Intelligent Programmable Prosthetic Arm (IPPA)

Matthew Bald, Ivette Carreras, and Andrew Mendez

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

Abstract — **The objective of this project is to provide a low cost prosthetic arm with advanced functionality and features that compare to commercial prosthetics. This is accomplished by utilizing 3D printing technology and off the shelf electronic devices that incorporate multiple features such as automatic grasping, pointing and other natural gestures that are standard in expensive prosthetics. The IPPA 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.**

Index Terms — **Prosthetic arm, microcontroller intercommunication, electromyography sensor, infrared emitter, pressure sensor, Bluetooth, mobile application.**

I. Introduction

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. Even then, there are not that many commercial prosthetic hands in the market that take full advantage of the technologies available today [1]. The latest and most advanced 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 include in the prosthetic. This drastically reduces the availability of prosthetic technology for children and adults all over the world.

3D printed arms are part of a current trend to provide solutions at a more affordable price. However, most of those limbs are very limited in their functionality. 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 $1,000. The Intelligent Programmable Prosthetic Arm (IPPA) contains multiple sensors that allow it to perform automatic grasping of objects, gentle handshakes, and other one-motion-gestures.

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 will satisfy 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. This project is targeted towards people who are missing a hand, wrist, and part of their forearm; not a full arm.

II. Subsystems

The IPPA system is best presented in terms of subsystems; that is, the five modules—whether purchased or designed—that are interfaced to create the final product. This section describes the software and hardware design overviews of the Servo Subsystem, Sensor Subsystem, Main System Controller, Communication Module, and the Power Subsystem.

A. Servo Subsystem

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.

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. Each finger is controlled by Pololu 1501MG series servos. The strength of the servos is adequate (16 kg*cm), since according to a NASA study, the average adult male hand is capable of producing about 8 kg*cm of torque [4]. These servos 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 1 below shows the ATmega with the required power supply, ground, and reset pin wired.

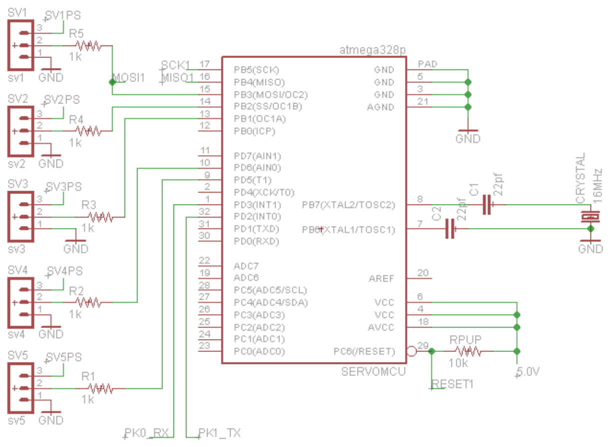


Fig. 1. A schematic of how to wire five servos to the ATmega328P.

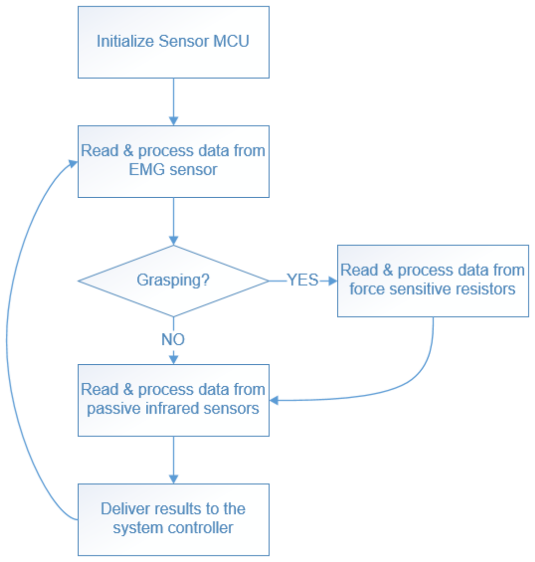
B. Sensor Subsystem

This subsystem is composed of three different sensors: EMG, pressure, and distance. The electromyography (EMG) sensor allows 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 (PIR), to detect when an object is near the hand. The hardware components used for these sensors are listed in Table I.

