LOW-COST RIGID-FRAME EXOSKELETON GLOVE WITH FINGER-JOINT FLEXION TRACKING MAPPED ONTO A ROBOTIC HAND

A Thesis

By

ROBERTO GUERRERO, JR.

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Won-jong Kim
Committee Members,	Daniel A. McAdams
	Mehrdad Ehsani
Head of Department.	Andreas A. Polycarpou

May 2015

Major Subject: Mechanical Engineering

ABSTRACT

This thesis provides a representation of a low-cost rigid-frame exoskeleton glove that is used to track finger-joint flexion mapped onto a robotic hand to mimic user movements. The overall setup consists of an exoskeleton glove (exo-glove), sensors, a microcontroller, and a telerobotic hand. The design of the exo-glove is crafted to fit onto a left hand. SolidWorks was used for the prototype designs which were then sent to the Stratasys 400 rapid prototyping machine to be 3D printed in ABS-M30 plastic.

The exo-glove houses five rotary position sensors and three flexible sensors to track angle changes of the finger joints from two fingers and a thumb. Five low-pass filters are implemented as signal filtering for the rotary position sensors. An Arduino Mega microcontroller is connected to the sensors of the exo-glove and processes the input values. Using an open-loop controller to control the robotic hand, the values processed by the microcontroller from the exo-glove are sent to the servo motors on the robotic hand to operate the corresponding fingers of the user.

Throughout the initial calibration and testing phase, each sensor was tested individually to ensure the sensor functionally performs well. Signal analysis was conducted on the sensors at steady state and while in operation to show fluctuations in sensor readings and response to finger flexion. Experimental results show that averaging sensor data in the processing code yields smoother values and better precision. Due to the use of low-pass filtering with the rotary position sensors, the data sets collected were grouped together tightly compared to the flex sensors without filtering. However, the

actual angles measured were not accurately portrayed in sensor readings. The true flexion angles were compared in the data samplings to find a variety of ranges spanning around the angles desired to track. Many of the actual flexion angles were offset from the sensor readings by a variation of degrees, but the data shows the sensor readings were able to follow the general magnitude of the true flexion angles.

The precision seen in the data was also apparent in the robotic hand mirroring the posture. Changes in sensor readings caused jerking movements to occur in the robotic fingers but were able to maintain an overall flexion mirroring of the RF exo-glove. There is quarter-second delay between the exo-glove sensor reading and the robotic hand mirroring capability when not implementing averaging. When averaging the sensor values, there was a delay of more than half a second between the exo-glove posture and robotic hand mirroring.

DEDICATION

To my family and my future

ACKNOWLEDGMENTS

First of all, I would like to thank my advisor, Dr. Won-jong Kim, for helping me throughout my research. Dr. Kim was always patient and encouraging with great advice and insight towards my research goals. I would also like to thank Dr. McAdams for agreeing to be one of my committee members and always approachable. Additionally, I would like to thank Dr. Ehsani and Dr. Enjeti for becoming part of my thesis committee and for being patient and understanding with me.

Thanks also go to my friends and lab mates in Dr. Kim's lab. They were always encouraging and interested in my research project. Many times I was helped with multiple questions and they were always willing to help even when they may have been busy.

Finally, I would like to thank my family for supporting me during my time spent in school and while I was very busy conducting research. Such support was very critical in helping me complete this thesis. Their complete understanding for my choice of attending graduate school helped put my mind at ease for better focus.

TABLE OF CONTENTS

Page

CHAPTER Page

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

Page

Page

CHAPTER I

INTRODUCTION

Hand and finger-motion tracking has been researched for years with new techniques. The human hand is subjected daily to dexterous use, and as such we rely on them to be tough and functional. Unfortunately our hands are not invulnerable to damage, and in some cases they need to be rehabilitated to regain their original function. In such a case or to prevent possible hand injuries, finger-flexion tracking is required to instrument new devices for aid. Tracking the flexion of individual fingers is a tedious task that involves precise measurements. Usually an apparatus with sensing capabilities is applied to a person's hand, allowing motion of the fingers to be acquired. The data obtained from the motion-tracking devices opened a new way for controlling robotic grippers. Early research into telerobotic technology produced robotic hands that were tethered to the controller and had only a few degrees of freedom. Recently, necessity to have a precise and dexterous robotic hand has brought up new research into human-hand mapping with robotic hands.

1.1 Flexion Tracking

Movements and flexing of the fingers is known as flexion of the finger joints. A human finger has an approximate total flexion range of about 260°, although finger joints have varying ranges of motion based on each individual person [1].

The finger joints acquainted with flexion are the distal-interphalangeal (DIP) joint also referred to as the fingertip joint, proximal-interphalangeal (PIP) joint or center joint of the finger, and the metacarpal-phalangeal (MP) joint commonly known as the joint of the knuckles [1]. Fig 1.1 shows the human hand flexed to a clenched fist posture with the main joints articulated to their flexion positions.

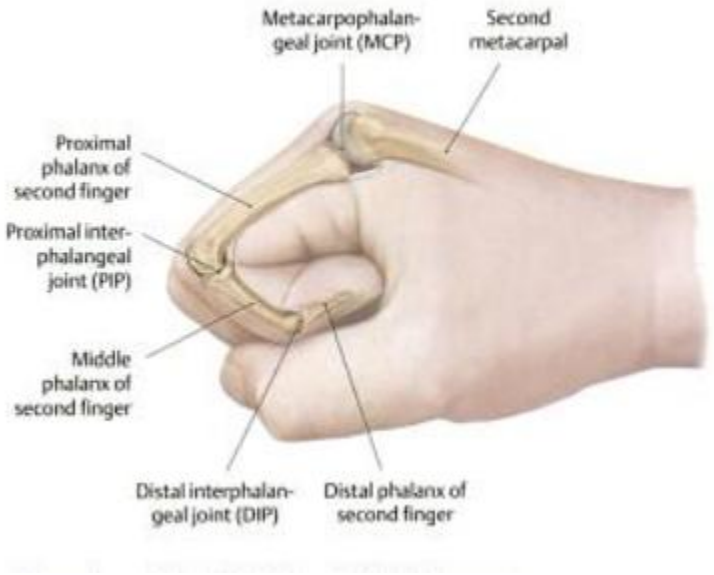


Figure 1.1 Finger flexion with joints identified [2]

1.2 Methods of Motion Tracking

Researchers have used methods for tracking motion ranging from simplistic designs to very complex systems. Exoskeleton type devices encompass a certain type

which uses external mechanical systems to function. By having the mechanisms outside of the device the user can wear the system without intrusive parts. Common mechanical designs in use are pivoting joints, slide mechanisms, and multiple linkages [3].

Gloves have been developed with flexible sensors integrated into the fingers that return changes of values based on the change in resistant across the glove [4]–[7] as seen in Fig 1.2. A more complex glove shown in Fig 1.3 uses a uniquely patterned glove that is tracked with a computer controlled camera. Using the nearest-neighbor technique, this glove can be tracked based on the value changes of the colors nearby [8]. Other methods used in finger tracking include: motion-detecting cameras and finger markers [9], light sensitive variances [10] and [11], magnetic field distortions [12], piezoelectric effect sensors, or optical position sensors [13].

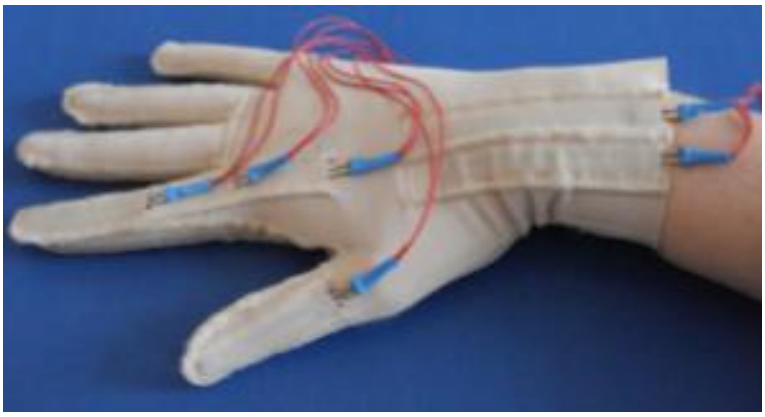


Figure 1.2 Soft glove with flexible sensors [7]

Figure 1.3 Unique patterned colored glove [8]

1.3 Biomimetic Hand

Biomimetics in general is known as using biological inspiration for purposes that differ from their natural use. The range of bio-inspiration varies from mimicking small attributes of biology to entire natural processes. Through many years of existence, biological organisms have adapted for survival out of necessity, and as such they prove to be an effective solution for certain applications [14]. Mimicking human anatomy is seen in many areas from folding limbs on a scissor lift to humanoid robotics. Naturally, human hands have highly desirable traits that we would like to mirror in other applications. This particular research involves the tracking of human finger joints because of the dexterous ability fingers support. In terms of biomimicry, the research involved with using human-finger tracking to develop new ideas follows a form and functional imitation inspired by observation of the human hand [15].

The human hand is often overlooked when a person thinks of the word tool. It is an appendage that we use to great extent and allows us to live our lives the way we do.

There are 23 degrees of freedom (DOFs) in the human hand allowing for excellent dexterity for complex movements. The human hand is composed of a thumb, index, middle, ring, and little finger. Based on the common physical shape of a human's hand, the rigid-frame exoskeleton glove (RF exo-glove) was designed to fit snugly onto a left hand. Human hands are very dexterous and well equipped to perform tedious tasks, the RF exo-glove will be unable to fully match the DOFs, rather it provides the basic movement a person would normally execute upon flexion.

1.4 Applications of Finger Tracking

Many applications in various fields make use of telerobotic controlled hand tracking technology. Possible fields in which this thesis can be traced to include: general robotics, military, medical, and hazardous environments.

1.4.1 Rehabilitation

Diminished hand and finger control gives the need for devices that can aid in rehabilitation for the patient. However, simply understanding and tracking progress in finger-joint angles is the start for future development in this area [16]. Finger rehabilitation occurs over several months with physical therapists assisting the patient in joint movement. During these visits from the physical therapist, each session is charged usually to the patient or insurance company adding up to a very costly rehabilitation [3].

Various devices used in the medical field are designed to aid victims of traumatic events, such as strokes, become rehabilitated [7]. Fig 1.4 shows a late model hand device developed by Kaiser Medical Inc. for patients requiring therapy for their hands [17]. This glove promotes the healing process for post-surgery finger joints by repeatedly moving the patient's fingers through a certain range. Correspondingly, data from the finger flexion tracking of the RF exo-glove can be used by doctors to provide useful information on the progress the patient is making [18]. The resulting changes in data from the RF exo-glove can show if progress in flexion is achieved.

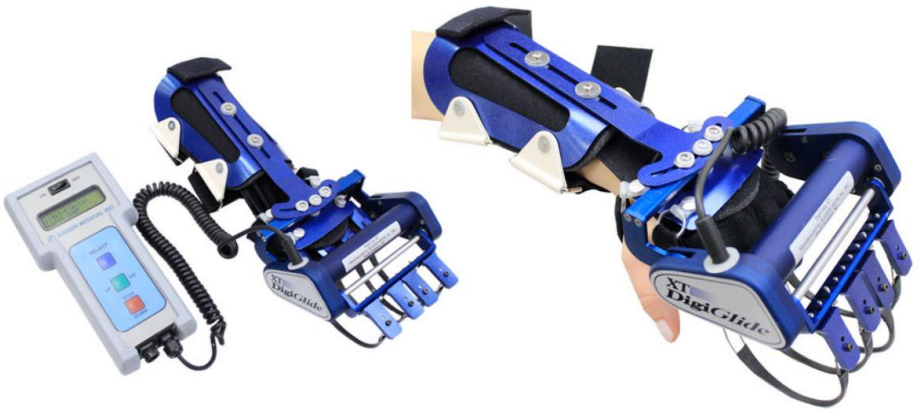


Figure 1.4 XT DigiGlide glove for rehabilitation [17]

Passive finger joint tracking, such as the RF exo-glove, holds merit for the users that have movement and pain sensitivities in their fingers or hands. Due to the lack of force-actuating systems in passive devices, they weigh less and are easier to use compared to active-controlled devices like the XT DigiGlide glove [19].

1.4.2 Robotics

Robotic grippers and semi-dexterous hands have been in service in the industry for decades. Fig 1.5 shows a three-fingered robotic-gripper attached to a mechanical arm used for grasping objects that may weigh more than an average person can carry. Many of these grippers and hands are used in hazardous environments such as highly toxic locations and can be controlled using a finger tracking method. With the recent influx of robotic hand applications spanning from military functions to the medical field, the need for precise robotic-hand control is inevitable. In some cases, the operator for these robotic devices cannot be in the same location due to constraints or harmful scenarios that could arise. These types of situations make use of telerobotics.

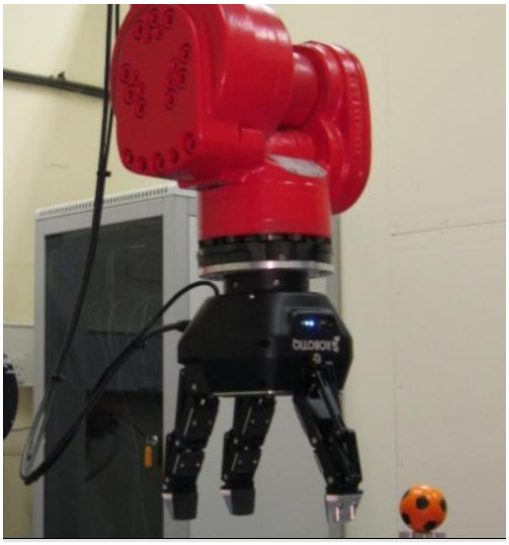


Figure 1.5 Robotic gripper attached to mechanical arm [20]

Telerobotics is a way for people to operate mechanical systems or robots from a remote location. Typically, telerobotics is used for operation in hazardous environments and places that would potentially be dangerous to life. A few applications for industrial use telerobotics include space exploration, extreme pressure environments, bomb disposal and handling, and chemical exposed areas. In order to use a telerobotic system, there must be a human-interface mechanism that communicates with the remote system to be operated. The control method for the human-operation mechanism varies based on the application and function of the telerobotic device to be operated. Control methods can involve traditional methods such as a typical control module with buttons and joysticks; however, using a human to interface with the telerobotic system sometimes requires more sophisticated controllers. Such controllers may use haptic feedback, optical tracking, voice commands, interactive sensors, or digitally rendered environments [21].

One field of telerobotics is in medical applications. Telemedicine and telesurgery allows patients to be monitored and cared for in their own homes or away from medical facilities [22]. People with little hand mobility or low strength could use a telerobotic hand to help grasp objects which may be out of their abilities [23]. Currently in the medical field there is a surgical robot named the da Vinci Surgical System seen in Fig 1.6, which is controlled remotely by the surgeon replicating their exact movements.

Figure 1.6 da Vinci surgical robot and surgeon [24]

1.4.3 Heavy Equipment

An alternative use for mapping hand movement in the industrial world would be in the use of tractor operation. There are several tractors that use a claw or multiple-limbed mechanisms to pick up objects such as the claw mechanism seen in Fig 1.7. Typically, a tractor operator must receive training and become proficient with the heavy machinery before they can perform such tasks. These industrial machines could greatly benefit their operators by reducing the learning curve and time to train for operation. Simple motions done every day such as attempting to grasp an object would directly translate to the use of the control system for operating these tractors.

Figure 1.7 Crane machine with claw mechanism [25]

1.5 Contributions of Thesis

This thesis was created to contribute experimental results for a new design of a finger joint flexion RF exo-glove. Most finger-flexion-tracking devices use a flexible glove substrate to house the tracking sensors. My research has shown a niche in experimentation done in the field of finger-joint flexion-tracking methods regarding the shape distortion caused by human fingers. Each design that used a flexible sensor on an amorphous glove did not take into account the slight transverse bending of the sensor across the finger. The distorted part of the sensor that is unaccounted for is very minuscule. Over many attempts of repeating the experiment, however fatigue may have affected the sensor readings. The RF exo-glove uses a rigid frame to maintain a controlled and repeatable position for the tracking sensors to adhere to.

The RF exo-glove shares some commonality with other finger-tracking gloves. Although there may be a variety of reasons for their creation, the similarity shared is the desire to track hand or finger motion. The DHM glove [6] and NeuroAssess glove [7] utilize potentiometer bend sensors similar to the ones in the RF exo-glove. Additionally, just as the RF exo-glove was designed to track finger flexion, so were the DHM glove, NeuroAssess glove, and SmartGlove [13]. Tracking the flexion of the thumb was not seen in all of these devices but was involved with the DHM glove, SmartGlove, NeuroAssess glove, and Color glove [8]. Sensor placement for the RF exo-glove occurs above the finger joints for the flex sensors. This was also the sensor positioning for the DHM glove, NeuroAssess glove, SmartGlove, and augmented-environment project device.

My decision to design the RF exo-glove was to create a unique device that had not been tested. In comparison to the previously mentioned devices, the RF exo-glove uses rotary position sensors and flex sensors rather than only a single type of sensors. Finger tracking in an augmented-environment was done using retroreflective markers [11]. The colored-glove experiment uses a unique patterned color glove for tracking. SmartGlove uses sliding optical encoders. Magnetic hand tracking uses magneto-resistive tracking [12].

Each experimental finger-tracking device was created with a certain number of DOFs and sensors. There are 8 DOFs on the RF exo-glove using 8 sensors. The DHM glove has 10 DOFs using 10 sensors. The augmented-environment project used 4 sensors

and had 6 DOFs. The color glove did not use sensors but was able to replicate 26 DOFs for a 3D model.

The RF exo-glove was designed for finger-flexion tracking. However, several devices designed for hand-tracking were intended for other uses. The augmented-environment glove and color-glove were designed for virtual reality, and the magnetic hand tracking prototype was designed for machine interaction similar to the function of a mouse.

The most discernible difference between the RF exo-glove presented here-in and the other tracking devices is the rigid base and structural components. As stated, many of the previously published researched gloves that did not use a camera or external tracking system were bound with sensors above the finger joints. The RF exo-glove uses sensors adjacent to the finger joint with the exception of the MP joint (knuckles) that has sensors positioned above the finger. This unique design will allow for new research areas in finger-flexion tracking.

1.6 Overview of Thesis

This thesis begins with the first chapter, giving an introduction to finger-flexion tracking and applications. This introduction will cover methods that have been researched and provide examples of current implementations of the topic.

In the second chapter, the experimental operation is discussed. This chapter reviews the conceptual design of the RF exo-glove, components, hardware, and wiring diagram. Features and constraints of the RF exo-glove and robotic hand are conferred with visual diagrams. The entire setup can be seen in Fig. 1.8.

The third chapter covers the calibration methods used to gather initial data from the sensors. This chapter consists of three sections−one section covers the calibration of the rotary position sensors, another for the calibration of the flex sensors, and the last section comparing data of the two types of sensors.

The fourth chapter details the experimental results and analysis. There are three individual sections in this chapter covering the data acquired from four positions with the RF exo-glove. The overall results are displayed for the RF exo-glove with the inclusion of the transmission of flexion angles onto the robotic hand.

The final chapter entails the conclusions of the thesis and provides an insight of the experiment with remarks for improvements.

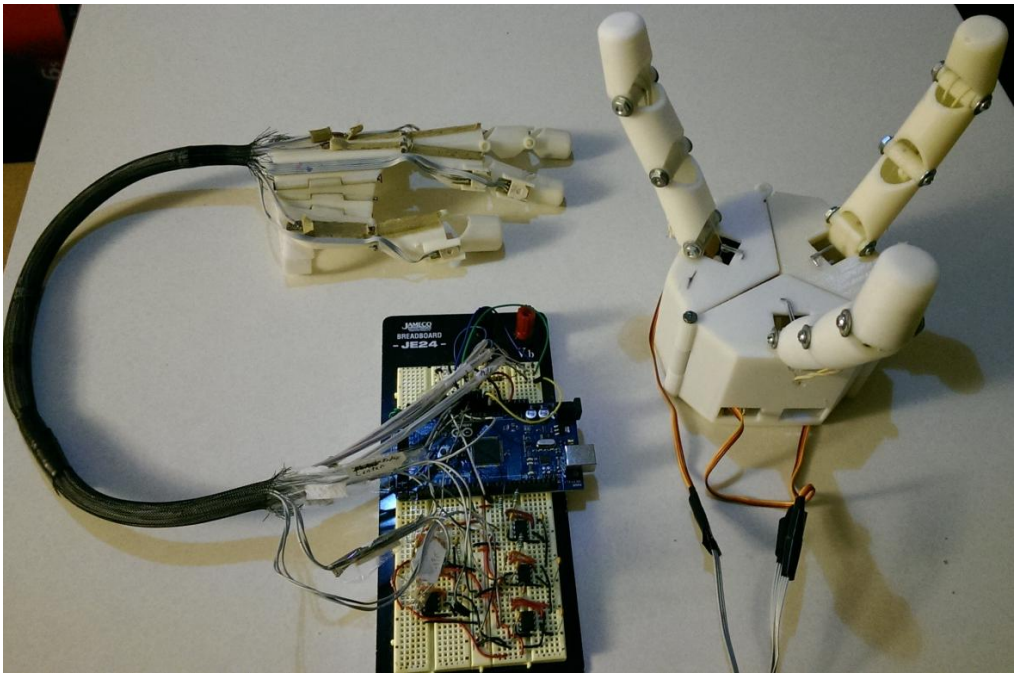


Figure 1.8 Complete setup of RF exo-glove, robotic hand, and electronic system

CHAPTER II

EXPERIMENTAL CONCEPT

2.1 Conceptual Design

The RF exo-glove seen in Fig. 2.1 consists of two fingers and a thumb. Each finger is comprised of three parts with the fingertip and center joint modified to accommodate the rotary position sensors. The thumb consists of two parts with the thumb tip similarly modified to accommodate a rotary position sensor. Each finger and the thumb have the knuckle-conjoining section modified on top to allow a flex sensor to lie completely flat across. There are three sectional joints for allowing the thumb to move more freely just as natural movement would occur. Finally, the RF exo-glove has a base for the hand that ties in all of the finger and thumb joints. There are a total of 12 parts that make up the RF exo-glove mechanical assembly which has an overall length of 19.8 cm. Part dimensions in appendix C.

