Final Report

Fyp20063 2020-2021

Robotic Hands Controlling by Smartphone App

By

Bert, Liu Kin Ning
3035376147
libert@connect.hku.hk

Supervisors:

Dr. T W Chim,
The idea of trying something new that I have not tried before drove me into choosing this topic as my final year project. And because of this, I stumbled upon obstacles and struggles. Some of them are duly surmounted, but some of them cannot be overcome and compromise the quality of the final results. Nevertheless, the overall performance of the project has reached the level of quality that I am satisfied with. And this cannot be done without the help and guidance of many.

I would like to thank Dr. Chim for opening the final year project topic so that I could engage myself in these three months in a fun and challenging environment in which I learnt a lot about 3d printing, 3d modeling, Arudino, React Native, web programming and such. Also Dr. Chim had provided me with samples of past projects of the Makerlab. These served as excellent examples for me to reference.

I would like to thank Dr. Chan’s instructive and insightful comments during the interim presentation session, with which I was able to improve on my current design of the software elements of the project. Specifically, his instructions over the internet control has been directly incorporated into the system.

I would also like to thank Mr. David Lee’s help and instructions over the past 8 months. He provided me and others in the Makerlab valuable instructions and guidelines. His code sample of the ESP32 programming is vital to the control mechanism of the servo motors in the project. In addition, his recommendations on the purchase and usage of electronic parts and the teaching of the use of 3d printers are indispensable. It cannot be stressed enough for his contributions to the completion of this project and I am forever grateful for it.
Finally, I would like to thank the schoolmates that I met in the Makerlabs, without their encouragement and friendship, it would have been much more difficult to finish this project on time given such a tight schedule.
Table of contents

I. Cover Page

II. Acknowledgements

III. Table of contents

1. Introduction
   1.1 Background
   1.2 Motivation
   1.3 Project Objectives and Requirements
   1.4 Contribution of the Project
   1.5 Structure of the Report

2. Project Background and Literature Review
   2.1 Robotic hand model
       2.1.1 Creating the 3d model from scratch
       2.1.2 Ada Hand by OpenBionics
       2.1.3 Inmoov robotic hand
   2.2 Mobile application
       2.2.1 PWA and webview
       2.2.2 Flutter
       2.2.3 React Native
   2.3 Controlling method
       2.3.1 Normal slider control
       2.3.2 Extra glove control
       2.3.3 Preset-configuration
       2.3.4 Gesture control and the Mediapipe project

3. Project Methodology
   3.1 Overview
   3.2 3d Printer
   3.3 Facility
   3.4 Robotic Hand
       3.4.1 Battery
       3.4.2 Voltage Regulator
3.4.3 Servo Motors p.21
3.4.4 PCA 9685 p.22
3.4.5 ESP32 p.22
3.4.6 Self-created 3d model p.23
3.4.7 Inmoov 3d model and the construction of the robotic hand. p.25
3.4.8 Receptive sensors attached to the fingertips p.28
3.4.9 Final look of the robotic hand p.28

3.5 Program of ESP32 p.29
3.5.1 Bluetooth part p.29
3.5.2 Logic of the receiver and the format of the message p.29
   3.5.2.1 Pre-determined messages p.30
   3.5.2.2 Custom gestures p.31
   3.5.2.3 Potential third mode p.31

3.6 Mobile application p.32
3.6.1 Connection p.32
3.6.2 Built-in gestures p.35
3.6.3 History page p.37
   3.6.3.1 How is the history page supported? p.37
3.6.4 Recording page p.38
3.6.5 QR Code binding p.40

3.7 Website p.41
3.8 Middle-man server p.44
3.8.1 AWS and Https server p.45

4. Results p.47
   4.1 Limitations and Future works p.48

5. Conclusion p.50

References p.52
1. Introduction

1.1 Background
Recent developments in computer vision and artificial intelligence have made high-accuracy gestures recognition possible. In addition, since the cost of 3d printing has dramatically decreased in recent years, there is an abundance of online communities who dedicate their time creating STL files of 3d models of robotic components of various human parts, including the human hand. These two recent advancements combined with the decreased cost of electronics boards and the growing popularisation of smart phones, have made the project this report describe: a mobile application controlling a robotic hand powered by arduino board, with the addition of function of gesture recognition, possible.

1.2 Motivation
The writer has experience working in a company that has created similar products of robotic hands that help curing patients suffering from strokes to recover the mobility of the impaired hand caused by the stroke. Such a product is meaningful to society as it helps people to regain their health. Seeing such a product, and recognising the possibility to improve certain features and functions their current software ecosystem provided fueled my motivations of choosing this topic as the final year project. The possibility to utilise the various skills I acquired with that working experience is another motivating factor as well.
1.3 Project Objectives and requirements

It should be noted that, there is a major shift of project objectives during the course of the project. In the interim report, I have stated that the previous goal of focusing on the modelling of the robotic hand had been discarded and the new focus is the creation of various softwares parts. To restate the renewed project objectives and requirements, they are listed as here.

<table>
<thead>
<tr>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) To create a robotic hand that is flexible and can mimic many motions that the human hand is capable of.</td>
</tr>
<tr>
<td>2) Via delicate programming and software design, ensuring the movement and control of the robotic hand is smooth and not rigid</td>
</tr>
<tr>
<td>3) To implement synchronization between the robotic hand and the mobile application such that the control is smooth and there is no issues concerning buffer overflow</td>
</tr>
<tr>
<td>4) To implement sensors on the robotic hands such that it can not only be controlled but can receive signals via touching with the materials attached to the surface of the hand</td>
</tr>
<tr>
<td>5) To implement a small-scale machine learning model on the mobile application so that it can use the camera to recognise hand gestures, then it can control the robotic hand based on what gestures the users make.</td>
</tr>
<tr>
<td>6) (Tentative) To create a set of APIs to control the robotic hand.</td>
</tr>
<tr>
<td>7) (Tentative) To allow the use of online platforms to control the robotic hand remotely via the Internet</td>
</tr>
</tbody>
</table>
There are parts that cannot be finished due to an estimation of the size of the project and some unforeseen limitations that cannot be easily surmounted with such limited time and manpower. I will restate such limitations in the later parts of this report.

