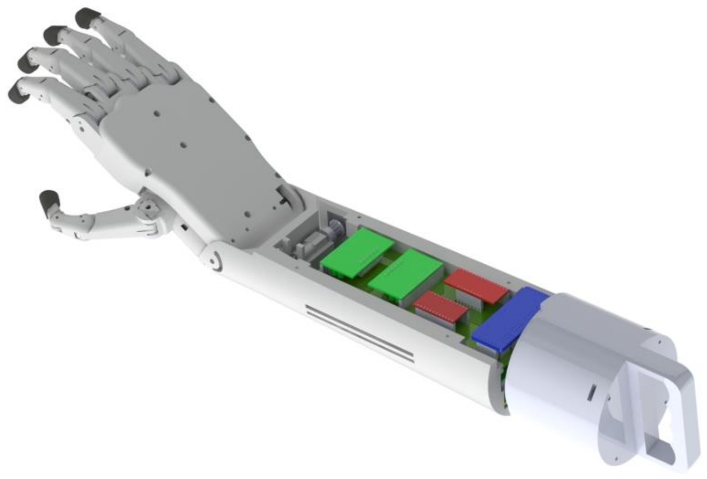
Vision-based Intelligent Prosthetic Robotic Arm



Written and Submitted by

Robert Edwards

Kyle Lafontant

Nuttaworn Sujumnong

Jared Wormley

Project Advisors:

Professor Cagdas D. Onal

Professor Taskin Padir

Abstract

Many intelligent trans-radial prosthetic devices require various forms of user input to properly form desired grasps and perform functions. This project is based around an already-existing design and involves the design and development of an anthropomorphic trans-radial prosthetic robot that offers low-level assistive intelligence. The device is capable of autonomously determining object positions through a palm-embedded camera, and utilizes a two degree-of-freedom wrist to track and orient itself to an object. As the arm moves, the camera and wrist allow for the hand to remain at a convenient orientation with respect to the object and prepare finger positions to facilitate a subsequent grasp. The fingers are actuated using a bevel-gear driven system that allows for variable grip strength through the relation between the forces generated at the fingertips and the current draw of electric motors. The core advantage of this system is its user-friendly human interaction, reducing cognitive burden during grasping activities of daily living.

Acknowledgements

- Taskin Padir and Cagdas Onal
For their assistance and guidance throughout the project

- Selim Ozel
For acting as a co-advisor and helping hand

- WPI Robotics Engineering Department
Including Joe St. Germain and Tracey Coetzee

- Thanacha Choopojchareon
For his insight into last year project

Contents

Introduction

Prosthetic devices have been developed for many decades to assist and compensate people who have lost their limbs to accidents or pre-existing disabilities. Many prostheses have tackled problems on how to achieve similar if not parallel capabilities to those of human limbs. With increasing complexity and functionality of modern prostheses, control architectures have evolved to make operation of the device more intuitive and convenient for amputees in their daily routines. Examples of these control architectures include passive motion of the body and electrical signals sent to muscles.

The previous iteration of this project, the IRIS Hand, involved the development of a smart robotic prosthetic device that utilized object recognition with a camera embedded within the device to identify objects and autonomously determine grasps prior to manipulating those objects. As a continuation of this technology, this project involves the realization of a prototype trans-radial prosthesis. This includes the redesign of the actuation system in the finger and the mechanical design for wrist’s motion.

Background Research

Prosthetic Devices

The advent of prosthesis opened up a field of engineering that would expand for many generations. The first prostheses were developed for basic functional use consisting of peg legs for foot prostheses and hooks for hand prostheses. As time passed, many different types of tradesman including blacksmiths, watchmakers, and locksmiths developed more complex prostheses for increased comfort and functionality. Materials evolved from steel and iron, to aluminum and plastic allowing for similar functionality, but with lighter materials that were more comfortable for the amputee. Devices were developed to incorporate natural body motions to execute various functions. Today, the evolution of prosthetics continues as robotic solutions for prosthetics become more prevalent (Norton, 2009).

Hand Anatomy

This project focuses on the development of a robotic prosthetic hand up to its forearm. In order to properly design an anthropomorphic design for this prosthetic device, there had to be understanding of the human forearm, wrist, and hand so that they can be properly approximated with the prosthetic.

According to George ElKoura, a scholar at the University of Toronto, the human hand consists of a total of 21 degrees-of-freedom (DOF), and the wrist has a total of 6 degrees-of-freedom (ElKoura, 2003). Each finger has 4 degrees-of-freedom, 3 of which are for flexion and extension motions and 1 for the abduction and adduction motion (ElKoura, 2003). Flexion motion describes motion where two links are coming closer together about an axis of a joint, extension motion is the converse (Bruenger, 1994). These types of motion refer to the curling and straightening of the finger. Adduction motion involves any motion where a joint it moving closer to the center point of the body, and the converse is abduction motion (Bruenger, 1994). For the human finger, when the joint at the knuckle moves toward the body it is adduction and when it moves the opposite direction it is adduction. The motions described for each of the fingers are accomplished through the mechanical links and joints that make up the finger. Each finger comprises of a total 4 bones and 3 joints. The bone connected to the knuckle is called the metacarpal, which is connected to the proximal phalanx by the metacarpophalangeal joint. The proximal interphalangeal joint connects the proximal phalanx to the middle phalanx, and the distal interphalangeal joint connects the middle phalanx to the distal phalanx (Bio-Mech). *Figure 1* below shows these bones and joints as described above (Parasuraman, 2009).

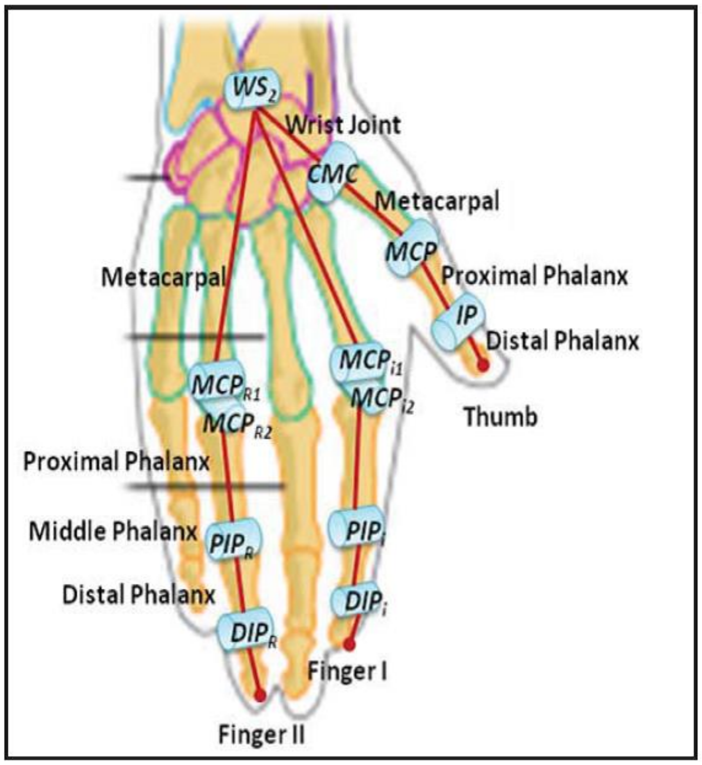


Figure 1: Shows Bones and Joints Locations of the Fingers and Thumbs

Note that the thumb does not contain a proximal interphalangeal joint, a middle phalanx, and a distal phalangeal joint, but instead has one interphalangeal joint between the proximal and distal phalanges. Though the wrist has one less link, it actually has a total of 5 degrees-of-freedom including the 2 flexion and extension degrees-of-freedom and the 3 degrees-of-freedom at its carpo-metacarpol joint in between the metacarpal and wrist joints (Parasuraman, 2009). The wrist itself is made up of eight bones called the carpals (Fincher, 1997) and has a total of 6 degrees-of-freedom one for each of translation and rotation about each of the three common axes (Parasuraman, 2009).

The bones described in the previous section make up the mechanical aspects of the hand and wrist, but the actual actuators for the arm are comprised of the muscles. The muscles in the forearm connect down to the wrist and the hand through tendons, meaning much of the actuation of the wrist and fingers comes from muscles in the forearm (Taylor). The electrical signals from the brain that tell the muscles how much force to apply and when to actuate come from the nervous system. This system extends through the body and drives all of its actuation, but one major nerve, the median nerve, drives the actuation within the forearm, wrist and hand (Taylor). In a sense the bones act as the structure for the body, the muscles are the motors, and the nervous system acts as the controller. Combining these three aspects with the brain makes the body similar to a robotic system, explaining the increase in robotic solutions to interface with the human body to solve prosthetic needs.

Available Prostheses

To come up with the best design for our trans-radial prostheses, we conducted research on similar products in the market. We observed that there were many different devices available for those who have had the misfortune of losing their hands. Among available options, there are both non-anthropomorphic and anthropomorphic (human-like) prosthetic devices giving users a wide variety of options from mechanically designed to electronically functional.

Non-Anthropomorphic Prosthetic Device

A non-anthropomorphic prosthetic device is one that does not look like that of a human body part. That being said, there have been many different non-anthropomorphic devices that have been created that vary in sophistication. The most commonly seen non-anthropomorphic prosthetic device is a hook. The use of prosthetic hooks started in the early 1900's giving those without a hand a new way of being able to grasp certain objects. Some of these hooks have a level of complexity that allows the users to use their back and several straps to actually articulate the hook itself. This system can be seen in *figure 2.*

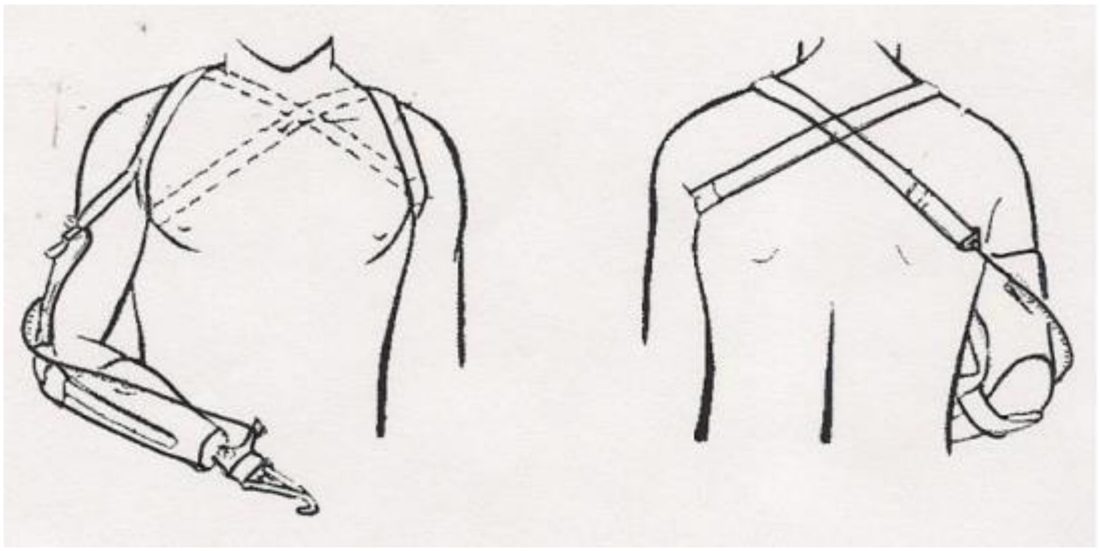


Figure 2: Prosthetic Hook System

As the user extends his or her arm, the cable opens the split hook allowing the user to grab objects. Though it can be very useful, this system lacks the appearance and functionality of a real hand, potentially making it a less desirable option.

Anthropomorphic Prosthetic Hands

An anthropomorphic prosthetic device is simply a device that has the characteristics of a human body part; in this case, the human hand is the main focus. Some of these are strictly aesthetic devices that only look like a hand and do not move in any way to those who have some sort of mechanical components to it that allow movement. These date all the way back to the 1400’s and can be seen in *figure 3.*

Figure 3: Prosthetic Hand from the 1400's

Devices like this were used simply to make those without the limb to look as if they did though there was no real intelligence or movement available. Two of the most sophisticated and popular anthropomorphic prosthetic hands were found to be the iLimb and the bebionic Myoelectric Prosthetic hand (iLimb, 2014). These two individual products, though very similar, have different levels of complexity.

Bebionic

The Bebionic Myoelectric Prosthetic hand is one of the most developed prostheses that have intelligence in it. The hand itself has two different sizes. The medium size is based off of the size of an average adult male's hand while the large is just slightly bigger. This system has independent motors for each finger which allows for each finger to have its own unique configuration to the others. This allows for a large amount of varying grip types since each finger is able to move on its own. Two of these grips include power grip and a static hook grip. The time it takes to go from an open hand to power grip is 0.5 seconds and the maximum grip force that can be applied on an object is 140.1 Newtons. The power grip can be used to throw a ball, eat a piece of fruit or grip other spherical objects. The maximum static load that the hook grip is able to hold for the whole hand is 45 kg while an individual finger in hook grip can hold up to 25 kg. The hook grip can be used to carry something like a grocery bag or even grasp a door handle. Currently in order to change the thumb from an opposed position to a non-opposed position, the user must manually move it. Though to move from grip to grip, electrodes are used to allow the user to use their muscles to control the opening and closing of the fingers. The manner at which they apply these signals (short bursts, quick rising stimulus, slow falling stimulus) determines what grip is chosen. In addition to having separate finger control, there are three different wrist attachments available for the bebionic3 myoelectric prosthetic which can be seen in *figure 4*.

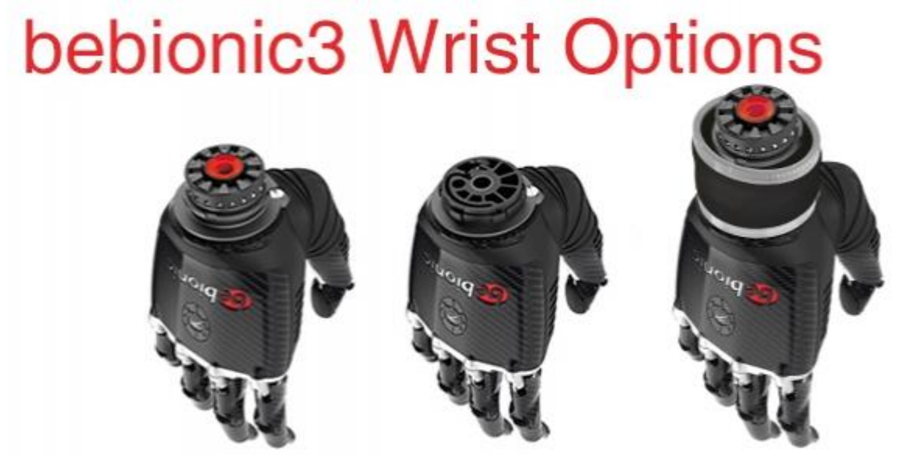


Figure 4: bebionioc3 Wrist Options for Users

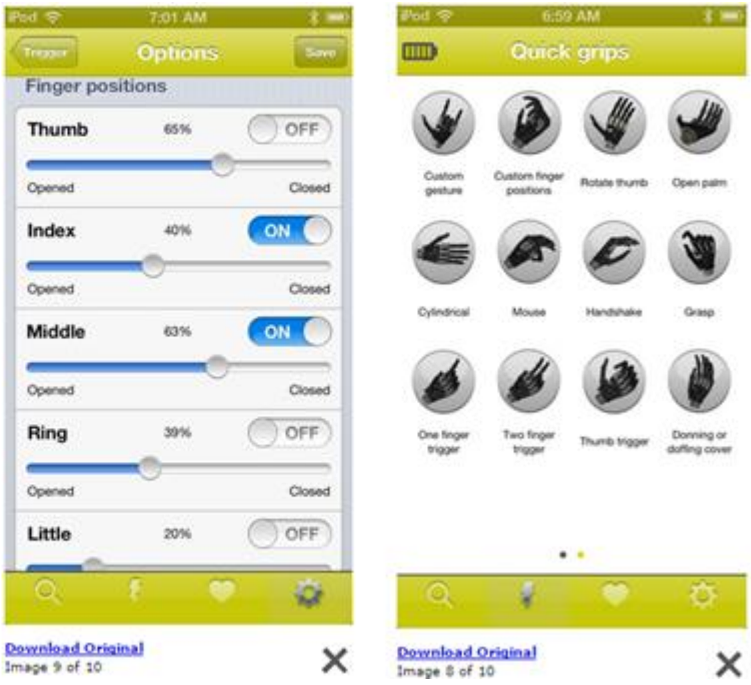
The electric quick disconnect wrist (*figure 4* left) allows the wearer to quickly rotate and remove the hand to interchange with other terminal devices. This particular wrist only allows for a rotation using powered wrist rotators. The multi-flex wrist (*figure 4* right) allows the wearer movement in all directions giving it 3 degrees-of-freedom in all rotational directions and in addition to control or the roll, pitch and yaw of the wrist, the wrist can lock in 30 degree flexion, 30 degree extension or a neutral position. This allows the user to be able to perform daily tasks while not having to worry about the hand slipping or rotating in any direction. This more sophisticated version of the EQD wrist also still has the ability to be interchanged with other terminal devices. The final wrist type that the user has the ability to choose from is the short wrist (*figure 4* middle). This offers a reduced build height to accommodate for those amputees with amputations closer to the wrist and would not otherwise be able to support either other wrists naturally without leaving that particular arm long. This wrist choice only allows for rotation around the arm though not the other two degrees of freedom allowed from the multi-flex wrist.

This sophisticated prosthetic even allows users to decide whether they want a glove to give a natural look and feel made of silicone. This silicone is made using multiple layers similar to human skin.

iLimb

The iLimb made by Touch Bionics is another example of an anthropomorphic myoelectric prosthetic device for hand amputees. Though it is very similar to the bebionic3, each finger is individually powered. The thumb must be manually rotated in order to achieve various grips. One key feature of the iLimb is its proportional control. Like the bebionic3, this iLimb is electrode controlled. The stronger the input signal that the electrode receives, the faster the fingers will move. This gives the user complete control over the speed at which the fingers both close and open their grip. In addition to this, if the user pulses the input, the group can increase the amount of force it has on the object though there is no real feedback that the user gets indicated how much force it is actually applying. The iLimb is made of an aluminum chassis for increased durability and long life span. There are two available sizes, one corresponding to the average size of an adult male's hand, and one corresponding to the average size of an adult female's hand. One useful feature is that after long periods of inactivity, the hand will automatically move from whatever grip configure it may have been in to a natural position.

