A Novel Exoskeleton Prototype Based on the Use of IMUs to Track and Mimic Motion

by

Harry Paul Draaisma

AN UNDERGRAD THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF APPLIED SCIENCES
in the School
of
Engineering Science

SIMON FRASER UNIVERSITY
February 5, 2021

APPROVAL

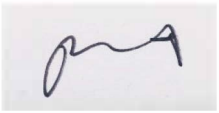
Name: Harry Draaisma

Degree: Bachelor of Applied Science (Honours)

Title of Thesis: A Novel Exoskeleton Prototype Based on the Use of IMUs to Track and Mimic Motion

Dr. Glenn Chapman, P.Eng
Director, School of Engineering Science

Examining Committee:



Dr. Andrew Rawicz, P.Eng (Supervisor and Chair)
Professor, School of Engineering Science

Dr. Ash Parameswaran, P.Eng
Professor, School of Engineering Science

Dr. Shahram Payandeh, P.Eng
Professor, School of Engineering Science

Date Approved July 18, 2021

Abstract

This thesis presents the development of a person-portable exoskeleton prototype which is designed to be controlled with Inertial Measurement Units (IMUs). It utilizes Euler angles calculated by the IMUs to track the rotation of the user's forearm and then performs the same rotation, mimicking the user. Special care is taken with the prototype's control algorithm to ignore changes in Euler angles caused by non-forearm rotations, which can otherwise cause erroneous prototype movements. The prototype is successful in demonstrating this method of control but does require the user to follow some specific guidelines to work at maximum effectiveness. Future iterations of the prototype can be easily improved by replacing some of the commercially available materials with more specialized ad-hoc products.

Keywords: Inertial Measurement Unit; IMU; Exoskeleton; Euler Angle; Control Methodology

Dedication

This thesis is dedicated to my parents, Harry and Maureen Draaisma. Their love, support, and patience were instrumental in helping me not only through this thesis, but my entire undergraduate career. Mom, Dad, I love you both so much, and I hope one day I can repay even a fraction of the kindness you have given to me.

Acknowledgements

I would like to express my thanks to both Dr. Andrew Rawicz and Dr. Ash Parameswaran, my academic supervisor and technical supervisor, respectively, for the time, effort, and constructive feedback given on my work. As both committee members and as professors during my undergraduate career, your guidance and assistance were invaluable to me getting to where I am now.

I would like to thank Dr. Shahram Payandeh for coming onto the thesis as a committee member with such short notice.

I would like to thank Dr. Carlo Menon, whose special topics course served as the inspiration for this thesis.

I would also like to thank Fred Heep and Ian McGregor for providing tools, equipment, and materials so I could work on the project from home to minimize my exposure to COVID-19.

I respectfully acknowledge the xʷməθkʷəy̓əm (Musqueam), Sḵwx̱wú7mesh Úxwumixw (Squamish), səl̓ilw̓ətaʔɬ (Tsleil-Waututh), q̓íc̓əy̓ (Katzie), kʷikʷəƛ̓əm (Kwikwetlem), Qayqayt, Kwantlen, Semiahmoo and Tsawwassen peoples on whose traditional territories the three SFU campuses reside.

Statement of Contribution

The general premise, design, overall code, and construction of the prototype described within this thesis are solely the work of the author. Code libraries used in the final code were provided by Mike McCauley of AirSpayce, AdaFruit Industries, and SparkFun Electronics. Pre-built sensors and microcontroller used in the construction of the prototype were provided by SparkFun Electronics, AdaFruit Industries, and Arduino.

Table of Contents

List of Figures

List of Acronyms

cm	Centimeters
DOF	Degree(s) of Freedom
g	Grams
IMU	Inertial Measurement Unit
I/O	Input/Output
mA	Milliamps
MEMS	Microelectromechanical Systems
SFU	Simon Fraser University
V	Volts

Glossary

Abduct	To move a limb away from the midline of the body
Accelerometer	A sensor that measures linear acceleration
Adduct	To move a limb towards the midline of the body
Axis of Rotation	A straight line through fixed points of a rigid body around which all other points of the body move in circles
Degree(s) of Freedom	The number of independent parameters needed to define a system's configuration
Electromyography	A technique of reading the electrical potential generated by muscle activation
Exoskeleton	A robotic assisting device with a wearable interface for rehabilitative or power amplification purposes
Gyroscope	A sensor that measures angular velocity
Magnetometer	A sensor that measures the intensity of a magnetic field or magnetic dipole
Overhead	A combination of excess computation time or computational resources required for a computer to perform a specific task
Pitch	The lateral axis that runs through a rigid body's center of gravity orthogonal to Yaw and Roll, one of the principal axes
Pronated	Having one's hand rotated so that the palm faces the opposite direction as the interior angle of the elbow
Roll	The longitudinal axis that runs through a rigid body's center of gravity orthogonal to Yaw and Pitch, one of the principal axes
Supinated	Having one's hand rotated so that the palm faces the same direction as the interior angle of the elbow
Yaw	The vertical axis that runs through a rigid body's center of gravity orthogonal to Roll and Pitch, one of the principal axes

Chapter 1.

Introduction

Over the last 30 years, development and research into devices designed to assist human movement have increased significantly [1]. Chief among this research is the creation and testing of powered exoskeletons, as they have shown significant promise as not only human power amplifiers [2] but also as rehabilitative tools [3]. Depending on the application, different control methodologies are used to control the exoskeleton. Rehabilitative exoskeletons mostly use impedance control to either resist or assist the motion of the limb they are donned upon, but occasionally also follow pre-programmed or fixed motions depending on the therapy and device used [4]. In contrast, amplifying-type exoskeletons detect the user's movement intentions and then moves in tandem with them or provides a proportional response based on the user's input. This movement detection is a critical factor for these exoskeletons, so they do not interfere with the natural movement of the limb or limbs they are attached to. Movement detection is most often acquired with direct torque/force measurements of the relevant limb or limbs [5]-[6] but research has been conducted into using alternative methods, such as using electromyography to detect muscle activity [7].

These alternative methods of detecting movement allow for more freedom in how the exoskeletons can be designed, such as allowing for teleoperation or allowing the exoskeleton manipulators to be free floating and not directly attached to the user. One such alternative method of movement detection that is has shown promise is the use of Inertial Measurement Units (IMUs). IMUs are devices combining accelerometers, gyroscopes, magnetometers and occasionally data fusion software into a single integrated package. Originally quite large and expensive devices, advances in microelectromechanical systems (MEMS) sensor technology have significantly reduced their size and cost, making them extremely portable and purchasable to the general population. They have been shown to be a convenient and cost-effective way to track an object's motion and position compared to more costly and cumbersome optical-based methods [8]. IMUs have also been successfully used to teleoperate a stationary robotic manipulator [9] and integrated into a person-portable exoskeleton prototype to assist its

other systems [10]. However, despite these demonstrations, they have never been used as the sole method of controlling an exoskeleton.