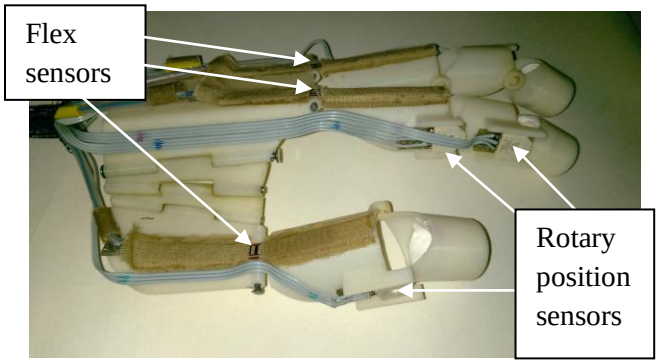


Figure 2.1 RF exo-glove

The design of the RF exo-glove was steered to the final concept by previous flexion-tracking research methods. With the same intent as Yamaura *et al.* [3], this research project was developed to provide experimental data for a device that could possibly be used to supplement the need for a physical therapist to assist a patient in rehabilitation. The prototype developed in this research was not designed to be the physical therapist replacement mentioned but rather demonstrate the base for such a device and show how finger flexion data can be used. Looking at the devices created by Lee and Cho [26] and others in the area, these prototypes were designed to actuate a force onto the user's fingers to assist in physical therapy. The RF exo-glove in this experiment was designed to measure the joint angle rather than manipulate the user's fingers. Improvement in joint-angle data would allow for more precise and accurate devices to aid in physical therapy for finger rehabilitation [26].

As stated in [27], optical hand tracking is far more complex and variably involved. One major drawback or difficult-to-control variable is the variance in light and contrast used when tracking such motions. Textures of the objects being tracked often change when flexed, causing the light which is reflected off to vary. Just as Ghosh proposed to introduce an alternative to optical hand tracking this experiment provides a mechanical design used in tracking finger flexion.

The RF exo-glove provides a rigid platform to replicate the experimental result with precision and accuracy. Similarly to how the exoskeleton glove by Noaman *et al.* [6] was fabricated to be a medium between the user and the sensors, my RF exo-glove was designed to hold each sensor in place for flexion readings. Guo and Nguyen [9]

proposed to use multiple methods in combination for hand tracking leading to the RF exo-glove combining the use of rotary position sensors and flexible sensors attached to a RF exo-glove for improved precision. Differing from Guo *et al*. [9], the tracking of an entire hand is not part of this experiment and therefore is beyond the scope of this thesis.

Research into previous finger tracking methods has inspired my design to use rotary sensors along the exoskeleton glove to track joint flexion of the DIP joint (fingertip joint) and PIP joint (center joint of the finger). The rotary position sensors used were applied onto a small rectangular piece of printed circuit board (PCB) and soldered to ribbon wire seen in Fig. 2.2. The resulting product allowed the rotary position sensors to be mounted onto the RF exo-glove while maintaining a solid signal connection.

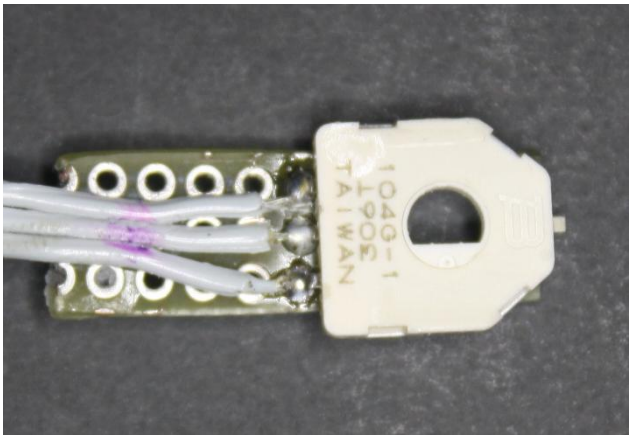


Figure 2.2 Bourns rotary position sensor mounted onto a custom PCB

To track finger joint flexion through the knuckles, the RF exo-glove consists of flexible sensors attached to hand body of the design. Flexible sensors (or flex sensors for short) are used across the MP joint on top of the exoskeleton glove. Contrary to the term "flex sensor" used in this thesis and in [6], the flex sensors used in this thesis vary resistivity when the sensor is bent causing a drop in resistivity due to the ink on the sensor. The flex sensors used in [6] operate by using a small movable sensor over the flexible base to track changes in resistivity. To implement the RF exo-glove and the flex sensors, a soft fiber cloth was used to keep the sensors aligned throughout the flexion process while preventing the resistive ink on the surface of the flex sensor from damage. Each flex sensor is held in place on the finger by the connecting end tabs with the opposite ends free to translate across the back of the RF exo-glove fingers shown in Fig. 2.3.

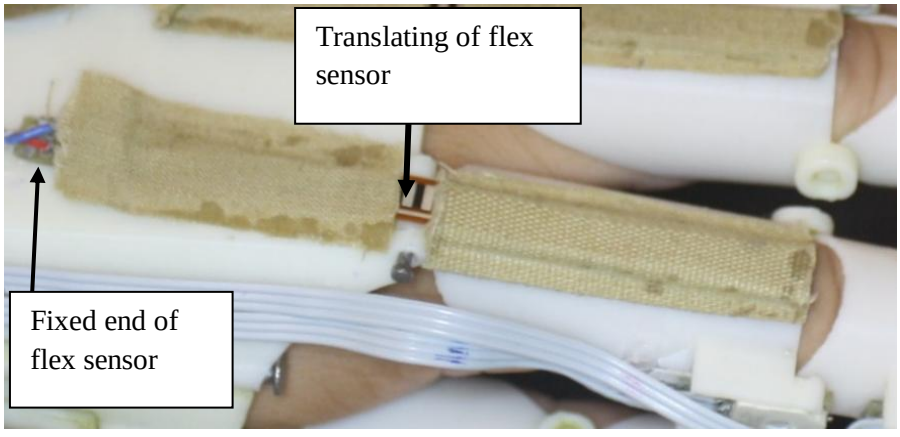


Figure 2.3 Flex sensor mounted on RF exo-glove

As shown in Fig. 2.4, only two fingers and the thumb are used in my RF exo-glove to reduce possible errors from hardware limitations and constraints. The limitations of the RF exo-glove design lay with the bulkiness of the components. Using two more fingers with this design would cause collision interference between the adjacent fingers particularly at the pivot points. Any collisions of the RF exo-glove finger joints could adversely affect the sensor data leading to highly mixed results. Ghosh's thesis [27] discusses the notion of tracking all joints in the hand that requires a high level of precision to track the motion of the bulk part of the hand. The basic design shown here in Fig. 2.4 represents the limited finger joints that are tracked in the exo-glove. The numbered links are the joints of the hand which the joint tracking follows while the remaining links show the other fingers of the human hand but are not used in flexion tracking for this experiment. The limited flexion tracking reduces the dexterous ability of the mimicking robotic hand, however, it does allow for the minimum necessary amount of fingers for complex grasping. As stated in Ghosh's thesis [27], humans can manipulate objects while using great dexterity requiring very little thought.

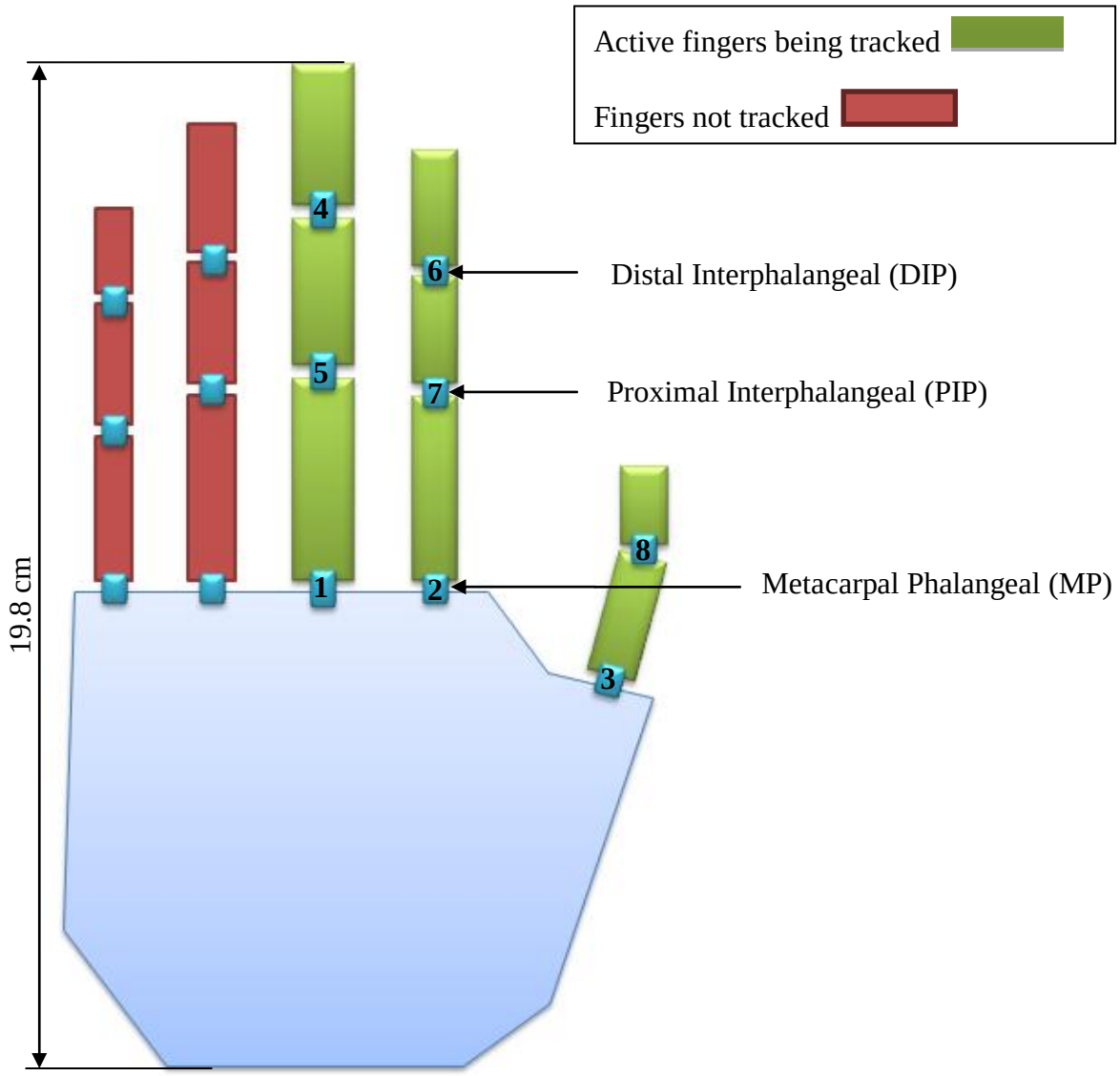


Figure 2.4 Active finger joints tracked

The range of sensors in Fig 2.4 for 1–3 represent the flex sensors while 4-8 represent the rotary position sensors. The sensor numbers are assigned accordingly:

1. Middle finger MP
2. Index finger MP
3. Thumb MP
4. Middle finger DIP
5. Middle finger PIP
6. Index finger DIP
7. Index finger PIP
8. Thumb DIP

2.2 RF Exo-Glove Features

The design was implemented to encase my own hand in an exoskeleton glove that could house rotary sensors and flex sensors for flexion tracking. The RF exo-glove is made using a base housing that connects all pivoting parts. There are three joints used to support sensors for tracking MP joints and two separate joints for sensors on the PIP joints. The thumb does not have a PIP joint, there are only two for the remaining fingers used. There are also three joints to support sensors tracking the DIP joints. Each finger joint was designed with two protruding edges used to hold the rotary position sensors in place parallel to my fingers. The knuckle-conjoining section of the RF exo-glove seen in Fig. 2.5 was designed to have a flat parallel surface to the base of the RF exo-glove in order to retain the flex sensors in their equilibrium state. The angle Ω is the difference between the actual finger and the top of the RF exo-glove which has the flex sensor. This angle is consistent throughout the RF exo-glove allowing the sensor values to be obtained directly without any offset required. Since the thumb, middle, and index fingers are being tracked, the RF exo-glove has a total of eight DOFs.

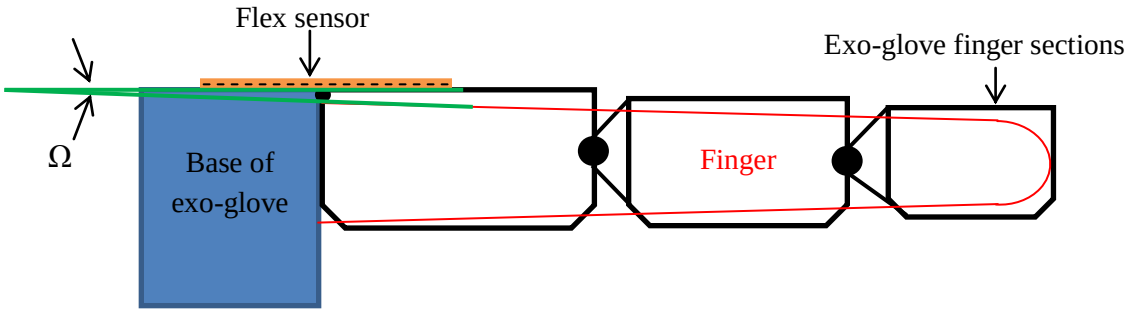


Figure 2.5 RF exo-glove finger relative to flex sensor angle

2.2.1 Fabrication

Measurements of my left hand were taken, and a SolidWorks CAD model was created using the measurements to form an exoskeleton glove with DOFs. The SolidWorks model was then fabricated using a Stratasys 400 Rapid Prototyper with ABS-M30 plastic for a sturdy prototype. The Stratasys 400 Rapid Prototyper has a resolution of ±0.127 mm (±0.005 in) resulting in fine and smooth fitting parts.

2.3 Hardware and Components

The RF exo-glove experiment is made up of various components and hardware used to gather data for finger flexion. Basic components used were resistors, capacitors, an assortment of wires, PCB, and protective tubing. The following section will discuss the major components used in the research.

2.3.1 Rotary Position Sensors

The sensors integrated into the RF exo-glove include five 3382G Bourns rotary position sensors that operate in a similar manner to a potentiometer. There are three connections on the sensor used for operation. Fig. 2.6 displays the connection tab functions for the rotary position sensor. The two outer edge connections are used for ground and positive voltage while the middle connection is the wiper that gives the value of the sensor reading.

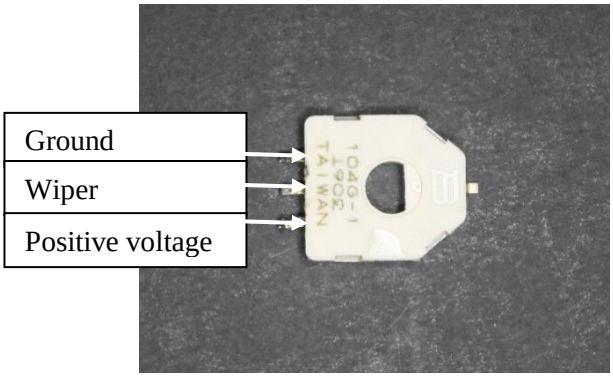


Figure 2.6 3382G Bourns rotary position sensor

2.3.2 Flex Sensors

Fig. 2.7 shows an example of the three Spectra Symbol flex sensors used on the RF exo-glove for knuckle flexion. Each flex sensor used had an effective sensor length of 2.2 inches with a normal flat resistance of 25 kΩ. There are only two connection tabs on the flex sensors for a power supply input and ground.

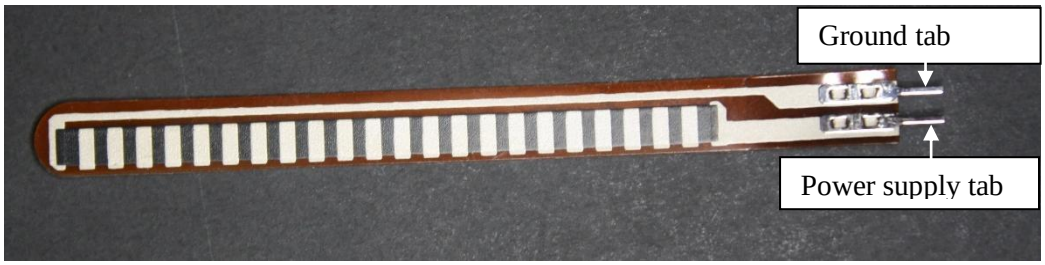


Figure 2.7 Spectra Symbol flex sensor

2.3.3 Arduino Mega Microcontroller

The microcontroller used in the system was an Arduino Mega 2560 displayed in Fig. 2.8. This microcontroller has a serial port component built onto the board allowing direct communication to a computer using a universal serial bus (USB) cable. Since the sensors used in this experiment are analog, conversion from analog to digital signals need to be processed. The analog to digital converter (ADC) built on the Arduino Mega has a 10 bit resolution giving values from 0 to 1023.

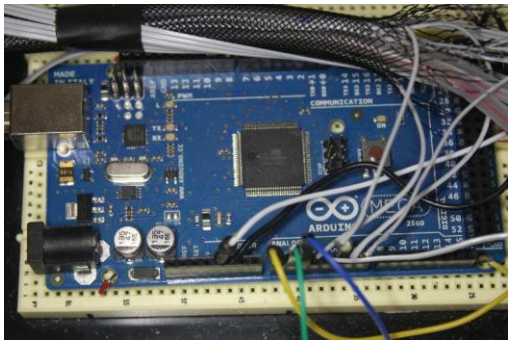


Figure 2.8 Arduino Mega 2560

2.3.4 Low-Pass Filter

Five anti-aliasing low pass filters (LPFs) were designed and implemented with the rotary position sensors to reduce the rapid fluctuation of the sensor reading. The LPFs were designed to only allow signals below a cutoff frequency to pass through while rejecting all others above it. The circuit designed is an active LPF utilizing a UA741CP operational amplifier (Op Amp), 1-kΩ resistors, and 100-pF capacitors. The effective filter operates at about half the noise frequency seen from the sensors allowing the signal through the sensor to pass without ever reaching the noise frequency.

The design for the LPF was dictated by the distortion found in the sensors. Each rotary position sensor was analyzed with an oscilloscope to find the noise frequency exhibited by the sensor. All sensors tested experienced the same noise frequency resulting in five identical low pass filters being created.

The noise frequency exhibited by each sensor was approximately 2.05 MHz. To have an effective filter I decided to limit the passing frequencies to 75% of the noise frequency giving the cutoff frequency of $fc = 1.54$ MHz. I chose capacitors, C = 100-pF to use in my circuit and using the cutoff frequency I used the following equation (2.1) to find the resistor values.

$$R_1 = R_2 = \frac{1}{\sqrt{2}*(2\pi fc)C} \tag{2.1}$$

This equation gave a resistor value of 1030 Ω and thus 1-kΩ resistors were chosen to be used. Once the filter was complete, a simple Arduino sketch was used to test the

effectiveness of the LPF circuit on the sensor data values. As expected, the hardware noise filtering was able to cut the fluctuation of data values considerably giving smooth data points.

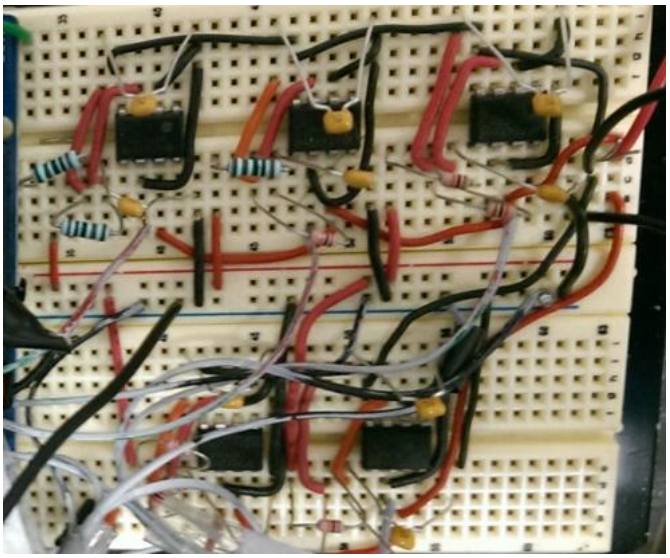


Figure 2.9 Low pass filters

2.3.5 Component Cost

The major components used for the RF exo-glove can be seen in Table 2.1. The 3D printed parts were made using the university's rapid prototyper at no cost.

Table 2.1 Component Cost

Description/Quantity	Price per each item ($)	Price for quantity ($)
10 x 100-pF Capacitors	0.03	0.30
10 x 1-kΩ Resistors	0.13	1.30
3 x 10-kΩ Resistors	0.10	0.30
5 x UA741CP Op Amp	0.55	2.75
3 x 2.2” Flex Sensors	7.95	23.85
5 x Rotary Position Sensors	2.60	13.00
Arduino Mega Microcontroller	No cost/39.54	-
3D Printed Parts	No cost	-
	Total $	41.50

With a total of $41.50 for all major components needed, this RF exo-glove has a relatively low cost to assemble. Comparing other flexion tracking gloves to this total cost, the price for components used here under-cuts various designs by quite a margin. For example, [7] uses a glove with six flex sensors which may be similar to the ones used in this experiment which alone would cost around $47.70.

2.4 Component Wiring

The block diagram in Fig. 2.10 shows the basic connection setup of the Arduino Mega microcontroller with the components that make up each finger of the RF exo-glove and robotic hand. The flex sensors and rotary position sensors are grouped together within a box representing the RF exo-glove. There is also a box labeled "Robotic Hand" that groups the servo motors that are used for the finger motion of the hand. Each sensor is connected to the analog input pins of the microcontroller while the servos are connected to the pulse-width-modulation (PWM) pins used for digital output.