One key feature that truly sets the iLimb apart from the bebionic3 is the ability to customize the configurations of the hand. Using the my "iLimb" app that can be downloaded to any phone or computer for free, the user has the ability to not only customize the types of grips most frequently used, but even customize specific gestures that are commonly used. This app can also control the performance output of the hand, the different grips in use at that moment, all the way down to the exact position of each individual finger. Two pictures from the mobile app of picking a grip as well as changing the finger position can be seen in *figure 5*.



Finger positions

Thumb 65% OFF
Opened Closed

Index 40% ON
Opened Closed

Middle 63% ON
Opened Closed

Ring 39% OFF
Opened Closed

Little 20% OFF

Download Original
Image 9 of 10

Download Original
Image 8 of 10

Figure 5: iLimb Mobile App Interface

In addition to these features, the iLimb also has multiple available coverings. These coverings are to ensure that the user can use their device while handling dust as well as water and neither will affect the internal circuitry. Two popular non-human like coverings that the iLimb have are the active skin and the active TS. These are made of semi-transparent, clear or black materials that are computer modeled to fit perfectly on the device. The difference between the two is that the TS version includes a conductive tip on the index finger making it compatible for touch screens on smartphones or tablets.

Object Recognition

Object recognition is an application of computer vision. Video recorded from a camera is parsed through a recognition algorithm to generate useful data, generally lines, geometry or points associated with objects within the camera's view. Recognition can be divided into two major categories: specific and generic. The generic method utilizes categorical traits of a type of item, and compares the image read against these categories to determine the type of object being viewed. The specific category, however, attempts to categorize an object as a specific item, such as a particular person's face or a particular structure; the bounds of the recognition are more strictly defined to determine the likelihood of the object being identified (Grauman, 2011).

For our robotic prosthesis, object recognition will be used to identify objects and determine appropriate grasps to lift or grasp the items that the arm is expected to interact with. To do this, further research would need to be done into the various recognition algorithms to determine the most appropriate direction.

There are two major methods of object recognition: appearance-based and feature-based.

Appearance-based Recognition

Appearance-based object recognition is the utilization of example images, which are used to compare against the live data set pulled from the camera input to determine likelihood of identification.

Direct Correlation Method

The direct correlation, or template matching, method is a comparison algorithm that utilizes stored images and compares the captured image data to that particular image. This data in some instances can be converted to a vector and directly compared to a vector generated from the templates (Heseltine, 2013). Through this, image similarity can be obtained and if the threshold of the algorithm for the difference in the two image vectors is not surpassed, the object is identified.

Some of the more primary concerns of the algorithm are variances due to lighting, orientation of the object, or distortion due to viewpoint or illumination. This can be accounted for through the use of multiple template images to correlate to different instances, or through applying a Gaussian blur to the template image to allow for a more coarse comparison (Berg, 2005).

Edge Detection

Edge detection is used to determine edges within a captured image of an object. These edges can be used in software to simplify a digital image grabbed from a camera or file system down to a collection of pixels representing edges, to put through additional filters. One of the more commonly used edge detection methods is the Canny Edge detection algorithm (Karla, 2009). The focus of the Canny edge detection is to maximize the detection of real edge points and lowering the probability of detecting non-edge points (false positives), and only detecting one edge per real edge in the image (Karla, 2009).

The approach to a Canny edge detection algorithm is as follows:

1. Initial conversion of the image to grayscale. Histogram-stretching to utilize full gray-scale range (this step may not be completely necessary depending on the tuning of the constants in your algorithm).
2. Application of a Gaussian filter, or blur, in order to reduce potential noise from the system.
3. Calculation of the gradients of the image. This is done through the use of the *Sobel-operator*, which performs a 2-D spatial gradient measurement (Karla, 2009). This can be used to determine direction of changes in the gradient.
4. Non-maximum suppression. All local maxima within the gradient image are preserved, while removing other values. The thick approximate lines generated from the gradient calculation are thinned to finer edge approximations.
5. Double thresholding. Two thresholds are used to mark strong, weak, and negligible edges. Only strong and weak edges are preserved, while negligible edges are deleted.
6. Edge tracking by hysteresis. Only weak edges connected to strong edges are preserved, while isolated weak edges are removed from the model.

From this, the image has been refined and isolated to only a collection of edges, which can be used by the device for further analysis. Canny edge detection is often used in conjunction with other imaging software to simplify input images for comparison to templates with reductions in effects from illumination or noise. This edge detection model is useful due to the use of two thresholds to determine important lines, which allows for more refined parsing of the image.

Feature-based Recognition

Feature based object recognition is the extraction of features from the contours and data of an image. These features are comprised of lines, arcs and lobes, which can be refined further and combined to make larger features (Howarth, 2009). From this reduction an input image can be reduced to a series of shapes or features, which can be used to determine the geometry of an object. These can generally be used in conjunction with appearance based recognition methods and a database of images and templates to determine the most likely object held within the image.

SIFT Detection

Scale-Invariant Feature Transformation, or SIFT, is a descriptor that is calculates points of interest within an image, utilizing directions of local gradients. At each of these points of interest, image descriptors are calculated, which are normalized to be scale-invariant and rotation-invariant. These points of interest are then used to compare and match to points of interest generated within other images.

From here, local image descriptors between two images can be matched to perform object recognition using existing templates; these matches can be visualized as a line between each match, which takes into account changes in scale or orientation of the item within the image. An example of two images being compared can be seen in *figure 6*.There are a few methods for determining point matches within two images, such as the application of a Best-Bin-First (BFF) algorithm to scale for larger numbers of image descriptors (Lindeberg, 2012).

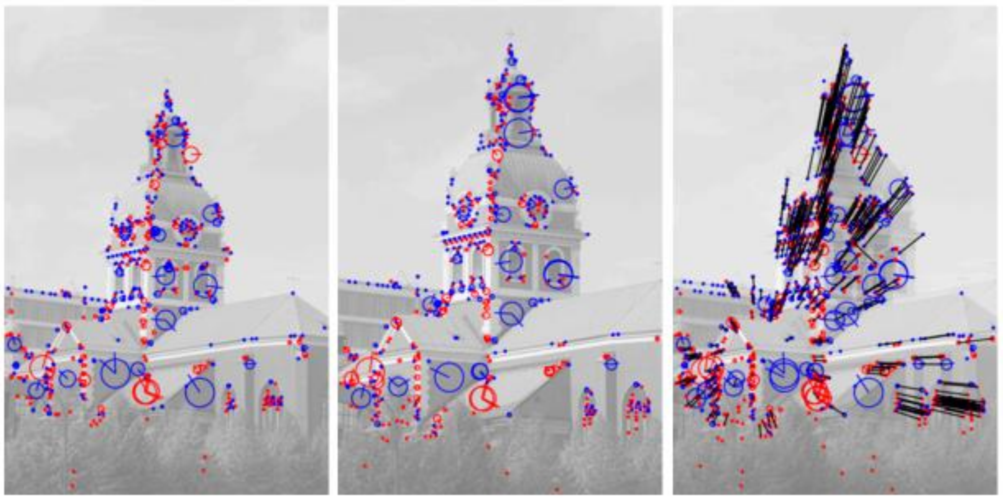


Figure 6: Interest Point Matching of Two Images with Different Scales

Fiducial Markers

Fiducials are small markers that are used in both human sight applications and computer vision systems to obtain data of an object or environment (Fiala, 2004). For computer vision, a code base utilizing a computer vision library such as OpenCV can read in the image and parse for the desired information. Fiducial markers are used in several industries, such as for industrial use, commercial use and other position tracking systems. An example of some types of marker patterns can be seen in *figure 7*.

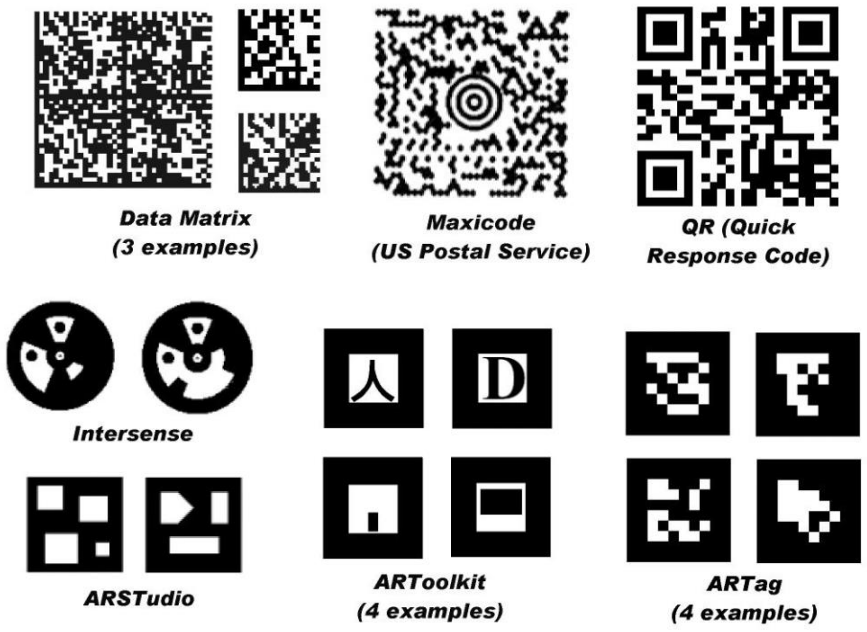


Figure 7: Several Types of Planar Pattern Marker Systems (Fiala, 2004)

Using custom fiducial patterns, it is possible to store information that can be queued or utilized by a fiducial. One such application of this feature is augmented reality. Through processing of the image for the fiducial, orientation and position of the fiducial with respect to the camera can be determined (Fiala, 2004).

The ARTag system is an example of a fiducial marker system utilized for generation of augmented reality (Fiala, 2004). This system utilizes a 36-bit code through use of a 6-by-6 grid within a bounding area. The values of the grid can be parsed and read in as a 32-bit value. The ARTag is first parsed using a form of object recognition, whereafter a quad is generated denoting the position and orientation of the identified fiducial (Fiala, 2004). From this, the value of the tag is stored at that location in the camera. A main application of this software is 3D model localization for augmented reality software. However, this could be applied to a prosthesis or robotic arm to determine the position and orientation of an object stored in the system.

ARToolkit is another fiducial-based marker system, utilizing a more varied set of input markers, such as symbols, letters and more abstract fiducial patterns.

The advantage of this system is that many assumptions about the object that we are visualizing can be made based off of the orientation and position of the fiducial tag with respect to the camera. This means that computer generated graphics could be rendered in real time overlayed in a recorded environment. This could also be used to approximate the position of an object given that the robotic system reading the tag can pull the associated object from a database. One potential issue with the markers in this use case is that, while accurate in finding the tags, there are still many assumptions being made on the system; it is assuming that the object always has a consistent geometry and does not deform. Though a safe assumption depending on the object being identified, this would limit the mechanism's capabilities of what types of object it could handle. If used in conjunction with more appearance-based, or feature-based, recognition systems, non-standard objects could be accounted for. However, that would require testing of the system to understand the feasibility.

Sensing Technology

Sensory Feedback

Sensory feedback is one of the most important functions of human body that allows a person to interact with or operate certain objects, which also help identify and determine the items by their shapes and surface, or engage in specific activities that require dexterity and force control. For most limb amputees, most prosthetic devices can sufficiently provide assistance for their everyday activities. However, without any feedback from prosthetic devices to the users, performing certain tasks that require sensory information is still proven to be difficult since they cannot predict the force exerted by the devices on an object (Talbot, 2014).

As the prosthetic technology is developed, many companies have invented sensors that allow users to receive sensory feedback from prosthetic devices. There are two types of sensory feedback that are currently used for prosthetic industries; invasive feedback and non-invasive feedback.

Invasive Feedback

Invasive feedback normally requires direct connection between the sensors and amputees' nervous system. This type of feedback provides accurate and realistic sense of touch to the users as the signals are sent directly through the nervous system to users' brain. A notable example of a device with this function is called cuff electrode, an implantation that encircle nerve bundles of particular limbs and stimulated based on signals from external sensors (e.g. pressure/force sensors) ("Amputees", 2014).

Non-invasive Feedback

Non-invasive feedback does not require any connection with users' nervous system or implantation in users' limbs. Instead, the feedback is provided to users by external equipment that delivers various forms of force or energy to users' body parts. There are several prosthetic devices that utilize this type of feedback, such as the BioTac® (Jiminez, 2014).

BioTac®

The BioTac®, a multimodal tactile sensor designed by SynTouch, LLC, is a device that provides feedback to the users through three different tactile displays (Jiminez, 2014). Each display is connected to a specific sensor of different functions; force, temperature, and vibration.

Force Display

For force or pressure feedback, the BioTac is equipped with air muscle, which reads pressure on particular spot and send the signal to force display, which is loosely wrapped around the test subject's upper arm during the experiment, to increase stiffness and produce squeezing force around subject's arm.

Thermal Display

The thermal feedback is determined based on the temperature detected by a thermistor on the BioTac device. This display utilizes a Peltier element to increase and decrease temperature at subject's skin, which is controlled by the signal from BioTac to produce +/- 3˚C changes in skin temperature and also maintain signal range to prevent any injury in user.

Vibration Display

Using the polyharmonic tactor, vibration feedback is measured from the intensity of vibration at the sensor, which is proportionally delivered to the subject through the vibration display.

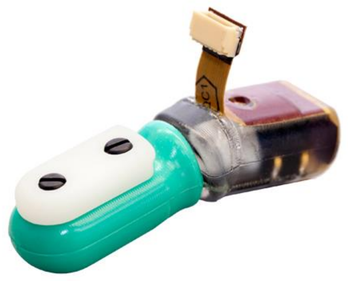


Figure 8: BioTac Multi-Modal Tactile Sensor

System Overview

Based on our previous research, we decided what functionality we wanted to incorporate into our system. Using the IRIS hand as a base design, a trans-radial prosthesis with six degrees-of-freedom in the palm and two degrees-of-freedom in the wrist was developed. The actuation of each finger was driven by a bevel gear system. One degree-of-freedom in the wrist used a belt and pulley while the other degree-of-freedom was actuated by a directly driven motor. We were able to measure the force exerted by each finger using the current spike from the motors. To achieve autonomous grip selection, a camera system located in the palm was used to detect AR markers corresponding to different objects.

Electrical Architecture

One of the major focuses of the project was to develop an electrical architecture that allowed for force sensing based on the electrical current in the motors when torque was applied. In order to determine how to best incorporate this in our system, we started by analyzing the electrical components used in the project last year to see which components could be recycled for our system.

Iris Hand Electrical Architecture

The IRIS hand electrical architecture involved a number of different pieces of hardware to achieve Communication between the different hardware, finger actuation, and object recognition.

Communication

At the highest level of the electrical system, the team used an Arduino Pro Micro board to communicate between the various components, and act as the brain of the system.

This board was responsible for sending signals that would actuate the motors, read in motor positions for positional control and force control, and interpret data from the object detection system. The Pro Micro communicated directly with a 16 channel Analog Multiplexer (MUX) which was capable of reading and writing to 16 analog inputs or outputs using only 5 pins from the Arduino.

Similar to the MUX, a 16 channel PWM driver was used to send out a total of 16 Pulse-Width Modulated (PWM) signals to the motor drivers which can then use those signals to drive the motors. The PWM driver allowed for these 16 signals to be sent simultaneously using only two pins on the Pro Micro that setup an I^2C communication using features of the Arduino IDE.

Finger Actuation

Utilizing both the MUX and the PWM driver, the finger actuation was achieved. As mentioned the PWM driver was able to send out a total of 16 PWM signals. The dual motor driver was connected to four of the PWM Driver pins. In order to drive a single motor two of the pins needed to be connected to the motor inputs for one motor on the dual motor driver allowing for speed and directional control of that motor.

The motors driven by the dual motor driver were the Pololu Micro-gear motors with a 1000:1 gear-ratio, allowing for high torque output from compact motors. These motors drove the series-elastic-actuated system by rotating the pulleys to actuate the four-bar linkages driving the fingers.

Though communication with the motor driver through the PWM driver allowed for both the speed and directional control of an individual motor, the scope of the project required control of the position of the motor. To achieve this the team used rotary potentiometers which would vary their resistance based off of the angle that their shaft was rotated.

These potentiometers were used in the series elastic actuation system to determine the position of the motor shaft and the spool connected to the finger allowing for calculations of the force on the fingers to be made based off of the offset between the two potentiometers.

Object Recognition Components

Due to the large amount of time it takes to process data from a camera, the team last year decided to incorporate the use of a second device to process the camera data, and then send it to the main processor. The PcDuino was an ideal choice because, it was a mini computer able to Run Linux allowing for the PcDuino to download the necessary software to communicate with the computer and run edge detection algorithms for object recognition. The PcDuino communicated with the Pro Micro using serial communication pins on both boards.

The last component in the object recognition system was the camera which sent data over to the PcDuino to be interpreted. The camera was stripped down to allow for it to fit within the palm.

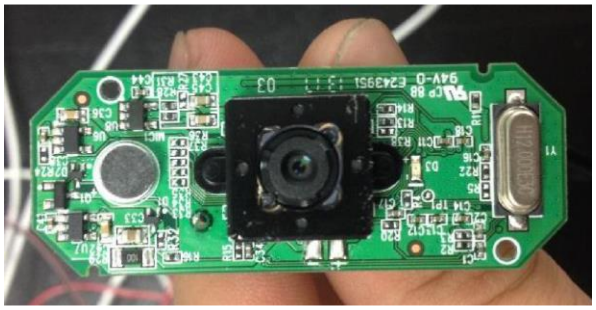


Figure 9: Stripped Logitech c525, Webcam

Analysis of IRIS Hand System

The Iris Hand electrical system components all were logical choices for the desired task. One issue the team had with their motor drivers was that they would burn out when the motors stalled. Upon further investigation we learned that the stall current of the 1000:1 pololu micro motors was 1.6 amps (reference pololu page) which is greater than the free run current of their motor driver which was 1.2 amps (reference pololu page) with a peak of 1.5 amps. We theorized this discrepancy was the main reason for their motor drivers burning out. We also saw that the old camera used in the arm was no longer functional. Majority of the components in the electrical system were all still fully functional, so with the change of the motor driver, addition of some form of current sensing, replacement of the old camera, and the addition of a higher torque motor for the wrist roll rotation, we were able to create the electrical architecture for the VIPeR Arm.

