ECE/COE 1896

Senior Design

Tendon-Driven Soft Hand Exoskeleton for Self-Training and Rehabilitation of Dysfunctional Hands

Final Report – Team #4

Prepared By:

Daniel Braunstein

Tyler Sheetz

Bryan Hess

Michael Etter

Table of Contents

Table of Figures

Table of Tables

1. Introduction

The goal of this project was to construct a wearable device that could actuate the fingers of a patient suffering from weakness or paralysis in the hand to assist them in strengthening their muscles through physical therapy. It was necessary for the device to be unobtrusive for the patient and be able to fully actuate their fingers through their entire natural range of motion. Other qualities which were desired by the patient were for the glove to be lightweight and portable since they might lack strength in their hand or arm, for it to be controlled wirelessly by a mobile phone app by the patient or the physical therapist, and battery power. Design goals created to set the device apart from other market solutions were making it as inexpensive as possible, the ability to finely control the motion of individual fingers, and for the physical therapist to have the ability to design custom exercises within the mobile app.

To accomplish these goals, a custom PCB was designed to interface with each of the hardware components. Flex sensors were fabricated by hand to reduce the device's cost and tested to ensure they met basic functionality that would be found in pre-built flex sensors available on the market. A custom exoskeleton was designed, and 3D printed to actuate the fingers in concert with servo motors, and to contain each of the hardware components discretely on the wearer's forearm. A nontrivial algorithm was written to control the movement of the fingers based on exercises and other parameters selected in a custom-designed mobile app with GUI.

Once completed, the device performed beyond the minimum standards set by the team. Any number of custom exercises could be designed and performed via the app GUI, and the servo motors precisely actuated the fingers to the positions and speeds set by the app. The hand-fabricated flex sensors performed better than expected and were able to determine the finger position accurately enough to act as a safety feature and a metric of the wearer's performance during the exercises. They were also integral to tracking performance during the non-assistive training mode, where the servo motors were disengaged, and the user attempts the exercises unaided. The prototype exceeded the capabilities of similar devices available on the market by offering greater customization, portability, longevity, and affordability. However, there was still room for improvement by the end of the semester. The exoskeleton was composed of low-quality material, and a greater focus on the mechanics would have yielded an even more discrete and lightweight design. The flex sensors would have benefitted from a standardized manufacturing process beyond what their human fabricator could offer. The PCB could have been greatly reduced in size if a machine were able to solder SMD components, instead of a human who needed to solder larger through-hole pieces. Finally, as the device is a prototype, the aesthetics could have been greatly enhanced by a manufacturing and artistic design process to make it more attractive to consumers.

2. Background

Treating hand dysfunction is a problem which requires knowledge beyond the electrical and computer engineering skills we've developed during our undergraduate studies. This project requires a basic understanding of physical rehabilitation, biomechanical engineering, anatomy, and biomechanics. For the prototype to achieve its design goals, these disciplines must work in tandem with the electrical and computer engineering design. Current treatments for hand weakness or paralysis employ physiotherapy and occupational therapy [1]. These symptoms are common amongst a variety of conditions, and present differently for different patients, so treatment plans are individually tailored to satisfy the needs of the individual patient. The greatest barrier to a comprehensive solution is the anatomical complexity of the human hand, which contains over 30 muscles involved in basic motor control. Assisted movements have proven effective treatment for such conditions by strengthening the patient's muscles and tendons and are facilitated by commercial and clinical rehabilitation gloves.

Commercially, a wide variety of mechanical and electrical solutions are available to patients. For mechanical solutions, there are puppeteering products that allow the user to use their free hand to aid in controlling the injured hand. This system has limited rehab mobility as its fully mechanical nature only allows for a very specific range of motion, and fine control of individual fingers is difficult [2]. As for electrical solutions, tendon driven soft exoskeletons allow for automated control over the hand. These systems use air pressure to extend and retract tubes that are mounted on the backside of the hand, and also come with basic training such as passively gripping an object and moving one to two fingers at a time [3]. These systems lack more precise control over the finger's movements due to the pneumatic control. The current market solutions offer some training regimens; however, they are lacking more complex rehabilitation programs. In contrast, our design uses a servo motor system to actuate the fingers more precisely than can be achieved with pneumatics.

Figure 1: electro-pneumatic rehabilitation glove [3]

Servo motors are compact and energy efficient rotary or linear actuators. They can be driven either by AC or DC. Our device will be utilizing DC servo motors as our application does not require AC voltage nor a significant amount of torque. Precision is the key advantage that servo motors provide since they are commanded to turn to a specific angle. If a heavy workload is placed on a servo motor, the motor will continue to work at the same pace. Therefore, if something was to oppose the motion of the tendon, the servo motor would automatically generate torque to stabilize. Servos allow more precise control of the exoskeleton's tendons compared to pneumatics.

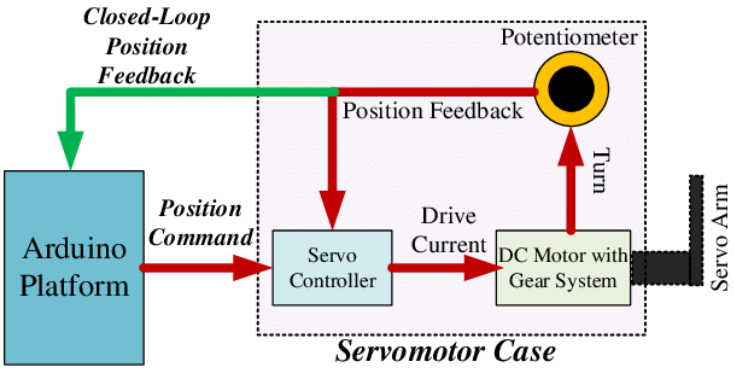

Figure 2: Block Diagram of DC Servo Motor [4]

3. System Requirements

3.1 General

- The system as a whole must aid in performing basic hand motor functions.
- The system will operate in the 3-5V range, powered via a battery.
- The battery life of the device is 550mAH with 9hrs of idle usage and 30 minutes of active usage.
- The device must be lightweight and unobtrusive.

These requirements all pertain to the portability of the device and the fact that it is wearable technology. Since the patient suffers from hand weakness or paralysis, the device must be lightweight enough for easy operation, and have sufficient battery life to perform the exercises independent of external power sources.

3.2 Wearability

This section details the requirements for the device exoskeleton. Given the biomechanical nature of this problem, it requires a substantial mechanism to interact with the human hand.

3.2.1 Exoskeleton

- The exoskeletal components fit snugly and completely over the glove and don't interfere with movement of the user's fingers.

This feature is required by the user since the goal of the project is to create a device that strengthens the hand and fingers through motor therapy. If the fingers can't move due to interaction between exoskeleton components, then the device fails to accomplish its goal. Full coverage of the hand and fingers is required to enable complete range of motion.

3.2.2 Portability

- The device is lightweight and can be easily worn by the user without strain.
- The device is entirely contained on the glove and forearm module. All necessary hardware is discreetly contained on the user's hand and arm.
- The device is battery-powered, and the battery can be changed or charged by the physical therapist.
- The device communicates wirelessly with a mobile phone.

These features are desired by the user. They are unnecessary for the base functionality but make it significantly more convenient and attractive.

3.3 Hardware

3.3.1 Servo Motors

- The motors provide enough rotation to completely curl the user's fingers.
- The motors are connected to the exoskeleton in such a way that they curl the user's fingers when they rotate.

These features are required by the user to ensure basic functionality of the device.

3.3.2 Flex Sensors

- The sensors provide a voltage reading that can be converted to an angle reading by the software based upon experimental data.
- The sensors can interface with the PCB and the microcontroller.

These features are required by the user to provide basic safety functionality. The flex sensors are used to track the position of the fingers to prevent the motors from rotating too far and injuring the user.

3.3.3 Embedded Hardware

- All components except the motors and sensors are integrated onto the PCB.
- The device has a Bluetooth module that is capable of interfacing with an android cell phone to initialize training routines.
- The device utilizes a programmable microcontroller that is capable of setting motor position and reading sensor data to determine finger position.

The first two features are desired by the user. The device could have components spread across multiple boards or connected some other way. Integrating all components onto a single PCB is a best practice, but not technically required. The Bluetooth module communicating with a cell phone is also not required. The device could communicate using some other wireless technology, with a wired connection, or even with a computer instead of a cell phone. Bluetooth communication and a cell phone are most convenient for the user. The last feature is required to ensure basic device functionality.

3.4 Software

3.4.1 Mobile Application Compatibility

- There will be a mobile application for exercise variability
- The application will be supported for Android cell phones.
- This application will come with an interactive GUI
- The user GUI must allow for the execution of preset and custom exercises.
- The user GUI must allow for the creation of custom exercises.
- Exercises have variable speeds for completion

In conjunction with the physical therapy device, a mobile application is required to perform exercises utilizing the device. A mobile app is needed to allow for complex calculations that are resource intensive. This mobile app must be on android phones as it is open source and allows for more libraries and programmability. Multiple GUIs are needed, one for exercise setting, and one for exercise execution. Custom exercises and varying speeds are both necessary in the build as they allow for different patients with different needs to fully utilize the device.

3.4.2 Bluetooth Compatibility

- The Bluetooth connection will be established with UART at 9600 baud with HW flow control.
- Bluetooth connections must hold over a distance of 30cm from mobile phones.
- Bluetooth cannot have packet loss over 2% to ensure the correct data is being sent

Bluetooth is needed in the system in order to complete more complex computations not supported by the microcontroller. These measurements were chosen in line with Bluetooth device standards outlined by various Bluetooth integrated devices.

4. Design Constraints: Standards and Impacts

This section provides the design constraints facing the implementation and development of the rehabilitation device.

4.1 Time

The entirety of this project is to be completed in the span of a semester. This gives the group roughly 3 months to complete the project and have it ready for the design expo. Time is one of the biggest constraints regarding this project.

4.2 Manpower

A common problem in the industry today is manpower. This section is moreover a constraint imposed by the manpower of the businesses in which the purchased materials come from. With lower manpower in the overall workforce, shipping speeds have plummeted; demand far exceeds supply.

4.3 Budget

The scope of this project only allows a $200 budget. With this in mind, the parts available for purchase are limited. This is a common occurrence in any project; though very manageable, cost remains a constant constraint.

4.4 RoHS Compliant

If the rehabilitation device is to be marketable, it would have to follow RoHS (Restriction of Hazardous Substances) standards. This means that every device purchased would have to contain safe levels of lead, cadmium, and other harmful substances. To adhere to this, all of the parts that are purchased for the device must already follow these standards. This would ensure that the overall product is compliant with the RoHS standards [5].

4.5 IEC 60601-1-11:2015 Compliant

To pass FDA testing, the device will need to meet medical standards. The ISO has standards on how to craft medical electrical equipment. This includes safeties in place so a patient will not be injured by a device. Likewise, it has requirements for operational use so that it may be used by both trained personnel as well as patients if deemed applicable [6].

4.6 CITI Biosensors Compliant

The Collaborative Institutional Training Initiative (CITI Program) has a standard list of practices to employ when using biosensors in biomedical human subject research. This includes wearable device safety concerns and Bluetooth reliability. Having CITI certification in biosensors will ensure that all practices will be followed, and medical safety considerations will be taken into account [7].

4.7 Weight

The device should weigh a minimal amount when it comes to the force on the wearer. It would be a hindrance if the device applied too much force on the wearer as it is meant to assist in rehabilitation. Many going through this process are weak and in the process of getting stronger; therefore, it is consequential that the device is not heavy.

4.8 Size

The device must be as small as possible. It needs to fit comfortably on the user and cannot be too much of a burden to take on and off.

4.9 JLC PCB

The PCB printing company has constraints regarding PCB creation. There are restrictions on trace lengths, maximum board dimensions, drill/hole sizes, annular ring sizes, and solder mask dimensions. These constraints will need to be kept in mind throughout the entire PCB creation process as the PCB cannot be created otherwise.

5. Summary of Design

5.1 Design Alternatives

Before describing the final prototype in detail, an alternative design that was considered during the planning stage will be examined. This section will explore alternative components or design choices that were considered across various modules and explain why they were ultimately rejected in favor of those presented in the following sections.

5.1.1 Flex Sensors

Custom flex sensors were fabricated for the device; however, flex sensors are a discrete item that may be easily purchased from several online suppliers. Purchasing sensors would have eliminated the time commitment required to fabricate and test the custom sensors but would have been substantially more expensive [8]. Since Design Constraint 4.3 encourages the design to be as inexpensive as possible, purchasing the flex sensors was rejected and they were instead custom fabricated by hand.

5.1.2 Windows App via USB Connection

Rather than an android phone app with Bluetooth connection, a windows computer app with USB connection was a considered design alternative. This would have eliminated the Bluetooth module completely, significantly reducing workload. However, System Requirement 3.2.2 establishes that portability is desired by the user. A mobile phone and Bluetooth connection are less cumbersome than requiring the user to be in close proximity to a PC over a wired connection.

5.1.3 Servo Motors

SER0043 servo motors were considered for their unique ability among servo motors to rotate 360 degrees, twice the normal range of motion for servo motors [9]. This capability would consequently allow for a greater range of articulation in the user's fingers. However, these motors drew significantly more current than our final selection, and they were larger as well, which conflicted with System Requirements 3.1 and 3.2.2.

5.1.4 Exoskeleton

A robust exoskeleton could have been designed to encase the user's hand and stiffly guide the finger articulation. Such a design was rejected on the basis of its complexity and price. The time required to design such an exoskeleton is misaligned with the focus of this course: to conduct rigorous ECE design. Furthermore, the price point for such exoskeleton files that could have been directly 3D-printed opposes Design Constraint 4.3 by consuming a large amount of the budget [10].

5.2 System Description

The Tendon-Driven Soft Hand Exoskeleton is a device that is controlled by a phone app. There are different exercise selections that can be chosen in the ap, or custom ones made by the user that are performed by the glove. Exercises vary in speed, finger selection, and how far the

fingers may curl. The glove will then perform the exercise by pulling at a thin wire line (the synthetic tendons) attached to the fingertips of the glove and fed through guides onto pulleys attached to motors contained within the wrist mount of the glove. The glove is able to detect the position of the fingertips to ensure that the exercise is being performed properly via flex sensors. This positional data is sent to the app, and the app determines when to stop a motor once at the desired position. Once complete, the user will select to stop the exercise on the app side and the motors will return to their default positions.

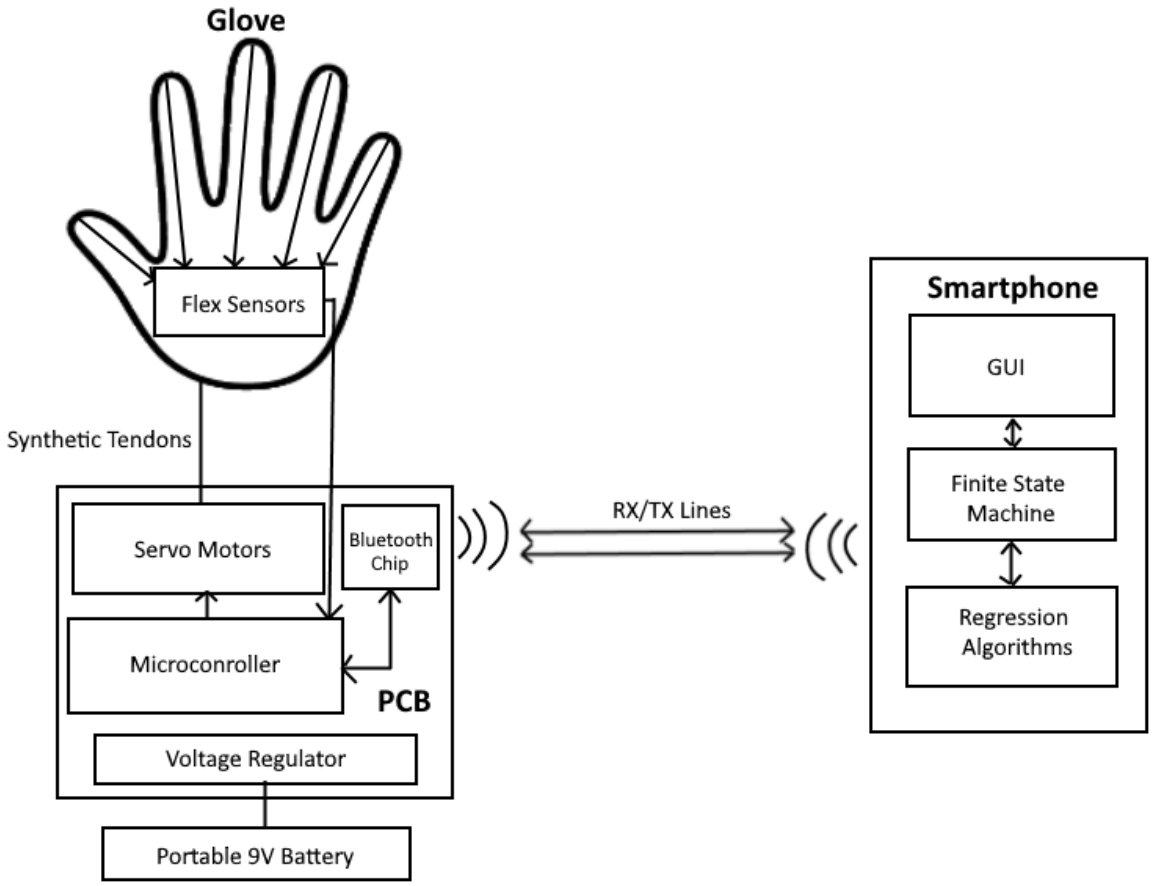

Figure 3: system data flowchart

Should the exercise malfunction, there are fail safes in place to ensure the user's safety. There is a set stop point to ensure that the fingers do not go past a safe position on the hand. The motors are attached in such a fashion that a maximum rotation from the pulleys cannot pull any finger beyond a full closed position. The motors cannot provide torque in the opposite direction, so fingers cannot be bent backwards. Along with this, if the motors stall for any reason, the system will also reset to the original position.

In addition to assistive exercises, the app also offers a non-assistive mode where a more rehabilitated user can move their hand themselves to execute any exercise and see their progress in real time. The app will show a progress bar for each finger, as it does in assistive exercise mode, and the user can track exercise completion. These exercises can be completed with no resistance/assistance or with resistance utilizing a separate resistance band if the user owns one.

5.3 Hardware

5.3.1 PCB Design

The PCB was designed to be compact and easy to assemble. For these reasons, the board was designed to be a square 2.5 inches wide, and the ATMega328P was placed in the center to allow traces to be easily connected to the correct pins. The traces are 15 mils wide to decrease the board's overall complexity. Finally, similar discrete components were placed together to make it easier to assemble and debug. The PCB is powered by a 9V alkaline battery and is regulated by a 5-volt, 5-amp linear voltage regulator to decrease and stabilize the input voltage for the ATMEga328P, Bluefruit LE module, Pocket Programmer, and the servo motors. Capacitor 3 is a 10µF electrolytic capacitor to stabilize the input voltage. The 16MHz crystal oscillator and two 22 pF capacitors are connected to the microcontroller so it can be reprogrammed on the PCB. An LED with a 330Ω resistor in series was placed to allow easy visual debugging. Finally, resistors 3-7 are the upper resistors in a voltage divider, with the flex sensors acting as the variable bottom resistance to ground. The voltage between the two is read by the analog input to the ATMega328P.

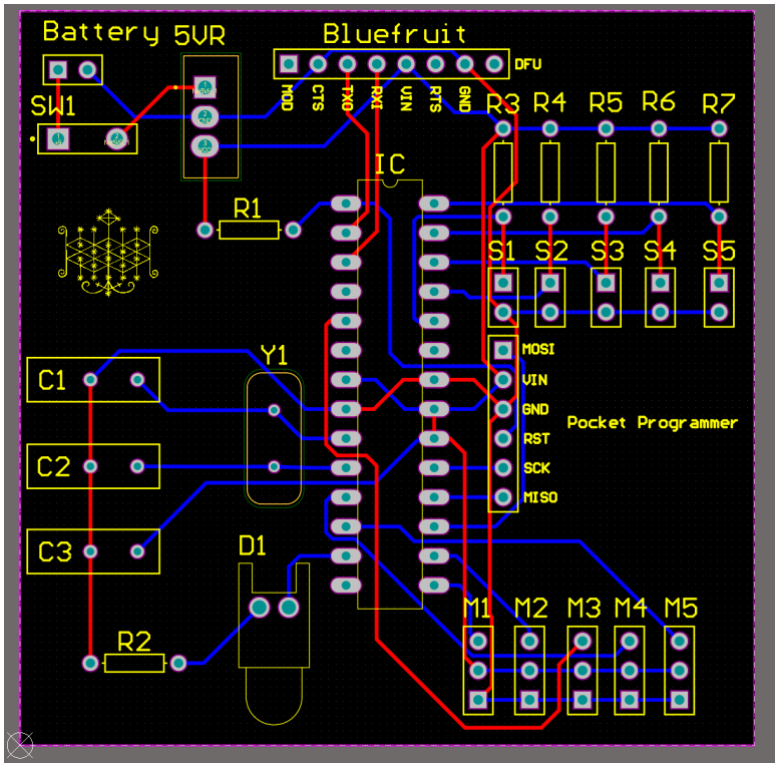

Figure 4: PCB Layout

5.3.2 Flex Sensors

The flex sensors were custom fabricated by hand following a design procedure acquired through researching custom flex sensors online [11]. These flex sensors have an inverse behavior to those found for purchase online: their resistance is high when straight

and low when bent [8]. They achieve this behavior by employing a material called Velostat which possesses a material property whereby its resistance sharply decreases when stress or strain is applied [12]. The basic construction methodology remained constant for the duration of the project, but the outer layer changed as design iterations took place. A length of Velostat was cut to size, depending on the finger for which the sensor was being fabricated, and sandwiched between two lengths of copper tape. The conductive side was in contact with the Velostat, and the adhesive side was placed on the inside of the outer layer of the sensor. There was a small tab of copper tape left out of the outside layer, to which wires were soldered so the sensor could connect to the PCB. Such wires may be seen in the pictures in Appendices 11.2 and 11.5.

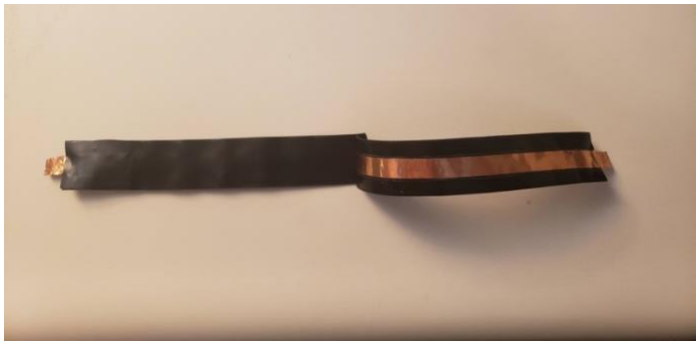

Figure 5: interior of a final revision electrical tape sensor

Initially, based on preliminary research, laminate sheets were used for the outer layer [11]. This plastic material was light, but there were issues with the tightness of the seal around the Velostat from the laminating process. Too much air around the Velostat meant it would not bend in the desired path in concert with the finger. Furthermore, the laminate was not flexible enough to accommodate the multiple bending points of a finger without impinging upon the Velostat, distorting the resistance more than what it should have been just from the bending motion.

Once the laminate was determined to be inadequate, the designer explored using duct tape as an outer layer. Duct tape was attractive because it was more secure than the laminate and was easier to work with since lengths of it could be readily cut to size. The adhesive side of the duct tape also holds pencil marks well, which enable precise placement of the copper tape to ensure each length cleanly overlaps across the Velostat. At the time of fabrication, it was though that this made the sensors more precise, later it was discovered through testing that in fact the inverse was true. However, duct tape also presented a similar issue to the laminate of inflexibility. This inflexibility was more severe than the laminate and was so severe as to impinge upon the Velostat even when the sensor was straight. This resulted in high variance of the sensor resistance, too high for the sensors to be usable.

Figure 6: example duct tape (top) and laminate (bottom) sensor

With assistance from the instructors, the designer established contact with a former student who had also constructed flex sensors for his senior design project. After a brief discussion, the designer was encouraged to explore using electrical tape as an outer layer. This proved to be a good choice, as the electrical tape allowed for greater flexibility in the sensors and was light enough to not impinge the Velostat. The electrical tape design was the final design used, and substantial testing was conducted to understand the flex sensor behavior under different bending conditions and construction styles. Pictured in Figure 7, from top to bottom: short, medium, long, slight-misalign (not tested) and misaligned.

