Luke's Hand

Senior Design Final Documentation

Group 6 - Fall 2009

Alexander Bermudez

Esteban Guerra

Cesar Oleas

Jose Suarez

College of Engineering and Computer Science,
University of Central Florida,
Orland, Florida, U.S

Table of Contents

CHAPTER 1 EXECUTIVE SUMMARY

The field of robotics has been rapidly growing during the last decade, opening a new window to bring robotics into everyday applications. The range of applications goes from simple toys, to medical equipment and space missions conducted by NASA. Thus, the team has decided to jump in the wagon with the Luke's Robotic Hand.

The idea behind the *hand* is to provide the user with an economic while efficient alternative to similar products currently available on the market. Its most valuable use is in the prosthetics industry, but it can also be used on industrial processes that cannot be done directly by humans (extreme temperatures).

The hand itself is a mechanical limb with seven (7) degrees of freedom, one for each finger, two for the thumb, and one for the wrist. It will be designed using lightweight materials, such as aluminum, for better usability, and it will feature an array of sensors to provide feedback to the users and system (MCU) to perform task such as: sense objects slipping, detect object within reach, identify object, determine temperature of the object, and measure force being applied. It will also feature a wide array of user interfaces, having the Glove and Pod interfaces as the more relevant ones. The *hand* will mimic to a certain accuracy the actions performed by the human hand wearing the Glove, while the Pod interface will allow the user to give a limited set of instructions to the *hand* by applying forces to a specially design *insole*.

This project will consist of hardware and software design, as well as efficient and effective planning. Software will be used to effectively manipulate the date received from the sensors and motors, and to design an easy to use graphical user interface (GUI). Components such as sensors, motors, and RF modules will be acquired separately and integrated into the system, while all other circuitry will be designed in-house. Apart from functionality and efficiency, cost will be another important factor to be taken into consideration, and all components will be selected by not only looking at the specifications, but also the cost per unit.

CHAPTER 2 INTRODUCTION

2. Introduction

2.1 Motivation

The Engineering profession was conceived as one that would serve the people, and as such, all engineers try to apply their knowledge to the design and development of new products and technologies that would improve quality of life in one form or another. Engineers try to achieve those goals by working in their fields of interest, and often try to come up with ideas that are best suited to Sci-Fi flicks. In research labs is where engineers put their minds together and try to create new technology, which changes the vision of the world. All these make possible all the technological advances seen in the last few decades. And lately, the two fields of engineering that have been growing most rapidly are Electrical and Computer engineering, creating a great influx of ideas and an opportunity for new engineers to contribute to the steady growth of the profession.

While watching our favorite movie "Star Wars", we saw how Luke lost his hand when he was fighting Darth Vader. He replaced his hand with a very futuristic robotic hand. This movie is where the name of the project comes from. The idea of recreating a human like hand using robotics inspired the senior design group to build one. It is thought that working on a project like this will enhance all the members' engineering knowledge in electronics, mechanics and programming. The major motivation of this project is to build a prototype of a robotic human like hand so it could assist unfortunate people that have lost a limb.

It is important to give short information of how many people in the United States are living without a limb. The National Limb Loss Information Center reports that there are 1.7 million people living with a limb loss and it is estimated that one of every 200 people has had an amputation. Based on information obtained from the National Center for Health Statistics, there are 50000 new amputations every year in the United States and the ratio of upper limb to lower limb amputation is 1:4. Actually, there are 350,000 people with amputations and 30% of them have upper limb loss and the 10% of this quantity is related wrist and hand amputation. In the United States are 41,000 people are registered who had an amputation of hand or complete arm. There are several reasons why people are losing their limbs but the most important and relevant are: soldiers coming from Iraq, but the majority of new amputations occur due to complications of the vascular system (of or pertaining to the blood vessels), especially from diabetes. It is of great concern for us to see how all these people are living in these conditions, for that reasons this problem has caught our attention. It is an inspiration and motivation for us to use all the knowledge learned so far at the University of Central Florida and we will try to come out with a useful design.

Taking the above into consideration, the fact that the members of the groups are both Electrical and Computer Engineers, with interest in robotics and communications, and some ideas seen in movies during the last decade, it was determined that the members of the group wanted to contribute to the field by designing a robotic hand that could be used in prosthetics and/or industries that require external assistance to handle small objects.

First of all, what is a robot? A robot is a device is an instrumented mechanism used in science or industry to take the place of a human being. It may or may not physically resemble a human or perform its tasks in a human like way, and the line separating robot devices from merely automated machinery is not always easy to define. In general, the more sophisticated and individual the machine, the more likely it is to be classed as a robot devices. But, the robotic hand can be classified as a manipulator robot; a manipulator may be defined as a mechanism usually consisting of a series segments, jointed or sliding relative to one another, for the purpose of grasping and moving objects usually in several degrees of freedom.

A computer through a human interface may remotely control the Luke's hand. During the design, built and test of the robotic hand it is the group's intend to attempting mimic a human one as much as possible, while limiting it's functionality due to multiple factors such as time constrains, knowledge and budget matter. The forecasted limitations will play a bigger role in the mechanics of the robot, which is not the area of study in this specific scenario. But will be compensated by a diverse array of controlling devices that will be implemented, thus expanding the possible uses of the hand to multiple fields.

One of the problems with current prosthetic hands is the cumbersome mechanisms and electronics that require the individual to be extra careful with everyday functions. There are still several limitations with prosthetic hands in particular; the ability to apply appropriate pressure in order to hold items without dropping them. Another disadvantage of the current systems is the cumbersome electrical signal cords used to transmit and receive commands to and from the hand. Depending on the system configuration, there are usually two specific methods used to generate signals for the manipulation. One of these methods is the use of neurological impulses. The other is the use of the human musculoskeletal system to manipulate sensors in order to generate signals that will manipulate the prosthetic hand.

2.2 Goals

During the first half of the year, most of the efforts will be on understanding the anatomy of the human hand, and what needs to be done to reproduce its functionality. This will require the use of certain motors, servos as of now, and their integration with the microcontroller. The microcontroller will have the ability to link the robotic hand with the computer and a separate remote control, which will manage the robot hand manually.

The hand distinguishes humans from mostly all species in our planet. The hand is a very complex interface with the brain when it comes on sensing any kind of different surfaces and objects. The complexity of the hand's skeleton consists in many diverse bones with different sizes and functions. The palm is made of five metacarpus bones, which are the base of the hand. In our project, we are not going to need to build artificial metacarpus, because the motion of the fingers will not depend on these five bones.

One of the purposes of the robotic hand is that it could be used as prosthesis, but in order to behave like a real human hand. It is important that the time taken for the chain of operation from the disabled person's desire to carry out an action to the action itself must be minimal. In addition, the means and the system itself must be compact and light. It must no created additional difficulties for the owner. The most effective way of achieving this aim is to exploit the control commands given via the bioelectricity of the relevant muscles, to amplify them, to analyze them, and to transform them into the desire actions of a mechanical device like prosthesis.

2.3 Objectives

The objective of this robotic hand project is to simulate the behavior of tendons and muscle by using strings and servomotors. A strong metal sheet that would hold the fingers tightly will replace the metacarpus. The fingers are a grouping of three phalanges. The distant phalanges make the tips of the fingers; the proximal phalanges are the closest to the metacarpus, and the intermediate phalanges fall in between both. The thumb is the only finger that does not have intermediate phalange. The mechanics of the fingers will be mimicked in our robotic hand. The robotic fingers will have three metallic phalanges connected by joints.

The human hand uses its natural tendons around the fingers in order to connect the muscles within the bones. Those tendons mainly function by transmitting forces from one place of the whole hand to another to create a set of different movements, which include the manipulation of all the fingers. To recreate the functions of the tendons, strings are going to be implanted in the inner front part of the finger going from the tip of the finger and all the way down to the arm, somehow imitating the anatomy of a real human hand. For the robotic hand application, the above-mentioned strings would be individually connected directly to a particular servomotor, which will control the movement of each one of the fingers. The servo will function as the muscle of a human hand by pulling the string down and force the fingers from the robotic application to flex.

The figure 2.3.1 below shows the different bones of the hand mentioned in the hand description for a better understanding.

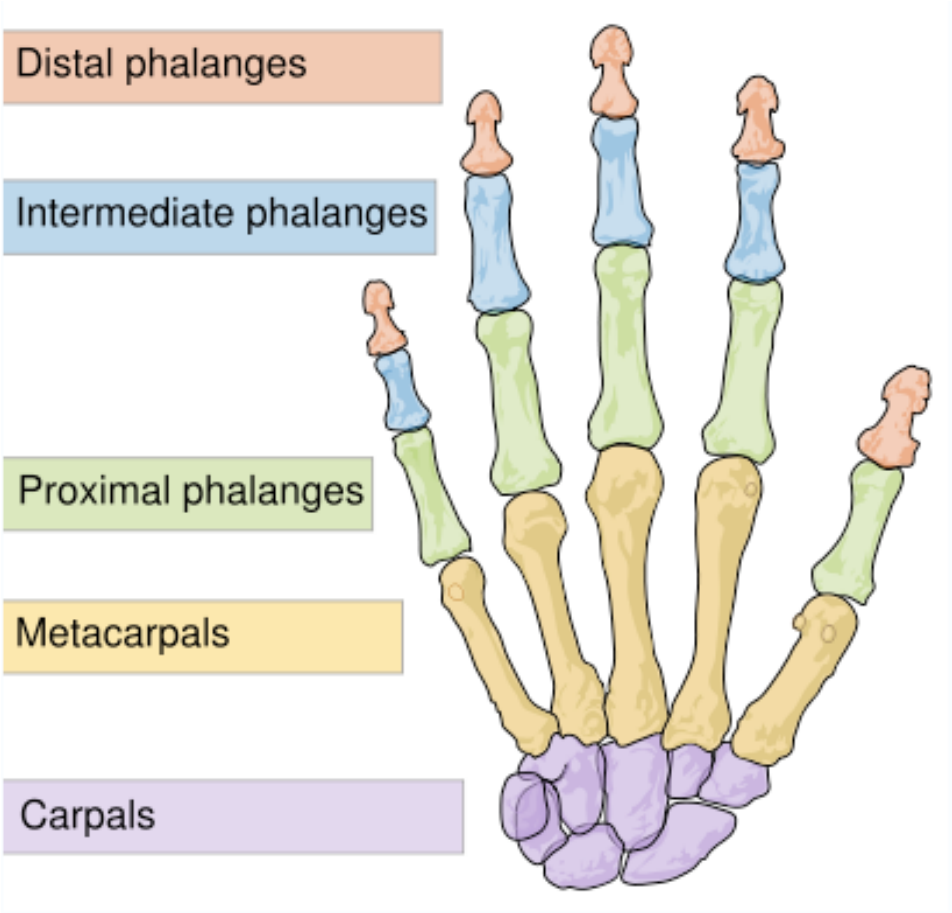


Figure 2.3.1: Skeleton of the Human Hand
Figure printed with permission granted by its author, Mariana Ruiz Villareal

To make the fingers go back to the original position, which would be an open hand with the fingers extended, the servomotors will return to its starting position and the extension of the fingers will be executed by the force of an elastic string attached on the back of the fingers. This elastic will pull the fingers back to the desired position without using any additional motors, mechanical rods, or any additional kind of power source. Instead it will use the elastic energy stored when they deform while the finger is flexing.

The figure 2.3.2 below shows how the tendons in a human hand are position. The strings in the robot hand will resemble the mechanical operation of tendons in a real human hand.

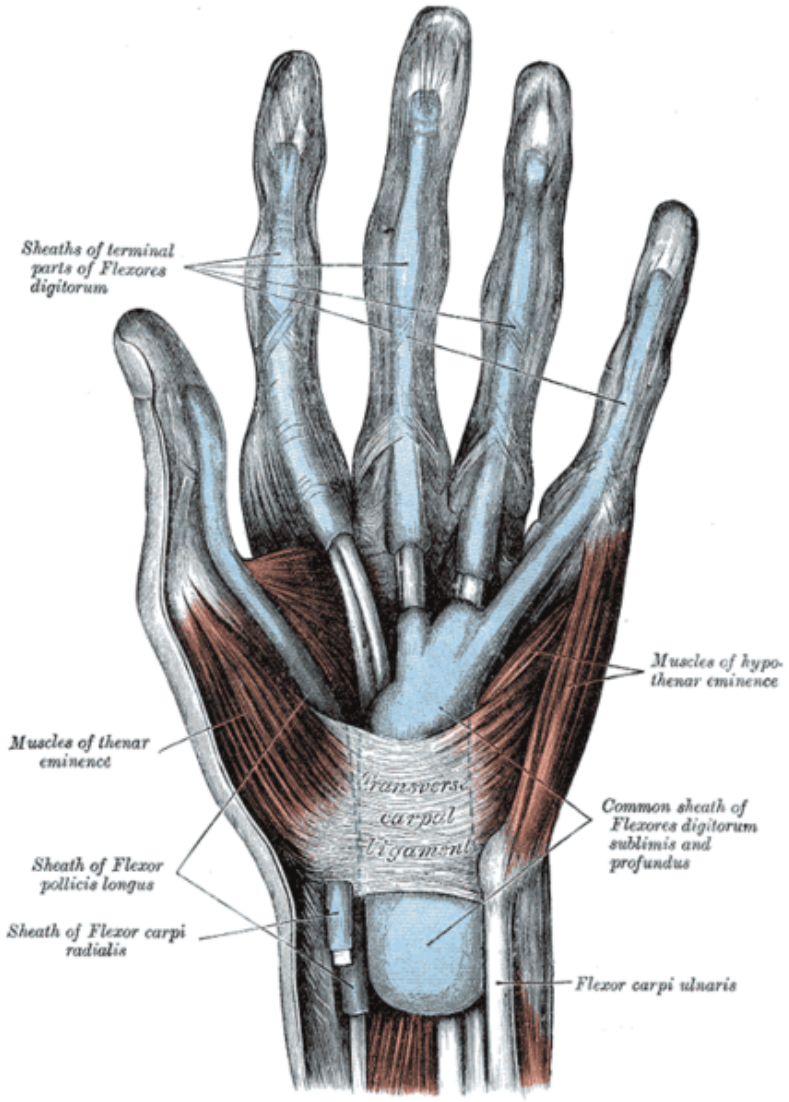


Figure 2.3.2: Tendons and Muscles of the Human Hand

Figure can be printed without permission because it was originally published in 1918 and therefore lapsed into the public domain because its copyright has expired

2.4 Specification and Requirements

The robotic hand is going to simulate some of the movements' human hand. The sizes of the fingers are taken from a real hand as well as the palm dimensions in order to have a more realistic solution with the application to remotely emulate a real human hand. Our goal is to recreate human like movements of the hand, the Luke's robotic hand must have the ability to move each finger separately.

The decision to use aluminum was taken to exploit the natural characteristics of such material, which helps to construct the skeleton of the fingers strong, resistant, and tremendously light in weight. The hand's palm would be constructed with a regular metal sheet. The size of the robotic hand is going to be around 15 cm long and about 9cm wide. The table 2.4.2 below shows the size of each finger and their parts. The measurements are given in cm:

Name	Size(cm)	Distal Phalanges	Intermediate Phalanges	Proximal Phalanges	Total
Index Finger	7.5X1	2	2.5	3	7.5
Middle Finger	9X1	2	3	4	9
Ring Finger	8X1	2	2.5	3.5	8
Baby Finger	6.5X1	2	2	2.5	6.5
Thumb	5.5X1	3	2.5	0	5.5

Table 2.4.2: Size of the different parts of the finger

The dimensions of the palm are going to be 5.5 cm long in the front and 6.5cm for the backside of the hand. The joints are intended to be out of metal rods, which are going to be attached on the finger. Each finger is going to have an elastic band, and they are going to act as tendons. The importance of the elastic band would be the pulling of the fingers back to their home place one the servomotors has no more power. Strings are going to be connected to servomotors, which would control the movements of each finger. In total five of them are needed.

For the electronics components, a microprocessor is needed and the options are endless due to the wide spectrum of microcontrollers available in the market. The PIC family is a natural selection because of it popularity among the educational community as well as robotics industry. There are some high-performance microcontrollers other than PICs that also need to be considered. A research is needed to decide the best option according to the application to be developed among with its specifications. Interesting characteristics are desired such as low power consumption microprocessor, well documented embedded applications for the microcontroller is a must, and communications as well as Ethernet capabilities perhaps would be required.

The specifications of the required microprocessor could be:

- Low power consumption
- At least 30 I/O pins
- USB to Serial Interface for programming and communication
- Enough RAM and ROM memory space for programming
- Instructions need to support 32-bit values and math operations
- Control of clock speed by software allows dynamic trading of power vs. speed
- Small in size and DIP package style preferred
- Needs to have a well source of information of how to use it

The microprocessor is going to help the senior design group to control the robotic hand; however, it is intended to build a custom made PCB (Printed Circuit Board) for the application. Additionally, an USB interface is going to be needed in order to connect it to the computer and program the microprocessor using the appropriated programming language for the microchip selected.

The microcontroller will be the central part of the project, and it will be integrated with a development board, a custom PCB board, or an already built Single Board Computer (SBC). The development board will be used to upload the software into the microcontroller and to test the system. The PCB board, if used, will either be ordered from a specialized vendor, or designed and printed on the lab. Depending on complexity of the final requirements and specifications, the use of the SBC will be considered as possible substitute to be PCB board the central part of the system.

The servomotors, together with the microcontroller, are the most important elements of the project. The motors will allow the individual motion of each appendage of the hand. Additional electronic components will be needed, but cannot be include it in the budget at this time, since it is not clear which kind of control will be used to interact with the robot. Depending on the controlling device used, the budget will change substantially, since the difference in cost between a wireless device and wired one is significant.

The robotic hand is going be controlled using the computer, a remote controller, and a glove. Through, the computer one can tell the hand to perform all the movements of the fingers and the different sign functions to be implemented. The controller is going have five buttons, one for each finger, but some combinations of the fingers' movements are needed so that the hand can also perform sign functions. The remote controller is going to be custom made by the members of the group, and its circuitry will be located in a separate PCB. The remote controller is going to be connected to the microcontroller using an USB cable. A single battery of 12Volts might be required as a power source for the robotic hand. The power is going to be distributed to the servos, the microcontroller, the sensors, and the different components of the PCB. In addition, it would be necessary to build a voltage

converter and/or voltage regulator, which is going to takes the voltage supply and convert it to the right voltage needed by each component. Another battery will be used as the power supply for the remote controller.

Software Requirements:

- Well-known programming language (C, Java, etc)
- Robust Integrated Development Environment (IDE)
- Low cost IDE (preferably sold with the MCU and/or RF module)
- Instruction Set Architecture (ISA) support.
- Memory allocation through assembly instructions
- Libraries should include support for modern interfaces (USB, FireWire)
- Preferably System independent (C or Java)
- Ability to program in machine code
- Modularity of the program.

CHAPTER 3 RESEARCH

3.1 Reference Projects

Extensive research was conducted in order to fulfill some of the major steps required to bring to life Luck's hand application. Important ideas and breakthroughs were taken in consideration from previous and ongoing projects, where robotic hands were design and implemented with about similar specifications and goals to the one specified in this document. Although the mechanics of this project is inspired from a real human hand and its characteristics, obvious limitations restrings mobility and functionality. From the mechanical research point of view, most of the information was taken from personal and collective observations of real hands. Since none of the group members are experts in biological or anatomy matters, some human technical information was taken from specialized web pages.

From the sources mentioned above, the most important and suitable approaches were taken in consideration, but also at the same time, learning from previous documented mistakes and downfalls were importantly noted to streamline the design process. Some of these projects were reach in features implemented and at certain degree constructed out of high technology that requires fundamental monetary resources. Others were simplistic ideas, and some of them even no realistic to be used as an application of this nature. Therefore, the group members took such applications as reference to avoid their weak points and lack of certain features. So in terms of budget, the middle point approach was taken in order to achieve the robotic hand application counting with the obvious limitations such as time required, instrumentation, facilities, complexity, field of engineering (usually Mechanical), and scope of the project.

3.1.1 Wireless Prosthetic Hand (UCF Spring 2007)

The purpose of the *Wireless Prosthetic Hand* (Figure 3.1.1) was to prevent it from crushing items by adding pressure-sensing capabilities, and to add wireless communications between the control and the hand to "alleviate the user from cumbersome signal cables" [Group 20, 1].

This project considered many approaches regarding the slipping of objects, and mainly focused on the possible use of microphones and Dynamic Force Sensors. The latter uses piezoelectric materials that produce a charge one a force is applied to it. Lead Zirconate Titanate (LZT), and Polyvinylidene Fluoride (PVDF) are materials that show piezoelectric characteristics, and were considered on this project. The microphone option was taken into consideration, but was discarded because it will not only detect the vibration generated by the object slipping but also surround noise, which will make signal processing a lot more difficult.

Figure 3.1.1 Wireless Prosthetic Hand

3.1.2 Other Projects

Other projects such as the CyberHand, iLimb, and the SmartHand were taken into consideration, but most of these were projects that have been in the works for many years, have undergo multiple redesigns, and are either commercial products (iLimb) or being developed by a groups of universities (CyberHand). Even though these offer great ideas to approach many of the main challenges in the design process, they are not suitable for this project due to some known limitations.

3.2 Interfaces

3.2.1 Overview

The interface selection for each of the components (subsystems) of the robotic hand is a fundamental aspect of the design process. Considering the fact that the system itself consists of the hand and at least two control interfaces, one wired and one wireless, multiple current technologies will be considered for use in different subsystems.

Two (2) IEEE and FCC approved technologies will end up being used, one wired, which will be selected from either USB, Serial (RS232), or FireWire, and one wireless, which is to be selected from either *RF, Bluetooth (802.15.), Wi-Fi (802.12),* and *ZigBee/MiWi (802.15.4).*

The selection of these technologies will be based on different aspects that need to be taken into consideration, such as *size, price, development cost, industry support, availability, power consumption,* and *data rate,* amongst others.

3.2.2 RF

Radio Frequency (RF) modules that do not include any standard, or pre-installed software stack will not be considered since the added load in the programming side will be too costly for the overall project. It is best to use any of the standards discussed below since the connectivity between multiple devices is much simpler, and allows the developers to focus on the actual functionality of the hand, rather that how to interface the control and the hand.

3.2.3 Wi-Fi (802.11)

Wireless Local Area Networks, commonly known as Wi-Fi, are aimed at data exchange between devices that require high data rates, such as portable computers and PDAs. The power consumption is relatively high when compared to other technologies, but the data rates are considerable higher, usually ranging from 11Mbps to over 100Mbps. It operates in the ISM (2.4Ghz) band, and its cost is considerably higher than other widely available wireless technologies.

3.2.4 Bluetooth (802.15.1)

Bluetooth is a standard and communications protocol used for wireless personal area networks, and is directed towards connecting and exchanging information between two (2) or more devices that require low power consumption and moderately fast data rates. The standard operates in the ISM (2.4Ghz) band at different speeds depending on the version, and ranges based on the class of the device. The specific speeds and ranges can be seen in detail in Table 3.24.a and 3.2.4.b.

Class	Maximum Permitted Power mW (dBm)	Range (approximate)
Class 1	100 mW (20 dBm)	~100 meters
Class 2	2.5 mW (4 dBm)	~10 meters
Class 3	1 mW (0 dBm)	~1 meter

Table 3.2.4 Bluetooth Specifications

3.2.5 ZigBee (802.15.4)

The IEEE 802.15.4 standard, also knows as ZigBee, provides a simple protocol to implement wireless personal area networks (WPANs) intended for home automation and other applications that require low power consumption and do not have the need for high data rates. The technical aspects of this standard can be seen in Table 3.2

ZigBee operates on the three ISM bands (2.4Ghz, 915Mhz, and 868Mhz), and has a maximum data rate of 250Kpbs while operating on the 2.4Ghz ISM band. On this band, each channel has a bandwidth of 5Mhz, and a maximum output power 0dBm (1mW).

Wireless Parameters	WiFi (802.11b)	ZigBee (802.15.4)
Frequency Band[a]	2.4 GHz	2.4 GHz
Number of Channels	11[b]	16
Range	about 100 m	about 100 m
Power Consumption Tx Mode	400 mA	25 to 35 mA
Power Consumption Standby Mode	20 mA	3 µA
Battery Life	1 to 5 days	3 to 5 years
Data Rate	11 Mbps	250 Kbps
Architecture	Star-Access Point	Star, mesh, cluster tree
Protocol Stack Size	500 KB	32 KB 4 KB (for limited function end devices)
Best-Suited Applications	Standard TCP/IP	Monitor and control

a. Both WiFi and ZigBee use the industrial, scientific and medical (ISM) radio bands.
b. 11 for North America and Oceania (channels 1, 6 & 11 are recommended); 13 for most of Europe; 14 for Japan

Table 3.2.5.1 Comparison between Wi-Fi and ZigBee

3.2.6 Universal Serial Bus (USB)

Probably the most widely used interface in the current market, since it is used in any kind of devices from computer mice to external hard-drives (HDDs), including all kind of peripherals. Its popularity in the field is given by its ease, low cost, compatibility and fast data rates offered by this technology. There are currently three (3) IEEE approved versions of the interface, USB 1.0, 1.1, and 2.0 (Figure 3.2.6), although only the latter two are currently supported, due to its faster data rate, USB 2.0 is rapidly gaining market share over USB 1.1. For that reason, only USB 2.0 will be considered for this project.

USB 2.0 offers a data rate of up to 480Mbp (around 60MBps), which to the limited amount of data being shared between subsystems, it certainly more than meets data rate requirements. Also, since it is so widely available, and mass-produced, the ratio of cost and date rate is probably the lowest of all technologies.

Figure 3.2.6 USB Connectors

3.2.7 FireWire (IEEE 1394)

Commonly knows as FireWire (name given by Apple Co.), IEEE actually classifies this interface as IEEE 1394. The interface is not as widely used as USB, but has gained certain market dominance in devices that require high data rates, higher than those offered by USB, such as high quality audio, video, and external HDDs. Two versions are currently available, FireWire 400 (Figure 3.27) and FireWire 800. The first can transfer data at three different rates, known as S100 (100Mbps), S200 (200Mbps), and S400 (400Mbps). This version is the more popular of the two, and its cost is significantly less than that of FireWire 800.

FireWire 800, as inferred from its name, transfers data at 800Mbps, but at higher cost to the user because the production and demand of devices using it is lower. IEEE 1394 S1600 and S3200 offer data rates of 1.6Gbps and 3.2Gbps respectively, but not many products use these versions.

Since FireWire 400 offers a speed that more than meets the requirements of the project, it will be the only version of this interface to be considered for implementation, thus disregarding FireWire 800, 1600, and 3200.

Figure 3.2.7 FireWire Connector

3.2.8 Interface Selection

When looking at the technical aspects of each of the two wired-technologies taken into considerations, it can be noticed that both meet the data rate requirements, but

only one of them is widely available to use in conjunction with virtually any computer available nowadays. So USB2.0 seems to be the logical option when selecting a wired technology to interface the computer and the robotics hand, since the only other considering factor, price, it is under the budgeted price for both technologies. Plus, most development boards from Microchip, Atmel, Parallax, and similar vendors support USB2.0 out-of-the-box.

Interface	Availability	Data Rate	Compatibility	Cost	Overall
USB1.1	Poor	12Mbps	Regular	Low	Good
USB2.0	Excellent	480Mbps	Excellent	Fair	Best
FireWire 400	Good	400Mbps	Excellent	High	Good

USB vs. FireWire (Table 3.2.8.1)

Wireless modules will be used in at least three different subsystems on this project, these being the PodControl, Glove Interface, and the robotic hand it self, so the technology to be used has to be carefully selected since it will be quite difficult to change later in the design process.

For these devices (subsystems), certain requirements need to be met. Low power consumption is one of the most important factors, since it will reduce the operating cost (less energy/batteries needed), and the production cost and complexity, since the circuitry will be simplified. It will also directly affect other parameters taken into consideration, such as; size, since as more power is needed, a bigger battery is required. This pretty much eliminates Wi-Fi, since it consumes a much more power than the other interfaces.

Data rate is another big aspect of the selection process, and this case Wi-Fi has the lead, but it offers a data rate that far exceeds the requirements, so it needs not be included again in the pool of options. On the other hand, RF does not provide a data rate high enough to be usable on the project, and as such, it has to be discarded. Both Bluetooth and ZigBee offer similar data rates, and meet the current requirements, even though these can change in the future, possible altering the final selection of a wireless interface.

When taking into consideration other aspects, such as, price, size, development cost, industry support, and availability, both ZigBee and Bluetooth are almost identical, so a decision between these two will be left to the discretion of the project members, whom as of now are inclined to choose ZigBee for its slightly lower power consumption and data rates.

Interface	Data Rate	Power	Availability	Cost	Overall
Wi-Fi	54Mbps	High	High	High	Fair
Bluetooth	1-3Mbps	Fair	High	High	Good

ZigBee	250Kbps	Low	Fair	Fair	Good
RF	~	Low	Low	Low	Fair

Wireless Technologies (Table 3.2.8.2)

* The adjectives used in Table 3.2.8.1 and 3.2.8.2 to describe the difference interfaces were suitable for this specific scenario, and are not applicable to all cases.

3.2.9.1 USB Modules

Four (4) USB 2.0 interface modules have been selected as feasible options for implementation in the design. These are complete USB 2.0 interface cards that have the physical USB connecter on them already, thus ready to install and operate. Elexol makes all the modules except one. They are all reasonably priced and meet the requirements.

The fist USB module considered is the USBMOD5 by Elexol. This module offers many of today's best features and is a great all around module. It is available at a reasonable cost and can perform the required duties of a USB. It is one of the more expensive models that have been evaluated because it is a dual channel whereas the others are single channel. The dual channel configuration allows two independent lines of communication between the peripheral devices that you are connecting. This will ultimately allow for faster transfer rates and protect against any crossover. The next model is the USBMOD4, which is a slight step down from the USMOD5 but will perform well and is very similar. This particular model is a second-generation design with many modern features. The USBMOD4 is also a lower cost alternative to the more complex USBMOD5. The next USB module is the USB245R. This unit is very comparable to the USBMOD4 in that it offers many of the same features. The last model is a fully integrated unit manufactured by ApogeeKits. It is the model VM110 module and is a complete system card ready for any customization or personalization through the various programming languages available. This card also comes with all diagnostic software included in the bundle price. However, since it is a complete integrated unit, it is the most expensive.

The selected modules are good representation of the options currently available on the market. These were amongst the most economical and efficient that could be afforded for this project, and the fact that the same company produced them all made it easier to compare. Despite the huge market for this technology, its been considered that the selected modules represent the best options to interface the microcontroller, the development board, and the custom PCB. However, the drawbacks of using USB have been stated and they are still taken into consideration before final decisions are to be made. Table 3.2.9.1 shows the comparisons of the four different models.