VIPeR Arm Electrical Components

The first component we looked for was our new motor drivers. Our main considerations for the motor drivers were for them to be affordable and have a free run current higher than 1.6 amps. We also needed to figure out how to read the current from the motors. We researched a few motor drivers that pololu offered and found some potential motor drivers that met our specifications. The first motor driver we looked at was the DRV 8838 single motor driver.

This motor driver could drive a single motor using two inputs, and could run a motor with a continuous current of 1.7 amps and a maximum current of 1.8 amps. This system also could easily be integrated with the PWM driver, was small, and very cheap at roughly $3.00 a board. The only issue with the system is that we would need to add some sort of external current sensing circuitry to enable the current sensing. After researching we found that current sensing could be accomplished using a current sensing resistor or a circuit involving a differential amplifier and a resistor. Both options involved increasing the amount of hardware in our system, so we researched to see if we could find a motor driver with on-board current sensing. We found that pololu offered some boards with current sensing capabilities, and found the MC33926. This board was significantly more expansive than the DRV 8838, being around $30.00 a board, but could drive two motors simultaneously. Since we would need other components for the DRV 8838 which would add cost and complexity to the circuit, this board looked like a viable option and provided a lot of on-board protection to prevent the components on the board from being damaged. Due to it having on-board current sensing, being able to drive two motors, and supporting high currents being able to run continuous currents of 3 A we decided to choose this board. The other benefit was we were able to run a motor that draws a large amount of current which we would need for the forearms roll rotation.

We found our final motor for the forearm roll rotation by searching for a motor with a high stall torque able to run high payloads since the entire weight of the palm and forearm would be on this motor. The motor also needed to have a stall current lower than the continuous stall current of the motor driver. We found the Pololu Metal Gearmotor with a 227:1 gear ratio would be adequate since its stall current was 220 oz-in and its stall torque was 2.2 amps.

Due to the camera from the project last year not functioning, we replaced it with the Logitech c270 webcam, which had near identical features, functionality, size, and cost from the camera used last year.

Images for all electrical components can be found in appendix F.

VIPeR Arm electrical Architecture

Due to all of the electronics in our, system, the circuit requiring all of them to communicate involves multiple connections. The diagram below shows the number of connections involved in the circuit excluding power and ground which are connected to an external power supply that runs the system.

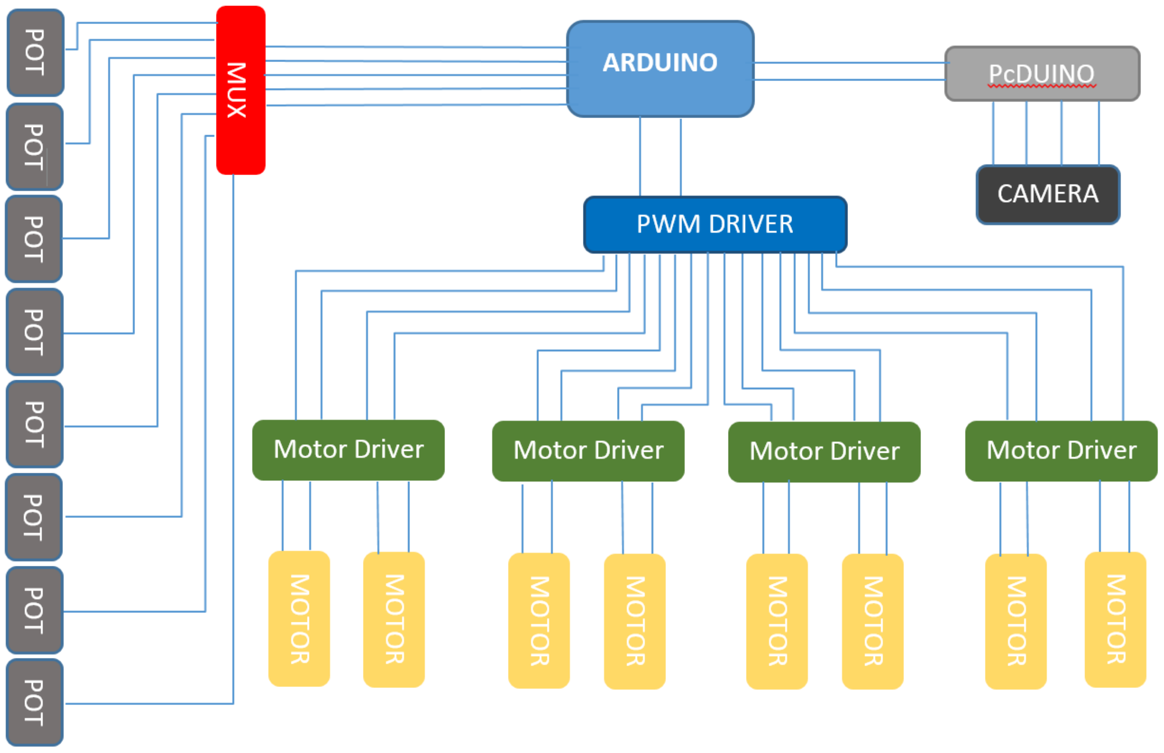


Figure 10: Electrical Architecture for VIPeR Arm

Each component integrates into our electrical system to create our control loop that actuates the fingers based off of the AR tag recognized the camera. The system diagram below shows the full functionality of the system driven by the Arduino main processor

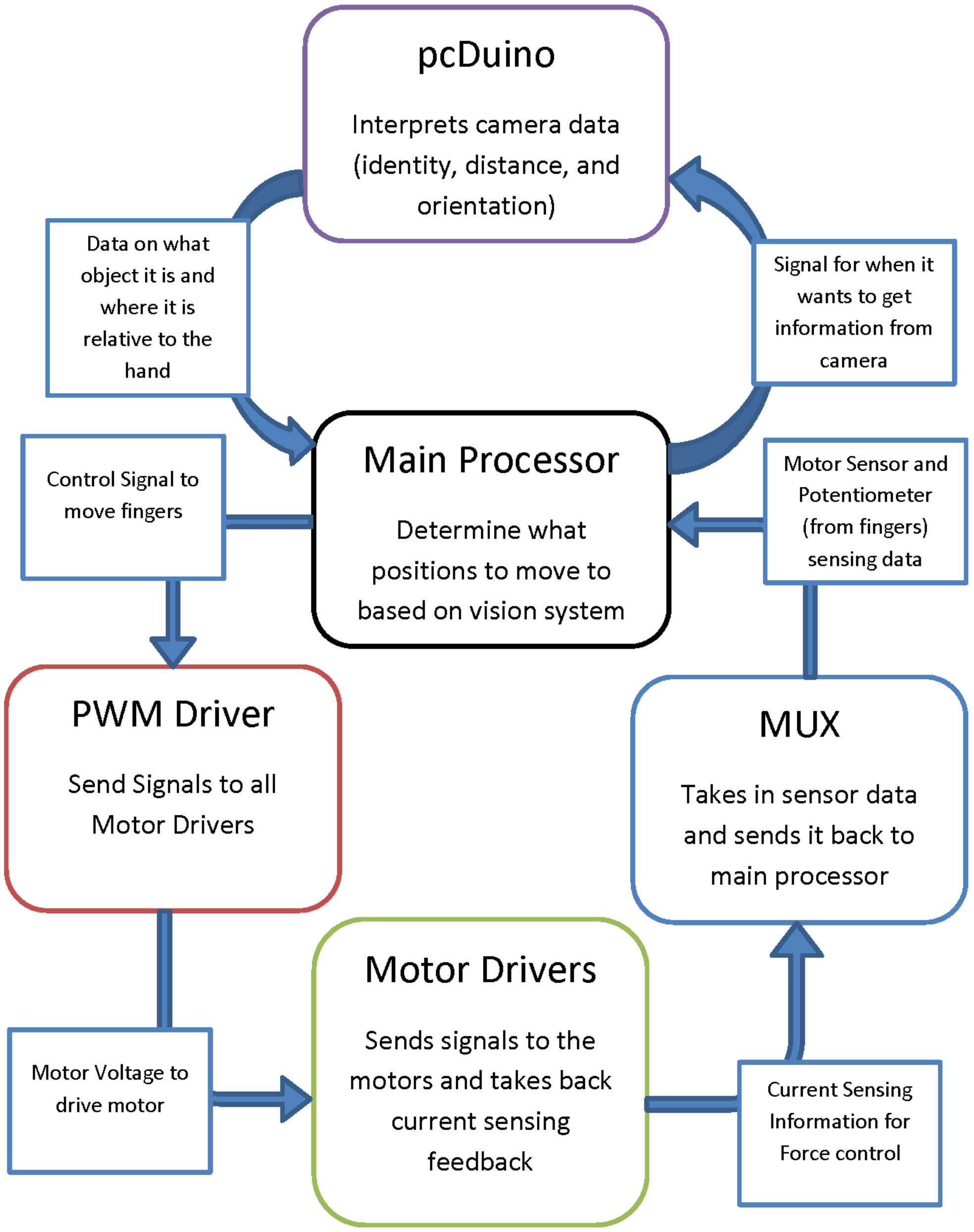

pcDuino
Interprets camera data (identity, distance, and orientation)
Data on what object it is and where it is relative to the hand
Signal for when it wants to get information from camera
Main Processor
Determine what positions to move to based on vision system
Control Signal to move fingers
Motor Sensor and Potentiometer (from fingers) sensing data
PWM Driver
Send Signals to all Motor Drivers
MUX
Takes in sensor data and sends it back to main processor
Motor Drivers
Sends signals to the motors and takes back current sensing feedback
Motor Voltage to drive motor
Current Sensing Information for Force control

Figure 11: System Diagram of the Electrical Architecture

The physical connections between the system were designed with the intention that the components could all be easily replaced if we were to have issues with a component failing. In order to achieve this we designed a PCB board in Ultiboard that would use female headers on the PCB board and male headers on the different components allowing for easy replacement of the various components. This design would have easily allowed for us to replace any components that broke, but was too large to fit in our forearm where we wanted to house all of our components. To remedy this we found some small prototyping boards (protoboards) which we could solder connections directly on too. These protoboards were small enough to slot in the forearm which allowed for us to not only remove components attached via female and male headers, but also remove the circuit entirely if one of the soldered connections broke off. An image of the developed PCB can be seen in appendix E.

After incorporating this into our electrical design we created mechanical models for all of the components including the female and male headers and created the forearm model with the intention of having these boards able to slot in and out as mentioned.

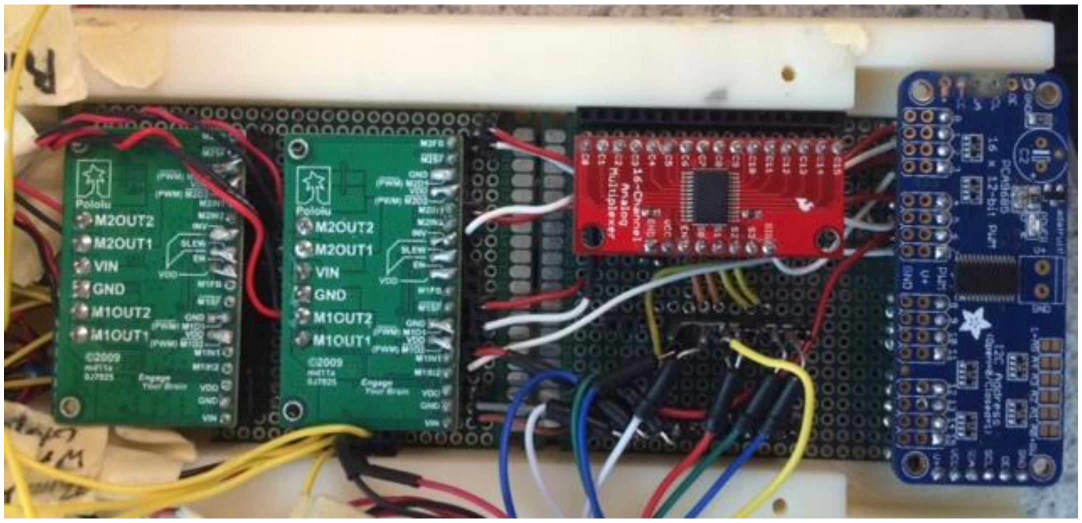


Figure 12: Electrical Circuitry

Mechanical Design

The focus of this track was to take the existing IRIS hand and modify it in a way so we could incorporate a two degree-of-freedom wrist (Casley, 2014). This addition makes the prostheses more anthropomorphic and allows for a larger variety of object orientations when grasping.

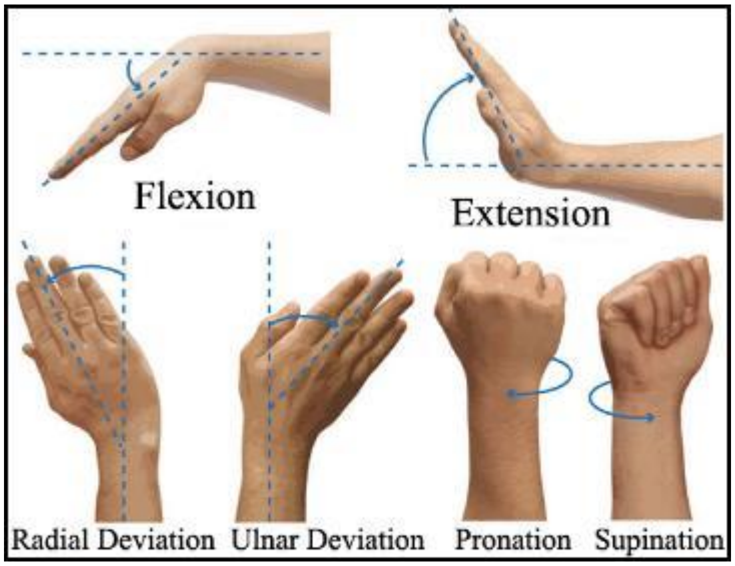


Figure 13: Wrist Degrees of Freedom

In *figure 13,* the flexion and extension (top) of the wrist is known as the "pitch" degree of freedom, the deviations (bottoms left) are known as the "yaw" DOF and the pronation and supination are known as the "roll" DOF. The two movements of the wrist that we included were both pitch and roll. With only these two additional rotations, we could orient the palm to face any desired object or surface while keeping the base of the forearm stationary. Yaw was not incorporated due to lack of space for all actuators and added complexity for both design and control. To incorporate the pitch and roll degrees-of-freedom, some changes to the old design needed to be made though there were some aspects that were kept more or less the same as the previous project, the IRIS hand.

Finger Design

One concept that was able to carry over from the previous project to this one was the finger design. This design is very important in giving the hand an anthropomorphic appearance. Not only is the appearance similar to that of a human, it also mimics the movement of a real finger the way it naturally curls while grasping different things. For this to be accomplished, two kinematic four-bars were coupled together by attaching the rocker of the first to the crank of the second. The design can be seen in *figure 14* which shows the cascading linkages.

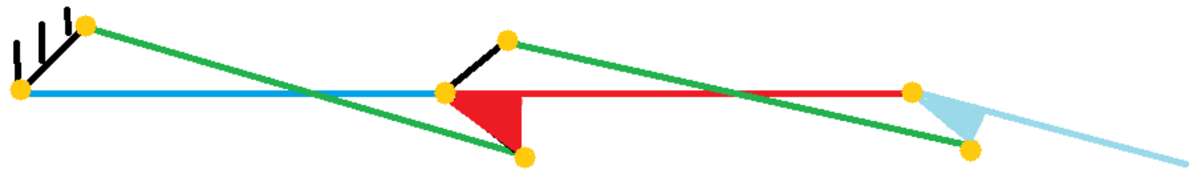


Figure 14: Finger Four-bar Kinematic Linkage Diagram

Last year, the team made different length fingers which meant each finger housed a unique linkage varying only in link lengths. This year we changed it to be modular between the four, one degree-of-freedom fingers (index, middle, ring and pinky) so that all the fingers were the same size and could easily be replaced. If one of the linkages were to break, a new one could easily be laser cut from a 2-mm thick acrylic sheet.

Design Options

Before settling on a final design to move forward with, there were several options that were considered which were all alterations of the previous hand design. Each was modified in a different way to achieve the addition of a two degree-of-freedom wrist while still keeping the actuation of the fingers the same.

Option 1

The first option that was explored kept the main force sensing and actuating concept of the old project the same with the series elastic actuation method (Casley, 2014). This case of series elastic actuation uses ropes and springs to move different components in order to effectively sense the force being applied at the end effector. In the old system, there were two strings acting as tendons that attached to each finger; one at the top and one at the bottom. These strings were the only manipulators of the fingers. If the bottom string was pulled in, the finger curled and vice versa. These strings were attached to springs that were housed in the forearm and the springs were connected to a spool using more string. This spool is finally what is actuated by the motor. *Figure 15* shows a picture representation of how the series elastic actuation system works.

