Robotic Hand in Motion Using Arduino-Controlled Servos

Nicholas Bonini
nicholas.bonini@gmail.com

Nithya Iyer
nithya_a_iyer@yahoo.com

David Kim
ycdavidkim@gmail.com

Katherine Mathison
kvmathison@gmail.com

Lauren Wellons
lewellons@comcast.net

New Jersey Governor's School of Engineering and Technology 2014

Abstract

Amputees often suffer from psychological and physical difficulties due to their inability to use their extremities. To aid the amputees in acquiring a functional replacement hand at a feasible cost, a prototype prosthetic was created utilizing Flexy Hand, a 3D printable hand model, and Arduino, an open source microprocessor. To avoid expensive and frustrating control methods associated with myoelectric prosthetics, an Android smartphone application allows the user to select a gesture that he wishes the hand to perform. The phone sends this information to the Arduino, which powers certain servos to actuate each finger individually. The simple construction and low cost of materials, as well as the use of common devices such as smartphones, enables amputees to gain access to new prosthetics with ease.

1. Introduction

With 3D printed prosthetics gaining popularity with the advent of consumer level 3D printers, practical applications of these prosthetics have also been increasing. Online open source development has allowed for printable hand models to be downloaded for free within minutes from websites such as Thingiverse, a free, open source 3D modeling site.

Contrary to traditional prosthetics, which often cost tens of thousands of dollars[1] and are usually unaffordable to many, these alternatives provide a relatively inexpensive option to the public. This allows for less expensive, yet effective prosthetics to be available to modern consumers, with increased personalization for the user at a speed unachievable by conventional methods.

Besides cost, another prevalent issue in high-level prosthetics is ease of control. The most common control system in use is electromyography, a medical technique in which electrical signals from the remaining muscles in an amputee's forearm are read by a device[2] attached to the muscle and are mimicked by the prosthetic. While people without any functional forearm muscles, and even those with intact forearm muscles can use this approach, electromyography can be difficult to use and is often imprecise[3]. Therefore, developing a 3D printable hand that is easy to assemble and control in a non-professional setting with commonplace products such as smartphones is vital for the average amputee. Smartphone based controls are more portable and convenient

than traditional myoelectric or motion sensing control systems.

Furthermore, the functionality and performance of consumer level robotic hands have advanced significantly in the past several years. Developments in the 3D printing industry have led to numerous designs of modeled hands from users all around the world, which the public can access online. 3D printed prosthetics are further advantageous in that they are ideal for children and teenagers; when one hand becomes too small, another can be readily printed in a larger size and reintegrated into the control system[4].

2. Background

2.1 3D Printing

Makerbot Replicator utilizes a technology called Fused Deposition Modeling (FDM), which involves the construction of parts in layers using high-grade thermoplastics[5]. To print a 3D model, an STL file of the design must be exported into the printer software. The Makerbot then creates g-codes that determine a pathway for the extruder. The extruder is the mobile head of the printer that first melts the filament and then deposits the molten thermoplastic into thin layers until the model is fully printed. As needed, the 3D printer will provide scaffolds for the design that act as supports. These are easily removed at the end[6].

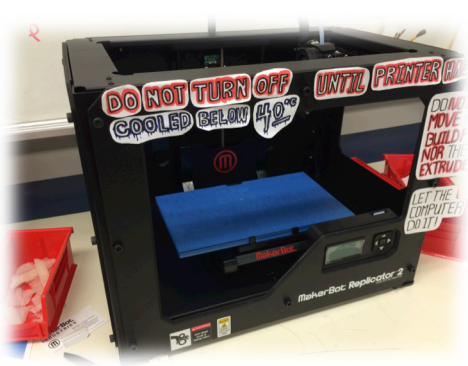


Figure 1: Makerbot Replicator 3D Printer

The most commonly used filaments in consumer printing are acrylonitrile butadiene styrene (ABS) and polylactic acid (PLA) thermoplastics. Each filament has its own unique characteristics, lending to different usages. While ABS is tougher and more flexible than PLA, due to the nature of the plastic, ABS requires a heated bed to prevent the outer layers from curling in or warping; this guarantees an even distribution of heat to both the outer and inner layers. PLA, however, does not require a heated bed and is more resistant to substances such as acetone, which dissolves ABS filament.

