DEVELOPMENT OF WEARABLE REHABILITATION DEVICE FOR WRIST-FINGER MOBILITY REHABILITATION

TAN CHIA WEN

UNIVERSITI TUNKU ABDUL RAHMAN

DEVELOPMENT OF WEARABLE REHABILITATION DEVICE FOR WRIST-FINGER MOBILITY REHABILITATION

TAN CHIA WEN

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Biomedical Engineering

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

September 2022

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : 

Name : Tan Chia Wen

ID No. : 18UEB07036

Date : 29 September 2022

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"DEVELOPMENT OF WEARABLE REHABILITATION DEVICE FOR WRIST-FINGER MOBILITY REHABILITATION"** was prepared by **TAN CHIA WEN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Biomedical Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : 

Supervisor : Mr Chong Yu Zheng

Date : 29 September 2022

Signature : 

Co-Supervisor : Dr Chan Siow Cheng

Date : 29 September 2022

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my research supervisor Mr Chong Yu Zheng and my co-supervisor, Dr Chan Siow Cheng for their continuous support and guidance throughout the entire development of the research.

Furthermore, I would like to extend my gratefulness to the founder of Malaysia Origami Academy, Mr Kenneth Ch'ng for inspiring me to incorporate origami theory into the design of the transmission mechanism for the rehabilitation system.

ABSTRACT

Many hand rehabilitation systems have only one preprogramed exercise protocol, do not measure the recovery progress, are costly, heavy and also do not have built-in safety mechanisms. Therefore, the aim of this study is to design and construct an affordable and light-weight hand rehabilitation exoskeleton system that could provide continuous passive movement to the finger and wrist joints, allow the patient to choose between different rehabilitation protocols and review their recovery progress. The rehabilitation system constructed incorporated ESP32, flex sensors, MG995 servo motors and android mobile application. Moreover, origami string theory and 3D printing technology was integrated into the transmission mechanism design. From the results obtained, the transmission mechanism can actuate MCP and PIP flexion and extension as well as radiocarpal extension movements that respect the static constraints of the hand and do not exceed the maximum angular velocities that can be naturally generated. In addition, the transmission mechanism was capable of actuating movements at 3 different angular velocities. Furthermore, the sensing system could measure maximum angle values that have an accuracy comparable to other studies except for 0° angles. Next, the total cost of the rehabilitation system was RM 533.70 and the segments attached to the hand weighed only 250 g. In conclusion, all the objectives were met. In the future, the transmission mechanism can be improved to generate more torque, formfitting gloves and goniometers can be used to increase the accuracy of the sensing system and a cloud database could be used to track recovery progress of patients.

TABLE OF CONTENTS

DECLARATION **i**

APPROVAL FOR SUBMISSION **ii**

ACKNOWLEDGEMENTS **iv**

ABSTRACT **v**

TABLE OF CONTENTS **vi**

LIST OF TABLES **xii**

LIST OF FIGURES **xiii**

LIST OF SYMBOLS / ABBREVIATIONS **xvii**

LIST OF APPENDICES **xix**

CHAPTER

1 INTRODUCTION

1.1 General Introduction 1

1.2 Importance of the Study 1

1.3 Problem Statement 1

1.3.1 Lack of Customizable Rehabilitation Mode 2

1.3.2 Lack of Recovery Progress Measurement 2

1.3.3 Costly and Heavy 2

1.3.4 Lack of Protection Against Unitended Motions 2

1.4 Aim and Objectives 2

1.5 Scope and Limitation of the Study 3

1.6 Contribution of the Study 3

1.7 Outline of Report 4

2 LITERATURE REVIEW **6**

2.1 Introduction 6

2.2 Stroke and its Prevalence 6

2.3 Post-stroke Effects 6

	2.3.1 Spasticity	7
2.4	Treatment for Spasticity	7
2.5	Anatomy of the Hand	7
2.6	Kinematics and Kinetics of the Hand	8
2.7	Hand Functional Requirements for Activities of Daily Living	10
2.8	Internet of Things Architecture	10
2.9	Market Research	11
2.10	Design Considerations	18
	2.10.1 Safety	18
	2.10.1.1 Static Constraints	18
	2.10.1.2 Dynamic Constraints	20
	2.10.2 Mobility	21
	2.10.3 Comfort	21
	2.10.4 Effective Force of Transmission	21
	2.10.5 Cost	22
	2.10.6 User-Friendliness	22
	2.10.7 Weight	22
	2.10.8 Adjustment to Different Hand Sizes	22
2.11	Exoskeleton System Design	23
	2.11.1 Transmission System	23
	2.11.2 Actuators	23
	2.11.3 Control System	25
	2.11.3.1 Operational Modes	25
	2.11.3.2 Microcontrollers	25
	2.11.4 Sensors	26
	2.11.5 Wireless Networks	27
	2.11.6 User Interface	28
2.12	Origami String as Transmission Mechanism	29
2.13	Motor Recovery Evaluation	32
2.14	Effectiveness of Origami Folding Activity as a Hand Rehabilitation Technique	32
2.15	Summary of Findings	33
3	**METHODOLOGY AND WORK PLAN**	**34**

3.1 Introduction 34
3.2 Specifications of Exoskeleton System 34
3.3 Work Plan 36
3.3.1 Work Breakdown Structure 38
3.3.2 Gantt Chart 40
3.4 Architecture of Hand Rehabilitation System 42
3.5 Anatomical Dimension Assumptions 43
3.5.1 Components Chosen for Each Element in the System Architecture 43
3.5.1.1 Power Source 43
3.5.1.2 Microcontroller 43
3.5.1.2.1 Programming ESP32 44
3.5.1.3 Sensors 44
3.5.1.4 Actuators 45
3.5.1.5 Transmission Mechanism 45
3.5.1.5.1 Mechanism Design Theory 45
3.5.1.5.2 Software to Design Mechanism 47
3.5.1.5.3 3D Printer 47
3.5.1.5.4 Filament 48
3.5.1.6 User Interface 48
3.6 Performance Testing and Results Analysis Methodology 48
3.6.1 Transmission System 48
3.6.1.1 Purpose 48
3.6.1.2 Steps 49
3.6.2 Sensing System 51
3.6.2.1 Purpose 51
3.6.2.2 Steps 51
3.6.3 User Interface 53
3.6.3.1 Purpose 53
3.6.3.2 Steps 54
3.6.4 Total Cost of Rehabilitation System 54
3.6.4.1 Purpose 54
3.6.4.2 Steps 54

3.6.5 Total Weight of Transmission Mechanism Attached to Hand 54
3.6.5.1 Purpose 54
3.6.5.2 Steps 54
3.7 Total Expenditure 54
3.8 Summary 55
4 REHABILITATION SYSTEM DESIGN AND CONSTRUCTION 57
4.1 Introduction 57
4.2 Electrical Circuit 57
4.3 Transmission System 58
4.3.1 Origami Theory 58
4.3.2 Final Hardware Design 59
4.3.3 Design Features 62
4.3.4 Code Design for Android Mobile Application Programmme 64
4.3.5 Code Design for ESP32 Programme 65
4.4 Sensing System 66
4.4.1 Voltage Divider Circuit 66
4.4.2 Code Design for Android Mobile Application Programmme 67
4.4.3 Code Design for ESP32 Programme 68
4.4.4 Calibration of Flex Sensors 68
4.5 Origami Tutorial Functionality 71
4.6 System Integration 71
4.6.1 Code Design for Android Mobile Application Programmme 71
4.6.2 Code Design for ESP32 Programme 72
4.8 Summary 72
5 RESULTS AND DISCUSSION 74
5.1 Introduction 74
5.2 Transmission System 74
5.2.1 Angles Generated 74
5.2.2 Angular Velocities Generated 76

5.3 Sensing System 79
5.4 User Interface 83
5.4.1 Activity Flow 83
5.4.1.1 Navigation 83
5.4.1.2 Hand Rehabilitation Activity 83
5.4.1.3 Recovery Progress Measurement Activity 86
5.4.1.4 Origami Tutorials Activity 88
5.5 Total Cost of Rehabilitation System 89
5.6 Total Weight of Transmission Mechanism Attached to Hand 89
5.7 Summary 89
6 CONCLUSIONS AND RECOMMENDATIONS 91
6.1 Conclusions 91
6.2 Recommendations for future work 92
REFERENCES 93

LIST OF TABLES

Table 1.1:	Outline of Report	4
Table 2.1:	Maximum Angular Velocity at the Finger Joints for Males and Females during Flexion and Extension Motions.	9
Table 2.2:	Peak Torque at the Finger Joints during Flexion and Extension Motions.	9
Table 2.3:	Maximum Angular Velocity at the Wrist Joint for Males and Females during Flexion, Extension, Abduction and Adduction Motions.	10
Table 2.4:	Peak Torque at the Wrist Joint for Males and Females during Flexion, Extension, Abduction and Adduction Motions.	10
Table 2.5:	Summary of Characteristics for 10 Exoskeleton Systems.	12
Table 2.6:	Maximum Angular Displacement for Wrist and Finger Joints.	19
Table 2.7:	Comparison between Linear DC Motor and Servo Motor.	24
Table 2.8:	Types of Sensors and their Common Functions in Exoskeletons..	26
Table 2.9:	Comparison between Flex Sensor and Rotary Position Sensor.	27
Table 2.10:	Comparison Between the 3 Types of User Interface.	28
Table 3.1:	Specifications for Exoskeleton System.	32
Table 3.2:	Function of Elements in Exoskeleton System.	40
Table 3.3:	Dimension Assumptions of the Hand.	41
Table 3.4:	Comparison between Compatible Microcontrollers for Arduino IoT Cloud.	42

Table 3.5:	Total Expenditure for the Construction of Rehabilitation System.	54
Table 3.6:	Elements in Rehabilitation System Architecture Design and Components Selected.	56
Table 4.1:	Activities Conducted to Select Rehabilitation Modes and their Functions.	65
Table 4.2:	Rehabilitation Mode Selected and Corresponding Data Received by ESP32.	65
Table 4.3:	Activities Conducted to Assess Recovery Progress.	67
Table 5.1:	Maximum Flexion / Extension Angles Generated for Three Different Rehabilitation Modes.	76
Table 5.2:	Maximum Flexion / Extension Angles Generated for Three Different Rehabilitation Modes.	78
Table 5.3:	Maximum MCP Flexion Angles and Maximum Radiocarpal Extension Angles Detected.	79
Table 5.4:	Mean Values and Mean Differences for Maximum MCP Flexion Angles and Maximum Radiocarpal Extension Angles Detected.	82

LIST OF FIGURES

Figure 2.1:	Joints and Bones in a Human Hand.	8
Figure 2.2:	Miura Vertex.	27
Figure 2.3:	(a) Parallel Configuration (b) Antiparallel Configuration	28
Figure 2.4:	Origami String Template with Three Vertices.	28
Figure 2.5:	Actuation Mechanism for each Finger on Gripper.	29
Figure 2.6:	Mechanical Stops on Hinges.	29
Figure 3.1:	Work Breakdown Structure for FYP Part 1.	36
Figure 3.2:	Work Breakdown Structure for FYP Part 2.	37
Figure 3.3:	Gantt Chart (Semester One).	38
Figure 3.4:	Gantt Chart (Semester Two).	39
Figure 3.5:	Architecture Design for Exoskeleton System.	40
Figure 3.6:	Crease Pattern and Dimensions for Transmission Mechanism (PIP and MCP joints).	44
Figure 3.7:	Transmission Mechanism in Antiparallel Configuration (PIP and MCP joints).	45
Figure 3.8:	Crease Pattern and Dimensions for Transmission Mechanism (Radiocarpal joint).	45
Figure 3.9:	Transmission Mechanism in Antiparallel Configuration (Radiocarpal Joint).	46
Figure 3.10:	Blue Coloured Backdrop.	49
Figure 3.11:	Ring Light and Stand Setup.	49
Figure 3.12:	Hand Attached with Transmission Mechanism and Markers.	50
Figure 3.13:	Sagittal View of the Hand.	50
Figure 3.14:	Trackers Attached Using Kinovea.	51

Figure 3.15: Angles Constructed on Paper. 52

Figure 3.16: Position of Middle Finger With 0° Flexion at MCP Joint. 52

Figure 3.17: Maximum MCP Flexion Angle Displayed on the User Interface. 53

Figure 4.1: Electrical Circuit Design for Rehabilitation Device. 57

Figure 4.2: Constructed Electrical Circuit for Rehabilitation Device. 58

Figure 4.3: Crease Pattern and Dimensions for Transmission Mechanism. 59

Figure 4.4: Transmission Mechanism in Antiparallel Configuration. 59

Figure 4.5 (a) & (b): Final Transmission Mechanism Design. 60

Figure 4.6: 3D-Printed and Assembled Transmission Mechanism. 60

Figure 4.7: Transmission Mechanism Attached to the Hand. 61

Figure 4.8: Transmission Mechanism Actuating MCP and PIP Joint Flexion and Radiocarpal Joint Extension. 61

Figure 4.9: (a) Initial Design of Loops that Connect Segments Diagonally. (b) Segments Cannot Bend Completely. (c) Modified Loops. (d) Segments Can Bend Completely. 63

Figure 4.10: Safeguard Plates Design Feature. 63

Figure 4.11: (a) Segments Lie Completely Flat. (b) Segments Do Not lie Completely Flat. 63

Figure 4.12: Elastic Bands (Circled in Red) and Extensions (Circled in Pink) Attached on Segments. 64

Figure 4.13: Different Activities Navigated During Selection of Rehabilitation Modes. 64

Figure 4.14: Voltage Divider Circuit for Sensing System. 66

Figure 4.15: Different Activities Navigated for Users to Assess Recovery Progress. 67

Figure 4.16: Hand Position Held for MCP Joint Flexion of 90°. 69

Figure 4.17: Iteration Values Shown on Serial Monitor 70

Figure 4.18: (a) Hand Position Held for Radiocarpal Joint Extension of 60°. (b) Hand Position Held for Radiocarpal Joint Extension and MCP Joint Flexion of 0°. 70

Figure 4.19: Complete Activity Flow in Android Application. 72

Figure 5.1: Angle vs Time graph for “Easy” Mode. 74

Figure 5.2: Angle vs Time graph for “Intermediate” Mode. 75

Figure 5.3: Angle vs Time graph for “Difficult” Mode. 75

Figure 5.4: Angular Velocity vs Time graph for “Easy” Mode. 77

Figure 5.5: Angular Velocity vs Time graph for “Intermediate” Mode. 77

Figure 5.6: Angular Velocity vs Time graph for “Difficult” Mode. 78

Figure 5.7: Boxplot: Maximum Angles Detected for MCP Joint Flexion. 80

Figure 5.8: Boxplot: Maximum Angles Detected for Radiocarpal Joint Extension. 81

Figure 5.9: (a) Launcher Icon (b) Navigation Page 83

Figure 5.10: (a) “Start Rehabilitation Activity” Button is Selected. (b) Prompt Displayed to Enable Bluetooth. (c) List of Paired Devices is Displayed. (d) Status Shown as Application is Attempting to Connect with ESP32. (e) Status Shown When Application Fails to Connect with ESP32. 84

Figure 5.11: (a) Three Different Rehabilitation Modes Displayed. (b) Page Shown When “Easy” Mode was Selected. (c) Page Shown When “Intermediate” Mode was Selected. (d) Page Shown When “Difficult” Mode was Selected. 85

Figure 5.12: (a) “Start Recovery Progress Measurement” Button is Selected. (b) Prompt Displayed to Enable Bluetooth. (c) List of Paired Devices is Displayed. (d) Status Shown as Application is Attempting to Connect with ESP32. (e) Status Shown When Application Fails to Connect with ESP32. 86

Figure 5.13: (a) Page Shown When the Sensing System is Measuring the Joint Angles. (b) Maximum Flexion Angle at MCP Joint and Maximum Extension Angle at Radiocarpal Joint Displayed on the Page. 88

Figure 5.14: (a) “Start Learning How to Fold Origami” Button is Selected. (b) Page Displaying a List of Origami Tutorials. (c) Tutorial Video Runs When It is Selected. 88

LIST OF SYMBOLS / ABBREVIATIONS

θ_{DIP}	flexion or extension angle for DIP joint, °
θ_{PIP}	flexion or extension angle for PIP joint, °
$\theta_{MCP_{Little}}$	flexion or extension angle for MCP joint of little finger, °
$\theta_{MCP_{Middle}}$	flexion or extension angle for MCP joint of middle finger, °
$\theta_{\mathrm{MCP_{Ring}}}$	flexion or extension angle for MCP joint of ring finger, °
C1, C2	two central spinal creases
P1, P2	two peripheral creases
α_1, α_2	angles between the peripheral creases and the central spinal creases, °
ϕ	angle offset for collinear crease *C1*, °

ADLs	activities of daily living
BLE	Bluetooth Low Energy
CMC	carpometacarpal
CPM	continuous passive movements
CRUD	create, read, update and delete
DALYs	disability-adjusted life years
DC	direct current
DIP	distal interphalangeal
DoF	degrees of freedom
EMG	electromyography
IDE	integrated development environment
IoT	Internet of Things
IP	interphalangeal
LCD	liquid crystal display
LPWAN	low-power wide area network
MCP	metacarpophalangeal
PIP	proximal interphalangeal
PWM	pulse-width modulation
ROM	range of motion
RTP	repetitive task practice

VR	virtual reality
WiFi	Wireless Fidelity
WLAN	wireless local area network
WMAN	wireless metropolitan area network
WPAN	wireless personal area network

LIST OF APPENDICES

Appendix A:	Android Application Code for “RehabConnect” Activity	98
Appendix B:	Android Application Code for “DevicesFragment” Activity	101
Appendix C:	Android Application Code for “RehabModesFragment” Activity	104
Appendix D:	ESP32 Code for Transmission System	111
Appendix E:	Android Application Code for “DevicesFragment2” Activity	115
Appendix F:	Android Application Code for “ProgressStartFragment” Activity	117
Appendix G:	ESP32 Code for Sensing System	124
Appendix H:	Ten Sets of Maximum Resistance Values for Sensing System Calibration	128
Appendix I:	Android Application Code for “OrigamiVideos” Activity	129
Appendix J:	Android Application Code for “HomeDirectory” Activity	131
Appendix K:	Complete Android Application Code.	132

CHAPTER 1

INTRODUCTION

1.1 General Introduction

With a lack of man power and the ongoing pandemic, it is important for post-stroke patients to utilise hand rehabilitative exoskeletons as a telerehabilitation alternative to physical physiotherapist sessions in order to carry out repetitive rehabilitation exercises.

This project aims to design a hand exoskeleton system that can provide continuous passive movement to the finger and wrist joints and allow the patient to choose between different rehabilitation protocols as well as review their recovery progress.

1.2 Importance of the Study

The results of this present study may provide insight in designing a hand exoskeleton rehabilitation system that is safe, cost-effective, light and allows users to interact with it via mobile phone. Moreover, this study will contribute to a better understanding on how to integrate origami string theory into designing transmission mechanisms.

1.3 Problem Statement

At present, there are many different hand rehabilitation exoskeleton systems that are in the market or under development. However, few of them generate movement at the wrist, have customisable rehabilitation modes or have user interfaces.

1.3.1 Lack of Customizable Rehabilitation Modes

Many hand exoskeleton systems can only perform one preprogrammed exercise protocol. This is not sufficient because post-stroke patients have varying severity of complications. For example, for patients with spasticity, their muscle tone increases with the increase in stretching velocity. Therefore, if they were to use hand exoskeleton that actuates high angular velocities, they may feel discomfort or pain.

1.3.2 Lack of Recovery Progress Measurement

Many hand exoskeletons can only generate movement of the finger and hand joints. They do not have sensors that measure the recovery progress of the patient. This would mean that the patients themselves would not be able to detect whether the rehabilitation exercises conducted by the hand rehabilitation exoskeleton system is effective.

1.3.3 Costly and Heavy

Most hand exoskeletons are priced around RM 3000. This may be unaffordable for patients to purchase for home rehabilitation. With physiotherapy sessions in Malaysia costing around RM 150 to RM 250 per session, RM 3000 can allow the patient to attend at least twelve sessions of physiotherapy. This means that it is still more cost effective to attend live physiotherapy sessions compared to purchasing a hand rehabilitation exoskeleton. Moreover, some hand exoskeletons found in the market survey are as heavy as 2.3 kg. With a weight this large, the hand will feel lethargic and uncomfortable after a while.

1.3.4 Lack of Protection Against Unintended Motions

Many exoskeletons like most tendon wire-based exoskeletons and some mechanical linkage exoskeletons do not have a built-in design their transmission mechanism to prevent actuating unintended motions. Over flexion or extension of finger and wrist joints exceeding their static constraints would injure and cause pain to the patient.

1.4 Aim and Objectives

The aim of this project was to design a hand rehabilitation exoskeleton system that could provide continuous passive movement to the finger and wrist joints and allow the patient to choose between different rehabilitation protocols as well as to review their recovery progress. The objectives were to:

- Design and construct a transmission mechanism that has 9 degrees of freedom and can generate extension movement at the radiocarpal joint

and coupled flexion and extension movements at the Metacarpophalangeal (MCP) and Proximal Interphalangeal (PIP) joints of the 4 fingers (index, middle, ring and little fingers).

- Design and construct a transmission mechanism that actuates movements that respect the static constraints of the hand and does not exceed the maximum angular velocities that can be generated by the hand naturally.
- Design and construct a sensing system that can measure the maximum angle of flexion at the MCP joint and maximum angle of extension at the wrist joint for recovery progress measurement.
- Design and construct a hand exoskeleton rehabilitation system that cost less than RM 1500 with the sections attached to the hand weighing less than 500 g.
- Design and construct a user interface that allows the patient to choose between 3 levels of angular velocity that is generated by the transmission mechanism and allows the patient view their recovery progress.

1.5 Scope and Limitation of the Study

The scope of this study was to design and construct a hand rehabilitation exoskeleton system that can generate extension movement at the radiocarpal joint and coupled flexion and extension movements at the MCP and PIP joints of the 4 fingers (index, middle, ring and little fingers), measure the maximum angle of flexion at the MCP joint and maximum angle of extension at the wrist joint, allows the patient to choose between 3 levels of angular velocity that is generated by the transmission mechanism and allows the patient view their recovery progress.

This study had to be completed within 8 months. As such, due to time limitation, the hand rehabilitation exoskeleton system was not tested on subjects. Therefore, the efficacy of the rehabilitation system in improving the finger and wrist movements of post-stroke patients are not known.

1.6 Contribution of the Study

This study will design and construct a hand exoskeleton rehabilitation system

that is safe, cost-effective, light and allows users to interact with it via mobile phone. Moreover, this study will integrate origami string theory into the transmission mechanism design for the hand rehabilitation system.

1.7 Outline of the Report

The content of the report was distributed as shown in Table 1.1.

Table 1.1: Outline of Report

Chapter	Content
1	This chapter explains the importance of this study, identifies the problem statements and objectives of the study, states the scope and limitations of the project and highlights the importance and contributions of the study.
2	This chapter contains the literature review on the following topics: the effects of stoke and its prevalence, the anatomy, kinematics and kinetics of the hand, Internet of Things architecture, market research, design considerations, exoskeleton system design, origami string theory, motor recovery evaluation and the effectiveness of folding origami for hand rehabilitation.
3	This chapter contains the specification for rehabilitation system, system architecture design and materials chosen, total expenditure, work plan, Work Breakdown Structure, Gantt Chart and methods used to test and analyse the performance of the rehabilitation system.
4	This chapter covers the design and construction of the following systems: electrical circuit, transmission mechanism, sensing system, and origami tutorial functionality. In addition, it also covers the method used to integrate these systems together.
5	This chapter displays and evaluates the performance of the following areas: transmission system, sensing system, user interface, total cost of rehabilitation system and total weight of transmission mechanism attached to the hand.
6	This chapter covers the conclusion of the entire study as well as

	possible future work that can be done.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

A literature review was conducted to study the prevalence and effects of stroke, the properties of the human hand, the architecture of Internet of Things, conduct market research on hand exoskeletons as well as determine the design considerations and components required. The results of this review will be used to set the specifications of the hand exoskeleton as well as select suitable technology for the design of the exoskeleton system.

2.2 Stroke and its Prevalence

Stroke is a cerebrovascular disease that hinders the provision of oxygen and nutrients to the brain cells thus causing the demise of these cells.

According to the World Stroke Organization, there are 12.2 million new strokes cases annually, with 101 million stroke survivors living with complications on a global scale. Stroke has the third highest disability-adjusted life years (DALYs) which indicates that this lifestyle disease places a huge burden on society, with 89 % of this burden concentrated in low- and middle-income countries (Feigin et al., 2022). World Health Organization explains that DALYs is the aggregation of years forfeited by premature demise and the years existing in suboptimal health (World Health Organization, 2013). This shows that there is a huge market for rehabilitation devices that target post-stroke patients.

2.3 Post-stroke Effects

They can be categorized into a few types: physical symptoms, cognitive symptoms and emotional symptoms. Some of the physical symptoms are loss in stamina, drop foot, loss in muscle strength, spasticity, contractures and sensory alterations (Stroke Association, 2013).

2.3.1 Spasticity

Spasticity occurs in around 30 % to 80 % of post-stroke patients. 66 % of patients suffering from spasticity have it affect their flexor muscles in their wrist and fingers (Kuo and Hu, 2018). Spasticity is a velocity-dependent increase in muscle tone that causes the muscles to become stiff, in constant contraction and resistive to stretching. When spasticity occurs in the flexor muscles of the wrist and fingers, it will cause a decrease in range of motion on the finger and wrist joints as well as evoke pain to the patient. (American Stroke Association, 2019) This impairment will restrict the patient from carrying out their daily activities such as grabbing objects and pinching. This lack of autonomy will reduce their quality of life because they will have be dependent on their caregivers to carry out basic tasks or even be placed in care facilities.

2.4 Treatment for Spasticity

Treatment for spasticity include both pharmacological and physical management techniques. Physical management techniques can be divided into 5 categories: active rehabilitation, passive rehabilitation / stretching, positioning, standing and supplementary exercises. Active rehabilitation exercises include: progressive resistance exercises to increase the strength of the affected limb, neuromuscular electrical stimulation to activate affected muscles, and using electromyography-driven robotics to provide active-assisted rehabilitation. On the other hand, passive rehabilitation exercises include: passive stretching of joints to increase the mobility of the joints, using continuous motion robots that provide continuous passive movements (CPM) to the affected limb, and wearing orthotics to prevent the affected muscle from contracting (Monaghan et al., 2011).

2.5 Anatomy of the Hand

A human hand consists of five fingers, a palm and a wrist. The little, ring, middle and index fingers each contain three phalanges: distal, intermediate and proximal phalanges. In addition, these four fingers each contain three joints: distal interphalangeal (DIP), proximal interphalangeal (PIP), and metacarpophalangeal (MCP) joints. On the other hand, the thumb contains

only two phalanges: distal and proximal phalanges. The thumb has three joints: interphalangeal (IP), metacarpophalangeal (MCP) and carpometacarpal (CMC) joints.

The palm contains five metacarpal bones while the wrist contains of eight bones: scaphoid, triquetrum, lunate, capitate, trapezoid, hamate, trapezium and pisiform. Furthermore, the wrist has four joints: ulnocarpal, distal radioulnar, radiocarpal and scaphotrapeziotrapezoid joints (American Society for Surgery of the Hand, 2022).

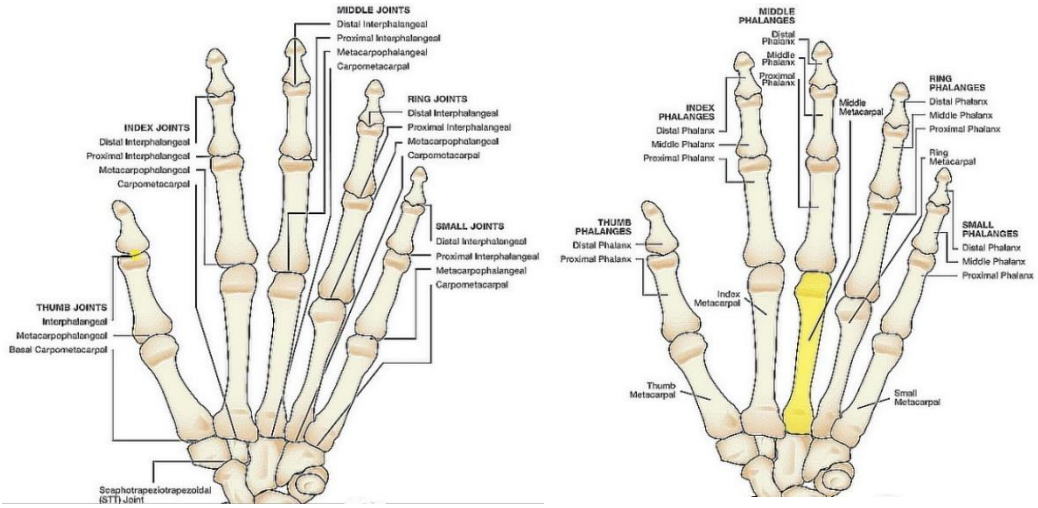


Figure 2.1: Joints and Bones in a Human Hand (American Society for Surgery of the Hand, 2022).