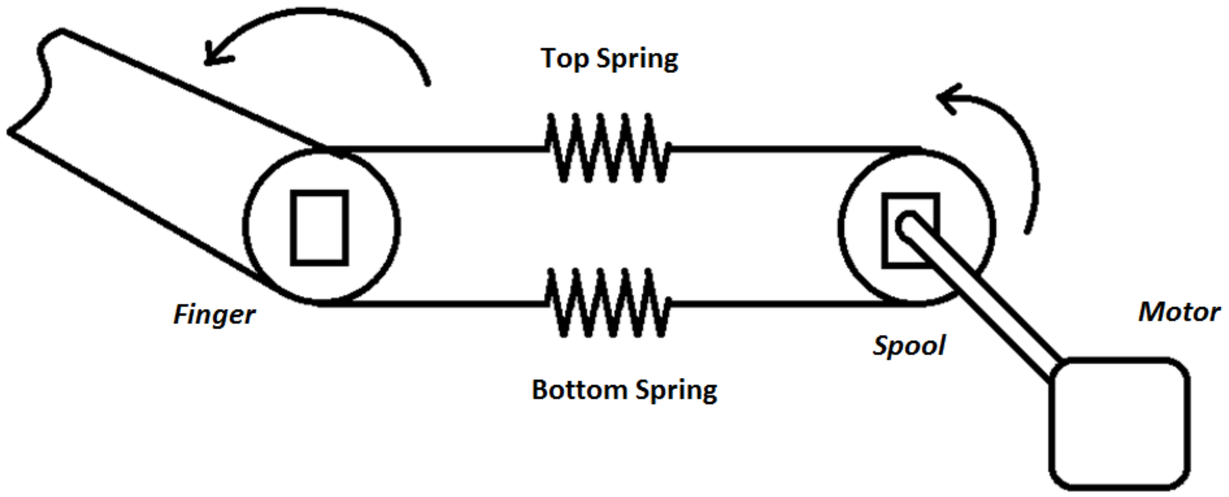


Figure 15: IRIS Series Elastic Actuation System (Casley, 2014)

In order to determine how much force is being applied at the end effector (in this case the fingertip), two rotational potentiometers are used; one at the motor and one at the base of the finger. When a force is applied, the string and spring system becomes displaced. Using the two potentiometers, a relationship can be formed, which directly correlates to the force being applied.

Though incredibly effective, the series elastic actuation method takes up space from the finger to the forearm which means that as the wrist's pitch changed, the string/spring system would also be displaced. To maintain the same length of tense cabling while rotating the wrist pitch, an intermediate set of spools was added to the design. These intermediate spools would be attached to a shaft that would rotate concurrently with the pitch rotation allowing for consistent tensioning during wrist actuation. *Figure 16* shows the intermediate spool design in the entire assembly.

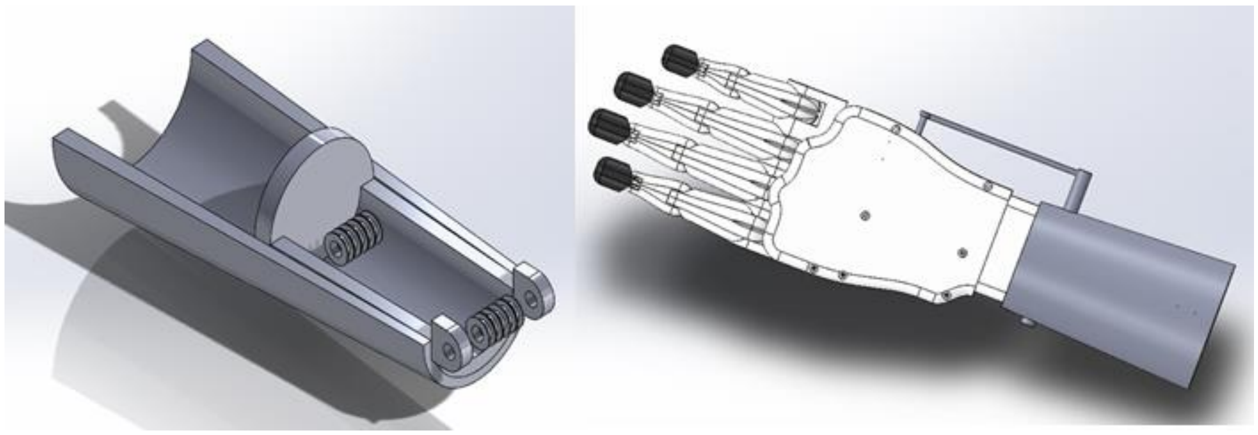


Figure 16: Design Option 1: Linear Series Elastic Actuation

Option 2

The second design option still incorporated series elastic actuation though in a different form. The previous system used a linear spring to do the series elastic actuation while this design uses a torsional spring. The spring is placed at the base of the finger which still allows for some elasticity in the system. As the finger is being pushed, the torsional spring will be actuated which will again allow us to determine the amount of force exerted at the fingertip. *Figure 17* shows the preliminary design of the second option.

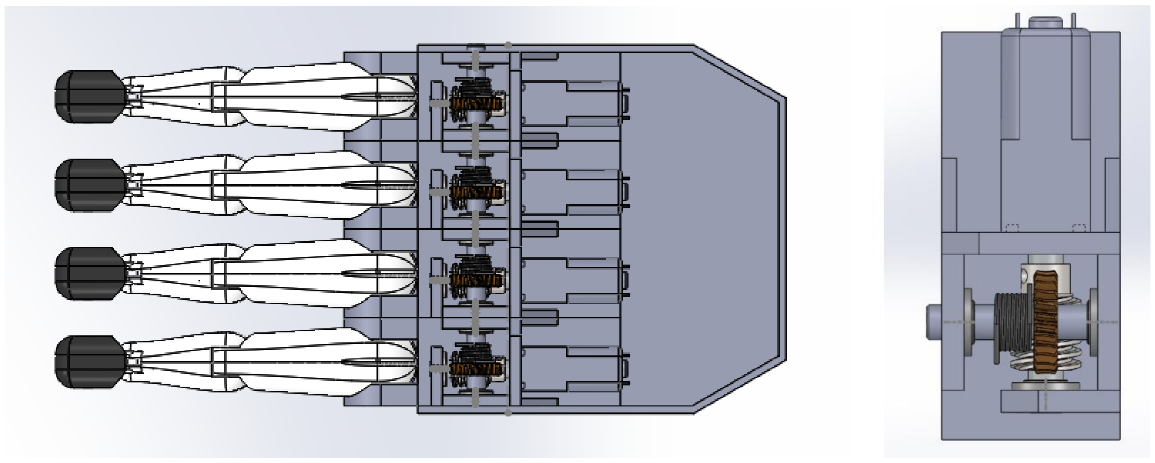


Figure 17: Design Option 2 - Torsional Spring Series Elastic Actuation

This design used a worm gear connected to a spur gear which drives the finger directly meaning the gear would be directly attached to the finger itself. The thumb would then be actuated using a similar system leaving no wires other than the motor and potentiometer wires to come through to the wrist. This would allow for easy actuation of the wrist and there would not be any strings needing to flow through which would make the wrist actuation straight forward.

One other reason to use the worm gear system is that they are not back-drivable. This means that once the motor drives the gear to its desired position, there is no way for the system to actually move the motor and gear. There are two main drawbacks to this approach, both involving the size of things. It would be difficult to find a torsional spring small enough to fit around the shaft but not contacting the worm gear while providing enough of a range of the spring. The size of the finger housing would have to be decided based on the size of the spur gear in addition to the size of the worm gear. Though it is a viable option, this would not fit the criteria of keeping the hand anthropomorphic.

Option 3

The third and final design option included a bevel gear driven system. This system also condensed the finger actuation by moving the motor up by the knuckle. *Figure 18* shows the bevel gear driven finger and knuckle assembly.

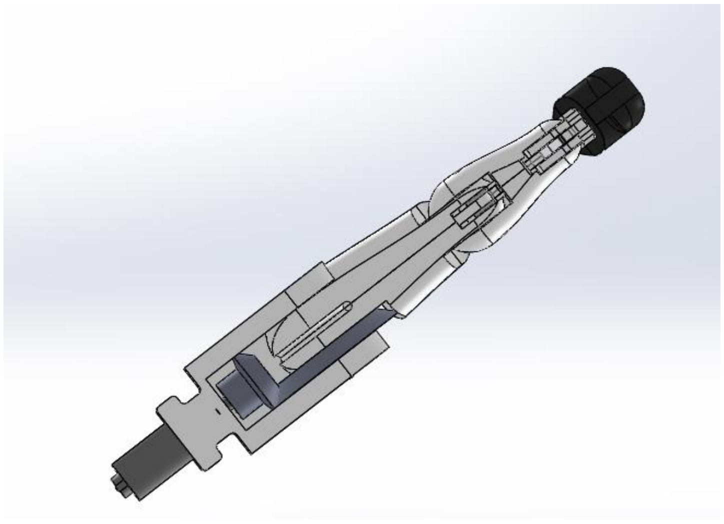
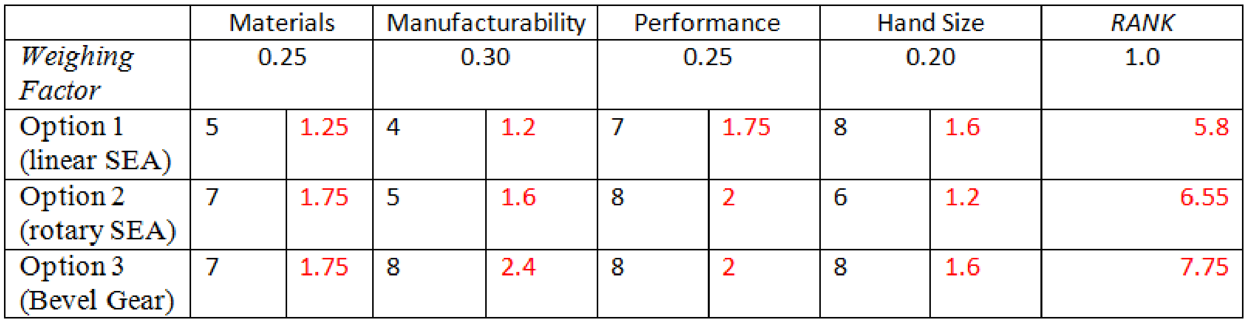


Figure 18: Option 3 - Bevel Gear Driven Finger System

Since a bevel gear size was not chosen at this time, the two bevel gears, seen as the grey figures behind and next to the finger, were roughly modeled. The bevel gear is fixed to the 3D printed finger case so as the motor turns the gear pinion which turns the gear, the finger rotates with it. This design also has the advantage compressing all manipulators and actuators of the fingers inside of the palm casing which allows for an easy addition of the degree of freedom to the wrist.

After coming up with the three different designs a decision matrix was made in order to decide which design would make the most sense to move forward with.



	Materials		Manufacturability		Performance		Hand Size		*RANK*
Weighing Factor	0.25		0.30		0.25		0.20		1.0
Option 1 (linear SEA)	5	1.25	4	1.2	7	1.75	8	1.6	5.8
Option 2 (rotary SEA)	7	1.75	5	1.6	8	2	6	1.2	6.55
Option 3 (Bevel Gear)	7	1.75	8	2.4	8	2	8	1.6	7.75

Figure 19: Decision Matrix for design

The materials column rates the designs based off of the raw materials needed in order to make the system work. The first design needed two sets of spools (one for the motor connection and one for the intermediate between the forearm and hand), string, and springs for each finger thus getting a lower grade than the other two options. The second option needed only the worm gear and the spur gear in order to make the system work. The third design similarly only needed the bevel gears to operate the system.

The manufacturability column explored how easy it would be to make each option. The linear series elastic actuation option would need custom spools to be made. It would also need a larger forearm section in order to house all the electronics needed for the system. The rotary series elastic actuation option needed for a custom housing for the worm gear as well as the motor in order to make it. There would also need to be modifications to the current finger to put the spur gear in the finger which would have moved the four-bar linkage to another spot. The third option needed only to shave down the side of the finger and readjust the current knuckle in order to house the bevel gear system which is why it received the highest rating.

The performance column helped determine which option would be efficient computationally. The only real difference in all the systems was the additional code that was needed to compensate for adjusting string tension when the wrist's pitch motion is actuated in option one.

The final column is the hand size in general. For the first and third options, the hand size could stay more or less the same as the old hand which was modeled roughly based off of the average male adults hand. The middle design however, would have to be larger as previously stated.

Based off the decision matrix, we chose to further develop design 3, the bevel gear driven system, since it had the highest overall ranking.

Preliminary System Designs

The first full hand design derived from option three of the decision matrix was a very crude design that did not completely look like a human hand. This design, seen in *figure 20*, used all the same length of fingers and since there was no offset, each finger would reach the same length.

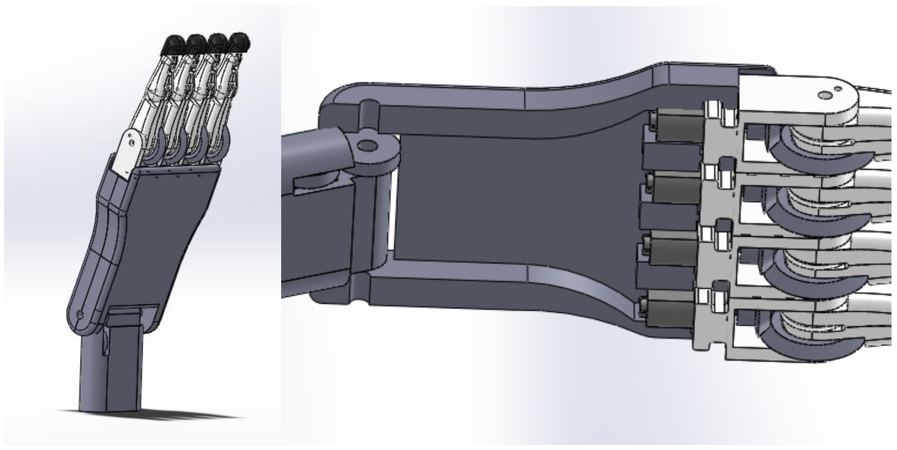


Figure 20: Preliminary Design for the Hand

This system also did not have the full wrist system. It only had one of the two degrees of freedom determined. The original design also used two large spur gears to actuate the pitch of the wrist. The wrist was also not anthropomorphic thus another version was necessary.

In the next iteration of the design, the wrist was made to include both degrees of freedom. As seen in *figure 21*, the prosthetic hand was modified to look more like that of a human hand.

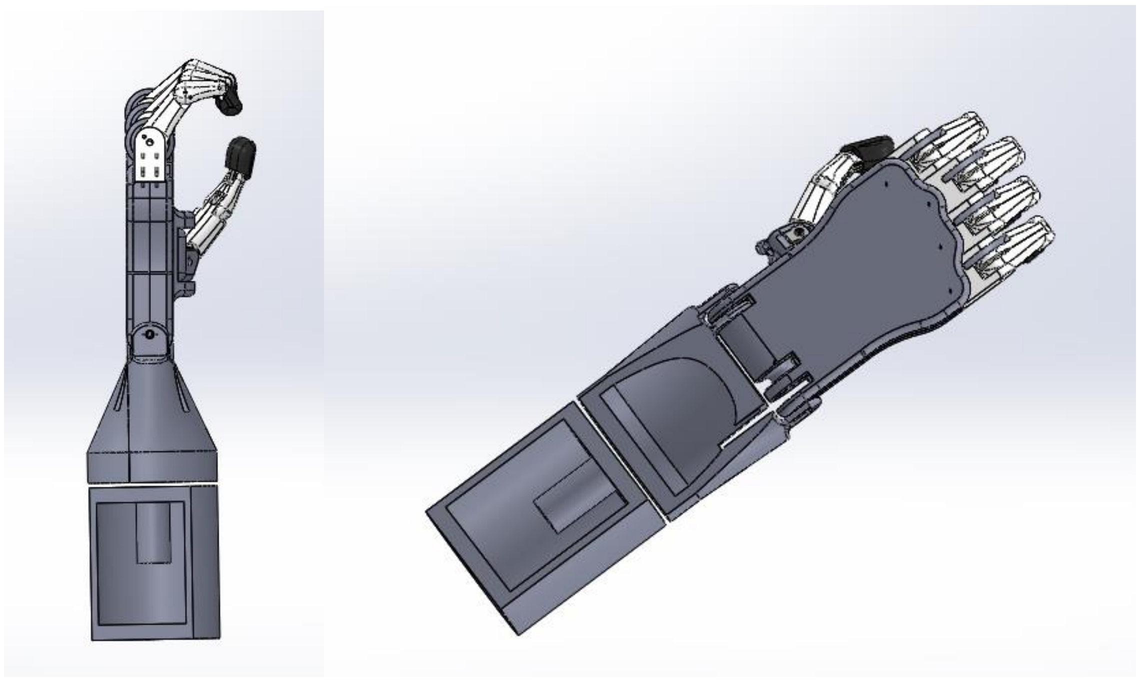


Figure 21: Second Iteration of the hand design

An offset for the fingers was determined in order to give it the illusion that the fingers were different lengths. In this design, the thumb design was kept the same as in the IRIS hand; the rotational degree of freedom was actuated by a motor inside the 3D printed thumb housing and the curling of the thumb was actuated through series elastic actuation. The motor for this was housed in the palm case. The issue with this is that the spool would not be parallel to the string, so there would be a potential that the system would fail. The IRIS hand's thumb can be seen in *figure 22*.

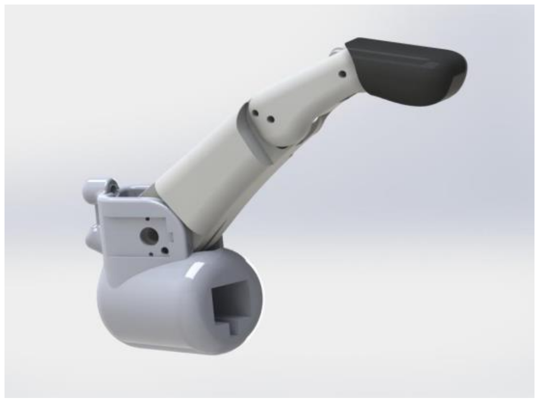


Figure 22: IRIS Hand Thumb Design

In addition to the thumb, the "roll" of the wrist was determined to be directly driven by a motor. The housing for the electronics would be shared within the forearm which was subject to change once the size of all electrical components was determined.

Final Design

The final design consisted of three major parts that were redesigned from older models; the hand case and fingers, the thumb, and the forearm.

Finger Design

There was only one change that was needed in the finger design. The side that contacts the bevel gear needed to be modified in order to accommodate the gear itself. The teeth of the gear was interfering with the side of the finger so an impression was made on the finger which can be seen in *figure 23*.

