INTELLIGENT PROGRAMMABLE PROSTHETIC ARM

Senior Design I
Group 5
Sponsored by: CECS Alumni Chapter

MAY 1, 2015

MATTHEW BALD, IVETTE CARRERAS, ANDREW MENDEZ

Table of Contents

List of Figures

List of Tables

1 Introduction

This document comprises the research, design, test, and finance documentation for the University of Central Florida's Computer Engineering curriculum's Senior Design I. Group 5 chose to advance the current technology available in low cost prosthetic arms. This document contains all of the research, design, and procedures needed in order to develop the Intelligent Programmable Prosthetic Hand.

1.1 Narrative

This section introduces the general description of the project as well as the motivation for the creation of the Intelligent Programmable Prosthetic Arm (IPPA).

1.1.1 Executive Summary

One considerable obstacle for people to acquire a major or minor upper limb prosthetic is their expensive cost. An industry quality upper limb prosthetic costs tens of thousands of dollars as they require many sessions for adjustments, and use expensive materials. This drastically limits the ability of affected children and adults all over the world to have a somewhat normal life. 3D printed arms are part of a current trend to provide a solution at a more affordable price. However, most of those limbs are very limited in their functionality. Available designs can be manually moved to a desired position, and some others allow you to open/close your hand using the electrical potential generated by muscle cells in the amputee's arm.

We propose a 3D printed prosthetic arm with off the shelf electronic devices that incorporates multiple features such as grasping, pointing and other natural gestures that are standard in expensive prosthetics. This project utilizes the advantages of 3D printing to reduce the cost of the prosthetic to less than one thousand dollars. The Intelligent Programmable Prosthetic Arm (IPPA) contains multiple sensors that allows it to perform automatic grasping of objects, gentle handshakes, and other natural gestures in addition to gestures triggered by the electrical potential generated by muscle cells of the person.

One of the problems that drives the cost of prosthetics up, is the complexity of a human hand and the wide variety of applications that this tool can be used for. It is difficult to design and program a single prosthetic that satisfies each individual. In order to solve this problem, the Intelligent Programmable Prosthetic Arm also includes a mobile application that allows the amputee to change the features in the arm from an available list or create their own and unique arm movement or hold patterns.

Amputees face many obstacles when adjusting to a prosthetic for the first time; a big one is the extensive learning curve and adaption to the prosthetic. Especially difficult is the ability to control their electromagnetic signals in the arm and learning to signal the prosthetic. Therefore, the IPPA project introduces the use of voice commands in order to support the full functionality of the prosthetic through this early adaptation stages. This allows unexperienced users to start using their prosthetic arm right away while they learn to control it with the electrical potential generated by muscle cells in their arm.

The IPPA project's goal is to provide a fully functional low cost prosthetic, as well as providing the correct support for those starting to learn how to send electromagnetic signals to their new limb. This project is targeted towards people who are missing a hand, wrist, and part of their forearm; not a full arm.

1.1.2 Motivation

This project was mainly motivated from the current statistics of amputees that require upper limb prosthetics, the challenges of designing an upper limb prosthetic, and the current work being done to improve the current state of affordable, functional prosthetic arms.

In 2005, an estimated 41,000 people were documented to suffer a major upper limb amputation, while 500,000 people were documented to have suffered of a minor upper limb amputation[12] These statistics do not account for people across the globe who have suffered upper-limb amputation from regional conflicts and wars. In addition to number of amputees who need upper limb prosthetics, there are many challenges that are involved in designing an upper limb prosthetic. A major problem is developing a prosthetic to maneuver and complete tasks similar to a human hand. Our natural hands preform a wide range of task-specific grasps ranging from complex and delicate, to strong, and forceful [16]. Another important challenge is incorporating natural hand gestures that people use in social settings.

Industry quality upper limb prosthetics that enable people to utilize basic grasping actions costs thousands of dollars to buy and are limited to adult amputees that have the insurance plan to cover it. An industry quality upper limb prosthetic is inaccessible to most adults because their insurance does not cover the fee. In the case of children it becomes a greatly expensive commodity because parents must acquire different sized prosthetics as children grow.

The Daniel project is a current project that served as motivation to pursue this project on developing a capable but affordable upper limb prosthetic. The Daniel project was an initiative from the company Not Impossible Labs, where they developed a training facility and laboratory to utilize 3D printing technology to develop 3D printed prosthetics for victims in a Sudan refugee camp.[6] The Daniel project illustrates how 3D printing technology is revolutionizing upper limb prosthetic technology

1.2 Project Description

Our goal is to develop a low cost, upper limb prosthetic that enables the amputee to perform automated grasping tasks, perform a wide range of hand gestures, and to incorporate a management system that allows the prosthetic to be taught user specified grasps and gestures. In this section the hardware and software specifications needed to accomplish this goal are discussed.

1.2.1 Hardware Specifications

The goal of the prosthetic arm is to be robust enough to be useful, while also being low cost and lightweight. Therefore, the team has created hardware specifications and requirements with the intention of satisfying those goals. The significant portions of hardware sections have been outlined in Figure 1.

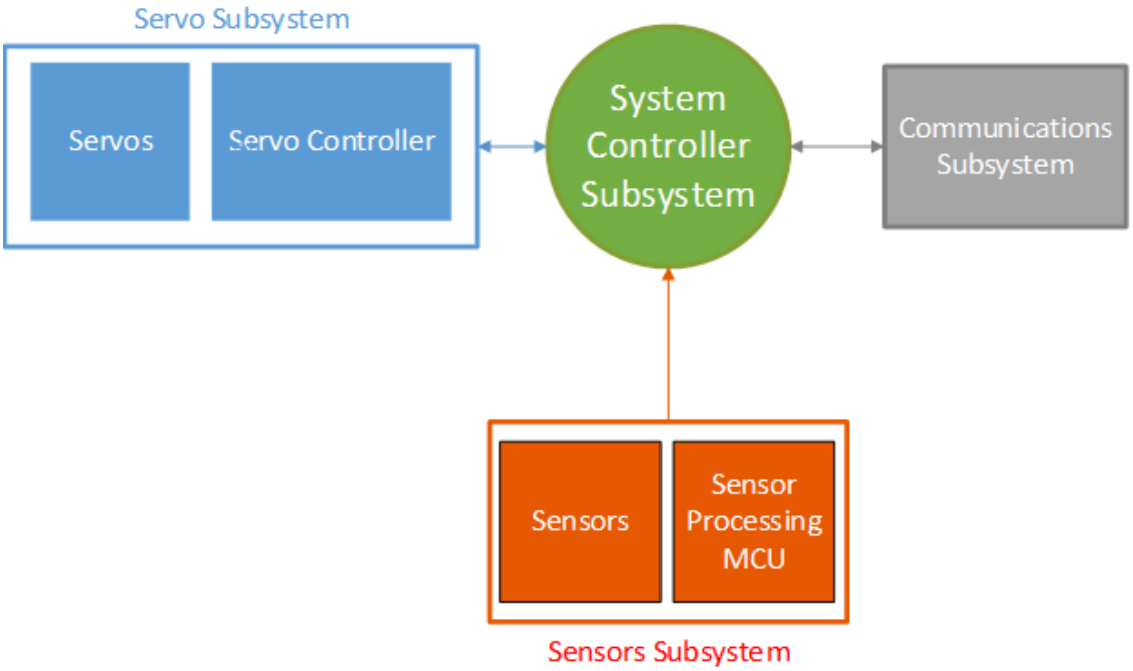

Figure 1. A high level block diagram showing major hardware components and how they are connected

Table 1 shows the list of all hardware requirements related to these major components. Following is the description of each software unit of the Intelligent Programmable Prosthetic Arm:

- **Servo Subsystem** Controls the servos linked with each individual finger of the prosthetic. The Servo Subsystem is composed from a secondary microcontroller and the five servos required to control the fingers of the prosthetic. The servos must be able to provide sufficient enough torque to

grasp objects firmly and be able to hang onto objects when the user is lifting the object.

- **Sensor Subsystem** Provides data to the System Controller Subsystem about the external environment of the hand. The Sensor Subsystem consists of a secondary microcontroller, a 16 to 1 multiplexing integrated circuit, and various sensors that provide data to the System Controller Subsystem. The sensors includes an electromyography sensor, force sensitive resistors, infrared distance sensors, and an inertial measurement unit. This allows the hand to detect pressures when grasping, detect distances from objects to trigger actions, and recognize its position and movement in space.

- **Communications Subsystem** Connects the prosthetic wirelessly with Bluetooth so that it may be controlled and programmed via a smartphone application. The Communications Subsystem consists of a Bluetooth enabled integrated circuit, such as the HC-05 or HC-06.

- **Power Subsystem** Supplies power to the servos, microcontroller, sensors, and other integrated circuits. The Power Subsystem consists of a rechargeable 7.2V battery, used to power the servos, and a 9V battery coupled with appropriate voltage regulators to supply the microcontrollers and sensors with power. The rechargeable battery used needs at least 3000 mAh worth of power to supply.

- **System Controller Subsystem** Coordinates all of the subsystems mentioned above. It directs the Servo Subsystem to the correct gesture, gather and interpret data from the Sensor Subsystem, and transmit and receive data from the Communications Subsystem. The System Controller Subsystem requires a powerful centralized microcontroller. Low power microcontroller units tend to have a maximum clock speed of 20 MHz. Whereas a higher power microcontroller could potentially clock as high as 120 MHz. The advantage of these clock speeds is that it enables the run of code and process incoming data much more quickly. As a result, the prosthetic is more responsive.

Requirement	Description
1	All grasping tasks should be able to hold a weight of 3-5 pounds.
2	The wireless communication should work within 8 meters.
3	Grasping tasks should withstand 5 minutes of continuous use.
4	At minimum, the hand should have 1 grasping setting and 5 miscellaneous gestures stored at all times.
5	While grasping and holding, the arm should be able to be lift the object from a downward position to a position perpendicular to the ground.
6	The arm's battery should last 1 hour before requiring a recharge, when extensively used.
7	The hand should grasp when an object is less than or equal to 1 inch.
8	The hand should stop grasping when it detects dangerous levels of pressure.
9	The hand should have a microphone which will be used for the voice commands.
10	The price of the prosthetic system should be under $500.
11	The arm will have a reset button to override the System Controller Subsystem servo position

Table 1. Hardware requirements for the IPPA project.

1.2.2 Software Specifications

There are five major software component that compose the whole IPPA system; these are represented in Figure 2. Data flows from one component to another as shown in Figure 2.

Following is the description of each software unit of the Intelligent Programmable Prosthetic Arm:

- **Main System** software consists of the main control of the entire system. It analyzes sensor input of EMG signals and voice commands to determine when to trigger a gesture. This is accomplished by using thresholds on the inputted sensor information, and data analysis from multiple experiments. Voice recognition for command triggered gestures are handled by this software unit. The resolved servo's position from voice command or EMG signal analysis is transferred to the Servo Controlling software. This software unit also receives input from the Communication unit, and has the necessary modules to change the gestures and triggers stored in the system.

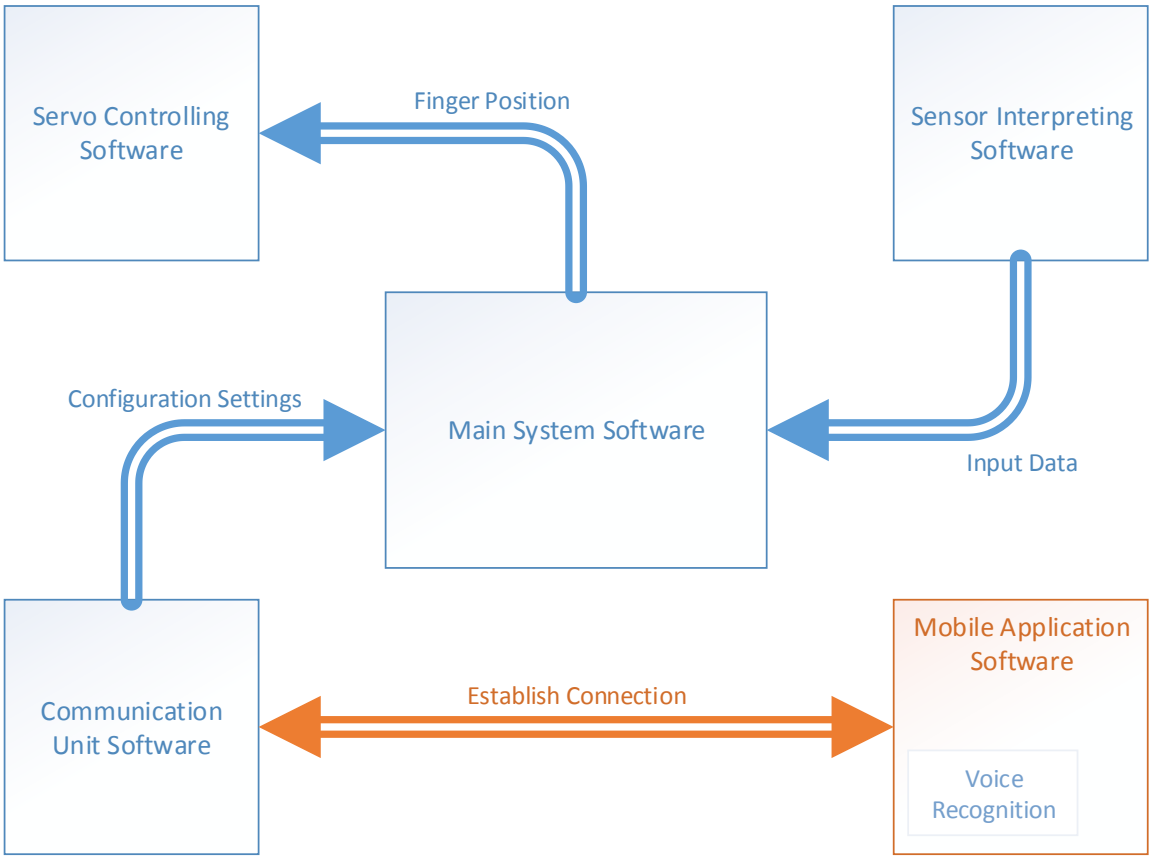

Figure 2. Data flow for the five major software components.

- **Servo Controlling** software sets the position of each finger in the prosthetic as dictated by the Main System software. This must be accomplished at a steady but gradual speed to provide a natural gesture. The Servo Controlling software must be able to override the position given by the Main System, with a reset (opened hand) gesture. This action is signaled by the Sensor Interpreting software to avoid the use of excessive pressure on any object or human body parts; of by a physical reset button in the arm.

- **Sensor Interpreting** software receives the input information from every sensor and provide an interpretation of the prosthetic's status and surroundings. The input from the sensors consists of applied pressure on specific points, the prosthetic's rotation, and others (see section 1.2.1 for more details on the sensors). This software unit must determine if the prosthetic arm must release; if the person wants to grab an object and what the position is for each individual finger (depends on the object). The EMG signals in the amputee's arm is transferred to the Main System for further analysis. This is accomplished by using thresholds on the inputted sensor information, and data analysis from multiple experiments.

- **Communication Unit** software provides an interface between the prosthetic arm subsystems and the mobile application. This provides the means of transferring data to the arm, such as real time position for the arm or new gesture settings.

- **Mobile Application** software communicates with the Main System software unit through the Communication Unit software. It provides a way to change the settings for the gestures as well as the triggering mechanism for those (i.e. command used). This software must interpret the user input through the User Interface of the application and translate it into an IPPA gesture format before sending the data to the Main System. It has pre-programmed gestures to send to the arm. Also, in the learning mode the user generates new gestures using this application, demo the gestures in real time, and add them to the arm if desired.

Requirement	Description
1	Device will provide a reset override for the arm for safety concerns
2	Device will provide interpretation of EMG signals for arm movement triggers
3	Device will provide an UI for changing the arm settings
4	Device will provide a way to re-program the IPPA with different gestures
5	Device will provide an application to add/remove gestures to/from the IPPA
6	Device will provide a smooth user interface through the mobile application
7	Device will provide precise and consistent voice recognition
8	Device will provide a learning mode, where the arm will learn new gestures
9	Device will provide reliable and real time communication between the prosthetic software and the mobile application

Table 2. Hardware requirements for the IPPA project.

2 Research

This chapter contains the research efforts dedicated to this project. Current commercial prosthetic limbs as well as current work in robotics, specifically robotic grasping has been researched. All the technology needed to implement the desired functionality and quality features for the IPPA also are researched. From this research many design decision were reached.

2.1 Related Work

There are many commercial products that relate to this project. In addition, there are some research publications that relate to the feature of automatic grasping and sensing. These sections describe commercial projects and research publications that are relevant to the project.

2.1.1 Commercial Prosthetic Limbs

There are not that many commercial prosthetic hands in the market that take full advantage of the technologies available today [6]. The latest and most advance prosthetics are: iLimb by Touch Bionic, Bebionic by RSL Steeper, and Michelangelo by Otto Bock. The cost of these ranges from $25,000 to $100,000 depending on durability and functionality, as well as the options the amputee decides to have in the prosthetic. The cost also increases because of the extensive adjustments and training required for each individual person. Figure 3 shows two of the latest commercial prosthetics available in the market, the bebionic 3 and the iLimb ultra.

Figure 3. The i-limb ultra prosthetic hand from touch bionics.

These three prosthetics work by interpreting the electromagnetic signals that are left in an amputee's arm and translating those signals into pre-programmed features or motions. The number of features programmed in the prosthetic is limited by the ability of the amputee to control and trigger the right signals in their arm. As the person learns more features can be added to the prosthetic, but this

continues to add to the cost of the prosthetic. The main features that are across all of these commercial prosthetics are:

- Five individually operated fingers
- Gesture and grip pattern selection to allow customization
- Automatic grasp
- Strength and speed variation for sensitive tasks
- Application for customization and training support
- Long continuous usage
- Light weight
- Durable material selection
- Optional thumb positions
- Natural looking design

Commercial prosthetic have similar physical characteristics, as shown Table 3. The more complex prosthetics have 11 joints and have an adaptive grip, which means these prosthetics can better adapt their grip to a given object. However, this raises the weight of the prosthetic which may limit the use of these by adults only.

Prosthetic Arm	Developer	Weight (g)	Overall Size	Number of Joints	Number of Actuators
SensorHand (2011)	Otto Bock	350 – 500	7 – 8 ¼	2	1
Vincent Hand (2010)	Vincent Systems	–	–	11	6
iLimb Pulse (2010)	Touch Bionics	460 – 465	180-182 mm long, 75- 80 mm wide, 35- 45 mm thick	11	5
Bebionic (2011)	RSL Steeper	495 – 539	198 mm long, 90 mm wide, 50 mm thick	11	5
Michelangelo	Otto Bock	~420	–	6	2

Table 3. Industry prosthetic arm and their physical properties.

2.1.2 Similar Projects

Several Capstone projects have been developed to solve a similar problem to this project. Two notable senior design projects that were focused on designing an intelligent robotic hand were the IRISHAND smart robotic prosthesis and the Design of a Human Hand Prosthesis.

Design of a Human Prosthesis This project was a senior thesis project completed by Paul Ventimiglia. He developed mechanical design of a cost effective, anthropomorphic prosthetic hand. The hand had five actuated fingers, a compound thumb gearbox that enabled thumb roll actuation, sensors for feedback control, and a Lithum Polymer battery and an Arduino Pro Mini micro controller. The hand is able to complete 4 grasps, has a minimum power grip of 150N and a minimum force of 15N for a precision pinch grip. The hand has force sensing on the fingertips using LED variable feedback and analog potentiometers to measure the rotational position of each finger joint.

The noteworthy aspects of the project was the design the thumb roll gearbox and the compact design. The thumb roll gearbox enabled was novel compared to other commercial applications and enabled the hand to incorporate a grip that could hold flat surfaces on top of the index finger using the thumb. The design was developed using off the shelf components and was designed to encase all the components compactly. Another aspect was that his first intention was to develop an entire prosthetic arm, however talking to a local amputee; he chose to focus his design on the hand for several reasons. One reason is that from the amputee's point of view, there are very few amputees that require a full arm amputation. Another reason is that every amputee's situation is unique. The design of the prosthetic has to accommodate the uniqueness of the amputee due to the medical problems that the amputee has. Also, the author chose to design a hand because he can incorporate a universal bolt that would fit to an existing socket that most amputees take the time to have a custom fit. The critical aspects of this project was the user did not work on a control system to control various grasps nor performed test on the functionality of the hand actuation. His future recommendation was to integrate the system with commercial myo-electric sensors to enable his design to be ubiquitous to commercial non-anthropomorphic prosthetic arms [13].

IRISHAND Students at WPI developed a capstone project called IRISHAND. IRISHAND is an anthropomorphic robotic hand that intelligently automates grasping an object with minimal user input. The hand uses a vision sensor and an object recognition algorithm to analyze what object it is about to grab, adjust the prosthetic hand to the most appropriate grasp, and execute the grasp automatically.

They wanted to develop a prosthetic that was low cost to manufacture using 3D printing, highly versatile with intelligent sensing, and exceeded the potential of current upper limb prosthetics. Their idea to use a camera and use object recognition was inspired to develop a system that could adjust the grip to best fit the object the user is about to grasp and execute the grasp automatically similar to how the brain executes this subconsciously. Their vision sensing system was implemented using a pcDuino that used an ARM A8 cortex processor. Using OpenCV and embedded Linux, developed a general object recognition system that recognized the general shape of the object and recognized an AR Tag if the tag

was present on the object. The general shape was determined using color segmentation, canny edge detection, and Hough transform to find the principal lines that defined the object the hand was about to grasp. Once the general shape was determined, they compared the shape to templates to determine if the object was a sphere, a cube, or a cylinder. If the shape was not determine, the recognition system would check to see if there is an AR Tag in the scene. The tag would be used to identify an object that is known in the internal system.

When the object was classified, that information would be sent over to the main controller using serial UART. The main controller would then signal the motor controls to adjust the motors to actuate the fingers to a pre-defined grasp. To automate the fingers to grasp the object correctly, the team used series elastic actuators to detect force from feedback and use rotary potentiometers directly attached to shaft of rotating joints to control the position of the fingers. The strong features of this system was the algorithm to detect general object shape and the force feedback sensing. When the object recognition system was tested to determine the general shape of an object, it had an accuracy rate of 93.3 percent overall. Another benefit was the force sensing using the series elastic actuators was able to detect when hand has applied enough force to grab an object successfully. In addition, the entire system was successfully constructed to fit in a prosthetic that was the size of an average adult male hand.

The poor features of the system was the object recognition had poor accuracy to recognize AR tags and the general shape recognition was accurate with the constraint that the object's color was a certain color. Finally, they used the pcDuino as a co-processor to process the camera frame significantly increased the cost of the system and made recognition of objects a computationally expensive operation. As the IRISHAND utilized object recognition to automate grasping gestures, this is a computationally expensive process and limits the amount of objects you wish to grab because the recognition requires an object to either be a simple recognizable shape or have a QR code [28].

2.2 Relevant Technologies

This section describes the technologies that are relevant to this project. The purpose of this research into relevant technologies is to help assess the components and technologies that are involved in designing a low cost prosthetic limb. This section surveys 3D printing, sensors, mobile platform, communication methods, servos to actuate the hand, and human-technology interaction.

2.2.1 3D Printing

Since 3D printing became available in the market, there has been a wide spread movement to fulfil the need for cheaper prosthetic limbs. There are hundreds of cases of children and adults whose lives have been improved by a simple 3D printed prosthetic arm, leg or hand [13]. A lot of the open source work done in this

area has been inspired by children and veterans. A major benefit of 3D printed prosthetic is not only the cost to make it, but also the fact that maintenance is very affordable and somewhat easy to achieve [13]. Since the prosthetic is composed of 3D printed parts, all it takes it to 3D print the part that is broken or worn out, and install it onto the prosthetic.

There are a variety of open source designs from the simplest to more complex prosthetics available online for replication and continued development. This project uses this technology to create the main body of the prosthetic hand. The hand is composed of multiple parts, which were printed separately. The use of different types of plastic and/or printing configurations could be use in order to achieve maximal performance and durability. The design was based on the InMoov hand, which is available for continue development and improvement from their website.

The 3D printed hand is hollow in order to provide space for the sensors, and the servo strings. The forearm is hollow as well to provide space for the IPPA embedded system. The performance of the prosthetic is directly correlated with the grip and motion precision of the hand.

With the permission of the University of Central Florida Idea Lab, the 3D Printer was used to manufacture all the plastic components of the IP Prosthetic Hand. The type of material is ABS plastic. The 3D prosthetic design could not be modified with the provided files from InMoov. The design details of the prosthetic are further discussed in the 4.7 section of this documentation. After printed, the hand needs to be assembled and modified as necessary.

2.2.2 Sensors

The IPPA features several sensors to allow the prosthetic to make automated decisions, such as automatically opening or closing, depending on the sensory input. The sensors chosen need to be useful, low cost, low power, and low weight in order to be viable options. Research on various sensor technologies is described below.

2.2.2.1 Initial Measurement Unit (IMU)

The IMU is a combination of three different sensors. An accelerometer, which could be used to measure changes in the arm's rate of acceleration or movement. A gyroscope, which detects changes in rotation, specifically, pitch, roll, and yaw. Lastly, the unit also includes a magnetometer, to detect magnetic north [9, 10].

The MPU-9150 provides an accelerometer, gyro, and magnetometer in one package. The chip communicates over I^2C and can be purchased individually as a chip, or in breakout board form. The chip is also low cost, runs at a low voltage level, about 5V, and provides 16 bit readings [14].

2.2.2.2 Electromyography

Electromyography measures muscle activation via electric potential. Normally, EMG is used to diagnose the health of muscles, such as the heart, and the cells that control them, called motor neurons. The motor neurons create electrical signals to generate a response from muscles [17]. The EMG sensor detects these signals into numerical values that this team could use to be able to leverage to generate a response in the prosthetic. Advancer Technologies has created an EMG breakout board that operates at 3.5V to 9.0V. With the inclusion of this sensor, the prosthetic could respond to the user flexing his/her arm muscles [2]. The board itself is very small, 1 inch by 1 inch, and translates the input signals from 3 uniquely placed electrodes.

2.2.2.3 Pressure Sensors

Force sensitive resistors (FSR) are made from material whose resistance changes when a force or pressure is applied to it. The resistors are constructed from a sheet of conductive polymer, which consists of electrically conducting and non-conducting particles. When a force is applied to the layer of polymer, particles touch the conducting electrodes, which changes the resistance of the film [27].

By placing this material on the hand in strategic locations, such as the fingertips or palm, the team could detect when the hand has a strong grasp on an object or detect when the hand is exerting too much force on an object, preventing potential damage. The FSRs on SparkFun.com read a resistance larger than 1 Mega-Ohms when no pressure is applied. They can sense forces ranging from 100g to 10 kg. To sense the change in pressure, a simple voltage divider circuit would be used [24], as shown in Figure 4.

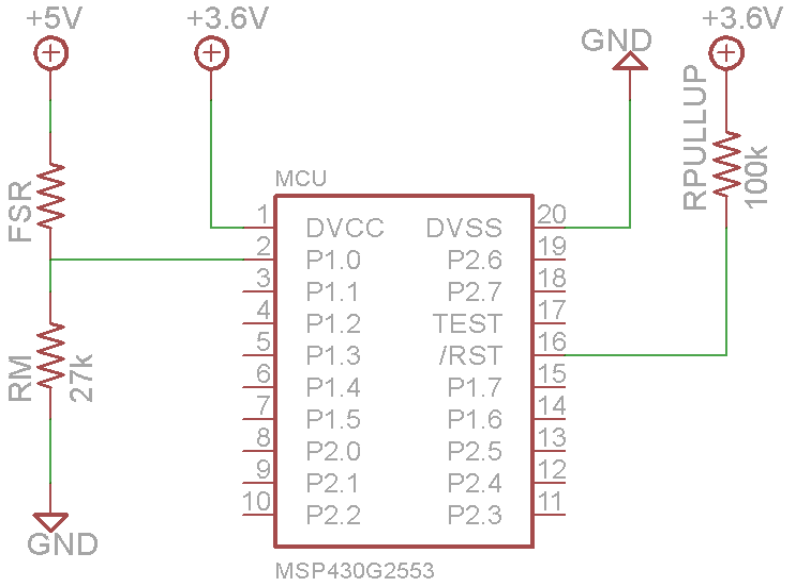

Figure 4. A schematic showing how to create the voltage divider circuit for the FSR sensor and observe its voltage with the MSP430.

In this example an MSP430G2553 microcontroller is used to read the voltage on the FSR. The voltage changes depending on the resistance of the FSR. Figure 5 shows how to calculate the resistance of the force sensitive resistor as a function of the resistors used and the detected voltage on the bottom resistor.

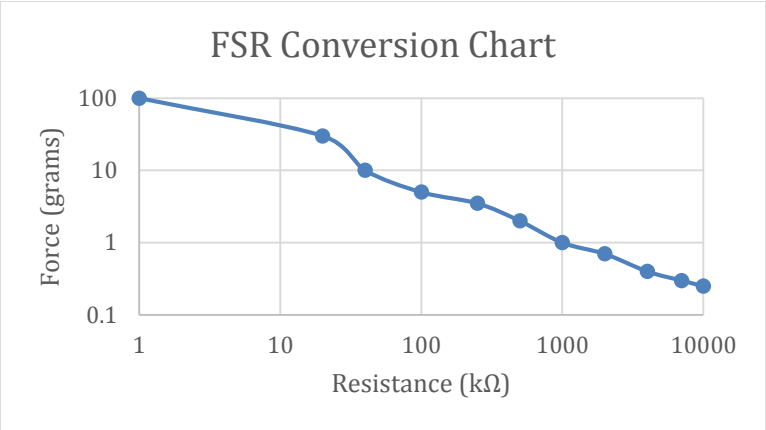

Figure 5. A chart that can be used to convert resistance of the FSR to force being applied to the FSR.

Using the equations below, the team can match the force sensitive resistors resistance to the chart in Figure 5 above to calculate an approximate force reading, in grams.

$$V_{out} = \frac{R_M * V^+}{R_M + R_{FSR}}$$

$$R_{FSR} = \left(\frac{R_M * V^+}{V_{R_{FSR}}}\right) - R_M$$

2.2.2.4 Distance Sensors

Ultrasonic An ultrasonic wave is emitted and the reflections from nearby objects are received. One of the requirements of the hand is that it can detect an object is near the hand and initiate a grasping gesture to pick up that object. The frequencies of these sensors typically are emitted at 30 to 50 kHz, which would be nearly inaudible as to not interfere with the life of the user. When objects are not moving, the amount of time it takes for a wave to return remains the same. When objects are far away, it takes the wave longer to return to the sensor than when an object is close to the sensor. Ultrasonic sensors detect within ranges of 2 cm to 3 m [23].