2.6 Kinematics and Kinetics of the Hand

In total, there are twenty-seven degrees of freedom (DoF) in a human hand. One DoF means that the joint can move about in one axis of rotation. The little, ring, middle and index fingers each contain four DoF: one DoF at the DIP joint (motions: flexion / extension), one DoF at the PIP joint (motions: flexion / extension), and two DoF at the MCP joint (motions: flexion / extension and adduction / abduction). The thumb contains five DoF: one DoF at the IP joint (motions: flexion / extension), two DoF at the MCP joint (motions: flexion / extension and adduction / abduction), and two DoF at the CMC joint (motions: flexion / extension and adduction / abduction). The palm contains six DoF:

three DoF for translation and three DoF for rotation motions. (Rahman and Al-Jumaily, 2013)

The twenty-seven DoF human hand model does not contain DoF for wrist. Therefore, besides finger and palm movements, the wrist contains three DoF: two DoF at radiocarpal joint (motions: flexion / extension and abduction / adduction) and one DoF for rotation motion. (Palmer et al., 1985)

Referring to the results obtained from Chen Chen et al. (2013a), the maximum angular velocities of the finger joints of healthy subjects during flexion and extension can be seen at Table 2.1.

Table 2.1: Maximum Angular Velocity at the Finger Joints for Males and Females during Flexion and Extension Motions.

Joint	**Maximum Angular Velocity (°/s)**	
	Males	**Females**
DIP	574	572
PIP	861	858
MCP	697	694

Referring to the results obtained from Chen et al. (2011), the peak torque of the finger joints of healthy subjects during flexion and extension can be seen at Table 2.2.

Table 2.2: Peak Torque at the Finger Joints during Flexion and Extension Motions.

Joint	**Peak Torque (Nm)**			
	Little	**Ring**	**Middle**	**Index**
DIP	0.398	0.550	0.850	0.775
PIP	1.200	1.800	2.890	2.280
MCP	N/A	3.700	5.000	4.630

Referring to the results obtained from Xia and Frey-Law (2015), the maximum angular velocities and peak torque of the wrist joint of healthy subjects can be seen at Table 2.3 and Table 2.4 respectively.

Table 2.3: Maximum Angular Velocity at the Wrist Joint for Males and Females during Flexion, Extension, Abduction and Adduction Motions.

Type of Motion	Maximum Angular Velocity (°/s)	
	Males	Females
Flexion	240	240
Extension	240	180
Abduction	180	120
Adduction	180	180

Table 2.4: Peak Torque at the Wrist Joint for Males and Females during Flexion, Extension, Abduction and Adduction Motions.

Type of Motion	Peak Torque (Nm)	
	Males	Females
Flexion	17.5	11.2
Extension	10.9	7.0
Abduction	14.3	8.8
Adduction	13.5	8.8

2.7 Hand Functional Requirements for Activities of Daily Living

Activities of Daily Living (ADLs) are the everyday tasks that are independently carried out by healthy individuals to maintain a good quality of life. To carry out most ADLs, a person's hand need to at least have the ability to operate three fingers and each of these fingers need to at least have two degrees of freedom (DoF); one DoF at the PIP joint and one DoF the MCP joint to generate flexion and extension movement. This is because abduction and adduction at the MCP is not essential for the performance of ADL. In addition, since movement of the DIP joint is dependent on the movement of the PIP joint, the DIP joint is usually not targeted in ADLs training as well. (Sarac, Solazzi and Frisoli, 2019)

Referring to the minimum requirements needed to perform ADLs, we can set the minimum functional requirements for a hand exoskeleton rehabilitation device to be: able to generate flexion and extension at the MCP and PIP joints of at least three fingers.

2.8 Internet of Things Architecture

Internet of Things (IoT) is the interconnection of devices (e.g., sensors and personal servers) that can communicate with each another through via the

Internet. There are four layers in the IoT architecture: sensing, network, support and application layers.

The first layer is the sensing layer. This layer consists of different types of sensors (e.g., flex sensors, thermal sensors, and glucometers). Its function is to collected data (e.g., angular displacement, temperature, and blood oxygen saturation) from the environment and conduct some minor signal processing.

The second layer is the network layer. This layer functions to transmit data sent by components from the sensing layer to the support layer for processing and storage. There are two types of networks: wired and wireless. Wired networks require data to be sent to the Internet via cables. Some wired networks include: Ethernet, Meter Bus, and Power Line Communication. Some wireless networks include: Bluetooth, Wireless Fidelity (WiFi), and ZigBee.

The third layer is the support layer. This layer is used to process, analyse, store and retrieve data sent from the network layer. This layer also handles the security and maintenance of the system. Some components in this layer include: databases, authentication systems, and data processing centres.

The fourth layer is the application layer. This layer provides human interaction with the system and also executes orders provided by the support layer. For example, this layer can allow the users to select different modes of operation as well as display statistics to the users through interfaces like web applications and mobile applications. In addition, this layer can also control actuators to perform a function (e.g., turn on a nightlight) as dictated by the support layer.

2.9 Market Research

In order to get a better view on the aspects that should be considered when designing a hand rehabilitation exoskeleton, ten exoskeleton systems were analysed and summarised in Table 2.5.

Table 2.5: Summary of Characteristics for 10 Exoskeleton Systems.

Paper	**Weight**	**Body Structures Interacted**	**Hardware**	**Command method**	**Transmission mechanism**	**Operational Modes**	**Network**	**Patient Interface**	**Force created**
(Yang et al., 2021)	150 g	5 fingers, 10 DoF	▪ Actuator: 5 Linear DC motors ▪ Microcontroller: Arduino MEGA2560 ▪ Sensors: None	Bilateral hand training	• Type: Sliding springs • Placement: dorsal	Continuous passive movement	Bluetooth	VR game on phone / computer	10 N
(Rahman and Al-Jumaily, 2013)	1.8 kg	5 fingers, 15 DoF	▪ Actuator: 5 Linear DC motors ▪ Microcontroller: ATmega 328 ▪ Sensors: None	Bilateral hand training	• Type: Mechanical linkages • Placement: dorsal	Continuous passive movement	Zigbee	None	-
(Ahmed et al.,	280 g	5 fingers, 14 DoF	▪ Actuator: 14 Rotational DC	Pre-set programme	• Type: Tendon wires	Continuous passive	Wired	None	-

2021)			motors ▪ Microcontroller: Arduino Sketch running on personal computer ▪ Sensors: None		• Placement: dorsal and palmar	movement			
(Yurkewich et al., 2020)	284 g	5 fingers, -	▪ Actuator: 2 Linear DC motors ▪ Microcontroller: tinyTILE by Intel Curie ▪ Sensors: Gyroscope	Grasp intent detection	• Type: Tendon wires • Placement: dorsal and palmar	Active-assisted movement	-	None	16 N
(Decker and Kim, 2017)	719 g	5 fingers, 12 DoF	▪ Actuator: 5 Rotational DC motors ▪ Microcontroller: - ▪ Sensors: Flex	• Pre-set programme • Grasp intent detection	• Type: Mechanical linkages • Placement: dorsal	• Continuous passive movement • Active-assisted	-	VR	14 N

			sensor and Inertial Measurement Unit			movement • Haptic interaction			
(Jo et al., 2019)	156 g	4 fingers (little, ring, middle, index), coupled, 8 DoF	▪ Actuator: 1 Linear DC motor ▪ Microcontroller: - Sensors: Potentiometer	Pre-set programme	• Type: Mechanical Linkages with Spring Guidance • Placement: dorsal	Passive movement	-	None	-
(Ates, Haarman and Stienen, 2017)	650 g	5 fingers and wrist, 14 DoF, only extension	▪ Actuator: 1 electric motor ▪ Microcontroller: - ▪ Sensors: Rotary position sensors and flex sensors	-	• Type: Spring • Placement: dorsal	• Continuous passive movement • Active-assisted movement	-	-	-

(Yap et al., 2016)	150 g	5 fingers, 14 DoF, only extension	▪ Actuator: 1 pneumatic actuator ▪ Microcontroller: Arduino Mega ▪ Sensors: Air pressure sensor	Pre-set programme	• Type: Pneumatic • Placement: dorsal	Continuous passive movement	None	None	-
(Kang et al., 2019)	104 g	2 fingers (index and middle), 6 DoF	▪ Actuator: 1 dual-slack enabling actuator ▪ Microcontroller: Custom electric board (TMS320F2808) ▪ Sensors: None	Button intention detection	• Type: Tendon wires • Placement: dorsal and palmar	Continuous passive movement	None	None	-
(Singh et al., 2019)	2.3 kg	4 fingers (little, ring, middle, index) and	▪ Actuator: 1 rotational DC motor ▪ Microcontroller:	Pre-set programmes with customizable	• Type: Mechanical Linkages with Spring	• Continuous passive movement • Active-	Wired	LCD display with buttons	-

		wrist, 5 DoF	ATmega328 ▪ Sensors: Potentiometers and EMG units	parameters	Guidance • Placement: dorsal	assisted movement		for mode selections	

From Table 2.5, we can see that the weight of the wearable exoskeleton ranges from 104 g to 2.3 g. Some of the exoskeletons are also lighter because their actuators, control systems and power sources are placed remotely. Since they are not attached to the arm or hand, these masses are not factored into the total weight.

In addition, the body structures that the exoskeleton interacts with range from two fingers (index and middle fingers) to all five fingers. Only two out of the ten exoskeletons reviewed interact with the wrist. Most exoskeletons here provide flexion and extension motions of joints while two of them provide only extension.

Other than that, the exoskeletons reviewed have different operational modes. Some exoskeletons provide the patients with the option of multiple operation modes. Most of them operate on continuous passive movement mode (CPM). A few operated on active-assisted movement mode and only Decker and Kim (2017) operated on haptic interaction mode.

Furthermore, methods used to dictate the movements of the exoskeletons include: bilateral hand training (where the exoskeleton copies the movement of the healthy hand), pre-set programme (common for those using CPM), grasp intent detection (common for those using active-assisted movement mode) and button intention detection (where the patient uses their healthy hand to press a button when they want the exoskeleton to perform flexion).

In addition, the exoskeletons implement various transmission mechanisms such as: tendon wires, sliding springs, mechanical linkages and pneumatics. The placement of these transmission mechanisms are either dorsal or palmar or both.

Moreover, most exoskeletons reviewed used either linear or rotational DC motors as actuators. Other than DC motors, pneumatic pumps (for pneumatic transmission systems) and dual-slack enabling actuators (for tendon driven transmission system) were also utilised. Most exoskeleton designs try to have less actuators because they are expensive and if attached directly to the exoskeleton itself, they will contribute to the weight of the exoskeleton. Sensors used by the exoskeletons include: flex sensors,

gyroscopes, inertial measurement units, potentiometers, air pressure sensors, Electromyography (EMG) units, and rotary position sensors.

A few exoskeleton systems have patient interfaces in the form of a Virtual Reality games or just a Liquid Crystal Display (LCD) display that allows patient to select rehabilitation parameters by selecting buttons. The types of networks that allow data to be transmitted between the exoskeleton and the patient interface are: wired or wireless (Bluetooth and Zigbee).

Only three papers stated the force that can be generated by the exoskeleton: 10 N, 14 N and 16 N. This gives us the range of forces that we should achieve from our own exoskeleton design.

Some of the terminology used above will be elaborated in section 2.11.

2.10 Design Considerations

It is important to consider these following aspects before the specifications and design of the exoskeleton is set.

2.10.1 Safety

Safety is the most important aspect that should be considered when designing an exoskeleton system. The design of the transmission mechanism and control programme need to consider the static and dynamic constraints of the human hand. The exoskeleton device also needs to be electrically safe.

2.10.1.1 Static Constraints

Static constraints are limitations imposed on the movement of joints (Rahman and Al-Jumaily, 2013). Table 2.6 shows the maximum angular displacement that can occur for different joints in the fingers and wrist.

Table 2.6: Maximum Angular Displacement for Wrist and Finger Joints. (Chen Chen et al., 2013b) and (Palmer et al., 1985).

Joints	**Flexion (°)**	**Extension (°)**	**Abduction / Adduction (°)**
Little Finger			
DIP	90	5	0
PIP	135	0	0
MCP	90	30 – 40	50
Ring Finger			
DIP	80 - 90	5	0
PIP	120	0	0
MCP	90	30 – 40	45
Middle Finger			
DIP	80 – 90	5	0
PIP	110	0	0
MCP	90	30 – 40	45
Index Finger			
DIP	80 – 90	5	0
PIP	110	0	0
MCP	90	30 – 40	60
Thumb			
IP	75 – 80	5 – 10	5
MCP	75 – 80	0	5
Wrist			
Radiocarpal	78	60	21 – 38

Referring to Table 2.6, we need to ensure that the exoskeleton designed does not generate motions that exceeds these static constraints.

2.10.1.2 Dynamic Constraints

Dynamic constraints are limitations imposed on the movement of finger joints when the finger is moving (Rahman and Al-Jumaily, 2013). Dynamic constraint can be divided into intrafinger and interfinger constraints.

Intrafinger constraints are limitation of joints imposed by the movement of other joints that are located on the same finger. Intrafinger constraints are listed below:

Equation 2.1 shows the intrafinger constraints on the index, middle, ring and little fingers (Chen Chen et al., 2013b).

$$\theta_{DIP} \approx \frac{2}{3}\theta_{PIP} \tag{2.1}$$

where

$\theta_{\mathrm{DIP}} = \text{flexion or extension angle for DIP joint}$

$\theta_{\mathrm{PIP}} = \text{flexion or extension angle for PIP joint}$

Interfinger constraints are limitations on joints due to correlation of joint motion between joints from different fingers. Interfinger constraints are listed below:

Equation 2.2 shows that when there is flexion or extension of the ring finger at MCP joint, the MCP joints at middle finger and little finger will also flex or extend to the same degree (Chen Chen et al., 2013b).

$$\theta_{MCP_{Ring}} \approx \theta_{MCP_{Middle}} \approx \theta_{MCP_{Little}} \tag{2.2}$$

where

$\theta_{\mathrm{MCP_{Little}}} = \text{flexion or extension angle for MCP joint of little finger}$

$\theta_{\mathrm{MCP_{Middle}}} = \text{flexion or extension angle for MCP joint of middle finger}$

$\theta_{\mathrm{MCP_{Ring}}} = \text{flexion or extension angle for MCP joint of ring finger}$

The exoskeleton design should follow these dynamic constraints to ensure that the movement generated on the hand follow the natural hand

motions. This will help to reduce the patient's discomfort or fatigue when using the exoskeleton.

2.10.2 Mobility

Referring to the hand anatomy, we need to determine which fingers and what joints on each finger that we want the exoskeleton to interact with. Furthermore, we also need to determine whether we want to include wrist interaction. We should also determine whether we want to control each finger / wrist individually or couple the movements together (Sarac, Solazzi and Frisoli, 2019).

2.10.3 Comfort

We need to ensure that patients are comfortable during the usage of the exoskeleton because rehabilitation sessions have long durations. One method to ensure comfort is to evaluate whether the parts of the exoskeleton that interacts physically with any part of the patient's body does not cause the patient any pain. We should also ensure that the static and dynamic constraints of the hand adhered to ensure the movement generated by the exoskeleton feel natural (Sarac, Solazzi and Frisoli, 2019).

2.10.4 Effective Force of Transmission

We need to ensure that the forces generated by the transmission system of the exoskeleton are sufficient to generate torque at the joints. From the market survey, we gather that the target forces that should be generated is in the range of 10 N to 16 N. However, we should also keep in mind that patients with spasticity have stiffer joints due to an increase in muscle tone. They may require a larger force as compared to healthy subjects. Moreover, forces applied need to be perpendicular to the bones to ensure that the connectors of the exoskeleton do not slip off the hand during actuation (Sarac, Solazzi and Frisoli, 2019).

2.10.5 Cost

Since the goal is to have this exoskeleton purchased by the patient for personal use at home, it will have to be affordable. The cost of the

exoskeleton includes the cost of components, the manufacturing cost and the cost to modify the exoskeleton design to fit different hand sizes (Sarac, Solazzi and Frisoli, 2019).

From the market research done, the cost of the exoskeleton should be in the range of 300 to 500 USD.

2.10.6 User-Friendliness

The goal is to ensure the exoskeleton can be worn and operated independently by the patients themselves with minimal training required.

One criterion is that the patient must be able to utilise their less impaired hand to wear and take off the exoskeleton from the hand with spasticity. As such, they should be able to do so within 5 minutes (Ates, Haarman and Stienen, 2017).

In addition, since 86 % of stroke cases occur in people who are 50 years old or older, the user interface design must be intuitive and easy to understand. Words displayed must be easily readable and the design of the user interface must be simple and direct.

2.10.7 Weight

The weight of the exoskeleton that is worn on the hand and arm must be light (within 500 g) to ensure the it is portable without causing arm fatigue to the patient (Ates, Haarman and Stienen, 2017).

2.10.8 Adjustment to Different Hand Sizes

People have different hand sizes due to their age, height and sex. Therefore, strategies should be created on how to customize the exoskeleton to function effectively for patients of various hand sizes.

One such strategy is to scale and manufacture each exoskeleton individually according to the hand dimensions of the patient. Another method is to have a technician manually adjust the mechanical connects to align with the finger joints before operating the exoskeleton. Another strategy is to predesign and manufacture a set range of sizes. The range of sizes should be small enough to still allow mass production but large enough to ensure that accommodate most hand sizes (Sarac, Solazzi and Frisoli, 2019).

2.11 Exoskeleton System Design

There are many components in an exoskeleton system. The subsections below will review these components.

2.11.1 Transmission System

Transmission systems transform forces generated by actuators into movement at the patient's joints.

These transmission system units can be planted on either on the palmar, lateral or dorsal sides of the hand. Palmar placements mean that the transmission components are installed on the palm of the hand. (Sarac, Solazzi and Frisoli, 2019) This placement is almost never used for mechanical linkages or pneumatic systems because the bulky components would obstruct the movement of the finger joints when flexion takes place.

Lateral placements mean that the transmission components are placed at the left and right sides of the fingers as well as the wrist. However, this placement is not suited for bulky units found in mechanical linkages and is also prone to movement collisions between different fingers due to the adduction and abduction of the MCP joints. (Sarac, Solazzi and Frisoli, 2019)

Lastly, dorsal placements mean that the transmission components are placed on top of the fingers and wrist. This placement is most common especially for mechanical linkage mechanism because it will not obstruct the movement actuated by the exoskeleton, there are less collisions between different fingers and the palm is bare and can interact with real objects. (Sarac, Solazzi and Frisoli, 2019)

2.11.2 Actuators

Actuators are used to generate forces on to the transmission mechanisms. There are four types of actuators: direct current (DC) motors, servo motors, ultrasonic and pneumatic actuators.

DC motors are the most widely used type of actuator because they are easily sourced, cheap, reliable and can be easily controlled. DC motors can be further divided into linear DC motors and rotational DC motors. Linear DC motors generate forces that produces linear motions. It is suitable

for mechanical linkage or spring transmission systems that require linear forces. Rotational DC motors generate forces that produces rotational motions. It is suitable for tendon wire transmission systems because rotational DC motors can coil and uncoil the wires. (Sarac, Solazzi and Frisoli, 2019)

Servo motors are used to generate specific angular displacements. These motors can provide accurate positioning, and generate high output torque. However, they are costlier compared to DC motors. (Sarac, Solazzi and Frisoli, 2019)

Ultrasonic motors also generate rotational motions using ultrasonic vibrations. These motors are light and silent but are prone to temperature increase and hysteresis after a certain operational period. (Sarac, Solazzi and Frisoli, 2019)

Pneumatic actuators utilise a combination of pneumatic pumps and valves to control the air pressure that is sent to inflate or deflate the components in the transmission system. These motors can produce adjustable forces and speed easily but they need to be attached remotely because they are large in size and weight. (Sarac, Solazzi and Frisoli, 2019)

Based on the analysis above, servo motors and linear DC motors are suitable for this project. Table 2.7 makes a comparison between these 2 actuators to determine which one is more suitable.

Table 2.7: Comparison between Linear DC Motor and Servo Motor.

Specification	**Linear DC motor**	**Servo motor**
Cost	RM 65.10	RM 14.90
Weight	Heavier	Lighter
Force	Larger force.	Smaller force (maximum torque for MG995: 10 kgfcm)
Attachment	• Linear DC motors have long dimensions which will take up too much space on the arm.	• Servo motors have smaller dimensions.

Referring to Table 2.7, the servo motor is selected because it is cheaper, lighter and would not take up too much space.

2.11.3 Control System

2.11.3.1 Operational Modes

There are three different operational modes for rehabilitation exoskeletons: Continuous Passive Movement (CPM), Active-assisted Movement and Active-resisted Movement.

Exoskeletons applying CPM will passively move the joints on the hand without any assistance from the muscles in the hand. The patient does not exert any force during the entire rehabilitation exercise. This is used to the replace repetitive task practice (RTP) performed by therapists in rehabilitation sessions. CPM has the ability to restore the patient's range of motion and is most effective when each session conducted over a long duration of around 45 minutes. (Ahmed et al., 2021)

When patients use exoskeletons applying active-assisted movement, patients have to use their muscles to contribute some force in rehabilitation exercises.

Meanwhile, when patients use exoskeletons applying active-resisted movement, patients have to use their muscles to apply forces larger than what they would normally exert without the exoskeletons. These exoskeletons apply forces that oppose that generated by the patient's muscles. (Sarac, Solazzi and Frisoli, 2019)

2.11.3.2 Microcontrollers

The function of a microcontroller is to control the actuators, receive data from sensors and send as well as receive data to and from a user interface. Some microcontroller specifications that should be considered during selection process are: compatible programming languages, number of input and output pins, input power and additional integrated modules.

2.11.4 Sensors

There are many types of sensors used in the exoskeleton in order to achieve different functionalities. Table 2.8 elaborates the types of sensors as well as its common functions.

Table 2.8: Types of Sensors and their Common Functions in Exoskeletons. (Tiboni et al., 2022)

Type of Sensors	Common Functions
Bending / Flex Sensors	• Measure the orientation or magnitude of bending force generated by transmission system. • Measure the angular displacement of a joint.
Potentiometers / Rotary Position Sensors	• Measure angular displacement of joint.
Pressure Sensors	• Measure air pressure in pneumatic transmission systems.
Force Sensors	• Measure force exerted by the patient or force exerted by transmission system.
Torque Sensors	• Measure torque generated by transmission system.
Gyroscope	• Measure angular velocity generated by transmission system.
Inertial Measurement Units (IMUs)	• Measure the orientation of the phalanges. • Measure angular velocity generated by transmission system.
Electromyography (EMG) Sensors	• Measure EMG signals to anticipate movement from patient.

Based on Table 2.8, both flex sensors and the rotary position sensors are suitable because they can detect the range of motion of the joint by varying their resistance. Table 2.9 makes a comparison between these two sensors to determine which one is more suitable.

Table 2.9: Comparison between Flex Sensor and Rotary Position Sensor.

Specification	**Flex sensor**	**Rotary position sensor**
Cost	RM 49	RM 15
Setup	• Simpler setup. • The flex sensor just has to be attached to the glove and connected to the microcontroller.	• Complicated setup. • Requires a lever system to be attached to the gloves. Then the lever system needs to rotate the shaft of the sensor.
Attachment	• Can be attached at the dorsal part of the fingers and wrist.	• Will have to be attached to the lateral sides of the fingers and wrist because that will where the lever system will be installed. • This bulky system may hinder the movement of the fingers.

Even though the flex sensor is more expensive, it is selected for the exoskeleton system because of the ease of setup and its attachment would not disturb the movement of the fingers.

2.11.5 Wireless Networks

Wireless Network technology is often used to connect the microcontroller to the user interface. There are four main types of wireless technologies.

Low-power Wide Area Network (LPWAN) can be ruled out as potential wireless technology for this project because it is not widely supported which would make installation more complex and not compatible with many devices. Since the microprocessor on the exoskeleton will be transmitting data over a short distance, Wireless Metropolitan Area Network (WMAN) is also unsuitable because it is meant to transfer data over a longer distance. It would be costly to use WMAN.

Two possible technologies that can be considered are Wireless Local Area Network (WLAN) and Wireless Personal Area Network (WPAN) because they are both relatively low cost, interoperable with suitable

coverage range. However, WPAN would be more suitable due to having lower power consumption (exoskeleton can operate longer before bring charged), cheaper and has less signal interference. Even though WPAN has a shorter range of coverage and data transfer rate, it will not interfere with the functionality of the exoskeleton system because the exoskeleton aims to transfer a small amount of data infrequently through a short distance. (Rackley, 2011)

2.11.6 User Interface

User interface is the point where the patient can interact with the exoskeleton system. These interfaces are usually the applications on the patient's phones, laptops or custom-LCD screen displays with button selections. These interfaces will run the application layer of the exoskeleton IoT system.

There are 3 possible types of user interface that can be constructed: Android mobile application, web application or Arduino IoT Cloud. Table 2.10 compares the attributes of these 3 types of user interface.

Table 2.10: Comparison Between the 3 Types of User Interface.

Attribute	**Android Mobile Application**	**Web Application**	**Arduino IoT Cloud**
Type of Network Layer with ESP32	Bluetooth (More reliable and does not require a router)	Wi-Fi (Less reliable and requires a router)	Wi-Fi (Less reliable and requires a router)
Construction of Network Layer	Difficult to construct network.	Difficult to construct network.	Easy to connect to ESP32. Just have to link the ESP32 to a "Thing".
Integrated Development Environment	Android Studio	Web Application Builder	Arduino Web Editor
Interface Construction	Difficult to build. Require	Difficult to build. Require	Easy to build built by selecting the widgets

	knowledge of JavaScript programming language.	knowledge of JavaScript programming language.	and dropping it into the display area.
Interface Design	Customizable and can embed origami video tutorials.	Customizable and can embed origami video tutorials.	Not customizable and cannot embed origami video tutorials.

Referring to the comparisons made in Table 2.10, Android mobile application was chosen as the user interface type. This is because it can create a more reliable network with the ESP32 and does not require a router. Even though it is more difficult to construct, it has a customizable interface design which can allow the integration of origami video tutorials.

2.12 Origami String as Transmission Mechanism

Origami-influenced engineering designs have been implemented in many different applications such as: antenna deployment, air bag systems, stent grafts and drug delivery systems.

One keen interest of this project is to investigate the viability of designing a transmission mechanism inspired by origami theory. This is conducted by reviewing the origami claw gripper designed by Liu et al. (2021). In this paper, they incorporate the theory of the Miura vertex and origami string into designing the fingers for the grippers.

The basic component that makes up each foldable finger is called the Miura vertex. A vertex is the point where two or more lines or creases meet. As seen in Figure 2.2, the Miura vertex consists of four creases: two central spinal creases (C1, C2) and two peripheral creases (P1, P2). The angles between the peripheral creases and the central spinal creases are α_1, α_2. The lengths of the central spinal creases can be varied to alter the location of the vertex. When the lengths of the peripheral creases change, the angles α_1, α_2 will also change. (Liu et al., 2021)

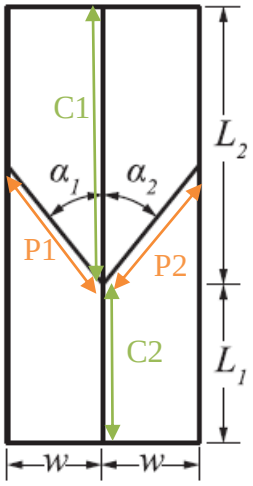


Figure 2.2: Miura Vertex. (Liu et al., 2021).

Each Miura vertex has two configurations: parallel configuration (when the central spinal creases, C1 and C2 are collinear) and antiparallel configuration (when the central spinal creases are not collinear). From Figure 2.3, we can see that the angle offset for collinear crease C1 is ϕ. When the Miura vertex is in antiparallel configuration, angle ϕ is more than 0°.

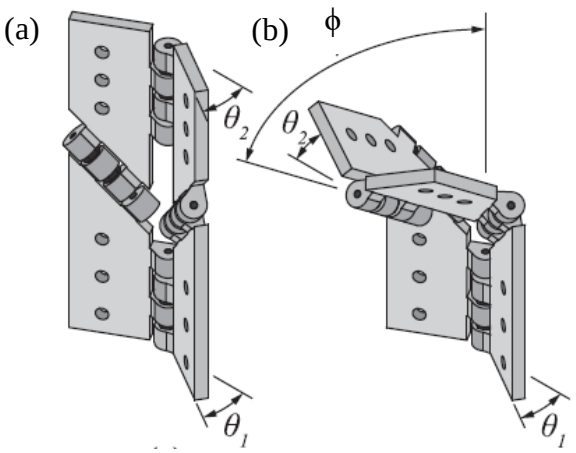


Figure 2.3: (a) Parallel Configuration. (b) Antiparallel Configuration. (Liu et al., 2021)

During transition from parallel to antiparallel configurations, magnitudes for angles θ_1, θ_2 and ϕ will increase. At any point of the transition, magnitudes of θ_1 will always be equal to θ_2.

An origami string is a multivertex template that contains multiple Miura vertices. Figure 2.4 shows an example of an origami string template that contains 3 vertices.

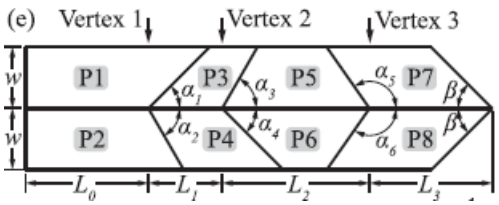


Figure 2.4: Origami String Template with Three Vertices. (Liu et al., 2021)

Figure 2.5 shows the method (Liu et al., 2021) used to actuate one finger on the gripper. The actuation mechanism consists of one rotational DC motor and shaft linkage mechanism.