2.2 Arduino

Arduino is a brand of open-source microcontrollers frequently used in at-home, do-it-yourself electronics projects[7]. It can be programmed in a version of C, and the Arduino website contains software for programming the device. There are also a host of straightforward online tutorials that make it easy and quick to learn. A variety of electronic components can be connected via breadboard as inputs and outputs for the code, making Arduinos incredibly versatile. Arduino microcontrollers are intuitive, inexpensive[8], and readily available—three factors critical to accessible, easy-to-use prosthetics.

2.3 Servos

Servos are small but powerful motors that can be used in a multitude of products ranging from toy helicopters to robots. A servo consists of three basic parts: an electric motor, a feedback potentiometer that connects to the output shaft, and a controller. This allows the servo to rotate to specific angles by keeping track of its current angular position. The servo is controlled via Pulse-Width Modulation, or PWM. The motor aligns the shaft to a specific angle depending on the duty cycle of the signal that is sent. The ability to

rotate to a certain position rather than at a certain speed makes servos very useful in prosthetic devices by making very precise movements through the elimination of the time variable. Normal DC motors require running the motor for a given amount of time at a certain speed to derive a distance; servos can directly choose a position[9].

2.4 Bluetooth

Using low-power radio waves, Bluetooth can connect up to eight devices simultaneously[10]. Due to its strong connectivity, this technology is preferred over other wireless communication techniques such as infrared networks. To achieve this, Bluetooth devices send out very weak signals, preventing interference with other systems. Although this limits the range to about 10 meters, it provides enough range for a keyboard, computer, mouse, or desktop printer. Furthermore, weak transmissions reduce power consumption and do not require a direct line of sight between the connected devices, allowing the user to be in a separate room from the second device while still maintaining full functionality[11].

2.5 Android Application

MIT App Inventor 2 is the most ideal platform for developing an app to control the hand. This platform is free to use and simple to learn, even for those with very little experience in programming. MIT App Inventor 2 is in the style of a blocks editor: rather than writing lines of code in the traditional manner, the designer drags and drops blocks to represent functions and variables. Every segment of code begins with a condition given by a "when" block, and continues with "get" blocks, or "set" blocks—what the application will do when the "when" condition is met. Each "when", "get", and "set" block is specific to each component; for example, a button has "when Button1.Click", a "when" condition that is true when the button is clicked[12]. The full block code can be viewed in Appendix B.

2.6 Myoelectric Prosthetics

Myoelectric prosthetics generally weigh anywhere from 400 to 600 grams, can use a combination of DC motors and servos, and have around 11 joints and 6 degrees of freedom. On the contrary, the human hand has an average weight of 400 g, taking up 0.6 percent of the total body weight, and has 27 degrees of freedom in terms of finger functionality[13]. In 2007, results of a survey taken by myoelectric prosthetic users showed that many consumers wanted better movement of the thumb, index finger, and wrist. Although they use one of the most advanced control methods on the market, new myoelectric devices are not responding to users' complications with finger functionality by only having 6 degrees of freedom[14].

3. Methods/Experimental Design

3.1 Hand Assembly

The hand used in the prototype was a 3D printed version of the Flexy Hand[15]. The STL file was exported into the Makerbot platform and directly printed without any scaling or modification. Taking into account the separate parts of the hand, printing took about eleven hours. The completed hand was strung with fishing line and a stretched disposable pipette. The pipette was used to break up any excess material from the 3D printing that would hinder the fishing line's path through the interior of the hand and fingers, and to help thread the line through the palm of the hand. Each finger was strung with about two feet of fishing line to ensure that there would be enough material to reach down the length of the arm and attach to the servos.

Molds for the hinge joints (see Figure 2) were downloaded from Github[16] and 3D printed.