Figure 7: electrical tape sensors used in testing.

In total, 10 laminate sensors, 18 duct tape sensors, and 14 electrical tape sensors were fabricated. These are documented in Appendix 11.5, except for five of the electrical tape sensors which were used in the prototype.

5.4 Software

5.4.1 Bluetooth

The Bluetooth Low Energy (BLE) connection between the microcontroller and the mobile app is facilitated via an Adafruit breakout board, the Bluefruit UART Friend [13]. This board was chosen due to its extremely low price point while still not requiring custom programming of XML code to have it operable. This device allows for connectivity using the universal transmitter and receiver (UART) protocall. This protocol uses two lines, a receiver for incoming data (RX) and a transmitting line for outgoing data (TX). Likewise, variables are sent to and from the peripheral utilizing the generic attribute profile (GATT) protocol stack. This protocol assigns a 128-bit id to services and characteristics in order to send/receive data. Services are the connection lines between two devices, and characteristics are the actual variables being sent. We use two services, one for data being sent from the microcontroller, and one for data being sent from the app. The microcontroller characteristics include the voltages for each flex sensor reading and a done signal from the microcontroller. The app characteristics include a bit shifted integer value for which motors to move, an integer value from 1 to 100 to determine how closed the hand should be, an integer value from 1 to 100 to determine how fast the exercise should complete, and a done/reset signal from the app to tell the motors to stop at the correct position and uncurl.

The BLE UART Friend does not use the standard BLE libraries, it uses custom ones designed by Adafruit. Since most of Adafruit's BLE chips are designed to work with their reprogrammable boards, the libraries are not optimized for microcontroller bootloading and use. As a result, the way in which communication between chip and app was handled had to be completely overhauled from the base code. For one, data transmission speed was a priority, so the designer implemented the hardware UART setup as opposed to the base software one. This reduced data loss and increased transmission speed. Similarly, hardware UART uses more connection lines, so the designer disabled the mode input from the chip to reduce the number of physical connections needed (the ATMega328P has a limited number of input pins). As a result, all code had to be written in the Hayes/AT command style of programming opposed to using pre-built functions. An example of how a service is instantiated using this protocol opposed to using prebuilt functions is shown below. In order to automate this process instead of using a command line manually, a custom method for writes/reads was created to still be able to send/receive data. This process had to be done for creating services/characteristics, sending data, and receiving data. Likewise, this change also made Serial printing to the command line impossible as it would try to send these prints as data through the RX line. This was a particularly large issue to debug as Adafruit's preset code would use the command line for debugging, so that prebuilt code had to be removed entirely. Lastly, given the difference in internal clock speed of the ATMega328P and Arduinos, a 5ms delay needed to be added in between the creation of each characteristic and service (discovered through experimental observation) and results in a 13 second system startup. The Bluetooth connectivity code accounts for roughly 1⁄3 of all Arduino code on the chip.

```
htsMeasureCharId = gatt.addCharacteristic(0x2A1C, GATT_CHARS_PROPERTIES_INDICATE, 5, 5, BLE_DATATYPE_BYTEARRAY);
```

Figure 8: characteristic creation with pre-built functions

```
// Adds characteristics
success = ble.sendCommandWithIntReply( F("AT+GATTADDCHAR=UUID128=00-00-00-02-62-7E-47-E5-A3-fC-DD-AB-D9-7A-A9-66,"
"PROPERTIES=0x2, MIN_LEN=1, MAX_LEN=20,VALUE=5,DATATYPE=1"), &thumbCharID);
if (! success) error(F("Could not add thumb measurement characteristic"));
delay(500);
```

Figure 9: characteristic creation with custom ATT-line Commands

5.4.2 Mobile App and Finite State Machine

In order to control the rehab glove, a mobile app for Android devices must be connected. This app was written using Android Studio and uses a combination of Java, Kotlin, and XML languages. Initially the app was being adapted from Google's BLE Scan example, however the application programming interfaces (API) were 8 years out of date and incompatible with modern displays. This resulted in a complete overhaul of the app from scratch and custom APIs to be developed. The new UI for the application is shown in the figure below. The app allows for users to scan for nearby BLE devices. After 3 seconds, the device completes the scan, and the rehab glove will appear in available devices to connect to (the interface is programmed to only show the glove and no other devices). Once connected, the device will display the flex sensor voltages for each finger at the bottom of the screen and can now perform exercises. An exercise is performed by one of two ways, either through preset exercises or custom exercises. The user may select the drop-down bar to choose from one of several exercises adapted from the University of Texas Orthopedic Physicians recommended exercises for hand rehabilitation [1]. The user also has the option to use the UI to create their own exercises. Individual fingers can be selected by clicking the fingertips of the desired finger. The speed of the motors can be selected using the speed bar. The degree of how closed the user wants the fingers to be also can be selected using the position bar (a 0 indicating a fully open hand, and a 100 indicating fully closed). This exercise data can be named and saved alongside the other preset exercises using the save button. The assistive mode bar at the top of the UI allows the user to turn on/off the motor assistance, this allows the user to still use the glove for exercises even when they have moved past the need for assistive rehab. The UI is handled by 13 functions (the largest being onCreate which handles the majority of data structures and implements the device listeners for interactive elements of the UI) and 19 sub-functions.

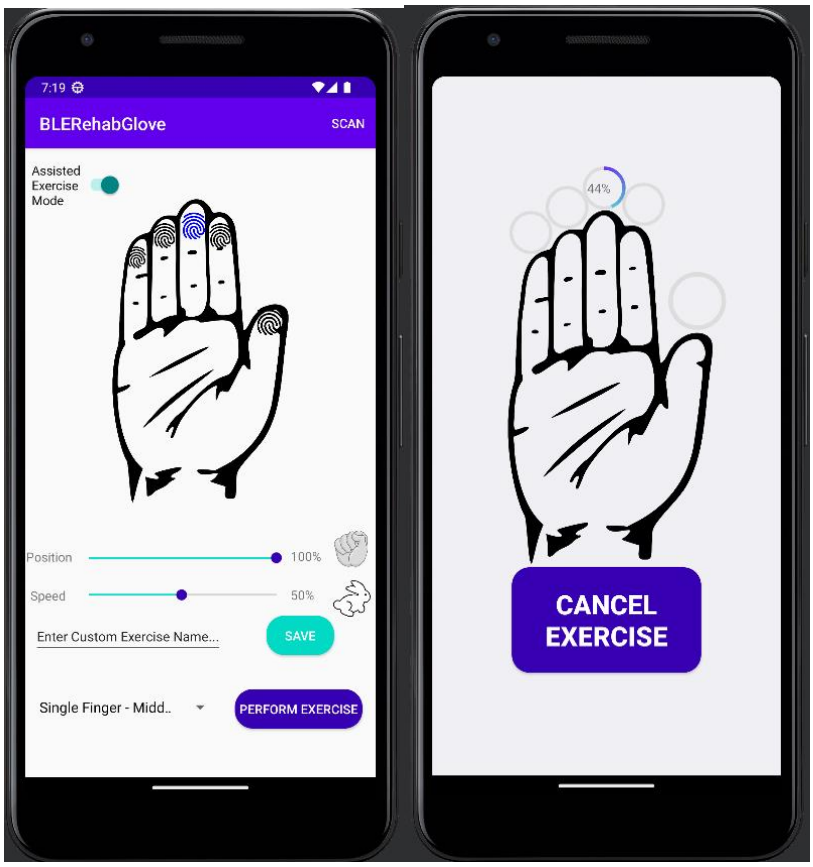

Figure 10: UI for application (Left) and UI for exercises (Right)

The Bluetooth is handled in the application via a finite state machine (FSM). Due to the GATT protocol constantly advertising data with assigned ids, there needs to be a way for the application to parse this information. The FSM allows for the app to systematically parse through each characteristic for reads and writes and assign the proper incoming/outgoing data. The FSM, shown in the figure below, allows for the system to properly read in the flex sensor values and assign them to the correct fingers, while still broadcasting which motors to move. The three major sections of the FSM are as follows: the startup (shown in blue), the read cycle (shown in red), and the write cycle (shown in green). The read/writeNextSensor states pull the next GATT characteristic in line, and the onRead/WriteCharacteristic states interact with the microcontroller to send and receive the data. The onReadCharacteristic also interacts with a custom message handler that updates the UI in real time with the data being read in from the sensors. This message handler also calls a FlexSensorCalc class that allows for the regression algorithms to determine the position of the fingers. These positions will update in the UI, as well as set internal flags to stop the motors if the fingers have reached their desired position.

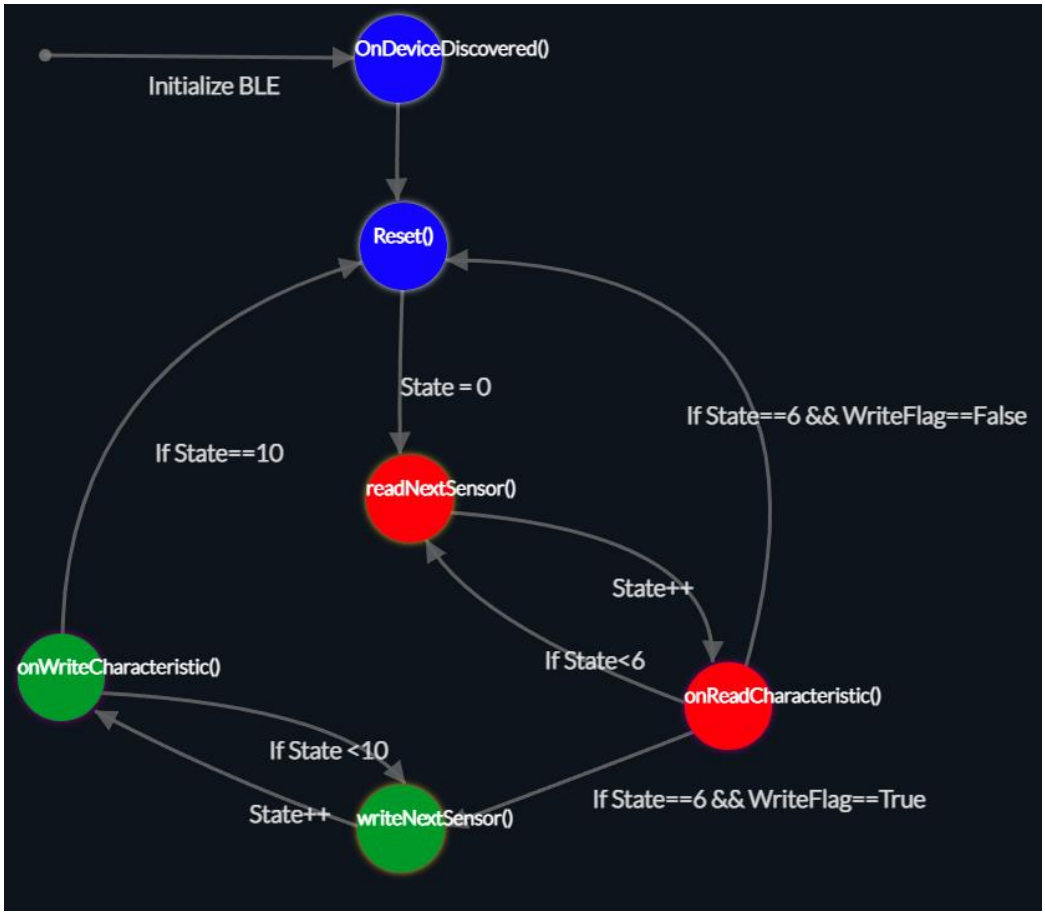

Figure 11: finite state machine used for mobile app

5.4.3 Motors and Regression Testing

In order to determine position, a regression had to be completed to determine the relationship between degrees of motor rotation and voltage read at the top of the flex sensors. This was done by altering the motors to rest at a certain degree (0° being completely unwound, and 180° being fully wound) with someone wearing in the glove while the flex sensor voltages were read. 30 flex sensor readings were attained per finger for every 10 degrees of rotation. After this degree range was complete, we would move to the next degree range. The trials went from 10 degrees to the complete rotation of the motors at 180 degrees. This means that for each finger 540 data points were generated, thus giving 2,700 data points overall. A trend line was then created in MATLAB to show how the points correlated on average. The equation from this trend line was retrieved from MATLAB and put onto the ATMega328P along with being implemented on appside. Figure 12 is an example of a trend line that was created from the data of the pinky tests.

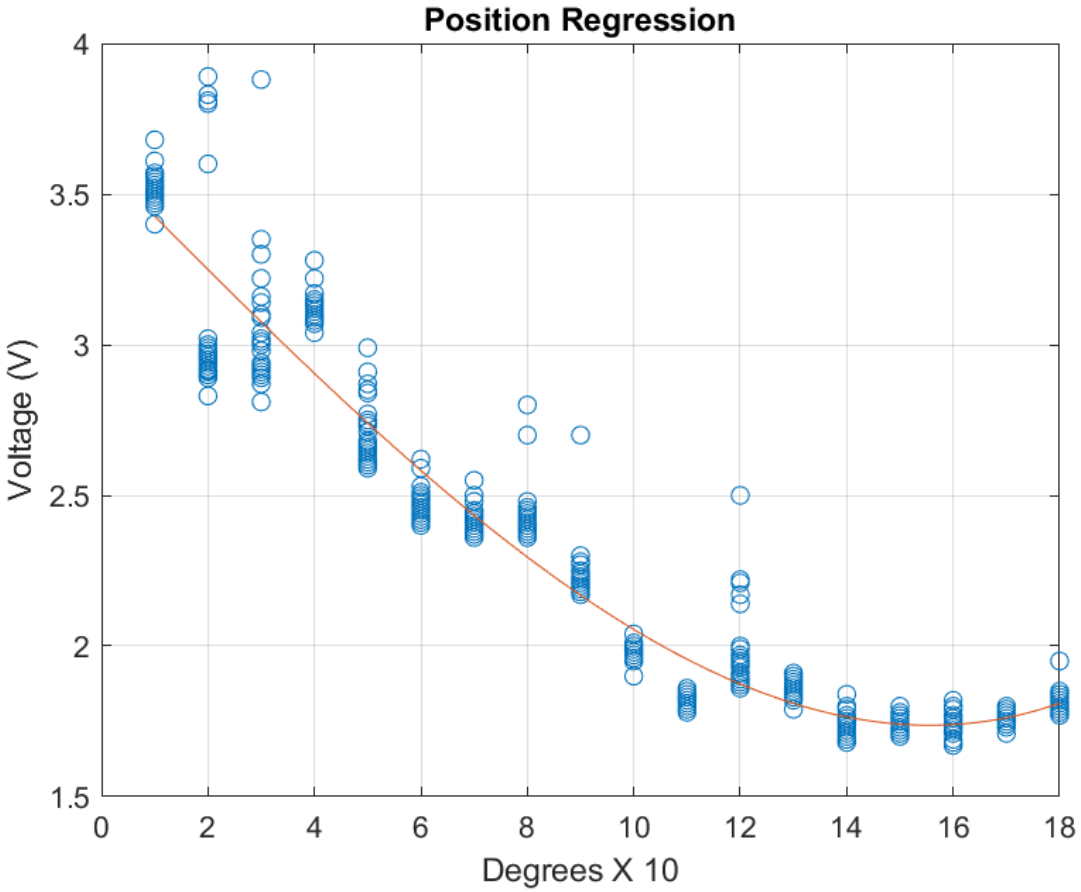

Figure 12: Trinomial Pinky Regression

Once the equation was implemented onto the ATMega328P, it was quickly discovered that having voltage as the independent variable was a poor decision. The algorithm needed to compute a bending angle for the finger from the voltage measurement in the divider. Therefore, the plots variables were inverted, and new plots were generated. With these new best-fit equations, the thumb and pinky were able to make effective use of a trinomial fit. Initially, we tried using a trinomial fit for the index, middle, and ring fingers, but found that a linear fit gave more accurate results in determining position. Due to the length of these three fingers, inconsistent data would lead to fingers not being able to partially curl, only fully curl. The switch to linear would result in a better overall accuracy in positional testing. Once these criteria were finalized, accuracy testing began.

FINGER	TRINOMIAL BEST-FIT EQUATION	LINEAR BEST-FIT EQUATION
THUMB	$Degrees = -11.8725V^3 + 83.643V^2 - 239.0979V + 380.274$	
INDEX	$Degrees = 338.014V^3 - 3572.9V^2 + 12360V - 13887$	$Degrees = -175.3966V + 712.1525$
MIDDLE	$Degrees = 1091V^3 - 13238V^2 + 53203V - 70725$	$Degrees = -261.2576V + 11141.9$
RING	$Degrees = 329.2518V^3 - 3566.3V^2 + 12662V - 14627$	$Degrees = -158.1201V + 665.6175$
PINKY	$Degrees = -9.7278V^3 + 116.477V^2 - 479.5077V + 685.3639$	

Table 1: trinomial best-fit (and linear where applicable) equations for each flex sensor

5.5 Exoskeleton

The exoskeleton design was constructed of two broad modules: the components mounted on the fingers which connected to the strings and flex sensors, and the components mounted on the user's wrist which encapsulated all the electrical hardware of the prototype. Images of all Fusion 360 models are left to Appendix 11.2.

5.5.1 Finger Guides

The finger guides underwent several revisions. Originally conceived as a ring that would fit snugly around the user's fingers, three of these rings would be worn on each finger - one between each joint and one on the tip - with two worn in a similar manner on the thumb. The rings had three geometries supporting extraneous hardware and physical components. The first was a small loop for the strings to fit through, the strings would be tied off at the loop on the ring worn at the end of each finger. This mechanically set the force from the servo motors at the fingertip, curling the finger when the motors rotated by allowing the string to slip through the loops on the lower rings. The second functionality was a small slot for the flex sensors to fit through. The sensors would fit snugly, but without excess pressure, through these slots and bend along with the fingers when they curled. The final functionality was a small notch which was intended to secure a rubber band to assist with uncurling the fingers. This component was dropped in subsequent revisions after consultation with a licensed physical therapist informed the team that the fingers do not require assistance to uncurl.

Several issues were discovered with the ring design that forced iteration. The first was the restrictive nature of wearing a ring around the finger, which made curling the fingers difficult. The rings were size by measuring the designer's fingers between each joint. The first issue arose because the actual joints are larger than the area between them. For some fingers where the disparity was too great, the rings wouldn't fit over the joints to get to their designated finger areas. Others were simply difficult to squeeze over the joints. Instances such as these made the rings difficult to put on the user when they were wearing the glove and would have also made it difficult for the user to remove the glove - or for them to have it removed - when the rings were secured to the glove. The second issue was that when the fingers are curled, the areas between the joints expand because the flesh there

is compressed. This increases the diameter of these sections and, in some instances, makes the rings too snug and uncomfortable to wear since they impinge on the fingers. While the pain of the impingement might not be a concern for someone whose hand is paralyzed, the restriction it imposes on the fingers' range of motion is a concern for the functionality of the device. The finger guides were revised to be less restrictive by eliminating the rings.

The iterated design kept the necessary components from the rings: the loops for guiding the strings and the slots for guiding the flex sensors. Instead of connecting them with a ring, these components were designed as distinct pieces to be attached to the glove at the appropriate points. In addition to string guides on the fingers, as were present with the ring design, string guides were also attached to the palm of the glove. This prevented the strings from interfering with objects the user sought to hold in their palm and added more torque to the strings which allowed the fingers to curl to a greater degree.

5.5.2 Wrist Mount

With the goal of portability in mind, the exoskeleton was also designed to contain all of the electrical hardware - the PCB, the battery, and the motors. The wrist mount is composed of four pieces and secured to the user's wrist using Velcro straps fitted through slots on the side. In a second revision, these slots we made in the wrist mounts themselves, rather than through tabs jutting from the mount. This made the design more durable. There are two beds, which are curved on the inside to fit comfortably over the user's forearm, and flat on the outside. On the flat side, various mounts are attached. One side attaches to a mount for the PCB and battery, the other to a mount for the motors. The PCB fits snugly from the top into its walled enclosure and is held in place with binder clips on the sides of the wall. The battery slides easily into its enclosure, and the mount is situated on the outside of the user's arm such that the opening is facing upwards, and the battery can never fall out through any natural motion. The second mount is attached to the inside bed and holds the motors in place.

The motor mount underwent several revisions through the design process. The first iteration had the motors laying sideways in their enclosures, with armatures attached to the shafts tied to the strings along the fingers. Loops similar to those attached to the fingers and palm are placed in line with the armature to guide the string to the motor. The radius the armatures rotated through was too small to provide sufficient torque to curl the fingers, so the motors were moved to sit upright in the enclosures. To augment the radius of rotation, a design for pulleys was acquired from an online source and attached to the shaft of the servo motors [14]. This placement resulted in the strings having to move in too severe of an angle from the guides to the groove in the pulleys, and the apparatus did not function as intended. Therefore, the design was revised for a third time to bring the string guides up to the same level as the pulley grooves. This ensured the strings were always fed into the pulleys so the motors could effectively curl the fingers.

Occasionally, the strings became caught on the guiding loops when the motors were released, and the fingers uncurled. A further revision of the motor mount where the strings ran through the mount itself and were guided into the pulleys was designed and printed. However, this design was not printed in time to be implemented, so it exists as an example of how the exoskeleton might have evolved given more time.

6. Team and Timeline

6.1 Daniel Braunstein: Flex Sensor and Exoskeleton Designer

Daniel's role in the group was to design and fabricate the flex sensors, and to design and oversee the 3D printing of the exoskeleton. Daniel had basic experience with CAD and 3D printing, and extensive practice with various low-resolution fabrication methods.

6.1.1 Skills

Daniel learned how to develop proper experiments for electrical systems through his experience in ECE 1212 Electronic Circuit Design Lab, and he used these skills to develop a testing plan to verify and understand the behavior of the flex sensors. Further, Daniel's experience with PCB design in ECE 1895 Junior Design Fundamentals allowed him to assist Michael in verifying the PCB design, debugging the board, and resoldering components when necessary.

Outside of ECE coursework, in The Art of Making and Hands-on System Design, Daniel gained experience with hands-on fabrication techniques and designing components for wearable devices. This course, and Introduction to Mechanical Engineering Design, served as an introduction to CAD and 3D printing.

6.1.2 Timeline

When the semester began, Daniel needed to quickly understand how to construct flex sensors by hand. He conducted research on fabrication methods, and how Velostat works. In tandem, he began developing and printing the rings for the fingers of the exoskeleton. At the first checkoff, Daniels presented the laminate flex sensors with appropriate functionality, and the fitting of the exoskeleton rings to fingers.

Between the first and second checkoff, Daniel pivoted to the duct tape flex sensors and fabricated a large number to understand how variations in construction altered their behavior. After a certain point, he pivoted again to the electrical tape sensors and began fabricating those. He also began developing the wrist mount components of the exoskeleton. At the second checkoff, Daniels presented an early iteration of the wrist mount as a completed print, and preliminary regressions for the electrical tape sensors.

After the second checkoff, Daniel worked to finalize the design of the motor bed to ensure proper functionality. He also conducted extensive tests on a curated set of electrical tape flex sensors to understand their behavior under various conditions in isolation from the larger device. These tests were conducted with assistance from Michael and Tyler, who recorded data while Daniel adjusted the sensors.

6.2 Tyler Sheetz: System Software

Tyler's job in the group is to program the microcontroller alongside the sensor connections and safeguards. He is well suited for this task because of his familiarity with sensor functionality and safety implementation.

6.2.1 Skills

Through ECE coursework, Tyler has learned how to wire and test sensor connections by both breadboard testing and PCB debugging. He has become familiarized with programming processor chips with C along with the wiring process of processor to sensor. Tyler put this into practice during Junior Design as his role for that class was the same as it is now. Alongside of this, Tyler has learned how to alter registers bitwise to adjust innate circuit designs to more fit the needs of the project.

Outside of his coursework, Tyler has worked extensively with PLC and sensor connections. He has created safeguards to ensure that the machinery attached would not exceed a certain limit depending on the sensor readouts. With these projects, He was able to design the connections by himself. For the sake of this project, that helps Tyler better understand the physical process of the system.

