UNIVERSITY OF MALTA

MASTER OF SCIENCE IN ENGINEERING DISSERTATION

A Real Time hand-movement motion capture system for rehabilitation of children with Cerebral Palsy

Mario Farrugia

Supervised by:

Prof. Ing. SImon Fabri

Co-supervised by:

Prof. Ing. Owen Casha

A dissertation submitted in partial fulfilment of the requirements for the degree of Master of Science in Engineering

by the

Faculty of Engineering

June 2023

Copyright Notice

Abstract

Cerebral palsy is a physical disability that affects movement and posture. It is the most common physical disability in children. Motivated by the child functional goals set during occupational therapy for children with cerebral palsy, the SMARTCLAP project (funded by the MCST TDP Project R& I-2019-003-T) places the user at the centre of the design process to develop a revolutionary device. Through this design approach, SMARTCLAP aims to increase the motivation of a child with cerebral palsy during therapy sessions, in which guardians/parents can also be involved. This contributes towards developing a positive behaviour and improving the social interaction of the child.

As part of the SMARTCLAP project, this work has developed a novel sensor-based, hand and fingers motion capture device (glove) to be used in the rehabilitation of children suffering from Cerebral Palsy, through interaction with an Augmented Reality (AR) system. The system aims to offer a cheap, user-centred design option, which is tailor-made for a particular child and makes use of current motion capture technologies and 3D printing.

The glove is based on eleven ICM 20948 Inertial Measurement Units, which capture the movement of the fingers and the hand and send this data to a central processing unit to be analysed and used by a Motion Capture Algorithm (MCA), based upon a model of the hand and its movements using the kinematics defined by the International Society of Biomechanics, which are interfaced with the AR system. The AR game then sends back visual and auditory feedback to the user. The motion capture device is battery powered, re-chargeable and can run uninterrupted for over 45 minutes (the time of one therapy session with a child).

Sensing of the movements and positioning of the different sensors with respect to each other is achieved using the mathematical theory of dual-Quaternions, which offer a highly computationally efficient and unique way of representing rotations and translations in a three-dimensional space. Furthermore, the device and MCA are designed with a short and easy calibration process.

The device was tested in comparison to an industry standard vision-based motion capture system (VICON) and results achieved show a discrepancy of around 3.5°, which is comparable to other similar studies and also within the range of movement which is distinguishable by humans.

Acknowledgement

I would firstly like to thank my supervisor Prof Ing. Simon Fabri and co-supervisor Prof Ing. Owen Casha for their patience, time and guidance. Further thanks goes to the SMARTClap project team, headed by Prof Ing Philip Farrugia and Matthew Bonello, for including me in this interesting project and supporting me throughout it.

I would also also like to thank the people that helped me with the work which I was not able to do, specifically Ing. Rueben Debono and the team in the Electronics System Laboratory and Adrian Axiaq for helping me with soldering the electronic boards, Riccardo Gatt Ellis for designing the test rig and John Paul Borg for building it, as well as Ing. Rachael Duca and Ing. Jean Gauci *dont know surnames here* at the Biomedical Engineering Lab for helping me out with the VICON system.

I also acknowledge the financial support I received for this work through the MCST TDP Project R& I-2019-003-T.

Contents

List of Figures

List of Tables

List of Acronyms

- CP - Celebral Palsy
- AR -Augmented Reality
- 3D - 3 Dimensional
- IMU - Inertial Measurement Unit
- MCA - Motion Capture Algorithm
- RAGT - Robot-Assisted Gait Training
- EMG - Electromyography
- VR - Virtual Reality
- DoF - Degrees of Freedom
- ADC - Analog to Digital Convertor
- BLE - Bluetooth Low Energy
- MCP - Metacarpal
- TMCP - Thumb- Metacarpal
- TCMC - Thumb CarpoMetacarpal
- PIP - Proximal Intermidiate Phalange joint
- DIP - Distal Intermidiate Phalange joint
- PP - Proximal Phalange
- IP - Intermidiate Phalange
- DP - Distal Phalange
- LCS - Local Coodrinate System
- GHCS - Global Hand Coordinate System
- WCS - Wrist Coordinate System
- MCS - Metacarpal Coordinate System
- PCS - Phalangeal Coordinate System

Chapter 1 – Introduction

Modern technology is contributing to medical and health sciences in many ways. This includes the use of computer technology in medical devices to aid in therapy and rehabilitation. From monitoring, for example, gait movement for patients suffering from neuromuscular and orthopaedic problems to be able to give them better treatment [1] to treatment such as continuous glucose monitoring and automatic insulin pumps for type-1 diabetes sufferers [2]), computer technologies have become part and parcel to the medical world.

This dissertation deals with the hardware and software design and manufacture of a device that will capture the the finger and hand movement of children with Cerebral Palsy (CP) during their therapy sessions while they are playing a serious Augmented Reality (AR) game, in order to make therapy fun and engaging and thus motivating them to improve the function of their upper limbs.

1.1 What is Cerebral Palsy?

Cerebral Palsy is a group of disorders that affect the movement and muscle tone or posture, of an individual. It is caused by damage to the brain while developing, usually during pregnancy [3]. It is one of the most common disabilities in childhood. Symptoms of the disorder usually become apparent as the child is growing, although these do not usually worsen with age. Such symptoms can vary, from affecting one limb or side to the whole body, while always involving difficulty with movement and coordination. These can include [3]:

- Variations in muscle tone, such as being either too stiff or too floppy
- Stiff muscles and exaggerated reflexes (spasticity)
- Stiff muscles with normal reflexes (rigidity)
- Lack of balance and muscle coordination (ataxia)
- Tremors or involuntary movements
- Slow, writhing movements
- Delays in reaching motor skills milestones, such as pushing up on arms, sitting up or crawling
- Favouring one side of the body, such as reaching with one hand or dragging a leg while crawling

- Difficulty walking, such as walking on toes, a crouched gait, a scissors-like gait with knees crossing, a wide gait or an asymmetrical gait
- Excessive drooling or problems with swallowing
- Difficulty with sucking or eating
- Delays in speech development or difficulty speaking
- Learning difficulties
- Difficulty with fine motor skills, such as buttoning clothes or picking up utensils
- Seizures

Being a disorder emanating from the brain, CP can also be associated with side neurological problems.

CP patients can be categorised in different (or a combination of) classes, which, when compared to the type of movement disorder that is present, can lead the therapist towards the best clinical procedure, depending on the Functional Classification recommended. Two most commonly used are the ABILHAND-Kids [4] and the Manual Ability Classification System (MACS) [5], which define various levels of the extent of the disorder.

Lacking a cure (which would require brain damage repair), CP is usually managed using a number of different therapies, based on the principle of giving the child a normalisation of the quality of movement and developing a set of functional skills in order to be able to lead as independent a daily life as possible [6]. Simply, this means that the child needs to relearn the functional movement of their limbs such as hands and fingers..

In [7], Bayon lists a number of therapies used for the rehabilitation of CP patients, namely: physical and occupational focusing on walking, standing stretching and flexibility, oral medication (most for spasticity), orthotics to prevent deformities, Botulinum toxin (for local spasticity), ferule and plaster to avoid contractures, orthopaedic surgery, neurosurgical procedures, movement and treadmill therapies based on motor learning theories and finally Robot-Aided Gait Therapy (RAGT). This latter therapy is emerging as both an alternative and complementary treatment in the push to diminish the time, physical and financial strain on caregivers and family of children with CP (as well as on the children themselves). These robotics and computer-assisted rehabilitation systems appear promising, shown to increase the motivation of a child to train their motor functions intensely in a playful therapeutic environment [8].

1.2 The SMARTClap project

A ‘smart user centred product service system for evaluating and developing functional hand skills in children with Cerebral Palsy (SMARTCLAP)‘is a project funded by the Malta Council for Science and Technology (project reference R & I-2019-003-T). The work reported in this dissertation was one component the SMARTCLAP project.

The SMARTClap project aims to build on the promises shown by RAGT therapies by using a User-centred Design (UCD) approach that places product users at the centre of the design process. Motivated by the customised occupational therapy required for children with CP, this project exploits UCD to develop a product service system, which offers an enhanced personalised experience and remote monitoring of the child's progress by a paediatric Occupational Therapist (OT).

The overall aim of the SMARTCLAP project is to design and manufacture a 3-D printed, user-centred wearable device that will be used by children suffering from Cerebral Palsy (specifically to do with the hand) during their rehabilitation while interacting with a serious AR game. It has been shown that AR games improve the motivation of children suffering from motor function disorders [6]. The data captured from the device during the sessions will also be used by the child's occupational Therapist to analyse any improvement in hand and finger movement. The subtasks of the SMARTClap project are depicted in figure 1.1.

The wearable device must capture the hand and finger movements of the user and wirelessly transfer the orientation and coordinate data to a Central Processing Unit (PC, laptop, tablet) that will convert this data onto an Augmented Reality screen to be used by the user in a serious game, as shown in the 3rd frame of figure 1.1.

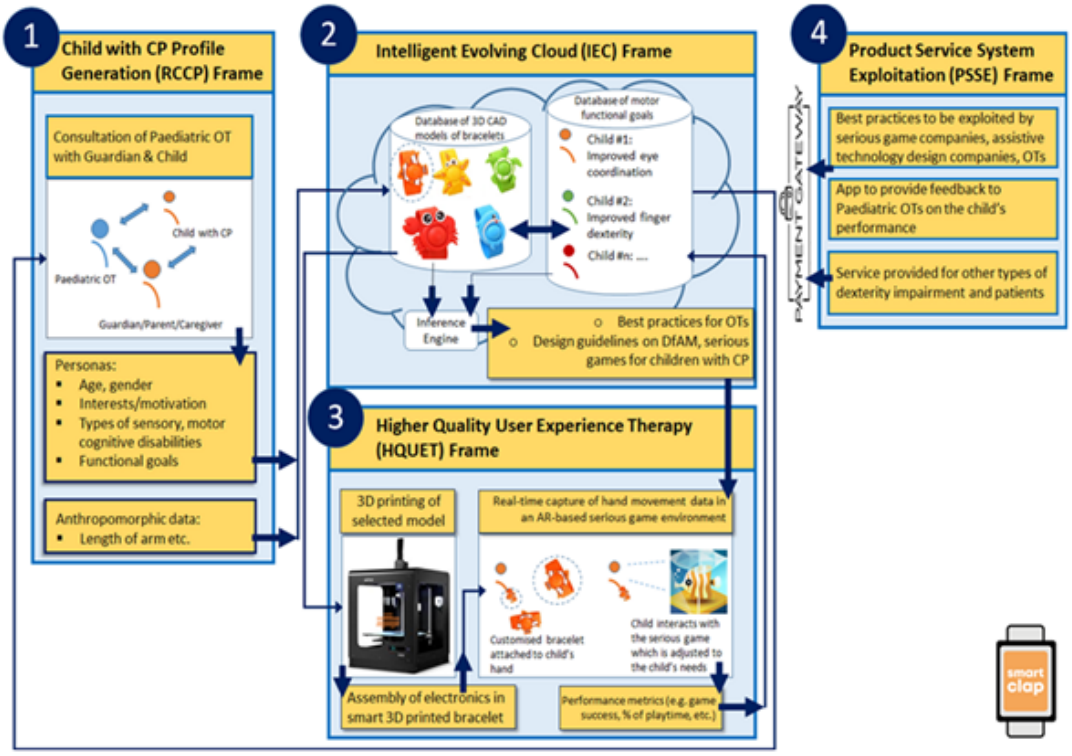


Figure 1.1: SMARTCLAP Project framework with this project's objectives shown in frame 3, where the device will capture the child's hand motion to help them interact with a serious game through AR, adjusted to the child's needs

1.3 Motivation for this work

The SMARTCLAP project is divided in several work packages including project management, hardware and software design, dissemination, and evaluation. The work presented in this dissertation presents the work carried out concerning:

- the development, implementation and testing of the sensory hardware of the wearable device and the corresponding motion capture algorithm (MCA) required to estimate and track the finger and wrist movements in real-time of the user with CP, whilst playing the serious game with the wearable device.
- the design, fabrication and testing of the back-end hardware supporting the relevant functions of the motion capture device (e.g. to send/receive information to/from the game etc.).

As shown in the literature review presented in Chapter 2, various products are available on the market which offer some kind of hand-motion capture. However, these devices tend to be prohibitive to buy and use due to the price tag, they do not always give enough data on the motion of the fingers (specifically motion data of the separate phalanges of the fingers), and are not customisable for the needs of the person using them.

The availability of on the positioning of the finger tips and the phalanges while using the device would help the Occupational Therapist to monitor and evaluate better any tiny improvement made by the child between therapy sessions.

The constraints of the project dictated that the device needs to be wearable and portable (thus wireless and small).

Thus, building on research already available in literature, this work aimed to create a cost-effective, customizable motion-tracking hand-wearable device which can be easily accessible for children needing to use it for therapy.

It is to be noted that the wearable device was aimed for children with CP aged 4-11 year and whose functional mobility can be classified as being on scale 1 and 2 of the MACS cited previously [5]. This is because children that are higher on the mobility scale might have difficulty opening their hand in order for the device to be correctly worn.

1.4 Methodology

Figure 1.2 shows the high-level design of the system. The sensors, attached to the wearable device will capture the movement of the hand, transmit the data to a central processing unit which outputs the movements to a screen.

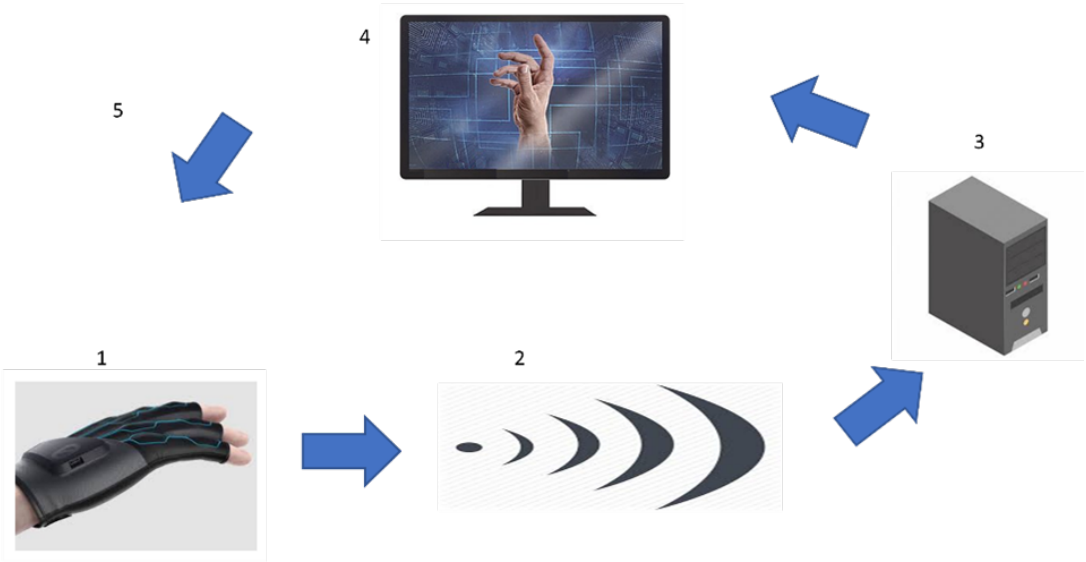


Figure 1.2: High Level diagram of the system. The research reported here deals mainly with (1) and (2)

Thus, the first step was to research and evaluate currently available systems and sensors that are suitable for the motion capture of the hand and fingers. The sensors used most where bending sensors, electromyography (EMG) sensors, accelerometers and

magnetometers, and, more recently, Inertial Measurement Units (IMUs), which are accelerometer, gyroscope, and magnetometer in one chip. Bending sensors are cheap and lightweight, but they cannot track specific phalanges of the fingers and their output is analogue, which would need to be converted to digital before processing. EMG technology is not advanced enough to track finger movements accurately, although META is working on such technology [9]. Accelerometers, magnetometers and Inertial Measurement Units can track both rotational and translational movement in 3D and come with a small footprint. To balance between complexity and accuracy, in this project it was decided to use only IMUs to track the finger motion, so that each finger will have 2 IMUs connected to track the proximal and intermediate phalanges, with the movement of the distal phalange extrapolated mathematically from the other 2.

The next step involved the identification of the necessary components to use for the device. The main components are:

1. An IMU sensor that measures orientation (need 11 of these, as shown in figure 1.3). A new IMU from Bosch, BNO055 [10] was originally selected, however, due to this device running out of stock worlwide due to the Covid-19 pandemic during development, a 2nd option, the ICM-20948 from TDK-Invensense [11] was used as a replacement. This microchip has the ability to output orientation data in the format that was used in the motion capture algorithm, i.e. Quaternions. This made interfacing all sensor modules and the data capture device easier.

2. A module which captures the orientation data from all the sensors and transmit them wirelessly to the processing unit. For this, the Arduino Nano board was chosen because it gave the best balance between price ($30) and functionality (multiple inputs and a wireless Bluetooth output).

3. A battery pack

4. Leds and/or vibrational module for visual feedback and battery power/charging monitoring.

Having acquired the hardware off the shelf, the sensor modules, were redesigned to fit the needs of a 4-11 year old child. The development boards used were too big to fit on the phalange of a child, and hence needed to be redesigned to fit accordingly. Using the acquired hardware, the motion capture algorithm (MCA) was then developed to capture the orientation data from every sensor and "construct" the model of a hand and its movements using kinematics defined by the International Society of Biomechanics (IBS) [12]. Finally, the wearable device was tested and evaluated against industry-standard motion-capture systems, namely the VICON system.

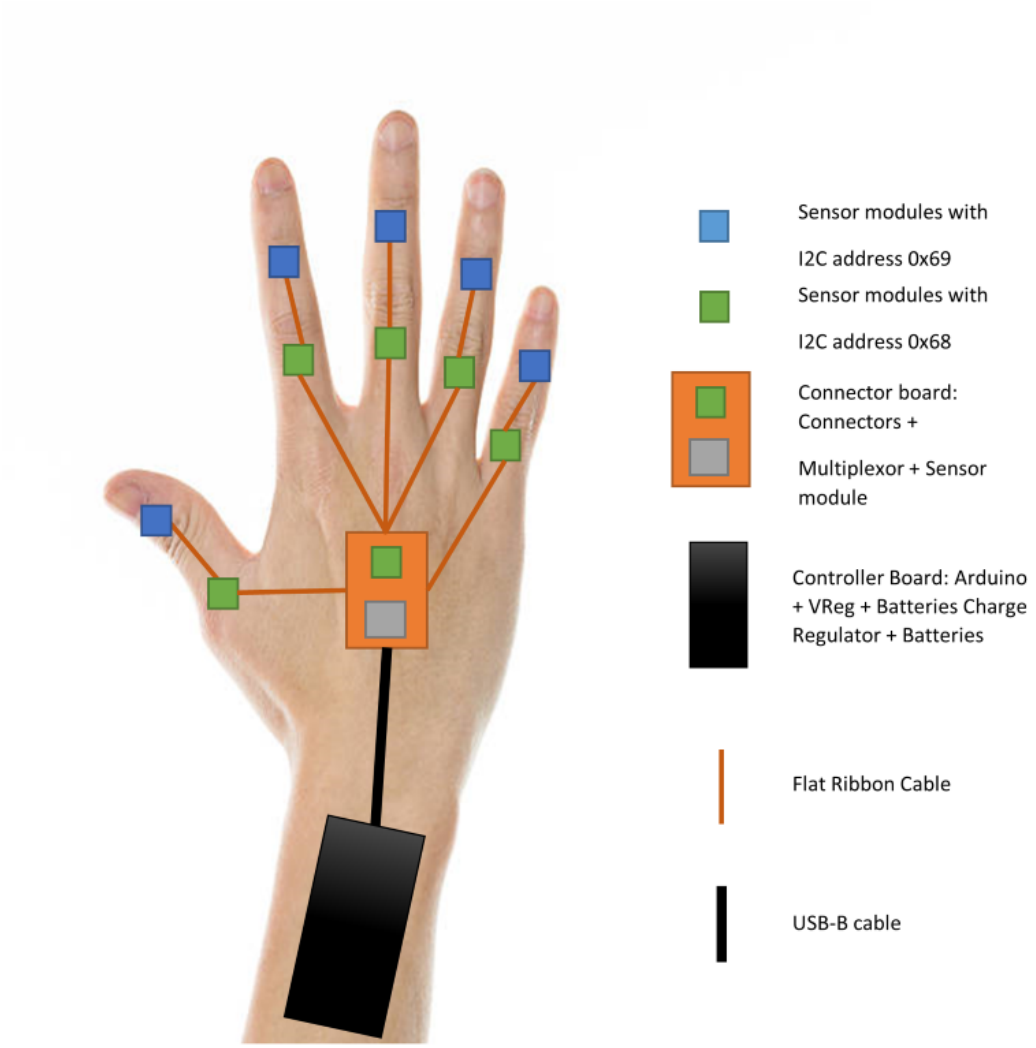


Figure 1.3: Component schematic

1.5 Project Outcome

This project successfully designed and produced a hand-wearable device able to capture the movement of the hand and each phalange of every finger, using the minimum amount of sensors possible (11, one on each proximal and intermediate phalange plus one on the back palm) and extrapolating on the data from the intermediate phalange sensors to estimate the positioning of the distal phalange.