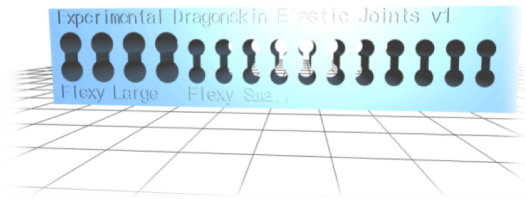


Figure 2: 3D Model of Joint Mold

Various types of silicone were molded into joints and placed in the joint slots in each finger to test their properties. GE™ Supreme Silicone was selected to be the main material for joint casting. Using a caulk gun and the 3D printed molds, the silicone was shaped into connectors that cured in about 24 hours. These were then fit into slots between each finger segment.

3.2 Arm Assembly

A PVC pipe with an inner diameter of one and a half inches and a length of twelve inches was used to house the servo motors. Rectangular holes three fourths of an inch wide by one and one half inches long were cut into the PVC in a staggered pattern down the tube, as shown in Figure 3. The holes were spaced a quarter of an inch apart. Five ⅛ inch round holes were drilled into the pipe at one end, one for each finger's control string to enter the pipe and meet the hand at its center.

A plastic ring was 3D printed and attached to the end of the pipe with superglue in between the 1/8 inch holes and the servos. The servo motors were placed in their housing and superglued into place. The fishing lines were run from the fingers through the interior of the PVC tube, out the holes drilled into the tube, through the 3D printed ring, and to the servos. They were tied to the prongs on the servos that were pointed directly toward the hand when in the zero position. Eyeglass screws were inserted into the two prongs of each servo to the right of the prong with the fishing line.

Figure 3: PVC pipe with a staggered spiral motion

Holes were drilled into the hand and the sides of the PVC tube. L-brackets were fastened between the hand and the inner pipe by driving screws into these holes and reinforcing the connection with superglue. This holds the hand to the inner tube.

3.3 Circuitry

Each servo has three wires: a power wire, a ground wire, and a Pulse-Width Modulation (PWM) wire. The PWM wire enters one of the six PWM ports on the Arduino Uno board. The power and ground of each servo were attached to the horizontal positive and negative rows on the breadboard, which were connected to the 6V battery pack. The battery pack housed four 1.5V D size batteries. The Bluetooth module's four pins were plugged into adjacent columns on the breadboard. The power and ground pins connected to their

own power and ground ports in the Arduino. The other two pins are RXD and TXD, and connected to the TX and RX ports on the Arduino, respectively. These pins told the Arduino what commands the phone was sending (see Figure 4).