TABLE I
SENSOR COMPONENTS

Type	Part	Quantity
EMG	Advancer Technologies Muscle Sensor v3	1
PIR	Constructed: 470Ω resistor, a 47nF capacitor, an infrared emitter, and an infrared sensor	1
C	FSR400 by Interlink Electronics	3

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. 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. Figure 2 shows the flow of the servo microcontroller software.



Initialize Sensor MCU
Read & process data from EMG sensor
Grasping?
YES
Read & process data from force sensitive resistors
NO
Read & process data from passive infrared sensors
Deliver results to the system controller

Fig. 2. Flowchart showing the algorithm in the servo controller.

Each sensor is crucial for the objective of project to incorporate autonomous behavior of the prosthetic. The EMG sensor gives full control to the person using the prosthetic by allowing them to use their muscle impulses to open and close the hand. 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.

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. In the sensor controller an output pin is set high to indicate an action. A threshold on the change in value approach is used to determine when to set the pin. Since a spike-like behavior was observed when a muscle is contracted and relaxed, the PIN state only changes on the increasing side of the spike. This supports the possibility of the user not having to hold the flexion of their muscle.

The force sensing resistors have been added to assist the person with the grasp/gesture being performed. Measuring the pressure exerted at different points in the hand provides some feedback about the object in the hand.

The resistors sensing pad is circular, with an area of 0.3 square inches. A voltage dividing circuit is required to determine the resistance of the sensor, which will change when force is applied to the sensor. Since multiple force sensitive resistors were used, each one required a voltage division circuit and an available input pin on the multiplexer. The ATmega uses a GPIO to measure the voltage on the resistor, then applies equation (1) to determine the resistance, which is then converted to grams using equation (2).

$$V_{out} = \frac{R_M * V^+}{R_M + R_{FSR}} \quad (1)$$

$$R_{FSR} = \left(\frac{R_M * V^+}{V_{R_{FSR}}}\right) - R_M \quad (2)$$

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. The design was taken from [2], Figure 3 shows the schematic to construct the PIR sensor.

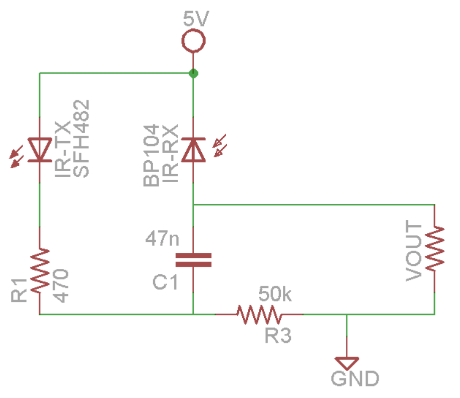


Fig. 3. A schematic of the PIR sensor circuit.

This sensor is used to decide if an object of interest is very close to the hand. This sensor provides another feature to the autonomous mode of the IPPA. The goal is to facilitate the triggering of grasping motions. Since the natural motion of humans is to position the hand near an object and then grab it, the IPPA has this distance sensor in the middle of the palm. Once the distance between the palm and the object is less than ~1 cm, the output pin that corresponds to the distance sensor is set high for the main controller to trigger the grasp. To connect the distance sensor to the microcontroller, a wire was attached from the top of the Vout resistor, to an available input pin on the multiplexer.

C. Main Controller

System 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 decides how to proceed with the current grasp and gesture being performed. 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 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 gestures the prosthetic hand can complete. The main controller has two different running modes: autonomous and teaching mode. Figure 4 shows the overview of the flow between these two modes.

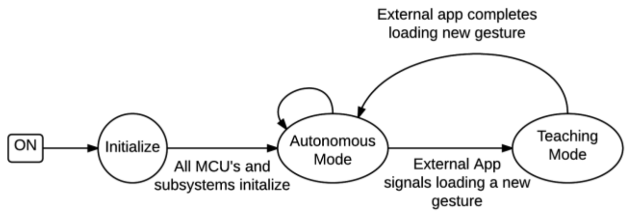

External app completes loading new gesture
ON
Initialize
All MCU's and subsystems initalize
Autonomous Mode
External App signals loading a new gesture
Teaching Mode

Fig. 4. Flowchart showing the general algorithm in the main controller.

The main controller initializes by initializing the UART communication between the servo controller and Bluetooth communication subsystem. Also, the GPIO pins that interface the sensor controller are initialized. Once the main controller is initialized and the Bluetooth communication subsystem is initialized, the main controller starts in autonomous mode.