1.4 Contribution of the project:
This is an individual project so everything delivered is solely the product and responsibility of me. By this time, however, I have to say the workload of this project is a bit too much and it would be much better if I had asked a friend to join to compete this fyp cooperatively. That way, the current finished parts can be more polished, and the parts that can be done with more manpower can be implemented much more easily as well.

1.5 Structure of the report
As there are four different parts of codes written in different platforms, which include the ESP32 board, the mobile application, the middleman online server, and the facial recognition website, it would only make sense for the exposition of them in sequence. Such a sequence of discussion is going to be present in the subsequent parts of 1) Project background and literature review, 2) Project methodology, 3) Experiments and Results, 4) Conclusion and future Works. Diagrams will be scattered in the whole report to show additional useful information.
2. Project Background and Literature Review

The project background will talk about the existence of technologies available in the public domain. And how they enabled me to finish the project in a desirable way.

2.1 Robotic hand model

There are a total of three possible approaches considered possible during the intermediate phrase period, during which most of the software part of the mobile application has been completed. The approach includes: creating the 3d model of the robotic hand completely from scratch, using the model of the Ada Hand by OpenBionics, using the robotic hand created by the Inmoov project. Each approach has its pros and cons.

2.1.1 Creating the 3d model from scratch

This is of course the most difficult approach. Instead of counselling on others’ 3d models, the approach requires me to create the 3d model ourselves. The advantages, however, are plenty. First, by customising the 3d model from scratch, we can create it in such a way that can fit our specific needs and demands. Second, doing this can allow me to learn the most in the fyp because the additional requirement is very challenging and requires fresh perspective and skills. Third, this will increase the originality of the project. Unfortunately, there are several significant disadvantages involved in this approach. First off, the writer of this report does not have any expertise in 3d modelling, creating a 3d model from scratch would cost considerable time. Second, the writer cannot guarantee that the 3d model must work perfectly without printing it out. If the 3d model turns out not working, the created 3d models would be a waste.
2.1.2 Ada Hand by OpenBionics

This is indeed an excellent open source 3d model of the human hand available online. Out of the 3 options of the hand model, this perhaps looks most elegant. One advantage of using this design is that it consists of not as many parts and its construction is more simple than that of the others. However, there are a few disadvantages accompanying choosing this model. First, there are not a lot of tutorials and information available online about the construction of this 3d model. There are scarcely any youtube videos demonstrating how the individual parts are connected to each other. Second, there are certain non-printable components that were not rapidly available at the moment of printing the 3d model. Third, and most importantly, each individual part of the 3d printing components are really large, each may require over 10 hours for printing out. The writer considers it to be dangerous to print this because it would waste a lot of material if the 3d printer suddenly does not work. The power outages that happened several times in the Haking Wong building has confirmed this fear, which, luckily, did not disrupt the project as most 3d printable parts has already been printed.

![Ada Hand by Open Bionics](image)

Figure 1. Ada Hand by Open Bionics
2.1.3 Inmoov robotic hand

The Inmoov project is a well-known French robotic project that aims at creating a full-scale humanoid robot constructed with 3D-printable human body parts. Most known human parts have their 3D models of Inmoov freely available online in the public domain and they allow the public to access and download their STL files. Although the robotic hand is not as elegant in structure compared to the Ada hand and its consistence of a large number of different 3D printing parts, there are a lot of advantages because of which making this an excellent option for our project. First, it consists of many small parts that can be printed. Because of this, I can print the parts individually and avoid the risk of the malfunctioning of the 3D printer. If a tiny part is misprinted, just printing that again in 10 minutes. Second, because it is the most popular design of robotic hands online, there is a lot of readily available information online about how to construct the individual parts together. For instance, there are a number of different YouTube videos demonstrating how to put the parts together. Finally, one can incorporate the functions of sensory response in its tips.

![Inmoov Hand](image)

Figure 2. Inmoov Hand
2.2 Mobile application

Since the inception of the release of the iPhone, mobile application developments became one of the hottest businesses in the industry. The two dominating platforms on which software is created are the Android operating system created by Google and the iOS system owned by Apple. However, in the past, because the two different systems were in two separately ecosystems, the codebase used in Android, which is largely consisting of Java and Kotlin code, and the codebase used in Apple, which is based on Objective C and Swift, were not mutually operable and intelligible. This results in the common problem in the industry where the programmers have to be separated into two teams, one team working on the iOS release, another team working on the Android release. In recent years, cross-platform software development has finally reached the state of being able to create products that can be released to the public in the Apple appstore and Google Playstore. The writer deems it worth writing the software as one that is both available in Android and iOS. There are, however, three possible approaches for writing cross-platform applications. Three will be considered here.