A drawback from using an ultrasonic sensor is that they require a continuous supply of energy to function. Another drawback is that they may trigger a false detection since the ultrasound detection range can leak into unintended spaces. Compared to passive infrared sensors, ultrasonic sensors require much more power. Depending on the angle of the object being detected and the size of the

object, the sensor may not function correctly. The PING))) sensor by Parallax is a commonly used ultrasonic sensor. It can detect when an object is more or less directly in front of it.

Microwave Microwave sensors detect motion through the same principle as Doppler radar, like ultrasonic sensors. The difference is that microwaves are emitted. Phase shifts in the reflected microwaves are created due to motion of an object moving towards or away from the sensor. Compared to ultrasonic sensors, microwave sensors are usually more costly and consume more energy. However, there is not a significant difference in the performance of the two [23].

Passive infrared Passive infrared (PIR), sensors are the most commonly used sensors in the market. They are typically seen used in burglar alarms and automatic lighting systems. They achieved this widespread use because of their utility, energy savings, and cost savings. They are composed of at least two components: a photo-transistor and an infrared LED. PIR sensors detect changes in infrared radiation triggered by any hot moving body such as a human hand or arm, an automobile moving, or even sometimes a warm breeze. The small fluctuations detected are amplified and processed by a controller. An advantage of the PIR sensor is that there is a passive energy component which requires little to no energy to detect motion. During idle operations when there is little to no movement to be detected, the sensors are much more energy efficient.

2.2.3 Mobile Platform

This section discusses the major mobile platforms available for development of the mobile application that is used to manage the prosthetic arm. As shown in Table 4, the mobile market share is dominated by the Android OS and iOS. This section discusses the characteristics, benefits and drawbacks of these two mobile platforms.

Mobile Operating System	Market Share July 2014
Android	52.5 %
iOS	41.4 %
Microsoft	3.3 %

Table 4. Mobile platforms and their market share.

Android OS is an open source operating system that has shown a remarkable growth in the global market share in the last 4 years [8], with approximately 50% of the US market share today. Smartphones with Android OS can be purchase from many vendors and at a cost as low as $20, which makes it very accessible to most people. This mobile platform has a very prevalent app market, and there are many tools and forum support available for development. In order to develop and publish an application for this market, there would be no extra cost and no license is required. The development environments available are Eclipse IDE (open source software) and the Android Studio Beta, with the Android SDK, which are

free of cost. Simulation of all versions of the operating system are available for testing through these two environments, which is very beneficial for development and eliminate the need to purchase a device. The team's Computer Engineers are trained and experienced in developing applications for Android OS which reduces the cost of this part of the project in terms of time.

iOS is a widely used operating system as well. Its market share varies from 40% to 50%, making it a strong competitor of the Android OS. However, it is a proprietary software and runs exclusively in Apple's iPhones. This series of smartphones are exclusively expensive; they are sold for approximately $500. The app market for iOS is known for its pricy but great applications. Development for this mobile platform is costly since a license is needed and a permission to publish an application in their market. However, there is a lot of forum support for development in this platform. An additional cost to developing in this mobile platform is the time and resources required to train the team's Computer Engineers since this development environment is not known to any of the engineers in the project.

2.2.4 Communication Methods

In order to meet the functional requirements of the external mobile application to manage the prosthesis, research needs to be done to determine the best communication channel for the external application to communicate to the main controller of the prosthesis. The possible communication methods are wired communication and wireless. To make sure ease of use is assured, wireless communication was used for this system. The types of communication that are discussed are the IEEE 802.11 (Wi-Fi) and Bluetooth wireless communication. Before discussing the various wireless communication, several factors were considered to determine what the best communication service for the system is.

Range The range from the external application to the main processing unit is important to consider due to how the user would use the external application along with the prosthesis. It is assumed that the most frequent use case would be that the user is using the external application when wearing the prosthesis, which limits the distance between systems to within one meter. Another use case that can be considered is whether the user could be managing the prosthesis with the external application when not wearing the prosthesis. In this case, a safe estimate on the maximum range the external application is away from the prosthesis would be in the range of 6 to 10 meters, considering the prosthesis could be the other side of a room; but this should be unusual. Usually, the prosthesis would be used and managed within close proximity of the amputee [6].

Data Rate The rate of data transmission is a very important factor. The system uploads several different types of data and requires to have bi-directional communication in real-time. Features where this data transmission is large when the user loads new grasps and gestures for the prosthesis to complete. To

complete new types of gestures, each gesture contains instructions for the Main Controller to handle actuation of the servos, which triggers the gesture or grasp. Depending on how much data is required for the Main Controller to run a range of gesture and grasps effectively, it would be important to have the ability to send a large amount of data at one time [22].

Energy Consumption Energy consumption is a very important factor because this is the biggest limiting factor for the IPPA. Wireless communication factor is viable to meet the functional requirement of time of usage. The functional requirement of usage is that grasping tasks should withstand 5 minutes of continuous use and the arm's battery should last one hour before requiring a recharge [22].

Frequency Band This is another important factor because the team would need to evaluate whether a wireless communication channel could have a lot of interference [22].

Security This is another important aspect of choosing a wireless communication channel. The team needs to ensure that there is a level of security on the communication channel so no one can acquire unwanted access to the control and data of the prosthesis main processing unit or the external application [22].

2.2.4.1 IEEE 802.11

The IEEE 802.11 wireless communication protocol is the wireless communication commonly known as Wi-Fi. The data rate for this protocol ranges from 11Mbps to 150 Mbps depending on the type of IEEE 802.11 protocol. There are three different types of protocols: 802.11b, 802.11g, and 802.11n and each one has a maximum data rate of 11Mbps, 54Mbps, and 150Mbps respectively. The inner range of the wireless communication is 35 meters, 38 meters, and 70 meters respectively. Full-featured Wi-Fi modules usually consume 3.3V and are priced between $30 to $50 dollars. Important features using IEEE 802.11 are that it has high data-transfer reliability and speed, and operates in different frequency bands depending on the protocol. 802.11b and 802.11g use the most common frequency of 2.4GHz, but 802.11n can be configured to either 2.4GHz or 5GHz. For further data transfer reliability and speed, 802.11n provides Spaced-time block coding (STBC) which reduces error rate at the price of higher power consumption. Also, most modules that use IEEE 802.11 offer WPA and WPA2 encryption for secure data transmission [22].

2.2.4.2 Bluetooth

Bluetooth radio operates at different frequencies and has different power consumption and data rate depending on class of Bluetooth Radio. The one most commonly used as modules for embedded systems is Class 2 Bluetooth Radio. Data rate for using Bluetooth wireless communication is between 1- 3Mbps. The range is of wireless communication is around 10 meters. Class 2 Bluetooth Radio

runs on 2.5mW power consumption. The price range of a common Bluetooth module is between $11 and $40 dollars, excluding shipping and handling. Current security of Bluetooth Radio is based on PIN Code security, where the devices that are connecting have to enter a PIN code. Features of Bluetooth Radios are that it is designed to be low cost, low range, and use low power supply. A drawback is that data transmission is not as reliable as IEEE 802.11. For reliable transmission and even better power consumption, a designer can choose Bluetooth Basic Rate (BR) Bluetooth radio, which has an Enhanced Data Rate Mode and High Speed Mode [22].

2.2.5 Powering the Prosthetic Hand

There were several choices regarding how to mobilize the fingers of the hand. The fingers would need to be strong enough to perform everyday tasks such as opening doors, lifting groceries, and shaking hands. According to a NASA study, the average, adult male hand is capable of producing about 8 kg-cm of torque [19]. The motors that are chosen for this project must be capable of producing at least that much torque.

Linear Actuators Linear actuators create motion in a straight line, unlike motors or servos, which create motion radially. Electric linear actuators actually use an electric motor and convert the radial motion into linear motion. Linear actuators can be constructed to move at high speed or high force. The DC motors of the actuators are either mounted on the side of the actuator or in-line with the actuator. The disadvantage of linear actuators is that they are quite large, too large to be used in a prosthetic limb.

Stepper Motors A stepper motor divides a full rotation into a number of equal steps. DC motors rotate continuously when voltage is applied to the terminals. Stepper motors can be turned by very precise angles, which can allow for greater precision and accuracy in preprogrammed gestures. They also provide very good holding torque, which would be good for holding objects with the hand. However, to achieve a holding torque of about 9 kg-cm, the weight of the motor increases to over 1.5 lbs. The high weight of stepper motors makes them an unattractive choice in the hand.

Servos Servos are composed of an electric motor mechanically linked to a potentiometer. A controller transmits pulse-width modulation (PWM) signals to the servo, which translates the signals into a position. The potentiometer can be tapped, observed, by an analog input pin of a microcontroller, which provides positional feedback to the controller. Unlike stepper motors, servos can sustain high torque for a short amount of time, limiting the length of holding tasks, which creates a disadvantage. There are many servo models that are high performance, low weight, and low cost, which makes servos the ideal mechanisms to power the fingers.

HK15298 The HobbyKing HK15298 servo has a rotational range of 90º with a torque of 14 kg-cm at 6V and 15 kg-cm at 7.4V. They may only be purchased from HobbyKing directly at a price of $20. It weighs 66 grams, with a length of 1.6 inches, width of 0.79 inches, and a height of 1.65 inches.

Pololu 1501MG The Pololu 1501MG has a rotational range of 90º with a torque of 16 kg-cm at 6V. They retail for about $20. The 1501MG weighs 60 grams with a length of 1.6 inches, width of 0.8 inches, and a height of 1.55 inches.

HS-805BB The HS-805BB has a 180º rotational range with a torque of 25 kg-cm at 6V. The 805BB retails for about $40. It weighs 152 grams with a length of 2.59 inches, a width of 1.18 inches, and a height of 2.26 inches.

Seiko PS-050 The Seiko PS-050 provides 65 kg-cm of torque at 8.4V. The price of the PS-050 is $228. It weighs 280 grams with a length of 3.9 inches, a width of 1.73 inches, and a height of 3.65 inches.

2.2.6 Voice Recognition

Another feature of this project is the ability to do voice commands to get the IPPA to do selected gestures. Voice recognition is a technology that has had great improvements in the past years, and has become available to the non-speech recognition experts through open source libraries. During the adaptation phase, the time it takes the amputee to learn how to send/control the electromagnetic signals to their arm, the amputee may not be able to use their prosthetic arm as much as they would like. In order to allow the person, especially kids, to use their prosthetic arm at full potential during this period the system introduces the use of voice commands. The utilization of this technology further extends the capabilities of the prosthetic since it does not depend on the available EMG signals. The module that is used to interpret human speech uses voice recognition technologies to accomplish this task. There are two possible solutions to incorporating voice commands: one is to add this functionality to the embedded device in the arm, the other is to add it to the mobile application. The speech recognition accuracy is affected due to the constraints introduced by handheld devices, such as processing power and storage.

Embedded There are many speech recognition libraries available, but not all satisfy the requirements for audio processing in embedded processors due to the complexity of their algorithm. Therefore, the options of available free speech recognition libraries are further reduced. Having the speech recognition processing done in the arm as part of the embedded system has the following benefits: always available to the user when wearing the IPPA; commodity of not having to carry any other device; and does not require internet connection. However, it remarkably affects the power utilization of the IPPA depending on the usage of this feature. This impacts the microcontroller unit chosen for the Main System Controller, and adds the need for a microphone component.

Mobile In the case of mobile speech recognition there are many possibilities, especially because most mobile platforms have already developed a speech recognition API and made it available to developers. Most of these APIs stream the audio to remote servers to perform speech recognition, which adds a requirement for the user to use this functionality. Having the speech recognition processing done in a mobile device as part of the IPPA's mobile application has the following benefits: maintain the power consumption of the arm embedded device to a minimum; use of advance speech recognition at no cost; easy development, testing and integration to the system. This impacts the choice of mobile platform for the IPPA's application.

Possible open source libraries/software available to use are listed in Table 5.

Library/Software	Development Environment
CMUSphinx – PocketSphinx	Embedded and handheld devices
Android Speech API	Android
Windows Phone API	Windows Phone
OpenEars	iPhone

Table 5. Speech recognition software available for mobile platforms.

2.2.7 Human Technology Interaction

The hand is responsible for a variety of tasks. Important tasks are grasping and gestures. The hand should be capable to of doing a tremendous number of grasps that range from small and intricate to strong and forceful. Gestures are an important form of nonverbal communication for human interaction. This section discusses the issues with designing prosthetic limbs and discussing important human prosthesis interactions to consider.

This project's goal is to design a hand that is capable of a large array of grasping tasks by analyzing the anatomy of the human hand. The difficulties to match complexity of movement and sensing electrically and mechanically have resulted in prosthetic limbs only preforming a fraction of functions that our natural hand can do.

As important as the hand serves for numerous grasping tasks and gestures, another important factor is the significance the hand represents to a person's self-image. A person's hand is an important part of their self-image socially and psychologically. Because of this, amputees often deal with discomfort in society as being seen differently. They suffer from a tradeoff of choosing prosthetics that are functional versus visually similar to a regular hand. As the prosthetic hook is the most functional in handling simple grasping tasks, their unusual shape and functions stands out. On the other hand, the user can decide to get a cosmetic prosthetic at the sacrifice of the prosthetic being functionality useful.

The IRIS team addressed the issue of appearance by designing their prosthetic hand to be anthropomorphic and addressed the issue of functionality by developing an object recognition system with a webcam embedded in the hand to intelligently sense the object the user wishes to grab and innately decide the best grip for grasping.

The designer of the MPQ capstone project addressed the issue of appearance by deciding his prosthetic design to resemble the size and appearance of an average adult human hand. It did not address the issue of functionality, but rather focused on the mechanical design of an anthropomorphic robotic hand. The goal of that project was for the hand to be used as a research platform for developers and research to focus on the issue of functionality.

This project addresses the issue of appearance by choosing the design of the prosthetic hand to be anthropomorphic and resemble and function like a human hand. The IPPA system addresses the issue of functionality by developing an intelligent, programmable system that enables the user to automate a variety of grasping tasks and gestures using several sensors for contextual understanding and feedback.

2.3 Possible Architecture

This section discusses the possible architectures and major components that are being consider for the design of the IPPA. These components are the main controller microcontroller, the secondary microcontrollers needed for servos and sensors, and the components that make up the power system; batteries, voltage regulators, etc. Principal concerns researched in this section are: processing speed, I/O capabilities, and power consumption.

2.3.1 Main Processor Unit

The main processor unit has control of the entire system. It handles information received from the wireless communication unit, handle information sent and received from the servo control unit, and signal servo control unit to which gesture or grasp to execute based on handle information received from the EMG sensor and from the information received from other sensors.

Since this is the unit that requires the most computation, memory storage and access, and I/0, important factors to consider are architecture, processor speed, memory, I/O, and power consumption. Additional factors that were considered were the cost and complexity of development.

BeagleBone Black The beaglebone black is a low-cost high performance system on a chip. The SoC is an embedded computer with MCU capabilities that was designed for hobbyist and developers to use. The board runs with the AM335x ARM-Cortex A8 architecture and has a whopping 1GHz processor speed. The high

processor speed enables the system to run a distribution of the Debian Linux operating system. The board has 512 MB of DDR3 RAM and 4GB eMMC Flash memory. The board has HDMI output with a mini HDMI port, USB host and USB client ports, Ethernet port, UART, I^2C, JTAG, and more than 46 GPIO ports. Additional features of this board include a 3D graphics accelerator and NEON floating point accelerator. Some disadvantages are that it cost $55.00, making this the most expensive choice in main processor unit among the choices discussed in this section. Also this board has high power consumption because for full reliability, the board needs 5V and a 2A power supply and the board consumes up to 2W of power. Overall, this system is not ideal for this project because the system has more computational power than what is needed for this project and the power consumption hinders the IPPA system's time of use.[1][5]

TM4C1294. The TM4C1294runs a 32 bit ARM Cortex M4. This architecture has several benefits, which include the clock speed at 120 MHz and cost around $20 to purchase. This architecture has less computational power than the BeagleBone Black, but has enough processing power to meet the requirements of the IPPA system for a lower cost. A disadvantage of this unit is the low amount of memory. The processor ranges from 256KB of Flash to 1MB of Memory. This is enough memory to handle data during the operations, however at the time to store information about gestures and grasps, the amount of memory the processor has may limit the system. Additionally, the processor is capable of handling up to 90 GPIO's and has 10 I^2C ports and 8 UART ports.

Tiva CC3200 The Tiva C series is a microcontroller unit that was designed to be the ideal solution for creating IoT applications. The biggest feature of this MCU is that it has a separate chip WLAN & TCP/IP stack capable of running IEEE 802.11 b/g/n wireless signal. The CC3200 runs a 32bit ARM Cortex – M4 core at 80 MHz. The board has 256KB of RAM and has 2 UART ports, 2 SPI's, audio output, camera output, I^2C, and 27 GPIO ports. As built in Wi-Fi enables easy to set up wireless communication, the system has powerful security features, which include a 256-bit encryption WPA security. Another strong feature is the board's power consumption. The board is functional between 2.1 – 3.6 V and can be powered from USB or 2xAA or 3xAAA batteries. As this board seems fit for wireless communication and low power, the board cost $29.99, which is more expensive than the TM4C1294. [25][18]

2.3.2 Secondary MCUs

The decision of which MCU to use as the controller of the servos and processing sensor inputs came down to two options, the Atmel ATmega328P and Texas Instrument's MSP430G2553. The important factors to base the decision on are:

- Power Consumption – The amount of power consumed by the microcontroller should be very small since all of the subsystems are powered by the same battery, except for the EMG sensor.

- Cost – Cost should be minimal to be in-line with the project objective of the whole system being low cost.
- Flash Memory Size – Storage used to store code and data. Needs to be large enough to hold all code written and any external libraries used for servo control or sensor processing.
- Operating Frequency – How quickly instructions are fetched, decoded, and executed. Operating frequency is directly related to the performance of the microcontroller.
- General Purpose I/O Count – Used to control external devices such as servos or to read data from external devices, such as sensors. The servo controller likely requires at least five GPIOs with pulse width modulation capabilities.

Table 6 shows hardware specification comparisons between two popular microcontroller units.

Features	ATmega328P	MSP430G2553
Operating frequency (MHz)	Up to 20	Up to 16
Max I/O Pins	23	24
Flash memory (KB)	Up to 32	Up to 16
Power consumption	0.2 mA at 1 MHz, 1.8V 7 mA at 16 MHz, 4.0V 14 mA at 20 MHz, 5.5V	0.23 mA at 1MHz, 2.2V 4.5 mA at 16 MHz, 3.6V
Cost	$3.85 each	$2.80 each p

Table 6. Comparisons of important features of the MSP430G2553 and the ATmega328P.

ATmega328P The ATmega328P is an 8-bit AVR RISC-based microcontroller [4]. In addition to its 32KB of flash memory, it has 23 GPIOs, six of which are PWM enabled. Flash is important because libraries to control the servos and to process incoming sensor data requires storage space. Libraries may also be required to enable I^2C, SPI, or USART communication, which are all features of the ATmega328P. The chip also contains several timer options, and a programmable watchdog timer with an internal oscillator, which could be used for programmable interrupts, or speed settings for the pulse-width modulation signals to control the servos.

To save on power, the chip can be set into several different power modes. Active mode consumes 0.2 mA, power-save mode consumes 0.75 µA, and power-down mode consumes 0.1 µA. The chip is able to operate with voltages from 1.8 V to 5.5V. With operations at maximum clock speed, 20 MHz, the chip consumes just under 14 mA. The chip is capable of operating at 16 MHz while requiring 4V supply voltage and about 7 mA of supply current, or about 28 mW of power [4].

MSP430G2553 The MSP430G2553 is a Texas Instruments microcontroller. Its main feature is that it requires very little power to run, which has the beneficial effect of extending battery life. It runs a 16-bit RISC architecture, 16-bit registers, and also has five different low-power modes. The clock is capable of running at 16 MHz. Included in the features are UART, SPI, and I^2C communication interfaces. The G2553 has 16 KB of flash to store the program code, libraries used, and any settings created during teaching mode [26].

The MSP430G2553 is capable of quite low power consumption. At 1 MHz in active mode, it consumed 0.23 mA of current, at 2.2V. In comparison to the ATmega328P, it has slightly worse performance in terms of power consumption at this level. When raising the clock speed to 16 MHz, the MSP430G2553 requires 3.6V and 4.5 mA, or about 16.2 mW of power. This is nearly a 70% improvement over the ATmega328P [26] at near maximum clock speeds.

2.3.3 Power

This section describes the possible methods of powering the IPPA system of microcontrollers, servos, and wireless communications. It discusses factors that are important when choosing power systems, such as life expectancy and maximum current output. Different batteries are discussed along with how to distribute their power to components that require specific voltage levels.

2.3.3.1 Batteries

A combination of batteries are used to power the entire system. At least one nine volt battery is required to supply power to the microcontrollers, sensor devices, and wireless communications systems. Typical alkaline 9V batteries have a lifespan of about 9 hours when continuously supplying 50 mA [10]. This lifespan could be doubled by using two 9V batteries. This lifespan could be doubled by using two 9V batteries.

The remaining unpowered subsystems consist of the five servos that are used to provide movement to the prosthetics fingers. Servos require large amounts of power, and can potentially draw up to 5 Amps when in heavy use. To supply this kind of power, it needs to use a high power rechargeable battery with a high capacity. Popular rechargeable batteries for remote controlled devices satisfy these needs. These batteries supply 7.2V, have a maximum current draw of around 38 Amps, and have a range of capacities of at least 3800 mAh. This satisfies the requirement of the arm being in continuous use for at least an hour before requiring a recharge.

2.3.3.2 Voltage Regulators

Since the prosthetic arm is built from many different electronic components, each with different required voltages, voltage regulators are required to regulate the voltage supplied by the 9V battery or batteries. A 5V regulator is required to supply power to the microcontrollers that processes incoming sensor data, communications data, and communicate with the servos. The EMG sensor requires a voltage of ± 3.5 to ± 9.0 Volts. An appropriate voltage regulator is selected to satisfy this need, although it would be possible to use the 5V regulator that is being used to supply the microcontrollers with power. The wireless communications device use 3.6V-6V (if the HC06 is used) or 2.7V – 4.8V (if the CC3000 is used). In the event of using the CC3000 as the wireless communications device, it would be recommended to use another 3.3V voltage regulator, instead of the 5V regulator that is powering the microcontrollers.

3 Research Prototype

Multiple prototypes have been created to prove the main concepts and requirements of this project are achievable. These prototypes for the main components of the IPPA's system are also very important for the decisions taken in the final design of the system. The subsystems that are lightly covered by these prototypes are: 3D printed hand, servo, EMG sensor, Bluetooth, main MCU, mobile application. These prototypes are discussed in this chapter.

3.1 Hardware

Prototyping is a necessary, preliminary step in the design process. It allows the team to test each part in the application that are being designed. In this case, one of the most important components to prototype with was the 3D printed hand itself. It allowed insight into what kind of mobility could be expected, the strength of the prosthetics construction, and the alterations that it might need in order to add sensors or microcontrollers.

A servo motor was also acquired, so that the team could test the prosthetic fingers being controlled by a servo and microcontroller. This allows the team to have a better idea of the kind of algorithm needed for the servo controller, and also how to best assemble the tendon system inside of the prosthetic hand.

The EMG sensor also arrived, allowing the team to see the output signals it generates from an arm. This provides the team with some understanding of how sophisticated the physical interactions between the user and the arm could be.

3.1.1 3D Printed Hand

A 3D hand was printed from the InMoov right hand design, which is an open source design. Only the hand was printed since they forearm needs to be designed differently from the InMoov forearm in order to fit all of the IPPA's components. The 3D printing took approximately 26 hours in the ABSplus – P430 3D, Dimension sst 1200es 3D modeling printer in the Texas Instruments Innovation Lab at the University of Central Florida. The hand uses 9.66 in^3 of model material and 3.62 in^3 of support material, which would have cost $66.40 to print. Since this hand is just a prototype the setting for the printer was sparse, high density. A sample of the individual parts needed for each finger is shown in Figure 6.

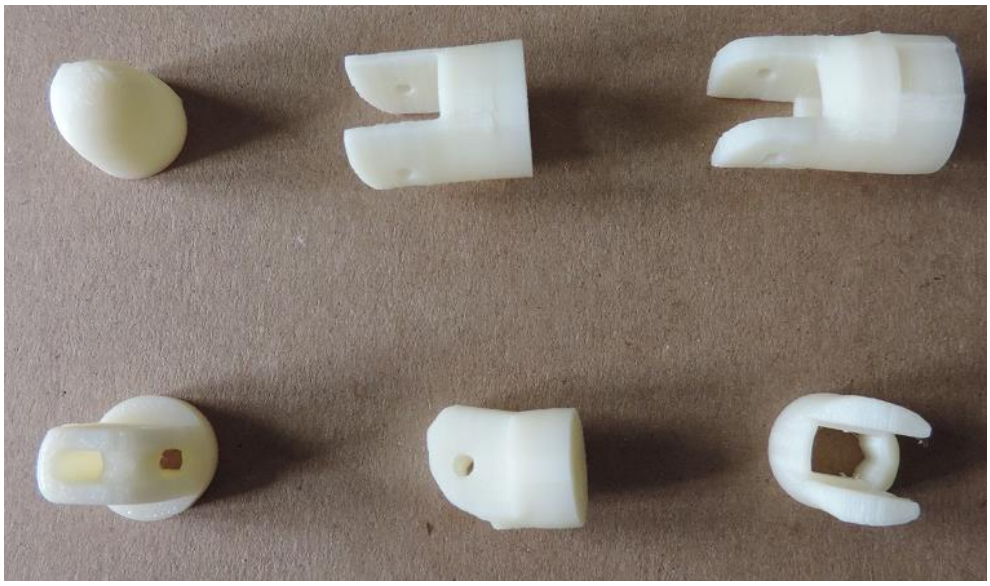

Figure 6. 3D printed parts that make up the index finger of a right hand.

A sample of the finger is shown in Figure 7. There are three joints per finger, and two additional joints on the hand for the ring-finger and the pinky-finger. Each finger joint rotates approximately 90 degrees, which give the hand the capability of grabbing a wide range of objects.

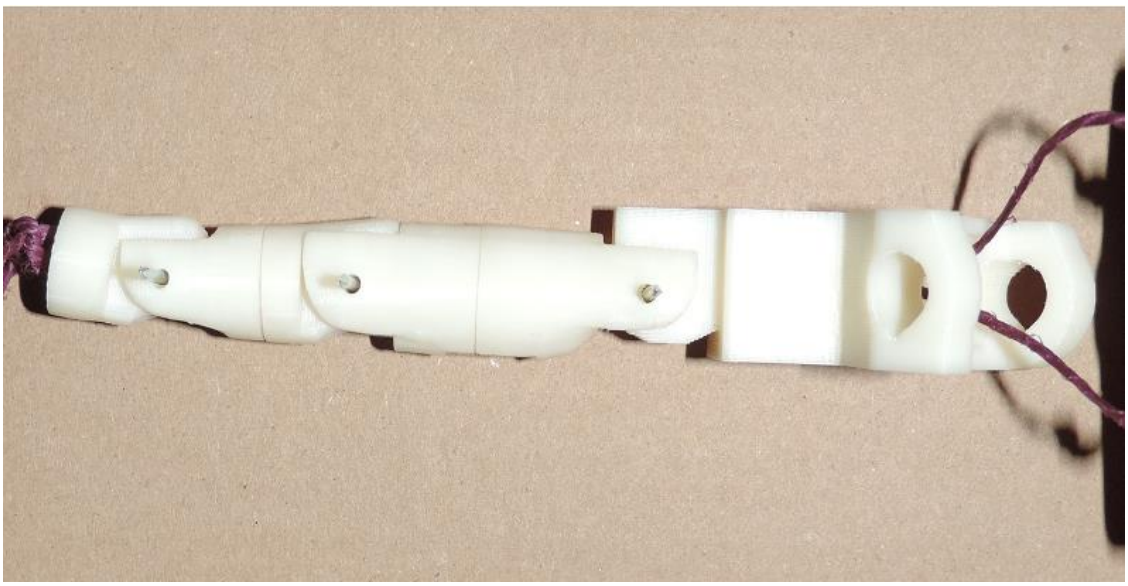

Figure 7. Assembled finger with the proper wiring for motion control.

Figure 8 shows the finger assembled and flexed. Note in both figures the finger is missing the finger tip, this was left out since it adds little value to the prototyping phase.

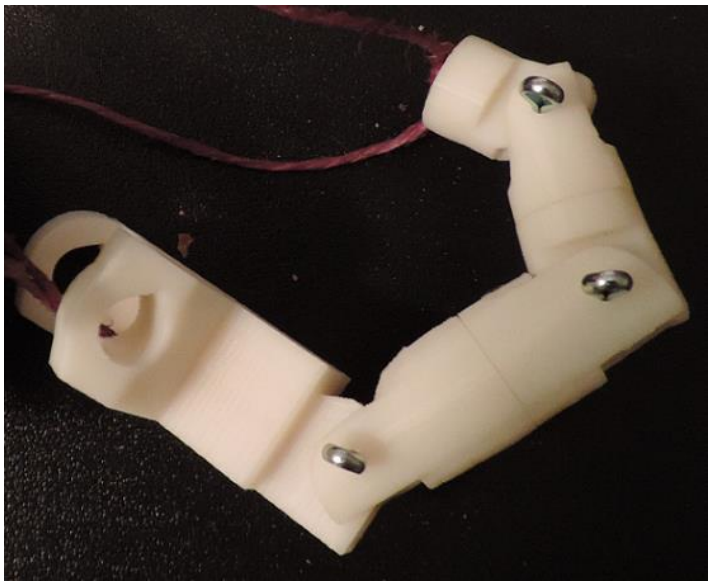

Figure 8. The 3D printed thumb in the flexed position.

3.1.2 Main Controller

This section talks about the initial prototype of the Main Controller. The initial design of the Main Controller is based off of the role the Main Controller plays in the system and the functionality of the system. Also the prototype serves as an aid in developing the design of the Main Controller.

Hardware Main controller's purpose is to control the coordination between the Sensor Control Unit, the Servo Control Unit, and the Communication Unit. The Main Controller receives input from the Sensor Microcontroller Unit and analyze the object the hand is about to grab. From the sensor information, the Main Controller directs which grasp to complete and send that information to the Servo Controller. The Main controller contains information about the set of grasps and gestures the hand is capable of completing. In order to complete a gesture, the Main Controller unit listens to any messages received from the Communications Unit for voice triggers for the main controller to trigger a certain gesture to complete. The Main Controller also listens on the communication module to update and manage the set of grasps and gestures the prosthetic hand can complete.