A list of features characterising the developed system include:

- A MCA based on dual quaternion theory, which, although complex, is one of the most efficient methods to represent motion in 3D space;
- Bring together work from different authors to provide an effective way to use dual quaternions to get positioning of the hand and fingers in 3D space;
- No long and complex calibration process to start using the glove (maximum stabilization length of 25 seconds);
- Uses a minimal amount of hardware while still giving positioning data of all the phalanges of the fingers;
- Providing flexibility of customizing the number of fingers to be used during the therapy session;
- Possible to use on both hands and provides the possibility of detecting on which hand the glove is worn;
- Hardware is small enough to easily fit on a child's finger;

This work brought together research from other studies [13, 14] that effectively used dual quaternions with a minimum number of sensors possible to capture the free movement of the fingers and hand. It provides a finished solution of a chargeable, battery-powered, motion capture wearable device, capable of running for longer than the 45 minutes, which is the time that CP therapy sessions typically take. Results show that the device can capture motion data and send it via wireless at a rate of 8Hz (enough for an AR game) with a discrepancy in finger joint angles that is comparable to other studies of around 3°when compared to the VICON industry standard reference system.

1.6 Dissertation Outline

The rest of the dissertation is divided in four sections.

Chapter 2 provides a review of RAGT and computer-assisted therapies currently used for CP rehabilitation. It also reviews current motion capture technologies and smart wearables available on the market. A theory of how these technologies are used in algorithms to represent motion in 3D space is also provided.

Chapter 3 describes the mathematical theory of the Motion Capture Algorithm. To achieve this, a detailed description of the physiological structure of the hand is provided, describing the hand model on which the MCA is based.

Chapter 4 provides a description of the theory of quaternions which leads to the actual MCA description and the procedure of how positioning data coming from each IMU is used to capture the motion of that phalange and the whole hand.

Chapter 5 highlights the design process behind the hardware that was developed. It discusses the decision-making process of selecting the technology and electronics to use for capturing the hand movements and for sending data to the AR game. It also describes the overall system, lists the software used for designing the selected hardware, and provides details of all the electronics used.

Chapter 6 explains how the MCA was implemented, coded, tested and validated. It highlights the use of quaternions for implementation and the software used to implement it.

Chapter 7 describes the testing and validation carried out and the results achieved are discussed and compared to previous works documented in the literature.

The conclusion highlights the results, outcomes and deliverables of the work done. Further improvements, issues and problems encountered during the research are also discussed.

Chapter 2 – Literature Review

2.1 Introduction

This chapter will provide a review of computer assisted therapies currently used in CP rehabilitation. It also reviews current motion capture technologies and smart wearable devices available on the market (not necessarily used for CP or rehabilitation). Finally, the theory of how these technologies are used in algorithms to represent motion in 3D space is described.

2.2 Computer-Assisted Therapies for CP Rehabilitation

An increase in rehabilitation robotic therapies for adults after a stroke has led to similar therapies being tried more and more in paediatrics [15]. Motivation and active participation are important factors in the learning of sensory-motor functional skills in children [16]. However, neuropsychological studies have shown that children with developmental disorders appear less motivated and more passive during playtime than their physically typically developed peers, although having the same pleasure [17]. Computer-Assisted therapy combined with Virtual/Augmented Reality (VR/AR) looks to be an interesting concept in increasing the motivation levels of children undergoing therapy [18].

VR therapy is an interactive and enjoyable intervention which has recently been shown to improve upper extremity motor function in adults with chronic hemiparesis with greater compliance by the patient [19]. However, the need of wearing a headset in VR technologies has sometimes resulted in CP patients getting disoriented after a certain amount of time [20], and thus, AR was seen as a better technology to use in this project. However, the use and functionality of AR is not within the remit of this specific thesis.

Therapies are available for both the upper and lower limbs. However, since this project is aimed at hand motion gestures, this study with focus mostly on upper limb therapies, specifically AR-aided therapies that do not involve any robotic or mechanical support, as shown in the taxonomy chart [2.1 – [21]] in Figure 2.1.

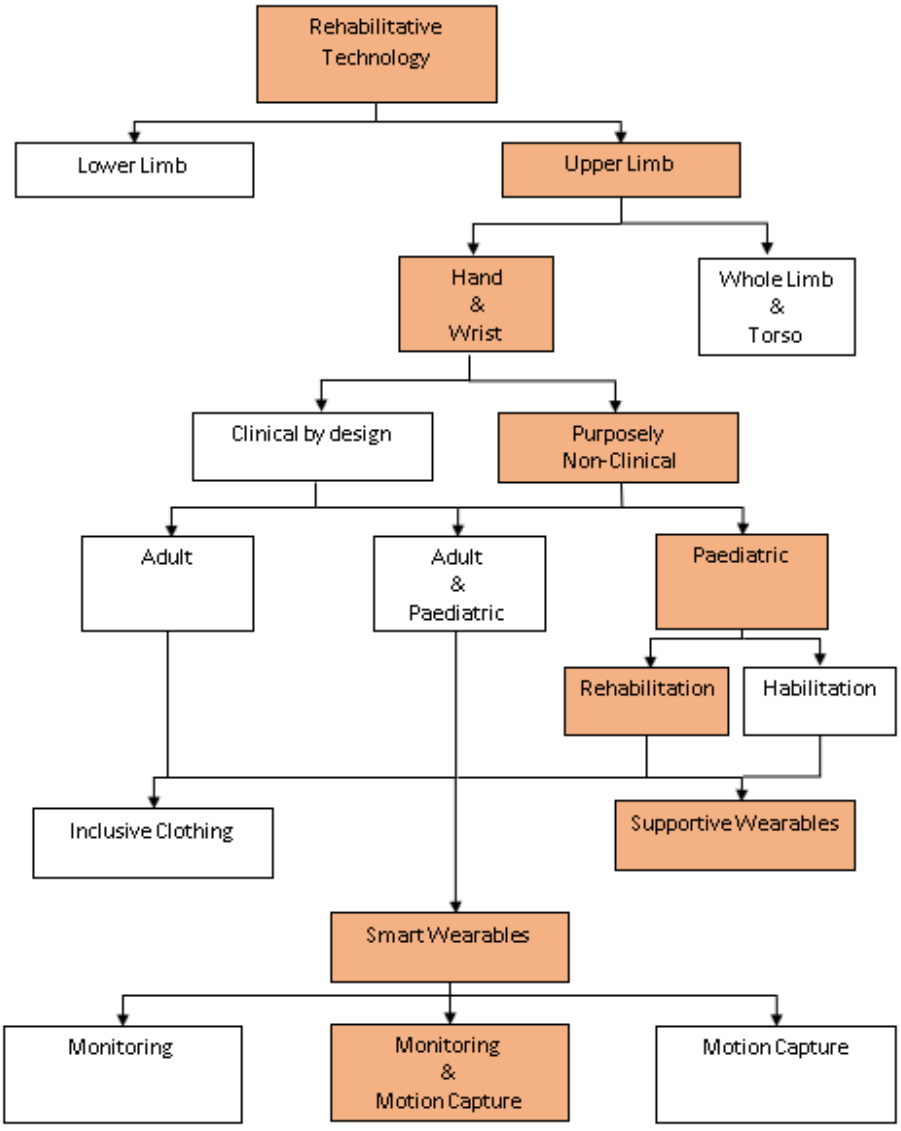


Figure 2.1: Upper Limb Taxonomy Chart [21], highlighting this project's "path"

In-depth systematic reviews of robotic-assisted therapies for children with CP (both upper and lower) is provided by Bayon et. al [7] and Meyer-Heim et al. [8] while Ahn et. al [22] systematically reviewed robots developed for motor function therapies for upper extremities (although this was for stroke survivors and not CP).

In the next section, a number of wearable devices used in the aforementioned reviews and the technologies used in each device are highlighted.

2.3 Smart Wearables and VR Systems

This section will introduce several systems which use hand-wearable, unaided devices with a type of AR/VR system. Not all of them have been used in the rehabilitation of children with CP. Some of them have not been used in rehabilitation at all. However, all of them, to a degree, contain similarities with what this project wants to achieve and build upon.

All these systems - apart from the P5 glove, which was developed purely for entertainment purposes - seem to have had a degree of success in rehabilitation when compared to conventional methods of rehab. This success was seen on both patients recovering from stroke, and children with CP, helped mainly by the higher degree of intensity of the rehabilitation brought about by the interaction with the system.

The Eye to Play gaming console by Nintendo [23] uses a camera as a sensor which tracks and captures movement of the user to place them in a VR environment. The IREX-VR by Gesture-Tek [24, 25] also uses a camera to track the user's hand movements. In this case, the user needs to use a DataGlove for motion capture. The P5-Glove [26] is a gaming device by Essential Reality which uses Bending Sensors and InfraRed(IR) receptor with an optical tracking device. It uses a USB connection and needs to point to the IR reciever all the time. The reading changes depending on how well the glove fits, which also can be a source of noise. Another glove that uses bending sensors is the Cyber Glove from CyberGlove Systems [27]. It uses 4G tracking but also needs a camera marker to track the movements. The Hand Tutor by MediTouch [28] also uses a similar technology and is used as a training glove for ' *impairment-oriented Training system whose exercises are based on repetitive and intensive active flexion and extension movements of the finger's and the wrist*'. Similarly, the Bi-Manu-Trainer (Reha-Stim) [29, 30, 31, 32] uses three bending sensors, an accelerometer on the back palm and a camera to capture pitch and roll movements. Neofect's SMART-Glove [33, 34, 35] is the nearest product to the SMARTClap wearable device in terms of design, however it uses bending sensors and an Inertial Measurement Unit (IMU) on the back palm for 3D imaging. The SMARTClap device developed in this project does not use bending sensors. Furthermore, the SMARTGlove uses a Hall Effect Sensor to

zero the IMU. The Accelle Glove by IDRT [36] uses 6 x 3-axis accelerometers (1 per finger and 1 on the backhand) to capture Sign Language positions of the finger.

The GestoGlove [14], which is not on the market, is based on 11 Inertial sensors and Magnetometers (not IMUs) and a haptic device on each finger tip for force feedback. Although not a marketable glove, [13] also uses IMUs (16 of them) for tracking and positioning of the fingers and hand.

Figure 2.2 shows a few of these gloves.

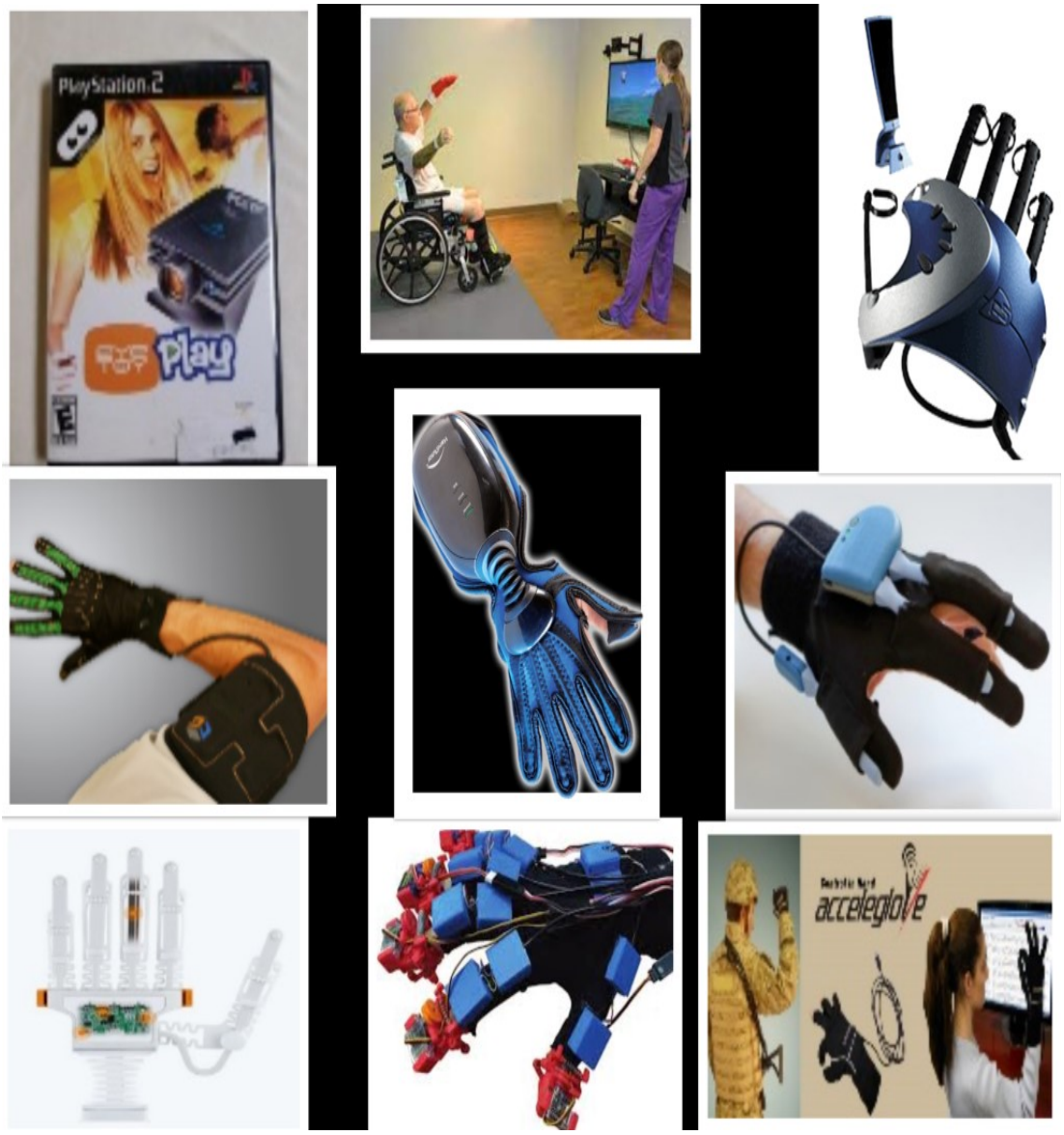


Figure 2.2: Available Motion Capture wearable systems. Starting from top left and going clockwise: Eye to Play, IREX-VR, P5-Glove, Cyber Glov, Hand Tutor, Bi-Manu Trainer, SMARTGlove, GestoGlove, AccelleGlove

These systems detect the movement of the upper limb by use of a wearable glove which the patient uses to interact with a game or function on a screen. The next section will provide some ideas about how such interactions can be developed, using Motion Sensing devices which capture the movement of the patients and feeds data back into

the system.

2.4 Motion Capture Technology

Pradhan et al [37] define motion capture as follows: *"Motion capture is the capturing of live human motion events and translating them in 3D positional and orientation information of joints in space over time"* .

Several different types of sensors have been used to capture motion of the hand for Virtual Reality devices. This literature review will focus on the ones that were used for wearable upper limb extremities.

When one thinks of motion capture, the first thing that usually comes to mind is motion sensing technology commonly used for movies, called video-based capture. This normally requires the user (the person whose motion is to be captured) to work freely within a limited space surrounded by special cameras using IR sensors, allowing for all movements to be recorded via reflective markers attached on the body (usually via skin-tight clothing to remove the chance of a marker being covered up by clothing). Due to the number of cameras needed to capture all the marker reflections all the time, this system tends to be quite expensive.

Another form of motion capture can be instrument-based. This involves the user wearing tracking sensors (e.g. accelerometers, IR, resistive bending sensors and more) that capture the motion and location with different resolutions (depending on the sensor used) which then relay the data back to a centralized processing unit. Depending on the mode in which the data is relayed back (e.g. cable, Bluetooth, Wi-Fi . . .), the user is constrained to be within reach, or line of sight of the Processing Unit.

The SMARTClap wearable device is envisaged as a low-cost instrument-based lightweight device, and thus this review will consider the instrument-based form of motion capture. Consequently, in this section, will look at sensors that have been used for this technology but not necessarily specific to rehabilitation.

2.4.1 Bending Sensors

Bending sensors can be cheap and lightweight and can easily fit on a glove to measure the bending of the fingers [38]. In electro-mechanical bending sensors, the angular bending of the fingers can be evaluated by repeatability measurements, with change of resistance being proportional to varying degree of bending. Simone et al. [38, 39] used bending sensors made of specific materials in their testing glove prototype (the Shadow Glove) due to these falling under all their four requirements: (1) Easy to put on/remove,

(2) comfortable and durable, (3) detects main functions of motor activities and (4) cost-effective. The 'glove' uses bending sensors attached to the dorsal phalanges of the hand so they could be easily attached to any hand, functional or otherwise. As the authors themselves mentioned, this was not a generic solution, and there were repeatability issues with using only bending sensors. Gentner et al. [40] improved on this by increasing the number of bend sensors (from 5 to 10) and adding an extra 4 sensor to measure abduction (the movement of the limbs away from the body). Also, they improved the calibration period from minutes to seconds. Bakhsi et al. [41] used cheap, readily available bending sensors attached on a supportive cloth, to monitor the bending of the knee, using previous data gathered over time and an Extended Kalman Filter technique. Notable in most of these works is the bulky electronics equipment needed to convert the analog output of the bending sensors to digital data to be used by the processing unit. In addition, there are sensitivity issues if the glove on which the sensors are attached does not fit perfectly with the hand. A loosely fitted glove might introduce extra movements (giving false readings) and thus it is always ideal that the glove is specific to the hand that is using it, while attaching sensors directly to the hand is not an option.

Fibre-optic bending sensors have also been used to measure flexing, although never for hands [42]. A light source coupled to a light sensor at each end of an optic fibre pipe measures the bend which is proportional to the light arriving at the sensor from the source. They seem to offer the highest degree of accuracy ($< 1^\circ$) of bending sensors [43]. That said, the necessity of the source, sensors, and fibre to go through all the joints make this system too bulky and expensive for this work.

More recently, Saggio [44] and Tongrod [45] used printed flexi sensors with a resistive polymer on a plastic substrate to measure the change in electrical resistance when bending. Several printed layers on the substrate are needed for the sensors to be effective and give consistent readings.

The problem with most bending sensors is that they bend but are not stretchable. Kusada [46], Meguc [47] and Vogt [48] used conductive liquid metal injected into a soft chamber to solve this issue. The resistance of the fluid changed with the deformity of the chamber. Later, Shen [49] used a stretchable sensor based on ethylene–propylene rubber (EPR). The resistive sensing material was moulded inside the EPR polymer to help minimise its stretch.

Bending sensors can be cheap to produce and can easily attach themselves to wearable gloves. However, they can become bulky due to the electronics needed to convert their analogue signal to digital, and also, while they can measure angular flexing of the fingers (albeit with a number of repetitive measurements for repeatability and calibration), they cannot be used alone to measure abduction, and the tracking of the hand (or any part of the body) in space.

2.4.2 Tracking and positioning Sensors

A downside of bending sensors is that they cannot provide for positioning of the hand in space, a crucial issue for any A/VR System. They only provide data about the flexion of the finger as a whole.

Infra-Red (IR) devices can create accurate positioning using Radio Frequency Identification (RFID) tags and receiving units [26]. They are relatively inexpensive; however, they require a constant line of sight and can create interference since RFID operates at near-wireless spectrum.

Camera-based systems can provide free movement to users without the need of wearables. These systems can also incorporate the user into the VR system itself [24]. In other systems, an (external) camera can use IR to detect patched reflectors on gloves/garments worn by the user whose motion is being tracked [37, 50]. While these systems can be quite accurate, multiple, external cameras are needed to sense rotation and remove any "blind spots", while at the same time needing a large amount of processing power to handle all the data gathered by these specialized cameras.

Inertial Measurement/Monitoring Units (IMUs) are packaged devices that nowadays typically incorporate a 3-axis accelerometer, a gyroscope, and a magnetometer, which together can be used to measure positioning and movement of the hand. They come with a small footprint and are easily available on the market [51].

Using six, 2-axis accelerometers, Hernandez-Rebollar [36] managed to design a glove that simulated all 26 letters of the American Sign Language (ASL). Five of the sensors were positioned on the middle phalanges of each finger, while another was placed on the wrist for measuring roll and hand posture. An issue with this design was that with two-axes both parallel to the hand palm, one could not know if the hand was facing up or facing down. This was later improved but using 3-axis accelerometers, which, with the inclusion of a 3rd axis perpendicular to the other two, solved the hand direction ambiguity. A similar device was developed by Kim et al. [52].

The Rapeal SmartGlove [33, 35] uses a 3-axis IMU to introduce a 6-Degrees of Freedom (DoF) detection for their glove, coupled with bending sensors to measure the curving of the fingers. The first known glove using IMUs for both measuring hand orientation and finger positioning used the aid of magnetometers (as a separate device, to measure the drift of accelerometer measurements due to the number of integrations needed to get displacement out of it – see later section) in order to sense hand and finger movements [53]. An accelerometer was placed on each of the distal and medial phalanges of the thumb and each finger while a magnetometer was placed on each fingertip and at the back of the hand. Three IMUs were used to gather the data of the thumb, index and middle fingers, and ring and pinky fingers respectively while a main processing board was placed on the back of the hand. The sensors were connected to

each other using plastic ribbon connectors and cables. To complement this, multiple Extended Kalman Filters (EKF) were used to gather the appropriate data and estimate optimal orientation trajectories.

The use of magnetometers to measure motion is not ideal since they measure the magnetic field of the earth, which changes from place to place, and in closed quarters (i.e. indoors) it might not be strong enough to be measured accurately [54]. Otherwise, for magnetometers to perform ideally, a magnetic source needs to be included in the setup, which would be quite cumbersome. The Gesto glove [14], due to its use of motor-based device for haptic (touch) feedback, used the bio-mechanic constraint predictions of the hand to avoid interference between the magnetic field of the motor and the magnetic sensors [55, 56]. Having said that, they are used to compensate for the drift created in the accelerometer [57, 58].