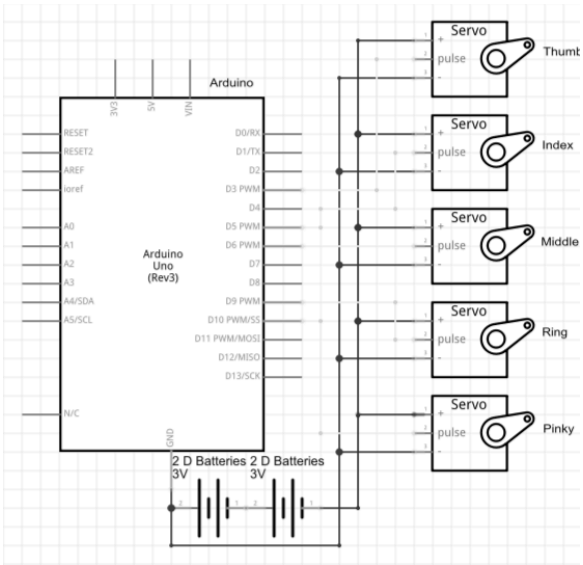


Figure 4: Arduino Servo Circuit Schematic

3.4 Arduino Programming

The Arduino was programmed using the development environment available on the Arduino website. The most important aspect of this project's code was setting the position of the servos , which determined the fingers' movement. The servo library, which contained functions for attaching the servos to pins and setting their angles, established fine control over the servo positioning. The attach() function associated a certain pin on the Arduino with each servo, and the write() function set the angular position of the servo to a value between 0 and 180 degrees. The position of the servo determined how far the finger that it controls would bend, with a lower angle leaving the finger more relaxed and a higher angle bending the finger in toward the palm of the hand. Gestures were made by setting each servo to a specific angle that corresponded to how far the finger was bent when the hand made this gesture. Initially, a potentiometer was used to manually control the position of the fingers, and when each gesture was made, the angles of the servos were recorded and later configured as preset gestures. In the final design, gestures were controlled via an Android application connected by Bluetooth to the Arduino board. When a gesture to be performed was selected in the application, the Arduino received an integer between zero and five and used this data to select which gesture function to run. The functions set the servos' positions to match the data gathered for the selected gesture in the manual tests. The full programmed code can be viewed in Appendix A.

3.5 Application Development

The application for this hand was a button-based application with only one screen (see Figure 5). When a button was pressed for a gesture, one byte number was sent to the Bluetooth module. The Arduino used this byte as a trigger to run the block of code that executes that gesture, setting the servo motors for the corresponding fingers to specific angles. When any button that results in a gesture was selected, the other gesture buttons became hidden until the reset button was pressed. Once the hand reverted to its initial position, the other buttons became available again.

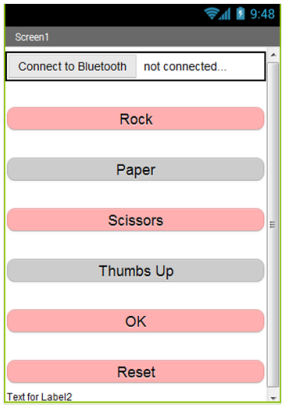


Figure 5: Menu interface on the Samsung Nexus S

To connect to Bluetooth, a list picker function was created as a "connect to Bluetooth" button. When this button is pressed, a list of devices that have been paired with the phone materializes. When the HC-06 Bluetooth module is selected, the Android phone establishes a connection and a green label appears that reads, "Connected!" This means that the device is ready to receive orders from the phone. If the Android phone cannot connect to the Bluetooth device, the label appears red and reads, "Not connected…" The application automatically checks the phone for connectivity every 50 milliseconds and updates the label to the current status of the Bluetooth connection.

The Samsung Nexus S was selected as the smartphone to be used with the prototype. To test the application, Android drivers were installed to download the apk file to the phone. Once the drivers were installed, the Android had to be opened as a USB storage device. The application was placed into the phone's folder, which appeared when USB storage mode was activated. The file appeared in the application folder "ES File Explorer" on the phone's display, where it could be selected for installation. Once the application was installed and the phone connected to the Bluetooth module, the flashing red light on the module would stop blinking and stay lit. Once the Bluetooth was connected, each servo was hooked up to the breadboard one at a time and tested with the app.

The second component of testing involved the application's performance to button response. First we tested how the menus would navigate; each gesture button was pressed in turn and the result was assessed. Asking random people to compare different color schemes and button layouts simulated user perception of the application.

4. Results and Discussion

4.1 Hand Functionality

The fully assembled hand can easily perform the everyday gestures of rock, paper, scissors, the "OK" sign, and thumbs up at the push of a button. The user interface is simple to use and easy to learn. The application allows the user to control five different gestures via Bluetooth from an android smartphone. The fingers can be configured to a variety of gestures due to the strings that are manipulated by the servos. Also, rubber bands superglued to the back of the fingers allow them to snap back into place when the servos release the pressure (see Figure 6).

Figure 6: Hand with rubber bands attached

This entire project can be easily built with commonly available materials at a low cost of $475, and can be programmed using the open source Arduino with minimal experience in programming. A smartphone application for controlling the hand can be designed by following the tutorials on the MIT App Inventor website. Although the device is simple and affordable, it cannot grab things very easily or respond to stimuli, and it is heavy and bulky. Therefore our device is easily reproducible but not robust.

4.2 Discussion

New technological advancements in the field of myoelectric prosthetics have led to development of hands with multiple degrees of freedom. Nonetheless, these devices are still inadequate in terms of performance, stability, aesthetics, and affordability. Anthropomorphic hands such as iLimb™ and Bebionic™ have received media attention—yet, they are not as consumer friendly as advertised.