Aside from work experience, Tyler has plenty of hobby experience that makes him a good fit for this role. He often tinkers with Arduino and raspberry pi programming to connect to sensors to fulfill a pet project of his own.

6.2.2 Timeline

In the beginning of the semester, Tyler began working on effective motor code with a breadboard and an Arduino. By the time the first checkoff was due, he had been able to complete motor code that addressed each motor individually based off of a decoded integer. Shortly after this, he had created code that would show the analog value that the ATMega328P would read once on the PCB and the system integrated.

There was a portion of time between the first and second checkoff where he was waiting on the system to be integrated enough so that the code could be improved and further tested with the system rather than the breadboard. Alongside this, the system must have been put together to perform the regression testing necessary in finding the position of the finger through flex sensor values. Due to this lull in time, he was helping other team members wherever needed. One big aspect that he helped with was the flex sensor testing. Tyler helped Daniel in getting over one thousand data points for his angle tests. Other small items that he assisted on were PCB soldering and debugging, slight code improvements, and CAD model aid for the wrist mount creation.

Once CAD models were finalized, he was able to use his 3D printer at home to create some pieces for the wrist apparatus. With these being some of the final pieces for the system to be put together, he was able to begin fully implementing the motor and flex sensor code with the Bluetooth module along with regression testing roughly a week before the second checkoff.

Between the second checkoff and the design expo, alterations to the motor code were made which included having the motors change speed and position in response to user input on appside. If the user chose to make the motors move slower on their phone, the physical system would respond in kind for the next exercise. Once done, everything was set to begin performing regression tests. The tests themselves involved collecting 30 samples of data for each 10 degrees of rotation on the motor for each finger. This ended up giving 2,700 data points altogether which would then be taken into MATLAB to run a regression. This was then used to determine the position of the hand and fingers during the

course of the exercise. Once the equation was implemented in the code, Tyler made the motors return to their natural state if the position was deemed to be too astray from the expected position value.

6.3 Bryan Hess: App Algorithm and Bluetooth Functionality

Bryan's primary role is to design and implement the UI system via a mobile app, incorporating the algorithm for finger controls, as well as handle the Bluetooth communication protocol between the microcontroller and the app. Bryan has experience in medical R&D at Co-Op at the University of Pittsburgh's Health and Rehab Science Center where he helped conduct a study on wheelchair maintenance data collection. During his time there, he received CITI certification in biosensors. Likewise, he has worked a summer internship at the Bettis Atomic Power Laboratory (or Naval Nuclear Lab) as well as a current internship for the Regional Industrial Development Corporation of Southwestern Pennsylvania. Alongside his Computer Engineering degree, Bryan is pursuing a BA in Economics.

6.3.1 Skills

Inside the classroom, Bryan has accrued a multitude of skills through his coursework while enrolled at The University of Pittsburgh. He has experience in algorithms from ECE 1148 Algorithmic Thinking. Likewise, he has machine learning principles learned in ECE 1395 Into Machine Learning. His time in ECE 302 Data Structures and Algorithms alongside microcontroller functionality learned in ECE 1895 Junior Design Fundamentals will allow him to help bridge the gap between the software GUI and the microcontroller code.

Outside of lectures, Bryan has experience in the health-tech industry with his co-op at the Health and Rehab Science Center. During that time, he worked on R&D for two wheelchair reliability studies. During this program, he worked writing custom data analysis code as well as sensor programming on a breakout board. As part of the study, he worked in clinical trial testing, receiving CITI certification, and developed 3D printing skills to develop an enclosure for the device. His time in other industries, such as the public sector or in the economics field, also taught him the importance of clear communication in project management and to plan for delays.

6.3.2 Timeline

At the start of the semester, Bryan took fast to learning as much as possible about Android app development, given his lack of experience. He spent much of the first few weeks watching tutorials and developing basic applications to ease the process once materials arrived. Once the Bluetooth chip arrived, Bryan took fast to developing the connectivity between both app and device. He worked on an Arduino Mega 2560 for fast prototyping and ease of development. The first major roadblock came in the way of poor documentation of Adafruit libraries for BlueFruit Bluetooth Low Energy. He had to change the configuration files to allow for hardware UART connectivity, and also update the board's firmware. Once this was done, he had to adapt the wiring schematic and reconfigure the software to accept UART data in command mode in order to reduce the number of required pins, as well as keep the transmission rate as fast as possible.

For the first checkoff, Bryan presented his connectivity using the UART protocol between app and chip. He used prebuilt apps from Google and Adafruit to test the read/write capabilities of the chip as well as the ability to create custom GATT ids. Two major roadblocks would arise at this point that would slow software development to a grinding pace. First, the switch from Arduino to the PCB microcontroller was a major, unforeseen hurdle. Hardware wise, the RX and TX pins on the board were reversed from the ones on the UART chip, so Bryan and Michael worked to remedy this issue and allow for Bluetooth communication on the PCB via the microcontroller. The second, and more cumbersome of the two, came in the way of software implementation. Microcontroller side, the Adafruit Bluetooth chip that was being used had not been optimized to work on the ATMega328P. In order to fix this problem, Bryan changed how GATT services were instantiated. Likewise, there was very little documentation on how to write a value to the microcontroller from the app, so he had to overhaul that process through experimentation as well. Another major blow was the base connectivity app he had been working off of from Google had APIs 8 years out of date and incompatible to modern Gradle builds. As a result, he had to fully remake the app from scratch. This took a considerable amount of time and ultimately led to the redevelopment of the Finite State Machine on the app. This would ultimately be a benefit as the new app ran better and the new FSM allowed for quicker data transmission.

For the second checkoff, Bryan presented the new app as well as the 2-way communication from chip to phone. He showed a simple close hand exercise using the app to close the hand around a stress ball. From here, the UI and layout were heavily developed, with custom graphics and real time feedback. Likewise, data structures were implemented into the code to allow for variable exercises and saving custom exercises. Bryan worked with Tyler to implement the regression algorithms for each finger in order to track position. This was integrated into the code and now allowed from the app to stop the motors when fingers reached their desired positions.

6.4 Michael Etter: Hardware Engineer and Project Manager

Michael's primary role is to design and manufacture the PCB for the device. He is responsible for designing the device's schematic and PCB layout using Altium Designer. He is also responsible for soldering the microcontroller and the other components to said PCB. He will calculate the device's power output to optimize device usage. Lastly, Michael will act as a Project Manager and will keep track of the budget and the schedule.

6.4.1 Skills

Michael learned how to design and manufacture PCBs in ECE 1895. He knows the best practices when designing a PCB and knows how to properly solder components to a PCB. Furthermore, from ECE 1776, Michael knows how to properly calculate a battery's operating life. During his Co-op at Emerson Automation Solutions, Michael learned about being a Project Manager. He has led multiple projects before and will use the interpersonal communications and budgetary skills to help manage the project.

6.4.2 Timeline

Michael began researching microcontrollers and other components for the exoskeleton. He determined that the ATMega328P was the best option, due to its 5 PWM outputs, and its simplicity. The ATMega328P could easily be programmed and attached to the PCB. During the first couple of weeks, Michael designed and manufactured three boards in Altium Designer. One utilized the RN4870-V Bluetooth Module, but it was scrapped due to the complexity of programming the RN4870-V and the soldering method being difficult as well. The second board was removed from the project because the flex sensor traces were incorrect. The third PCB was the one utilized in the final design of the exoskeleton. Michael ran the Flying Probe Test and Optical Inspection to verify its connectivity and made sure the PCB matched the design. During this time, Michael also breadboarded the PCB and verified the design. Moreover, he also scheduled meetings and managed the budget.

For the first checkoff, he presented the PCBs and the breadboard prototype. He also determined key PCB metrics like the propagation delay. Immediately after the first check off, he began soldering the components and headers to the PCB. Once the PCB was soldered completely, he began the functionality tests. The first issue was Bluetooth connectivity. Michael and Bryan worked to troubleshoot the issue immediately. Michael redid the flying probe tests and optical inspection to reverify the hardware. He also designed three different PCBs as a potential hardware solution. The Bluetooth connectivity issue was resolved, when Michael and Bryan realized that the RX and TX wires connecting the Bluefruit to the 8-pin header were switched. This fixed the Bluetooth connectivity problem. The second issue Michael ran into was the microcontroller not receiving enough current. The linear voltage regulator selected did not output enough current for the motors and the microcontroller to run simultaneously. Once the issue was identified, he replaced the linear voltage regulator with another that had a greater output current and it fixed the problem. Michael then finished conducting the functionality tests and presented his demonstrations for the second checkoff. He also attempted to make a rechargeable battery module, but it failed due to a diode shorting, and this failure led to him burning his laptop's motherboard.

After the second checkoff, Michael would assist his teammates. He would assist Daniel with the construction of the exoskeleton, tying knots to the motors and braces. Furthermore, he helped Daniel document some of his regression tests. Michael would also be present to answer any questions about the PCB/Power, fix anything that broke off the PCB and to assist his teammates with small tasks and administration work to allow them to focus on more pressing issues.

7. Testing, Data Analysis, and Results

7.1 PCB Testing

The first test conducted on the PCB was a Flying Probe Test, which checks the board for opens and shorts. Trace connectivity was tested by placing probes to the solder pads and vias. The probe would then sound an alarm if current passed through the trace, demonstrating the trace's connectivity. The Flying Probe Test was successful as there were no shorts or opens on the PCB. The second test was an optical inspection. If the PCB did not match the schematic, or a trace was incorrect, the board would be deemed a failure. The PCB passed optical inspection and progressed to the final stage of testing, Functional Testing.

Functional testing sees if the PCB could successfully operate and utilize the components soldered onto it. The first functionality test was making sure the ATMega328P could be reprogrammed on the PCB and use a digital I/O pin to turn on an LED. The PCB was able to do this, so it progressed to the second functionality test, Bluetooth connectivity. The Bluefruit LE Module was connected to the PCB via 8-pin male headers and wires. The ATMega328P and Bluefruit LE were able to communicate using USART communication (data could be transmitted and received) therefore it passed the second functionality test and progressed to the third functionality test, servo motor functionality. The servo motors were connected to the PCB via 3-pin male headers and were controlled by 5 PWM outputs from the ATMega328P. The ATMega328P was able to rotate the servo motors, so the PCB passed the third functionality test. The final functionality test was flex sensors. The flex sensors were attached to the PCB via 2-pin male headers and were bent to observe the voltage drop. The voltage across a 510 Ω resistor above the flex sensor was input to an ADC pin. Utilizing the LED again, the hardware engineer would observe the LED turn on or off depending on the voltage. The PCB passed the flex sensor functionality and was approved for use in the exoskeleton. In summary, the PCB passed a Flying Probe Test, Optical Inspection and Functionality Testing.

Other metrics of the PCB were calculated to provide more information. Since, the PCB would be communicating via Bluetooth, trace impedance and propagation delay were calculated. Trace impedance does not impact communication, but the metric was still calculated to understand the impedances within the PCB. The trace's characteristic impedance, Z_o, was calculated using

$$Z_0 = \sqrt{\frac{R + j\omega L}{G + j\omega C}}$$

The trace's resistance, R, inductance, L, and capacitance, C, and conductance were provided by Altium Designer. The end result was the RX trace's characteristic impedance being 6.3723Ω and the TX trace characteristic impedance being 5.9790Ω. The propagation delay was the more important metric as it determines communication quality. If the propagation delay eclipsed 500 picoseconds, the signals would be distorted and communication between the Bluefruit and ATMega would be hindered. The propagation delay is calculated automatically by Altium Designer and is determined by the trace length and transmission speed. The RX and TX propagation delays

were 110.45 and 97.242 picoseconds respectively. Therefore, the RX and TX traces were suitable for Bluetooth communication.

7.2 Power Analysis

A 9V alkaline battery was used to power the exoskeleton. The battery capacity is 550 mAh. The exoskeleton would draw 1.5952 amps in a worst-case scenario. This is due to the five servo motors drawing 300 mA, the Bluetooth drawing 15.2 mA and the ATMega328P drawing 80 mA. Therefore, the battery life was approximately 21 minutes if the exoskeleton was left on. Moreover, a single exercise would take approximately 15 seconds. Therefore, each battery contained approximately 84 exercises.

7.3 Flex Sensor Testing

It was important to understand the precision of the flex sensors independently of their functionality in the larger prototype. How did variations in their construction and bending affect the precision of their resistance measurements? Once the final electrical tape design for the flex sensors was reached, testing began on a curated set of four flex sensors that were constructed especially for such a purpose. Three of the flex sensors varied the length of Velostat used: a short sensor had 5 cm of Velostat, a medium sensor had 10 cm, and a long sensor had 15 cm. The fourth flex sensor was the same length as the medium sensor but varied in that the lengths of copper tape were offset from each other, with 1 mm of space between. The other three sensors were constructed as normal, with the copper tape on each side overlapping across the Velostat.

The testing procedure was as follows: the sensors were bent to a series of five angles of 0, 45, 90, 135, and 180 degrees and their resistance was measured by connecting the digital multimeter leads to the tabs of copper tape. Another variable introduced to the test was the bending radius of the flex sensor. It was crucial to understand how curvatures in the bending angle influenced the resistance, since a bending finger would introduce such curvature to the sensors. To enforce the bend radius, cylindrical structures were acquired from the lab and the sensors were bent around them. Four radii were utilized: 0 mm (sensor not bent around anything), 25 mm, 50 mm, and 75 mm. The shorter sensor was only able to adequately circle the lowest two radii, the medium sensors only the lowest three, and the longest sensor was tested at each of the radii. In total, for each bending angle, at each viable bending radius, 25 measurements were taken. A bend angle of 0 degrees was only measured for the radius of 0 mm, since no variation would have been introduced at larger radii. In total, 1300 data points were collected.

The mean and standard deviation of the resistance at each angle for each radius was calculated in Excel to develop an understanding of the flex sensor behavior. From the mean and standard deviation, the percent error in precision was also calculated using the following equation:

$$percent\ error\ in\ precision(\%) = \frac{\sigma}{\mu} * 100\%$$

Where σ is the standard deviation and μ is the mean, and a higher value indicates worse precision. This value is recorded in the "Precision (%)" column in each table. Furthermore, for each sensor at each bend radius, the resistance vs. the bend angle was plotted on a scatter plot with connecting lines to provide a visualization of the flex sensor behavior across the trials. A selected plot and

tabulation of mean, standard deviation, and precision are shown below as an example, with the entirety of the collected data shown in the Appendix.

Misaligned - 10 cm				
Radius (mm)	Angle (deg)	Mean (kOhm)	Deviation	Precision (%)
0	0	8.592	1.17541482	13.6803401
	45	6.684	0.95073656	14.2240658
	90	5.24	0.77995726	14.8846806
	135	5.076	0.6653821	13.1083943
	180	3.376	0.70727175	20.9499927
25	Angle (deg)	Mean (kOhm)	Deviation	Precision (%)
	45	7.108	0.66828138	9.40182017
	90	6.068	0.81584312	13.4450086
	135	4.484	0.45522888	10.1522944
	180	3.584	0.48792076	13.6138605
50	Angle (deg)	Mean (kOhm)	Deviation	Precision (%)
	45	7.992	1.21515431	15.2046335
	90	5.08	0.82865353	16.3120773
	135	3.292	0.24310492	7.38471797
	180	3.008	0.20800641	6.91510672

Table 2: metrics for the misaligned sensor

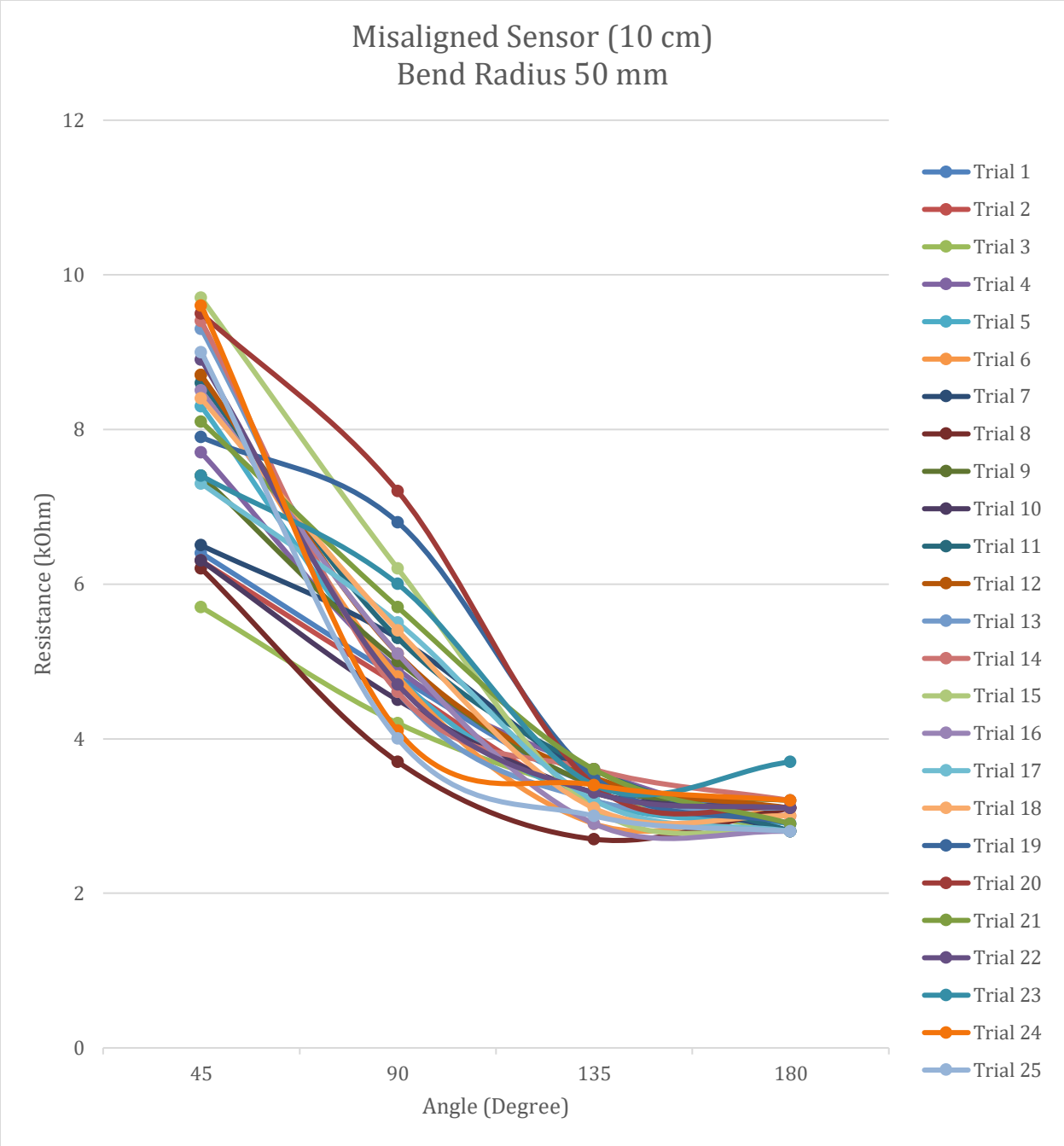

Figure 13: resistance vs angle of the misaligned sensor for a radius of 50 mm

From the collected data, it was observed that the sensor which had misaligned lengths of copper tape had a much higher resistance than the other sensors with overlapping copper. This resulted because the measured flex sensor resistance is actually the volume resistance of the Velostat, which is to say that it increases proportional to the volume of Velostat between the multimeter leads. When the two pieces of copper tape are misaligned, there is a greater amount of Velostat between them than when they overlap perfectly across it. Furthermore, this sensor also exhibited a lower overall percent error in precision than any of the other sensors. While the other

sensors became much less precise at higher bend angles, the misaligned sensor's precision remained consistent throughout the bend. The short and middle sensors in particular were much less precise at a bend radius of 0 mm than at 25 mm, which is due to the natural error from a human tester attempting to hold the sensor in precisely one bend. Velostat is highly sensitive to small fluctuations, which was observed throughout the design of these sensors and in particular with the duct tape sensors discussed in 5.3.2. Bending the sensor around a solid radius served to stabilize it.

In terms of the absolute mean resistance of the sensors, a correlation was observed where the mean resistance of the sensor when held straight decreased proportional to length. Although this violates the concept of volume resistivity discussed earlier in this section, this correlation is due to the Velostat sensitivity. It is much harder to hold a longer sensor perfectly straight than it is to hold a smaller sensor perfectly straight. The tiny, unintentional movements of the human tester can more easily distort the resistance of a longer sensor by introducing small bends and stretches even when attempting to hold the sensor as straight as possible. There was no strong relationship observed between bending radius and mean resistance for sensors of any length.

7.3.1 Flex Sensor Price Analysis

Given that the primary incentive to fabricate the sensors by hand was the financial benefit, a rough calculation of the cost per flex sensor was calculated to determine the savings. The calculation will assume a default medium-length flex sensor with 10 cm of Velostat, and the price per unit of Velostat, copper tape, and electrical tape taken from the appropriate distributers [12] [8] [15]. One sensor requires approximately 11 cm of electrical tape, 9 cm of copper tape for each side, and one 10 cm by 1.9 cm piece of Velostat. The cost per piece may be calculated for each of these given the cost per unit and size per unit:

$$C_{Velostat} = \frac{(10\ cm)(1.9\ cm)}{(28\ cm)(28\ cm)} * \$4.95 = \$0.12$$

$$C_{copper\ tape} = \frac{18\ cm}{500\ cm} * \$5.95 = \$1.06$$

$$C_{electrical\ tape} = \frac{0.75\ in}{66\frac{ft}{roll}} * \frac{1\ ft}{12\ in} * \frac{\$2.29}{roll} = \$0.002$$

$$C_{tot} = C_{Velostat} + C_{copper\ tape} + C_{electrical\ tape} = \$0.12 + \$1.06 + \$0.002 = \$1.18$$

This is approximately 9% the price of the closest-sized flex sensor available wholesale from Adafruit [16]. It is appropriate to conclude that fabricating the sensors by hand significantly reduced the financial cost of this prototype.

7.4 Bluetooth Testing

There were a few tests conducted on the Bluetooth module to ensure compliance with NIH wearable technology standards. The first test was a distance connectivity test for Bluetooth. The test was conducted by connecting the microcontroller to the app and testing successful data transmission at varying distances. This process would be repeated until the two devices lost connection to one another. This process was repeated both unobstructed (i.e., no physical interference between devices) and obstructed. For the obstructed tests, the device was placed behind

plaster and concrete walls. The distance between the two devices was measured in meters. The results from the tests showed a mean 31.924 meters for signal loss when the connection was unobstructed, with a standard deviation of 2.988 meters. For an obscured connection, the mean was 20.343 meters for signal loss with a standard deviation of 2.814 meters. Compared to Adafruit's commercial beacons and microcontrollers, the expected range of their commercial products are 10-20 meters [17]. Our device testing outperformed Adadruit's commercial products and met the marker for our 30cm range outlined in the requirements.