However, as previously mentioned, magnetometers now come incorporated within IMUs, together with accelerometers and gyroscopes, and hence might be less bulky to use while removing extra hardware.

One issue with using IMUs is "Dead Reckoning", which is constant feedback that each sensor has to relate back to the processing unit with regards to the starting/previously known ("home") position and the previous measurement(s)/position(s) [33]. This requires a fixed-position sensor to act as a sort of reference point [58]. This drawback, however, can be solved (at least for hand measurements) by using a sensor at the back of the palm or wrist as a reference point, while using quaternions to represent the motion of the whole hand [14]. In the same research, flexibility of the glove to fit different hand sizes (using the 50th percentile measurements of men and women) was also achieved by estimating the length of the phalanges of each user by using estimates from a priori knowledge.

Baldi et al [14] were the first to use only IMUs to capture the motion of the hand. Ten accelerometers and gyroscopes (one chip) plus ten magnetometers (separate chip) where placed on the back of the proximal and intermediate phalanges of the fingers, and an extra sensor on the palm for rotational sensing. Positioning and movement was extracted by using known biomechanical constraints of the hand [55] and a model for the relation between the distal phalange and the rest of the hand [56]. This was because no sensors were used to measure the distal phalange movement (since this was used for the cutaneous device).

[13] obtained similar results as [14] but by using dual quaternions to calculate the position and rotation of the fingers. In this work, 16 sensors per hand (3 on each phalange and one on the backpalm) were used.

[59] improved on [13] by using only six IMU sensors (one on the back palm, one on the distal phalange of the thumb and one on the intermediate phalange of the other

fingers) to estimate the positions of the finger and hands. The result was a slimmer and less bulky design, which, however limited the captured tracking of certain hand configurations, especially of the thumb.

IMUs do not require line of sight communication like video based systems, and are not restricted in space to be in the field of view of the cameras around the subject under investigation [60]. They can be restricted, like many candidates mentioned in this review, due to their connection to the main AR processing unit, thus not allowing total freedom of movement in space (this would depend on the cable length), as well as the possibility of loss of connectivity due to wear and tear of the cable. This study has avoided the latter restriction by using wireless technology to transmit the data to the AR system.

2.4.3 Other Sensors

Other sensors used in glove applications for rehabilitation are Hall effect and EMG sensors. The former was used in [33] to calibrate the glove when placed flat on a base. The magnetic source (a magnet) was placed on the palm while five magnetic sensors were placed on each fingertip. Once these align with the magnet, in combination with a base, the glove is calibrated, and the other sensors are in their 'rest' position.

More recently, a similar hand-tracking device was proposed, which used a magnetic localization system for tracking the position of the index finger compared to the wrist, and IMUs to track the pose of the thumb, index and the whole hand. This system uses less components than other gloves; it uses three IMUs and two magnets compared to sixteen IMUs in [13] or the twelve plus the haptic devices used in [14]. The sensors are positioned exclusively on the fingertips of the thumb and index finger. Hence, the positioning of the fingers can only be estimated according to the position of the fingertips only.

Baldi et al. [14] used a servo motor-based contraption attached to the fingertips as a cutaneous (touch) device to provide haptic (force) feedback to the user. This idea however looks very bulky and might be too much for the sensitivity of children with CP, and is thus not going to be considered further for this work.

2.5 The SMARTClap Glove

From the gloves available on the market and in research, none seem to be designed specifically for children suffering from Cerebral Palsy. The hand of a child affected by this disease can 'rest' in its natural place in a whole range of positions, from totally closed, to totally open. Furthermore, movements of fingers and hands of CP-affected

patients can occur as well on a large range, from normal movement to no movement at all.

Designing a normal, 'five-finger' glove with sensors attached to it is easy to do for children whose palm can easily open. With CP, a palm can be 'massaged' open in most cases, but depending on the child, this can stay open for minutes or only seconds. Donning the glove becomes difficult for children whose hand does not have the capability to fully open. Furthermore, children with CP tend to have very sensitive skin and thus wearing a whole garment-made glove or using sticky materials to attach the sensors to the hand is also not ideal due to possible irritability of the skin.

The glove needs also to be versatile, with the 'base' i.e. the backhand area (and maybe the thumb and index finger), as a base platform, with the rest of the fingers/joints where the sensors are placed having the possibility of being detached/attached according to the necessities of the child. This since most therapies of CP focus on the function of the thumb and index/middle finger. Hence, there will not always be the need of the ring and little finger to be used.

The wearable device needs to be lightweight and cost-effective, hence the least number, and types, of sensors used, the better. This also provides for a less cumbersome algorithm since there are fewer inputs from different sources. However, the algorithm used to process the sensor data might need to be more complex using inputs from fewer sensors.

Thus, a balance must be reached between the number of sensors used and the types of sensors used. For example, a bending sensor and an accelerometer can be used to measure the angular and transactional/rotational movement of a finger, respectively. This means that the microcontroller needs to get input from 2 types of sensors, one which gives an analog output, and one with a digital one. The analog output of the bending sensor would need to be translated into a digital signal to be read by the digital microcontroller. After this extra analog to digital convertor (ADC) is inserted, then the digital input from the bendy sensor needs to be interfaced (algorithm in software) with the accelerometer data, which creates and an extra layer of complexity.

If, however, two accelerometers are used for both angular and translational/rotational movement of the finger, there is no need for an ADC in between, and the interface of both with the microcontroller is the same. However, an accelerometer does not give the angular position data of the finger directly (since it only outputs its own rotational data around its own axis), meaning a more complex algorithm needs to be used to extract this data from the accelerometer, with the data of the translational/rotational accelerometer as well.

After a consultation process with the different multidisciplinary experts in the SMART-Clap consortium, considering the information and constraints of the device mentioned

previously and to keep the hardware as less complex as possible, it was decided that the wearable device will be based only around IMU motion sensors and a controller device. This process is explained in more detail in Chapter 5 - Hardware Design.

The SMARTClap glove algorithm and hardware builds on the work of [13] and [14] to create a final product which can be used with a serious AR game for children with CP for their rehabilitation. Five fewer sensors where used than [13] while the concept of dual quaternions reduced the complexity of [14]'s algorithm.

This chapter has provided a literature review of hand and finger motion capture technologies, with an emphasis on those that have the potential to be used for computer-assisted therapies that encourage motivation in the rehabilitation of patients with CP.

The limitations and parameters within which this project needed to work were clearly described. The next chapter will describe the theory underpinning the different algorithms based on IMUs used to capture hand and finger motion, including definitions and a detailed explanation of Quaternion and Dual Quaternion theory and how these are used to sense rotations and translations.

Chapter 3 – Quaternion Theory

This chapter will discuss the theory of quaternions and justify why they where chosen for the SMARTClap's Motion CaptureAlgorithm (MCA), which is described in Chapter 4. This is done by first presenting different possible algorithms which are used to describe motion and rotation in 3D space. Next after describing how quaternions are used to describe rotations, the concept of dual quaternions is introduced, including the theory behind how they are used to describe both rotations and translations.

3.1 IMUs and Motion Capture

This section will briefly discuss Inertial Measurement Units (IMUs) and how they are used in motion capture algorithms.

The wearable device was based on IMU as motion sensors, placed on particular parts of the hand, specifically the back palm and the fingers, to capture the motion of the hand and fingers. Each of these IMUs consists of an internal accelerometer, magnetometer and gyroscope which output the data that could be used to determine the position and orientation of the sensor in three-dimensional (3D) space.

Data outputted from IMUs can take several formats that can represent movement of a rigid body in 3D space. The most common formats are Euler angles, Matrices, Axis Angles and Quaternions [61].

Euler Angles are the simplest to understand and visualise, consisting of three angles in $[X,Y,Z]$ 3D space. However, they suffer from the Gimbal Lock (or singularity) problem, when two of the dimensions 'lock' in the same plane, thus reducing the 3D space to 2D (Figure 3.1, [62]).

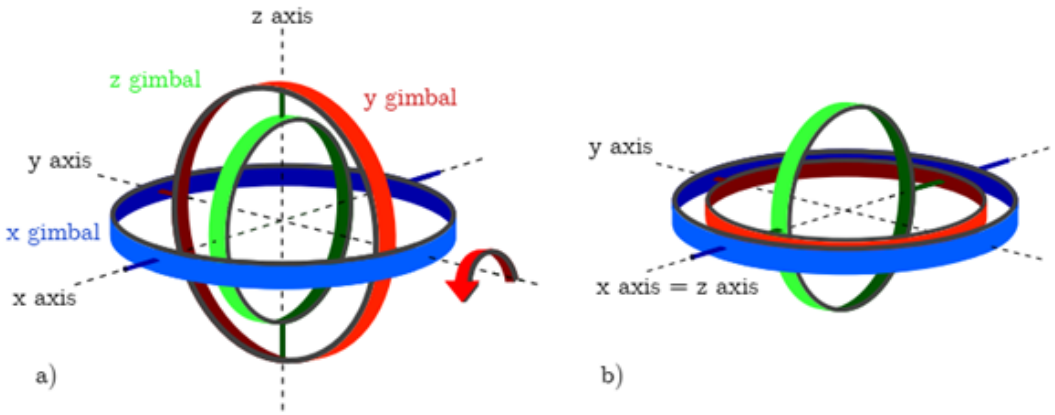


Figure 3.1: Gimbal Lock. In (a) there are 3 axes perpendicular to each other. In (b), the x and z axes are 'locked' in parallel, thus removing one dimension [62]

Axis-Angles are another straightforward way to represent a rotation of a body by an angle around an axis, and it uses only two parameters. However, there can be infinite possibilities of the same pair doing the same rotation (multiples of 2 π radians) and linear interpolation between two points can introduce discontinuities [61].

Transform Matrices use 3x3 matrices to rotate a point in 3D space using simple algebra, with the columns of the matrix representing the vectors in 3D space and the bottom row the translation. A multiplication of a vector coordinate by the matrix gives the new position of the vector. Though simple, once expanded, matrix transformation can be very time and memory consuming due to the number of computations needed to achieve a transformation [61].

Quaternions, while having more parameters than Euler and Axis angles (4 as compared to 3 and 2 respectively) do not have the problem of gimbal lock. Matrices neither have issues with gimbal lock, however they need to store 9 parameters (as compared to the 4 of quaternions) and quaternion multiplication is faster than matrix multiplication, making them more attractive computationally for real-time systems [61].

3.2 Quaternions and Dual Quaternions

3.2.1 Quaternion Definitions

Quaternions represent one type of a number system within the class of hypercomplex numbers. They were invented by William Rowan Hamilton in 1843 and can be seen as an extrapolation from 2D imaginary/complex numbers to describe rotations in space.

Instead of the one real and one imaginary elements associated with complex numbers $(a+bi)$, quaternions use one real and three imaginary numbers, as shown in (3.1).

The mathematical form of a quaternion q consists in a scalar element a and a 3-dimensional vector *[b c d]* , typically denoted as:

$$q = a + ib + jc + kd \tag{3.1}$$

where *a; b; c; d* are real numbers, and *i; j; k* (also known as the basis elements) denote the usual 3-dimensional orthogonal axes that follow the right-hand rule and $[b,c,d]$ represents a vector in 3D space.

The basis elements satisfy the following conditions:

$$\begin{aligned} i^2 = j^2 = k^2 = ijk = -1 \\ ij = -ji = k \\ jk = -kj = i \\ ki = -ik = j \end{aligned} \tag{3.2}$$

Quaternion q is sometimes also denoted as follows:

$$q = (a; v) \tag{3.3}$$

where a is the scalar component and v is the vector *[b c d]*.

Quaternion (and dual Quaternion) mathematical operations follow and expand on the same rules as those of complex numbers, as described by [61] , where, considering 2 quaternions $q1 = (a1; v1)$ and $q2 = (a2; v2)$:

$$\begin{aligned} &Addition : q1 + q2 = (a1 + a2, v1 + v2) \\ &Multiplication : q1q2 = [a1a2 - v1 \cdot v2, a1v2 + a2v1 + v1 \times v2] \\ &Conjugate : q^* = (a; -v) \\ &Magnitude : |q| = \sqrt{qq^*} \end{aligned}$$

A quaternion whose magnitude is equal to 1 is called a unit quaternion.

3.2.2 Quaternion Rotations

A special unit quaternion q can be used to determine the effect of a rotation by an angle θ of a vector p around an axis of rotation represented by a unit vector e passing through the origin, expressed as:

$$q = [\cos\frac{\theta}{2}, e\sin\frac{\theta}{2}] \tag{3.4}$$

For the unit vector *e = [x,y,z]*, the rotation quaternion can thus can be denoted as:

$$q = [\cos\frac{\theta}{2} + i(x\sin\frac{\theta}{2}) + j(y\sin\frac{\theta}{2}) + k(z\sin\frac{\theta}{2})]$$

or, as more normally denoted:

$$q = [q_0, q_1, q_2, q_3] \tag{3.5}$$

If we denote the original vector p as a quaternions q_p defines as $q_p = (0, p)$, we can obtain the quaternion representing the rotated vector p',denoted as $q\prime_p$ by the quaternion multiplication:

$$q\prime_p = q\ q_p\ q^* \tag{3.6}$$

where q is the rotation quaternion 3.4 and $q*$ is the complex conjugate of q. The new rotated vector can then be extracted from $q\prime_p$ (i.e. the vector part of quaternion $q\prime_p$).

3.2.3 Dual Quaternion Definitions

Quaternions by themselves can only handle object rotations. This concept was expanded by William Kindon Clifford in 1873 [63] to come up with Dual Quaternions, which, based on the dual-number system, can incorporate both rotations and translations in a single coordinate frame [64, 61]. A dual quaternion is composed of 2 quaternions, a real part and a dual part, with the real part denoting the rotation and the dual part denoting the translation away from the origin of rotation.

More specifically, a dual quaternion Q is represented as:

$$Q = q_r + \varepsilon q_d \tag{3.7}$$

Where q_r and q_d are the real and dual parts of the dual quaternion respectively, each of which is a quaternion defined as :

$$q_r = q_{ra} + iq_{rb} + jq_{rc} + kq_{rd}$$
$$q_d = q_{da} + iq_{db} + jq_{dc} + kq_{dd}$$

and ε is called the dual unit, that satisfies the property

$$\varepsilon^2 = 0; \varepsilon \neq 0$$

Consider two Dual Quaternions, $Q_1 = q_{r1} + \varepsilon q_{d1}$ and $Q_2 = q_{r2} + \varepsilon q_{d2}$. The mathemat-

ical operations used for dual quaternions are:

$$
\begin{aligned}
&Addition : Q_1 + Q_2 = q_{r1} + q_{r2} + \varepsilon(q_{d1} + q_{d2}) \\
&Multiplication : Q_1 Q_2 = q_{r1} q_{r2} + \varepsilon(q_{r1} q_{r1} + q_{r2} q_{d1}) \\
&Conjugate^1 : Q^* = q_r^* + \varepsilon q_d^* \\
&Magnitude : |Q| = QQ^* \\
&Unitcondition : ||Q|| = 1
\end{aligned}
$$

3.2.4 Dual Quaternion Rotations and Translations

The dual quaternion can be used to represent both a translation and rotation of a vector by using q_r as a quaternion representing a rotation and $\frac{q_d}{2}$ to represent the translation.

A pure rotation around axis e by angle θ is expressed by converting the rotation quaternion to a dual one and setting the dual part to 0:

$$
\begin{aligned}
Q_r &= q_r + \varepsilon q_d \\
&= [q; \varepsilon 0] \\
&= [\cos\frac{\theta}{2}; e\sin\frac{\theta}{2}], \varepsilon[0,0,0,0] \\
&= [(q_0, q_1, q_2, q_3) + \varepsilon(0,0,0,0)]
\end{aligned} \tag{3.8}
$$

Similarly, a pure translation by a vector t is expressed by a dual quaternion where the real part is set to 1 and the dual part to $\frac{t}{2}$:

$$
\begin{aligned}
Q_t &= q_r + \varepsilon q_d \\
&= [1;0], \varepsilon[0;\frac{t}{2}] \\
&= [(1,0,0,0) + \varepsilon\frac{1}{2}(0,x,y,z)]
\end{aligned} \tag{3.9}
$$

where $[x,y,z]$ represents the translation vector t in 3D space.

[1]The definition of the Dual Quaternion conjugate is not unique Sometimes it is defined as $Q^* = q_r^* - \varepsilon q_d^*$ or even $Q^* = q_r - \varepsilon q_d$

It can be shown that the above rotation and translation dual quaternions can be combined to provide a unit dual quaternion representing a rotation followed by a translation using:

$$Q_{rt} = Q_t \, Q_r \tag{3.10}$$

Thus, the dual quaternion of a vector p, Q_p , going through a translation and rotation can be expressed by

$$Q'_p = Q_{rt} \, Q_p \, Q^*_{rt} \tag{3.11}$$

from which the new position of p (i.e. p') can be extracted from Q'_p which will have the structure $Q'_p = [1;0] + \varepsilon[0;p']$.

Furthermore, according to [61], the vector p' can also be extracted from Q'_p from:

$$p' = 2 \, q_{p'd} \, q^*_{p'r} \tag{3.12}$$

where $q_{p'd}$ and $q_{p'r}$ are the translation and rotation quaternions that constitute Q'_p respectively.

Example of Dual Quaternion Rotation and Translation

Consider a rotation by angle $\theta = 180^\circ$ of a vector $p = [1,0,1]$ around the axis defined by unit vector $e = [0,0,1]$, i.e. the z-axis, followed by a translation along vector $t = [0,0,-1]$ i.e. along the z-axis by a distance of 1 unit, as shown in figure 3.2.

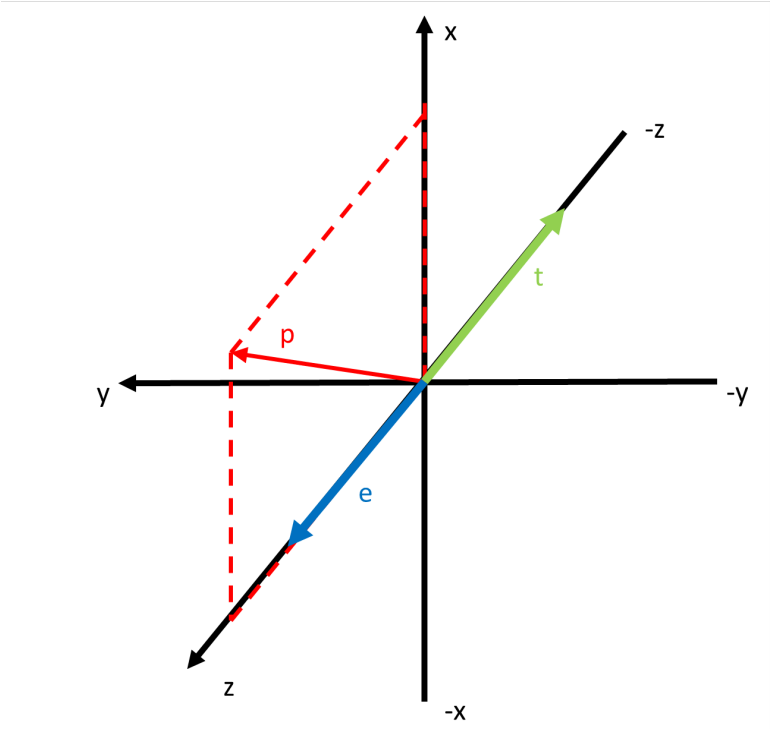


Figure 3.2: Visualisation of the original vector p at point [1,0,1] (red), with the rotation and translation vectors e and t respectively

Hence by (3.8), the dual quaternion of the original vector p is

$$Q_p = (1;0) + \varepsilon(0;[1,0,1])$$

From (3.4), the rotation quaternion of $q = [\cos 90, [e_1, e_2, e_3] \sin 90$. Hence, by (3.8) and (3.9) respectively :

$$Q_r = (0;[0,0,1] + \varepsilon(0;[0,0,0])$$
$$\textit{and}$$
$$Q_t = (1;[0,0,0]) + \varepsilon(0;[0,0,-\frac{1}{2}])$$

Using (3.10):

$$\begin{aligned} Q_{rt} &= Q_t\, Q_r \\ &= (1;[0,0,0]) + \varepsilon(0,[00-\frac{1}{2}])(0;[0,0,1] + \varepsilon(0;[0,0,0]) \\ &= (0;[0,0,1]) + \varepsilon(\frac{1}{2};[0,0,0]) \end{aligned}$$

Hence, from (3.8):

$$Q^*_{rt} = (0;[0,0,-1]) - \varepsilon(\frac{1}{2};[0,0,0])$$

Finally from (3.11):

$$
\begin{aligned}
Q'_p &= Q_{rt}\, Q_p\, Q^*_{rt} \\
&= (0;[0,0,1]) + (\frac{1}{2};[0,0,0])(1;[0,0,0]) + \varepsilon(0;[1,0,1])(0;[0,0,-1]) - \varepsilon(\frac{1}{2};[0,0,0]) \\
&= (0;[0,0,1]) + \varepsilon(-\frac{1}{2},;[0,1,0])(0;[0,0,-1]) - \varepsilon\frac{1}{2};[0,0,0]) \\
&= (1;[0,0,0]) + \varepsilon(0;[0,0,-\frac{1}{2}]) + (0;[-1,0,\frac{1}{2}]) \\
&= (1;[0,0,0]) + \varepsilon(0;[-1,0,0])
\end{aligned}
$$

from which the position of newly rotated and translated vector $p' = [-1,0,0]$ is obtained, visualized in figure 3.3

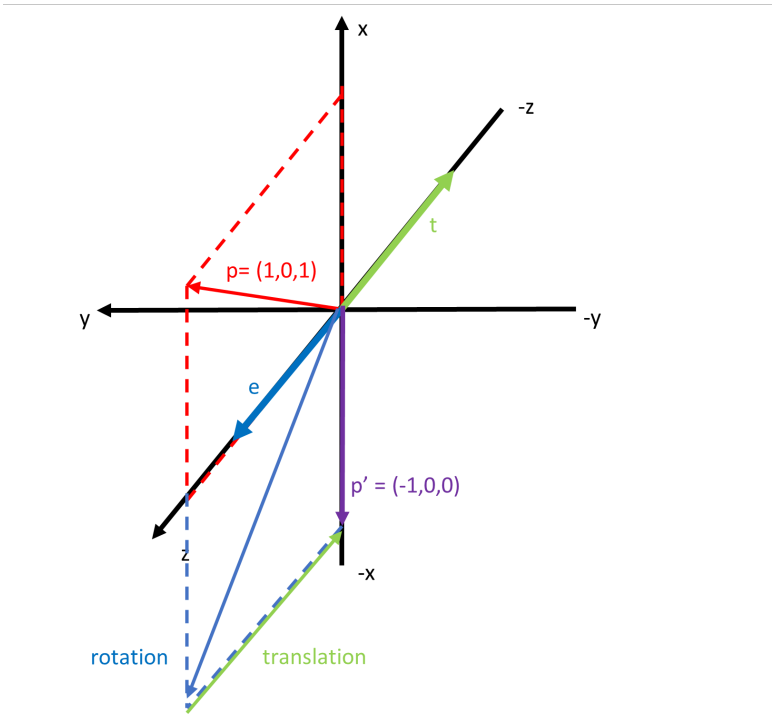


Figure 3.3: Visualisation of the final position of vector p at point [-1,0,0] (purple) after rotated around vector e (blue) and translated by vector t (green) respectively

This chapter has explained the theory behind the concept of quaternions and dual quaternions, and how they can be used to describe both rotation and translation in 3D space. The next chapter builds upon this information and upon the constraints of the wearable device presented in Chapter 2, to describe in detail how the Motion Capture Algorithm was designed and developed using dual quaternions

Chapter 4 – The Motion Capture Algorithm

With the knowledge of the hardware based on IMUs, to be used for the wearable device (hardware details will be described in full in Chapter 5) , the Motion Capture Algorithm (MCA) was developed in conjunction with sensing hardware. The wearable device is thus based on a set of inertial measurement units (IMUs), placed specifically on the back palm and the finger phalanges to capture their motion in three-dimensional (3D) space. Each of these IMUs consist of an internal accelerometer, magnetometer and gyroscope which output the required data to determine the position and orientation of the sensor in 3D space. The data is outputted in quaternion format, which, while being more complex when compared to other 3D-space-positioning representation methods, like Euler angles, provides a much simpler way of manipulation. The quaternions outputted from every IMU are captured by a controller device (Arduino Nano 33 BLE), which sends them wirelessly to the (Main Controller Unit) MCU based on a processing unit, such as a PC, to visualise them onto an Augmented Reality (AR) screen to be able to play the serious game.