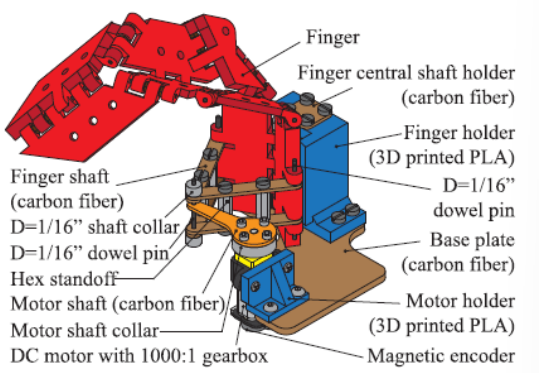


Figure 2.5: Actuation Mechanism for each Finger on Gripper.
(Liu et al., 2021)

Figure 2.6 shows that mechanical stops are designed on the hingers to prevent the finger of the gripper from folding in the other direction. This would be a useful design in the exoskeleton to restrict the angular displacement of the transmission system within the static constraints of the finger and wrist joints.

Figure 2.6: Mechanical Stops on Hinges. (Liu et al., 2021)

2.13 Motor Recovery Evaluation

It is important to evaluate the motor recovery of patients with hand spasticity after their rehabilitation sessions in order to gage how far along is their healing progress and whether the rehabilitation protocol is effective. There are many methods to evaluate motor recovery such as: Modified Ashworth Scale, Fugl-Meyer Assessment (Upper Extremity) and active range of motion.

The Modified Ashworth and Fugl-Meyer Assessment evaluation methods are widely used. However, since they require a therapist to be present, it cannot be carried out by the exoskeleton itself.

On the other hand, recovery progress of the patient can be evaluated by detecting the active range of motion that can be performed by the patients themselves. Since this range of motion can be detected using flex sensors, it can be carried out using the hand exoskeleton.

2.14 Effectiveness of Origami Folding Activity as a Hand Rehabilitation Technique

The effectiveness of applying the origami folding activity as a hand rehabilitation therapeutic method was reviewed to decide whether the user interface for the hand rehabilitation system should contain a functionality that provides users with tutorials on how to fold origami.

One of the studies reviewed whether origami folding sessions contributed to the improvement of the hand functions of patients. These patients were given a weekly, 1 hour session of origami classes for the span

of 6 weeks. These classes thought the patients how to fold a series of origami models that has a range of complexities. Patients participating in these sessions displayed a larger progress in sub-test scores for the Jebsen-Taylor Hand Function Test compared to the control group. (M Wilson et al., 2008)

Another study investigated whether subjects can improve their dexterity by folding origami cranes. Subjects were instructed to fold origami cranes for 40 to 50 minutes a day for a span of 4 weeks. Those participating in these folding sessions shown a significant improvement in the Purdue Pegboard test and the Grooved Pegboard test compared to the control group. (Bae, 2013)

The studies above shows that folding origami is effective as a rehabilitation technique. Therefore, tutorials on how to fold origami will be provided in the user interface.

2.15 Summary of Findings

A hand exoskeleton should respect the static and dynamic constraints of the hand, be light and comfortable to be worn for long periods of time, be cost-effective, generate sufficient forces, user-friendly and adjustable to different hand sizes.

Moreover, the hand exoskeleton system should contain a transmission system, servo motor actuators, a control system, flex sensors, data transmission network and an Android mobile application user interface. Furthermore, one possible transmission mechanism design can be based on the Miura vertex theory. In addition, the hand exoskeleton should be able to measure the patient's joint range of motion to determine their recovery progress. Lastly, a functionality that provides the user with origami tutorials can be designed in the user interface to increase the recovery rate of the patients.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

This section covers the specifications for the rehabilitation system, the system architecture design and materials chosen, total expenditure, the work plan, Work Breakdown Structure and Gantt Chart for this project as well as the methods used to test and analyse the performance of the rehabilitation system.

3.2 Specifications of Exoskeleton System

Referring to literature review conducted on the design considerations of exoskeleton systems, Table 3.1 shows the specifications set for this exoskeleton system:

Table 3.1: Specifications for Exoskeleton System

Specifications	Justification
• Actuate movement in four fingers (index, middle, ring, little fingers) and wrist. • For fingers: one DoF at the PIP joint and one DoF at the MCP joint. • For wrist: one DoF at Radiocarpal joint. • Type of motion: ▪ flexion and extension for MCP and PIP joints ▪ extension for Radiocarpal joint.	• Need at least three functional fingers to carry out most ADLs. • Each of these fingers need to have at least two DoF to carry out most ADLs. • Adduction and abduction at the joints are not essential to perform ADLs.
• Movement of the MCP joints for all 4 fingers are coupled. • Movement of the PIP joints for all	• To reduce the complexity of the design system. • To reduce cost by enabling the

4 fingers are coupled.	movement of all 4 fingers to be actuated by using only 2 actuators.
The maximum angles generated by transmission mechanism on PIP, MCP and radiocarpal joints should respect the static constraints of the hand. Moreover, the maximum angular velocities generated by transmission mechanism should not exceed the maximum angular velocities that can be generated by the hand naturally.	To ensure that the transmission mechanism does not cause the patient discomfort or pain.
Total cost of the system must be less than RM 1500.	To ensure that it will be affordable for patients to purchase for home use.
Total weight attached to the hand must be less than 500 g.	To ensure that it is portable without causing fatigue to the patient.
User interface allows patients to select from 3 levels of angular velocity that is generated by the transmission mechanism.	Allows users to reduce the angular velocity generated by the transmission mechanism if they feel pain or discomfort.
Sensing system detects the flexion angle of MCP joint and extension angle of radiocarpal joint. Then, the maximum angle of flexion for the MCP joint and the maximum angle of extension for the radiocarpal joint is displayed to user via the user interface.	To allow the patient review their recovery progress.

3.3 Work Plan

A work plan is developed for this project.

1. Planning Phase
    1.1. Title Selection
    1.2. Work Plan Formulation
    1.3. Gantt Chart Construction
    1.4. Problem Formulation
    1.5. Setting Objectives
2. Preliminary Design phase
    2.1. Literature Review
        2.1.1. Stroke and its Effects
        2.1.2. Anatomy, Kinetics and Kinematic of Hand
        2.1.3. IoT Architecture
        2.1.4. Market Research
        2.1.5. Design Considerations for Hand Exoskeletons
        2.1.6. Components Required in Exoskeleton System
        2.1.7. Origami String as Transmission Mechanism
        2.1.8. Motor Recovery Evaluation Techniques
    2.2. Preliminary System Design
        2.2.1. Determine Specifications for Exoskeleton
        2.2.2. Determine Architecture of Exoskeleton System
        2.2.3. Preliminary Design of Solution 1 and 2 for Exoskeleton System
        2.2.4. Preliminary Design and Modelling of Transmission Mechanism
        2.2.5. Prototyping User Interface Design
        2.2.6. Determine Budgeted Expenditure
3. Documentation Phase
    3.1. Progress Report Writing
    3.2. Log Book Compilation
    3.3. Presentation
4. Design and Construction Phase

4.1. Purchase Components

4.2. Hardware Design and Construction

4.2.1. Design, 3D-Print and Assemble Transmission Mechanism

4.2.2. Design and Construct Sensing Glove

4.3. Software Design and Construction

4.3.1. Develop Code for ESP32

4.3.2. Develop Code for Android Mobile Application

5. Functionality Testing Phase
6. System Integration Phase (Sensing, Transmission and User Interface)
7. System Testing Phase
8. Documentation Phase

8.1. FYP Poster Designing

8.2. Final Report Writing

8.3. Log Book Compilation

8.4. Presentation

3.3.1 Work Breakdown Structure

Figure 3.1 shows the Work Breakdown Structure for tasks conducted in the first part of the study.

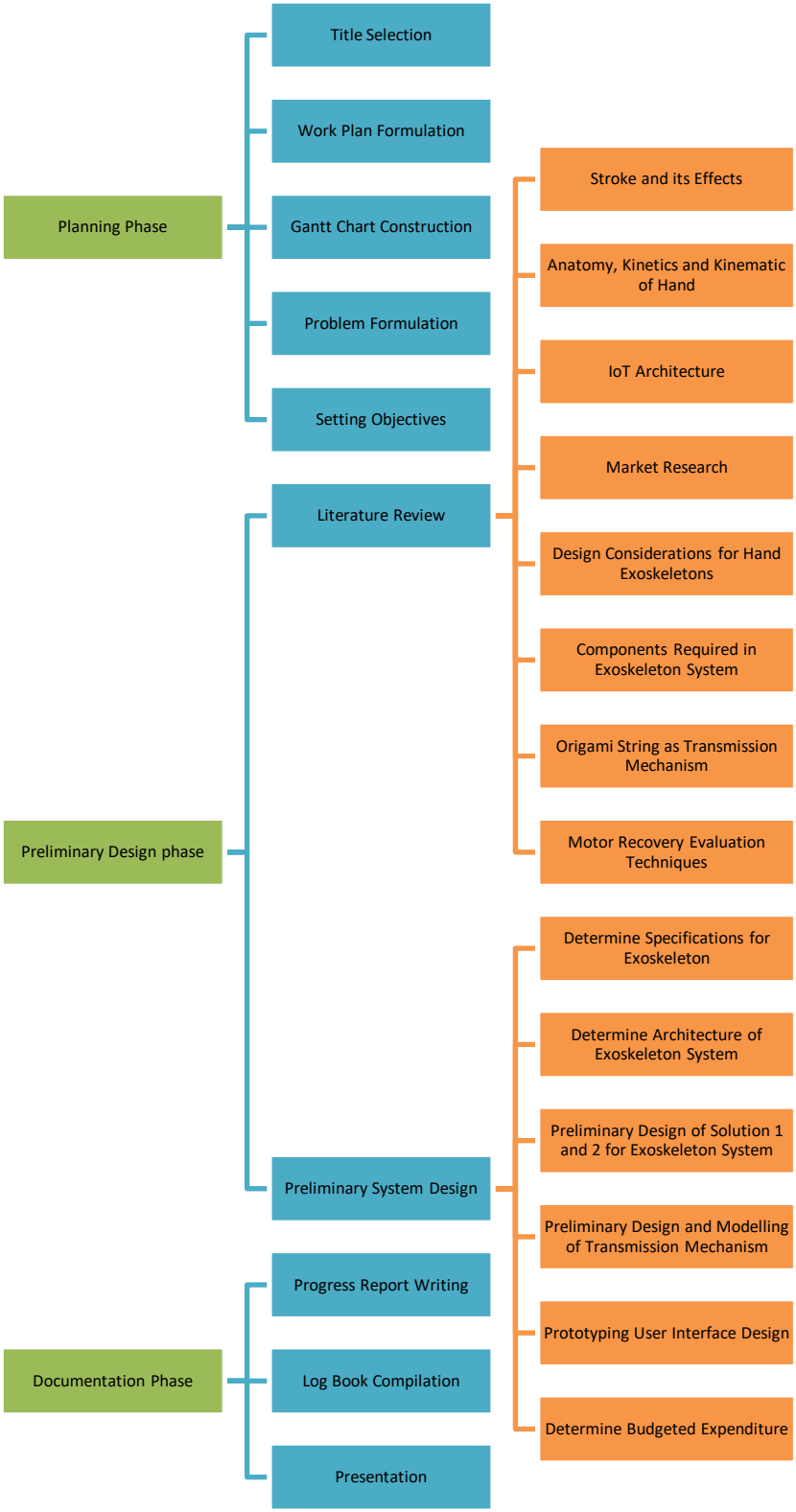


Figure 3.1: Work Breakdown Structure for FYP Part 1.

Figure 3.2 shows the Work Breakdown Structure for tasks conducted in the second part of the study.

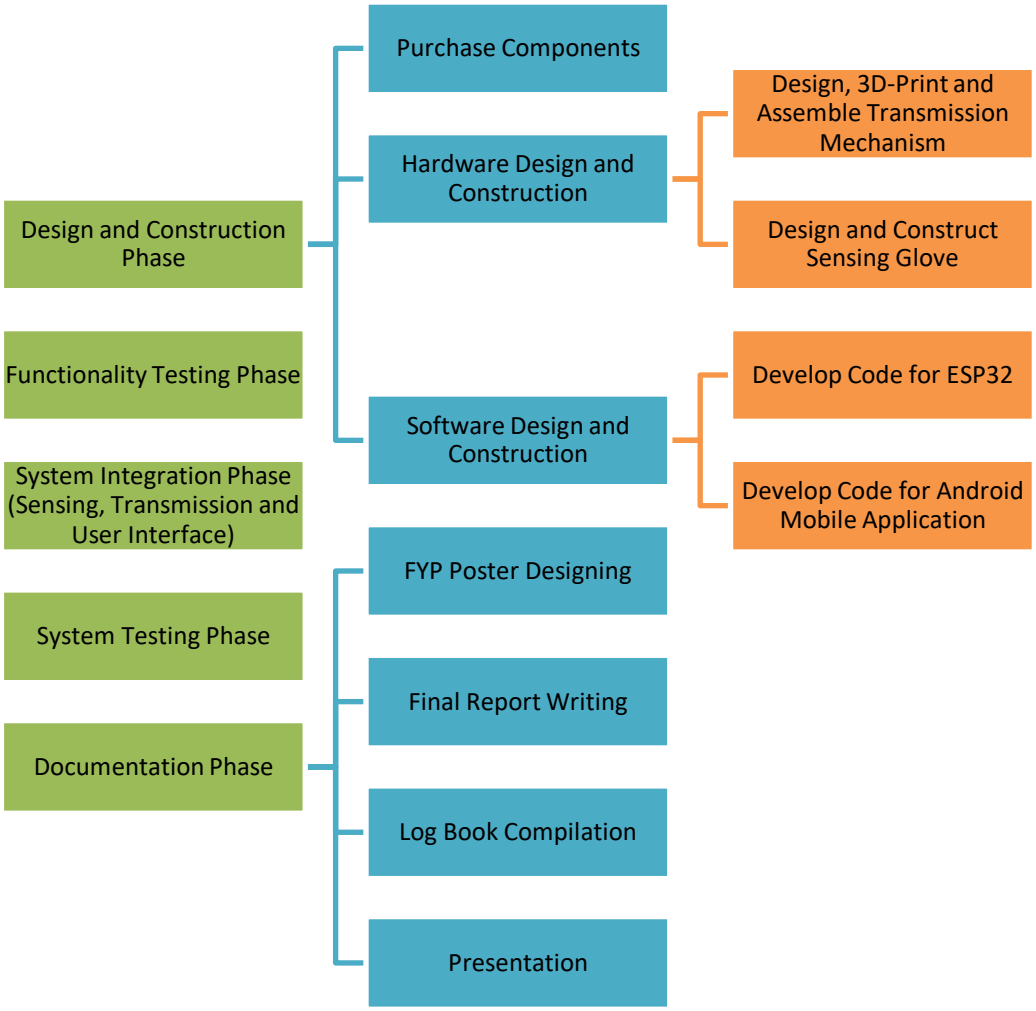


Figure 3.2: Work Breakdown Structure for FYP Part 2.

3.3.2 Gantt Chart

Gantt Charts were constructed using ProjectLibre. Figure 3.3 illustrates the Gantt Chart for tasks conducted this semester. The work duration was fourteen weeks.

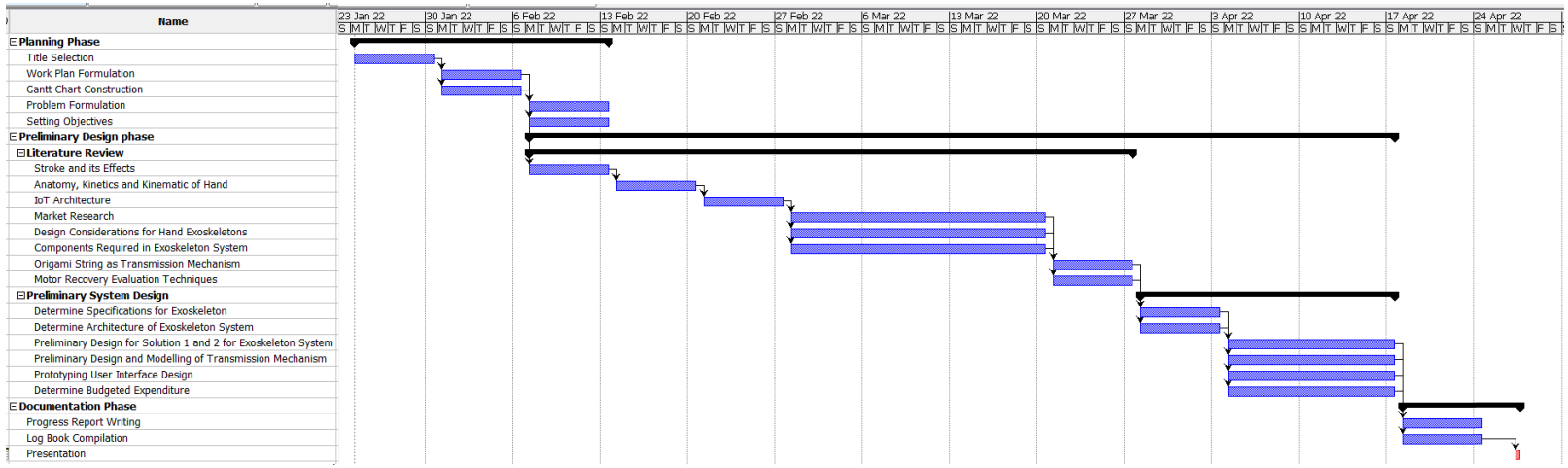


Figure 3.3: Gantt Chart (Semester One)

Figure 3.4 illustrates the Gantt Chart for tasks conducted in the second part of the study. The work duration was fifteen weeks because tasks commenced immediately during the semester break.

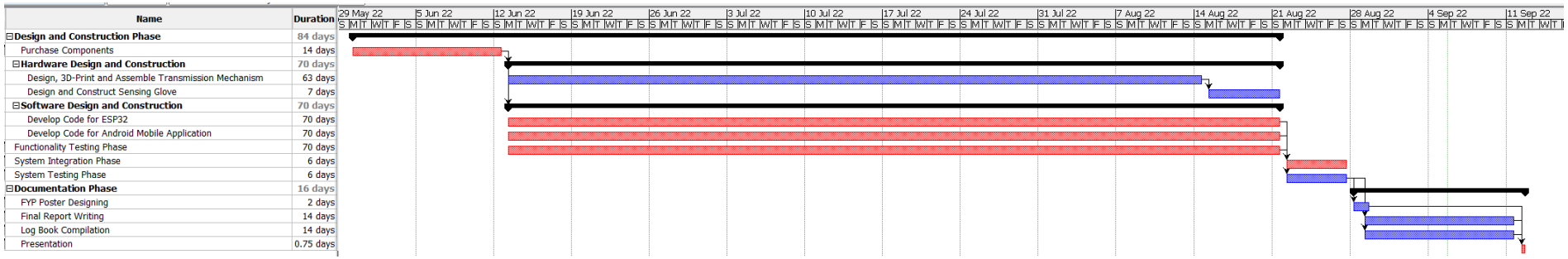


Figure 3.4: Gantt Chart (Semester Two)

3.4 Architecture of Hand Rehabilitation Exoskeleton System

Figure 3.5 illustrates the architecture design for the exoskeleton system.

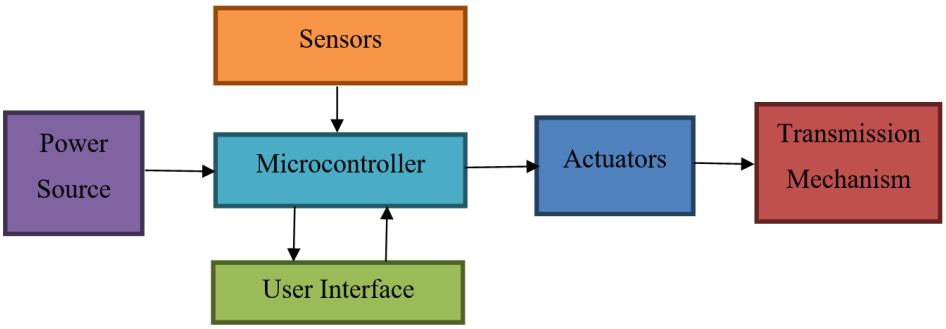


Figure 3.5: Architecture Design for Exoskeleton System.

Table 3.2 describes the function of each element in the exoskeleton architecture.

Table 3.2: Function of Elements in Exoskeleton System.

Elements	**Function**
Power Source	To supply power to actuators, microcontroller and sensors.
Microcontroller	▪ To control the actuators. ▪ To read data from the sensors. ▪ To transmit data to user interface. ▪ To read data from the user interface.
Sensors	▪ To detect the range of motion of the MCP and radiocarpal joints.
User Interface	▪ To allow the user to change the angular velocity generated by the transmission mechanism. ▪ To display the recovery progress to the user. ▪ To display video tutorials on how to fold origami.
Actuators	To drive the transmission

	mechanism.
Transmission Mechanism	To generate motion at the MCP, PIP and radiocarpal joints.

3.5 Anatomical Dimension Assumptions

To create the first exoskeleton prototype, the dimensions of the hand are assumed to be as shown in Table 3.3. Since the movement of all four fingers will be coupled, the lengths for the finger phalanges on the four fingers are assumed to be equal to that of the middle finger.

Table 3.3: Dimension Assumptions of the Hand.

Appendage	Dimensions
Distal Phalange	Length: 2 cm
Intermediate Phalange	Length: 3.5 cm
Proximal Phalange	Length: 6 cm
Palm	Length: 8.5 cm

3.5.1 Components Chosen for Each Element in the System Architecture

3.5.1.1 Power Source

The power source chosen should be portable to allow the user to carry out the rehabilitation activities anywhere. Therefore, a power bank was chosen. This power bank will supply the microcontroller, sensors and actuators with power via a modified Micro USB cable.

3.5.1.2 Microcontroller

This project would require 1 microcontroller for the entire exoskeleton system. This microcontroller should have a minimum of two pulse-width modulation (PWM) output pins to control the actuators as well as a minimum of nine analog input pins to receive data from the sensors.

Table 3.4 compares some of the microcontrollers that are available in the market.

Table 3.4: Comparison between Compatible Microcontrollers for Arduino IoT Cloud.

Specifications	**ESP32**	**Arduino Nano 33 IoT**	**Arduino MKR WAN 1300**	**Arduino MKR WiFi 1010**
Cost	RM 21.80	RM 143	RM 177	RM 199
Processor	32-bit	32-bit	32-bit	32-bit
Network	WiFi, Bluetooth, and BLE	WiFi, Bluetooth, and BLE	LoRaWAN	WiFi, Bluetooth, and BLE
Powered by	5 V	5 V	5 V	5 V
PWM pins	16	11	12	13
Analog input pins	18	8	7	7
Weight	-	5 g	32 g	32 g

From table 3.4, ESP32 is selected because it is the cheapest option, has enough PWM and analogue pins and has an integrated network module with WiFi, Bluetooth or Bluetooth Low Energy (BLE) network capabilities.

3.5.1.2.1 Programming ESP32

The programming language for ESP32 is C++. The Arduino Web Editor will be used to construct, debug and upload the C++ programme to the ESP32 microcontroller.

3.5.1.3 Sensors

This project would require 2 sensors to measure the angular displacement of the MCP joint of the middle finger and the radiocarpal joint. From the literature review conducted, flex sensors were selected for the exoskeleton system because of the ease of setup and its attachment would not disturb the movement of the fingers.

3.5.1.4 Actuators

This project would require two actuators to actuate the transmission mechanism. Based on the literature review conducted, servo motors were selected because it is cheaper, lighter and would not take up too much space.

3.5.1.5 Transmission Mechanism

3.5.1.5.1 Mechanism Design Theory

The transmission mechanism design selected is inspired by the Miura vertex and origami string concept. This design is chosen because the maximum flexion angles at MCP and PIP joints as well as the maximum extension angle of the radiocarpal joint can be explicitly designed into the transmission mechanism. This would remove the possibility of the exoskeleton generating angles of rotation that exceed the static constraints of the finger and wrist joints.

Figure 3.6 visualises the first version of crease pattern and dimensions for the transmission mechanism that will move the PIP and MCP joints. Red lines are creases that are folded inwards (valley folds) while blue lines are creases that are folded outwards (mountain folds). The Miura vertex on the left has 55° angles to generate the maximum flexion angle of 110° at the PIP joint when the Miura vertex is in its antiparallel configuration. On the other hand, the Miura vertex on the right has 45° angles to generate the maximum flexion angle of 90° at the MCP joint when the Miura vertex is in its antiparallel configuration. Figure 3.7 illustrates the transmission mechanism in its antiparallel configuration.

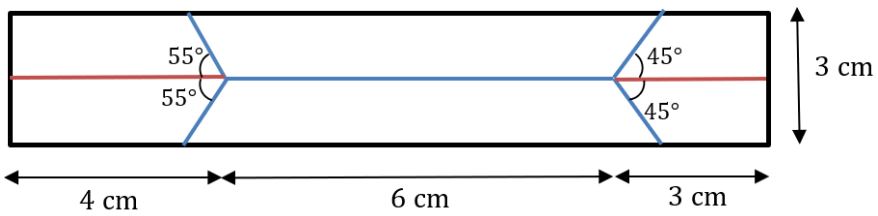


Figure 3.6: Crease Pattern and Dimensions for Transmission Mechanism (PIP and MCP joints).

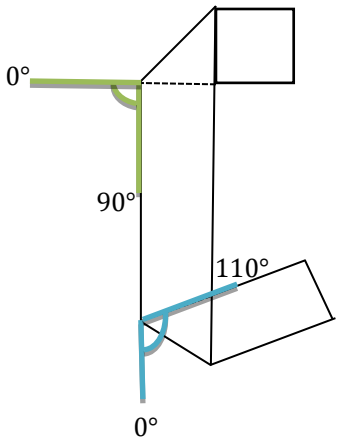


Figure 3.7: Transmission Mechanism in Antiparallel Configuration (PIP and MCP joints).

Figure 3.8 visualises the first version of the crease pattern and dimensions for the transmission mechanism that will move the radiocarpal joint. The Miura vertex has 30° angles to generate the maximum extension angle of 60° at the Radiocarpal joint when the Miura vertex is in its antiparallel configuration. Figure 3.9 illustrates the transmission mechanism in its antiparallel configuration.

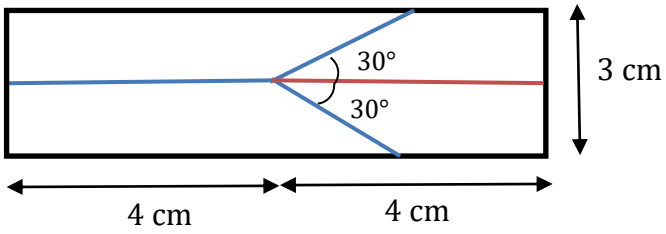


Figure 3.8: Crease Pattern and Dimensions for Transmission Mechanism (Radiocarpal joint).

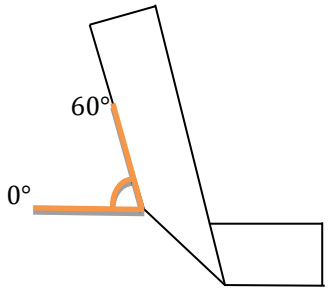


Figure 3.9: Transmission Mechanism in Antiparallel Configuration (Radiocarpal Joint).

The crease patterns shown in Figure 3.6 and 3.8 is subjected to change as newer versions of the transmission mechanism is designed.

3.5.1.5.2 Software to Design Mechanism

SOLIDWORKS 2019 will be used to design all the sections of the transmission mechanism that will be 3D printed. This software was chosen because it is an industry standard software, it is easy to use, and allows the assembly all the sections together before the design is printed out. The objects designed will be exported as STL (Standard Tessellation Language) files.

In addition, Ultimaker Cura will be used to slice the objects created by SOLIDWORKS to prepare them for 3D printing. This software will convert the STL files into G-Code (Geometric Code) files. These files can be readily read by the 3D printer.

3.5.1.5.3 3D Printer

The 3D printer chosen for this project is the Creality Ender 3. This 3D printer is compatible with PLA (Polylactic acid), ABS (Acrylonitrile Butadiene Styrene) and TPU (Thermoplastic Polyurethane) filaments. In addition, it has a printing volume of 220 x 220 x 250 mm which is sufficient to print the transmission mechanism. It also has a layer resolution of 0.1 mm which is adequate for this project.

3.5.1.5.4 Filament

The material chosen for the filament that will be used in the 3D printer is PLA. This is because PLA does not produce hazardous fumes when heated like ABS. Furthermore, PLA is not prone to warping when ambient temperature decreases like ABS. In addition, PLA does not have the high elastic properties that TPU has. Elastic properties are not ideal for the transmission mechanism because force has to be effectively transferred from segment to segment.

3.5.1.6 User Interface

Referring to the literature review conducted, Android mobile application was chosen as the user interface type. This is because it can create a more reliable network with the ESP32, does not require a router and has a customizable interface design which can allow the integration of origami video tutorials.

3.6 Performance Testing and Results Analysis Methodology

3.6.1 Transmission System

3.6.1.1 Purpose

To determine whether these specifications listed below are met:

i. Can generate flexion and extension at MCP and PIP joints, and extension at radiocarpal joint.
ii. Maximum MCP flexion generated should be 90° and below.
iii. Maximum PIP flexion generated should be 110° and below.
iv. Maximum radiocarpal extension angle generated should be 60° and below.
v. “Difficult” mode capable of generating the highest magnitude of angular velocities, followed by “Intermediate” mode and then “Easy” mode.
vi. Maximum magnitude of angular velocity at PIP joint is 858 **°/s.**
vii. Maximum magnitude of angular velocity at MCP joint is 694 **°/s.**
viii. Maximum magnitude of angular velocity at radiocarpal joint is 180 **°/s.**

3.6.1.2 Steps

i. Setup the blue backdrop to create a clearer background.