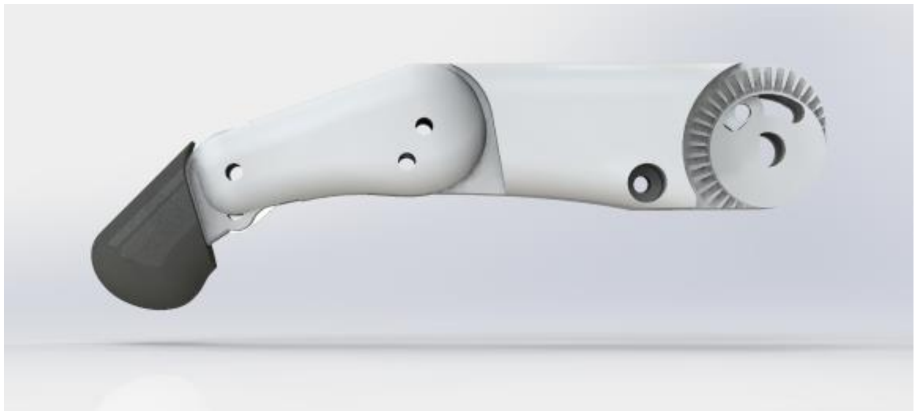


Figure 23: Finger Modification to Accommodate Bevel Gear Teeth

Thumb Design

The next modification that was made was the thumb. The thumb had to be completely redesigned because the curling motion became bevel gear driven. This meant that the motor housed inside the thumb had to be changed from rotating the entire unit to being the motor used to drive the curling motion. To achieve this, an assembly of two separate parts were needed to determine the gearing necessary. The housing holds a motor in place that has a mitre gear attached to it. This mitre gear is simply used to turn the intermediate shaft between the motor and the bevel gearing system. To actually manufacture the entire gearing system, the thumb needed to be redesigned into two different parts that easily slide in place to mesh the gears. A picture of the system can be seen in *figure 24.*

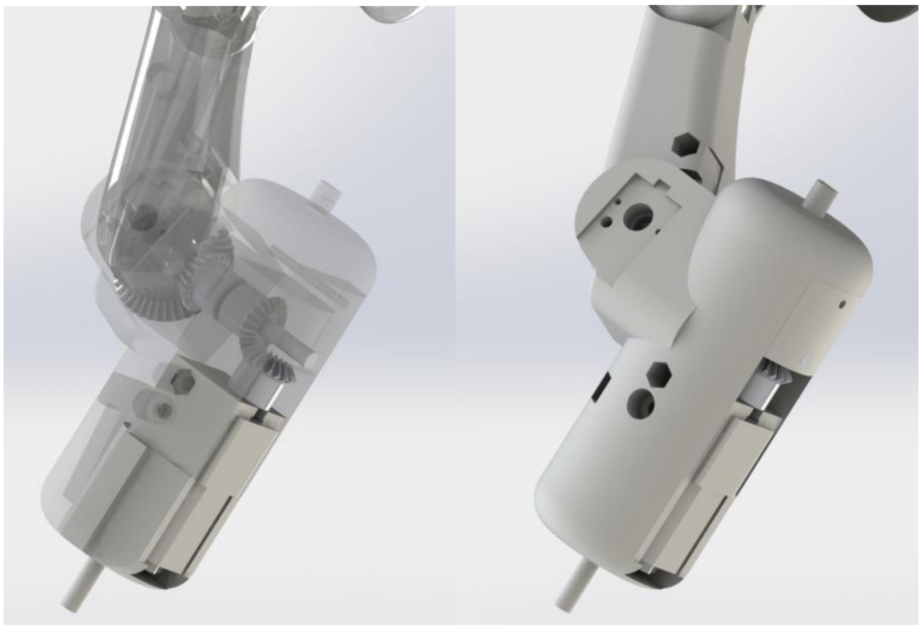


Figure 24: Thumb Curling Actuator

To rotate the thumb, another motor is mounted to the palm of the hand. This also uses a mitre gear system where the other gear is attached to the bottom of the thumb casing. *Figure 25* shows the motor and thumb assembly in the palm.

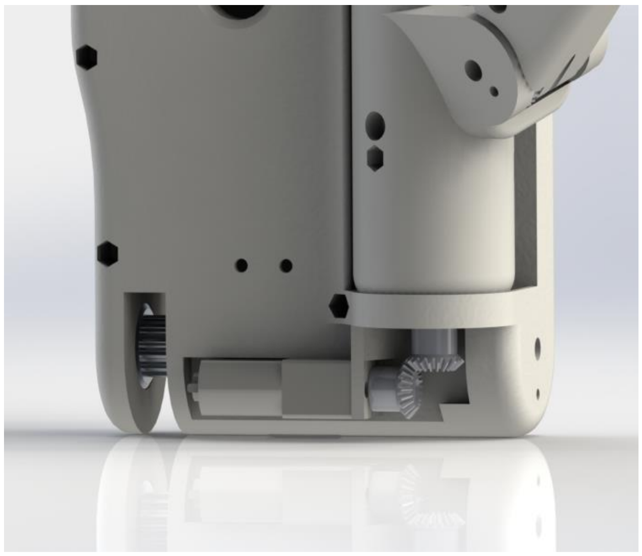


Figure 25: Thumb Rotational Actuator

Palm Design

Next the palm was designed. The case had offset knuckles and a rounded top of the case in order to give it a more realistic look. A small hole in the bottom of the palm was made for the camera to fit in. The slot for the thumb was made larger as well to fit in the new thumb design. *Figure 26* shows the final palm casing design.

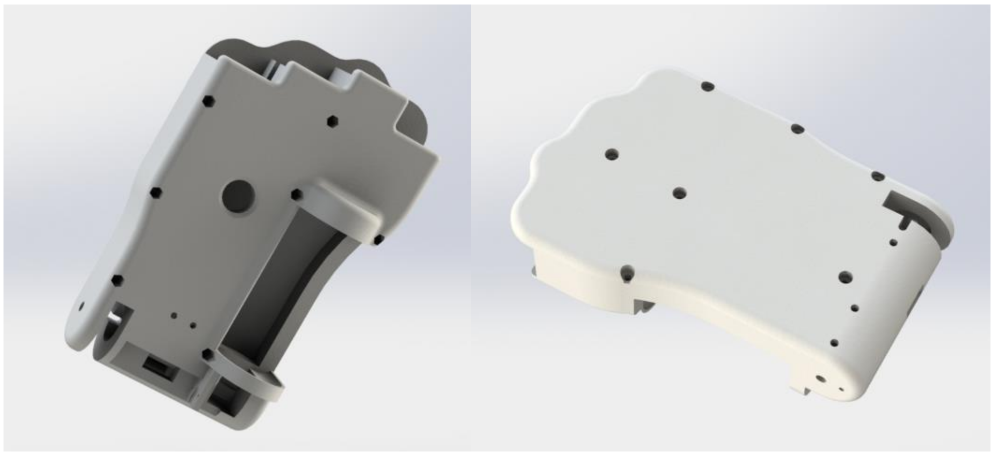


Figure 26: Hand Case Design

Forearm Design

The final part of the mechanical design is the forearm. To actuate the pitch motion of the arm, a belt and pulley system is used. One pulley is rigidly attached to the shaft that is in between the forearm and the hand. The other pulley is on a motor shaft which is mounted just under the top of the forearm. In order to ensure the hand rotates with the shaft, a pin goes through one side of the palm, the shaft and partially through the bottom of the palm. To ensure that the system does not try to continue moving past human limitations, a physical stop was applied to limit the motion. This physical stop (*figure 27*) limits the range to be 90 degrees flexion and 70 degrees extension, similar to a human hand.

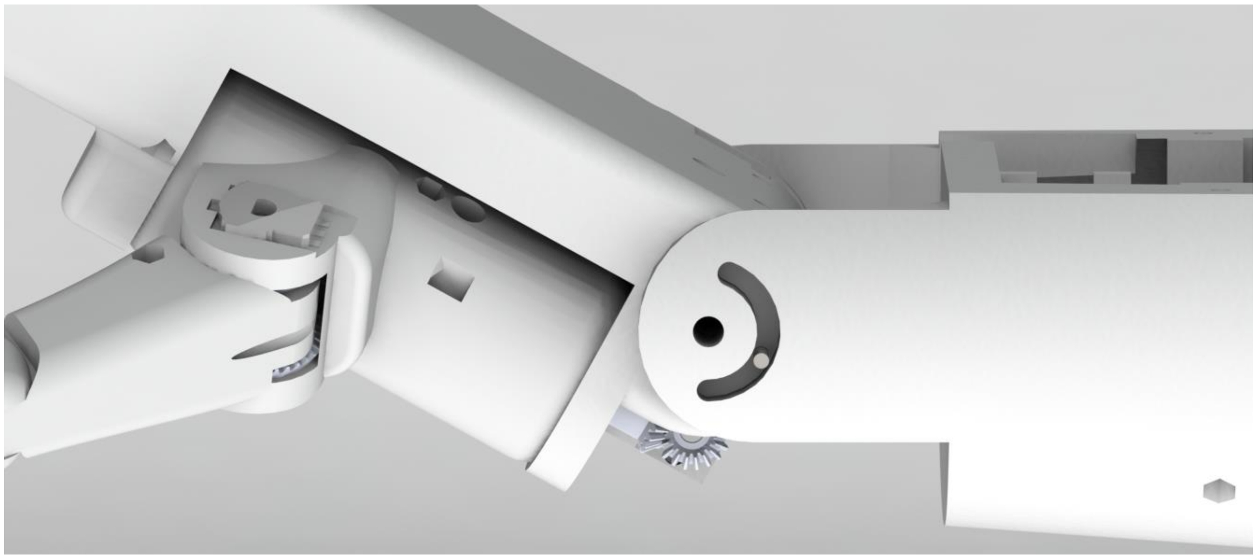


Figure 27: Pitch Motion Physical Stop

The physical stop was made using a slot and pin. The slot is made cut directly into the wrist while the pin lies in a hole that's on the hand case. As the hand moves, the pin will move within the slot until it reaches one of its physical limits. These limits could also be made in code just to ensure that the hand never tries to move too far in a certain direction.

To actuate the roll of the wrist, a motor is directly driven to rotate the entire forearm. A small slot allows for the motor and potentiometer wires to be channeled to the main cavity of the forearm where all electrical hardware is housed. There is a physical stop on both wrist degrees-of-freedom.

The final aspect of the design is the housing for all electronic components. The electronics are connected to each other using matrix boards. Knowing the size of the matrix boards, two slots were designed for them to slide in; one below the pcDuino and one above.

Two cylindrical protrusions were also made to ensure that the pcDuino had a place to fit in as well. Finally, two slots were cut on either side of the forearm to dissipate some of the heat that could build up from the electrical components. *Figure 28* shows the final wrist design with all electrical hardware housed inside.

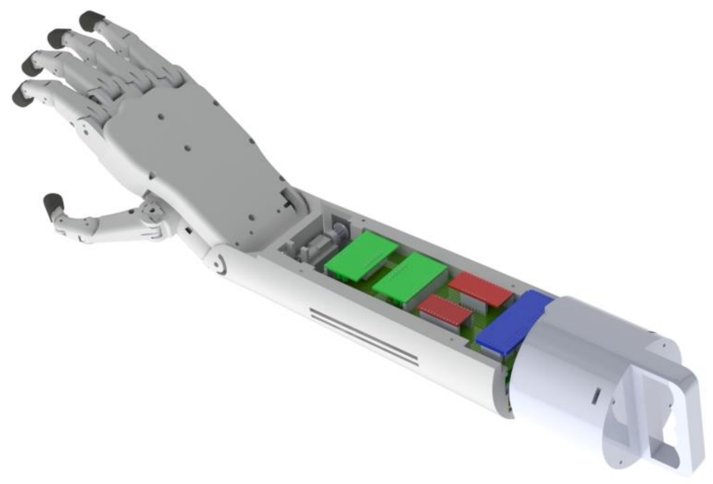


Figure 28: Forearm and Hand Final Design

Force Analysis

An analysis was conducted on the cascaded four-bar linkage finger design to calculate the force exerted by a finger based off of the motor torque. To analyze the finger, we utilized a mechanical simulation tool in MATLAB called SimMechanics and created a model representing the finger. In SimMechanics, the cascaded four-bar linkage was simulated using a combination of rigid bodies to represent the links and revolute joint blocks acting as the joints connecting the links. The SimMechanics model of the finger is shown below in *figure 29*.

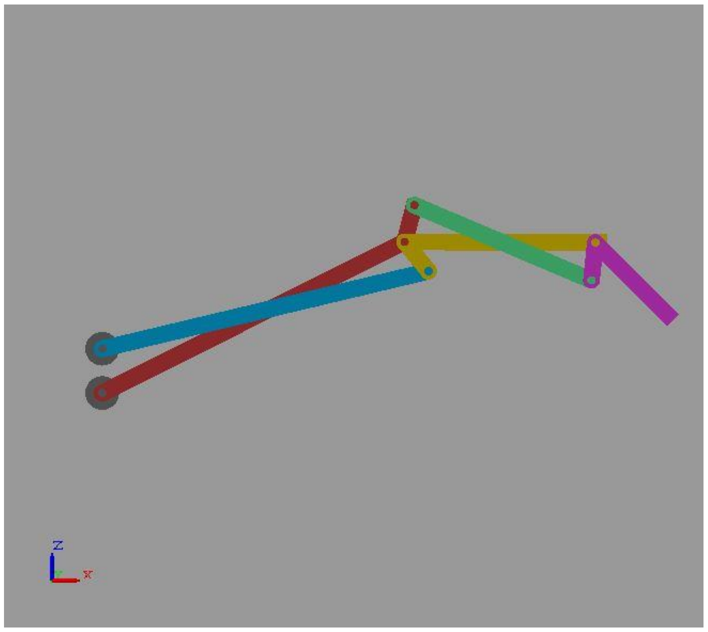


Figure 29: Cascaded Four-bar linkage representation of finger

We were able to apply arbitrary forces on this model which we used to determine the torque on the driving motor. Using constant forces applied to the fingertip, we applied force to the fingertip analogous to that of an average male's gripping force for one finger to determine the average torque applied to the driving joint of the cascaded four-bar, where the second gear in the bevel gear system would drive the finger's curling rotation. Adding in a spring component to the SimMechanics model allowed for us to simulate a more realistic human hand grasp since the force applied to the fingertip adjusted depending on the position and orientation of the finger.

Another approach we took to determine the force and torque relationship involved a mathematical approach using both the vector loop method of four-bar linkage analysis and the Jacobian Matrix. Because our linkage configuration was a cascaded four-bar linkage, it was a connection between two four-bar linkages, the first driving the second. The Vector Loop method of positional and velocity analysis for a four-bar linkage allows for complete analysis of a linkage based on the link lengths of the four links and the angle of the driven joint (Norton, 2011). Computing the positional vector loop equation for the first four-bar linkage allowed for us to determine the driving angle for the second four-bar. Using the driving angle for the second four-bar we computed the position of the link representing the fingertip.

With the values for all of the joint angles and positions calculated, we could then compute the velocity vector loop equations for the cascaded four-bar linkage. After we found the linear and angular velocities of each link in the cascaded four-bar linkage we were able to use a Jacobian Matric which relates the angular velocity to linear velocities. A general Equation for the Jacobian is shown below:

$$V = J * \omega \tag{1}$$

Where V is a 2x1 matrix representing the x and y values of the linear velocity, J is a 2x1 matrix representing the Jacobian Matrix used to relate Linear and Angular Quantities, and ω is a 1x1 matrix representing the angular velocity of the driven joint of the cascaded four-bar. The Jacobian is a mathematical tool used to relate linear and angular quantities, since we want to relate the input torque to the Force we can use the Jacobian for this relationship with the following equation:

$$F = J * \tau \tag{2}$$

In this equation J is the same as before, τ is a 1x1 matrix representing the torque of the driven joint, and F is a 2x1 matric that tells the x and y components of the force applied to the fingertip.

With this relationship, the gear ratios of the motors, and the current feedback of the motors, we can relate the motor torque to the force of the fingertip.

After calculation of the Jacobian equation, we could relate the torque at the input of the cascaded four-bar linkage to the forces at the fingertip, but we still needed to find a way to relate the voltage signal that the Arduino gets in from the analog-read function from the feedback pin on the motor driver to the torque driving the four-bar.

To compute the overall relationship between the motor torque and the force on the fingertip, we needed to find the relationship between the current sensor reading, as an analog input to the Arduino and the torque at the input to the cascaded four-bar linkage. Once we have that relationship we can simply multiply our analog reading from the Arduino by a conversion factor to get the input torque to the four-bar.

The first relationship we need is the relationship between the motor torque and the current read by the motor. This is an easy calculation since the motor constant is equal to the Torque of the motor divided by the current (Voss, 2007). The pololu micro metal gearbox motors with the 1000:1 gear rations have a maximum torque of 125oz-in and a maximum stall current of 1600mA or 1.6A. Using these two quantities we calculated the motor constant:

$$K_T = \frac{\tau}{I} = \frac{125}{1.6} = 78.125 \tag{3}$$

Rearranging this equation we can find the current with respect to the torque by incorporating the motor constant:

$$I = \frac{\tau}{78.125} \tag{4}$$

The motor drivers give a current feedback in the form of a voltage where 525mv or 0.525V is equivalent to an Amp of current. This means that:

$$V * 0.525 = I \tag{5}$$

If we want to solve for the voltage we get:

$$V = \frac{I}{0.525} \tag{6}$$

The Next step in our equation for the conversion is to find out the relationship between the analog read and the supplied 5 volts to the signal. The Analog read function outputs a value from 0 to 1024, where 0 is 0 volts and 1024 is 5 volts. Because of this we can make a simple equation to relate the analog read value to a voltage:

$$V = \frac{AnalogRead}{1024} * 5 \tag{7}$$

Solving for the analog read value yields:

$$AnalogRead = \frac{V}{5} * 1024 \tag{8}$$

Combining this equation with the other equations yields the full relationship between the analog read value and the torque at the driving joint of the four-bar:

$$AnalogRead = \frac{1024 * \tau}{5 * 0.525 * 78.125} = 4.99 * \tau \tag{9}$$

Since there is a 3:1 gear ratio from the motor output shaft bevel gear to the bevel gear at the input angle of the four-bar, the relationship between the input angle and the analog read value is:

$$AnalogRead = 3 * 4.99 * \tau = 14.98 * \tau \tag{10}$$

Lastly, by solving for the torque we can find an equation we can use to calculate the torque in oz-in based off of the analog read value:

$$\tau = \frac{AnalogRead}{14.98} \tag{11}$$

With this relationship, when we read in values from the analog read function within Arduino we can calculate the output torque at the input the four-bar linkage and then put the output torque into the Jacobian we derived earlier. With that we will be able to find the force the fingertip is exerting in both the x and y direction.