D. Communication

The communication with the IPPA system from the mobile application 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. Different package types have been designed to reduce the amount of information that needs to be transmitted and to facilitate the understanding between devices. There is a total of 10 packages, which are listed in Table II.

The communication subsystem is composed of the main controller the Bluetooth HC-06 module, and the external mobile application. The Bluetooth module transmits and receives information from the main controller via hardware UART.

TABLE II
BLUETOOTH INTERFACING PACKAGES

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	Full gesture
D	Delete an existing gesture	Gesture identifier
E	Temp gesture storage and trigger	Full gesture
F	Request command strings stored in the arm	N/A
G	Switch IPPA mode	N/A
H	Send voice command in arm	Variable # of strings
I	All gestures stored in the arm	Variable # of gestures

E. Power Subsystem

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. The total current drawn by the system sums up to a minimum of about 650 mA and a maximum of about 3000 mA. The current draw may increase to around 6000 mA if the user is lifting an object of moderate weight. The worst case scenario is if the user attempts to lift an object beyond the capabilities of the servos, which would cause the system to shut down due to a lack of power. The battery we have chosen to use contains 7800 mA hours of charge. It can discharge about 7 amps maximum. Assuming the servos will not be continuously moving, the lifespan of this battery should exceed 1 hour. Table III lists the power requirements for each component used.

The intelligent programmable prosthetic arm utilizes two different power sources in order to power the various subsystems. A pair of 9 volt batteries will be responsible for

TABLE III
COMPONENTS POWER REQUIREMENTS

Component	Voltage	Current	Power
EMG [7, 8]	± 5V	1.8mA	8.8mW
FSRs (Rm = 27 kOhm)	5 V	0.2 mA	1.0 mW
PIR [5]	5 V	6 µA	0.03 mW
Servo MCU [9]	5 V	16 mA	80 mW
Sensor MCU [9]	5 V	16 mA	80 mW
Main MCU [6]	3.3 V	80 – 320 mA	0.3 – 1.2 W
Servo (idle) [6]	6 V	5 mA	30 mW
Servo (no load) [6]	6 V	0.1 – 0.5 A	0.6 – 3 W
Servo (stalled) [6]	6 V	2.5 A	165 mW
Bluetooth [5]	3.3 V	50 mA	165 mW

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.

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. There were five 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. The passive infrared sensor requires a 5V power supply as well. These two sensors draw very little current, about 10 mA. The MIC5205 satisfied 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. The EMG sensor will be powered separately by two 9V batteries which will create a +9V and -9V for the sensors V+ and V- terminals. Five LM317 voltage regulators, adjusted to 6V, are responsible for powering their own servo. Each servo having its own regulator reduces the heat generated in each regulator.

III. MAIN MICROCONTROLLER

This section will describe how the System Controller operates. The System Controller is composed of the Main Controller and Communication modules for the sensor controller, servo controller, and the bluetooth. As mentioned before, it operates in two modes: autonomous or teaching mode.

A. Autonomous Mode

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. 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 main 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.

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 moves 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. All of these state changes are shown in Figure 5.

During the autonomous mode the expected Bluetooth packages are of type A, F, G, and the type H package is sent from the main controller to the mobile application as a response to the package type F. When a package type F is received the main controller, gathers the voice command strings associated with each gesture currently stored in the arm, and sends it to the mobile app. This action happens every time the IPPA is connected to the mobile application.

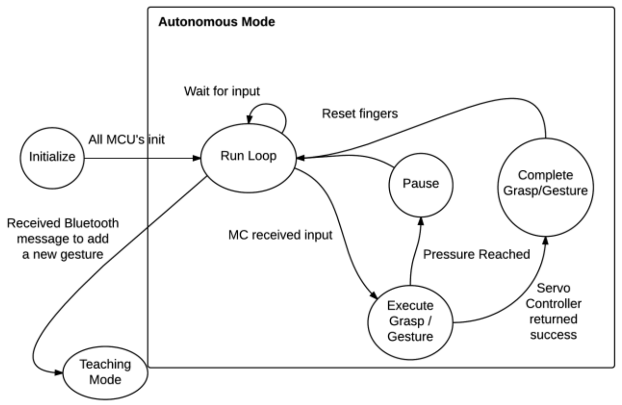


Fig. 5. Flowchart showing the Autonomous Mode in the main controller.