The raw data outputted by the specific IMUs does not describe their actual position with respect to each other, or in a 3D space frame. It only describes the rotation of each specific IMU. The aim of the MCA is to get the IMUs to describe their position and rotation in relation to each other on the screen. For instance, indicating the rotational position of the thumb in relation to the wrist.

This chapter will start by describing the physiological structure of the hand, including all the bones, joints, and the degrees of freedom (DoFs) (i.e. in how many directions can each specific joint rotate) of every joint of the hand. After this, the hand model on which the algorithm was based is described. This includes:

- description of the coordinate systems of all the joints (both global and local);
- the setting out of the hand motion constraints assumed by and included in the algorithm, describing also how the cost and bulkiness is reduced by extrapolating certain finger motions on other IMUs without having to place an IMU on each phalange of the fingers;
- the hand measurements needed to be able to create the model of the hand;
- how the estimated starting position of the IMUs on the hand is calculated (the calibration); and finally
- a short description of the motion capture algorithm based on quaternion theory

Within this section, physiological hand measurements which needed to be used for the algorithm were also collected and presented. This is not readily available in the literature and will be considered in the future for the possibility of publishing it as a novel study.

Finally, the actual algorithm and the procedure of how data from each IMU was used to capture the motion of each finger phalange and of the whole hand is described, based on the detailed description of Quaternion Theory presented in Chapter 3. This starts by describing the initial position of the hand, the theory behind the rotation and how the initial positional readings were used to avert complex and time consuming calibration methods. The theory of rotation between 2 joints is then described, followed finally by the actual description of the MCA, specific to the SMARTClap glove.

4.1 Hand Structure

The bones of the hand can be categorised into three different classes, namely the carpal bones, the metacarpal bones and the phalanges (Figure 4.1[65]).

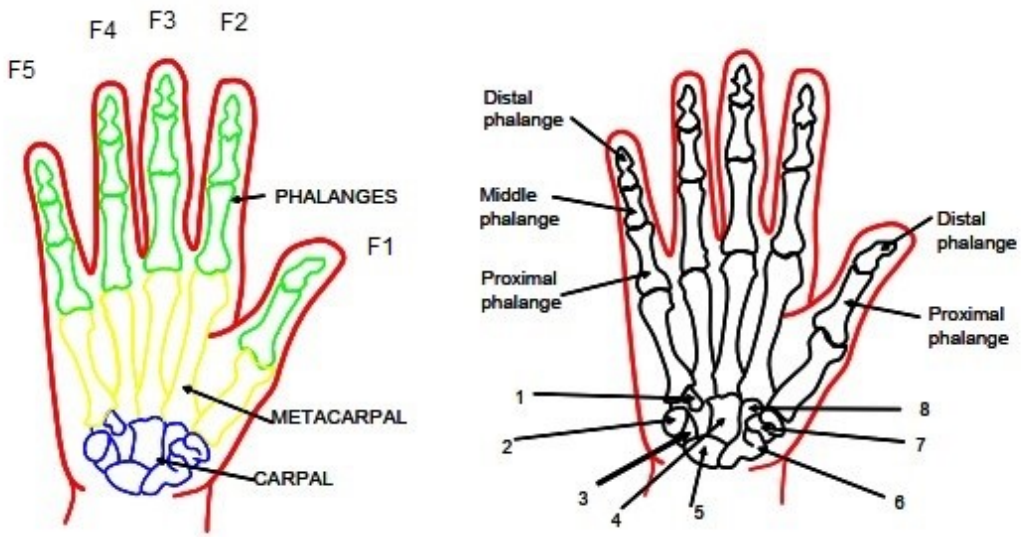
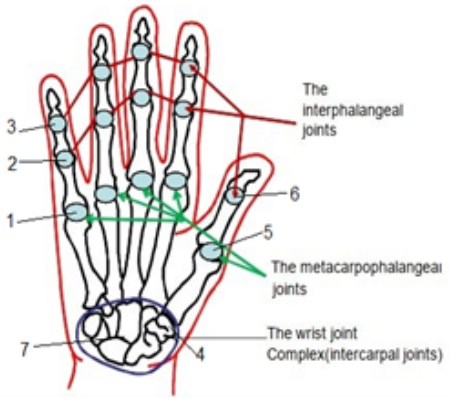


Figure 4.1: Physiological hand bone structure (Source: [65])

The carpal bones are the eight small bones that form the wrist (carpus), namely the hamate (1), the pisiform (2), the triquetrum (3), the capitate (4), the lunate (5), the scaphoid (6), the trapezium (7), and the trapezoid (8). The metacarpals (MCPs) are a set of 5 bones that connect the carpal bones in the wrist to each of the five fingers, and which form the palm of the hand. The phalanges are the bones that make up the fingers (named as F1 (thumb), F2 (index), F3 (middle), F4 (ring) and F5 (little)). F2 to F5 have 3 phalanges, the Proximal (attached to the Metacarpal bone and nearest to the wrist) – (PP), the Intermediate (IP) and the Distal (DP) phalanges. The thumb (F1) has two phalanges, the Proximal and the Distal.

The carpal, metacarpals and phalanges are held together by joints as shown in Figure 4.2. The carpal (wrist) joint holds the metacarpals. The phalanges are attached to the MCPs by the metacarpal-phalangeal joints (MCPj). The phalanges of fingers F2 to F5 are connected by inter-phalangeal joints (IPj): the distal interphalangeal joint (DIPj) and the proximal interphalangeal joint (PIPj). Finally, the thumb metacarpal is assumed to join the wrist using a ball joint, called the thumb-carpo-metacarpal joint (TCMCj), while the finger itself is connected to its metacarpal by the thumb-metacarpal phalangeal joint (TMCPj) and its two phalanges are joined by the thumb interphalangeal joint (TIPj). Figure 4.2 also shows the number of degrees of Freedom (DoFs) for each joint. These describe the direction of possible motion of each component of the hand. This knowledge and that of how the joints are connected together was essential in being able to describe a workable hand model (described in the next section) on which the algorithm was based.



Number	Joint Name	Abbv	DoFs
1	Metacarpal-Phalangeal Joints	MCPj	2
2	Proximal Interphalangeal Joint	PIPj	1
3	Distal InterPhalangeal Joint	DIPj	1
4	Thumb Carpo Metacarpal Joint	TCMCj	2
5	Thumb-Metacarpal phalangeal Joint	TMCPj	2
6	Thumb Interphalangeal Joint	TIPj	1
7	Carpal (Wrist)		6

Figure 4.2: Physiological hand joints and respective degrees of freedom (DoFs) [65]

4.2 The Hand Model

Numerous models of the hand have been described in literature [65, 14, 55, 66, 67, 68, 69, 70], and the model proposed and used in this development is based on these works.

The joints allow every part of the hand to move according to a different number of degrees of freedom (DoF), as tabulated in figure 4.2. These describe the direction of possible motion of each component of the hand. A normal functional hand has a total of 27 DoFs [71] shown in table 4.1 and figure 4.3 [13], with the possible motions of each joint shown in figure 4.4 ([13]).

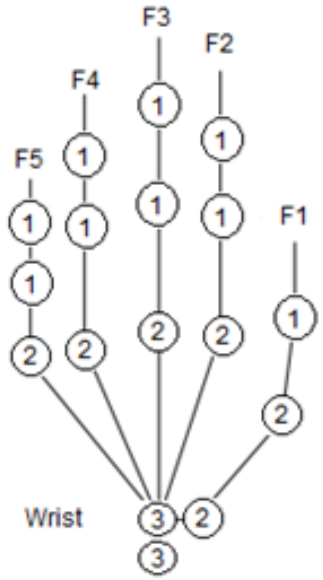


Figure 4.3: Hand joints model and the number of Degrees of Freedom for each joint (in circles) (Source: [13])

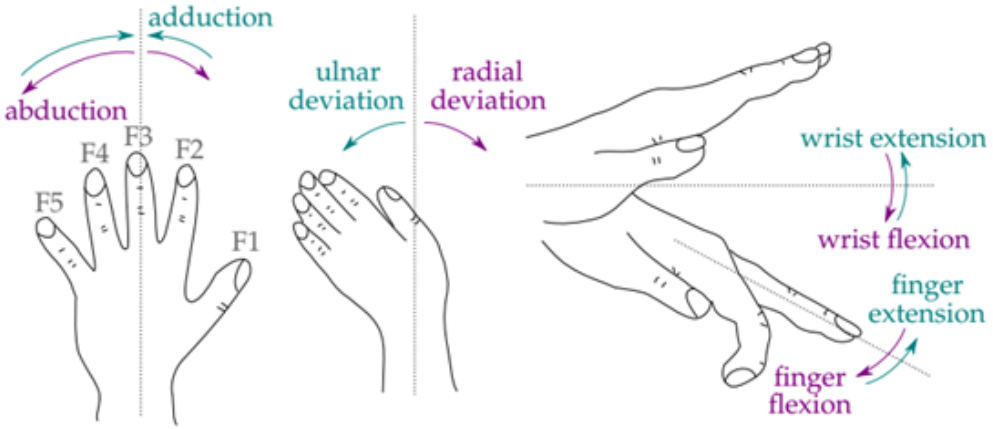


Figure 4.4: Hand and finger motions (Source: [13])

The wrist is the only part of the hand that has six degrees of freedom (three for rotation and three for translation). This means that its rotational movement can be tracked around the three axes of rotation (x, y, z). For this model, the translational movements are not included and thus, all the motions being considered are rotational. The thumb has five DoFs, two in the TCMCj and TMCPj respectively and one in the DIPj. The MCPs have two DoFs, since the fingers can only move in two directions, sideways (abduction and adduction) and up or down (flexion and extension). The phalanges of each finger (F2 to F5) can only flex and extend, and thus, they only have one DoF each.

In the simplified version of the proposed model, a DoF (the adduction/abduction movement) for the TMCPj will be removed since it follows a similar movement of the TCMCj.

Also, as mentioned previously, the translational movements of the wrist will be neglected thus concentrating only on the rotational movements. Hence the DoFs considered for this simplified model are shown in the last column of Table 4.1 .

Table 4.1: Degrees of Freedom for the joints of the fingers (F1 - F5) and the wrist used in SMARTCLAP model

Component	**Joints**	**DoFs**	**DoF Description**	**Total DoF**	**SmartClap DoFs**
F2 -F5	MCPJ	2	Abduction/Adductoin + Flexion/Extension	8	8
	PIPj	1	Abdcution/Adduction	4	4
	DIPj	1	Abdcution/Adduction	4	4
F1	TCMCj	2	Abduction/Adductoin + Flexion/Extension	2	2
	TMCPj	2	Abduction/Adductoin + Flexion/Extension	2	1
	TIPj	1	Abdcution/Adduction	1	1
Wrist		3	Ulnar + radial Deviation + Rotation	3	3
		3	Translation	3	0
Total				27	23

However, the hand and finger movements are too complex and articulated to be simulated perfectly, hence one needs to define a hand model that includes certain constraints and assumptions to simulate as much as possible the movements of the hand. The hand model in this work is based on a number of previous studies and will be defined by:

1. A system of local coordinates (LCS), one for each independent moveable part of the hand, incorporated within a Global Hand Coordinate System (GHCS), which describes the motion of the whole hand [12].

2. The motion constraints of the fingers and the phalanges depending on the joints attached to them as described by [55, 14].

3. An intra-dependence of the movement of certain joints on others. This will help simplify the model, calculations, and hardware by removing five possible Inertial Measurement Units (IMUs) from the hand and instead estimating the movement of the dependent joints by calculation, similar to the model proposed by [14].

4. Calculations on the positioning of the IMUs on the fingers which depend on the hand measurements [56].

5. The use of Quaternions to estimate the orientation and positioning of the fingers [13].

4.3 Coordinate Systems of the Joints

The International Society of Biometrics (ISB) has standardised a definition of the coordinate system of the body [12]. This model will base the movements of the hand on this Joint Coordinate System (JTS). Using this coordinate system, one can move from the wrist to the fingertips, one bone at a time, with each bone having its own local coordinates, to describe the movements of each segment. These local coordinates (LCs) are then included in the Global Hand Coordinate System (GHCS), which describes the global wrist motion,defined as the movement of the third metacarpal with respect to the radius (refer to Figure 4.5). This is because wrist movements will affect all the segments of the hand, independently of their individual movements.

Figure 4.5: 3^{rd} (Middle) Metacarpal and Radius (stock photo). The Global Coordinate System (GCS) of the hand will move as the former does with respect to the latter.

The ISB recommends the point of origin of the LCs to be at the centre of mass of each phalange and at the volumetric centre of the carpals (wrist). In the proposed model, the carpals and metacarpals will be considered as one rigid plate, with the start of the Global Coordinate System (GCS) being in the centre of the carpals (where the capitate carpal is). Each of the phalanges of F2 to F5 will have their LCs in the same orientation as the GCS. Each phalange (or bone) will be considered as a beam with the LCs' origin at the middle of the phalange, also assuming the phalange width is homogenous (unlike what is shown in figure 4.6). This will also be the position where the IMU modules of the proposed wearable device will be positioned (figure 4.6). The hardware design of the IMU modules is described in detail in Chapter 5.

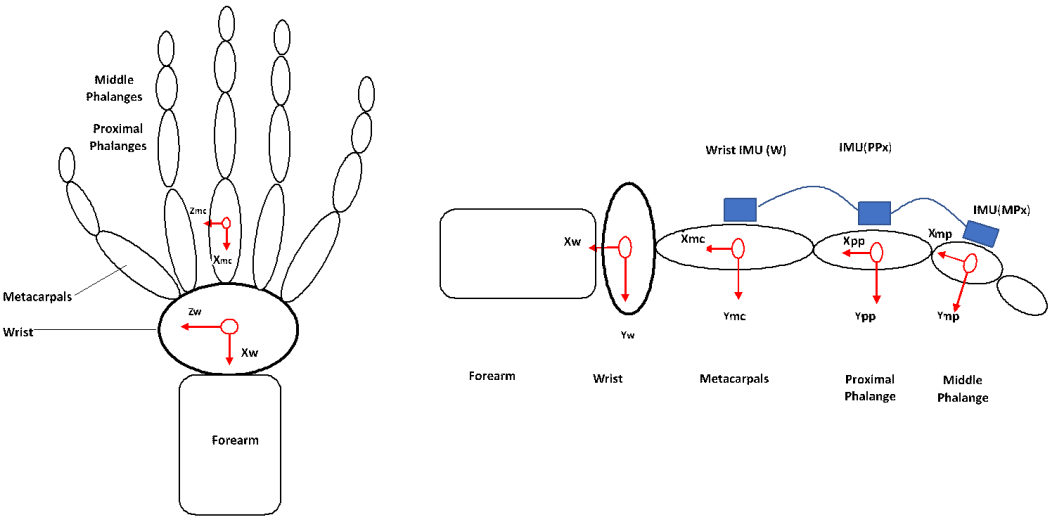


Figure 4.6: Local and global coordinate systems of the hand model

4.3.1 Wrist Coordinate System

The Wrist Coordinate System (WCS) has its origin at the volumetric centre of the wrist, with (for the right-hand placed palm down), the X-axis (X_w) pointing backwards toward the forearm, the Z-axis (Z_w) pointing inside toward the thumb (i.e. left), and the Y- axis (Y_w) pointing downwards towards the palm (figure 4.6). For the left hand, the x- axis points forward towards the fingers, the z- axis points inside towards the thumb (i.e. right) and the y-axis points upwards towards the backhand.

4.3.2 Metacarpal Coordinate System

The Metacarpal Coordinate System (MCS) is described similarly to the WCS. Positioning the right - hand palm down, the X-axis (Y_{mc}) points towards the wrists, the Y-axis (X_{mc}) points downwards and the Z-axis (Z_{mc}) inside towards the thumb. Here, the 3^{rd} metacarpal is often used to describe the movement of the whole hand as well, due to the position it has, as a continuation of the forearm and connected to the centre of the wrist. Hence, the 'wrist' IMU will be positioned as close as possible to the centre of this metacarpal that, for the proposed model, which considers the carpals and metacarpals as one rigid body, would be at the centre of the back palm.

4.3.3 Phalanges Coordinate System

The methodology of the Phalanges Coordinate System (PCS) is like that of the MCS. There are 14 phalanges in all, three each for F2 to F5 and two for the thumb. Note that to make the SMARTCLAP device less complex, cheaper, and as minimal as possible, IMUs were placed only on the first two phalanges of each finger (i.e. the Proximal

and Intermediate). The movement of the distal phalange (and hence the fingertip) were approximated according to constraints and the dependency of the rotation of the DIP joint on the rotation of the PIP joint, accepted in literature [55, 56]. These constraints are explained in the next section.

4.4 Motion Constraints of the hand

According to [55], there are two types of constraints of the finger joints, intra- and inter-finger constraints.

4.4.1 Intra-finger Constraints

Intra-finger constraints are based on the dependency of the joint's movements within the same finger. Thus, for example, due to the constraint of the Distal-Interphalangeal joint (DIP), the Distal Phalange's movements have a dependence on the Intermediate Phalange movement in fingers, while it will rotate with the same angle as the Proximal phalange when this latter is moved. The Thumb's T-DIP – T-MCP has a different constraint than the other four fingers, whose constraints in this regard are similar [56]. Since no IMU shall be used to track the position of the Distal Phalange (for cost and complexity reasons as explained in Chapter 5), the motion of this phalange was approximated according to the following equations (from [56]):

$$\theta_{DIP} = 0.88\theta_{PIP} \tag{4.1}$$

for the index, middle, ring and little fingers, and

$$\theta_{TDIP} = 0.77\theta_{TMCP} \tag{4.2}$$

for the thumb.

Since the other phalange rotations shall be tracked using IMUs, these were the only equations needed.

4.4.2 Inter-Dependence Motion Constraints

Inter-dependence constraints are the ones between the fingers themselves. This is the effect on a neighbouring finger when a particular finger is flexed/extended or abducted /adducted (Figure 4.7).