Object Recognition

The vision system for the proposed hand must be able to accomplish a number of tasks to adequately support the controller and the user in grasping and tracking objects. The first task is to identify the object that is in the camera's point of view and differentiate this object from its surroundings. In this sense, it should not misidentify parts of its environment as desired objects.

The second task for the vision system is to determine the position of the object we wish to grasp in relation to the camera and hand. This information can be required by the wrist to make the necessary joint angle calculations for keeping an object in sight of the hand.

The third task for the vision system is to send this identification and position information to the main processor in a timely manner. It was unclear at what rate would be optimal for our system, due to the variability in hand speeds when moving to grasp an object, so an evaluation of the object identification speed will be made relative to responsiveness achieved by the previous iteration of the hand.

AR Marker Detection System

The selected approach for object identification is the use of an augmented reality (AR) tag system. There are multiple open-source libraries available for implementation, and unique identifiers for each tag allow for simplified object identification. The abstract patterns of the tags also aid in reducing the likelihood of identifying a false object. Drawbacks to using this system are that now all objects are required to use an AR tag, meaning that heavy environmental modifications will have to be made for prosthetic users to utilize the system as efficiently as a user with a healthy hand would. However, the design of a robust and modular object detection system is outside of the scope of our project, and the system is designed in a modular fashion that a more defined vision program could easily be implemented without modifying any of the connections or boards used in the hardware control architecture.

In the current implementation of the AR tag system, the ArUco marker detection library is used. This library was developed by Rafael Munoz-Salinas at the University of Cordoba. The library used OpenCV, an open-source image processing library. Below is a brief overview of the implementation of the library:

1. An image is read in from the program using OpenCV.
2. An adaptive threshold is applied on the image. Lines are generated at boundaries of contrasting color.

3. Contours are calculated, which approximate the curvature of the lines.
4. Another filter is applied which removes more sparse line clumps.
5. A transformation is applied on the remaining closed-loop four-contour clusters to map the image to the proper perspective and orientation. These are then matched to an associated AR marker.

When using a camera feed, this marker operation is repeated on each image frame that is pulled from the active camera source. From this process we get an ID for each marker identified. The system can also determine a pose estimate of the marker in relation the center point of the camera. To accomplish this, however, the intrinsic properties of the camera must be taken into account. These properties are defined by the physical properties of the camera, including resolution, cone of view and lens curvature. Without this information, the system has no parameters to transform the angles produced the contour detection into positions and rotations.

The library accounts for this issue by utilizing a camera calibration program provided directly by OpenCV. Using this program, a .yml file is produced. The file contains basic information concerning resolution and other parameters, as well as a map of points concerning the positions of various points located on a calibration board.

When this intrinsic file is given to the marker detection algorithm, a pose estimate can be generated for the marker. The units for the generated positions are relative to the given marker size and the sizes used for the intrinsic file generation. One of the highlights of this system is that conversions to different units or unit systems are very straightforward. All that needs to be updated in order to generate a correct estimate is the .yml file corresponding to those desired units and the number used to evaluate the tag size. Due to the values being independent, though, both must be updated to remain consistent; a linked unit conversion is more desirable to avoid user error.

Feature Approximation using Markers

The system is also capable of determining the positions of features not directly tagged by a marker in relation to the camera position. This is completed by using an AR marker that is positioned on a surface near the grasping feature. Due to the parsing method for the tag in the image, the tag cannot be adequately and consistently detected if formed over a curved or spherical surface. If the tag were to be identified, we would experience errors in rotation, which would lead to inaccurate orientations computed for the hand. Due to the system not being able to compute partial identifications because of the nature of the key inside of the marker, we also must be careful of the size of the tag on a feature. By reducing the size, the system has a reduced effective range of tag identification, minimizing the effectiveness of the system.

The pose estimate is generated in the form of two 3x1 column vectors. One denotes the translation of the center of the marker, and the other denotes the rotation in Rodrigues vector form. This rotation can be computed into a 3x3 rotation matrix. From here, a 4x4 matrix of the transformation of the point can be generated.

To generate the pose estimate, the system is given a transformation matrix relating the marker coordinate system to the feature coordinate system. These matrices are determined by the user and are tied to a particular marker ID. The formula for calculating the transform of the camera to the feature can be written as:

$$T_f^c = T_m^c T_f^m \tag{12}$$

The sub- and superscripts c, m, and f represent the camera, marker and feature, respectively. The resultant transformation directly correlates to the position of our feature. An example of the offset feature detection can be seen in *figure 30*. The arrow drawn denotes the feature that is being estimated, which is a door knob. This is an example of a feature that would not be appropriate for direct marker identification.

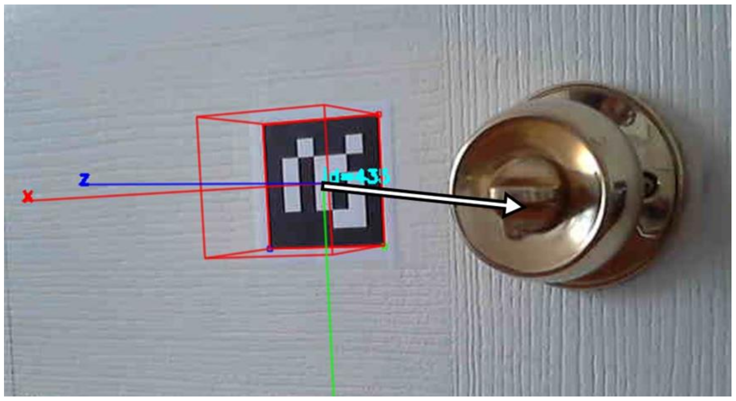


Figure 30: AR marker detection with door knob

After the feature's pose is estimated, the data must be transmitted to the main processor for tracking and identification. This is done using a serial connection between the pcDuino and the main processor. The selected baud rate for communication is 9600.

Joint Angle Approximation

For tracking and stabilizing on an object, there are two methods that can be used to approximate the joint angles required for the two DOFs of the wrist. The first method discussed utilizes a kinematic model of the arm to provide transformations relating the position of the feature with respect to the base of the hand. The second method uses the position related to the camera, calculated previously in equation 3, and analyzes two separate planes to determine the necessary joint angles.

3-Dimensional Kinematic Method

The two DOFs in the wrist can be modeled as a robot arm with a set of rotary joints at each of the degrees. Each degree corresponds to an individual coordinate system. Using DHparameterization, relations between each of the coordinate systems can be calculated. Two of these coordinate system transformations, namely the transform from the camera to the wrist pitch and the transformation from the wrist pitch to the wrist roll (which acts as the base coordinate system), are dependent on the pitch angle and roll angle, respectively. A diagram of the model and the various transforms can be seen in *figure 31*. This diagram shows the various coordinate systems associated with different parts of the system as well as the transformations between each of the coordinate systems. Depending on either using the tag as the feature or using an offset feature, an additional coordinate system transform may be required, as depicted.

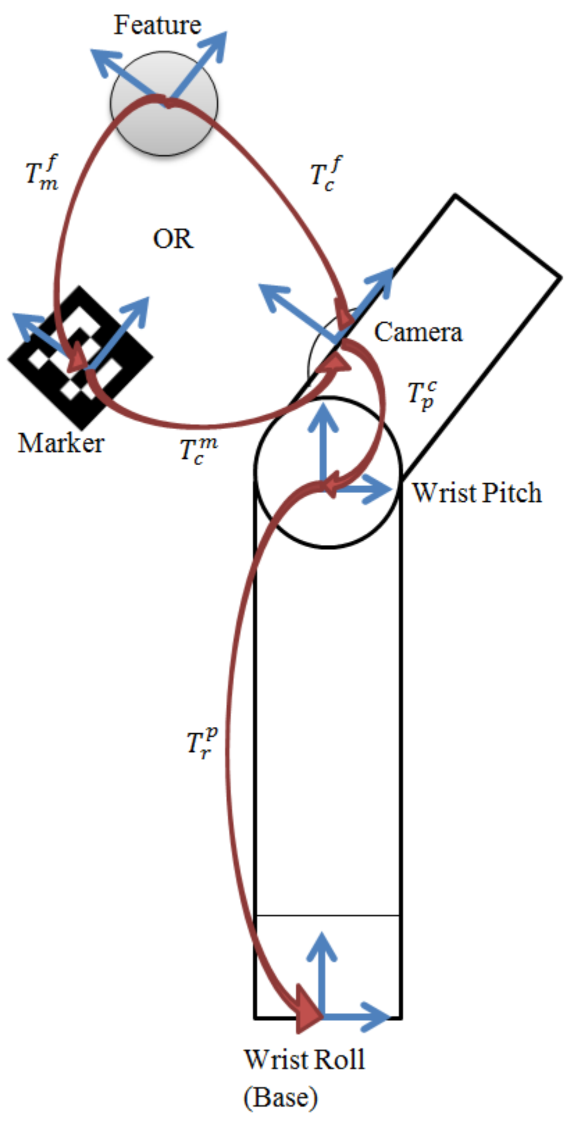
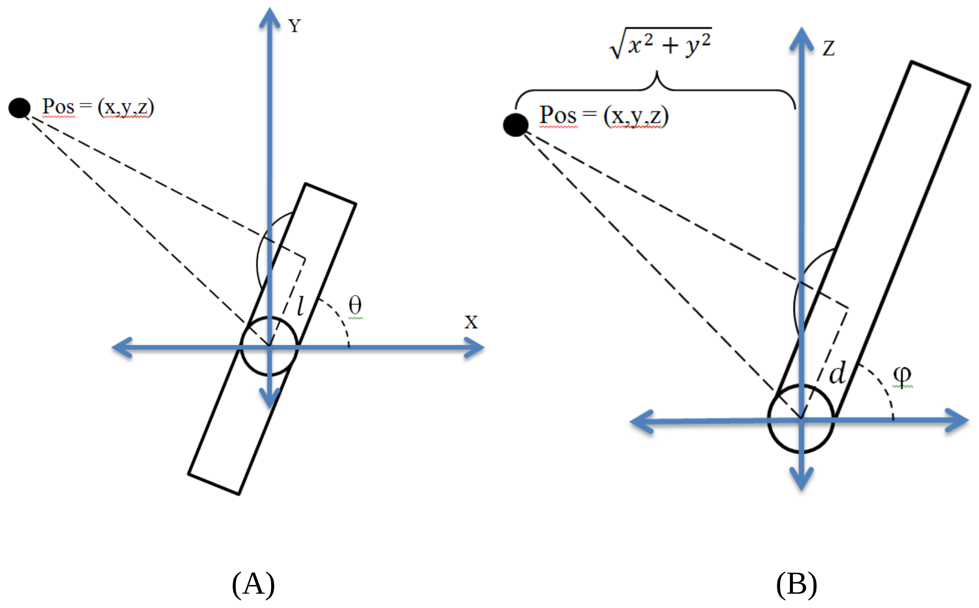


Figure 31: Diagram of Arm with modeled coordinate systems and transformations

Once the transformation matrices are obtained, we can then apply our transformation to the feature-to-camera transformation detailed earlier to obtain an overall transformation between the feature and the base of the arm. The resultant formula for the transformation is:

$$T_r^f = T_m^f T_c^m T_p^c T_r^p \tag{13}$$

Once the final transformation and the feature's position relative to the base of the arm as a result, joint angles must be calculated. To compute both of these angles, the system is analyzed on two planes: on the x-y plane (perpendicular to the axis of rotation for the roll DOF) and on a plane parallel to the z axis. From these planes, geometric relations between the angles of the joints and the position of the feature can be found. Diagrams of both of these displays can be found in *figures 32A and 32B*. θ represents the roll angle and φ represents the pitch angle.



Figures 32A and 32B: Planar drawings of hand on x-y plane and z-a plane, respectively

Equations for both the pitch angle and roll angle are:

$$\theta = 180 - tan^{-1}\left(\frac{y}{x}\right) - cos^{-1}\left(\frac{l}{\sqrt{x^2+y^2}}\right) \tag{14}$$

$$\varphi = 180 - tan^{-1}\left(\frac{z}{\sqrt{x^2+y^2}}\right) - cos^{-1}\left(\frac{d}{\sqrt{x^2+y^2+z^2}}\right) \tag{15}$$

One note on this method is the use of parameters l and d, which denote the position of the camera relative to center of the pitch joint axis. By modifying these values, the feature can be tracked at a different point on the palm. This can be advantageous depending on the object the hand is tasked with gripping

Non-Absolute Joint Difference Method

The second method for calculating joint angles for the wrist involves the pose generated for the detected feature with respect to the camera frame. Similar to the previous method, the system is analyzed from two planes: the y-z plane and the x-z plane (both of which are perpendicular to the camera's viewing plane). From here, we can analyze the trigonometric properties formed above to achieve the equations for two angles, which can be written as:

$$d\theta = \tan^{-1}\left(\frac{x}{z}\right) \tag{16}$$

$$d\varphi = \tan^{-1}\left(\frac{y}{z}\right) \tag{17}$$

The angles that are derived are the difference between the actual and desired angles. Without a base reference frame for the position to use as a reference, our angles are without reference; due to this, the computational output is relative angle changes rather than absolute angles.

Method Comparison

With the first modeling method, multiple transformations are utilized to determine the absolute pose estimate. Due to requiring the angles of the joints in order to compute the two transformations T_p^c andT_r^p, the rest of the transformation is computed on our main processor rather than still on the coprocessor. Though this alleviates some of the computational load from the processor and avoids negatively impacting the frame rate, the burden results in longer processing time for parsing the information sent to the main processor from the pcDuino.

For the second method, however, the system is left relative and reduces the amount of computation needed to determine the joint angles. A similar result is achieved, and the second method could be modified to account for different target positions on the palm. For this reason, the second, non-absolute method is favorable.

Control

The control algorithm used to move the fingers is known as Proportional Integral Derivative (PID) control. In this system, the position can be controlled using rotational potentiometers which output the finger position. By knowing what position we want the finger to be at, an error can be found based off of where the finger is which can be seen in the equation below.

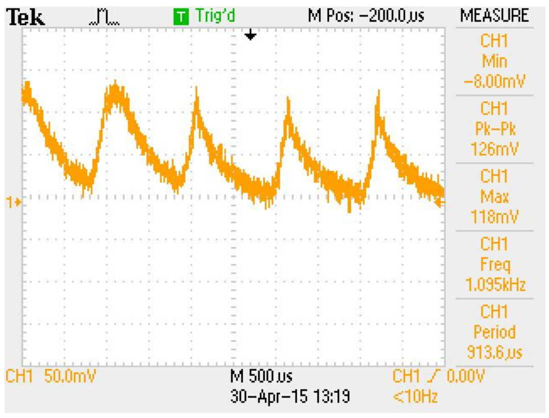
$$E = |actual - desired| \tag{18}$$

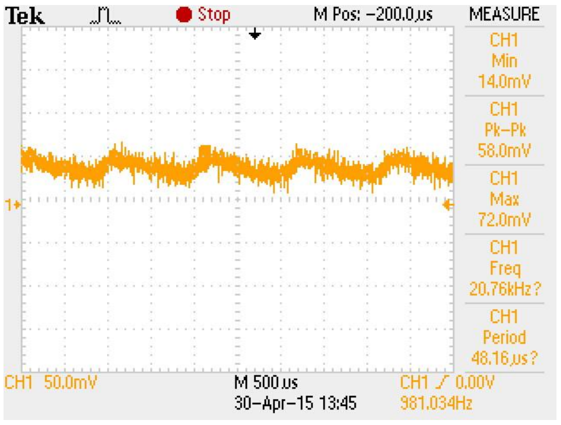
This error can then be used to determine how fast the motor has to drive in order to go to the intended position. By multiplying this error by a constant, the motor response will either speed up or slow down. Tuning the constant that is multiplied by our error, *E*, will give us solely a proportional control. We determined that ideally our finger should take approximately 1 to 1.5 seconds to move from an open grip to a closed grip. Since we do not need high speed or precision in the system, we believed that using only P control, we could achieve the desired positional control meeting the previously stated parameter.

Results

Though we were unable to achieve all goals, the project still yielded deliverables. The mechanical design proved to be robust with some minor issues. The full arm was manufactured and assembled, aside from the top of the forearm and rear, which included the roll DOF. This was due to the lack of space from wiring within the bottom of the forearm. The bevel gear driven system successfully moved the fingers. Due to the bevel gear not being fixed to the finger itself, there was a lot of slop in the finger. The thumb curling actuation worked, though due to the cracking of a thin wall between the mitre gear and bevel gear on the intermediate shaft, the curling speed decreased significantly. Due to manufacturing issues, part of the hand case broke which was supposed to act as a support for the motor that controlled the thumb rotation. This left the motor with only one point of contact to the hand causing instability when the motor drove. The belt and pulley system which drove the pitch of the wrist had about 40 degrees of slop. This can be attributed to the imperfect connection between the set screws and the shaft. Another contributing factor could be the un-tensioned belt driving the entire system.

The electrical system was fully operational for all five fingers and set up for the three remaining degrees-of-freedom. There were no major failures though we were unable to gather good current sensing feedback. Due to the large gear ratio of the system, the amount of current drawn to increase the torque was miniscule. This led to low changes in the signal from the feedback pin on the motor driver to the Arduino. When a force was applied to the fingertip, the signal coming from the feedback pin became a sawtooth wave because of the PWM signal being sent to the motor driver inputs. To fix this, a low-pass filter was implemented to cut out half of the frequency at which the PWM signal was being sent. This left us with a much cleaner signal as shown in *figures 33A and 33B.* As shown in the figures, the change in the peak to peak voltage went down by about 50% when the filter was applied to the signal.