In this thesis, we introduce a person-portable exoskeleton prototype that focuses on showcasing using IMUs as its control methodology and describe its design and capabilities. This prototype would be controlled through monitoring and reacting to changes in an IMU's angular position and not through direct force measurements or extracting the movement from velocity/acceleration data. While this prototype's performance lacks power amplification or rehabilitative ability, it succeeds in demonstrating this control methodology and its potential.

Chapter 2.

Prototype Design

2.1. Hardware

The initial concept of a device to demonstrate IMUs as a control methodology was rather straightforward. An IMU would be attached to the end of a limb, and as the IMU and limb were moved, the device would mimic this movement based on the readings from the IMU. The prototype realizes this by attaching the IMU to the user's hand to track the motion of the forearm about the elbow. This motion was chosen to minimize the complexity of the prototype; as it is limited to one degree of freedom (DOF) and thus can be replicated using a single source of motion, as opposed to other joints that would require multiple sources. The overall hardware design of the prototype remained very consistent with this initial concept; and as such, most of this stage of the design was simply acquiring the necessary materials to realize it. Following that, the only iteration from the original concept was determining how to make all the hardware person portable.

The prototype, shown in Figure 1, accomplishes this by utilizing a custom-made 3D printed ABS plastic frame (which is worn on the upper arm) and a glove. Three belts with ladder locks are used to secure the frame to the upper arm such that the prototype's actuator (a 1.8° step 12 V stepper motor) sits parallel to the elbow joint (see Figure 2). The frame was designed to accommodate this positioning first and foremost, and then simply made large enough to hold most of the other hardware comfortably. An Arduino Uno microcontroller board running its native software is used to control the prototype. The microcontroller is connected to three devices: an Adafruit BNO055 Absolute Orientation Sensor attached to the glove, a Sparkfun ICM-20948 9 DOF IMU attached to the frame, and a Sparkfun Big Easy Driver motor driver, which controls the motor. The prototype is powered by a 12 V battery pack, which provides power to the driver, which distributes power to the other components.

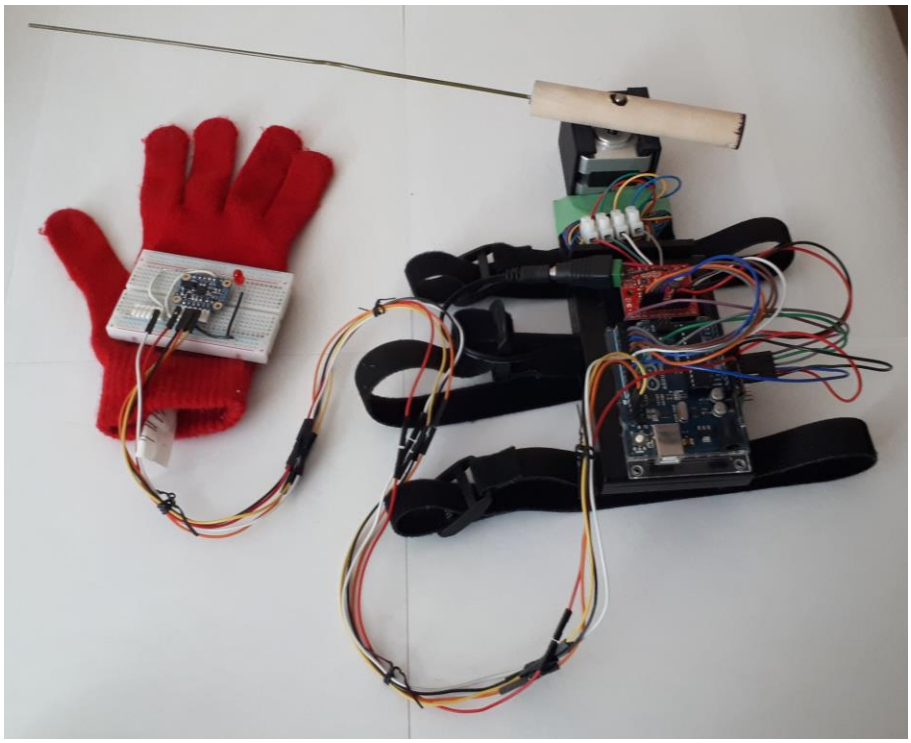


Figure 1: The prototype, with the BNO055 and glove on the left and the frame with the Arduino Uno, ICM-20948, and motor driver and motor attached to it on the right.

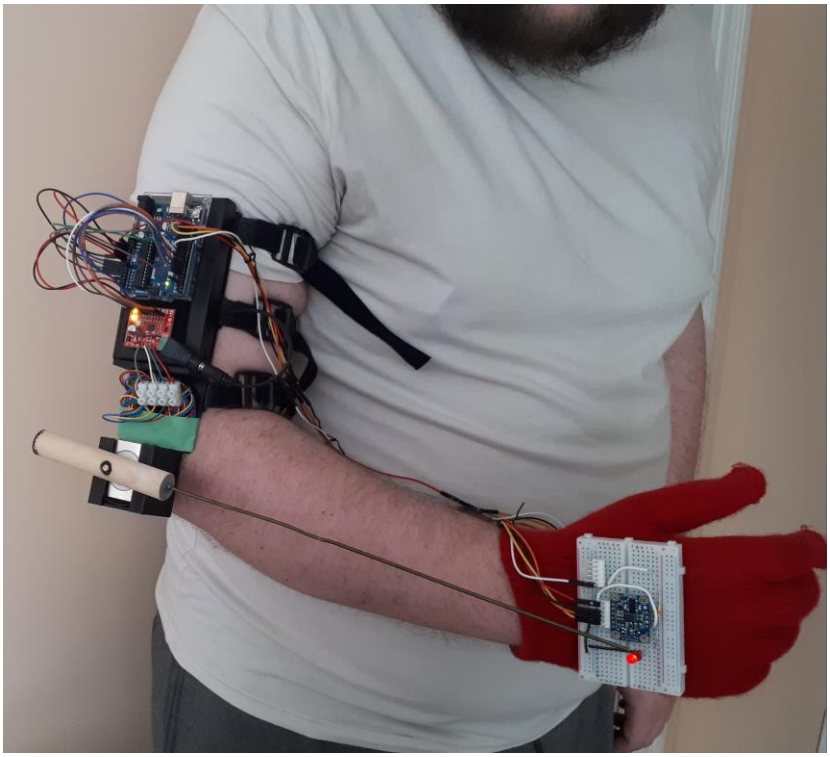


Figure 2: The active prototype being worn by a user, in which the prototype arm has rotated to align itself with the user's forearm.

2.1.1. Hardware Selection

An underlying requirement for most of the materials used in the prototype's construction was that they had to be commercially available products. This was necessary for two reasons; the first was for convenience, as it ensured that every part could be quickly acquired through local vendors. The second reason was for redundancy. If any part of the prototype became damaged or malfunctioned; the offending part could simply be replaced without affecting any other part of the prototype. The only exception to this rule is the prototype's frame, for which no commercially available equivalent exists. As such, it was necessary to have the frame be custom made.