The use of servos, rather than DC motors, in this new prototype proves to be a more viable option when controlling finger kinematics. Servos are lighter than DC motors and are more precise. Due to their simple, continuous rotation mechanisms, DC motors are more commonly used in prosthetics, even though servos demonstrate a higher degree of accuracy through proper control of angular displacement. Servos are also much easier to program than DC motors; a DC motor's control is dependent on the time the action will take to complete, rather than the motor's final position. With the use of servos, the time variable is eliminated along with any other factors for possible error.

Moreover, most myoelectric prosthetics possess six degrees of freedom in finger movement as opposed to only five in this particular prototype. To negate this disadvantage, different hand designs can be 3D printed to achieve this sixth degree, the Trapeziometacarpal joint. Because of time constraints, this joint was not added to the prototype due to the added complexity in the programming and assembly of the hand. As a result, the fingers cannot move in a lateral direction, lessening the number of degrees of freedom the prototype has overall. This joint does play an imperative role in the functioning of a normal thumb, so alternative designs can be found to ensure the thumb will be fully utilized. The dexterity of the average human hand is far more intricate with 27 degrees of freedom. This new prototype and even the myoelectric devices are not advanced enough to reach that level, but with new developments in the future, prosthetics could eventually replace the lost limb or hand entirely.

4.3 Problems Encountered

Although this project met most of its initial goals, there were several setbacks in the process, which hindered overall productivity. Small errors during hand printing were compounded by stress from us, causing cracks to appear in the palm. Acetone was used to soften excess ABS filament so that the filament could be placed inside the cracks for restoration.

In the initial tests, the fishing line would pass under the servo arms and through the diameter of the rotation. To achieve maximum distance with every rotation, the eyeglass screws were put into the holes in the servo blades. This kept the lines within the circumference of the arms, which is longer than the diameter and could pull the fingers further. Then, the fishing line would often wrap underneath the screws in the servo instead of around them so that the line was not pulled the full length, preventing the hand from fully closing. To resolve this, a plastic ring was added to the internal pipe in the arm. The fishing line was threaded through holes in the ring at the same height as the screws.

Another issue involved the inability of the hand to return to its initial resting position. To correct for this flaw, rubber bands were attached to the back of the fingers. The first set of rubber bands was too inelastic and prevented the servos from pulling the fingers in toward the palm. However, when the rubber bands were replaced with longer, thinner elastic bands, the servos had enough power to move the fingers once more. This problem would not

have arisen had the hinge joints been printed with strong, flexible nylon. The process for molding and curing silicone joints could have been completely avoided.

Arm assembly also provided challenges: the original outer casing of the arm was just big enough to fit the servos and allow them to function properly (see Figure 7). Carving holes for the servos in the outer pipe alleviated any casing issues.

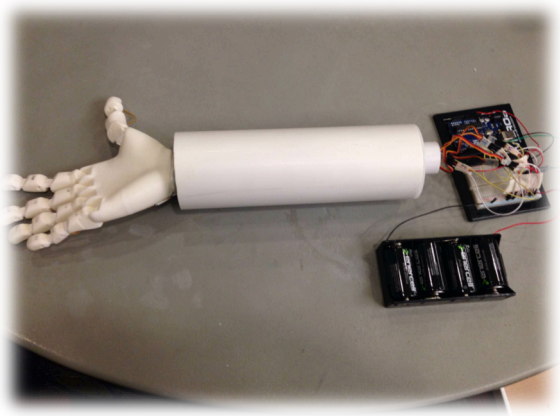


Figure 7: Full Hand and Arm Assembly

One problem encountered during app development was that the Bluetooth device would become disconnected, but the connectivity light would stay green. To remedy this, a clock timer was added to update the connectivity light's status every 50 milliseconds. The reason the app makes the user reset the hand before beginning another gesture is to prevent confusion. If more than one gesture button is pressed, the servos may receive more than one piece of information, ultimately causing them to fail.