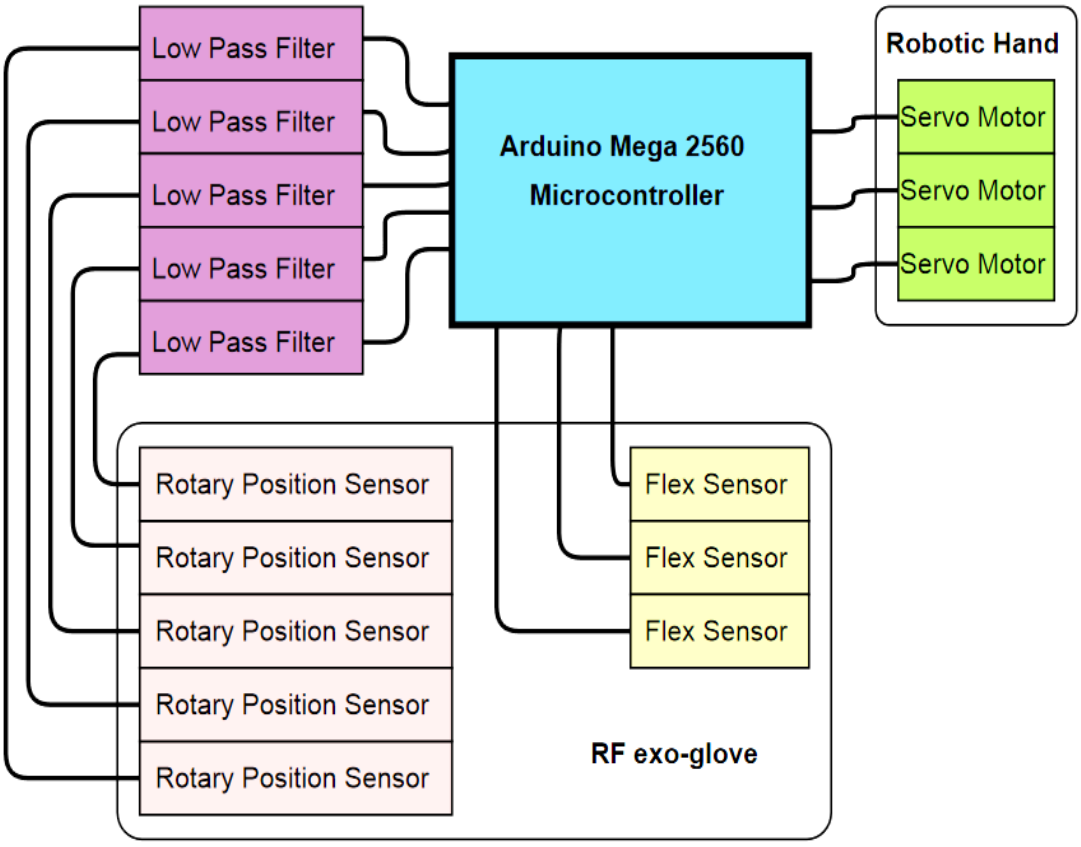


Figure 2.10 Block diagram of electronic component systems

2.4.1 Rotary Position Sensor Circuit

The circuit for the rotary position sensors is very simple and seen in Fig. 2.11. This sensor acts as a potentiometer varying resistance when turned. There are three connection tabs used for operation with one tab connecting to ground and the opposite end tab connected to the LPF circuit. The center tab or wiper is connected to the Arduino Mega using an onboard analog pin.

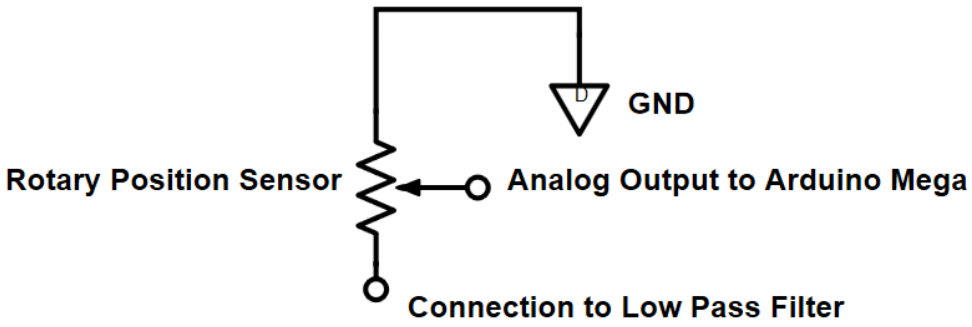


Figure 2.11 Rotary position sensor circuit diagram

2.4.2 Flex-Sensor Circuit

The flex sensor circuit shown in Fig. 2.12 is simple but involves an additional pull up resistor. The flex sensor itself varies in resistance similar to a potentiometer. A 10 kΩ resistor is connected to one tab on the flex sensor and a 6-V power supply shown as Vcc. Between the junction of the flex sensor and resistor, the signal connection is made with the Arduino Mega using one of the onboard analog pins. The last remaining tab of the flex sensor is connected to ground.

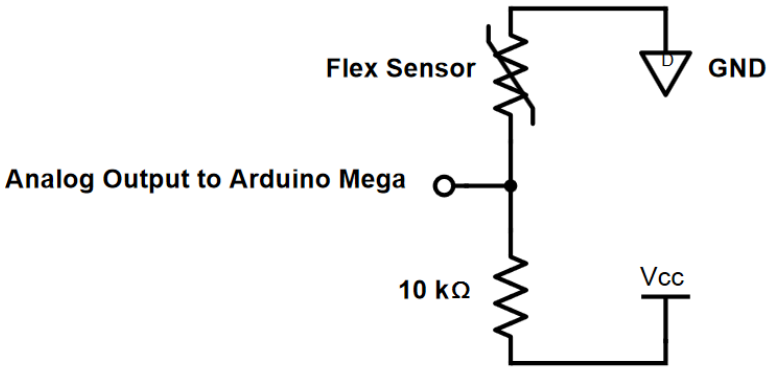


Figure 2.12 Flex-sensor circuit diagram

2.4.3 LPF Circuit

In the LPF circuit shown in Fig. 2.13 Cl and C2 consist of 100-pF capacitors while R1 and R2 consist of 1-kΩ resistors. The operational amplifier used to make this active filter is a UA741CP. An external power supply of 9-V is used to power the Op Amp and there is a 5-V input at Vin passing through the filter circuit.

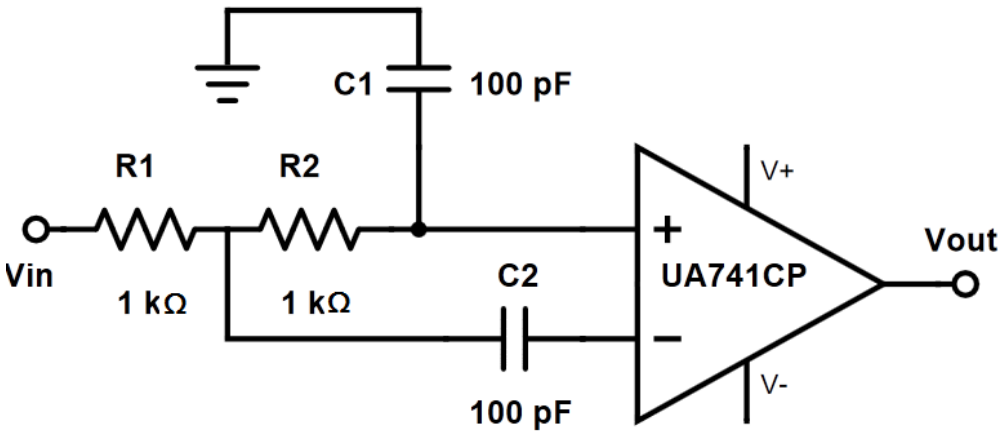


Figure 2.13 LPF circuit diagram

2.4.4 Robotic Finger Servo Motor Circuit

Fig. 2.14 shows the servo motor's internal circuitry and connection with the Arduino Mega. The servo motors used to animate the robotic fingers have built-in potentiometers and error-detecting amplifiers to determine when the motor has achieved the desired position. A 5-V power supply powers the motor. The signal connection of the motor is attached to the PWM pin of the Arduino Mega to communicate the angles necessary to mirror the RF exo-glove finger flexion.

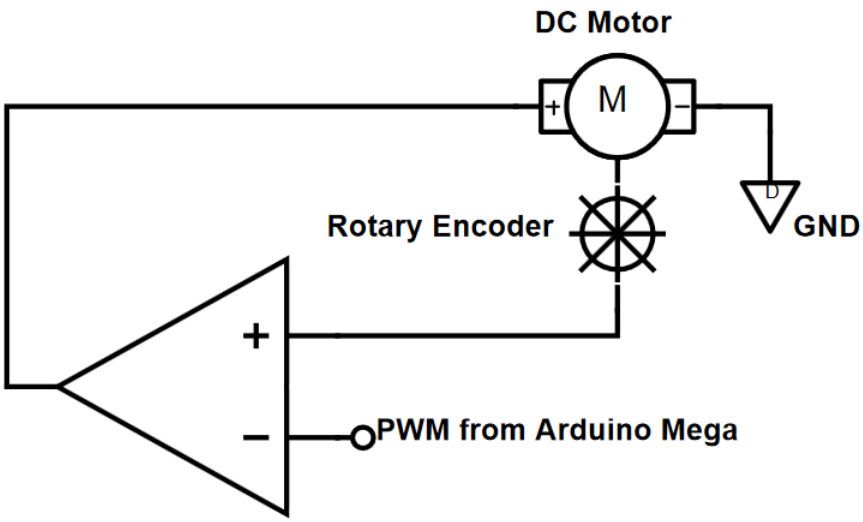


Figure 2.14 Servo motor circuit diagram

2.5 Telerobotic Hand

A robotic hand seen in Fig. 2.15 was created for this experiment to demonstrate how the flexion data can be used in robotic applications. This telerobotic hand is controlled directly through the Arduino Mega by the flexion data received from the RF exo-glove allowing it to mimic finger flexion.

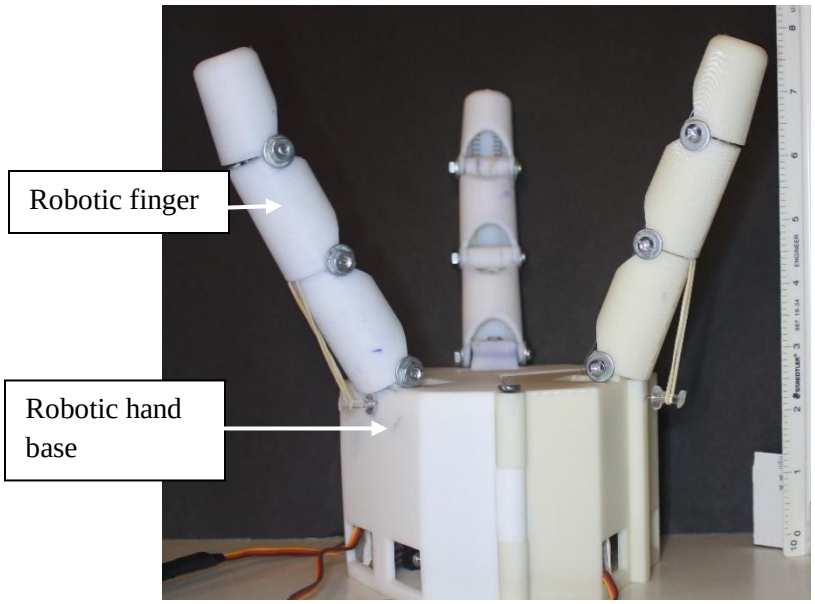


Figure 2.15 Robotic hand

2.5.1 Features

The robotic hand is made up of three 3D printed base housings for servo motors and three 3D printed fingers each consisting of three-finger sections. Each base was designed to pivot at the adjoining connection point giving the robotic hand multiple grasping functions as in Figs. 2.16 and 2.17. Each base has the capability to become

separated from each other allowing the fingers to be operated as standalone robotic finger mechanisms.

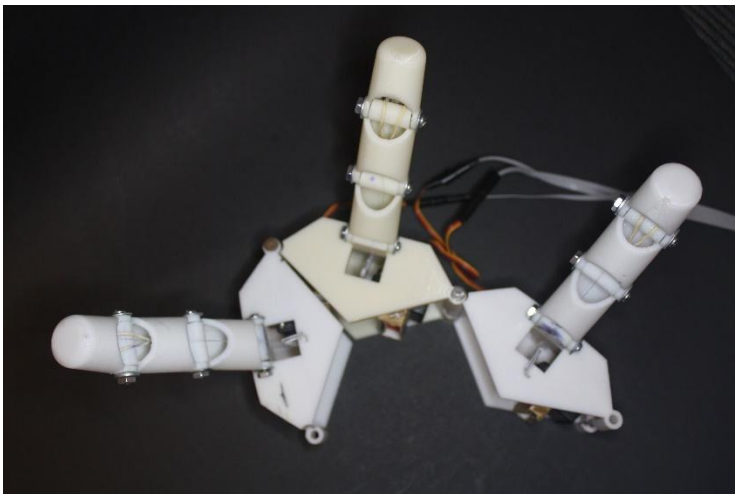


Figure 2.16 Top view of robotic hand with base pivoted outwards

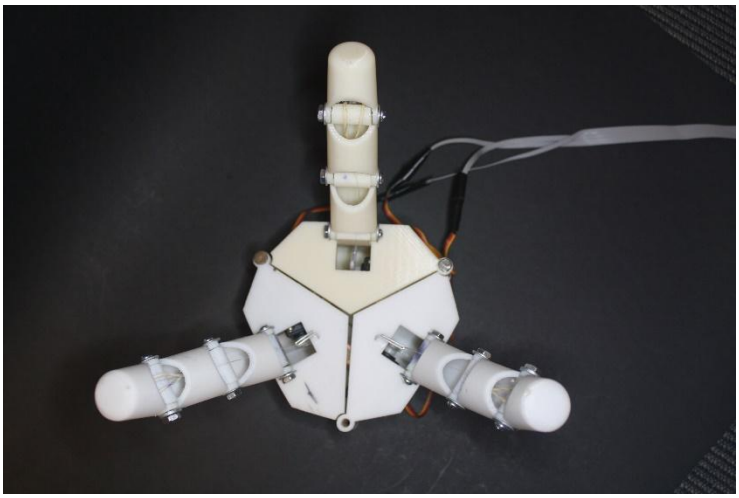


Figure 2.17 Top view of robotic hand in initial position

Each robotic finger is made up of a DIP joint (fingertip), PIP joint (center of the finger), and a MP joint (knuckles). All three fingers were made identical including the thumb. The human thumb is only comprised of two joints whereas for this application, symmetry was desired to show the robotic hand functioning similarly in any directional position. Standard Radioshack servos in Fig. 2.18 were used to operate the robotic fingers. These servo motors provided sufficient torque and speed to mirror the flexion of the RF exo-glove.

Figure 2.18 Standard servo [28]

Each robotic finger is operated by a series of pulleys and guides that is tensioned by a servo motor shown in Fig. 2.19. There are five pulleys, three springs, and one cable in each robotic finger assembly. With this simplified design, a single motor controls the flexion of the finger allowing for a single DOF. If two more motors are added per finger we can achieve control for each individual joint of the robotic finger. As seen in the

figure, when the cable is not in tension the springs will hold the robotic finger in a vertical position causing the robotic hand to be open. The motor is then operated to tension the cable and overcome the spring force causing the robotic hand to close.

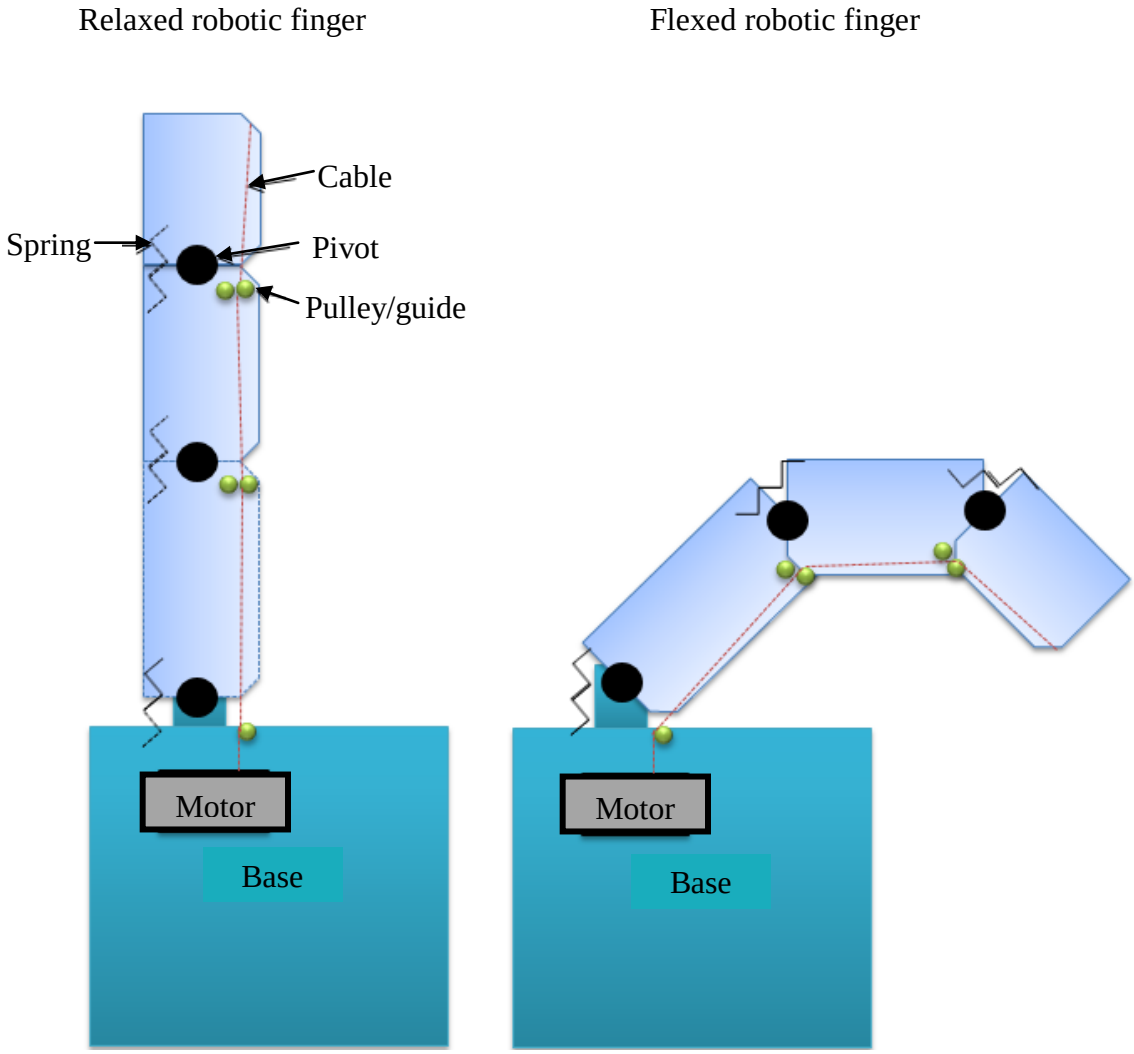


Figure 2.19 Robotic-finger operation mechanism

2.5.2 Constraints

Unlike the eight individual finger joint DOFs seen in the RF exo-glove, the robotic hand has only a single DOF per finger. This constraint of the robotic hand limits the precision of the imitation from the RF exo-glove. However, based on values received for each individual finger of the RF exo-glove, the robotic fingers use an algorithm to achieve a similar result. Each finger joint angle was measured at the maximum flexion achieved amongst all postures tested. The combination of the finger joint angles for each finger and thumb at maximum flexion were set to be the mapping value to the servo motors in the robotic hand.

2.6 Mapping the RF Exo-glove to the Robotic Hand

An open loop controller is the backbone of the RF exo-glove and robotic hand integration. A flowchart in Fig. 2.20 shows the basic operations which take place in the experiment. The setup used in the experimentation uses the human hand as the input in conjunction with the rotary position and flex sensors to relay their signals to the microcontroller. The microcontroller then processes the input values and maps them to the corresponding finger joints in the robotic hand. During this process, values interpreted by the microcontroller for each finger are reduced from individual multiple DOFs, seen in the RF exo-glove, to a single DOF on the robotic hand finger. The encoders in the servos give feedback to the internal circuitry when the desired position has been achieved.

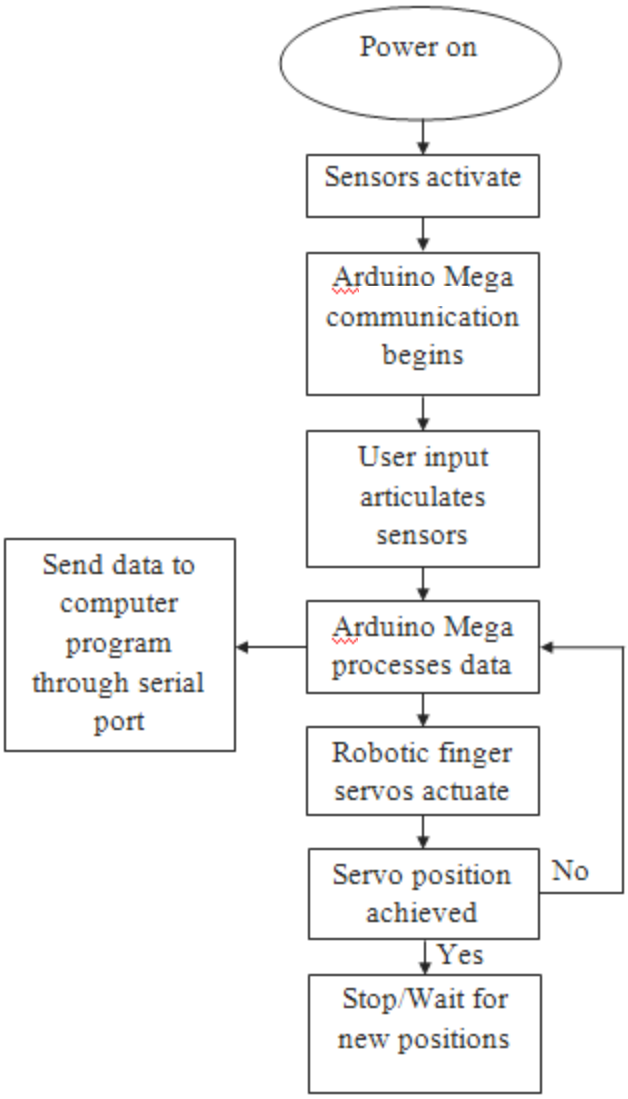


Figure 2.20 Flowchart of operation for RF exo-glove and robotic hand

CHAPTER III

SENSOR CALIBRATION

Before the experiment was performed, each sensor was subjected to calibration. This calibration test was done to reduce fluctuations in data and set initial reference points to track finger flexion.

3.1 Rotary Position Sensors

Initial readings taken from the Arduino serial monitor for the rotary position sensors showed rapidly fluctuating signals within a range of five digits. In order to process these fluctuating signals obtained from the sensors, an algorithm was implemented into the Arduino code that averages the sampled data for every 10 readings. This averaging algorithm allowed the values to be read by observation and improved the accuracy of the sensor reading by reducing the effect of outlier data points.

3.1.1 Calibration Testing

The rotary position sensors on the RF exo-glove vary in values observed due to the position and orientation they were mounted. No two rotary position sensors were in identical situations thus giving unique values for each sensor. Fig. 3.1 shows the RF exo-glove with two rotary position sensors on the middle finger facing opposite of the index finger sensors in Fig. 3.2. This change in orientation is due to the constraints on the RF

exo-glove that allows room for the sensors to pass in proximity to the neighboring finger.

Figure 3.1 RF exo-glove middle finger

Figure 3.2 RF exo-glove thumb and index finger

Each rotary position sensor was measured individually using the Arduino serial monitor to obtain values with a constant power supply input voltage of 5-V. The rotary sensors were rotated from a 0° finger joint angle to 90° as seen in Fig. 3.3.