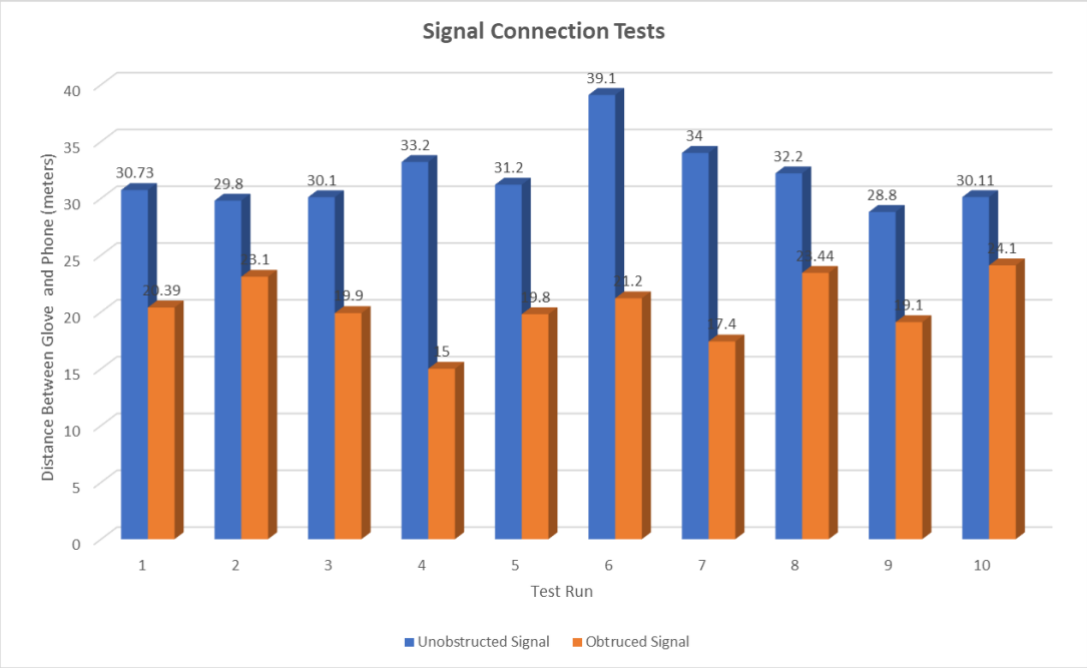

Figure 14: Distance when Signal was Lost Between the Two Devices

The second Bluetooth test conducted was a test on packet loss. Packet loss occurs when the transmitted data packets fail to arrive at their destination. Due to the high frequency nature of BLE, packet loss is inevitable. For wearable health technology, mitigating packet loss is important. For our packet loss test, we connected the microcontroller to the app for a minute, recording read and write data on both the microcontroller and app. This process was repeated 10 times in total, and the data between the two sets were compared. The results from the test showed that there was very minimal packet loss. There was an average of 0.62% missed packets across all 10 tests. Compared to the NIH Asthma Research Tool under low frequency testing, their device resulted in a 1% data loss using newest generation phones [18]. Our device outperformed the NIH device, and falls in compliance with the NIH wearable technology standards set by the study as well as the CITI biosensors standards for Bluetooth.

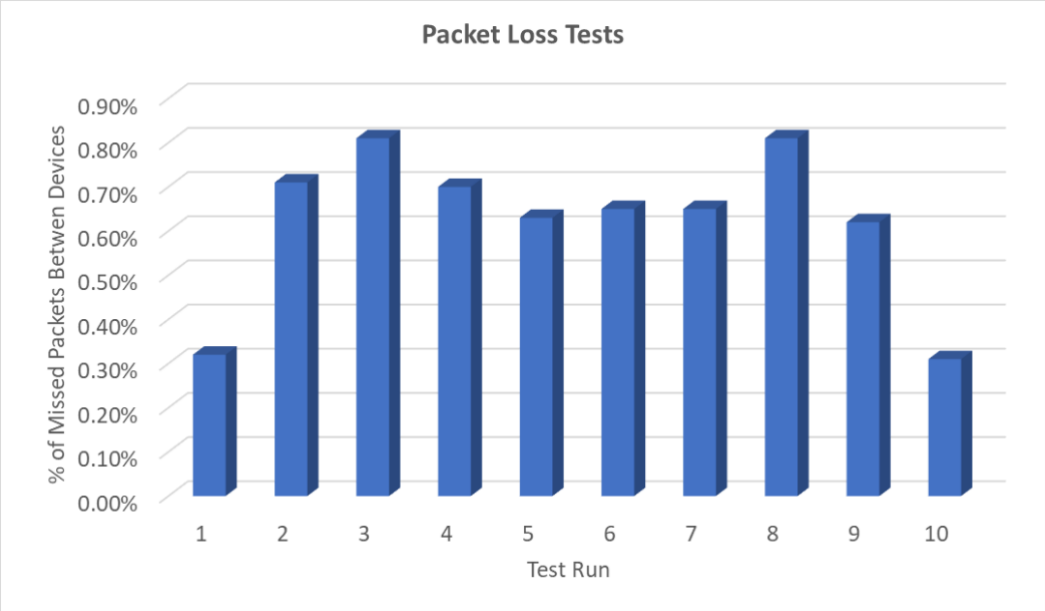

Figure 15: Packet Loss Between the Two Devices

7.5 Finite State Machine Testing

The system's Finite State Machine determines when a read or write action can take place. Given that this system queues actions in a given sequence, a more optimized FSM will result in better read and write times. The system originally ran an FSM that had less computational logic connecting the states. The system would always move to the write state opposed to having a flag set for writes. Similarly, the "sensor read" state was not separated from the "get next sensor" state. This caused an unintended redundancy when iterating to the next sensor value, causing there to be a lag when resetting to read the first sensor. This system was optimized to use the new FSM as listed in section 5.3.2. To test the optimization of the two FSMs, sensor read data was collected for a minute using both FSMs, in the same conditions. The timestamp of each sensor reading was collected and plotted against one another as seen in the figure below. The results from this test show that the newer FSM was much better optimized and able to get more sensor readings per second. Using the old FSM, a single flex sensor's reading would only update 1.224 times/second. Utilizing the new FSM, a single flex sensor's reading now updates 2.773 times/second.

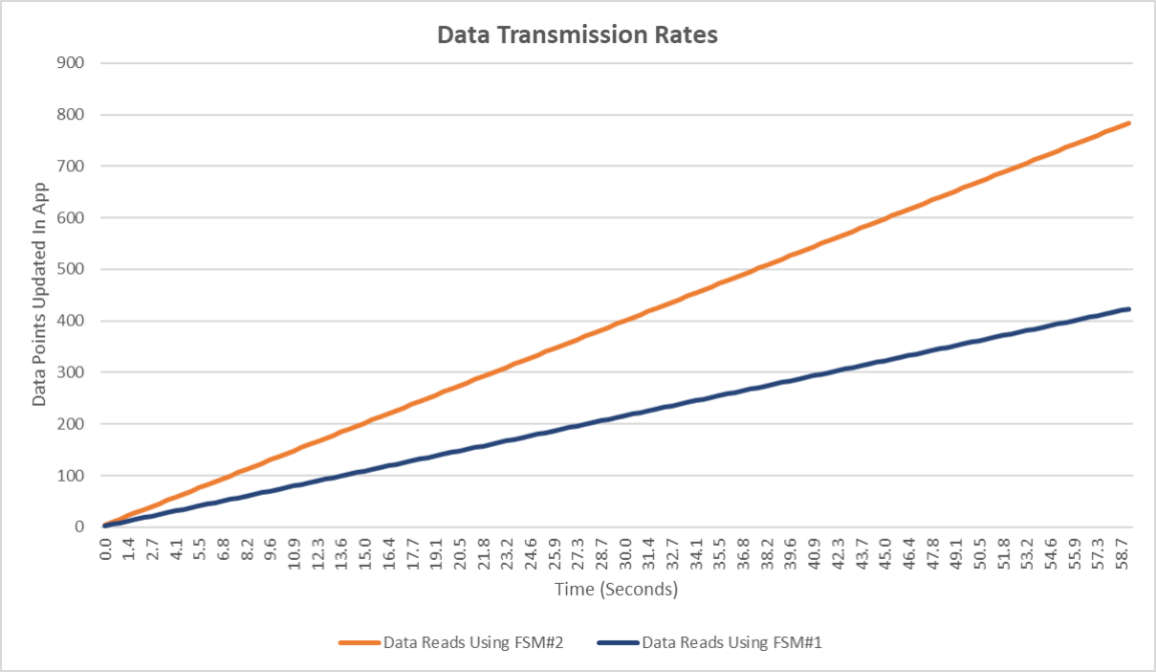

Figure 16: Data Transmission Rate Test Between FSMs

7.6 System Accuracy Testing

For overall system accuracy testing, the glove was affixed to a user's hand, and the same exercise was performed across all fingers. These exercises tested for two positions, halfway closed, and fully closed. Each test was conducted 20 times, for the two positions, for each of the five fingers, totaling 200 test points. The step size of each motor turn was hard coded to be 10° per motor step. This means a full 180° rotation of the motors would take 18 steps. The expected step value for each finger was parsed from the regressions for a halfway closed hand and a fully closed hand. The actual step value (or number of degrees turned on the motor) was recorded from the test trials and plotted against the expected. This is how we determined our accuracy metric for the device. The figure below shows a graph for thumb accuracy testing – with other graphs placed in Appendix 11.4 – alongside a table for the percent error. Error was found to be minimal in testing as it usually stayed within a 5% margin. The largest error was seen in the pinky. The largest standard deviations were found in the middle and ring fingers due to them being the longest. Error in testing found that even in the worst-case scenario, the fingers would always reach their general desired location. This is indicative of how human hands curl. As long as the fingers reach the general locations desired, the effects of the exercise will be the same. Given that the most common rehab glove on the market, the Syrebo [3], can only perform a full close and open, any degree of precision movement is better than what is available in commercial products. Likewise, the Syrebo glove uses an air pump to close fingers, meaning constant pressure is being applied even when fully closed. Regardless of if the servo motors overshoot their position, the tendon length is set up so that even a full 180° rotation cannot exceed a full closed position on any finger (as the motors stall due to torque). This puts the device in ISO IEC 60601-1-11:2015 compliance for medical electrical equipment and passes the system's safety requirements [6].

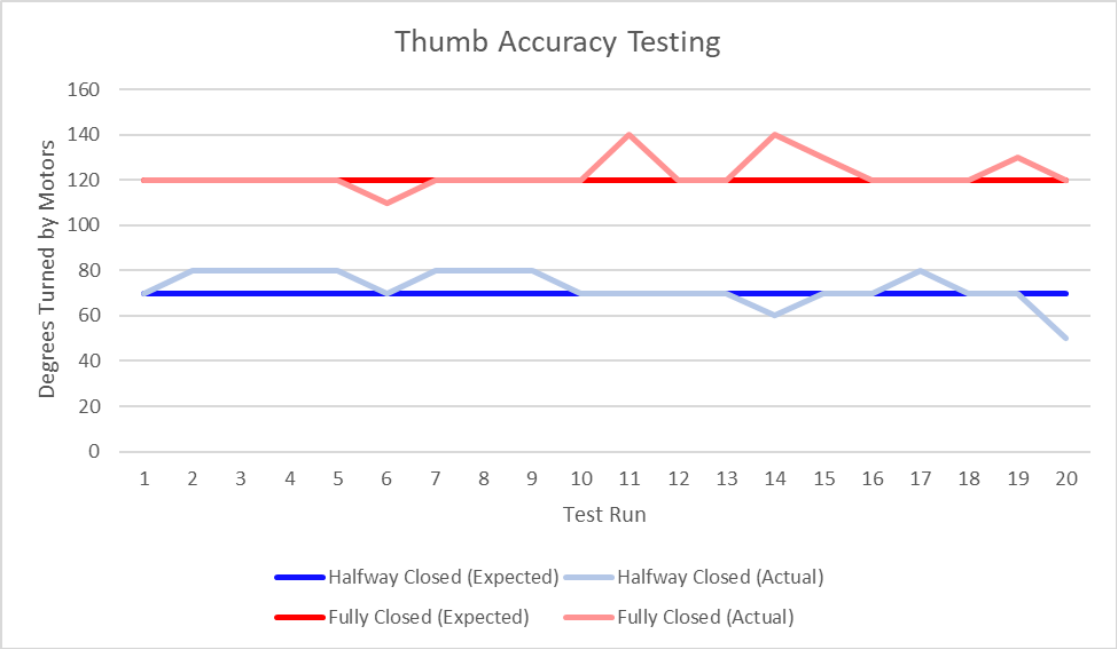

Figure 17: Accuracy Statistics of Thumb

	Half Closed			**Fully Closed**		
Finger	Mean (Degrees)	Std Dev (Degrees)	% Error	Mean (Degrees)	Std Dev (Degrees)	% Error
Thumb	72.5	122.5	7.66	6.98	5.00	5.00
Index	75.5	171.5	8.65	7.92	5.00	1.67
Middle	94.0	175.0	12.41	6.71	4.44	5.56
Ring	78.0	171.5	12.49	8.53	5.00	5.00
Pinky	65.5	130.5	8.65	8.65	7.86	0.71

Table 3: Accuracy Statistics of Full System Per Finger

8. New Skills Acquired and Learning Strategies

During the span of this project, the group's communication skills improved greatly. It seemed as if every member had a different way of explaining a problem. In the beginning, it was extremely hard to understand an issue that needed solving. Near the end, we learned to elaborate better so that the other members can understand easily.

8.1 Daniel Braunstein

Over the course of this project, I enhanced my CAD skills in Autodesk Fusion 360 to a higher level. My previous experience with CAD was an introductory course on Solidworks four years ago, and since then the most work I'd done in Fusion was sketching the profile for the lid of a box for Junior Design. In designing the exoskeleton, I learned to measure accurately and thoroughly and to use those measurements to construct original 3D models. I already had a basic grasp of Fusion from prior experience, so no research was necessary to develop my skills. I improved my abilities through trial and error, my own ability to interpret Fusion's interface and reference its documentation.

The second most significant skill I developed was the ability to formulate and execute an engineering test plan to gather data about – and extract metrics for the performance of – designed components. This plan came at the encouragement of Dr. Youngblood, who pushed me to be rigorous in testing my custom flex sensors. Without these rigorous tests, my understanding of their functionality would not have been nearly as thorough, and I could not have spoken about my designs as deeply as I did to judges at the Design Expo, or in this report.

8.2 Tyler Sheetz

During the span of this project, I learned how to better design models in CAD. The way I learned this was through another group member, Dan. As he was the main architect of the glove's wrist mount, Dan was using Fusion 360 a fair amount during the project. During this time, I was able to absorb some tactics on how to better design models in Fusion. With this knowledge, I did some rudimentary designing to help prototype our device. One example would be the creation of small blocks to prop up our motors during the prototype phase. I measured the necessary dimensions, created the model in CAD, and then printed it on my printer.

Another skill I acquired through this project was the ability to sell a product and advertise well. Though not an engineering skill exactly, it proved to be extremely useful for the exposition. The way I learned this was through observing another group member Bryan at the expo. I noticed some of the conversational tactics he was using and employed them myself to better explain and showcase our project.

8.3 Bryan Hess

Throughout the duration of this project, I acquired a much deeper understanding of mobile device application development, the Bluetooth communication protocols, and Arduino IDE programming. To learn more about these subjects, I learned primarily from YouTube tutorials [19] as well as the developer documentations for the UART Friend [13] and the Android Studio API developer [20]. From the YouTube tutorials, I learned how to write the basic skeleton for a BLE compatible application. Likewise, I learned how to make custom XML documents to create custom

graphics for the interface. The UART Friend documentation taught me about the UART and GATT protocols. I learned how to create services and characteristics from them as well as how to utilize Adafruit's libraries to operate the Bluetooth chip. The Android Developer resource aided me in making the finite state machine. I was able to adapt the APIs provided from the resource and use them to create each state of the machine. All of these resources provided a piece of the puzzle I needed; however, each were missing crucial parts. The YouTube tutorial was over nine years old, and as such, some of the Bluetooth documentation was out of date (this was the case for almost any tutorial online). The Android Developer resource filled in those gaps; however, the documentation was overly technical and hard to parse through. Lastly, the Adafruit documentation was severely lacking in examples and tutorials. Similarly, they mix up terminology often for sending and receiving data, so it was confusing to read as they would miss attribute some functions. Overall, I greatly expanded my knowledge in these subjects and feel comfortable applying them in industry.

8.4 Michael Etter

Over the course of the semester, I learned more about hardware design communication. In the beginning my PCBs were lacking in compactivity and were too complicated. I had to spend time reducing the board to a more manageable size and redesign the schematic to account for my teammates' modules. For example, I manufactured a couple PCBs that were incorrect. The flex sensors did not have an input to the microcontroller. This board had to be scrapped as it would fail the flex sensor functionality test. I learned a great deal in testing PCBs and making sure they are usable before soldering components. My communication skills also improved. Since the PCB would integrate everyone's components, I needed to communicate efficiently. There were a few mistakes, but I eventually improved my communication and got the PCB designed and manufactured to make it easier for Tyler, Dan and Bryan to integrate their modules.

9. Conclusions and Future Work

The final iteration of our device is able to articulate the user's fingers based off of an app command. The motors have been adjusted to set speed and reset when the position becomes unfavorable. Our hand fabricated flex sensors ended up working better than we initially anticipated. For these reasons, our team would deem the project a success when compared to our goals in the conceptual document.

Some limitations of our device are flex sensor reliability, hand size, and quality of manufacturing. With the flex sensors being made by hand, they are much more variable in readings than that of market fabricated ones. A standardized process for fabrication would minimize differences across different sensors. Hand size would be an issue should the glove go to public due to no feasible "one-size fits all" application. This means that there would have to be at least a small, medium, and large option. Similarly, the quality of the glove is not well suited for modification. If given more time, a custom glove, made wholly from differing types of fabric would be created in order to ease putting it on and reduce interference from glove resistiveness when curling. Lastly, the quality of the wrist mount is relatively poor due to our group not having access to high quality manufacturing methods. The entire mount is from a 3D printer that used PLA as its material. This means the mount is much bulkier and more cumbersome than it could be.

If we had to start over again, we would have begun the 3D printed aspects of the glove sooner than we did. There was a press for time near the end of the term in getting the mount fully printed which led to last minute implementations. Regarding possible improvements to the design, there were difficulties encountered with the motor bed due to the exposed strings. As discussed in section 5.5.2, a fourth revision of the bed had been designed and printed to remedy this, but there was not enough time to implement it into the design. Likewise, a more permanent affixing for flex sensors would also help to reduce variability between users. Other than these aspects, aesthetic improvements would be been pursued to make the device more marketable and appealing to consumers. Once complete, we would pursue FDA approval for the device. Should this be successful, marketing and advertising would commence with rehabilitation facilities.

10. References

[1] Texas Orthopedics, "Wrist and Hand Physical Therapy Exercises," Fort Worth Hand Center, 24 September 2022. [Online]. Available: https://fortworthhandcenter.com/orthopedic/wrist-and-hand-physical-therapy-fort-worth/.

[2] "Finger Support Training Brace, Stroke Rehab Arthritis Glove," Xemz, [Online]. Available: https://www.amazon.com/XEMZ-Arthritis-Separator-Orthotics-Corrector/dp/B088JZDJYN.

[3] "Hand Rehabilitation Exercise Treatment Mechanical Gloves, Soft Exoskeleton Robot Stroke Hemiplegia Hand Electro-Pneumatic Training Equipment, Suitable For Passive Finger Exercises(Color:Orange,Size:Le," Asewq, [Online]. Available: https://www.amazon.ca/Rehabilitation-Mechanical-Exoskeleton-Hemiplegia-Electro-Pneumatic/dp/B09PV4WQFB?th=1.

[4] A. S. Sadun, J. Jalani and J. A. Sukor, "A Comparative Study on the Position Control Method of DC Servo Motor with Position Feedback by using Arduino," in *Proceedings of Engineering Technology International Conference*, Bali, 2015.

[5] RoHS Guide, "RoHS 2 vs. RoHS 3," 16 December 2022. [Online]. Available: https://www.rohsguide.com/rohs3.htm.

[6] ISO, "IEC 60601-1-11:2015," 9 September 2020. [Online]. Available: https://www.iso.org/standard/65529.html.

[7] Collaborative Insitutional Training Initiative (CITI Program), "Research, Ethics, and Compliance Training," 22 June 2020. [Online]. Available: https://www.citiprogram.org/verify/?kca0e43e9-f102-46fa-8b6d-78c667b94ce9-37150978.

[8] Adafruit, "Short Flex Sensor," [Online]. Available: https://www.adafruit.com/product/1070.

[9] DFROBOT, DFRobot DF9GMS 360 Degree Micro Servo (1.6Kg).

[10] YouBionic, "Exohand Servo," 2022. [Online]. Available: https://www.youbionic.com/exohandservo.

[11] TwidgeVR, Composer, *How to Make a DIY Flex Sensor for VR Gloves with Velostat and Copper Foil.* [Sound Recording]. YouTube. 2021.

[12] Adafruit, "Pressure-Sensitive Conductive Sheet (Velostat/Linqstat)," [Online]. Available: https://www.adafruit.com/product/1361.

[13] K. Townsend, "Introducing the Adafruit Bluefruit Le Uart Friend," Adafruit Learning System, [Online]. Available: https://learn.adafruit.com/introducing-the-adafruit-bluefruit-le-uart-friend/ble-gatt.

[14] InMoov Open Source 3D Printed Life-Size Robot, "InMoov body parts library: Forearm-and-Servo-Bed," [Online]. Available: https://inmoov.fr/inmoov-stl-parts-viewer/?bodyparts=Forearm-and-Servo-Bed.

[15] "3M 03429NA 051131034297 Scotch Electrical Tape, 3/4-in by 66-ft, Black, 1-Roll, 3/4 Foot," 3M, [Online]. Available: https://www.amazon.com/Scotch-Electrical-Tape-4-Inch-66-

Foot/dp/B001ULCB1O/ref=sr_1_4?keywords=electrical%2Btape&qid=1671244131&sr=8-4&th=1.

[16] Adafruit, "Long Flex sensor," [Online]. Available: https://www.adafruit.com/product/182.

[17] L. Ada, "All the Internet of Things - Episode One: Transports," Adafruit Learning System, [Online]. Available: https://learn.adafruit.com/alltheiot-transports/bluetooth-btle.

[18] V. V. Tipparaju, K. R. Mallires, D. Wang, F. Tsow and X. Xian, "Mitigation of Data Packet Loss in Bluetooth Low Energy-Based Wearable Healthcare Ecosystem," *Biosensors*, vol. 11, no. 10, p. 350, 2021.

[19] MarakanaTechTV, Composer, *Developing Bluetooth Smart Applications for Android Tutorial.* [Sound Recording]. YouTube. 2013.

[20] Android Developers, "android.bluetooth.le," [Online]. Available: https://developer.android.com/reference/android/bluetooth/le/package-summary.

11. Appendix: Figures and Tables

This appendix collects various figures and tables that would be inconvenient to present in the main body of the text. Each set is organized into subsections which indicates the related module and relevant section in the text.