2.2.1 PWA and webview

Progressive web applications are applications just like the websites that we see online on a web browser. By creating a website that caters our needs, we put it on top of the layer called webview in our mobile application. So when we open the application, what is shown to the user is actually a stored webpage. The user thinks he is using a native application but is actually browsing a website. The application itself will link several functionalities that require the use of several native functionalities that use the native api of the mobile phone such as Bluetooth connectivity, online access, et extra. Because in our approach, we want to control our robotic hand via Bluetooth, if we were to adopt this approach, we have to link the webpages to the native Bluetooth api calls. The main advantage of this approach is that if you are a web developer, you can create a PWA very quickly.
2.2.2 Flutter

As the major cross-platform application writing platform created by Google, it has attracted a considerable number of developers to develop their apps in this platform. There are several advantages of using this approach. First, most functionalities that you require in the native Android and iOS libraries are available because of the meticulous preparations by Google. Second, the literature of how the software libraries and the underlying software architecture are made extremely easy to read and understand. One disadvantage in adopting this approach, however, is that the software platform uses a programming language called Dart, which is not a very common language to use. So extra learning has to be done to program a software in Flutter.

2.2.3 React Native

React Native is a technology that is based on the website writing framework React, created and maintained by Facebook. As the most popular cross-platform development platform, React-Native not only supports making software projects available in iOS, Android, you can also release the products in Windows, Mac and even Xbox with just some minor modifications of code. The most significant disadvantage of using this approach is that the original first-hand literature of the platform is written in such a way that it is not easy for beginners to read. But there are several huge advantages in using this approach compared to that of the others. First, as the most popular cross-platform mobile application platform, there are the largest number of libraries catering for different software needs. For instance, all Bluetooth serial and Bluetooth BLE calls, as well as the access of cameras, are supported by not only one but a large number of similar libraries from which the writer can choose the best one. Second, as the platform is largely written in Javascript, the coding speed is definitely faster than Dart. For instance, in Dart, if you miss the character “;” in the ending of a programming statement. You won’t be able to compile, whereas in React Native, you can miss the character without breaking the functionality of the code. This permits rapid development.
2.3 Controlling method

There are 4 possible approaches for us to control the robotic hand once it is connected to the mobile application. Once it is connected, how should we control the robotic hand. There are, I can think of, 4 total common methods: normal roller control, extra glove control, preset-configuration control and gesture recognition camera.

2.3.1 Normal slider control

The most straightforward and the supposed easy way to control the hand, is to have the mobile application a number of 5 sliders, each representing a finger. When we slide it to one extreme, the finger contracts to the maximum, to the other extreme, we have the fingers completely relaxed.

2.3.2 Extra glove control

This is a common approach in many diy projects. The application maker will create an extra glove that can detect the wearer’s hand to different extent of the fingers’ contraction and relaxation. By wearing the glove, it is possible to do a hand gesture, then the robotic hand correspondingly moves. This approach, however, is deemed impractical in our case, as the current project already has incorporated 4 different code bases. An extra component will make the completion of the project impossible.
2.3.3 Preset-configuration

This approach is basically resetting a set of different gestures’ parameters both in the electronic board and the mobile application. For instance, in the electric board we preset that if there is a bluetooth signal sending over a message of “number5”, the robotic hand will do the gesture of the number five, or paper in rock, paper and scissor. After that, every time we open the app, if we connect the application with the robotic hand we then click on the button representing number five, a bluetooth signal will send the message to the robotic hand and the robotic hand will then move accordingly.

2.3.4 Gesture control and the Mediapipe project

The recent improvements in deep learning and neural networks algorithms have made the reality of pattern recognition of voice, written characters and even faces possible. The recognition of the position and movement of the human hand, is under active development as well. The recent breakthroughs have resulted in the Mediapipe project, where the researchers in Google have rolled out a project where they can use the camera to capture the movement of the human hand. Their system is so powerful that it can accurately capture the location and movement of even very delicate and nuanced movements of the small parts of the human finger bones. Their approach is to firstly recognise the location of the human palm first, then the finger and knuckles locations are subsequently handled. They have used 30000 images to model the neural network which resulted in an extremely accurate network that can be used to recognise the most nuanced movements.

The Mediapipe project is available in both the mobile platform, the web platform and the local computer. The major difficulties involved using their library in not the mobile
application, however, is the extreme complexity in setting the development environment, where there are a lot of extra tools and libraries needed to incorporate into. The website api, however, is relatively easy to use and programme in. Several wrapper libraries of the Mediapipe hand tracking algorithm have been released for extra easy programming.

This approach of using this Mediapipe, where a camera captures the gestures of the physical human hand, and signal is then sent to the robotic hand, which then moves accordingly is an interesting and excellent approach to control the robotic hand.

Figure 3. The accurate results resulted from the Mediapipe library
3. Project Methodology

3.1 Overview

A higher level overview of the project would be beneficial at this stage as this project involves complexity to a certain extent. As you can see, in the diagram. There are four major parts in the system: the part of the robotic hand consisting of the ESP32 board which receives bluetooth signal from the phone, the part of the mobile application that connects and sends the message to the ESP32 board, the online middleman server that redirects data flow between the website hosted on Github and the mobile application, and the website that does gesture recognition. The exposition will start from the Robotic Hand, then the mobile application, then the bridge server, then the website.

Figure 4. A summary of the architecture of the entire system
3.2 3d Printer

Since the components of the robotic hand have to be constructed, we need to create it out of certain materials. Plastic is the ideal material as it is cheaper and lightweight thus suitable for projects of this kind. The easiest way to such construction is to use a 3d printer to print the gcode exported from the precreated stl files. The 3d printer in the Makerlab serves exactly our needs. It took GCode in and printed the file in plastic form. It is of the model Anycubic Mega i3, which is cheap and affordable. However, the 3d printer has a few problems that will lead to results that are not ideal. First of all, of certain shape, it cannot print properly and the printed shape will not be identical to the stl file. In addition, very often, the printer was jammed and the printing continued but the printing was not actually working. Several models have to be reprinted multiple times just because the printing stops midway through the printing process.