The initial prototype was designed by defining the requirements of the system. The prototype consists of interfacing the communication module and the communication driver. Hardware requirements for the system include the communication module, a button to start and reset the Bluetooth module, and an LED to indicate when the Bluetooth module is initializing and when it is accepting information from the hardware module. The communication hardware module passes information through UART Communication to the Main Controller.

3.1.3 Servo Motors

In order to prototype with a servo motor, a Pololu 1501MG was ordered. The size profile for this servo fits with the requirements of the 3D printed hand, in case the team decides to use these servos for the final design. A picture of the servo tested is shown below in Figure 9.

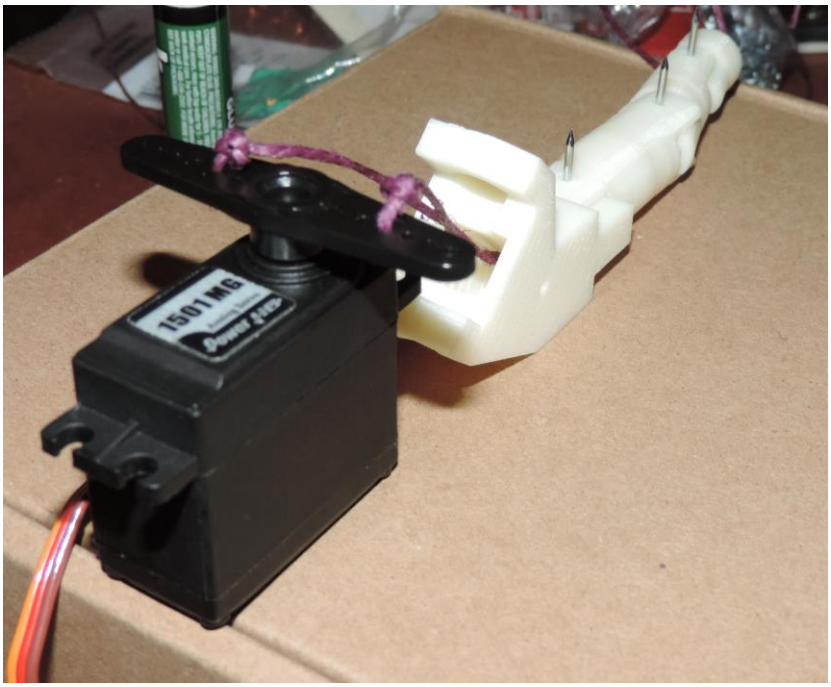

Figure 9. The servo motor connected to a 3D printed finger. The purple string acts as a tendon.

Initial testing was conducted with an ATmega328P based microcontroller with similar specifications and features as the MSP430G2553. There are servo libraries available to be used with both microcontrollers that makes controlling servo motors simple.

Setting up the servo did not cause much trouble. The ATmega328P based microcontroller was capable of providing proper voltage, current, and a ground path to power the servo enough to observe its range of motion. It is not clear how powerful the servo is until supplying it with more voltage and current. Servo motors have three different wires. The wiring for the Pololu 1501 MG servo is simple. The servo bundles its power, ground, and control lines into a 3-way connector. The colors coordinate with their function. Red being the power line, orange being the control line, and black being the ground line. One is a power supply line, usually RED. Another is the ground wire, which is either BLACK or BROWN; it is BROWN on the Pololu. The control line, which reads pulse-width modulation signals from the control unit is ORANGE, at least for the Pololu. The Pololu 1501MG is described by the manufacturer as being able to move through 90 degrees. However, with certain microcontrollers, it is possible to extend this range to 180

degrees, or close to it. From initial testing with the ATmega328P, it could be seen that its range of motion was slightly less than 180 degrees.

An attached finger is visible in Figure 11 above. Also visible is a piece of thin, but strong, colored rope. This rope acts as tendons for the finger, allowing the finger to open and close. The finger has two pathways that the string can travel through. Tightening one side causes the finger to close, tightening the other side causes the finger to open. The ends of the rope are tied to the servo, which rotates and pulls one side of the string, tightening it. In Figure 10 below, the finger is shown in flexed position.

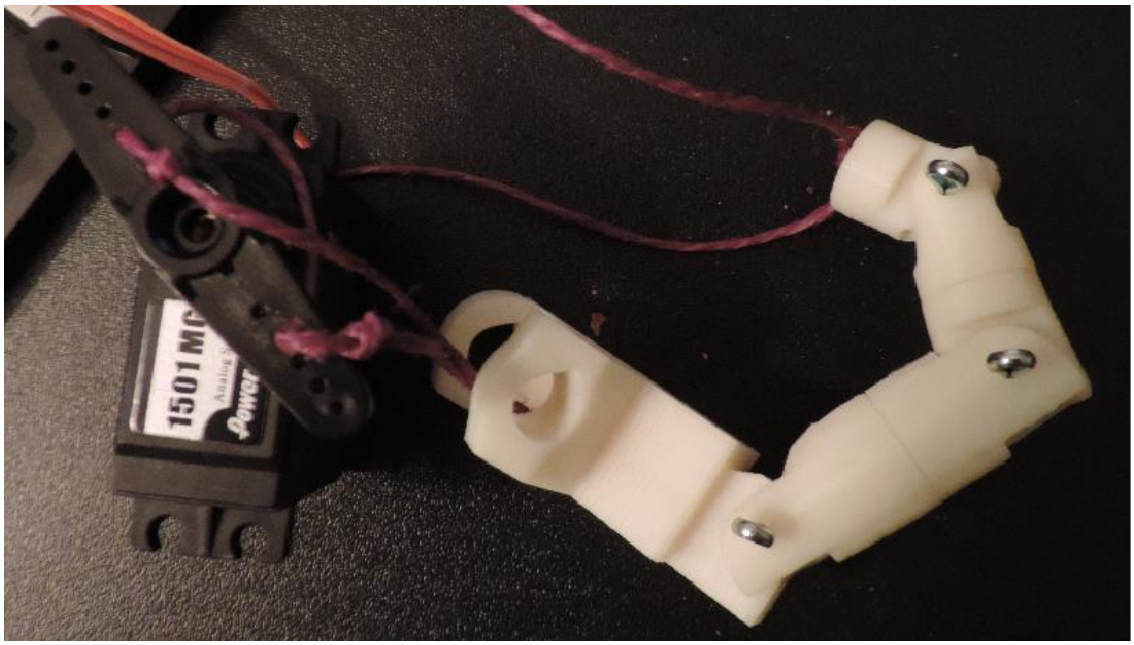

Figure 10. Servo motor with a finger in the flexed position.

From the prototype, the engineer discovered several important factors worth taking into account. First, a microcontroller cannot supply enough current to the servos to power them, except for prototyping purposes. When using the microcontroller to power the servos, it drained so much power from the microcontroller that the microcontroller lost its connection to the computer. In order to fix this, the engineer had to remove the power connection from the servo to the microcontroller. To resolve this in design, the system uses an external power source to power the servos, such as a high-current, rechargeable battery.

3.1.4 Sensors: EMG

The team decided to prototype with the electromyography sensor breakout board developed by Advancer Technologies. Setting up the device was fairly straightforward for someone with basic knowledge of soldering and electrical wiring. Additional parts were required, such as two 9-Volt batteries and snap connectors for the batteries. It was required to set up the batteries in such a

configuration so that they could supply the board with a +9 Volts and a -9 Volts. Figure 11 describes the configuration.

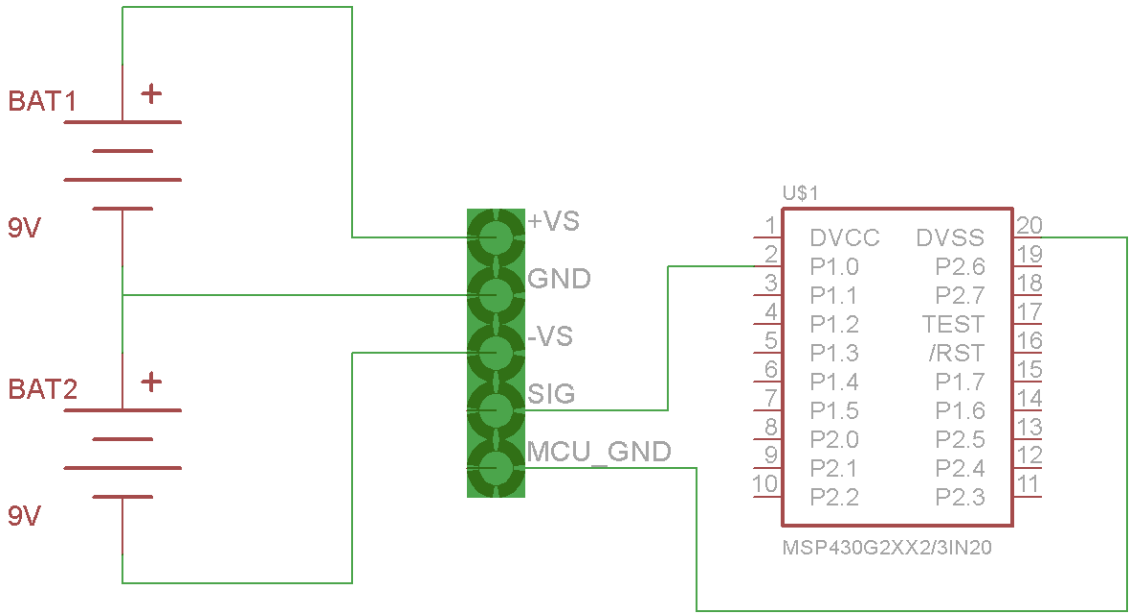

Figure 11. How to wire the EMG sensor to its power source and the MSP430G2553.

Once the wiring was complete, it was possible to do initial testing on the board. Observing the output from the EMG sensor would be important for designing and calibrating the sensor controller algorithm. With the electrodes placed on one of the team members' arm, the team observed the values read and printed onto the computer via serial communication. A screenshot from the serial monitor shows the results observed from an arm being flexed while the electrodes were attached.

The serial monitor showed a spike in the magnitude read from the signal output of the EMG sensor. This spike is correlated with the EMG wearer flexing his/her arm muscle. The values of the readings spiked from about the 180s to the mid-300s at that time. When the user relaxed their arm, the readings began to fall into the 200s and, eventually, back into the 180s range.

Depending on where the electrodes are placed, one can observe readings from different muscles. In particular, the team is looking to sense when the user desires to grasp an object. However, the muscles involved with that action are closer to the hand, where the user most likely does not have muscles in that area. An alternative would be to place the electrodes closer to the bicep area, which is a larger muscle that could produce a stronger EMG reading.

3.2 Software

Multiple iterations have been created to prove the main software concepts and requirements for this project are achievable. These software modules prototype important parts of the IPPA's software system. This is very important to understand how to interface and control the various hardware components of the IPPA's system, and is very important for the decisions taken in the final design of the software system. The software prototype modules that were developed are the Servo Controller Algorithm, Teaching Mode, and the Mobile application.

3.2.1 Servo Controller Algorithm

The servo controller is responsible for controlling the fingers of the prosthetic. It receives directions from the system controller related to positioning the fingers correctly. The servo controller algorithm consists of three, major parts:

1. Receive communications from the system controller
2. Compute the appropriate positioning of each servo
3. Set the servos into the computed positions

The communications from the system controller is sent via UART to the TX and RX pins of the servo controller. These communications is the positioning information for the servo controller. The system controller is responsible for telling the servo controller what gesture to create, when to release a grasp based on sensor data, and how to position the fingers during a user teaching session.

During prototyping, it was discovered that there would need to be an expansive set of global variables in order to store positioning, speed, and position updating information. A servo controlling library exists that automatically converts a position set, in degrees, to the appropriate pulse-width modulation signal required to set the servo to that position. This allows the team to prototype and program the algorithm with less time spent on setting a hardware clock to create PWM signals.

3.2.2 Teaching Mode and Mobile Application

As part of the mobile application and main MCU prototype, a switching mechanism has been implemented at the software level. The two possible modes of operation for the IPPA are: teaching mode, and autonomous mode. The teaching mode allows the user (amputee) to change settings, hand gestures, and gesture triggering mechanism. This mode is not engaged when the user is using the voice command triggers, either through the mobile application or the (if available) arm system.

A status variable keeps track of what mode is currently running in the Main System Controller. When the IPPA is operating in the teaching mode the sensor information is ignored by the Main System Controller, since the user has full control

of the hand position through the mobile application interface. In this mode the user is allowed to test pre-programmed gesture as well as previously saved gestures. When the user selects a gesture to demo, the arm performs that gesture. In the prototype the two available gestures are open and close finger.

From the research done about mobile platforms (see section 2.2.3), it was determined to use the Android platform for the prototype. This provides a quick insight to the possibility of developing all necessary features for the IPPA in the Android platform. The prototype requires Android 4.0 as the minimum version which is supported by the IPPA application. This major version demands mobile devices with certain hardware requirements, which is needed for the application to run smoothly.

3.2.2.1 Graphical User Interface (GUI)

In this section, the Graphical User Interface for the prototype is discussed. Since the prototype of the mobile application does not include all the features and functionality of the full application, a smaller number of pages were designed and developed. For this prototype there is no capability to create new gestures or modify the existing gestures. The features and functionality implemented for the mobile prototype include:

- Recognize if the IPPA system is currently paired with the smartphone
- Provide instructions on how to connect the device
- Confirm that the user wants to change the arm mode
- Indicate the IPPA system to change to the Teaching Mode
- Receive confirmation from the IPPA system of mode change
- Send gesture information to the arm
- Indicate the IPPA system to temporally store the gesture
- Trigger the performance of a gesture (used for both voice commands and gesture replay)

The graphical user interface (GUI) is as simple as possible, but its main infrastructure is used later in the final design. Table 7 lists the components, their functionality, and the page number where they belong.

The user is only able to do voice commands and enter Teaching Mode if the phone is paired with the IPPA system. If the phone is not connected or even connected to a different device then the user is presented with Page 1, where instructions are given in order to connect the devices. Page 2 gives the user the option of just doing voice commands for the arm, which is not implemented in the prototype, or entering the Teaching Mode of the IPPA.

Page Number	Component	Description
1	Text	Provides step-by-step instructions for the user to connect to the arm through their Bluetooth
2	Button 1	Voice Command button. No functionality for the prototype since this is not part of the Teaching Mode, but it will be implemented for the full mobile application
2	Button 2	Teaching Mode button. Entry way to the Teaching Mode.
2	Pop-up Dialog	Will confirm that the user wants to enter the Teaching Mode
3	Tab	Allows the user to switch between creating a gesture and replaying a gesture in the arm.
3	List 1	Provides a list of gestures stored already in the arm. Every element in the list could be selected for playing the gesture
3	List 2	Provides a list of gestures stored in the phone. Every element in the list could be selected for playing the gesture
3	Pop-up Dialog	Will confirm that the user wants to demo a selected gesture

Table 7. UI components used for the prototype application.

For the prototype the 3 pages have been designed and mockups were created. In Figure 12 the first 2 pages are shown.

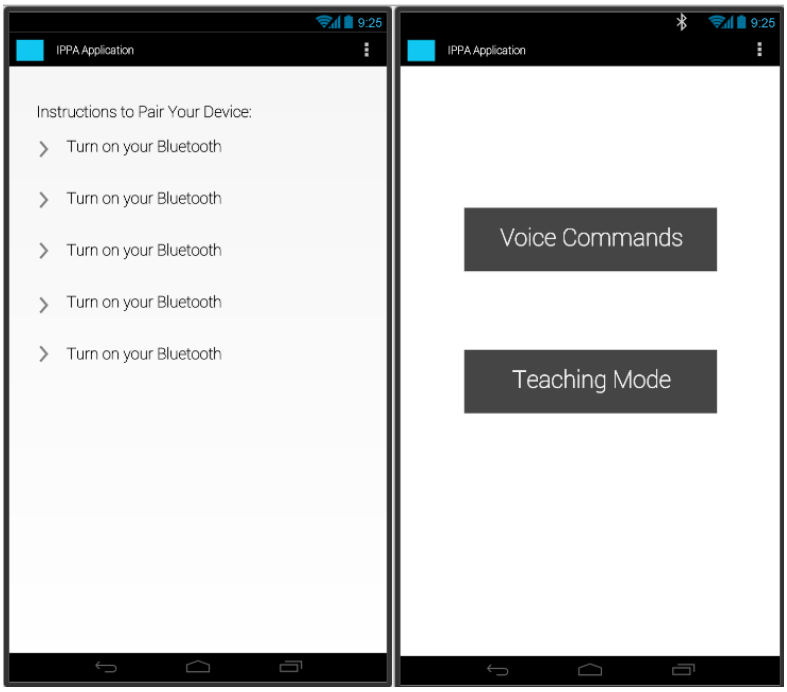

Figure 12. Left: Entry page with instructions for connection. Right: Page available once connected, displays options for the user.

In Figure 13 pages 2 and 3 are shown. In page 2, if the user selects to go into the Teaching Mode, a dialog is displayed to allow the user to confirm the selection or cancel it. This precaution is needed, since Teaching Mode is going to change the behavior of the IPPA. Page 3 presents the user with two tabs: Create Gestures and Demo Gestures. In this prototype the Create Gestures' tab is not implemented. In the Demo Gestures tab two lists are displayed: one with the gestures that are currently stored in the arm; and another one with the gestures stored in the phone. If the user decided to demo a gesture in either list a pop-up dialog confirms this intention or cancel the demo.

This design is sufficient to test the core capabilities needed from the Android platform for the final mobile application that add to the IPPA system.

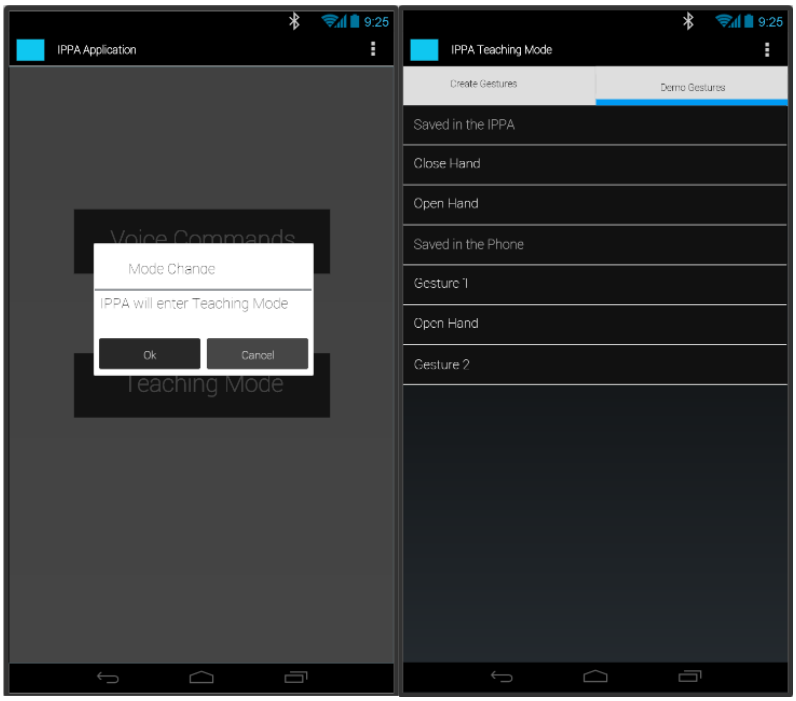

Figure 13. Left: Displays the dialog to confirm entering the Teaching Mode. Right: Example of possible gesture list in the Demo view.

3.2.2.2 Connection with the Mobile Application

Research was done in order to determine which communication technology was IPPA project (see section 2.2.4.). From the comparison of Bluetooth and Wi-Fi, the team of engineers decided to use a Bluetooth component and prototype a communication solution between the Main System Controller and a mobile application. By doing this, the team is able to evaluate the performance of the Bluetooth solution for the IPPA. In this case, the Bluetooth component acts as a slave and the mobile application (phone device) acts as a master.

To establish the connection, the Android Bluetooth network stack APIs are used. These APIs allow the application to:

- Scan for other Bluetooth devices
- Query the local Bluetooth adapter for paired Bluetooth devices
- Connect to other devices through service discovery
- Transfer data to and from other devices

For this prototype, once the connection is establish the user is able to demo a gesture. This proves the possibility of implementing all other described features of the IPPA (see section 1.2.). Table 8 lists the steps followed by both, the Mobile Application Software and the Main System Software, which are performed to demo a gesture while in the IPPA is in the Teaching Mode.

Step	Description
1	Mobile application first sends an encoded message to the IPPA Main System with the intended action to perform (i.e. demo gesture, add new gesture, etc.)
2	Main System confirms that it is ready to receive the gesture information
3	Mobile application transmits the gesture information in the IPPA's encoded format
4	The information gets copied to a temporary memory space
5	The Main System Software triggers the gesture

Table 8. List of steps to demo a gesture.

Once the application is launched, the application checks if the smartphone is currently connected to the Bluetooth device. If not, the user is provided with instructions to connect their smartphone to the Bluetooth device. After the smartphone has been paired with the IPPA system, the application allows the user to interact with the arm (see section 4.5 for more details about the application).

4 Design

This chapter contains the design efforts dedicated to this project. The design was a crucial aspect to this project because it became a great reference when the team developed the IPPA system while meeting all of the functionality previously defined. This chapter details the software and hardware design of the Main System Microcontroller, the Servo Microcontroller, and the Sensor Processing Microcontroller. This chapter also includes the design of the mobile application, the Power System, and the 3D printed arm.

4.1 System Controller

System controller's purpose is to control the coordination between the sensor control unit, the servo control unit, and the communication unit. The main controller receives input from the sensor microcontroller unit and analyze the object the hand is about to grab. From the sensor information, the main controller directs which grasp to complete and send that information to the servo controller. The Main controller contains information about the set of grasps and gestures the hand is capable of completing. In order to complete a gesture, the Main Controller unit listens to any messages received from the communications unit for voice triggers for the main controller to trigger a certain gesture to complete. The main controller also listens on the communication module to update and manage the set of grasps and gestures the prosthetic hand can complete.

4.1.1 Overview

This section describes a high level overview of the two modes running in the Main controller: the autonomous and teaching mode. Sections 4.1.4 and 4.1.5 have a more detail explanation. The diagram in Figure 14 a, represents how the Main Controller operates. Once the device is powered on, the Main controller initializes. The Main controller initializes by initializing the UART communication between the servo controller and Bluetooth communication subsystem. Also, the GPIO pins that interfaces the sensor controller is initialized. Once the Main controller is initialized and the Bluetooth communication subsystem is initialized, the Main Controller starts in autonomous mode.

To illustrate the hardware requirements and software requirements, a call flow diagram, a data flow diagram, and a software flowchart is used. The Call Flow Diagram illustrates the high level design software modules and hardware modules and their interactions. A data flow diagram shows the format of the input data, how it is processed through different hardware modules, and illustrates a high level passage of information. The pseudo-code flowchart gives a high level description of all the software modules, how they interact with the hardware modules, and the algorithmic process they entail during run time. The System Controller Call Flow

Diagram, in Figure 14 b illustrates the high level design of software modules and hardware modules and their interactions.

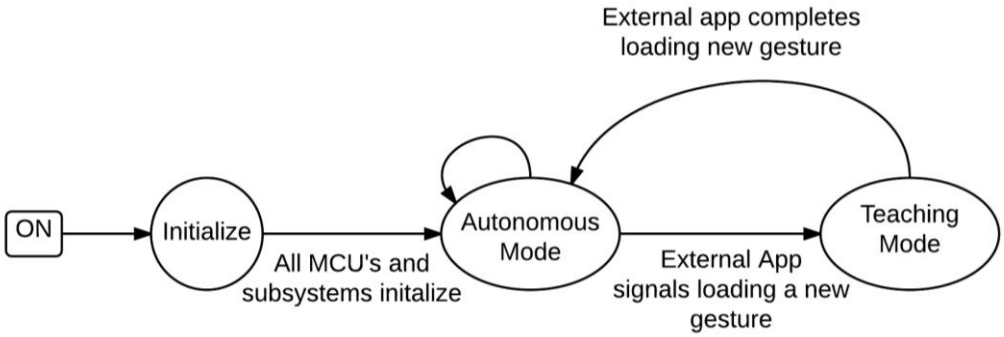

Figure 14 a. System Controller Mode Flow Diagram

The servo controller unit is implemented in an ATmega328P. This board is a 16 bit Reduced Set Instruction Count Architecture that is capable of UART, I^2C, or SPI communication. The Bluetooth module is implemented using a HC-06 Bluetooth Module that can communicate using UART.

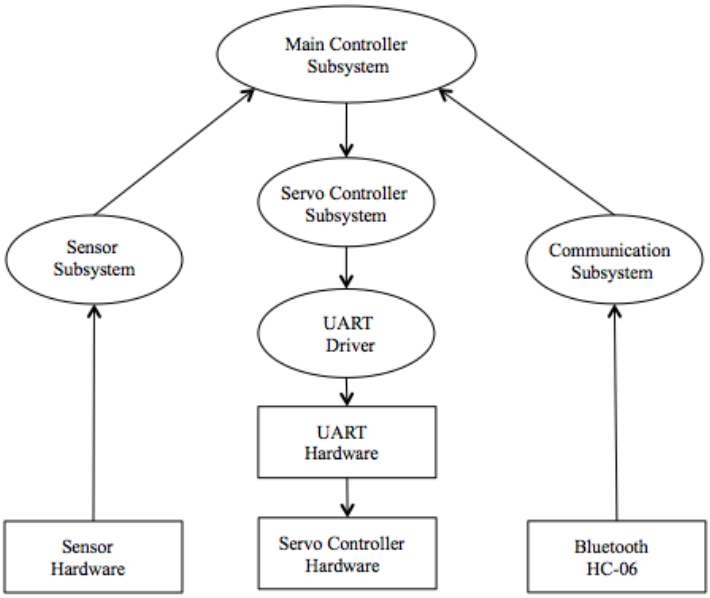

Figure 14 b. System Controller Call Flow Diagram

Following is a list of the members and functions that is used to handle all of the System Controller's functionality:

- TEACH_MODE_NEW:boolean
- TEACH_MODE_LOAD:boolean
- TEACH_MODE_TEMP:boolean

- AUTONOMOUS_MODE:boolean
- gestures:Gesture[5]
- gesture:defaultGest
- int:MC_tempServoPos[5]
- int:MC_defaultServoPos[5]
- char:MC_servoPos[3]
- uint32_t g_ui32SysClock
- Initialize()
- +MC_initalizeEEPROM()
- +MC_resetEEPROM()
- +retreiveTask(int index, uint32_t arr[])
- +initalizeTask(index index, uint32_t arr[])
- +loadAllGestures(uint32_t arr[])
- +MC_initalizeAllGestures()
- +MC_initalizeDefaultGesture()
- +MC_ResetServos(int pos[])
- +MC_SendServoPos(int pos[],int end[])
- +UARTIntHandler(void)
- +StoreGest(Gesture g)
- +removeGesture(int id)
- +SendGestureToServo(int id)

4.1.2 Servo Controller Subsystem

The purpose of this system is to transfer resolved position from a voice command or EMG signal to the servo controller unit. The main controller contains in its memory a set of gestures and grasps that the hand can complete at any time. It is important to know that every finger is controlled by a servo motor. A servo motor runs by receiving a position in degrees to rotate to and completes the rotation accordingly by pulse-width modulation. Thus defining an array where every element contains a servo position for each finger is required to generate a grasp. A grasp and a gesture also requires a string that is the name of the gesture. The name is used to compare the voice command received from the external application to trigger a gesture. Below is a description of a grasp data structure and a gesture data structure. Table 9 shows the gesture and grasp data structure and the data elements and types inside each structure.

Gesture	Grasp	Description
unsigned_ 32_int Servo Position Array[5]	unsigned_ 32_int Servo Position Array[5]	array -every element contains a servo position for each finger is required to generate a grasp
char name[20]	char name[20]	Name of gesture/grasp
	Char 8 bit signal	For grasp only, contains analog signal to determine trigger from EMG

Table 9.Grasp and Gesture data structure

Note that the range of each element in the servo position is the range of rotation the servo can complete, which is between 0 – 178 degrees. The servo controller subsystem is illustrated using a pseudo code flowchart. Figure 15 shows the three main modules in the servo subsystem.

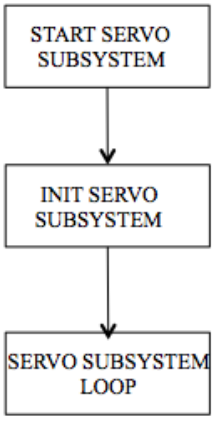

Figure 15. Servo Subsystem pseudo-code flowchart

The diagram in Figure 16 illustrates the START SERVO SUBSYSTEM MODILE. This module describes how the servo subsystem is initialized. The modules that initialize the servo subsystem is the INIT SERVO SUBSYSTEM and the SERVO SUBSYSTEM LOOP module.

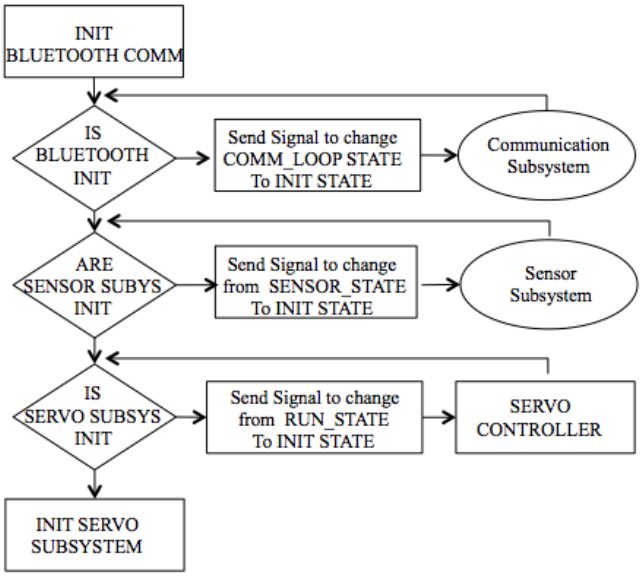

Figure 16. Start Servo Subsystem module

The diagram in Figure 17 describes the design of the INIT SERVO SUBSYSTEM module. This is comprised of a module to initialize UART communication, a module to test the servo communication, and the main module, which communicates to servo controller to trigger a grasp or gesture. The hardware switch specifies how the switch triggers the servo subsystem to re-initialize.