USB Module Comparison

Manufacturer	Module #	Transfer Rate (bps)	Channels	Price
Elexol	USB245R	8Mbps	1	$30
Elexol	USBMOD4	8Mbps	1	$30
Elexol	USBMOD5	12Mbps	2	$40
Apoogee	VM110	N/A	1	$75

Table 3.2.9.1

3.2.9.2 ZigBee Modules

Since ZigBee is a relatively new wireless technology that's is being rapidly accepted by the industry, most MCU vendors design and sell their own ZigBee modules for added compatibility and lack of standalone options, as it is the case for companies such as Rabbit, Texas Instruments, NEC, Microchip, and Atmel. Unfortunately, the Propeller line of microcontrollers by Parallax is quite new, and the company only offers in-house modules for regular RF and Bluetooth. Thus, these market conditions considerably narrowed down the search for RF modules that are compatible with the selected MCU and meet the specified requirements of the project. RF modules from Digi (market leader), Texas Instruments, Ember, MeshNetics, CEL and FreeScale will be considered, being cost, size, and interface the predominant factors to take into consideration. Special consideration will be given to Digi and Ember since these two are leaders in the market.

Digi (MaxStream) Modules

Digi is probably the largest ZigBee module producer on the market, and features two different modules, each one with 3 different versions depending on the antenna used. One of the modules offered by Digi is the XBee® 802.15.4, which operates at 2.4Ghz, 1mW(+0dBm), 250kbps, and up to 90 meters range. It sells for $21 per unit, and the development kit is sold for $99 (includes two modules). The other module offered by Digi® is the XBee-PRO® which differs from XBee® only on power consumption (63mW - +18dBm), range (1.6Km) and price ($32). Given the limited range of use for the *hand,* the XBee® more than meets the requirements for the project. If selected, the XBee® module will be bought with either and U.FL RD Connector or a Chip Antenna due to the size factor.

Ember Modules

"Ember's Semiconductor products enable developers to choose between the EM250, a highly integrated and cost-optimized System-on-Chip, and the EM260, a ZigBee network co-processor that can be paired with almost any microcontroller. The EM2xx family minimizes external components and provides multiple RF connection

options for easy use with or without external PAs" [Ember, 1]. The module is priced at $9.68 at DigiKey.

The **EM260** is a module with a flash-based microprocessor running the EmberZNet ZigBee stack that easily allows developers to add ZigBee networking to their favorite microcontroller. A flexible RF interface minimizes external components in configurations with or without external LNA/Pas. The EM260 has the following specifications:

- -99dBm receive sensitivity
- +2.5dBm transmit power (+4.5dBm boost)
- Excellent 802.11b/g rejection and interference immunity
- Integrated TX/RX switch, minimizing external components
- Optional 2nd RF path for easy addition of external PA/LNA
- Single external crystal (24MHz)

The **EM250** will not be considered because of its architecture, that require the on-chip microcontroller to run the application, rather than using an external MCU as the EM260, which will imply learning a new language, which is not appropriate as of now.

MeshNetics

The company offers four (4) different modules under the line ZigBits, three (3) of which operate on the ISM 2.4Ghz band, while the other operates on the 900Mhz band. The three modules that operate on the 2.4Ghz band are the Balanced RF Output (BRFO), Dual Chip Antenna (DCA), and the AMP modules. The AMP module is designed specifically for long-range applications, and therefore its current consumption is much higher at 23mA in receive mode (RX), and 50mA in transmitting mode (TX). The current consumption is the same, 6μA, for all three modules in sleep mode. The DCA and BRFO modules have the following common specifications; Supply voltage 1.8-3.6V, current consumption 19mA (RX) and 18mA (TX), Flash memory 128kB, RAM 8kB, EEPROM 4kB, and operating temperature -40 — + 85ºC. The compact size of these modules, 13.5mm x 18.8mm for the BRFO and 13.5mm x 24mm for the DCA, makes them perfect for the Glove Interface and PodControl respectively.

MeshNetics also offers development kits for the platform, which include "3x MeshBean development board featuring different antenna types, 3x ZigBit OEM modules mounted on MeshBean boards, 3x USB cables, 2x external interface cables, and 1x software & documentation CD" [Meshnetics, 1]. Being a perfect kit to streamlined the development and testing procedure.

CEL (California Eastern Laboratories)

CEL offers the option to acquire the ZigBee transceiver alone, a ZigBee module, and/or the development kit. Their development kit unfortunately includes more tools than needed for this project, and its price is prohibitive on the budget. Their modules are targeted to applications that are out of the scope of the project, and therefore are not viable alternative when considering that much better alternatives has been already discussed. Regarding the transmitter, the specifications are pretty much standard with the only notable advantage of having an embedded *voice CODEC*, but since no voice recognition is intended for the hand, this feature does not offers any benefits.

Texas Instruments (TI)

Texas Instruments offers a wide variety of ZigBee RF ICs providing great flexibility to the final user. Some of the ICs offered by the company are the CC2520, CC2420, CC2430, CC2431, and CC2480A1. Most of them operate 2-3.6V, and a few operate at 1.8-2.6V. The CC2420 offers the lowest RX current consumption at 19.7mA, while all other range between 25-27mA. The company also offers specific development kits or each of their commercial ICs, but the lack of documentation on these may hinder development.

FreeScale

FreeScale offers ZigBee ICs, transceivers, and controllers. But only the transceivers are of interest for the projects, since the idea is to integrate it into the designed PCB. Five transceivers are currently being offered by MC13191, MC13192, MC13201, MC13202. All of them offered standard specifications, but use O-QPSK modulation. These transceivers are moderately priced, and meet all the requirements for the project, their only downfall in the development boards offered by the company is that the provided documentation is intended for engineers with vast experience in the field of wireless communications.

3.3 Sensors

A sensor is a piece of equipment that measures any kind of entity; it could be force, temperature, light, motion, and many more things. Sensors transfer that information into digital or analog data and provide a link from the outside world to a machine. Luke's hand must have the assistance of sensors to provide a great sensitivity to everything around it. Luke's hand has to be able to detect the presence of object when they are close to it. It has to be able to feel the objects when the fingers are touching something in order to know the shape of any kind of thing that it is about to grab. Luke's hand also has to be able to control the amount of force that it applies to the object in order to grab it and at the same time not to crash it. One of the most important functions that this robotic hand must have is the ability of sensing slipping to correct the amount of force applied to the object for a better grapple. For

this numerous functions it is very important the integration of sensors into the project.

3.3.1 Touch Sensor

Touch sensor works like a resistor; it shows how much resistance is applied to a surface and that is converted into pounds of force. There are going to be at least five sensors integrated into the hand. Each sensor will be attached to the tip of every finger. The way the touch sensor is going to be used is somewhat alternate of its original purpose of sensing how much force is applied to a surface.

The touch sensors give an accurate measure of the force applied to an object, but the problem is that every object has a different force tolerance. For example, an egg will crack if the hand will use the same force as if a bottle was grabbed. The touch sensors are going to be used as a way to tell the hand that when the fingers are closing they have touch some object so they need to stop contracting. By integrating one sensor into each finger, it will alert the hand that all the fingers are in contact with an object and that they are ready to grab. The touch was considered into the project because it will provide a good link with the object and the fingers.

Trossen Robotics will provide the touch sensor that will be used. The .2 Inch Force Sensing Resistor (FSR) is the force sensor that fits the projects criteria. The FSR has diameter of .2 inches and a length of 1.75 inches. The actual sensing part will be located in the tip of the sensor, where the .2 inch circle is located. This circle has a rear adhesive capacity, that will be glued to the tip if the finger. Two legs follow the sensing circle; which conduct the electricity and provides the interface with the voltage divider. This sensor must be connected to a voltage divider before it is connected to the microcontroller.

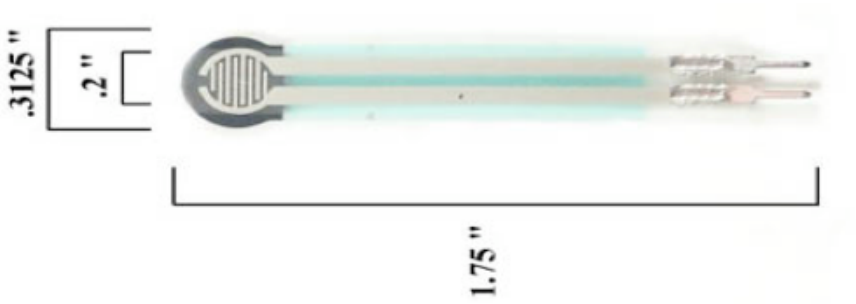


Figure 3.3.1.1 FSR touch sensor

3.3.2 Optical Sensor.

The Optical sensor was one of the sensors considered to integrate to the hand for the utility of sensing slipping. The optical sensor is going to be detached from a mini

mouse and integrated to the palm of the hand. The optical sensor is a considerable idea because of its capacity of sensing movement in any direction. This sensing ability can be used as a tool to sense slipping. If an optical sensor is attached to the palm of the robotic hand, it will be able to sense when an object is moving in any direction. When the fingers don't apply the right force to grab an object the only way to correct that mistake is if the hand could detect that the object is falling. The optical sensor provides a great feedback to the hand because it will sense that the object is falling in any direction and that the hand must apply more force to stop slipping.

When the optical sensor was extracted from the mini mouse the following parts were observed: Circuit board with the optical sensor, chip, and all the electrical parts. A plastic mount where the prism was located so the LED can be detected when reflected from a surface. In order for the sensor to work, the plastic mount has to be perfectly horizontal and aligned with the surface of the hand. In senior design two, with some more field research, the faith of the optical sensor will be determined. The only way to know if the optical sensor will work in favor of the

Project will be by attaching it directly and testing it with different type of materials. The optical sensor has to be able to sense any type of surface and color, even colorless. Ideally, this sensor could provide a great asset to the hand to perfect its functions and stop the object before it falls.

3.3.3 Potentiometer.

The potentiometer is no more than a touch sensor that not only provides information about how much force is being applied; but also shows the mark where that force is being applied. The fact that the potentiometer has the ability to show where force is applied on a plane made it a candidate for the task of slipping.

The idea was to stick a potentiometer in the palm of the hand since every object grabbed will touch the palm and apply a force towards it. The potentiometer will detect the force of the object and the position on the palm where it is applied. If the object is slipping then the force applied will shift down, so the hand will assume the object is slipping because of the absence of force on the top of the potentiometer. This absence of force will indicate the fingers to grab harder to stop slipping. Like the other sensors, the final word will be given after testing. The problem with idea has to do with the different kind of objects that the Luke's hand has to grab. Objects like tennis balls, small cups and apples are fine, but when it come to larger objects like bottles, rods and sticks the potentiometer hypothetically wont work. If the potentiometer is measuring a force across the whole palm, and a bottle is slipping, the potentiometer won't be able to detect a change in the force until the top of the bottle has fallen and passed the top of the palm. Then the potentiometer will be able to detect a change in absence of force. This doubt will be confirmed when a further research and testing takes place in senior design two.

The SoftPot Membrane Potentiometer – 50mm was chosen. This potentiometer can detect and vary the resistance from one hundred Ohms up to ten thousand ohms. It has a total length of two and a half inches but only two inches are for active sensing. It has three pins, which one is for ground, the other for voltage and the last one for sending the signal to the computer of how much force is applied linearly. The potentiometer has to be connected to a voltage divider before it is connected to the microcontroller.

Figure 3.3.3.1

3.3.4 Proximity Sensors

A proximity infrared sensor is one sensor that the Luke's hand must have in its arsenal. Luke's hand has to be able to detect if there is an object in between the fingers and the palm. One of the functions that the robot hand must do is to detect any kind of object and after detection, start contracting the fingers to grab it. For this function, a proximity sensor will make that task easy. The proximity sensor is attached to the palm of the hand, where is the center and place where all the objects will meet.

The sensor has to be able to detect an object that is located from zero centimeters up to two centimeters from the palm. The sensor has to detect the object so close to the hand because if the fingers start contracting too soon, then the hand will miss the object and won't be able to grab it. The proximity sensor only detects a mass approaching to the palm; which means that it does not matter if the objects have different color, surface and shape. The proximity sensor will be activated only when the fingers are open, because it will be programmed to detect motion before the fingers grab an object. This proximity sensor needs a voltage divider also before it is connected to the microcontroller.

The sensor chosen is the Sharp IR Distance Sensor GP2D120 found in Trossen robotics. This sensor can detect accurately objects from four centimeters up to 30 centimeters but it can also be used from zero centimeters to fifty centimeters. This proximity sensor works along the voltage divider and needs a voltage between 3 and 5 volts.

Figure 3.3.4.1 proximity sensor

The proximity sensor will probably be placed in the bottom center of the palm so if any object of any size approaches the hand, it will be detected and grab with accuracy. The wires will be run through the inside of the palm towards the bottom to be connected with the microcontroller.

3.3.5 Sonar Sensor

Ping Ultrasonic Sensor is a parallax product that assists the application to detect an object within certain distance range and triggers the robotic hand to react according to the program loaded in the microcontroller. This type of detector is very effective and works with more accuracy than IR sensors to detect objects because the light of the room where the test is performed is not a factor, also the objects to be detected can be of white color or transparent, which is an inconvenient for IR sensors. This sonar sensor can perform measurements for moving or stationary objects. The pulse is transmitted from the component and distance-to-target is determined by measuring the time required for the echo to go and return to the receiver portion of the device.

The sonar sensor provides precise, non-contact distance measurements in the range from 2 cm to 3 m, and requires 20 mA to function with very low power consumption. The figure below shows the PING Sonar sensor that is used in the project.

Figure 3.3.5.1 Sonar Sensor

3.3.6 Optical Interrupter

The idea of using an optical interrupter as a way to detect slipping was obtained by looking a computer mouse. The rolling motion of the upper wheel to move the page up and down inspired the idea of sensing slipping. Taking out the optical interrupter off the mouse and place it on the palm of the hand. When the object is secure, after the fingers grabbed the object, it will make contact with the wheel attached to the palm. If the object starts slipping then the wheel will start moving; which will alert the hand that there is not enough forced applied and the object is falling down.

The optical interrupter is actually a laser that goes across two little towers. Between this two towers there is a wheel that looks like a gear. This wheel has wholes and they are called digital wholes. When the wheel interrupts the laser, the sensor detects movement and in the case of the robot hand, it senses slipping.

There are a couple of concerns about the optical interrupter that has to be researched in senior design two before it is chosen for the project. Some of the materials of the objects that are going to be grabbed can affect the rolling of the wheel. Glass is very slipper and because it does not create a much friction with the wheel, it cannot give accurate data when the object is falling. Other concern is the position where the wheel of the sensor will be located in the palm. There is a variety of object with many different shapes that will be grabbed by the Luke hand. The fact that those object have different shapes present the possibility that when the fingers contract and grab the object it will not make full contact with the wheel. If there is not enough contact between the wheel and the object, the data will not help the hand to accomplish the function to its full extent.

The figure 3.3.6.1 bellow will show how the interrupter could be attached to the palm of the hand if it is chosen in the project. There are many places that the interrupter can be attached to the palm but in the figure below shows hypothetically the best location for it to accomplish it function.

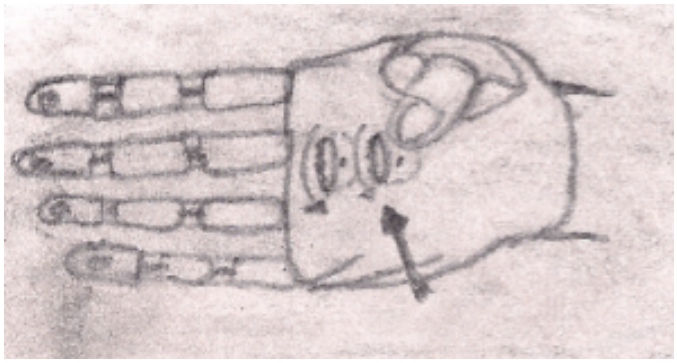


Figure 3.3.6.1 Optical interrupter

3.3.7 Flex Sensor

The flex sensor manufactured by Spectra Symbol is utilized in the globe interface designed as one of the control modes for the robotic hand. It is commercially available in Trossen Robotics and comes in 4.5 inches of length. The flex sensor varies its resistance across the pads while it is bent. The flex sensor resistive statistics are: Straight (extended): about 9000 Ohm; 90 degree bent: about 14000 Ohm; 180 degree bent: about 22000 Ohm.The flex sensors runs with 3.3 volts and they are connected directly to an A/D converter that will give a 12-bit representation of the voltage variation. This flex sensors are used in the Glove Interface as the primary sensor for detecting how much the finger is being flexed. The sensor will be attached to the back of the fingers and will provide an accurate reading relationship between bending and voltage. Figure 3.3.7.1 shows the flex sensor use in the project.

Figure 3.3.7.1 Flex Sensor

3.4 Motors

A motor is a device that will take electrical energy and convert it to mechanical energy. There are two types of electric motors AC (Analog Current) and DC (Analog Current); but the one by far used in the robotics and the one chosen for this project will be DC Motors. However, there are different types of DC motors, but the most commons are: Continuous, Stepping and Servos. A continuous DC motor rotates continually when power is supplied and the shaft stops only when power is removed or if the motor is stalled because it cannot longer drive the load attached to it. Motors can be used in small devices like a remote control car to big engines to produce energy in a power plant. A stepping DC motor is not very different from the continuous motor. The stepping motor rotates a few degrees when the power is supplied and then it stops. If a continuous motor wants to stop after a few degrees,

the power supply is going to have to be pulsed to mimic the stepper. Stepper motor will not be used in this project because there is not a valuable function for them in order to use them

3.4.1. DC Continuous Motor

DC motors can have different speeds and size. The speed of the motor depends on how much current it is supplied to it, and also can be manipulated by adding gears. The gears have different number of sparks and the ratio between two of them can either transform a fast motor into a torque motor or a torque and slow motor into a faster one. A DC motor was considered for the individual movement of the fingers in the Luke's Hand. Many design obstacles had to be taken into consideration for the use of this motor. In order to get single finger movement, Luke's Hand needs a motor for each finger. Each finger will have a string attached to the tip, and by pulling that string the fingers will be able to contract and bend. This means that the only way that a DC motor can pull the string down will be by rolling it on a rod. To rotate the rod, the mechanical force of the motor will have to be transferred to the rod by using gears.

The whole mechanism of pulling the string will need more space for the motor with the gears together that are in charge of moving the rod in order to roll the string so the finger can be contracted. There are four fingers in the hand that will have independent movement, which means that a lot more space will be needed to have five different mechanisms to move each finger independently. It is too much space and pieces used to control only the movement of the fingers and that is not counting sensors and power. Figure 3.4.1.1 shows the design and layout of the robotic hand if a DC motor were to be used to control the individual movement of the fingers and the amount of space that it will be lost if a continuous DC motor will be used in this project.

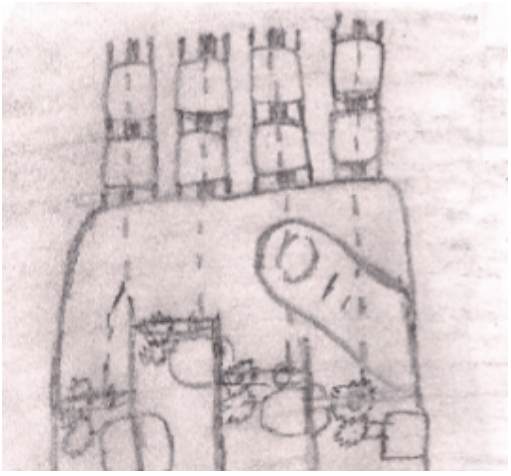


Figure 3.4.1.1 DC motor finger movement

Continuous motors will not be totally excluded from the project since the good thing about the DC motor is that we can create a lot of torque when we play with the gears. Torque is very important for the performance of the robotic hand and arm. It is the force applied to grab and hold objects of any kind of weight and also being able to raise the arm. A motor will be stronger depending on its size. For this project we cannot use a big motor to move the fingers and get more torque because we have limited space. This means that if a small motor is used then more gears are going to have to be added in order to transfer speed into torque, so more space will be needed. For this various reasons a DC motor is not convenient for the individual movement of the fingers and will not be used in the robotic hand.

One of the requirements for the robotic hand is to grab objects without dropping them. The hand needs some kind of vertical movement of the arm in order to test the grappling of the object that was held. When it comes to producing a lot of force to raise the arm, a DC torque motor will be excellent. The DC torque motor will have the sufficient strength to raise the whole weight of the arm and hand with no problem. Since it is a torque motor, it won't count on speed, so the arm will be raised slowly but firm. In the design section, where the elbow mechanism is analyzed, the way the torque motor will be integrated is shown and explained. Figure 3.4.1.2 shows a torque motor similar to the one that will be attached to the robotic arm used to apply the force.

Figure 3.4.1.2 DC torque motor

3.4.2 DC Servomotors

Servo Motors is a special type of DC motor; which combines a continuous DC motor with a feedback loop to ensure the accurate positioning of the motor. There are two types of servomotors: Hobby and robot servomotors. The difference between them is not their function, but most of it is their physical and mechanical performance.

Hobby servomotors are made of plastic shell, with plastic gears and components, while the robot servos are made of metal shells, gears and components to increase their endurance. A hobby motor usually has a maximum range of rotation of one hundred and sixty degrees. A robot servomotor usually has twice as much as the rotating range of a hobby servo and it is usually around three hundred degrees but can be greater than that. Hobby servos do not have too much torque force and it is another category in which the robot servomotor is superior. Hobby servos can deliver a 2.6/3.0 Kilograms per centimeter of torque. Having a greater torque capacity signifies a huge advantage when designing robots. With more torque this means more power needed. The hobby servomotor draws much more less current than a robot servomotor. A hobby servo uses around 3v-6v while the robot servo draws 7v and can be up to 12v. All of these big different specifications will vary the price of the servo. There is a mini-servo that is also used in robotics. This mini-servo is usually the size of a twenty-five cents coin. This servo is usually placed in the joints of the robots because it is small and actually a decent amount of torque.

The hobby servo was selected to control the movement of the fingers in the Luke's hand. This type of motors requires a constant 4.8 to 6 V and in order to turn the motor a digital signal is required. A servomotor is composed of three cables a +Voltage, Ground and signal. Servomotors are designed for limited rotation rather than for continuous rotation, like a Continuous DC or a Stepper, but the goal of these motors is to achieve accurate rotational positioning over its range. The servomotor is positioned by using pulse width modulation (PWM), this pulse controls the movement of the motor and determines the position of the servo.

Usually the pulse varies from about 1ms to about 2ms; these pulses are sent some 50 times per second. When a 1ms pulse is sent the servo is commanded to turn all the way in one direction, but when a 2ms pulse is sent it moves all the way in the other direction. So pulse of 1.5ms commands the servo to move to its center or neutral position. An average servo can rotate a full 60˚ in a quarter to half second. It's good to mention that the actual length of the pulses is fairly constant at 1ms to 2ms for most manufactures. But it is always good to check the specifications for the servo firsts.

DC servos are often used in robotics because it is easier to control their position and speed. Servos also come in different sizes. Their speed or torque can be controlled by gear ratio; the only difference is that it will only move on a range of rotation instead of a whole three hundred and sixty degrees. A servomotor can also be manipulated as the DC motor to create more speed or torque. As mentioned before, we need to pull a string in order to contract the finger. A servomotor is perfect for the job because the string can be attached directly to the arm of the servo. The servo

will pull straight down without the need of wrapping the string on a rod like the DC motor. Another advantage of using a servo is that there are zero chances of having gears getting out of place and causing a failure or ropes wrapping around the rod in a different way every time. A lot of space is saved since there is no need for gears or rods and since the only thing going though the palm now are five ropes. Depending on how heavy objects we want the Luke's hand to grab, we can get a specific servo with the necessary torque force.

The chosen servomotor, Futaba S3004, has a maximum degree of rotation of One hundred and sixty degrees when a 2ms square wave signal is applied. It has a speed of 0.11/0.09 seconds per sixty degrees. This servo can deliver a 2.6/3.0 Kilograms per centimeter of torque. This torque is enough for contracting the fingers and giving the force necessary to grab objects with in specifications of the project. The selected PWM given for expanding the tendons is 1.9ms and for contracting is 0.9 ms. Futaba S3004w servomotor used in this application is shown below in Figure 3.

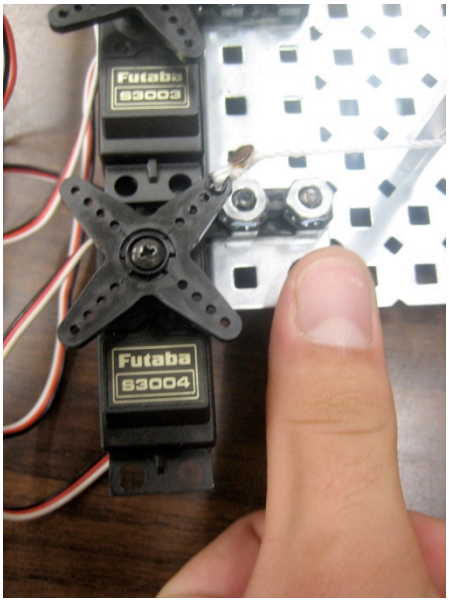


Figure 3.4.2.1 Futaba S3004

3.5 Research on microcontrollers

3.5.1 Overview

This research is intended to obtain better insides on what kind of microcontrollers are available in the market that are suitable for the type of project the group has embarked on. A Robotic hand is a complex task in the sense that it requires several degrees of freedom and coordination on several movements in order to have a

remote approach to what a real human been hand performs. Although, there is a vast variety of microcontrollers and manufactures on the market, the architecture of them is whether RISC or CISC; therefore, the selection of the microcontroller for a robotic hand needs to be carefully considered based on critical parameters such as documentation available for the chip, compatibility with the design specifications of the robotic hand, technical support offered by the vendor, cost of the microcontroller, integrated development environment (IDE), and overall efficiency.

Searching for microcontrollers available in the market is an endless task because there are so many options, making it inconvenient to consider every possibility due to the time constrain for the project, so the search was narrowed to the microchips most used in the robotics industry as well as the educational community. After finding the most commonly used in the industry by robotics developers, only few microcontrollers were considered to be evaluated taking in consideration the critical parameters previously mentioned. The brands and manufactures most representative in the microchips industry considered for this research are the following:

3.5.2 Microchip Technology Inc

One of the most important products of Microchip Technology is the Series of microcontrollers called PIC. This set of microcontrollers comes in different packages and pin sets as well as a diverse range of memory technologies such as Serial SRAM, EEPROM, and Flash, which make the company to provide a huge selection of chips for almost every necessity. The diversity in technology, size, and package also allows Microchip to offer a wide range of chips suitable for nearly almost every application at competitive prices. Today, the company has more than 400 different options to choose from, which make it one of the leaders in the world of microcontrollers.

Advantage of using Microchip products is the fact that for more than 20 years this company has been present in the market positioning its products and developing new ones at the point that some universities teach embedded systems as well as other electronic classes based on the PIC series of microcontrollers. As a consequence, the documented use of this product in the industry is one the most complete to rely on because present and future professionals would incline for using a very popular chip, which they already know how to use.

The integrated development environment is another important factor, which has been evolving during the time. Recently Microchip has launched its MPLAB IC3 after having the ICD2 for several years. Despite the quality of the IDE, which has not been disputed, there are criticisms for the high price that Microchip charges for the standard edition of that software. Although, there is a free student edition version of the ICD2, it lacks of several functionality that the standard edition has such as code optimization and other interesting features. Another down side of the IDE usage is that there is a learning curve, which can be inconvenient for a short and /or single project needing to be developed with time constrains. Nevertheless, learning the

integrated development environment can payoff in a long run if there are several projects implementing the usage of PIC microcontrollers.

PIC microcontrollers are programmable by assembly language, Basic, or C language. The compilers provided by Microchip in certain series of chips are free to be downloaded, and additionally the GCC C compiler can also be used to program the PICs, giving it the flexibility of cross platforms and universality among different operating systems. Among many other advantages of using the PIC microcontrollers are the facilitation of tutorials on line from Microchip and other web sources, sample code over the internet, entire internet blogs of information regarding the microchip, reference manuals, training sections, extensive libraries, and additional literature.

Prices on the PIC family from Microchip are competitive compared with the other manufactures on the market; therefore, cost is not a fundamental difference to be considered at the time of making a selection. PIC microcontrollers can be purchased by amounts from one to as many as needed, and the price range goes from $0.60 cents up to $8.00 for a single unit.

Atmel Corporation

In the industry of microcontrollers, Atmel has been gaining terrain with a series of microchips, which arguably are a bit faster than the PIC family from Microchip at comparable prices. The Atmel family of microcontrollers is known as AVRs and they share similarities with the PIC in its Harvard architecture implementation, packages, functionality, memory technology, pin sets, and even popularity among the robotics industry. Maybe some of the differences are the instruction set number (35 for PICs and 130 for AVRs) and the implementation of real interrupts by AVR, whereas PIC pulls all its registers to check which needs service.

Perhaps the greatest advantage of using AVR chips instead of PICs for the senior design project would be the AVR Studio Integrated Development Environment. The well known inconvenient of the learning curve required to program PICs from Microchip is one of the main reasons for people migrating to AVR microchips. The AVR Studio IDE for Atmel microcontrollers have been gaining popularity due to the simplicity of the software and the powerful debugger that comes along with the IDE. Those outstanding tools are available to download free of charges for the users. Like the IDE from Microchip, AVR Studio is also developed only for Windows operating systems, but the chip can be programmed with a GCC C language, or Assembly language as well. Moreover, there is a Unix command line assembler, which allows Mac OS and Linux users to program the chips in its own platform.

3.5.4 Parallax Inc

Parallax offers two different valid options to be considered for the robotic hand project. The architecture of microcontrollers offered by Parallax is Harvard Architecture with two separated memories and separated address busses, one to address the program set of instructions and the other to handle the data set. Even though Parallax like any other company has a wide set of options, this research is going to focus on two of specific products such as Basic Stamp and Propeller.

3.5.4.1 Basic Stamp

The Basic Stamp is a high-level language programmable microcontroller. It is based on the PIC16C56a and PIC16C57c from Microchip Technology, which has been modified with a firmware to maximize the potential usability implementing interpretation of BASIC Tokens. Also additional hardware and software are included in this product. Those modifications allowed positioning the microcontroller on top of the market for users beginning in robotics as well as hobbyist.

Parallax has today six different versions of the Basic Stamp solution for customers looking into an easy solution with a high-level language approach. The next figure shows the different solutions offered by the company in the market.