As the 3d printer has several problems, the 3d printing process is far from ideal, and considerable time is added to grind the printed plastic parts. In addition, the resulting robotic hand is far from reaching the level of movement smoothness that I had expected.

Figure 5. Anycubic Mega i3 in the Makerlab
### 3.3 Facility

The project is largely done and conducted in the Makerlab in the Haking Wong building at HKU. It is a large room with many facilities and tools, such as screws, 3d printers and grinder. The problem, however, is that there are multiple blackouts happening in April. This drastically increased the risks of 3d printing as electricity might run out anytime. For this reason, reprinting portions of the 3d printing that are in bad shape became impractical as there would be enough time.

### 3.4 Robotic Hand

As the robotic hand is the major part of the project, it is relatively complex. The following diagram shows how the various components of the robotic hand come together. As you can see, it consists of the ESP32 board connected to various electronic parts including battery, current changer, servo motors, and the printed 3d models. The way the electronic parts come together is thanks to Mr. David Lee in the Maker Lab, who instructed me to buy the parts and construct them together. Such a construction allows the robotic hand to operate on its own without plugging it into a charger directly all the time.

![Diagram of Robotic Hand Components](image)

Figure 6. A simple diagram of the components of the robotic hand.
3.4.1 Battery
In a case where only 1 low-powered servo motor is needed, a power bank sold by Xiaomi or other brands might surface the need to power both the servo motor and the ESP32 board. The problem, however, is that we have a total of 5 servo motors, which are not low powered at all, because we need to exert enough force to use threads to control the fingers to move, the simple mechanism of plugging the micro usb cable from a battery bank to the micro usb slot of the ESP32 board might not be sufficient. For this reason, with the guidance of David, I bought one bank of li-on battery, A high discharge lipo battery of 1800mac of the brand ACE from Taobao. It is able to power both the ESP32 main board and the PCA 9685 servo motor controller board. However, as I was developing on the ESP32 board, the battery is unable to be charged. This does not turn out to be a problem thanks to David’s specific instructions to buy 2 extra spares for the materials.

3.4.2 Voltage Regulator
As we do not need as much power as the battery is able to offer, we want to cap it to a limit. In our case, it is enough to cap it to 5v. Because of this, we need a voltage regulator to ensure that the voltage is actually 5v. I bought a HiModel 6-50V 20A, which is a voltage regulator that allows you to change the voltage to a range between 5v and 9v. With this we are able to charge both the ESP32 and PCA9685 boards properly! Now the battery set up of our project is finished. An extra on and off switch is connected to allow us to turn the battery on and off.

Figure 7, Figure 8, The Battery ACE 1800mac and Voltage regulator HiModel 6-50v 20A
3.4.3 Servo Motors

As the following diagram demonstrates the mechanism through which the individual fingers are moved, namely via the pulling of threads connected to the tips of the fingers, as our hand is an robotic hand without human’s direct pulling the strings, we should use another mechanism to pull the strings.

Figure 8. The mechanism through which our hand moves.

And in the design we have chosen, we use servo motors to pull the strings. Our servo motors chosen are of the model MG996R. They are extremely reliable and affordable. In addition, they are quite powerful, surely enough to move the fingers in our case. Furthermore, they fit perfectly within the slots of the 3d model we have chosen.

Figure 9. Servo Motor MG996R
3.4.4 PCA 9685
The board ESP32 does not have enough pins to be connected to a total of 5 (or potentially 6 if we have to add the ability of wrist rotations). So, in addition to the ESP32 board, we need an extra piece which can be connected to many servo motors so that the fingers can move. In our case, I bought a servo controller board of the model PCA9685, it allows me to control at most 16 servo motors. By attaching the battery unit and ESP32 with a few cables, in addition to the preliminary program that helps control the servo motors written by David, now it became extremely easy to write a program to control the servo motors. However, as the board is not exactly reliable, all three of the units I originally bought broke and I had to buy an extra one in Sham Shui Po.

Figure 10. PCA9685 Servo motors controller

3.4.5 ESP32
In the beginning, I was programming on an Arduino Uno board, attached with an extra Bluetooth Serial unit. However, it has a few imperfections. First, it is not a small unit and with the extra bluetooth unit it occupies a considerable space. Second, it is not very powerful and for many functionalities like wifi connection it has to be implemented by attaching extra modules.

David informed me to buy the ESP32 board. Compared to Arduino Uno, which has only 1 core, no wifi, no Bluetooth support and 2 kb ram, ESP32 has 2 cores, support for both wifi and Bluetooth and a stunning amount of 512km ram. Such additional power can allow us to do extra programming even if we have a lot of extrafunctionalities in the aftermath of this
project. And most importantly, with the installation of a few libraries, the same style of code that I wrote for Arduino Uno can be reemployed.

There are some problems encountered in the project related to ESP32. There were two scenarios that occurred that I had to replace my ESP32. First, I incorrectly attached the 5v power supply to the 3.3v input pin on the board, which led to the board suddenly on fire. Luckily no injuries happened eventually. Second, after a prolonged session of programming where I turned on the power supply from my battery, the ESP32 overheat and stopped working. Luckily, the cost of the ESP32 board was extremely low, and I bought extra spares.

Figure 11. ESP32 board

3.4.6 Self-created 3d model

As said in the literature review, one option for printing the 3d model is to adopt the 3d model I created. Indeed, there was an attempt to do so. However, as I tried to create the 3d model, I faced a number of issues.
Unlike my fellow labmates in the makerlabs, I did not have much experience using 3d modeling softwares. So I had to do everything from scratch. As a result, I spent a considerable amount of time learning how to use the software of 3d modeling. Using Both Sketchup and Shap3r, I was able to create a 3d model in the following diagram, which is a useful prototype. The problem, however, is that after I printed one of the fingers in the stl files, there are quite a large number of adjustments needed to be made. For instance, the angle of curling of fingers, the size of the holes through which the threads ran through are not proper.