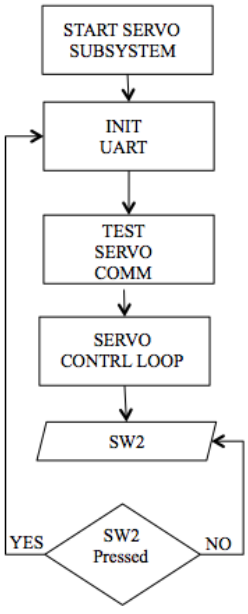

Figure 17. INIT SERVO subsystem pseudo-code

The diagram in Figure 18 describes, INIT UART MODULE, which describes how the initialization of the UART communication between the main controller and the servo controller. If the servo module does not initialize, some information alerts the user that there is a problem with the UART communication.

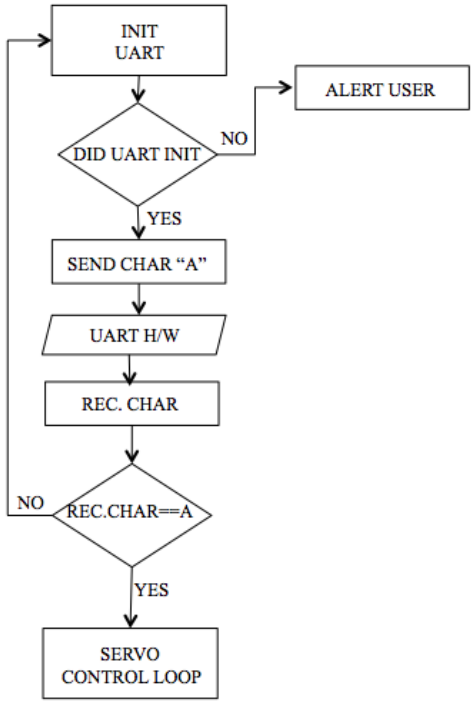

Figure 18. INIT UART module

The diagram in Figure 19 describes TEST SERVO module. This module describes how the main controller tests whether the UART communication to servo controller is working properly. The servo subsystem sends a basic position to the servo controller, the servo controller completes the information sent, and receives the same information back. If this fails, the system triggers to re-initialize the UART communication.

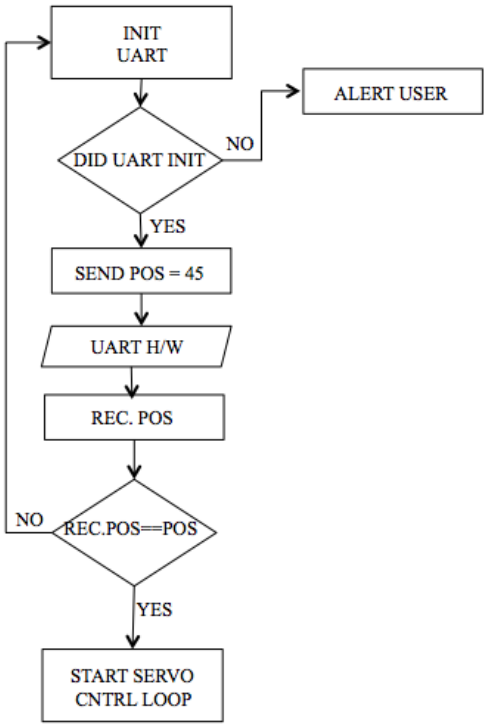

Figure 19. TEST SERVO module

The diagram in Figure 20 describes the servo control loop. The left part of the diagram on the left describes how the servo control loop waits for a signal from the start control loop. When the signal is received, the loop sets the GLOBAL_SERVO_FLAG, which sets an interrupt to run. This interrupt, shown in the right part of the diagram, is comprised of determining whether to complete a grasp or a gesture based on the information received from the sensors. Once a grasp or gesture is determined, send the information to the servo controller.

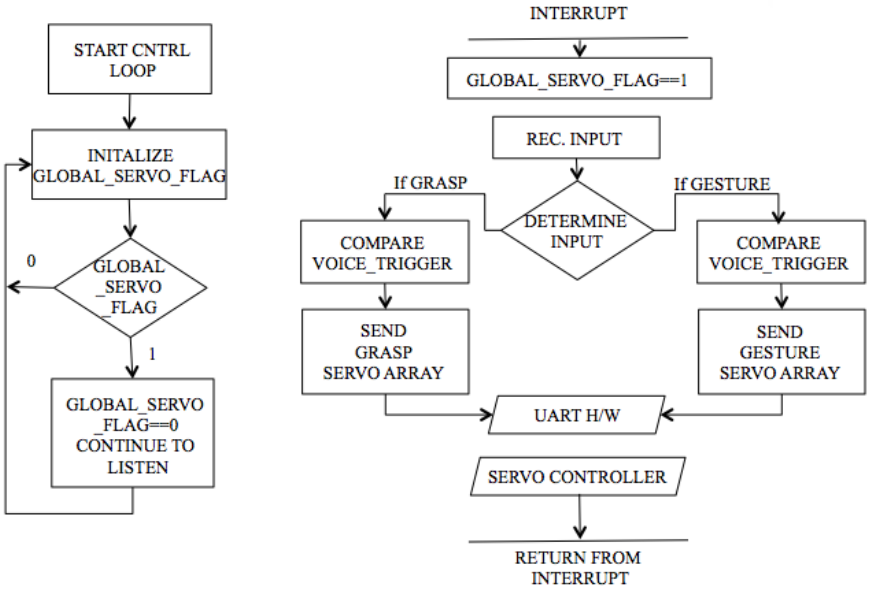

Figure 20. SERVO CONTROL LOOP module

Following is a list of the members and functions that handle all of the functionality in the Servo Subsystem:

- moveToDefaultPos()
- SERVO_ATMegahandler()
- initalizeServoController()
- receiveGesture():Gesture g
- startGesture(Gesture g)
- endGesture()
- ServoErrIntHandler

4.1.3 Sensor Subsystem

The sensor subsystem software component serves as an input to the main controller for the autonomous grasping of the prosthetic hand. The sensor subsystem receives input information from the sensor from every sensor and provide an interpretation of the prosthetics status and surroundings. Sensor information consists of applied pressure at specific points on the hand, the instance the user wants to start grasping or release grasp, and the distance each finger is to the object the user wants to grab.

To illustrate the hardware requirements and software requirements, a call flow diagram, a data flow diagram, and a software flowchart are used. The Call Flow Diagram illustrates the high level design software modules and hardware modules and their interactions. A data flow diagram shows the format of the input data, how it is processed through different hardware modules, and illustrates a high level passage of information. The pseudo code flowchart gives a high level description of all the software modules, how they interact with the hardware modules, and the algorithmic process they entail during run time. Because this is the subsystem that is running all the time, there is not end state in this system, the sensor subsystem is always in a continuous loop. The diagram in Figure 21 the Sensor software Subsystem.

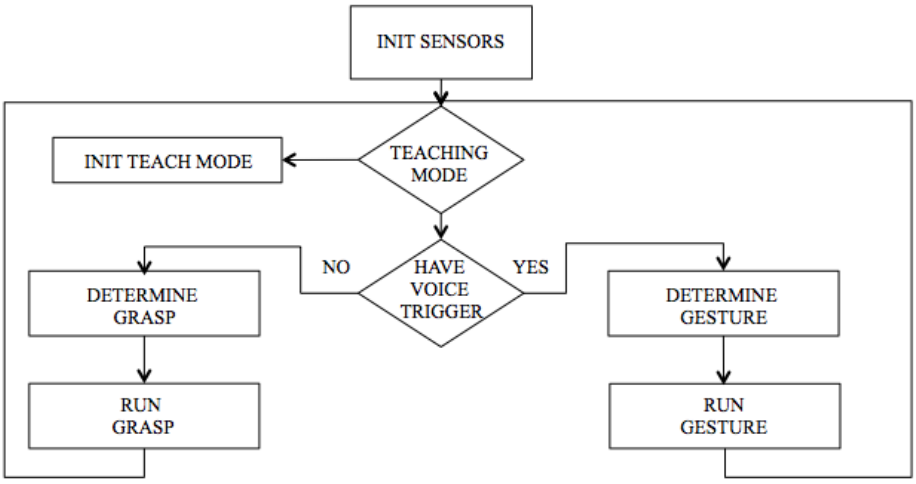

Figure 21. Sensor Subsystem pseudo-code

The Start State of the sensor subsystem is when the entire prosthetic is powered on and all of the ports used in each subsystem must be initialized. For the sensor subsystem, that would mean to initialize the analog pin to read for one distance sensor, 2 analog pins must be initialized to read for each pressure FSR sensor, and one analog pin must be initialized to read in order to receive input from the Electromyography (EMG) sensor. It is important to note that all the sensors requires calibration to determine appropriate thresholds for the sensor subsystem to trigger the list of actions.

The list of actions that are completed are to have a global flag set whether to read the sensors or not. This is mentioned in more detailed in Teaching Mode section where this flag determines that the system is either in autonomous mode or teaching mode. If this flag is not set to teaching mode, the sensor subsystem checks whether the main controller received a voice trigger. The condition whether the main controller received a voice trigger determines whether the subsystem runs the DETERMINE GESTURE module or DETERMINE GRASP module.

The diagram in Figure 22 describes the run gesture module. The module begins by first reading the distance sensors. The distance sensors indicate if an object is close by. When the threshold is reached that the object is at a certain distance, the sensor subsystem waits, if the EMG sensor reaches a threshold to trigger grasp. If EMG sensor reaches threshold to indicate that the user wants to grasp, the sensor subsystem sends the action to grasp to the servo controller and the sensor subsystem immediately reads the pressure sensors.

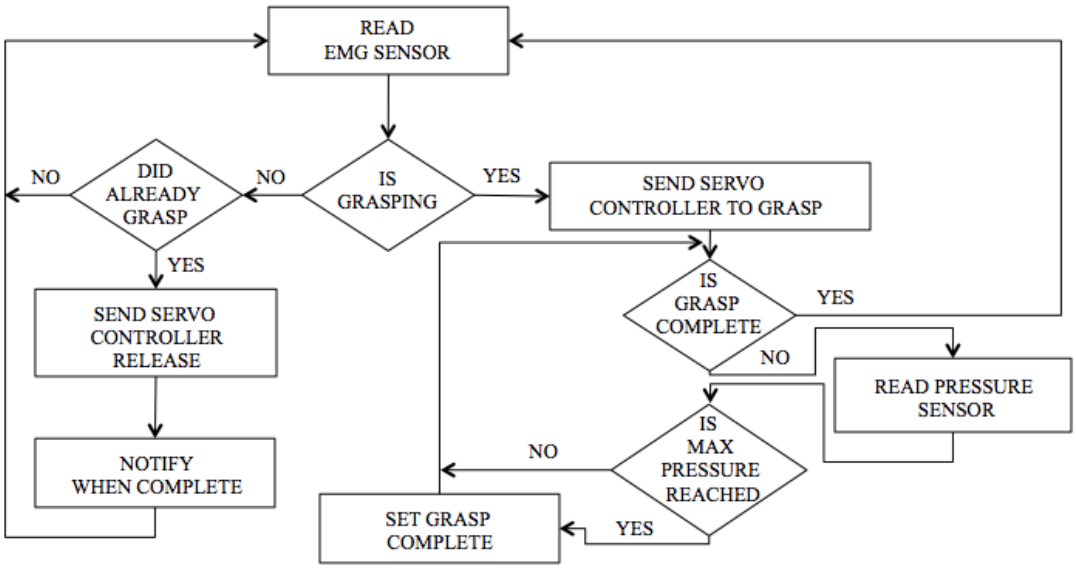

Figure 22. RUN GRASP Module

The sensor subsystem reads the pressure sensors to make sure that the hand reaches the appropriate grip pressure and also prevent the hand from gripping an object too strongly. If at any time the pressure sensors reach the threshold limit of pressure, the sensor subsystem sends a signal to the servo controller to stop grasping. The arm is now in grasping state and the sensor subsystem is reading information from the EMG sensor. A threshold value reached from the EMG sensor indicates that the user wants to release grasp. If this threshold is reached, the sensor subsystem sends a signal to the servo controller to release grasp. This cycle repeats, as the user can trigger a new grasp.

Following is a list of the members and functions that encapsulate the functionality of the Sensor Subsystem:

- int32_t : i32val
- bool : SC_TaskTriggered
- initalizeSensorController();
- SC_checkTriggerPins()

4.1.4 Autonomous Mode

This section describes how the System Controller operates in Autonomous Mode. Autonomous mode can be represented by the high-level state diagram seen in Figure 23. The diagram illustrates how the main controller is running autonomous mode. The first part of autonomous mode is the Run loop. The main controller is in a long lasting execution loop. In this loop, the main controller waits for input from the Bluetooth communication subsystem and the sensor control unit. Whenever a Bluetooth message is received an interrupt is generated to handle the Bluetooth message. When a GPIO pin is set to HIGH an interrupt to handle the information sent from the sensor controller is generated and starts the action of triggering a grasp. Once the interrupt is triggered, either interrupt enters the Execute/Grasp phase. In this phase the main controller processes the input information and triggers a grasp or a gesture.

The messages received from the Bluetooth need to be process in order to determine if a gesture needs to be triggered. Several actions can be done here, but generally the message indicates to complete a gesture or start TEACHING Mode. When teaching mode is described in the later section, there are other various events that can occur. From the interrupt that was triggered to handle the information sent from the sensor controller or external app, a temporary loop is started in the Execute/Grasp phase to execute a grasp. During this loop, the main controller sends the servo positions desired to the servo controller to tell the servo controller where to move the servos to in order to complete the gesture. Note that the main controller sends 5 sub servo positions to the servo controller in order to take 5 incremental moves to reach the full grasp. The goal of this is to allow a halt in the grasping motion in case a high pressure reading occurs, the main controller can halt the servo controller.

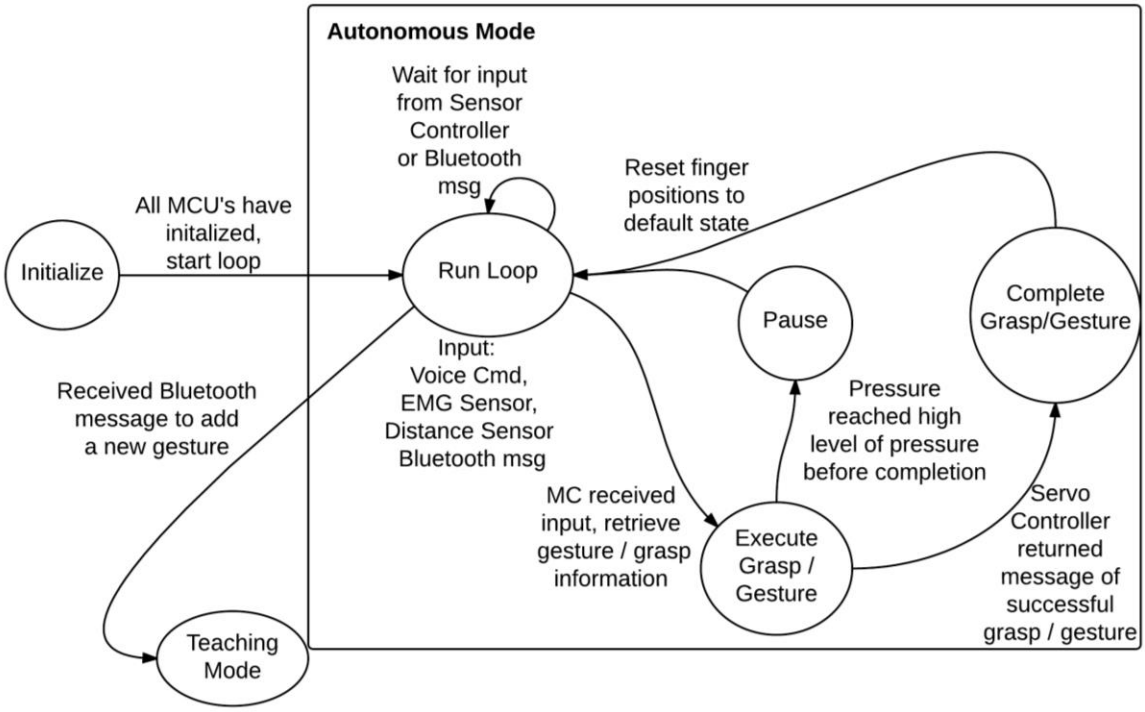

Figure 23. Autonomous mode high-level state diagram

There is a possibility that when the main controller is executing a grasp/gesture in the Execute/Grasp phase, the hand may reach high levels of pressure which would hurt the functionality of the hand completing the grasp/gesture. The main controller is monitoring the sensor controller to identify if high levels of pressure are occurring. If a high-level amount of pressure does occur, the autonomous mode moves to a Pause State. During this phase, the main controller stops the servo controller from incrementing any more to the desired gesture/grasp. Then the main controller move back to the RUN LOOP, enabling the user to complete another gesture/ grasp.

If a gesture/grasp is triggered in the RUN LOOP and the main controller is completing a gesture/grasp in the Execute/Grasp phase, if no high levels of pressure occur, then the main controller transitions to a complete grasp/gesture phase. If the main controller is completing a grasp or the gesture, the servos all reached its desired position and hold. This is when the user can lift objects up. For both a grasp and a gesture, the position of the servos is held until the EMG sensor is triggered to reset the servo positions back to open or a new gesture is triggered.

4.1.5 Teaching Mode

Teaching mode is the software component of the system that enables the user to add new gesture and grasps to the system, as well as allow the user to generate new gesture and grasps. The initial state of the system consists of 4 steps. The first step in teaching mode's start state is the mobile app sending an encoded message to IPPA with intention to perform Teaching Mode. Additional messages

indicate whether the user desires to upload a pre-existing gesture on the application, or to create a new gesture/grasp. The encoded messages are received from the Bluetooth communication subsystem, and transferred to the main controller. When the message is received, the main controller sets a flag that indicates to ignore any sensor readings from the sensor subsystem. After this step, the teaching mode is enabled and the main controller waits for more message. The operation of teaching mode is shown in Figure 24.

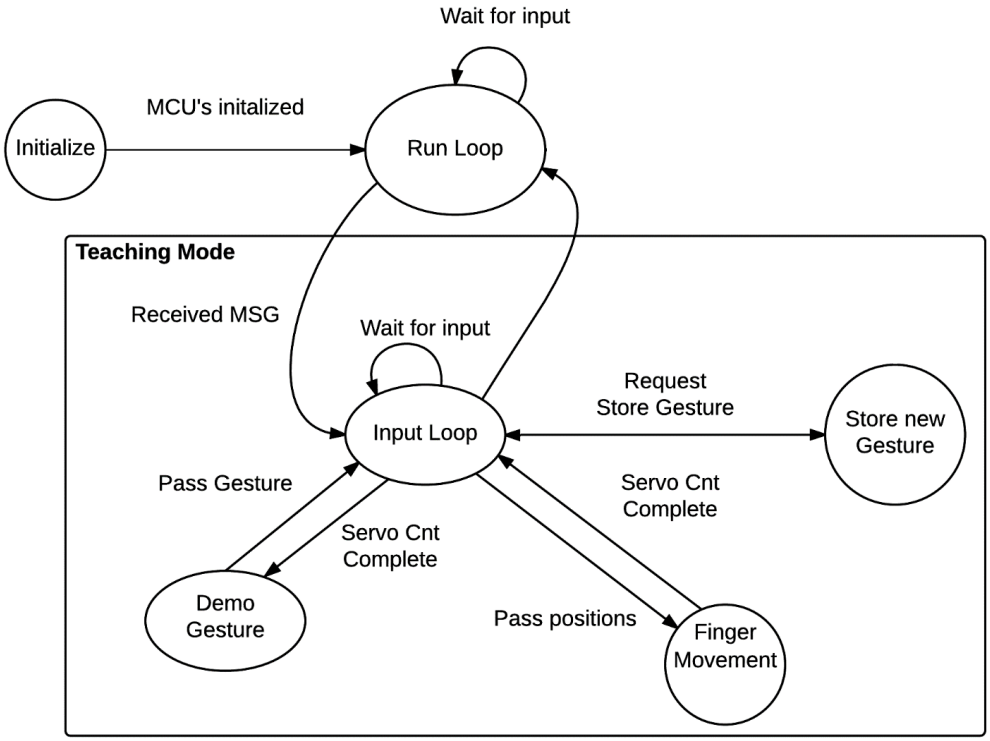

Figure 24. Teaching mode high-level state diagram

First action in the list of teaching mode actions is to load a pre-existing gesture/grasp or create a new grasp is to open up space for the pre-defined gesture or the new gesture. The end action for the teaching mode of the new gesture/grasp is that the user accepts and the gesture/grasp is uploaded. This is completed by the name of the gesture/grasp and the voice trigger of the gesture is stored. The gesture/grasp information is moved from the temporary memory space to the memory area that contains the gestures/grasps that are completed during autonomous mode. Once the move of information is completed, the Main Controller exits Teaching Mode and reverts back to the normal operation of autonomous mode. To illustrate the hardware requirements and software requirements, a call flow diagram, a data flow diagram, and a software flowchart is used. The diagram in Figure 25 is the Teach Mode Call Flow Diagram. This diagram illustrates the high level design of software modules that interacts with and hardware modules of the Bluetooth UART hardware and the Servo Controller.

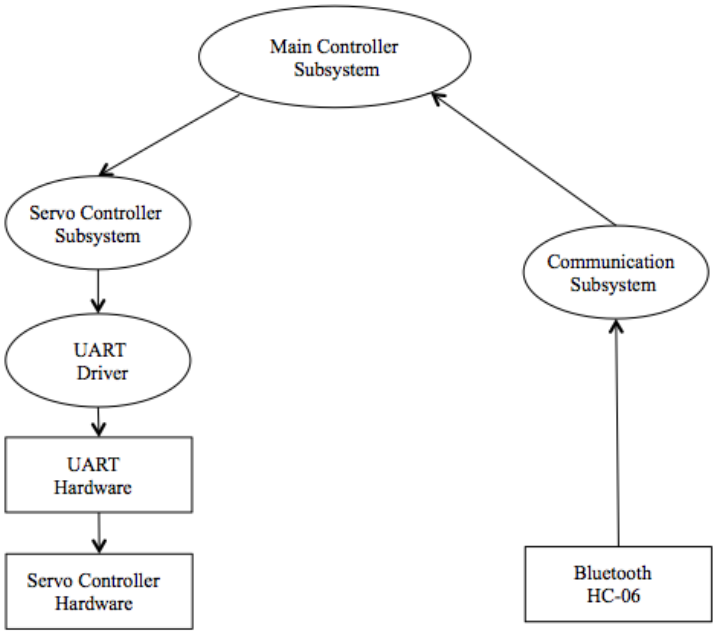

Figure 25. Teach Mode Call Flow Diagram

The diagram in Figure 26 describes the Bluetooth communication data flow diagram. This data flow diagram shows, how the data is processed through different hardware modules during teach mode and illustrates a high level passage of information form the Bluetooth hardware to the servo controller from the main controller subsystem.

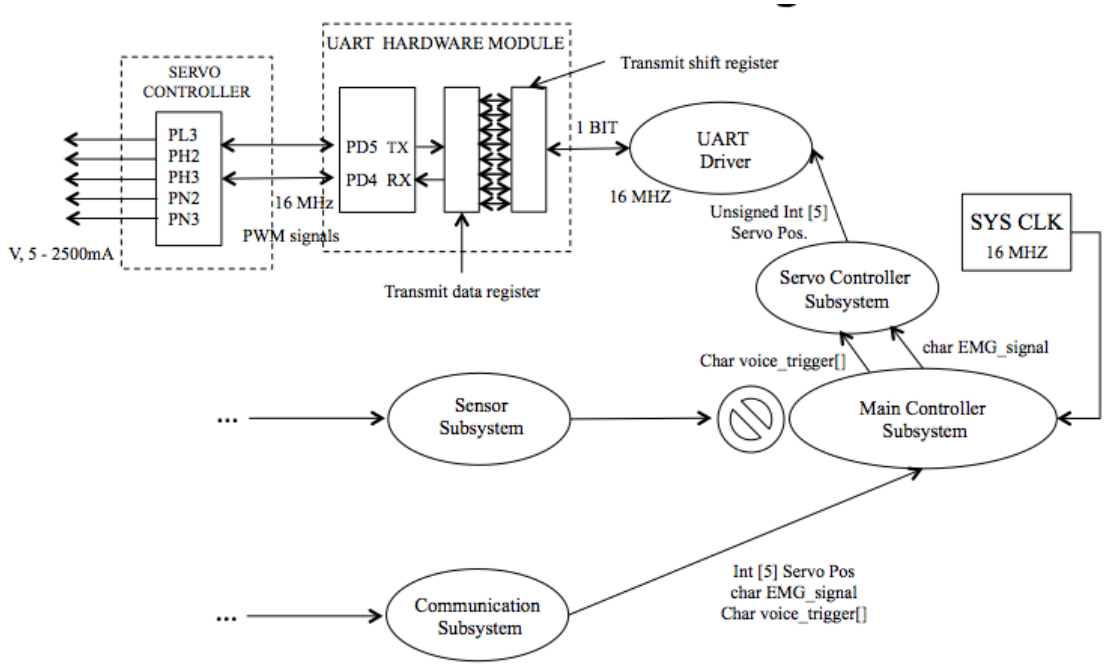

Figure 26. Teach Mode Data Flow Diagram

Figure 27 describes the Teach Mode software subsystem. The subsystem is composed of the INIT TEACH MODE Module. The module contains the initialization and the details the each mode software entails. Once that module is completed, Tech Mode ends and the main controller returns back to autonomous mode. The diagram in Figure 27 describes the INIT TEACH Mode module.

First action in the list of teaching mode actions is to load a pre-existing gesture/grasp or create a new grasp is to open up space for the pre-defined gesture or the new gesture. Next are the steps to determine if the gesture wanting to add is a pre-defined gesture/grasp. A pre-defined gesture/grasp is a gesture/grasp that already has the information containing the voice trigger needed to trigger the gesture, the name of the gesture/grasp, and the position array for each servo that controls each finger. The next step to load a pre-defined gesture in the app allow the gesture/grasp to demo it.

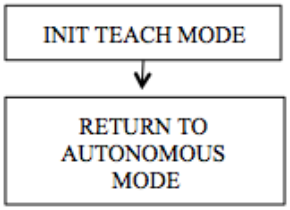

Figure 27. Teach Mode Software Module

The diagram in Figure 28 describes the teach mode control loop. The diagram on the left describes how the control loop waits for a signal from the INIT TEACH MODE module. When the signal is received, the loop sets the TEACH_MODE_FLAG, which sets an interrupt to run. This interrupt is comprised of determining whether to complete a new gesture or a demo gesture based on the information received from the external application. Once a grasp or gesture is determined, send the information to the servo controller. The determination of the demo grasp and new grasp would be similar.

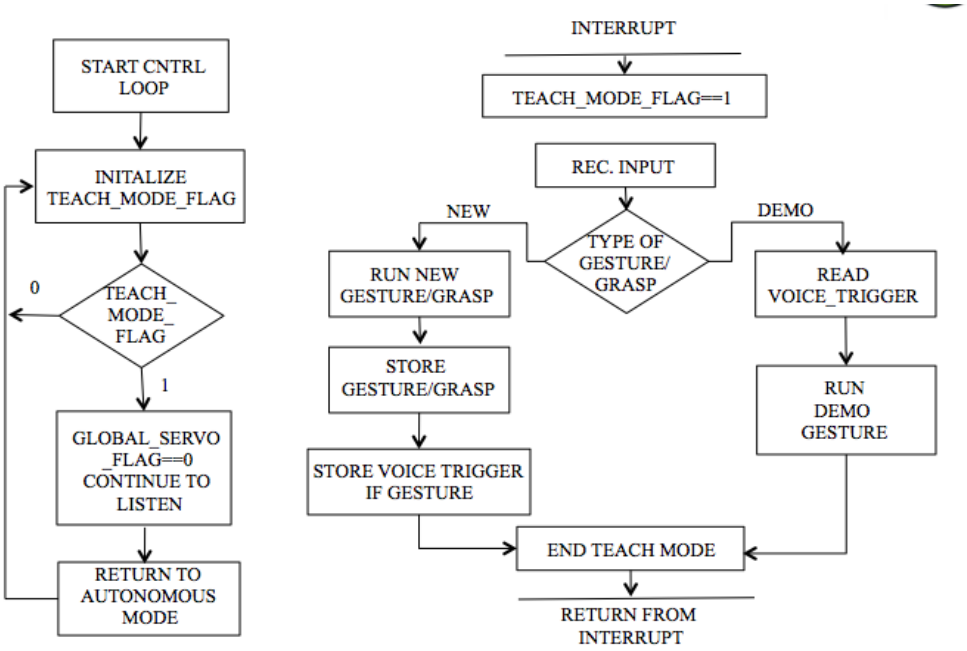

Figure 28. Teach Mode Control Loop Module

The diagram in Figure 29 describes the RUN DEMO GESTURE module. To complete the demo gesture/grasp, the mobile app transmits all the information required to run the gesture/grasp. The information is sent over Bluetooth and received in the Bluetooth communication subsystem. The information is then copied to the area that was freed in memory to store the gesture. And the main controller triggers the gesture/grasp to be completed.

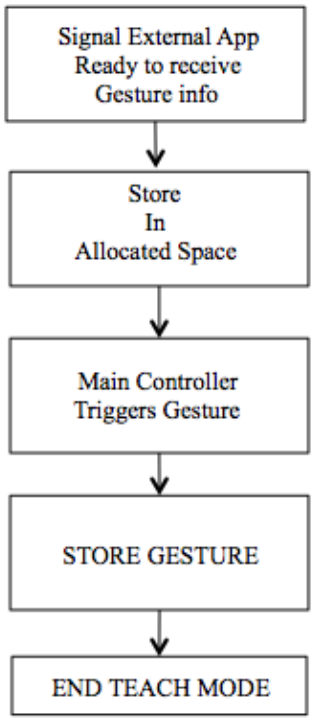

Figure 29. Run Demo Gesture Module

The diagram in Figure 30 describes the RUN NEW GESTURE module. Next are the steps to determine if the gesture wanting to create a new gesture/grasp. A new gesture/grasp is a gesture/grasp that the user interactively creates the voice trigger needed to trigger the gesture, the name of the gesture/grasp, and the position array for each servo that controls each finger. The next step to create a new gesture/grasp is the app opens up a socket communication system where the application transmit servo positions for all fingers. The finger positions are then passed to the servo controller, which executes the new position, providing real time feedback during the creation of a gesture.