In order to move the prototype's arm in the same manner as the forearm, an actuation system that can rotate the arm in a similar way would be necessary. While any DC motor could perform this rotation (if it were attached at the end of the prototype arm), only a stepper motor could provide precise control over the rotation while also preventing the arm from rotating freely whenever it is at rest. Of the potential stepper motors that could be used in this project, a Mercury Motor SM-42BYG011-25 was chosen as the prototype's actuator. This stepper motor was chosen as its size and weight (occupying approximately 60 cm^3 and weighing about 200 g) ensured it would not be too heavy or cumbersome to be supported by a single limb while providing enough torque to manipulate the prototype arm. As well, the motor's operating voltage of 12V was more convenient than other motor's operating voltages, which were either too high to be powered by commercially available battery packs or were at lower voltages and thus would not be able to easily interface with other components. With the motor chosen, a motor driver was also necessary to act as an interface between the motor and prototype's controller. The Big Easy Driver was chosen over other drivers for having the capability to provide power to both the motor and the microcontroller, despite them operating at different voltage levels. This feature eliminated the need to have two separate power sources, greatly simplifying the prototype's design.

Due to the prototype needing to work in-sync with the user's movements, a microcontroller with as little overhead as possible was a necessity. Otherwise, the overhead would propagate timing delays across the entire prototype; potentially causing it to consistently lag behind the user's movements. This issue would make the prototype much harder to control and less intuitive to use. Although there are many microcontroller

boards that have such low overhead, the Arduino Uno was ultimately chosen to be used as the prototype's main controller. This was due to two features that were unique to Uno. The first being its larger array of I/O pins, which not only allowed for multiple sensors and components to be connected to a single microcontroller; but also provided extra power pathways to those components, simplifying the overall design of the prototype. The second feature is the ubiquitous nature of Arduino products. Due to their prevalence, most if not all commercially available sensor and component boards are designed to easily interface with Arduino boards. Furthermore, these products also often offer support materials to aid in the interfacing process. This support could range anywhere from simply providing the code that initializes the sensor to the Arduino, all the way to fully detailed initialization and troubleshooting guides with detailed example projects. Thus, Arduino boards are much more convenient to work with compared to other boards that would not have these considerations and/or levels of support.

Since the prototype's whole concept would rely upon an IMU, the choice of which sensor board to use to track the forearm's motion was one of the more critical design decisions that had to be made. The BNO055 was eventually chosen to act as this tracking sensor due to it having built-in data fusion software, which simplified the software design (this will be elaborated on in the next section of this thesis). Originally, the BNO055 was the only IMU being used in the prototype; but during testing it became clear that movement data from an IMU placed elsewhere on the prototype would be needed to reduce potential errors. The ICM-20948, which was being considered as the forearm tracking sensor before the BNO055 was chosen, could provide this data and thus was selected to perform that function. Although it was implemented after the prototype was already built, the extra I/O provided by the Arduino Uno allowed it to be implemented seamlessly, without requiring any changes to prototype's design.

2.2. Software

2.2.1. The Role of the BNO055 and Euler Angles

The BNO055 acts as the linchpin of the prototype's control algorithm due to its built-in data fusion software, which is absent in the ICM-20948. This software allows the BNO055 to determine its orientation from the data acquired by its sensors. Specifically, the BNO055 can calculate what its documentation refers to as its "X, Y, and Z Euler

angles”. These correspond to the BNO055’s orientation with respect to its yaw, roll, and pitch axes respectively [11], which are determined when the BNO055 is turned on (i.e., at power-on all the Euler angles will read zero). These angles are measured from the BNO055’s frame of reference, which has the effect of making a rotation about one of its principal axes and the same rotation about an axis parallel to its principal axes indistinguishable to itself (see Figure 3).

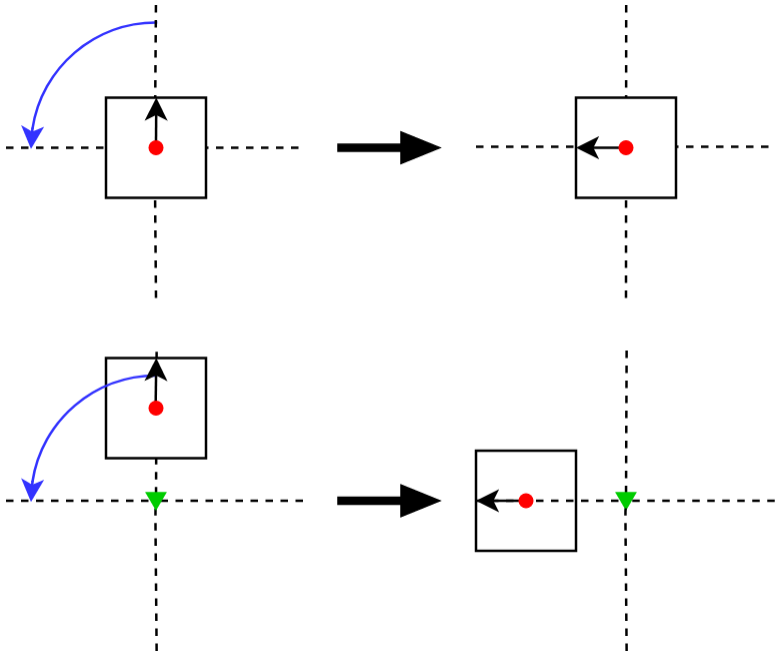


Figure 3: Rotation about a principal axis (red circle) and a parallel axis (green triangle). While the end position is different between rotations, from the body's perspective, its end orientation is the same.

Therefore, if the BNO055 is affixed to the end of an object whose axis of rotation is parallel to a principal axis, the rotation of that object can be tracked using the relevant Euler angle. The elbow, being a joint with only 1 DOF, allows the forearm to rotate about one axis. The BNO055’s yaw axis can be oriented parallel to the elbow by affixing it to the hand or wrist. As the forearm rotates, the BNO055’s orientation about its yaw axis changes. Therefore, the rotation of the forearm can be tracked by tracking the change in the X Euler angle value.

2.2.2. The Prototype Control Algorithm in Detail

Once the prototype is donned but before turning it on, the user must align their forearm with the arm of the prototype, as this alignment is required for the algorithm to work properly. Once the prototype is powered on, a moment is taken by the programming to initialize the IMUs so they can communicate with the microcontroller. During this initialization, the X Euler angle is set to zero as well as setting the stepper motor to half step mode (so the motor moves approximately 1° per step). This limits how fast the motor is able to rotate, but the smaller step size allows for greater precision. Once this initialization is complete, the algorithm activates. The core of the algorithm is rather simple in its operation: it continuously retrieves the X Euler angle reading of the BNO055 and then sets this value as the target position. It then checks if this target position is different from the previously read target position (i.e., its current position); and if it is, finds the integer difference between the two, stepping the motor that many times (with the direction determined by the difference's sign). If the target position changes while the stepper motor is running, the algorithm can update the number of steps such that the motor will reach the new position. Once the target position is reached, it is then set as the current position, and this cycle repeats. Due to the motor's step size and the alignment done earlier, this causes the arm attached to the motor shaft to rotate with the rotations of the BNO055, effectively mimicking the motion of the forearm when rotated by the elbow. This forms the core of the control algorithm, with the rest of the algorithm designed to support this core.