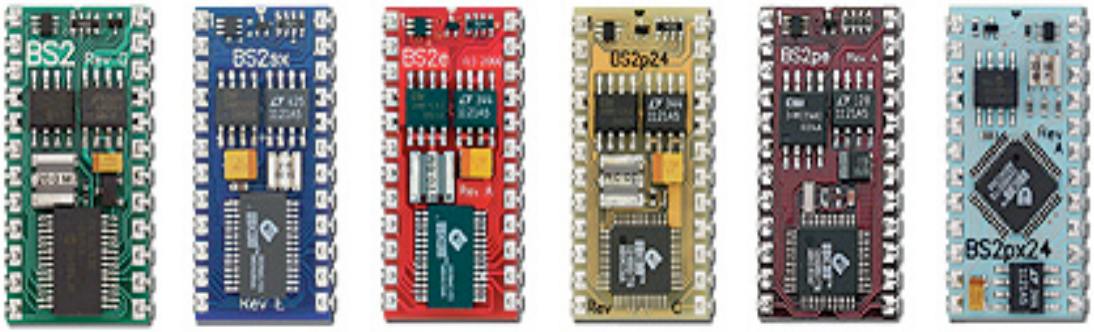


Figure 3.5.4.1

This is the set of Basic Stamp controller offered by Parallax. Reprinted with permission of the copyright owner, Parallax, Inc © 2008. All rights reserved.

Thus, the success of the chip is due to the combination of the PIC characteristics such as low power consumption due to its fully static CMOS controllers, the chip's speed performance of up to 5 MIPs (millions of instructions per seconds) at clock frequency of 20MHz, the Harvard Architecture, which exploits the concept of pipelining by overlapping cycles for instruction fetch and instruction execution, and the friendly interface added for people without experience in programming languages. Such interface is base on the computer programming language BASIC (Beginner's All-purpose Symbolic Instruction Code.) Thus, for the senior design project, where results are expected in a timely manner, Basic Stamp microcontroller is a valid option.

3.5.4.2 Propeller

Parallax also offers its new product as a big contender in the field of microcontrollers. The Propeller microchip, as many people describes it is "a different beast." Propeller has not only Harvard Architecture to separate the management of memory for data and program instructions, but also a truly pipeline design with 8 32-bit processors in series called cogs, which can execute different tasks in parallel simultaneously, giving a throughput of up to 160 MIPs, 20 MIPs per cog. Those cogs may work independently or in cooperation's by sharing resources through a hub, which is the coordinator of resources in the chip.

This microcontroller is a brand new development, which requires an input voltage of 3.3 volts giving very low power consumption at a tremendous high speed as mention above. The speed of the clock is controlled by specific quartz, which can be replaced to obtain faster rate of data to process, or slower rates and therefore, higher or lower throughput given in MIPs by each of the cogs that sum up to the total performance of the microcontroller.

Next figure is presented as a graphical representation of the microchip to illustrate the architectural development of the truly parallel microcontroller.

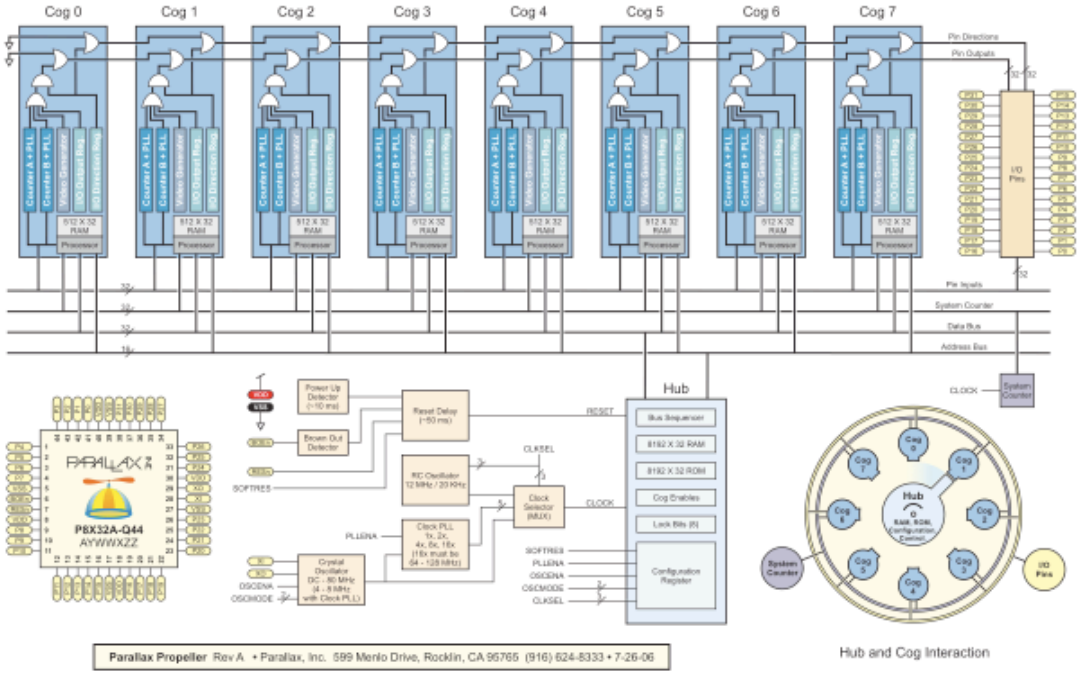


Figure 3.5.4.2.1

This is the Block Diagram of the Propeller Microcontroller. Reprinted with permission of the copyright owner, Parallax, Inc © 2008. All rights reserved.

At the time of programming this chip, developers have each one of the processor available to decide how and when they should be employed. All the cogs share the same clock signal to give them a needed synchronization. For the purpose of a robotic hand, these characteristics are desirable and it is easy to visualize one of the cogs generating a pulse width modulation, while other cog could be controlling the motor servos in order to move the fingers at the same time if required; in the mean time, other cog could be supervising the input values of implemented sensors in the application. The possibilities are wide to exploit this microcontroller, and in at later section that topic will be fully developed.

Parallax provides two different programming languages to code the Propeller: Spin, which is an object-oriented programming language, and Propeller Assembly as the second alternative. Spin, is a programming language developed exclusively for Propeller by its manufacturer as a response to complex integrated development environments, which sometimes turns away users even from good products. There is also a third party IDE called ImageCraft, by ImageCraft, which has an ANSI C development tool for C language. The inner Propeller libraries are directly supported by this IDE.

Parallax has three different packages and two pin sets for its Propeller microchip. Its internal RC oscillator runs at a maximum of 12 MHz, but with some hardware manipulation achieves up to 80MHz of external clock speed. The chip also has 32 I/O pins and current source sink per I/O of 40 mA. Additionally, Propeller comes with pre-set libraries for interfacing with mice, keyboards, RF, video, LCDs, stepper motors, and sensors. Thus, Propeller provides a high level integration for the users to speed up development of applications.

The next figure shows the described integrated circuit packages that Parallax has available for users today in the market.

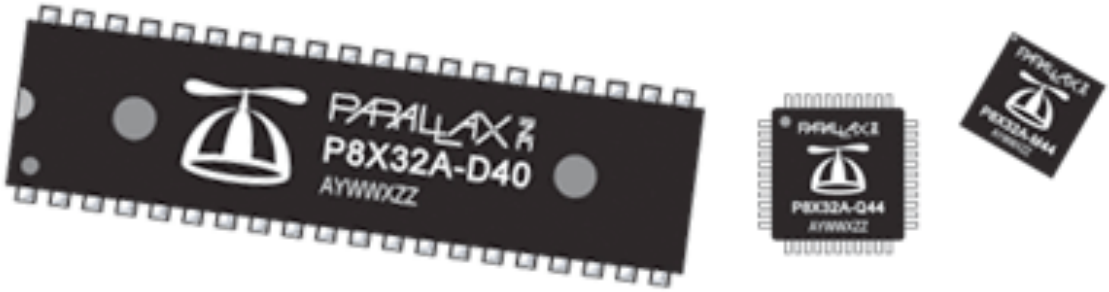


Figure 3.5.6.2

These are the Different IC Packages offered by Propeller. Reprinted with permission of the copyright owner, Parallax, Inc © 2008. All rights reserved.

At simple glance, the down side of this powerful microcontroller is the price. The Propeller is available in the market for $12.99 for a single chip. This price compare with other microchips in the market is one of the highest, but also at the same time there is no other microcontroller with similar characteristics to make a fair comparison. Now from an operational prospective, Propeller easily does the job of several other microcontroller together due to its unique architectural design. Therefore, after math Propeller is right for the money and unbeatable for its functionality.

Documentation has been building up quickly since 2006 when first appear Propeller in the market. So far the microchip have been attractive to engineers as well as hobbyist for very different reasons, which are efficiency and architectural design from one side and easy to work with due to the friendly interface from the other side. Propeller is found in applications related to industries such as Art, Commercial / Industrial applications, Medical, Home, Robotics, Science and many more applications.

3.5.5 Zilog, Inc

Zilog is another manufacturer of microcontrollers with a great line of product oriented toward wireless and wired telecommunications as well as embedded health and fitness applications, intelligent home controls and others. Their most popular line of microchips is the Z family with an architectural design based on CISC technology. Most recently Zilog have introduced their new line called ARM-based with RISC architectural design and 32-bits MCUs.

Before Parallax in conjunction with ImageCraft developed the ANSI C compiler for the Propeller chip, some C programmers started to migrate to ARM-based microchips, in which Zilog provides a free ANSI C compiler for the ARM-based line of microcontrollers in order to provide robust development applications to the users. There is not much information other than from the vendor about the integrated development environment for the different series of microchips. The IDE is called Zilog Developer Studio II, and it is claimed to be flexible and functional to speed up the final product of developed applications with any of their microcontrollers.

Although Zilog's chips come with a broad range of memory sizes, very useful peripheral sets, and widely used in several embedded applications that enable the company to offer solutions for about every application, there is not substantial information related to the Z family of chips being used for implementations such as the robotic hand or any other similar application that can give to the senior design group fundamental feedback.

3.5.6 Texas Instruments

One of the indisputable leaders in the industry of microchips is Texas Instruments with a huge variety of product. Despite their big presence in the world of microcontrollers, their impressive goodwill, and great documentation and support on their products, TI has not been taking the lead in the robotics industry as it is in scientific and medical applications.

A promising product is their microchip MSP430 Ultra-low power, which has been used for several scientific applications and now promoted for the robotics industry. The company is making the effort by putting a set of videos on the Internet, where they show the power and versatility of the chip to be used in a wide range of applications. But still it is to soon to see the expected results translated into spread documentation over the Internet.

TI also offer a line of chips called ARM-Based. That line of microcontrollers comes with 16-bits and 32-bits as well as and interesting variety of memory sizes and technologies. So far the main consumer of this line of products is the automotive industry. Another product from TI is the chip so called C2000 claimed to perform up to 150 MIPs of DSP processing power on its RISC architecture platform. The microcontroller comes with peripheral integration for an easy to use implementation such as motor driving controllers, sensing applications and others.

There are several software development tools available to the user among with a huge amount of libraries and header files ready to be used. One of the software tools is the Code Composer Studio suitable for assembly code or C programming language developed specially for the C2000 microchip. TI also has a great set of sample code for different type of implementations. But outside TI, online there are not so many discussions about using TI chips for robotics or even for hobbyist.

The prices of TI microchips vary but tend to go toward the upper side. As an example, a 16-bit, general purpose chip with 32k of flash memory from TI cost about $9.80, whereas a PIC with similar specifications can cost around $5 each. Nevertheless, customer support from TI seems to be unbeatable from their web page.

Bottom line, while TI provides a great source of documentation to back their wide portfolio of products, there is not enough signal of acceptance in the robotics industry or related fields that can be used as reference to overcome any inconvenience at the time of using a TI microcontroller for the senior design project.

A summary of the research about suitable microcontrollers for implementing a robotic hand is presented below in a table format with the microcontrollers previously discussed. The table includes the chips' manufactures and their most important characteristics such as architectural design, their most representative product line that can be used for the robotic hand, and the most important advantages and disadvantages of those chips. Table 3.5 below presents a summary of the research for selecting the appropriate microcontroller.

Manufacturer	Product	Architecture	Advantage	Disadvantage
Microchip	PIC series	Harvard Architecture, mono-processor	Widely used for more than 20 years	Learning curve of IDE interface, which slows the developing process. Some issues reported with compilers and compatibility
Parallax	Propeller	RISC multi-processor	Easy interface for fast results, object oriented programming language	Programming language developed exclusively for this chip; called Spin. Price
Manufacturer	**Product**	**Architecture**	**Advantage**	**Disadvantage**
Parallax	Stamp series	Harvard Architecture	Commonly used by hobbyist, easy IDE based on Basic programming language. Huge amount of discussion groups sharing their thoughts.	IDE developed for people without experience in microcontroller leaving the code expose to the world without security, small glitches with the debugger has been reported
Zilog	Z series	CISC Architecture	Price.	Convoluted instruction set. Not so popular in the robotics industry thus, lack of documentation on solving issues.
Atmel	AVR series	Harvard Architecture, mono-processor	Free compiler constantly updated with cross platforms and high portability of code.	Ethernet, CAN, USB, and RF are not supported for the smaller chips.

Table 3.5

The above table is a summary of the research about suitable microcontrollers for implementing a robotic hand

Even though cost, efficiency, and technical support were important in the process of selecting the most appropriate microcontroller, documentation among Integrated Development Environment offered with the microchip were the key factors for the final decision. As a result, this research is inclined toward choosing Propeller microchip based on the wide and complete documentation already existing on line as well as complete solution projects for diverse applications posted on Parallax web site to demonstrate the simplicity of using Spin programming language to code and integrate this revolutionary microcontroller with about any peripheral device available in the market.

3.6 Development Boards

3.6.1 Introduction

A development board is recommended to be use in order get familiar with the instruction set as well as the environment implemented to program any of the microcontrollers today offered in the market. Each one of the vendors offers several development boards, and depending on the customer needs and budget, that fundamental tool is selected.

3.6.2 Propeller's Starter Kit

Most of the development boards come with the necessary voltage regulator, power supply, USB port and other hardware interfaces. Additionally, C program and/or Assembly language are the default computer programs to code the microcontrollers. This process is done with the assistance of an editor computer program where the code is written, a linker, a loader, and a compiler.

The IDE provided by Parallax works basically in the same way as the other IDEs provided by their manufacturers or some times third party solutions. Perhaps one of the major differences is that Propeller's development board from Parallax comes with a unique computer language, which is claimed to be intuitive and English like high-level language, designed exclusively for Propeller microchip.

Spin software was developed and written by Chip Gracey, the owner and founder of Parallax, who very well knew the insides of the PICs from Microchip. Mr. Gracey of course understood the weakness and lack of simplicity of Microchip's IDE and used it as the starting point to develop Parallax's first product, the Basic Stamp, which used Basic as the programming language. Later, after having success with their first product, and many years of hardware design and architectural development for their latest product, the Propeller, they decided to write their own programming language written exclusively for that microcontroller, Spin.

Parallax claims that the IDE they offer alone with Propeller is an intuitive language that can be easily learned in couple days, leaving behind the straggle of adapting to a new programming environment. Parallax even took in consideration clear and colorful divisions between the main sections of the code. By implementing different colors for each of the main sections in the code, Parallax accomplished to get the users focus on the area where they most likely could have concerns. In the next figure a sample of the code is presented to illustrate the computer language Spin for the microcontroller Propeller.

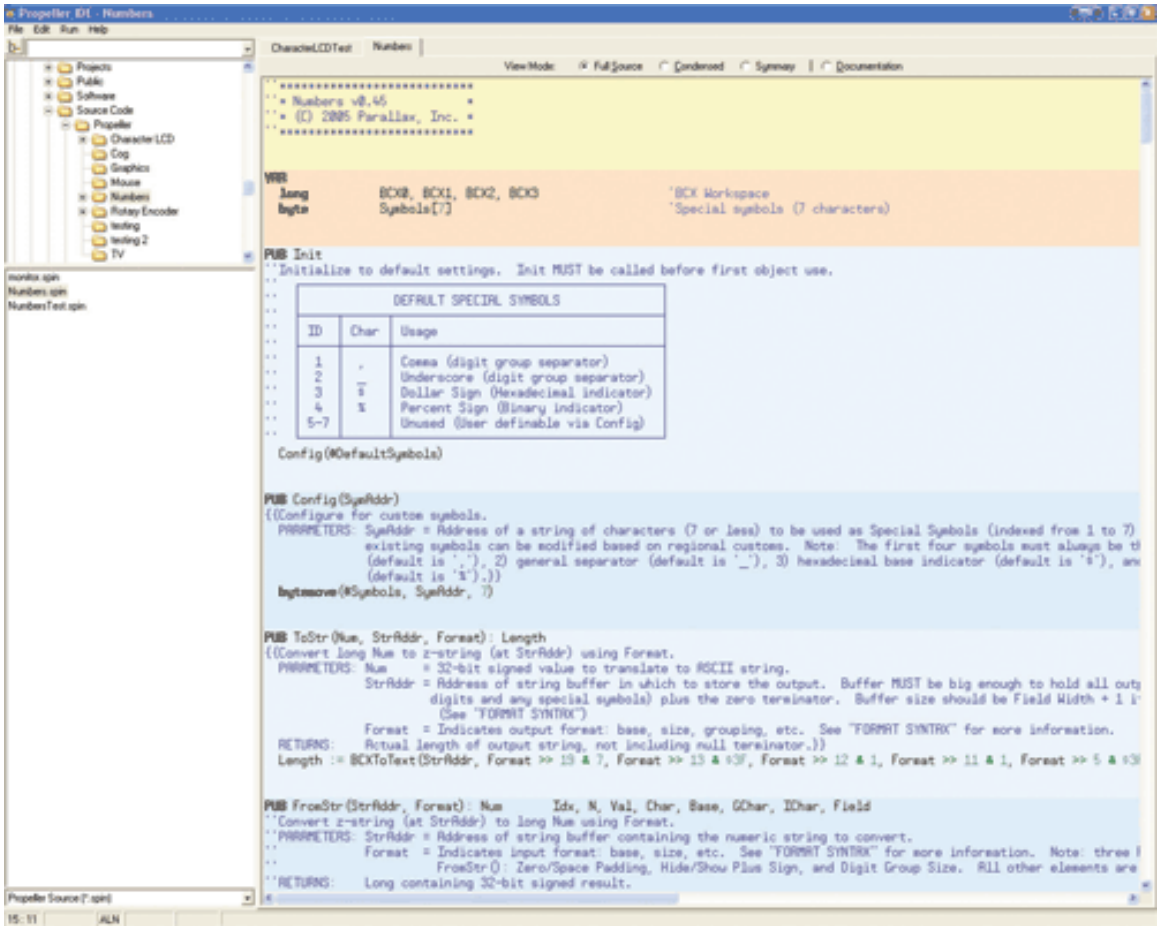


Figure 3.6.2.1

Above is a screenshot of the Propeller software. Reprinted with permission of the copyright owner, Parallax, Inc © 2008. All rights reserved.

One could think that the uniqueness of the Propeller's programming language as a down side of this chip. But at the same time Spin, the programming language, has been claimed by many to be developed to alleviate the highly known learning curve of IDEs provided with most of the development kit for microcontrollers available in today's market. The time one could take to overcome the learning curve could be very well used for implementing optimizations or even more applications.

In addition to Spin, any Propeller's development board comes with assembly language if preferred. Also, as most of the products today in the market, these electronic boards come with USB ports by default to easily download the computer program from the PC to the board. Along with the board and the interface to program the Propeller microchip, there is an extensive library of drivers for diverse hardware components that can be integrated in the robotic hand application.

All those previously mentioned hardware and software advantages in using the Propeller microcontroller are important in essence that the developing group might be more concentrated in implementing the actual functionality of the robotic hand rather than troubleshooting and using precious time in discovering how to incorporate a hardware device to the board, or even being worry about the proper usage of the IDE and its debugging tools.

3.7 Power Supply

3.7.1 Overview

One of the most important aspects of any project is how to efficiently deliver power to each of the components in the circuit; however, lowering cost is also a priority. The more power that is consumed the more expensive the project will become, and for that reason in order to accomplish the right amount of power needed for the robotic hand with the proper amount of expenses, it is key to have an understanding of the behavior of each component. However, it is important to mention that electronics equipment requires DC power for proper operation and a good power supply circuit has to maintain a constant voltage across the load with changes in load current.

The Robotic hand is controlled by a 4-Dip Switch, and a glove. All of these materials need power supply in order to work and deliver the expected results. Initially, the communication Computer-Robotic Hand is made using a device called "propeller plug"; it provides a USB-to-serial port connection that is convenient for microcontroller programming and communication. This device is capable of asynchronous communication of up to 3M baud with both 3.3 V and 5.0 V devices such as the propeller, which is the microcontroller used for this project. It important to mention that it is not require to supply voltage to the propeller plug because this device is power up by the USB port from the computer. This port supplies the voltage needed by the propeller plug to work. There are other components that also require power supply and it is worth mention them: servo motors, microcontroller, sensors, A/D Converter, Ping Sonar Sensor, EEPROM, and other electronic components.

It is worth to note that safety is an important issue in the design of any electrical equipment, which is able to come into contact with a human operator. Electrical

hazards could occur in the circuitry harming the user. It is important to prevent any electric shock which results from the passage of electric current through the human body. Fire, heat related hazards and mechanical issues can also be present. Therefore, the power supply design must consider not only operating conditions, but also likely faults, foreseeable misuse, external influences and environment, and over-voltages that might occur on input or output line.

It is essential to pick components and materials, which have prior safety certification. When using certified components the only thing it needs to be taken care of is placing correctly the components and the physical testing is done as a complete system. All components should be securely attached and not sharp edges or corners should exist. The design must also ensure that no single failure causes any of the above hazardous conditions.

Most power supplies have built-in protection circuitry to prevent over current, over voltage, or over temperature conditions. It is possible that the power supply could be operating normally but another section of the system has failed and may be loading down the power supply. The appropriate design of the power supply circuitry for the robotic hand has to be made in order to prevent all the problems just mention from happening; however, in order to have a better understanding it is important to describe the power requirements for each component used in the Robotic hand. All of them will be describe in the following sections.

3.7.2 Servomotors

The robotic hand consists of four servomotors, each of them moving a single finger independently. The servo motors used in this project need around 4-6V in order to work; this type of servomotors have three cables: one for voltage (red), one for ground (black) and one for the signal (white). In our circuit design there are four servo connectors, which are located in a Printed Circuit Board. The servo motors have their own power supply source, which is two 9 Volts batteries connected in parallel. The Servo motor chosen for this project is:

- Futaba S3003 Standard Servo

As explained before, the servos need only 4-6Volt to work, therefore, a voltage regulator also known as a DC to DC converter is needed to lower down the voltage from 9 Volts to about 5Volts. A common and inexpensive way to lower the voltage is to create a series circuit using some resistors; usually called a voltage divider but the problem is that if there is change in the load resistance the output voltage will change and the servos always need a constant output voltage supply; also a voltage divider consume battery faster.

Servomotors can produce high current draws from the battery. For this reason it is important to have a separate power supply source for them because they can create a lot of noise which can interfere with the rest of the circuitry, and also the excessive

current draw can leave the rest of the components without power. For that reason, it was decided that all the servo motors have their own battery source; and the rest of the circuitry also has its own battery source. This would help to prevent the problems mention earlier to happen. Some characteristics about the servomotors is that at not load operation the servos are not pulling the fingers, but as the motors start pulling the fingers, they will need more current in order to work and this action will change the resistance in the motors. That's why a voltage regulator is needed because it provides a constant DC output voltage and this value is maintain regardless of the changes in the load current and input voltage, as well voltage regulators reduce the amount of current that it lost in the conversion of one voltage to another. However, each voltage regulator has its limits and operating ranges so it is important to verify its specifications by checking its datasheet.

A linear regulator is a special component, which circuitry has been divided into four blocks: sense circuit, comparator circuit, reference voltage circuit, and series pass transistor. The regulated output is monitored by the sense circuit, which feeds back a portion of the output voltage to the comparator circuit. The comparator circuit compares the signal from the sense circuit to a signal from a reference voltage circuit. If there are any changes in the output voltage, the comparator circuit will send a correction signal to the series pass transistor, causing it to adjust its voltage drop to correct output voltage.

There are two types of voltage regulators: Linear Regulators and Switching Regulators. Their difference is based on the type of circuit used. Power supplies with power ratings less than 50W often use linear regulators. Power supplies with power ratings above 50W use switching regulators. The following calculation shows the total minimum and maximum power calculations consumed by the servos motors.

To ensure that the voltage supply stays within the range of 4.5 and 6 V, the voltage supply to the servo motor is going to be 5 V. All the motors are connected in parallel with respect to the source. All the fingers have the same type of servo moving them, so the total resistance will be:

$$R_{eq=}\frac{1}{1/R+1/R+1/R+1/R}=\frac{R_{servo}}{4}$$

The equivalent resistance decreases, which will allow more current to flow increasing the power demand. Therefore, it is essential to check the servo motor specification and find out the minimum and maximum current draws. With no motor movement the servo draws 9.6mA, with motor movement and no load the servo draws around 115mA. Therefore, the minimum current required by all the servos with motor movement and no load is:

$$I_{\min}=4(115mA)=460mA$$

The minimum power needed is:

$$P_{\min} = 5V(460mA) = 2.3W$$

Now, if it is assumed that when all the servomotors are functional working with a load operation (pulling the fingers to the maximum strength). By doing experiments in the lab with this servo, it was found that each servo draws at most 250mA, so the maximum power is:

$$I_{\max} = 4(250mA) = 1A$$

$$P_{\max} = 5V(1A) = 5W$$

There are a variety of linear regulators in the market which can give the right amount of voltage needed, but it is important to find one that provides 5V output at load currents up to 1A. Linear regulators are constructed using discrete components but are usually encountered in integrated circuit form. IC linear voltage regulators have built-in protection from over current conditions and overheating. Therefore, if the circuit encounter a situation when the output current reaches its maximum value, then the current limiting would kick in and would hold the output current constant. But, if the temperature increases and passes the range of operation, a built-in circuit will sense an overheating condition and would shut down the regulator until the temperature falls within the safe range of operation. These IC regulators are extremely popular for applications when power requirements are under 25W like the robotic hand application, which maximum power values for the servos were calculated above.

An example of an IC linear voltage regulator is the LM323, which gives 5V output and load current up to 3A. It is recommended to go a little bit higher than the calculated specifications in order to prevent any lack of power for the servomotors. This IC linear voltage regulator is a three terminal device; an unregulated DC (up to 20V) is applied to the input terminal, the circuit ground is connected to the ground pin and the regulated DC is available at the output pin. The LM323 regulator provides an output power of up to 15 Watts and as it was calculated before this value surpass the maximum power needed when all the motors are working at full load. Each servo motor receives 5V and enough current to perform at its best.

Although, when connecting a voltage regulator, it is important to connect a small bypass capacitor between the +Vin and the ground to eliminate the noise generated by fast-acting logic chips. Also a small capacitor has to be connected between the output of the regulator and ground to reduce load transient spikes created by fast switching digital logic or to swamp out stray load capacitance and to improve the transient response of the regulator. Usually, the datasheet of the IC linear voltage regulator tell us the recommended size of these capacitors and also the ranges of operation of the IC. As it was mention this regulator provides power supply to the servomotors, but it is important to mention that motors create excessive current spike and noise. The figure below shows how to connect an LM323 with the right

sizes capacitors recommended in the datasheet. A switch is also included to turn on/off the power distribution to the servo motor.

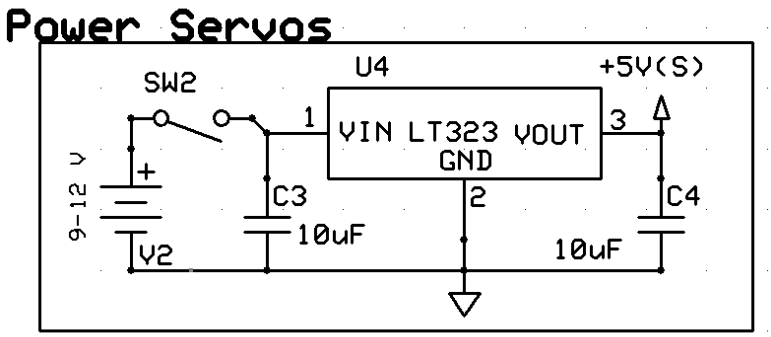


Figure 3.7.2.1: Connection of a LM323 to regulate voltage with capacitors

As general rule the input voltage has to be 2 to 3V above the output voltage, but keep in mind that the input and output difference appears as heat. This voltage difference is known as the dropout voltage. If the input voltage is unnecessarily high, the regulator will overheat; consequently, using a 9V battery is a safe power supply. It is important to quote that IC linear voltage regulators come in different type of packages for example the LM323 and other regulators come in two different packages TO-220 and TO-3. This will only be important when designing the Printed Circuit Board.

The linear regulator package, which is going to be used in this project, is the TO-220 because it is a heat sinkable device; thus it can be used in projects where a large amount of power is being drawn. The grounding area makes handling easier by reducing the possibility of damage from electrostatic discharge and the mounting with the tab ensures the component to be held in place. This device package is more than good to do the regulation of the voltage needed for the servo motors. Even though the Robotic hand will not draw too much power; this regulator package will help to prevent unexpected problems that may happen in the circuitry protecting the circuit.

3.7.3 Microcontroller and EEPROM

Another main important component that needs power supply is the microprocessor. The Luke's hand robot uses a Propeller P8X32A-D40; this microprocessor is a high-speed processing for embedded systems and has a low current consumption, which is one of the main difference compare to the other microprocessors available in the market that usually work with 5V. The Propeller P8X32A-D40 is a 40 pin package device with an operating supply voltage of 1.8 V to 3.6 V. The manufacturer

recommends supplying 3.3 V. Another important factor of using the Propeller microcontroller is that there is a lot of information available on how to interact with it as well as how to supply power to the microprocessor. The figure below shows how to communicate to the microprocessor from the computer using a USB connection and the pins number where voltage is needed to be applied.

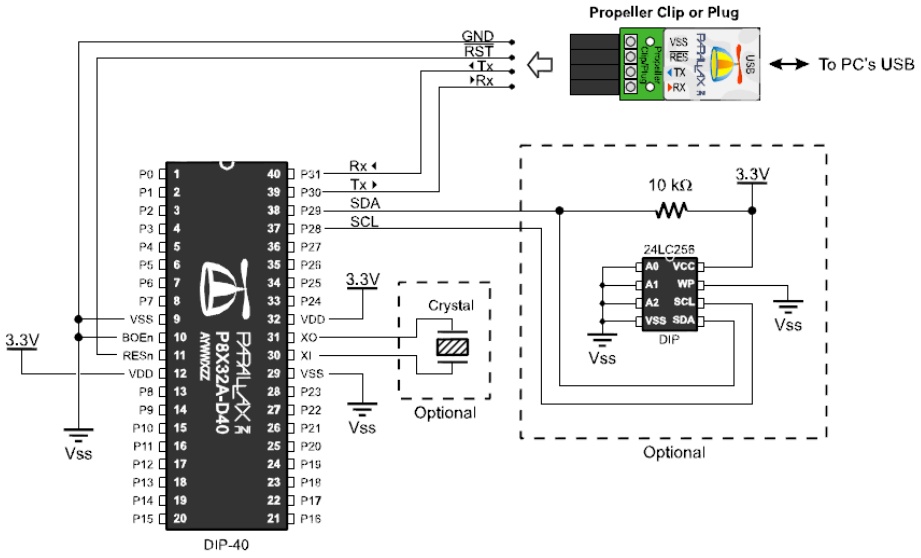


Figure 3.7.3.1: Communication to the Microprocessor
"Photo Courtesy of Parallax Inc."