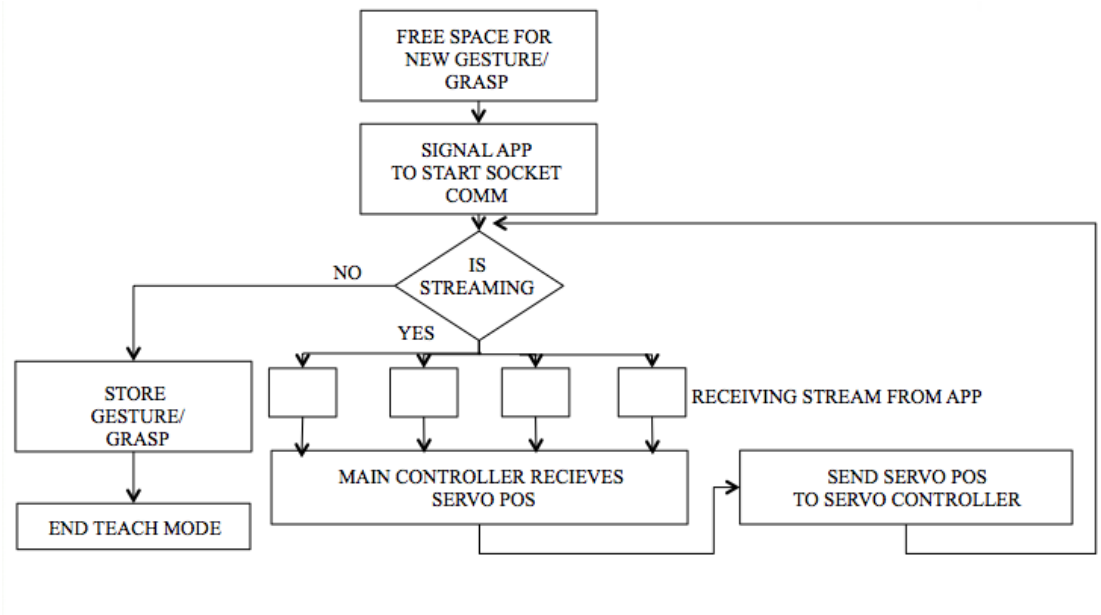

Figure 30. Run New Gesture Module

4.1.6 Bluetooth Communication Subsystem

This section outlines the overall function of the Communication System on the Main Controller and design the interface between the Main Controller Subsystem and the Bluetooth Communication Module. The System Controller subsystem interfaces and communicates with the communication subsystem.

The communication system collects messages in the format of 8bit ASCII characters that are messages transmitted from the external application. If there are any problems that occur with communication between the external application and the main controller, the user can reset the communication using the reset button on the main controller. If the user presses the reset button, the communication system switches state from COMMUNICATION STATE to INITALIZATION STATE. From here, the communication subsystem follows the steps of initialization and re-initialize Bluetooth communication.

The diagram in Figure 31 describes the Bluetooth Communication Subsystem Call Flow Diagram. The Bluetooth Communication Subsystem Call Diagram illustrates the high level design of software modules that interact with and hardware modules of the LED, Bluetooth UART hardware, and the hardware switch.

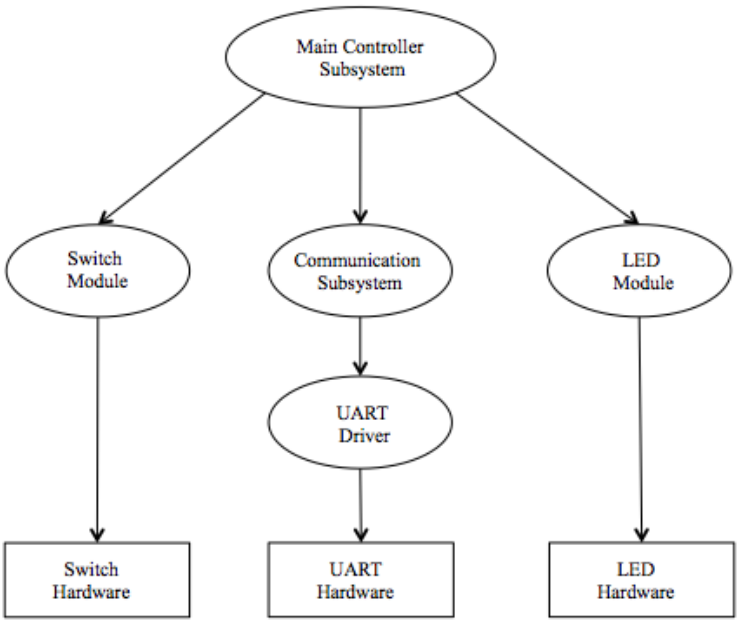

Figure 31. Bluetooth Communication Call Diagram

To illustrate the hardware requirements and software requirements, a call flow diagram, a data flow diagram, and a software flowchart is used. The Call Flow Diagram illustrates the high level design software modules and hardware modules and their interactions. A data flow diagram shows the format of the input data, how it is processed through different hardware modules, and illustrates a high level passage of information. The pseudo code flowchart gives a high level description of all the software modules, how they interact with the hardware modules, and the algorithmic process they entail during run time.

The data flow diagram, in Figure 32 shows the format of the input data of the Bluetooth communication subsystem, how the data is processed through different hardware modules, and illustrates a high level passage of information form the Bluetooth hardware to main controller.

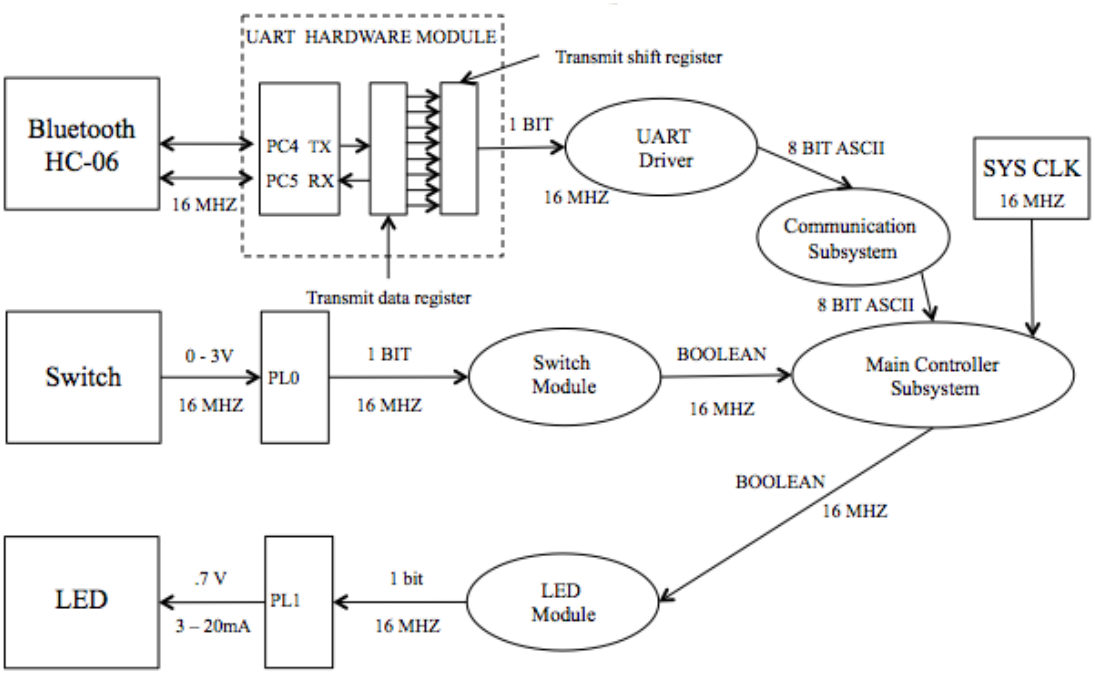

Figure 32. Bluetooth Communication Data Flow Diagram

The communication system interfaces with the Bluetooth HC-06 communication hardware module. The diagram in Figure 33 shows the Bluetooth Communication High Level Pseudo-Code. There are three states in the communication subsystem: a START STATE, an INITALIZE STATE, and COMMUNICATION STATE.

The START STATE is the state when the communication subsystem is powered on and all of the ports and processes are starting up. This state only occurs when the prosthetic is powered on and the entire system is initializing all of its ports and processes. The START STATE waits until the necessary ports are initialized and immediately transition to the INITALIZATION STATE. The INITALIZATION STATE signifies that the communication subsystem is initializing the UART serial communication to the Bluetooth module. In this state, an LED that is turned on and emits a red color to notify the user.

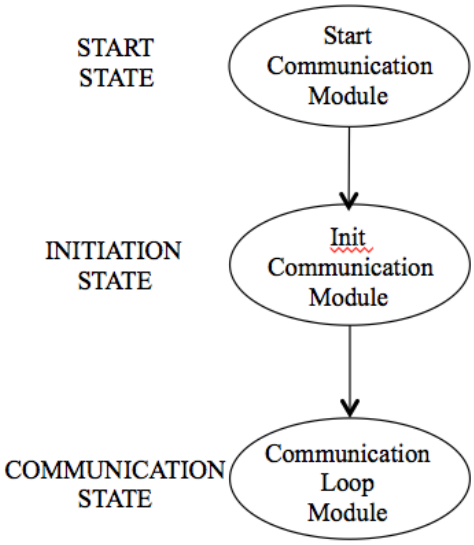

Figure 33. Bluetooth Communication High Level Pseudo-Code

The diagram in Figure 34 describes the design of the INIT Communication Module. The module calls a module to initialize the UART communication of the Bluetooth Hardware and determine if the initialization was a success or not. After the initialization, the module conducts a test to make sure the communication between the main controller and the Bluetooth module works correctly. Finally, the module completes and start the COMM LOOP module.

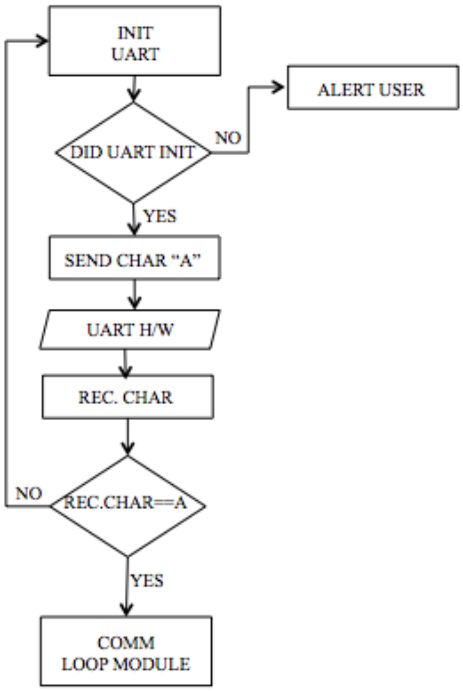

Figure 34. Init Communication Module

The diagram in Figure 35 is pseudo-code on how the Bluetooth Communication subsystem initializes communication to the Bluetooth hardware. There are two main modules that complete this function, the Init Communication Module and the Communication Loop module.

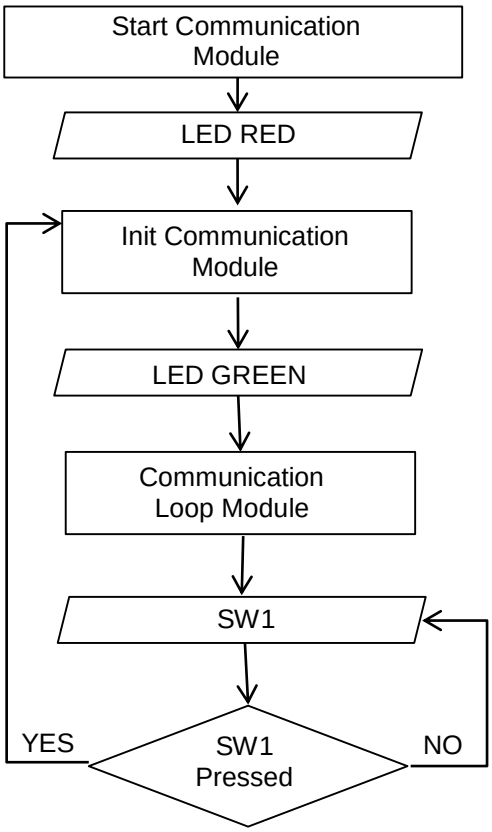

Figure 35. Bluetooth Communication High Level Pseudo-Code

When the signal is received, the loop sets the GLOBAL FLAG, which sets an interrupt to run. This interrupt is comprised of passing the information received the message from the Bluetooth signal and store it to a buffer to be processed

The diagram in Figure 36 describes the COMMUNICATION LOOP Module. The diagram on the left describes how the module waits for the Bluetooth communication to initialize to the external application and then completes a communication test by sending the character ‘a’ to the external controller. If the test passes, the module waits for a signal from the start the control loop.

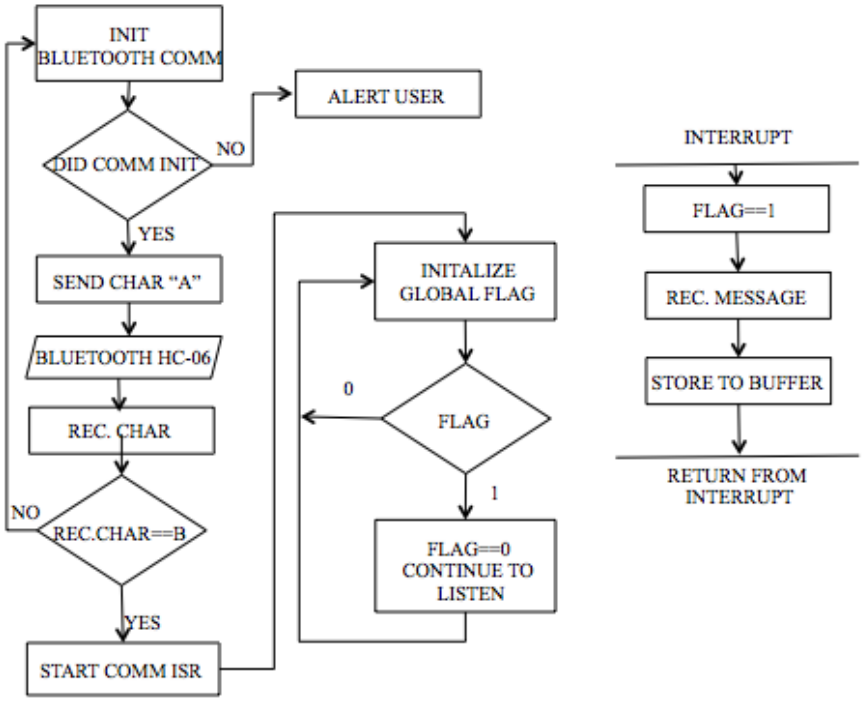

Figure 36. Communication Loop Module

4.2 Servo Controlling Microcontroller

The design for the servo controller includes the software programmed on the controller and the hardware components used. The software is responsible for processing communications from the system controller and sending PWM signals through the control lines of the servos to set their positions. The hardware for the servo controller mostly consists of the microcontroller itself, the servo motors, and the fishing lines which act like tendons for finger control.

4.2.1 Software

Before entering the loop structure and the main algorithm, two processes must occur. Setup, which involves the initialization of UART communications with the system controller and GPIO initialization; and Data Structure Initialization, which sets up the required data structures that holds information critical for controlling the servo motors.

Setup:

1. The servo controller must establish communications with the system controller via UART
2. The servo controller must set up five, appropriate GPIOs as output pins, to be used to send pulse-width modulation signals to the control lines of each servo

Data structure initialization:

1. A servo position array to store the current positions of each servo
2. A servo step size array to store the amount (in degrees) of how much to change the servo each iteration
3. A servo step speed array to store the amount of time in between servo steps
4. A servo step count to store the amount of steps before arriving at a final position
5. A new position array to store servo position settings as dictated by the system controller
6. A servo pin array to store the pin identifiers corresponding to each servo's control line
7. An array containing the last times the servo positions were updated, measured in milliseconds since the microcontroller powered on

The above data structures are initialized as global variables, as they need to be saved over continuous microcontroller cycles. The servo controller update algorithm is run inside of a loop structure, which runs indefinitely for as long as the servo controller is powered. This structure is shown as a flowchart in Figure 37.

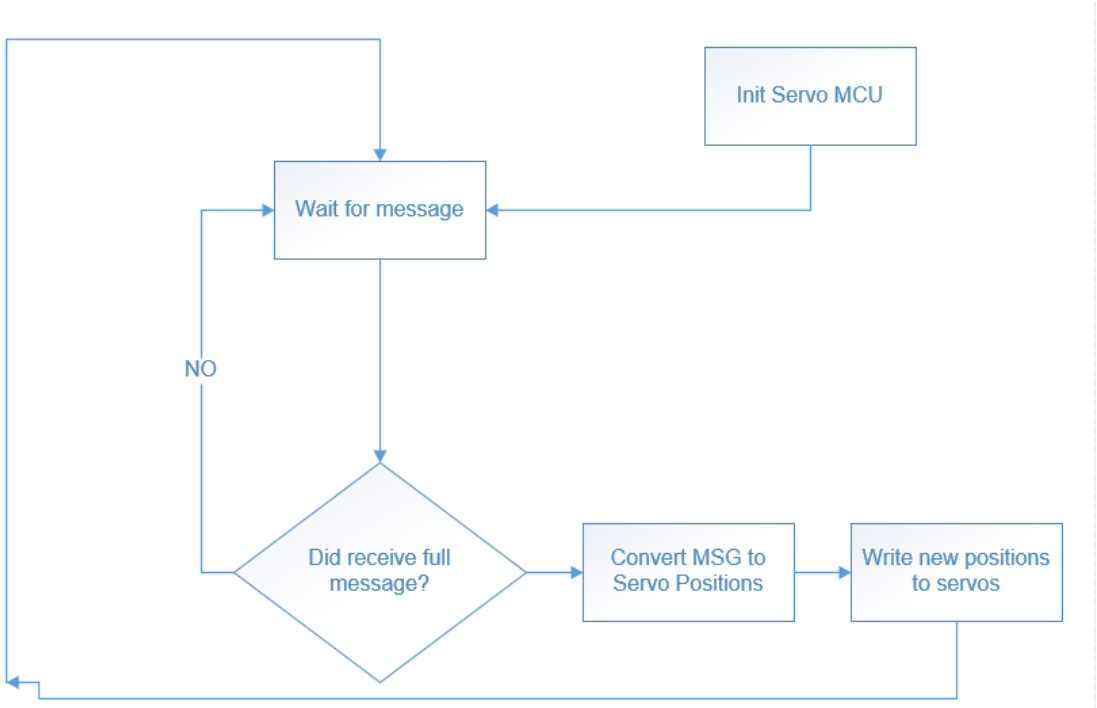

Figure 37. Flowchart showing the algorithm inside the servo controller.

In order to implement this algorithm, several function had to be written. These are listed below:

1. Initialize() – Sets up specific GPIOs as control lines for all 5 servos. It also establishes initial communication with the system controller via UART.
2. ReceivePositions() – Reads serial data from UART specifying where to position the servos
3. DetermineStepSize() – Performs calculations on the servo positioning data to determine how many degrees to move each servo by each loop iteration

4.2.2 Hardware

The microcontroller we used as the servo controller was an ATmega328P. The reason for choosing this microcontroller is because it requires little power, yet still boasts the speed and features needed to control the servos. Wiring the ATmega328P to a breadboard or printed circuit board is fairly simple, only requiring a source voltage of 5.0V, the reset pin pulled up using the voltage source, and a ground. Any code on the chip should begin to run once the chip is powered on. Figure 38 below shows the ATmega with the required power supply, ground, and reset pin wired correctly.

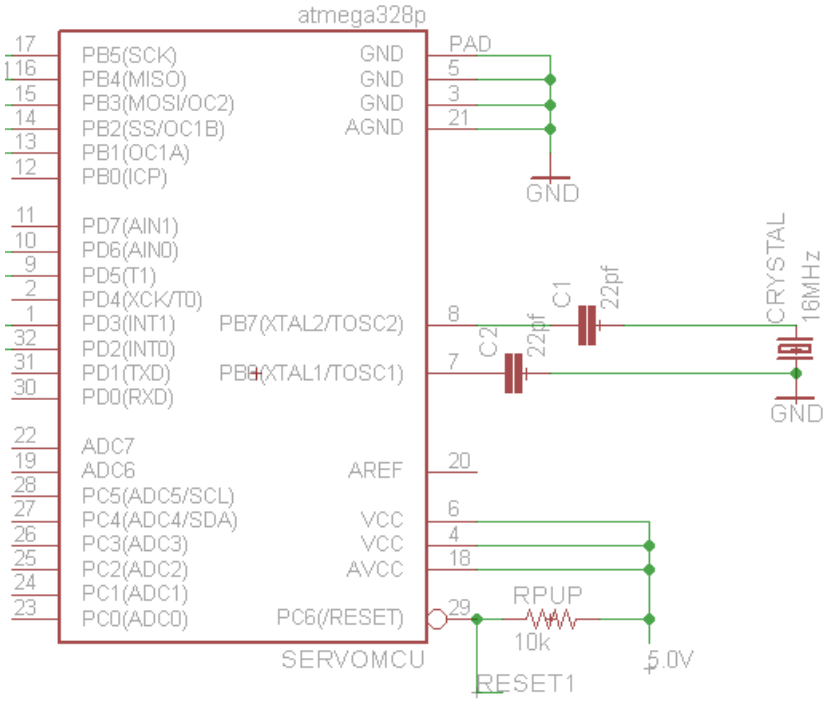

Figure 38. A schematic showing how to wire the ATMega328P on a breadboard or PCB.

The servos used in conjunction with the microcontroller are the Pololu 1501MG series servos. The strength of the servos is adequate, they are reasonably priced,

their arms can be positioned from 0 to almost 180 degrees, giving the range of motion needed, and the servos do not consume more power than the system is capable of supplying.

As far as controlling the servos, the control lines of each servo are connected to a unique output pin on the ATmega. They are also wired in parallel, so that they all receive the same voltage, but may receive more or less current depending on their current motion. Figure 39 below shows the schematic of how we wired the servos to the ATmega. The schematic symbol of a variable resistor was used to illustrate the behavior of a servo.

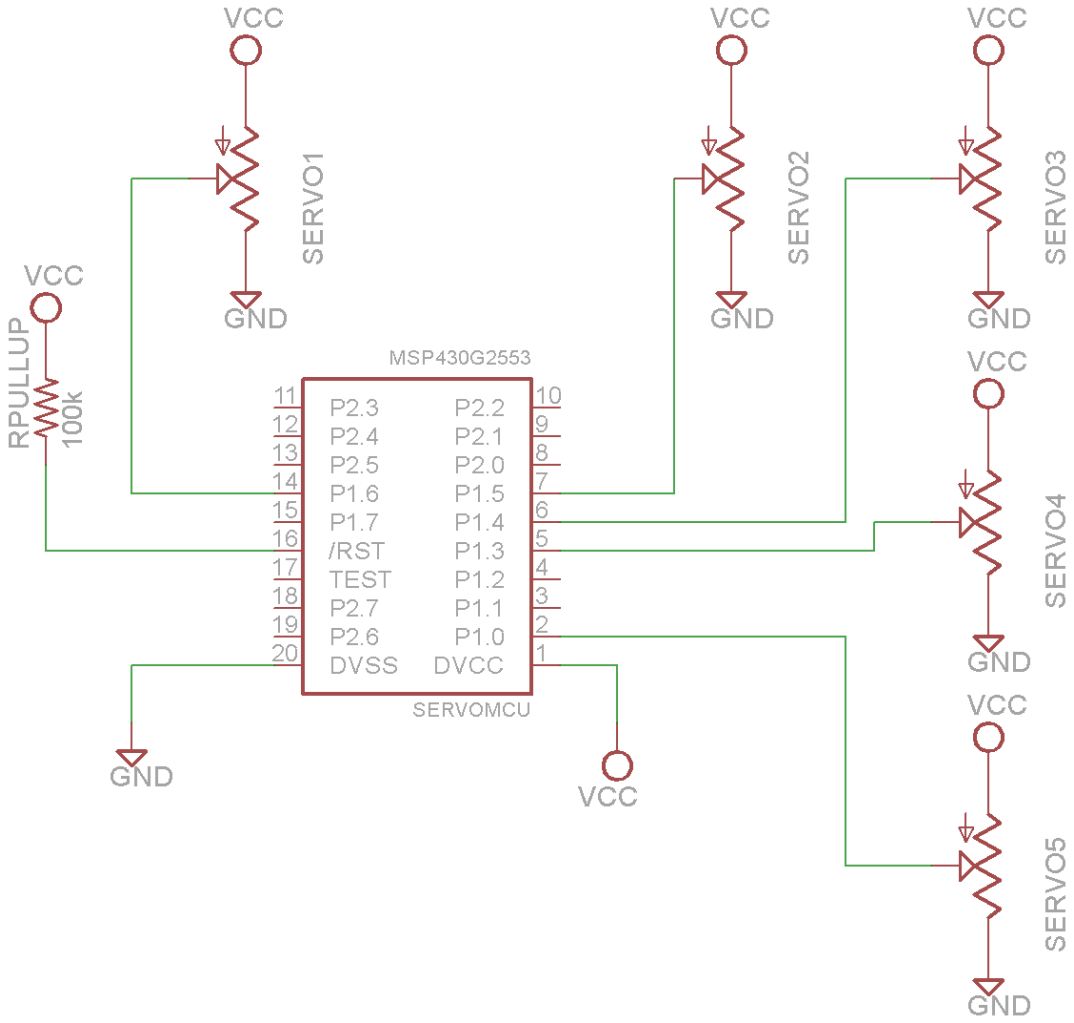

Figure 39. A schematic of how to wire five servos to the ATmega328P.

4.3 Sensor Processing Microcontroller

For the IPPA system, the team decided to implement three different sensors into the prosthetic arm. An electromyography sensor, to allow the prosthetic to interface with the electrical impulses generated by the user. Multiple force sensing resistors were implanted into the palm and fingers to determine the amount of pressure the

hand is generating on an object and to sense when a strong grip has been established. And a passive infrared sensor, to detect when an object is near the hand.

The sensor processing microcontroller is responsible for interpreting the inputs from all of the above mentioned sensors. In addition to processing these incoming signals, the microcontroller also communicates to the system controller the results of these computations.

4.3.1 Software

The sensor processing microcontroller required software designed to cycle through the three, different sensors and process the incoming data. Upon interpreting that data, the results are sent to the system controller. Like the servo controller, the sensor processing microcontroller enters a setup phase and initializes any required data structures before entering the main, repeating loop.

Setup:

1. The sensor processing microcontroller must establish communication with the system controller
2. The microcontroller must establish GPIOs 1.6, 2.3, 2.4, and 2.5 as output pins, to control the select lines of the multiplexer
3. The microcontroller must establish GPIO 1.7 as an input pin, to read the incoming data from the multiplexer
4. The microcontroller must read several values from the EMG sensor to determine what data a ‘relaxed’ arm provides

Data structure initialization:

1. An array to store the electromyography sensor history to computer the average over time
2. A cutoff variable for muscle relaxed state versus muscle flexed state
3. An array to store the voltage detected from the force sensitive resistor circuits
4. A conversion table or function to convert between resistance and force (in grams)
5. An array to store the voltages read from the passive infrared sensor over time
6. A threshold to determine if an object is near the hand or not

Like the servo microcontroller, the above variables are declared as global variables. Then, the microcontroller enters its infinitely repeating loop structure. The loop structure contains three, major sections of code that deal with the three, different sensors integrated into the hand. In Figure 40 below, the general algorithm is shown.

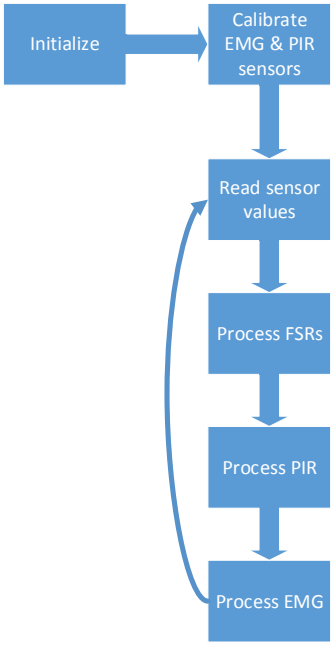

Figure 40. A flowchart of the algorithm inside the sensor microcontroller.

The algorithm described above requires several functions to be written.

1. Initialize() – Sets up the appropriate GPIOs as input and output pins. It also reads several values from the EMG sensor to establish a 'relaxed' state. Lastly, it establishes initial UART communication with the system controller.
2. ProcessEMG() – Reads a new value from the EMG sensor, determine the average of the last 5 readings, and determine if the average has risen above the discovered threshold
3. ProcessFSR() – Reads the current value on the FSRs, then runs a formula on the detected voltage to convert the reading to a resistance value. It uses a conversion formula to convert the discovered resistance to a force (in grams). The result is stored in the PIR buffer.
4. ProcessPIR() – Reads the voltage on the PIR circuit and determines if the reading is above a threshold.
5. TransmitData() – Sends the processed sensor readings to the system controller via UART.

4.3.2 Hardware

The team decided to use an ATmega328P as the sensor processing microcontroller. It has a variety of GPIOs for use when listening to incoming sensor data. For the electromyography sensor, the team chose to use the Advancer Technologies Muscle Sensor v3. It features an adjustable gain knob, a wide range of supply voltages, comes with electrodes, and showed promising results in the research prototype. What makes this sensor slightly difficult to use is that it requires a positive and negative power supply. However, as shown in Figure 11,

section 3.1.4, the method of wiring 9V batteries to supply a +9 and a -9 volts is fairly simple.

While there are many passive infrared motion sensors available commercially, the team opted to construct one. The reason for this is because they are simple to construct and have a smaller footprint than the commercial versions. Size is important to fit in the hand without obstructing the hands ability to perform tasks such as grasping and lifting. The PIR was constructed from a 470Ω resistor, a 47nF capacitor, an infrared emitter, and an infrared sensor [11]. The schematic below, in Figure 41, shows how to construct the PIR sensor.

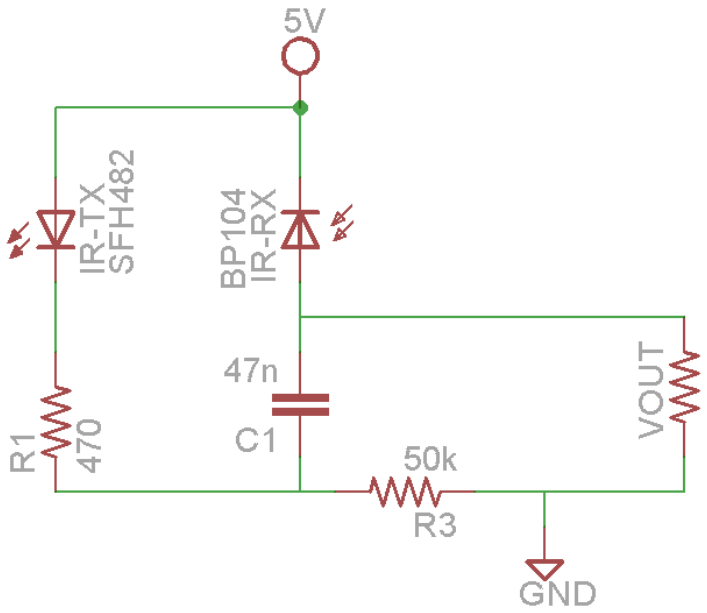

Figure 41. A schematic of how the team created a PIR sensor circuit.