The first of these supports is to deal with how the BNO055 reports the X Euler angle, which is as an angle ranging from 0° to 360° [11]. This means if the BNO055 is rotated past its initial position, the X Euler angle value will wrap-around from 0 to 360 (and vice versa if it is moved back). This wrap-around and other similar discontinuities can cause erroneous rotations as the target position is suddenly up to several hundred degrees away from the current position. However, such large sudden changes in orientation are physically impossible for the elbow to perform, and therefore are anomalous and easy to identify amongst the angle data. The algorithm handles this by actively looking for these discontinuities, checking to see if the current position and target position differ by an impossible amount. If they do, the algorithm simply overrides the value of the current position and sets it equal to the value of the current target position. This prevents the

discontinuity from being seen by any other parts of the algorithm, while doing so quickly and precisely enough that minimal angle data is lost in the transition.

Another support to the core of the algorithm is to manage non-elbow rotations. Since any rotation about an axis that is parallel to the BNO055's yaw axis can affect the X Euler angle, certain rotations about the shoulder will cause erroneous changes in the angle data. Indeed, even the slight change in position from the user rotating their wrist is enough to shift the X Euler angle by a few degrees. The algorithm handles these rogue rotations by using the gyroscopes of the BNO055 and ICM-20948 (which is attached to the upper-arm) to detect rotations of the wrist and rotations about the shoulder respectively. If the gyroscopes read an angular velocity beyond a threshold value (indicating they are being rotated), the algorithm continuously sets the current position equal to the target position until the rotation stops. This allows the algorithm to effectively ignore any changes to the X Euler angle caused by these rotations and resume normal operations once the rotations have stopped.

The last support for the core involves handling a particular edge case. Specifically, while the X Euler angle is quite robust in tracking the forearm's position when used with the algorithm, certain rotations combined with certain hand orientations make the X Euler angle less effective at this task. One such combination is the basic bicep curl: rotation about the elbow while the arm is against the side of the body with a supinated hand. This movement is not about the yaw axis of the BNO055 and thus cannot be tracked by the X Euler angle. However, it can be tracked using the Z Euler angle, as the rotation is parallel to the BNO055's pitch axis. Therefore, if the algorithm knows that the user is about to begin a bicep curl, it can simply change which Euler angle it is reading to maintain tracking of the forearm. The algorithm handles this by using the BNO055's accelerometers to detect when the hand is oriented for a bicep curl, as the supination of the hand creates accelerometer readings unique to that orientation. If the hand is in that position, the algorithm sets the current and target positions to the current Z Euler angle value. So long as hand remains supinated, the algorithm behaves exactly as it did for non-bicep curl movements, only now using the Z Euler angle readings instead of the X Euler angle. When the BNO055 detects that the hand is no longer supinated, the current and target position are set to the current X Euler angle value and the algorithm goes back to reading the X Euler angle data.

Together, these supports and the core they underpin make up the control algorithm which the prototype uses. The code which implements the control algorithm on the microcontroller can be found in Appendix A.

2.2.3. Algorithm Design

While the idea of using differences in an IMU's angular position to control a stepper motor was a constant throughout the software development; how that idea was implemented had to be adjusted between iterations. Originally, once the difference between positions was determined, the motor would be controlled using a simple loop mechanism. The loop would simply move the motor in the specified direction by a single step every time the loop was ran, repeating a number of times equal to the positional difference. While this method worked for small differences (in the range of 1-2 degrees), larger differences would cause the loop to run much longer; and while the loop was running, positional data could not to be updated. This led to issues where if the forearm were moved during one of these longer loops, the prototype would not be able adjust for this change and thus cause the prototype arm to end up at the wrong position. This issue was solved by having the movement be handled by specialized stepper motor code libraries. These libraries allow the algorithm to perform as explained in the previous section, which includes giving the prototype the ability to update positional data while the motor is running.

Initially, the prototype's software was evaluated using a simple testing setup, which was effectively a deconstructed version of the prototype. This setup was used to test the core of the control algorithm; both to ensure that it worked with the selected hardware and to determine how certain parameters (such as step resolution and motor speed) affected the core's performance. Once it was determined that the core was functioning optimally, the testing setup was dismantled, and its parts used to construct the prototype. With the prototype built, testing could now be performed while it was being worn and operated by a user. This testing revealed the issues discussed in the previous section (i.e. the X Euler angle wrap-around, non-forearm movement causing rotations, and the curl edge case). The supports to the control algorithm which either eliminate or minimize these issues were implemented in the order they were described earlier.

This control algorithm allows the prototype to be controlled entirely through the monitoring of an IMU's angular position. While it is true that angular velocity and linear acceleration are also used in the algorithm, this data is used entirely for error checking. If the algorithm were to simply ignore this data, the prototype could still be operated but with errors being more likely to occur. Technically, the BNO055 does also use angular velocity and acceleration data to determine its angular position via its data-fusion software. However, since the algorithm is only reliant on the resultant position data, the velocity and angular acceleration are not considered necessary variables to the prototype's control algorithm. This is because, from the algorithm's perspective, it does not care how the BNO055 determines its position, just that is able to do so. Since the control algorithm uses only angular positioning, other control methods found on currently existing exoskeletons (such as directly measuring torque and forces created by the limb) are also not used.

Chapter 3.

Prototype Evaluation

The prototype is successfully capable of mimicking the user's forearm motion in most circumstances. It is comfortable to wear for extended periods of time, requires very little training to use effectively, is inexpensive to produce, and can be made entirely from commercially available products. However, the limitations to the prototype's motion tracking ability do limit its robustness, and its current battery limits usage of the device to only a few hours of continuous use.

3.1. Movement Tracking and Mimicry

When dealing with movements just about the elbow, the prototype performs marvellously. When properly aligned, the prototype can track and follow the rotations of the forearm, keeping its arm aligned to the BNO055 on the user's hand through a variety of different orientations. This tracking is effective at forearm rotations up to approximately 90° per second, with faster rotations increasing the potential for misalignment. This is because the more sudden deceleration needed to stop a faster rotation is more likely to trigger the wrist rotation detection by accident, which has the effect of stopping any prototype arm movements in progress. However, even if this was not the case, the motor being driven in half step mode creates a limit to how fast the motor is able to turn; and thus faster forearm rotations could still potentially cause issues due to the user being able to "outrun" the prototype arm.

The prototype can also maintain alignment during complex arm movements (i.e., movements involving elbow, shoulder and/or wrist rotations) but this requires a caveat. Due to the way the algorithm handles rotations of the wrist and about the shoulder, any forearm movement while the wrist or shoulder is rotating is not tracked by the prototype's arm. Therefore, wrist and shoulder rotations must be done before or after rotating the forearm, which has the effect of requiring the user to perform any complex arm movement as a series of individual steps (e.g., rotate about shoulder, then rotate about elbow, then rotate the wrist) to maintain alignment.