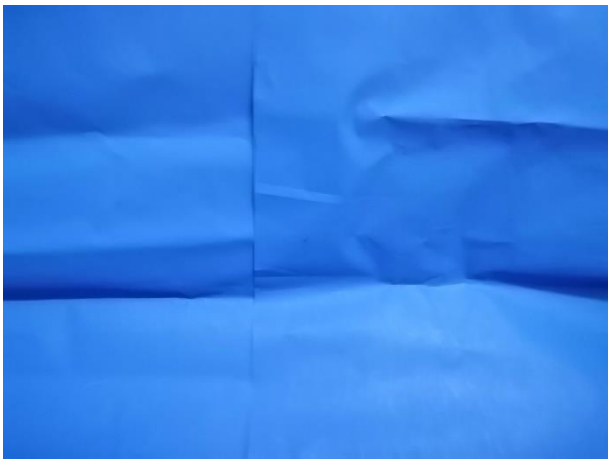


Figure 3.10: Blue Coloured Backdrop.

ii. Setup the ring light and stand so that the light shines perpendicular to the ground.

Figure 3.11: Ring Light and Stand Setup.

iii. Attach the transmission mechanism to the hand.

iv. Attach markers (i.e., black stickers) to the PIP, MCP and radiocarpal joints of the index finger. In addition, one marker is attached to the transmission mechanism, in front of where the DIP joint should be located.

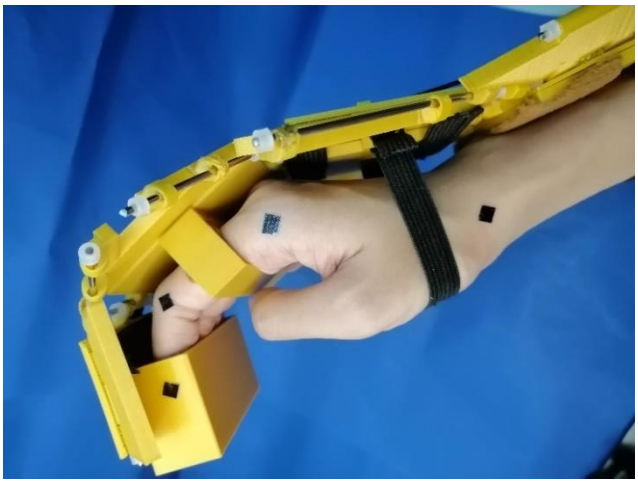


Figure 3.12: Hand Attached with Transmission Mechanism and Markers.

v. Activate the “Easy” mode continuous rehabilitation activity from the user interface.
vi. Start the video recording.
vii. Position the hand so that the sagittal view of the hand is being captured by the camera.

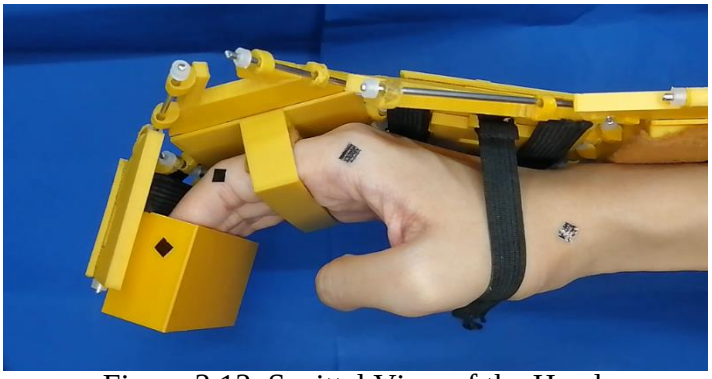


Figure 3.13: Sagittal View of the Hand.

viii. Record the movement actuated by the transmission mechanism for 60 seconds.
ix. Upload the video to Kinovea and attach trackers to measure the angles generated and the angular velocity for the extension of radiocarpal joint as well as the flexion of the MCP and PIP joints.

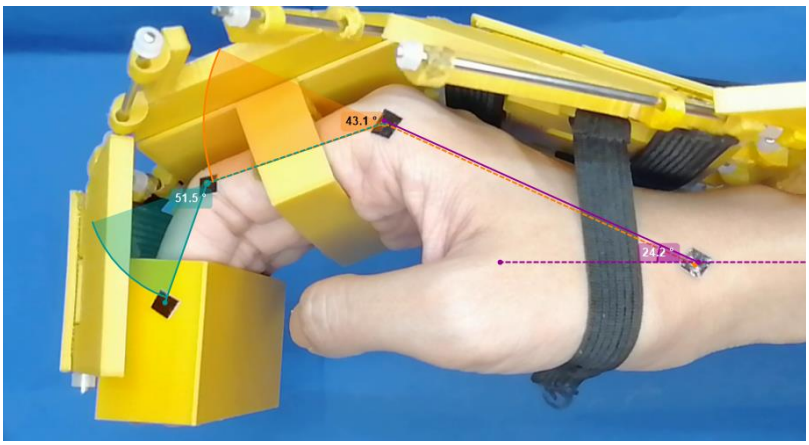


Figure 3.14: Trackers Attached Using Kinovea.

x. Export the raw data.

xi. Use Microsoft Excel to visualise the raw data in graphs. Then, using the "Max" function, locate the maximum angles generated by radiocarpal extension, as well as MCP and PIP flexion. Moreover, using the "ABS" and "Max" functions, locate the maximum angular velocity magnitudes generated at the radiocarpal, MCP and PIP joints.

xii. Repeat steps v to viii for the "Intermediate" and "Difficult" modes.

3.6.2 Sensing System

3.6.2.1 Purpose

To measure the performance of sensing system in detecting the maximum flexion angle of the MCP joint of middle finger and the maximum extension angle of the radiocarpal joint.

3.6.2.2 Steps

i. 0°, 30°, 60°, and 90° angles are constructed on paper.

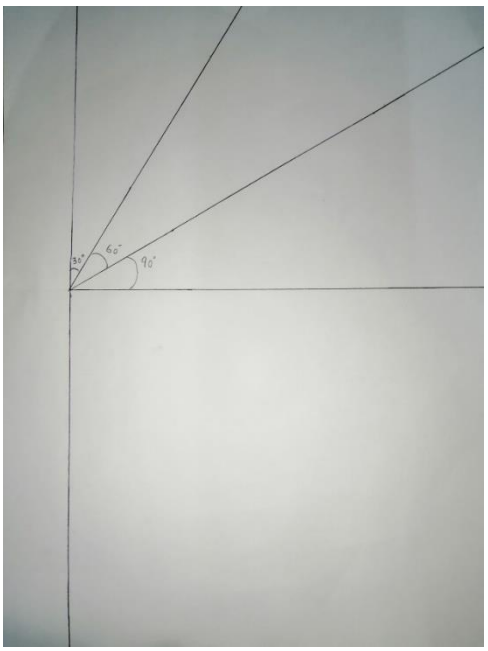


Figure 3.15: Angles Constructed on Paper.

ii. The sensing glove is worn.

iii. The middle finger is positioned so that the MCP joint has a flexion of 0°.

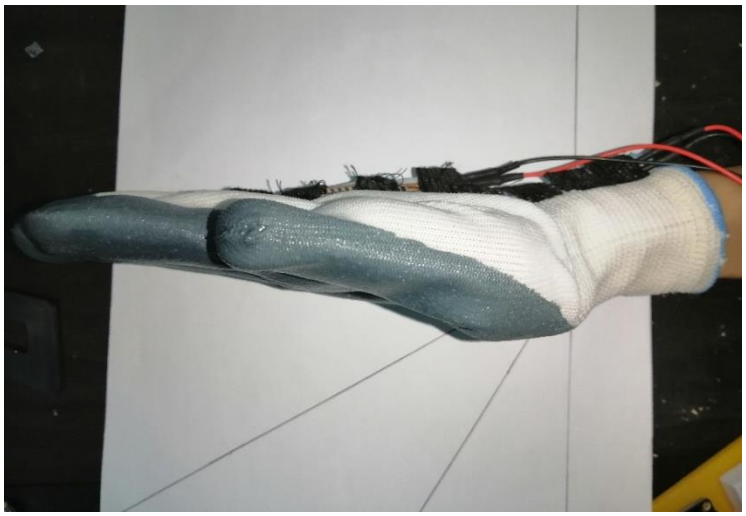


Figure 3.16: Position of Middle Finger With 0° Flexion at MCP Joint.

iv. Obtain and record the value for the maximum flexion angle of MCP joint from the user interface.

Figure 3.17: Maximum MCP Flexion Angle Displayed on the User Interface.

v. Steps iii and iv are repeated for MCP flexion angles of 30°, 60°, and 90°.
vi. Steps iii and iv are repeated for radiocarpal extension angles of 0°, 30° and 60°.
vii. Steps iii to vi are repeated 9 more times to obtain 10 sample values.
viii. Construct boxplots to visualise the distribution of sample values.
ix. Calculate the mean values of angles measured and the mean differences between these values and the actual angles at the MCP and radiocarpal joints.

3.6.3 User Interface

3.6.3.1 Purpose

To determine whether these specifications listed below are met:

i. Users can select between different activities.
ii. Mobile application can connect to ESP32 via Bluetooth.
iii. Users can select different rehabilitation modes.
iv. Users can see the maximum angles for radiocarpal extension and MCP flexion.
v. Users can view origami tutorial videos.

3.6.3.2 Steps

Functionality testing is conducted for the mobile application by exploring the different functionalities offered in the application.

3.6.4 Total Cost of Rehabilitation System

3.6.4.1 Purpose

To evaluate whether the total cost of the entire rehabilitation system is less than RM 1500.

3.6.4.2 Steps

Expenditure table is constructed to calculate the total cost.

3.6.5 Total Weight of Transmission Mechanism Attached to Hand

3.6.5.1 Purpose

To evaluate whether the total weight of the transmission mechanism attached to the hand is less than 500 g.

3.6.5.2 Steps

Weigh the transmission mechanism segments on an electronic weighing scale that has an accuracy of 0.1 g.

3.7 Total Expenditure

Table 3.5 shows the total expenditure for the construction of the entire rehabilitation system. The total expenditure was RM 533.70. Since it costs less than RM 1500, the rehabilitation system has fulfilled part of an objective set in Chapter 1.

Table 3.5: Total Expenditure for the Construction of Rehabilitation System.

Components	Quantity	Cost per Unit	Total Cost Design
ESP32 Microcontroller	1	RM 21.80	RM 21.80
Flex Sensors (2.2	2	RM 49	RM 98

inch)			
MG995 Servo Motors	2	RM 10.30	RM 20.60
Metal Rods (2 mm diameter)	12 x 15 cm	RM 1.60	RM 19.20
2mm Brushings	56	RM 0.20	RM 11.20
Pure PLA Filaments	1 kg	RM 47.50	RM 47.50
Super Glue	2	RM 1.60	RM 3.20
Elastic Bands (Flat)	1	RM 3.50	RM 3.50
Elastic Bands (Thin)	1	RM 2.00	RM 2.00
Gloves	1	RM 4	RM 4
Power Bank	1	N / A	RM 0
Breadboard	1	N / A	RM 0
Connecting Wires	-	N / A	RM 0
Soldering Iron	1	N / A	RM 0
Solder	1	N / A	RM 0
Creality Ender 3	1	N / A	RM 0
SOLIDWORKS	1	N / A	RM 0
Project Libre	1	RM 0	RM 0
Android Studio	1	RM 0	RM 0
Arduino Web Editor	1	RM 0	RM 0
Total Cost			RM 533.70

- N / A: Not applicable because the components are readily available.

3.8 Summary

The entire Final Year Project took 29 weeks to complete. Table 3.6 summarizes the elements inside the rehabilitation system architecture design along with the components selected.

Table 3.6: Elements in Rehabilitation System Architecture Design and Components Selected.

Elements	**Components Selected**
Power source	1 power bank
Microcontroller	ESP32 + Arduino Web Editor
Sensors	2 Flex sensors
User interface	Android mobile application + Android Studio
Actuators	2 servo motors
Transmission mechanism	Origami String theory + 3D-printing technology + Creality Ender 3 + PLA + SOLIDWORKS + Ultimaker Cura

Next, the total expenditure for the entire rehabilitation system was RM 533.70. Lastly, using tools such as Kinovea, Microsoft Excel and an electronic weighing scale, the performance of the rehabilitation were tested and analysed for the following areas: transmission system, sensing system, user interface, total cost of rehabilitation system and total weight of transmission mechanism sections attached to the hand.

CHAPTER 4

REHABILITATION SYSTEM DESIGN AND CONSTRUCTION

4.1 Introduction

This chapter elaborates on the design and construction of the entire rehabilitation system. The design and construction of the following systems covered here are: electrical circuit, transmission mechanism, sensing system, and origami tutorial functionality. Lastly, the method to integrate these systems together was also explained.

4.2 Electrical Circuit

Figure 4.1 shows the electric circuit design that connects the MG995 servo motors and flex sensors to the ESP32 microcontroller. The microcontroller, servo motors and flex sensors are powered externally by a 5V power bank. Moreover, 10 kΩ resistors are used to construct the voltage divider circuit for the flex sensors. Furthermore, D14 and D15 pins of the microcontroller are output pins that produce pulse-width modulation (PWM) output signals. As such, they are connected to the signal input pins of the servo motors to drive the servo motors. Lastly, the D34 and D35 pins of the microcontroller are analog-to-digital converter (ADC) input pins that connect to the flex sensor voltage divider circuits to receive variable voltage signal. Figure 4.2 shows the constructed electrical circuit.

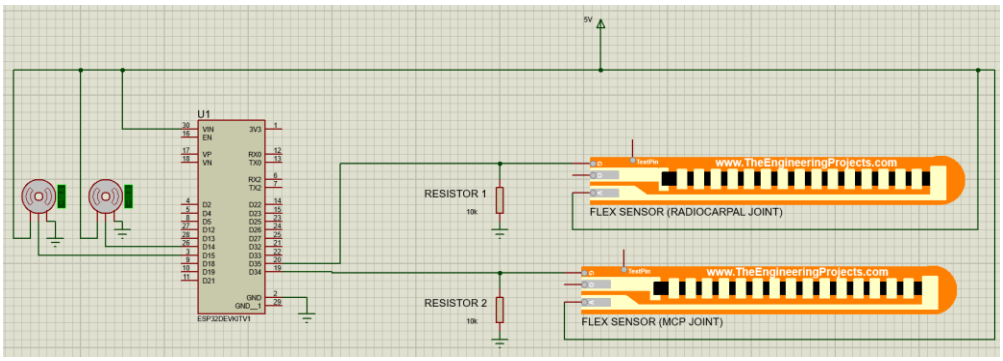


Figure 4.1: Electrical Circuit Design for Rehabilitation Device.

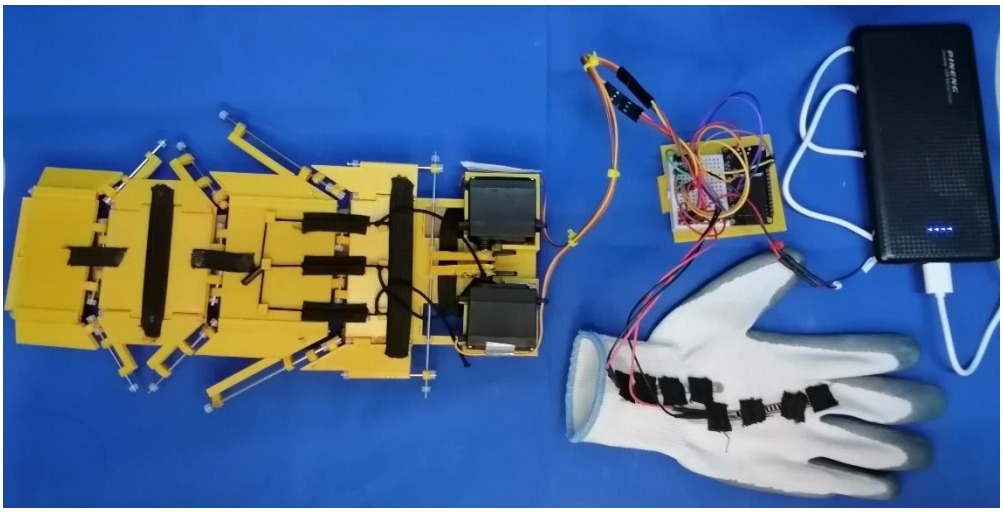


Figure 4.2: Constructed Electrical Circuit for Rehabilitation Device.

4.3 Transmission System

The transmission system includes the transmission mechanism and actuators.

4.3.1 Origami Theory

Figure 4.3 visualises the latest crease pattern and dimensions for the transmission mechanism that will generate movement at the PIP, MCP and radiocarpal joints. Compared to the previous design found in Section 3.5.1.5.1: Mechanism Design Theory, this new design combines the transmission mechanism of the MCP and PIP joints with that of the radiocarpal joint to create a more efficient design. The Miura vertex on the left has 55° angles to generate the maximum flexion angle of 110° at the PIP joint when the Miura vertex is in its antiparallel configuration. On the other hand, the Miura vertex in the middle has 45° angles to generate the maximum flexion angle of 90° at the MCP joint when the Miura vertex is in its antiparallel configuration. Lastly, the Miura vertex on the right has 30° angles to generate the maximum extension angle of 60° at the radiocarpal joint when the Miura vertex is in its antiparallel configuration. Figure 4.4 illustrates the transmission mechanism in its antiparallel configuration.

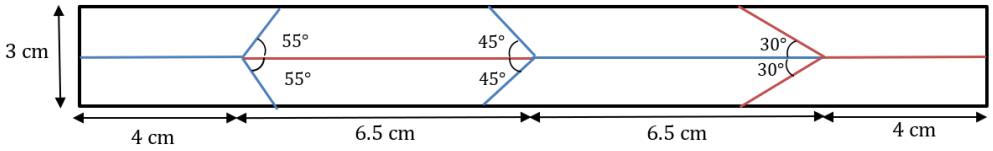


Figure 4.3: Crease Pattern and Dimensions for Transmission Mechanism

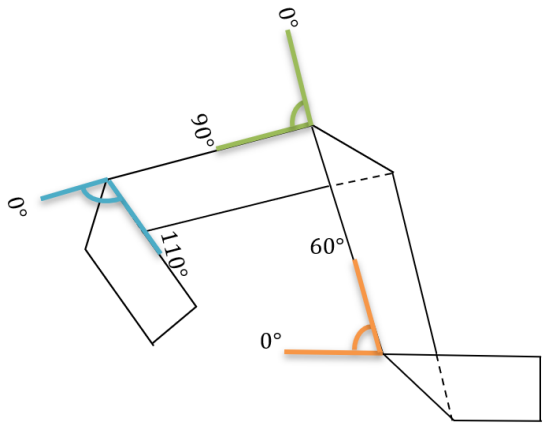
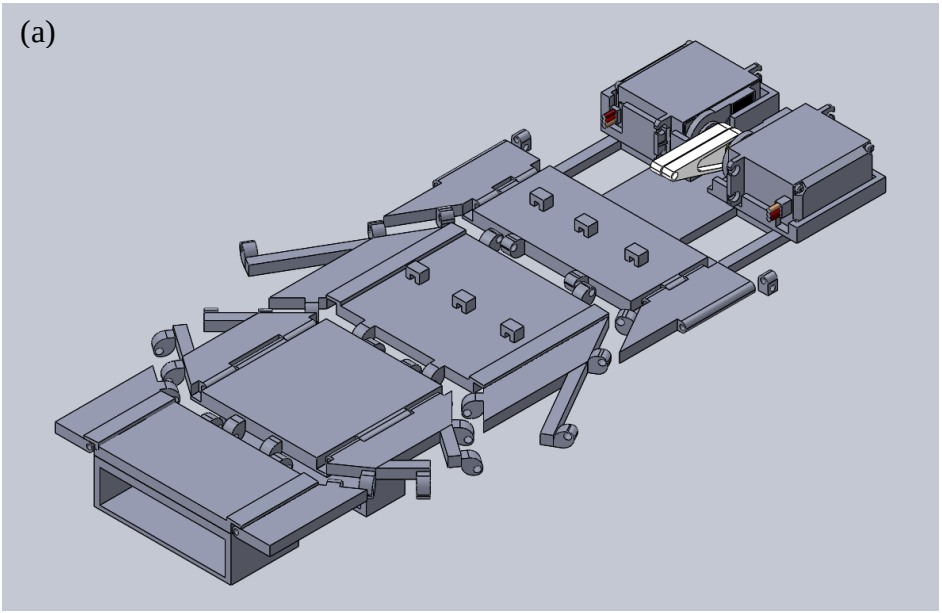


Figure 4.4: Transmission Mechanism in Antiparallel Configuration.

This design ensures that the transmission mechanism does not actuate movements that exceed the static constraints of the hand.

4.3.2 Final Hardware Design

After ten rounds of designing, 3-D printing, testing and modifying the transmission mechanism, the final design is as shown in Figure 4.5 (a) and (b). There are 34 separate segments in total (excluding the 30 separate loops that attach the sections together). Figure 4.6 shows the 3D-printed and assembled transmission system.



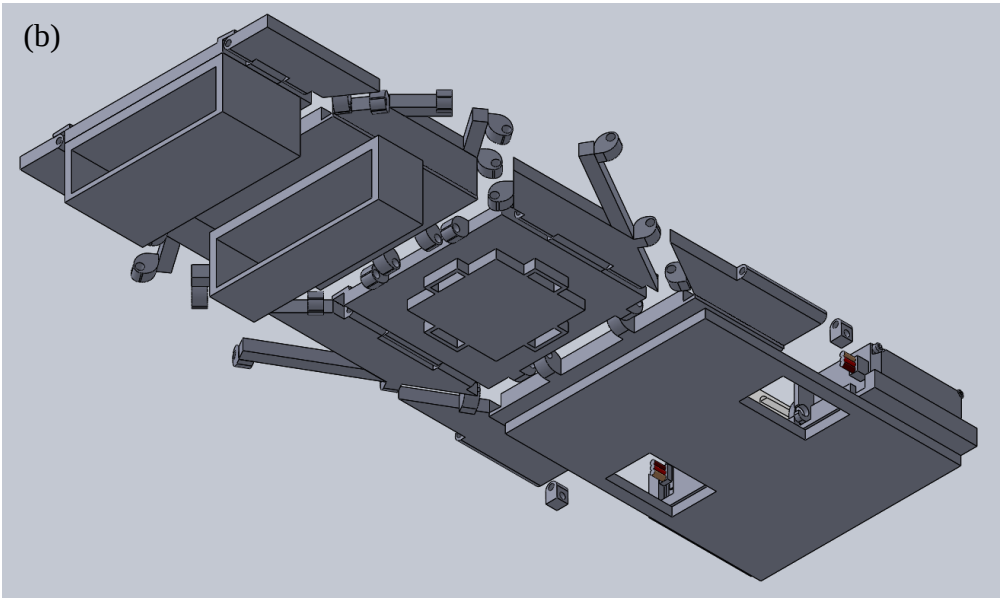


Figure 4.5 (a) & (b): Final Transmission Mechanism Design.

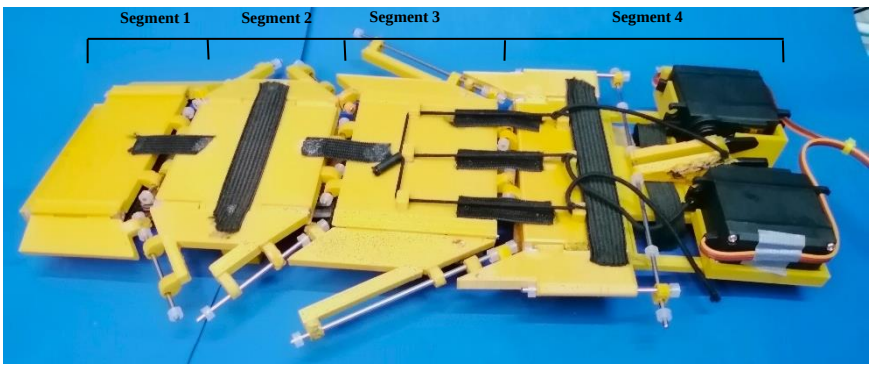


Figure 4.6: 3D-Printed and Assembled Transmission Mechanism.

In general, the transmission mechanism consists of 4 main segments as shown in Figure 4.6. Segments 1, 2, 3, and 4 will be attached to the distal-middle phalanges, proximal phalanges, palm and wrist respectively as shown in Figure 4.7. When the servo motor arms move upwards in segment 4, the forces will be transferred to segment 3, then segment 2 and subsequently segment 1. This would result in the flexion of the MCP and PIP joints and the extension of the radiocarpal joints as shown in Figure 4.8.

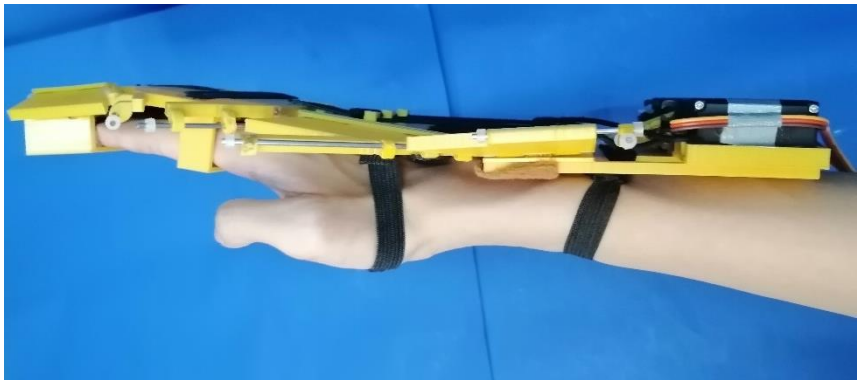


Figure 4.7: Transmission Mechanism Attached to the Hand.

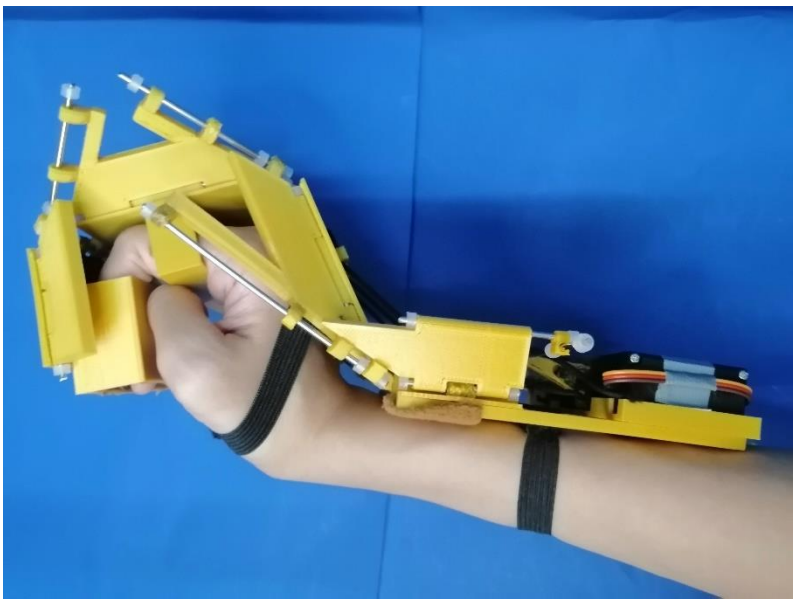


Figure 4.8: Transmission Mechanism Actuating MCP and PIP Joint Flexion and Radiocarpal Joint Extension.

4.3.3 Design Features

The following design features are incorporated to ensure that the transmission mechanism can actuated the movements as intended.

Initially the loops designed to connect the segments together diagonally as shown in Figure 4.9 (a) have an axis of rotation that is located the middle of the segment. As such, the segments cannot bend completely as shown in Figure 4.9 (b). To solve this issue, the loops were designed so that the axis of rotation between the segments are now located between the 2 segments as shown in Figure 4.9 (c) which enabled the segments to bend completely like folds on a paper as shown in Figure 4.9 (d).

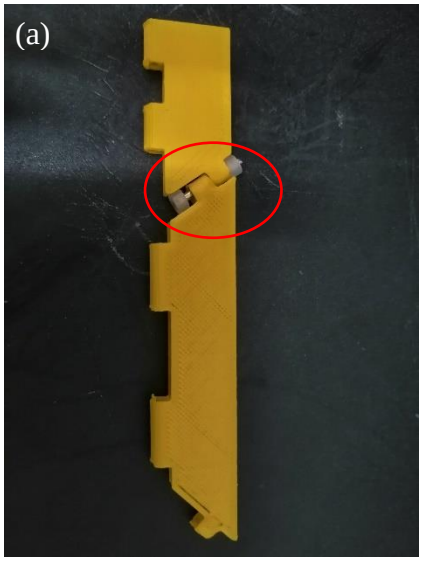


Figure 4.9: (a) Initial Design of Loops that Connect Segments Diagonally. (b) Segments Cannot Bend Completely. (c) Modified Loops. (d) Segments Can Bend Completely.

Moreover, the transmission mechanism has safeguard plates designed as shown in Figure 4.10 to ensure that the segments do not bend in the direction that is not intended. This prevents the transmission mechanism from actuating movements that violate the static constraints of the hand.