11.1 Motors and Regression Testing

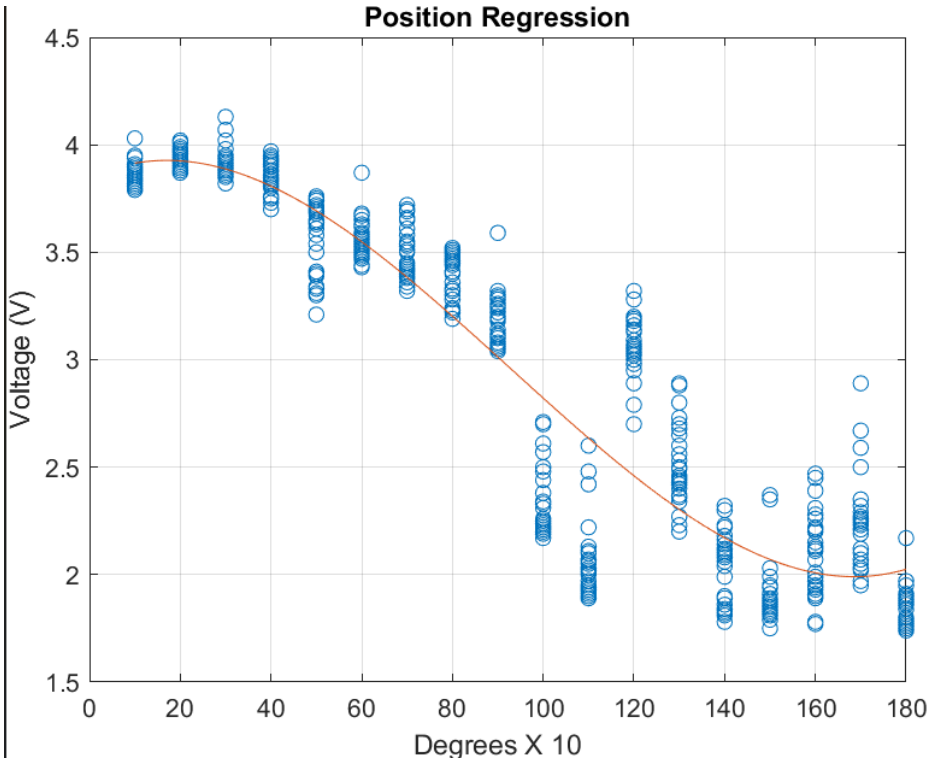

Figure 18: thumb sensor voltage vs. motor rotation with trinomial best-fit line

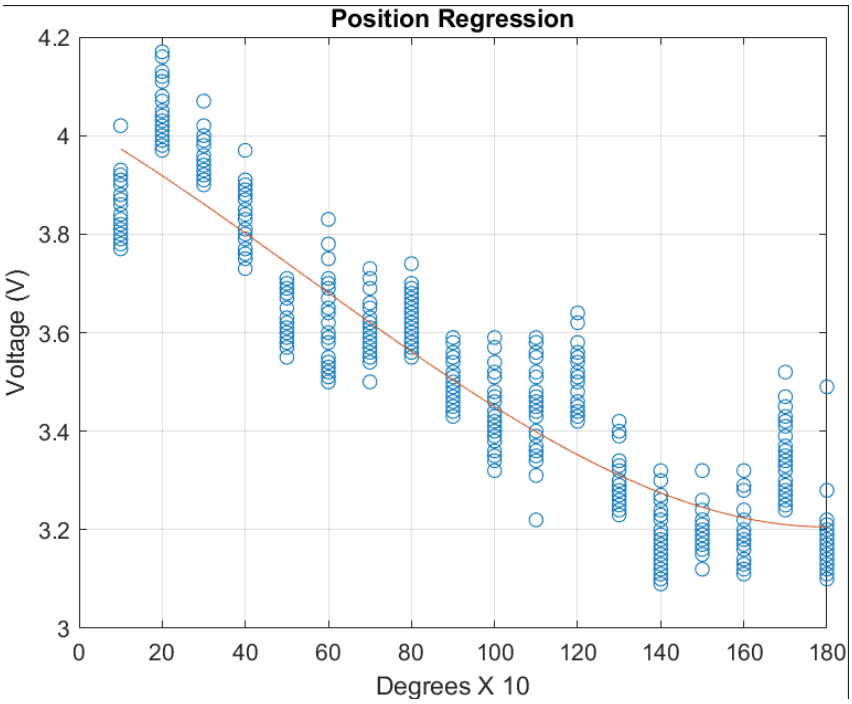

Figure 19: index finger sensor voltage vs. motor rotation with trinomial best-fit line

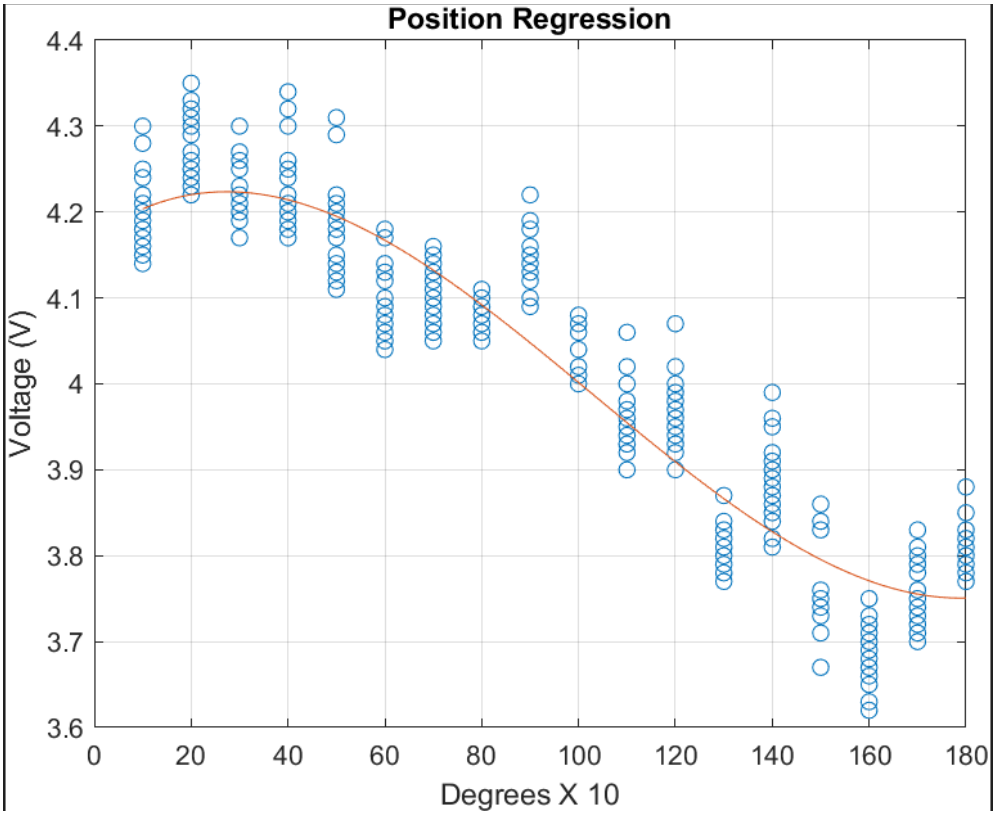

Figure 20: middle finger sensor voltage vs. motor rotation with trinomial best-fit line

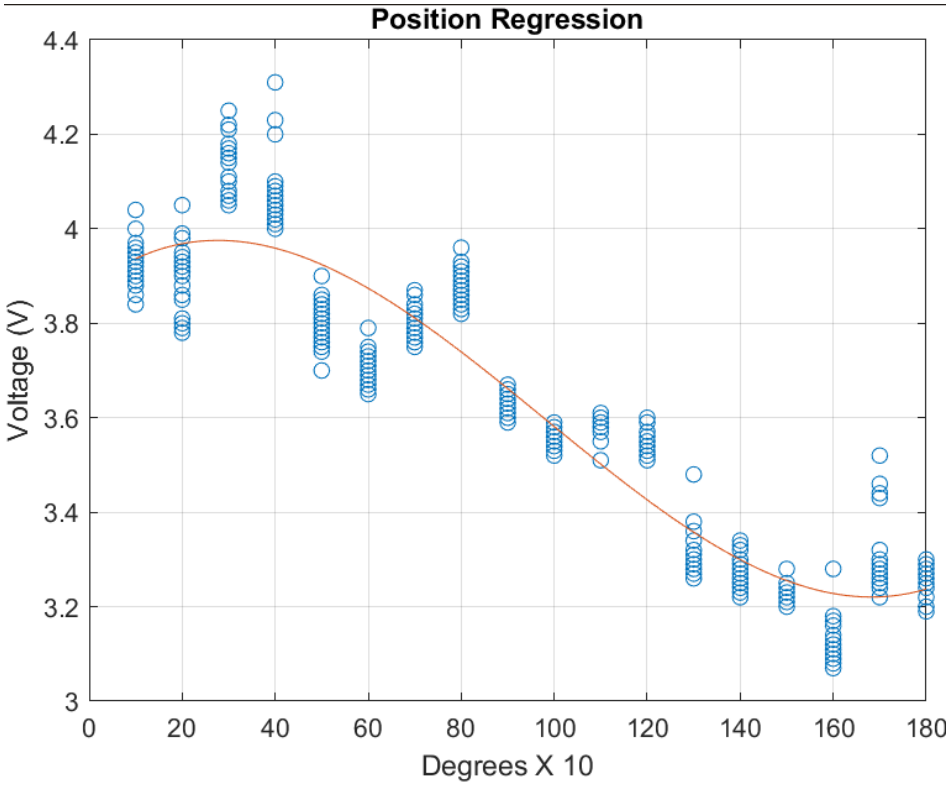

Figure 21: ring finger sensor voltage vs. motor rotation with trinomial best-fit line

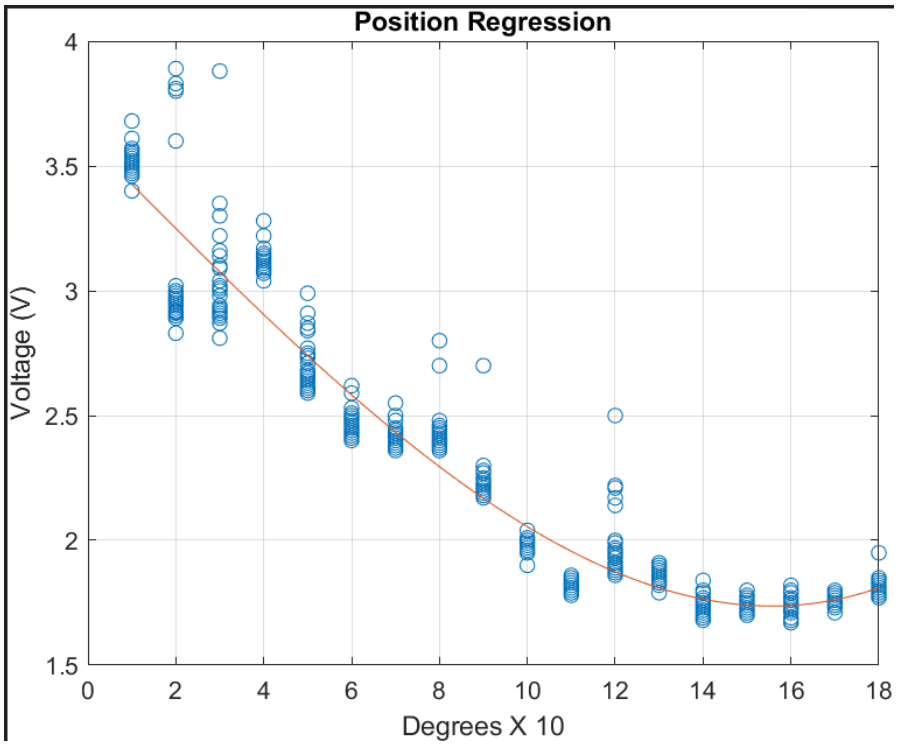

Figure 22: pinky sensor voltage vs. motor rotation with trinomial best-fit line

11.2 Exoskeleton

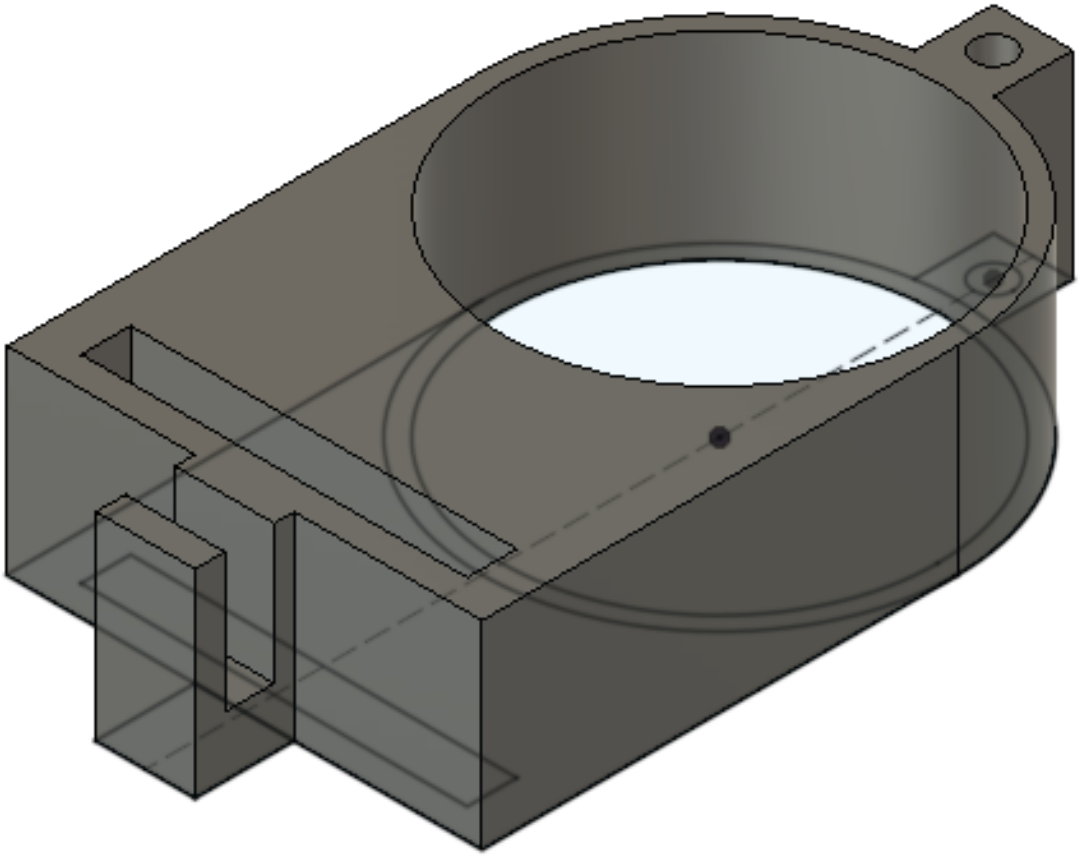

Figure 23: Fusion 360 model of one of the finger rings

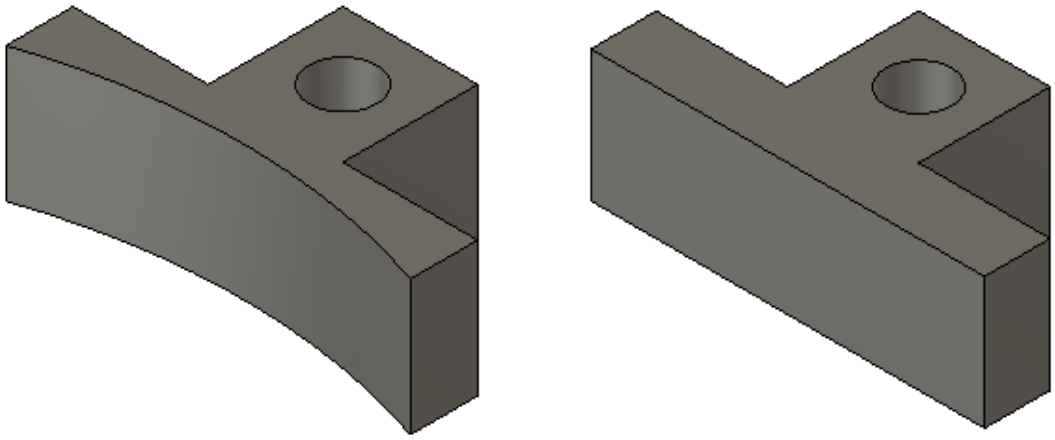

Figure 24: Fusion 360 model of the revised guide loops for the fingers (left) and palm (right)

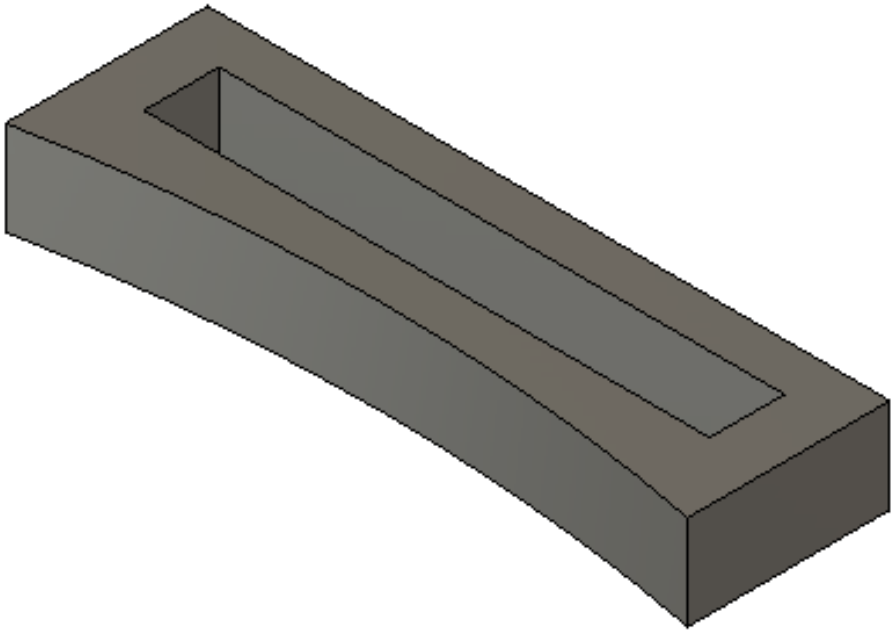

Figure 25: Fusion 360 model of the revised flex sensor guide slot

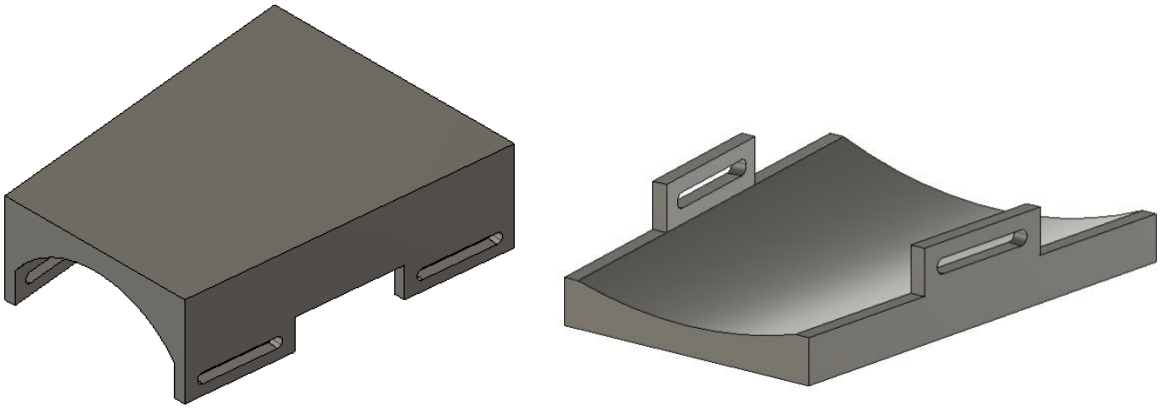

Figure 26: Fusion 360 model of the original wrist mounts.

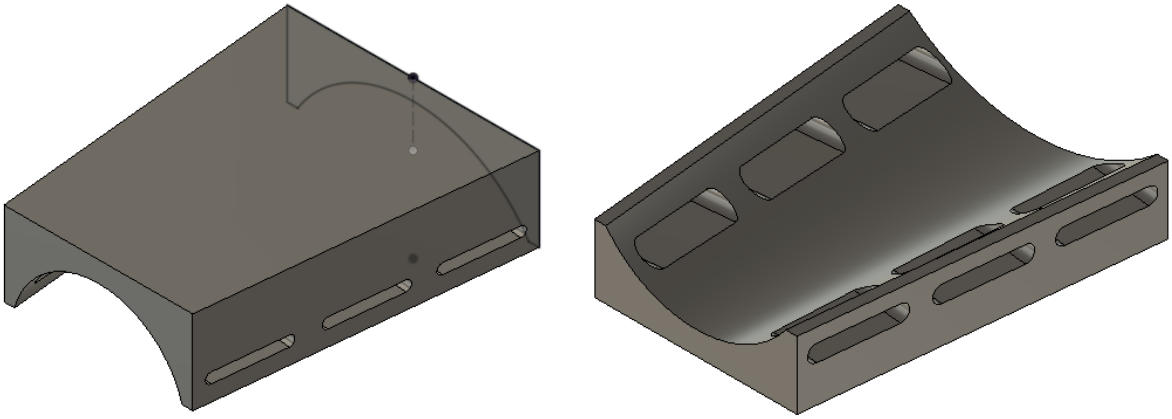

Figure 27: Fusion 360 model of the top (left) and bottom (right) second-revision wrist mounts

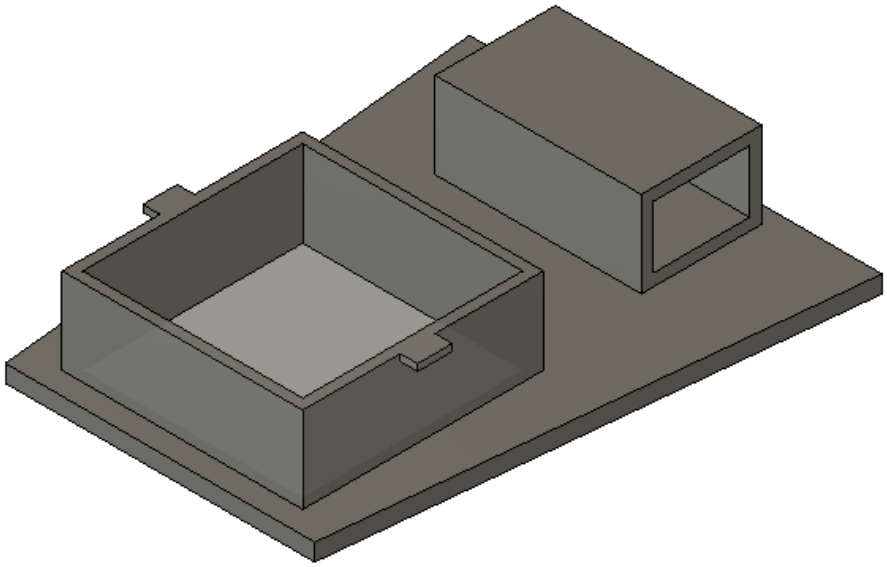

Figure 28: Fusion 360 model of the PCB bed

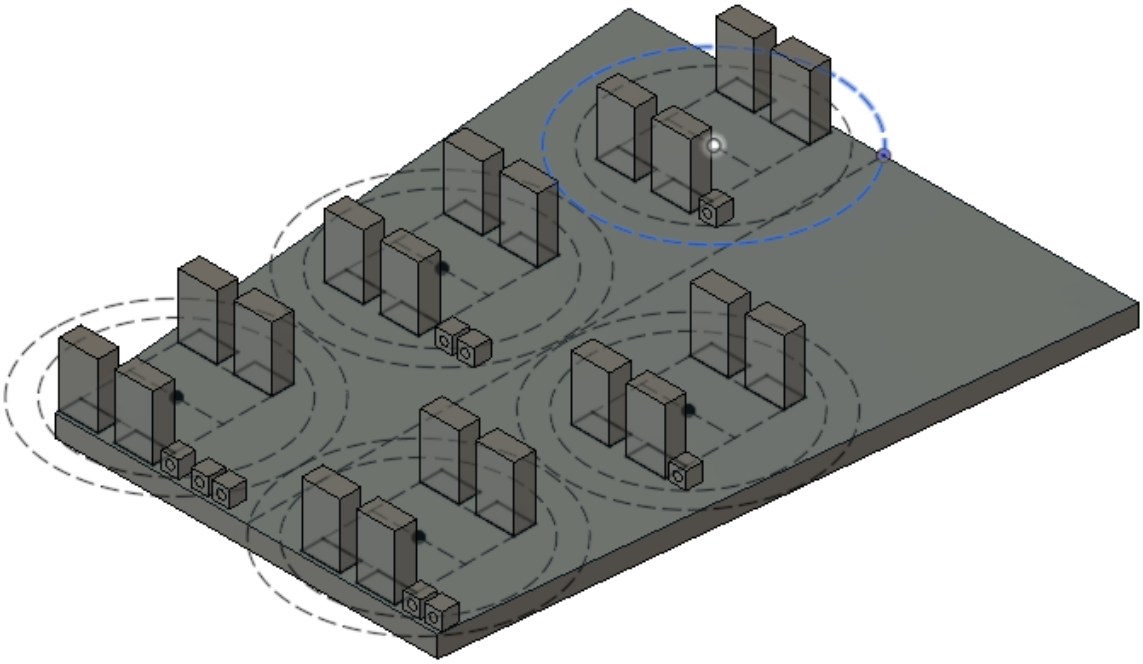

Figure 29: Fusion 360 model of the first motor bed revision.

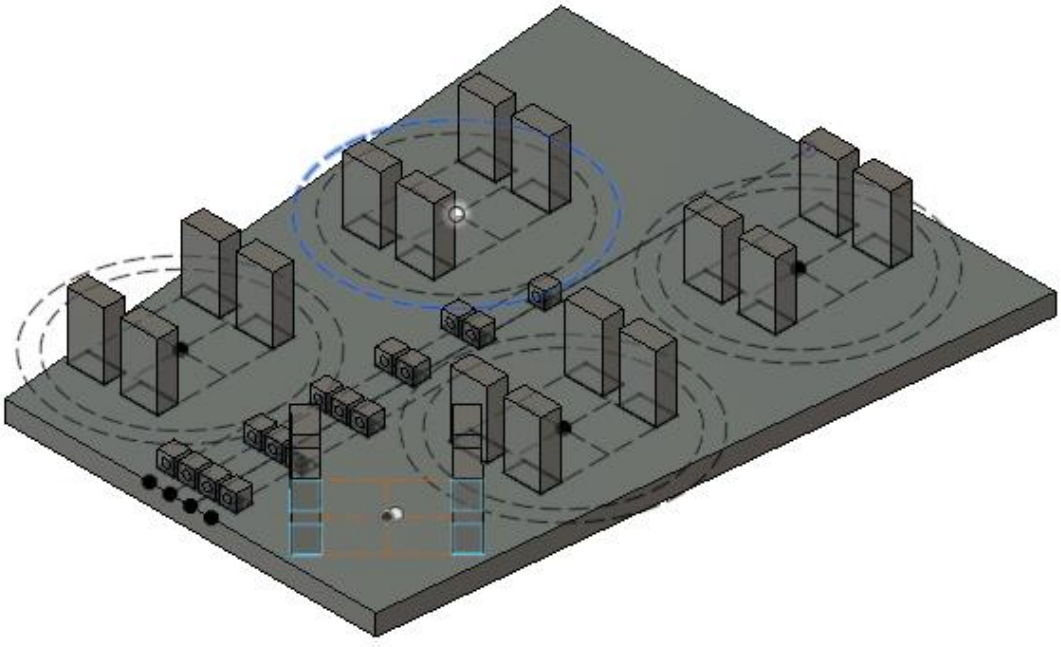

Figure 30: Fusion 360 model of the second motor bed revision.