For previously discussed reasons, the rest of the circuitry has another power source (9-V Battery). Two regulators are used to step down the voltage to 3.3 V. The first regulator is a LM2940, which output 5V and up to 1A of current. The output of this regulator is used to power the Ping Ultrasonic Sensor, which needs 5 V to work. The second regulator is LM2937; this IC linear voltage regulator deliver 3.3V and up to 500mA of current. This output is used to power up the propeller microcontroller.

The EEPROM (24LC256) is a 32KX8 (256Kbit) serially electrically erasable PROM capable of operating across a broad voltage range (1.8 - 5.5 V). It has been designed as a low power consumption device, which is ideal for low power applications such as personal communication and data acquisition. The maximum write current is 3mA at 5.5V. But, for this project the EEPROM is being supplied 3.3V from the same source as the microcontroller; therefore, the maximum current consumption at this voltage is around 1.98mA. The connection to the microcontroller is shown in the figure 3.7.3.1. The main function of the EEPROM is to store the final program, so any time the user wants to use the robotic hand; the microcontroller first has to pulled the program from the EEPROM and store it in its RAM memory; this is done really fast then the robotic hand can be used. The download of the program is made through the propeller plug using the USB connection from the computer port.

Input and output capacitors need to be used nearby the regulators in order to prevent the problems mention earlier. The recommended size of these capacitors can be found in the datasheet of the regulator. The TO-220 package will be also used for this linear regulator for all the same reasons mention earlier. A switch is used to turn on/off the power supply going to this portion of the circuitry.

3.7.4 Sensors and MCP3208

Another important component of the robotic hand are the sensors, these devices allow the robotics application to have a better understanding of the environment in which it is working. The robotic hand would be controlled by an adapted glove that the user would wear; this glove has four flexible resistive sensors, which controls the movement of each finger. This flex sensor produces a resistance output correlated to the bend radius; the smaller the radius the higher the resistance values. The initial resistance when it is extended is about is 9Kohms and as the flex increases, the resistance increases up to 22Kohms. The flex sensors are connected to a voltage divider circuit in order to read the changes in voltages as the resistance changes. The size of the other resistor is 100Kohms. The figure 3.7.4.1 shows the voltage divider circuit designed.

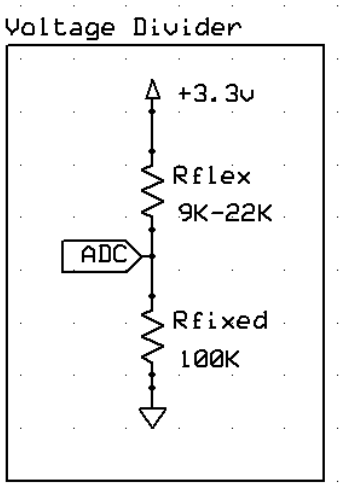


Figure3.7.4.1: Voltage Divider Recommended to be used with the Flex Sensor Resistor

There are four circuits as the one shown above for each flex sensor. A voltage of 3.3 V is connected from the voltage divider to the flex sensors, and by reading the changes in voltage as the flex sensors starts increasing their flexion at the glove, we can calculate the pulses needed to send to the servomotor of the robotics application, so they can move the fingers of the robotic hand.

$R_{flex} = 9k\Omega$

$R_{fixed} = 100k\Omega$

$V_o = 3.3V \frac{100k\Omega}{109k\Omega} = 3.03V$

$R_{flex} = 22k\Omega$

$V_f = 3.3V \frac{100k\Omega}{122k\Omega} = 2.70V$

The calculation above shows how the voltage changes as the bent increases. An A/D converter is used to take the changes in voltage and to convert it to digital values. The A/D is a MCP3208, which is a 12 bit, 8-channel, Serial Peripheral Interface Analogue to Digital Converter. The MCP3208 device operates over a broad voltage range (2.7 V-5.5 V). Low current design permits operation with typical standby and active currents of only 500nA and 320uA respectively. The MCP3208 converter employs a conventional SAR architecture. With this architecture a sample is acquired on an internal sample/hold capacitor for 1.5 clock cycles starting on the fourth rising edge of the serial clock after the start bit has been received. Following the sample time, the device uses the collected charge on the internal sample/hold capacitor to produce a serial 12-bit digital output code. Conversion rates of 100ksps are possible on the MCP3208. This device offers the choice of using the analog input channels configured as single-ended inputs or pseudo-differential pairs.

The MCP 3208 can be configured to provide four pseudo-differential input pairs or eight single-ended inputs. Configuration is done as part of the serial command before each conversion begins. Communication with the device is accomplished using a simple serial interface compatible with the SPI protocol. With most microcontroller SPI ports, it is required to send groups of eight bits. It is also required that the SPI port be configured to clock and latch data in on the rising edge. Because communication with the MCP3208 device may not need multiples of eight clocks, it will be necessary to provide more clocks than are required. This is usually done by sending “leasing zeros” before the start bit.

For this project, the MCP 3208 is connected to 3.3 V and it is programmable as single-ended inputs, so each channel can read an AC voltage. This converter is a 16-pin device and it has eight channels to perform A/D conversion, but only four are used for this project. Also, this device has Voltage reference pin, which is connected to 3.3 V. The reference input (Vref) determines the analog input voltage range. As the reference input is reduced, the LSB (low significant bit) size is reduce accordingly. The theoretical digital output code produced by the A/D converter for this project is a function of the analog input signal and the reference input, as shown below.

$2^{12} = 4096$

Digital Output Code (DOC) = $V_{in} \times 4096 / V_{ref}$

V_{in} = Analog Input Voltage

V_{ref} = Reference Voltage

$$\text{DOC} = 3.03 \times \frac{4096}{3.3} = 3761$$

$$\text{DOC} = 2.70 \times \frac{4096}{3.3} = 3351$$

With the digital code, it is easier to calculate the pulses that the microcontroller has to send to the servomotors in order to move the fingers as accurate to the glove user's movement. The 3.3 V supply voltage for these components come from the same source use for the propeller microcontroller and the EEPROM. Bypass capacitors are used in the A/D converter chip between the VCC and Ground as well as Vref and ground. They help to filter the electrical noise out of the circuits. They do this by removing the alternating current caused by ripple voltage. The sizes of the bypass capacitors are 10uF and they would help to have a cleaner signal.

A Ping Ultrasonic Sensor is located at the palm of the robotic hand, and its function is to sense if any object is getting close to the hand. This sensor provides a non-contact distance measurement from about 2 cm (0.8 in) to 3.3 m (129.92 in). The Ping sensor requires a supply voltage of 5V and supply current is 30mA typically, but the maximum can reach up to 35mA. The voltage supply for this sensor comes from the LM2940 regulator output, and as it was mention earlier this regulator can output 5 V and up to 1A of current. This sensor has three pins 5V, ground, and signal. It is worth to mention that a large resistor is important for connecting a 5 V output device to the propeller chip's 3.3 V input; 1Kohm resistor is used between the signal pin and the propeller chip's P26 I/O in order to prevent the propeller pin to get damaged. Because, the ping ultrasonic sensor is located at the palm of the robotic hand, a three pin connector is located at the schematic and PCB to connect the sensor to it as shown below in the figure 3.7.4.2

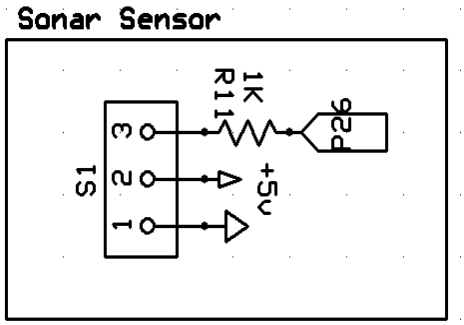


Figure3.7.4.2: Ping Ultrasonic Distance Sensor Connector

3.7.5 4-bit Dip Switch Controller

There is also a 4-bit Dip switch in the circuit designed, which is used for controlling purposes. The switch is connected to 3.3V and to four pins of the microcontroller (P8 - P11) using 10K ohms resistor between the microcontroller pins and the switch for protection purposes. The microcontroller is going to read high or low values, and based on the different combinations, the robotic hand will perform a different function. There are sixteen different combinations that can be used and for example if the combination is 0010, it means that for this application, the number 2 in decimal will be emulated by the hand. The selection of the functions implemented for each position of the switch is done by software in the propeller.

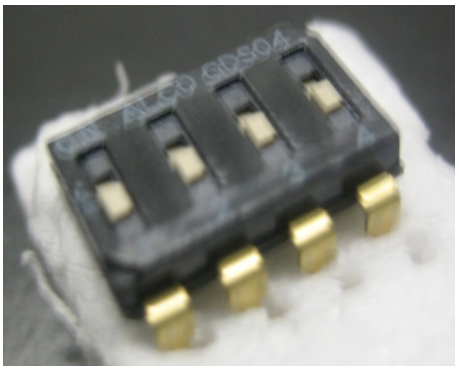


Figure 3.7.5.1: 4-bit Dip Switch

3.7.6 5MHz Crystal, Reset bottom and LEDs

Other important components that are part of the circuitry need to be discussed. These are the Crystal, Reset bottom, and LED indicators. The 5MHz crystal is important for this project because for application that are timing sensitive like serial communication, tone generation, servo control and time keeping instructions have to be done fast. The propeller chip needs to be connected to a 5MHz crystal oscillator through its X0 and X1 pins, and this crystal would make the system clock of the propeller to run at 80MHz. this can be accomplished by just implementing these three lines of code at the beginning of the program.

```
CON

_xinfreq = 5_000_000

_clkmode = xtal1 + pll16x

System Clock = (5MHz * 16) = 80MHz
```

The 5 MHz crystal chose for this project is the HC-49US Quartz Crystal because it has the following features:

- Cost effective
- Excellent aging,
- Wide frequency range,
- Low profile, excellent reliability
- “AT strip” blank technology

Next is the picture of the crystal is shown below as figure 3.7.6.1

Figure3.7.6.1: HC-49US Quartz Crystal

A reset switch is also included in the circuit design in order to restart the program if it is the user desired. This switch is connected to REsn pin of the propeller microcontroller as well as the RST pin of the propeller plug. So, every time the user pressed the Reset bottom, the REsn in will go low and the propeller will start again by loading the program saved in the EEPROM and stored it in its RAM memory. Then, the user can start using the robotic hand. A group of three LEDs are used as indicators. The voltage supply is coming from one of the I/O pins of the propeller, and then a 100ohm resistor is connected in series with the LED to get necessary current to light it up.

There are three LEDs connected as explain before. One is a Red LED, which is used to indicate when the motors are moving the fingers. A Green LED is used to indicate when the Glove is in use to control the robotic hand. A third Yellow LED is used to indicate when the robotic hand is doing any other function. The figure below shows the connection of the LEDs to the microcontroller pin.

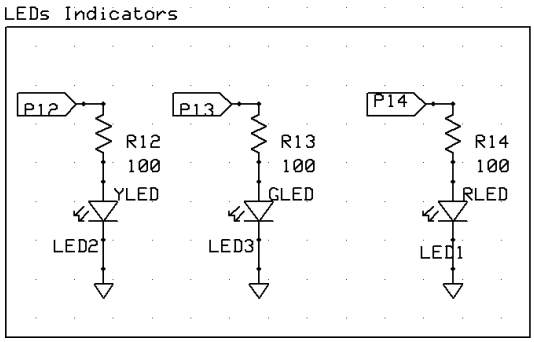


Figure3.7.6.2: LEDs Indicator Connection

3.7.7 Battery Selection

3.7.7.1 Overview

A battery is a type of linear power supply that offers benefits that traditional line-operated power supplies lack: mobility, portability, and reliability power supply. But, when choosing a battery it is important to consider the battery's internal resistance. Remember that the cheapest batteries have very high internal resistance, which will shorten their lives in the robot application and make it difficult for the robot to work continuously. The battery's internal resistance creates heat which is power lost in its most basic form, which reduces the amount of power available from the battery, but it also could harm anyone close to the battery. In a robot application, which draws considerable amount of current, the internal resistances of the batteries have a voltage drop across them which become larger as more current is drawn from the batteries. This voltage drop is actually power loss for the robot.

Another important factor is the capacity of the battery, which is rated in amp-hour current. This is the amount of power that a battery can deliver over a specified period of time. The current ratings are as important as the voltage ratings. If the battery does not provide enough current, it will not be able to turn on the motor or sufficiently power all the electronic devices in the robot hand. It is important to note that the current rating of batteries, or any voltage source, is only a guide to typical values permissible for normal service life. The actual amount of current produced when the battery is connected to a load resistance is equal to I=V/R, by Ohm's Law.

The robotic hand is going to have several motors and it is important to quote that motors can draw excessive current when first switched on, then they settle down to a more reasonable level. Therefore, it is recommended to use a battery which offers 20 to 40 percent more than the power required for the robotic hand because the battery has to be able to accommodate the surge. The Luke's robotic hand is going to have motors, sensors, and other different electronics components, so there is going to be a considerably amount of current flowing, which will increase the power consumption. Therefore, the chosen battery needs to have low internal resistance in order to help deliver the required current to every component in the circuitry. There is a variety of internal resistance batteries in the market and it is important to talk about them and know their respective advantages and disadvantages. They all will be discussed briefly in the followings paragraphs.

3.7.7.2 Types of Batteries

There are several types of batteries that can be used in this project. Those batteries come in different sizes, shapes, and configuration.

Zinc

Zinc batteries are a type of battery, which are very commonly used due to its low cost. They come in two forms: carbon zinc and zinc chloride. Carbon zinc or regular duty batteries die out the quickest and are not recommended to be used in robotic applications. Zinc chloride or heavy duty batteries provide a little more power than regular carbon zinc cells. Zinc chloride or heavy duty batteries last 25 to 50 percent longer than the regular ones. Regardless of the added energy, they are not suitable for most robotic applications.

Alkaline

Alkaline batteries last up to 800 percent longer than carbon zinc batteries. But the actual increase in life expectancy ranges from about 300 to 800 percent depending on the application. The outstanding performance of the alkaline battery is due to its low internal resistance and it will perform satisfactory at low temperature. However, the increase in power and life expectancy is worth in cost, they cost twice as much as zinc batteries. In robotics, where the batteries are driving motors, solenoid, and electronics, the average increase is a reasonable 500 percent or five times the life of carbon zinc batteries.

In recent years, high-tech alkaline batteries can be found, and the new type of batteries offers two or three times the life of regular alkaline batteries. The dramatic improvement in battery life is by decreasing even more the internal resistance of the batteries and improving their ability to respond to changing current demands such as motors turning on and off without damaging the internal structure of the battery. They are more expensive than regular alkaline, but they can do a better job when working in a robot application.

Nickel Metal Hydride

These batteries are probably the optimal rechargeable battery technology for use in robots. They can be recharged 4000 or more times and have a low internal resistance, so they can deliver large amount of current. Nickel metal hydride batteries are about the same size and weight as alkaline cells, but they deliver about 50 percent more operating current. Such batteries work best when they are used in very high current situations. It is recommended when the battery is discharging to be located away from the circuitry or any other component like the microcontroller that may be affected by the heat. One disadvantage of these batteries is that they do not hold the charge well, the charge in the battery is depleted over time even when the battery is in storage. For that reason, it is a good idea to put the batteries on a recharger at regular intervals even when they have not been used.

Nickel-Cadmium

These types of batteries are popular because of the ability to deliver high current and to be cycled many times for recharging. Also the cell can be stored for a long time, even when discharged, without any damaged. But in recent years, nickel-cadmium batteries have fallen out of favor due to the memory effect in which the battery will tend to last only as long as it is usually asked to. If it is placed in longer service periodically, it will not respond very well. But, it is important to be careful when using these batteries because they tend to change output polarity, positive becomes negative and vice versa. Polarity reversal is common if the battery is left discharged for too long or if it is discharged below 75 or 80 percent capacity.

Lithium and Lithium Ion

These types of batteries are popular in laptops computers; they are best used at a steady discharge rate and tend to be expensive. However, they offer high output voltage, long shelf life, low weight and small volume. All these features make them an excellent choice for special applications. A lithium battery can provide at least 10 times more energy than equivalent carbon-zinc cell. These batteries provide the highest energy density of most any other commercially available battery and retain their charge for months, even years.

Lead-Acid

The battery that comes in vehicles is a lead-acid battery. It is made up of not much more than lead plates crammed in a container that is filled with an acid-based electrolyte. No all lead-acid batteries are as big as the one in a car. It can also be found in 6V batteries and they are about the size of a small radio. Although, lead acid batteries are powerful, they are heavy which could be one disadvantage for using it in robotic applications. These batteries are used where high values of load current are necessary. Lead-acid batteries come in self-contained packs, six volt packs are the most common, but 12 and 24 packs can be found. These batteries come in various amp-hour capacities, so they can work in different applications. The lead-acid type is a secondary cell or storage cell, which can be recharged. The charge and discharge cycle can be repeated many times to restore the output voltage, as long as the cells are in good physical conditions.

Final Selection:

After reviewing the different types of batteries on the market; it was decided that for the Robotic hand 9 V alkaline batteries are going to be used because they have low internal resistance and their voltage is high enough for the regulators to be able to lower it to the required one by the components. Also, they are cheap and easy to find in any convenience store.

3.8 Software

3.8.1 Introduction

The software part of this project is divided in two parts, the programming of the behaviors for the robotic hand, and the required coding for the multiple interfaces planned. The latter will be discussed in the following section, titled *User Interfaces*.

Regardless of the microcontroller selected, which was eventually the deciding factor in language selection, there are two options for programming a microcontroller. One is using a supported language by the manufacturer of the MCU, which would be a popular language such as C, C++, or Java, or a proprietary one such as Spin (by Parallax). The second is using assembly code, which is unique to the type of MCU, and is dependant of the Instruction Set Architecture (ISA). These two options can be combined seamlessly most of the time, since the Integrated Development Environment (IDE) is able to differentiate between the two at compile time.

There are certain aspects of the design process that are language independent, such as; data structures needed, algorithm analysis, running time, and modularity. These aspects will be used to illustrate the process that will be needed to achieve a reliable piece of software that meet all pre-established requirements.

3.8.2 Languages

Language selection is strictly dependant in the MCU selected, and as it has been previously mentioned, the Propeller (Parallax) has been selected as the MCU for the project, which implies that both SPIN and Propeller Assembly can be used. C could also be used, but a proprietary IDE is needed, which represents a considerable increase in the cost of the project. The Instruction Set Architecture (ISA) for the assembly code provides 120 instructions, providing a wide array of possibilities for the programmers in case they need to use it in conjunction with SPIN to better manage memory allocation.

3.8.3 Data Structures

Due to the nature of the data to be manipulated, complex data structures will not be required. The microcontroller stores at all times somewhere between ten (10) and twenty (20) variables. Some of these variables store inputs (locomotion and control indicators) and some are used to store outputs (sensors).

Inputs, the type of data used to store the voltages that need to be passed to the servomotors are defined as *Doubles*, while the most adequate type for the control

indicators (LEDs) is a *Boolean*, since the LED is either *on* or *off*. All LEDs are stored in an array to facilitate the handling of these values. The voltages passed to each servomotor will be stored in the same fashion, using an array of Doubles. If more than one *degree of freedom* is used per finger/thumb, a 2D array will be used, for easy access to the relevant data.

The data generated by the sensors will be treated in a different manner, since each finger will possess a different kind of sensor, and using indexes in an array to identify its purposes it is inefficient since it can lead to misunderstandings, so descriptive names will be assigned to each variable storing the data received from the sensors.

3.9 User Interfaces

3.9.1 Overview

The robotic hand is being conceived as a tool intended for use in different fields, and not restrained to single task, thus it needs to provide the final user with an interface that is appropriate and functional for the intended task. These interfaces should present a wide array of functions, in and easy and intuitive way, so that the user does not feel overwhelmed by the product, and finds it useful and adequate for his/her need(s). Hence, four (4) possible interfaces have been taken into consideration: Computer Driven, Portable Device (iPhone, G1), Glove Interface, and a PodControl.

3.9.2 Computer Interface

This interface is intended for testing and development, and as such will offer the most flexibility to the user when interacting with the hand, it will not only allow the user to test every single feature, but also to see how each independent part (motor, sensor, etc) is behaving.

Different programming languages were taken into consideration at this stage (C, C++, C#, Objective-C), but Java was finally selected as the intended language for this interface for multiple reasons. First, Java provides an extensive object oriented framework, with a vast library of classes that will simplify the design and deployment process, while allowing the development of a robust application. Java has also been adopted for its portability characteristics that will allow the control interface to be *system independent,* meaning it will run on *almost* any *operative system* (OS) with little or no changes at all, and will only require one package to be compiled and built. Other important factor taken into consideration was the fact that the developers were highly familiar with the Java IDE when it came to object oriented programming (OOP), and given the time constraints, learning a new language for the purpose of this project will be highly inefficient.

The data received from the Robotic Hand will be handle based on the nature of it, and will be allocated using different data structures (which will be discussed later on, on the design stage). All data structures to be used are already supported and included into the Java IDE, considerably reducing the time required to implement the code needed to properly manipulate the data. Figure 3.5.2 illustrates some of the desired functionality of this interface. The left block in the image allows the user to identify the object, and enter certain characteristics of it such as width and weight such that the hand knows beforehand which kind of object is being picked. The center block allows the user to activate/deactivate sensors from the array for testing purposes, or for better functionality depending on the object being grabbed. The right block lets the user close the hand at a manual rate by using sliders, allowing the user to close/open the fingers (as a whole) and the thumb at different rates.

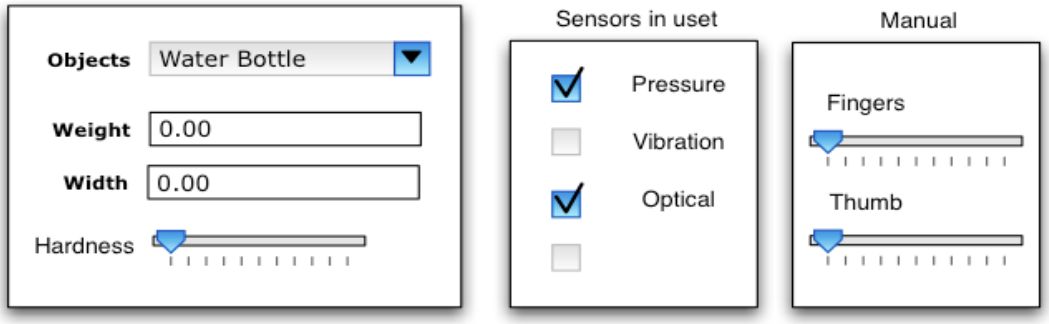


Figure 3.9.2 Sketch of the GUI

This interface was not finally implemented partly because of the difficulties of maintaining constant communication between the MCU and the Computer, and because Parallax already provides an application that is excellent for debugging purposes, since it provides a GUI that can output any desired values to the screen.

3.9.3 Portable Device Interface

The use of portable device such as a cell phone or PDA as a controller for the hand would provide the user with a limited array of functions, and would be mainly focused on portability. It will certainly enhance the use of the hand by providing a mobile and easy to use platform to control the hand, but was not a viable option for the project in its current form. Even thought this interface would have added many features, the increase in cost it implies would have overcome the benefits. The cost of acquiring such a device (iPhone) would have been around $300, which is almost 30% of the final budget. It would have also added complexity to the circuitry since either a Wi-Fi or Bluetooth would have

been required, adding $20-30 to the cost. Aside from the cost increase, time was also a big constraint, since this device is programmed using Objective-C, which none of the developers is familiar with, the needed time to implement this interface would have certainly affected the other areas of the project, hence it was disregarded as of now.

3.9.4 PodControl (Foot control)

This is probably the most useful interface in terms of functionality to the user, since it can be used in prosthetics. Unfortunately, it is also the most limited in terms of functionality, since the foot can only apply pressure to the *insole* at certain points. *The foot can only apply identifiable forces at three (3) different points, the thumb, heal, and the metatarsals,* and as such, the sensors will be located at those points (Figure 3.5.4). Due to its limitations, the only functions that could have been implemented would have been grabbing and releasing objects and turning on/off the control. The importance of the latter one lays in the fact that power consumption in this interface was high in the list of requirements since it would have been used during extended periods of time, and the sensors, and most importantly the wireless module, will be draining current permanently and having to change the battery every so often will make the product unviable.

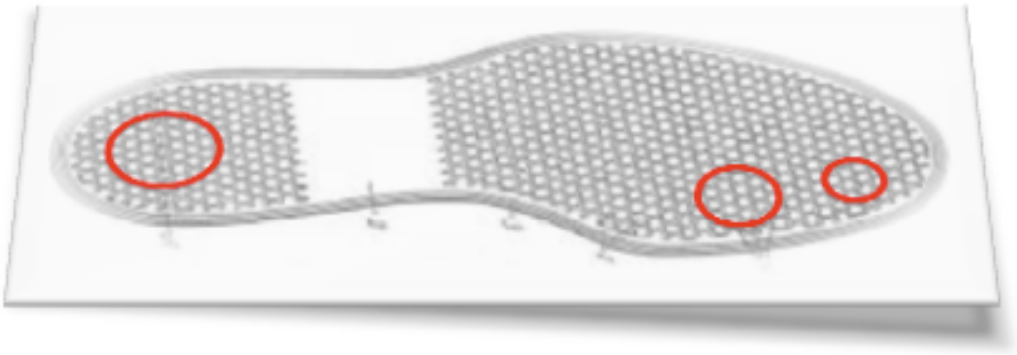


Figure 3.9.4 Insole with tentative sensor locations

Size needed to be taken into the equation when selecting the elements of this platform since it needs to fit in the shoe while being comfortable to use. Hence small factor force sensors would have been used, as well as button cell batteries (discussed on the power section). Other types of sensors were considered for this platform, such as an accelerometer, which would allow the MCU to

determine if the person using the hand is currently is motion, or in a fixed position. This particular interface was not implemented because of the fact that wireless connectivity was omitted in the final design due to its cost and the difficulty of its implementation, thus rendering the PodControl unviable, since wiring it to the hand was not adequate.

3.9.5 Glove Interface

As it can be inferred from its descriptive name, this interface allows the user to freely control the robotic hand through a glove featuring an array of sensors that will read the movements of the user's hand and closely imitate them on the mechanic hand. This kind of interface can be useful in a wide variety of fields, particularly in those were certain objects or task cannot be handle properly due to the nature of the human hand, which is sensitive to extreme temperatures, corrosive compounds, and similar hazards. Unfortunately, to be used under such conditions, a great amount of time has to be allotted for research, development and testing, and considerable resources are needed. So for practical purposes, and for the scope of the project, this interface will be focused on the "*toy industry"*, which has lower requirements and regulations.

The glove features sensors in all four fingers to detect all movements to an accuracy that will be determined in the design process where a final sensor is selected and tested. For this purpose, force or pressure sensors are the best suited to perform the task efficiently, and out of all the sensors previously considered, a potentiometer seems to be best option.

Tentatively, the glove was going to have feedback mechanisms, that would transmit relevant information to the user, such as; if the object has been grabbed, force being applied to the object, if the object is slipping, and which fingers are actually in contact with the object. This was going to be implemented trough the use of color-coded LEDs. Due to the difficulties placing force sensors on the robot hand, some of the feedback mechanisms were not included, but an LED was added to PCB board that indicates when the Glove is in use.

The glove was originally planned to communicate with the hand wirelessly thought its own ZigBee module, but it was finally wired directly to the MCU. It is battery powered, and controlled by a Propeller MCU. Some functionality was either added or removed from the glove depending on the actual features implemented in the robotic hand, and the design and implementation difficulties that were found in the building and testing stage.

3.10 Transmission and Reception

3.10.1 Introduction

This project will require the wireless transmission of data between the robotic hand and two control interfaces, the PodControl and the Glove interface. No communication is needed between the controls, but it is essential that the robotic hand can communicate with any other device in the network. Due to nature of the wireless personal area network (WPAN), the ***star*** *(Figure 3.10.1)* topology is the most adequate for this specific application. In this topology the hand will be the central node while the user interfaces will be peripheral nodes.

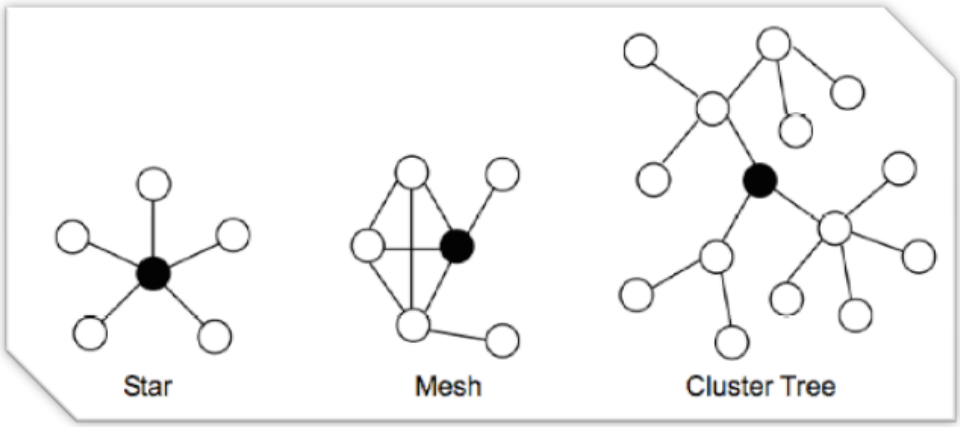


Figure 3.10.1 Physical Network Topologies Supported by ZigBee

3.10.2 ZigBee Protocol

The ZigBee protocol specifies the Physical (PHY) and the Media Access Control (MAC) layers.

3.10.2.1 PHY Layer

"The PHY layer defines the physical and electrical characteristics of the network. The basic task of the PHY layer is data transmission and reception. At the physical/electrical level, this involves modulation and spreading techniques that map bits of information in such a way as to allow them to travel through the air. Specifications for receiver sensitivity and transmit output power are in the PHY layer." [Rabbit®, 1]. This layer is also responsible for the link quality indicator (LQI) of the received packets, energy detection (ED) within the current channel, enabling/disabling the radio transceiver, and clearing the channel assessment (CCA).