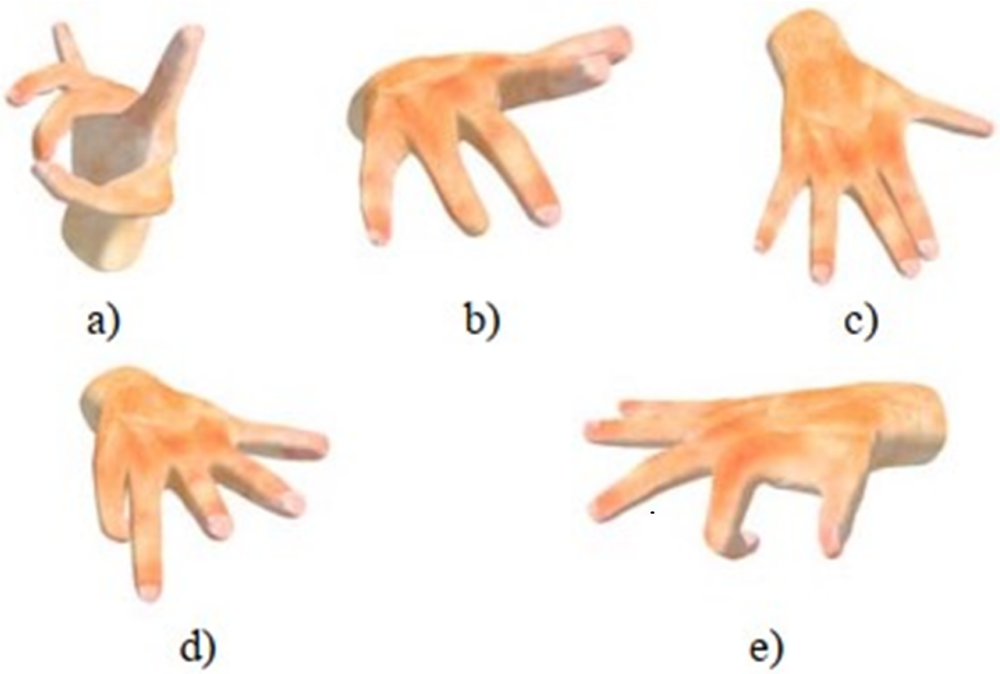


Figure 4.7: Main Hand motion constraints (from [55])

Since most of the phalanges will have an IMU to track their motion, not all these constraints were used. In fact, they were mostly used to set a maximum and minimum angle for every phalange flexion/extension and finger abduction/adduction in the computations.

A number of studies have measured the angles of flexion/extension and abduction/adduction of the hand in a number of real-life positions ([71, 72, 73, 74]). However, for this work, the constraints used are the maximum joint angles (in degrees) for when the hand is in a neutral position. Specifically, though there are numerous works in literature ([75, 76, 77, 78, 79, 80]), this work will follow the constraints (in degrees) as used by Lisini et. al in a similar work [14], tabulated in Table 4.2.

Furthermore, [56] has shown that these constraints are similar for both the left and the right hand, and at different hand positions. The mentioned research was done on adults between 20 and 70 years old, and not children. Hence, since no study was found specifically targeting children between 7 and 11 years old, it was assumed that the constraints are similar to those of adults.

Table 4.2: Maximum Angle constraints for each joint (in degrees)

Digit	Joint	Flexion	Extension	Abduction / Adduction
Thumb	TCMC	90	15	60
	TMCP	80	0	0
	TIP	80	10	0
Index	MCP	90	40	60
	PIP	110	0	0
	DIP	90	5	0
Middle	MCP	90	40	45
	PIP	110	0	0
	DIP	90	5	0
Ring	MCP	90	40	45
	PIP	120	0	0
	DIP	90	5	0
Little	MCP	90	40	50
	PIP	135	0	0
	DIP	90	5	0

4.5 Hand Measurements

In order to determine the position of the IMUs (i.e. the phalanges) on the child's phalanges, a number of measurements of the hand were taken beforehand.

Since the IMUs shall be positioned on the centre of the proximal and intermediate phalanges, the sizes of these need to be known, as does the size of the distal phalange for estimation of the fingertip position. Measuring all the phalanges of every child every time they need to start to use the wearable device would be a tedious process and the accuracy depends on who is doing the measuring. There are several studies which have defined the functional length ratios between the different phalanges, which ratios are summarised in Table 4.3 [81, 82]. Thus, by measuring the lengths of the fingers (from the fingertip to the Metacarpal joint), one can easily compute the length of the specific phalanges of each finger through these ratios

Table 4.3: Ratios of lengths of different phalanges. L = Length, p=proximal, i =intermediate, d = distal, taken from [81, 82]

	Thumb		Index		Middle		Ring		Little	
	$\frac{Lp}{Li}$	$\frac{Li}{Ld}$	$\frac{Lp}{Li}$	$\frac{Li}{Ld}$	$\frac{Lp}{Li}$	$\frac{Li}{Ld}$	$\frac{Lp}{Li}$	$\frac{Li}{Ld}$	$\frac{Lp}{Li}$	$\frac{Li}{Ld}$
Ratio	n/a	1.5	1.86	1.24	1.72	1.36	1.7	1.29	1.91	1.06

Thus, two length measurements were taken for each finger, the length between the fingertip and the wrist (l) and the length from the fingertip to the MCP joint (lf), as

shown in Figure 4.8 for the thumb and middle finger. The lengths of the specific phalanges Figure 4.9 where then calculated from the ratios in Table 4.3.

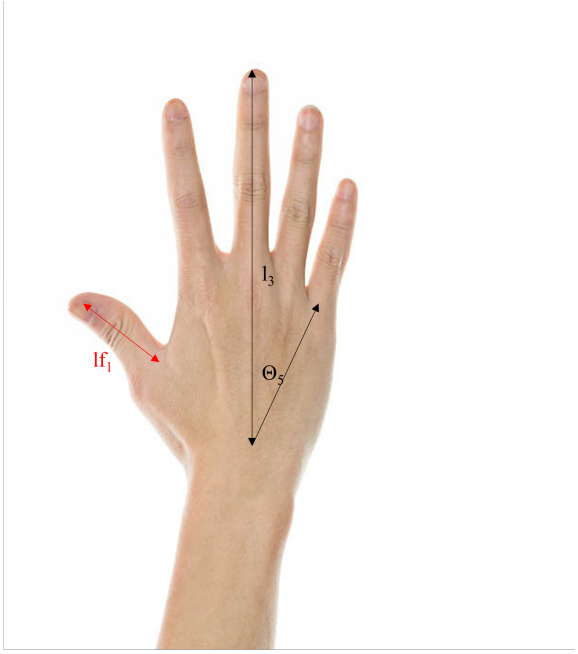


Figure 4.8: Hand measurements to be taken for each child, for each finger

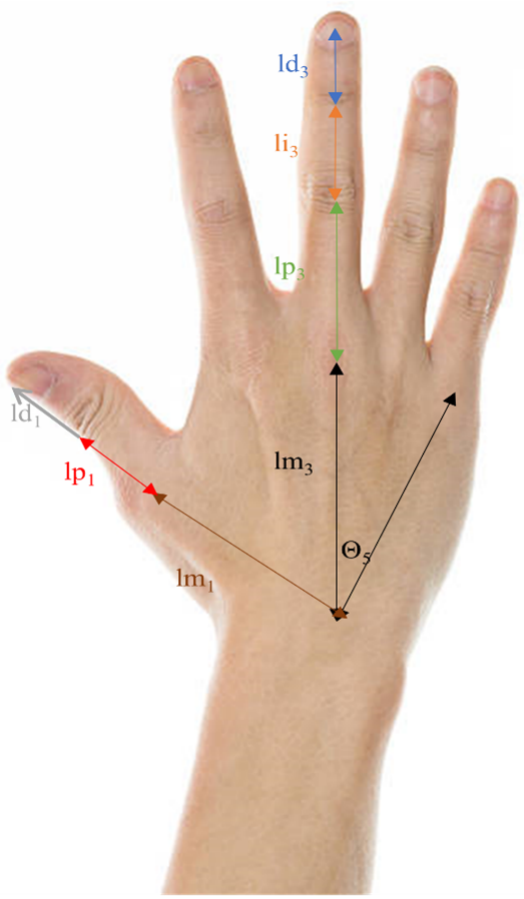


Figure 4.9: Hand measurements calculated after measurements in 4.8 are taken

Thus, for example, for the middle finger, if l_3 is measured to be 17.5cm and lf_3 is 9.5cm, then, from table 4.3:

$$lm_3 = 8cm \quad ld_3 = \frac{9.5}{(1.72 * 1.36) + 1.36 + 1} = 2.09cm li_3 = 1.36 * 2.09 = 2.59cm \quad lp_3 = 1.72 * 2.59 =$$

Additionally, determining the estimated position of the IMUs required the angle at which each finger protrudes out from the wrist, as is the example of the little finger in Figures 4.8 and 4.9. This helped in determining the starting position of each finger compared to the wrist. Unfortunately, there is nothing in literature that confirms what theses angles are, if they are similar for all people with normal motor functions, if they change with growth and what the relationship, if any, between the angles is. In [13] a reference is made to the angle between the wrist joint centre and the T-CMC (Thumb-carpometacarpal) joint as being 58°, but nothing more. The authors of [13] took their measurements from their study cohort without publishing them. From early trials done on two people involved in the SMARTCLAP consortium, the thumb angle was found to be similar to that of [13]. Hence, it was assumed that these angles must be similar for most people. However, to test if this was the case, it was decided to make a small study

of these angles on more people. The preliminary study was done on six participants (all male, age ranging from 23 to 56), using members from the SMARTCLAP consortium and the average of all the measurements for every finger was taken as the angle for the model. Taking the angle between the wrist and the MCPj of middle finger as 0°, the four angles of the other MCPs were measured using a goniometer. The results are shown in Table 4.4 including the confidence intervals.

Table 4.4: Average angle measurements between wrist and 3^{rd} MCPj and wrist and other finger MCPjs

	Thumb	**Index**	**Middle**	**Ring**	**Little**
Angle (°)	-52.5	-15.8	0	14.5	28.8
95% CI (%)	8.5	5.3	n/a	5.28	7.75

Data regarding the gender, age and dominant hand of each participant was also taken so that more studies on the relationship between this and the angles can be carried out in the future.

4.5.1 Sensor-positioning calculations from hand measurements

Starting with the wrist joint as the origin, the palm IMU shall be positioned at the midpoint between the wrist joint and the MCPj of the middle finger. The Middle finger's IMU positions are calculated by adding the length of MCP 3 (using l and l_f measured in Section 4.5 (figure 4.8)) and the phalange lengths calculated from table 4.3. According to the Coordinate Systems of the Joints described previously, the IMUs are positioned (initially) along the x-axis of the hand. Hence for the IMUs on the middle finger, the position vector in 3D space (in the format $[x_{\text{pos}}, y_{\text{pos}}, z_{\text{pos}}]$) was calculated as:

$$F3_1 = [lm_3 + \frac{lp_3}{2}, 0, 0] \tag{4.3}$$

$$F3_2 = [lm_3 + lp_3 + \frac{li_3}{2}, 0, 0] \tag{4.4}$$

where $F3_1$ and $F3_2$ are the positions of the IMUs on the proximal and intermediate phalange of finger 3 respectively, l_m is the length between the metacarpal joint and the wrist, and l_p and l_i the lengths of the proximal and intermediate phalange respectively.

The rest of the IMU positions are calculated in a similar fashion but rotated at an angle around the wrist joint according to the angles of Table 4.4. Hence, as an example, the index finger's IMU positions (using the same format as the Middle Finger), are calculated from:

$$F2_1 = [cos\theta(lm_2 + \frac{lp_2}{2}), 0, sin\theta(lm_2 + \frac{lp_2}{2})] \tag{4.5}$$

$$F2_2 = [cos\theta(lm_2 + lp_2 + \frac{li_2}{2}, 0, sin\theta(lm_2 + lp_2 + \frac{li_2}{2}] \tag{4.6}$$

The final model (for the right hand), was programmed using Python and Visual Python, as can be seen in Figure 4.10.

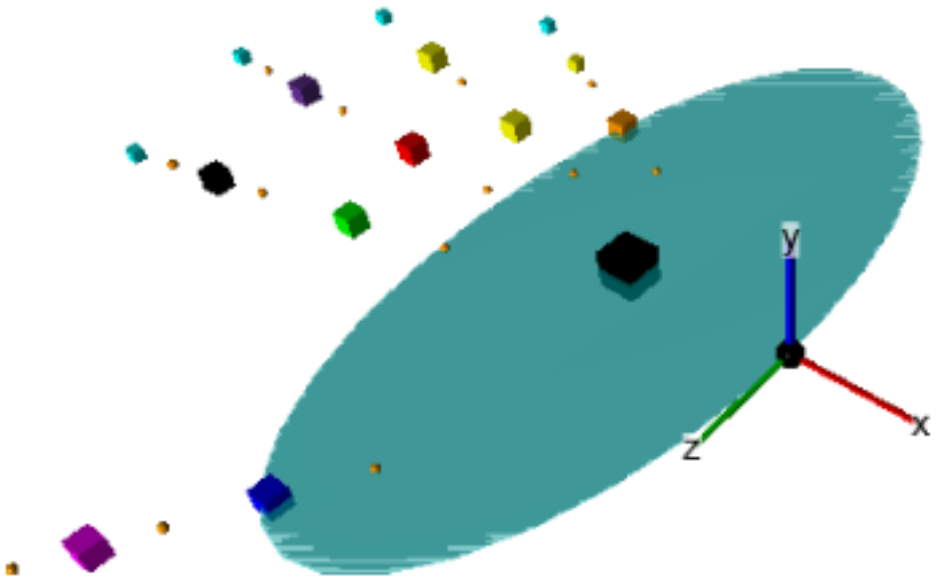


Figure 4.10: Computer model output of hand joints and IMUs (using Visual Python). Spheres represent joints, and cubes represent the estimated positions of the IMUs.

This section discussed how the model of the hand was designed to estimate measurements and constraints of the hand. This was used at the start of the game program to help calibrate the SMARTCLAP device once donned by the user and kept static for 25 seconds until calibrated. Each specific IMU sensor (the design of which is described in Chapter 5) was then programmed (automatically) with its specific position in 3D space to be displayed on the Augmented Reality (AR) screen. Once set up, the sensors (IMUs) will describe their position and motion in 3D space with respect to each other through the concept of quaternions, as presented in the next section.

4.6 Motion Capture Algorithm using Quaternions

This section describes the MCA and how the quaternions were used to define the position of the fingers and hand in 3D space.

4.6.1 Initial Positioning

Figure 4.11 shows the position of the main components of the wearable device, which were required to capture the movement of one finger. A controller board was placed on the forearm, a sensor (IMU) module was connected on the back palm of the hand, and two IMU modules were each placed on the proximal and intermediate phalanges respectively. The electronic design of these boards is described in Chapter 5. However, it is necessary to point out that the IMU selected (ICM-20948 from Invensense [11]), outputs data in quaternion format directly (which are transformed in dual quaternions in code, as described in Section 2 of Chapter 3), and hence there was no need for an external 'fusion algorithm' to be created so as to get the data from the accelerometer, gyroscope and magnetometer within the IMU and fuse them into quaternions. This was done internally in the IMU contrary to other systems such as [13, 14].

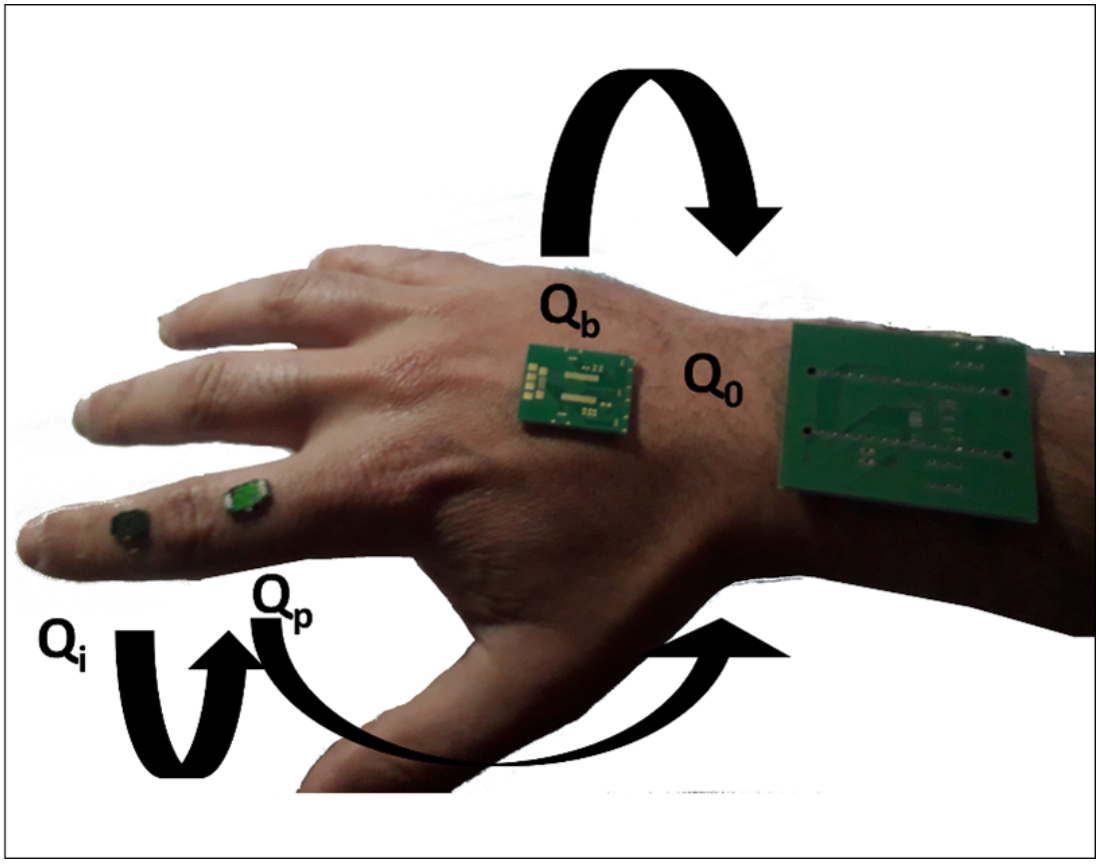


Figure 4.11: Positioning of IMUs on the wrist with the corresponding reference IMU being pointed at. Q_0 is the origin imaginary quaternion on the wrist, and reference to Q_b and Q_p. Q_p is the reference IMU of Q_i

Initially, let every sensor's initial position on the hand be described by a dual quaternion Q_x, where x denotes the phalange or back palm on which the sensor is positioned. Thus, Q_b, Q_p and Q_i denote the initial positions of the back palm, proximal and intermediate phalanges respectively. Q_0 denotes the origin quaternion, a specific reference point, set to position (0,0,0) on the wrist. Thus, the wrist is the origin position of the Global

Hand Coordinate System (GHCS) described in Section 4.1 (Coordinate Systems of the Joints) and the reference point from where the model is built. All other sensors can trace back their position with reference to this quaternion Q_0 (point).

Since these initial dual quaternions have no rotation, following equation (3.9) in section 3 of Chapter 3, they were thus initially described as pure translation dual quaternions:

$$Q_x = 0 + \frac{1}{2}[1, x, y, z]$$

where $[x, y, z]$ represents the vector position of the specific sensor in 3D space . Thus for example, if the sensor on the back palm Q_b is situated at 5cm along the x-axis from the origin, then its original dual quaternion would be denoted as:

$$Q_b = 0 + [1, 2.5, 0, 0]$$

4.6.2 Initial positioning for Rotation

The normalised rotational quaternion outputted by the ICM20948 sensors describes a certain 'non-zero' angle of rotation depending on the sensor's actual angle and direction. The only point that a 'zero' angle (i.e. no rotation, with an output rotation quaternion $q = [1, 0, 0, 0]$) is given is if the sensor is positioned perfectly flat, and with the x-axis pointing towards the Earth's magnetic North direction. This is due to

- the magnetometer encompassed inside most IMUs, on which data the sensor calculates is tilt, yaw and heading (pitch, yaw, roll),
- lack of accuracy of the accelerometer

as described in detail in the calibration application note of another IMU, the LSM303 from STMicroelectronics [83].

Hence, before normal operation, the sensors have traditionally needed to be calibrated using complex equations and also by rotating them in a 'figure-of-8' manner ([83, 84, 85], that is both impractical and time consuming.

Furthermore, the output quaternion has a rotation angle of between -180°and 180°. This means that depending on the heading and tilt, the sensor can give an initial reading of any angle within that range. This was not ideal for this work since it was imperative that the initial angle be 0°.

In order to bypass these complex calibration techniques, the initial Euler Angles (pitch, yaw, roll) outputted by the quaternion (equation (4.7)) were used as the reference angle throughout the full usage time of the device. In order to obtain this, it was required that

the whole device with all the sensors remained unmoved for 25s after initialization (5s). This downtime, calculated through trial and error, allowed for the initial rotation angle of all the sensors to stabilize. Thus, the sensors and the glove could be used starting from any heading, and not necessarily when pointing along the perfect magnetic North. After the stabilization period and to compensate for glitches, the device needed to be stationary (not necessarily pointing in same direction of the stabilization period) for a further 5s, to give time to a number of calibration points to be taken and averaged. This value was then taken to be the initial orientation/angle.