Since there is no mechanism to provide positional feedback to the control algorithm, if the prototype arm becomes unaligned it is incapable of re-aligning itself without user input. The most common way the arm can be unaligned is due to detecting X Euler angle changes from wrist and shoulder rotations that do not exceed their angular velocity detection thresholds. These occur most often if the shoulder is rotated horizontally by less than approximately 15° or the wrist is rotated less than approximately 45° per second. However, there are methods to realign the arm with the forearm without fully resetting the prototype. Abduction or adduction of the wrist can be used to adjust the prototype arm's position by up to approximately 5° in either direction, which can be used to make small adjustments. If a larger positional adjustment is needed, the user can adjust their forearm to realign it to the prototype arm while rotating their wrist, as the prototype arm will not move while sufficient wrist rotations are detected.

While the bicep curl edge case is mentioned specifically, there are other edge cases that exist where the Z Euler angle would be a more effective at tracking the forearm's position. These include a bicep curl with a pronated hand, forearm rotations involving a supinated hand with the arm parallel to the ground, and forearm rotations towards the body's midline while the arm is held vertically above the head with the palm also facing the midline. However, these edge cases are not handled by the algorithm in the same way as the basic bicep curl; and as such if the user performs any of these specific motions, it will cause the prototype to become unaligned. The primary reason they are not handled is that these configurations create no data values that are unique to them over their full range of motion. Thus, there would be no way for the prototype to reliably identify that it is in one of those configurations. Therefore, even if there was something that handled these edge cases, there would very likely still be loss of positional data which would cause the prototype to become misaligned. There would even be a possibility that, due to the lack of any unique identifiers, any handling protocol built for these edge cases could trigger and interfere with routine operations. As such, the prototype does not handle these edge cases to minimize any potential disruptions to its other processes.

Although the prototype is portable, it operates at maximum effectiveness when the user is stationary. If the user walks around while operating it, there is a significant chance that the prototype will become misaligned as the motion of walking can change the Euler angle value even if the user's arm is held stationary. As with the edge cases mentioned

previously, there is no data that can uniquely identify walking against other movements. Therefore, the prototype does not handle any angle change caused by walking, as any attempt to do so could negatively affect its overall performance. However, while walking there is very little to no movement about the elbow. Therefore, if there was a way to uniquely identify walking, the Euler angle changes it creates could be managed in a very similar way to how the prototype handles shoulder and wrist rotations (i.e., ignoring any Euler angle change while walking is detected).

3.2. Comfort and Ease of Use

The prototype is capable of being entirely donned, doffed, and activated with the user's opposite arm (i.e., the arm not wearing the prototype). The harness consists of three straps with ladder locks which allow the prototype frame to be securely fastened to the upper arm, while spreading the constrictive pressure between them such that they do not cut off the user's circulation. The stepper motor is also well insulated to prevent the user from being burned by the motor heating up during prolonged use. These factors, combined with the fact the prototype weighs only approximately 650 grams, allows it to be worn for extended periods of time without causing significant discomfort or fatiguing the user. However, the harness may occasionally need to be retightened over time, as the straps may shift in the ladder locks.

Since the prototype is entirely controlled by the user's forearm movement, learning how to use the prototype requires no special training beyond learning how to break down complex arm movements into individual steps and how to re-align in the event of the arm falling out of alignment. This allows for the prototype to be learned quickly, and thus easy to use for a variety of users.

3.3. Battery Life

In a worst-case scenario, the prototype draws approximately 360 mA of current during operation. Although the actual current draw of the prototype may be less than this, it is a fair assumption that the prototype's typical current draw is approximately this value; as the stepper motor draws an effectively constant 330 mA of current while the prototype is active, and thus the difference between any typical scenario and the worst-case scenario is at most a few tens of milliamps. Considering the prototype's power source (Eight 1.5 V

AA Duracell alkaline batteries) and assuming that each of the battery cells are used equally, the prototype is estimated to be able to run continuously for up to three to five hours [12]. This range is because the prototype can work at voltages lower than its power source's typical output. The electrical components of the prototype require only 5 V to operate, and the stepper motor can run at voltages lower than 12 V but drives with less torque if it does so. Thus, the range of voltages that would still allow the prototype to be functional is potentially quite wide, as very little torque is required to rotate the prototype's arm.

3.4. Cost

In terms of material cost, the prototype costs just under $360 to build (see Appendix B for a full cost breakdown). This makes the prototype relatively inexpensive to build, owing to the fact that it is made entirely with commercially available sensors and materials. The material costs can be broken down into seven categories: the electronics (which includes the motor, motor driver, IMUs, and microcontroller), the textiles, the power source, the frame, the arm, the connecting materials (such as wires), and the assembly materials (such as solder, glue, Velcro, etc.). Figure 4 shows the percentage of the material costs that each category represents.

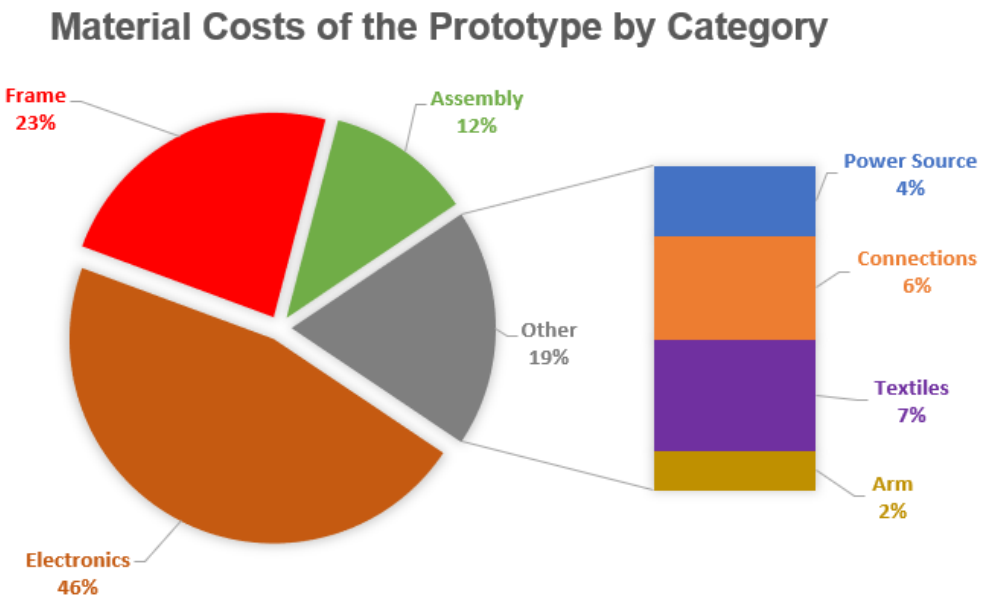


Figure 4: Percentage of Material Cost per Category

The electrical components being responsible for a majority of the material costs is expected, as they are critical to the prototype's function. However, the frame being responsible for nearly a quarter of these costs is unusual. This is due to the frame being a 3D printed item, which means unlike the prototype's other materials, it is not mass-produced, making it significantly more expensive than if it were.

Chapter 4.