Figure 12. The 3d model prototype I created

Figure 13. One of the fingers of the prototypes.
It would have been possible to perfect the design in a good form if I had an extra month. But as I am not really experienced in using 3d modelling software, a tiny adjustment takes at least a few hours. Most importantly, 3d printing, unlike 3d modelling and software programming, takes a lot of time. If I continued to adopt the approach of creating the 3d model by myself, it would have been impossible to finish other more important aspects of the project, namely, the programming aspect of the software.

3.4.7 Inmoov 3d model and the construction of the robotic hand

As the constraints of time and experience had convinced me that I did not have enough time to finish the project on time, I decided to use an available 3d model of robotic hands online. I considered both the aforementioned Ada Robotic Hand and the Inmoov hand. In the end, I decided that the Inmoov hand was a superior choice for a number of reasons. First, most materials required for the Inmoov hands are more straightforward and simple than the Ada hand. In addition, there are a large number of tutorials both in text formats and youtube videos format online that I could consult to build the robotic hand.

At the end of the day, I printed every component (and because I printed quite a lot so the resulting hand consists of a total of three colors: silver, pink and brown). There were a number of issues I faced though. First off, there were countless occasions when 3d printing stopped in the middle of printing, leading to the problems of wastage and unnecessary reprints. Second, 3d printed products are not precise because of the limitations of the 3d printers, I had to do a lot of extra grinding of the different parts of the fingers so that various parts fit together. But even then, there were some problems that could be fixed even with grinding so the movements of the robotic hands are not as precise as I wanted it to be.
Figure 14. These are just some of the wastage because the sudden jam of the 3d printer.

After the 3d printing, the next step is the construction of these 3d printing parts. The most difficult part is attaching and putting the lines into the appropriate holes of the fingers and palm, and ultimately attaching them to the servo motors. The thread I use is a fishing line of the type J-Braid Grand x8 40lb.

Figure 15. The fishing braid line J-Braid Grand x8 40lb.
There are certain problems involved in attaching the thread. First off, as the original document of the Inmoov Hand suggested we should use fishing braid lines of 200lb, which are extremely expensive and could not be bought in HK easily, by using these fishing lines of 40lb make them relatively fragile. On several occasions they broke and I had to reattach them again. Second, the original design required me to attach two fishing braid lines to each finger, which is impractical in our case because the imprecision of the 3d printed fingers shapes. At the end, I followed an instruction online which used a modification of the robotic hand, which attached each finger with one fishing braid line and one rubber elastic band. This adoption significantly simplified the mechanism of movement of the fingers of the robotic hand.

Figure 16. Threading those fishing lines in the holes of the 3d model
3.4.8 Receptive sensors attached to the fingertips

Despite the design of the Inmoov hands could cater for adding receptive sensors in the fingertips, because of the time constraints and extra complexity in programming given that the receptive sensors are added. But the ESP32 board can easily support receptive sensors, this could be added after the completion of the fyp.

3.4.9 Final look of the robotic hand

The left part is the electronics component, consisting of the battery, voltage regulator, servo motors controller and the ESP32. The right part consists of Servo motors,

Figure 17. Final look of the robotic hand
3.5 Program of ESP32

The movement of the robotic hand requires the programming of ESP32. The program basically does two things: receiving messages from the mobile application and based on these messages, controlling the servo motors. Thanks to David's generosity, who wrote an easy to use API based on the library of PCA9685. A simple instruction can instruct each finger to a certain degree.

3.5.1 Bluetooth part

Although ESP32 supports both Bluetooth classic and ble, as the original code I have written for the mobile application was based on Bluetooth classic. Adopting Bluetooth classic will give us the extra advantage of simpler code and more bandwidth and quicker speed of transmission. Online tutorials and instructions provided me with the setup part of the Bluetooth part code, which is relatively simple.

3.5.2 Logic of the receiver and the format of the message

The current form of the program works like this. I based my program on the idea that gesture control was a good idea, so the ESP32 mainly intercepts messages which controls the robotic hand. Our program ensures that each each There are two forms of messages: predetermined messages and normal messages.
3.5.2.1 Pre-determined messages

There are a total of 13 gestures that are pre-determined. As long as messages of the names of the gesture are received, the robotic hand will do the corresponding gestures. The following table summarises the messages, their name, and the extent of the fingers curled.

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
<th>Fingers position (thumb, index, middle, ring, pinky) (70 is totally uncurled, 165 is totally curled)</th>
</tr>
</thead>
<tbody>
<tr>
<td>rock</td>
<td>Rock in rock paper scissors</td>
<td>165,165,175,165,165</td>
</tr>
<tr>
<td>paper</td>
<td>paper in rock paper scissors</td>
<td>70,70,70,70,70</td>
</tr>
<tr>
<td>thumpUp</td>
<td>Thumb Up</td>
<td>70,165,175,165,165</td>
</tr>
<tr>
<td>gun</td>
<td>Gun gesture</td>
<td>70,70,175,165,165</td>
</tr>
<tr>
<td>middleFinger</td>
<td>Middle Finger</td>
<td>165,165,70,165,165</td>
</tr>
<tr>
<td>pinkyPromise</td>
<td>Just extend the pinky</td>
<td>165,165,175,165,70</td>
</tr>
<tr>
<td>number1</td>
<td>Gesture of number 1</td>
<td>165,70,175,165,165</td>
</tr>
<tr>
<td>number2</td>
<td>Gesture of number 2</td>
<td>165,70,70,165,165</td>
</tr>
<tr>
<td>number3</td>
<td>Gesture of number 3</td>
<td>165,70,70,70,165</td>
</tr>
<tr>
<td>number4</td>
<td>Gesture of number 4</td>
<td>165,70,70,70,70</td>
</tr>
<tr>
<td>number5</td>
<td>Gesture of number 5</td>
<td>70,70,70,70,70</td>
</tr>
<tr>
<td>number3 German</td>
<td>Gesture of number 5 in Germany</td>
<td>70,70,70,165,165</td>
</tr>
<tr>
<td>Spiderman</td>
<td>Famous Spiderman gesture</td>
<td>70,70,175,165,70</td>
</tr>
</tbody>
</table>
3.5.2.2 Custom gestures

