 84.69 | 1.145699 |
| 844 | 3.052009 | 16036.1 | 78.5 | 1.014195 |
| 845 | 3.06884 | 15691.0247 | 85.662 | 1.16635 |
| 846 | 3.066038 | 15748.2197 | 84.494 | 1.141553 |
| 847 | 3.079165 | 15481.2088 | 89.862 | 1.255598 |
| 848 | 3.070011 | 15667.1558 | 86.146 | 1.176642 |
| 849 | 3.072212 | 15622.3586 | 87.051 | 1.195866 |
| 850 | 3.071292 | 15641.0677 | 86.674 | 1.187852 |
| 851 | 3.077628 | 15512.3543 | 89.247 | 1.242532 |
| 852 | 3.070754 | 15652.0334 | 86.452 | 1.183145 |
| 853 | 3.080668 | 15450.7937 | 90.46 | 1.268292 |
| 854 | 3.075195 | 15561.7282 | 88.266 | 1.221685 |
| 855 | 3.074504 | 15575.7559 | 87.986 | 1.215732 |
| 856 | 3.082535 | 15413.0455 | 91.197 | 1.283954 |
| 857 | 3.081653 | 15430.8774 | 90.849 | 1.276568 |
| 858 | 3.058169 | 15909.3593 | 81.162 | 1.070748 |
| 859 | 3.085594 | 15351.2816 | 92.392 | 1.309352 |
| 860 | 3.073015 | 15606.0264 | 87.379 | 1.202844 |
| 861 | 3.091432 | 15233.7724 | 94.628 | 1.356852 |
| 862 | 3.077507 | 15514.8098 | 89.199 | 1.241499 |
| 863 | 3.08851 | 15292.5339 | 93.516 | 1.333238 |
| 864 | 3.079703 | 15470.3222 | 90.076 | 1.260149 |
| 865 | 3.081418 | 15435.6215 | 90.757 | 1.274599 |
| 866 | 3.085222 | 15358.7885 | 92.248 | 1.306281 |
| 867 | 3.074646 | 15572.8646 | 88.044 | 1.21696 |
| 868 | 3.089546 | 15271.6777 | 93.912 | 1.341652 |
| 869 | 3.078141 | 15501.9545 | 89.453 | 1.246902 |
| 870 | 3.081719 | 15429.5276 | 90.876 | 1.277128 |
| 871 | 3.08703 | 15322.3286 | 92.948 | 1.321158 |
| 872 | 3.072518 | 15616.1215 | 87.176 | 1.198533 |
| 873 | 3.069206 | 15683.5783 | 85.813 | 1.169564 |
| 874 | 3.070382 | 15659.6083 | 86.299 | 1.17989 |
| 875 | 3.083114 | 15401.335 | 91.425 | 1.288791 |

| | | | | |
|---|---|---|---|---|
| 876 | 3.075204 | 15561.5442 | 88.27 | 1.221763 |
| 877 | 3.085021 | 15362.8433 | 92.169 | 1.30462 |
| 878 | 3.081915 | 15425.575 | 90.953 | 1.278767 |
| 879 | 3.081118 | 15441.6876 | 90.638 | 1.272079 |
| 880 | 3.084573 | 15371.8784 | 91.995 | 1.300914 |
| 881 | 3.097361 | 15114.8737 | 96.835 | 1.403742 |
| 882 | 3.075171 | 15562.222 | 88.256 | 1.221475 |
| 883 | 3.081581 | 15432.3237 | 90.821 | 1.275968 |
| 884 | 3.079339 | 15477.6937 | 89.931 | 1.257068 |
| 885 | 3.078307 | 15498.5966 | 89.519 | 1.248312 |
| 886 | 3.082057 | 15422.6928 | 91.009 | 1.279961 |
| 887 | 3.075458 | 15556.3841 | 88.373 | 1.223949 |
| 888 | 3.085777 | 15347.6033 | 92.463 | 1.310856 |
| 889 | 3.074644 | 15572.9179 | 88.043 | 1.216937 |
| 890 | 3.075687 | 15551.7378 | 88.465 | 1.225916 |
| 891 | 3.080854 | 15447.0265 | 90.533 | 1.269859 |
| 892 | 3.078705 | 15490.5294 | 89.679 | 1.251695 |
| 893 | 3.083454 | 15394.4678 | 91.558 | 1.291623 |
| 894 | 3.078491 | 15494.8767 | 89.593 | 1.249872 |
| 895 | 3.077448 | 15516.0183 | 89.175 | 1.240991 |