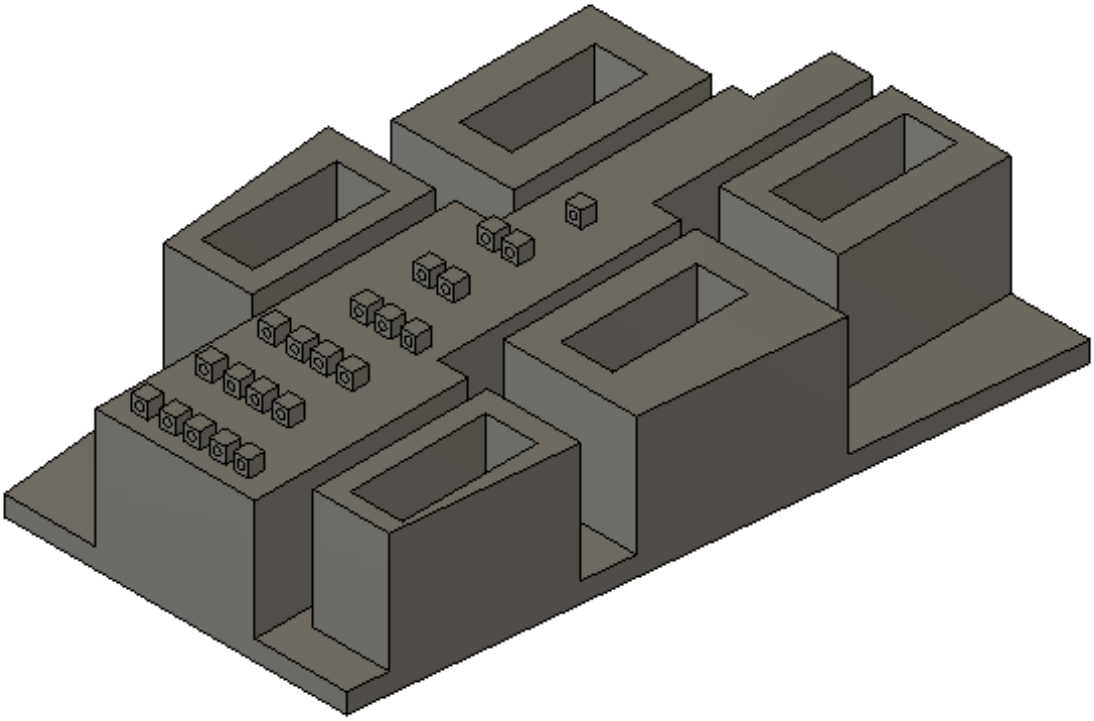

Figure 31: Fusion 360 model of the third revision motor bed.

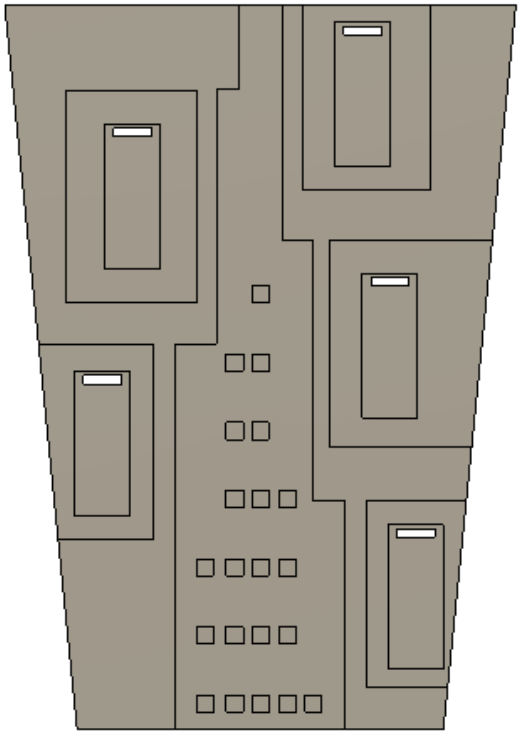

Figure 32: Top-down view of the Fusion 360 model of the third revision motor bed.

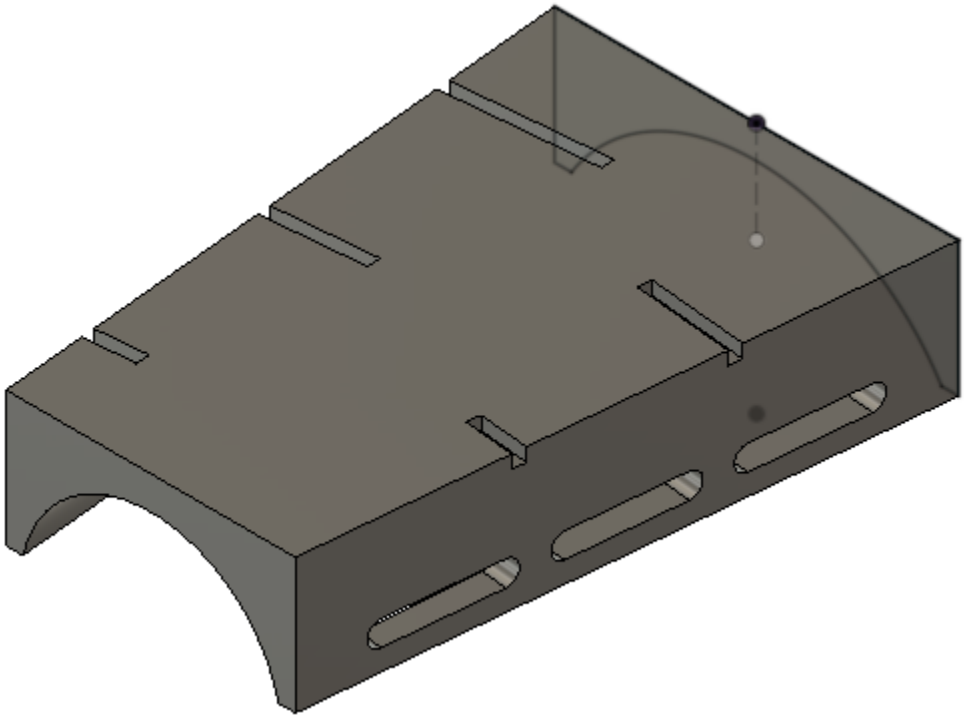

Figure 33: Fusion 360 model of the third revision top wrist mount.

The slots cut into Figure 33 accommodate the servo motor power wires, which slip through the holes shown in each motor bed in Figure 32.

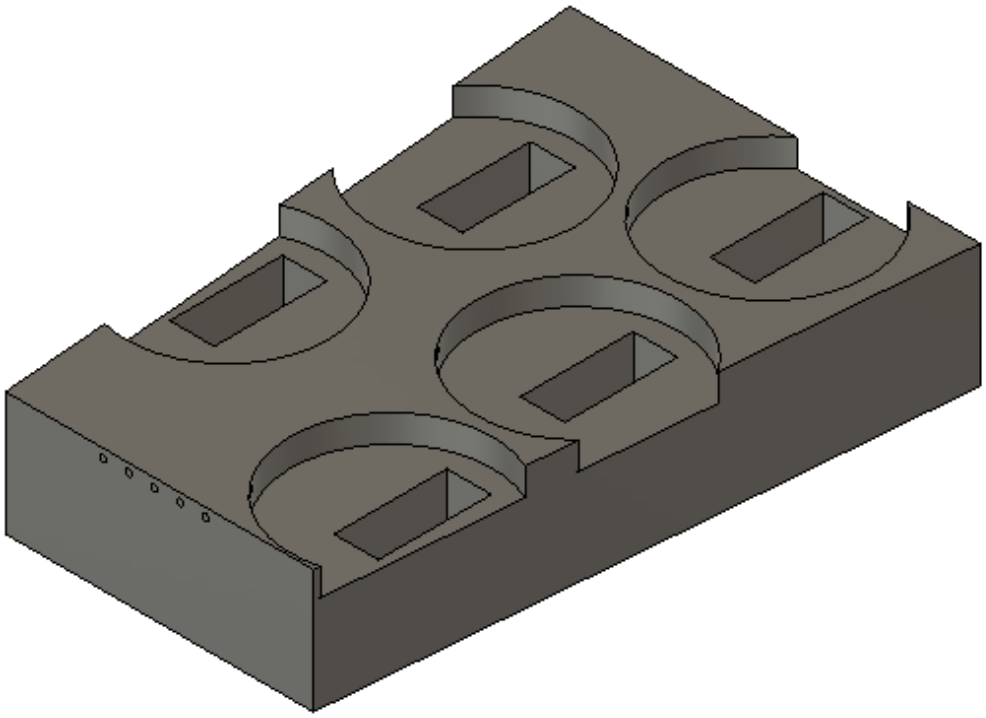

Figure 34: Fusion 360 model of the fourth revision motor bed.

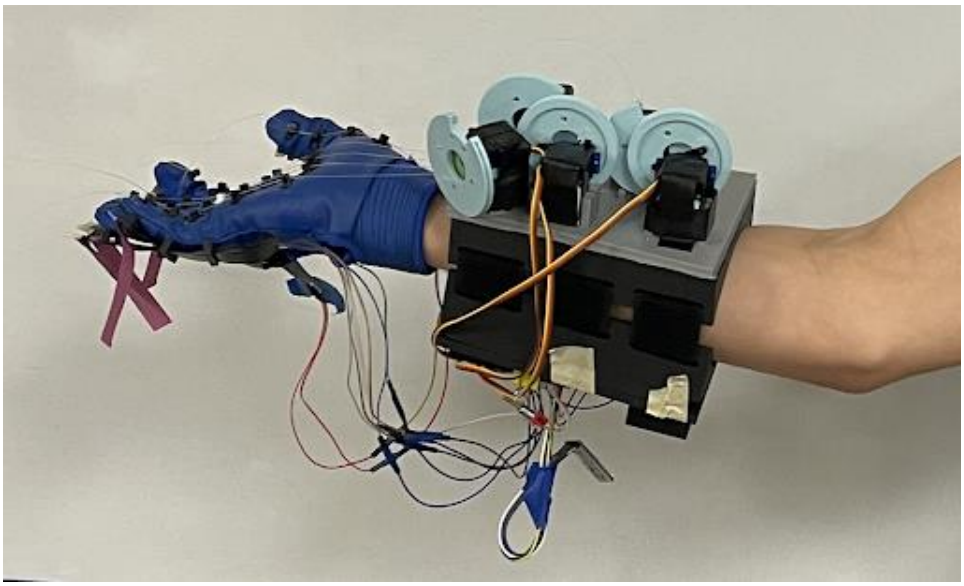

Figure 35: second revision exoskeleton being worn by a user.

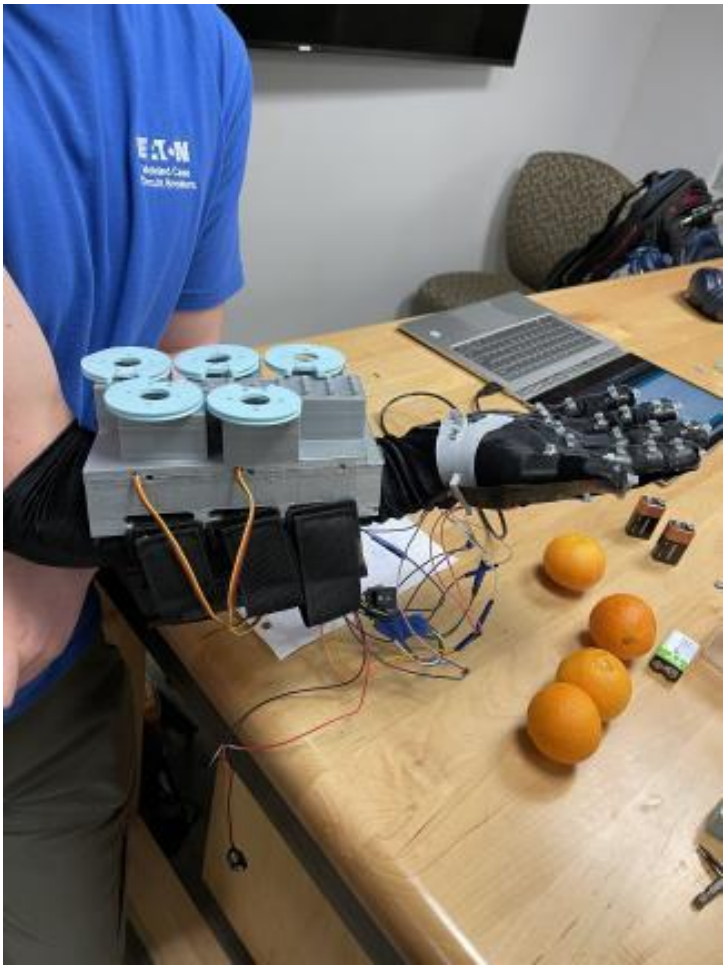

Figure 36: third revision exoskeleton being worn by a user.

11.3 Flex Sensor Testing

Short - 5 cm				
Radius (mm)	Angle (deg)	Mean (kOhm)	Deviation	Precision(%)
0	0	2.2668	0.409163	18.050246
	45	0.9816	0.3958291	40.324887
	90	0.4368	0.2079207	47.600883
	135	0.4308	0.2318929	53.828444
	180	0.52944	0.2027712	38.29919
25	Angle (deg)	Mean (kOhm)	Deviation	Precision (%)
	45	1.02	0.3074221	30.139419
	90	0.4024	0.1332066	33.103033
	135	0.1436	0.0495715	34.520541
	180	0.09216	0.0306862	33.296607

Table 4: metrics for the short flex sensor (5 cm)

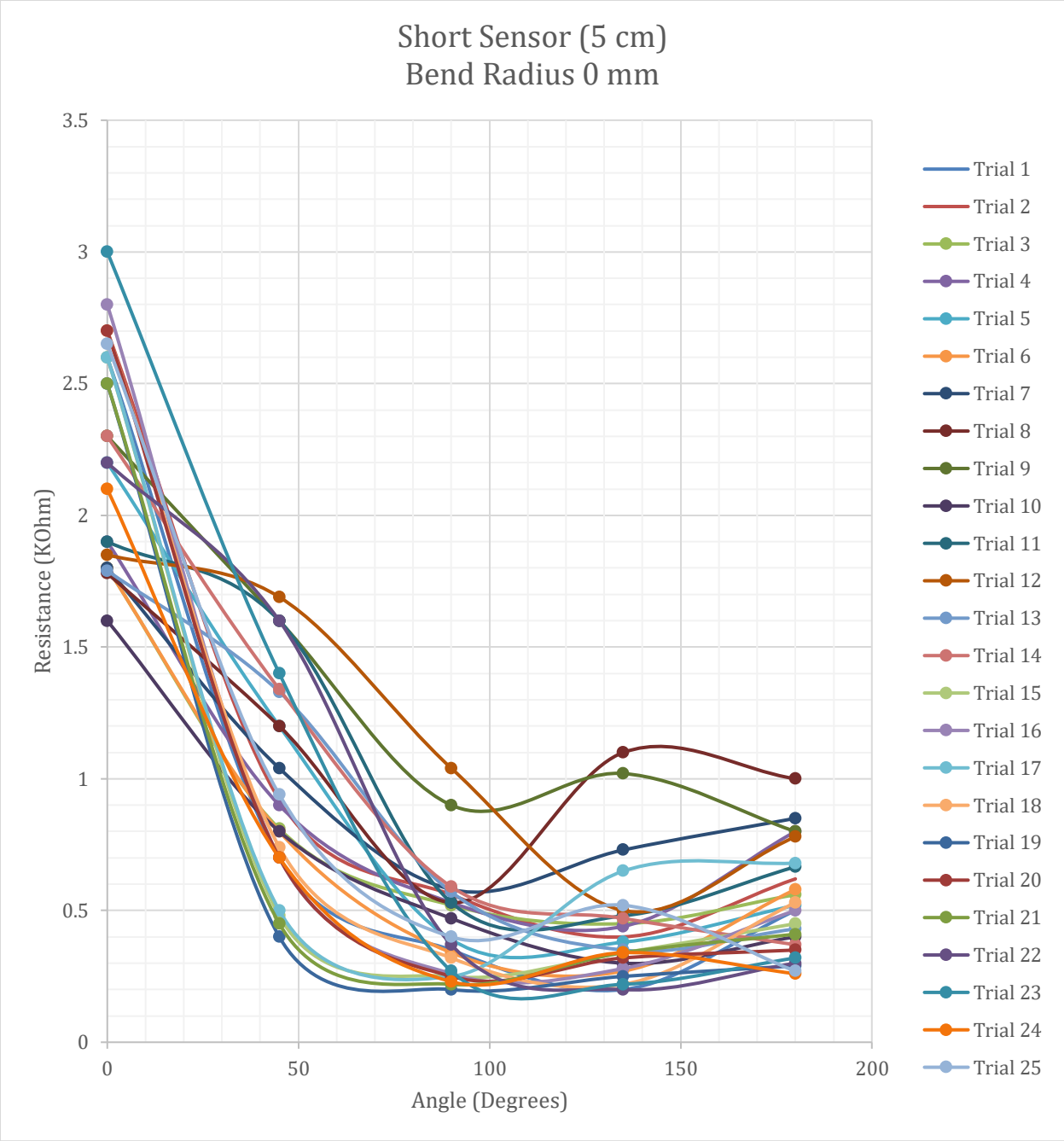

Figure 37: short sensor resistance vs angle for a radius of 0 mm

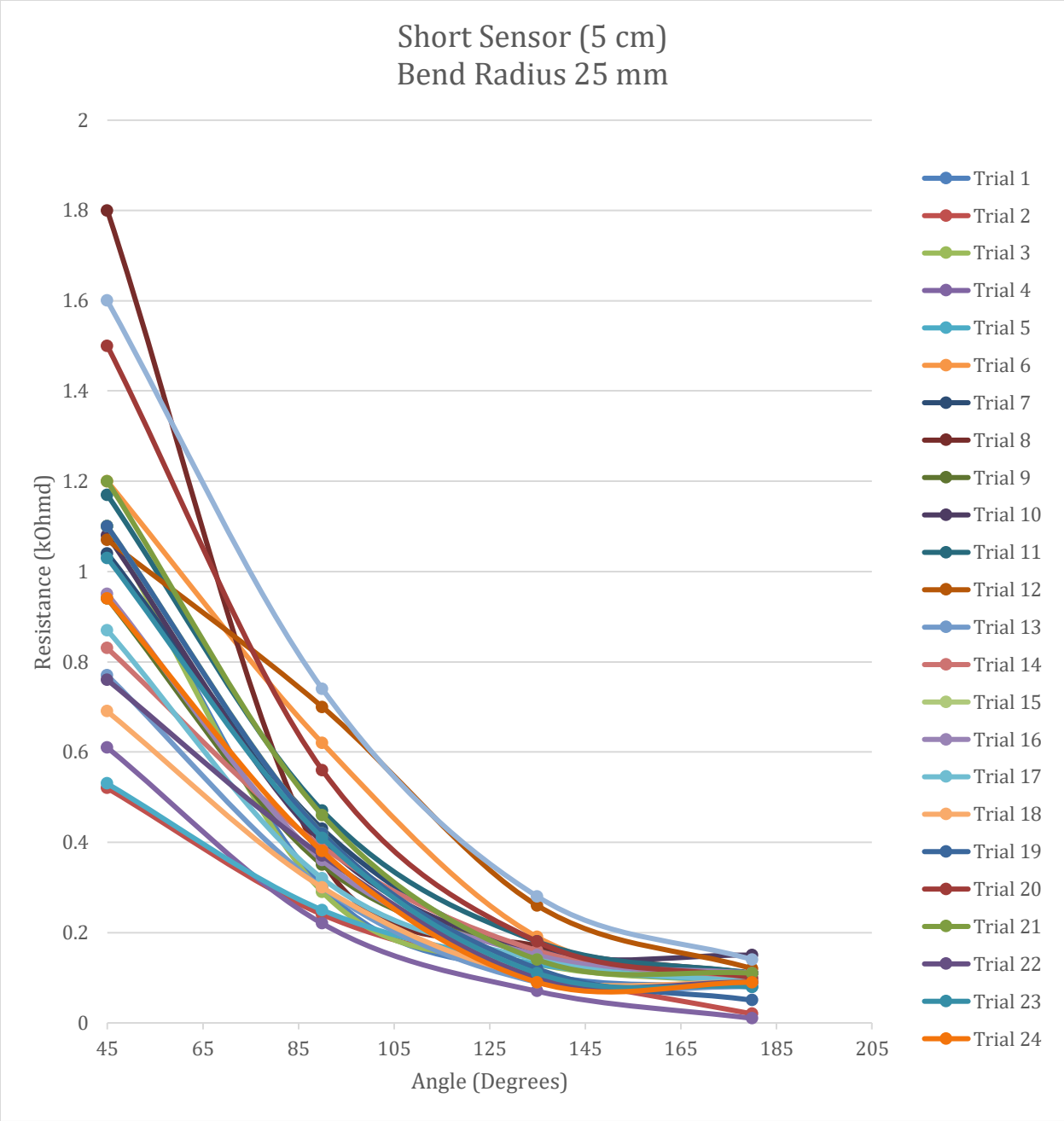

Figure 38: short sensor resistance vs angle for a radius of 25 mm

Medium - 10 cm				
Radius (mm)	Angle (deg)	Mean (kOhm)	Deviation	Precision (%)
0	0	1.3232	0.1802017	13.618632
	45	0.65384	0.2577214	39.416581
	90	0.1984	0.0400708	20.196961
	135	0.134	0.0618466	46.154167
	180	0.1678	0.0852946	50.831101
25	Angle (deg)	Mean (kOhm)	Deviation	Precision (%)
	45	1.1788	0.2579619	21.883431
	90	0.41532	0.1570803	37.821516
	135	0.10324	0.0384592	37.252185
	180	0.06736	0.021391	31.756181
50	Angle (deg)	Mean (kOhm)	Deviation	Precision (%)
	45	0.9164	0.2042645	22.289888
	90	0.406	0.0589491	14.51949
	135	0.174	0.031225	17.945397
	180	0.0756	0.0437874	57.919795

Table 5: metrics for the medium flex sensor (10 cm)