B. Teaching Mode

In order for the IPPA to switch to Teaching Mode the external application must send a message to switch modes, which is received in the Run Loop that tells the main controller to go into teaching mode. In this case the main loop in the Autonomous Mode is discontinued and the non-ending Input Loop is started.

The first phase in Teaching Mode is the Input Loop. The input loop is an interrupt-waiting loop that is waiting to for input. In the Input Loop, the main controller waits for the user input through the application. It could be to add/delete a gesture already defined in the android application; temporarily store and trigger a gesture once; or the user is designing a new gesture. In this later case, the IPPA allows the user to have real time feedback regarding the position of the fingers. The mobile application communicates the user intent during this execution mode with the Bluetooth packages types B, C, D, E, and G.

When a Bluetooth message of type C is received, the main controller decodes the transmitted information for the gesture to be stored in the arm and transitions to the Store New Gesture phase. This temporary structure replaces one of the 5 gestures that are permanently stored in the main controller's memory. For the message type B, the main controller transitions into the Finger Movement state, where it transmits the new desired servo positions to the servo controller. The decoding of the message is very fast since it is short, and the communication to the servo controller and execution of such servo positions is done quickly. In the case of a type E message, the gesture is only stored temporarily, and executed immediately after.

After any of the above Bluetooth messages have been handled, the execution of the program returns to wait in the

Input Loop. If a message of type G is received, then the main controller moves back to the Run Loop, and starts to run the Autonomous Mode. All of these states are shown in Figure 6.

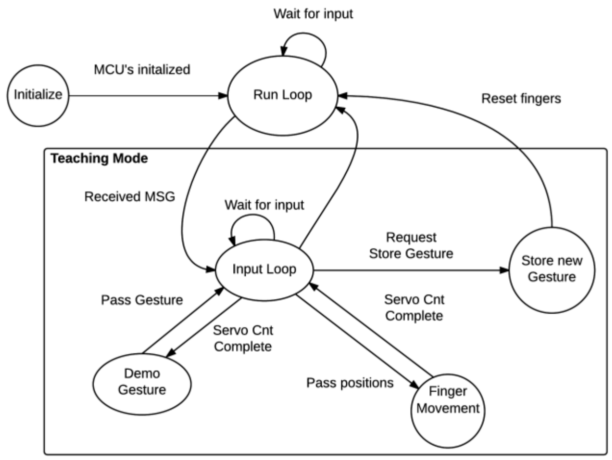


Fig. 6. Flowchart showing the Teaching Mode in the main controller.

IV. System Graphical Interface

The mobile application was designed to provide the user with an easy but capable interface to the IPPA. 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

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.

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. Figure 7 shows the main page for the mobile application. There are a total of four activities for this application: Main activity, Help activity, TeachingMode activity and DeviceDiscovery activity. The application has been divided in two packages: the Bluetooth functionality and the application itself. The objective of this is to provide a robust enough structure for future work.

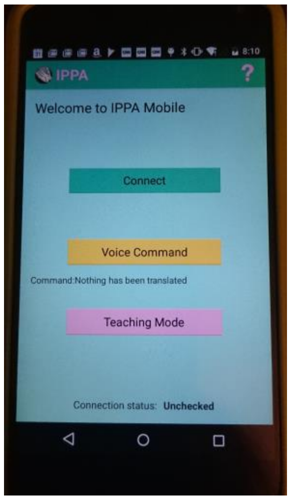

IPPA
?
Welcome to IPPA Mobile
Connect
Voice Command
Command:Nothing has been translated
Teaching Mode
Connection status: Unchecked

Fig. 7. Mobile Application Main page.

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 is done and the buttons are enabled or disabled. If the check passed then application will proceed to the selected page.

This application needs to store multiple files in the phone. There is a file to store all the gestures in the phone and arm. 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 will guarantee only one Bluetooth service instance, that cannot be directly modified.

A. Voice Commands

The speech recognition will be 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 will be started (Google API). The onActivityResult() method is used to handle the result obtained from the launched activity. Multiple translated texts is obtained and compared to the available gestures. Once the audio input has been translated to text, a package of type A 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.

B. Teaching Mode

The Teaching Mode will be represented in two major views. These views are the "Create Gesture" and "Demo Gesture" fragments. These two 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. 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.

V. 3D Hand Design

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 [3].