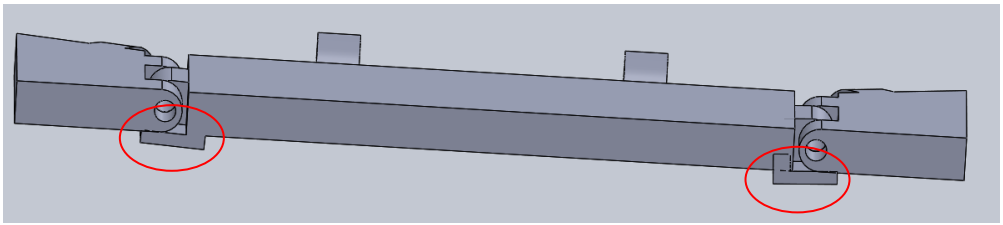


Figure 4.10: Safeguard Plates Design Feature.

In addition, the initial designs had the segments lie completely flat as shown in Figure 4.11 (a). This made the movements actuated jerky. Unlike paper that encodes the creases in its fibres when it is folded, this transmission mechanism does not remember which direction it should be folding towards. Even though there are safeguard plates too prevent the segments from bending the wrong direction, there was also no design to encourage it to bend in the correct direction. As such, the safeguard plates were later designed to extend outward with a 3° angle to allow the segments to "remember" which direction it should bend towards as shown in Figure 4.11 (b).

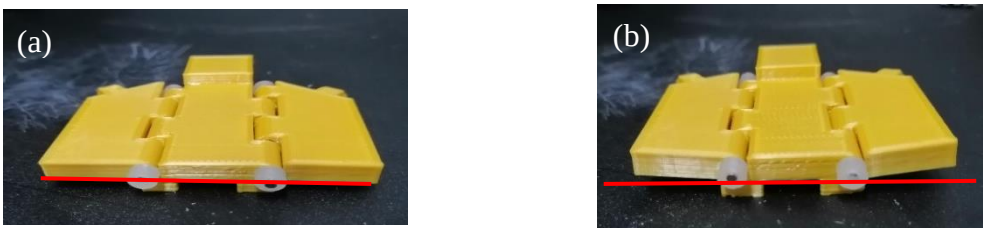


Figure 4.11: (a) Segments Lie Completely Flat. (b) Segments Do Not lie Completely Flat.

Furthermore, when the initial versions of the transmission mechanism were attached to the hand and attempted to actuate the joints, it was discovered that due to the weight of the hand, the forces transferred from the servo motors to the subsequent segments were not adequate to generate flexion at the MCP and PIP joints and extension at the radiocarpal joints. As such, two additional design features were added to aid the transmission mechanism. Elastic bands were attached between segments to exert elastic forces on the segments while extensions were attached on some segments to increase the length of the lever thus increasing the torque generated on the segments.

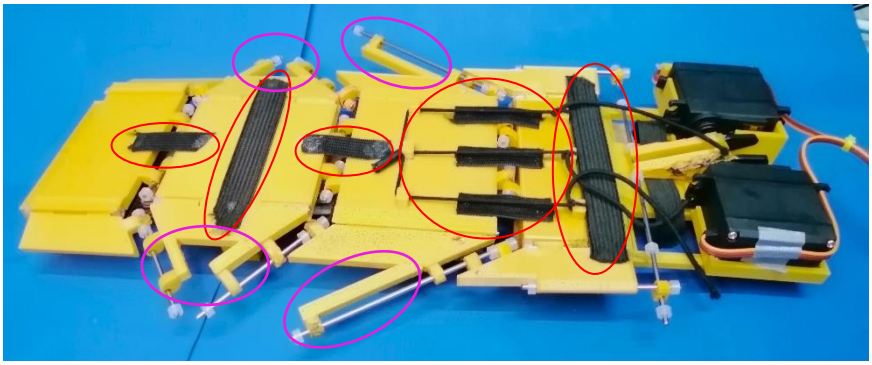


Figure 4.12: Elastic Bands (Circled in Red) and Extensions (Circled in Pink) Attached on Segments.

4.3.4 Code Design for Android Mobile Application Programme

The purpose of this code is to allow the user to select the rehabilitation mode and to send this information to the ESP32.

Figure 4.13 illustrates the different activities that the user navigates through to select the rehabilitation mode.

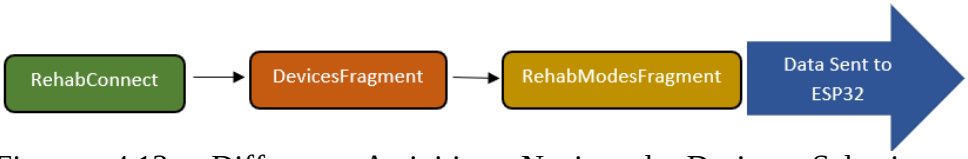


Figure 4.13: Different Activities Navigated During Selection of Rehabilitation Modes.

Table 4.1 summarizes the function of each activity.

Table 4.1: Activities Conducted to Select Rehabilitation Modes and their Functions.

Activity	Function
RehabConnect	Enable Bluetooth on user's mobile device.
DevicesFragment	Query the mobile device for paired devices and displays this list of devices for the user to select.
RehabModesFragment	Connect with the device selected by user. Then, displays 3 modes of rehabilitation for user to select. Once selected, this information is sent to ESP32 and user is notified that the rehabilitation has started. In addition, this activity listens to the status of the Bluetooth connection. If connection fails, it will attempt to reconnect with the ESP32 again.

The complete code annotated with comments can be found in Appendix B, C, and D.

4.3.5 Code Design for ESP32 Programme

The purpose of this code is to receive the rehabilitation mode sent by the Android mobile application and to control the servo motors.

Firstly, the header files that enable Bluetooth and servo motor control functions are imported. Then, a variable, "Mode" is initialised to store the mode of the rehabilitation activity. After that, the Bluetooth serial service is initialised. Next, the code will loop as it attempts to detect whether the android mobile application has sent any message. Once a message is detected, it is stored in the "Mode" variable to detect which rehabilitation mode was selected by the user. Table 4.1 shows the value that the "Mode" variable will contain corresponding with the rehabilitation mode chosen by user.

Table 4.2: Rehabilitation Mode Selected and Corresponding Data Received by ESP32.

Rehabilitation Mode Selected	Data Received by ESP32
Easy	1
Intermediate	2
Difficult	3

After detecting the rehabilitation mode, an if-else loop will be used to execute the correct rehabilitation mode. For all modes, the step angles are set to be the same at 1°. Moreover, the servo motor will also rotate from 0° to 42° for all modes. The only difference is the delay set between angle increments. The delay set for "Easy" mode is the shortest, followed by "Intermediate" mode and then "Difficult" mode. The complete code annotated with comments can be found in Appendix E.

4.4 Sensing System

4.4.1 Voltage Divider Circuit

When the flex sensors are bent away from the direction containing the conductive ink, their resistance will increase. To enable the ADC input pin of the ESP32 to detect a change in input voltage, a voltage divider circuit was constructed as shown in Figure 4.14 so that the ESP32 will be measuring the change of voltages across the 10 kΩ resistor. When the flex sensor is bent and its resistance increases, the voltage across the 10 kΩ resistor will drop thus the voltage received by the ESP32 will decrease.

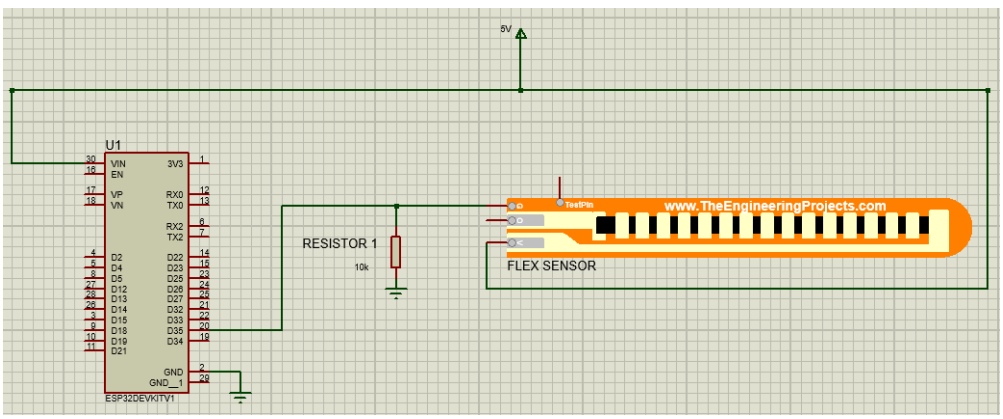


Figure 4.14: Voltage Divider Circuit for Sensing System.

4.4.2 Code Design for Android Mobile Application Programme

The purpose of this code is to allow the users to assess their recovery progress. Figure 4.15 illustrates the different activities that the user navigates through in order to access their recovery progress.

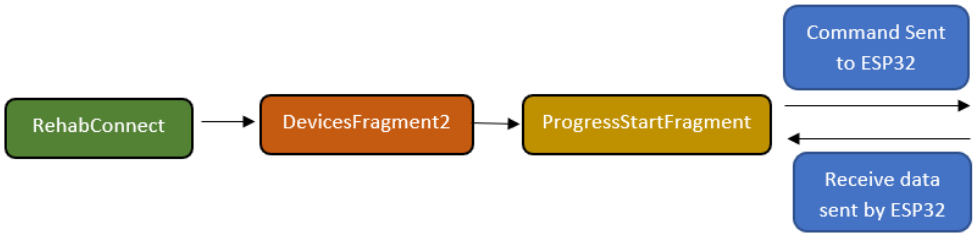


Figure 4.15: Different Activities Navigated for Users to Assess Recovery Progress.

Table 4.3 summarizes the function of each activity.

Table 4.3: Activities Conducted to Assess Recovery Progress.

Activity	Function
RehabConnect	Enable Bluetooth on user's mobile device.
DevicesFragment2	Query the mobile device for paired devices and displays this list of devices for the user to select.
ProgressStartFragment	Connect with the device selected by user. Then, sends a command to the ESP32 to begin joint angle measurement. Once the measurement process is completed, it receives the maximum flexion of MPC joint and maximum extension of radiocarpal joint values from ESP32 and displays this information to the user. In addition, this activity listens to the status of the Bluetooth connection. If the connection fails, it will attempt to reconnect with the ESP32 again.

The complete code annotated with comments can be found in Appendix B, F, and G.

4.4.3 Code Design for ESP32 Programme

The purpose of this code is to detect the digital input voltages provided by the flex sensors, process them into MCP flexion angles and radiocarpal extension angles, locate the maximum angles and subsequently transmit this data back to the Android mobile application.

Firstly, the header file that enables Bluetooth functionality in ESP32 was imported. Then, the true values for the voltage of the power source and resistance of the resistors used in the voltage divider circuit as well as the resistance of the flex sensors when they lie completely flat or are bent to 90° (for MCP joint flex sensor) and 60° (for radiocarpal joint flex sensor) was measured using a multi-meter. Next, these values are stored in variables.

When the programme detects a prompt from the Android application to start measuring the angles of the joints, the programme will detect the digital signal provided by the flex sensors. Then, these values will be converted into analog voltages and subsequently into the resistance of the distorted flex sensors. Next, these values are mapped onto the resistance range of flex sensors so that MCP flexion angles and radiocarpal extension angles can be obtained.

After that, these values are passed through an if-else loop to determine whether their values were larger than those in the previous iterations. This programme will run for 11 iterations and the maximum MCP flexion angle and radiocarpal extension angle obtained will be catenated and converted into unsigned character format where they can now be sent to the Android Application.

The complete code annotated with comments can be found in Appendix H.

4.4.4 Calibration of Flex Sensors

Once the code in section 4.5.3 was constructed, the values for the resistance of flex sensors when they lie completely flat or are bent to 90° (for MCP joint flex sensor) and 60° (for radiocarpal joint flex sensor) have to be calibrated to ensure that the calculated joint angles adhere as close to the true angles as possible. To do so, the following steps were taken:

i. The angles for 60° and 90° were drawn on paper.

ii. The sensing glove was worn.

iii. The hand was held at the position shown in Figure 4.16 to detect the resistance calculated by the ESP32 programme when MCP joint was flexed at 90°.

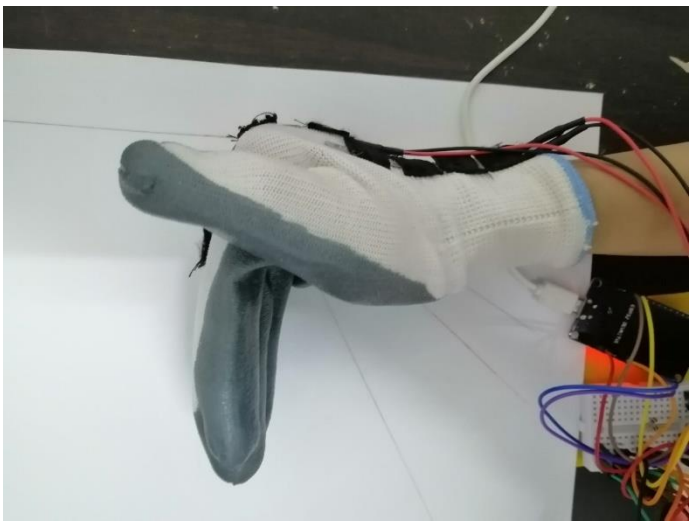


Figure 4.16: Hand Position Held for MCP Joint Flexion of 90°.

iv. The ESP32 programme is imitated via Android mobile application.

v. Once the measuring process was completed, the maximum value for “Resistance MCP” shown on the serial monitor of the Arduino Web Editor (Figure 4.17) was located among the iterations. This value represents the maximum resistance that was detected during the measuring process.

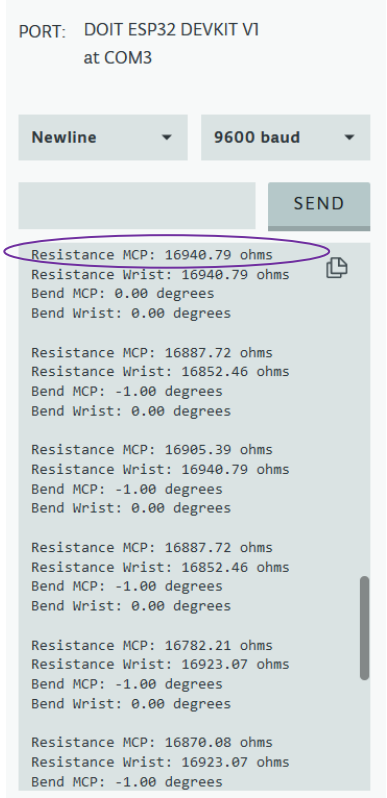


Figure 4.17: Iteration Values Shown on Serial Monitor.

vi. Steps iii to v were repeated to obtain that maximum resistance measured for 0° MCP joint flexion and radiocarpal joint extension as well as 60° radiocarpal joint extension. Step iii was modified to obtain the different angles, whereby the hand was held at the position shown in Figure 4.18 (a) to measure angles for 60° radiocarpal joint extension and held at the position shown in Figure 4.18 (b) to measure angles for 0° MCP joint flexion and radiocarpal joint extension.

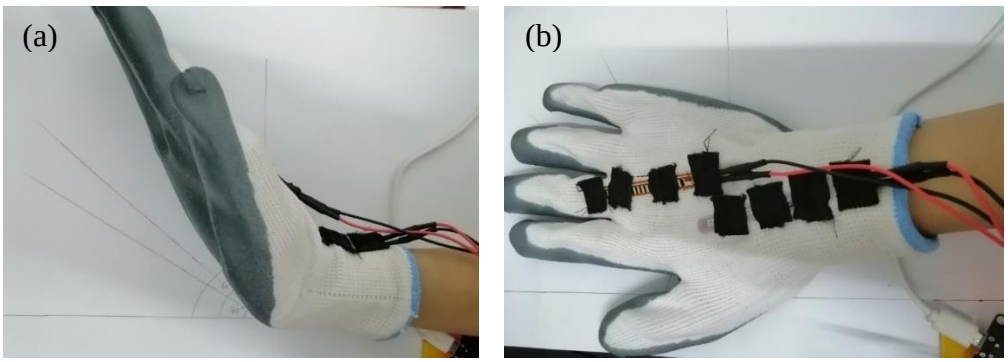


Figure 4.18: (a) Hand Position Held for Radiocarpal Joint Extension of 60°. (b) Hand Position Held for Radiocarpal Joint Extension and MCP Joint Flexion of 0°.

vii. Steps iii to vi was repeated to obtain 10 sets of values and which were tabled.

viii. The mean for these 10 sets of values were obtained and the values of the variables in the ESP32 programme were altered accordingly. (Table containing the 10 sets of values can be found in Appendix I)

4.5 Origami Tutorial Functionality

The purpose of this code was to display a list of tutorial videos to the user and enable the users to select and play the videos. For this functionality, the user interacts with only 1 activity: “OrigamiVideos”. The function of this activity is to embed HTML content from YouTube into WebView elements that the user can interact with.

The complete code annotated with comments can be found in Appendix J.

4.6 System Integration

4.6.1 Code Design for Android Mobile Application Programme

To integrate the 3 main systems / functionalities (transmission, sensing and origami tutorials) together, a menu page was added to enable the user to navigate to their preferred functionality. The “HomeDirectory” activity functions to display buttons for the users to click and navigate to activities that correspond to the user’s selection. As such, Figure 4.19 illustrates the different activities that the user navigates through in order to perform their chosen functionality.

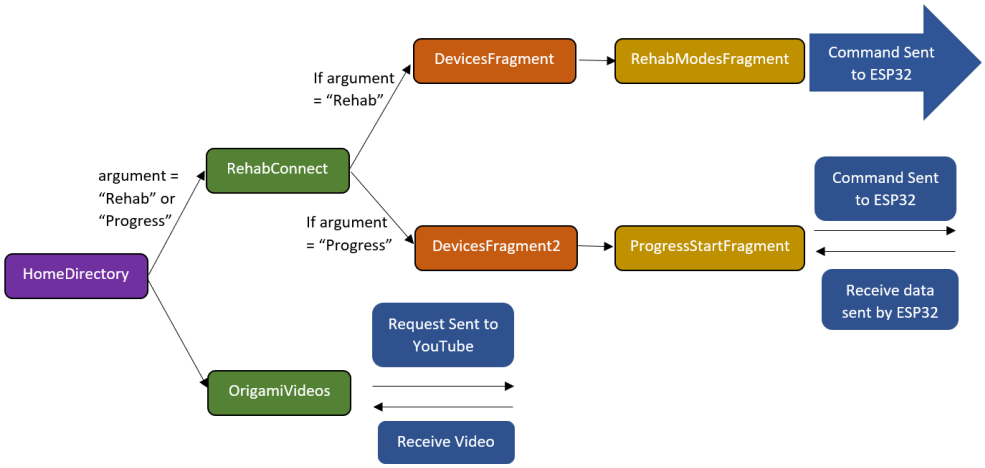


Figure 4.19: Complete Activity Flow in Android Application.

The complete code for the "HomeDirectory" activity that is annotated with comments can be found in Appendix K.

4.6.2 Code Design for ESP32 Programme

To integrate the transmission and sensing systems together, an infinite loop is created to constantly check whether the Android application has sent a command to the ESP32. When the android application sends the first command which contains information about which system was chosen by the user, the ESP32 programme passes this command to an if-else loop to trigger the correct system. The complete code for the ESP32 programme can be found in Appendix L.

4.7 Summary

To summarise, transmission mechanism was designed using origami theory. Furthermore, this transmission mechanism was designed to actuate the MCP, PIP and radiocarpal joints together and contains special design features that help to optimise its movements.

In addition, the sensing system utilised voltage divider circuit method for its design. This system was also calibrated to increase the accuracy of its measurements.

Moreover, the Android mobile application programme was designed to cover the transmission system, sensing system and origami tutorials

functionalities while the ESP32 programme was design to cover the transmission system and sensing system functionalities. Once the systems were completed, they were integrated to enable the user to select their preferred activity.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 Introduction

This chapter will assess whether the study has achieved its goals by displaying and evaluating the results produced for the analysis of the following areas: transmission system, sensing system, user interface, total cost of rehabilitation system and total weight of transmission mechanism attached to the hand.

5.2 Transmission System

5.2.1 Angles Generated

Figure 5.1, 5.2 and 5.3 show the angles generated at the joints within a timeframe of 60 seconds for the “Easy”, “Intermediate” and “Difficult” modes respectively.

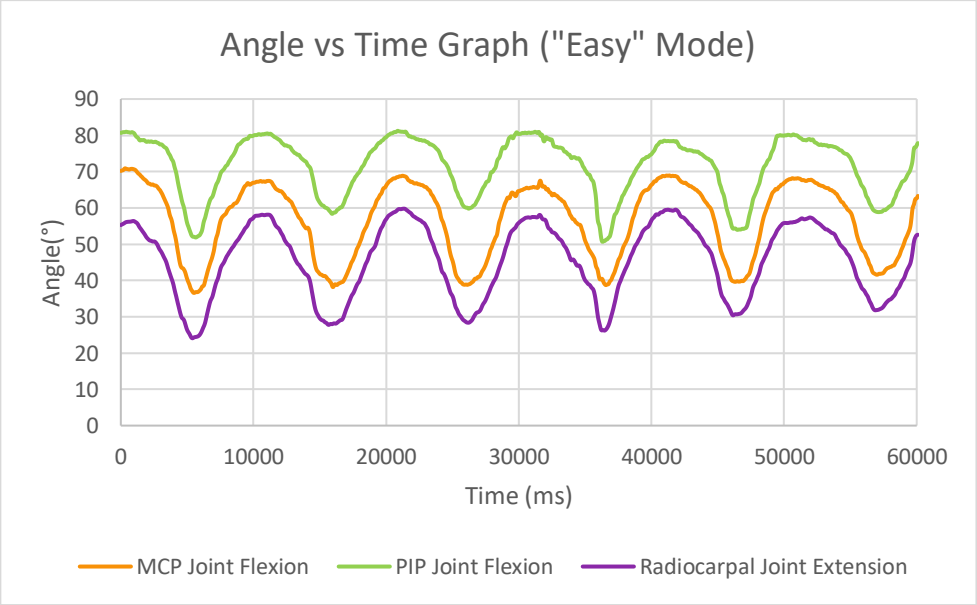


Figure 5.1: Angle vs Time graph for “Easy” Mode.

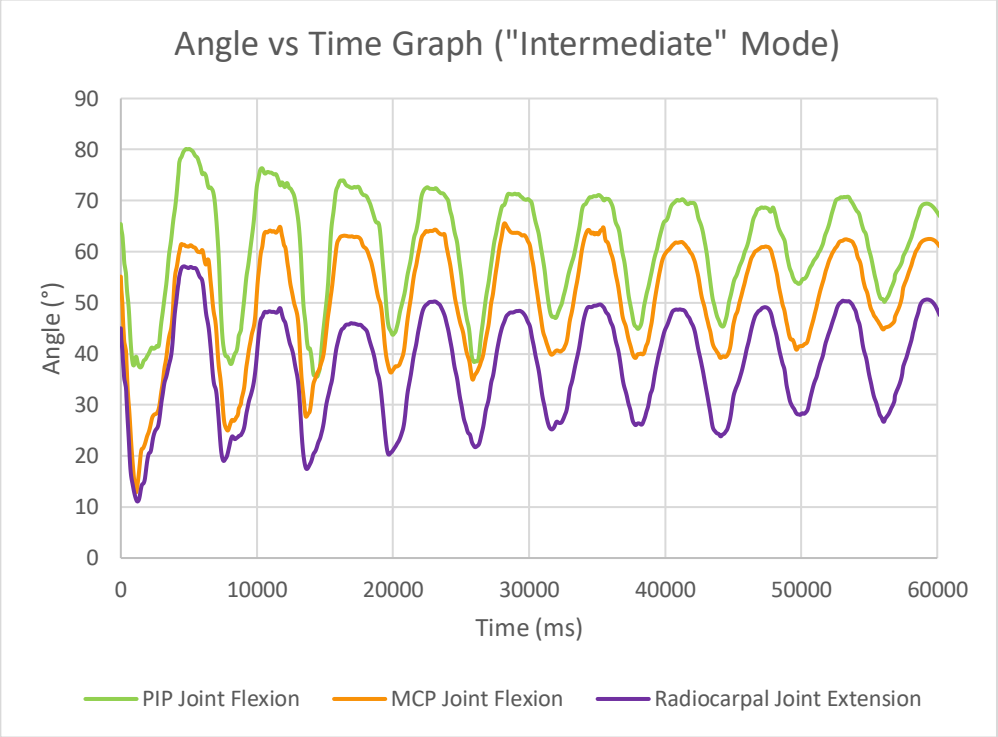


Figure 5.2: Angle vs Time graph for “Intermediate” Mode.

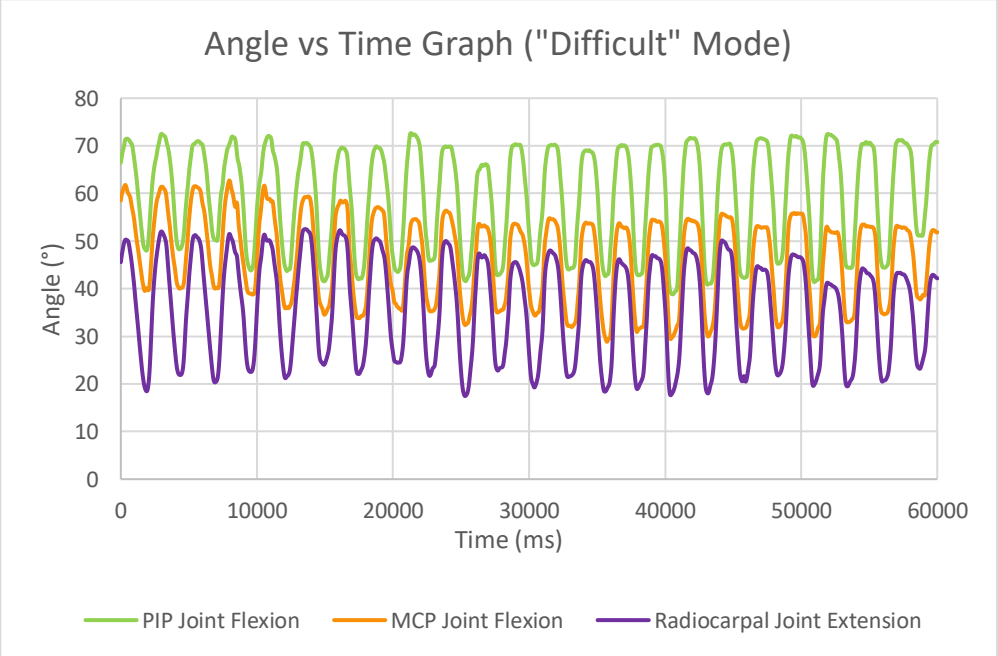


Figure 5.3: Angle vs Time graph for “Difficult” Mode.

From Figure 5.1, 5.2 and 5.3, it is observed that all modes were able to produce flexion and extension at the MCP and PIP joints and extension at the radiocarpal joint. In addition, the “Difficult” mode generated the most

flexion-extension cycles followed by the “Intermediate” mode and then the “Easy” mode. This proves than the transmission mechanism can actuate the fastest for the “Difficult” mode followed by the “Intermediate” mode and then the “Easy” mode, thus allowing the user to select which mode they are comfortable with. Moreover, it can be seen that the transmission mechanism can produce angular displacements with a range of 25° to 30°.

Using the “Max” function from Microsoft Excel, the maximum flexion or extension angles for the joints were calculated as shown in Table 5.1.

Table 5.1: Maximum Flexion / Extension Angles Generated for Three Different Rehabilitation Modes.

Rehabilitation Mode	**Maximum Flexion Angle for PIP Joint**	**Maximum Flexion Angle for MCP Joint**	**Maximum Extension Angle for Radiocarpal Joint**
Easy	81°	71°	60°
Intermediate	80°	66°	57°
Difficult	73°	63°	53°

From Table 5.1, the maximum flexion angles generated at the PIP joints were less than 110° for all rehabilitation modes. Furthermore, the maximum flexion angles generated at the MCP joints were less than 90° for all rehabilitation modes. Lastly, the maximum extension angles generated at the Radiocarpal joints were 60° and below for all rehabilitation modes. Therefore, the transmission mechanism did not exceed the static constraints of the hand.

5.2.2 Angular Velocities Generated

Figure 5.4, 5.5 and 5.6 show the angular velocities generated at the joints within a timeframe of 60 seconds for the “Easy”, “Intermediate” and “Difficult” modes respectively.

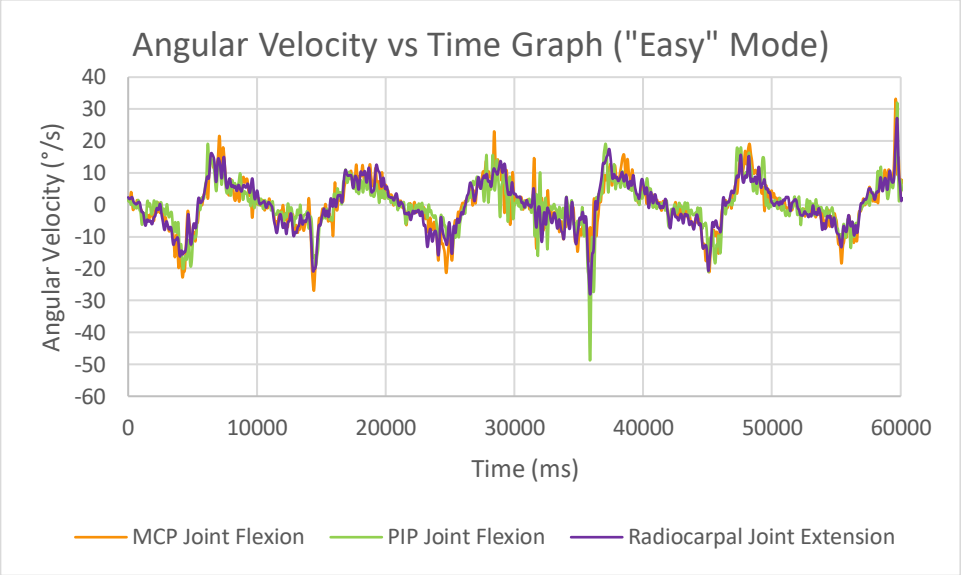


Figure 5.4: Angular Velocity vs Time graph for “Easy” Mode.