3.10.2.2 MAC Layer

"The MAC layer defines how multiple 802.15.4 radios operating in the same area will share the airwaves. This includes coordinating transceiver access to the shared radio link and the scheduling and routing of data frames." [Rabbit®, 2]. This layer is

also responsible for the following tasks:

- Beacon generation if device is a coordinator
- Implementing carrier sense multiple access with collision avoidance (CSMA-CA)
- Handling guaranteed time slots (GTS) mechanism
- Data transfers services for upper layers

3.10.3 Properties of 802.15.4

The ZigBee protocol features different properties depending of the frequency of operation. 802.15.4 defines operation in three license-free industry scientific medical (ISM) frequency bands. A parallelism between the 915Mhz and 2.4Ghz bands is made in the table 3.10.3 below.

Property Description	Prescribed Values	
	915 MHz	2.4 GHz
Raw data bit rate	40 kbps	250 kbps
Transmitter output power	1 mW = 0 dBm	
Receiver sensitivity (<1% packet error rate)	-92 dBm	-85 dBm
Transmission range	Indoors: up to 30 m; Outdoors: up to 100 m	
Latency	15 ms	
Channels	10 channels	16 channels
Channel numbering	1 to 10	11 to 26
Channel access	CSMA-CA and slotted CSMA-CA	
Modulation scheme	BPSK	O-QPSK

Table 3.10.3 Comparison between 802.15.4 Frequency Bands

3.10.3 Transmitter and Receiver

"The higher the transmitter's output power, the longer the range of its signal. On the other side, the receiver's sensitivity determines the minimum power needed for the radio to reliably receive the signal. These values are described using dBm (decibels below 1 milliwatt), a relative measurement that compares two signals with 1 milliwatt used as the reference signal. A large negative dBm number means higher receiver sensitivity" [Rabbit®, 4]

3.10.5 Channels

The standard specifies that only the three ISM bands can be use for transmission. The 2.4Ghz band operates worldwide, while the 868Mhz only operates in the EU,

and the 915Mhz only operates in North and South America. "If global interoperability is not a requirement, the relative emptiness of the 915Mhz in non-European countries might be an advantage for some application" [Rabbit®, 4]. For applications that require the full 250kbps data rate, the 2.4Ghz is the only option, and as specified by 802.15.4, "communications should occur in 5Mhz channels ranging from 2.405 to 2.480Ghz" [Rabbit®, 5].

3.10.6 Network Devices and their Operating Modes

Two types of devices can participate in a Low-Rate WPAN (LW-WPAN): a full function device (FFD) and reduced function device (RFD). "An RFD does not have routing capabilities and can be configured as end nodes only" [Rabbit ®, 6]. They communicate with their parent, which is the node that allowed the RFD to join the network. "An FFD has routing capabilities and can be configured and the PAN coordinator" [Rabbit®, 7]. In a **star** network all nodes communicate with the PAN coordinator only so it does not matter if they are FFDs or RFDs.

Three operating modes are supported by IEE 802.15.4: PAN coordinator, coordinator (router), and end device. FFDs can be configured as any of the three modes, while RFDs can only

3.10.7 Addressing Modes

802.15.4 supports two different addressing methods: short (16-bit) and extended (64-bit) addressing. An extended address (also called EUI-64) is assigned to every RF module that complies with the 802.15.4 specification. When a device is associated with a WPAN it can receive a 16-bit address from its parent node that is unique in that network.

3.10.7.1 PAN ID

Each WPAN has a 16-bit number that is used as a network identifier. It is called the PAN ID. The PAN coordinator assigns the PAN ID when it creates the network. A device can try and join any network or it can limit itself to a network with a particular PAN ID. ZigBee PRO defines an extended PAN ID. It is a 64-bit number that is used as a network identifier in place of its 16-bit predecessor.

CHAPTER 4 Design

4.1 Robotic Hand

The robotic hand is intended to be capable of having independent movements on each finger as well as be able to execute actions that require the coordination of the whole hand in a human-like fashion. The group is fully aware of the complexity of the task in the sense that any robotics project requires an important amount of mechanical knowledge or money to invest in the skeleton design. The monetary resources are limited and coming directly from members expenses. Therefore, from the best of the group members' knowledge the hand's skeleton design would be made as inexpensive as possible but at the same time carefully maintaining the required functionality to succeed in the approach.

4.1.1 Different Approaches of Designing the Robotic Hand

First design on paper to be constructed had multiple pieces united by a cross over metal pin with some threaded mechanism to keep the parts together. It was discarded because after a thoughtful analysis of the idea, it was found that the robotic hand would have so many small pieces that will make the process difficult to execute and guaranty a success given the resources available and background of the group members. The task of coordinate all the small parts is a time consuming and meticulous process, but since the project is not a mechanical engineering venture, the group decided to make some changes to simplify the skeleton of the human like hand design.

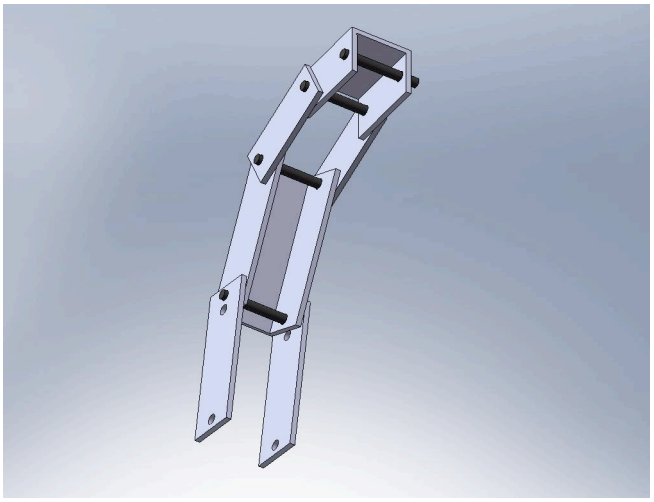


Figure 4.1.1.1: Shows the first design of the finger using several independent pieces.

Evolving from the first prototype to the second one designed on paper, which was the first prototype to be actually built, a single finger was put together and the mechanical part was tested. Then, with the first finger of the project being built, it was presented to Dr. Richie in order to gain feedback and his recommendation was to use two C-channels instead of one C-channel and two flat pieces to build the fingers of the robotic hand. The advice was followed to reduce the overall number of pieces. By having less number of parts at the time of assembling the robotic hand, the degrees of freedom as well as the number of variables were also reduced and therefore the mechanical design was simplified.

The following figure is presented to illustrate the design of a single finger using only one C-channel and flat individual pieces on the sides to complement the union of two phalanges. Within the figure 4.1.1.2 is also identified the mechanism to put together the phalanges. That mechanism is a long threaded rod, which goes from one side to the other, and is locked by a nylon washer and two-nut system.

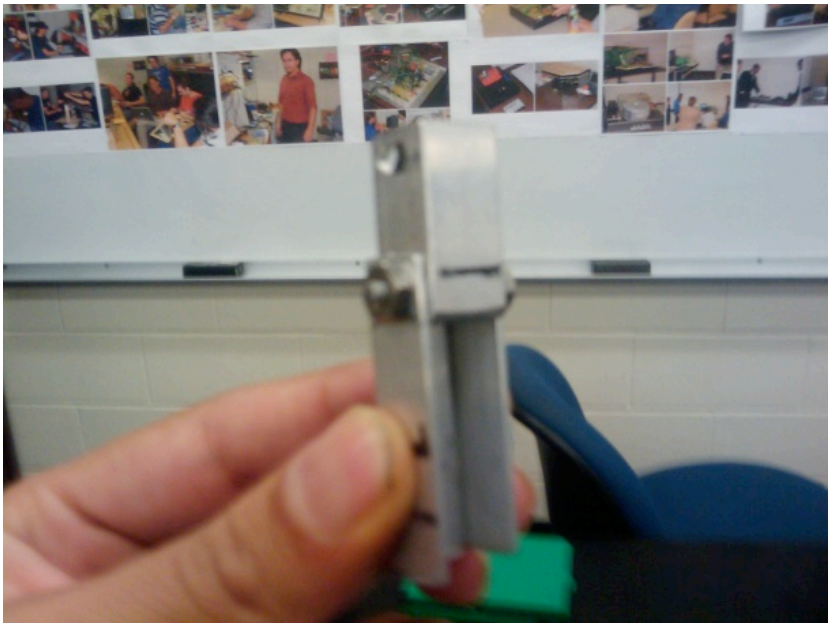


Figure 4.1.1.2: Finger with a single C-channel and two flat individual pieces complementing the other phalange.

After learning from previous prototypes, another trial product was sketch and this time all the fingers were built using two C-channels. One of the C-channel had to be wider than the other to let the inner C-channel fit inside the other as well as provide some room to accommodate a nylon washer in between the metal parts in order to reduce undesired friction. The dimensions for the inner C-channel were 3/8 wide by 3/8 deep inches and the second one was 1/2 wide by 3/8 deep inches.

Below is present the figure 4.1.1.3 with the second built prototype two illustrate the implementation of using two C-channels jointed by a metal rod and locked by nylon washers and nuts on the sides.

Figure 4.1.1.3: Implementation of fingers using two C-channels

Following the pattern of a human hand, where every finger is unique in size due to the irregular dimensions of its phalanges, the group decided to give the same trend to this sample prototype. Thus, for each phalange, it was given a different measure of length depending on each of the fingers. The dimensions of the phalanges for this prototype can be found in the Table 4.1.1.1, which is presented below. It is also important to mention that in addition to the naturally required length of each phalange, some length was added to the phalange to support the pivot point (joint) in each finger.

Phalange	Large (cm)	Width (cm)	Angle Deviation
Tip	3.2±.1	1.2±.1	45
Middle	4.5±.1	1.5±.1	-
Lower	5.9±.1	1.2±.1	63.43-50
Piece in Palm	3.7±.1	1.5±.1	-

Table 4.1.1.1

As mentioned before with the respective illustration, commercially available threaded rod (6-32 by 36) was used to get through the union of the C-channels

implementing the joint of two the phalanges. Each 'joint' has a length of 230mm, regardless of which joint or finger since all fingers were given a standard width for this application designed. It is intended that each rod will be holding two phalanges in place by using one (1) hexagonal metal nut at each side of each phalange. The metal hexagonal nuts would have the appropriate dimensions to match the rod's thread. Other important suggestion made by Dr. Richie during one of the group meetings was to place between the two C-channels a flat nylon washer to take care of the friction caused by the metal pieces.

Once modifications were made, a new single finger was assembled to do the necessary testing. The results of testing the free mobility of a single finger helped to determine that putting the two nylon washers in between the two aluminum C-channels, the joint get too tight that it limited a natural movement because the space was not enough to fit them. So instead of improving the design by avoiding metal friction, the washers augmented the friction at the time when they were forced to fit in the assigned space. Unfortunately, there are not so many possibilities to replace one of the C-channels by other of couple millimeters wider or narrower to make the phalanges fit perfectly with the required nylon washers in between.

Accommodating the sample prototype to the above described limitations, new testing was made by placing only one nylon washer in one of the sides of each phalange with better results than expected, but knowing that it was not the final solution. A final test was made without any nylon washer and the results were as good as the ones with only one washer in one of the phalange's side. The reason for the unexpected result have foundations on the fact that the space between two C-channels is about 2 millimeters, which is about the width of a single nylon washer that was available in the store where the materials were purchased. Therefore, the friction of metal against metal is transferred to the friction of pieces without space regardless of the material. A temporary decision of building the four fingers with three phalanges with the material on hand was taken; then, fingers were assembled without nylon washers for now while thinner nylon washers or any other suitable material can be find for the final design and implementation.

Considering the limitations of implementing most of the natural movements of a human hand that is tremendously complex due to its nature, the angle as well as the order at which each phalange of the robotic hand is going to be flexing was limited because in the design of this project, where a single servomotor would control a whole finger. This implies that a finger of the robotic hand could be completely close beginning from the tip of the finger and following the successive phalanges before even grabbing an object because of the order at which the phalanges would be closing. Of course that is not the natural behavior of a human hand, but it is what is accomplished based on the method implemented in this project.

To avoid the described situation and taking in considerations the limitations formerly sited, the angles of inclination between phalanges were carefully designed. It is how the first phalange corresponding to the fingertip has about 55 degrees allowing most objects to get in between the fingers and palm of the hand to achieve

grabbing functionality; whereas the other two phalanges have about 90 degrees of inclination with respect to their base to finish the encapsulation of an object in addition to performing the rest of the traction necessary for hold something. The length of each phalange was taken from measuring the hand of one of the members of the group plus one quarter of an inch to allow some room for overlapping the C-channel pieces where the joins are thought to be.

The next figure shows the angles of inclination of the fingertips as well as the bending degree of the other phalanges.

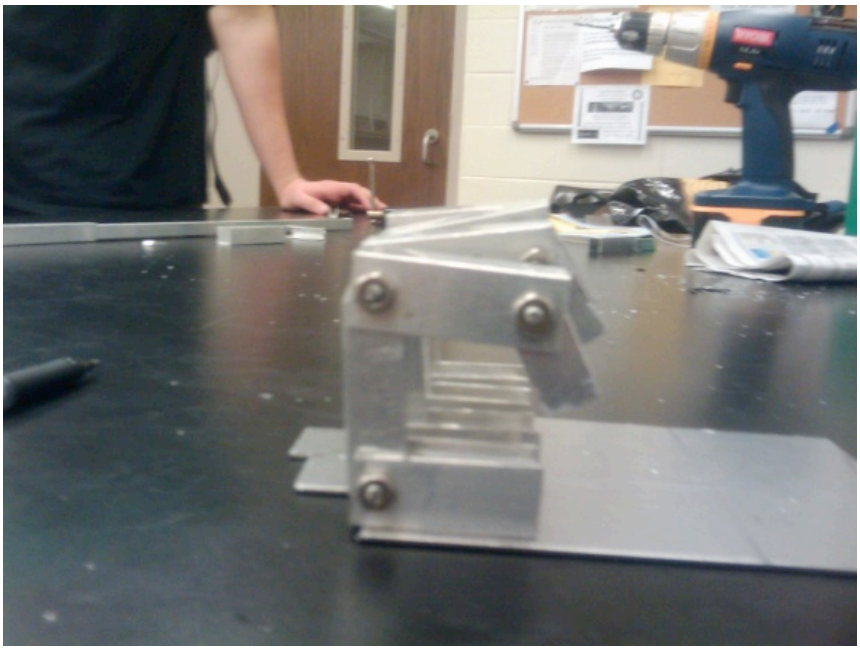


Figure 4.1.1.4: Showing the angle of inclination for 4 fingers

In the tip of each finger a rod piece was added with the corresponding hexagonal metal nuts on the sides and in the future the appropriate nylon washer. The purpose of that accessory is intended simulate the union of a tendon and the bone, which in the case of the robotic hand it ties a nylon string to the rod, which is the actuator mechanism to make the phalanges close the angle of bending in order to react at the given commands. Although in a human hand, there are several tendons controlling each one of the phalanges and therefore, more degrees of freedom for controlling each finger, the design of the project uses only a single spot for pulling the whole finger. The 'fake' joint then connects the string to the servomotor, which does the pulling of the string to generate the dynamic motion of the finger like the tendons and muscles relationship in a human hand. The next figure helps in visualizing a servomotor pulling a string to make a finger close when necessary.

Figure 4.1.1.5: Finger with a rod and string attached to it implementing the tendon muscle mechanism.

A gap was also necessary to be taken in consideration in the designed to make available the seamlessly pass of the nylon string when the fingers were bent in order to avoid pinching the tendon-muscle like mechanism. The strings passing through the insides of the hand's skeleton is visualized with the former and next figure.

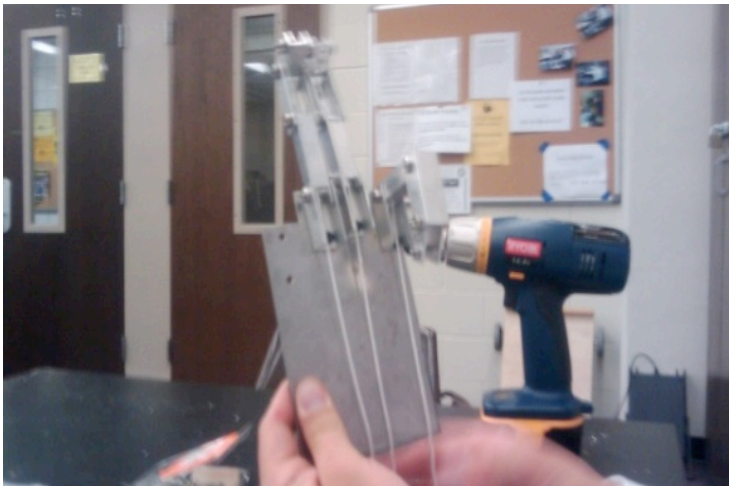


Figure 4.1.1.6: Shows the gaps made to let the strings pass through the inside of the fingers.

Other fundamental part of the design was to make the phalanges stop moving when the finger was totally stretched; in other words no significant negative angles are

allowed because it increases the required amount of work to perform any movement. There were several ideas implemented by adding extra pieces to each one of the finger's joints in the design. As mention before, adding more pieces was undesired because it gives more complexity to the design of the robotic hand. Finally, one of the members of the group came out with a simple but powerful solution. The power of such solution is appreciated in the sense that with little modification of the original design of the phalanges, the 'stopper' mechanism was incorporated without augmenting the number of pieces for a complete finger.

Thus, if one servomotor is used to control the motion of a complete finger, an 'stopper' will be needed at each joint to limit the minimum angle of inclination (zero degree) for each phalange, which due to the design of the finger, will create a sequential movement of all phalanges, from top to bottom cascade like movement. This type of design let the tip of each finger to have the capacity to apply the maximum amount of pressure at the time of holding an object. The figure below illustrates the solution implemented to avoid a phalange going into negative degrees of rotation at the time of starching the fingers.

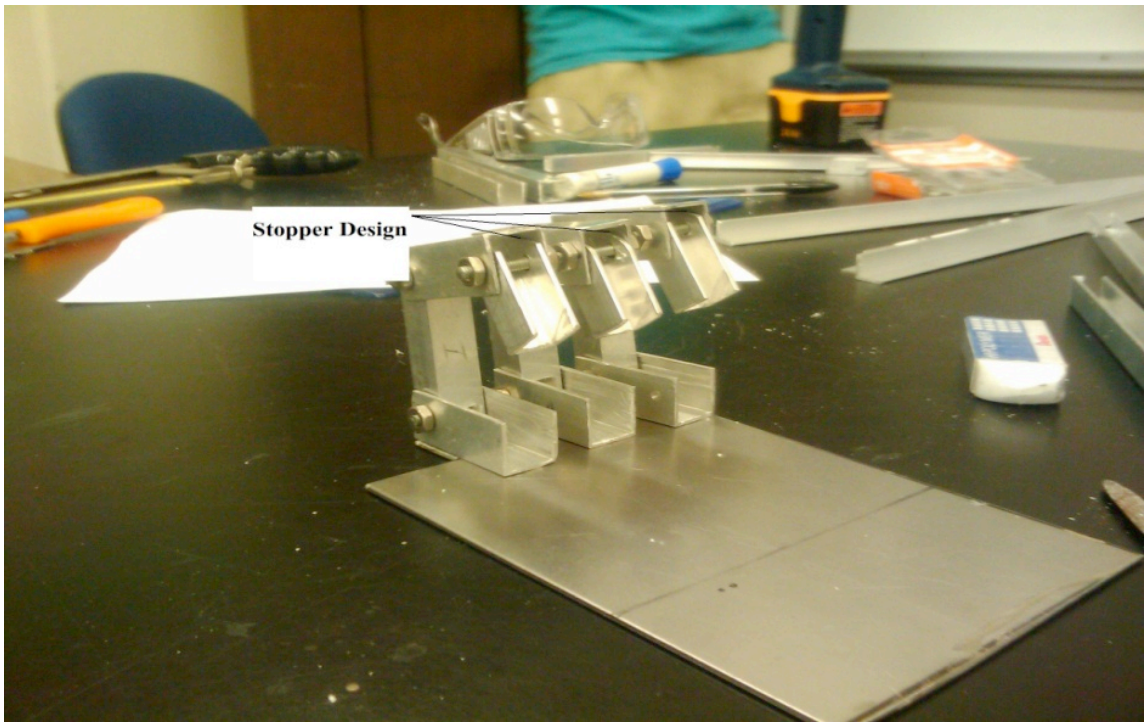


Figure 4.1.1.7: Shows the stopper design implementation limiting the angle of inclination

Several types of sensors are needed to control the strength of the robotic hand required to grab objects of different kinds and shapes. Moreover, actuators are important to command the robotic hand performing certain actions at the time of sensing an object whether by approximation of the object to the application

designed or touching pre-determined parts of the hand. The sensing process of this application is discussed later in a different section of this chapter.

For budget reasons, the execution of building the fingers was completely executed by trail and error. As a result, those pieces constructed have consistent alignment issues that could be solve by using an auto CAD type computer aid program to design where exactly each cut must go and where every hole should be open to accomplish an acceptable or desired degree of alignment. The computer-aided design also should be complemented by using appropriated tools to cut the pieces, and open the holes in the exact place or by sending the pieces to a specialized shop to be set ready for the assembly process. Figure 4.1.1.8 shows the alignment issues present in the application designed.

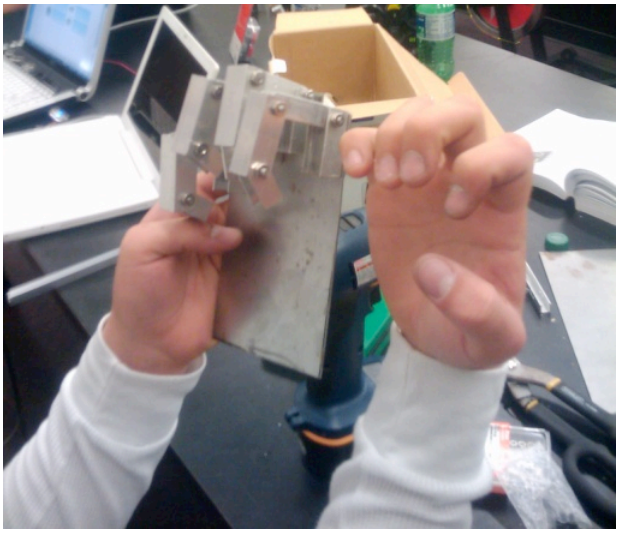


Figure 4.1.1.8: Shows the problem with alignment in the fingers

Figure above also illustrates how the back of the palm of the hand is implemented to hold the fingers in place.

The actual palm have not been physically implemented yet, but there are several theoretical ideas from where some of them are going to be build in order to test the performance of the built ones and decide which is more suitable for the application. The decision making factor would be based on how the design can be incorporated with the integration of necessary sensing functionality. The prototype idea for the palm of the robotic hand is center on a metal plate capable of folding at two thirds of the height from the base of the actual palm to the base of the fingers.

Back of the hand is implemented as a fixed metal plate to support the whole hand as well as holding the fourth phalange of each finger. That fourth phalange was screwed against the inner section of the back of the palm so that, the joint of the third phalange and the fourth one is going to emulate the knuckle where the finger

joins the hand. Since those fourth phalanges and the back of the robotic hand are static pieces, there is not much variation that can be made on their design having in mind the concept of optimizing the sample prototype. The above Figures 4.1.1.7 and 4.1.1.8 illustrates the integration of the fingers with the hand.

Skeleton of the palm, as mentioned before is intended to be able to fold about 30 degrees toward inside of the hand in order to facilitate the hand grabbing thinner objects as well as it would for wider things. The actual implementation of such palm would be made by using a metal hitch to join to flat metal pieces, one representing the upper part of the hand, and the other as the bottom part of the palm, which would be about two thirds the size of the whole palm. The hitch is going to be installed internally welded for cosmetic purposes but preserving the required functionality to execute desired operations. The degrees of folding the palm toward the inside could be adjusted once the physical implementation is built because it could vary depending on how much space would be available for such operation without compromising fundamental functionality inside the skeleton of the hand. Perhaps additional features would be needed from the design point of view to optimize the folding mechanism.

A hand in general is desired to have some artificial intelligence, which would be accomplished by incorporating several sensors and a computer code to act and react to certain external and/or internal stimulus. In order to do that, the palm would have several openings on the metal plates that represent the palm of the hand to place the selected set of sensors as seamless as possible. The current prototype of the robotic hand conceptually has proximity sensors on the palm to read an object getting closer or actually touching the hand.

There is an important feature that the group wants to implement on the hand application; that is the detection of a slippery object from the hand. When an object is grabbed, depending on its weight or composition, it could start gently sloping down from the hand if the necessary corrections are not taken. In order to solve that potential issue, the group would use a dedicated sensor installed on the palm of the hand to detect if the object is in place or on the contrary a delta position have been applied to the original position.

Based on the reading of the sensor's input, a sequence of computer instructions would take the adequate corrections with the help of other sensors installed in the tip of the hand's fingers, where most of the force-pressure would be made to maintain the position of an object without compromising the integrity of its material composition due to unnecessary force applied when the issue is not addressed. To install the dedicated sensor into the palm of the application hand, and opening would be made on the bottom part of the palm, which is the wider part, so that it would function for most of the objects on which the robotic hand would be tested. It is intended to use a computer mouse like sensor to detect any motion of the object once it has been grabbed.

Next is presented the state diagram of the finite state machine, which would control the process of interaction between the involved sensors for the grabbing functionality.

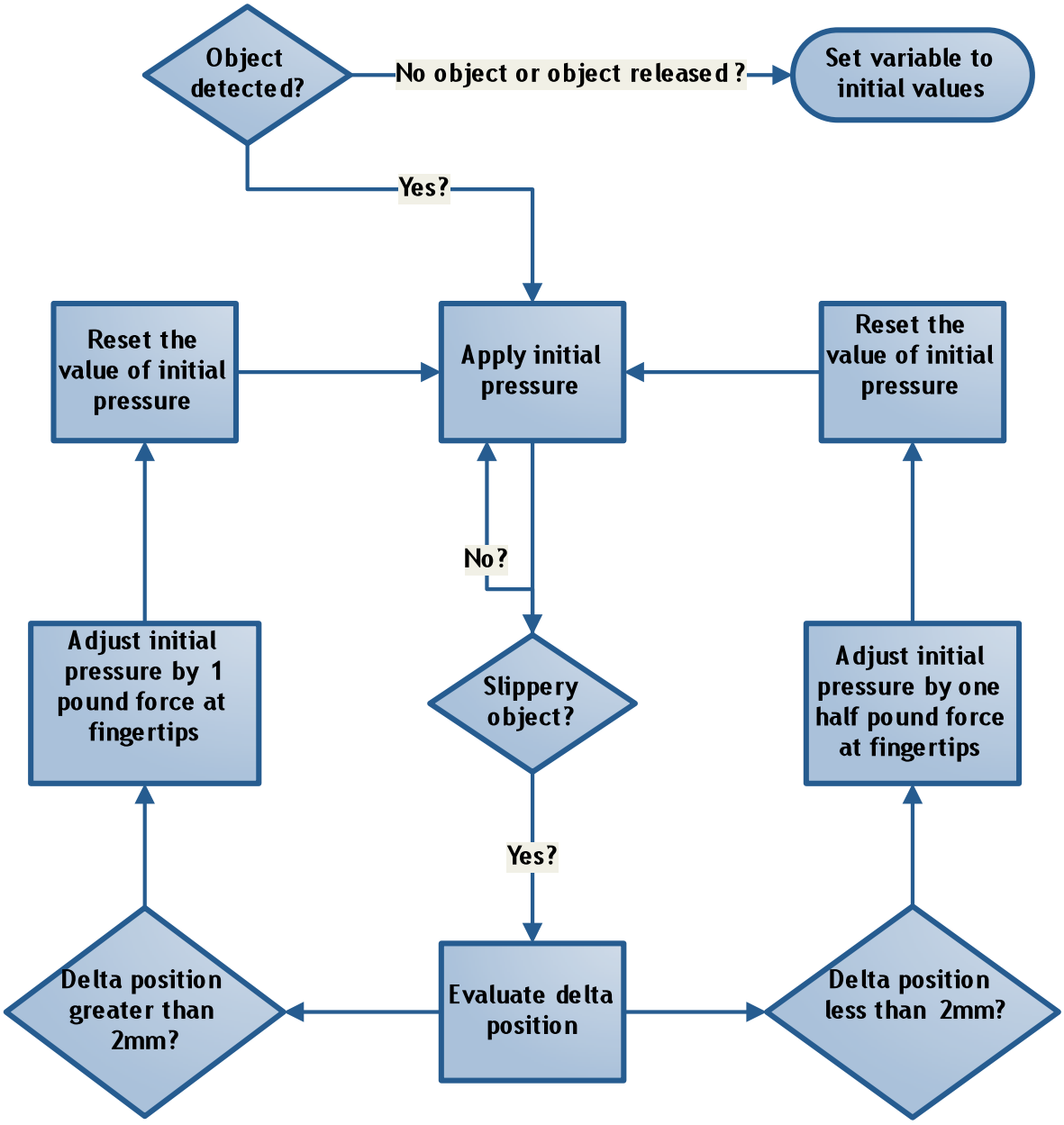


Diagram 4.1.1: Finite state machine to control the sensing process for grabbing functionality.

As mentioned before, if there is a difference between the original position at when the initial pressure was applied, and the position evaluated after 100 milliseconds,

then that variation of the original position with respect to a small amount of time is the so called delta position with respect to time. Also, the adjustment of the pressure-force executed at the time of evaluating the delta position would be calibrated at the time of testing when the application would be in its final stages.

A press sensor is planned to be installed in each one of the fingertips of the robotic hand to complement the process of grabbing and holding tight enough an object, and varying the pressure depending on the nature of that object. At the beginning of the grabbing stage, the variables would be set to a minimum prefixed number, and the units would depend on the variables. Therefore, at the time of evaluating if an object is slippery, there would not be the possibility for reducing the pressure applied to the object because up to now the group have not found a suitable procedure to evaluate the type of object before grabbing it or the way to evaluate if there were to much pressure applied to the object in order to reverse the action.

Since the thumb is a special finger in different forms for a human hand, the group has decided to take a different approach than the one taken for the remaining four fingers up to now, so that the implementation of Luke's hand would resemble a realistic implementation of a hand. The differences begin with the number of phalanges implemented in the design for the other finger compared with the thumb, continue with the mechanical operation of the different type of fingers constructed for this application, and end with the functionality that the artificial thumb can execute within the application of this robotic hand.