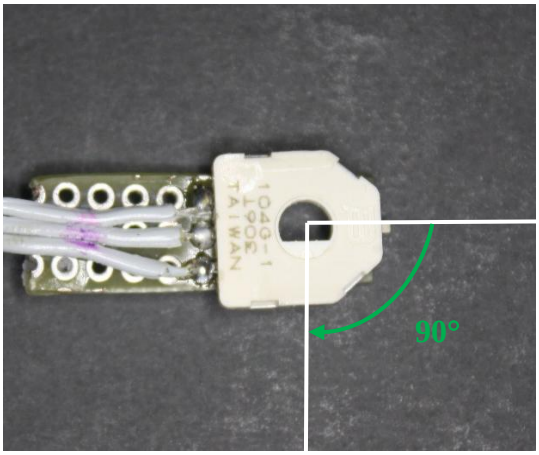


Figure 3.3 Rotary position sensor at 90° angle

The values from the serial monitor were returned as bits ranging from 0 to 1023 due to the built-in 10 bit ADC onboard the Arduino Mega. Based on the values obtained from the rotary position sensors, the fluctuation was obtained using the following formulas:

$$\text{Resolution} = \frac{\text{Flexion angle}}{\text{Change in bits from } 0^\circ \text{ to } 90^\circ} = \frac{90^\circ}{\Delta \text{Bits}} \tag{3.1}$$

Changes in bits for the equilibrium state (ΔBES) at 0° and 90° for each sensor was recorded and integrated into the fluctuation formula.

$$Fluctuation, \quad f = \pm\Delta\text{BES} * \frac{90^\circ}{\Delta Bits} \tag{3.2}$$

In Table 3.1, the results of the rotary position calibration testing is shown. Two hundred and fifty data points were taken for each sensor during 25 seconds at equilibrium. Each sensor had at least one bit of fluctuation when tested at equilibrium while three sensors were shown with two bits of fluctuation.

Table 3.1 Rotary Position Sensor Calibration Results

	Middle Finger DIP	Middle Finger PIP	Index Finger DIP	Index Finger PIP	Thumb DIP
Fluctuation of bits at equilibrium (ΔBES)	±2	±2	±1	±1	±2
Bit value at 0°	585	523	573	598	648
Bit value at 90°	374	305	852	845	857
$\Delta Bits$	211	218	279	247	208
Degrees of fluctuation	0.852	0.826	0.323	0.364	0.862
Resolution (°/bit)	0.426	0.413	0.323	0.364	0.431

Comparing the rotary position sensors at 0° and 90° orientations, it is shown that sensor values for the middle finger experience a drop in bit values. This is caused by the orientation of the sensors on the RF exo-glove being mounted in an opposite manner compared to the remaining rotary position sensors.

3.2 Flex Sensors

Each of the three flex sensors used in this experiment was subjected to a calibration test before experimentation. As previously stated, the values obtained through the serial monitor were averaged for every 10 readings.

3.2.1 Calibration

Each flex sensor was measured individually using a multimeter and the serial monitor of the Arduino software. The Arduino Mega microcontroller was powered at 5-V from an external power supply. To conduct the calibration process, each flex sensor was placed completely flat on a smooth surface and had a small block with a flat surface placed on top of the sensor. The values of the flex sensors were measured once completely flat shown in Table. 3.2. Flex sensor 3 has an equilibrium resistance of over 5-kΩ more than Flex sensors 1 and 2 causing its sensitivity to change.

Table 3.2 Flex Sensor Measurements

	Flex sensor multimeter values (kΩ)	Serial monitor values (bits)
Flex sensor 1	24.05	837
Flex sensor 2	24.33	822
Flex sensor 3	29.68	888

Immediately after the flat measurements were taken, each flex sensor was bent in a 90° angle as in Fig. 3.4 and data was again acquired using the Arduino serial monitor.

Figure 3.4 Flex sensor bent 90°

Using the same equations as used for the rotary position sensors, fluctuation results were obtained. Table 3.3 shows the calibration results for the flex sensors used over the knuckle of the RF exo-glove. Each flex sensor had one bit of fluctuation in the bit values received.

Table 3.3 Flex Sensor Calibration Results

	Middle Finger MP	Index Finger MP	Thumb MP
Fluctuation of bits at equilibrium (ΔBES)	±1	±1	±1
Bit value at 0°	837	822	888
Bit value at 90°	660	568	737
$\Delta Bits$	177	234	151
Degrees of fluctuation	0.510	0.385	0.597
Resolution (°/bit)	0.510	0.385	0.597

Comparing the bit values of the flex sensors at 0° and 90° orientations, we can see that the flex sensor of the index MP joint has the best resolution amongst the three sensors. All three flex sensors had bit values in the eight hundreds but were no identical. There was a 66 bit difference between the highest and lowest flex sensor at 0°.

3.3 Signal Analysis

A signal analysis was conducted on both flex and rotary position sensors while at steady state and in the process of being flexed. The flex sensors used in this calibration test were bent 50° and the rotary position sensors were rotated 40°. Each of the following signal analysis graphs was conducted over five seconds to demonstrate the different rates of sensor value readings. During these five second data gatherings, 40 data points were acquired without using averaging and during times using the LPFs. When averaging was used, only 16 data points were collected due to the delay in response time.

3.3.1 Steady-State Signals

Steady-state signal analysis was implemented to acquire fluctuation of the sensor data while at rest or without flexion. In Fig. 3.5 we have a plot of the steady state values of the flex sensors with and without averaging. The averaged data clearly has fewer fluctuations than the data without averaging. A majority of the data points for both signals remained at 0° with the averaged sample showing 61.1% less fluctuation.

However, using the averaging code to reduce fluctuations caused the fewer fluctuations to increase in flexion angle by 0.053° over the non-averaged data. It is apparent though that the averaged data exhibited the fluctuations during the first half of the data sample only, whereas the sample without averaging had fluctuations throughout the plot.

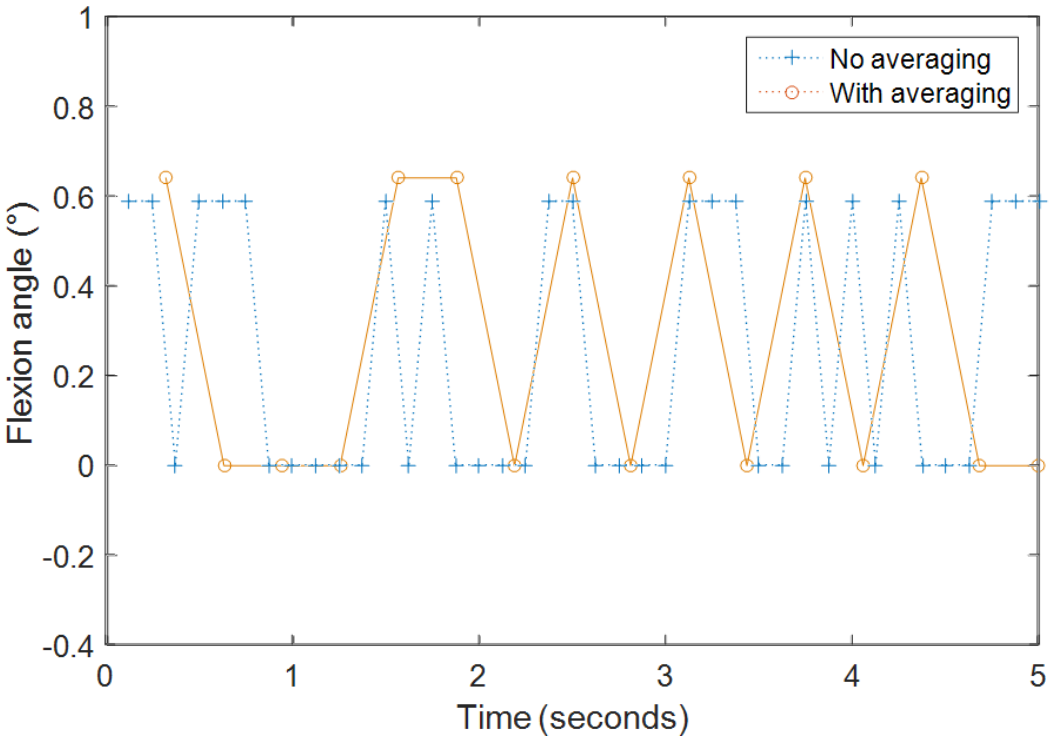


Figure 3.5 Steady-state signals of flex sensors

In Fig. 3.6 the plot shows the rotary position sensor signals obtained without averaging, with averaging, and using the LPFs. With the rotary position sensors, we can see a reduction in data fluctuations when using averaging and LPFs compared to the data that were not averaged. Compared directly to the data sample without

averaging, there was a 72.3% decrease in fluctuations while using LPFs and a 100% decrease in fluctuations using averaging. Once again, there was a trade off in the degree of flexion angle seen when reducing the amount of fluctuations. Using LPFs caused the flexion angle of fluctuations to increase by 0.227° over the non-averaged data.

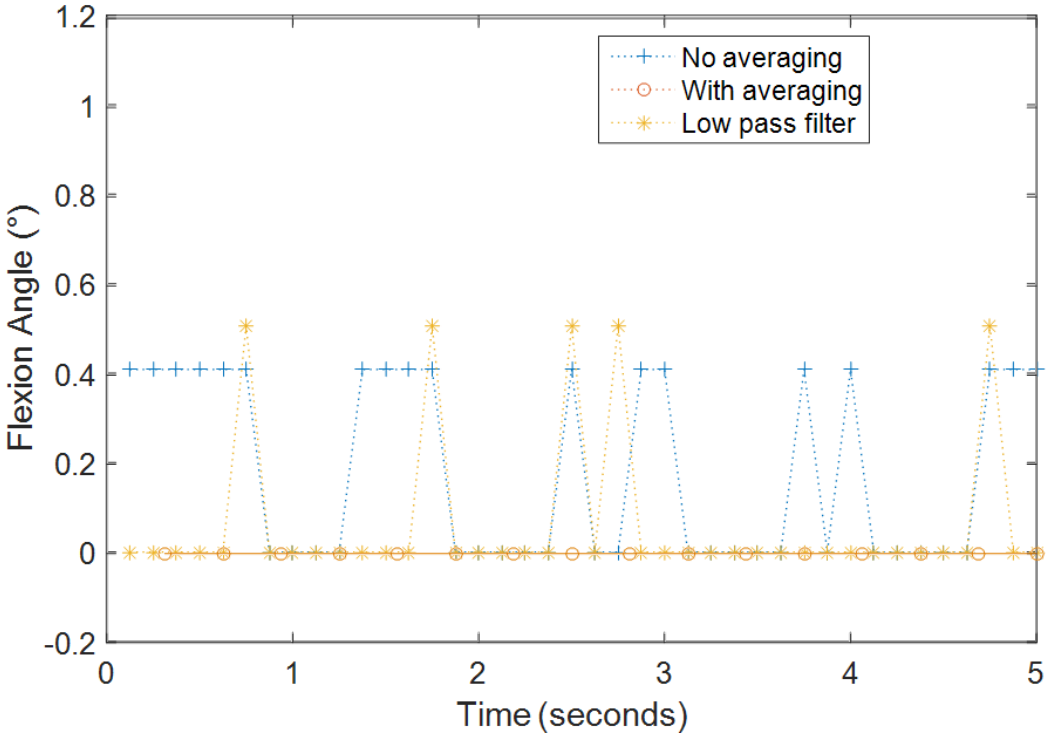


Figure 3.6 Steady-state signals of rotary position sensors

3.3.2 Flexion Signals

A test for sensor flexion signals was conducted to demonstrate the sensor's response to flexion from the initial relaxed position to the rotated orientation and returned back to the relaxed position.

In Fig. 3.7 the plot shows the response of the flex sensor with and without averaging. The data set without averaging has immediate leaps in flexion angle as well as a small negative slope a third of the way through the plot. Using averaging creates a smoother and gradual transition to the peak angle and return. However, there considerably less data points involved which causes sudden jumps in values from one data point to the next.

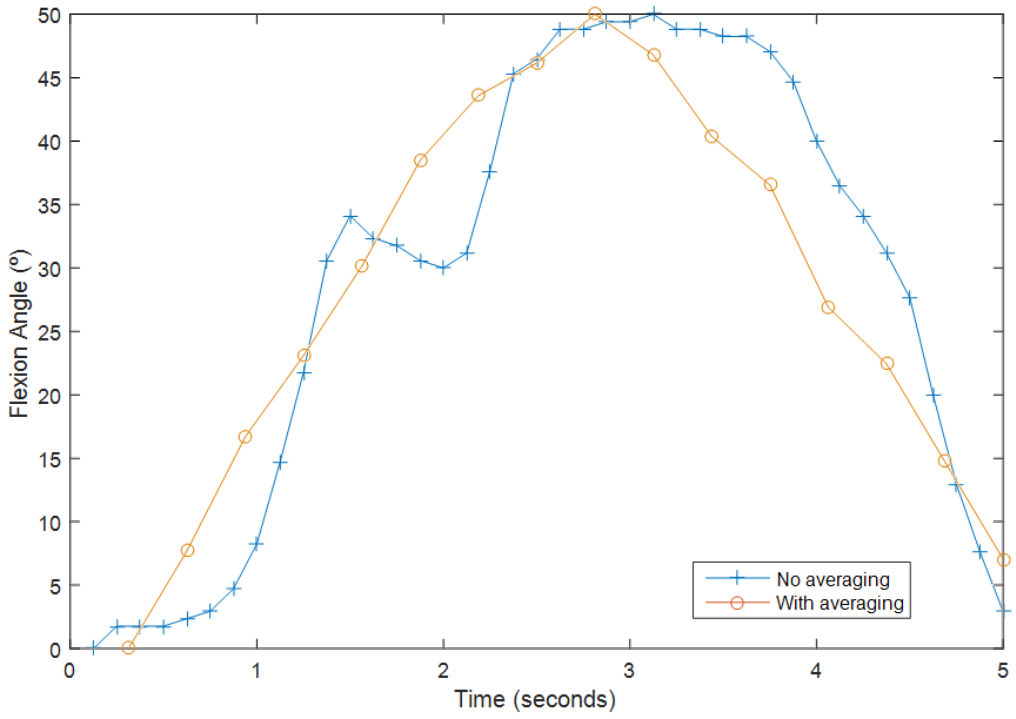


Figure 3.7 Signal of flex sensor while bent

Fig. 3.8 shows the plot of the rotary position sensors while being rotated without averaging, with averaging, and while using LPFs. Without averaging we have a plot similar to the results of the flex sensor testing. There is a sudden leap of values rising to the peak along with a negative slope in the flexion angle one third of the way through the plot. Using LPFs and averaging give similar results to one another. There are some points in the plot that are not as consistent in the slope as other parts but overall the averaging, and the LPF data were able to show better signal response throughout the plot.

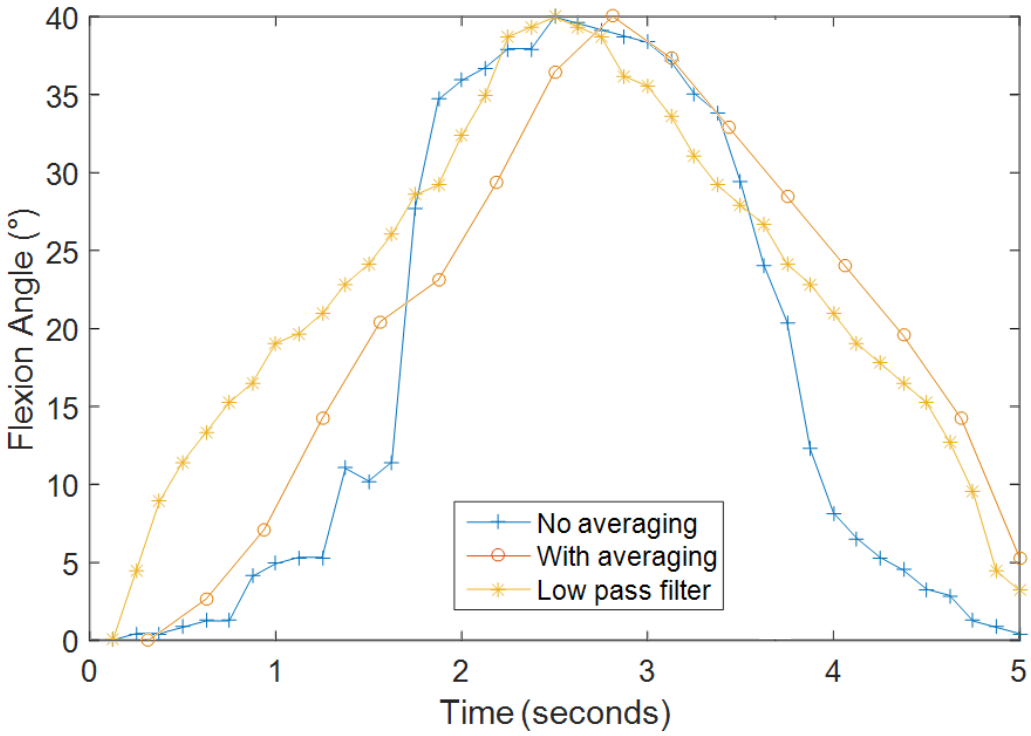


Figure 3.8 Signal of rotary position sensors while rotated

CHAPTER IV

EXPERIMENTAL RESULTS

After the intermediate calibration was complete, the flex sensors wee placed into the RF exo-glove pockets created to house the flex sensors and the rotary position sensors placed into their appropriate positions. The low pass circuit was connected to the rotary position sensors and external power supplies were attached. To begin the initial testing phase, each finger joint was flexed around the knuckle and measured using the Arduino serial monitor.

4.1 Experimental Testing

In Fig. 4.1 we can see each finger joint has the flexion angle measured against the previous finger joint or base of the hand. Angle θ is the change in angle from the initial position of the finger at rest to the final position where the finger completes flexion. Essentially, θ corresponds to the flex sensor values for the finger with respect to the knuckle.

Since the finger joints farthest away from the hand use rotary position sensors, the angles measured are taken from the revolute point of the finger joints in the center of the joint. Angles α and β are the flexion angles of the RF exo-glove center finger section and fingertip respectively. Similarly to the measurement of angle θ, angles α and β are measured with respect to the previous finger joint position as shown in Fig. 4.1.

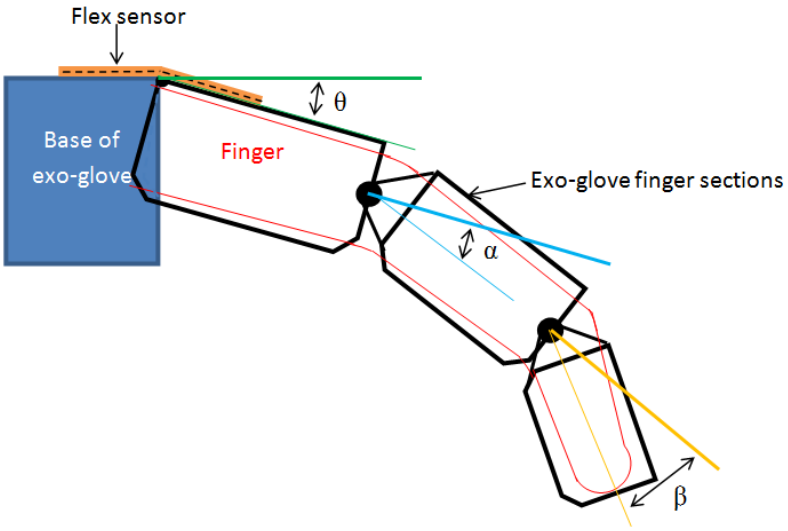


Figure 4.1 RF exo-glove sensor angles

4.1.1 Postures

The experiment was carried over four different postures using the RF exo-glove. As seen here in Fig. 4.2, the four different postures include a relaxed hand pose, grasping a water bottle, holding a pen, and a clenched fist.

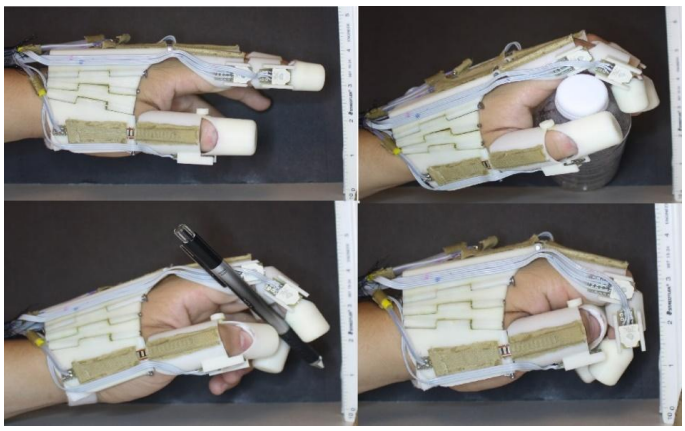


Figure 4.2 RF exo-glove postures (a) relaxed sensors, (b) grasping water bottle, (c) holding a pen, and (d) clenched fist

These postures are very common ones that people make and allow for great data acquisition by flexing each finger joint in multiple ranges. The items in Fig. 4.3 were used for grasping postures in this experiment which include a half-liter water bottle and a standard pen.

Figure 4.3 Water bottle and pen used for grasping postures 2 and 3

4.2 Experiments in Multiple Postures

The first posture tested was the RF exo-glove placed in the initial relaxed position resulting in the fingers being nearly flat across the joints. Generally a person does not hold their hand, when relaxed, perfectly straight which would mean their finger joints are already angled. This posture was tested first to set reference values the additional postures could be measured against. In this position 30 seconds was allowed

to pass before retrieving data to ensure the sensors reached steady state equilibrium. Data was logged from the serial connection of the Arduino Mega using the Parallax Data Acquisition tool (PLX-DAQ) [29]. The PLX-DAQ gathered flexion information of the sensors from the Arduino serial port and transferred them to a Microsoft Excel spreadsheet. The data logged for the relaxed sensors was obtained in bits ranging from 0 to 1023 due to the 10 bit ADC on the Arduino Mega. Using the serial monitor data taken from each sensor reading during the calibration process, each column of sensor data was evaluated for accuracy at steady state.

Table 4.1 shows the actual angles measured from the RF exo-glove and the offset of the reference posture angles.

Table 4.1 RF Exo-glove Posture Angles

	Flexion Angles of RF exo-glove							
Posture	Middle Finger MP	Index Finger MP	Thumb MP	Middle Finger DIP	Middle Finger PIP	Index Finger DIP	Index Finger PIP	Thumb DIP
Bottle	22	18	15	26	62	11	68	54
Fist	22	25	13	60	68	45	66	65
Pen	15	12	15	34	62	41	55	57
	Reference Posture Angles							
	13	10	10	5	5	5	5	5
	Adjusted Angles							
Bottle	9	8	5	21	57	6	63	49
Fist	9	15	3	55	63	40	61	60
Pen	2	2	5	29	57	36	50	52

Once the reference postures were acquired, each of the three remaining postures underwent the same process for data acquisition. Using the resolution found during the sensor calibration, I was able to take the difference between the reference posture and the alternative posture to obtain a difference in bit values sensor by sensor. The difference in bit values was then multiplied by the resolution to find the change in flexion angles for each finger joint.