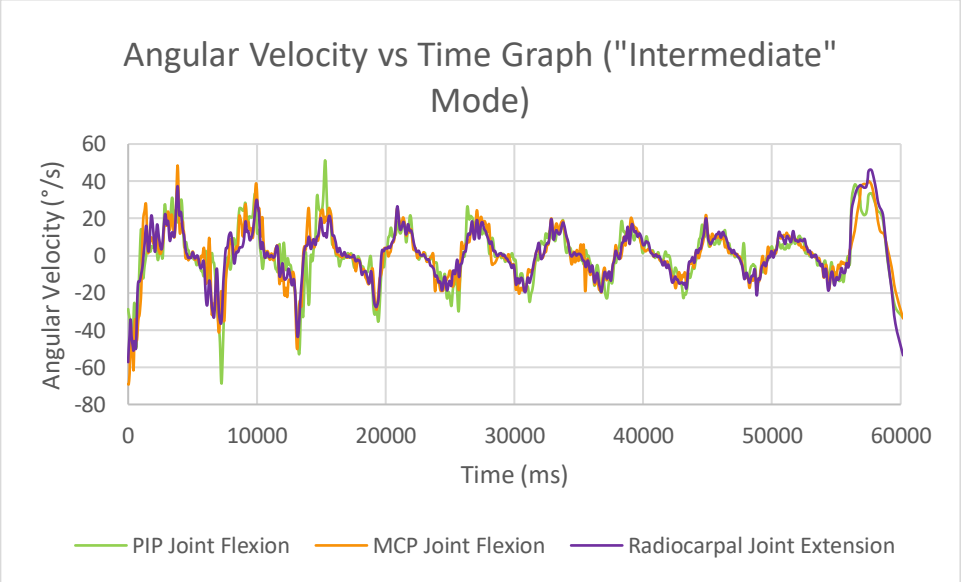


Figure 5.5: Angular Velocity vs Time graph for “Intermediate” Mode.

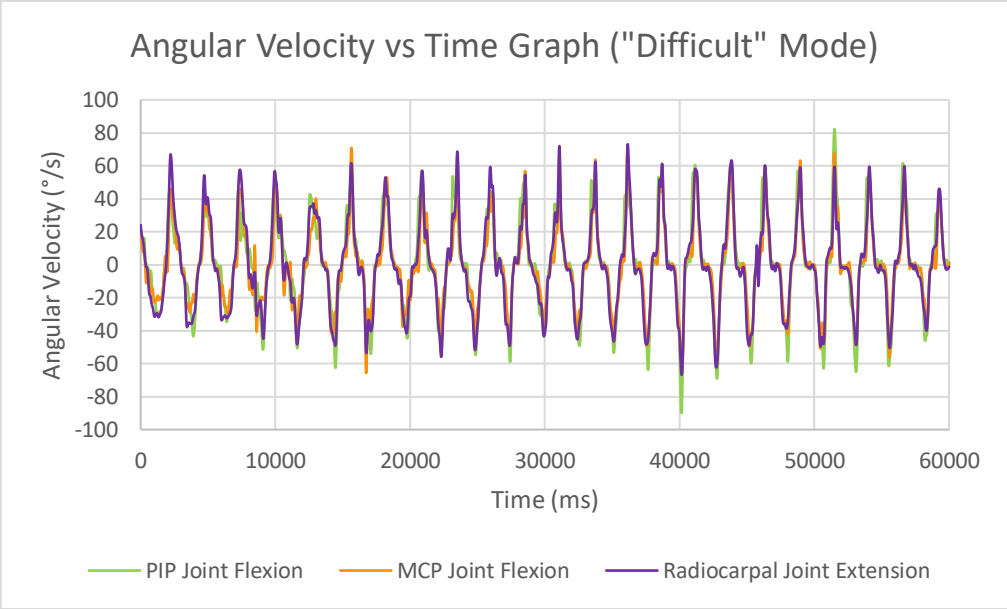


Figure 5.6: Angular Velocity vs Time graph for "Difficult" Mode.

From Figure 5.4, 5.5 and 5.6, it can be observed that the angular velocities generated at the joints for the "Difficult" mode is the highest, followed by "Intermediate" mode and then "Easy" mode.

Using the "ABS" function from Microsoft Excel, the angular velocities were converted into absolute values. Then, using the "Max" function, the maximum angular velocity magnitudes for the joints were calculated as shown in Table 5.2.

Table 5.2: Maximum Flexion / Extension Angular Velocity Magnitudes Generated for Three Different Rehabilitation Modes.

Rehabilitation Mode	**Maximum Flexion Angular Velocity Magnitude for PIP Joint**	**Maximum Flexion Angular Velocity Magnitude for MCP Joint**	**Maximum Extension Angular Velocity Magnitude for Radiocarpal Joint**
Easy	49°/s	33°/s	28°/s
Intermediate	69°/s	69°/s	57°/s
Difficult	90°/s	71°/s	73°/s

From Table 5.2, the maximum magnitude for angular velocities generated at the PIP joints were less than 858°/s for all rehabilitation modes. Furthermore, the maximum magnitude for angular velocities generated at the MCP joints were less than 694°/s for all rehabilitation modes. Lastly, the maximum magnitude for angular velocities generated at the Radiocarpal joints were less than 180°/s for all rehabilitation modes. Therefore, the transmission mechanism generates movements that did not exceed the maximum angular velocities that can be produced naturally.

To summarise, this study managed to design and construct a transmission mechanism that could generate flexion and extension at the MCP and PIP joins and extension at the radiocarpal joint. In addition, this transmission mechanism actuates movements that respect the static constraints of the hand and does not exceed the maximum angular velocities that can be generated by the hand naturally. Moreover, the "Difficult" mode of the transmission mechanism is capable of generating the highest magnitude of angular velocities, followed by "Intermediate" mode and then "Easy" mode.

5.3 Sensing System

Table 5.3 shows the maximum MCP flexion angles and maximum radiocarpal extension angles detected by the sensing system for 10 trials.

Table 5.3: Maximum MCP Flexion Angles and Maximum Radiocarpal Extension Angles Detected.

Trials	**Maximum MCP Flexion Angle**				**Maximum Radiocarpal Extension Angle**		
	0°	**30°**	**60°**	**90°**	**0°**	**30°**	**60°**
1	8°	35°	53°	95°	9°	34°	58°
2	7°	36°	56°	89°	8°	28°	59°
3	5°	30°	58°	91°	8°	26°	53°
4	7°	28°	59°	87°	9°	27°	59°
5	4°	30°	55°	94°	10°	33°	55°
6	5°	36°	52°	90°	8°	30°	55°

7	7°	35°	56°	84°	5°	30°	57°
8	7°	35°	56°	88°	7°	29°	55°
9	6°	36°	57°	87°	7°	35°	65°
10	3°	29°	55°	93°	8°	32°	64°

Figure 5.7 and 5.8 represent the boxplots created to visualise the distribution of sample values obtained in Table 5.3.

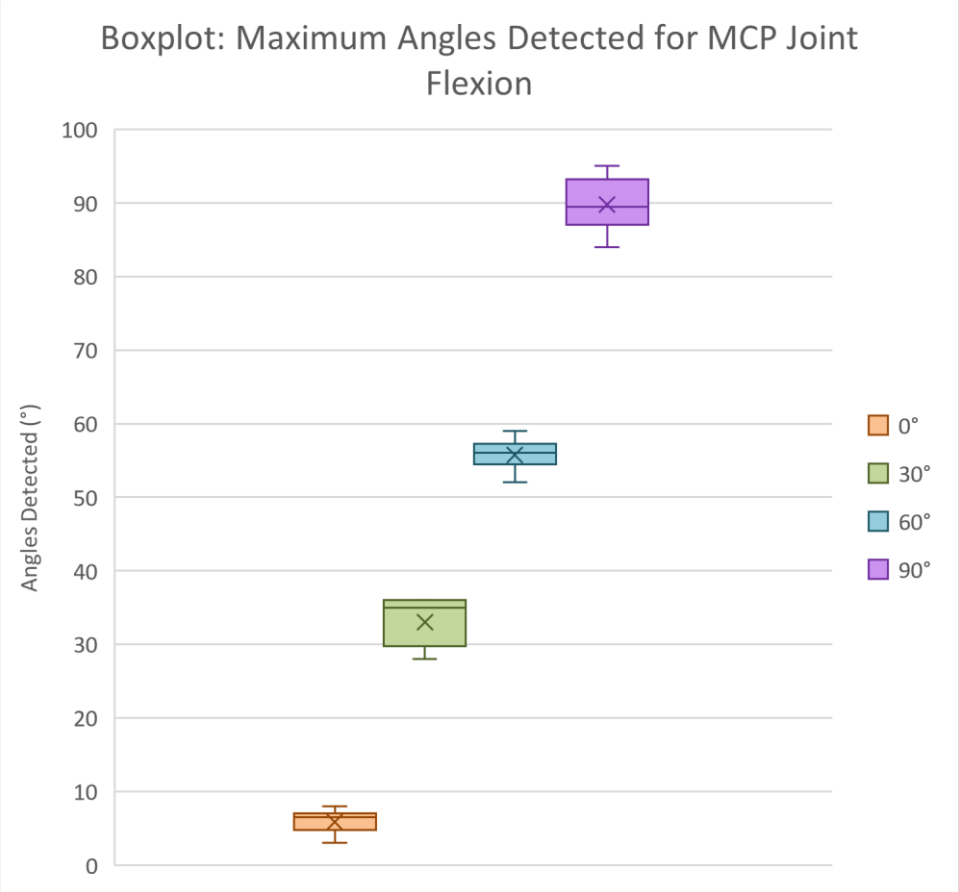


Figure 5.7: Boxplot: Maximum Angles Detected for MCP Joint Flexion.

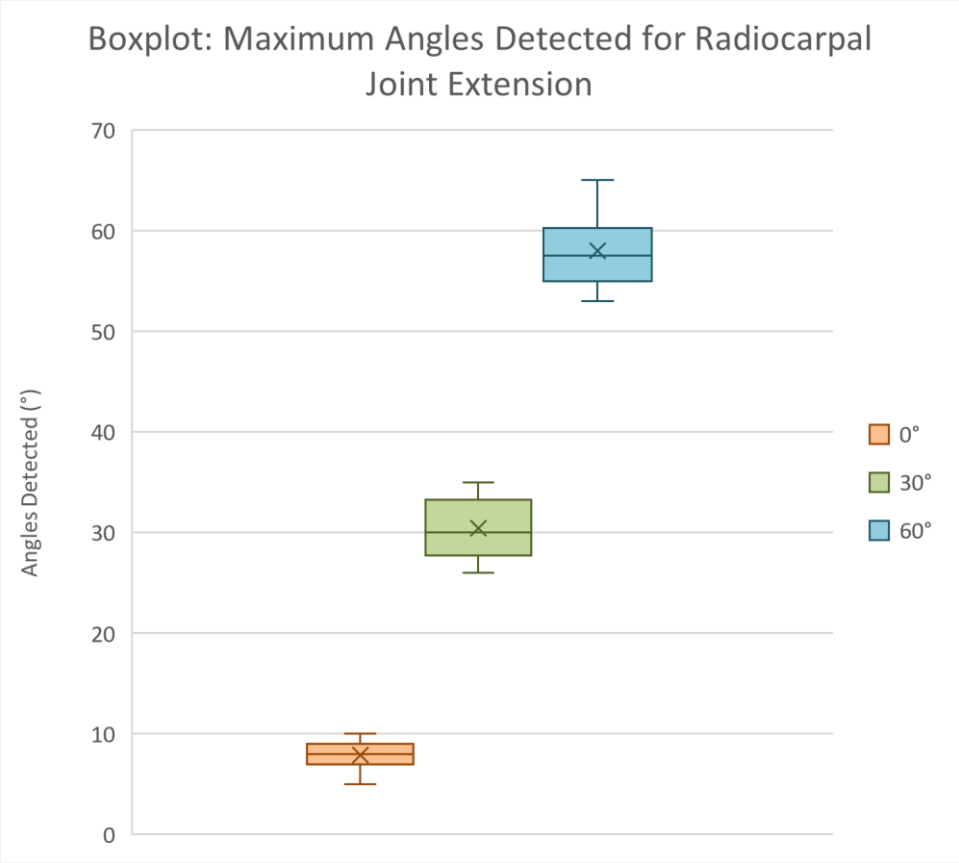


Figure 5.8: Boxplot: Maximum Angles Detected for Radiocarpal Joint Extension.

Referring to Figures 5.7 and 5.8, there are no outliers found in the measured values. Moreover, the sample values for maximum angles detected for MCP joint flexion of 60° and 90° as well as the sample values for maximum angles detected for radiocarpal joint extension of 0° display a symmetric distribution. Furthermore, the sample values for maximum angles detected for radiocarpal joint extension of 30° and 60° are slightly positively skewed while the sample values for maximum angles detected for MCP joint flexion of 0° and 30° are negatively skewed. In addition, the dispersion of sample values for maximum angles detected for MCP joint flexion of 30° and 90° and for maximum angles detected for radiocarpal joint extension of 30° and 60° are larger compared to the other samples.

Table 5.4 shows the mean values for the maximum MCP flexion angles and maximum radiocarpal extension angles detected by the sensing

system along with the percentage difference between these values and the actual MCP flexion and radiocarpal extension produced.

Table 5.4: Mean Values and Mean Differences for Maximum MCP Flexion Angles and Maximum Radiocarpal Extension Angles Detected.

	Maximum MCP Flexion Angle				**Maximum Radiocarpal Extension Angle**		
Actual Angles	**0°**	**30°**	**60°**	**90°**	**0°**	**30°**	**60°**
Mean for Angles Measured	5.9°	33°	55.7°	89.8°	7.9°	30.4°	58°
Mean Difference Between Measured and Actual Angles	5.9°	3°	-4.3°	-0.2°	7.9°	0.4°	-2°

Referring to Table 5.4, the mean differences between the measured and actual angles for MCP joint flexion and radiocarpal joint extension fall below ±5° for all flexion and extension angles except those for 0°. Therefore, excluding the joint measurements for 0°, the accuracy of this sensing system is similar to the results obtained in (Williams et al., 2000) where the maximum mean difference was 5.10°. The joint measurements for 0° exceeded a mean difference of ±5 because the gloves used in the sensing system was large and not formfitting. As such, the excess material tends to distort the flex sensors especially when the joint flexion or extension angles are really small. Despite this issue, this study has managed to design and construct a working sensing system.

5.4 User Interface

5.4.1 Activity Flow

5.4.1.1 Navigation

Android users can access the mobile application by selecting the launcher icon shown in Figure 5.9 (a). Once the application loads, users can navigate to the different activities from the navigation page shown in Figure 5.9 (b).

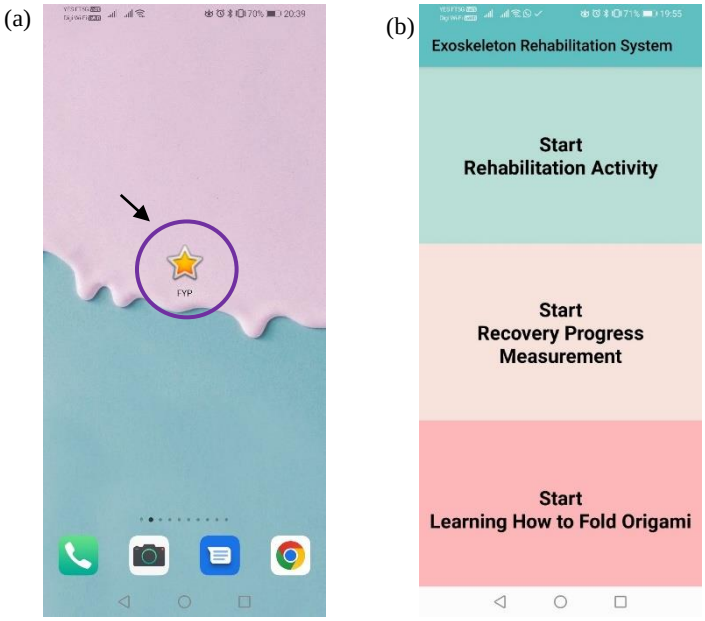


Figure 5.9: (a) Launcher Icon (b) Navigation Page

5.4.1.2 Hand Rehabilitation Activity

When users select the button shown in Figure 5.10 (a), the users will navigate to the page shown in Figure 5.10 (b). Here, they will be prompted to provide their permission to allow the application to enable the Bluetooth feature on their mobile devices. If they select “Allow”, they will navigate to the page shown in Figure 5.10 (c) which displays the list of external devices that are paired with the mobile phone. Here, the users can select “ESP32_Control” to connect with the ESP32 microcontroller. They will then be navigated to the page that will display the status of the Bluetooth connection. Figure 5.10 (d) is the status shown when the application is in the attempting to connect with ESP32. If the ESP32 is not turned on, status shown in Figure 5.10 (e) will be displayed to inform the user that the attempt has failed. The application will

loop between status shown in Figure 5.10 (d) and Figure 5.10 (e) until the ESP32 is turned on.

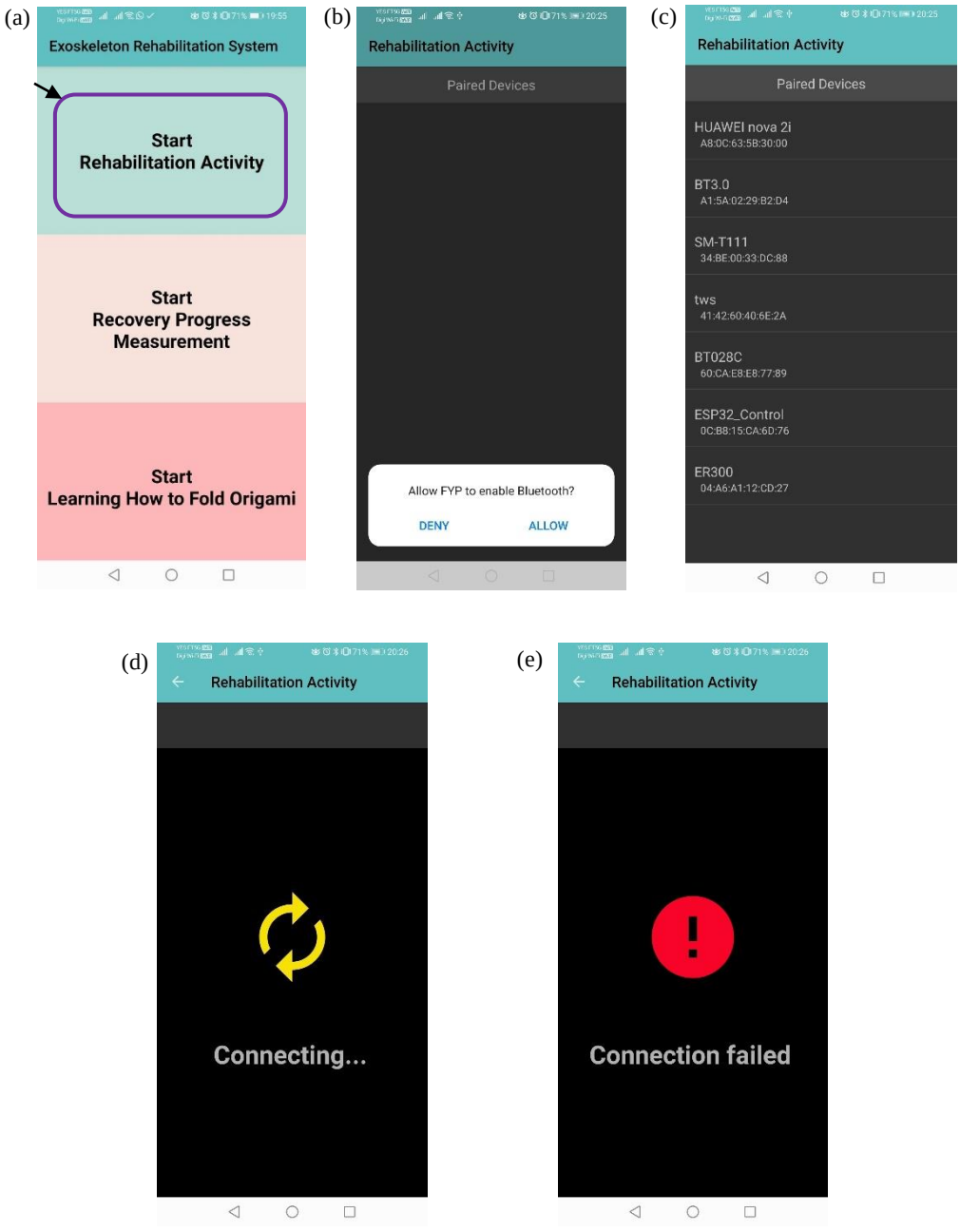
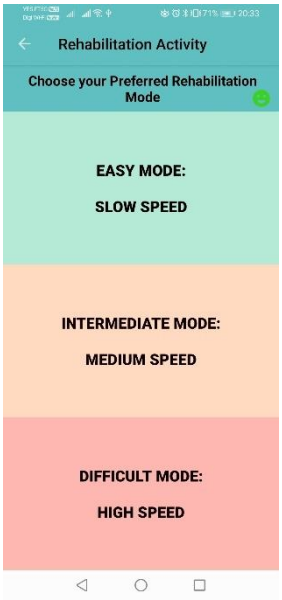


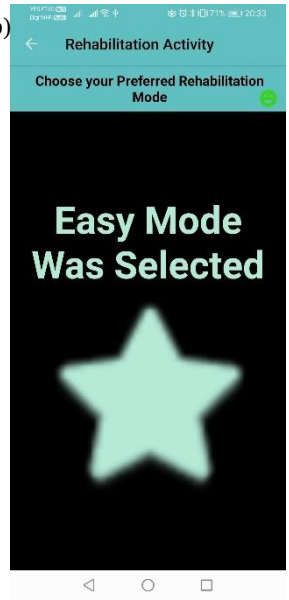
Figure 5.10: (a) “Start Rehabilitation Activity” Button is Selected. (b) Prompt Displayed to Enable Bluetooth. (c) List of Paired Devices is Displayed. (d) Status Shown as Application is Attempting to Connect with ESP32. (e) Status Shown When Application Fails to Connect with ESP32.

Once the ESP32 is turned on, the application will successfully connect with it and navigate to the page shown in Figure 5.11 (a). If the “Easy” mode is selected, the application will navigate to the page shown in Figure 5.11 (b) and the transmission mechanism will start to actuate. If the “Intermediate” mode is selected, the application will navigate to the page shown in Figure 5.11 (c) and the transmission mechanism will start to actuate. If the “Difficult” mode is selected, the application will navigate to the page shown in Figure 5.11 (d) and the transmission mechanism will start to actuate.

(a)



(b)



(c)

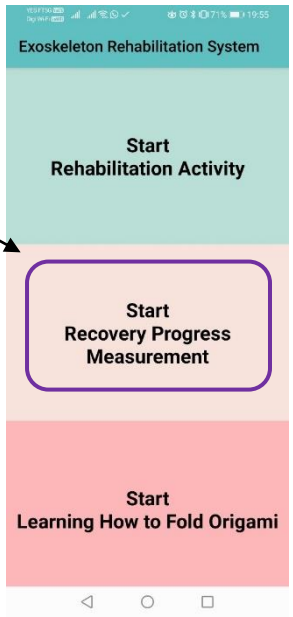
(d)

Figure 5.11: (a) Three Different Rehabilitation Modes Displayed. (b) Page Shown When “Easy” Mode was Selected. (c) Page Shown When “Intermediate” Mode was Selected. (d) Page Shown When “Difficult” Mode was Selected.

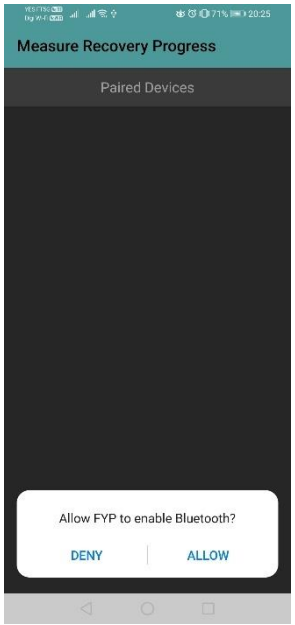
5.4.1.3 Recovery Progress Measurement Activity

When users select the button shown in Figure 5.12 (a), the users will navigate to the page shown in Figure 5.12 (b). Here, they will be prompted to provide their permission to allow the application to enable the Bluetooth feature on their mobile devices. If they select “Allow”, they will navigate to the page shown in Figure 5.12 (c) which displays the list of external devices that are paired with the mobile phone. Here, the users can select “ESP32_Control” to connect with the ESP32 microcontroller. They will then be navigated to the page that will display the status of the Bluetooth connection. Figure 5.12 (d) is the status shown when the application is in the attempting to connect with ESP32. If the ESP32 is not turned on, status shown in Figure 5.12 (e) will be displayed to inform the user that the attempt has failed. The application will loop between status shown in Figure 5.12 (d) and Figure 5.12 (e) until the ESP32 is turned on.

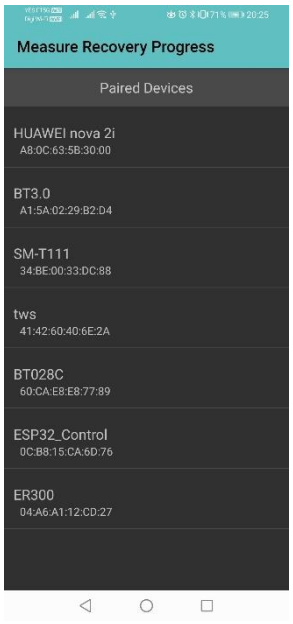
(a)



(b)



(c)



(d)

(e)

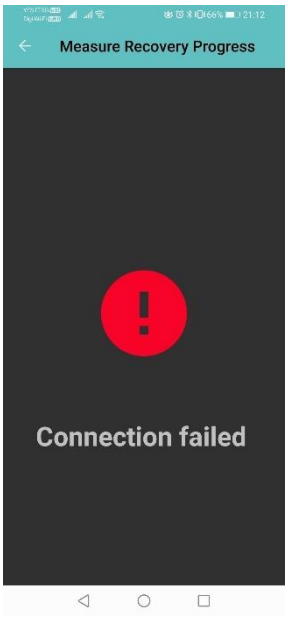


Figure 5.12: (a) “Start Recovery Progress Measurement” Button is Selected. (b) Prompt Displayed to Enable Bluetooth. (c) List of Paired Devices is Displayed. (d) Status Shown as Application is Attempting to Connect with ESP32. (e) Status Shown When Application Fails to Connect with ESP32.

Once the ESP32 is turned on, the application will successfully connect with it and navigate to the page shown in Figure 5.13 (a). Users can now move their hand has donned on sensing glove. Once the detection cycle is complete, the maximum flexion angle at the MCP joint of the middle finger and the maximum extension angle of the radiocarpal joint are displayed.

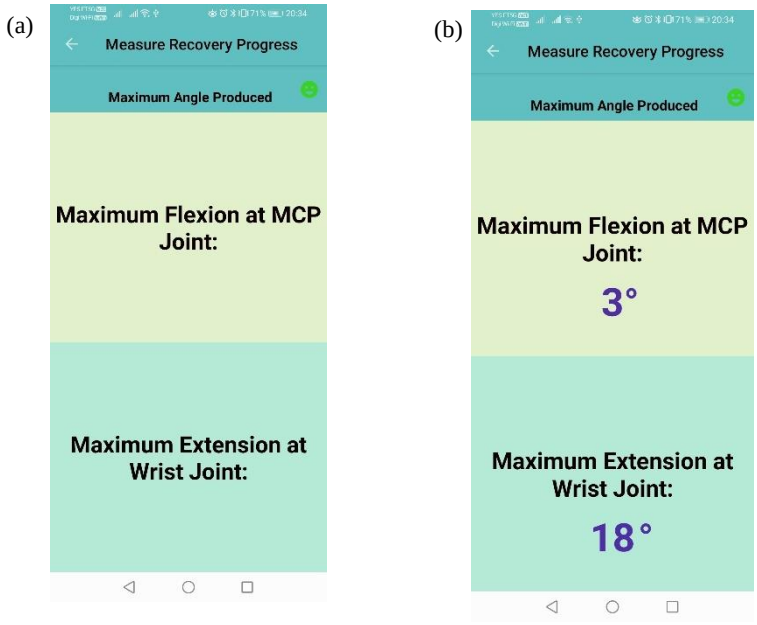


Figure 5.13: (a) Page Shown When the Sensing System is Measuring the Joint Angles. (b) Maximum Flexion Angle at MCP Joint and Maximum Extension Angle at Radiocarpal Joint Displayed on the Page.