Unfortunately, a system for turning the wrist using a DC motor and sprockets was in the initial stages of design, but could not be implemented as a part of the arm due to time constraints.

5. Conclusion

Simple prosthetics have the potential to make a measurable impact in an amputee's daily life. Since this particular prosthetic is controlled by an Android application, its use is straightforward and does not carry a steep learning curve, unlike many of the more advanced prosthetics which require an inordinate amount of time to master. Construction and assembly of the hand calls for a short list of materials and tools that are easy to access. With the availability of this technology, amputees have the necessary tools to manufacture their own personalized prosthetics that will improve their quality of life in addition to the prevention of the mental degradation that oftentimes comes with physical deformation[17].

The objective of this project consisted of the creation of an inexpensive 3D-printed robotic prosthetic hand powered by Arduino that could perform several gestures. The proof-of-concept prototype accomplishes these tasks effectively with an easy method of control. Several enhancements to the design would boost functionality to the hand. Wrist actuation of the hand would prevent the amputee from having to turn his shoulder to properly grasp something; instead, the app would turn the hand to the desired setting. Further upgrades would include more gestures and tactile feedback. A bank of specialized gestures could be programmed into the app, offering the user more flexibility with his fingers. Tactile feedback, on the other hand, would alert the amputee to the temperature and texture of objects, establishing a natural perception of touch. These ameliorations would serve to create a product that could revolutionize the prosthetics industry.

6. Acknowledgments

The authors would like to acknowledge our project mentors, Mohit Chaudhary, Dr. Kang Li, and Dr. William Craelius for all of their help and guidance in planning and building our project, and our assistant project mentor, Julian Hsu, for his assistance in designing the Android app. We would also like to thank our RTA, Mary Pat Reiter, for her assistance with this paper, Director Dean Jean Patrick Antoine, and all our GSET sponsors for their support: Rutgers University; The State of New Jersey; Morgan Stanley; Lockheed Martin; Silverline Windows; South Jersey Industries, Inc.; The Provident Bank Foundation; and Novo Nordisk. We would also like to thank Steve Wood for his 3D model of the Flexy Hand. Without the generous help and donation of these people and companies, our project would not have been possible. Our team expresses much gratitude to everyone who contributed to this project.

7. Notes:

[1]W. Craelius, K. Li, I. Ali, A. Alvi, E. Chu, K. Lin, S. Nazare, N. Plichta, "Transhumeral Prosthesis with a Dexterous Hand," pp. 1-3, (unpublished).

[2]"Myoelectric Prosthetics 101," *Ottobock*, <http://www.ottobockus.com/prosthetics/info-for-new-amputees/prosthetics-101/myoelectric-prosthetics-101/> (19 July 2014).

[3]Joseph T. Belter, Jacob L. Segil, Aaron M. Dollar, Richard F. Weir, "Mechanical design and performance specifications of anthropomorphic prosthetic hands: A review", JRRD, **50**, p. 611, 2013.

[4]"John Hopkins and e-NABLE," *Enabling the Future,* 17 July 2014, <http://enablingthefuture.org/tag/3d-printed-hands> (19 July 2014).

[5]"FDM Thermoplastics," *Stratasys*, <http://www.stratasys.com/materials/fdm> (19 July 2014).

[6]Annelise, MakerBotting 101 - How Does It Work, *MakerBot*, January 10, 2012, <http://www.makerbot.com/blog/2012/01/10/makerbotting-101-how-does-it-work/> (23 July 2014)

[7]"Frequently Asked Questions," *Arduino*, <http://arduino.cc/en/Main/FAQ> (19 July 2014).

[8]"Arduino Uno-R3," *Sparkfun*, <https://www.sparkfun.com/products/11021> (19 July 2014).

[9]"What's A Servo?," *Seattle Robotics Society*, <http://www.seattlerobotics.org/guide/servos.html> (19 July 2014).

[10]How It Works," *Bluetooth*, <http://www.bluetooth.com/Pages/How-It-Works.aspx> (19 July 2014).