The following graphs consist of experimental results from the previously stated postures with and without averaging. Each posture was tested five times over separate data-acquisition samplings. In total there were 40 samples taken. Each graph shows five separate tests run on the particular posture with or without averaging. The x-axis is labeled with the corresponding fingers tracked with the RF exo-glove. The y-axis is labeled with the flexion angle to show the degrees of flexion angles of the data acquired. Each PIP and DIP joint is tracked with a rotary position sensor. Each MP joint uses a flex sensor for flexion tracking.

4.2.1 Bottle Grasping Posture

In Figs. 4.4 and 4.5 we have box plots of the adjusted actual flexion angles measured directly off of the RF exo-glove and the range of sensor data obtained in the samplings for the bottle grasping posture. From direct observation, the box plots for the averaged data in Fig. 4.5 shows better precision for the rotary position sensors. However, the non-averaged data in Fig. 4.4 was more precise for the flex sensors. Evaluating each

finger and thumb in both figures gives mixed results. Each MP joint performed better without averaging in terms of precision, but had an even trade-off in the accuracy shown in comparison with both plots. The PIP joints were more accurate without averaging but compromised the precision of the data. The DIP joints were slightly more accurate without averaging, However, the precision gained by averaging is a great improvement in performance.

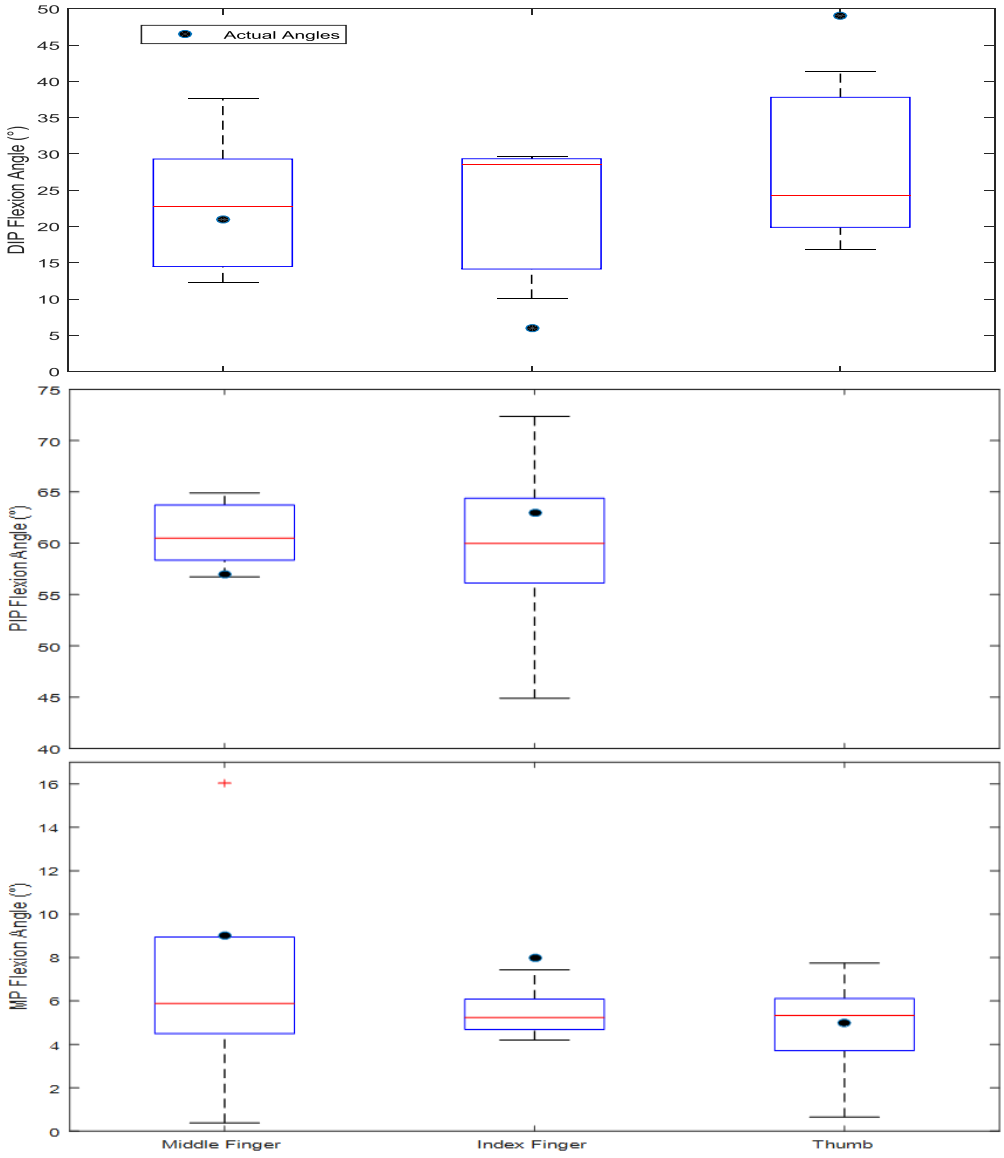


Figure 4.4 Bottle grasping posture without averaging

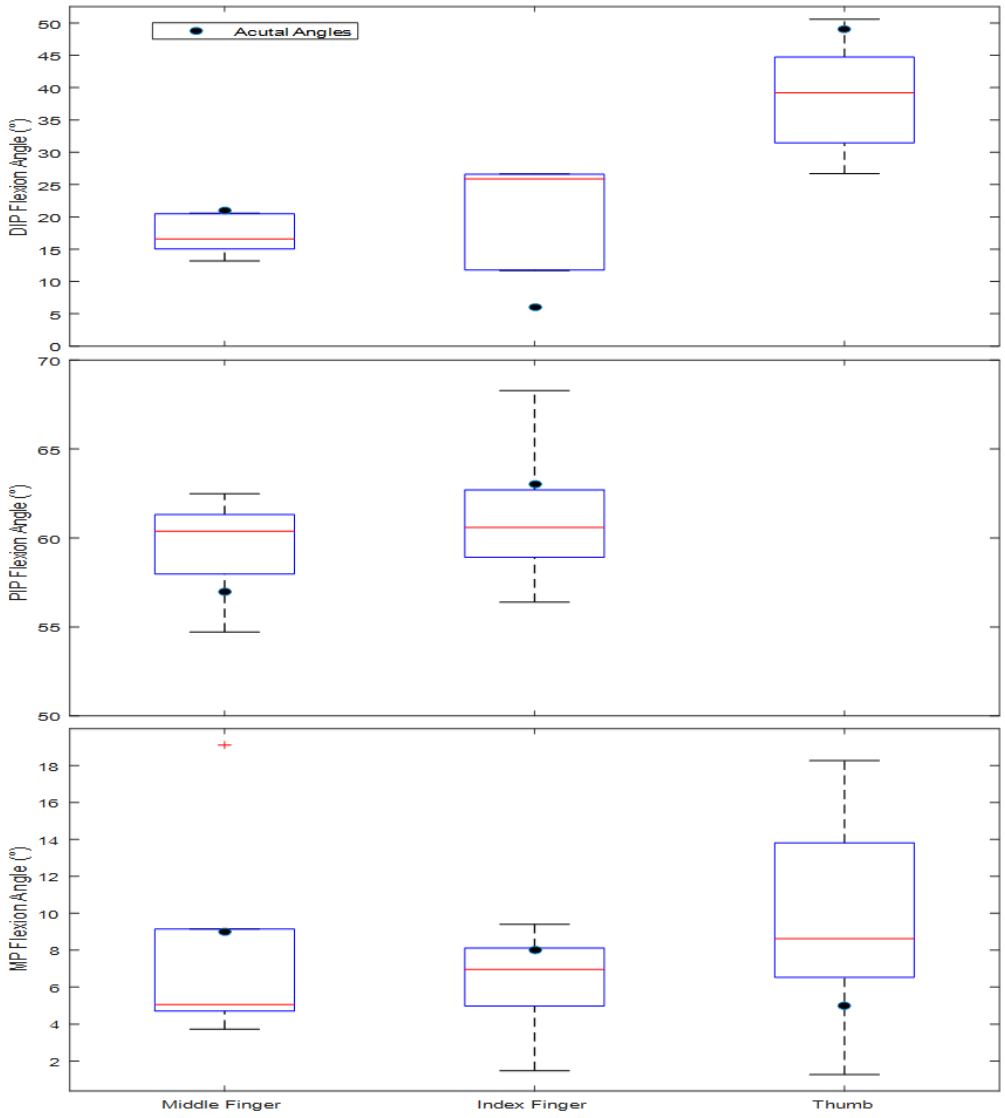


Figure 4.5 Bottle grasping posture with averaging

The observations from Figs. 4.4 and 4.5 can be seen directly on table 4.2. Table 4.2 and proceeding Tables following the posture graphs use the following convention to identify the sensors:

1. Middle finger MP
2. Index finger MP
3. Thumb MP
4. Middle finger DIP
5. Middle finger PIP
6. Index finger DIP
7. Index finger PIP
8. Thumb DIP

The standard deviation of the averaged data is lower for each sensor with the exception of sensor 2 which is greater by 0.24°. The average flexion angle difference also has better performance with the averaging data which displays lower angle differences for all but three sensors. Given the difference in angles and standard deviation from the true angle that was too be met, there was an increase in performance using averaging.

Table 4.2 Flexion Angle Average Difference and Standard Deviation for Bottle Grasping Posture

Average Flexion Angle Difference								
Sensor	1	2	3	4	5	6	7	8
No averaging	4.55	2.55	1.86	8.19	4.24	13.18	8.47	16.87
Averaging	5.29	2.23	6.25	3.72	3.43	14.52	3.94	11.18
Standard Deviation of Flexion Angles								
No averaging	3.09	1.18	1.72	5.84	3.42	9.60	6.16	12.55
Averaging	2.79	2.47	4.25	3.19	1.38	8.00	1.94	8.13

4.2.2 Clenched Fist Posture

In Figs. 4.6 and 4.7 a box plot of the clenched fist posture data can be seen with the adjusted actual angles. Looking at the range of data in the plots, it is immediately apparent that the flexion data obtained does not fit the actual flexion angles. There were very few data points that coincided with the true angles, but it should be noted that the data was offset almost consistently across the plot. Flexion values in Fig. 4.6 varied by 46.1° for the middle finger DIP joint, but when compared to the averaged data plot, the same sensor improves precision by 72.6%. Again, the averaged data remains more precise but does not have the accuracy to display the true angles.

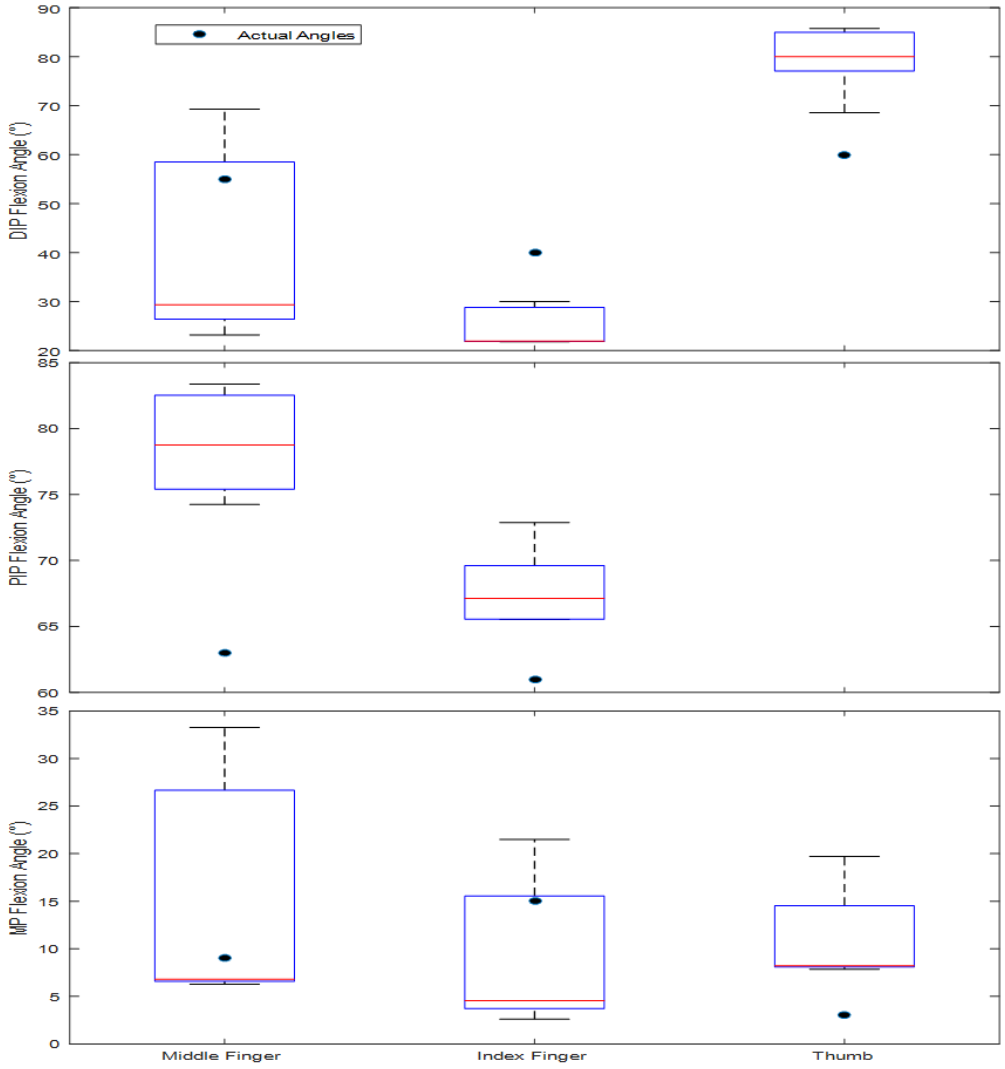


Figure 4.6 Clenched fist posture without averaging

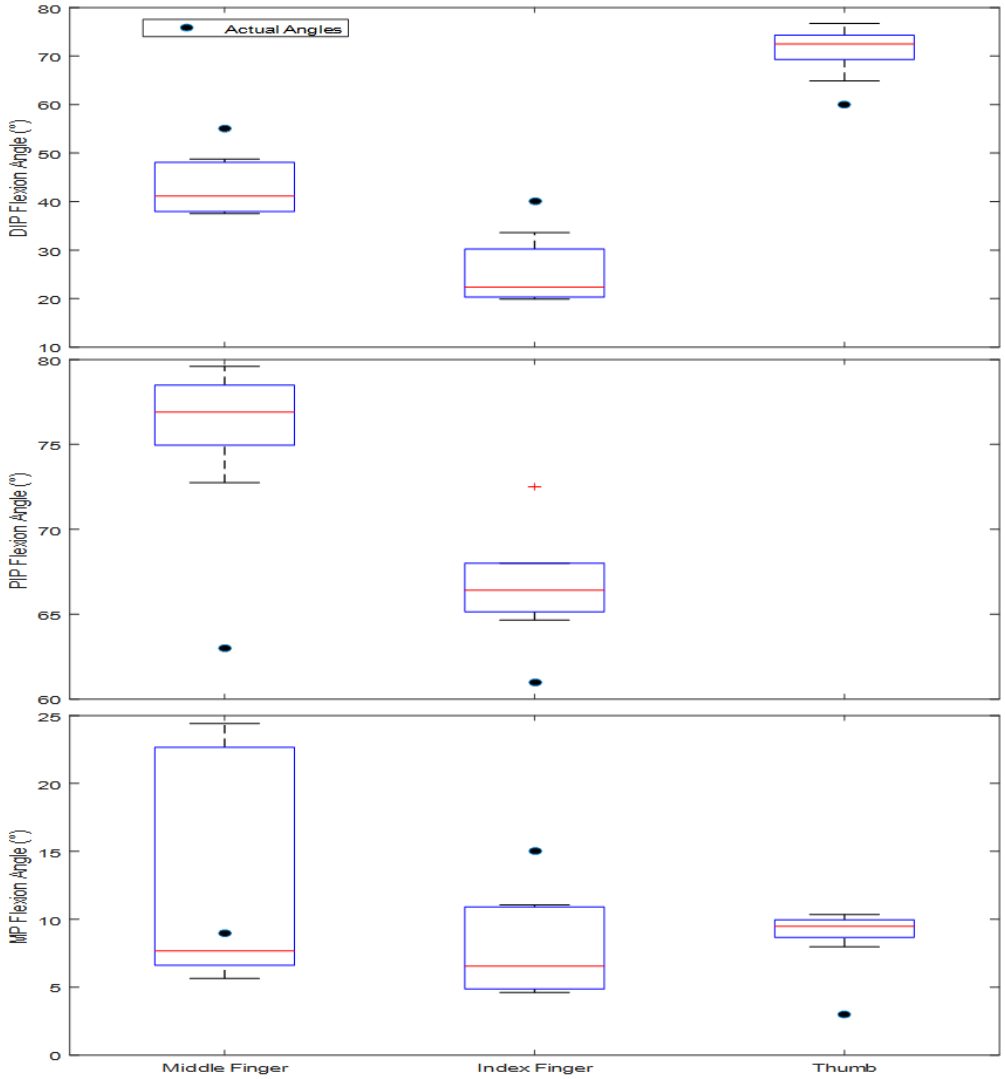


Figure 4.7 Clenched fist posture with averaging

Table 4.3 shows the average flexion angle difference and standard deviation for the data samplings of the clenched fist postures. Overall, using averaging resulted in a lower standard deviation and average flexion angle difference from the true angles. There is at least a 34% improvement in standard deviation of the flexion angles with averaging.

Table 4.3 Flexion Angle Average Difference and Standard Deviation for Clenched Fist Posture

Average Flexion Angle Difference								
Sensor	1	2	3	4	5	6	7	8
No averaging	9.39	8.34	8.35	19.86	15.88	15.17	6.92	19.82
Averaging	7.05	7.39	6.31	12.33	13.61	14.94	6.07	11.67
Standard Deviation of Flexion Angles								
No averaging	10.05	4.44	5.10	12.82	3.96	4.07	3.04	6.81
Averaging	6.66	3.15	0.91	5.34	2.60	6.01	3.13	4.38

4.2.3 Pen-Holding Posture

In Figs. 4.8 and 4.9 we have box plots that show the flexion angle data for the pen holding posture. The Fig. 4.8 plot shows a slightly better performance in accuracy for the middle finger MP over Fig. 4.9 but does not hold this accuracy throughout the plot. Once again, the plot without averaging displays overall greater accuracy but lacks the sensor precision gained by averaging. Both plots follow the general pattern of the true flexion angles but are offset by a similar degree value. The offset angle varies from sensor to sensor ranging from an average of 1.91° to 13.86° for the averaged data and 4.15° to 14.54° for the non-averaged data. Fig. 4.8 has two outliers that are effectively removed with the averaging function.

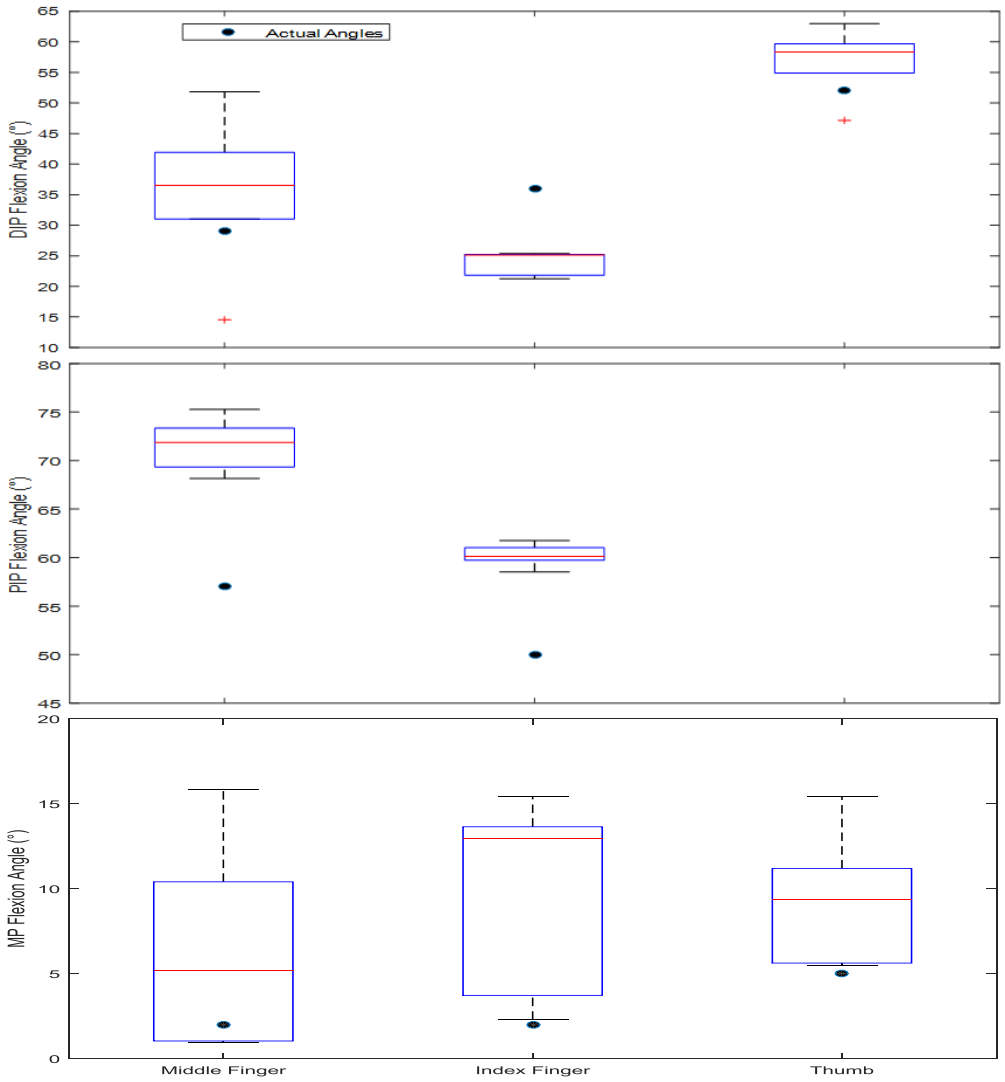


Figure 4.8 Pen holding posture without averaging

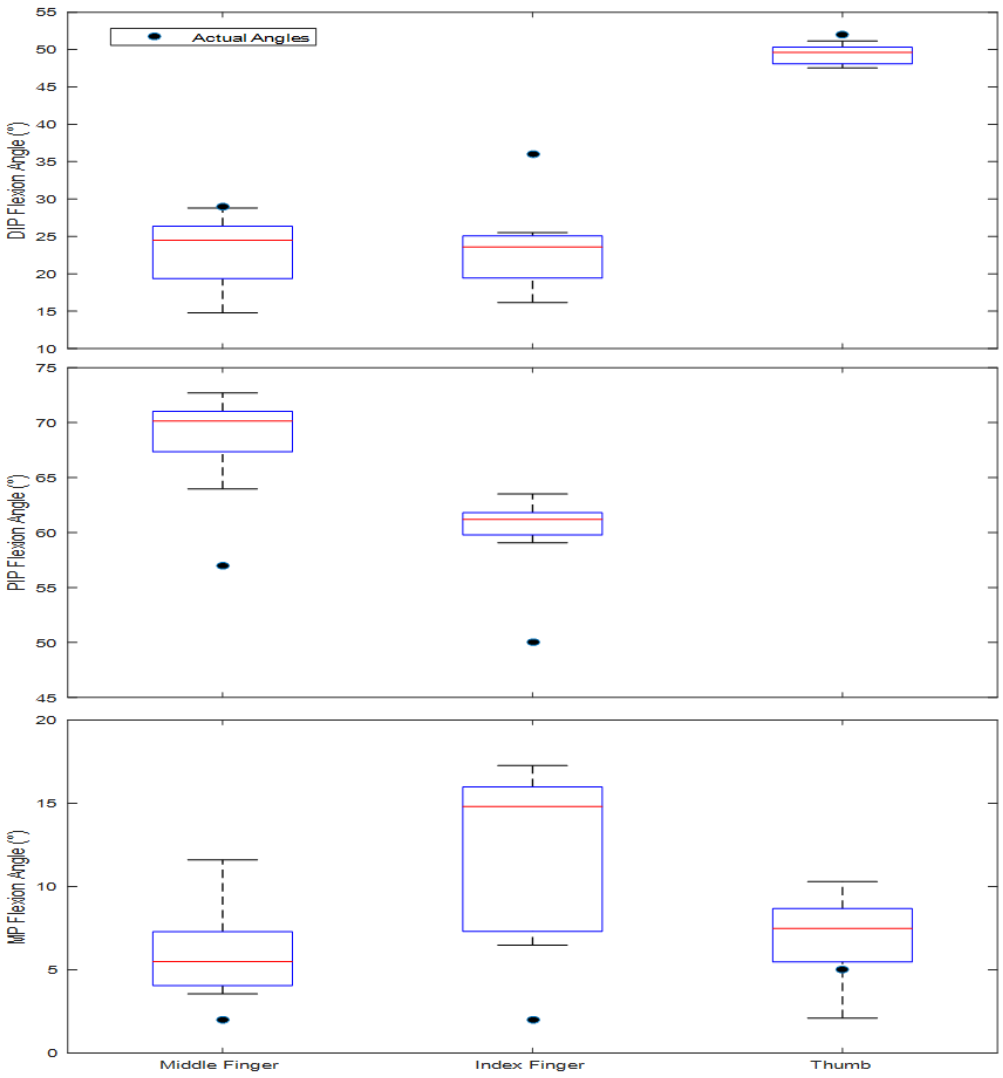


Figure 4.9 Pen holding posture with averaging

Table 4.4 shows the average flexion difference and standard deviation of the pen holding posture data compared to the true flexion values. In this pen holding posture we see that the flexion angle difference is better when using averaging for five of the eight sensors. However, looking at the standard deviation of the flexion angles shows that

there are four sensors in favor for averaging and without averaging giving a mixed result.