5.4.1.4 Origami Tutorials Activity

When users select the button shown in Figure 5.14 (a), the users will navigate to the page shown in Figure 5.14 (b). When users select the video that they wish to view, the video will start to play as shown in Figure 5.15 (c).

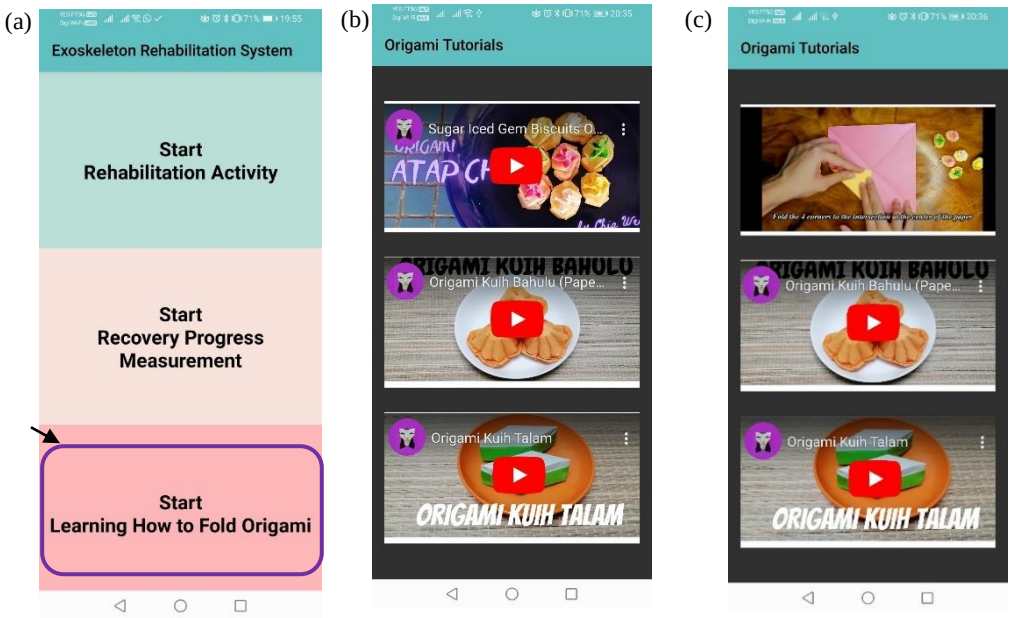


Figure 5.14: (a) “Start Learning How to Fold Origami” Button is Selected. (b) Page Displaying a List of Origami Tutorials. (c) Tutorial Video Runs When It is Selected.

Therefore, this study managed to design and construct a fully functional user interface that can connect to the ESP32 via Bluetooth, allow users to select between different activities, allow users to select between different rehabilitation modes, allow users to see their recovery progress as well as view tutorial videos on how to fold origami.

5.5 Total Cost of Rehabilitation System

Referring to the expenditure table constructed in Section 3.7, total cost of the rehabilitation system was RM 533.70. Therefore, this study managed to construct the entire rehabilitation system for less than RM 1500.

5.6 Total Weight of Transmission Mechanism Attached to Hand

The total weight of the transmission mechanism segments that are attached to the hand was 250 g. Therefore, this study managed to construct a transmission mechanism that weighed less than 500 g.

5.7 Summary

This study managed to design and construct a transmission mechanism that could generate flexion and extension at the MCP and PIP joins and extension at the radiocarpal joint. In addition, this transmission mechanism actuates movements that respect the static constraints of the hand and does not exceed the maximum angular velocities that can be generated by the hand naturally. Moreover, the “Difficult” mode of the transmission mechanism is capable of generating the highest magnitude of angular velocities, followed by “Intermediate” mode and then “Easy” mode.

Furthermore, this study managed to design and construct a working sensing system that can detect most angles with an accuracy on par with other studies.

Other than that, this study managed to design and construct a fully functional user interface that can connect to the ESP32 via Bluetooth, allow users to select between different activities, allow users to select between different rehabilitation modes, allow users to see their recovery progress as well as view tutorial videos on how to fold origami.

Next, this study managed to construct the entire rehabilitation system for RM 533.70, which is less than RM 1500. Lastly, this study managed to construct a transmission mechanism that weighed 250 g, which is less than 500 g.

In conclusion, this study managed to fulfil all 5 objectives set.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

To conclude, the aim of this study was to design a hand rehabilitation exoskeleton system that could provide continuous passive movement to the finger and wrist joints and allow the patient to choose between different rehabilitation protocols as well as to review their recovery progress. After testing and evaluating the system constructed, it was determined that the transmission system could generate 3 different angular velocities for the extension movement at the radiocarpal joint and coupled flexion and extension movements at the Metacarpophalangeal (MCP) and Proximal Interphalangeal (PIP) joints of the 4 fingers. In addition, these movements respect the static constraints of the hand and does not exceed the maximum angular velocities that can be generated by the hand naturally. Moreover, the sensing system could measure the angle of flexion at the MCP joint and maximum angle of extension at the wrist joint with an accuracy similar to that in the (Williams et al., 2000) study for all angles except 0°. Furthermore, the total expenditure for the construction of the rehabilitation system was RM 533.70 and the sections of transmission mechanism attached to the hand weighed 250 g in total. Lastly, the Android mobile application was successfully constructed to allow the patient to choose between 3 levels of angular velocity that is generated by the transmission mechanism and allows the patient view their recovery progress. As such, all the 5 objectives set in this study was met.

One of the limitations of this study is that the torque generated on the segments in the transmission mechanism are only sufficient to produce angular displacements with a range of 25° to 30°. In addition, the accuracy of the sensing system when measuring joint flexion or extension angles at 0° falls below that of a comparable study.

This study has provided insight on designing a hand exoskeleton rehabilitation system that is cost-effective and allows users to interact with it

via mobile phone. In addition, this study has proven that it is possible to integrate origami string theory into designing transmission mechanism for rehabilitation systems.

6.2 Recommendations for future work

In the future, the design of the transmission mechanism should be improved to increase the torque that can be generated at each of the segments in order to enable the actuation of a wider range of motions for the joints. One idea is to increase the width of the segments that are located lateral to the hand

In addition, the sensing system should be improved by using a formfitting glove to prevent the excess material from distorting the flex sensors. Moreover, the sensing system should be calibrated by positioning the joints using a goniometer instead of a paper that has angles drawn to increase its accuracy.

Lastly, a cloud database could be utilised in the future to record the patient's previous maximum MCP flexion and maximum radiocarpal extension angles. This will allow the user to keep track of their recovery progress throughout the entire rehabilitation journey.

REFERENCES

Ahmed, T., Assad-Uz-zaman, M., Islam, M.R., Gottheardt, D., McGonigle, E., Brahmi, B. and Rahman, M.H., 2021. Flexohand: A hybrid exoskeleton-based novel hand rehabilitation device. *Micromachines*, 12(11). https://doi.org/10.3390/mi12111274.

American Society for Surgery of the Hand, 2022. *Anatomy*. [online] Available at: <https://www.assh.org/handcare/anatomy> [Accessed 11 April 2022].

American Stroke Association, 2019. *Spasticity After Stroke*. [online] Available at: <https://www.stroke.org/-/media/stroke-files/lets-talk-about-stroke/life-after-stroke/ltas_spasticity_english_0419.pdf> [Accessed 3 March 2022].

Ates, S., Haarman, C.J.W. and Stienen, A.H.A., 2017. SCRIPT passive orthosis: design of interactive hand and wrist exoskeleton for rehabilitation at home after stroke. *Autonomous Robots*, 41(3), pp.711–723. https://doi.org/10.1007/s10514-016-9589-6.

Bae, J.H., 2013. The Effects of Origami on the Improvement of Hand Dexterity. *Journal of International Academy of Physical Therapy Research*, 4(2), pp.588–594. https://doi.org/10.5854/JIAPTR.2013.10.25.588.

Chen Chen, F., Appendino, S., Battezzato, A., Favetto, A., Mousavi, M. and Pescarmona, F., 2013a. Constraint Study for a Hand Exoskeleton: Human Hand Kinematics and Dynamics. *Journal of Robotics*, 2013. https://doi.org/10.1155/2013/910961.

Chen Chen, F., Appendino, S., Battezzato, A., Favetto, A., Mousavi, M. and Pescarmona, F., 2013b. Constraint study for a hand exoskeleton: Human

hand kinematics and dynamics. *Journal of Robotics*, 2013. https://doi.org/10.1155/2013/910961.

Chen, F.C., Favetto, A., Mousavi, M., Ambrosio, E.P., Appendino, S., Battezzato, A., Manfredi, D., Pescarmona, F. and Bona, B., 2011. Human Hand: Kinematics, Statics and Dynamics. In: *41st International Conference on Environmental Systems 2011, ICES 2011*. https://doi.org/10.2514/6.2011-5249.

Decker, M. and Kim, Y., n.d. *A Hand Exoskeleton Device for Robot Assisted Sensory-Motor Training after Stroke*.

Feigin, V.L., Brainin, M., Norrving, B., Martins, S., Sacco, R.L., Hacke, W., Fisher, M., Pandian, J. and Lindsay, P., 2022. World Stroke Organization (WSO): Global Stroke Fact Sheet 2022. *International Journal of Stroke*, 17(1), pp.18–29. https://doi.org/10.1177/17474930211065917.

Jo, I., Park, Y., Lee, J. and Bae, J., 2019. A portable and spring-guided hand exoskeleton for exercising flexion/extension of the fingers. *Mechanism and Machine Theory*, 135, pp.176–191. https://doi.org/10.1016/j.mechmachtheory.2019.02.004.

Kang, B.B., Choi, H., Lee, H. and Cho, K.J., 2019. Exo-Glove Poly II: A Polymer-Based Soft Wearable Robot for the Hand with a Tendon-Driven Actuation System. *Soft Robotics*, 6(2), pp.214–227. https://doi.org/10.1089/soro.2018.0006.

Kuo, C.-L. and Hu, G.-C., 2018. Post-stroke Spasticity: A Review of Epidemiology, Pathophysiology, and Treatments. *International Journal of Gerontology*, 12(4), pp.280–284. https://doi.org/10.1016/j.ijge.2018.05.005.

Liu, C., Wohlever, S.J., Ou, M.B., Padir, T. and Felton, S.M., 2021. Shake and Take: Fast Transformation of an Origami Gripper. [online] https://doi.org/10.1109/TRO.2021.

M Wilson, L., W Roden, P., Taylor, Y. and Marston, L., 2008. The Effectiveness of Origami on Overall Hand Function After Injury: A Pilot Controlled Trial. *The British Journal of Hand Therapy*, 13(1), pp.12–20. https://doi.org/10.1177/175899830801300102.

Monaghan, K., Horgan, F., Blake, C., Cornall, C., Hickey, P.P., Lyons, B.E. and Langhorne, P., 2011. Physical treatment interventions for managing spasticity after stroke. In: *Cochrane Database of Systematic Reviews*. John Wiley & Sons, Ltd. https://doi.org/10.1002/14651858.cd009188.

Palmer, A.K., Werner, F.W., Murphy, D. and Glisson, R., 1985. Functional wrist motion: A biomechanical study. *Journal of Hand Surgery*, 10(1), pp.39–46. https://doi.org/10.1016/S0363-5023(85)80246-X.

Rackley, S.A., 2011. *Wireless networking technology: From principles to successful implementation.*

Rahman, A. and Al-Jumaily, A., 2013. Design and Development of a Bilateral Therapeutic Hand Device for Stroke Rehabilitation. *International Journal of Advanced Robotic Systems*, 10. https://doi.org/10.5772/56809.

Sarac, M., Solazzi, M. and Frisoli, A., 2019. Design Requirements of Generic Hand Exoskeletons and Survey of Hand Exoskeletons for Rehabilitation, Assistive, or Haptic Use. *IEEE Transactions on Haptics*, 12(4), pp.400–413. https://doi.org/10.1109/TOH.2019.2924881.

Singh, N., Saini, M., Anand, S., Kumar, N., Srivastava, M.V.P. and Mehndiratta, A., 2019. Robotic Exoskeleton for Wrist and Fingers Joint in Post-Stroke Neuro-Rehabilitation for Low-Resource Settings. *IEEE*

Transactions on Neural Systems and Rehabilitation Engineering, 27(12), pp.2369–2377. https://doi.org/10.1109/TNSRE.2019.2943005.

Stroke Association, 2013. *Physical effects of stroke*. Available at: <https://www.stroke.org.uk/sites/default/files/physical_effects_of_stroke.pdf> [Accessed 2 March 2022].

Tiboni, M., Borboni, A., Vérité, F., Bregoli, C. and Amici, C., 2022. *Sensors and Actuation Technologies in Exoskeletons: A Review*. *Sensors*, https://doi.org/10.3390/s22030884.

WILLIAMS, N.W., PENROSE, J.M.T., CADDY, C.M., BARNES, E., HOSE, D.R. and HARLEY, P., 2000. A Goniometric Glove for Clinical Hand Assessment. *Journal of Hand Surgery*, 25(2), pp.200–207. https://doi.org/10.1054/jhsb.1999.0360.

World Health Organization, 2013. *WHO methods and data sources for global burden of disease estimates 2000-2011*. [online] Available at: <https://www.who.int/healthinfo/statistics/GlobalDALYmethods_2000_2011.pdf> [Accessed 1 March 2022].

Xia, T. and Frey-Law, L.A., 2015. *Wrist joint torque-angle-velocity performance capacity envelope evaluation and modelling*. *Int. J. Human Factors Modelling and Simulation*

Yang, S.H., Koh, C.L., Hsu, C.H., Chen, P.C., Chen, J.W., Lan, Y.H., Yang, Y., Lin, Y. de, Wu, C.H., Liu, H.K., Lo, Y.C., Liu, G.T., Kuo, C.H. and Chen, Y.Y., 2021. An instrumented glove-controlled portable hand-exoskeleton for bilateral hand rehabilitation. *Biosensors*, 11(12). https://doi.org/10.3390/bios11120495.

Yap, H.K., Lim, J.H., Goh, J.C.H. and Yeow, C.H., 2016. Design of a Soft Robotic Glove for Hand Rehabilitation of Stroke Patients with Clenched Fist

Deformity using Inflatable Plastic Actuators. *Journal of Medical Devices, Transactions of the ASME*, 10(4). https://doi.org/10.1115/1.4033035.

Yurkewich, A., Kozak, I.J., Hebert, D., Wang, R.H. and Mihailidis, A., 2020. Hand Extension Robot Orthosis (HERO) Grip Glove: Enabling independence amongst persons with severe hand impairments after stroke. *Journal of NeuroEngineering and Rehabilitation*, 17(1). https://doi.org/10.1186/s12984-020-00659-5.

APPENDICES

Appendix A: Android Application Code for "RehabConnect" Activity

```
package com.project.rehabilitation_system;

import android.Manifest;
import android.bluetooth.BluetoothAdapter;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.widget.Toast;

import androidx.core.app.ActivityCompat;
import androidx.fragment.app.FragmentManager;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

public class RehabConnect extends AppCompatActivity implements
FragmentManager.OnBackStackChangedListener {

    // declare variables
    Toolbar toolbar;
    private BluetoothAdapter mBlueAdapter;
    private String activity_mode;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // retrieve activity mode
        Bundle extras = getIntent().getExtras();
        activity_mode = extras.getString("activity_mode"); //
success extraction


        // initialise views
        toolbar = findViewById(R.id.toolbar);
        // sets the toolbar for this activity
        setSupportActionBar(toolbar);


getSupportFragmentManager().addOnBackStackChangedListener(this
);

        if (savedInstanceState == null) // check fragment not
there in first place
            if (activity_mode.equals("Rehab")) {

getSupportFragmentManager().beginTransaction().add(R.id.fragme
nt, new DevicesFragment(), "devices").commit();
            }
            else {

getSupportFragmentManager().beginTransaction().add(R.id.fragme
nt, new DevicesFragment2(), "devices").commit();
            }
```

```
        else
            onBackStackChanged();

        if (activity_mode.equals("Rehab")) {
            getSupportActionBar().setTitle("Rehabilitation
Activity");
        }else {
            getSupportActionBar().setTitle("Measure Recovery
Progress");
        }


toolbar.setTitleTextColor(getResources().getColor(android.R.co
lor.black));



        // on btn click, turn on bluetooth
        // create bluetooth adapter
        mBlueAdapter = BluetoothAdapter.getDefaultAdapter();


        // check is bluetooth is already enabled

        // if not enabled, turn on
        if (!mBlueAdapter.isEnabled()) {


            // turn on bluetooth
            if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.BLUETOOTH_CONNECT) !=
PackageManager.PERMISSION_GRANTED) {

            }
            mBlueAdapter.enable();


            // if already on...
            if (mBlueAdapter.isEnabled()){
                showToast("Bluetooth is now on");}
            } else {
               showToast("Bluetooth is already on");
                   }

    }

    @Override
    public void onBackStackChanged() {

getSupportActionBar().setDisplayHomeAsUpEnabled(getSupportFrag
mentManager().getBackStackEntryCount()>0);
    }

    @Override
    public boolean onSupportNavigateUp() {
        onBackPressed();
        return true;
    }

    // toast message function
    private void showToast(String msg) {
```

```
            Toast.makeText(this, msg, Toast.LENGTH_SHORT).show();
        }

    }
```

Appendix B: Android Application Code for "DevicesFragment" Activity

```
package com.project.rehabilitation_system;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.pm.PackageManager;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.ListFragment;

import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;

import java.util.ArrayList;

public class DevicesFragment extends ListFragment {


    private BluetoothAdapter bluetoothAdapter;

    // initialise array for paired devices
    private final ArrayList<BluetoothDevice> listItems = new
ArrayList<>();

    // adapter to create a view for each item (paired device)
    private ArrayAdapter<BluetoothDevice> listAdapter;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // if device has bluetooth capabilities

if(getActivity().getPackageManager().hasSystemFeature(PackageM
anager.FEATURE_BLUETOOTH))

            // create bluetooth adapter
            bluetoothAdapter =
BluetoothAdapter.getDefaultAdapter();

        // create list to show all paired devices and their
addresses
        listAdapter = new
ArrayAdapter<BluetoothDevice>(getActivity(), 0, listItems) {
            @NonNull
            @Override
            public View getView(int position, View view,
@NonNull ViewGroup parent) {
                BluetoothDevice device =
listItems.get(position);
                if (view == null)
                    // show the fragment view
                    view =
```

```
getActivity().getLayoutInflater().inflate(R.layout.device_list
_item, parent, false);
                TextView text1 =
view.findViewById(R.id.text1);
                TextView text2 =
view.findViewById(R.id.text2);
                text1.setText(device.getName());
                text2.setText(device.getAddress());
                return view;
            }
        };
    }

    @Override
    public void onActivityCreated(Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
        setListAdapter(null);
        View header =
getActivity().getLayoutInflater().inflate(R.layout.device_list
_header, null, false);
        getListView().addHeaderView(header, null, false);

        setListAdapter(listAdapter);
    }


    // when go out of app and come back
    @Override
    public void onResume() {
        super.onResume();

        refresh();
    }

    // finding all paired devices and store them into an array
    void refresh() {
        listItems.clear();
        // if bluetooth is on, get list of paired devices
        if(bluetoothAdapter != null) {
            for (BluetoothDevice device :
bluetoothAdapter.getBondedDevices())
                if (device.getType() !=
BluetoothDevice.DEVICE_TYPE_LE)
                    listItems.add(device);
        }

        listAdapter.notifyDataSetChanged();
    }

    // once a paired device is selected, change fragment to
RehabModesFragment
    @Override
    public void onListItemClick(@NonNull ListView l, @NonNull
View v, int position, long id) {
        BluetoothDevice device = listItems.get(position-1);
        Bundle args = new Bundle();
        args.putString("device", device.getAddress());
        Fragment fragment = new RehabModesFragment();
        fragment.setArguments(args);

getFragmentManager().beginTransaction().replace(R.id.fragment,
```

```
fragment, "rehabmodes").addToBackStack(null).commit();
    }

}
```

Appendix C: Android Application Code for "RehabModesFragment" Activity

```
package com.project.rehabilitation_system;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.Bundle;
import android.os.Handler;
import android.os.IBinder;
import android.os.Looper;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

public class RehabModesFragment extends Fragment implements
ServiceConnection, SerialListener {

    private enum Connected { False, Pending, True }

    // store address of microcontroller
    private String deviceAddress;
    // bluetooth connection service
    private SerialService service;
    private TextView receiveText;

    // to show status of pairing
    private ImageView pairStatusImage;
    private ImageView miniPairStatusImage;

    // variable storing pairing status
    private Connected connected = Connected.False;


    private boolean initialStart = true;
    private boolean hexEnabled = false;


    private Button easyModeBtn;
    private Button intermediateModeBtn;
    private Button difficultModeBtn;

    private TextView rehabText;
```

```
    private TextView easySelected;
    private TextView intermediateSelected;
    private TextView difficultSelected;

    private ImageView easyStar;
    private ImageView intermediateStar;
    private ImageView difficultStar;


    @Override

    // on create of fragment, get microcontroller id
    public void onCreate(@Nullable Bundle savedInstanceState)
{
        super.onCreate(savedInstanceState);

        //setHasOptionsMenu(true);
        setRetainInstance(true);
        // get device address from devices fragment
        deviceAddress = getArguments().getString("device");

    }

    // when activity ends, disconnect from microcontroller
    @Override
    public void onDestroy() {

        // if still connected, disconnect
        if (connected != Connected.False)
            disconnect();
        getActivity().stopService(new Intent(getActivity(),
SerialService.class));
        super.onDestroy();
    }

    // when activity starts,
    @Override
    public void onStart() {
        super.onStart();
        if(service != null)
            service.attach(this);
        else
            getActivity().startService(new
Intent(getActivity(), SerialService.class)); // prevents
service destroy on unbind from recreated activity caused by
orientation change
    }

    // when activity stop
    @Override
    public void onStop() {
        if(service != null &&
!getActivity().isChangingConfigurations())
            service.detach();
        super.onStop();
    }

    @SuppressWarnings("deprecation")
    @Override
    public void onAttach(@NonNull Activity activity) {
```

```
        super.onAttach(activity);
        getActivity().bindService(new Intent(getActivity(),
SerialService.class), this, Context.BIND_AUTO_CREATE);
    }

    @Override
    public void onDetach() {
        try { getActivity().unbindService(this); }
catch(Exception ignored) {}
        super.onDetach();
    }

    @Override
    public void onResume() {
        super.onResume();
        if(initialStart && service != null) {
            initialStart = false;
            getActivity().runOnUiThread(this::connect);
        }
    }

    @Override
    public void onServiceConnected(ComponentName name, IBinder
binder) {
        service = ((SerialService.SerialBinder)
binder).getService();
        service.attach(this);
        if(initialStart && isResumed()) {
            initialStart = false;
            getActivity().runOnUiThread(this::connect);
        }
    }

    @Override
    public void onServiceDisconnected(ComponentName name) {
        service = null;
    }


// when activity starts, user only sees pairing status
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState) {

        View view =
inflater.inflate(R.layout.fragment_terminal, container,
false);

        pairStatusImage =
view.findViewById(R.id.pairStatusIm);
        miniPairStatusImage =
view.findViewById(R.id.miniPairStatus);
        receiveText = view.findViewById(R.id.receive_text);

        // mini icon is initially invisible
        miniPairStatusImage.setVisibility(View.INVISIBLE);

        easyModeBtn = view.findViewById(R.id.easyMode);
        intermediateModeBtn =
view.findViewById(R.id.intermediateMode);
        difficultModeBtn =
```

```
view.findViewById(R.id.difficultMode);
        rehabText = view.findViewById(R.id.rehabText);

        // notification text
        easySelected = view.findViewById(R.id.selected_easy);
        intermediateSelected =
view.findViewById(R.id.selected_intermediate);
        difficultSelected =
view.findViewById(R.id.selected_difficult);
        easyStar = view.findViewById(R.id.easy_star);
        intermediateStar =
view.findViewById(R.id.intermediate_star);
        difficultStar =
view.findViewById(R.id.difficult_star);

        // hide modes
        easyModeBtn.setVisibility(view.INVISIBLE);
        easyModeBtn.setEnabled(false);
        intermediateModeBtn.setVisibility(view.INVISIBLE);
        intermediateModeBtn.setEnabled(false);
        difficultModeBtn.setVisibility(view.INVISIBLE);
        difficultModeBtn.setEnabled(false);
        rehabText.setVisibility(view.INVISIBLE);
        easySelected.setVisibility(view.INVISIBLE);
        intermediateSelected.setVisibility(view.INVISIBLE);
        difficultSelected.setVisibility(view.INVISIBLE);
        easyStar.setVisibility(view.INVISIBLE);
        intermediateStar.setVisibility(view.INVISIBLE);
        difficultStar.setVisibility(view.INVISIBLE);


        return view;

    }


    // connect the socket
    private void connect() {
        try {
            BluetoothAdapter bluetoothAdapter =
BluetoothAdapter.getDefaultAdapter();
            BluetoothDevice device =
bluetoothAdapter.getRemoteDevice(deviceAddress);
            receiveText.setText("        Connecting...");

pairStatusImage.setImageResource(R.drawable.loading_foreground
);

miniPairStatusImage.setImageResource(R.drawable.loading_foregr
ound);

            connected = Connected.Pending;
            SerialSocket socket = new
SerialSocket(getActivity().getApplicationContext(), device);
            service.connect(socket);

        } catch (Exception e) {
            onSerialConnectError(e);
        }
    }
```

```
    // unpair, disconnect socket
    private void disconnect() {
        connected = Connected.False;
        service.disconnect();
    }

    // send message to microcontroller
    private void send(String str) {

        // if not paired
        if(connected != Connected.True) {
            Toast.makeText(getActivity(), "not connected",
Toast.LENGTH_SHORT).show();
            return;
        }
        try {

            byte[] data;

            data = (str).getBytes(); // only string can use
this function to encode into array of bytes

            // send data to microcontroller
            service.write(data);

        } catch (Exception e) {
            onSerialIoError(e);
        }
    }


     // listen for the following statuses
    // status: paired
    @Override
    public void onSerialConnect() {
        receiveText.setText("          Connected");

pairStatusImage.setImageResource(R.drawable.success_pair_foreg
round);

miniPairStatusImage.setImageResource(R.drawable.success_pair_f
oreground);

        connected = Connected.True;

        // hide big pair status picture
        pairStatusImage.setVisibility(View.INVISIBLE);
        receiveText.setVisibility(View.INVISIBLE);

        // Show the button selections
        easyModeBtn.setVisibility(View.VISIBLE);
        easyModeBtn.setEnabled(true);
        intermediateModeBtn.setVisibility(View.VISIBLE);
        intermediateModeBtn.setEnabled(true);
        difficultModeBtn.setVisibility(View.VISIBLE);
        difficultModeBtn.setEnabled(true);
        rehabText.setVisibility(View.VISIBLE);


        // send activity type to microcontroller
        send("1"); // rehabilitation activity
```

```
        easyModeBtn.setOnClickListener(new
View.OnClickListener() {

            public void onClick(View v) {

                // send mode to microcontroller
                send("1");

                // tell user they selected easy mode
                easySelected.setVisibility(View.VISIBLE);
                easyStar.setVisibility(View.VISIBLE);
                easyModeBtn.setVisibility(View.INVISIBLE);

intermediateModeBtn.setVisibility(View.INVISIBLE);

difficultModeBtn.setVisibility(View.INVISIBLE);

            }
        });

        intermediateModeBtn.setOnClickListener(new
View.OnClickListener() {

            public void onClick(View v) {

                // send mode to microcontroller
                send("2");


intermediateSelected.setVisibility(View.VISIBLE);
                intermediateStar.setVisibility(View.VISIBLE);
                easyModeBtn.setVisibility(View.INVISIBLE);

intermediateModeBtn.setVisibility(View.INVISIBLE);

difficultModeBtn.setVisibility(View.INVISIBLE);
            }
        });

        difficultModeBtn.setOnClickListener(new
View.OnClickListener() {

            public void onClick(View v) {

                // send mode to microcontroller
                send("3");

                difficultSelected.setVisibility(View.VISIBLE);
                difficultStar.setVisibility(View.VISIBLE);
                easyModeBtn.setVisibility(View.INVISIBLE);

intermediateModeBtn.setVisibility(View.INVISIBLE);

difficultModeBtn.setVisibility(View.INVISIBLE);
            }
        });


        // CHANGE Pairing status to smaller icon
        miniPairStatusImage.setVisibility(View.VISIBLE);
```

```
    }

    // status: cannot pair
    @Override
    public void onSerialConnectError(Exception e) {
        receiveText.setText("    Connection failed");

pairStatusImage.setImageResource(R.drawable.fail_pair_foregrou
nd);

miniPairStatusImage.setImageResource(R.drawable.fail_pair_fore
ground);

        disconnect();

        // timer
        Runnable runnable = new Runnable() {
            @Override
            public void run() {
                connect();
            }
        };

        Handler handler = new Handler(Looper.getMainLooper());

        handler.postDelayed(runnable, 2000); // delayed 2 s
    }

    // when data is read
    @Override
    public void onSerialRead(byte[] data) {

    }

    // when pairing is lost suddenly
    @Override
    public void onSerialIoError(Exception e) {
        receiveText.setText("     Connection Lost");

pairStatusImage.setImageResource(R.drawable.fail_pair_foregrou
nd);

miniPairStatusImage.setImageResource(R.drawable.fail_pair_fore
ground);

        disconnect();

        // timer
        Runnable runnable = new Runnable() {
            @Override
            public void run() {
                connect();
            }
        };

        Handler handler = new Handler(Looper.getMainLooper());

        handler.postDelayed(runnable, 2000); // delayed 2 s

    }
}
```