[11]"How Bluetooth cuts the cord," *TechTarget*, March 2005, <http://searchmobilecomputing.techtarget.com/feature/How-Bluetooth-cuts-the-cord> (19 July 2014).

[12]"MIT App Inventor," <http://appinventor.mit.edu/explore/> (19 July 2014).

[13]Pylatiuk C., Schulz S., Döderlein L., "Results of an Internet survey of myoelectric prosthetic hand users", *PubMed*, Dec. 2007, <http://www.ncbi.nlm.nih.gov/pubmed/18050007> (19 July 2014).

[14]"Flexy Hand," *Thingiverse*, <http://www.thingiverse.com/thing:242639> (19 July 2014).

[15]"Flexy-Joint," *GitHub*, <https://github.com/daprice/Flexy-Joint/wiki/Casting> (19 July 2014).

[16]"Coping with Your Amputation," *Capital Health*, <http://www.cdha.nshealth.ca/amputee-rehabilitation-musculoskeletal-program/patients-families-amputee-rehabilitation/coping-your-> (19 July 2014)

[17]Joseph T. Belter, Jacob L. Segil, Aaron M. Dollar, Richard F. Weir, "Mechanical design and performance specifications of anthropomorphic prosthetic hands: A review", JRRD, **50**, pp. 599-606, 611, 2013.

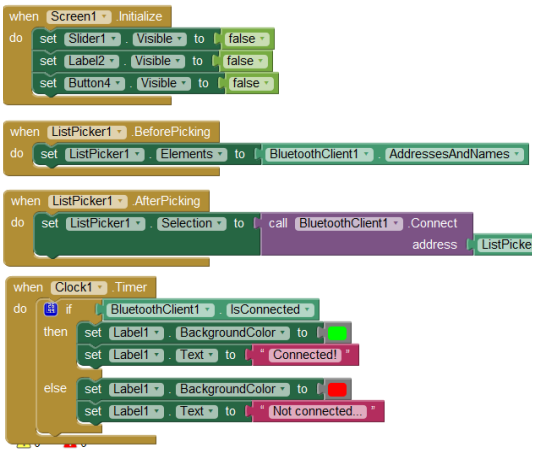
8. Appendix

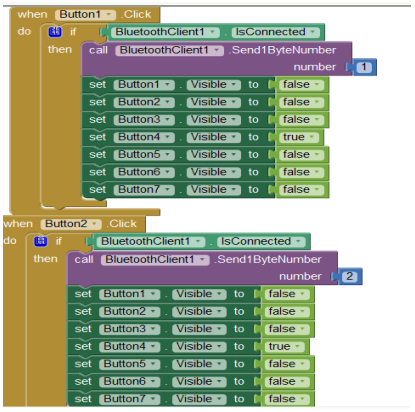
8.1 Appendix A: Arduino Code

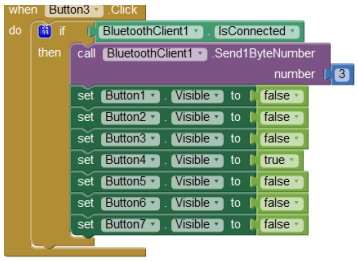
```
#include <Servo.h>          //Import servo
library

Servo thumb;                //Initialize
finger servos
Servo index;
Servo middle;
Servo ring;
Servo pinky;

byte serialA;               //Input from
app

void reset() {              //Resets
fingers to a natural
 thumb.write(30);           //relaxed
position
 index.write(30);
 middle.write(30);
 ring.write(30);
 pinky.write(30);
}

void setup() {
 thumb.attach(3);           //Sets each
servo to a pin
 index.attach(5);
 middle.attach(6);
 ring.attach(9);
 pinky.attach(10);

 reset();
 Serial.begin(9600);        //Can receive
input via BT
}

/*void jelly() {
 //Jellyfish hand
 for(int i=0;i<180;i++) {
   thumb.write(i);
   index.write(i);
   middle.write(i);
   ring.write(i);
   pinky.write(i);
   delay(8);
 }
 for(int i=180;i>0;i--) {
   thumb.write(i);
   index.write(i);
   middle.write(i);
   ring.write(i);
   pinky.write(i);
   delay(8);
 }
}*/

void rock(){                //RPS rock
   thumb.write(180);
   index.write(180);
   middle.write(180);
   ring.write(180);
   pinky.write(180);
}

void paper() {              //RPS paper
 thumb.write(0);
 index.write(0);
 middle.write(0);
 ring.write(0);
 pinky.write(0);
}

void scissors(){            //RPS scissors
 thumb.write(180);
 index.write(0);
 middle.write(0);
 ring.write(180);
 pinky.write(180);
}

void thumbsUp() {           //Thumbs up
gesture
 thumb.write(0);
 index.write(180);
 middle.write(180);
 ring.write(180);
 pinky.write(180);
}
```