| 896 | 3.076932 | 15526.4663 | 88.968 | 1.23659 |
| 897 | 3.077205 | 15520.9445 | 89.077 | 1.238917 |
| 898 | 3.085822 | 15346.6898 | 92.481 | 1.311229 |
| 899 | 3.085518 | 15352.8251 | 92.363 | 1.308721 |
| 900 | 3.077733 | 15510.2325 | 89.289 | 1.243424 |
| 901 | 3.059906 | 15873.7233 | 81.904 | 1.086519 |
| 902 | 3.057278 | 15927.6675 | 80.78 | 1.062623 |
| 903 | 3.060508 | 15861.3747 | 82.161 | 1.09197 |
| 904 | 3.057344 | 15926.2961 | 80.808 | 1.063232 |
| 905 | 3.059236 | 15887.4661 | 81.618 | 1.080444 |
| 906 | 3.060133 | 15869.0737 | 82.001 | 1.088572 |
| 907 | 3.058609 | 15900.3193 | 81.351 | 1.074755 |
| 908 | 3.065766 | 15753.7623 | 84.381 | 1.13914 |
| 909 | 3.054332 | 15988.2356 | 79.509 | 1.035634 |
| 910 | 3.060851 | 15854.347 | 82.307 | 1.095068 |
| 911 | 3.061003 | 15851.2345 | 82.371 | 1.09644 |
| 912 | 3.067693 | 15714.4204 | 85.185 | 1.156229 |
| 913 | 3.070103 | 15665.2955 | 86.184 | 1.177443 |
| 914 | 3.062678 | 15816.9139 | 83.081 | 1.111532 |
| 915 | 3.067245 | 15723.5773 | 84.998 | 1.152259 |

| | | | | |
|---|---|---|---|---|
| 916 | 3.064284 | 15784.0545 | 83.759 | 1.125924 |
| 917 | 3.071534 | 15636.1467 | 86.773 | 1.189962 |
| 918 | 3.066681 | 15735.088 | 84.763 | 1.147263 |
| 919 | 3.068729 | 15693.2998 | 85.615 | 1.165367 |
| 920 | 3.077388 | 15517.2365 | 89.151 | 1.240478 |
| 921 | 3.072964 | 15607.0591 | 87.359 | 1.202403 |
| 922 | 3.070835 | 15650.3779 | 86.486 | 1.183856 |
| 923 | 3.069336 | 15680.9103 | 85.867 | 1.170715 |
| 924 | 3.069498 | 15677.6254 | 85.934 | 1.172132 |
| 925 | 3.079093 | 15482.6864 | 89.833 | 1.25498 |
| 926 | 3.073598 | 15594.1706 | 87.617 | 1.207899 |
| 927 | 3.071591 | 15634.9967 | 86.796 | 1.190455 |
| 928 | 3.065571 | 15757.7567 | 84.299 | 1.1374 |
| 929 | 3.07384 | 15589.257 | 87.716 | 1.209991 |
| 930 | 3.067718 | 15713.9145 | 85.196 | 1.156448 |
| 931 | 3.074731 | 15571.1552 | 88.078 | 1.217686 |
| 932 | 3.082556 | 15412.6071 | 91.205 | 1.284135 |
| 933 | 3.071103 | 15644.9215 | 86.596 | 1.186199 |
| 934 | 3.073086 | 15604.5721 | 87.408 | 1.203465 |
| 935 | 3.075618 | 15553.1461 | 88.437 | 1.22532 |
| 936 | 3.083264 | 15398.3154 | 91.483 | 1.290037 |
| 937 | 3.084435 | 15374.6645 | 91.941 | 1.299771 |
| 938 | 3.072338 | 15619.7928 | 87.102 | 1.196963 |
| 939 | 3.067906 | 15710.0813 | 85.274 | 1.158109 |
| 940 | 3.075561 | 15554.2883 | 88.414 | 1.224836 |
| 941 | 3.080492 | 15454.3636 | 90.39 | 1.266805 |
| 942 | 3.078361 | 15497.5047 | 89.541 | 1.24877 |
| 943 | 3.087665 | 15309.5429 | 93.192 | 1.32635 |
| 944 | 3.079424 | 15475.975 | 89.965 | 1.257787 |
| 945 | 3.076041 | 15544.5567 | 88.608 | 1.228954 |
| 946 | 3.083577 | 15391.9881 | 91.606 | 1.292645 |
| 947 | 3.082745 | 15408.8011 | 91.279 | 1.285708 |
| 948 | 3.085854 | 15346.0407 | 92.493 | 1.311494 |