Appendix D: ESP32 Code for Transmission System

```
// include this header file from the library
#include <Servo_ESP32.h>
#include "BluetoothSerial.h"

// initialize variable
char Mode; // store mode of rehabilitation activity

// assign GPIO pin 14 as control pin for left servo
static const int servoPinLeft = 14; //printed G14 on the board
// assign GPIO pin 15 as control pin for right servo
static const int servoPinRight = 15;

int angle =0; // initial angle
int angleStep = 1; // incremental angle

// max angular displacements
int angleMin =0;
int angleMax = 42;

//check if bluetooth is properly enabled
#if                                    !defined(CONFIG_BT_ENABLED)
|| !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable
it
#endif

// create an instance of bluetooth serial
BluetoothSerial SerialBT;

  servoLeft.attach(servoPinLeft); // attaching servo to pin
  servoRight.attach(servoPinRight);
```

```
  // initialise bluetooth serial device
  SerialBT.begin("ESP32_Control"); //Bluetooth device name

void loop() {
  // receive data from mcu
        if (SerialBT.available()){
        Mode = SerialBT.read();

        if (Mode == '1'){
          // actuate
          // need to alter timer
          for (int timer = 0; timer <=73; timer +=1){

            for(int angle = angleMin; angle <= angleMax; angle +=angleStep) {

            int angle2 = 40 - angle;

            servoLeft.write(angle);
            servoRight.write(angle2);
            delay(120);
            }

            for(int angle = angleMax; angle >= angleMin; angle -=angleStep) {

            int angle2 = 40 - angle;

            servoLeft.write(angle);
            servoRight.write(angle2);
            delay(120);
            }

          }
```

```
} else {
 if (Mode == '2'){
  for (int timer = 0; timer <=73; timer +=1){

   for(int angle = angleMin; angle <= angleMax; angle +=angleStep) {

    int angle2 = 40 - angle;

    servoLeft.write(angle);
    servoRight.write(angle2);
    delay(70);
   }

   for(int angle = angleMax; angle >= angleMin; angle -=angleStep) {

    int angle2 = 40 - angle;

    servoLeft.write(angle);
    servoRight.write(angle2);
    delay(70);
   }

  }
 } else {

  for (int timer = 0; timer <=73; timer +=1){

   for(int angle = angleMin; angle <= angleMax; angle +=angleStep) {

    int angle2 = 40 - angle;

    servoLeft.write(angle);
```

```
            servoRight.write(angle2);
            delay(30);
           }

           for(int angle = angleMax; angle >= angleMin; angle -=angleStep) {

            int angle2 = 40 - angle;

            servoLeft.write(angle);
            servoRight.write(angle2);
            delay(30);
           }

         }


       }
      }
     }
}
}
```

Appendix E: Android Application Code for "DevicesFragment2" Activity

```
package com.project.rehabilitation_system;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.pm.PackageManager;
import android.os.Bundle;
import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.ListFragment;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;

import java.util.ArrayList;

public class DevicesFragment2 extends ListFragment {


    private BluetoothAdapter bluetoothAdapter;

    // initialise array for paired devices
    private final ArrayList<BluetoothDevice> listItems = new
ArrayList<>();

    // adapter to create a view for each item (paired device)
    private ArrayAdapter<BluetoothDevice> listAdapter;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);


if(getActivity().getPackageManager().hasSystemFeature(PackageM
anager.FEATURE_BLUETOOTH))

            // create bluetooth adapter
            bluetoothAdapter =
BluetoothAdapter.getDefaultAdapter();

        // create list to show all paired devices and their
addresses
        listAdapter = new
ArrayAdapter<BluetoothDevice>(getActivity(), 0, listItems) {
            @NonNull
            @Override
            public View getView(int position, View view,
@NonNull ViewGroup parent) {
                BluetoothDevice device =
listItems.get(position);
                if (view == null)
                    // show the fragment view
                    view =
getActivity().getLayoutInflater().inflate(R.layout.device_list
_item2, parent, false);
                TextView text1 =
```

```
view.findViewById(R.id.text3);
                TextView text2 =
view.findViewById(R.id.text4);
                text1.setText(device.getName());
                text2.setText(device.getAddress());
                return view;
            }
        };
    }

    @Override
    public void onActivityCreated(Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
        setListAdapter(null);
        View header =
getActivity().getLayoutInflater().inflate(R.layout.fragment_de
vices_header2, null, false);
        getListView().addHeaderView(header, null, false);

        setListAdapter(listAdapter);
    }

    // when go out of app and come back
    @Override
    public void onResume() {
        super.onResume();

        refresh();
    }


    // finding all paired devices and store them into an array
    void refresh() {
        listItems.clear();
        if(bluetoothAdapter != null) {
            for (BluetoothDevice device :
bluetoothAdapter.getBondedDevices())
                if (device.getType() !=
BluetoothDevice.DEVICE_TYPE_LE)
                    listItems.add(device);
        }

        listAdapter.notifyDataSetChanged();
    }

    // once a paired device is selected, change fragment to
message terminal
    @Override
    public void onListItemClick(@NonNull ListView l, @NonNull
View v, int position, long id) {
        BluetoothDevice device = listItems.get(position-1);
        Bundle args = new Bundle();
        args.putString("device", device.getAddress());
        Fragment fragment = new ProgressStartFragment();
        fragment.setArguments(args);

getFragmentManager().beginTransaction().replace(R.id.fragment,
fragment, "terminal").addToBackStack(null).commit();
    }

}
```

Appendix F: Android Application Code for "ProgressStartFragment" Activity

```
package com.project.rehabilitation_system;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

import android.os.Handler;
import android.os.IBinder;
import android.os.Looper;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

public class ProgressStartFragment extends Fragment implements
ServiceConnection, SerialListener {

    private enum Connected { False, Pending, True }

    // store address of microcontroller
    private String deviceAddress;
    // bluetooth connection service
    private SerialService service;
    private TextView receiveText;

    private TextView mcpStatus;
    private TextView wristStatus;

    // to show status of pairing
    private ImageView pairStatusImage;
    private ImageView miniPairStatusImage;

    // variable storing pairing status
    private Connected connected = Connected.False;

    private boolean initialStart = true;
    private boolean hexEnabled = false;

    private TextView rehabText;

    private StringBuilder message_whole;
    private TextView MCPAngle;
    private TextView WristAngle;
    private TextView MCPUnit;
```

```
    private TextView WristUnit;


    @Override

    // on create of fragment, get microcontroller id
    public void onCreate(@Nullable Bundle savedInstanceState)
{
        super.onCreate(savedInstanceState);

        //setHasOptionsMenu(true);
        setRetainInstance(true);
        // get device address from devices fragment
        deviceAddress = getArguments().getString("device");

        // store entire message received by microcontroller
        message_whole = new StringBuilder(100);

    }

    // when activity ends, disconnect from microcontroller
    @Override
    public void onDestroy() {

        // if still connected, disconnect
        if (connected != Connected.False)
            disconnect();
        getActivity().stopService(new Intent(getActivity(),
SerialService.class));
        super.onDestroy();
    }

    // when activity starts,
    @Override
    public void onStart() {
        super.onStart();
        if(service != null)
            service.attach(this);
        else
            getActivity().startService(new
Intent(getActivity(), SerialService.class));
    }

    // when activity stop
    @Override
    public void onStop() {
        if(service != null &&
!getActivity().isChangingConfigurations())
            service.detach();
        super.onStop();
    }

    @SuppressWarnings("deprecation")
    @Override
    public void onAttach(@NonNull Activity activity) {
        super.onAttach(activity);
        getActivity().bindService(new Intent(getActivity(),
SerialService.class), this, Context.BIND_AUTO_CREATE);
    }

    @Override
```

```
    public void onDetach() {
        try { getActivity().unbindService(this); }
catch(Exception ignored) {}
        super.onDetach();
    }

    @Override
    public void onResume() {
        super.onResume();
        if(initialStart && service != null) {
            initialStart = false;
            getActivity().runOnUiThread(this::connect);
        }
    }

    @Override
    public void onServiceConnected(ComponentName name, IBinder
binder) {
        service = ((SerialService.SerialBinder)
binder).getService();
        service.attach(this);
        if(initialStart && isResumed()) {
            initialStart = false;
            getActivity().runOnUiThread(this::connect);
        }
    }

    @Override
    public void onServiceDisconnected(ComponentName name) {
        service = null;
    }


     // view displayed to users when activity starts

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState) {

        View view =
inflater.inflate(R.layout.fragment_progress_start, container,
false);

        pairStatusImage =
view.findViewById(R.id.pairStatusIm);
        miniPairStatusImage =
view.findViewById(R.id.miniPairStatus);
        receiveText = view.findViewById(R.id.receive_text);
        mcpStatus = view.findViewById(R.id.MCP_Status);
        wristStatus = view.findViewById(R.id.Wrist_Status);

        // mini icon is initially invisible
        miniPairStatusImage.setVisibility(View.INVISIBLE);
        mcpStatus.setVisibility(View.INVISIBLE);
        wristStatus.setVisibility(View.INVISIBLE);

        rehabText = view.findViewById(R.id.rehabText);

        // hide text
        rehabText.setVisibility(view.INVISIBLE);
```

```
        // display MCP and Wrist angles
        MCPAngle = view.findViewById(R.id.MCP_angle);
        WristAngle = view.findViewById(R.id.Wrist_angle);
        MCPUnit = view.findViewById(R.id.unit_MCP);
        WristUnit = view.findViewById(R.id.unit_Wrist);

        MCPUnit.setVisibility(View.INVISIBLE);
        WristUnit.setVisibility(View.INVISIBLE);

        return view;

    }


    // pairing, connect socket
    private void connect() {
        try {
            BluetoothAdapter bluetoothAdapter =
BluetoothAdapter.getDefaultAdapter();
            BluetoothDevice device =
bluetoothAdapter.getRemoteDevice(deviceAddress);
            receiveText.setText("        Connecting...");

pairStatusImage.setImageResource(R.drawable.loading_foreground
);

miniPairStatusImage.setImageResource(R.drawable.loading_foregr
ound);
            connected = Connected.Pending;
            SerialSocket socket = new
SerialSocket(getActivity().getApplicationContext(), device);
            service.connect(socket);

        } catch (Exception e) {
            onSerialConnectError(e);
        }
    }

    // unpair, disconnect socket
    private void disconnect() {
        connected = Connected.False;
        service.disconnect();
    }

    // send message to microcontroller
    private void send(String str) {

        // if not paired
        if(connected != Connected.True) {
            Toast.makeText(getActivity(), "not connected",
Toast.LENGTH_SHORT).show();
            return;
        }
        try {

            byte[] data;

            data = (str).getBytes(); // only string can use
this function to encode into array of bytes

            // send data to microcontroller
```

```
                service.write(data);

            } catch (Exception e) {
                onSerialIoError(e);
            }
        }

        // receive data from microcontroller
        private void receive(byte[] data) {
            if(hexEnabled) {

            } else {

                message_whole.append(new String (data));

            }

        }


         // Listening to the following statuses

        // status: paired
        @Override
        public void onSerialConnect() {
            receiveText.setText("          Connected");

pairStatusImage.setImageResource(R.drawable.success_pair_foreg
round);

miniPairStatusImage.setImageResource(R.drawable.success_pair_f
oreground);
            connected = Connected.True;

            // hide big pair status picture
            pairStatusImage.setVisibility(View.INVISIBLE);
            receiveText.setVisibility(View.INVISIBLE);

            rehabText.setVisibility(View.VISIBLE);
            mcpStatus.setVisibility(View.VISIBLE);
            wristStatus.setVisibility(View.VISIBLE);

            // CHANGE Pairing status to smaller icon
            miniPairStatusImage.setVisibility(View.VISIBLE);

            // send activity type to microcontroller
            send("2"); // rehabilitation activity
        }

        // status: cannot pair
        @Override
        public void onSerialConnectError(Exception e) {
            receiveText.setText("    Connection failed");

pairStatusImage.setImageResource(R.drawable.fail_pair_foregrou
nd);

miniPairStatusImage.setImageResource(R.drawable.fail_pair_fore
ground);

            disconnect();
```

```
        // timer
        Runnable runnable = new Runnable() {
            @Override
            public void run() {
                connect();
            }
        };

        Handler handler = new Handler(Looper.getMainLooper());

        handler.postDelayed(runnable, 2000); // delayed 2 s
    }

    // when data sent by microcontroller is read
    @Override
    public void onSerialRead(byte[] data) {

        receive(data);


        if (message_whole.indexOf(":")!=-1){

            // split message into string arrays
            String[] split =
message_whole.toString().split(":");

            // issue with split again
            StringBuffer mcp_b = new StringBuffer();
            StringBuffer wrist_b = new StringBuffer();

            mcp_b.append(split[0]);
            wrist_b.append(split[1]);

            String mcp_s = mcp_b.toString();
            String wrist_s = wrist_b.toString();

            // display angles
            MCPAngle.setText(mcp_s);
            MCPUnit.setVisibility(View.VISIBLE);
            WristUnit.setVisibility(View.VISIBLE);
            WristAngle.setText(wrist_s);
        }
    }


    // when pairing is lost suddenly
    @Override
    public void onSerialIoError(Exception e) {
        receiveText.setText("     Connection Lost");

pairStatusImage.setImageResource(R.drawable.fail_pair_foregrou
nd);

miniPairStatusImage.setImageResource(R.drawable.fail_pair_fore
ground);

        disconnect();

        // timer
        Runnable runnable = new Runnable() {
```

```
            @Override
            public void run() {
                connect();
            }
        };

        Handler handler = new Handler(Looper.getMainLooper());

        handler.postDelayed(runnable, 2000); // delayed 2 s

    }

}
```

Appendix G: ESP32 Code for Sensing System

```
// include this header file from the library
#include "BluetoothSerial.h"

// initialize variable
char MCP_data[20]; // character array to store MCP angle
char Wrist_data[20]; // character array to store Wrist angle
float MCP_max_data = 0;
float Wrist_max_data = 0;

// assign GPIO pin 34 as input pin for MCP flex sensor signal
static const int FLEX_MCP_PIN = 34;
// assign GPIO pin 35 as input pin for Wrist flex sensor signal
static const int FLEX_Wrist_PIN = 35;

// Measure the voltage at 5V and the actual resistance of
// 10 k resistor, and enter them below:
const float VCC = 5.17; // Measured voltage
const float R_DIV = 10000.0; // Measured resistance of 10k resistor

// values below can be adjusted during callibration
const float STRAIGHT_RESISTANCE_MCP = 17203.4; // resistance when straight
const float BEND_RESISTANCE_MCP = 33311.9; // resistance at 90 deg
const float STRAIGHT_RESISTANCE_Wrist = 16864.1; // resistance when straight
const float BEND_RESISTANCE_Wrist = 26740; // resistance at 60 deg

float angle_MCP;
float angle_Wrist;
```

```
// create an instance of bluetooth serial
BluetoothSerial SerialBT;

void setup() {
// initialise the serial monitor
  Serial.begin(9600);

  //for MCP joint
  pinMode(FLEX_MCP_PIN, INPUT);
  // for Wrist joint
  pinMode(FLEX_Wrist_PIN, INPUT);

  // initialise bluetooth serial device
  SerialBT.begin("ESP32_Control"); //Bluetooth device name
}

if (SerialBT.available()) {
      for (int timer = 0; timer <=10; timer +=1)
      {
        // Read the ADC values and calculate voltage and resistance
        int flexADC_MCP = analogRead(FLEX_MCP_PIN);
        float flexV_MCP = flexADC_MCP * VCC / 4095.0;
        float flexR_MCP = R_DIV * (VCC / flexV_MCP - 1.0);

        int flexADC_Wrist = analogRead(FLEX_Wrist_PIN);
        float flexV_Wrist = flexADC_Wrist * VCC / 4095.0;
        float flexR_Wrist = R_DIV * (VCC / flexV_Wrist - 1.0);

        // Use the calculated resistance to estimate the sensor's
        // bend angle:
        float angle_MCP_F = map(flexR_MCP,
STRAIGHT_RESISTANCE_MCP, BEND_RESISTANCE_MCP,
```

```
                0, 90.0);
  float          angle_Wrist_F          =          map(flexR_Wrist,
STRAIGHT_RESISTANCE_Wrist, BEND_RESISTANCE_Wrist,
                0, 60.0);

   // display on Arduino Web Editor's serial monitor for calibration
  // activities
  Serial.println("Resistance MCP: " + String(flexR_MCP) + " ohms");
  Serial.println("Resistance Wrist: " + String(flexR_Wrist) + " ohms");
  Serial.println("Bend MCP: " + String(angle_MCP_F) + " degrees");
  Serial.println("Bend Wrist: " + String(angle_Wrist_F) + " degrees");
  Serial.println();

  // obtain the maximum angles generated at MCP and Wrist
  if (angle_MCP_F > MCP_max_data){
    angle_MCP = angle_MCP_F;
    MCP_max_data = angle_MCP_F;
  }

  if (angle_Wrist_F > Wrist_max_data){
    angle_Wrist = angle_Wrist_F;
    Wrist_max_data = angle_Wrist_F;
  }

  delay(500);
 }

 // convert float to int
 int   angle_MCP_int = (int)angle_MCP;
 int   angle_Wrist_int = (int)angle_Wrist;

 // convert from float to character array
 sprintf(MCP_data, "%d", angle_MCP_int);
```

```
      sprintf(Wrist_data, "%d", angle_Wrist_int);

      // combine the MCP angle data and Wrist angle data together
      strcat(MCP_data, ":");
      strcat(MCP_data, Wrist_data);

      // Send data to android application
      for (int i = 0; i <= 15; i++)
      {
        Serial.println(char (MCP_data[i]));
        SerialBT.write((uint8_t) MCP_data[i]);
      }
    }
  }
  delay(20);
}
```

Appendix H: Ten Sets of Maximum Resistance Values for Sensing System Calibration

Maximum Resistance Generated During 0° MCP Joint Flexion	Maximum Resistance Generated During 0° Radiocarpal Joint Extension	Maximum Resistance Generated During 90° MCP Joint Flexion	Maximum Resistance Generated During 60° Radiocarpal Joint Extension
17263	16782	33471	26047
17119	16852	33060	27431
16958	15885	33425	29337
17137	17101	33241	27092
16764	16852	32700	24880
16782	17048	33333	26111
17318	17282	33288	28235
17300	16539	33379	23292
18202	17191	33843	26891
17191	17119	33379	28093

Appendix I: Android Application Code for "OrigamiVideos" Activity

```
package com.project.rehabilitation_system;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.os.Bundle;
import android.webkit.WebView;
import android.webkit.WebViewClient;

public class OrigamiVideos extends AppCompatActivity {

    // declare variable
    Toolbar toolbar;

    @Override

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_origami_videos);

        // initialise views
        toolbar = findViewById(R.id.toolbar);

        // sets the toolbar for this activity
        setSupportActionBar(toolbar);

        getSupportActionBar().setTitle("Origami Tutorials");


toolbar.setTitleTextColor(getResources().getColor(android.R.co
lor.black));

        WebView wv = (WebView)findViewById(R.id.my_webview);
        WebView wv2 = (WebView)findViewById(R.id.my_webview2);
        WebView wv3 = (WebView)findViewById(R.id.my_webview3);

        wv.getSettings().setJavaScriptEnabled(true);
        wv2.getSettings().setJavaScriptEnabled(true);
        wv3.getSettings().setJavaScriptEnabled(true);

        wv.setWebViewClient(new WebViewClient() );
        wv2.setWebViewClient(new WebViewClient() );
        wv3.setWebViewClient(new WebViewClient() );

        // embed HTML content into WebView elements
        String customHTML = "<iframe width=\"350\"
height=\"160\"
src=\"https://www.youtube.com/embed/aBRUb0TOHik\"
title=\"YouTube video player\" frameborder=\"0\"
allow=\"accelerometer; autoplay; clipboard-write; encrypted-
media; gyroscope; picture-in-picture\"
allowfullscreen></iframe>";
        wv.loadData(customHTML, "text/html", "UTF-8");

        String customHTML2 = "<iframe width=\"350\"
height=\"160\"
src=\"https://www.youtube.com/embed/vs14JXq8XSk\"
```

```
title=\"YouTube video player\" frameborder=\"0\"
allow=\"accelerometer; autoplay; clipboard-write; encrypted-
media; gyroscope; picture-in-picture\"
allowfullscreen></iframe>";
        wv2.loadData(customHTML2, "text/html", "UTF-8");

        String customHTML3 = "<iframe width=\"350\"
height=\"160\"
src=\"https://www.youtube.com/embed/mnRMxb8r4v8\"
title=\"YouTube video player\" frameborder=\"0\"
allow=\"accelerometer; autoplay; clipboard-write; encrypted-
media; gyroscope; picture-in-picture\"
allowfullscreen></iframe>";
        wv3.loadData(customHTML3, "text/html", "UTF-8");

    }

}
```

Appendix J: Android Application Code for “HomeDirectory” Activity

```
package com.project.rehabilitation_system;

        import androidx.appcompat.app.AppCompatActivity;
        import androidx.appcompat.widget.Toolbar;
        import android.content.Intent;
        import android.os.Bundle;
        import android.view.View;

public class HomeDirectory extends AppCompatActivity{
    // declare variable
    Toolbar toolbar;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home_directory);

        // initialise views
        toolbar = findViewById(R.id.toolbar);
        // sets the toolbar for this activity
        setSupportActionBar(toolbar);

        getSupportActionBar().setTitle("Exoskeleton
Rehabilitation System");

toolbar.setTitleTextColor(getResources().getColor(android.R.co
lor.black));
    }

    // when "start rehabilitation activity" button clicked,
launch paired list activity
    public void GoPairing(View v){
        String value = "Rehab";
        //"this" refers to navigation activity
        Intent i = new Intent(this, RehabConnect.class);
        i.putExtra("activity_mode", value);
        startActivity(i);
    }

    // when "start progress measurement" button clicked,
launch rehabilitation activity
    public void GoProgress(View v){
        String value = "Progress";
        Intent i = new Intent(this, RehabConnect.class);
        i.putExtra("activity_mode", value);
        startActivity(i);
    }

    // when "start learning origami" button clicked, launch
paired list activity
    public void GoVideos(View v){
        //"this" refers to navigation activity
        Intent i = new Intent(this, OrigamiVideos.class);
        startActivity(i);
    }
}
```

Appendix K: Complete Android Application Code

```
// include this header file from the library
#include <Servo_ESP32.h>
#include "BluetoothSerial.h"

// initialize variable
char Mode; // store mode of rehabilitation activity
char Activity; // store type of activity

char MCP_data[20]; // character array to store MCP angle
char Wrist_data[20]; // character array to store Wrist angle
float MCP_max_data = 0;
float Wrist_max_data = 0;

// assign GPIO pin 14 as control pin for left servo
static const int servoPinLeft = 14; //printed G14 on the board
// assign GPIO pin 15 as control pin for right servo
static const int servoPinRight = 15;

// assign GPIO pin 34 as input pin for MCP flex sensor signal
(voltage divider output)
static const int FLEX_MCP_PIN = 34;
static const int FLEX_Wrist_PIN = 35;

// create an instance of esp32 class
Servo_ESP32 servoLeft;
Servo_ESP32 servoRight;


int angle =0; // initial angle
int angleStep = 1; // incremental angle

// max angular displacements
int angleMin =0;
int angleMax = 42;



// Measure the voltage at 5V and the actual resistance of your
// 47k resistor, and enter them below:
const float VCC = 5.17; // Measured voltage of Ardunio 5V line
const float R_DIV = 10000.0; // Measured resistance of 10k
resistor

// Upload the code, then try to adjust these values to more
// accurately calculate bend degree.
const float STRAIGHT_RESISTANCE_MCP = 17203.4; // resistance
when straight
const float BEND_RESISTANCE_MCP = 33311.9; // resistance at 90
deg
const float STRAIGHT_RESISTANCE_Wrist = 16864.1; // resistance
when straight
const float BEND_RESISTANCE_Wrist = 26740; // resistance at 60
deg

float angle_MCP;
float angle_Wrist;
```

```
//check if bluetooth is properly enabled
#if !defined(CONFIG_BT_ENABLED) ||
!defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig`
to and enable it
#endif

// create an instance of bluetooth serial
BluetoothSerial SerialBT;


void setup() {
  // initialise th serial monitor
  Serial.begin(9600); // make sure match at serial monitor
  servoLeft.attach(servoPinLeft); // attaching servo to pin
  servoRight.attach(servoPinRight);

  //for MCP joint
  pinMode(FLEX_MCP_PIN, INPUT);
  // for Wrist joint
  pinMode(FLEX_Wrist_PIN, INPUT);

  //////////
  // initialise bluetooth serial device
  SerialBT.begin("ESP32_Control"); //Bluetooth device name
  Serial.println("The device started, now you can pair it with
bluetooth!");
}


void loop() {

  // transmit data out of mcu
  if (Serial.available()) {

    // send data using bluetooth serial, data is obtained from
serial
    SerialBT.write(Serial.read());
  }

  // receive data from mcu
  if (SerialBT.available()) {

    Activity = SerialBT.read();

    if (Activity == '1'){

      // rehabilitation activity
      //void loop()
      for (int timer = 0; timer <=10000; timer +=1) {
        delay(100);

        if (SerialBT.available()){
        Mode = SerialBT.read();

        if (Mode == '1'){
          Serial.write("easy");
          // actuate
          // need to alter timer
          for (int timer = 0; timer <=73; timer +=1){
```

```
            for(int angle = angleMin; angle <= angleMax; angle
+=angleStep) {

            int angle2 = 40 - angle;

            servoLeft.write(angle);
            servoRight.write(angle2);
            //Serial.println(angle);
            delay(120);
            }

            for(int angle = angleMax; angle >= angleMin; angle
-=angleStep) {

            int angle2 = 40 - angle;

            servoLeft.write(angle);
            servoRight.write(angle2);
            //Serial.println(angle);
            delay(120);
            }

          }

        } else {
          if (Mode == '2'){
            Serial.write("intermediate");
            for (int timer = 0; timer <=73; timer +=1){

              for(int angle = angleMin; angle <= angleMax;
angle +=angleStep) {

                int angle2 = 40 - angle;

                servoLeft.write(angle);
                servoRight.write(angle2);
                //Serial.println(angle);
                delay(70);
              }

              for(int angle = angleMax; angle >= angleMin;
angle -=angleStep) {

                int angle2 = 40 - angle;

                servoLeft.write(angle);
                servoRight.write(angle2);
                //Serial.println(angle);
                delay(70);
              }

            }
          } else {
            Serial.write("difficult");

            for (int timer = 0; timer <=73; timer +=1){

              for(int angle = angleMin; angle <= angleMax;
angle +=angleStep) {

                int angle2 = 40 - angle;
```

```
                servoLeft.write(angle);
                servoRight.write(angle2);
                //Serial.println(angle);
                delay(30);
              }

             for(int angle = angleMax; angle >= angleMin;
angle -=angleStep) {

                int angle2 = 40 - angle;

                servoLeft.write(angle);
                servoRight.write(angle2);
                //Serial.println(angle);
                delay(30);
              }

           }


         }
       }
     }

     }

   }else{

     // progress measurement activity
     //void loop()
     for (int timer = 0; timer <=10; timer +=1)
     {
       // Read the ADC, and calculate voltage and resistance
from it
       int flexADC_MCP = analogRead(FLEX_MCP_PIN);
       float flexV_MCP = flexADC_MCP * VCC / 4095.0;
       float flexR_MCP = R_DIV * (VCC / flexV_MCP - 1.0);

       int flexADC_Wrist = analogRead(FLEX_Wrist_PIN);
       float flexV_Wrist = flexADC_Wrist * VCC / 4095.0;
       float flexR_Wrist = R_DIV * (VCC / flexV_Wrist - 1.0);

       // Use the calculated resistance to estimate the
sensor's
       // bend angle:
       float angle_MCP_F = map(flexR_MCP,
STRAIGHT_RESISTANCE_MCP, BEND_RESISTANCE_MCP,
                          0, 90.0);
       float angle_Wrist_F = map(flexR_Wrist,
STRAIGHT_RESISTANCE_Wrist, BEND_RESISTANCE_Wrist,
                          0, 60.0);



       Serial.println("Resistance MCP: " + String(flexR_MCP)
+ " ohms");
       Serial.println("Resistance Wrist: " +
String(flexR_Wrist) + " ohms");
       Serial.println("Bend MCP: " + String(angle_MCP_F) + "
degrees");
```

```
        Serial.println("Bend Wrist: " + String(angle_Wrist_F)
+ " degrees");
        Serial.println();

        // obtain the maximum angles generated  MCP and Wrist
        if (angle_MCP_F > MCP_max_data){
          angle_MCP = angle_MCP_F;
          MCP_max_data = angle_MCP_F;
        }

        if (angle_Wrist_F > Wrist_max_data){
          angle_Wrist = angle_Wrist_F;
          Wrist_max_data = angle_Wrist_F;
        }


        delay(500);
      }

      // convert float to int
      int   angle_MCP_int = (int)angle_MCP;
      int   angle_Wrist_int = (int)angle_Wrist;

      // convert from float to character array
      sprintf(MCP_data, "%d", angle_MCP_int);
      sprintf(Wrist_data, "%d", angle_Wrist_int);

      strcat(MCP_data, ":");
      strcat(MCP_data, Wrist_data);

      for (int i = 0; i <= 15; i++)
      {
        Serial.println(char (MCP_data[i]));
        SerialBT.write((uint8_t) MCP_data[i]);
      }

    }

  }
  delay(20);
}
```