The conversion equations from the quaternions coming from the IMU (in the form $[q_0, q_1, q_2, q_3]$ to Euler angles (yaw (ψ), pitch (θ), and roll (ϕ)) are defined as [13] :

$$\begin{aligned}\psi &= \arctan \frac{2(q_0q_1 + q_2q_3)}{1 - 2(q_1^2 + q_2^2)} \\ \theta &= \arcsin(q_0q_2 + q_3q_1) \\ \phi &= \arctan \frac{2(q_0q_3 + q_1q_2)}{1 - 2(q_2^2 + q_3^2)}\end{aligned}$$

However, *arcsin* and *arctan* function implemented in computer languages only give results between -90°and 90°(i.e.180°range), and so these equations are not adequate to achieve the whole 360°range required in the rotations and orientation of quaternions in Euler space. Thus, the $arctan2(x,y)$ function is used in order to achieve the required orientation in space, giving angle results with the range $-\pi$ to π radians. This changes the quaternion to Euler equations to:

$$\begin{aligned}\psi &= \arctan 2[2(q_0q_1 + q_2q_3), 1 - 2(q_1^2 + q_2^2)] \\ \theta &= \arcsin(q_0q_2 + q_3q_1) \\ \phi &= \arctan 2[2(q_0q_3 + q_1q_2), 1 - 2(q_2^2 + q_3^2)]\end{aligned} \tag{4.7}$$

Thus, having established the initial reference angles (yaw, pith, roll), after each quaternion acquisition (for every sensor), the respective angle of rotation was calculated from the rotation quaternion and this angle was subtracted from the reference angle. The difference was used to calculate the new angle of rotation after the acquisition as well as applied to the motion constraints of the hand as described in section 4.4. This latter option was also used to help remove any glitches in the data, for example, if an erroneous quaternion occured whose angle described a rotation well outside the physiological constraints of the finger/hand described.

4.6.3 Rotation Theory

In this section, two kinds of rotations are described. There is the rotation due to the Local Coordinate System (LCS) and the rotation due to to the Global Hand Coordinate System (GHCS), both of which were explained in section 3 of this chapter. The LCS rotation describes the rotation of a single sensor (i.e. phalange) when only it moves, whereas the GHCS rotation describes the rotation of the sensors with respect to the whole hand (e.g. when the backpalm is rotating around the wrist, all the fingers move with the same angle around the wrist, even if the specific phalanges are not moving). As mentioned previously, the sensor will only output a rotation quaternion, not a dual one. Let q_{r} be the incoming rotational quaternion outputted by each sensor. Furthermore, let Q_{in} denote the incoming dual quaternion of the sensor, achieved by adding the dual (or translation) quaternion q_d to q_{r}, as described in equation (3.7)

$$\begin{aligned} Q_{\mathrm{in}} &= q_{\mathrm{r}} + \varepsilon q_d \\ &= [q_0, q_1, q_2, q_3] + \varepsilon[0,0,0,0] \end{aligned}$$

Then, the LCS rotation of the sensor (Q_{xrot}) with respect to its initial position (denoted by Q_x) is described by the mathematical operation:

$$Q_{\mathrm{xrot}} = Q_{\mathrm{in}} Q_x Q_{\mathrm{in}}^*$$

where Q_{in}^* is the complex conjugate of dual quaternion Q_{in}.

The GHCS rotation of the sensor was then calculated from the equation [14]:

$$Q_{\mathrm{xout}} = Q_{\mathrm{xref}}^* Q_{\mathrm{xrot}} \tag{4.8}$$

where Q_{xref} is the output dual quaternion of the joint around which the sensor is rotating, and which contains a reference to the rotation of previous sensors (described in more details below), while Q_{xout} is the final positioning quaternion of the sensor, from which the final position , Pos, in 3D space of the sensor can be extracted using the equation (from equation (3.12)):

$$Pos = 2\, q_d\, q_r^* \tag{4.9}$$

where q_d and q_r are the translation and rotation quaternions that constitute Q_{xout} respectively.

The position of the distal phalange virtual sensor was then calculated from the position

and angle of the intermediate phalange using equations (4.1) and (4.2).

The SMARTClap MCA

The hand, including the wrist (Origin) and the back-palm, was modelled as a series of vectors (phalanges), connected by joints (Points *A*,*B*, *C* and *D*), with sensors (S0, S1, S2 and S3) positioned in the middle of the vectors (phalanges), as shown in Figure 4.12. Thus, the Metacarpal joint (MCP) is represented by Point A, Proximal-Intermediate Phalange Joint (PIj) by Point B, and so on. Similarly, the quaternions of the sensors' position on the back palm (S0), and each of the phalanges (S1-S3) were denoted as Q_b, Q_p Q_i and Q_d respectively (with this latter one being a virtual sensor in this work).

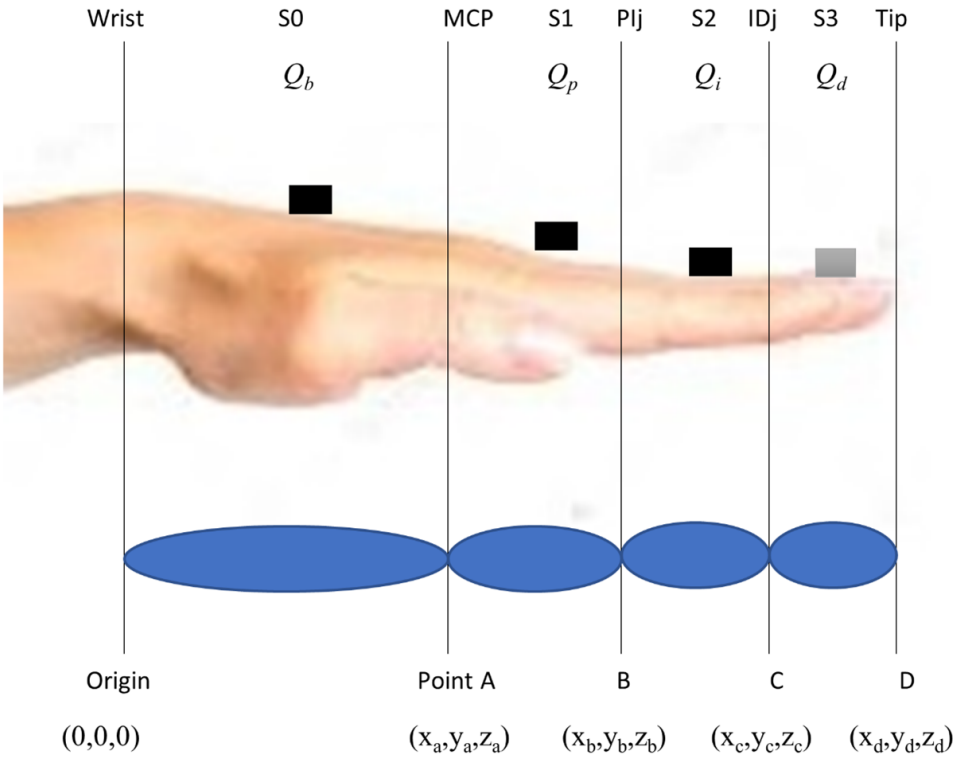


Figure 4.12: Imagination of the hand for the SMARTClap MCA purposes

Each sensor's **rotation** was described by an incoming rotation quaternion, q_{in}, which, when translated into a dual quaternion, yields Q_{in}. Each sensor's and joint's **position** was described by a dual quaternion, Q_{trans} in the form of a vector that defines its translation from the origin, in the form seen in equation (3.7) in Chapter 3. It is important to note that each joint 'after' each sensor (e.g. MCP after the S0, PI after S1 etc...) rotates with the same rotation quaternion and around the same reference joint as the sensor, albeit at a different distance from the reference joint.

Thus, the rotation dual quaternion (from equation (3.8)) of the back palm sensor (S0)

and MCP joint was given by:

$$Q_{\text{rot}} = (q_{\text{b0}}, q_{\text{b1}}, q_{\text{b2}}, q_{\text{b3}}) + \varepsilon(0,0,0,0)$$

while their translation dual quaternions(from equation (3.9)) were, respectively, calculated from:

$$Q_{\text{transSO}} = (1,0,0,0) + \varepsilon\frac{1}{2}(0, x_a, y_a, z_a)$$
$$Q_{\text{transMCP}} = (1,0,0,0) + \varepsilon(0, x_a, y_a, z_a)$$

The dual part of the translation quaternion denotes the vector that describes the position between the reference joint and the quaternion position. Hence, the MCP translation quaternion denotes the length between the origin and the MCP joint (i.e. between $[0,0,0]$ and $[x_a, y_a, z_a]$), while the translation quaternion of the sensor denotes the length between the origin and the sensors, which sits in the middle between the orign and the MCPj, i.e. between $[0,0,0]$ and $[x_a, y_a, z_a]$. This is similar for the sensors on the phalanges.

Each subsequent sensor/joint pair are taken to rotate around the previous joint. Thus, S0 and MCPj rotate around the wrist (origin), while S1 and PIj rotate around the MCP joint and so on. This means that the quaternion of the joint becomes the reference quaternion for the next sensor/joint pair. The rotation of the joint is the same as the one of the sensors, but with a different translation from the reference joint. Thus, the quaternion of S0 (Q_b) describes the rotation both of S0 and the MCP, Q_p describes rotation of S1 and PIj, and so on.

Not all joints of the hand can rotate around all three of the axes (yaw, pitch, roll) angles that are described by the quaternion. The PI and ID joints, for example, can only rotate around the y-axis of the sesnor while the MCP can rotate around the y and z axis. And even these rotations are constrained to the limits described in table 4.2 [14].

Thus, returning to Figure 4.12, the aim of the algorithm is to find the positioning and rotation of fingertip Point D shown in Figure 4.12 from the inputs of S0 – S2.

Working from the wrist (origin), first the rotation of S0 and MCP was calculated, using equations:

$$Q_{\text{rotSO}} = Q_{\text{inS0}}\, Q_{\text{transSO}}\, Q^*_{\text{inS0}}$$
$$Q_{\text{rotMCP}} = Q_{\text{inS0}}\, Q_{\text{transMCP}}\, Q^*_{\text{inS0}}$$

where Q^*_{inS0} is the quaternion conjugate of Q_{inS0}. Q_{rot} incorporates the rotation (from Q_{in}) and translation (from Q_{transSO}) of the sensor/joint pair around the wrist. Furthermore, Q_{rotMCP} becomes the quaternion of the reference joint around which S1 and the PIj rotate. Thus:

$$Q_{\text{rotS1}} = Q_{\text{inS1}}\, Q_{\text{transS1}}\, Q^*_{\text{inS1}}$$
$$Q_{\text{rotPI}} = Q_{\text{inS1}}\, Q_{\text{transPI}}\, Q^*_{\text{inS1}}$$

where $Q_{transS1}$ and $Q_{transPI}$ are the translation dual quaternions from the MCP (i.e. reference) joint and Q_{rotS1} and Q_{rotPI} describe the rotation and translation of this sensor/ joint pair around the MCP joint **only**, i.e. the LCS rotations of the named sensor and joint.

However, there is still the GHCS movement to consider. Thus, if the back palm (i.e. S0) moves, then S1 and PI joint also move accordingly (even though there is no movement of the phalange around the MCP joint), as do the S2 and ID joint etc... Incorporation of this movement into the movement of sensor/joint pair S1/PI was achieved by multiplying Q_{rot} by the conjugate of the reference dual quaternion Q_{rotMCP}, because this also incorporates in it the movement of sensor S0 and hence of the back palm. Thus, the final output positional quaternion of the S1/PI pair was calculated from:

$$Q_{\text{outS1}} = Q^*_{\text{rotMCP}}\, Q_{\text{rotS1}}$$
$$Q_{\text{outPI}} = Q^*_{\text{rotMCP}}\, Q_{\text{rotPI}}$$

with Q_{outS1} and Q_{outPI} being the positional quaternion of S1 and the PI joint. Note that the same equation is used for the S0/MCP pair, but since the reference quaternion in this case is the static wrist at (0,0,0), Q_{out} and Q_{rot} are the same.

Now Q_{outPI} becomes the reference quaternion of S2 and the ID joint, and the output position can be calculated in the same way. The values of Q_{rotS3} and Q_{rotFT} are calculated by taking the Euler angles from input quaternion Q_{inS1} , multiplying the yaw angle by 0.88 (equation (4.1)) and converting the resulting Euler angles back to Quaternion form, as described in [56].

Finally, the actual positions in 3D space of the finger part with which one holds an object was calculated by first extracting the vector element from the dual quaternion and then removing the finger thickness offset (also measured initially, with the hand measurements), from equation (3.12).

$$[x, y, z] = 2q_{\text{outtrans}} q^*_{\text{outrot}} - [x_t, y_t, z_t] \tag{4.10}$$

where q_{outtrans} and q_{outrot} are the translation and rotation quaternions that constitute Q_{out} (e.g. Q_{outS1}) respectively while x_t, y_t and z_t are the thickness of the fingertip in the directions of x , and z respectively. In our case, the thickness is only measured along the y-axis since the sensors are placed flat on the back of the fingers with respect to the finger pad. Thus, if the finger thickness is y_t, $[x_t, y_t, z_t]$ becomes $[0, y_t, 0]$. Thus, a generic step by step summary of the algorithm is as follows:

- Device switched ON and keep device stationary with palm facing downwards for 30 seconds for the quaternion readings to stabilise and for the calibration data to be taken.

  For every incoming quaternion q_{in} (one from each sensor),the algorithm will:

- create unit dual quaternion Q_{in} (equation (3.7))
- convert to Euler (equation (4.7))
- apply constraints (table 4.2)
- convert back to unit quaternion Q_{in} (by reversing equation (4.7))
- calculate Local Rotation (LR) by multiplying with its respective initial position dual quaternion $Q_{xrot} = Q_{in} Q_x Q_{in}^*$
- calculate Global Rotation by multiplying LR with previous joint rotation $Q_{xout} = Q_{xref}^* Q_{xrot}$
- extract final position from dual quaternion by $[x, y, z] = 2q_{\text{xouttrans}} q_{\text{xoutrot}}^* - [x_t, y_t, z_t]$ (from equation (4.10))

4.7 Conclusion

This chapter has described the mathematics behind the motion capture algorithm to be used in conjunction with the hardware described in Chapter 5 of this dissertation.

The mathematics is based on dual quaternions, which can represent both rotation and translation of a rigid object. Compared with ways that the motion of rigid bodies can be described in 3D space, quaternions can be less intuitive to understand when compared for example, to Euler angles, and their algebra is more complicated. However, as described previously in Chapter 3, they provide the best compromise between memory usage (less elements required than matrixes) and computational efficiency. They also do not present the problem of gimbal lock.

Whereas other studies exist that have based their tracking algorithm on (dual) quaternions and/or positional sensors ([13, 14]), this work introduced the innovation of combining dual quaternions and IMUs with inter and intra finger constraints, and estimating the position of the distal phalange (and hence fingertip position) without using a sensor for that phalange, by utilising equations found in literature [56]. This ensured that less IMUs are used in the design of the final wearable device, and hence resulting in a less bulky and costly device.

Furthermore, complex and time consuming calibration methods for the sensors were averted (calibration only takes 30 seconds before the glove is worn) by using Euler angles captured from the initial reading as reference throughout the usage.

The final output of the algorithm are Cartesian coordinates of the sensor positions and the finger tip pads (through the virtual sensors on the distal phalange) in 3D space ([x,y,z] format) which are sent to the machine hosting the game. This format was extracted from the calculated positional dual quaternions using a script in Python code, which is discussed in more detail in Chapter 6, together with the implementation and testing of the MCA.

During the course of this dissertation, a possible relationship between hand measurements was observed. This involves the measurement of the angle between the third Metacarpal Joint and the other MCPjs, of which there is no reference in literature. Thus, measurements of these were taken from a group of people, all involved in the SMARTCLAP consortium and others, in order for this relationship to be studied and hopefully published in the future.

In conclusion, this chapter gave a physiological description of the hand in section 4.1 while section 4.2 described the hand model that the MCA was based on. The latter provided the assumptions and hand movement constraints that were assumed by the algorithm. Furthermore, hand measurements were collected that would be needed to calibrate the device at start of usage, and some of this data might be published in the future.

Finally, section 4.6 described the MCA, how each IMU provides positional and orientation data in quaternion format, and how this data is manipulated using reference IMUs to capture the actual motion of the hand and provide the final positioning of each phalange and the fingertip.

Based on the MCA described in this chapter, the next chapter will describe how the hardware of the SMARTClap device was selected, designed and implemented.

Chapter 5 – The Hardware design

5.1 Introduction

Following on from Chapter 4, where the mathematical framework of the Motion Capture Algorithm (MCA) was presented, this chapter will discuss the process involved in the design and manufacture of the hardware, which the SMARTClap's user would wear while playing the serious game over AR.

For the SMARTClap project which this work was part of, a user-centred design approach was adopted, whereby the child, guardian and occupational therapist were placed at the centre of the design process. For instance, 3D printing was exploited to incorporate the child's heroes and/or favourite TV/cartoon characters on the device. For this aim, the device needed to be something that the child could see during the game-play and thus, wearable. It could not be (for example) a smart phone or VR glasses which the child wears.

Design specifications for the device arose following meetings and focus groups with other stakeholders of the projects, including Occupational Therapists, technical experts, and parents whose children were going to be the ultimate beneficiaries of the device.

The main specifications which emerged were that the device be light (since it needed to be used by young children), portable (hence battery-powered), and wireless. Additionally, considering that the usual therapy session for a child with Cerebral Palsy (CP) lasts around 45 minutes, the device needed to provide this time of uninterrupted game-play, without the need of being recharged.

This chapter will highlight the design process behind the hardware used. Section 5.2 will discuss the decision-making process of selecting the technology and electronics used for capturing the hand movements and for sending them to the game-play. Section 5.3 lists the software used for designing the selected hardware, whose components' detailed description is laid out in Section 5.4, including schematics and 3D models of the designs of the Electronics Boards (PCBs) used. Finally, the final system model and connections are shown in Section 5.5.

5.2 Motion Capture and Control Technology selection

As previously mentioned, the device to be designed needed to have sensors that capture the motion of the hand and individual fingers, plus a control unit to gather the data sent by the sensors and sending it to the processing unit.

5.2.1 Controller Selection

The control unit needed to have a small form factor, since it was going to be placed on the arms of children aged between 3 and 11. It also had to have an I^2C communications output (to connect to the sensors) while being able to output data wirelessly, via Blue-tooth or WIFI to the processor hosting the AR game. Due to its small size, it was decided to use an Arduino Nano as the controller. A comparison sheet between possible Nano contenders to be used as a controller was drawn as shown in 5.1.

Table 5.1: Arduino Nano Options

	Nano 'Classic'	**Every**	**IOT**	**33 BLE**	**33 BLE Sense**
Controller	AT-mega328P	AT-mega4809	SAMD-21G18A	nRF52840	nRF52840
Operating Voltage (V)	5	5	3.3	3.3	3.3
Flash (KB)	32	48	256	1000	1000
SRAM (KB)	2	6	32	256	25
EEPROM(b)	1000	256	Emulated	Not listed	Not listed
Clock(MHz)	16	20	48	64	64
ADC In	8	8	8	8	8
Digital I/O	22	23	23	23	23
PWM OUT	6	5	5	5	5
Size(mm)	18 x 45	18 x 45	18 x 45	18 x 45	18 x 45
Sensors	None	None	None	9-axis IMU temperature	9-axis IMU temperature pressure humidity light color gesture sound
Wireless	None	None	Wi-Fi BT4.2	BT 5.0	BT 5.0

The Arduino IOT, 33BLE and 33BLE Sense microcontrollers were the three contenders considered for the control unit, with the main difference between them being either use of WIFI or Bluetooth. Table 5.2 shows the characteristic differences between these two wireless protocols used by theses controllersDue to the Bluetooth's less power consumption (the device needs to work on batteries), the protocol was chosen for wireless data transfer. The lower speed of Bluetooth was not deemed to be an issue, since 3Mb/s is enough speed for the data to be sent.

The Arduino Nano 33 BLE (Figure 5.1) was the one that most suited the required specifications. It comes in a small form-factor of 45 x 18 mm, has dedicated pins for

Table 5.2: Specification differences between Bluetooth and Wi-Fi

	Bluetooth 5.0	Wi-Fi Direct
Peer-to-peer sharing	Yes	Yes
Speeds	1-3 Mbit/s	$>$ 54Mbits/s
Range	100m	46-100m
Energy consumption	0.01–1.0 W	2 to 20 watts
Frequency	2.4Ghz	2.4 or 5.0Ghz
Service discovery	Yes	Yes
Supported devices	Smartphones Smart TVs Laptops Smartwatches	Smartphones Smart TVs Laptops Smartwatches Desktop computers

the selected I^2C communications protocol, provides enough digital input/output pins (14) for the peripherals needed (LEDs, switches etc.) and has the possibility to be both a Bluetooth client and host (meaning that external devices can both connect to it, and it can connect to external devices to send data wirelessly). With an input voltage of 3-16V, it can be powered by small form factor batteries and at a 64 MHz clock speed provides one of the fastest CPUs for such a small microcontroller. The BLE Sense provides similar specifications, but the extra sensors it contains were out of scope for this project.

Figure 5.1: The Arduino Nano 33 BLE

5.2.2 Sensors Selection

To select the sensor technology to be used for motion capture, a breakdown of similar current systems and the technology used in them was conducted. A review of similar devices on the market and in research, including the sensors being used and links to the literature has already been provided in Section 2.4 of the Literature Review (Chapter 2). From that review, a list of motion capture sensors used and available in research was also generated Table (5.3), highlighting the pros and cons of each technology.