As we do not want the robotic hand only supporting a set of simple gestures, we should have a more elastic way to support a larger number of potential gestures. Because of this, I allow a different message that is allowed to be sent to the ESP32 board. It is of the following format.

<table>
<thead>
<tr>
<th>thumb</th>
<th>Separator</th>
<th>Index</th>
<th>Separator</th>
<th>Middle</th>
<th>Separator</th>
<th>Ring</th>
<th>Separator</th>
<th>Pinky</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>70</td>
<td>#</td>
<td>70</td>
<td>#</td>
<td>70</td>
<td>#</td>
<td>70</td>
<td>#</td>
<td>70</td>
</tr>
</tbody>
</table>

For instance, this can be a customised message representing the paper gesture. The message would be “!70#70#70#70#70^”. In this way, we can support any new gestures that a mobile application sends.

3.5.2.3 Potential third mode

In the future, a new additional mode of the message can be added. As the current messages are either in the form of predetermined gestures or customised messages. Yet both of these messages are discrete, which means we cannot yet continuously send signals to control the robotic hand with the direct changing of position from the phones. So certain movements such as changing movements of the fingers immediately when we change movement signal from the phone. This function, however, can be supported in the future release of the project.
3.6 Mobile application

This is the main part of the project. In the end, out of three different development technologies, PWA, Flutter and React Native, we have chosen React Native because of its abundance of libraries, easy-to-program language Javascript and the writer's familiarity with developing with this language. The following pages will focus on the five functionalities of the application: connection, built-in gestures, recording-gestures, history and QR-code binding.

3.6.1 Connection

We want our application to be able to connect to the robotic hand via Bluetooth. In React Native, the library React Native Bluetooth Serial serves exactly our purposes. With the function

i) BluetoothSerial.connect(id).

We will be able to connect and pair our mobile phone with the robotic hand. There is one limitation, however, which is that it takes around 15 seconds to scan the devices. The library allows simultaneous connection to different devices, so in the future, if there are multiple robotic hands, they can be all simultaneously connected to the application. Then with one application, we can make the robotic hands move at the same time.

The connection portion consists of four pages, which are listed below
Connection page 1: Waiting page

This page waits for the user to click connect, which then initiates the searching of devices process.

Connection page 2: Waiting screen

This page is shown during the scanning process. It takes around 15 seconds.
Connection Page 3 (Showing the scanned devices)

This page shows the devices that are available for connection. As in this case ESP32 is shown.

Connection page 4 Connection success page

This page is shown when you successfully connect with the bluetooth device. When you click disconnect, you can reconnect the device.
3.6.2 Built-in gestures

Once connected you are allowed to press the pre-configured modules now in the Hand page. The following are the currently supported gestures. They all have their corresponding gestures in the ESP32 board.

**Rock**

Gesture of the Robotic Hand
Rock in Rock, Paper, Scissor

**Thumb up**

Gesture of the Robotic Hand
A normal thumb up

**Gun gesture**

Gesture of the Robotic Hand
Point a gun to the right

**Pinky Promise**

Gesture of the Robotic Hand
Pinky promise

**Number 1**

Gesture of the Robotic Hand
Number 1

**Number 2**

Gesture of the Robotic Hand
Number 2
Number 3

Gesture of the Robotic Hand
Number 3

Number 4

Gesture of the Robotic Hand
Number 4

Number 5

Gesture of the Robotic Hand
Number 5

German Number 3

Gesture of the Robotic Hand
Number 3 German

Spiderman

Gesture of the Robotic Hand
Spiderman!!
3.6.3 History page

This is a page where all the past history of the actions are recorded. They include connection, message sending, message recording and QR code binding.

3.6.3.1 How is the history page supported?

The application has a built-in database system that adopts a sql style database system called WatermelonDB. It supports the CRUD (Create, Read, Update, and Delete) operations of a full-scale database system. With it, each time an operation is done, the operation will be recorded into the database. This is how the history is recorded. In addition, when the remove is pressed, the R operation of CURD will be done. The database supports another important function, which is the recording of customised messages, explained in the next section.
3.6.4 Recording page

As explained above, I wanted the whole system able to support more than the predetermined gestures. For this reason, I included a portion of the code to implement this function. 5 sliders are added, each for one finger. The further you pull the slider to the right, the more curled the fingers are going to be. You get to decide the name of the gesture you like. In the future, the functionality of customising the image representing the gesture you like can be added.