Future Work

While the prototype being made entirely of mass-produced products makes it very replicable and inexpensive, utilizing more bespoke products in future iterations would significantly increase the prototype's functionality. Two areas where this is most apparent are in the prototype's harness and battery. The current harness, while functional, does suffer from needing to be occasionally retightened or readjusted. Furthermore, it concentrates most of the weight of the prototype onto one limb. This limits the weight of the prototype to weights that can be comfortably managed with just that limb. An improvement would be for the prototype to utilize a harness system like those found in upper-arm prosthetics (see Figure 5 for an example). These would still allow the user don, doff, and activate the prototype with only one arm, but also spread the prototype's weight across the body. Depending on how this harness is implemented, the prototype could even be rearranged such that parts of it could be distributed across the harness, to further reduce the amount of weight acting solely on the limb.

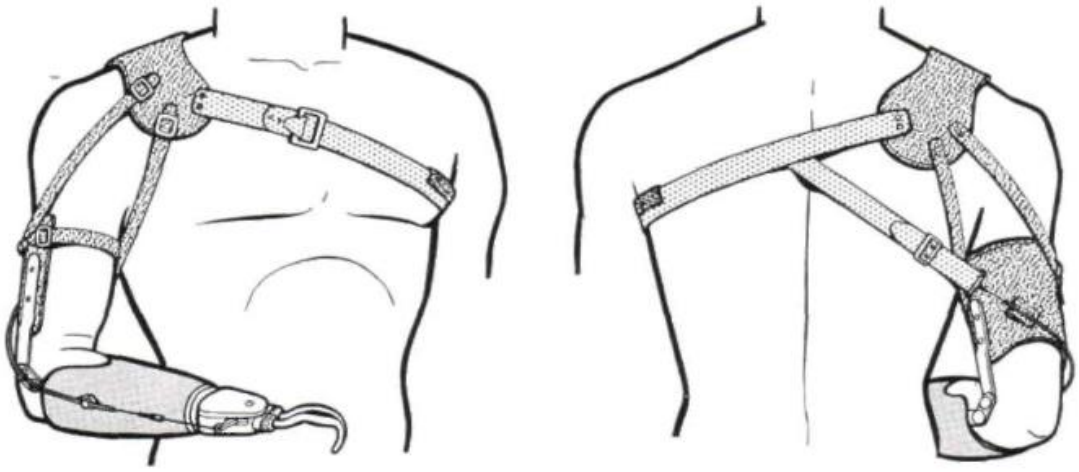


Figure 5: A Below-Elbow Chest-Strap Harness for an Upper-Extremity Prosthetic. Adapted from [13]

Likewise, the current battery is only guaranteed to power the prototype for a few hours and is quite heavy, being responsible for nearly a third of the prototype's weight. A high-capacity Lithium-based battery would not only allow the prototype to run for longer, but

possibly be even lighter than the current battery. The battery could also be positioned such that it is attached to the user's belt or elsewhere on an improved harness, to further reduce any discomfort caused by its weight or size.

One of the biggest limitations the current prototype has is the inability to reliably identify certain movements and configurations of the hand and arm. Future iterations can solve these issues by using additional sensors to provide greater data resolution. One example is adding an accelerometer which would sit over the user's pectoral area. This would detect whether the user is walking (from lateral acceleration readings), without accidentally detecting motion from the user just moving their arm. This would provide unique data for walking, and thus allow the prototype to handle any Euler angle changes caused by walking without potentially interfering with anything else. Another example would be adding a gyroscope to the user's hand specifically designed to track the hand's orientation. This, along with data about the position of the arm (from an IMU such as ICM-20948), would allow the prototype to know the exact orientation of both the user's hand and arm. This in turn would allow the prototype to easily identify when the user is in an orientation that corresponds to an edge case, which would allow for the prototype to be handle those cases more effectively. Of course, these sorts of changes cannot be made in a vacuum and would require other modifications to be implemented for them to work properly. The pectoral accelerometer would require a change to the harness such that a sensor could be placed there, and the addition of more sensors would require a microcontroller with more data inputs and power pathways to handle them.

Although misalignments caused by shoulder and wrist movements can still happen, the difference between the user's forearm and the prototype arm are small, usually no larger than approximately 15°. While the user can manually adjust their forearm to re-align themselves following these misalignments, an improvement that future iterations could implement is a way for the prototype to automatically correct these small positional discrepancies. This would not only reduce the instances of the user having to manually re-align the prototype but could also be used to ensure proper initial alignment during the prototype's start-up. One way this could be implemented is with some form of intensity detection. The idea would be that a detector of some specific stimuli (e.g., a certain frequency of light or sound) would be attached to the prototype's arm, while an emitter of that stimuli is attached to the glove of the prototype. After every movement of the prototype arm, the detector would check if the intensity it is reading is within a range that

indicates it is aligned (i.e., within a certain distance of the emitter). If it is not, the prototype would automatically move the arm towards the emitter until values in the alignment range are detected. This method would require that whatever stimulus is chosen is not present in the environment the prototype is being used in, otherwise the detection would be vulnerable to confused false positive readings. Another method could be to attach an IMU to the prototype's arm and compare the Euler angle readings from it and the IMU on the user's hand to determine if a misalignment has occurred. If the readings do not match then the prototype can rotate the arm back into alignment, rotating to minimize the difference between the two angle readings until alignment is maintained (i.e., the angle reading from both IMUs match or are within a certain tolerance). However, this method would require that both IMUs be precisely synchronized with each other and remain that way for as long as the prototype is being used. These are just two of potentially many different methods that could implement this automatic correction feature; because of this, further research would need to be performed to determine which method would work the best for this particular purpose.

The actuation system is an obvious area in which future iterations of the prototype could be improved. While the current method of actuation does effectively demonstrate an IMU based control system, it lacks the power necessary to rotate any arm that was not built specifically to weigh as little as possible. Naturally, this design constraint limits the construction of the prototype arm: both in what materials can be used to build the arm itself and what (if any) sensors and accessories can be attached to it. Future iterations could implement stronger actuation systems, which in turn would allow the prototype arm to be heavier. This extra weight allowance could allow sensors to be attached to the arm to provide more positional data to the control algorithm; or potentially even allow for the addition of some form of manipulator to the arm, to give it some form of functionality. Although this change could be as simple as using a stepper motor capable of driving more torque, this could also include implementing a completely different style of actuation. Since any actuation method that can perform rotations as small as ±1° could technically work as a potential replacement, further research would be needed to determine which methods would provide more power while minimizing any increases in weight and maintaining similar levels of precision as the current method.

Chapter 5.

Conclusion

In this thesis, we developed a person-portable exoskeleton prototype that demonstrates an IMU-based control system. This prototype uses the Euler angles generated by the IMU to track the motion of the forearm. This is possible because the Euler angles can detect rotations not only of the IMU itself, but also rotations about a parallel axis (such as about the elbow when the IMU is attached to the hand). The prototype detects these forearm rotations, and then has its own arm perform the same rotation, essentially mimicking the motion of the forearm. The prototype is successful in demonstrating this control methodology; being capable of mimicking rotations of the forearm in a variety of different orientations while being relatively inexpensive to build. However, the inability to reliably identify and handle certain movements and a lack of power in the actuation system does significantly limit the prototype's overall functionality; but these problems do have potential solutions which could be implemented in future iterations of this exoskeleton. Despite these issues, the prototype still demonstrates how IMUs can be used to control an exoskeleton with a minimal apparatus; in the hope that with future research building on this idea, powered exoskeletons can be built lighter and smaller than ever before. A live demonstration of the prototype will be performed during the thesis defense as a compliment to this thesis.