Table 5.3: Advantages and disadvantages of possible motion-capture sensors

Sensor Type	**Technology**	**Info**
Bending	Electro-Mechanical	Resistive: Change in resistance, of a carbon-based, electrically conductive ink as a stretched part would be proportional to bend angle. Capacitative: More reliable than resistive, and can measure direction of bend apart from the magnitude.
	Optical-fibre	Light source coupled to a sensor by optic fibre. Less light received, more bend Less cost effective and bulky since need optic fibres to run to all joints Direction of bend can be measured by pre-bending the fibre in advance of measurement
	Printed Sensors (MEMS)	Small size, low cost, flexibility. Number of printed layers on a substrate needed for sensors to work correctly. Can be used for bending/pressure/resistance.
	Conductive liquid metal	Liquid-embedded elastomer electronics (LE3) [1]. Sensors still resistive, but wearable on skin using the elastomer film around it. ethylene–propylene rubber (EER).
Tracking and Positioning	IR	Can create accurate positioning using RFID tags and receiving units. Inexpensive but require constant line of sight and can create interference since RFID operates at near- wireless spectrum.
	Camera	IR reflected from patch reflectors on tracked object Needs external camera, not ideal. Probably multiple cameras needed to sense rotation of objects.
	Electro-myography (EMG)	Physiological info of muscles while in motion. Measures electric current generated in muscles during motion. Electrodes attached to skin. Can be used to sense change in tonality of limbs. Final algorithm might be bulky and complicate.
	Triaxial Accelero-meters	Measure changes in acceleration in 3 direction Linear (acc. wo gravity) or isolated gravity acceleration. Need to remove noise by filtering, which creates delays. Double integral to get positioning creates drift.

	Gyroscope	Measures angular velocity relative to itself using Coriolis effect. Power hungry due to constant vibrations. Can be effected by other vibrations. Senses angular velocity, thus 1 integration gets angular position – creating drift.
	Magneto-meters	Can be used to correct for drift included from previous calculations (acc. and gyr.) Senses Earth's magnetic field – although this is difficult in interior environments. Susceptible to influence of anything magnetised nearby. Needs tilt compensation for precise readings. Usually used as a 'fusion sensor' with other sensors.
	Inertial Monitoring Units (IMUs)	Packaged, integrated 3-axial accelerometers, gyroscope and magnetometer with a very small footprint, readily available on the market. Been used for motion capture without using other sensors [8] (albeit with quite a complex algorithm). Algorithm still has deficiencies (e.g. large deviation in magnetic field data) but provides cheaper and less amount of different sensors. Can be improved upon.
Touch	Hall-Effect	Measure changes in Magnetic field. Need a magnetic source so can be bulky.

For simplification and space on the controller board purposes, it was decided to use only one type of sensor in the wearable device.

After consultation with the different experts involved in the SMARTCLAP consortium, covering the areas of product design, electronics design, occupational therapy and others, the 3 main sensing technologies which showed most promise were Inertial Measurement Units (IMUs), Electromyography (EMG) and a camera tracking motion sensors on the hand. As will be explained later, a decision matrix was employed to arrive at the most feasible option for our application.

IMUs are packaged devices that nowadays incorporate a 3-axis accelerometer, gyroscope, and a magnetometer, which together can be used to measure positioning and movement of the hand. They come with a small footprint and are easily available on the market.

EMG sensors provide physiological info of the muscles while in motion by measuring electrical current generated in said muscles during motion [3].

The third option uses patched reflectors placed on the hand and fingers, whose movements can be detected by IR sensors of a camera positioned externally.

The three technologies could have been used both separately or in tandem, and there were also a number of wearability options as to how to have the device worn: as a glove, as an armband or as a wristband. To decide on which was the best, a decision matrix was designed where all the options available were scored on several criteria deemed important for the device to have. Such criteria include the sensitivity to motion, modularity of the device, cost, availability, weight and others.

Each of the criteria were also given an importance weighting both from a technical point of view and from an occupational therapy point of view since these latter ones are the ones to use the device (Table 5.4).

Table 5.4: Characteristics necessary for the motion capture device, and their corresponding (importance) weight as given by the Occupational Therapists

	Characteristic	Weight
1	Sensitivity	10
2	Availability of Sensor	1
3	Sensors + Development Costs	5
4	Tracks finger movement	7
5	Tracks hand motion	7
6	Tracks arm motion	7
7	Real-time tracking	7
8	3D printability	4
9	Wearability	10
10	Modularity (personalise + adaptable to their condition)	10
11	Weight	8
12	Feedback Response	6
13	Grasping Ability	8
14	Hand preparation (calibration)	6

Finally, each technological option was marked from 1 to 10 by the 2-person technical team according to their design characteristics (with 1 being the lowest and 10 the highest importance), the mark was multiplied by the weight of the specific criteria and the option with the highest mark was chosen. As can be seen by the decision matrix in 5.5, the winning option was that of a device consisting solely of IMUs.

With the main decisions taken, the first version of the system setup looked like the one in figure 5.2. It includes IMU sensors on each phalange each housed inside a small 3D printed casing that attaches to the fingers by a ring-like structure. Each of the IMUs are connected to the control unit by wires. The control unit is powered by batteries and would provide light feedback to the user. This feedback involves a power indicator, battery charging and error status LEDs. Furthermore, it was decided to use only two sensors for each of fingers 2 to 5, one on the proximal phalange and one on the intermediate phalange, and another two on the thumb, on the Thumb Metacarpal and the proximal phalange of the thumb, similar to others in literature [14, 13]. This was to make the device cheaper and less bulky and knowing the fact that the motion of the distal phalange can be extrapolated from the movement of the other phalanges, as shown in research [55].

However, during subsequent testing of the glove with the users, later on during the project, difficulties arose in connecting the sensor of the Thumb Metacarpal correctly, as the 3D printed casing of the sensor could not be attached easily in this position.

Table 5.5: Final decision matrix table to choose the ideal sensor option for the device

Char.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
Weights	10	1	5	7	7	7	7	4	10	10	8	6	8	6	Total
IMUs only	9	10	5	9	9	5	9	9	7	7	6	7	9	7	631
IMU wrist band	4	10	3	1	5	8	6	9	9	4	7	5	8	7	447
EMG arm band	6	6	8	5	1	1	5	9	9	4	7	7	8	9	460
EMG Wrist band	4	6	9	1	1	1	4	9	9	4	8	5	8	9	406
IMU + EMG glove	8	6	5	9	8	6	7	9	7	5	6	8	9	8	599
IMU EMG wrist band	4	6	4	2	7	8	7	9	9	5	7	5	8	8	496
IMU + EMG arm band	4	6	3	7	8	8	7	9	9	5	6	8	8	8	543
Camera with motion tracking sensors	8	10	4	9	9	6	9	1	1	1	10	3	1	10	509

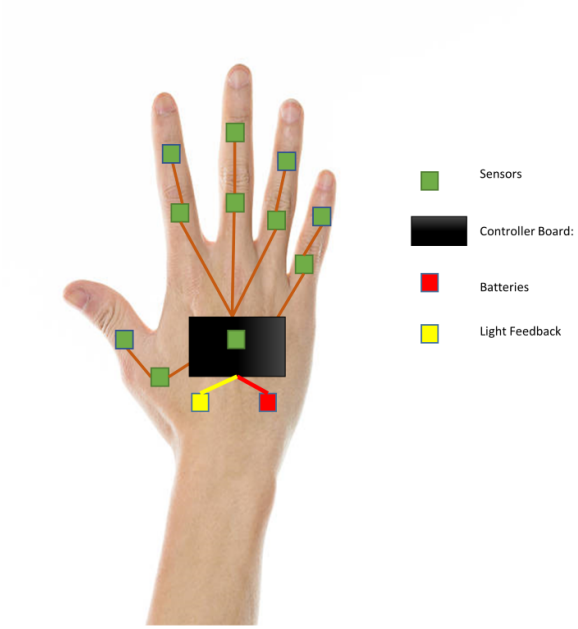


Figure 5.2: Original system schematic showing the IMU sensors, the main board, peripherals, and the interconnectivity

Thus, upon consulting with the health professionals of the SMARTClap consortium, it was decided that for the thumb, the 2 sensors are attached to the proximal and distal phalanges, as shown in figure 5.3. The MCA was not affected by this as the Thumb Metacarpal was deemed to not have much function for children with CP and thus was assumed that its motion was the same as the proximal phalange motion.

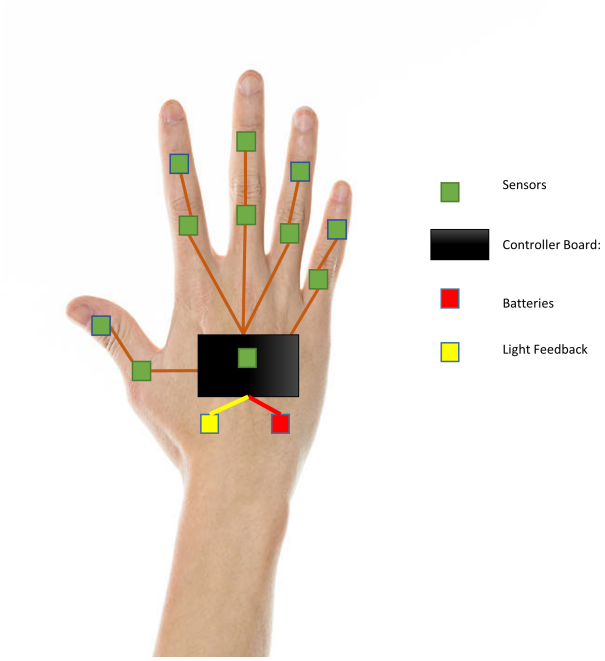


Figure 5.3: Updated schematic with Thumb sensors on proximal and distal phalanges

5.3 Hardware design

This section will describe the design of the hardware used for the SMARTClap wearable device.

The electronics boards and schematics were designed with the Eagle PCB software, using a University of Malta student license. This software integrates with Autodesk Fusion 360 (also provided through the same license) to help create CAD models of the hardware designed. This was important for this project as the 3D models of each PCB version design could be sent to the 3D printing manufacturer (Invent3D – part of the SMARTCLAP consortium) to create actual scale models of the casing housing the sensor boards in a short time, and thus help the device designer to build enclosures around these models.

As described in Section 5.2, two modules needed to be designed: the sensor module and the main control unit module. However, due to size issues, three boards were finally designed (see explanation later on).

A simple block diagram of the whole system is shown in Figure 5.4 consisting of a

control unit and a number of sensor module(s). The control unit board was divided into two sections, the sensor interface circuitry and the battery charging circuitry.

The Arduino microcontroller was powered by an unregulated battery input through the V_{in} pin, with an internal regulator adjusting the input voltage accordingly. To power the rest of the circuitry, a regulator was used to output regulated 5V and 3.3V supplies. Two 250mAh, 3.7V batteries were used for powering up. Further details are discussed later in this document in the Power Source section.

Since the Arduino only contains one I^2C communications channel, an I^2C multiplexer was used to connect the 11 IMU sensors.

The battery charging circuit consists of a charge regulator and battery switch. The latter is required to switch the two batteries between parallel and series connection when charging and operating respectively. The batteries charge through an audio jack connector, while connected to the control circuit by battery headers. When the charge cable is connected, the batteries are charging, and are disconnected from the control circuitry. Without the charging connection, the batteries power the control circuitry. Thus, for safety reasons, the system cannot be in use while charging.

Each section is discussed in detail further ahead in the document.

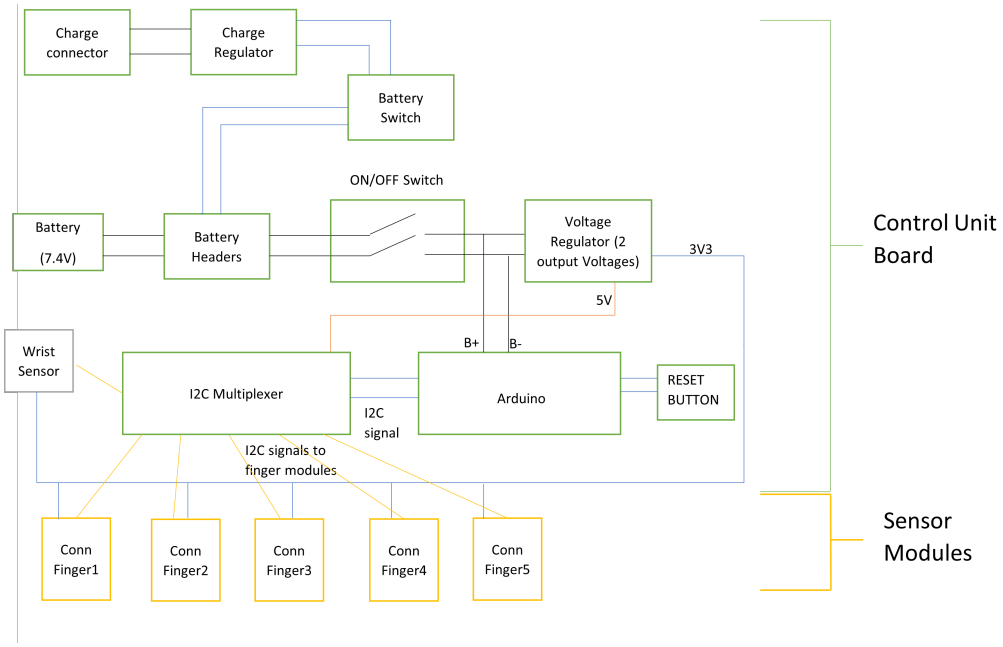


Figure 5.4: Block diagram of the hardware design, showing the components in the control unit board above, and the sensor module connections below.

5.3.1 Sensor Module Design

Eleven IMU sensor modules (based on the ICM20948 IMU) will be attached to some of the phalanges and back palm of the hand so they can track the finger and hand

movements respectively and relate this data back to the controller board via I^2C. The main design constraint of this sensor board is the size, since it needed to fit comfortably on the fingers of children aged between 3 and 11. In order to get the dimension limits of the board, the fingers of the smallest child to take part in this research were measured. Then the board dimension limits were set by the size of the distal phalange (this is the tip part of the finger, situated furthest from the palm) of the pinkie finger (thus making it the smallest area where the sensor board will be placed). The length of the phalange of the 3-year-old child was of 17mm and the approximate diameter (i.e., flat area on the top) of the phalange was 12mm. Thus, the board dimensions were set to a maximum of 17x12mm, and it was assumed that the board could easily fit on all other, (bigger) phalanges of the child, and older children as well.

The BNO055 [10] from Bosch was originally earmarked as the IMU to build the system around. The main reason for this was that it can output orientation and translation data directly in quaternions, which make algorithm and data manipulation easier to implement than with conventional Euler Angles. The definition and explanation of how quaternions work, and how they are manipulated to provide coordinates in 3D space were described in Chapter 3.

Development boards and code to connect to the Arduino for the BNO055 were also readily available online from Adafruit, so interfacing the development board and getting the first visualisation was envisaged to be straightforward.

Unfortunately, once the design was generated and the prototype was going to be ordered, a sourcing issue for the BNO055 cropped up. The device was out of stock for the near future with no date of stock availability in sight. This was mainly due to the disruption caused by COVID-19, that also led to an increase in demand for electronic equipment for people working from home, lack of raw materials to keep up with the demand and worldwide shipping disruptions [86, 87]. This was not the only time that this issue was encountered during this project and considerable time was spent in trying to source components and changing different designs according to available electronics. In this case, changing the IMU sensor module and redesigning of the board and algorithm/interface code, put the project back by around two to three months.

The replacement IMU sensor selected was InvenSense's ICM20948. It has similar characteristics as the BNO055 and the only other sensor available that could output data in quaternions, albeit not as easily as the BNO055. A similar procedure as previous was taken to interface it to the Arduino board, using development boards available on the market from Adafruit. The sensor boards for the SMARTClap device were redesigned based on these development boards, because the size of the latter (2.5 x 1.5 cm) was too large to fit on a child's finger.

The schematic and 3D model of the sensor module is shown in Figures 5.5 and 5.6 respectively. The final size is 5.5 x 10.1mm, well within the limits discussed previously.

The Bill Of Materials (BOM) for the parts can be found in Appendix 1.

The module was based on the ICM20948 and connected according to the basic application in the datasheet.

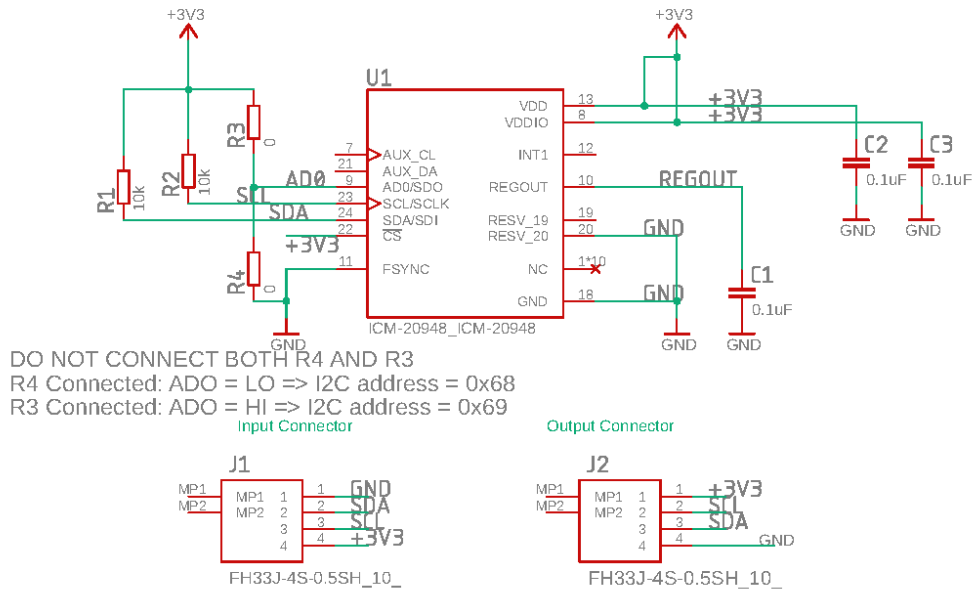


Figure 5.5: Schematic Diagram of the Sensor module based on the ICM20948

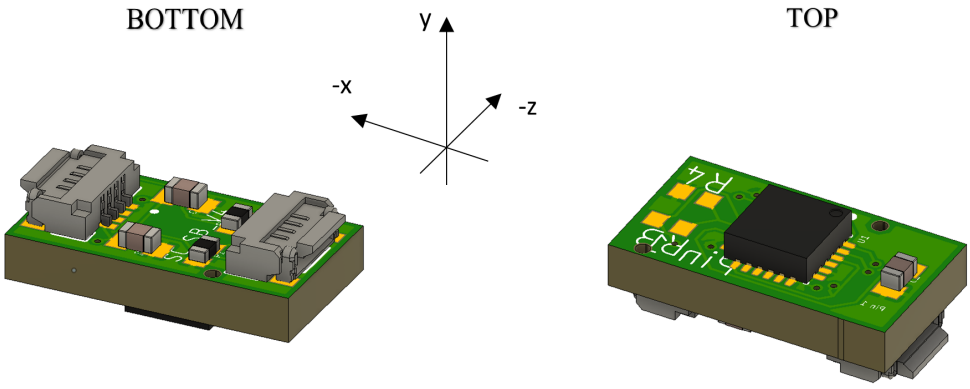


Figure 5.6: 3D model of the redesigned Sensor module with the orientation axis of the sensor

The sensor module was designed to have 2 FPP connectors from Hirose [88], one at each end, the reason being so that one board could connect to another board through these connectors by flat ribbon cables. In Figure 5.3, two sensor modules connected on each finger are shown, both of which must connect to the main control unit board.

In order to avoid having multiple cables running alongside each other on the device, the boards were designed to have the same signals passing through each of them, thus

only one cable connecting the module on the Intermediate Phalange with the one on the Proximal Phalange, and one cable connecting the module on the Proximal Phalange to the controller board. This is possible because each sensor can be programmed to have one of two I^2C identification addresses, 0x68 or 0x69, depending on whether the ADO pin (pin 9) of the ICM20948 is connected to ground (or floating) or to 3V3, respectively. In this way, the two sensors (Slaves) could be on the same I^2C connection, and the Arduino (Master) could still tell them apart (Figure 5.7). When the Arduino needs to send or receive data from the sensors, the address is first sent/received before the data, and depending on the address, the Arduino knows which sensor it is communicating with. The user can change the default I^2C address by connecting one of the 2 solder pads shown on the "top side" of Figure 5.6. Unfortunately, due to the size of the module, it was not possible to put header pins to connect this, so a solder connection or a 0 ohm resistor must be used.

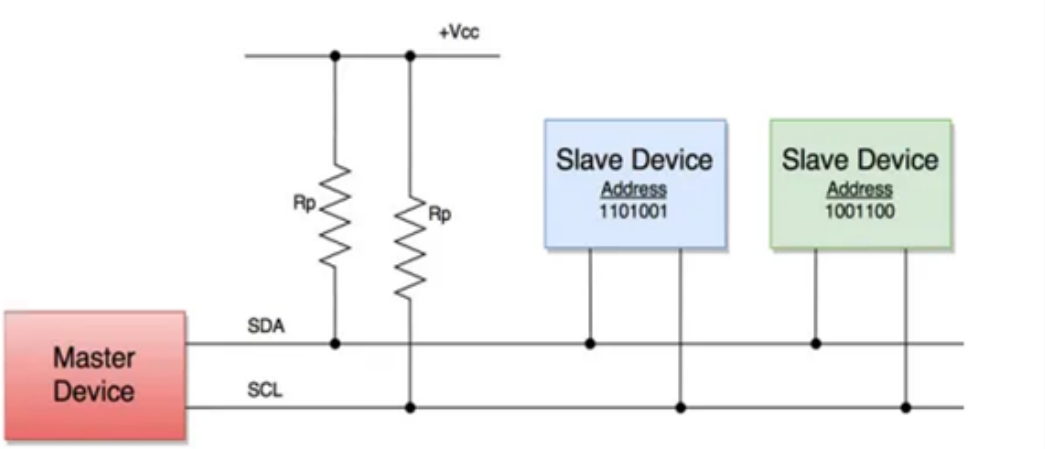


Figure 5.7: I^2C Communications protocol Schematic

The orientation of the module was dependent on the orientation of the sensor chip. The MCA depended on the orientation of the sensor in XYZ space and on the kinematic model of the hand (discussed in Chapter 4). Simply put, according to this model, when in the natural state (as in Figure 5.8), the hand is pointing in parallel to the x-axis plane. Thus, the x-axis of the sensor must point in the same direction as the fingers, as shown in Figure 5.8. To make for easier design of the sensor module (due to the connections needed to be made on a board with a diminutive size), the x-axis of the sensor was actually oriented in reverse order (i.e. parallel with the negative x-axis of the kinematic model) but this was easily manipulated in software by negating the x-axis data coming from the sensor.