The gesture is stored in the exact form mentioned above, namely this:

<table>
<thead>
<tr>
<th>thumb</th>
<th>Separator</th>
<th>Index</th>
<th>Separator</th>
<th>Middle</th>
<th>Separator</th>
<th>Ring</th>
<th>Separator</th>
<th>Pinky</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>70</td>
<td>#</td>
<td>70</td>
<td>#</td>
<td>70</td>
<td>#</td>
<td>70</td>
<td>#</td>
<td>70</td>
</tr>
</tbody>
</table>
With the support of the WaterlemonDB, the application to store the recorded gestures. Even after you turn off the application, the recorded messages are still there. Because of the support of data synchronisation of the WaterlemonDB with a database online, in the future, it is possible to reroute the data from the app to the cloud, or from the cloud to the application. One possible scenario is the client will be given more gestures to use if they suppliers decide to update the database.

**New recorded gestures**

After the gestures are recorded, in the original page of Hand, they will be shown.

As you can see the image of added new gestures are only `</>`, which represents code as an icon. In the future release, I can improve it by allowing customising images as mentioned before.
3.6.5 QR Code binding

As mentioned by Dr. Chan, the application would be better with the support of online access. In addition, Mr. David provided me with the excellent idea of allowing the user to use gesture recognition of the camera. As React Native has yet to support the gesture library most accurately for this task, I have decided to combine the gesture recognition and online access to one unit. By going to a certain website on your browser on your computer, the computer will show a QR code and do pattern recognition with the camera. Once we use our mobile phone to scan the QR code, we will be able to do gestures in front of the camera, the browser runs Javascript code and determines what gestures we are doing. Then it sends back the signal to the mobile application, which instructs the robotic hand to move.

The page of QR code binding
3.7 Website

Our website is consisting of four parts: Mediapipe, Github hosting, FingerPose.js and QR Code. On the surface it is a relatively simple static page, but it involves various technologies. The basic overview of the process of the website is like this.

![Workflow of the website](image)

As the literature suggests, we use the Google Mediapipe technology. But instead of using directly their APIs, we use a third-party layer called Fingerpose.js. Fingerpose.js is an excellent library if you want to do simple hand gesture recognition.

The way fingerpose.js works is this. Based on the numbers and values the Mediapipe library got from recognising the hands movement, it calculates the level of curliness and towards which positions (upwards, downwards, left and right) each finger are doing. And based on this, it calculates the confidence level of which precoded gestures match the characteristics that the camera has captured. With five fingers, each fingers have 3 degrees of curling (no curl, half curl, and full curl), plus each fingers supporting 7 directions (vertical up, vertical down, horizontal left, horizontal right, diagonal up right, diagonal up left, diagonal down right and diagonal down left), a large number of hand gestures can be recognised.
However, the website cannot be hosted just on an unencrypted normal http server. And the reason is that in recent years, modern browsers have upgraded the security level so that if the website you are on is not encrypted, you are not allowed to have access to cameras or microphones. Because of this, the website has to be hosted in a location with https encryption. The solution I adopted is to use Github’s free website server where I can host one website there. It is free and extremely easy to program on as you can directly modify the code in the github page.

For the mobile application and the website to be connected, we need to use a qr code mechanism. By scanning the qr code shown on the website, your application will be connected to the website. Each time the website recognises the gestures, the message will be sent to the application, which then controls the robotic hand.

As the gesture recognition is not 100 percent perfect, in the design of the whole system, the code of the mobile application has been modified such that you have to hold a certain gesture for more than 2 seconds so that it is recognised for more than 2 seconds so that the system actually recognizes the gestures you are making.

Figure 19. The look of the website
But as there is no support for point-to-point connection between the mobile application and the website, which is impossible because our phones and computers do not have fixed ip addresses, we need an extra middleman server between the mobile application and the websites, both of which connect to a fixed ip address. The middleman will be explained in the next section.

In the future, continuous gestures recognition should be able to be implemented as Mediapipe is extremely accurate for the location of each finger and its position.
3.8 Middle-man server

The middleman server is based on the technology of Node.js and its library of Socket.io. Node.js is a runtime that allows you to run Javascript code as a scripting language like Python instead of just running it on the browser. Socket.io is a library allowing you to do simple chat-room style messages broadcasting and transmission with simple code without complex handling of web socket.

Both the application and website have installed the Socket-io Client. With it, it can connect to a certain ip address of the Socket-io Server, which then redirects messages from the application to the website and from the website to the application.

In the past, the writer had experiences writing software systems involving communications between socket io server and socket io clients. The past approach is to use two extra tables to store the matching between the mobile application and the website so as to support multiple devices and multiple websites. Experiences have told me that this approach drastically increases the complexity of the project. So, in our case, we adopt a new approach: the QR code approach.

In our QR code approach, the whole thing works like this: You open the website, as it connects to the middleman server, there is a connection id created between them. This connection id will be shown on the web page as a qr code. Now, when you scan the qr code using the mobile application, the mobile application will then send the connection to the middleman server. Now the middleman server knows which mobile application has known which website it wants to be matched to, the connection id between the middleman server and the mobile application will then be sent to the website. The website stores this id. Now, each time the website recognises a gesture, it will send the message in addition to the id to the middleman server. The middleman server will then send the message according to which id it has received.
Because in the whole process, the middleman server is basically stateless that it does not have to have extra variables or arrays to store the ip addresses of the websites and the applications opened, the complexity is simplified beautifully.

The whole process can be summarised in the following diagram

Figure 20. The process of the interactions between the middleman server, the website and the mobile application.

3.8.1 AWS and Https server
As the server has to be connected to the website we mentioned, which is https encrypted, the server itself has to be https encrypted as well. As I want the project to be able to be run even after my graduation from HKU, instead of choosing HKU’s free https server, I decided to set up my own server. I chose AWS’s EC2 server as it allows a free-tier usage of one year. In addition, I am able to get a fixed ip address for the project for free.