References

[1] H. Lee, W. Kim, J. Han, and C. Han, "The technical trend of the exoskeleton robot system for human power assistance", *International Journal of Precision Engineering and Manufacturing*, vol. 13, no. 8, pp. 1491-1497, 2012. Available: 10.1007/s12541-012-0197-x.

[2] M. de Looze, T. Bosch, F. Krause, K. Stadler, and L. O'Sullivan, "Exoskeletons for industrial application and their potential effects on physical work load", *Ergonomics*, vol. 59, no. 5, pp. 671-681, 2015. Available: 10.1080/00140139.2015.1081988.

[3] H. Lo and S. Xie, "Exoskeleton robots for upper-limb rehabilitation: State of the art and future prospects", *Medical Engineering & Physics*, vol. 34, no. 3, pp. 261-268, 2012. Available: 10.1016/j.medengphy.2011.10.

[4] L. Marchal-Crespo and D. Reinkensmeyer, "Review of control strategies for robotic movement training after neurologic injury", *Journal of NeuroEngineering and Rehabilitation*, vol. 6, no. 1, 2009. Available: 10.1186/1743-0003-6-20.

[5] H. Kazerooni, "The human power amplifier technology at the University of California, Berkeley", *Robotics and Autonomous Systems*, vol. 19, no. 2, pp. 179-187, 1996. Available: 10.1016/s0921-8890(96)00045-0.

[6] K. Kong and D. Jeon, "Design and control of an exoskeleton for the elderly and patients", *IEEE/ASME Transactions on Mechatronics*, vol. 11, no. 4, pp. 428-432, 2006. Available: 10.1109/tmech.2006.878550.

[7] C. Fleischer, C. Reinicke and G. Hommel, "Predicting the intended motion with EMG signals for an exoskeleton orthosis controller", *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005. Available: 10.1109/iros.2005.1545504.

[8] A. Filippeschi, N. Schmitz, M. Miezal, G. Bleser, E. Ruffaldi and D. Stricker, "Survey of Motion Tracking Methods Based on Inertial Sensors: A Focus on Upper Limb Human Motion", *Sensors*, vol. 17, no. 6, p. 1257, 2017. Available: 10.3390/s17061257.

[9] A. Noccaro, F. Cordella, L. Zollo, G. Di Pino, E. Guglielmelli and D. Formica, "A teleoperated control approach for anthropomorphic manipulator using magneto-inertial sensors", *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2017. Available: 10.1109/roman.2017.8172295.

[10] K. Little et al., "IMU-based assistance modulation in upper limb soft wearable exosuits", *2019 IEEE 16th International Conference on Rehabilitation Robotics (ICORR)*, 2019. Available: 10.1109/icorr.2019.8779362.

[11] "BNO055 Intelligent 9-axis absolute orientation sensor", *Bosch-sensortec.com*, 2021. [Online]. Available: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bno055-ds000.pdf. [Accessed: 21- Feb- 2021].

[12] "LR6 MN1500 Technical Datasheet", *Duracell.com*, 2021. [Online]. Available: https://www.duracell.com/wp-content/uploads/2020/02/MN15US11191.pdf. [Accessed: 20- Mar- 2021].

[13] R. Pursley, "Harness Patterns for Upper-Extremity Prostheses", *Artificial Limbs*, vol. 2, no. 3, pp. 26-60, 1955. Available: http://www.oandplibrary.org/al/pdf/1955_03_026.pdf. [Accessed 27 March 2021].

Appendix A.