The thumb of Luke's hand has three phalanges from which one is the base of the finger whereas the remaining two are the component of the actual finger like in the human hand. The base phalange is going to be fixed somewhere in between 45 and 60 degrees in the X-Y axis plane, but that phalange is at the same time going to be movable on the Z-axis from zero up to about 45 degrees in order to provide the thumb with additional realism by moving it some like a limited joystick. The other two remaining phalanges would behave like the other finger, except that in this case the knuckle is at the joint of the base and second phalanges.

For the mechanical operation of the thumb, the design counts on controlling it in two different manners as follow: First, a DC motor, fixed against the inner part of the back of the palm, would control the rotation of the base phalange at the specifications described above. Second, the mechanism to control the other two phalanges of the thumb is going to be similar to the one implemented to control the other four fingers, with a string and a dedicated servomotor like the tendon-muscle relationship in a real human hand.

Thumb's functionality, like in a human hand, is extremely important within the role of the whole robotic hand. Therefore, to provide the hand with higher precision at the time of grabbing a thin object similar to a pen, the thumb should move toward the other fingers, and it may need to be calibrated to lie between the index and the middle fingers leaving the ring and the little fingers static in home place. Additional functionality is provided to the thumb when it closes toward the palm of Luke's

hand, and it could be exploited by using it as an independent grabbing functionality for more limited type of objects.

So far the materials discussed for Luke's hand are aluminum, metal sheets, hexagonal metallic nuts, and metallic rods for the external part of the hand because it is pretty much the skeleton of the hand. Thus, some type of cover material is required for the robotic hand, but specially on the fingertips as well as the palm's surface, where must of the contact for the grabbing functionalities would happen. Such material has not been decided yet, but it has been thought for providing gripping for the application hand at the time of execution. Some of the alternatives for the cover material are leather gloves or rubber of silicon type.

Inside the Luke's hand, at the skeleton level, strings would be running to control the functionality of most of the fingers mobility. The wiring of the sensors located at the fingertips as well as the ones at the palm would be run alone the mentioned strings toward the arm of the application in order to feed the microcontroller with input data. Also, a DC motor is going to be located inside the hand in order to control certain mobility of the thumb. The bulk part of electronic components such as actuators, controllers, and the main PCB are going to be located at the extension of the robotic hand like an arm in a human hand.

Like any human hand, Lucks application would have a wrist, which would be implemented to give important functionality in case of lifting heavy objects in a different procedure than the one put into operation for the grabbing functionality. Using a stepper motor, which would control metal plates at the joint of the arm and the Luke's hand, would do the mechanical execution of the wrist. A nylon washer and metallic nuts would be required to keep the metal plates at the wrist in place but avoiding metal friction at the time of rotation. The rotation at the wrist could be accomplished manually by executing the portion of the computer program coded for that purpose when the size or the weight of the object is suited for that special procedure in order to keep the object under control.

The arm of Luke's hand is going to serve for several purposes such as the needed space for locating the electronics components that are not buried inside the palm of Luke's hand; as a pivot point at the elbow level to lift a grabbed object after balancing the object by adjusting the wrist rotation. The elbow design is going to be explained in details in the next section of this chapter. The object would be lifted from the 'ground level' to demonstrate the functionality of this procedure; and as support for the joint of the hand and the wrist, where an important rotational functionality is implemented. The arm, as an extension of the robotic hand adds important functionality to overcome certain limitation and with the assistance of the sensors installed in order to maintain the control of objects at the robotic hand.

Next is presented the block diagram of the procedure for grabbing special objects, from which the rotation of the wrist is fundamental for balancing whether a large object or an uneven distributed object's weight. The definition of special objects is explained in detail in the testing section of this document.

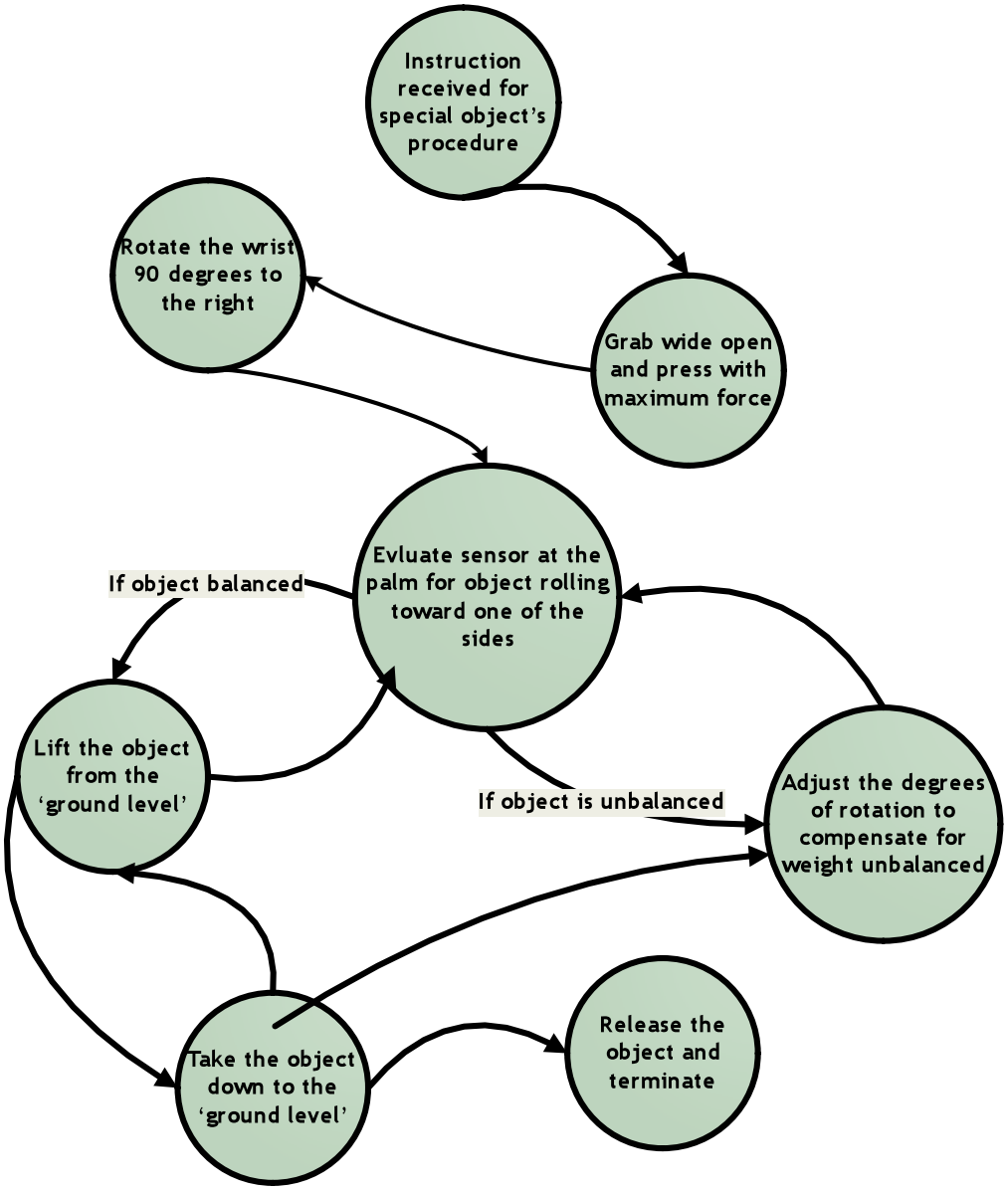


Diagram 4.1.2: Definition of the procedure to control the sensing process for grabbing special objects.

4.1.2 Elbow Mechanism

The Elbow mechanism is going to be the instrument that is in charge of the movement of the arm. The arm has to be able to move up and down so when the robotic hand grabs an object the whole system can lift it. The elbow will use one

DC torque motor in order to move. This DC torque motor will be slow but very strong; which will allow the robotic hand to grab a more variety of objects.

The elbow will have one axis, which is going to be built in with a gear. This gear will create the link that the axis of the elbow needs to have in order to connect with the torque motor. The torque motor will provide with the enough torque to lift the whole arm. If the torque motor doesn't have the enough force to lift the hand then both of the gears that link the motor and the axis will come in play.

The gears can be manipulated to create more torque; the only problem that the whole system can face is the instability of the gears when it comes to staying in place and not getting out of the shafts. To solve that problem, two stoppers are going to be placed right next to the gears to keep it in place. The torque motor will be chosen in senior design two but the motor will have to surpass the estimated weight of the whole hand, arm and object. The estimated total weight that the whole system will have is from five to seven pound depending on the weight of the object. The torque motor will have to go beyond those requirements. Figure 4.1.2.1 shows the design of the elbow mechanism and how the torque motor will be connected to the arm to make it go up or down.

Figure 4.1.2.1 Elbow mechanism

4.1.3 Software

The governing firmware for the robotic hand will be composed of several sections, divided in separate files for better modularity, and used as libraries by the main header file. These libraries will contain specific functions to properly handle data used/produced by all motors, sensors, and wireless communications. The file containing the information's for sensors (*sensors.h*) will have the specific algorithms to be used for the temperature, force, vibration, and proximity sensors. As of now, an IR sensor will be used as the proximity sensor, and its only function is to determine if an object is within reach to be grabbed. The temperature sensors will work on similarly to the IR sensors, the MCU will compare the temperature of the object with certain. The pseudo code for these algorithms can be seen on *Figure 4.1.1*.

```
int withinReach(double distanceIR){
    // distance in millimeters
    const double minimumDistance = 20;

    // 1 = true, 0 = false
    if (distanceIR <= minimumDistance)
        return 1;
    else
        return 0;
}
```

```
int temperatureLevel(int objTemp){
    // temperature in celsius
    // MIN and MAX are based on specifications of
    // all parts used by the hand
    const int minTemp = MIN_VALUE;
    const int maxTemp = MAX_VALUE;

    // 1 = true, 0 = false
    if (objTemp <= maxTemp && objTemp >= minTemp)
        return 1;
    else
        return 0;
}
```

Figure 4.1.1 IR and Temperature sensors algorithms

The vibration sensor will be used to determine if the object is sliding, in which case the force applied to the area of the sensor will change. The MCU will receive the signal as a DC voltage that is proportional to the force, and will calculate the rate at which this force is changing based on the sampling rate. Based on this rate, the MCU will send the appropriate signals to the motors to apply a higher force to the object and avoid the object from slipping. An initial version of the algorithm for the vibration sensors can be seen in the following block of code:

```
// detect if object is slipping using two consecutive sample from the sensors
// and the sampling ratio
int detectSlipping(double Force1, double Force2, double sampling){
    double tolerance = 0.97;
    if (force1 < Force 2 && (Force1/Force2) < tolerance)
        return 1;
    else
        return 0;
}
```

```
// determine the slipping rate of the object
// sampling is being passed in seconds
double slippingRate(double Force1, double Force2, double sampling){
    double rate = 0;
    rate = (Force2-Force1)/sampling;
    return rate;
}



// Send the proper signal to the motors to increase the force applied
// pulse is sent in milliseconds
double correctPulse(double rate, double pulse){
    return (pulse+ pulse*(1-rate));
}



// Correct Slipping of the whole hand
void correctSlipping(double *pulse0, double *pulse1, double *pulse2, double *pulse3,
                     double *pulse4, double *pulse5){

    // if object is slipping find the rate at which it is slipping
    // and uptate the pulse being sent to each motor
    if (detectSlipping(Force1, Force2, sampling)){
        double rate = slippingRate(Force1, Force2, sampling);
        *pulse0 = correctPulse(rate, pulse0);
        *pulse1 = correctPulse(rate, pulse1);
        *pulse2 = correctPulse(rate, pulse2);
        *pulse3 = correctPulse(rate, pulse3);
        *pulse4 = correctPulse(rate, pulse4);
        *pulse5 = correctPulse(rate, pulse5);
    }
}
```

The force sensors on the tip of each finger/thumb will send a signal as a voltage that will be proportional to the force being applied; the MCU will take this information and use it to determine if the force being applied to the object will not break it. The application will control each motor in an independent manner, sending the correct pulse to each servomotor depending on factors such as; current force being applied, force needed to avoid slipping, and whether the finger is actually in contact with object.

If the object is within reach, and needs to be grabbed, the MCU will update the pulse of each finger using the *updatePulse* function until the forces sensors send a positive voltage indicating contact with the object, then the *correctSlipping* function will come into play to ensure the object is properly held (Figure 4.1.2).

```
void closeHand (double force1, double force2, double force3, double 4){
    double forces = force1 + force2 + force3 + force4;

    // Increase the pulse until the hand is touching the object
    while (forces == 0 || forces <= TOLETANCE_VALUE) {
        *pulse0 = correctPulse(rate, pulse0);
        *pulse1 = correctPulse(rate, pulse1);
        *pulse2 = correctPulse(rate, pulse2);
        *pulse3 = correctPulse(rate, pulse3);
        *pulse4 = correctPulse(rate, pulse4);
        *pulse5 = correctPulse(rate, pulse5);
    }

    // avoid slipping of the object by correcting the forces applied
    void correctSlipping( *pulse0,  *pulse1,  *pulse2,  *pulse3, *pulse4,  *pulse5);
}
```

Figure 4.1.2 Code to Close the Hand

Lastly, an interface selection algorithm needs to be implemented to make the appropriate function calls depending on the interface currently ordering the hand, and to prevent more than one interface to establish a connection with the hand. This part of the program will have to deal directly with RF module if finally implemented, and can be seen in the following blocks of code:

```
// Ligh up the LEDs based on precedence
if (connectionStatus == 0){
    switch (activeControl){
        case 0;
            GloveControl = 1;
            PodControl = PCInterface = RControl = 0;
            break;
        case 1;
            PodControl = 1;
            GloveControl = PCInterface = RControl = 0;
            break;
        case 2;
            PCInterface = 1;
            GloveControl = PodControl = RControl = 0;
            break;
        case 3;
            RControl = 1;
            GloveControl = PodControl = PCInterface = 0;
        default:
            RControl = PodControl = PCInterface = GloveControl = 0;
    }
}
```

```
// LED Indicators
int GloveControl, PodControl, PCInterface, RControl;
// GloveControl=0, PodControl=1. PCInterface=2, RCControl=3
int activeControl;
// 0 if no connection exists, 1 is being used by a controller
int connectionStatus;
```

```
    // Create the connection with the corresponding interface
    switch (activeControl) {
        case 0:
            GloveControl();
            break;
        case 1;:
            PodControl();
            break;
        case 2:
            PCInterface();
            break;
        case 3:
            RControl();
            break;
        default:
            break;
    }
```

4.2 PCB Design

There are multiple ways to solder together a custom circuit:

- Point-to-Point Prototyping Wiring
- Wire Wrapping
- Prototype Printed Circuit Board

4.2.1 Point-to-Point Prototyping Wiring and Wire Wrapping

The Point-to-point wiring refers to the practice of mounting electronics components to a prototyping PCB and connecting the leads directly together with the soldering iron. This method requires more manual labor so it is the most difficult method of creating the circuitry. Also it is the slowest and there could lead to multiple errors in the wiring. Building our circuit design using this method would not be a good idea. The second method is Wire Wrapping; it is used to create very complex circuits and it is extremely robust. The connection is made using a bare wire which is wrapped around a long post; it could be connected to the pin of an IC or some other component. This method seems pretty efficient for our circuit design but it requires

specialized tools and components sockets. Also, it is time consuming and as the design gets more complex there could be errors which could be difficult to trace.

4.2.2 Prototype Printed Circuit Board

The third method and by far the most highly efficient and inexpensive is a Prototype Printed Circuit Board. The big advantage of using a PCB is the ability to change and replicate the circuit very quickly. A PCB is composed of one, two or multiple insulator material layers with electrical conductors; one side has the components, such as resistors, capacitors, coils, transistors, diodes, and integrated-circuit units. The other side has the conducting paths printed with silver or cooper on the board instead of using wiring. Small metal eyelets or just holes in the board are used to connect the components to the wiring. The PCB can have through-hole connections or surface mount. It is up to the designer to choose, but circuit design for the robotic hand will be the through-hole connection. But, when soldering the components, it is important not to use too much heat in soldering or disordering. Otherwise the printed wiring can be lifted off the board. As a safe mechanism a small iron of about 25 to 30W rating has to be used. The picture below shows how a PCB board would look like. The following PCB board figure 4.2.2.1 illustrates a circuit designed.

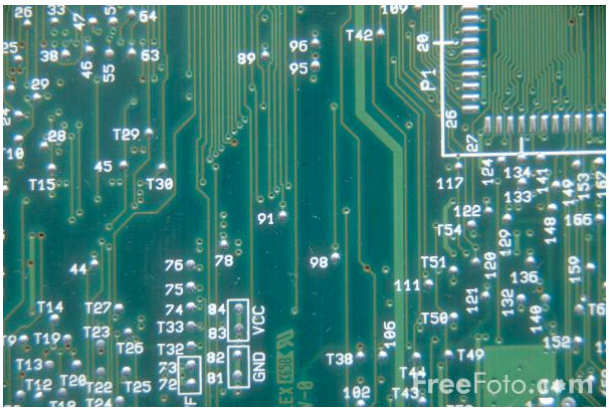


Figure 4.2.2.1: PCB Board Example
"Courtesy of FreeFoto.com"

It is worth noting that currently the main generic standard for printed circuit board design, regardless of materials is IPC-2221A. Whether PCB board is single-sided, double-sided or multilayer, this standard provides rules for manufacturability and quality such as requirements for material properties, criteria for surface plating, conductor thickness, component placement, dimensioning and tolerance rules, and more. For a specific technology the designer can then choose the appropriate sectional standard from the IPC-2220 series. For power conversion devices, additional parameters are recommended by IPC-9592. The width of the circuit conductors should be chosen based on maximum temperature rise at rated current and acceptable impedance. The spacing between the PC traces is determined by

peak working voltage, the coating and the product application. The minimum possible width of the traces and spacing between them are limited by the manufacturing capabilities of the manufacturer and should not be less than 2 mils. Typical minimum values are 6/6 mils. Depending on the application, other standards may also apply. IPC and other standards do not tell how to properly route the board. Good PCB layout techniques require understanding of the effects of non-zero trace impedance and coupling of signals from one circuit to another through parasitic capacitances and radio transmission, as well as basic understanding of circuit operation. Auto-routing may be done for most parts of control circuits, but power, ground and high di/dt circuits should be routed by hand.

There are different types of software in the market for designing PCBs, but Express PCB is the software chosen for this project because it is one of the most reliable, easy to use, and it is free to download. Express PCB software includes an ExpressSch for drawing schematics and ExpressPCB for circuit board layout. A circuit design has to be made and tested before drawing the schematic of the circuit. Then, when the schematic and the PCB layout of the circuit are done, this program offers error checking, which alert the designer of any miss connection or error in the schematic. There are different tutorials in the internet dedicated to PCB design. Afterward, when the PCB layout is done, ExpressPCB offers a low cost printed circuit manufacturing service which can be ordered online. This service is fast, inexpensive and the most important it will save the project a lot of time.

4.3 Circuit and Power Supply Design

The circuit schematic will be designed using ExpressSch; for all reasons mention in the previous section.

4.3.1 Design Rules for PCB Layout

There are important rules to follow when designing a PCB. The traces (Red lines on the PCB) for power supply has to be bigger that traces for signal because power supply carries large amount of current and space it's needed to carry all of it. In this PCB the traces from the battery supply are 0.060', the traces from the LM323 (5V-3A) regulator to the motors are 0.060'', the traces from the LM2937 (3.3V-500mA) to the microcontroller are 0.0030'', the traces from the LM340-5 (5V-1A) to the different sensors are 0.025'', and the traces for the signal connections are 0.015''.

When building a PCB is it important to have a common ground for all the components in order to reduce noise. Express PCB has a nice tool to accomplish this, place filed plane, it makes a whole layer ground (Green Plane on PCB) and by using the solid pad to filled plane function all the ground connections are connected to this plane. There are times

when two or more traces crossed each other which are not good at all. To prevent this from happening there is a function where two 0.031 ''round via with 0.014'' holes can be placed; this will allow placing a pad under the copper layer and crossing between copper traces are eliminated. This can be seen on the PCB graph above on the small green traces surrounding by a black line.

The Express PCB website tells the respected traces length with respect to the current that needs to be carry. These are mention in the table below:

Traces	Current
0.010''	0.3Amps
0.015''	0.4Amps
0.020''	0.7Amps
0.025''	1.0Amps
0.050''	2.0Amps
0.100''	4.0Amps
0.150''	6.0Amps

Table 4.3.1.1

This picture below was obtained from the Express PCB schematic software by using the export image of mechanical drawings tool.

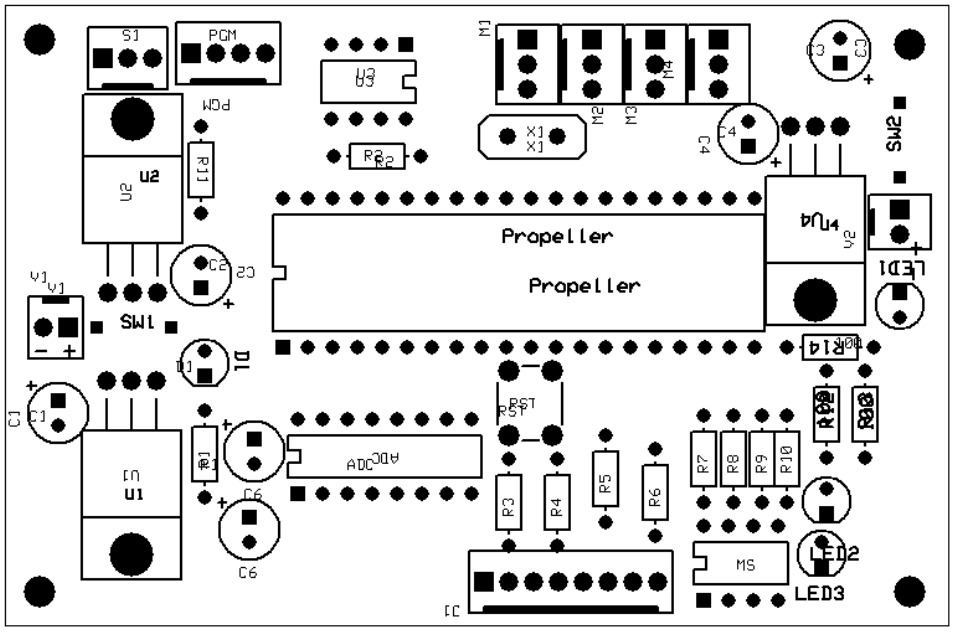


Figure 4.2.2.2: Luke's Hand PCB Mechanical Drawing

4.3.2 Luke's Hand Schematic

The Luke's hand schematic shows the circuitry connection of the microcontroller to the servomotors and the sensors. The circuit design shows the connection between the different components and the power supply required to work. All these components are located in a single PCB. The table below shows all the parts that have been included in the schematics:

Component	Name/Size
Microcontroller	P8X32A-D40
C1-C2	1000uF
C3-C7	10uF
PW LED-LED1-LED3	4-different colors
Servo Connector	M1-M4 (3-Pin)
PGM	Propeller Plug
5MHz Crystal	HC-49US
A/D Converter	MCP3208
R1	100
R2	10K
R3-R6	100K
R7-R10	10K
R11	1K
R12-R14	100
SW	4-Dip Switch
SW1-SW2	2-on/off switch
Voltage Regulators	LM2940 LM2937 LT323
EEPROM	U3-24LC256
V1-V2	9 Volts Battery (2-Connector)
Flex Connector	J1 (8-Pin)
Reset Bottom	RST (Tact Switch)
Sonar Connector	S1 (3-Pin)

Table 4.3.2.1: Components of Luke's Hand Controller

The schematic of Luke's hand shows that the communication to the computer is through USB 2.0 connection, but in order to have a better understanding with the Propeller microcontroller, the Propeller plug is recommended to be used for communication.

Also, all the servo motors have their signal wire connected to selected pins of the propeller (P20-P23). These pins are set to be outputs and they send a pulse, known as PWM (Pulse Width Modulation) to the servo motors. This PWM is a digital output that has a square wave of varying duty cycle. Depending on the length of the PWM, the servo motors will rotate certain degree. For the servo motor used in the project, the PWM has to be from 0.8ms to 1.9ms. As it was mention earlier, each motor will move a single finger. The table 4.3.1.2 shown below, list the motor number, the finger that it is moving, and the microcontroller pin number to where it is connected. Also, the PMW that need to be sent to the servo motor for flexion and extension of the finger is shown.

Servo	Finger	Microcontroller-Pin	PWM-Flexion(ms)	PWM-Extension(ms)
M1	**Pinky**	23	0.9	1.9
M2	**Ring**	22	0.9	1.9
M3	**Middle**	21	0.9	1.9
M4	**Index**	20	0.8	1.9

Table 4.3.2.2: Motor distribution to each finger

Using an 80 MHz system clock, the output pulse resolution is between 12.5ns - 50ns, which in theory corresponds to about 140000 possible positions for a Futaba standard servo like the one used in this project. The signal pin from the microcontroller creates a steady stream of signal pulses with high signals that are equal to the position value of the servo, times the clock period in length, and low signals that are about 20ms in length.

There are also four 2-pin connectors, one for each flex sensor arranged to incorporate those sensors in the PCB. The voltage divider and those flex resistive sensors are connected as shown in the figure 4.3.1.1 below. Additionally, there are four signals coming from the 4-Dip switch controller, one signal for each finger. Also, a 3-pin male connector is used to connect the Ping ultrasonic sensor to the circuitry; one of its pin is connected to the microcontroller. The signals coming from the sensor are connected to the microcontroller as inputs. Therefore, the propeller will be alerted to any signal coming from them or the switch control and it will deliver the right output to the motors. An external 5MHz crystal is used to speed up the processing of information, and as it was mention earlier this crystal would make the system clock of the microcontroller run at 80MHz. The schematic of the circuitry is divided by sections, each having its own name and function. This is done to make it easy to understand how the components are distributed in the schematic. The final schematic for this project is shown in the figure 4.3.1.1.

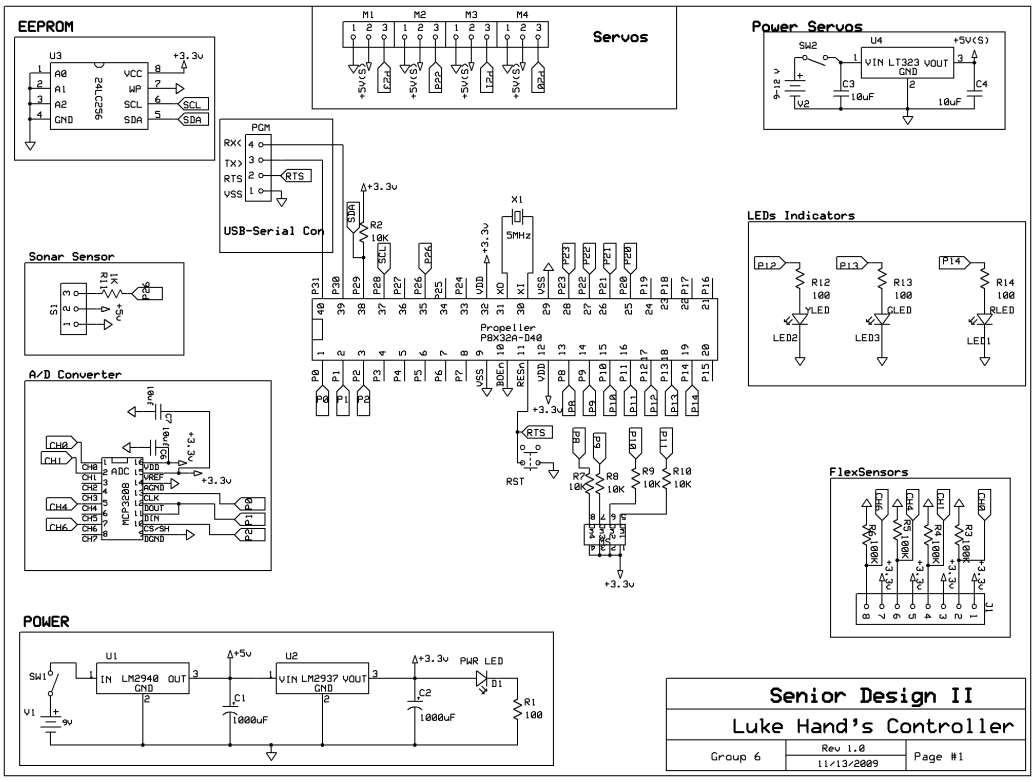


Figure 4.3.2.1: Schematic of the Luke's hand Circuitry

As we can see in the figure above, the schematic is divided in section in order to make it easy to understand how all the components are distributed in the circuitry. It is important to mention that the Propeller Microcontroller has two pins which are already reserved for the connection to the computer. These are Pin 30 (Serial TX to Host) and P31 (Serial Rx from host); so these pins cannot be utilized to drive motors or to received any information from sensors or remote controllers. Two pins also important to mention are P28 (I2C SCL) and P29 (I2C SDA), which are connected to an external EEPROM in order to store the final program.

4.3.3 Luke's Hand PCB Layout

As it was mention before, the PCB layout design was done using Express PCB, which offers an error detection tool to save time at the development stage of the robotic hand. The figure shown below is one of the views offered to visualize the final product of PCB design. This view presents all the motor, sensors, and servo connectors. In the PCB view, the components such as microcontroller, the voltage regulators, capacitors, resistors, switches are also included in order to be able to have a complete picture of what would be the final product from the circuitry point of view. It is worth noting the PCB is made up of three layers: silkscreen layer (yellow), top copper layer (red) and

bottom copper layer (green). The figure below shows all the connections paths, components packages, and the different layers.