By opening a certain port from the ec2 instance, which is running Ubuntu Linux, I was able to run softwares on top of it. In the security setting, I opened an extra port for outside connections to the server in the ec2 instance, which can now open service in applications that have to use that port.
But in the end, as I had to create a https server, I had to get necessary public key certificates from a recognised online body. Fortunately, there are many free online ssl certificates online. I was able to get one from ZeroSSL. However, as it is free, it can only last up to 90 days. In the future, I will have to renew the certificate by applying a new one. But for now, everything is set and done and the whole process of online gesture recognitions is complete.

Figure 21. The AWS ec2 instance created for this project.

Figure 22. The SSL certificates for the middleman server.
4. Results

In the previous part, methodology, the whole system is shown, which consists of a number of 7 functions, they include:

i) Robotic Hand Connection
ii) Predetermined gesture making
iii) Recording self-made gestures
iv) Making self-made gestures
v) History browsing
vi) Connecting with the Internet website
vii) Recognition of gestures in the website
viii) Moving robotic hands based on recognised gestures

On a large scale, despite some tiny problems, such as the lack of smoothness of movement because of the limitation of the 3d printer, all these 7 functions have been successfully implemented to a very satisfactory level.

Each part's success can be verified by the following video links.

Connection
https://drive.google.com/file/d/1MXh_iGrY2FhTQoJ0vJnoHAEPtX7vYjdv/view?usp=sharing

Predetermined gesture making
https://drive.google.com/file/d/1bctpFXmH8ng_cFOOfjcr3G07hVhRtnf6/view?usp=sharing

Recording and making self-made gestures
https://drive.google.com/file/d/1KPX4olRtK7lD4SWmVInrkTKJJkhHbrX1/view?usp=sharing

History browsing
4.1 Limitations and Future works

Despite what I was able to complete, there are still a large number of limitations where I can focus my future work on. Below are the most significant ones.

**No self-made robotic hand model**

Currently, the robotic hand is based on the Inmoov robotic hand. Given more time, I will be able to create a robotic hand that matches my own sense of style and utility.

**No sensor**

The ESP32 board directly supports adding capacitive sensors. In the future design of the robotic hand, the tips of the fingers can be added.

**A lack continuous control of fingers.**

As the current design is still based on the model of gestures, in the future release, I can directly use the angle and location of the joints and palm and fingers so that as the camera capture the precise movements, the exact same movements are replicated in the robotic hand.
Awkward hand movements
The lack of precision of the 3d printer has resulted in the lack of precision of the printed products, so that there are certain movements looking awkward. For instance, certain fingers have problems completely uncurling after curling.

Image is not added
Now all custom gestures use the same icon. In the future, the user should be able to use his own jpeg files as the image.

Interface too simplistic
The current interface powered by Native Base is still too simplistic for an actual release. A better GUI version can be released in the future.

No backend database synchronization
Currently all data is only stored in the device, in the future, they can be synchronised with a cloud database.

Proper case for everything
For programming convenience, the electronics are now all exposed outside without being covered properly. In the future a proper case can be made.
5. Conclusion

Our robotic hand project consists of 4 parts: the robotic hand part, the mobile application part, the middleman server part and the website part. On the whole, they work extremely well with each other and implement most functionalities they were designed for accomplishing.

Our robotic hand consists of the battery unit, the board unit and the physical hand unit. The battery consists of a 1800mac li-po battery, whose output is capped to 5v with a voltage regulator. The board unit consists of ESP32 and the servo motor controller PCA9685. With the power of the battery and program of the board unit, the servo motors of the physical hand unit are moved, which then move the fingers. More expensive and more reliable components involving the electronics part can be sought for future release.

The program of our ESP32 program opens a Bluetooth Serial server and based on the messages it receives, sends the appropriate signals to the servo motors so that the fingers do certain gestures appropriately. Currently, two types of message are supported: pre designed gestures and customised gestures. A specific protocol of the custom gestures are included. In the future, the support of continuous finger movements may lead to a 3rd extra mode of messages.

Our mobile application supports connection, gestures control, recording gestures, history, and online binding. All these functionalities work properly. There is space, however, for improvements over the GUI and minor bugs to improve the UI flow of the program. In the future, the writer may write a wrapper to the Mediapipe hand recognition library for React Native so that the recognition works alone in the mobile application without having to rely on an extra website.

The website and the middleman server works quite well, the recognition works quite well for now. In the future, improvements of the accuracy of recognition, and the continuous
movement tracking can be included as well. In addition, the website can have more functionalities built in, such as synchronising other information such as history and customs gestures recorded in the mobile application.

Indeed, the current form of the whole system solves already one of the problems my previous employer meets. As my employer wanted to the patient to wear a robotic hand glove in one hand that is unable to move, and another hand to do a certain gesture, and the hand will do the gesture that the other hand does, thus training the patient's electricity circuit of his hand, and the patient will gradually regain his hand's mobility. The current form of the project already satisfies this requirements and because of this, I find the project attaining a satisfactory result.
References

https://www.sketchup.com/

https://github.com/andypotato/fingerpose


https://github.com/derektu/react-native-bluetooth-serial


http://inmoov.fr/hand-and-forarm/

https://www.himodel.com/electric/HiModel_6-50V_Input_20A_Large_Current_Switching_Mode_Voltage_Regulator_SBEC.html

https://openbionicslabs.com/obtutorials/ada-v1-assembly
https://google.github.io/mediapipe/solutions/hands.html

Progressive web apps (PWAs) | MDN. (2021, February 24). Progressive Web Apps (PWAs) | MDN.


https://randomnerdtutorials.com/esp32-bluetooth-classic-arduino-ide/