In the research section regarding pressure sensors, section 2.2.2.3, the team has provided a chart relating the resistance of the force sensitive resistor to the amount of pressure, in grams, being exerted on that sensor. The specific sensor chosen is the FSR400 created by Interlink Electronics. The resistor's sensing pad is circular, with an area of 0.3 square inches. A voltage dividing circuit, as shown in 2.2.2.3, Figure 41, is required to determine the resistance of the sensor, which changes when force is applied to the sensor. The ATmega uses a GPIO to measure the voltage on the resistor, then applies a formula to determine the resistance.

Integrating all of the sensors mentioned above was a relatively simple task in terms of hardware. To connect the EMG sensor to the microcontroller, the SIGNAL output pin was attached from the EMG sensor to an available pin on the multiplexer. To connect the distance sensors to the microcontroller, a wire was attached from the top of the Vout resistor, shown in Figure 11, in section 3.1.4, to an available pin on the multiplexer. Since several force sensitive resistors were used, each one required a voltage division circuit and an available pin on the multiplexer.

4.5 Mobile Application

The mobile application was designed to provide the user with an easy but capable interface. This section discusses all the design details regarding graphical user interface (GUI), algorithm used to create gestures from user input, communication with the IPPA system, and voice commands. As part of the IPPA system a mobile application was developed: IPPA Mobile Support. This mobile application provides the user with the following features:

- Create new hand gestures
- Add new gestures to the arm
- Specify voice command to trigger gesture
- Edit previously created gestures
- Delete gestures from the arm
- Save gestures in the phone itself
- Sync arm gestures with phone to enable multiple devices

The Android platform was selected as the platform of choice for the development of this application. This decision was based on the team's engineer's familiarity with it, the vast online support for development, as well as the low cost of developments and mobile devices that use this platform (see section 2.2.3 for Android platform details). Figure 42 shows a high level overview of this application.

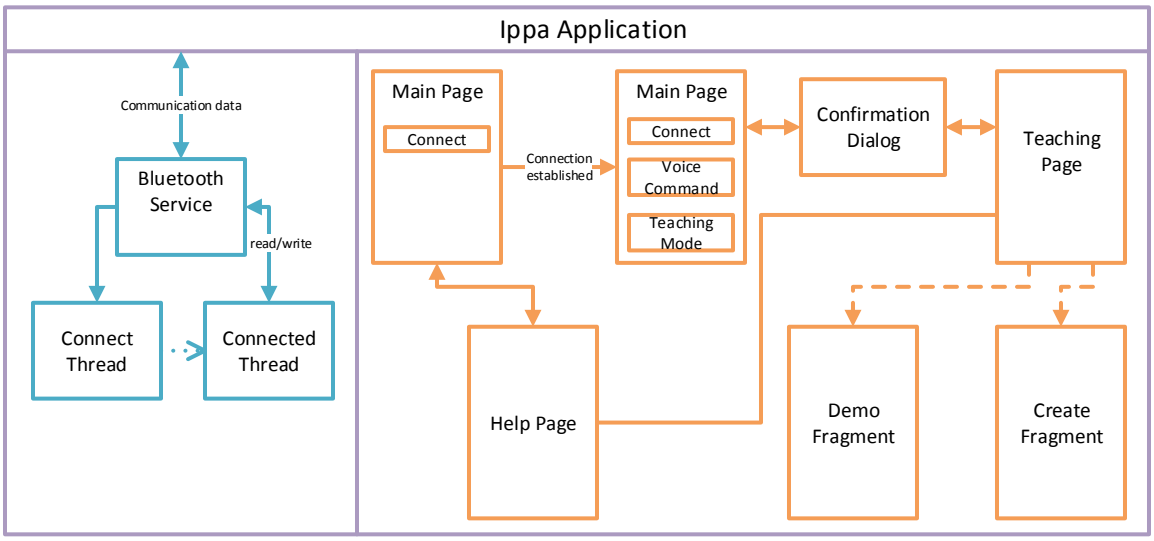

Figure 42. This is the high level view of the entire application

4.5.1 Graphical User Interface (GUI)

The application has been designed to be simple and have all the necessary components to provide the expected functionality and quality. The major colors for the application are different tones of green, pink and yellow. The elements used have been customized to use the desired color, however no complex components

have been designed for the application in order to reduce the cost and time of production. Following the look and functionality of each page of the application is described.

Entry (Main) Page It contains three buttons for: establishing connection, voice command translation, and entering teaching mode. When the user clicks on the connection button the app tries to establish a Bluetooth connection with the IPPA system. Multiple toast messages are possible to inform the user of the progress or lack thereof. If the device cannot connect to the IPPA system, a dialog is displayed to communicate this to the user. If the problems with communication are due to the lack of Bluetooth capabilities the application is terminated. When the user clicks on the voice command button the google speech translator dialog is displayed and the voice of the user is recorded, once the recording is done, the dialog goes away. When the user clicks on the teaching mode button a dialog is displayed to confirm the user's selection. This avoids mistaken transition into this mode. The details of this page's design are discussed further below in section 4.5.1.3. The action bar (top bar) has been enabled to provide a help icon action, this takes the user to the help page, and a sync icon action to get the gestures stored in the arm. There is an introductory text area bellow the action bar. A text is displayed with the status of the connection. This is shown in Figure 43.

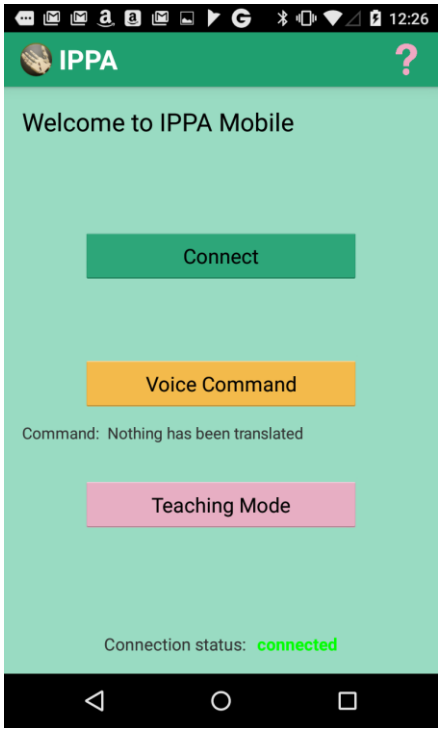

Figure 43. GUI of the entry page of the application.

Teaching Mode Page This page is composed of two fragments, or sections: "Create Gesture" and "Demo Gesture". These sections appear as tabs at the top

of the screen. The action bar is enabled, and allows the user to go back to the main page or to go to the help page, and a sync icon action to get the gestures stored in the arm. The user is be able to click on the tab or swap to switch between fragments.

The create gesture fragment provides the user with the functionality to create new and custom gestures. It contains the following components: titles for each subsection, text input for gesture name, checkbox to allow the user to change the start position of the arm, five sliders to set start position of each finger appear in this case, five sliders to set the end position of each finger, radial buttons to select pressure allowed, an edit text for the user to input the voice command for the gesture, two buttons to clear or save the gesture.

The demo gesture fragment supports the test of previously saved gestures or gestures in the arm itself. It contains the following components: two text views with the title for the following lists, list of gestures stored in the phone, list of gestures stored in the arm. Depending on the location of the gesture the user is presented with different options. If the user selects an item in the list stored in the arm a dialog appears with the options: to delete the gesture, to demo it, or to transfer it to the phone. If the user selects an item in the list stored in the phone a dialog appears with the options: to delete the gesture, to save the gesture into the arm, to edit it, or to demo it. Figure 44 shows part of this page.

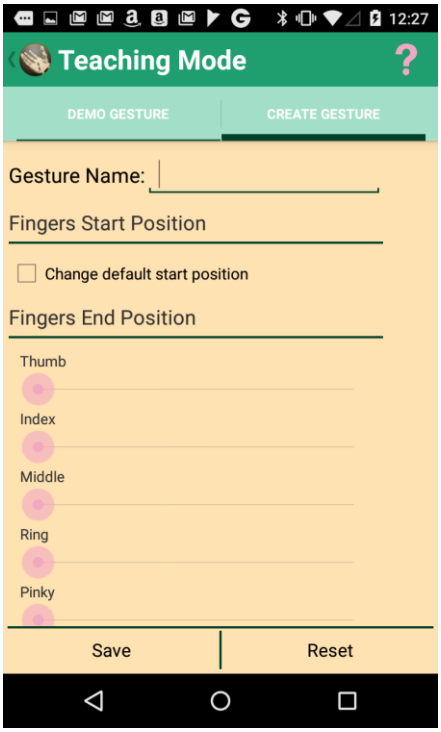

Figure 44. GUI of the teaching mode under the create tab

Help Page In this page, instruction are given to the user on how to create a new gesture, save it and copy it into the IPPA system. This page is composed of text. The action bar has been enabled to allow the user to go back to the previous view (main page or teaching mode page).

4.5.2 Algorithm

In this section, the control flow of the application is discussed. There are a total of four activities for this application: Main activity, Help activity, TeachingMode activity and DeviceDiscovery activity. If the connection is lost at any point during the execution of the application, a dialog is displayed to reconnect. Even though the triggering voice command for a gesture can be modified and set by the user, the "reset" command is reserved and can't be deleted from the system. This guarantees the user has a reset option for the hand, as a safety precaution. Each gesture has an identifier (name) that is given by the user and it is used to identify the gesture. A feature to support the change of the phone is the sync action within the action bar.

Storage This application needs to store multiple files in the phone. There is a file to store all the gestures in the phone and arm.

IPPA Application As a global state of the application the connection threads are kept in this custom application object in order to maintain the connection, and have access to the transmitting streams from anywhere in the application. This guarantees only one Bluetooth service instance that cannot be directly modified.

Main Activity A major requirement for the application is for the phone to be connected to the IPPA system through Bluetooth. In order for the user to do voice commands or proceed to the Teaching Mode page, a check of connection and status of the connection must be done. If the check passed then application proceeds to the selected page. The main APIs from the android.bluetooth library that was used in order to accomplish this are listed in section 4.5.3.

Voice Command The speech recognition is done using the Android's built-in Speech Recognizer activity. When the user clicks on the button to input a command, a speech recognition activity is started (Google API). The onActivityResult() method is used to handle the result obtained from the launched activity. Multiple translated texts are obtained and compared to the available gestures. Once the audio input has been translated to text, a package of type A (see section 4.5.3) is created and sent to the IPPA to trigger the gesture. If the given input does not match the strings for the current gestures in the IPPA system, then no package is sent to the arm. Figure 45 shows the speech recognition process.

Teaching Mode Activity This activity can only be started by the Main Activity. The global Bluetooth connection is used through the application's send methods. The

main purpose of this activity is to provide flow between the Create Gesture fragment and the Demo Gesture fragment.

Create Gesture Fragment This fragment gathers all the information needed in order to create a new gesture. Once the user is satisfied and clicks on the save button the Gesture object is populated with each customization and saved in the file in the phone. During the creating process, every time the position of a finger is changed, a package of type B is sent to the IPPA system; this provides a live feedback during the creation of the gesture. This application supports gestures with a custom start position, but it is not the default; the user must check the "Change Start Position" checkbox, this enables the change of the start sliders. The position of the sliders is discrete to provide precision. During the creation of a gesture many packages are transmitted.

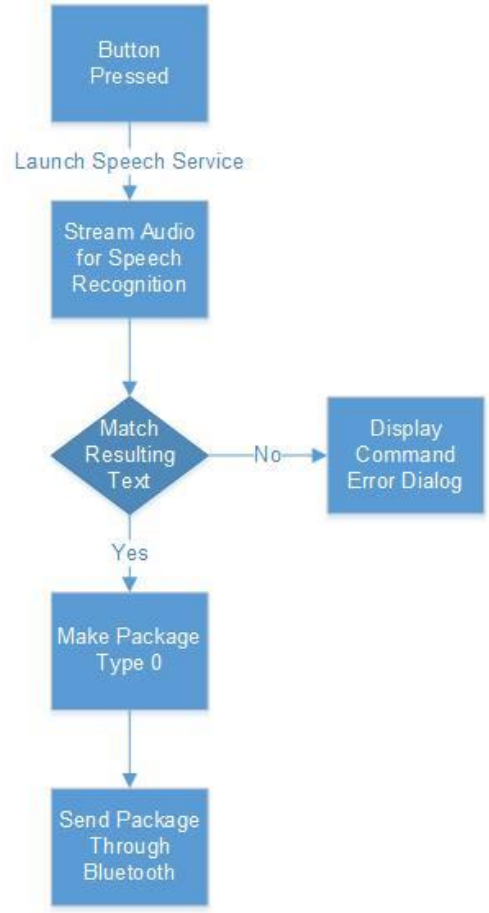

Figure 45. Control and data flow to obtain speech recognition triggered gestures

Demo Gesture Fragment This fragment allows the user to delete, add, demo, or edit a gesture in the IPPA system. Each gesture has possible actions to perform, which depend on the stored location of it. In this fragment the file with all the stored gestures are loaded and the two lists of Gestures are populated. The generation

of the package is done by the activity, which is then sent to the Bluetooth service thread through the global state.

4.5.3 Communication

The communication with the IPPA system, as mentioned before, is accomplished through a Bluetooth connection. The team decided on Bluetooth because it is a simpler solution, with less power consumption and it satisfies the distance and safety requirements of the project. In order to have a responsive connection and live communication with the IPPA system, a second thread is running the Bluetooth service. The communication between the threads are asynchronous.

Different package types have been designed to reduce the amount of information that needs to be transmitted and to facilitate the understanding between devices. By doing this the interface between devices is smoother. Each package is composed of a package type and the data for that package, which is defined based on the type. Table 10 lists all the package types that are implemented for communication.

Type	Description	Data
A	Trigger a gesture	Gesture identifier
B	Update the position of each finger	Position of all five fingers
C	Add a new gesture to the arm, no trigger	Full gesture information
D	Delete an existing gesture	Gesture identifier
E	Temporary store a gesture and trigger it (Demo)	Full gesture information
F	Request command strings stored in the arm	N/A
G	Switch IPPA modes	N/A
H	Send voice command in arm	Variable number of strings
I	All gestures stored in the arm	Variable number of full gestures

Table 10. Package types designed for the IPPA system communication with the mobile application

Due to the package specialization, each package has different lengths. This reduces the number of bytes needed to transmit information. Figure 46 shows how the data is structured within each package. Package types B and D contain all the information that makes up a gesture: start position for all fingers, end position for all fingers, command string, and the sensor information. The sensor information is the pressure levels that are the maximum allowed.

Figure 46. Structure of the different package types to be transmitted

The packages are formatted as strings, with a white space in between each piece of information. This facilitates the parsing of the information by the modules in the main system.

4.6 Power Unit

There are over a dozen separate components that require power. Some have almost negligible power consumption rates, such as the force sensitive resistors, electromyography sensor, and even the servo and sensor microcontrollers only draw a small amount of power. However, since all of these components needed to be integrated together and draw power from the same power source, it required calculating the maximum expected power consumed by all the devices when they are running. The important factors to be kept in check are battery life, battery output, and, as a requirement of the entire project, weight.

4.6.1 Power Specifications

Below is Table 11 describing the recommended operating specifications for the components integrated into the prosthetic. The total current drawn by the system sums up to about 650 - 2800 mA.

As for powering the servos, which are capable of drawing a combined 12,500 mA when exceeding their load limit, and up to 2,500 mA when no load is attached, they required a large, rechargeable battery with a high output current. Such a battery is used as the battery pack for an emergency light [3]. The battery we have chosen to use contains 2200 mA hours of charge. It can discharge about 9 amps maximum. Assuming the servos are not continuously moving, the lifespan of this battery should exceed 1 hour.

Component	Voltage	Current	Power
IMU [7]	3.3V	8 mA	26.4 mW
EMG [24, 25]	± 5V	1.8 mA	8.8 mW
FSRs (Rm = 27 kOhm)	5.0V	0.2 mA	1.0 mW
PIR [19, 20]	5.0V	10.5 mA	52.5 mW
Servo MCU [26]	3.6V	4.5 mA	16.2 mW
Sensor MCU [26]	3.6V	4.5 mA	16.2 mW
System MCU [21]	3.3V	30 mA	99 mW
Single Servo (idle) [21]	6.0V	5 mA	30 mW
Single Servo (no load) [21]	6.0V	500 mA	3000 mW
Single Servo (stalled) [21]	6.0V	2500 mA	15000 mW
Bluetooth [20]	3.3V	50 mA	165 mW

Table 11. Power requirements of the electrical components in the IPPA.

4.6.2 Sources

The intelligent programmable prosthetic arm utilizes two different power sources in order to power the various subsystems. A pair of 9 volt batteries are responsible for powering the EMG sensor, to create the positive and negative voltage references. A larger, rechargeable, 7.4V battery is used to supply power to the rest of the system.

Taking into consideration that the prosthetic arm requires about 200 - 350 mA of table 11current in order to supply the non-servo components. Our 2200 mAh battery can supply these components for 8 hours.

The largest power consumers during operation are the servo motors. Each one is capable of consuming over 100-2500 mA of current and 600-25000 mW of power when engaged. In order to supply the servo motors with sufficient power, the team chose to use a high-output rechargeable battery. Lithium Ion was concluded as the battery of choice. Five loadless servos in motion would draw at least 500 mA of current from the battery, resulting in about 4 hours of battery life. With added load, as much as 2500 mA could be drawn from the battery, reducing battery life to less than an hour. We assume that the servos are not continuously engaged, which extends the expected battery life to well over one hour.

4.6.3 Voltage Regulators

Voltage regulators are placed between the battery and the electrical devices that require power. These regulators convert the 7.4 volts provided by the rechargeable battery into the various voltages required by the project's components. Table 11

above, in section 4.6.1, describes the five different voltage levels required. For each of these levels specified, a different voltage regulator was used.

There were four voltage levels required to power all of the remaining devices. A 3.3 volt regulator was used to power the Bluetooth module and the system controller microcontroller. An adjustable LM317 voltage regulator met the required specifications. It was adjusted to supply 3.3V as its output voltage, while providing as much as 800 mA of output current.

A 5 volt regulator was used to supply power to two of the sensors and the ATmega microcontrollers. The force sensitive resistors use a 5V supply to create the voltage divider circuits and the EMG sensor uses 5V of power at its positive terminal. The passive infrared sensor requires a 5V power supply as well. These two sensors draw very little current, about 10 mA. A 5V regulator, such as an MIC5205 satisfies the project's needs. It provides 5 volts of output voltage, and can supply up to 1.5 amps of current. It can accept up to 18 volts as its input voltage. Figure 47 shows the voltage regulator wiring schematics.

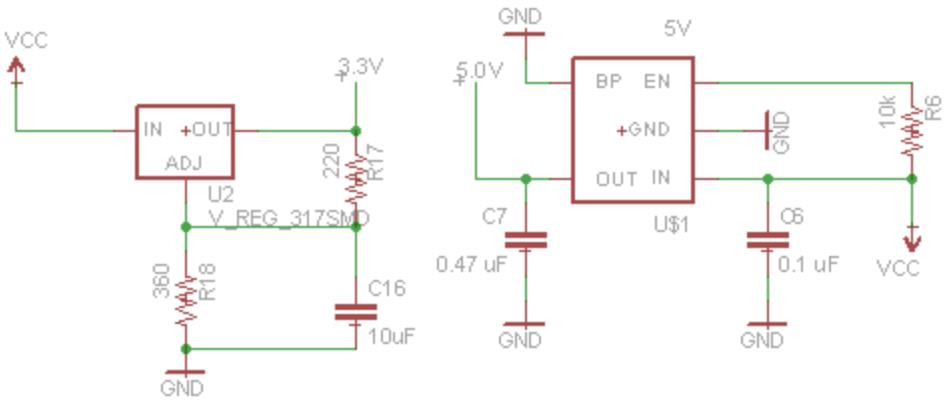

Figure 47. Voltage regulators' wiring schematics.

4.6.4 Power distribution

Power is distributed by wiring four voltage regulators to the 7.4V battery in a parallel configuration. This prevents voltage levels from changing at the inputs of the voltage regulators, and allows different amounts of current to flow through the regulators. This configuration is shown below, in Figure 48. The components are wired to the outputs of the voltage regulators, also in parallel.

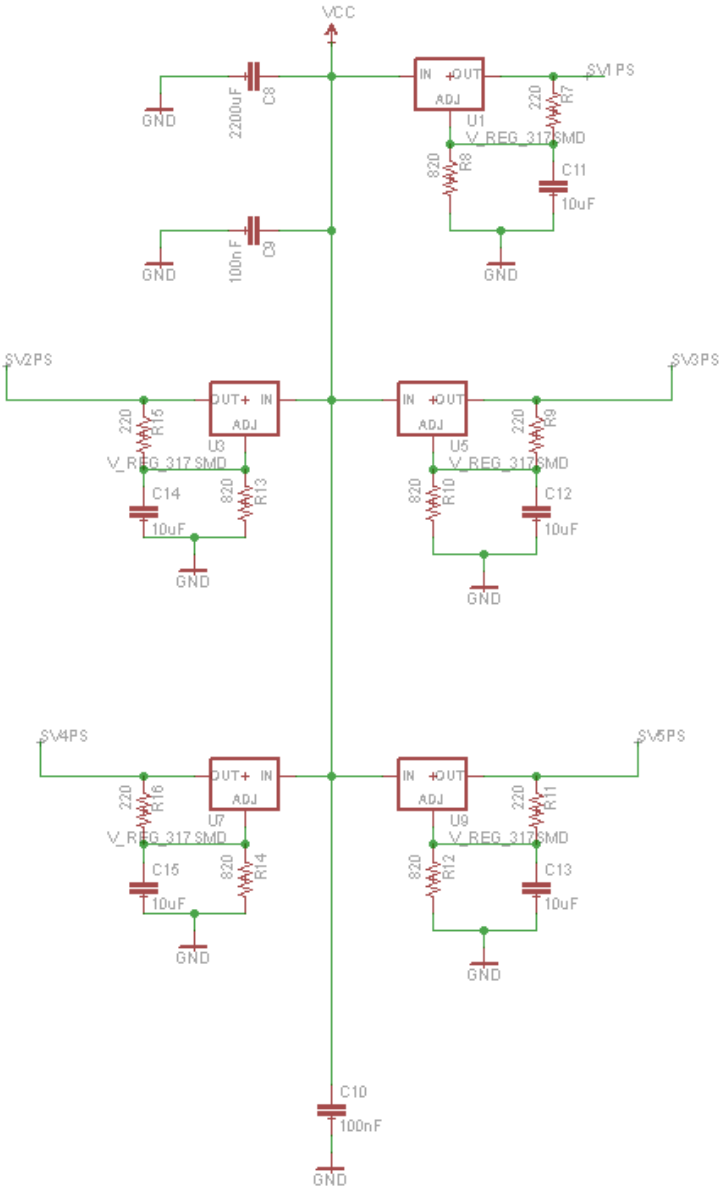

Figure 48. Schematic showing how the team wired electrical devices to their power sources.

4.7 3D Printed Arm

This section describes the 3D arm design selected for the final product. The main focus of this project is not the mechanical aspects of a hand design. Therefore, a hand design from an open source project has been used. After investigating multiple open source hand designs, the team decided to use the InMoov arm because of its completeness and capabilities. There is also a vast documentation available from this project on how to assemble it. The right hand has been chosen

for the final product; however, this project could be easily adapted for the left hand. All of the 3D parts needed are provided in the InMoov's project website.

In order to make the other needed parts, such as the PCB bed the team used the following software: MeshLab, which is open source software; and SOLIDWORKS, which is available to students through the Harris Lab in the Engineering building. Unfortunately the provided stl files for the InMoov hand are not editable; therefore all the changes made to the hand have been done after printed (i.e. drilling for distance sensor location).

4.7.1 Hand Unit

The InMoov hand design provides three joints for each finger, which is the same number of joints in prosthetics that cost $10,000. This gives each finger a wide range of motion, and the possibility of closing on relatively small objects. There is a hand base where the index and the middle fingers attach to. The other three fingers have an additional joint in the hand that contributes to a better grip of different shapes, such as a ball. There are wires running within the hand, for the sensors and the opening/closing functionality. On the top left the additional joints are placed, one for the pinky finger and another one for the ring finger. The thumb attachment joint goes in the middle right open space as seen on the left of Figure 49.

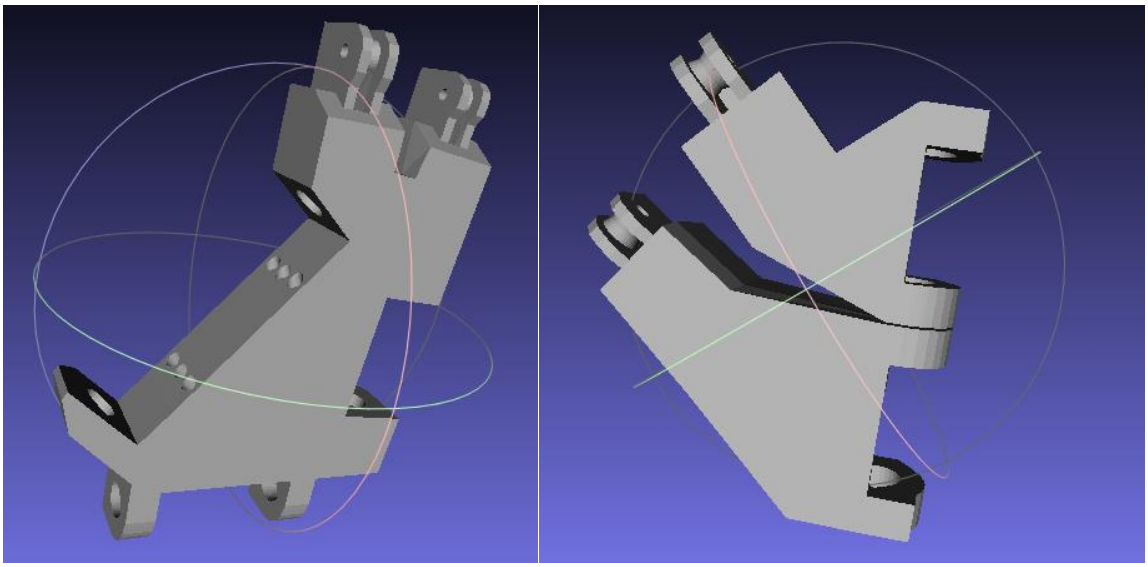

Figure 49. 3D design as seen in MeshLab. Left: the main hand base. Right: the hand joints for the ring and pinky fingers.

Since this project added pressure and distance sensors to the hand, the design for the main base as well as the fingers must be slightly altered. Space for 3 pressure sensors are needed. These were not be added to each finger but to two key ones: the thumb, and the index. The other two are placed on the main base of the hand. The location for this is at the 1/3 and 2/3 from the bottom to the top of the hand.

Silicon pads were placed on top of each sensor to supply a wider area of contact, and to reinforce the grip strength of the hand.

4.7.2 Forearm Unit

The InMoov forearm has been designed to be hollow, with sufficient space for five servos which control the hand movement. It was also designed to contain batteries and an Arduino board, which is very beneficial to this project since there must be space for the Power Subsystem, the Printed Circuit Board (PCB), and the Bluetooth component. The forearm was made up by five major 3D printed parts, seen in Figure 50.

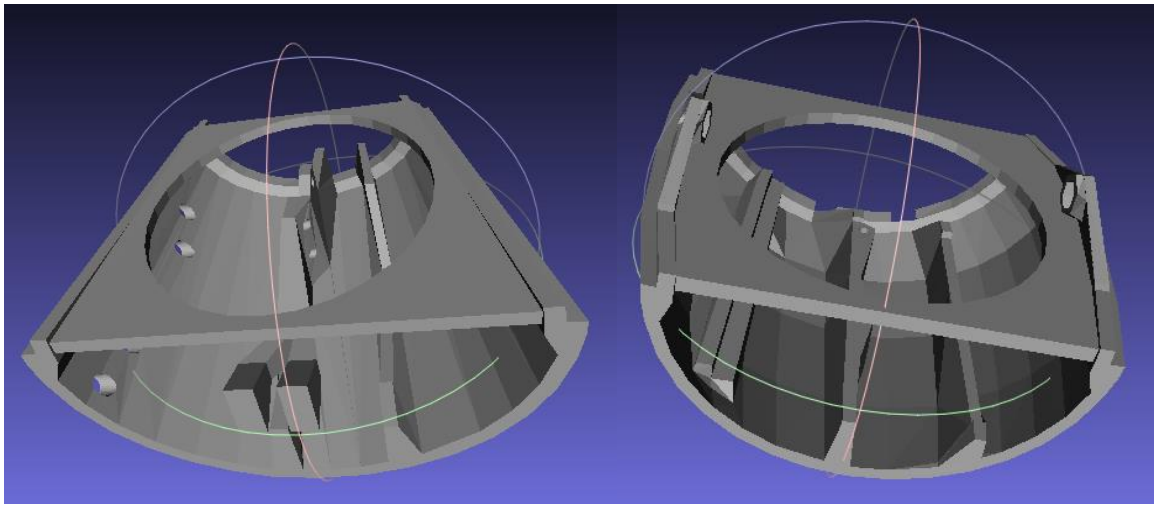

Figure 50. 3D design as seen in MeshLab. Left: forearm part closest to the wrist. Right: forearm part where servos are positioned.

Instead of fully redesigning the forearm, only the inside of was modified to provide a better fit the following components:

- Five servos (1.6” X 2.2” X 0.6”)
- EMG sensor (1” X 1”)
- PCB (1.7” X 3.1”)
- Power Supply
- Cables

Since the IPPA has an EMG sensor, three wires run outside of the forearm and are implanted on the user’s arm. The user is responsible with placing the sensors on the indicated muscles. Two buttons are placed on the surface of the forearm. These are located in the inside of the forearm 1/3 of the way from the wrist. The forearm has the area needed carved-in in a circular shape. This avoids the user pressing the buttons by mistake, such as placing the arm on a hard surface.