| 949 | 3.095103 | 15160.1027 | 96.002 | 1.38605 |
| 950 | 3.094953 | 15163.0993 | 95.947 | 1.384871 |
| 951 | 3.086173 | 15339.6087 | 92.617 | 1.31412 |
| 952 | 3.077409 | 15516.8062 | 89.159 | 1.240659 |
| 953 | 3.091124 | 15239.9526 | 94.512 | 1.354382 |
| 954 | 3.089965 | 15263.2567 | 94.072 | 1.345039 |
| 955 | 3.08621 | 15338.8541 | 92.631 | 1.314428 |

| | | | | |
|---|---|---|---|---|
| 956 | 3.081082 | 15442.4206 | 90.624 | 1.271775 |
| 957 | 3.084441 | 15374.5442 | 91.944 | 1.29982 |
| 958 | 3.085325 | 15356.7156 | 92.288 | 1.307129 |
| 959 | 3.085738 | 15348.3774 | 92.448 | 1.310539 |
| 960 | 3.086481 | 15333.4039 | 92.736 | 1.31665 |
| 961 | 3.081161 | 15440.8195 | 90.655 | 1.27244 |
| 962 | 3.09349 | 15192.44 | 95.402 | 1.373291 |
| 963 | 3.095741 | 15147.3024 | 96.239 | 1.391075 |
| 964 | 3.091916 | 15224.0447 | 94.811 | 1.360734 |
| 965 | 3.084193 | 15379.5636 | 91.847 | 1.297758 |
| 966 | 3.090368 | 15255.1448 | 94.225 | 1.348296 |
| 967 | 3.100346 | 15055.1758 | 97.921 | 1.426811 |
| 968 | 3.090695 | 15248.5734 | 94.349 | 1.350931 |
| 969 | 3.09953 | 15071.4833 | 97.626 | 1.420542 |
| 970 | 3.092193 | 15218.4854 | 94.915 | 1.362949 |
| 971 | 3.082702 | 15409.6634 | 91.263 | 1.285352 |
| 972 | 3.091115 | 15240.1251 | 94.508 | 1.354313 |
| 973 | 3.084309 | 15377.215 | 91.892 | 1.298723 |
| 974 | 3.092304 | 15216.2542 | 94.957 | 1.363837 |
| 975 | 3.086103 | 15341.0219 | 92.589 | 1.313543 |
| 976 | 3.09009 | 15260.7491 | 94.119 | 1.346046 |
| 977 | 3.091227 | 15237.8924 | 94.55 | 1.355206 |
| 978 | 3.096068 | 15140.7633 | 96.359 | 1.393637 |
| 979 | 3.096606 | 15129.9857 | 96.558 | 1.397851 |
| 980 | 3.091253 | 15237.3606 | 94.56 | 1.355419 |
| 981 | 3.105491 | 14952.5637 | 99.75 | 1.465682 |
| 982 | 3.090526 | 15251.9764 | 94.285 | 1.349567 |
| 983 | 3.090765 | 15247.1788 | 94.376 | 1.35149 |
| 984 | 3.09215 | 15219.3376 | 94.899 | 1.36261 |
| 985 | 3.090145 | 15259.6272 | 94.141 | 1.346497 |
| 986 | 3.080671 | 15450.7359 | 90.461 | 1.268316 |
| 987 | 3.094211 | 15177.9766 | 95.671 | 1.379009 |
| 988 | 3.096919 | 15123.7128 | 96.673 | 1.400299 |
| 989 | 3.098556 | 15090.9655 | 97.272 | 1.41302 |
| 990 | 3.088694 | 15288.8337 | 93.587 | 1.334733 |
| 991 | 3.091473 | 15232.9485 | 94.643 | 1.357182 |
| 992 | 3.102622 | 15009.7429 | 98.737 | 1.444146 |
| 993 | 3.100132 | 15059.4579 | 97.843 | 1.425167 |
| 994 | 3.090865 | 15245.1564 | 94.414 | 1.3523 |
| 995 | 3.096111 | 15139.8989 | 96.375 | 1.393975 |

| | | | | |
|---|---|---|---|---|
| 996 | 3.093916 | 15183.8922 | 95.561 | 1.376672 |
| 997 | 3.106275 | 14936.9592 | 100.02 | 1.471504 |
| 998 | 3.094034 | 15181.52 | 95.605 | 1.37761 |
| 999 | 3.092164 | 15219.0647 | 94.904 | 1.362718 |
| 1000 | 3.098257 | 15096.9456 | 97.163 | 1.410704 |