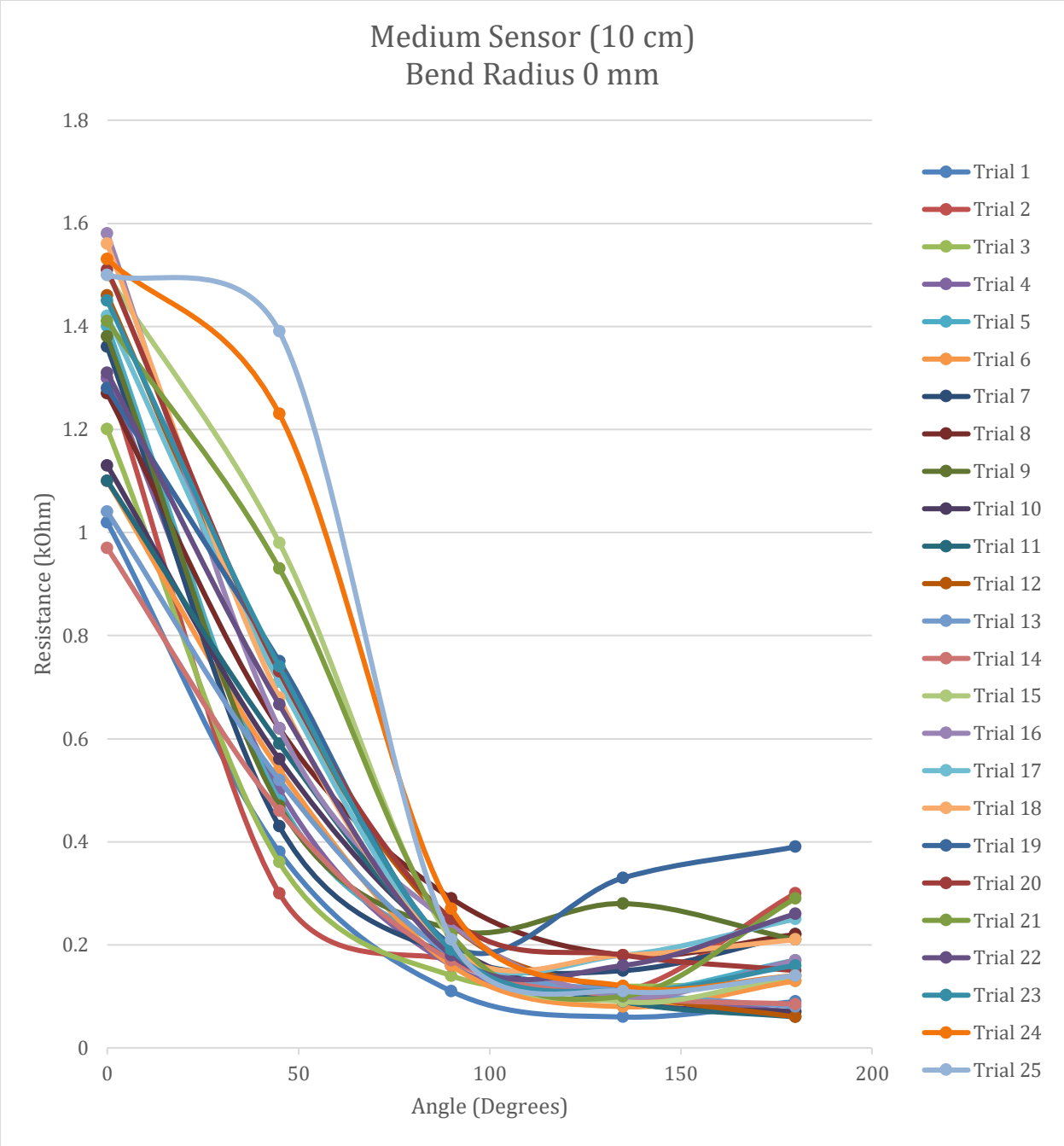

Figure 39: medium sensor resistance vs angle for a radius of 0 mm

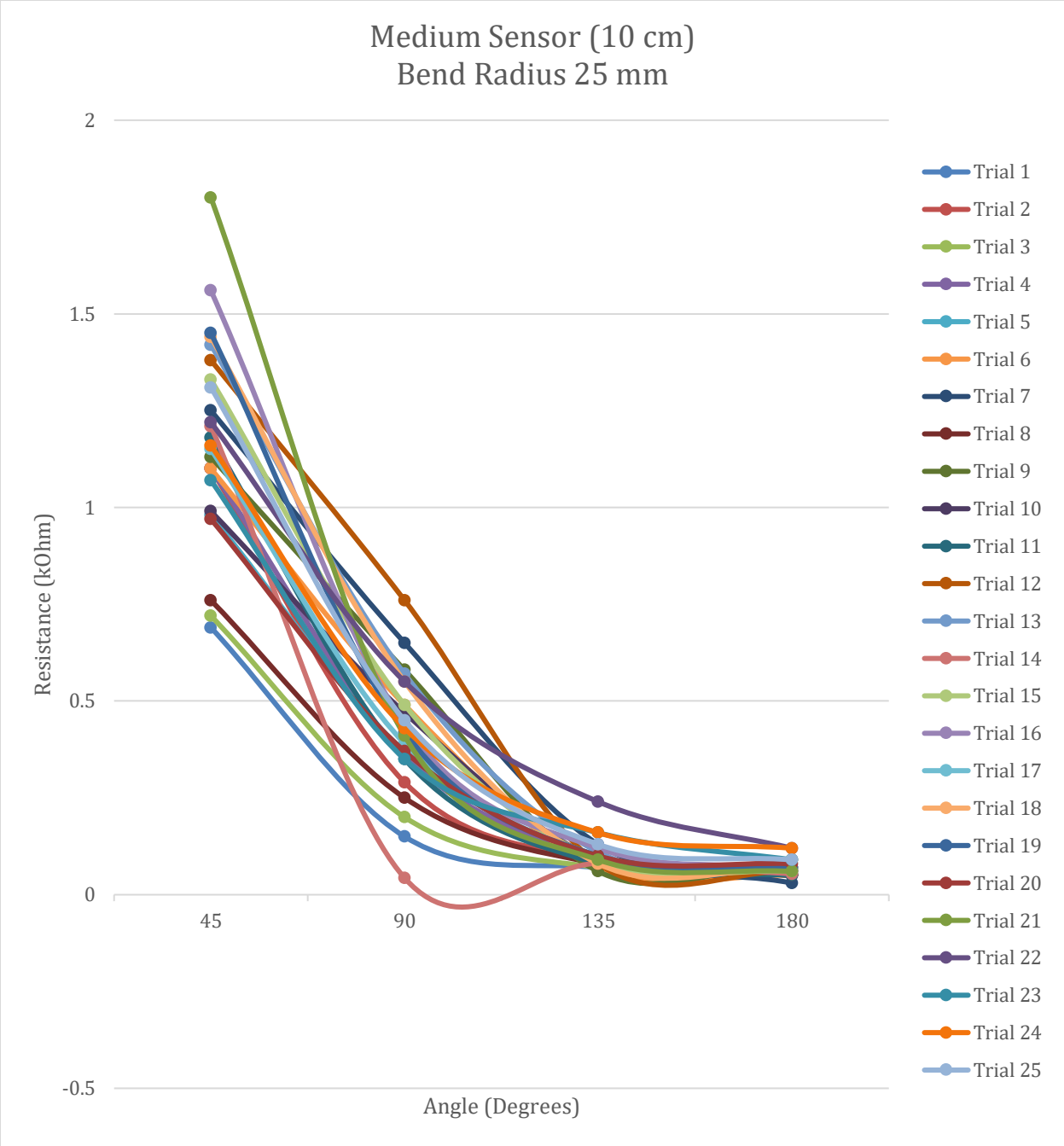

Figure 40: medium sensor resistance vs angle for a radius of 25 mm

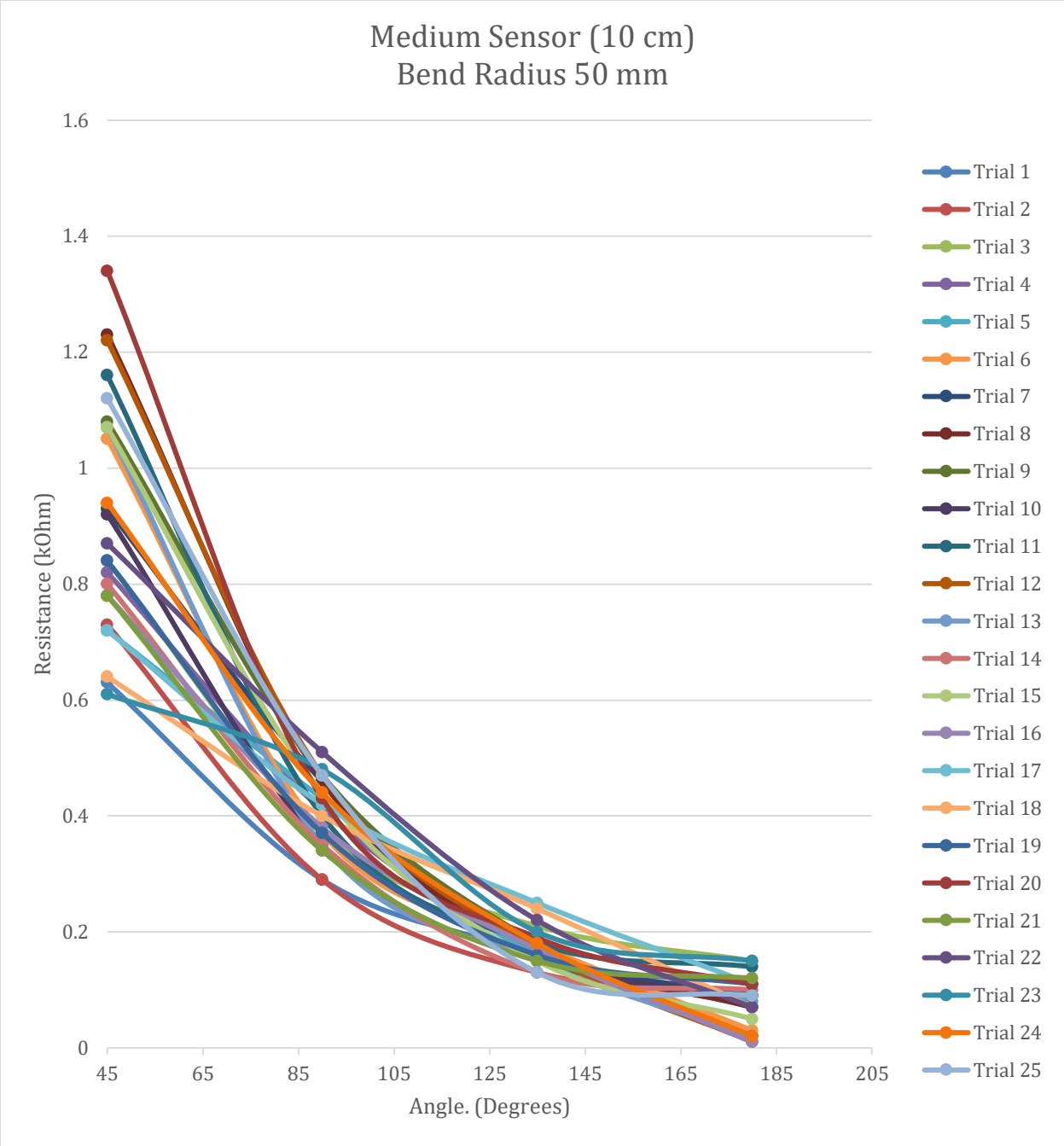

Figure 41: medium sensor resistance vs angle for a radius of 50 mm

Long - 15 cm				
Radius (mm)	Angle (deg)	Mean (kOhm)	Deviation	Precision (%)
0	0	0.8928	0.1227301	13.746646
	45	0.58144	0.0849127	14.60387
	90	0.2428	0.0411825	16.961499
	135	0.1244	0.0173397	13.9387
	180	0.0892	0.0193477	21.690242
25	Angle (deg)	Mean (kOhm)	Deviation	Precision (%)
	45	0.64832	0.0729147	11.246721
	90	0.364	0.0767572	21.087141
	135	0.154	0.0295804	19.208051
	180	0.0844	0.022	26.066351
50	Angle (deg)	Mean (kOhm)	Deviation	Precision (%)
	45	0.64728	0.0592055	9.1468171
	90	0.4216	0.0768483	18.227784
	135	0.1712	0.0330807	19.322844
	180	0.0884	0.0407922	46.144973
75	Angle (deg)	Mean (kOhm)	Deviation	Precision (%)
	45	0.57864	0.0628986	10.870083
	90	0.3552	0.0801103	22.553587
	135	0.1708	0.0419245	24.545981
	180	0.0808	0.0361617	44.754547

Table 6: metrics for the long flex sensor (15 cm)

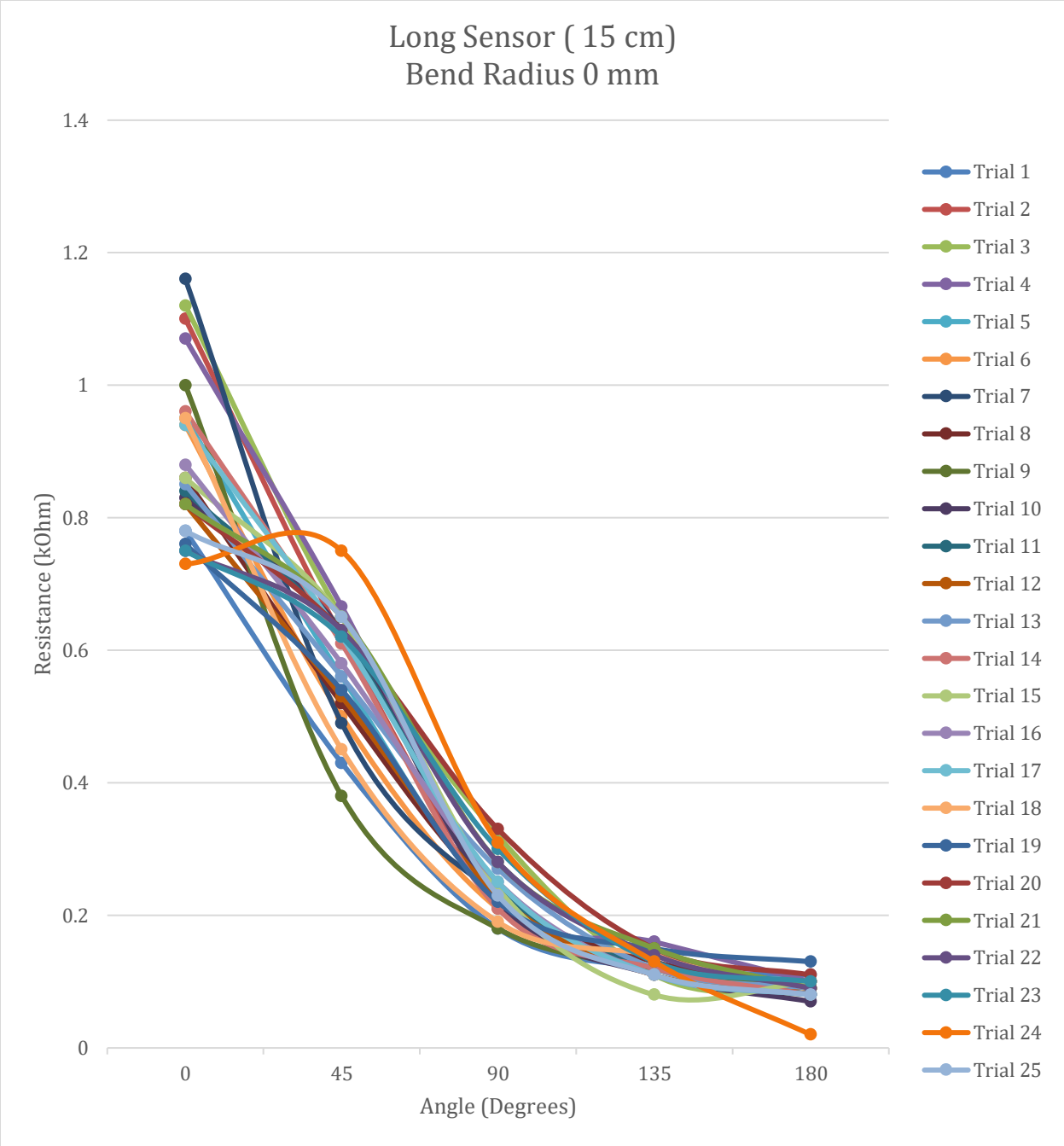

Figure 42: long sensor resistance vs angle for a radius of 0 mm

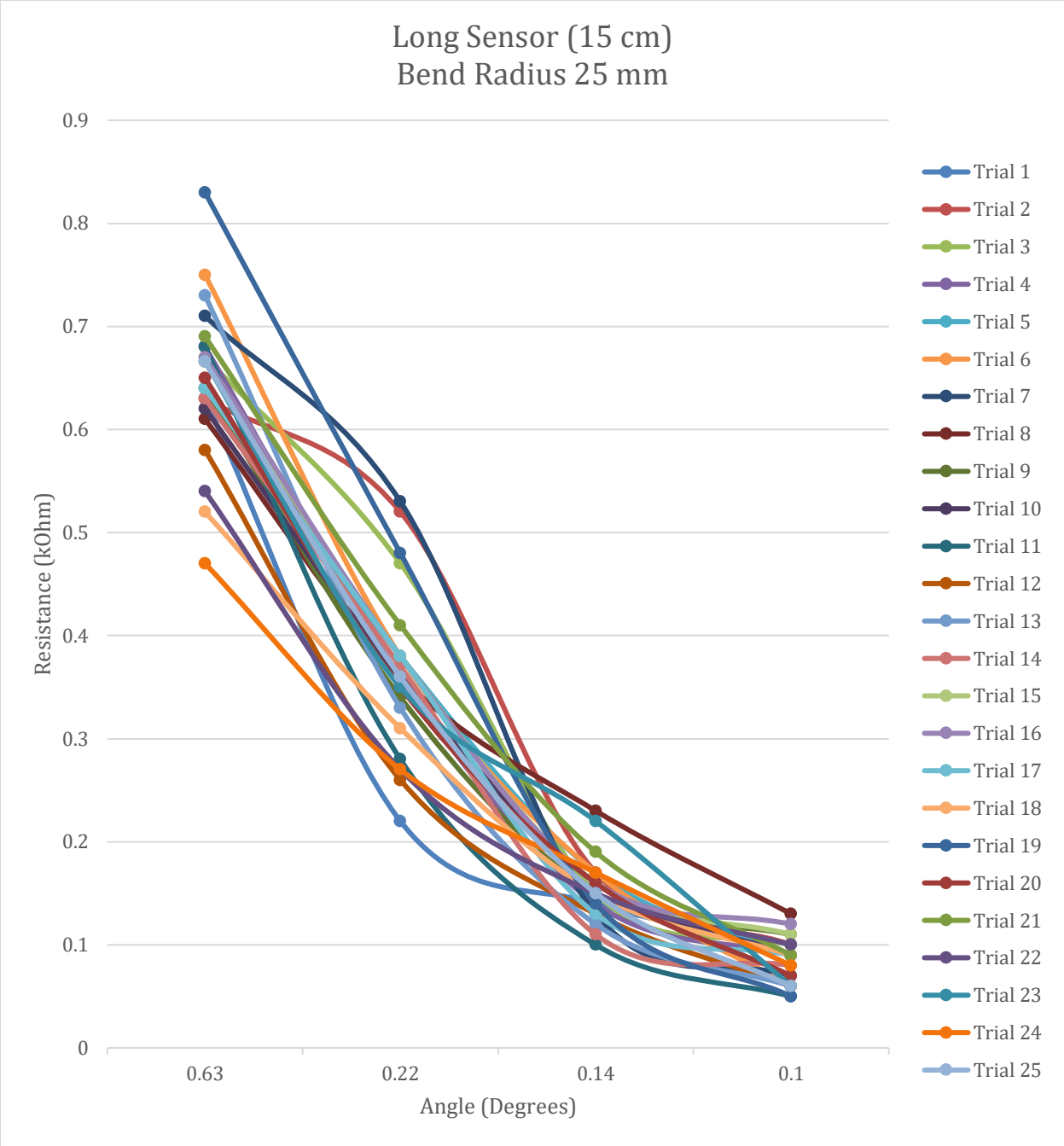

Figure 43: long sensor resistance vs angle for a radius of 25 mm

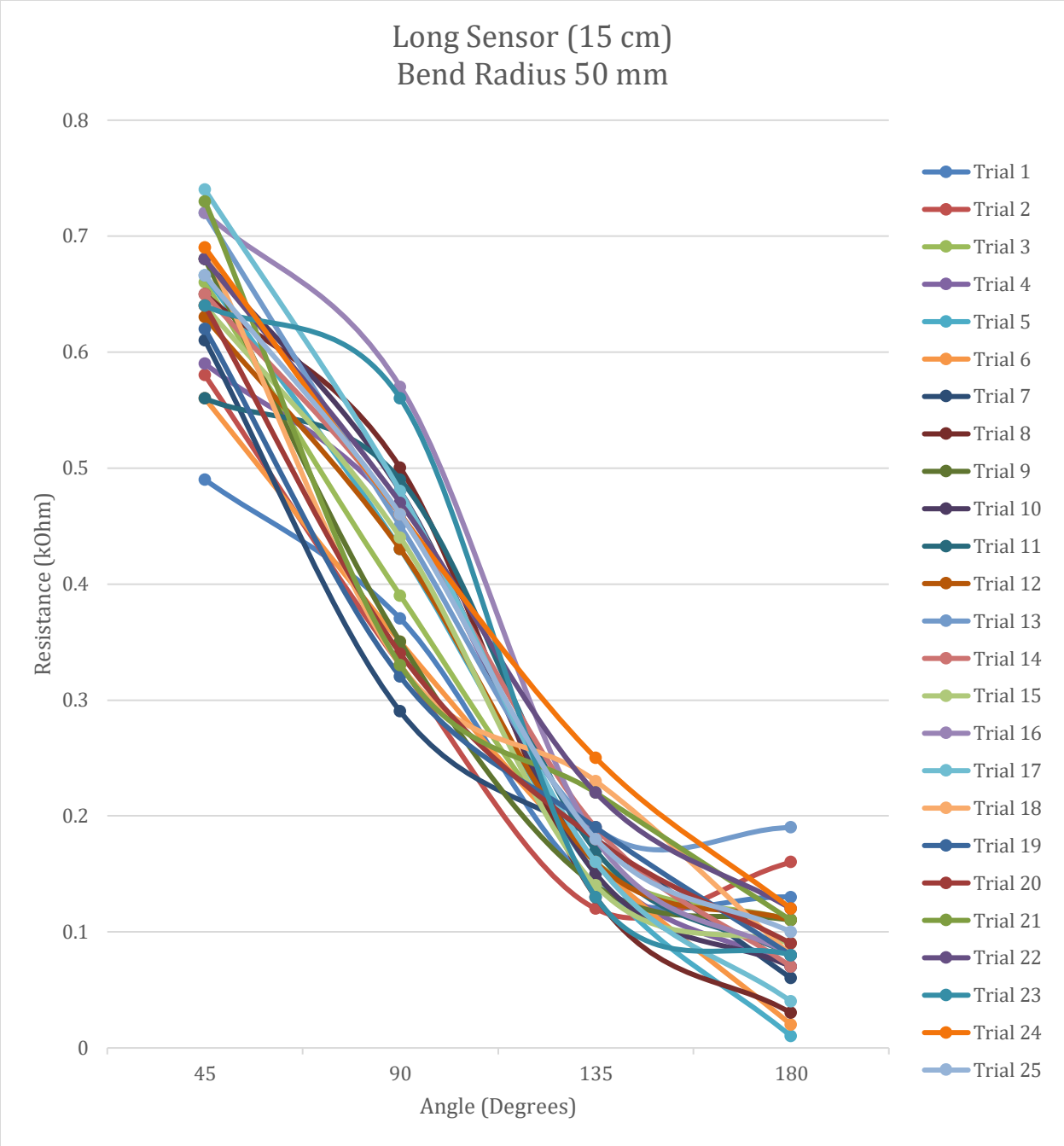

Figure 44: long sensor resistance vs. angle for a radius of 50 mm

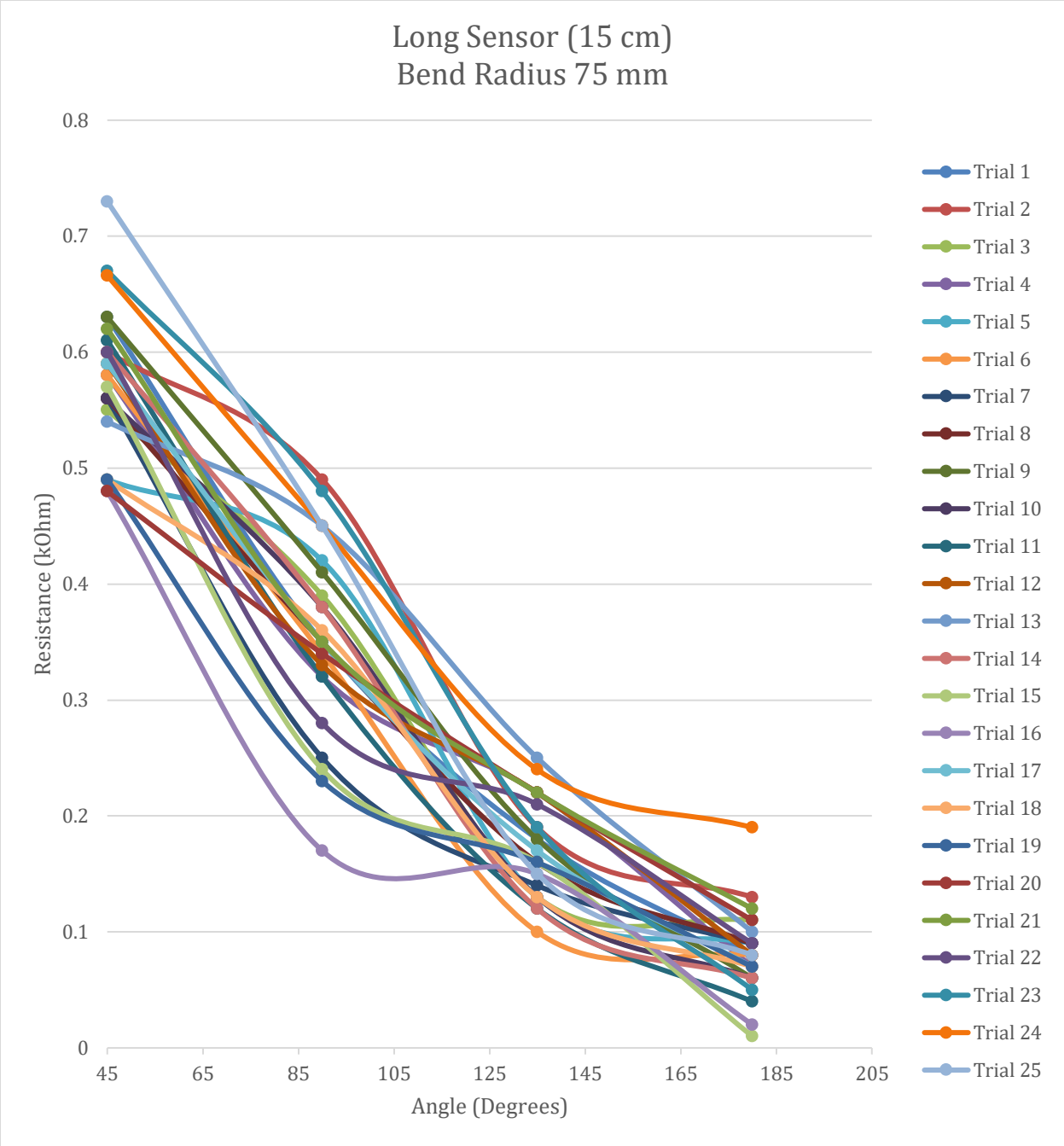

Figure 45: long sensor resistance vs. angle for a radius of 75 mm

Misaligned - 10 cm				
Radius (mm)	Angle (deg)	Mean (kOhm)	Deviation	Precision (%)
0	0	8.592	1.17541482	13.6803401
	45	6.684	0.95073656	14.2240658
	90	5.24	0.77995726	14.8846806
	135	5.076	0.6653821	13.1083943
	180	3.376	0.70727175	20.9499927
25	Angle (deg)	Mean (kOhm)	Deviation	Precision (%)
	45	7.108	0.66828138	9.40182017
	90	6.068	0.81584312	13.4450086
	135	4.484	0.45522888	10.1522944
	180	3.584	0.48792076	13.6138605
50	Angle (deg)	Mean (kOhm)	Deviation	Precision (%)
	45	7.992	1.21515431	15.2046335
	90	5.08	0.82865353	16.3120773
	135	3.292	0.24310492	7.38471797
	180	3.008	0.20800641	6.91510672