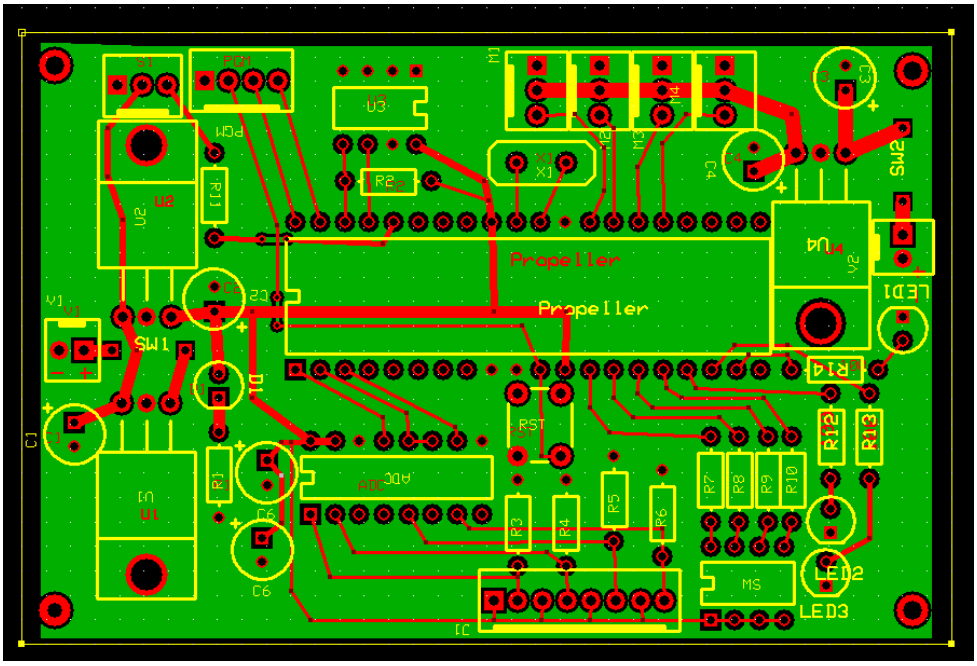


Figure 4.3.3.1: Luke's Hand PCB Layout

The PCB is made of the following components:

- The voltage regulators are TO-220 package with mounting hole.
- The Servo Motor Connectors are Molex, .1 inch KK, up, 3 pin.
- The USB computer is a Molex, .156 in KK, up, 4 pin
- The Voltage supply connectors are Molex, .1 inch KK, up, 2 pin.
- The capacitors are Radial electrolytic - Lead spacing 0.1 inch (2.5mm)
- The Resistors are 0.25 watt (lead spacing 0.35 inch)
- The Microcontroller is a Propeller P8X32A-D40, Dip 40 pin package
- The A/D Converter is a MCP3208, Dip 16 pin package
- The EEPROM is a 24LC256 (32KB), Dip 8 pin package
- The Switch Controller is a Dip 8 pin package
- The flex sensor connector is a Molex, .1 inch KK, up, 8 pin
- The Sonar Sensor connector is a Molex, .1 inch KK, up, 3 pin
- The reset bottom is a Switch - 6mm Push button
- The LEDs are 3mm T1 LED lamps

When the layout out for a printed circuit board uses analog components, care should be taken to reduce noise as much as possible. As it was mention earlier, bypass capacitors

are used with the MCP3208 in order to reduce the noise. Also this PCB design has common ground plane, which is recommended to keep the ground potential the same for all devices on the circuit board. It is worth to mention that the MCP3208 provide both digital and analog ground connections to provide another mean for noise reduction. The analog and digital circuitries are separated internally in the device. This reduces noise from the digital portion of the device. The two grounds are connected internally through the substrate, which has a resistance of 5-10 ohms. On the PCB, both grounds are connected to the common ground plane shown in the figure 4.3.2.1 above as the green plane.

In order to prevent the main components from getting damage when soldering to the PCB; it was decided to use Dip socket connectors. The PCB has two 4-Dip connectors for the EEPROM and the Switch, one 16-Dip connector for the A/D converter and one 40-Dip connector for the propeller microcontroller. After soldering the connectors it is easy to insert the components to it.

4.3.4 PCB with Component

The PCB of the robotic hand has two layers. The picture below shows the PCB with its respective components and connections. This is the final circuitry assembled for the Luke's Hand application and is shown in the figure 4.3.4.1 below.

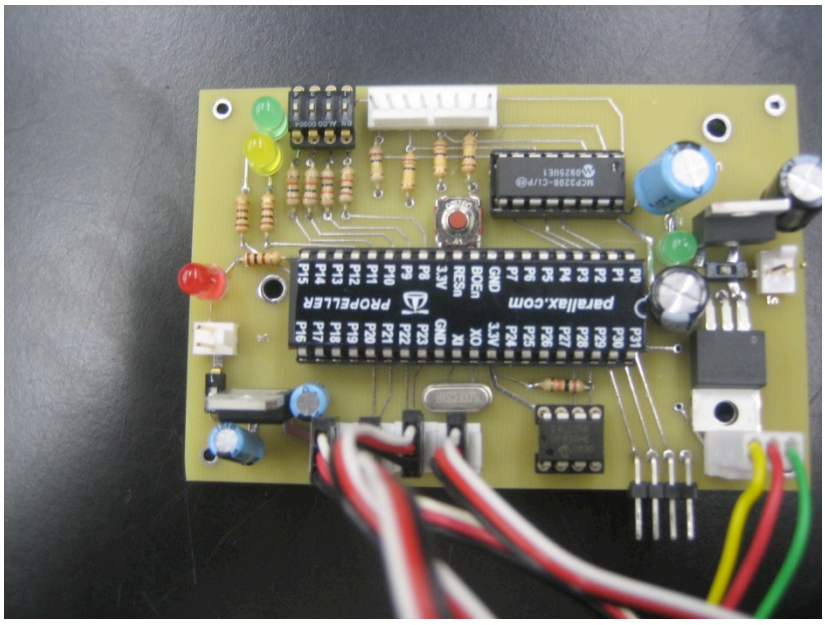


Figure 4.3.4.1: Luke's Hand Circuitry

4.4 Glove Interface

4.4.1. Glove Interface Function

The function of the Remote control glove is to create an interface between the user and Luke's hand. The main idea is that the user wears a normal size right hand glove; which as mentioned before will link him with the robotic hand. The glove's mains function is to mimic, as much as possible, the movements that the user is creating with the right hand. The glove is to translate the mechanical energy of the user's hand into electrical energy to communicate with the microcontroller. The microcontroller then will signal each servo to move independently to recreate the action that the globe is interpreting.

4.4.2 Glove Interface Design

The remote control glove needs to incorporate into its design some kind of sensing mechanism that will capture the flexing movement of the fingers when the hand is moving. The bending of the fingers is the principal function that the robotic hand must mimic since it is the source for grappling an object. If Luke's hand can imitate the gloves movement then it will be able to grab objects with out crashing the object because the user will be controlling the strength of the robotic hand. All the sensors that the Luke's hand has to detect the object on its self will be turned off. The force applied to the object will be based on what the user wearing the glove perceives visually.

Many ideas were brought up to the design of the glove in order to capture the movement of the fingers. The first idea was to use to use a FSR strip touch sensor. The FSR strip Touch sensor is the same exact sensor used in the tip of the fingers in the robotic hand, with the only difference that this touch sensor is long and has a grater sensing area. This Touch sensor was going to be integrated in the back of the finger inside the glove with the wires running down connected directed to the microcontroller to create a direct link with the robotic hand. When the user flexes the finger then the finger will make contact with the touch sensor. The more the user flexes the finger the greater the force that the finger will apply to the touch sensor; which means that the more the servo needs to turn in order to mimic the movement.

The glove needed to be a little tight so that the fingers and the sensors could touch each other when flexing but separated when the user is not flexing. The perfect glove for this design could be any standard utility glove. They have a little space between the finger when the hand is open but nice and tight when closed.

This glove also would be very light since it is a possibility to add a variety of components to achieve an accurate value. Also the glove will be cheap and easy to replace if there is any kind of wear and tear.

Another idea to recreate the movements of the glove was to use a variable switch used in houses to control the intensity of the light. These switches are basically variable resistors with a knob that can move up and down. This movement is great to translate the fingers movement into electrical data. When the fingers flex, the switch will release different resistance data as the fingers flex more and more. This will be the perfect link to the microcontroller so the servos can move as the fingers in the glove by getting a ratio between resistance and degrees of turn.

The glove was finally designed using the previously mentioned Flex Sensors, and its functionality was assigned to the option 12 (1100) in the 4-DIP switch implemented to control all functionality of the hand. The interface between the user and Luke's hand is a smart flex globe. This globe contains four flex sensors attached to the back of each globe's finger. The flex sensor's output is a direct correlation between the changes of human hand's position controlling the globe interface and the voltage drop generated at the flex sensors. While the finger flexes and extends, the voltage output changes and sends a signal to the analog to digital converter (A/D converter), where a 12-bit representation of the voltage is calculated. When the fingers are completely extended, the usual 12-bit value of the voltage output is about 3550. Subsequently, the usual 12-bit value of the voltage when the fingers are completely flexed is about 2100.

A simple relation between the range of finger's flexion and the range of the servos' motion is calculated on the globe controller program. The program calculates a percentage of the voltage range. Since the lower and higher boundaries are set for the 12-bit voltage value, a percentage from their difference is used to make a correlation with the servo's motion. The range of the pulse signal given to the servos to rotate clockwise and counterclockwise is also set. In conclusion, if the finger flexes 25% of the total flexing range, then the same percentage would be used to rotate the servomotor in order to flex the robotic hand's fingers and somehow emulate the human hand in the globe interface. Figure 4.4.2.3 shows the final implementation of the flex sensors and the final design of the glove interface.

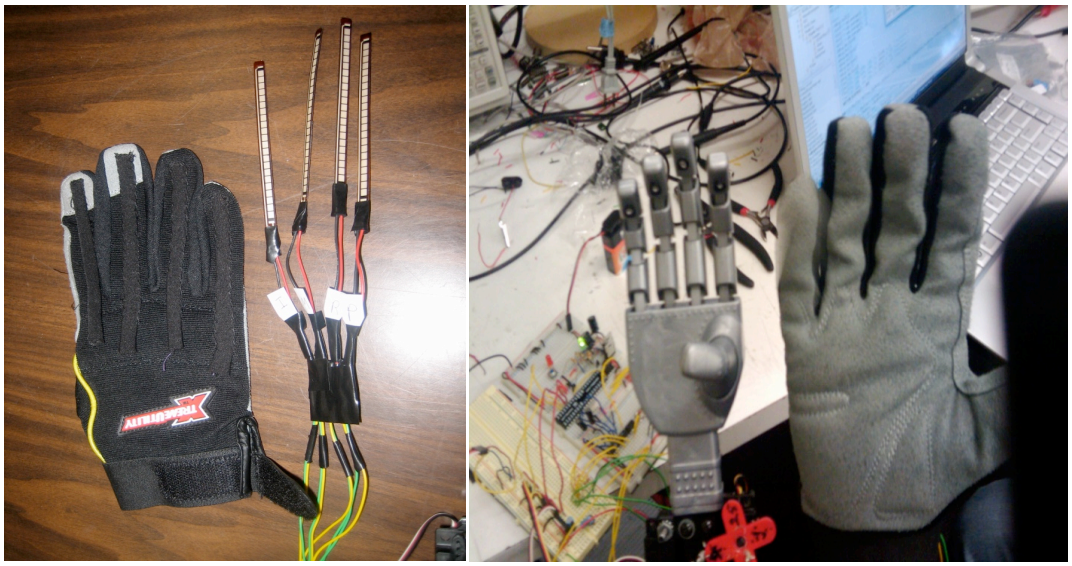


Figure 4.4.2.3 Final Glove interface

4.4.3 Software

The software side of the interface will be in charge of data and signal processing, and some minor aspects of the wireless/wired communication with the hand. The three most important things being handled by the software are the inputs from the potentiometers being used in each finger/thumb, the signal sent to the *hand* to operate each motor, and the sampling rate for the sensors being used. The sampling rate is dependent on the processor, and will be set to an optimal value that will generate enough data for high accuracy will keeping low power consumption.

Regarding the manipulation of data received and sent by the glove, an algorithm that is simple, yet elaborate enough to take as many factor in consideration is being considered. Some change will be made to final algorithm used, but in essence it will function as described below.

An array of floats numbers will be used to receive the signal coming from each potentiometer, these signals will be voltages, which have been previously converted from resistance values, and a second array of floats will be used to store the voltages that need to be sent to the hand to operate each finger/thumb (Figure 4.4.3.1).

```
// Feedback from fingers/thumb
// Index 0 = thumb, 1 = index, etc
float handMotion[5];

// Signal sent to motors
float motorsV[5];
```

```
// Roof and Ceiling for each joint
const float lowMin[5];
float lowMax[5];
float midMin[5];
float midMax[5];
float highMin[5];
float highMax[5];
```

Figure 4.4.3.1 Sensor/Motors Array and Roof and Ceiling Arrays

The data received from each sensor is going to be compared with some previously determined values that will constitute the Roof and Ceiling of certain ranges - these values will be obtained through extensive testing once the hand is built – and depending on the range that the value fall into, the proper voltage need by the motor to mimic the movement of the finger/thumb will be sent to the hand. The data structures used to store these ranges can be seen in Figure 4.4.3.1. The output of the sensor strictly depends on where the force is being applied. In this case the force is being applied at the joint, allowing the program to determine the current position of the finger/thumb. Depending on which joint is applying the force, an appropriate voltage will be sent to the hand (Figure 4.4.3.2)

```
// if the voltage received from the potentiometer is between a predefined range
// sent the assigned voltage to the hand
if (handMotion[0] > lowMin[0] && handMotion[0] < lowMax[0]) {
    motorsV[0] = signal;
} els if (handMotion[0] > midMin[0] && handMotion[0] < midwMax[0]) {
    motorsV[0] = signal;
} els if (handMotion[0] > highMin[0] && handMotion[0] < highMax[0]) {
    motorsV[0] = signal;
}
// do the same for each finger/thumb
```

Figure 4.4.3.2 Output Selection

Lastly, the PCB board includes an LED that indicates whether the Glove is in use or not. The value for each state will in theory be stored as *Booleans,* but given the constrains inherited from the SPIN language, in which *Boolean* is not a primitive type, these states will be stored as integers, where 0 (zero) will represent off, and 1 (one) will represent on. Certainly, the program will make sure that not two statuses are marked as current at the same time.

4.4.4 Software

The software side of the interface will be in charge of data and signal processing, and some minor aspects of the wireless/wired communication with the hand. The three most important things being handled by the software are the inputs from the potentiometers being used in each finger/thumb, the signal sent to the *hand* to operate each motor, and the sampling rate for the sensors being used. The sampling rate is dependant on the processor, and will be set to an optimal value that will generate enough data for high accuracy will keeping a low power consumption.

Regarding the manipulation of data received and sent by the glove, an algorithm that is simple, yet elaborate enough to take as many factor in consideration is being considered. Some change will be made to final algorithm used, but in essence it will function as described below.

An array of floats numbers will be used to receive the signal coming from each potentiometer, this signals will be voltages, which have been previously converted from resistance values, and a second array of floats will be used to store the voltages that need to be sent to the hand to operate each finger/thumb (Figure 4.4.4.1).

```
// Feedback from fingers/thumb
// Index 0 = thumb, 1 = index, etc
float handMotion[5];

// Signal sent to motors
float motorsV[5];

// Roof and Ceiling for each joint
const float lowMin[5];
float lowMax[5];
float midMin[5];
float midMax[5];
float highMin[5];
float highMax[5];
```

Figure 4.4.4.1 Sensor/Motors Array and Roof and Ceiling Arrays

The data received from each sensor is going to be compared with some previously determined values that will constitute the Roof and Ceiling of certain ranges - these values will be obtained through extensive testing once the hand is built – and depending on the range that the value fall into, the proper voltage need by the motor to mimic the movement of the finger/thumb will be sent to the hand. The data structures used to store these ranges can be seen in Figure 4.4.5. The output of the sensor strictly depends on where the force is being applied. In this case the force is being applied at the joint, allowing the program to determine the current position of the finger/thumb. Depending on which joint is applying the force, an appropriate voltage will be sent to the hand (Figure 4.4.4.2)

```
// if the voltage received from the potentiometer is between a predefined range
// sent the assigned voltage to the hand
if (handMotion[0] > lowMin[0] && handMotion[0] < lowMax[0]) {
    motorsV[0] = signal;
} els if (handMotion[0] > midMin[0] && handMotion[0] < midwMax[0]) {
    motorsV[0] = signal;
} els if (handMotion[0] > highMin[0] && handMotion[0] < highMax[0]) {
    motorsV[0] = signal;
}
// do the same for each finger/thumb
```

Figure 4.4.4.2 Output Selection

Lastly, the Glove will have an array the LEDs that will function as status indicators. Solid green if on and a connection with the *hand* has been established, blinking green if the glove is currently transmitting, and solid yellow if the glove is on but not connected to the *hand*. The values for each state will in theory be stored as *Booleans,* but given the constrains inherited from the C-Language, in which *Boolean* is not a primitive type, these states will be stored as integers, where 0(zero) will represent off, and 1 (one) will represent on. Certainly, the program will make sure that not two statuses are marked as current at the same time.

4.5 PodControl Interface

The PodControl interface is another way to create a link between the user and Luke's hand. The PodControl idea came up when various questions about what type of users are targeted with this project. People that have lost an upper limb are the most beneficial from this type of research and projects. If they consider the idea of wearing robotic prostheses to help them in their daily actions, a more sophisticated interface needs to be created.

The PodControl is a combination of two pads, one for each foot. The most important function of the pads is to difference when the user is walking and when the user is utilizing the pad to move the robotic arm. There are several ways to do accomplish this function. One of the designs is to integrate force sensors along the two PodControls. These force sensors will be various functions. FSR touch sensors similar to the ones integrated in the robotic hand will be use for this interface. The main difference of these force sensors is the area of sensing and shape. There are two different FSR sensors that will be used in this project. The first FSR touch sensor is the 1.5-inch square sensing FSR force sensor has a total length of 3.5-inches and a total width of 2.75 -inches. This sensor will be integrated in both PodControls and

they will have the function of recognizing if the person is walking or just standing up. The sensor will be place just underneath the big toes metatarsus where it is always under pressure when a person walks. When a person walks, the force applied on each foot changes drastically from all the weight of the body to zero. Only one foot remains in contact with the floor while the other shifts forward to remain walking. One way of programming the sensor and making them work together can be to detect the amount of force applied to each sensor when the person is walking and make a comparison between them. If they have a big difference in force that means that the person is walking; but is the weight is distributed between both feet the force applied to them will be almost the same. This way the robotic arm will know that the user is just standing up. Figure 4.5.1 shows the FSR Touch sensor that will be used in both PodControls.

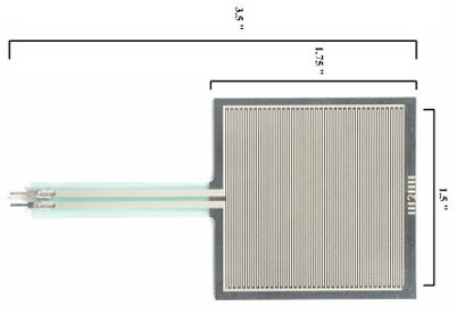


Figure 4.5.1 FSR 1.5-inch Force Sensor

After the robotic arm detects that the person is standing up, the second stage of the PodControl function can initiate activating the other set of sensors that are going to be added to one controlling PodControl for the movement of the robotic hand. Depending on the user preference in choosing which foot to use for the PodControl in charge of the direct controlling and movement of the hand can be adapted. The controlling PodControl will also have another FSR force sensor similar to the one used in the robotic hand just with a little more sensing area. A circular 0.5-inches FSR touch sensor with a total length of 2.375-inches and a width of 0.75- inches. There will be a total of two sensors integrated in the controlling PodControl. One sensor will be located just under the heal of the foot. When the user applies a force to the heal, it will activate the sensor and open the hand. The other FSR touch sensor will be attached underneath the big toe. The big toe sensor will close the hand when force is applied. Figure 4.5.2 shows the sensors that are going to be integrated for the opening and closing of the hand.

Figure 4.5.2 FSR 0.5 touch sensor

These sensors will be controlled by the microcontroller and also linked by wireless module. In the section of research more about wireless communications will be explained and how the PodControls can be controlled using this technology. The PodControl also will count with a gel membrane that wont let the person feel any type of electronics that run beneath the foot and can be comfortable for the person to wear. Figure 4.5.3 shows how the sensors are going to be integrated in the both of the PodControls.

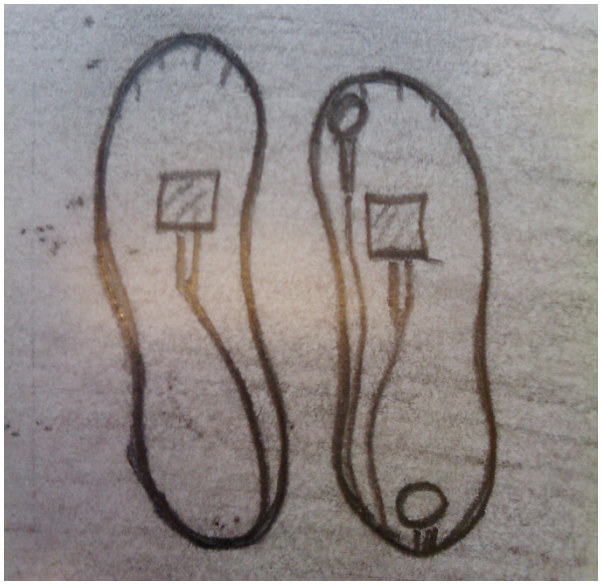


Figure 4.5.3 PodControl Design

4.5.1 Software

The data being handled by this device is relatively simple, much simpler than any of the other interfaces previously described. Since it will only have three (3) force sensors; one will be used to receive input from the user to close the hand, the other will serve the same purpose when the hand its to be opened, and the last will turn on/off the controller for better functionality, and lower power consumption. In essence, the algorithm will work pretty much like the one described for the Glove interface, reading the input from the sensors, and fitting it in certain ranges depending on the value received. The most complicated part of this interface is calibrating the sensors for each unique user.

Since the PodControl is to be used on top of the insole of the shoe, forces will be applied at all moments, and these forces will be based dependant on the weight of the person, and waking habits. The *main* function in the following block of code illustrates in an oversimplified manner the way the control is going to calibrate the sensors by first calling the respective functions. Once operating, the device will determine the current state of the user by comparing the current values of the forces on the sensors, and the ones previously found for each state. If the user is at a fixed position, the hand can be operated, and will send the proper signal to the *hand.* When calibrating the sensors for the first time, some statistical analysis will be done to correctly find the ranges of values for each state. Multiple reading will be used for this analysis.

```
static int main(){

    // variables needed to calibrate the control
    double* switchForce[2], release[2]; grab[2];

    int currentMode = 0;

    // call only the first time the control is used
    motionMode(*switchForce[], *release[]; *grab[], sensor2, sensor3, sensor1);
    standbyMode(*switchForce[], *release[]; *grab[], sensor2, sensor3, sensor1);

    currentMode = returnMode(*switchForce[], *release[]; *grab[], sensor2, sensor3, sensor1);

    if (currentMode == 0){
        checkSwithSensor();

        // each of these functions sends the proper signal to the motors
        // if required
        checkGrabSensor();
        checkReleaseSensor();
    }


}
```

```
// read in the forces on the sensors when the object is in motion
// In the future an accelerometer might be used to detect motion
void motionMode(double *switchForce[], double *release[]; double *grab[], double sensor2, double sensor3,
                double sensor1){
    *switchForce[1] =  sensor1;
    *release[1] = sensor2;
    *grab1[1] = sensor3;
}


void standbyMode(double *switchForce[], double *release[]; double *grab[], double sensor2, double sensor3,
                double sensor1){
    *switchForce[0] =  sensor1;
    *release[0] = sensor2;
    *grab[0] = sensor3;
}


// 0 = standby mode, 1 = object in motion
int returnMode(double *switchForce[], double *release[]; double *grab[], double sensor2, double sensor3,
               double sensor1){
    if (sensor1 <= switchForce[1] && sensor2 <= release[1] && sensor3 <= grab[1])
        return 0;
    else
        return 1;
}
```

4.6 Remote Control Panel

The remote control panel is another interface system that could be incorporated to the Luke's Hand arsenal to make the life easier to the user. This remote control panel will be only a direct link to microcontroller; which is connected with the servos of the robotic hand and arm. If any button or switch is pressed on the remote control, then it will signal the micro controller to send the square gave and pulse necessary to the servo to move in the direction desired in order to open and close the hand.

The buttons on the robotic hand will consist of five pares of FSR touch sensor. Two FSR sensors will be assigned for each finger. The button will consist on two parts since the hand must be able to open and also close the fingers. The FSR sensors will be places one by the other and they will be directly linked with the microcontroller that sends the signal to the servos. The way the touch sensors will work is very simple. One sensor will have the function to signal the servo to close the hand. This FSR sensor will tell the microcontroller to send a 2 ms pulse in order to close the hand. There is two positions of the servo that will be set as boundaries; which are zero degrees and one eighty degrees depending on the chosen servo. Zero degrees will be when the hand is opening and one eighty when the hand is closed. The FSR

touch sensor will give the direct signal to the microcontroller to either send a one millisecond to open the hand or a two millisecond to close it. The remote control will also have a switch that will be linked to the torque motor to move the arm up and down. This switch is located right next to the thumb button in the right part of the remote. This switch is only a on / off switch integrated to accomplished this simple task.

The sensors that are placed in the robotic hand will be turned off, since the user is controlling the robotic hand directly. When the hand is going to grab an object; it will be fully relaying on the users intuition and perception of the object that the user sees. Figures 4.6.1 and 4.6.2 show the design of the remote control panel and how the buttons are going to be arranged. The remote control panel will look like a computer mouse. This shape was chosen so the users hand can rest on it comfortably and also able to reach all the buttons without any problem. The length of the remote control panel will be around eleven centimeters, which allows plenty of space for any size hand to use it. The width of the control will be seven centimeters and the height will reach a maximum of three centimeters. Figure 4.6.1 shows the top view of the control with the four fingers and their respective buttons.

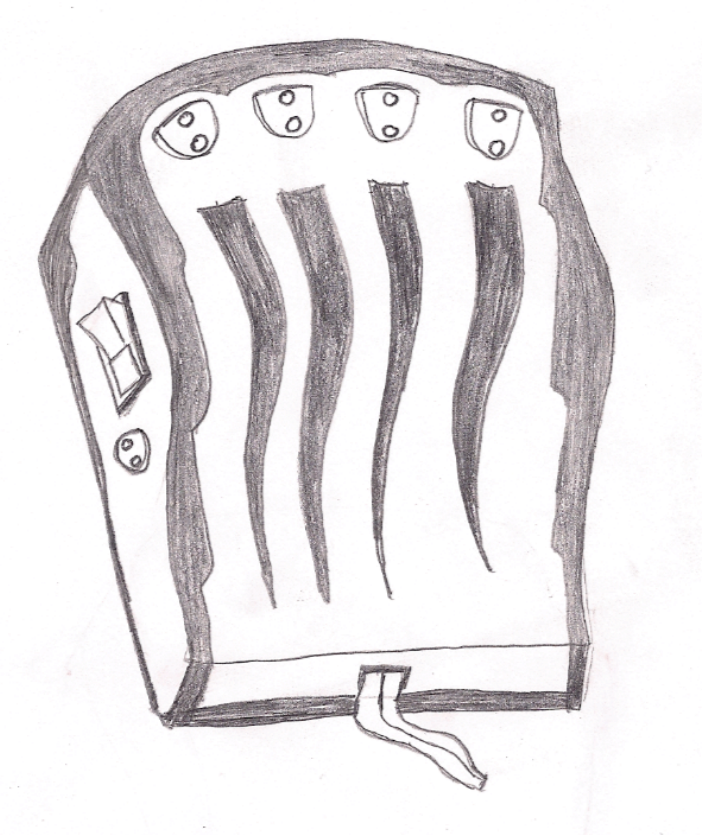


Figure 4.6.1 Upper view Remote Control Panel

The figure 4.6.2 shows the side view of the remote control for the robotic application, which shows the button that controls the thumb in addition to the on/off switch of the torque motor.

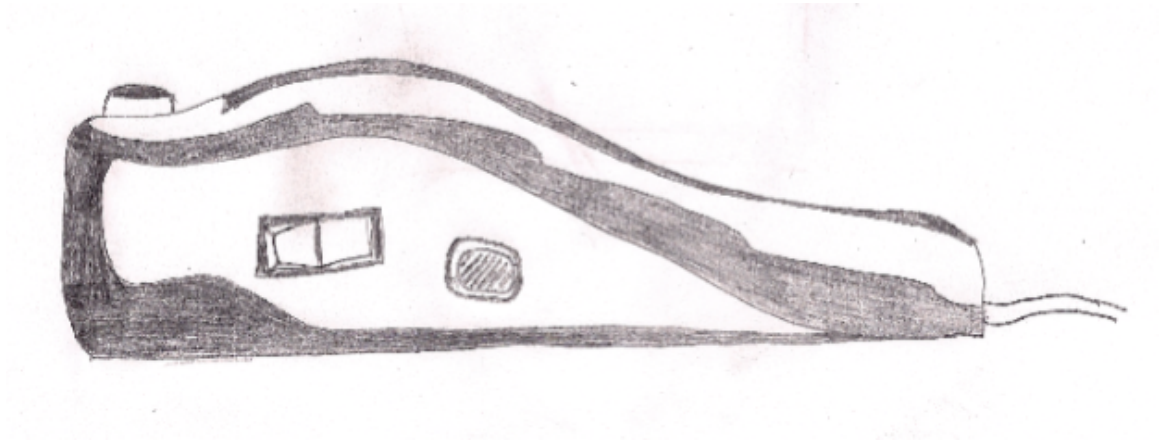


Figure 4.6.2 Side view Remote Control Panel

4.7 Computer Interface

This interface is quite complex, and requires much more coding that any of the others. For these reason, the algorithms to be used won't be explained in much details, since the many new factors will be taken into consideration as the design process advances. Rather, the desired functionality will be explained in as much detailed as currently possible.

Initially, the user must be able to control the hand directly from the computer, allowing him/her to activate/deactivate certain features that will enhance the functionality of the hand. Initially it will allow the user to select an object from a predefined list of known objects, passing parameters such as width, weight, and material to the hand. For example if the object is bottle of water, the hand will know that the IR proximity sensors will not work because of the translucent characteristics of the material, and therefore will deactivate the use of this sensor. It will also know beforehand the force that needs to be applied to the bottle to avoid slipping. It should also allow the user select which sensors to use independently, since for certain materials the microphone could be a better option to detect slipping that the vibration or optical sensors. The program will also let the user close/open the hand manually through the use of sliders, thus letting him/her control the hand freely.

It would have a display area, where it will show at all moments the status of each sensor, control interface, and motor, to easily monitor the behavior and state of the hand. This will be especially useful when testing the hand, since it will show the any discrepancies between actions sent to the hand, and the one actually performed. A control selection panel will also be added, this so that all external controls can be tested using the display features of the computer program. An extra panel will be used to let the user select certain predefined gestures, presenting these options as a list to the user. An initial sketch of the graphical user interface for this interface can be seen in the Figure 4.7.

Figure 4.7 Sketch of the computer interface

CHAPTER 5 Prototyping and Testing

5.1 Prototypes

When building the robotic hand, a lot of mechanical problems started come up. The pieces that form the fingers have to be built with extremely accuracy. Since the work team does not have the access to a laboratory or mechanical shop with sharp machines, which would assist the drilling of holes and overall the manufacturing of required pieces, a top of the line work cannot be done. This Group utilizes regular construction hardware to build the hand. The holes in the fingers need to be drilled with extremely care and accuracy since they are going to provide the pad of the joints. If one of the holes is slightly miss drilled, the contraction of the finger will be corked and the fingers will touch each other when closing the hand's fingers producing an undesired outcome.