5 Project Construction and Coding

This section discusses the main elements needed to be produced. This includes fabrication of the 3D printed prosthetic, the PCB that contains all of the elements need to implement the system controller and servo controller, the android application that runs on a mobile phone, and the software files that controls the system controller and servo controller. All of these elements were developed in parallel fashion, and when possible integrated when necessary into one final product. It discusses all of the necessary software, development environments, and other tools needed to develop the hardware and the software components.

5.1 Printed Circuit Board

For the IPPA to be a practical device, a printed circuit board was designed to house the majority of the electronic components used to control the arm. To save space, surface mount components utilize, that are a fraction of the size of drop in place components. By utilizing a PCB design, integration was done once large microcontrollers onto a much smaller footprint.

5.1.1 Design Environment

The printed circuit board design environment that was used is EAGLE 7.1.0. EAGLE offers an easy to use interface and produces schematics which are highly compatible with most PCB printing services. Our team has little experience with PCB design software, so it is important that the team chooses a program that has adequate support documentation, is low cost, and offers an array of functionality. With EAGLE, the team can produce not only the printed circuit board layout, but also supporting schematics that can be used in prototyping and debugging. EAGLE is free for students, and allows to create two-layer designs, which provides more than enough options for the relatively basic design. Creating designs greater than two layers would be out of the budget.

5.1.2 PCB Layout & Specifications

The printed circuit board must follow several requirements in order to 1. Fit within the budget and 2. Be compatible with the requirements of the off-site printed circuit board manufacturer. The most important feature of the board in order to fit within the IPPA forearm.

Therefore, the team only increased the size of the board when it absolutely cannot fit any more components in the design. In addition to the size of the board itself, the off-site vendor OshPark has several requirements of its own:

1. 6 mil minimum trace width
2. 6 mil minimum spacing
3. At least 15 mil clearances from traces to the edge of the board
4. 13 mil minimum drill size
5. 7 mil minimum annular ring

OshPark supplies a DRU (design rules) file, which verifies the design meets the requirements of OshPark. In Figure 51 below, the designed PCB is shown. It contains all of the modules and components required by the IPPA.

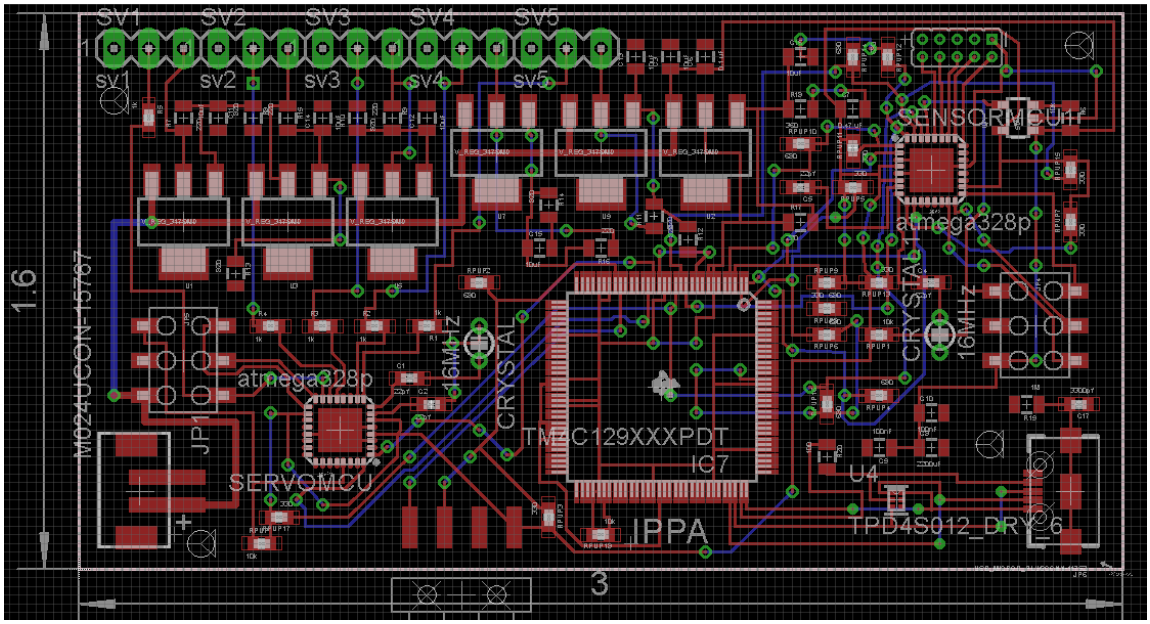

Figure 51. IPPA PCB.

5.1.3 PCB Vendor

The decision of which vendor to purchase from came down to two manufacturers, OshPark and 4PCB. The most important factors when deciding on the vendor would be pricing and shipping time. Shipping time should not matter much, as both sites offer shipping times of less than two weeks. This turnaround allows for time to design, submit, and assemble the PCB board within just a few months.

The second factor is pricing, which may be the most important. Both OshPark and 4PCB manufacture boards of similar quality and with the same number of layers (two). OshPark prices their boards at 5 dollars per square inch, while 4PCB prices their boards at a static 33 dollars per board, with a student deal. What this means

for the team is that if the board is designed to be under 6.6 square inches, OshPark becomes the better deal. The extra advantage provided by OshPark is that they send three copies of the same board. Considering the fact that the team has little soldering experience, and even less experience working with surface mount components, having two extra copies of the same board provides a safety net, should a mistake be made on the first board. Therefore, the team has decided to go with OshPark as the printed circuit board manufacturer.

5.2 Software Implementation

This section discusses the considerations during the overall software development and the environments used to develop the software for all major hardware elements. The major components that required software to be developed are the external application, the system controller, and the servo controller. For the system controller, the software was developed in the languages of C and ARM Cortex – M assembly language. During the development of the system controller, some functionality was prototyped on the TM1294 embedded system. This prototyping enabled the team to learn insights about memory management, processor optimization, and real-time data acquisition that is important in the algorithm design.

To develop the software for the main controller, the team used Energia during the prototyping stage of development as well as further development only when a team member needed to test certain functionalities quickly like UART and sensor calibration. For the overall development, the project needed a strong development environment that enables the team to develop, debug, and test the software and all of its modules. We decided to use Keil uVision4 compiler and the Texas Instruments and Code Composer Studio (CCS). The reason these were used is that the Keil uVision4 compiler can run on MacOSX and CCS can run on windows. The team developed using both a MAC and Windows operating system; compilers were needed to be used on both operating systems. To develop software for the servo controller, that was also developed on the TI CCS IDE. To develop the android application, the team used the Eclipse IDE to develop, debug, and test the android application.

5.3 Hand Fabrication and Assembly

As done for the prototype, the final hand product was manufactured through the Texas Instruments Innovation Lab located at the University of Central Florida, in the Engineering building. All of the components of the hand are 3D printed using the ABSplus – P430 3D, Dimension sst 1200es 3D modeling printer. In order to assemble the hand, all of the parts listed in Table 12 must be 3D printed. 3D custom bolts have been designed to attach the thumb, ring and small finger to the hand base. Besides the 3D printed parts, the team needed 3mm bolts or 3mm filament for each of the joints in the fingers.

Major Component	Parts	File
Index finger	6	Index3.stl
Middle finger	6	Majoure3.stl
Ring finger	7	Ringfinger3.stl, WristsmallV3.stl
Small finger (pinky)	7	Auriculaire3.slt, WristsmallV3.stl
Thumb	6	Thumb5.stl,
Hand base	1	WristlargeV4.stl
Bolts	6	Bolt_entretoise7.stl

Table 12. Hand 3D components needed and their respective files.

For better results, file all the finger parts to obtain a smoother interaction in the joints. Also, the holes in the joining finger parts should be drilled. ABS glue was used to glue different finger parts together since this provide the strongest union between parts. The tip of the fingers were not be connected until the hand is fully assembled, since these are not needed but are just for a more natural look of the hand. After the hand has been assembled, the two pressure sensors were added in their respective locations (see section 4.7.1 for details). All the cables were ran inside the hand, and into the forearm, where these were connected to the PCB. Braided fish line 200lbs was used for the servo lines.

5.4 Forearm Fabrication and Assembly

The forearm, just like the hand, was manufactured through the Texas Instruments Innovation Lab located at the University of Central Florida, in the Engineering building. All of the components of the forearm were 3D printed using the ABSplus – P430 3D, Dimension sst 1200es 3D modeling printer. In order to assemble the forearm, all of the parts listed in Table 13 must be 3D printed.

Major Component	Parts	File
Forearm shell	4	Robpart2V3.stl, Robpart3V3.stl, Robpart4V3.stl, Robpart5V3.stl
Forearm end caps	2	Robcap3V1.stl
Servo custom pulley	5	Servo-pulleyX5.stl
Servo positioning bed	3	RobServoBedV5.stl, RobCableFrontV3.stl, RobCableBackV3.stl

Table 13. Forearm 3D components needed and their respective files.

For better results when gluing the parts, file all the edges in the forearm shell. Also, the holes should be drilled with a 6mm drill. ABS glue was used to glue the four forearm shell parts together since this provide the strongest union between parts. After the top and bottom of the forearm (2 pieces) have been glued, the servo bed was assembled. All the cables entered the forearm at the “wrist” location and expand on the bottom of the forearm, leaving space for the PCB and batteries. Then the servos were mounted and the servo pulleys were be installed on them.

Before connecting the fishing lines coming from the fingers, set all the servos to zero degrees.

5.5 Bill of Materials (BOM)

In Table 14 below, is the list of all of the specific parts and components used in the creation of the IPPA. Components that are not mentioned are small parts that vary widely, such as resistors and capacitors. Also, depending on availability and price variations, the exact parts and prices may change. It also may be discovered that the PCB design needs to be modified, smaller or larger, thus altering the price listed below. The quantity of items may change as well, should the team receive a faulty component, or discover the system requires more or less of the components, such as certain sensors.

Item	Price	Quantity	Part No.	Vendor
Servo Motor	19.99	5	Pololu 1501MG	Pololu
Force Sensitive Resistor	5.95	5	FSR 400	SparkFun
ATmega328P	2.87	2	MSP430G2553IRHB32T	DigiKey
IR Emitter	1.95	4	LTE302	SparkFun
IR Receiver	1.95	4	LTR301	SparkFun
Tiva Series MCU	12.13	1	TM4C123GH6PMT	DigiKey
Bluetooth Module	5.95	1	HC-06	EBay
3.3V Regulator	0.64	1	LD1117V3	DigiKey
3.6V Regulator	0.67	1	MIC5205YM5 TR	DigiKey
5V Regulator	0.66	1	L7805CDT-TR	DigiKey
PCB	60.00	1	n/a	OshPark
9V Batteries	8.15	4	MN1604	Amazon
6V Battery	51.00	1	Unknown	All-Battery
Acetone	10.00	1	Unknown	Home Depot

Table 14. Continue: list of components, prices, quantities, part numbers, and vendors.

6 Testing and Calibrations

The testing and calibration section refers to specific tests that were conducted to determine the efficiency of specific components, verify that they function correctly, and make any adjustments to the hardware or software. In this section, it describes what component were tested, what feature of that component is in testing, how the team intends to perform the test, and the expected results.

6.1 Power Management

Since the IPPA is a portable unit, it must be wireless. This requires to have a design for the IPPA with battery power. The IPPA is estimated to consume two nine volt batteries in under 9 hours, while in continuous use. This is not very good, however, the lifespan of a cell phone is usually less than 12 hours. It would be useful to the users of the IPPA to invest in rechargeable nine volt batteries, so that they could charge them at night when they are not using their prosthetic. This way, they could convert their IPPA to being a completely rechargeable device, lowering the cost of buying new batteries.

We implemented power saving strategies that were built into the software of the IPPA. Simply turning off modules when they are not in use saves power and extend battery life to much more than 9 hours. Also, the microcontrollers to be used can be set into power saving modes, which reduce their required power levels and also their speed. They can be restarted into an active, powered on mode very quickly.

Another strategy that can be implemented is reducing the amount of polls that the microcontrollers perform on certain sensors. For example, the force sensitive resistors are only relevant when the user is performing a grasping gesture. While not grasping, the sensors can not only be turned off, but also the microcontroller can spend its resources on other sensors. In addition, the polling rate of the microcontroller can be reduced for all sensors. Polling sensors only once per second reduces the resources the microcontroller uses, without sacrificing noticeable reaction speed.

Implementing the above mentioned strategies could reduce the amount of power required by the microcontrollers significantly, which are some of the largest power consumers. However, the largest non-microcontroller power consumer is the Bluetooth module, which requires up to 50 mA of current. Turning this module off while not in use extends the battery life of a nine volt battery to twice the theoretical amount. Since the non-servo components draw about 100 mA total, removing the Bluetooth module from the equation reduces the amount of current required by half. Instead of 9 hours of battery life, the system would attain nearly 18 hours, which is quite significant.

If it is discovered that battery life is still not long enough, even with the strategies mentioned above, the clock frequencies could be lowered for the microcontrollers.

This would reduce how quickly the code is run and calculations are performed, but it might be worth testing whether the reduction is noticeable.

Lastly, it is worth noting the different power consumption levels of the servo motors, which is powered by a 7,800 mAh rechargeable battery. The worst case scenario regarding the servo motors is that the user is trying to lift an object so heavy it causes the servo motors to stall, peaking their current consumption to their maximum total of 12,500 mA. This would drain the battery in 48 minutes.

Another scenario is that they perform gestures only with the IPPA, which would cause all the servos a total current consumption of 2500 mA. This would give the battery 4 hours of use before being drained. The longest running scenario is that the user does not use gestures at all, never causing the servos to leave idle. At 25 mA total in idle mode, the servos would take 400 hours to drain the battery. The most likely scenario is that the user uses the servo motors to make gestures and occasionally lift light objects.

6.2 Servo Subsystem

The servo subsystem is crucial to obtain a good grasp. The precision as well as the speed of the movement of each servo was extensively tested to make sure the IPPA system fulfills all stated specifications and requirements. Most of the tests were via software since the servo subsystem is composed of a microcontroller and five servos that are controlled through software libraries. Testing the servos' movement also tests the strength and correctness of the attached fishing line. In order to determine if the servo subsystem is functioning properly the requirements listed in Table 15 must be met.

The servos first were tested independently, and then there were tests for concurrent movement. The following tests have been designed to test all of the requirements previously shown:

1. Write a software unit test that changes the position of the servo from 0° to the full extension of 180° without stopping. Do this for each finger. Verify that each finger goes from extended to flexed.

2. Write a software unit test that changes the position of the servo from 0° to 180° but stops every 30° for a noticeable amount of time. Do this for each finger. Verify that the movement is as expected when the servo stops moving, and when it starts to move from a given position.

Requirement	Description
1	Each finger must be able to move independently
2	The movement of the finger must be at a natural rate/speed
3	The servo subsystem must be able to override the main microcontroller's order and reset the hand to open
4	The servo microcontroller receives input from the main controller
5	The servos must be able to hold a given position for some time
6	The hand must be able to hold on tight to objects, without the servo giving out
7	The servos must be able to stop at the current position when signaled
8	The lines attached to the servos cannot stretch after extensive use

Table 15. List of software and hardware requirements for the servo subsystem.

3. Write a software unit test that changes the position of all the servos for the hand to be fully opened and change to a fully closed hand. Verify that the movement of the fingers is seems as a concurrent motion.

4. Write a software unit test that changes the position of all the servos from open to close randomly. Provide some time for the motion to happen, then run it again. Verify that the fingers are independent.

5. Write a software unit test to simulate a reset input from the button. Verify that the hand changes quickly to an open state.

6. Test the grasp capabilities of the hand by using different objects and a software unit test that closes all the fingers. Repeat with multiple object shapes and different weights. Verify that the hold pattern meets the requirements in Table _ and the specifications given in section 1.2.

7. Simulate a stop signal from the Sensor subsystem, and test that the servos stop. Verify that this happens without dropping the object being held.

8. After running all of the previous tests, verify that each finger line has not been stretched and that the fingers continue to have full motion by running test 1 again.

6.3 Sensor Subsystem Testing

This section contains the testing of all sensors, calibrating the threshold of sensor value that triggers a state change/action, and the tests to verify that the sensor

subsystem is working correctly. To review, the functions the sensor subsystem are to analyze the values of the FSR sensor, distance sensor, and EMG sensor. The FSR sensors are analyzed to prevent the IPPA from exerting dangerous levels of force on itself or others. EMG Sensor is analyzed in order to be able to detect when the wearer flexes his/her muscles. The PIR Sensor is analyzed to trigger activation of a grasp at a very specific distance.

Some expectations of the prosthetic that affect the sensor subsystem is the arm must grasp triggers if and only if an object is within ½ an inch from the sensor. The hand should stop grasping when it detects dangerous levels of pressure. Grasping tasks should withstand 5 minutes of continuous use. The hand should grasp when an object is less than or equal to 1 inch. In order to meet the expectations defined previously, the sensors have to be analyzed in order to determine strong thresholds that can be used to trigger certain events. The team needs to analyze the FSR sensor to determine the range of pressure it can sense, and determine a strong threshold that is uniform in a wide range of grasping situations.

An FSR sensor works as the pressure on the sensor increases, the resistance decreases. In order to calibrate, the range of resistance values was determined when there is high pressure, medium pressure, and low pressure. This enabled the team to develop a strong threshold of what resistance value reaches high levels of pressure. We can use a multi-meter can complete this survey of resistance values. Once these ranges are obtained, then research and test can be done for different grasps and what is the range of pressure humans exert.

Finally, the sensor connects to the main controller, read the analog value, and test values to find an appropriate threshold that the sensor subsystem can determine.

Calibrate PIR sensor: A PIR sensor works by an infrared LED emitting a pulse of light at some rate (ex. 38Khz) and an IR receiver able to detect light at the same rate as the emitter' emitting rate. The time delay between the emissions of light to the time received is calculated to find the distance of the object that is in front of it. One of the main functions and expectations of the PIR sensor is to sense when an object is within 1.3-2.5 cm of .5-1inch of the prosthetic hand.

Calibration is needed to determine if the signal receiving from the sensor is noisy or not. If the sensor is receiving a lot of noise, the IPPA system can have a algorithmic filter that uses statistics such as Gaussian smoothing to filter out a noisy signal. Testing was required to make sure that the sensor was able to determine objects of different sizes and material textures when they are within the distance range specified. Testing was be completed on the TM1294 system.

Calibrate EMG sensor: To review, the way an EMG sensor works is by muscle activation via electric potential. The electric potential is acquired as an analog signal, amplified by the EMG sensor hardware, and passed into the Main Controller for further analysis. The main function and expectation of the sensor is to easily

determine when the user wants to complete a grasp or release grasp. This may be difficult whether the system auto calibrates itself to every user or just hand calibrate. Auto calibration may be better as a production standpoint because the user can tailor the system to his/herself and the prosthetic is sensitive to his / her muscle activity. This completes by adding a calibration stage where the user has to do a series of tests. There is a software module with basic statistical machine learning to analyze the data received from training to infer the best threshold for the system to know when the user is completing a grasping action. This may take more time to build and test.

Manual Calibration is useful to prototype the sensor and may be good enough for the scope of this project to show that the proof of concept is there. The way manual calibration is completed, is by having a test module to receive analog signal from the arm, read muscle activity, and record a stream of input signal values. From there, the team analyzed the recorded analog values to find range of values and best possible threshold. The series of test in this action assume the sensor threshold was manually calibrated. The following lists the series of test that test the overall functionality of the sensor subsystem according to what was defined in the design:

1. Test – initialize the analog pin to read for one distance sensor,
   Test Method – the module of the program that initializes an analog pin by outputting the values of the distance sensor on the serial port
   Expected Result – should see value of centimeters output on terminal.
2. Test – initialize the analog pin to read for one FSR sensor,
   Test Method – the module of the program that initializes an analog pin by outputting the values of the pressure sensor on the serial port
   Expected Result – should see value of resistance output on terminal
3. Test – initialize the analog pin to read for one EMG sensor,
   Test Method – the module of the program that initializes an analog pin by outputting the values of the EMG sensor on the serial port
   Expected Result – should see value of muscle activity ranging from 900-1200 output on terminal
4. Test –receive input information from the sensor from every sensor and provide an interpretation of the prosthetics status and surroundings
   a. Test Method – test the module that reads all sensors and prints the value of the sensor next to the name of the sensor the value came from
   b. Expected Result – should see a value similar to this:
      PIR: 3cm FSR: 200 Ohm EMG: 957

5. Test –trigger grasping by having the distance sensor and the EMG value reaching the threshold pre-defined
   a. Test Method – test an if statement that prints “GRASP” in the terminal when the distance sensor is at 1.3 centimeters (.5 inches) and the EMG sensor is at a threshold defined from calibration
   b. Expected Result – should see “GRASP” printed in the terminal

6. . Test –trigger grasping, send action to servo controller, wait for servo controller to receive completion
    a. Test Method – test module that when grasp is triggered, send string to servo controller, and wait until receive string “GRASP COMPLETE”, then print string in terminal
    b. Expected Result – should see “GRASP COMPLETE”, printed in the terminal
7. Test –trigger grasping, send action to servo controller, wait for servo controller to receive completion, and add pressure sensor to see if pressure exceeds threshold define from calibration
    a. Test Method – test module that when grasp is triggered, send string to servo controller, force pressure sensor to reach threshold and force servo controller to stop grasping
    b. Expected Result – when pressure reaches max, the servos stop
8. Test –Hand is grasping an object, now test if user wants to release grasp
    a. Test Method – test system when after receive “GRASP COMPLETE”, listen to EMG sensor and force EMG threshold to be reached. In this case, the system should print to terminal “RELEASE” and the servo controller receives a message and start releasing grasp.
    b. Expected Result – see “RELEASE” printed in terminal and see the servos releasing grasp
9. Test –servo controller sends message done releasing grasp, and servo controller sends a signal to the sensor subsystem that it is complete releasing grasp and the sensor
    a. Test Method – test system when sensor subsystem receives message, “DONE RELEASING” , “READING DIST” prints with the distance sensor values on terminal
    b. Expected Result – see “DONE RELEASING” , “READING DIST” and distance sensor values on terminal

6.4 Bluetooth Module Testing

This section contains the testing for the communication system and the Bluetooth HC-05 module. To review, the function of the communication subsystem is to interface the Bluetooth communication HC-05 module to the main controller. The purpose of the communication subsystem is to communicate to the external application. Also the functions of the communication subsystem are to facilitate receiving information from external application, streaming data when in teaching mode, and sending information back to external application. An expectation of the Bluetooth module is the wireless communication should work within 8 meters.

This module does not need calibration to set up correctly, the initialization of the

Communication subsystem contains the setup of the Bluetooth module and test communication between the system controller and HC-05 and the external application. The below is the list of tests that were conducted to evaluate and guarantee all of the functions and expectations:

1. Test – start initializing the UART serial communication to the Bluetooth module
    a. Test Method – test module developed on TM1294 that handles initializing UART communication to send string “Test SENT”. Module receives and send the same message indicating that UART works correctly.
    b. Expected Result – LED that is turned on and emits a red color notifies the user. Next, we see “Test SENT” displayed twice on serial terminal.
2. Test – initialize the Bluetooth communication to external application
    a. Test Method – Send test string “B_Message SENT” and display it in the serial terminal. External app receives message and check if that if it receives all correct characters and information is not lost. After that passes, external app sends to main controller: “E _Message RECIEVED” and main controller displays it in serial terminal.
    b. Expected Result – “B_Message SENT” and “E _Message RECIEVED” is displayed on the serial terminal. LED changes to a green color where the communication subsystem
3. Test – collect both long information messages and external application.
    a. Test Method – Test module by external app sending long message, main controller receiving message, displaying it in serial terminal, then sending back the same message to external application. The external application should receive same message with no data loss.
    b. Expected Result – collect long message in the format of 8bit ASCII characters, see printed in terminal. External app prints the received message in either a pop up window or LogCat Android debug window.
4. Test – collect streaming information passed from external application.
    a. Test Method – Test module by communication system waiting in background, external app sending message to start streaming. Once Main controller enables streaming and confirms by sending message back to external app, the external app is able to pass rotational information for all servos and whenever the user changes position on app, the servo on prosthetic should change within time response of 1 millisecond.
    b. Expected Result – External app should be able to control all servos in real time or under the latency of one millisecond.
5. Test – reset communication subsystem
    a. Test Method – press the hardware reset button
    b. Expected Result – pressed reset button, see LED emit red.

If the user presses the reset button, the communication system switches state from COMMUNICATION STATE to INITALIZATION STATE.

6.5 Software Testing

This section describes the tests involved with verifying the functionality of the software written for the various modules and subsystems. The major subsystems to test are the sensor processing module, the servo controller module, the system controller module, and the training mode module. Each test consists of an intention, method, and expected result.

6.5.1 System in the Arm

Sensor Module Testing The sensor module test verifies that the sensor processing microcontroller is correctly reading and processing the incoming data from the EMG sensor, the PIR distance sensor, and the FSR sensors.

1. EMG Sensor Test – Verifies the EMG sensor is being correctly read
    a. Test Method – If the microcontroller detects that the EMG sensor has changed from a ‘relaxed’ reading to a ‘flexed’ reading, light an LED
    b. Expected Result – When the wearer of the electrodes flexes his/her muscle, the LED lights up

2. PIR Distance Sensor Test – Verifies the PIR sensor is being correctly read
    a. Test Method – If the microcontroller detects that an object is less than ½ inch from the PIR sensor, light an LED
    b. Expected Result – When an object is placed less than ½ inch from the sensor, the LED lights up

3. FSR Sensor Test – Verifies the FSR sensors are being correctly read
    a. Test Method – If the microcontroller detects a force above a certain threshold, light an LED
    b. Expected Result – When a weight above the threshold is placed on the FSR, the LED lights up

Servo Module Testing The servo module test verifies that the servo controlling microcontroller is correctly positioning the servos.

1. Iterative Finger Control Test – Verifies each finger correctly moves from completely open to completely closed, without positioning past the maximum flex of the fingers.
    a. Test Method – Iterate through each finger, moving each from totally open, to closed, to open again.

b. Expected Result – One at a time, each finger should close to the point that it cannot close any more, but stop without going further, then re-open.

System Controller Module Testing The system controller module tests verifes that the system controller is correctly communicating with the servo controlling microcontroller and the sensor processing microcontroller.

1. Servo Controller Communication Test – Verifies the system controller can transmit servo positioning information to the servo controller
    a. Test Method – Send a gesture to the servo controller and verify the servos move the fingers to that position
    b. Expected Result – The servos position the hand into the correct gesture

2. Sensor Processing Communication Test – Verifies the system controller can receive sensor information from the sensor processing microcontroller
    a. Test Method – Program the sensor processing microcontroller to transmit data to the system controller, if the system controller receives the correct data, light an LED
    b. Expected Result – The system controller lights up the LED

Training Mode Testing The training mode tests verifies that Bluetooth communication between the smartphone app and the system controller is functioning correctly

1. Packet Send Test – Verifies that the system controller can receive a data packet
    a. Test Method – Have the smartphone app send a packet of data to the system controller. If the packet matches a specific pattern, light an LED.
    b. Expected Result – The system controller lights up the LED

6.5.2 Mobile Application

The mobile application is one of the major contributions from this project to the area of 3D printed prosthetics. Therefore it is important that it is fully functional and that it is accessible and easy to interact. Most of the tests require input from a user. The responsiveness of the application is also tested since this reflects on the user's experience. In order to determine if the mobile application is functioning properly the requirements listed in Table 16 must be met.

Requirement	Description
1	The app must show if the mobile device is connected with the IPPA system
2	Instructions to establish connection must be available
3	The app must request the IPPA system to switch to Teaching Mode
4	The app is not in Teaching Mode when interpreting voice commands
5	The app must be able to do speech recognition
6	The app must be able to save new gestures
7	The app must be able to transfer information to the IPPA over the established connection
8	The app must be responsive at all times
9	The app must transfer data quickly, especially during the creation of a gesture. The arm must provide feedback

Table 16. List of software requirements for the mobile application.

Unit tests were written to make sure the small pieces of the application work properly. More extensive testing was be done to verify that all parts of the UI and the back end functionality work properly. As detailed in section 4.5.3, there are multiple types of packages that are sent to the IPPA system. For every test, it is implied that the UI components were checked for correctness. The tests must incorporate all the different types of packages. Some of the tests do not require connection with the IPPA system, more testing related to this was done in section 6.7. The following tests have been designed to test all of the requirements previously shown:

1. Download the application into an Android device and launch it. Verify that the application is properly installed and launches.

2. Download the application into an Android device and launch it. Test that each button in the main page works by pressing it. Verify that the user is taken to a different activity and that the standard back button works properly.
    a. For the communication button, instructions should be displayed
    b. For the voice commands button, a speak button and an empty text view should be displayed
    c. For the teaching mode, a dialog should pop asking for confirmation; confirm it. Then two tabs should be displayed, with the one on the left as selected.
3. Download the application into an Android device and launch it. Test that the voice commands page works. Press on the voice commands button. Then press on the speak button.
    a. Say “Open”. Verify that the text “open” is displayed in this page.
    b. Repeat, and say “Close”. Verify that the text changes from “open” to “close”.

   c. For both a and b, verify that the information is transferred to the IPPA system. A temporary LED flashes quickly when data is being transferred.

4. Download the application into an Android device and launch it. Test that the teaching mode page works. Press on the demo gestures tab on the top left.
   a. Press any item listed. Select delete from the displayed options. Verify that the item is deleted from the demo gesture's list
   b. Press any item from the list that is stored in the phone. Select move to arm from the displayed options. Verify that the gesture is transferred to the arm by using the voice commands to trigger it.
   c. Press any item from the list that is stored in the arm. Select demo gesture from the displayed options. Verify that the gesture is transferred to the arm, and automatically triggered.

5. Download the application into an Android device and launch it. Test that the teaching mode page works. Press on the create gestures tab on the top right.
   a. Create a gesture with the default start position for the fingers. Save it. Repeat test 4 for this particular gesture.
   b. Create a gesture with a different start position for the fingers. Save it. Repeat test 4 for this particular gesture.
   c. Start creating a test. Then press the reset button. Verify that the hand moves back to the open position and the sliders in the page are reset as well.

6.6 Calibration

Depending on the results discovered in the tests above, further calibration may be required. More specifically, adjustments may need to be made on the cutoff thresholds for the FSR, EMG, and PIR sensors.

FSR Cutoffs The implementation of force sensors in the IPPA design was to prevent the IPPA from exerting dangerous levels of force on itself or others. If the tests reveal that the hand is able to apply large amounts of pressure on objects in its grasp, the cutoff threshold must be lowered. However, it is possible that the threshold could be set too low, which could lead to the hand loosening its grip prematurely.