Table 7: metrics for the misaligned flex sensor (10 cm)

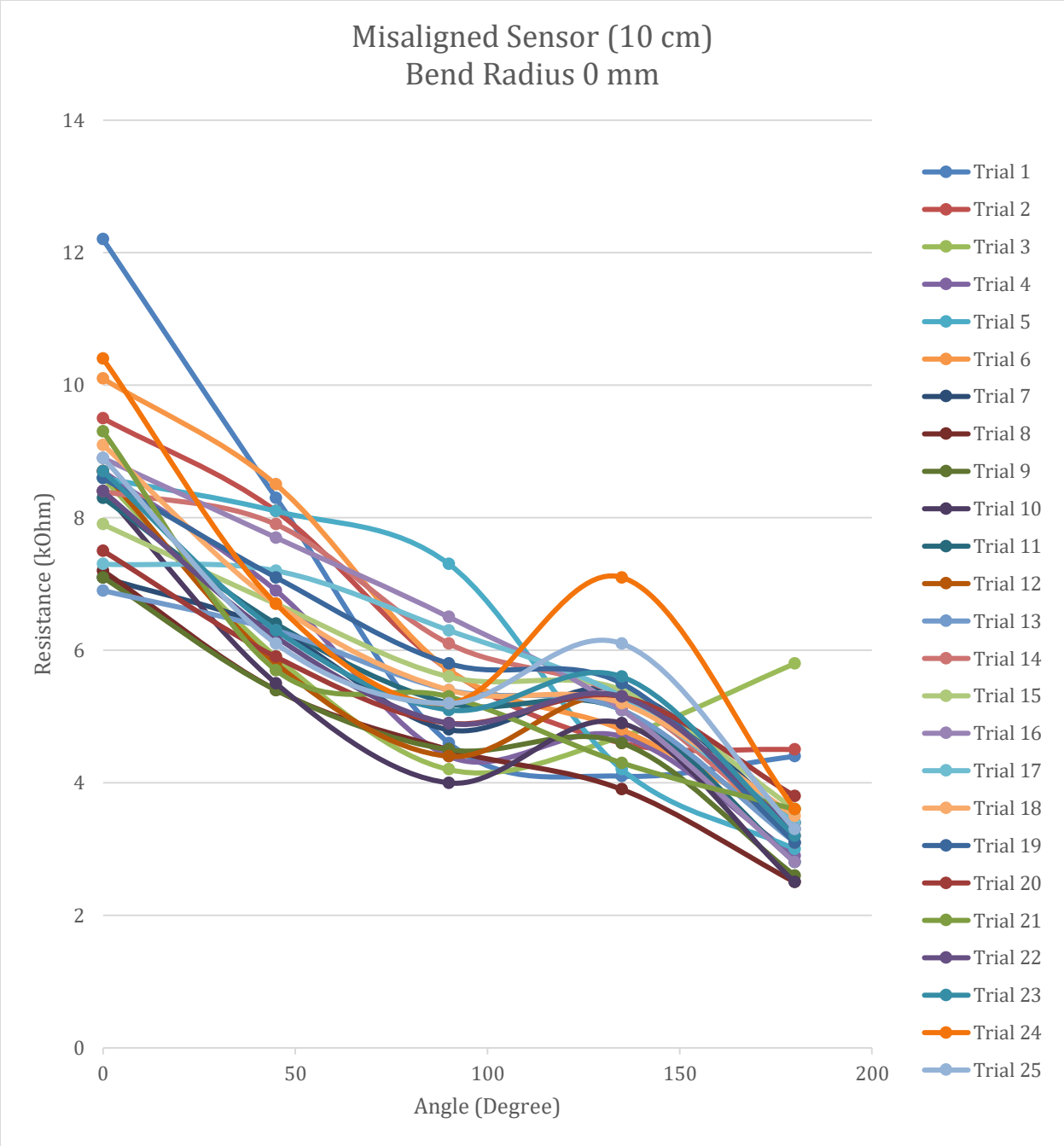

Figure 46: misaligned sensor resistance vs. angle for a radius of 0 mm

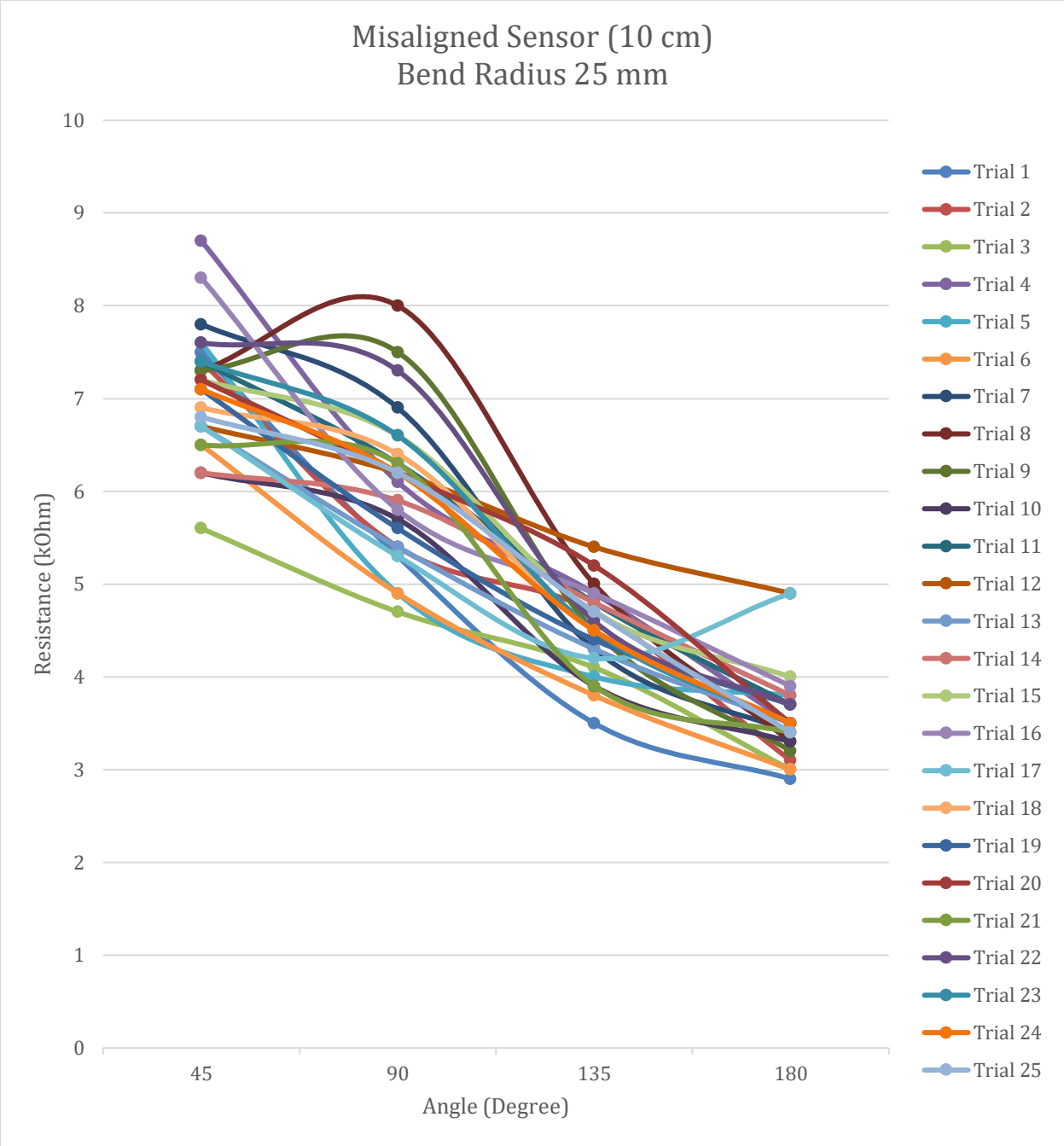

Figure 47: misaligned sensor resistance vs. angle for a radius of 25 mm

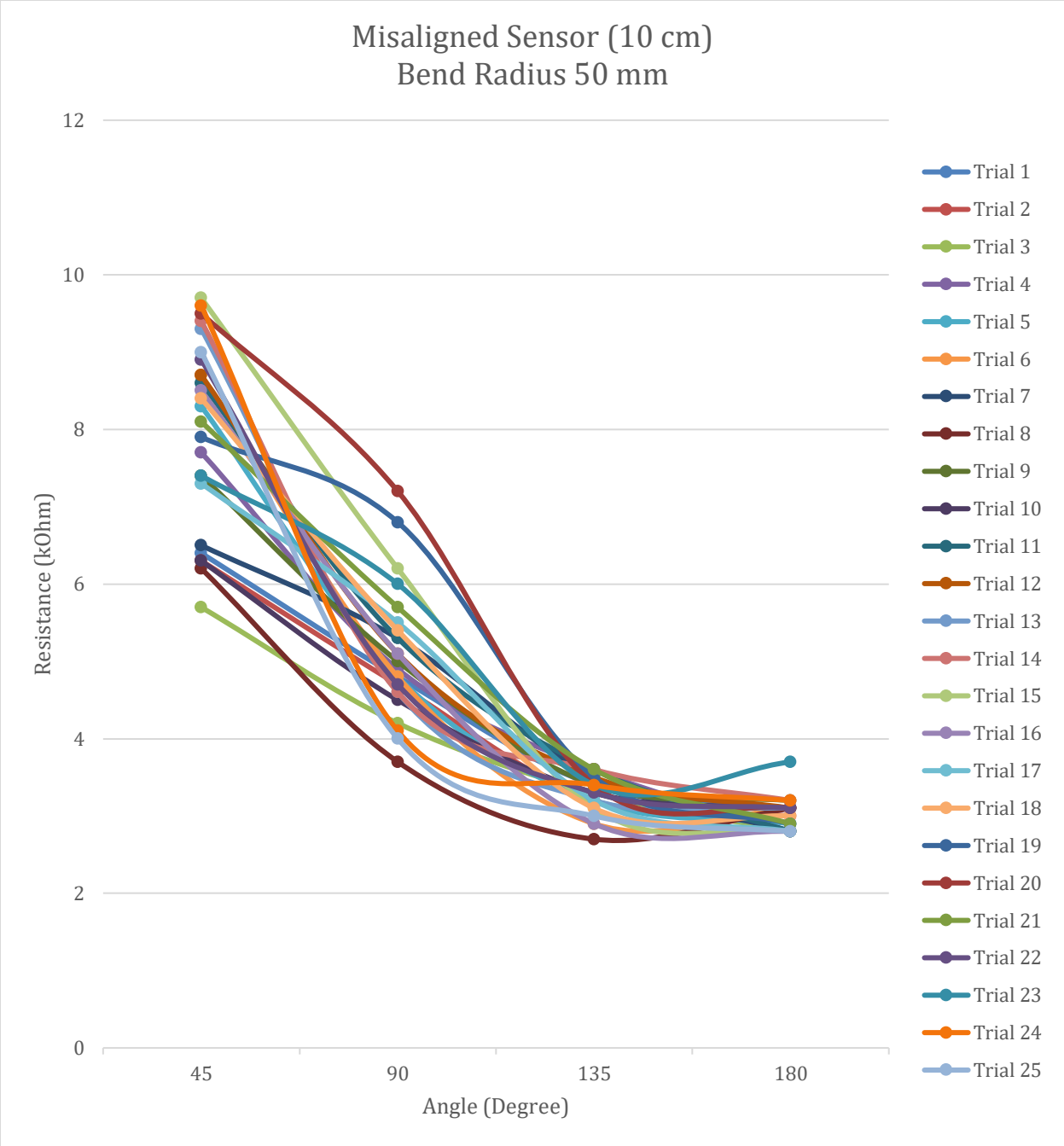

Figure 48: misaligned sensor resistance vs. angle for a radius of 50 mm

11.4 System Accuracy Testing

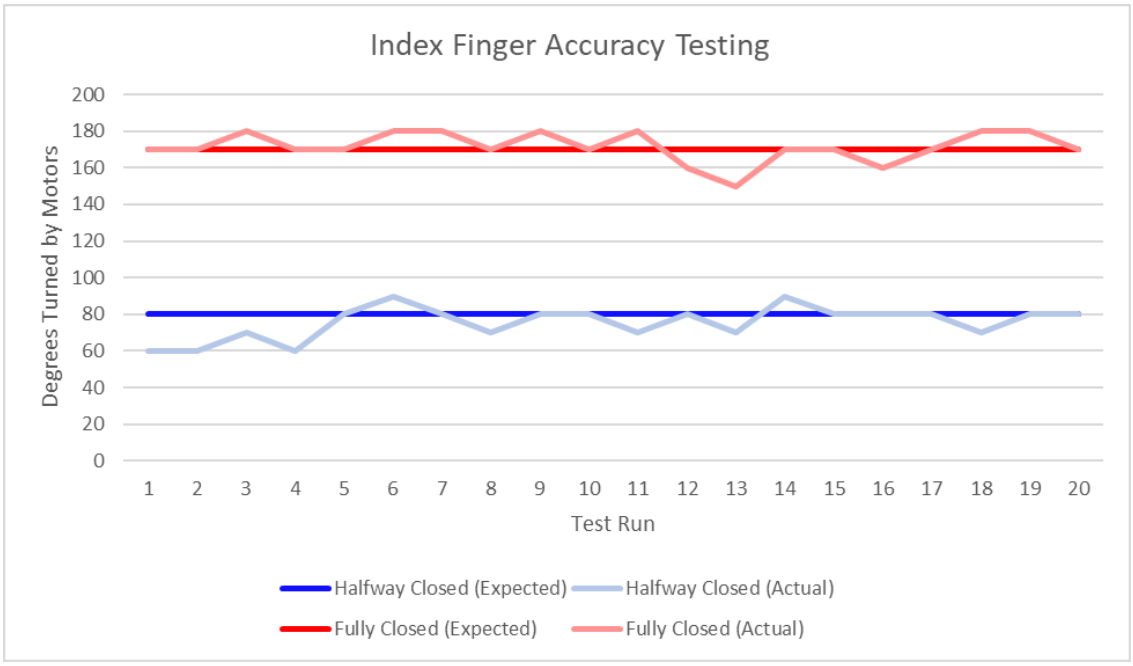

Figure 49: Accuracy Statistics of Index Finger

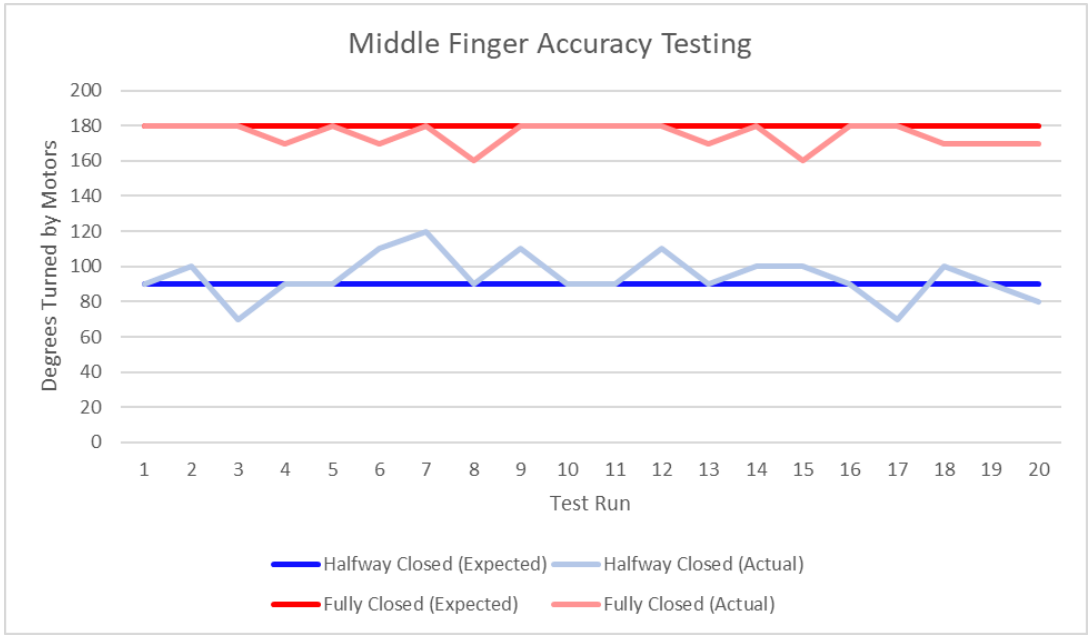

Figure 50: Accuracy Statistics of Middle Finger

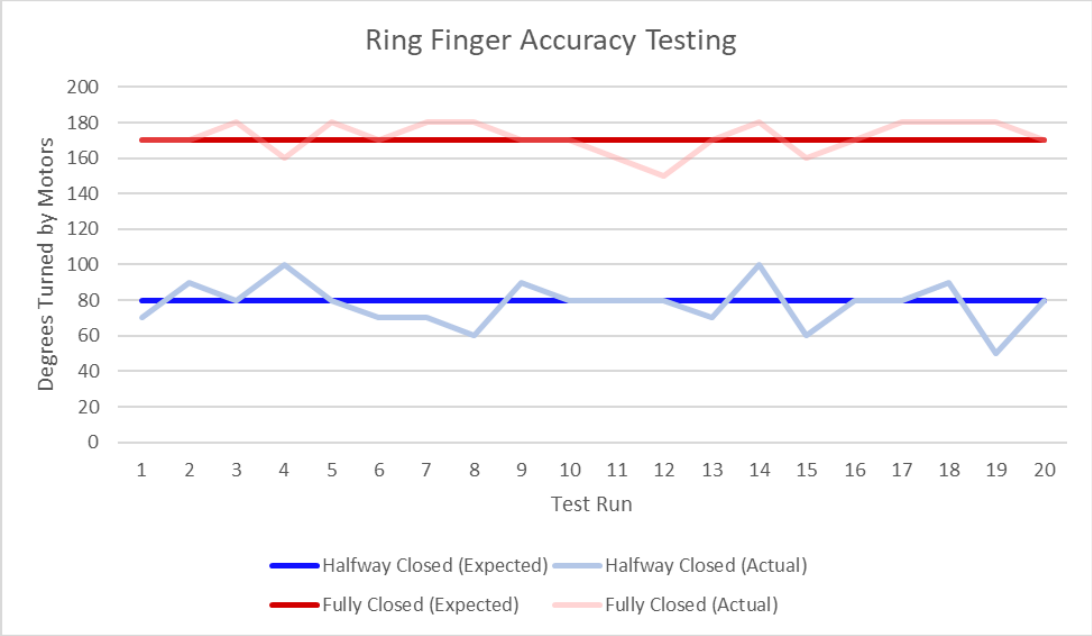

Figure 51: Accuracy Statistics of Ring Finger

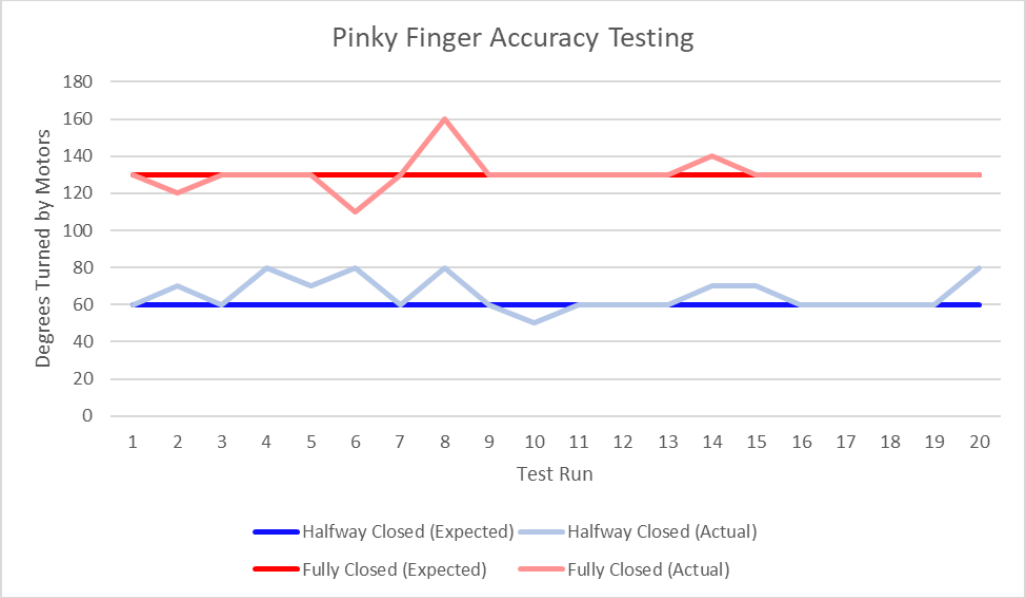

Figure 52: Accuracy Statistics of Pinky Finger

11.5 Constructed Flex Sensors

Figure 53: laminate flex sensors

Figure 54: duct tape flex sensors

Figure 55: electrical tape flex sensors