Table 4.4 Flexion Angle Average Difference and Standard Deviation of Pen Holding Posture

Average Flexion Angle Difference								
Sensor	1	2	3	4	5	6	7	8
No averaging	5.12	7.58	4.15	12.38	14.54	12.21	10.26	6.84
Averaging	4.14	10.33	3.08	6.10	12.15	13.86	11.00	2.67
Standard Deviation of Flexion Angles								
No averaging	5.38	5.91	4.04	6.50	2.74	1.98	1.18	2.40
Averaging	3.19	4.94	1.37	5.35	3.27	3.86	1.66	1.43

4.3 Telerobotic Hand Mapping

The lack of a closed-loop controller in this experiment causes the robotic finger flexion to be mimicked in a way that the finger jumps to the next position sent in the signal. The signal from the RF exo-glove is almost immediately sent to the robotic hand which will in turn make the servos react. Since there is no processed feedback to the servo motor, other than the internal circuitry confirming the desired position, it will constantly rotate at the same speed regardless of how close it may be to the next position.

In order to manipulate the robotic hand, the finger-joint flexion values needed to be processed for proper mapping. The average total joint flexion of each finger was taken to be mapped onto the robotic hand following Fig. 4.1. The flex sensor angle value is noted as θ, the center rotary position sensor angle value is represented as α, and the fingertip rotary position sensor angle value is shown as β.

The formula shown here was used to gather the average flexion of the middle and index fingers:

$$avg = (\frac{\theta+\alpha+\beta}{3}) \tag{4.1}$$

This averaging equation was used for tracking the flexion of the thumb:

$$avg = (\frac{\theta+\alpha}{2}) \tag{4.2}$$

The preceding equations were used to control the robotic fingers while reducing the multiple DOFs for each RF exo-glove finger into a single DOF. It was necessary to reduce the DOF to operate the robotic hand using only three servo motors. Using the averages shown above each servo was mapped to 180° allowing the robotic finger to mimic the flexion done by the RF exo-glove. Table 4.5 shows the averaged data, using (4.1) – (4.2), to map the finger flexion to the robotic hand. The "Begin" value is the initial start of the robotic hand mapping. The "End" values designate the terminal values the robotic hand receives when the servos are positioned at 180°.

Table 4.5 Averaged Data Used for Robotic Hand Mapping

Robotic Finger Map Values		
	Begin	End
Thumb	768	797
Middle Finger	648	446
Index Finger	664	762

The Arduino Mega used to communicate between the RF exo-glove and robotic hand sends signals through PWM at 490-Hz. This frequency is not particularly high, and due to the averaging there is a delay in real-time control. While operating the RF exo-glove to control the robotic hand, the signals of the sensors and PWM pins of the Arduino Mega were recorded to find the time delay between sensor reading and robotic hand motion. The delay in response for the non-averaged sensor values was a quarter of a second. This delay was increased when using the averaging of sensor values by 0.375 seconds over the non-averaging. The immediate delay for averaging the sensor values between posture and robotic finger motion was 0.625 seconds.

4.3.1 Mapping Relaxed Hand Posture

Here in Fig. 4.10 we can see the relaxed hand posture mimicked with the robotic hand. Since this is the reference posture, the robot hand's fingers remain in a relaxed position as well due to the sensor mapping. Monitoring the sensor values for this state will yield the initial robotic finger mapping values seen in Table 4.5. When testing the

relaxed hand posture, the data set from the sensors varied more than the servo mapping was created for. This caused the fingers to twitch and partially flex. Observations between using the averaging code and without averaging showed better results for the averaging attempt. Sensor values were smoother leading to less twitching.

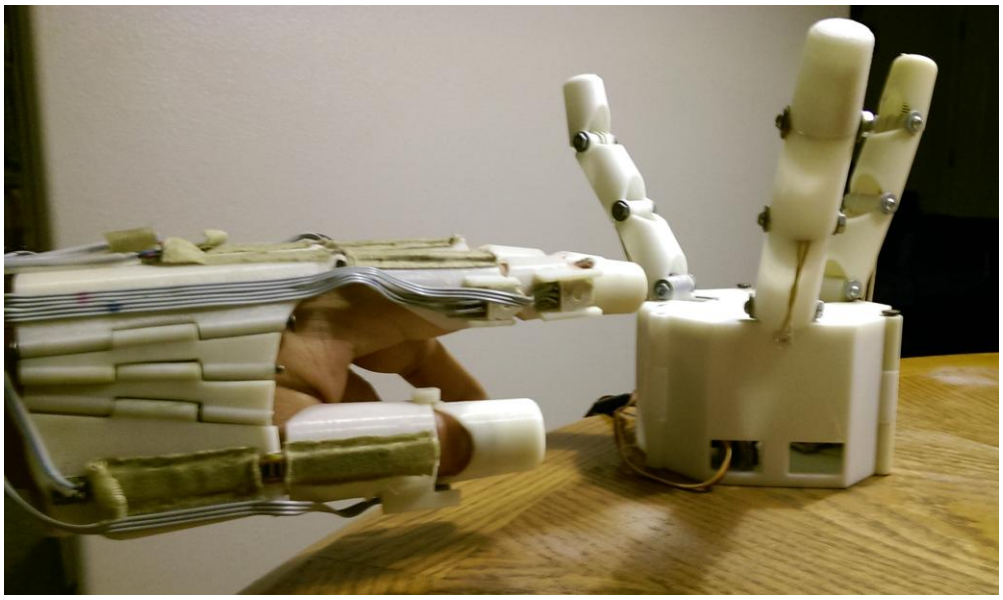


Figure 4.10 Relaxed posture with robotic hand

4.3.2 Mapping Bottle Grasping Posture

Depicted in Fig. 4.11 is the bottle grasping posture. The robotic fingers were able to flex in a similar manner to the degree given by the RF exo-glove but is unable to completely close the finger grip. This posture is just about mid way through the servo rotation leaving 90° additional flexion. As with the relaxed posture, using the averaging

code resulted in much less twitching in comparison to not using averaging. Unfortunately, using averaging in the code meant there was more delay in the sensor values being read. The delayed sensor reading directly translated to the reaction time the robotic fingers could move.

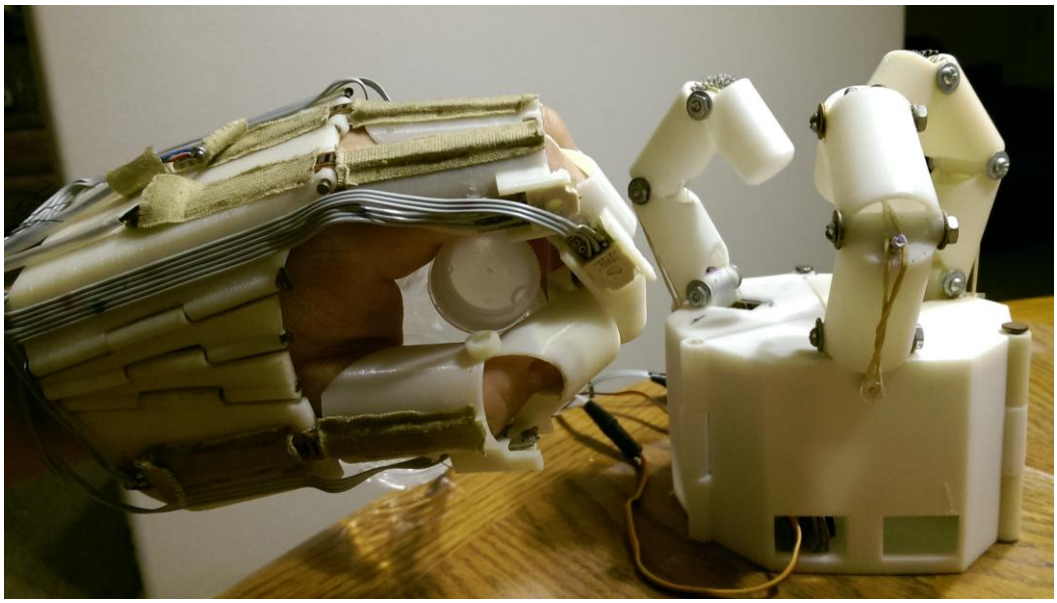


Figure 4.11 Bottle grasping posture with robotic hand

4.3.3 Mapping Clenched Fist Posture

With Fig. 4.12 we can see the clenched-fist posture mirrored by the robotic hand rather well to its abilities. Again, the fingers are unable to completely close against each other which is due to the single DOF for each robotic finger. At this posture, all servos have rotated 180° with sensors reading the end results in Table 4.5 for the averages.

Similarly for this posture, using averaging gave better results for holding the desired angles.

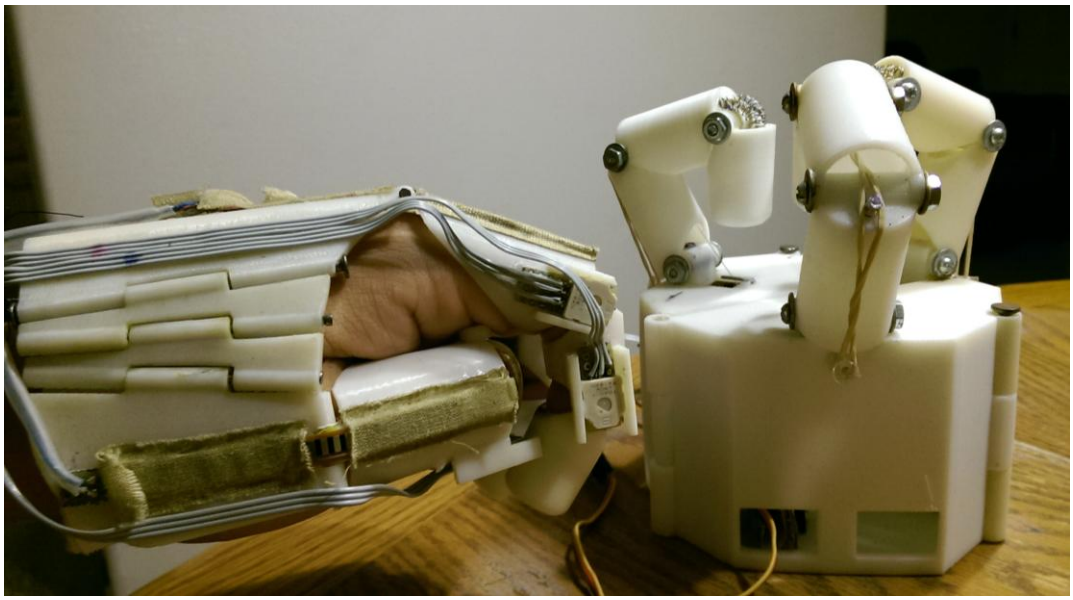


Figure 4.12 Clenched fist posture with robotic hand

4.3.4 Mapping Pen-Holding Posture

Fig. 4.13 shows the posture for holding a pen. This posture is similar to the bottle grasping posture while using the RF exo-glove due to its bulky parts. Of course the posture angles are not identical to the bottle grasping posture leading to the middle and index fingers to be less flexed. With this test the averaging code again present better results of holding the posture with less twitching.

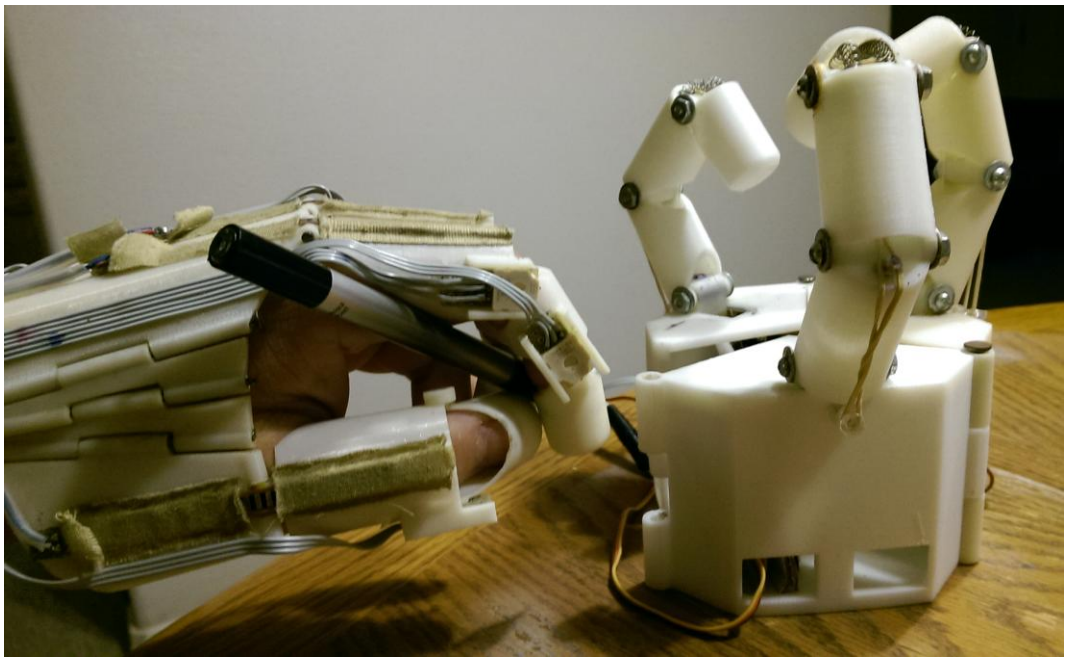


Figure 4.13 Pen holding posture with robotic hand

Although each posture was meant to represent common ones people make, the robotic hand itself was not designed to hold the bottle or pen and thus they are used for demonstration purposes only. The transitioning between postures displayed from the robotic hand was done very quickly. When experimenting without the use of averaging, the robotic hand had the ability to full close from the relaxed position in under half a second. However, when using averaging, the ability to react in real time was hindered by the code delays more than doubling the time to complete the same task. Mirroring the RF exo-glove posture was my goal in these experiments and thus averaging was more desirable.

CHAPTER V

CONCLUSIONS

5.1 Conclusions

The initial motivation to design this unique RF exo-glove was to experiment with a finger-flexion-tracking device that utilized a rigid frame for repeatable results. This holds true up to a certain point, however there were a few instances where precision was not seen in the data. Readings for the sensors were too varied to consider this to be a precise finger-flexion-tracking apparatus. The best results were seen with the bottle-grasping posture having six sensors with averaging be within 6° of the true flexion angles.

The lack of precision was mostly apparent in the sampling that was done without averaging. In some cases the range of values for a single sensor were scattered over 40°. Averaging proved to be effective in reducing fluctuations in data acquired but at the cost of somewhat greater flexion angle offset. The offset seen was manageable leading to only a fraction of a degree change making the compromise worth the vice. The data for using averaging showed up to a 45.2% increase in precision compared to non-averaged values.

Over several sets of sensor calibration, the resulting data were always within at least 2 bits of each set. The major contributing sources to the varied results are the dissimilar initial sensor values, slight changes in postures, and personal movement. The sensors used in this research were to be identical to one another per type. Unfortunately,

the reading of one flex sensor's value was considerably different from the other two. Similarly, testing all of the rotary position sensors revealed variation in sensor data.

The postures used in this research were meant to be easily repeatable, however, using an amorphous bottle and postures that could be manipulated with slight motion left mixed results. With over 40 experimental tests, some changes in flexion angles must have existed leading to data that would show up differently each time. Based on the fluctuation of the sensors while the RF exo-glove was on my hand and the sensor calibration testing alone, there seems to be interference in the data. During the experiments while using the RF exo-glove I could feel my fingers slightly moving which could have been the cause of the spikes in the results. On top of my natural movements when still, the RF exo-glove frame caused my blood to pulse in my fingers further changing minor sensor values. Each experiment was conducted in the same location under identical conditions, but my own variances in remaining still change very often.

5.2 Future Work

The design used in this experimental setup has several areas where improvements can be made for more accurate results. Looking at the tolerances of the finger shafts used to rotate the rotary position sensor, there is a small gap between the inner walls of the sensor and the shaft of the RF exo-glove finger joint. Future postures used in analysis should be easily repeatable with little variance in dimensional changes or characteristics of changing the postures.

Higher-strength materials could allow for a less bulky design which would lead to all five fingers being able to have tracking sensors. Higher-precision rotary position sensors and flex sensors could provide a superior range of accuracy for the flexion tracking.

Additional DOFs can be added to the design to make the RF exo-glove more operable and functional. A recommended site for and additional degree of freedom is the top of the knuckle. Since the finger joints at the knuckle allow the finger to move side to side, there can be another DOF to be exploited.

The Arduino Mega microcontroller has an onboard 10-bit ADC that was used in the experiments giving a maximum range of values to be from 0 to 1023. By adding an external analog to digital converter of 12 bits or more, the values obtained for the flexion angle can be far greater. For example, the 10-bit converter gives us 1024 values while a 12-bit converter would give us 4096 values. Using the simple formula of 2^x where x represents the bit value of the converter, we can see that a larger bit converter will yield a better resolution.

Signal-processing filters should be used on all sensor signals when gathering results from the system to remove outlier data points from being part of the analysis. Noise filters should be used on power supplies used in the system if they are known to be unstable or their quality is uncertain.

REFERENCES

[1] L. A. Jones and S. J. Lederman, *Human Hand Function*. Oxford: Oxford University Press, 2006. http://books.google.com/books?id=InyYROA6j_0C&printsec=frontcover&dq=human+hand&hl=en&sa=X&ei=R60LVPu7GYqdygTvhIHwAw&ved=0CB8Q6AEwAA#v=onepage&q=human%20hand&f=false

[2] M. Schünke, E. Schulte, U. Schumacher, L. Ross, and E. Lamperti, *Thieme Atlas of Anatomy*. Stuttgart: Thieme, 2006.

[3] Y. Liu and D. Sun, *Biologically Inspired Robotics*. Boca Raton, FL: CRC Press, 2012.

[4] S-H. Bae, Y. Lee, B. Sharma, H-J. Lee, J-H. Kim and J-H. Ahn, "Graphene-based transparent strain sensor," *Carbon*, vol. 51, pp. 236–242. Science Direct, Jan 2013. http://www.sciencedirect.com/science/article/pii/S0008622312007002

[5] A. Drimus, G. Kootstra, A. Bilberg and D. Kragic, "Design of a flexible tactile sensor for classification of rigid and deformable objects," *Robotics and Autonomous Systems*, vol. 62, no. 1, pp. 3–15. Science Direct, Jan 2014. http://www.sciencedirect.com/science/article/pii/S092188901200125X

[6] N. M. Noaman, A. R. Ajel, and A. A. Issa, "Design and implementation of DHM glove using variable resistors sensors," *Journal of Artificial Intelligence*, vol. 1, no. 1, pp. 44–52. Science Alert, 2008. http://www.scialert.net/abstract/?doi=jai.2008.44.52

[7] N. P. Oess, J. Wanek and A. Curt, "Design and evaluation of a low-cost instrumented glove for hand function assessment," *Journal of NeuroEngineering and Rehabilitation*, vol. 9, no. 1, doi: 10.1186/1743-0003-9-2. BioMed Central, Jan 2012. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3305482/

[8] R. Y. Wang and J. Popović "Real-time hand-tracking with a color glove," *ACM Trans. Graph.*, vol. 28, no. 3, doi: 10.1145/1531326.1531369. ACM Digital Library, Aug 2009. http://dl.acm.org/citation.cfm?doid=1531326.1531369

[9] J-M. Guo, Y-F. Liu, C-H. Chang and H-S. Nguyen, "Hybrid hand tracking system," *2011 18th IEEE International Conference on Image Processing (ICIP)*, Brussels, pp. 549–552. IEEE Xplore, Sept 2011. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6116404

[10] J-M. Guo, Y-F. Liu, C-H. Chang and H-S. Nguyen, "Improved hand tracking system," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 5, pp. 693–701. IEEE Xplore, May 2012. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6093954

[11] D-U. Klaus and D. Schmalstieg, "Finger tracking for interaction in augumented environments," in proceedings of *IEEE and ACM International Symposium in Augmented Reality*, New York City, NY, pp. 55–64. IEEE Xplore, 2001. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=970515

[12] Y. Ma, Z. Mao, W. Jia, C. Li, J. Yang and M. Sun, "Magnetic hand tracking for human-computer interface," *IEEE Transactions on Magnetics*, vol. 47, no. 5, pp.