Code for Prototype Control Algorithm

```
/*************************************************************
 * ugthesis3.0.ino
 * Arduino code for undergraduate thesis project "A Novel
Exoskeleton Prototype Based on the Use of IMUs to Track and Mimic
Motion"
 * For use with Arduino IDE ver. 1.8.13 and the following Arduino
libraries:
 * -AccelStepper ver. 1.61.0 by Mike McCauley
 * -Adafruit Unified Sensor ver. 1.1.4 by Adafruit
 * -Adafruit BNO055 ver. 1.4.2 by Adafruit
 * -Sparkfun 9DoF IMU Breakout - ICM 20948 ver 1.1.2 by Sparkfun
Electronics
 *
 * Code includes code taken from the following sources:
 * -"Arduino Code | Adafruit BNO055 Absolute Orientation Sensor"
by Kevin Townsend (Accessed: Feb. 02, 2021,
https://learn.adafruit.com/adafruit-bno055-absolute-orientation-
sensor/arduino-code)
 * -"ProportionalControl.pde" by Mike McCauley (Accessed: Feb.
02, 2021,
https://www.airspayce.com/mikem/arduino/AccelStepper/Proportional
Control_8pde-example.html)
 * -"Example1_Basics.ino" by Owen Lyke (Accessed: Feb. 02, 2021,
https://github.com/sparkfun/SparkFun_ICM-
20948_ArduinoLibrary/blob/master/examples/Arduino/Example1_Basics
/Example1_Basics.ino)
 *
 * Author: Harry Draaisma
 * Original Creation Date: January 18, 2020
 *
 * Distributed as-is, for use with project hardware.
 *************************************************************/
//include relevant libraries
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BNO055.h>
#include <utility/imumaths.h> //included with BNO055 library
#include <AccelStepper.h>
#include "ICM_20948.h"

//define variables for pins related to indicator LED and step
resolution selection
#define MS1 4
#define MS2 5
#define MS3 6
```

```
#define LED 10

//setup upper arm sensor (the ICM-20948) to communicate via SPI
(Serial Peripheral Interface)
#define USE_SPI
#define SERIAL_PORT Serial
#define SPI_PORT SPI    // Your desired SPI port.
#define CS_PIN    2     // Which Arduino pin you connect CS pin
to.

//construct object representing hand sensor (the BNO055)
Adafruit_BNO055 bno = Adafruit_BNO055(55, 0x29);
#ifdef USE_SPI
  ICM_20948_SPI myICM;  // If using SPI create an ICM_20948_SPI
object
#else
  ICM_20948_I2C myICM;  // Otherwise create an ICM_20948_I2C
object. Not actually used in this code
#endif

/* initialize global variables */
char curl = 'n'; //flag for if hand sensor is in "curl" position,
intialized to 'no'
float wrot; //wrist rotation
float sroty; //horizontal shoulder rotation
float srotz; //vertical shoulder rotation
float eXpres = 0;
float eXpast = eXpres;
float eZpres = 0;
float eZpast = eZpres;
float diffx;
float diffz;
//construct AccelStepper object to interface with stepper motor
connected via a motor driver
//8 refers to the Arduino pin which activates the motor driver, 9
is the Arduino pin which determines motor direction
AccelStepper stepper(AccelStepper::DRIVER, 8, 9);

void setup(void) {
//put your setup code here, to run once:
  //set LED and step resolution pins to be outputs, and turn off
indicator LED
  pinMode(LED, OUTPUT);
  pinMode(MS1, OUTPUT);
  pinMode(MS2, OUTPUT);
  pinMode(MS3, OUTPUT);
  digitalWrite(LED, LOW);

  //set baud rate for serial monitor (for debugging purposes)
  Serial.begin(115200);

  //initialize SPI port for upper arm sensor
```

```
  #ifdef USE_SPI
      SPI_PORT.begin();
  #else
      //initialize I2C bus for upper arm sensor. Not actually
used in this code.
      WIRE_PORT.begin();
      WIRE_PORT.setClock(400000);
  #endif

  //boot-up upper arm sensor, and verify it is both sending and
receiving data
  bool initialized = false;
  while( !initialized ){

  #ifdef USE_SPI
      myICM.begin( CS_PIN, SPI_PORT );
  #else
      myICM.begin( WIRE_PORT, AD0_VAL );
  #endif

  SERIAL_PORT.print( F("Initialization of the sensor returned: ")
);
  SERIAL_PORT.print( myICM.statusString() );
    if( myICM.status != ICM_20948_Stat_Ok ){
      SERIAL_PORT.println( "Trying again..." );
      delay(500);
    }else{
      initialized = true;
    }
  }

  //boot-up glove sensor, and verify it is sending and receiving
data
  if(!bno.begin())
  {
    /* There was a problem detecting the BNO055 ... check your
connections */
    Serial.print("Ooops, no BNO055 detected ... Check your wiring
or I2C ADDR!");
    while(1);
  }

  delay(1000);

  //finish hand sensor initialization
  bno.setExtCrystalUse(true);
  //set speed and acceleration limits for stepper motor
  stepper.setMaxSpeed(3000);
  stepper.setAcceleration(1000);
  //Set motor to half step mode and then turn on indicator LED to
alert user system is active
  digitalWrite(MS1, HIGH);
```

```
  digitalWrite(MS2, LOW);
  digitalWrite(MS3, LOW);
  digitalWrite(LED, HIGH);
}

void loop() {
  // put your main code here, to run repeatedly:

  //Read accelerometer, gyroscope, and euler angle position data
from hand and upper arm sensor
  imu::Vector<3> accel =
bno.getVector(Adafruit_BNO055::VECTOR_ACCELEROMETER);
  imu::Vector<3> gyro =
bno.getVector(Adafruit_BNO055::VECTOR_GYROSCOPE);
  imu::Vector<3> euler =
bno.getVector(Adafruit_BNO055::VECTOR_EULER);
  myICM.getAGMT();

  //debug(diffx, diffz, curl); //Test function used for
debugging. Uncomment to see data via Serial Monitor

  //record gyroscope readings that detect wrist rotations or
rotations about the shoulder
  wrot = gyro.y();
  sroty = myICM.gyrY();
  srotz = myICM.gyrZ();

  //if wrist rotations or rotations about the shoulder are
detected, ignore change in Euler angle as a result
  //then check to see if rotation has stopped
  while(abs(wrot)>60 || abs(sroty)>15 || abs(srotz)>15){
    switch (curl){
      case 'n':
       stepper.setCurrentPosition(eXpres);
       break;
      case 'y':
       stepper.setCurrentPosition(eZpres);
       break;
    }
    imu::Vector<3> gyro =
bno.getVector(Adafruit_BNO055::VECTOR_GYROSCOPE);
    myICM.getAGMT();
    wrot = gyro.y();
    sroty = myICM.gyrY();
    srotz = myICM.gyrZ();
  }

  //if hand sensor is in "curl" position, set "curl" flag to true
  //and change position tracking to use Z Euler angle
  if(accel.y() > 6.0 && accel.z() < 0){
    if(curl == 'n'){
      stepper.setCurrentPosition(eZpres);
```

```
    }
    curl = 'y';
  }

  //record current X and Z Euler angle positions
  eXpres = euler.x();
  eZpres = euler.z();

  //record difference between current and previous euler angle
readings
  diffx = round(eXpres - eXpast);
  diffz = round(eZpres - eZpast);

  //Main control section of code
  switch (curl){
    //if hand sensor is in curl position
    case 'y':
      //check to see if hand sensor is out of "curl" position
      //if it is, reset "curl" flag and change position tracking
to use X Euler angle
      if(accel.z() > 0){
        curl = 'n';
        stepper.setCurrentPosition(eXpres);
        break;
      }
      //if statement to ignore discontinuties in Z Euler angle
readings
      if(abs(diffz) > 15){
        stepper.setCurrentPosition(eZpres);
      }else{
      //move motor arm to new position dictated by Z Euler angle
reading
      stepper.moveTo(eZpres);
      stepper.setSpeed(100);
      stepper.runSpeedToPosition();
      }
      break;
    //if had sensor is not in "curl" position
    case 'n':
      //if statement to ignore discontinuties in X Euler angle
readings
      if(abs(diffx) > 50){
        stepper.setCurrentPosition(eXpres);
      }else{
      //move motor arm to new position dictated by X Euler angle
reading
      stepper.moveTo(eXpres);
      stepper.setSpeed(100);
      stepper.runSpeedToPosition();
      }
      break;
  }
```

```
  //set the current Euler angle reading to be the previous Euler
angle readings
  eXpast = eXpres;
  eZpast = eZpres;

}

void debug(int diffx, int diffz, char curl){
  //This function acts as a way to visually see the data via the
serial monitor
  //It is intended to be used as a way to debug the code during
the developent
  imu::Vector<3> euler =
bno.getVector(Adafruit_BNO055::VECTOR_EULER);
  imu::Vector<3> accel =
bno.getVector(Adafruit_BNO055::VECTOR_ACCELEROMETER);
  Serial.print(" X:");
  Serial.print(euler.x());
  Serial.print(" Z:");
  Serial.print(euler.z());
  Serial.print(" Curl:");
  Serial.print(curl);
  Serial.print(" Diffx:");
  Serial.print(diffx);
  Serial.print(" Diffz:");
  Serial.print(diffz);
  Serial.println("");

  delay(1);
}
```

Appendix B.

Material Costs of the Prototype

8 AA Battery Enclosure	$3.50
8 AA Batteries	$10.99
2.1 mm DC Barrel Jack	$3.50
Arduino Uno Rev3 Microcontroller	$33.00
Big Easy Driver Motor Driver	$25.00
1.8°/12V 2 Phase Stepper Motor	$31.80
Adafruit BNO055 9-DOF Absolute Orientation IMU Fusion Breakout Board	$50.41
SparkFun 9DOF IMU Breakout - ICM-20948	$24.45
ABS-M30 3D Printed Frame	$83.49
Solderless 54 x 83 mm Breadboard	$5.80
Second-hand Glove	$3.99
6” x 4” Sticky Back Velcro for Fabrics	$8.99
¾” x 48” Round Wooden Dowel	$5.19
3 ¼” Rubber Grommets	$2.91
12’ x ¾” Velcro Roll	$15.99
3 Ladder Locks	$2.98
80 Jumper Wires	$12.16
2 4” x 2” Velcro Strips	$5.99
2 1.875” Radius Velcro Coins	$5.99
Miscellaneous (Solder, Glue, etc.)	$20.00
Total	**$356.13**