The InMoov hand design provides three joints for each finger [2], 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.

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 will be needed. These were not be added to each finger but to three: the tip of the thumb, the tip of the middle dinger, and the center of the palm. The location for the distance sensor is toward the top left of the palm. In order to connect the sensors to the sensor controller, wires were ran through the needed fingers and palm. 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.

The InMoov forearm has been designed to be hollow, with sufficient space for the 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 Module. 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 will avoid the user pressing the buttons by mistake, such as placing the arm on a hard surface.

VI. Hardware Details

For the IPPA to be a practical device, a printed circuit board was design to house the majority of the electronic components used to control the arm. To save space, surface mount components are utilized, which are a fraction of the size of DIP components. The designed PCB is shown in Figure 8.

The printed circuit board IDE 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. With EAGLE, the team produced not only the printed circuit board layout, but also the supporting schematics that were used in prototyping and debugging. EAGLE is free for students, and allowed the team to create a two-layer design, which provided more than enough

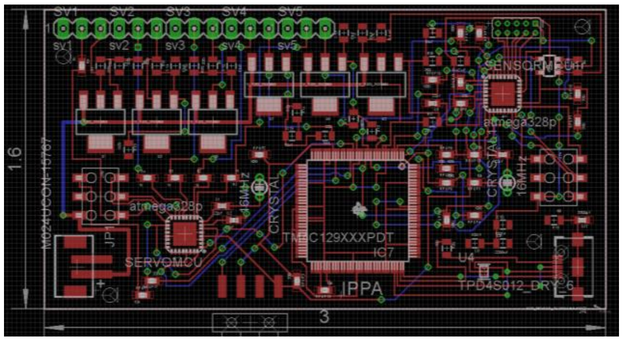


Fig. 8. Board layout of the IPPA PCB.

options for the relatively basic design. Creating designs greater than two layers would have been out of the budget

The most important feature of the board in order to fit in the forearm and within the budget was its size. To save money, the team minimized the size of the board.

VII. Conclusion

We can conclude that the IPPA is a step towards the development and advancement of affordable prosthetics that utilize technology. The IPPA provides a low-cost solution for adult amputees, with advanced features. This project is open source, and has been made available to developers and/or users through GitHub at https://github.com/icarreras/IPPA_application.git. The team hopes that this project is taken by other engineers and continue to expand on the possible features.

Ideas for future work: addition of a sync button for multiple mobile device support; design arm to fit children; or incorporation of consecutive gesture execution to support complex hand motion.

Acknowledgement

The authors wish to acknowledge the assistance and support of the members of the Texas Instruments Innovation Lab at the University of Central Florida.

Authors Biography

Ivette Carreras is a Computer Engineering student at the University of Central Florida. She will be graduating with a Bachelor of Science Degree in May 2015. During her time at UCF, she interned at both Texas Instruments and Microsoft as a Software Engineer. She will be joining the Operating System Group at Microsoft as a full time Software Engineer after graduation. Her interests are in low level software development.

Andrew Mendez is an undergraduate student at the University of Central Florida. He will be receiving his Bachelor's degree in Computer Engineering in May of 2015. During his time at UCF, he has participated in the MIT Summer Research Program, where he researched the areas of tangible user interfaces and augmented reality at the MIT Media Lab. His interests are designing intelligent interfaces for intuitive learning and interaction with information from our physical environment.

Matthew Bald is an undergraduate student at the University of Central Florida. He will be receiving his Bachelor's degree in Computer Engineering in May of 2015. During his time at UCF, he interned at OUC and created SalesForce applications to assist the customer resolution team. His interests are in designing electronics and music.

References

[1] 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
[2] 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/
[3] InMoov. Hand and Forarm. http://www.inmoov.fr/hand-and-forarm/
[4] NASA. Human Performance Capabilities. http://msis.jsc.nasa.gov/sections/section04.htm
[5] 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/
[6] Pololu. 1501MG Datasheet. http://www.pololu.com/file/0J729/HD-1501MG.pdf
[7] Sparkfun. Force Sensitive Resistor Tutorial. https://www.sparkfun.com/tutorials/269
[8] Texas Instruments. CC3200. http://www.ti.com/product/cc3200 ,
[9] Texas Instruments. MSP430G2x53 Datasheet. http://www.ti.com/lit/ds/symlink/msp430g2553.pdf