970–973. IEEE Xplore, May 2011. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5754773

[13] K. Li, I-M. Chen, S. H. Yeo and C. K. Lim, "Development of finger-motion capturing device based on optical linear encoder," *Journal of Rehabilitation Research and Development*, vol. 48, no. 1, pp. 69–82. Directory of Open Access Journals, 2011. http://www.rehab.research.va.gov/jour/11/481/pdf/li.pdf

[14] P. Gruber, *Biomimetics -- Materials, Structures and Processes*. Berlin: Springer Verlag, 2011. http://link.springer.com.lib-ezproxy.tamu.edu:2048/chapter/10.1007/978-3-642-11934-7_1

[15] A. von Gleich, C. Pade, U. Petschow, and E. Pissarskoi, *Potentials and Trends in Biomimetics*. Berlin: Springer, 2010. http://link.springer.com.lib-ezproxy.tamu.edu:2048/book/10.1007%2F978-3-642-05246-0

[16] S-y. Jung, S-k. Kang and I. Moon, "Design of biomimetic hand prosthesis with tendon-driven five fingers," in proceedings of *2nd IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics*, Scottsdale, AZ, 2008, pp. 895–900. IEEE Xplore, 2008. http://ieeexplore.ieee.org.lib-ezproxy.tamu.edu:2048/xpl/articleDetails.jsp?arnumber=4762803

[17] Thera Tech Equipment Inc., *XT DigiGlide Kaiser Portable Hand CPM*. 2014. http://theratechequip.com/products/xt-digiglide-kaiser-portable-hand-cpm/

[18] U. Jeong, H-K. In, and K-I. Cho, "Implementation of various control algorithms for hand rehabilitation exercise using wearable robotic hand," *Intelligent Service*

Robotics, vol. 6, no. 4, pp. 181–189. Berlin: Springer, Sept 2013. http://link.springer.com/article/10.1007/s11370-013-0135-5

[19] S. W. Lee, K. A. Landers and H-S. Park, "Development of a biomimetic hand exotendon device (BiomHED) for restoration of functional hand movement post-stroke," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 22, no. 4, pp. 886–898. IEEE Xplore, Jan 2014. http://ieeexplore.ieee.org.lib-ezproxy.tamu.edu:2048/xpl/articleDetails.jsp?arnumber=6710121

[20] Ardelis Engineering (Pty) Ltd, *3-Finger Robotic Hand on Comau Robot*. 2013. http://www.ardelis.co.za/products-grippers/

[21] M. Ferre, M. Buss, R. Aracil, C. Melchiorri, and C. Balaguer, *Advances in Telerobotics*. Berlin: Springer, 2007. http://link.springer.com.lib-ezproxy.tamu.edu:2048/book/10.1007%2F978-3-540-71364-7

[22] H. Karamanoukian, R. Pande, Y. Patel, A. Freeman, P. Aoukar and G. D'Ancona, "Telerobotics, telesurgery, and telemonitoring," *Pediatric Endosurgery & Innovative Techniques*, vol. 7, no. 4, pp. 421–425. Mary Ann Liebert, Inc., July 2004. http://online.liebertpub.com.lib-ezproxy.tamu.edu:2048/doi/abs/10.1089/109264103322614295

[23] F. Y. Wu and H. Asada, "Bio-artificial synergies for grasp posture control of supernumerary robotic fingers," *MIT Open Access Articles*. University of California, Berkeley, USA. MIT Press, July 2014. http://hdl.handle.net/1721.1/88457

[24] Robot Surgery: Dr. M. Hewitt, *Da Vinci robot and surgeon*. 2008. http://www.robotsurgery.ie/

[25] Teach Engineering: curriculum for k–12 teachers, *Crane Claw*. 2007. http://www.teachengineering.org/view_lesson.php?url=collection/van_/lessons/van_cleanupmess_less/van_cleanupmess_less1.xml

[26] Y. Lee and C. Cho, "A biomimetic hand employing a dual actuation scheme," *Journal of Mechanical Science and Technology*, vol. 26, no. 12, pp. 4131–4139. ProQuest, Dec 2012. http://search.proquest.com.lib-ezproxy.tamu.edu:2048/docview/1439740051?accountid=7082

[27] S. Ghosh, "Capturing human hand kinematics for object grasping and manipulation," Master's thesis, Texas A&M University, 2013. http://repository.tamu.edu/bitstream/handle/1969.1/149611/GHOSH-THESIS-2013.pdf?sequence=1.

[28] Radioshack, *RadioShack Standard Servo,* Date Accessed: 2. Sept. 2014. http://www.radioshack.com/product/index.jsp?productId=22472146#

[29] *Parallax Data Acquisition Tool (PLX-DAQ)*. Parallax Inc., 2014. http://www.parallax.com/downloads/plx-daq

[30] *Bourns 3382 - 12 mm Rotary Position Sensor,* 2nd ed. Bourns. Date Accessed: 2. Mar. 2014. http://www.bourns.com/pdfs/3382.pdf

[31] *Flex Sensor FS,* 2nd ed. Spectra Symbol, 2015. http://www.spectrasymbol.com/wpcontent/themes/spectra/images/datasheets/FlexSensor.pdf

[32] Ale., “Arduino and real time charts in excel,” Robottini RSS, 2011. http://robottini.altervista.org/arduino-and-real-time-charts-in-excel

[33] D. Mellis and T. Igoe, “Arduino – Smoothing,” Arduino, 2012. http://arduino.cc/en/Tutorial/Smoothing

APPENDIX A

BOURNS ROTARY POSITION SENSOR SPECIFICATIONS

Item #: 3382G

- *Features*
  - Surface mount and through-hole versions
  - 12mm Square/Dustproof
  - 1,000,000 rotation cycles
  - Thin profile
  - RoHS compliant

- *Electrical Characteristics*
  Standard resistance range: 2.5K to 100K ohms
  Resistance tolerance: ±30% std
  Linearity: ±2%
  Resolution: Infinite

- *Environmental Characteristics*
  Power rating: 16 volts max
  Operating temperature range: -40°C to 120°C
  Rotational life: 1,000,000 cycles
  Thermal Shock: 5 cycles

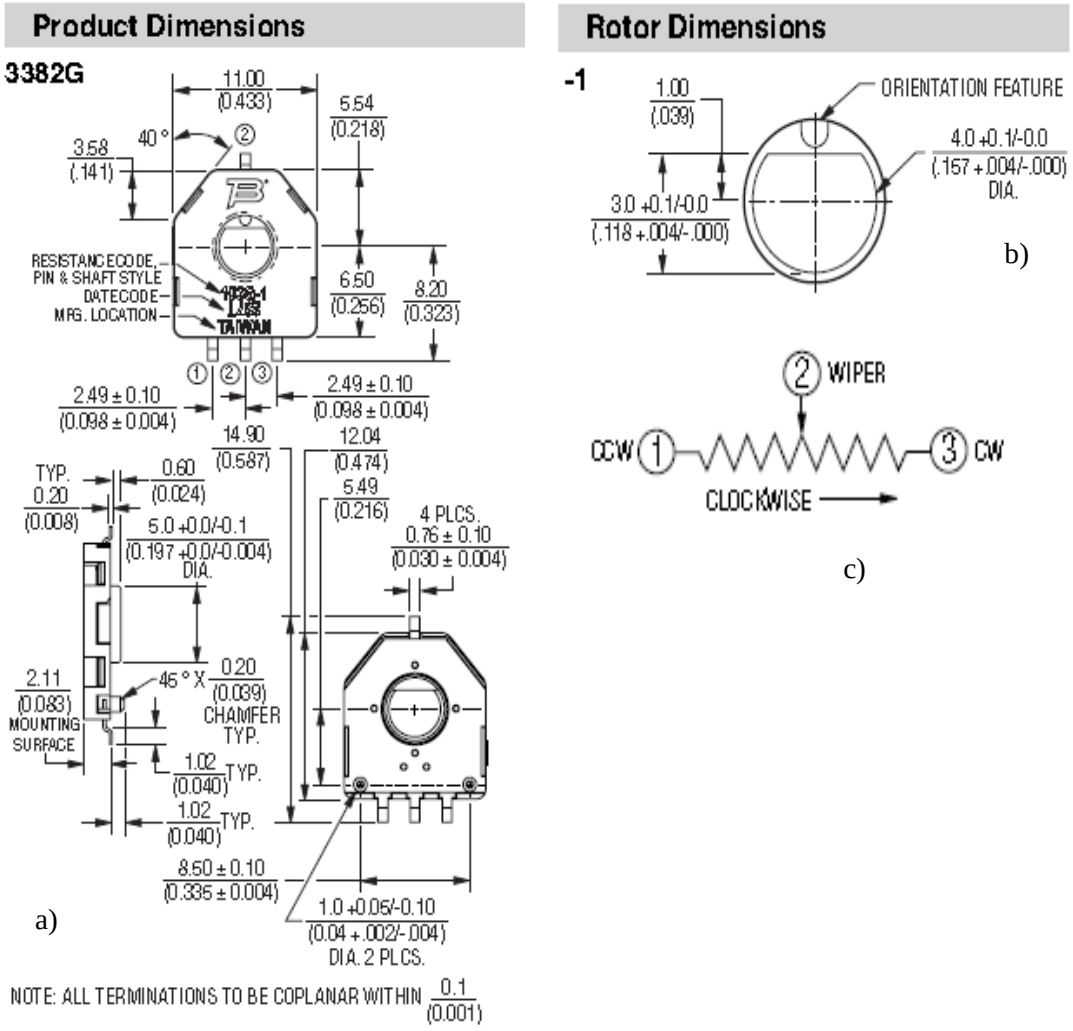


Figure Appendix A a) Dimensions of outer casing of rotary position sensor, b) rotation ring dimensions, and c) Operation diagram [30]

APPENDIX B

SPECTRA SYMBOL FLEX SENSOR SPECIFICATIONS

- *Features*
  - Angle displacement measurement
  - Flexible body for bending
  - Variety of applications
  - Simple design
  - Slim profile

- *Mechanical characteristics*
  Life cycle: over 1,000,000
  Temperature range: -35°C to +80°C

- *Electrical characteristics*
  Flat resistance: 25K Ohms
  Resistance tolerance: ±30%
  Bend resistance range: 45K Ohms to 125K Ohms
  Power rating: 0.5 Watts continuous to 1 Watt peak

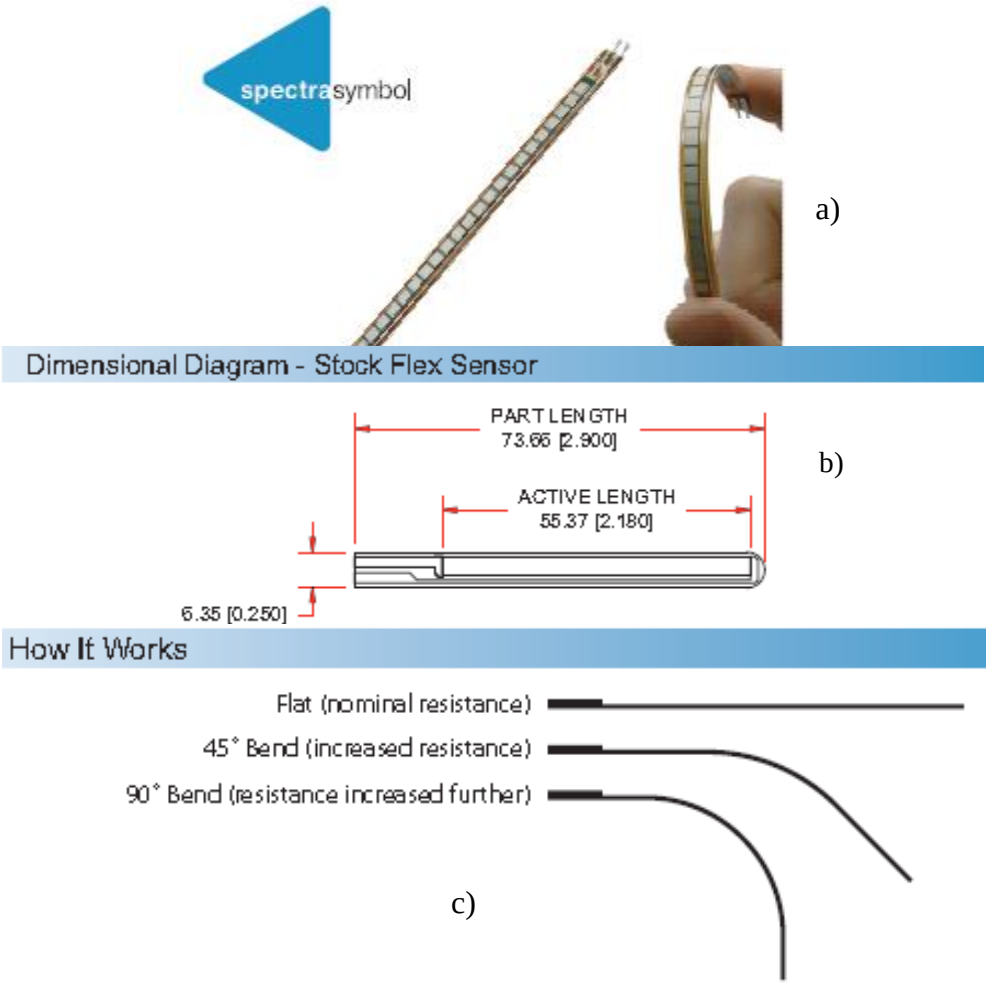
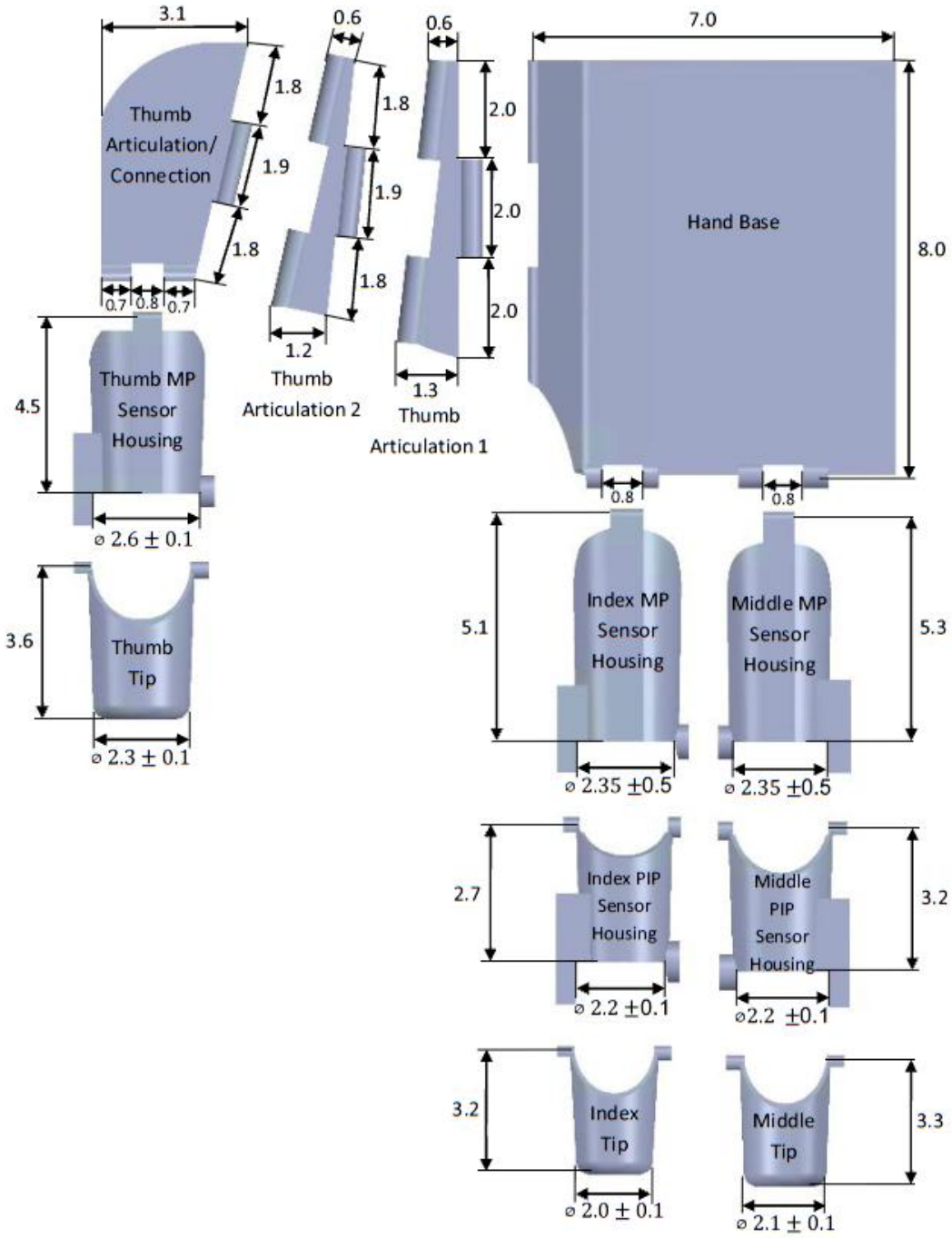


Figure Appendix B a) Example of an actual flex sensor, b) dimensional diagram, and c) flex sensor functionality [31]

APPENDIX C

RF EXO-GLOVE DIMENSIONS

Each dimension is given in centimeters. The dimensions alongside the finger and thumb components are given from pivot to pivot directly in the center were the sensors are attached. The diameter of each finger and thumb component is given for the center of the component. Since each finger and thumb component is tapered from the top to bottom the diameter changes along the length. Ex: Thumb Tip d = 2.3 ± 0.1, so the largest diameter which is the top is actually 2.4cm while the smallest diameter which is the bottom end is really 2.2cm.


3.1
Thumb
Articulation/
Connection
1.8
1.9
1.8
0.7
0.8
0.7
0.6
1.8
1.9
1.8
1.2
Thumb
Articulation 2
0.6
2.0
2.0
2.0
1.3
Thumb
Articulation 1
7.0
Hand Base
8.0
0.8
0.8
4.5
Thumb MP
Sensor
Housing
⌀ 2.6 ± 0.1
3.6
Thumb
Tip
⌀ 2.3 ± 0.1
5.1
Index MP
Sensor
Housing
⌀ 2.35 ±0.5
Middle MP
Sensor
Housing
5.3
⌀ 2.35 ±0.5
2.7
Index PIP
Sensor
Housing
⌀ 2.2 ±0.1
Middle
PIP
Sensor
Housing
3.2
⌀2.2 ±0.1
3.2
Index
Tip
⌀ 2.0 ± 0.1
Middle
Tip
3.3
⌀ 2.1 ± 0.1

APPENDIX D

ARDUINO CODES USED FOR DATA ACQUISITION

This is the original reference code from Robottini [32] used to create my own custom data acquisition code. This code was heavily modified for initial testing and servo control allowing me to use the PLX-DAQ software:

```
 int x = 0;

int row = 0;

void setup() {

  Serial.begin(128000); // opens serial port, sets data rate to 9600 bps

  Serial.println("CLEARDATA");

  Serial.println("LABEL,Time,x,sin(x)");

}


void loop() {

  Serial.print("DATA,TIME,"); Serial.print(x); Serial.print(",");
Serial.println(sin(x*PI/180));

  row++;

  x++;

  if (row > 360)

   {

    row=0;

    Serial.println("ROW,SET,2");

   }

  delay(100);
```

```
}
```

This is the original reference code from the Arduino website [33] I used to create my own custom code for averaging the sensors during initial testing:

```
/*

Smoothing

Reads repeatedly from an analog input, calculating a running average
and printing it to the computer. Keeps ten readings in an array and
continually averages them.

The circuit:
* Analog sensor (potentiometer will do) attached to analog input 0

Created 22 April 2007
By David A. Mellis <dam@mellis.org>
modified 9 Apr 2012
by Tom Igoe
http://www.arduino.cc/en/Tutorial/Smoothing

This example code is in the public domain.


*/


// Define the number of samples to keep track of. The higher the number,
// the more the readings will be smoothed, but the slower the output will
// respond to the input. Using a constant rather than a normal variable lets
// use this value to determine the size of the readings array.
const int numReadings = 10;

int readings[numReadings]; // the readings from the analog input
int index = 0; // the index of the current reading
int total = 0; // the running total
```

```
int average = 0; // the average

int inputPin = A0;

void setup()
{
// initialize serial communication with computer:
Serial.begin(9600);
// initialize all the readings to 0:
for (int thisReading = 0; thisReading < numReadings; thisReading++)
readings[thisReading] = 0;
}

void loop() {
// subtract the last reading:
total= total - readings[index];
// read from the sensor:
readings[index] = analogRead(inputPin);
// add the reading to the total:
total= total + readings[index];
// advance to the next position in the array:
index = index + 1;

// if we're at the end of the array...
if (index >= numReadings)
// ...wrap around to the beginning:
index = 0;

// calculate the average:
average = total / numReadings;
// send it to the computer as ASCII digits
Serial.println(average);
delay(1); // delay in between reads for stability
}
```

Custom code used for initial sensor testing without averaging:

```
const int flexpin = A0; // flexpin refers to the flex sensor on the index finger on analog pin 0

const int flexpin2 = A1; // flexpin2 refers to the flex sensor on the index finger on analog pin 1

const int flexpin3 = A2; // flexpin3 refers to the flex sensor on the middle finger on analog pin 2

const int potPin = A3; // potPin refers to the potentiometer on the thumb using analog pin 3

const int potPin2 = A4; // potPin2 refers to the potentiometer on the index finger center using analog pin 4

const int potPin3 = A5; // potPin3 refers to the potentiometer on the index finger tip using analog pin 5

const int potPin4 = A6; // potPin4 refers to the potentiometer on the middle finger center using analog pin 6

const int potPin5 = A7; // potPin5 refers to the potentiometer on the middle finger tip using analog pin 7


int flexPin = 0;

int flexPin2 = 0;

int flexPin3 = 0;

int potpin = 0;

int potpin2 = 0;

int potpin3 = 0;

int potpin4 = 0;

int potpin5 = 0;

int row = 0;

void setup() {
```

```
  Serial.begin(9600); // opens serial port, sets data rate to 9600 bps

  Serial.println("CLEARDATA");

Serial.println("LABEL,Time,flexPin,flexPin2,flexPin3,potpin,potpin2,potpin3,potpin4,p
otpin5,");

}


void loop() {

  flexPin = analogRead(flexpin);

flexPin2 = analogRead(flexpin2);

flexPin3 = analogRead(flexpin3);

potpin = analogRead(potPin);

potpin2 = analogRead(potPin2);

potpin3 = analogRead(potPin3);

potpin4 = analogRead(potPin4);

potpin5 = analogRead(potPin5);


  Serial.print("DATA,TIME,"); Serial.print(flexPin = analogRead(flexpin));
Serial.print(",");

  Serial.print(flexPin2 = analogRead(flexpin2)); Serial.print(",");

  Serial.print(flexPin3 = analogRead(flexpin3)); Serial.print(",");

  Serial.print(potpin = analogRead(potPin)); Serial.print(",");

  Serial.print(potpin2 = analogRead(potPin2)); Serial.print(",");

  Serial.print(potpin3 = analogRead(potPin3)); Serial.print(",");

  Serial.print(potpin4 = analogRead(potPin4)); Serial.print(",");
```

```
 Serial.println(potpin5 = analogRead(potPin5));
 row++;
 flexPin++;
 flexPin2++;
 flexPin3++;
 potpin++;
 potpin2++;
 potpin3++;
 potpin4++;
 potpin5++;

 if (row > 285)
  {
   row=0;
   Serial.println("ROW,SET,2");
  }
 delay(100);
}
```