EMG Sensor Cutoffs The EMG sensor must be able to detect when the wearer flexes his/her muscles. If it is discovered that the EMG sensor is not triggering when a user flexes, the threshold must be lowered. If the EMG sensor is triggering when the user is not flexing, the threshold needs to be raised.

PIR Sensor Cutoffs The PIR distance sensor requires activation at a very specific distance. We need to verify that the sensor triggers if and only if an object is within ½ an inch from the sensor. Using a ruler and a digital multimeter, the team can determine the correct reading to use as the cutoff threshold.

6.7 Final Integrated Tests

The final integrated test was conducted after all of the subsystems have been individually tested. At this point the entire hand and forearm was be assembled, and the PCB for the IPPA system was in place. No tests were conducted on an amputee, to avoid injury the tests were conducted by the project engineers who know how to operate the IPPA system. The final test consisted of general daily use intended for the IPPA system. This test must verify that all of the hardware and software specifications in section 1.2 are met. The tests' environment was: the IPPA was secured on a base and the EMG sensor was placed on the right hand of one of the project engineer. The base was constructed from wood; the arm just placed on it. The following list describes these tests:

1. The test subject moves right arm muscles to fully open the hand.
    a. Verify that the IPPA opens to a full extend

2. The test subject moves right arm muscles to fully close the hand.
    a. Verify that the IPPA flexes all the fingers entirely

3. Download the IPPA mobile application onto an Android phone. Launch the application and select the button for the Bluetooth connection instructions.
    a. Go to the phone settings and follow the instructions to connect to the Bluetooth in the IPPA system.
    b. Go back to the app and verify that the connection established icon is displayed.

4. Download the IPPA mobile application onto an Android phone. First connect the phone to the IPPA system.
    a. Launch the application and select the button for voice commands.
    b. Press the command button and say "Open".
    c. Verify that the application displays the correct command and that the IPPA opens the hand.

5. Repeat test 4, with the command "Close".

6. Download the IPPA mobile application onto an Android phone. First connect the phone to the IPPA system.
    a. Launch the application and select the button for teaching mode.
    b. Verify that a dialog pops up for confirmation.
    c. Confirm it.

   d. Select the Demo Gesture tab and click on an item already stored in the IPPA system.
   e. Select the “play demo” option for the gesture to be performed.
   f. Verify that the expected gesture is done by the arm.

7. Download the IPPA mobile application onto an Android phone. First connect the phone to the IPPA system.
   a. Launch the application and select the button for teaching mode.
   b. Verify that a dialog pops up for confirmation.
   c. Confirm it.
   d. Select the Demo Gesture tab and click on an item that is stored in the phone. Select the “move to arm” option for the gesture to be transferred to the IPPA system.
   e. Use the voice commands to trigger the gesture that was just transfer.
   f. Verify that the expected gesture is performed by the arm.

8. Download the IPPA mobile application onto an Android phone. First connect the phone to the IPPA system.
   a. Launch the application and select the button for teaching mode.
   b. Verify that a dialog pops up for confirmation.
   c. Confirm it.
   d. Select the Create Gesture tab.
   e. Create a new gesture: thumb up. Leave the default start position, and use the word “thumb” as the voice command.
   f. Save gesture.
   g. Go to the Demo Gesture tab, and repeat tests 7 and 6 for this particular gesture.

7 Design Constraints and Standards

This section describes the different design constraints and a sample of standards that apply to the IPPA.

7.1 Design Constraints

Economic One of the goals of this project was to lower the cost of functional available prosthetics. Therefore, keeping the cost of the IPPA to under $1000.00 was crucial.

Environmental Because we use wireless communication, an important environmental impact is the frequency at which this transmission happens. We used Bluetooth technology, has standards associate with it that already enforce environmental design constraints; such as the IEEE 802.15.1-2002 standard (mentioned in 7.2).

SocialThe IPPA will be used to interact with object but also with humans, having a direct impact on the amputee's social interactions, because of this the team choose a hand design that looks as much natural as possible compare to other designs.

Political The IPPA has no political impact or influence.

Ethical The IPPA has been designed to properly work and has been tested to the best of our abilities. There is no financial or other gain obtained from someone using the arm.

Health and Safety Safety is very important when developing robot-types of systems. The IPPA includes force sensing sensors in order to obtain some feedback on how strongly an object is being held. This avoids the crushing of objects or a human hand during a hand shake.

Manufacturability The IPPA is a complex system. Even though a PCB has been design for its electrical components; the mechanical components of the arm (the arm itself) require a long and detail process for assembling.

Sustainability This project used ABS to build the arm, which is a very durable type of plastic. In addition, a benefit of using 3D printed parts for the arm is that if any part were to break; it could be printed and only a subarea of the arm would be affected.

7.2 Standards

This is a small recollection of standards that apply to our project. Due to the large number of standards per component use, we limit the number of standards included in this document.

- IEC 60086-2 Ed. 12.0 b:2011
  - "IEC 60086-2:2011 is applicable to primary batteries based on standardized electrochemical systems. It specifies the physical dimensions; and the discharge test conditions and discharge performance requirements. Significant changes from the previous edition are the deletion of eight battery types from this standard, the addition of an air hole placement diagram and deletion of the resistive hearing aid tests for the P-system (zinc air) hearing aid batteries, standardization of a new form of alkaline (L-system) 9 volt battery (6LP3146), addition of a common designation reference as Annex D and general adjustment of application tests and their minimum average duration values to reflect changes in battery usage."
  - Since we opted to use standardized 9V batteries in our design, this standard is applicable.

- IEC 62133 Ed. 2.0 b:2012
  - IEC 6213 3:2012 specifies requirements and tests for the safe operation of portable sealed secondary cells and batteries (other than button) containing alkaline or other non-acid electrolyte, under intended us e and reasonably foreseeable misuse.
  - Included since we utilized a lithium ion battery for a mobile application.

- ASTM B286-07(2012)
  - Standard Specification for Copper Conductors for Use in Hookup Wire for Electronic Equipment
  - Included since we used hookup wire to connect our individual components.

- ISO 12224-1:1997
  - Solder wire, solid and flux cored -- Specification and test methods -- Part 1: Classification and performance requirements
  - Related to flux cored solder, which we used to permanently mount/join components.

- RS-232
  - Standard for serial communication transmission of data. The RS-232 standard is commonly used in computer serial ports.

  - The standard defines the electrical characteristics and timing of signals, the meaning of signals, and the physical size and pinout of connectors.
  - Because our system communicates and interfaces between the servo controller's ATMega328p MCU at a baud rate of 115200 and the HC-06 at a baud rate of 9600, this standard is applicable.

- IEEE 802.15.1
  - Standard for Bluetooth wireless communication. The standard defines the lower transport layers (L2CAP, LMP, Baseband, and radio) of the Bluetooth™ wireless technology. Bluetooth is an industry specification for short-range RF-based connectivity for portable personal devices.
  - Because we used Bluetooth communication to interface between the mobile phone application and the IPPA system, this standard was applicable.

- ISO/IEC TR 18037:2004
  - A standard specifies a series of extensions of the programming language C, specified by the international standard ISO/IEC 9899:1999. The standard includes am approach to codifying common practice and providing a single uniform syntax for basic basic I/O hardware (iohw) register addressing.
  - This standard was included since the IPPA system was implemented using the C programming language.

8 Administrative Content

Since this is a large and expensive project it is very important to make a detail finance budget as well as a well organize and realistic plan. This chapter discusses the tasks needed to complete this project, their projected completion time, and budget for materials and unexpected expenses.

8.1 Milestones and Project Planning

The planning of this project was done right at the beginning of the project, but it changed throughout. Flexibility was needed in order to accommodate for unexpected issues that could come up. This scheduling is essential to the successful completion of the project on time. The project expanded across approximately 8 month, starting in September 2014 and ending in late April 2015. A Gant chart has been used to organize and plan all tasks related to the project.

Each person in the team has a designated color to identify their corresponding task: Matt Bald is identified by green, Ivette Carreras is identified by orange, and Andrew Mendez is identified by blue. Some tasks were completed by multiple members in the team in which case dark red was used. Black represents the final deadline for the deliverables.

Figures 52 - 57 show the complete schedule of the project for the Research, Design, and Documentation phases of the project. Figures 50 - 55 show the tentative schedule for the Development and Build, and Test phases of the project. All the tasks have been distributed among the team engineers. The task assigned to each engineer reflects their areas of knowledge, as well as new areas that are of some interest to that person.

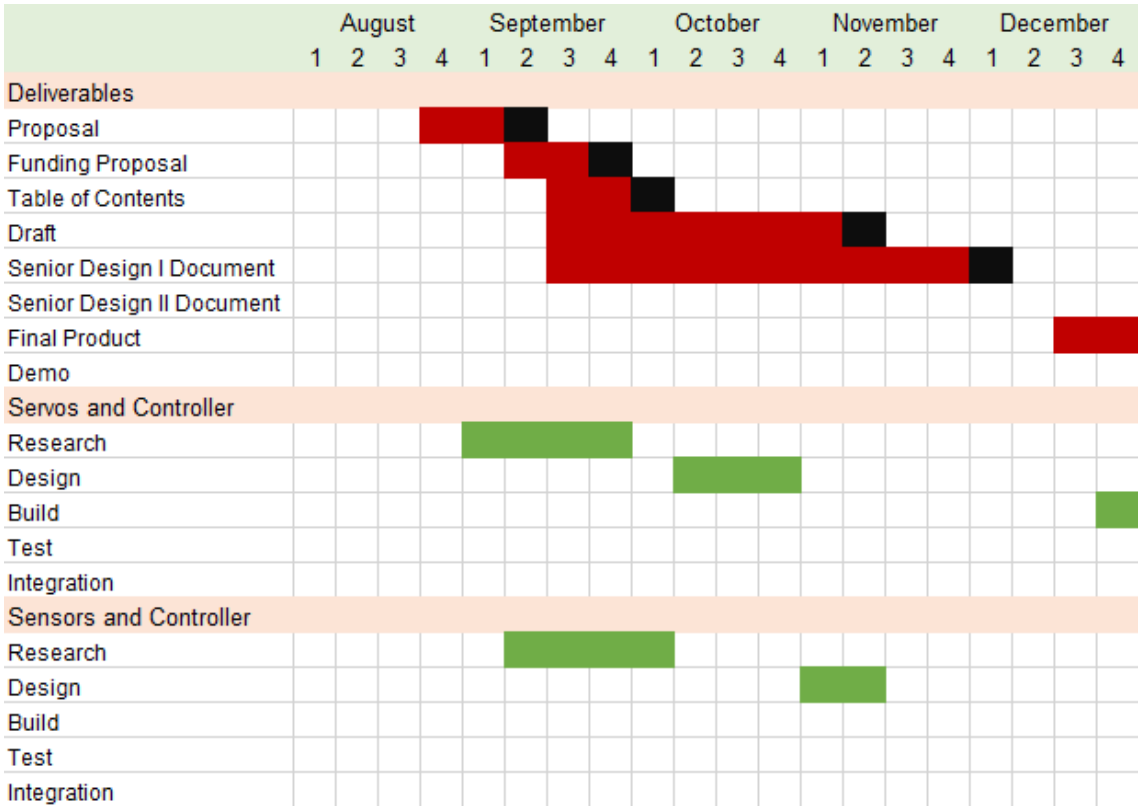

Figure 52. Milestone Chart. First semester for the project with schedule for the deliverables, servos and their microcontroller, and sensors and their microcontroller.

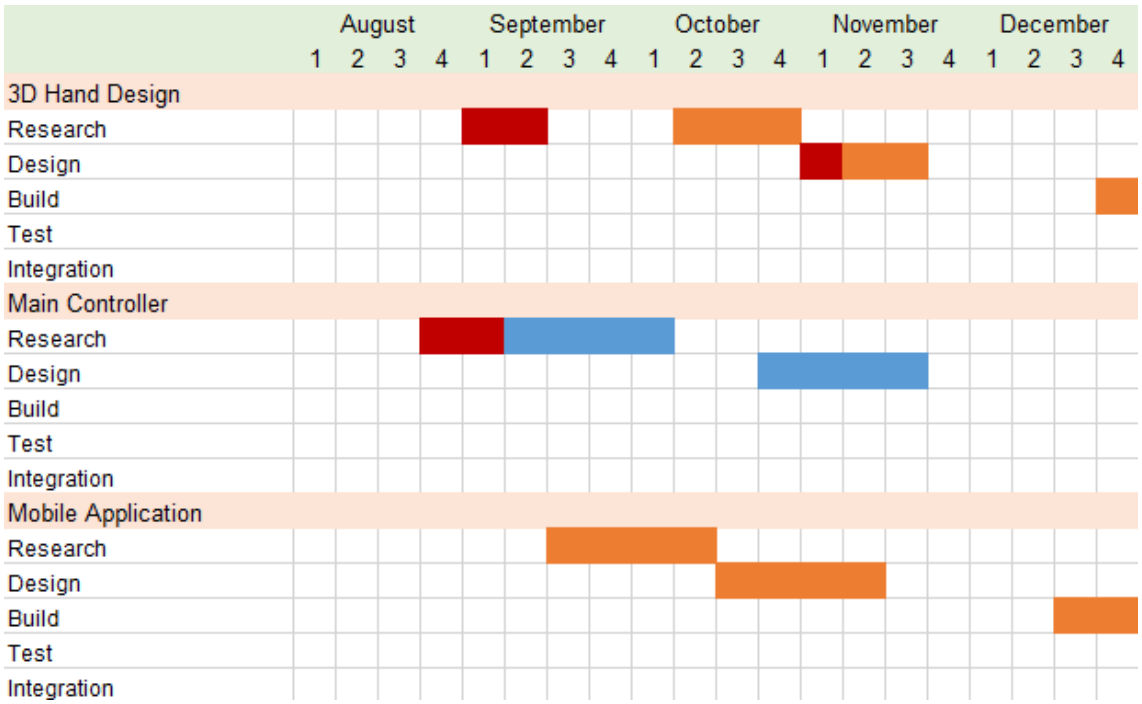

Figure 53. Milestone Chart. Continuation of the first semester schedule for the project; it includes the 3D hand physical design, main system controller, and mobile application.

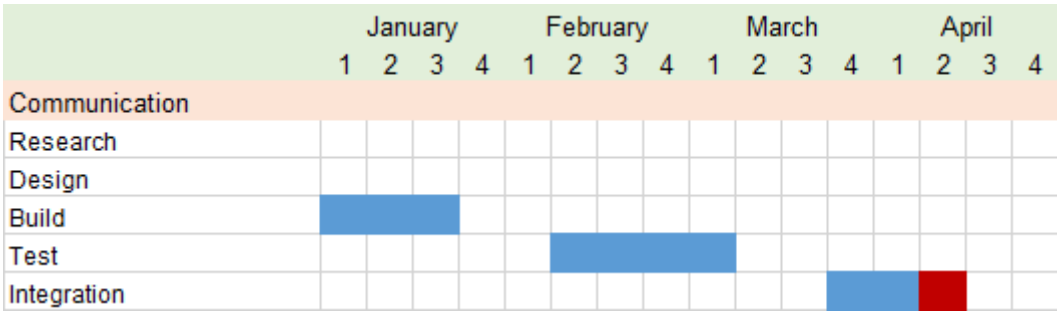

Figure 54. Milestone Chart. Continuation of the first semester schedule for the communication subsystem for the project.

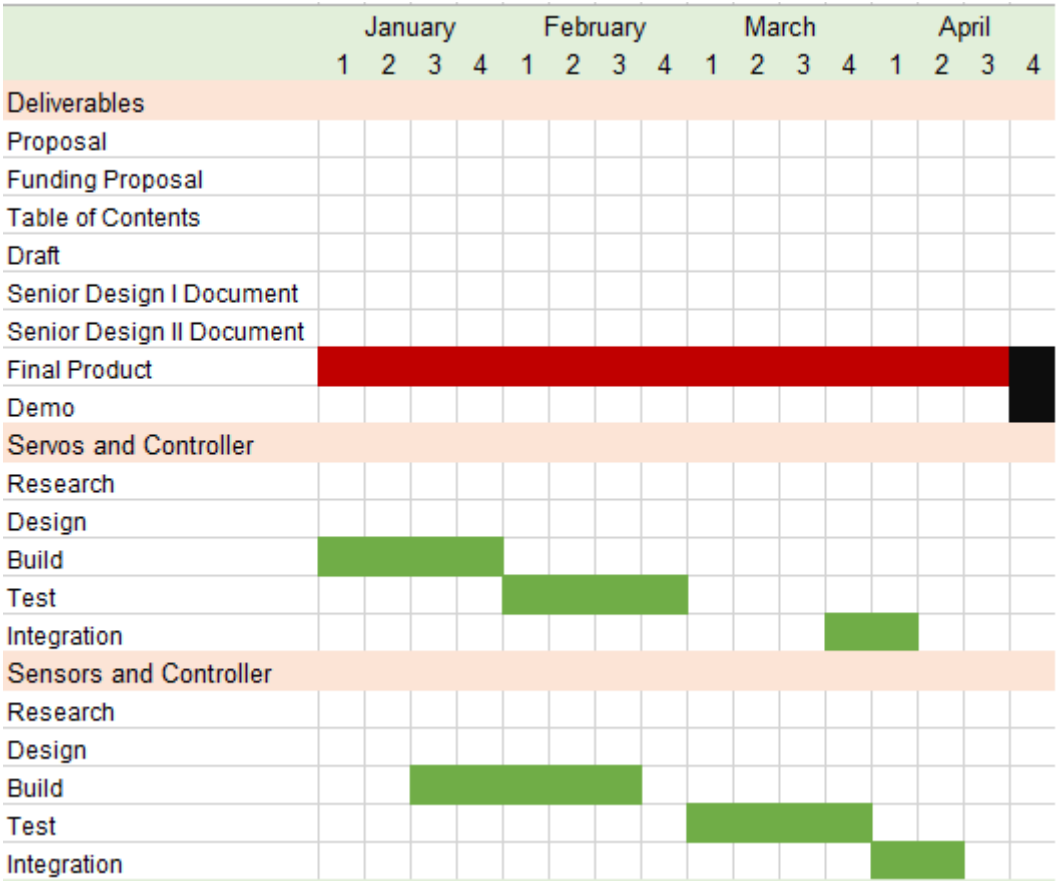

Figure 55. Milestone Chart. Second semester for the project with schedule for the deliverables, servos and their microcontroller, and sensors and their microcontroller.

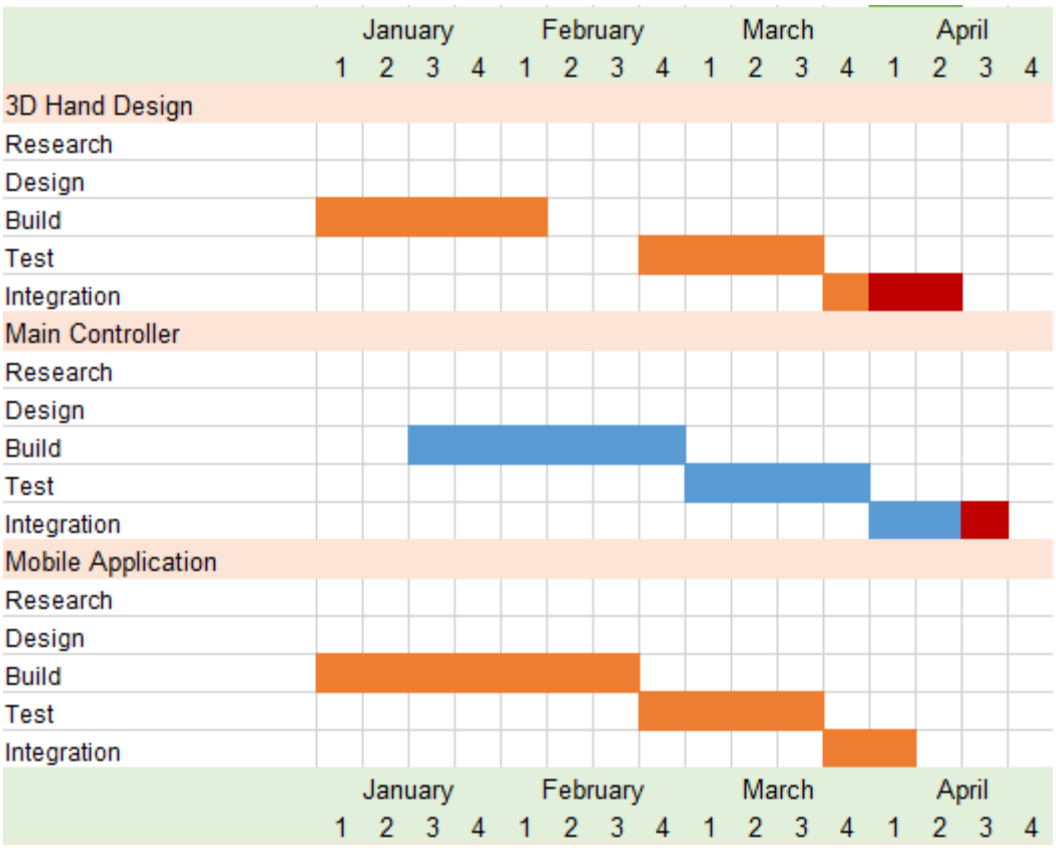

Figure 56. Milestone Chart. Continuation of the second semester schedule for the project; it includes the 3D hand physical design, main system controller, and mobile application.

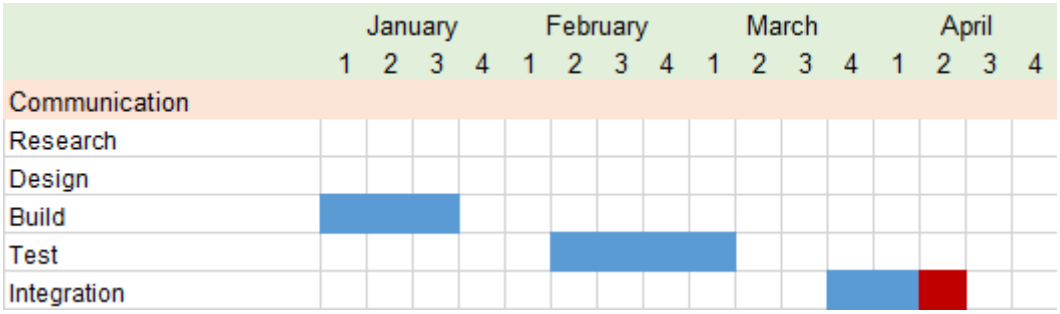

Figure 57. Milestone Chart. Continuation of the second semester schedule for the communication subsystem for the project.

8.2 Budget and Finances

The total cost of the project is shown below, in Table 17. A tradeoff was presented in the form of price versus performance. This was especially evident in the choice of servos to use. We have chosen to use strong servos that would best resemble the use of a real, human arm. This resulted in the servo motors, battery, and 3D printing becoming the most expensive investments of the project. However, the

team was able to satisfy the requirement to keep the cost of the arm to under $1000.

Quantity	Component	Individual Cost ($)	Total Cost ($)
5	Servos	19.99	99.95
1	TM4C1294	19.99	19.99
2	ATMega328p	6.87	13.74
2	Force sensitive resistors	5.95	11.90
2	Infared Emitters and Detectors	1.95	3.90
1	7.4V Rechargeable Battery & charger	111.95	111.95
1	Fishing Line	10	10
1	Grip material	5	5
1	Printed Circuit Board	30	30
1	Bluetooth module	8.99	8.99
2	5V Voltage Regulators	.67	1.34
2	3.3V Voltage Regulator	.67	1.34
5	LM317 Adj. Voltage Regulator	0.67	3.35
1	3D Printed Hand & forearm, 5 lbs. of ABS plastic	300	300
2	25 MHz Crystal	.53	1.06
4	16MHz Crystal	.56	2.24
1	EMG module	50	50
25 Components Total			**674.75**

Table 17. Total projected cost of the IPPA.

9 Conclusion

Considerable obstacles that hinder current prosthetic arms are their expensive to acquire, difficult to adjust to, and current advanced prosthetic arms are not affordable and capable to achieve a variety of tasks similar to the human hand.

The Intelligent Programmable Prosthetic Arm project's goal is to provide a fully functional low cost prosthetic, as well as providing the correct support for those starting to learn how to send electromagnetic signals to their new limb. This project is targeted towards people who are missing a hand, wrist, and part of their forearm and not a full arm. In addition to also satisfy the challenge of developing a single prosthetic that satisfies each individual, the IPPA's also includes a mobile application that allows the amputee to change the features in the arm from an available list or create their own and unique arm movement or hold patterns.

The design entails the hardware and software design of the Servo Controller, Sensor Processing Controller, the Mobile Application, System Controller, and the Power System. The Servo Controller controls the servos linked with each individual finger of the prosthetic. The Sensor Processing Controller reads and analyzes the information from the sensors to automate grasping. The Mobile Application provides a way to change the settings for the gestures/grasps as well as the triggering mechanism for the gestures. The System Controller that directs the Servo Controller to the correct gesture, gather and interpret data from the Sensor Controller, and transmit and receive data from the Communications Subsystem. The Power System to supply power to the Servo Controller, Sensor Processing Controller, and the System Controller.

WE utilize pressure sensor to enable the system to stop grasping when it detects dangerous levels of pressure. We utilize the distance sensor to automatically grasp when an object is less than or equal to 1 inch. We utilize an EMG sensor to automate when to grasp or release an object.

The system utilizes two ATmega328P micro-controllers as the servo controller and sensor controller. We use the TM4C123GH6PMT as the main controller. The system has two modes, Autonomous mode and Teach Mode. Autonomous mode performs automated grasping tasks and perform a wide range of hand gestures. Teach Mode allows the user (amputee) to change settings, hand gestures, and gesture triggering mechanism.

References

[1] Adafruit. BeagleBone Black https://www.adafruit.com/products/1876,

[2] Advancer Technologies. Muscle Sensor v3 Datasheet. https://docs.google.com/file/d/0B8Wy2qiwirwyNFV1dlA2T1JVZFU/edit?pli=1

[3] All-Battery. 6V 10,000 mAh Battery Pack. http://www.all-battery.com/nimh6v10000mahbatterypackwithbareleadscustomize.aspx

[4] Atmel. ATmega328P Datasheet. http://www.atmel.com/Images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Summary.pdf

[5] BeagleBoard. BeagleBone Black. : http://beagleboard.org/black

[6] Belter JT, Segil JL, Dollar AM, Weir RF. Mechanical design and performance specifications of anthropomorphic prosthetic hands: A review. J Rehabil Res Dev. 2013; 50(5):599–618. http://dx.doi.org/10.1682/JRRD.2011.10.0188

[7] Bluetooth HC-06 with serial port module Easy guide. http://www.puntoflotante.net/BLUETOOTH-HC-06-WITH-SERIAL-PORT-EASY-GUIDE.pdf

[8] ComScore. July 2014 U.S. Smartphone Subscriber Market Share. http://www.comscore.com/Insights/Market-Rankings/comScore-Reports-July-2014-US-Smartphone-Subscriber-Market-Share

[9] Crossplatform. Running MSP430 on a Breadboard. http://crossplatform.net/running-msp430-launchpad-on-a-breadboard/

[10] Duracell. 9V Coppertop Datasheet. http://ww2.duracell.com/media/en-US/pdf/gtcl/Product_Data_Sheet/NA_DATASHEETS/MN1604_6LR61_US_CT.pdf

[11] Elecrom.wordpress.com. Omkar. How to make simple Infrared Sensor Modules. http://elecrom.wordpress.com/2008/02/19/how-to-make-simple-infrared-sensor-modules/

[12] Estimating the Prevalence of Limb Loss in the United States:2005 to 2050 PAPER

[13] Forbes. TJ McCue. 3D Printed Prosthetics. http://www.forbes.com/sites/tjmccue/2014/08/31/3d-printed-prosthetics/

[14] Fuse Bits. Workit!- an Arduino based wristband motion controller. http://ssarbora.com/2014/05/18/workit-an-arduino-based-wristband-motion-controller/#more-43

[15] IDC. Smartphone OS Market Share, Q3 2014. http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[16] inMotion. Introduction to Upper-Limb Prosthetics: Part 1. http://www.amputee-coalition.org/inmotion/mar_apr_07/upper_prosth_pt1.html
[17] Mayo Clinic. Electromyography Definition. http://www.mayoclinic.org/tests-procedures/electroconvulsive-therapy/basics/definition/prc-20014183
[18] Mouser Electronics. Texas Instruments CC3100 & 3200. http://www.mouser.com/new/Texas-Instruments/ti-cc3100-cc3200-wifi-processor/
[19] NASA. Human Performance Capabilities. http://msis.jsc.nasa.gov/sections/section04.htm
[20] Pial. Using the HC-05 Bluetooth RS232 Serial Module. http://www.pial.net/using-the-hc-05-bluetooth-rs232-serial-module-for-cheap-wireless-communication-with-your-ucontroller/
[21] Pololu. 1501MG Datasheet. http://www.pololu.com/file/0J729/HD-1501MG.pdf
[22] Rose. Wireless Communication. Helen Fornazier, Aurelien Martin, Scott Messner. http://rose.eu.org/2012/wp-content/uploads/2012/03/Wireless-communication.pdf
[23] Sirius. Motion & PIR Sensor Technology. http://www.ecosirius.com/technology.html
[24] Sparkfun. Force Sensitive Resistor Tutorial. https://www.sparkfun.com/tutorials/269
[25] Texas Instruments. CC3200. http://www.ti.com/product/cc3200 ,
[26] Texas Instruments. MSP430G2x53 Datasheet. http://www.ti.com/lit/ds/symlink/msp430g2553.pdf
[27] University of Southampton. R M Crowder. Tactile Sensing. http://www.southampton.ac.uk/~rmc1/robotics/artactile.htm
[28] Worcester Polytechnic Institute. IrisHand. https://www.wpi.edu/Pubs/E-project/Available/E-project-043014-213851/unrestricted/IRIS_HAND_MQP_REPORT.pdf

Appendix

Copyright Permissions for Images Used

Melissa Peloquin
to ivette.carreras

Thu, Nov 20 11:03 AM

Re: Request Permission to use Pictures

Good afternoon Ivette,

We are happy to provide you with permission to use the images that you requested. Please do not hesitate to contact me if you require additional information or need a higher quality resolution of the image.

Kind regards,

Melissa

touch bionics

Melissa Peloquin – Vice President Operations
North America

Toll Free (855) MY iLimb x210
Mobile: (508) 254-6972
Fax: (508) 546-1288
Email: melissa.peloquin@touchbionics.com
Web: www.touchbionics.com

Touch Bionics Inc.
35 Hampden Road, Mansfield, MA 02048

Figure 58. Permission from touch bionics.