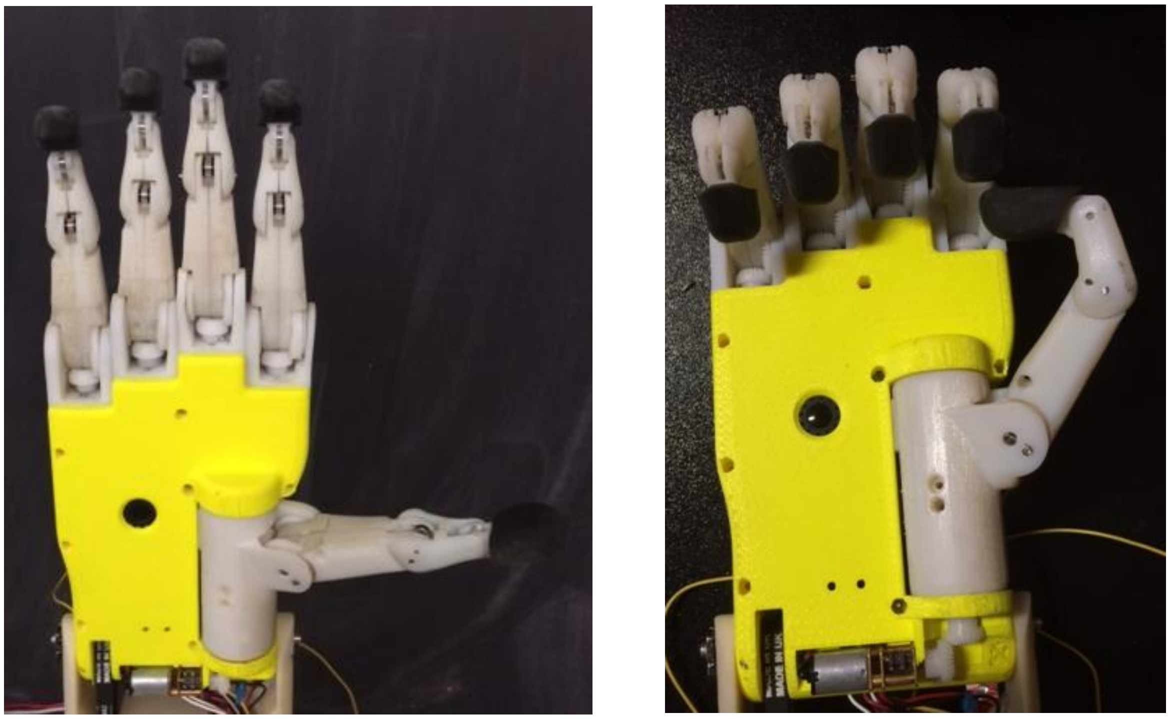


(A) (B)

Figures 33A and 33B: Unfiltered (A) and Filtered (B) current feedback from applied force

The object detection algorithm was successfully implemented into the control architecture of the hand. Multiple AR markers can be identified and used to drive the hand to desired grip formations. Pose data for markers can be sent over to the main processor, but is not currently handled for use in tracking and stabilization.

The control architecture for the hand is complete, allowing us to actuate all of the fingers to perform desired grips based on the AR markers that are shown to the hand. The “open” and “closed” grips for the hand can be seen in *figures 34A and 34B.*



(A) (B)

Figures34A and 34B: Completed hand in open and closed configurations

After tuning our PID constants, we found that only a proportional gain was necessary to actuate the fingers. *Figure 35* shows the PID control curve taken by converting potentiometer values into respective angles.

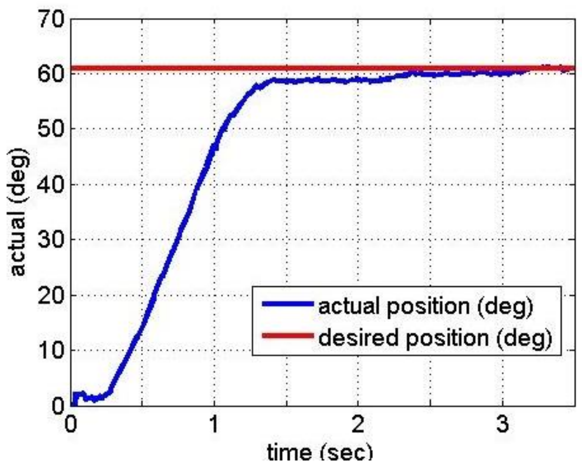


Figure 35: PID Control Curve

It took approximately one second for the finger to move from open to closed configuration.

Recommendations

A recommendation for future work on this project is to implement a more reliable form of force sensing. The current implementation draws too little electrical current from the motors to provide any meaningful information about the torques that the system is experiencing. Possible suggestions are to implement a tactile sensor in the fingertips to achieve direct forces at the tip, or to use a lower-geared motor with a voltage amplifier, providing a higher resolution in torque that we compute and allow for more accurate measurements. Further, the pitch actuation should be improved. The current design experiences slop in the belt system, as well as deflection of the motor shaft due to only a single point of rigid contact with the wrist. The forearm also needs to be re-examined. Space from wires was not taken into account in its design, and as a result debugging and wrist actuation became very difficult to perform. A more open design to account for these areas would be advised. For object recognition, a higher framerate, either through a more powerful coprocessor or through an optimized detection algorithm, is recommended. Without a high frame rate, the system will stutter and be unable to accurately track objects.

References

Abyarjoo, Fatemeh , Armando Barreto, Somayeh Abyarjoo, Francisco R. Ortega, and Jonathan Cofino. "Monitoring Human Wrist Rotation in Three Degrees of Freedom." *Academia.edu*. IEEE, 7 Apr. 2013. Web. 17 Oct. 2014.

Berg, Alexander C., and Jitendra Malik. "Geometric Blur for Template Matching." University of California, Berkley. Web. 14 Oct. 2014.

Bruenger, John, Ted Fischer, Chris Chapman, Jane Tucker, Amy VerHelst, and Wanda J. Wysor. "HYPERMUSCLE: MUSCLES IN ACTION." *www.med.umich.edu*. University of Michigan Learning Resource Center, 1994. Web. 17 Oct. 2014.

Casley,Sean, Thanacha Choopojcharoen, Adam Jardim, and Deniz Ozgoren. *IRIS Hand: Smart Robotic Prosthesis*. Worcester, MA: Worcester Polytechnic Institute, 21 April, 2014.

Christine Connolly, (2008) "Prosthetic hands from Touch Bionics", Industrial Robot: An International Journal, Vol. 35 Iss: 4, pp.290 – 293

ElKoura, George, and Karan Singh. "Handrix: Animating the Human Hand." *www.dgp.toronto.edu*. The Eurographics Association, 2003. Web. 16 Oct. 2014.

Fiala, Mark. "ARTag Revision 1. A Fiducial Marker System Using Digital Techniques." *NRC/ERB-1117* (2004): National Research Council of Canada. Web. 15 Oct. 2014.

Fincher, Louise A. *Elbow, Forearm, Wrist, and Hand*. La Crosse, WI: Orthopaedic Section, APTA, 1997. *www.uta.edu*. Web. 16 Oct. 2014.

Grauman, Kristen, and Bastian Leibe. "Visual Object Recognition." *Synthesis Lectures on Artificial Intelligence and Machine Learning* 5.2 (2011): 1-181. George Mason University. Web. 16 Oct. 2014.

Heseltine, Thomas, Nick Pears, Jim Austin, and Zezhi Chen. "Face Recognition: A Comparison of Appearance-Based Approaches." *Proc. VIIth Digital Image Computing: Techniques and Applications* (2013): University of York. Web. 14 Oct. 2014.

Howarth, J. W., H. H. C. Bakker, and R. C. Flemmer. "Feature-based Object Recognition." *Autonomous Robots and Agents, 2009. ICARA 2009. 4th International Conference* (2009): 375-379. *IEEE Xplore*. IEEE. Web. 16 Oct. 2014.

"i-Limb Ultra." Home. Web. 1 May 2014.

Jiminez, M.C., Fishel, J.A., “Evaluation of force”, *vibration and thermal tactile feedback in prosthetic limbs*, IEEE Intl. Conf. on Haptic Interfaces for Virtual Environment and Teleoperator Systems (Haptics), 437-441, 2014.

Karla, Prem K. "Canny Edge Detection." Indian Institute of Technology, Delhi, 23 Mar. 2009. Web. 16 Oct. 2014.

"Morphopedics." *De Quervain's Disease/Tenosynovitis*. Ed. Arianna Legatie. N.p., n.d. Web. 3 Nov. 2014.

Norton, Kim M. "A Brief History of Prosthetics." *www.amputee-coalition.org*. InMotion, Nov.-Dec. 2007. Web. 16 Oct. 2014.

Norton, Robert L. *Design of Machinery: An Introduction to the Synthesis and Analysis of Mechanisms and Machines*. Boston: McGraw-Hill Higher Education, 2011. Print.

Parasuraman, S., and Kee C. Yee. "Bio-Mechanical Analysis of Human Hand." *Bio-Mechanical Analysis of Human Hand*. IEEE, 10 Mar. 2009. Web. 17 Oct. 2014.

Smith, Matt. "New BeBionic Hand Almost Doubles Its Grip-strength, Steered by User's Electrical 'skin Signals' Alt." Engadget., 7 Sept. 2012. Web. 15 Oct. 2014.

Spong, Mark W., Seth Hutchinson, and M. Vidyasagar. Robot Modeling and Control. Hoboken, NJ: John Wiley & Sons, 2006. Print.

Talbot, David. "A Prosthetic Hand That Sends Feelings to Its Wearer." *MIT Technology Review*. 5 Dec. 2013. Web. 10 Oct. 2014. "Amputees Discern Familiar Sensations across Prosthetic Hand." *The Daily*. 14 Oct. 2014. Web. 13 Oct. 2014.

Taylor, Tim. "Median Nerve." *www.innerbody.com*. InnerBody, n.d. Web. 17 Oct. 2014.

Taylor, Tim. "Muscles of the Hand and Wrist." *www.innerbody.com*. InnerBody, n.d. Web. 17 Oct. 2014.

"The Hand." Life changing myoelectric hand packed with the latest technology. Web. 1 May 2014.

Tony Lindeberg (2012) Scale Invariant Feature Transform. Scholarpedia, 7(5):10491.

Walha, Chiraz, Hala Bezine, Nacer K. M'Sirdi, Aziz Naamane, and Adel M. Alimi. "HandGrasp: A New Simulator for Human Grasping." *Academia.edu*. Media.isr.uc.pt, n.d. Web. 17 Oct. 2014.

Appendices

Appendix A – Primary Arduino Code

```
#include <SoftwareSerial.h>

#include <Mux.h>
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>

Mux mux;
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();

SoftwareSerial pcSerial(2, 3);


//Marker IDs assoicated with various objects.
#define OPEN 24
#define CLOSED 435
#define DOORKNOB 0
#define CYLINDER 2
#define PHONE 3

//Detected marker ID
int id = 0;

//z rotation of the tag in relation to the tag
int tagAngle = 0;

//Current gripping configuration
int targetGrip[6];

//List of available gripping configurations
// Format: {PINKY, RING, MIDDLE, INDEX, THUMB, THUMBROT}
int openGrip[6] = {500, 480, 500, 520, 680, 700};
int closedGrip[6] = {320, 700, 295, 700, 620, 630};
int doorKnobGrip[6] = {400, 620, 400, 640, 650, 3};
int pointGrip[6] = {350, 700, 350, 550, 700, 3};


//Array for P values (proportional gain in PID loop)
int P_Arr[7] = {4, 6, 4, 4, 6, 2, 2};

//Initial values for the PID code
int potValue;
int potDiff;
int currValue;

//String for storing Serial byte data
String inputString = "";
```

```
//Connections from the Micro to the MUX
int S0 = 4;
int S1 = 5;
int S2 = 7;
int S3 = 8;
int SIG = A3;

//PWM DRIVER PINS
uint8_t PINKY1 = 10;
uint8_t PINKY2 = 11;

uint8_t RING1 = 2;
uint8_t RING2 = 3;

uint8_t MIDDLE1 = 4;
uint8_t MIDDLE2 = 5;

uint8_t INDEX1 = 6;
uint8_t INDEX2 = 7;

uint8_t THUMB1 = 8;
uint8_t THUMB2 = 9;

uint8_t THUMBROT1 = 0;
uint8_t THUMBROT2 = 1;

uint8_t WRISTPITCH1 = 14;
uint8_t WRISTPITCH2 = 15;

uint8_t WRISTROLL1 = 12;
uint8_t WRISTROLL2 = 13;

//EN pins
int EN = 9;

void setup(){
  Serial.begin(9600);
  //Initializes the software serial for the pcDuino
  pcSerial.begin(9600);
  //Sets the digital pins used with the MUX to be outputs to control the channel I/O
  pinMode(EN, OUTPUT);
  pinMode(S0, OUTPUT);
  pinMode(S1, OUTPUT);
  pinMode(S2, OUTPUT);
  pinMode(S3, OUTPUT);
  //Sets up and begins the mux and the pwm driver
  mux.setup(S0,S1,S2,S3,SIG); //initialise Mux

  //Initializes PWM driver
  pwm.begin();
  //This is the maximum PWM frequency
  pwm.setPWMFreq(1000);
  //Turns off all motors
  stopDrive();

  id = OPEN;
  //Initializes hand to open grip upon startup
  updateGrip(id);
}

void loop(){
  while (pcSerial.available()) {
    //Setting variable to tag id sent from pcDuino
    char inChar  = (char)pcSerial.read();
    //Initialization that tells when a string is being sent from pcDuino
    if(inChar == '['){
      inputString = "";
      Serial.println("Started serial data line");
    }
    //Appending the character to bit to the previous character bits
    inputString += inChar;
    Serial.println(inputString);
    //If the string has been fully received
    if(inChar == ']'){
      //Parsing string to determine what the id and orientation are
      int underScore = inputString.indexOf('_');
      String sub = inputString.substring(1, underScore + 1);
      String subAngle = inputString.substring(underScore + 1, inputString.length() - 1);
      id = sub.toInt();
      tagAngle = subAngle.toInt();

      Serial.print("Marker ID: ");
      Serial.println(id);

      //Updating the target grip based off of the ID
      updateGrip(id);
      Serial.print("Tag Angle: ");
      Serial.println(tagAngle);
      Serial.print("Target Grip: ");
      Serial.println(targetGrip[0]);
      Serial.println("Finished string, resetting input");
      inputString = "";
    }
  }
  //Enable's the MUX
  //Serial.println("Trying");
  digitalWrite(EN,HIGH);
  delay(50);
  ////Used to debug wrist control
  //driveToRoll(650, A1, WRISTROLL1, WRISTROLL2, -1, P_Arr[6]);
  driveGrip(targetGrip);
}

//Function that drives the motor
//inputs: motor in1, motor in2, duty cycle
//outputs: N/A
```

```
void drive_motor(uint8_t in1, uint8_t in2, int dutyCycle){
  int peak = 0;
  //convert duty cycle into decimal
  float frac = (float)dutyCycle/100;

  Adafruit_PWMServoDriver mot = Adafruit_PWMServoDriver();
  if (abs(dutyCycle) <= 100){
    if (dutyCycle >= 0){
      peak = (int)(4096 - (float)4096*frac);
      if (peak == 0){
        peak = 4096;
      }
      else if (peak == 4096){
        peak = 0;
      }
      mot.setPWM(in1, peak, 0);
      mot.setPWM(in2, 0, 4096);
    }
    else {
      frac = abs(frac);
      peak = (int)(4096 - (float)4096*frac);
      if (peak == 0){
        peak = 4096;
      }
      else if (peak == 4096){
        peak = 0;
      }
      mot.setPWM(in1, 0, 4096);
      mot.setPWM(in2, peak, 0);
    }
  }
  //Serial.println(peak);
}

//PID code
//input:
int driveTo(int desiredPos, int potNum, uint8_t in1, uint8_t in2, int dir, int P_Value){
  int dutyDiff = 0;
  potValue = mux.read(potNum);
  //potentiometer values
  potDiff = P_Value * (potValue - desiredPos);
  //difference between the potentiometer and desired position multiplied by the proportional gain
  dutyDiff = (int)((float)potDiff/(float)255*100);

  //converts the potentiometer difference into a duty cycle value to be used in the drive motor
function
  if(dutyDiff > 100){
    drive_motor(in1, in2, dir*100);
  }
  else if(dutyDiff < -100){
    drive_motor(in1, in2, -dir*100);
  }
  else drive_motor(in1, in2, dir*dutyDiff);
```

```
}

int driveToRoll(int desiredPos, int potNum, uint8_t in1, uint8_t in2, int dir, int P_Value){
  int dutyDiff = 0;
  potValue = analogRead(potNum);
  //potentiometer values
  potDiff = P_Value * (potValue - desiredPos);
  //difference between the potentiometer and desired position multiplied by the proportional gain
  dutyDiff = (int)((float)potDiff/(float)255*100);

//  Serial.print("Finger Pot: ");
//  Serial.print(potNum);
//  Serial.print("   DutyDiff: ");
//  if(potNum == 2){
//  Serial.print(dutyDiff);
//  Serial.print("   ");
//  }
//  else Serial.println(dutyDiff);

  //converts the potentiometer difference into a duty cycle value to be used in the drive motor
function
  if(dutyDiff > 100){
    drive_motor(in1, in2, dir*100);
  }
  else if(dutyDiff < -100){
    drive_motor(in1, in2, -dir*100);
  }
  else drive_motor(in1, in2, dir*dutyDiff);
}

//drives the fingers to the target grip configuration (no thumb rot. implemented yet)
void driveGrip(int desired[]){
  //drives the pinky finger
  driveTo(desired[0], 2, PINKY1, PINKY2, 1, P_Arr[0]);
  //drives the ring finger, currently functionl
  driveTo(desired[1], 3, RING1, RING2, -1, P_Arr[1]);
  //drives the middle finger, currently functional
  driveTo(desired[2], 4, MIDDLE1, MIDDLE2, 1, P_Arr[2]);
  //drives the index finger, currently functional
  driveTo(desired[3], 5, INDEX1, INDEX2, -1, P_Arr[3]);
  //drives the thumb curling motion
  driveTo(desired[4], 6, THUMB1, THUMB2, -1, P_Arr[4]);
  //driveTo(desired[5], 7, THUMBROT1, THUMBROT2, -1, P_Arr[5]);
}

void stopDrive(void){
  //stops all motors from driving
  drive_motor(PINKY1, PINKY2, 0);
  drive_motor(RING1, RING2, 0);
  drive_motor(MIDDLE1, MIDDLE2, 0);
  drive_motor(INDEX1, INDEX2, 0);
  drive_motor(THUMB1, THUMB2, 0);
  drive_motor(THUMBROT1, THUMBROT2, 0);
```

```
}

//updates the target grip configuration
void updateGrip(int markerID){
  if(markerID == OPEN){
    for(int i=0; i<6; i++){
      targetGrip[i] = openGrip[i];
    }
    //Serial.println(targetGrip[0]);
  }
  else if(markerID == CLOSED){
    for(int i=0; i<6; i++){
      targetGrip[i] = closedGrip[i];
    }
    Serial.print(targetGrip[0]);
    Serial.print("\t");
    Serial.print(targetGrip[1]);
    Serial.print("\t");
    Serial.print(targetGrip[2]);
    Serial.print("\t");
    Serial.println(targetGrip[3]);
  }
 else{
   for(int i=0; i<6; i++){
      targetGrip[i] = openGrip[i];
    }
  }
}
```