Custom code used for initial sensor testing with averaging:

```
const int flexpin = 0; // flexpin refers to the flex sensor on the thumb on analog pin 1
const int flexpin2 = 1; // flexpin2 refers to the flex sensor on the index finger on analog pin 2
```

```
const int flexpin3 = 2; // flexpin3 refers to the flex sensor on the middle finger on analog pin 3

const int potPin = 3; // potPin refers to the potentiometer on the thumb using analog pin 4

const int potPin2 = 4; // potPin2 refers to the potentiometer on the index finger center using analog pin 5

const int potPin3 = 5; // potPin3 refers to the potentiometer on the index finger tip using analog pin 6

const int potPin4 = 6; // potPin4 refers to the potentiometer on the middle finger center using analog pin 7

const int potPin5 = 7; // potPin5 refers to the potentiometer on the middle finger tip using analog pin 8


int row = 0;


const int numReadings = 10;

const int numReadings2 = 10;

const int numReadings3 = 10;

const int numReadings4 = 10;

const int numReadings5 = 10;

const int numReadings6 = 10;

const int numReadings7 = 10;

const int numReadings8 = 10;


int readings[numReadings];

int readings2[numReadings2];
```

```
int readings3[numReadings3];
int readings4[numReadings4];
int readings5[numReadings5];
int readings6[numReadings6];
int readings7[numReadings7];
int readings8[numReadings8];

int index = 0;
int index2 = 0;
int index3 = 0;
int index4 = 0;
int index5 = 0;
int index6 = 0;
int index7 = 0;
int index8 = 0;

int total = 0;
int total2 = 0;
int total3 = 0;
int total4 = 0;
int total5 = 0;
int total6 = 0;
int total7 = 0;
int total8 = 0;
```

```
int avg = 0;

int avg2 = 0;

int avg3 = 0;

int avg4 = 0;

int avg5 = 0;

int avg6 = 0;

int avg7 = 0;

int avg8 = 0;


void setup()

{

Serial.begin(9600);

Serial.println("CLEARDATA");
Serial.println("LABEL,Time,flexpin,flexpin2,flexpin3,potPin,potPin2,potPin3,potPin4,p
otPin5,");


  // initialize all the readings to 0:

  for (int thisReading = 0; thisReading < numReadings; thisReading++)

    readings[thisReading] = 0;

     // initialize all the readings to 0:

  for (int thisReading2 = 0; thisReading2 < numReadings2; thisReading2++)

    readings[thisReading2] = 0;

     // initialize all the readings to 0:
```

```
  for (int thisReading3 = 0; thisReading3 < numReadings3; thisReading3++)
    readings[thisReading3] = 0;
     // initialize all the readings to 0:
  for (int thisReading4 = 0; thisReading4 < numReadings4; thisReading4++)
    readings[thisReading4] = 0;
     // initialize all the readings to 0:
  for (int thisReading5 = 0; thisReading5 < numReadings5; thisReading5++)
    readings[thisReading5] = 0;
     // initialize all the readings to 0:
  for (int thisReading6 = 0; thisReading6 < numReadings6; thisReading6++)
    readings[thisReading6] = 0;
     // initialize all the readings to 0:
  for (int thisReading7 = 0; thisReading7 < numReadings7; thisReading7++)
    readings[thisReading7] = 0;
      // initialize all the readings to 0:
  for (int thisReading8 = 0; thisReading8 < numReadings8; thisReading8++)
    readings[thisReading8] = 0;
}


void loop()
{

//1111111111111111111111111111111111111111111111111111111111111111111
111
```

```
  // subtract the last reading:
 total= total - readings[index];
 // read from the sensor:
 readings[index] = analogRead(flexpin);
 // add the reading to the total:
 total= total + readings[index];
 // advance to the next position in the array:
 index = index + 1;

 // if we're at the end of the array...
 if (index >= numReadings)
  // ...wrap around to the beginning:
  index = 0;

 // calculate the average:
 avg = total / numReadings;
 // send it to the computer as ASCII digits


//22222222222222222222222222222222222222222222222222222222222222222222
222
  // subtract the last reading:
 total2= total2 - readings2[index2];
 // read from the sensor:
```

```
 readings2[index2] = analogRead(flexpin2);
 // add the reading to the total:
 total2= total2 + readings2[index2];
 // advance to the next position in the array:
 index2 = index2 + 1;

 // if we're at the end of the array...
 if (index2 >= numReadings2)
   // ...wrap around to the beginning:
   index2 = 0;

 // calculate the average:
 avg2 = total2 / numReadings2;
 // send it to the computer as ASCII digits


//3333333333333333333333333333333333333333333333333333333333333333333333333333
333
    // subtract the last reading:
 total3= total3 - readings3[index3];
 // read from the sensor:
 readings3[index3] = analogRead(flexpin3);
 // add the reading to the total:
 total3= total3 + readings3[index3];
```

```
  // advance to the next position in the array:
  index3 = index3 + 1;

  // if we're at the end of the array...
  if (index3 >= numReadings3)
    // ...wrap around to the beginning:
    index3 = 0;

  // calculate the average:
  avg3 = total3 / numReadings3;
  // send it to the computer as ASCII digits


//4444444444444444444444444444444444444444444444444444444444444444444444444444
444
    // subtract the last reading:
  total4= total4 - readings4[index4];
  // read from the sensor:
  readings4[index4] = analogRead(potPin);
  // add the reading to the total:
  total4= total4 + readings4[index4];
  // advance to the next position in the array:
  index4 = index4 + 1;
```

```
 // if we're at the end of the array...
 if (index4 >= numReadings4)
   // ...wrap around to the beginning:
   index4 = 0;


 // calculate the average:
 avg4 = total4 / numReadings4;
 // send it to the computer as ASCII digits


//5555555555555555555555555555555555555555555555555555555555555555555555555555
555
    // subtract the last reading:
 total5= total5 - readings5[index5];
 // read from the sensor:
 readings5[index5] = analogRead(potPin2);
 // add the reading to the total:
 total5= total5 + readings5[index5];
 // advance to the next position in the array:
 index5 = index5 + 1;


 // if we're at the end of the array...
 if (index5 >= numReadings5)
   // ...wrap around to the beginning:
```

```
  index5 = 0;


 // calculate the average:
 avg5 = total5 / numReadings5;
 // send it to the computer as ASCII digits


//6666666666666666666666666666666666666666666666666666666666666666666666666666
666
   // subtract the last reading:
 total6= total6 - readings6[index6];
 // read from the sensor:
 readings6[index6] = analogRead(potPin3);
 // add the reading to the total:
 total6= total6 + readings6[index6];
 // advance to the next position in the array:
 index6 = index6 + 1;


 // if we're at the end of the array...
 if (index6 >= numReadings6)
   // ...wrap around to the beginning:
   index6 = 0;


 // calculate the average:
```

```
 avg6 = total6 / numReadings6;
 // send it to the computer as ASCII digits


//77777777777777777777777777777777777777777777777777777777777777777777777
777
   // subtract the last reading:
 total7= total7 - readings7[index7];
 // read from the sensor:
 readings7[index7] = analogRead(potPin4);
 // add the reading to the total:
 total7= total7 + readings7[index7];
 // advance to the next position in the array:
 index7 = index7 + 1;

 // if we're at the end of the array...
 if (index7 >= numReadings7)
   // ...wrap around to the beginning:
   index7 = 0;

 // calculate the average:
 avg7 = total7 / numReadings7;
 // send it to the computer as ASCII digits
```

```
///888888888888888888888888888888888888888888888888888888888888888888888888
8888
   // subtract the last reading:
  total8= total8 - readings8[index8];
  // read from the sensor:
  readings8[index8] = analogRead(potPin5);
  // add the reading to the total:
  total8= total8 + readings8[index8];
  // advance to the next position in the array:
  index8 = index8 + 1;

  // if we're at the end of the array...
  if (index8 >= numReadings8)
    // ...wrap around to the beginning:
    index8 = 0;

  // calculate the average:
  avg8 = total8 / numReadings8;
  // send it to the computer as ASCII digits

  Serial.print("DATA,TIME,");
  Serial.print(avg); Serial.print(",");
  Serial.print(avg2); Serial.print(",");
```

```
  Serial.print(avg3); Serial.print(",");
  Serial.print(avg4); Serial.print(",");
  Serial.print(avg5); Serial.print(",");
  Serial.print(avg6); Serial.print(",");
  Serial.print(avg7); Serial.print(",");
  Serial.println(avg8);

delay(100); // wait 100ms between servo updates
}
```

Custom code used for postures without averaging and includes servo motor control:

```
#include <Servo.h>
Servo servo1; // servo1 is the first robot finger
Servo servo2; // servo2 is the second robot finger
Servo servo3; // servo3 is the third robot finger

const int flexpin = A0; // flexpin refers to the flex sensor on the index finger on analog pin 0
const int flexpin2 = A1; // flexpin2 refers to the flex sensor on the index finger on analog pin 1
const int flexpin3 = A2; // flexpin3 refers to the flex sensor on the middle finger on analog pin 2
const int potPin = A3; // potPin refers to the potentiometer on the thumb using analog pin 3
```

```
const int potPin2 = A4; // potPin2 refers to the potentiometer on the index finger center using analog pin 4

const int potPin3 = A5; // potPin3 refers to the potentiometer on the index finger tip using analog pin 5

const int potPin4 = A6; // potPin4 refers to the potentiometer on the middle finger center using analog pin 6

const int potPin5 = A7; // potPin5 refers to the potentiometer on the middle finger tip using analog pin 7


    int flexposition = 0; // Input value from the analog pin for the flexsensor on the thumb

    int flexposition2 = 0; // Input value from the analog pin for the flexsensor on the index finger

    int flexposition3 = 0; // Input value from the analog pin for the flexsensor on the middle finger

    int potposition = 0; // Input value from the analog pin for the potentiometer on the thumb

    int potposition2 = 0; // Input value from the analog pin for the potentiometer on the index finger center

    int potposition3 = 0; // Input value from the analog pin for the potentiometer on the index finger tip

    int potposition4 = 0; // Input value from the analog pin for the potentiometer on the middle finger center

    int potposition5 = 0; // Input value from the analog pin for the potentiometer on the middle finger tip

   int row = 0;

  int servoposition1; // Output value for servo1

  int servoposition2; // Output value for servo2
```

```
      int servoposition3; // Output value for servo3
      int average; // average of the thumb sensors
      int average2; // average of the index finger sensors
      int average3; // average of the middle finger sensors


void setup()
{
          Serial.begin(9600); // sets the baud rate at 9600
       Serial.println("CLEARDATA");

Serial.println("LABEL,Time,flexposition,flexposition2,flexposition3,potposition,potposition2,potposition3,potposition4,potposition5");


servo1.attach(9);// servo1 will be connected to output pin 9 on the PWM of the Arduino Mega
servo2.attach(8);// servo2 will be connected to output pin 8 on the PWM of the Arduino Mega
servo3.attach(7);// servo3 will be connected to output pin 7 on the PWM of the Arduino Mega
}


void loop()
{
       int row;
          int flexposition; // Input value from the analog pin for the flexsensor on the thumb
```

```
        int flexposition2; // Input value from the analog pin for the flexsensor on the
index finger

        int flexposition3; // Input value from the analog pin for the flexsensor on the
middle finger

        int potposition; // Input value from the analog pin for the potentiometer on the
thumb

        int potposition2; // Input value from the analog pin for the potentiometer on the
index finger center

        int potposition3; // Input value from the analog pin for the potentiometer on the
index finger tip

        int potposition4; // Input value from the analog pin for the potentiometer on the
middle finger center

        int potposition5; // Input value from the analog pin for the potentiometer on the
middle finger tip

    int servoposition1; // Output value for servo1

    int servoposition2; // Output value for servo2

    int servoposition3; // Output value for servo3

    int average; // average of the thumb sensors

    int average2; // average of the index finger sensors

    int average3; // average of the middle finger sensors


// Read the position of the flex sensor (0 to 1023).  I have limited the values to make it
more responsive.

            flexposition = analogRead(flexpin);

            flexposition2 = analogRead(flexpin2);

            flexposition3 = analogRead(flexpin3);

            potposition = analogRead(potPin);
```

```
potposition2 = analogRead(potPin2);

potposition3 = analogRead(potPin3);

potposition4 = analogRead(potPin4);

potposition5 = analogRead(potPin5);


average = (flexposition3 + potposition5)/2;

average2 = (flexposition2 + potposition3 + potposition4)/3;

average3 = (flexposition + potposition + potposition2)/3;


servoposition1 = map(average, 768, 797, 180, 0); // This maps the rotary position sensor averages onto the robotic finger

servoposition2 = map(average2, 648, 446, 180, 0); // This maps the rotary position sensor averages onto the robotic finger

servoposition3 = map(average3, 664, 761, 180, 0); // This maps the rotary position sensor averages onto the robotic finger

servoposition1 = constrain(servoposition1, 0, 180); // This maps the rotary position sensor averages onto the robotic finger

servoposition2 = constrain(servoposition2, 0, 180); // This maps the rotary position sensor averages onto the robotic finger

servoposition3 = constrain(servoposition3, 0, 180); // This maps the rotary position sensor averages onto the robotic finger


//command the servo to move to the appropriate position:

servo1.write(servoposition1); // This will move the robotic hand to mimic the exo-glove finger movement

servo2.write(servoposition2); // This will move the robotic hand to mimic the exo-glove finger movement
```

servo3.write(servoposition3); // This will move the robotic hand to mimic the exo-glove finger movement

```
Serial.print("DATA,TIME,");
Serial.print(flexposition); Serial.print(",");
Serial.print(flexposition2); Serial.print(",");
Serial.print(flexposition3); Serial.print(",");
Serial.print(potposition); Serial.print(",");
Serial.print(potposition2); Serial.print(",");
Serial.print(potposition3); Serial.print(",");
Serial.print(potposition4); Serial.print(",");
Serial.println(potposition5);

row++;
flexposition++;
flexposition2++;
flexposition3++;
potposition++;
potposition2++;
potposition3++;
potposition4++;
potposition5++;
average++;
average2++;
```

```
      average3++;


      if (row > 285) // resets the excel data once it has reached 285 rows of acquired data
      {
        row=0;
        Serial.println("ROW,SET,2");
      }
  delay(100); // wait 100ms between servo updates
}
```

Custom code used for postures with averaging and servo control:

```
#include <Servo.h>
Servo servo1; // servo1 is the first robot finger
Servo servo2; // servo2 is the second robot finger
Servo servo3; // servo3 is the third robot finger

int flexpin1 = A0;
int flexpin2 = A1;
int flexpin3 = A2;
int potpin1 = A3;
int potpin2 = A4;
int potpin3 = A5;
int potpin4 = A6;
int potpin5 = A7;
```

```
int flexPin11;
int flexPin12;
int flexPin13;
int flexPin14;
int flexPin15;

int flexPin21;
int flexPin22;
int flexPin23;
int flexPin24;
int flexPin25;

int flexPin31;
int flexPin32;
int flexPin33;
int flexPin34;
int flexPin35;

int potPin11;
int potPin12;
int potPin13;
int potPin14;
int potPin15;
```

```
int potPin21;
int potPin22;
int potPin23;
int potPin24;
int potPin25;

int potPin31;
int potPin32;
int potPin33;
int potPin34;
int potPin35;

int potPin41;
int potPin42;
int potPin43;
int potPin44;
int potPin45;

int potPin51;
int potPin52;
int potPin53;
int potPin54;
int potPin55;
```

```
int average1;
int average2;
int average3;
int average4;
int average5;
int average6;
int average7;
int average8;

      int servoposition1; // Output value for servo1
      int servoposition2; // Output value for servo2
      int servoposition3; // Output value for servo3
      int average10; // average of the thumb sensors
      int average11; // average of the index finger sensors
      int average12; // average of the middle finger sensors
int row = 0;

void setup()
{
  Serial.begin(9600);
  Serial.println("CLEARDATA"); // this clears the rows and columns of previous data to start with new values
```

```
Serial.println("LABEL,Time,flexpin1,flexpin2,flexpin3,potpin1,potpin2,potpin3,potpin4
,potpin5,"); // LABEL is used to setup the columns with the following headings such as
Time,....


servo1.attach(9);// servo1 will be connected to output pin 9 on the PWM of the Arduino
Mega

servo2.attach(8);// servo2 will be connected to output pin 8 on the PWM of the Arduino
Mega

servo3.attach(7);// servo3 will be connected to output pin 7 on the PWM of the Arduino
Mega

}


void loop()

{


//11111111111111111111111111111111111111111111111111111111111111111111
111

  flexPin11 = analogRead(A0);

  delay(10);

  flexPin12 = analogRead(A0);

  delay(10);

  flexPin13 = analogRead(A0);

  delay(10);

  flexPin14 = analogRead(A0);

  delay(10);

  flexPin15 = analogRead(A0);
```

```
  average1 = (flexPin11 + flexPin12 + flexPin13 + flexPin14 + flexPin15)/5;


//22222222222222222222222222222222222222222222222222222222222222222222222222222
222
  flexPin21 = analogRead(A1);
  delay(10);
  flexPin22 = analogRead(A1);
  delay(10);
  flexPin23 = analogRead(A1);
  delay(10);
  flexPin24 = analogRead(A1);
  delay(10);
  flexPin25 = analogRead(A1);

  average2 = (flexPin21 + flexPin22 + flexPin23 + flexPin24 + flexPin25)/5;


//33333333333333333333333333333333333333333333333333333333333333333333333333333
333
  flexPin31 = analogRead(A2);
  delay(10);
  flexPin32 = analogRead(A2);
  delay(10);
```

```
  flexPin33 = analogRead(A2);
  delay(10);
  flexPin34 = analogRead(A2);
  delay(10);
  flexPin35 = analogRead(A2);

  average3 = (flexPin31 + flexPin32 + flexPin33 + flexPin34 + flexPin35)/5;


//4444444444444444444444444444444444444444444444444444444444444444444444444444
444
  potPin11 = analogRead(A3);
  delay(10);
  potPin12 = analogRead(A3);
  delay(10);
  potPin13 = analogRead(A3);
  delay(10);
  potPin14 = analogRead(A3);
  delay(10);
  potPin15 = analogRead(A3);

  average4 = (potPin11 + potPin12 + potPin13 + potPin14 + potPin15)/5;
```

```
//55555555555555555555555555555555555555555555555555555555555555555555555555555
555

  potPin21 = analogRead(A4);

  delay(10);

  potPin22 = analogRead(A4);

  delay(10);

  potPin23 = analogRead(A4);

  delay(10);

  potPin24 = analogRead(A4);

  delay(10);

  potPin25 = analogRead(A4);


  average5 = (potPin21 + potPin22 + potPin23 + potPin24 + potPin25)/5;


//66666666666666666666666666666666666666666666666666666666666666666666666666666
666

  potPin31 = analogRead(A5);

  delay(10);

  potPin32 = analogRead(A5);

  delay(10);

  potPin33 = analogRead(A5);

  delay(10);

  potPin34 = analogRead(A5);
```

```
  delay(10);
  potPin35 = analogRead(A5);

  average6 = (potPin31 + potPin32 + potPin33 + potPin34 + potPin35)/5;

//7777777777777777777777777777777777777777777777777777777777777777777777
777
  potPin41 = analogRead(A6);
  delay(10);
  potPin42 = analogRead(A6);
  delay(10);
  potPin43 = analogRead(A6);
  delay(10);
  potPin44 = analogRead(A6);
  delay(10);
  potPin45 = analogRead(A6);

  average7 = (potPin41 + potPin42 + potPin43 + potPin44 + potPin45)/5;

//8888888888888888888888888888888888888888888888888888888888888888888888
888
  potPin51 = analogRead(A7);
  delay(10);
```

```
potPin52 = analogRead(A7);
delay(10);
potPin53 = analogRead(A7);
delay(10);
potPin54 = analogRead(A7);
delay(10);
potPin55 = analogRead(A7);

average8 = (potPin51 + potPin52 + potPin53 + potPin54 + potPin55)/5;

        average10 = (average3 + average8)/2; // thumb averaging
        average11 = (average2 + average6 + average7)/3; // index finger averaging
        average12 = (average1 + average4 + average5)/3; // middle finger averaging

          servoposition1 = map(average10, 768, 797, 180, 0); // This maps the rotary position sensor averages onto the robotic finger
          servoposition2 = map(average11, 648, 446, 180, 0); // This maps the rotary position sensor averages onto the robotic finger
          servoposition3 = map(average12, 664, 762, 180, 0); // This maps the rotary position sensor averages onto the robotic finger
          servoposition1 = constrain(servoposition1, 0, 180); // This maps the rotary position sensor averages onto the robotic finger
          servoposition2 = constrain(servoposition2, 0, 180); // This maps the rotary position sensor averages onto the robotic finger
          servoposition3 = constrain(servoposition3, 0, 180); // This maps the rotary position sensor averages onto the robotic finger
```

```
//command the servo to move to the appropriate position:

        servo1.write(servoposition1); // This will move the robotic hand to mimic the exo-glove finger movement

        servo2.write(servoposition2); // This will move the robotic hand to mimic the exo-glove finger movement

        servo3.write(servoposition3); // This will move the robotic hand to mimic the exo-glove finger movement


  Serial.print("DATA,TIME,");

  Serial.print(average1); Serial.print(",");

  Serial.print(average2); Serial.print(",");

  Serial.print(average3); Serial.print(",");

  Serial.print(average4); Serial.print(",");

  Serial.print(average5); Serial.print(",");

  Serial.print(average6); Serial.print(",");

  Serial.print(average7); Serial.print(",");

  Serial.println(average8);


   row++;

  flexpin1++;

  flexpin2++;

  flexpin3++;

  potpin1++;

  potpin2++;
```

```
  potpin3++;
  potpin4++;
  potpin5++;

  if (row > 285)
   {
    row=0;
    Serial.println("ROW,SET,2");
   }
  delay(100);
}
```