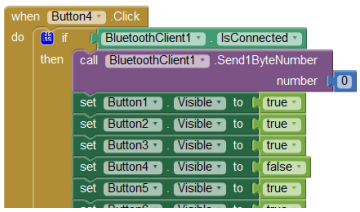
```
void okay() {                    //Okay sign
 thumb.write(60);
 index.write(150);
 middle.write(0);
 ring.write(0);
 pinky.write(0);
}

void loop() {
 if (Serial.available() > 0) {     //Read value
from app via BT
   serialA = Serial.read();
   Serial.println(serialA);
 }
 switch(serialA)        {                    //Input
determines gesture
   case 0:
     reset();
     break;
   case 1:
     rock();
     break;
   case 2:
     paper();
     break;
   case 3:
     scissors();
     break;
   case 4:
     thumbsUp();
     break;
   case 5:
     okay();
     break;
 }
}
```

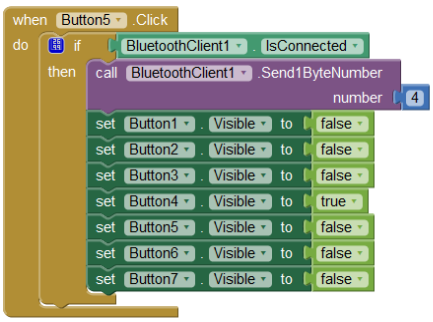
8.2 Appendix B: MIT App Inventor Code

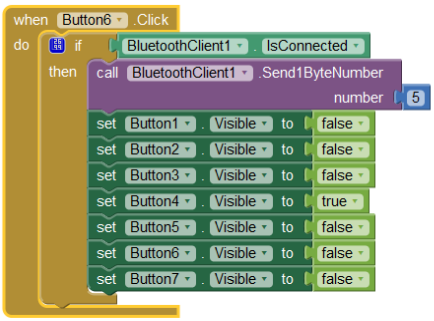





when Button3 .Click
do if BluetoothClient1 . IsConnected
then call BluetoothClient1 .Send1ByteNumber
number 3
set Button1 . Visible to false
set Button2 . Visible to false
set Button3 . Visible to false
set Button4 . Visible to true
set Button5 . Visible to false
set Button6 . Visible to false
set Button7 . Visible to false


when Button4 .Click
do if BluetoothClient1 . IsConnected
then call BluetoothClient1 .Send1ByteNumber
number 0
set Button1 . Visible to true
set Button2 . Visible to true
set Button3 . Visible to true
set Button4 . Visible to false
set Button5 . Visible to true


when Button5 .Click
do if BluetoothClient1 . IsConnected
then call BluetoothClient1 .Send1ByteNumber
number 4
set Button1 . Visible to false
set Button2 . Visible to false
set Button3 . Visible to false
set Button4 . Visible to true
set Button5 . Visible to false
set Button6 . Visible to false
set Button7 . Visible to false


when Button6 .Click
do if BluetoothClient1 . IsConnected
then call BluetoothClient1 .Send1ByteNumber
number 5
set Button1 . Visible to false
set Button2 . Visible to false
set Button3 . Visible to false
set Button4 . Visible to true
set Button5 . Visible to false
set Button6 . Visible to false
set Button7 . Visible to false