Appendix B – MATLAB Code

```
%Fingertip position analysis
%in this part we use the vector loop method to find the postion of the
%fingertip with respect to the driving joint4
%Please refer to Design of Machinery (Norton, R.)
%for the methods and analysis

%1st four bar

a1 = 40.08; %link4
b1 = 4.5;   %link3
c1 = 39.75; %link2
d1 = 5.25;  %grounded link

%Position analysis (Vector Loop method), compute for q3
K11 = -d1/a1;
K21 = d1/b1;
K31 = (c1^2-a1^2-b1^2-d1^2)/(2*a1*b1);
```

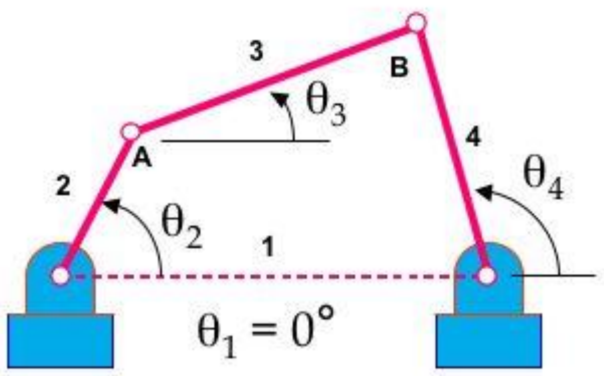
```
%driving joint angle
%the position is limited between 60 degree to 140 degree
q2 = (60:0.1:140)*pi/180; %angle of joint 4.

%Note that we call and treat it as theta 2 based
%on the vector loop method from Position Analysis chapter
%(Design of Machinery; Norton, R.).

A1 = cos(q2)-K11-K21*cos(q2)+K31;
B1 = -2*sin(q2);
C1 = K11-(K21+1)*cos(q2)+K31;

%Compute theta 3 for the first four bar linkage
q3 = 2*atan((-B1-sqrt(B1.^2-4*A1.*C1))./(2*A1));

% 2nd four bar
% link 3 from the previous four bar linkage is now the
% grounded link of this four bar linkage.
% All angles between each link will now be called "beta i," but still
% maintain the definition of "theta i"

%parameters (length) of all links in the second four bar
a2 = 22.61;
b2 = 4.5;
c2 = 22.80;
d2 = 4.5;

K12 = -d2/a2;
K22 = d2/b2;
K32 = (c2^2-a2^2-b2^2-d2^2)/(2*a2*b2);

%fixed angle between link 4 of first four bar and link 1 of second four bar
gamma1 = 138.58*pi/180;

%fixed angle between link 1 of first four bar and link4 of second four bar
gamma2 = 50.71*pi/180;


beta2 = gamma1-q2+q3+gamma2; %theta2 of the second four bar

A2 = cos(beta2)-K12-K22*cos(beta2)+K32;
B2 = -2*sin(beta2);
C2 = K12-(K22+1)*cos(beta2)+K32;

%theta3 of the second four bar
beta3 = 2*atan((-B2-sqrt(B2.^2-4*A2.*C2))./(2*A2));

%position of fingertip

gamma3 = 50.95*pi/180; %fixed angle between link 6 to fingertip
t = 12.06; %length from link 6 to the end of fingertip

P_tip = [a1*cos(q2)+a2*cos(q3+gamma2)+t*cos(beta3+gamma3+q2+gamma1);...
```

```
          a1*sin(q2)+a2*sin(q3+gamma2)+t*sin(beta3+gamma3+q2-gamma1)];

%plot
%plot the finger configuration from the computed numerical analysis above
%The plot will be separated for each four bar linkage
hold on;
%first four bar
plot([0 a1*cos(q2)],[0 a1*sin(q2)],'r');
plot([a1*cos(q2) a1*cos(q2)+b1*cos(q3)],[a1*sin(q2) a1*sin(q2)+b1*sin(q3)],'b');
plot([a1*cos(q2)+b1*cos(q3) -d1],[a1*sin(q2)+b1*sin(q3) 0],'k');
plot([-d1 0],[0 0],'m');

% plot the second four bar
plot([a1*cos(q2) a1*cos(q2)+d2*cos(q2+pi-gamma1)],[a1*sin(q2) a1*sin(q2)+d2*sin(q2+pi-
gamma1)],'r');
plot([a1*cos(q2) a1*cos(q2)+a2*cos(q3+gamma2)],[a1*sin(q2) a1*sin(q2)+a2*sin(q3+gamma2)],'r');
plot([a1*cos(q2)+a2*cos(q3+gamma2) a1*cos(q2)+a2*cos(q3+gamma2)+b2*cos(beta3-gamma1+q2)]...
    ,[a1*sin(q2)+a2*sin(q3+gamma2) a1*sin(q2)+a2*sin(q3+gamma2)+b2*sin(beta3-gamma1+q2)]);
plot([a1*cos(q2)+d2*cos(q2+pi-gamma1) a1*cos(q2)+a2*cos(q3+gamma2)+b2*cos(beta3-gamma1+q2)]...
    ,[a1*sin(q2)+d2*sin(q2+pi-gamma1) a1*sin(q2)+a2*sin(q3+gamma2)+b2*sin(beta3-gamma1+q2)],'k');

% plot from second four bar to fingertip
plot([a1*cos(q2)+a2*cos(q3+gamma2) a1*cos(q2)+a2*cos(q3+gamma2)+t*cos(beta3+gamma3-gamma1+q2)]...
    ,[a1*sin(q2)+a2*sin(q3+gamma2) a1*sin(q2)+a2*sin(q3+gamma2)+t*sin(beta3+gamma3-
gamma1+q2)],'m');

axis equal
xlabel('position in x-axis (mm)')
ylabel('position in y-axis (mm)')

%Determine the parameter of the Jacobian matrix for the fingertip
%In this section, we have to compute the forward velocity kinematics of the
%finger by computing all parameters and configuration of the second four bar
%linkage so that everything is in terms of theta 2

%solve for d(q3)/d(q2) (derivative of q3 in terms of q2)

% derive the parameters from first four bar linakge
% for forward velocity kinematics
dA = -sin(q2)+K21*sin(q2);
dB = -2*cos(q2);
dC = K11-(K21+1)*cos(q2)+K31;

%let d(q3)/d(q2) be dq3
dq3 = (-dA.*(tan(q3/2).^2)-dB.*tan(q3/2)-
dC)./(A1.*tan(q3/2).*(sec(q3/2)).^2+1/2*B1.*(sec(q3/2)).^2);
% derivative of beta 2 with respect to theta 2
dbeta2 = -1+dq3;

% derive the parameters from second four bar linkage
% for forward velocity kinematics
dA2 = -sin(beta2).*dbeta2+K22*sin(beta2).*dbeta2;
dB2 = -2*cos(beta2).*dbeta2;
```

```
dC2 = (K22+1)*sin(beta2).*dbeta2;

% derive theta3 of second four bar in terms of theta 2 of first four bar
dbeta3 = (-dA2.*tan(beta3/2).^2-dB2.*tan(beta3/2)-
dC2).*(dbeta2)./(A2.*tan(beta3/2).*sec(beta3/2).^2+1/2*B2.*sec(beta3/2).^2);

% fingertip Jacobian
% This is the Jacobian matrix from the fingertip forward velocity
% kinematics. The matrix has been solved numerically.
P_tip_J = [-a1*sin(q2)-a2*sin(q3+gamma2).*dq3-t*sin(beta3+gamma3+q2-gamma1).*(dbeta3+1);...
           a1*cos(q2)+a2*cos(q3+gamma2).*dq3+t*cos(beta3+gamma3+q2-gamma1).*(dbeta3+1)];

% To find the torque exerted on the driving joint, simply multiply
% the desired force to the Jacobian matrix

% Plot all possible configuration (position of the fingertip in x and y with
% respect to theta 4 (q2))
subplot(2,1,1)
 plot(q2,P_tip_J(1,:))
 xlabel('Driving Joint Angle (rad)')
 ylabel('position in x-axis (mm)')
subplot(2,1,2)
 plot(q2,P_tip_J(2,:))
 xlabel('Driving Joint Angle (rad)')
 ylabel('position in y-axis (mm)')



%NOTE: To see a specific configuration from this code, change
%the value of q2 to a specific angle (limit from 60 degree to 140 degree) and
%comment out the subplots above.
%Keep in mind that MatLab accepts input in radian only, so don't forget to
%convert degree to radian before running the code.
```

Appendix C – Vector Loop Equation

Parameter a, b, c, d, θ_2, θ_3, θ_4



$$K_1 = \frac{-d}{a}; K_2 = \frac{d}{b}; K_3 = \frac{c^2 - a^2 - b^2 - d^2}{2 \cdot a \cdot b};$$

$$A = cos(q2) - K_1 - K_2 \cdot cos(q2) + K_3$$

$$B = -2 \cdot sin(q2)$$

$$C = K_1 - (K_2 + 1) \cdot cos(q2) + K_3$$

$$q3 = 2 * atan\left(\frac{-B^2 \pm \sqrt{B^2 - 4 * A * C}}{2 \cdot A}\right)$$

Appendix D – Fingertip Position

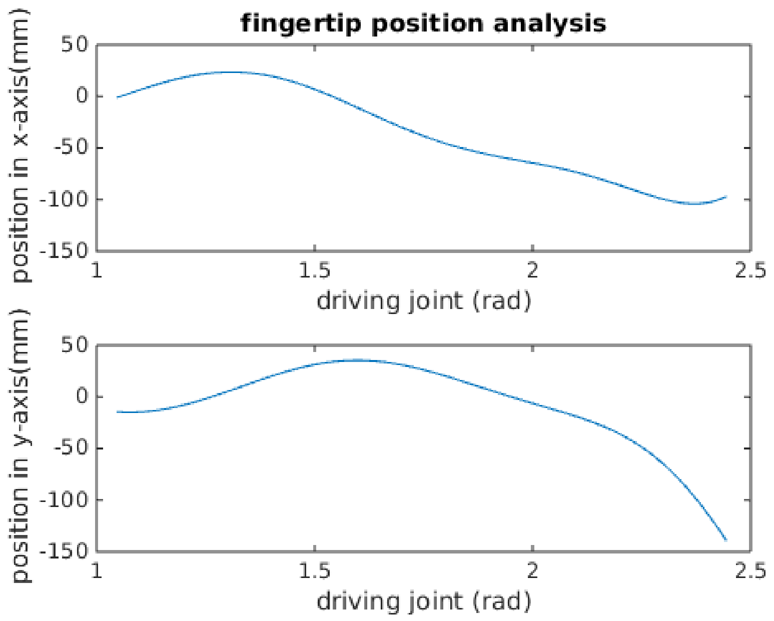


Figure 36: Driving joint angle related to fingertip position

Appendix E – PCB Design

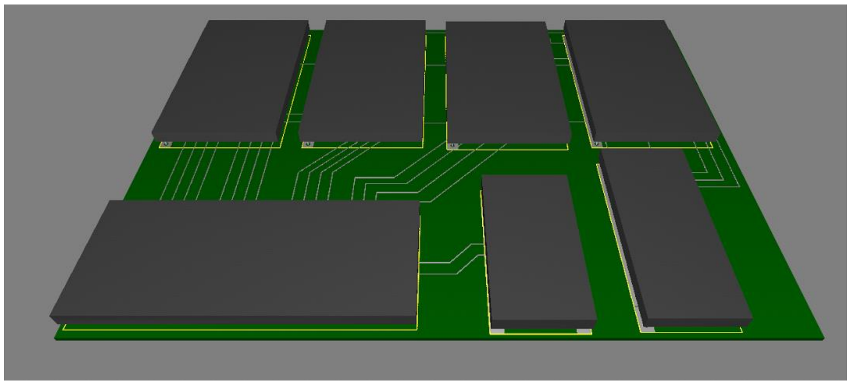


Figure 37: PCB developed for VIPeR Arm Components

Appendix F – Electrical Components

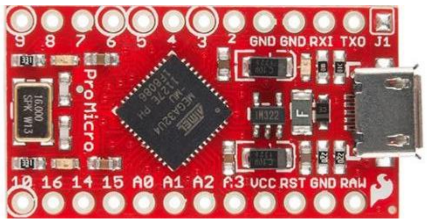

9 8 7 6 5 4 3 2 GND GND RXI TXO J1
Pro Micro
10 16 14 15 A0 A1 A2 A3 VCC RST GND RAW

Figure 38: Arduino Pro Micro Processor

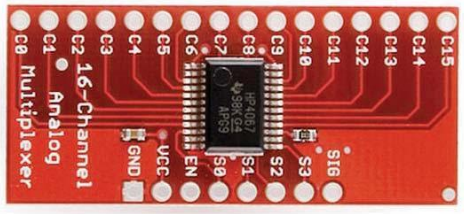

C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11 C12 C13 C14 C15
16-Channel Analog Multiplexer
GND VCC EN S0 S1 S2 S3 SIG

Figure 39: 16 Chanel Analog Multiplexer

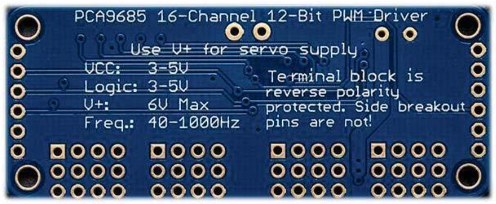

PCA9685 16-Channel 12-Bit PWM Driver
Use V+ for servo supply
VCC: 3-5V
Logic: 3-5V
V+: 6V Max
Freq.: 40-1000Hz
Terminal block is reverse polarity protected. Side breakout pins are not!

Figure 40: 16 Channel PWM Driver

Figure 41: DRV8835 Dual Motor Driver Carrier

Figure 42: Pololu 1000:1 Micro-gear motor

Figure 43: Rotary Potentiometer

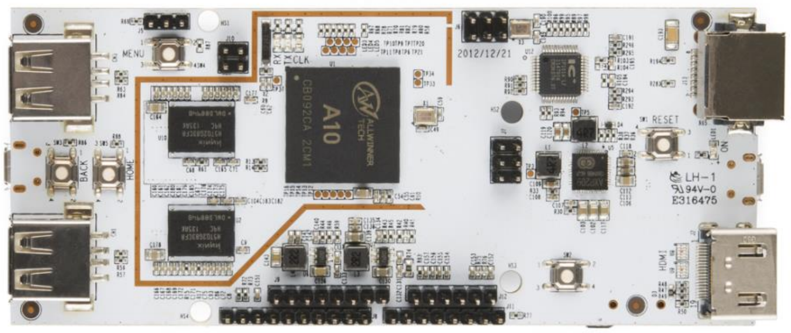


Figure 44: PcDuino Board

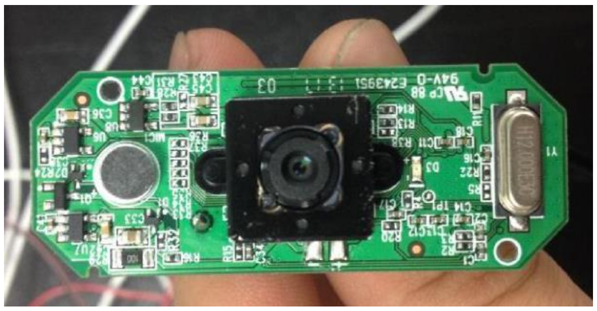


Figure 45: Logitech c525, Webcam

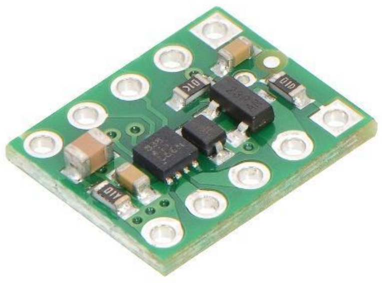


Figure 46: DRV 8838 Single Motor Driver Carrier

Figure 47: MC33926 Dual Motor Driver Carrier

Figure 48: Pololu Metal Gearmotor (227:1)

Figure 49: Logitech c270, webcam

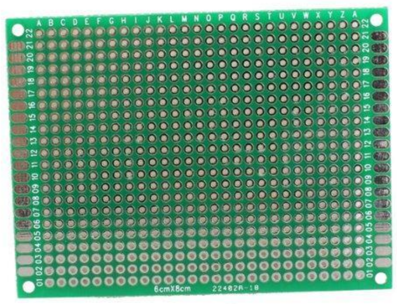


Figure 50: Protoboards for slotted PCB Design

Authorship

Abstract	RE, KL, NS, JW
Background Research: Prosthetic Devices	KL
Introduction	RE, NS
Background Research: Hand Anatomy	JW
Background Research: Available Prostheses	KL
Background Research: Object Recognition	RE
Background Research: Sensing Technology	NS
System Overview	RE, KL, NS, JW
Electrical Architecture	JW
Mechanical Design	KL
Force Analysis	NS, JW
Object Recognition	RE
Control	KL
Results	RE, KL, NS, JW
Recommendations	RE