The overall the design and construction of the first prototype was very well planned and executed, but when it comes to the evaluation of the mechanical performance of pieces working together, the prototype did not reach satisfactory results that would allow the team to continue further implementation of more complex functionalities. Because of the tools used in the process of building the hand, some holes where miss drilled and the fingers did not have a straight and consistent flex motion. For a good grapple of the object the robotic hand needs a very good flexing of the fingers. Additionally, the friction caused by metal pieces generates high margins of error, which would make almost impossible to give measurable and reproducible results. For this various problems the group decided to try a radically different approach. The new approach is a toy plastic hand, which is economically accessible, and easy to find in the market. Such hand found fits most of the specifications that Luke's hand needed and there has been planned several tests for the next academic semester in order to decide if the use of that plastic hand would be the most appropriate choice.

Visually, the fingers of the plastic hand are flexible with high precision and reproducible results when the hand is not loaded with any object or lightweight objects, and could contract the whole hand in a straight fashion. Apparently, the fingers can be manipulated and the sensors chosen for the main functions could be integrated for the robotic application. It is very light and it is able to grab various types of objects. The only impediment found was that the toy hand could not grab big and heavy objects due to the materials from which the plastic hand was built. If this toy hand is used for the final project, the size of the objects will be restricted by new specifications that would need to be redefined. Figure 5.1 shows the difference between the robotics hand built by team and the toy hand that might be used in the final project.

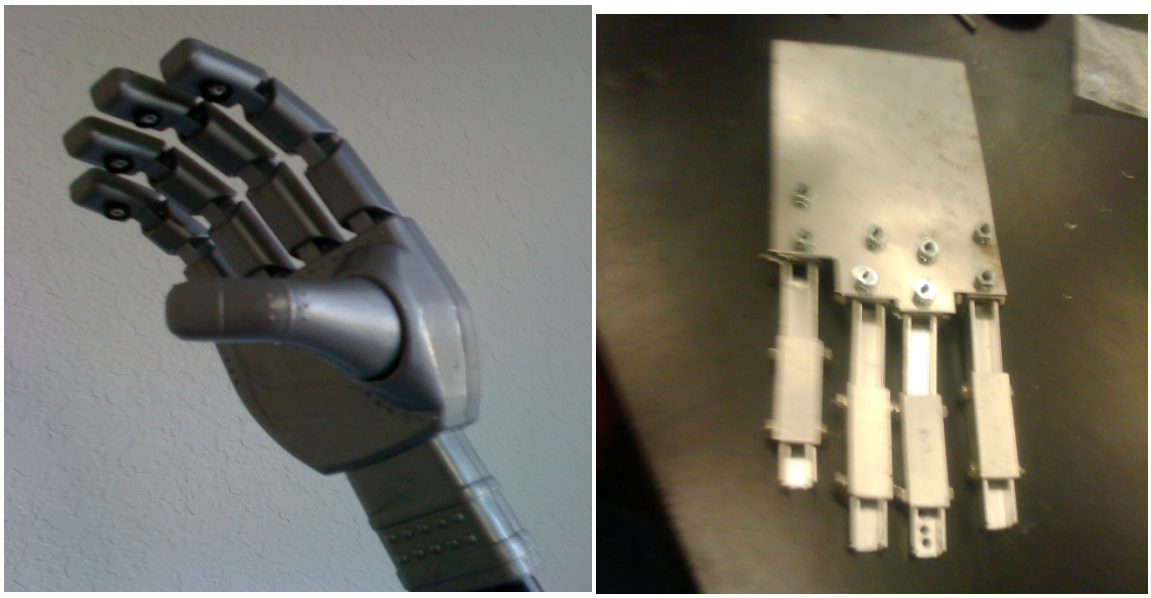


Figure 5.1. Difference between Toy hand and Groups hand

5.2 Testing

The validation of Luke's hand design is one of the primary steps during the process of the demonstrating its proper functionality. In order to corroborate the claimed characteristics and functions of this application there are going to be constructed different test cases where the robotic hand should be performing predetermine activities with reproducible expected results. Taking in account the specification of the design, the predefined procedures are going to be described in the following subtopics of this section.

5.2.1 Different procedures for testing the Robotic Hand

5.2.1.1 Count operation

Like any average normal human hand, the robotic hand application is going to be able to count decimal numbers from one to five. To procedure would be done by a computer program that would indicate the microcontroller to give the appropriate signal to the servomotors in order to move the phalanges of the fingers in such a way that the fingers would make the symbol of the cardinal number from one to five. The count operation is going to be executed in a procedural manner going in a loop counting from one to five and exiting the loop. Such loop could be repeated by

the program several times and exit the loop after executing the subroutine or it also could well be the execution of the count operation by a single time.

Since the fingers are going to be at home position when the fingers are stretched, then to flag the number one, only one finger would stay stretched whereas the other fingers would be contracted toward the palm of the hand. Because repetitively is fundamental for validating a procedure, each one of the numbers are going to be match from now on with one specific set of fingers, so that, when the application is instructed to show the representation of number one, always the same finger would be standing up. A relation of the number with the set of fingers is going to be shown in the following table.

Number to be represented	Set of fingers to represent the requested number
One	Index finger
Two	Index and middle fingers
Three	Index, middle, and ring fingers
Four	Index, middle, ring, and little fingers
Five	All the fingers standing up

Table 5.2.1.1: Representation of the numbers from one to five using the fingers of the robotic hand

As shown in the above table, the same would be valid for the rest of the numbers, where they would be represented by the same combination of fingers standing up, when those numbers would be requested by the program or by the user, depending on the mode of operation in which the application would be. Also inside the count operation, the hand would be able to respond to the instruction of representing a single number without going through the whole loop representing all the numbers involved in the count operation. Next is figure 5.2.1.1, which illustrates the process of representing the count operation, more specifically; the figure shows the robotic hand application symbolizing the number four.

Figure 5.2.1.1: Above is represented the number four with the appropriate fingers according to the table above.

5.2.1.2 Grabbing procedure

The process of grabbing an object has a lot more involved than the former operation. The members of the group have decide to make the task of grabbing objects with the Luke's hand application a more challenging operation than just giving the order to the microcontroller to retract the fingers. Instead, it was considered and is intended to be implemented some artificial intelligence, which would be accomplished by incorporating several sensors to read external stimulus as inputs signals to trigger the portion of the computer code to act and react accordantly to a specific situation.

The robotic hand as well as any human hand has natural limitations due to the size or the shape of an object intended to be grabbed. Of course, the limitations of the robotic hand are much greater than the ones for the human hand because of its natural versatility and ingenious design by the Mother Nature. Nevertheless, Luke's hand would have intelligence of its own to grab objects that are inside the range of the predetermined specifications.

Perhaps the most natural limitations for human and robotic hands alike are the size and shape of the object intended to be grabbed. Thus, the size of the object needs to be proportionally smaller to the Luke's hand size. As mention previously in chapter

two, the robotic hand was design to be of the size of a human hand; more specifically, its measurements were taken from one of the members of the group in order to have more realistic solutions with the application to remotely emulate a real human hand. Next are defined the types of objects that can be grabbed by the robotic hand application. Figure 5.2.1.2 helps to visualize the regular grabbing operation.

Figure 5.2.1.2: Illustration of the expected regular grabbing operation.

Any targeted object to be captured must weight less than three pounds, but with enough volume to be incorporated and pressed with the fingers toward the palm of Luke's hand. The shape of the object could be regular or irregular as long as the portion from where the object would be grabbed is about homogeneous and with enough volume to be incorporated and pressed with the fingers toward the palm of the robotic hand. The maximum diameter of the object in question should be about ten to twelve centimeters and the minimum about two to three centimeters. Such variation of the diameter is related to the material of the object to be captured, where if the nature of the object is slippery, then the margin of weight and diameter would be in its inner boundaries.

A better clarification of this process can be accomplished by looking into the Diagram 4.1.1. The Finite state machine is meant to control the sensing activity, which is located in chapter 4, and presents the process of interaction between the involved sensors for the grabbing functionality and an object.

5.2.1.2 Fine grabbing procedure

Besides the accuracy of its locomotion, human hands are provided with sharp senses to successfully grab and hold objects as thin as a hair and even thinner. One of the elements of the equation that enables the human hand to hold such objects is the ability of the flesh to be compressed with some pressure and vary its shape without compromising the integrity of its own composition. For Luke's hand, the design contemplates the incorporation of silicon or rubber material at the point of contact between the hand and the object to allow better grip and better stability at the time of holding a captured object.

Fine grabbing procedure is a functionality that has been defined for smaller elements, which cannot be successfully capture by the regular grabbing function. The operation of this procedure would be partially controlled by the sensors, which would decide the mechanical procedures with slightly modifications, where the fingers to be involved here would be only three instead of five; thumb, index and middle fingers. The decision of following this procedure instead of the regular grabbing function would be made manually by the user of the robotic hand.

Types of objects that can be grabbed by the robotic hand application are defined as follow: First, any targeted object to be captured must weight less than one pound, but with enough volume to be pressed with the thumb, index, and middle fingers of Luke's hand. The shape of the object could be regular or irregular as long as the portion from where the object would be grabbed is about homogeneous and with enough volume to be incorporated and held with the pressure and space proportioned by the mentioned fingers of the robotic hand. The maximum diameter of the object in question should be about two centimeters and the minimum about one half of a centimeter. Such variation of the diameter is related to the material of the object to be captured, where if the nature of the object is slippery, then the diameter would be required to be about 30% higher than the minimum value of the specification. The minimum weight of such elements is not defined as long as it has the required volume to be caught.

Importance of the thumb in the human hand is much greater than any other finger due to the versatility of its movements, the location in the hand, and the extra strength that it possesses. Thanks to the approach of Luke's hand design, some realism is incorporated at the time of using the thumb to perform certain functions that could only be accomplished by the use of several fingers. For instance, the fine grabbing functionality can be implemented with a little twist by using the thumb by itself pressuring the object against the palm of the robotic hand. Figure 5.3.1.4 illustrates the testing process of fine grabbing procedure.

Figure 5.2.1.3: Luke's hand executing Fine grabbing procedure.

5.2.1.3 Grabbing a large object procedure

Following the bases of balance, the human hand is constantly evaluating the position of an object, which has been taken in order to make the adequate corrections. This happens and is more evident especially with object that are larger or heavier than expected. The robotic hand application would be using a set of sensors to perform the necessary corrections of the hands position at the time of holding an object, which is greater than the upper limit of size and shape specifications of the regular grabbing functionality, so that the hand can balance the captured object.

Again, the sensing process at the palm of Luke's hand dominates the functionality of this procedure. The decision to perform this special operation would be controlled by the user of the robotic hand application because the hand application neither would have the necessary artificial intelligence to visually evaluate the size of the object nor would be able to evaluate the weight of that object before knowing which approach to take, so that the task can be executed accurately in the first try. Next is presented a graphical visualization of Luke's hand handling a large object by twisting the wrist 90 degrees and balancing the object with the assistance of the dedicated sensors implanted in the robotic application.

Figure 5.2.1.4: Graphical representation of Luke's hand executing testing procedure for grabbing a large object.

An important factor in the design of Luke's hand application is the mechanical implementation of the wrist. Because the wrist makes rotational movement, it is going to assist the actual hand to keep balanced the object if it is displaced toward one of the sides of the hand due to the inertia applied by the body's mass. This functionality is better explained with Diagram 4.1.2: 'Definition of the procedure to control the sensing process for grabbing special objects,' located in chapter 4, where the design of the hand is discussed.

Any targeted object to be captured must weigh less than five pounds and more than three pounds, appropriate volume to incorporate and press the object with the fingers toward the palm of Luke's hand. The shape of the object could be regular or irregular as long as the portion from where the object would be grabbed is about homogeneous and with enough volume to be incorporated and pressed with the fingers toward the palm of the robotic hand. The maximum diameter of the object in question should be about fifteen centimeters and the minimum about twelve to ten centimeters. Such variation of the diameter is related to the material of the object to

be captured, where if the nature of the object were flexible or compressible, then the margin of weight and diameter would be in its upper boundaries.

5.2.1.4 Lifting procedure

Human hands are not just limited to grab and hold and object, but instead, with the assistance of the arm, it is able to lift object and relocate them to a different relative position. The process of lifting an object by the robotic hand is a complementary operation that is implemented in cases where the user needs to relocate an object from its original position to a higher or lower relative position. To test the lifting procedure, an object is going to be given to the robotic application, and if must be lifted and released at higher position from the original. Since the elbow will not be able to rotate at its position, the object will need to be received by one of the member of the group who would be doing the demonstration of Luke's hand.

5.2.2 Results of testing

The results of testing all functionalities of Luke's hand application are expected to be within a tolerable margin of error that is difficult to quantify at this early stage of the application, but more details would be provided as soon as they would become available to the group.

5.2.3 Analysis of Results

This part of the process is pending for now due to the absence of a physical and complete robotic hand implementation.

CHAPTER 6 Budget and Financing

6.1 Budget

In order to understand better the expenditures of the project, the budget section will be divided in two sections: Senior Design I and Senior Design II expenses.

6.1.1 Senior Design I

A comprehensive list of expenses incurred during this first stage is shown in the Table 6.1. This table shows that percentage on the money was spent on miscellaneous (printing), tools, and accessories, which came represent about 15% of the total budget.

	Qty.	$ Per Unit	Free	D&R	Per Unit
Printing	1	80.00		80.00	0
Hacksaw	1	4.96		4.96	0
ANGLE1/2X96	1	6.64		6.64	0
3/8X96CCHNN	1	7.92		7.92	0
SCREWS	1	3.97		3.97	0
SHEET Metal	1	6.59		6.59	0
Threaded Rod	2	2.67		5.34	0
Washer	6	0.40		2.40	0
Alu Extrusions	1	2.53		2.53	0
ACME Misc	1	10.00		10.00	0
ACME Misc	1	4.50		4.50	0
Master Lock	1	5.98		5.98	0
Total				140.83	
Total + Tax				144.9	

Table 6.1 Senior Design I expenses

6.1.2 Senior Design II

The expenses incurred during this stage represented around 85% of the total, which is congruent with that happened since most of the design, implementation, and testing took place during this period time. Around $250 dollars were spent on tools, $200 on development boards, $65 on PCB boards, and the rest was spent on different components. Table 6.2 provides a breakdown of all of these expenses, and shows what was allocated to R&D, what was obtained free (if any), and what was actually spent in the final version of Luke's Hand.

	Qty.	$ Per Unit	Free	D&R	Per Unit
Plastic Hand	1	15.43			15.43
4 DIP Switch	2	3.49		3.49	3.49
8 DIP Socket	2	2.49		2.49	2.49
40 DIP Socket	2	2.49		2.49	2.49
16 DIP Socket	2	2.49		2.49	2.49
PING Sonar	1	29.99		22.49	29,99
PropPlug	3	24.99		49.98	24.99
VEX Parts	N/A	25.00	25.00		
SKYCRAFT		12.85		12.85	
Darth Vader H	1	32.03		32.03	
0.2" FSR	5	6.00		3.00	
Flex Sensors	5	12.95		3x12.95	4x12.95
7.5VDC 1A PS	1	10.99		10.99	
Cont Servo	1	12.99		12.99	
32K EEPROM	2	1.50		1.50	1.50
XTAL HC49US	1	1.10		1.10	
Propeller Kit	1	99.99		99.99	
Std Servo	8	23.37		4x23.37	4x23.37
PK1 ALK 9V	2	3.99		6.98	
Elec Tape	1	2.99		2.99	
75' #20 Solid	1	7.99		7.99	
IC VREF	2	1.60		3.20	
MCP3202	4	3.04		3x3.04	3.04
MCP3208	2	4.18		8.36	
Switch DIP	5	0.69		3.45	
CONN Header	10	0.306		3.06	
Rivets	1	5.47		5.47	
Rivet Tool	1	19.87		19.97	
Dremel Attach	1	29.98		29.98	
1/2X96CHAN	1	9.39		9.39	
3/8X96CCHN	1	7.92		7.92	
Plastic BAGGD	2	0.98		1.96	
PCB Boards	3	22.33		2x22.33	22.33
Total Shipping				35.99	
Total Taxes				8.98	
				837.61	253.39

Table 6.2 Senior Design II expenses

6.2 Financing

The total spent on the project came to about $1100, which was financed in its totality by the members of the group, each contributing the same amount of money, approximately $275. About 75% of the money spent went into the R&D category, where the mechanical design represented most of these expenditures, and the remaining expenditures in the R&D consisted on buying spare parts such as replacement servos, sensors, batteries, etc. The hand came to cost of around $250, thus representing 25% of the total budget.

Description	Cost	R&D	Per Unit
Tools/Supplies	$250	50>	--
Servos	25*8 = $200	8	4
Flex Sensors	13*10 = $130	10	4
FSR Sensor	7*7 =$49	7	0
Sonar Sensor	$32	2	1
Dev. Boards	$120	2	N/A
MCU/ADC/ROM	$80	8	3
PCB/Components	$80	3, 100>	1, >20
Other	$150	--	
Total	$1091	$791	$300

CHAPTER 7 Design Summary

The project and challenge in recreating as much as possible the human hand by the use of robotics was chosen for this senior design project. Luke's hand will have a different variety of requirements and also limitations. After a close observation to the human hand, the requirements and specific functions that can be recreated by the robotic hand was carefully chosen.

When a human hand is going to grab an object some important tasks have to be taken into consideration. A human hand knows that an object is in front of it because of visual feedback from the person. In the case of the Luke's hand there is no person to visually guide and detect if an object is in front of the robotic hand. To recreate this first step, the robotic hand is going to need some kind of sensing that will allow it to perceive objects that are close by so they can be grabbed. A proximity IR sensor will be integrated to the robotic arm to detect objects that are close by to the palm of the hand. This proximity sensor will only detect objects that are inside the palm of the hand, which are the only objects that the hand will be taking importance.

Another important and crucial characteristic function that was observed in a human hand was the ability of sensing touch. The human hand is able to perceive different texture, softness and an incredible variety of surfaces, temperatures and materials. When the human hand is going to grab an object, it senses the shape, texture, and temperature of the object before grabbing it. For this project the ability of sensing when the robotic hand is in contact with an object before grabbing will be a great feed back to distinguish which finger is in contact with the object and which one is not. A FSR touch sensor will be integrated in the tip of each finger in the robotic hand. These sensors can sense how much force is applied to an object and if they are in contact with any type of object.

The fingers will start closing right after the proximity sensor detects an object in the range of grabbing. When the finger flex and the touch sensor mounted on the tip touches the object detected, the finger will stop and wait until all the other fingers go through the same procedure. After all the fingers are touching the object detected they will apply a minimum force to ensure the grappling.

Another important function notice that the human hand performs and one of the most important ones is the ability of sensing slipping of an object when it is falling. When a human feels that an object is falling, the brain sends a signal to the hand to grab harder to stop slipping. This same idea has to be applied to the Luke's hand to prevent the drop of the object intended to grab. For this function many sensors were researched to come up with the best solution. In the section of sensors a more detailed explanation of each sensor is given, but for the overall function an optical sensor is preferred for the accomplishment of the task. The optical sensor will be integrated directly in the palm of the hand. , The perfect position to detect any object grabbed is the palm, since all of the objects will be supported on in when

grabbed. The optical sensor is the favorite because it can sense any type of horizontal and vertical displacement. This means that it does not matter how the object is falling, it will be detected before it does and the pressure applied to the object can be corrected on time.

After the hand grabs the object the arm needs to lift it to ensure that the object was grabbed correctly. The arm will be moved up and down with the help of a torque motor. The elbow mechanism, which is the system in charge of the arm movement, will also include gears between the axis of the arm and the torque motor. These gears will help the design to either increase the torque of the weight of the arm is greater than expected or the speed of the system if it is too slow.

One of the most important issues of the project is how Luke's hand is going to communicate with the user. The robotic hand will have three different links will the user; which consist on a remote control panel, a remote control glove and a remote control footpad. The remote control panel is more like a computer mouse with the function of controlling the individual movement of the fingers and arm. The user will have five buttons for the movement of the fingers and a switch for the arm. The buttons that control the fingers consist on two FSR touch sensors. One sensor sends the signal to contract the fingers to close the hand while the other sends the signal to open the hand. The buttons in the panel are going to be connected to the microcontroller. The switch that controls the movement of the arm will control the motion of the DC torque motor. The switch and the thumb button are going to be located on the side of the control panel while the others buttons will be on the upper part.

If the Luke's hand would ever help someone that does not have a limb the footpad link will be the perfect for them. Since the person that is going to be using the Luke's hand won't have on arm and it is trying to live life as normal as possible, a more discrete control had to be created. The footpad will actually have two pads, one for each foot. There is a couple of ways to control the system and a more detailed explanation is given in the footpad mechanism section. The most accepted is way of fabricating this footpad is having two FSR touch sensors in each foot. When a person is walking the forced applied in each foot will vary in every step. From step to step one foot will switch from resisting all the weight of the body to none. If the FSR sensor integrated in both feet registers a similar amount of force that means that the person is standing, therefore the system will activate stage two. When the second phase activates, this will turn on the other FSR touch sensors located in the right footpad. One sensor will be integrated below the heel of the foot and the other sensor will be below the big toe of the same foot. The big toe sensor will make the robot hand to close while the heal sensor will open the hand. Using this mechanism the person can walk with no problem and use the Luke's hand when needed. This interface can have various way of accomplishment as mention before, but this is the more accepted one of the group.

Another way to control the Robotic hand is by using a remote control glove. The main function of the remote control glove is to send a signal to the robotic hand,

when a person is wearing it, to mimic all the movements that the person is making with the hand. Like the footpad, there are many ways to accomplish this construction of the glove and they are analyzed in the respective section. The most accepted for idea for the construction is using a variable switch connected to the back of each finger to signal the servos to move alike with the hand. Each finger will have a string attached to the switch and a rubber band or elastic in the lower part of the switch. The string will pull the switch to make the servos to close the fingers and when the user opens the fingers the elastic will return the switch to the original place and signal the servos to return also. The glove will have five switched to make a direct link with ach servo to create the individual movement of the fingers. The section about the glove design will explain more about this method.

Luke's hand will also have the ability to perform task by itself so the help of all the many sensors mentioned before and a Propeller Microcontroller will accomplish the task. The microcontroller will have 32 input/ output ports with 256 ROM and 512 RAM. It will be programmed in spin, which is very similar to C ++.

Many different difficulties will make harder to accomplish the task and also to have a top of the line project. The tallest wall this project faces is budget. Group #1 does not have founding of any kind, so all the materials necessary for the construction the robotic hand have to be founded by the group. Time is also another issue because there is not enough time in one semester to do the appropriate research for the project and also to exercise trial and error to all the equipment to improve the system and design. The facility where the construction of the robotic hand was held is also not in favor. The tools used for the construction are not the right ones to build a perfect hand. The design of the Luke's hand has a lot of mechanics. The fact that group #1 is made of two electrical and two computer engineers, the mechanics of the robotic hand was not 100 % perfect. Figure 7.1 shows the layout of the whole prototype of Luke's hand, which consist of robotic hand and arm. The figure shows where the sensors, servos, motors and mechanism are going to be located in the robot.

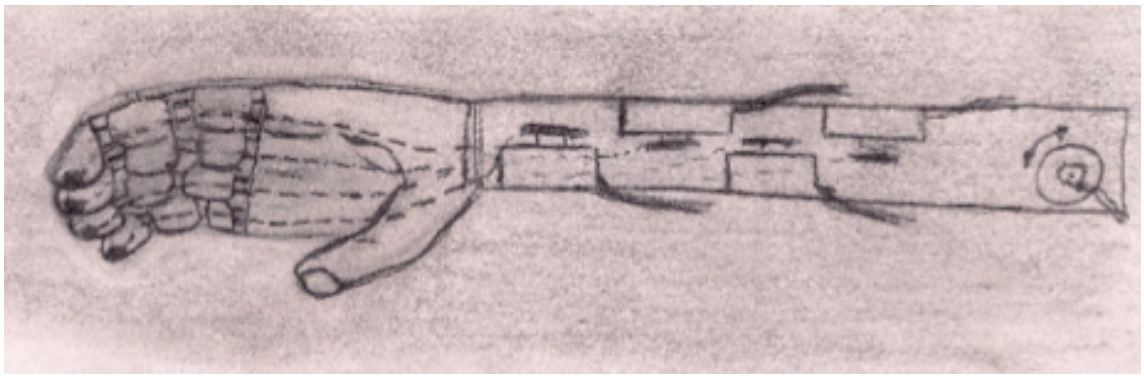


Figure 7.1 Luke's Hand

Bibliography

- Jame Cox, (2002). Fundamentals of Linear Electronics Integrated and Discrete, 2nd Edition. USA: Delmar Thomson Learning.
- Bernard Grob & Mitchel E. Schultz (2003). Basic Electronics, 9th Edition. USA: Glencoe/McGraw Hill.
- Gordon Mc Comb & Myke Preoko, (2006). Robot Builder's Bonanza, 3rd Edition. USA: McGraw-Hill Companies.
- Ben-Zion Sandler, (1991). Robotics: Designing the Mechanism for Automated Machinery. USA: Prentice-Hall, Inc.
- ExpressPCB, (2008). Tips for Designing PCBs. Retrieved April 24, 2009, from ExpressPCB. Web site: http://www.expresspcb.com/ExpressPCBHtm/Tips.htm
- Trossen Robotics. Web site: http://www.trossenrobotics.com/
- Rabbit®. Dynamic C® An Introduction to ZigBee. Retrieved 2008.
- Parallax. Web site: http://www.parallax.com/
- Texas Instruments. Web site: http://www.ti.com/
- Microchips. Website: http://www.microchips.com/
- Zilog. Website: http://www.zilog.com/
- FreeScale. Website: http://www.freescale.com/
- Digi. Website: http://www.digi.com/
- Atmel. Website: http://www.atmel.com/
- CEL. Website: http://www.cel.com
- Mariana Ruiz Villarreal. Scheme Human Hand Bones. Retrieved January 6, 2007.

Appendix A Copyright Reprinting Permits

Hello Mr. Oleas,

I hereby officially grant you permission to use any image or excerpt from the Propeller P8X32A Datasheet for the stated purpose of your senior design paper. Please include "Photo courtesy of Parallax Inc." or "(C) Parallax Inc.; used with permission" or any similar attribution format required by your senior paper guidelines.

I wish you well with your senior project! If you need anything else along these lines, please let me know.

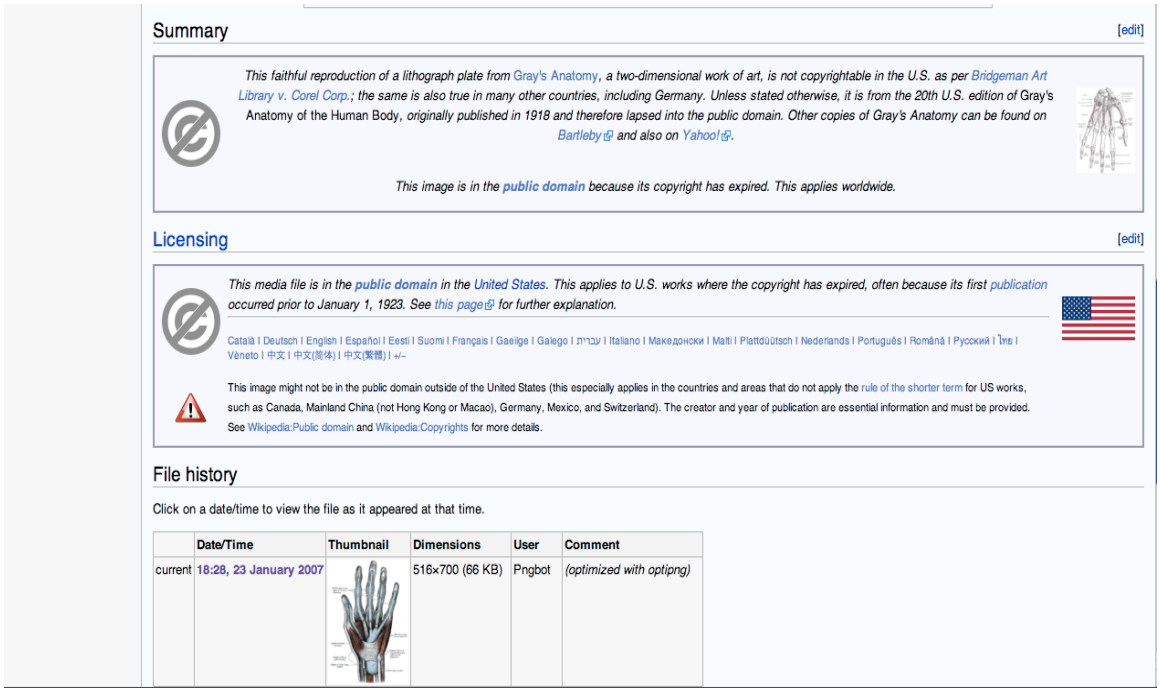
Best regards,

Stephanie Lindsay
Technical Editor
Parallax Inc.

www.parallax.com

Permission for the use of Figure2.3.2 is granted in the following link, additionally, the following screen shot shows the permission to use the image.

http://en.wikipedia.org/wiki/File:Gray423.png



Summary [edit]

This faithful reproduction of a lithograph plate from Gray's Anatomy, a two-dimensional work of art, is not copyrightable in the U.S. as per Bridgeman Art Library v. Corel Corp.; the same is also true in many other countries, including Germany. Unless stated otherwise, it is from the 20th U.S. edition of Gray's Anatomy of the Human Body, *originally published in 1918 and therefore lapsed into the public domain. Other copies of Gray's Anatomy can be found on Bartleby and also on Yahoo!.*

*This image is in the **public domain** because its copyright has expired. This applies worldwide.*

Licensing [edit]

*This media file is in the **public domain** in the United States. This applies to U.S. works where the copyright has expired, often because its first publication occurred prior to January 1, 1923. See this page for further explanation.*

Català | Deutsch | English | Español | Eesti | Suomi | Français | Gaeilge | Galego | עברית | Italiano | Македонски | Malti | Plattdüütsch | Nederlands | Português | Română | Русский | ไทย | Vèneto | 中文 | 中文(简体) | 中文(繁體) | +/–

This image might not be in the public domain outside of the United States (this especially applies in the countries and areas that do not apply the rule of the shorter term for US works, such as Canada, Mainland China (not Hong Kong or Macao), Germany, Mexico, and Switzerland). The creator and year of publication are essential information and must be provided. See Wikipedia:Public domain and Wikipedia:Copyrights for more details.

File history

Click on a date/time to view the file as it appeared at that time.

	Date/Time	Thumbnail	Dimensions	User	Comment
current	18:28, 23 January 2007		516×700 (66 KB)	Pngbot	*(optimized with optipng)*