Furthermore, the orientation of the y and z-axis of the sensor and the kinematic model are swapped and do not match. Similar to the x-axis issue, this was resolved in software by remapping the incoming data from the sensor accordingly.

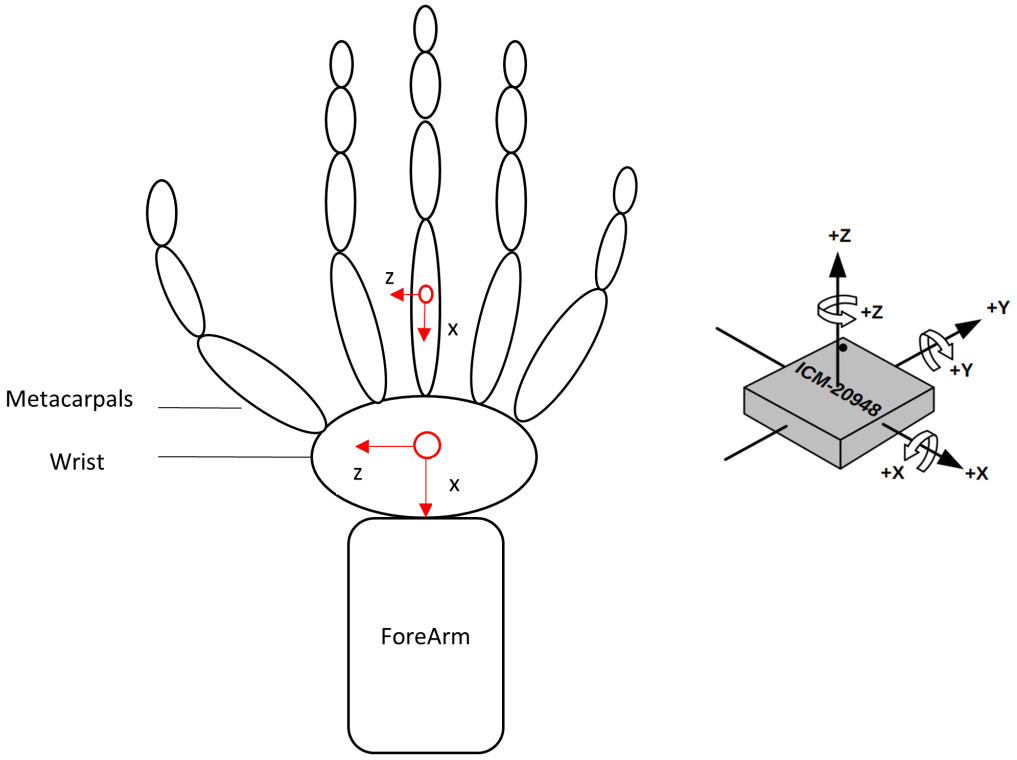


Figure 5.8: I^{2}C Communications protocol Schematic

5.3.2 Control Unit Board

The control unit board, based on the Arduino Nano 33 BLE, is the main board which gathered the data from the sensors and transmits them via Bluetooth to the main processing unit. Its block diagram can be seen in Figure 5.4. The main components and functions are the Sensor Control, the Power source, the charging circuit, the voltage regulator, and the peripherals.

Sensor Control

The main board's aim was to to control and receive data from eleven sensors, ten on the fingers, and one on the back of the palm. The communication was done through I^{2}C. However, due to only one I^{2}C port on the Arduino, and the maximum possible number of addresses of each sensor limited to two (see section 5.3.1), an I^{2}C multiplexor was used. The TCA9548A [89], from TI, could be controlled by one I^{2}C bus from the Arduino, while connecting up and switching between 8 different Slave devices, all with the same I^{2}C address (figure 5.9).

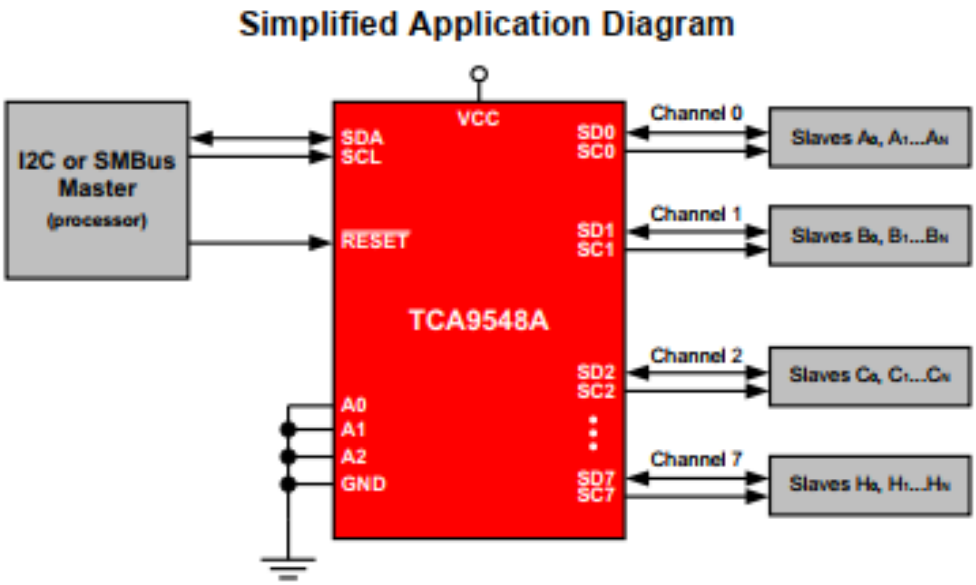


Figure 5.9: I^2C multiplexer application schematic

Since a maximum of two sensor modules can be connected on each channel, then the 8-channel multiplexer could have up to 16 sensors connected. This was more than enough for the eleven sensors used in this work.

Back Palm Sensor

The is the sensor attached to the back palm and upon which movement the whole hand moves. It has the same connections as the sensor modules, but since it was designed to be positioned in the same place where the connector board is, this was incorporated within the main board. It is connected on Channel 1 of the multiplexer, with address 0x68.

Power Source

Due to the wireless nature of the device, the power had to come from batteries. These needed to be lightweight and with a small form factor so that the final device weight is not bulky and heavy for the children using it. The batteries selected were rechargeable Lithium Ion Polymer (LiPo), which have a higher specific energy and lower weight then other polymer batteries (Figure 5.10). The nominal voltage for one battery pack is 3.7V.

Figure 5.10: Lithium Ion Polymer Battery Pack

Since 3.7V was not enough to power the Arduino and the Multiplexer (both need at least 5V), two battery packs in series were used for a total voltage of 7.3V. To calculate the current draw needed by the battery, an ammeter was connected in series with the Arduino, multiplexer and 2 sensor modules, from which an estimate of the total current for the system was extrapolated:

- Arduino alone: $\sim$15.8 mA
- Arduino + 1 Sensor: $\sim$18.5mA
- Arduino + 2 sensors: $\sim$ 22mA
- $\Rightarrow$ 1 sensor (dev board) draws between 3 and 5mA $\Rightarrow \sim$5mA
- Arduino + 11 sensors $\Rightarrow$ 71mA (push to 100mA, including all extra components on the main board)
- Therefore, for 1hr operation, the battery needed to have a capacity of at least 100mAh

Thus, two 250 mAh batteries in series were deemed enough to provide the necessary voltage and current to power the whole system for several hours. The batteries were designed to be physically placed underneath the main control unit board in the enclosure and connected to it through 2 JST pin headers [90].

Charging Circuit

The battery charging circuit is based on Microchip's MCP73833 battery management controller [91], as shown in 5.11.

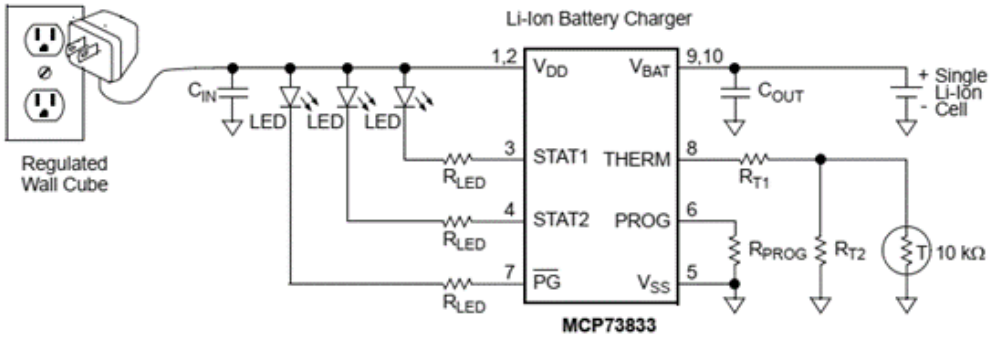


Figure 5.11: Schematic of charging circuit, from datasheet of [91]

Charging a LiPo battery is performed in three stages: first a preconditioning charge, then a constant-current fast charge and finally a constant-voltage trickle charge to keep the battery topped-up. The rate at which the constant-current fast charge operates is set by the resistor R_{Prog}, whose value is defined as:

$$R_{\text{Prog}}(k\Omega) = \frac{1000V}{I_{\text{charge}}(mA)}$$

I_{charge} is the constant fast charge current. Thus, for example, a 1kΩ resistor will set the charge current to 1000mA, a 5Ω resistor will set the charge at 200mA etc. Conversely, for example, if a 1000mAh battery pack needs to charge for 1 hour, I_{charge} will be 1A and will need an R_{Prog} value of 1kΩ .

The circuit also monitors the battery temperature by measuring the voltage across the 10kΩ negative-temperature coefficient (NTC) thermistor on pin 8. If the voltage goes outside the range of specific factory set defaults (and hence, the temperature of the batteries), the charge cycle is suspended until the reading is back within limit. LEDs (LED_{stat1}, LED_{stat2}, LED_{pg} – blue, green, and red respectively) give an indication of the status of the charge cycle according to Table 5.6.

Table 5.6: Description of Status LEDs. 0-OFF, 1-ON

	LEDstat1	**LEDstat2**	**LEDpg**
OFF	0	0	0
Standby	0	0	1
Charging	1	0	1
End of charge	0	1	1
Temperature Fault	0	0	1
Timer Fault	0	0	1
System Test Mode	1	1	1

The circuit in Figure 5.11 is designed to charge one, 3.7- 4.2V LiPo battery pack. However, the device needs 2 battery packs in series to run (see previous section) on 7.4V. Charging battery cells in series can cause an issue because they can become

unbalanced, with one battery being charged more than the other. This can cause overheating, damage, and failure to one or more of the cells which are working harder than the others. To make sure that the batteries are always balanced while charging, it is better to charge them in parallel. The solution was to have the batteries in parallel while charging, and in series while in use.

Thus, originally, a 3-channel electronic switch, the DG4053EEN from Vishay (Figure 5.12), was chosen to switch the battery connections accordingly. The functionality of the switch was that when there is no voltage on the input (i.e. charging cable was not connected), the batteries are in series and connected to the main circuit (i.e. Arduino). However if the cable is connected and a voltage is detected on the input, the internal switches would put the batteries in parallel and at the same time disconnect them from main circuit.

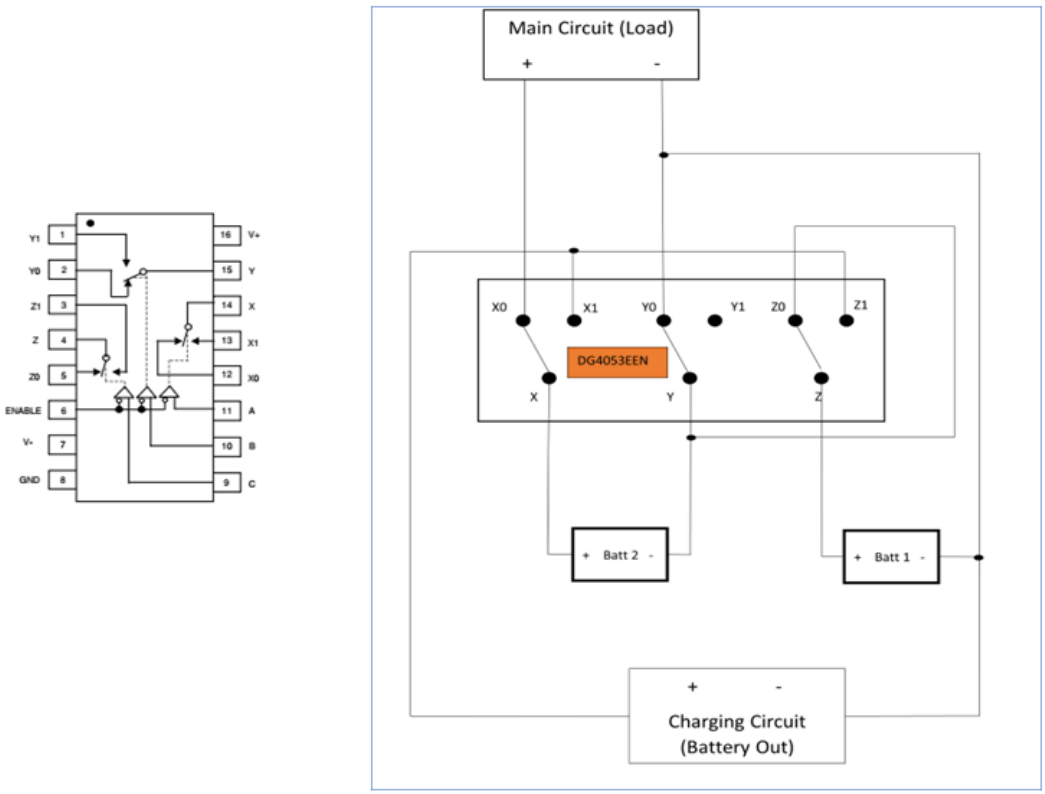


Figure 5.12: Schematic and block diagram of DG4053EEN switch

The functionality and size of the circuit was ideal for this purpose, however, once a prototype was built, issues arose with the voltage output to the main board due to a high 'ON Resistance'between internal contacts of the switch. This caused a large voltage drop across the said contacts and decreased the usable output voltage in a way that made it unusable.

The solution was found by using mechanical switching through an audio jack switch. Audio jack switches are an extension of normal audio jacks (figure 5.13a), where, apart

from having the main connection which transmits the audio signal, they also include a type of switch configuration which can be isolated from the audio signal and enabled to control other circuits (Figure 5.13b))

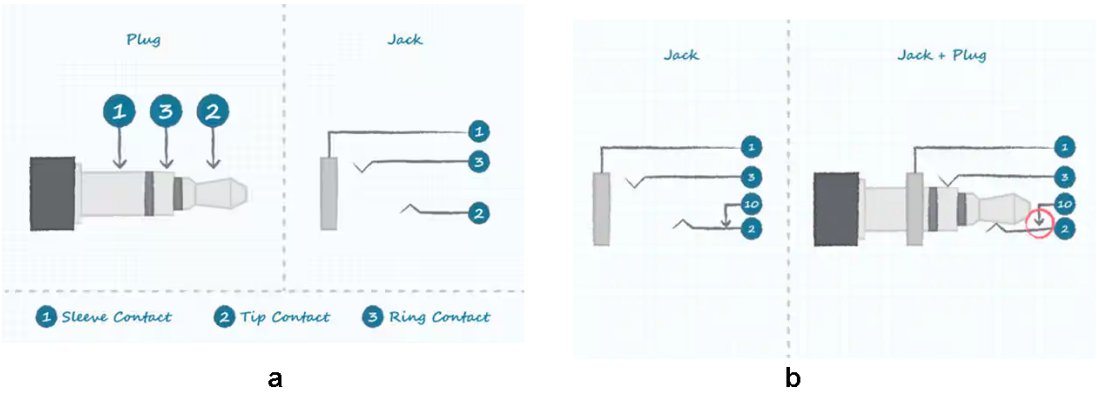


Figure 5.13: Audio jack switch configurations. Source: [92]

The switch chosen was the SJ3589ANG, from CUI devices (Figure 5.14)). It has a DPDT (Double Pole Double Throw) configuration, which can have two separate inputs and two separate outputs (meaning it can control 2 output circuits).

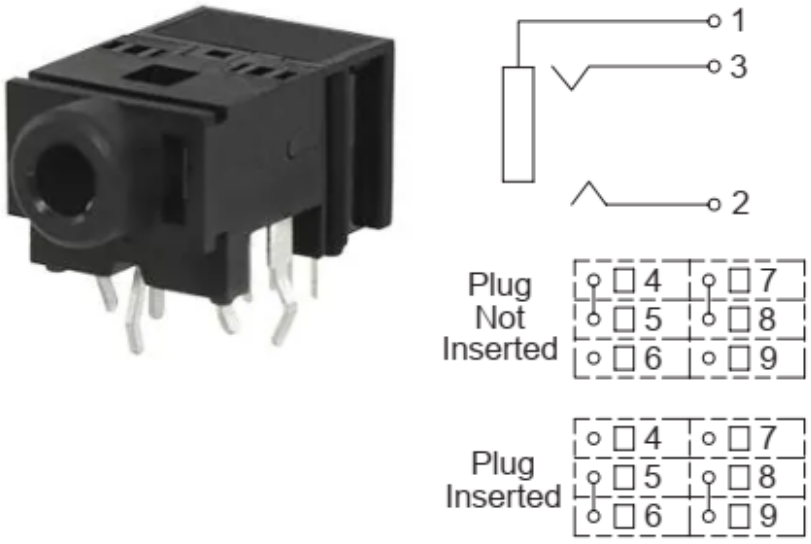


Figure 5.14: The SJ3589ANG (Left the component. Right On top, the audio (or in our case, voltage) input, and below, a separate circuit with 2 possible outputs

The connection circuit of the SJ3589ANG is shown in Figure 5.15. Inputs 2 and 3 are a separate circuit which provide the 5V input from any external source.

When the audio jack is not connected, contacts 7/8 and 4/5 are connected. This will put the 2 batteries in series, which in turn will provide the 7.3V output to the load, and switch on the Main Board circuitry. No voltage is on pins 2 and 3.

When the audio jack is inserted, contacts 5/6 and 8/9 are shorted, which put the batteries in a parallel configuration, charged by the 5V input to from contacts 2 and 3. The

voltage across the load is now 3.7V, which is not enough to power the main circuitry.

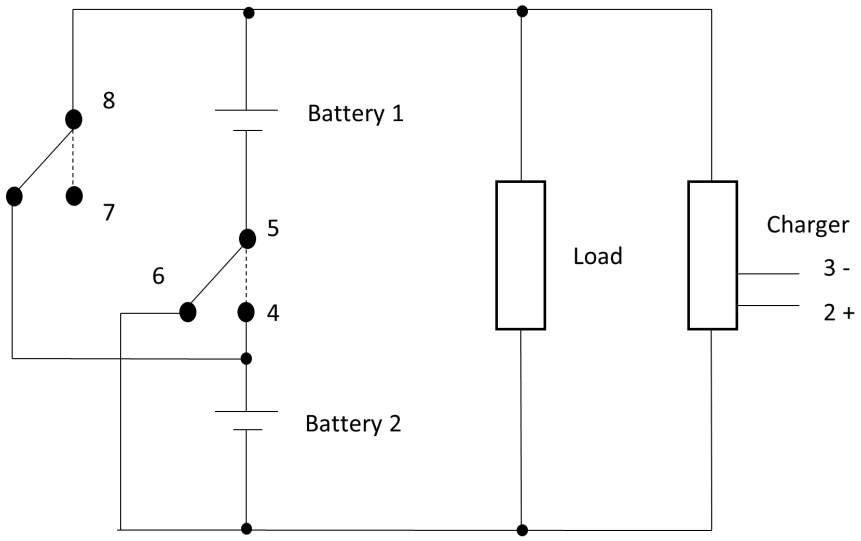


Figure 5.15: Connection circuit of the SJ3589ANG during charging

Voltage Regulator

The Arduino can operate from the unregulated voltage of the batteries due to an internal Voltage Regulator. However, the rest of the circuitry needs a regulated supply to avoid voltage fluctuations from the batteries. The sensor modules need a 3.3V supply to operate, while the I^2C Multiplexor needs a 5V input. Thus, a Voltage Regulator with 2 different outputs was chosen. The regulator originally chosen was Microchip's MIC5211. However, using its dual output option did not give enough current output (50mA) to power the sensors ($\sim$70mA). The choice then went onto the TI's TLV755 , which however turned out to be out of stock for the near future. Finally, TI's TPS51103 [93] was selected. Although this is a bit over-specified due to an extra function to output a clock to another chip once a certain voltage is reached (an option which was not necessary for this project), the output current and voltage is 100mA on the 2 pins, which is more than enough for what was needed.

Peripherals

An small-factor RGB LED was included as a status LED. This lights accordingly if one of the sensors on a particular finger was not connected correctly or has developed a fault. The lighting status is shown in Table 5.7.

Table 5.7: Error Status LEDs. 1:ON 0:OFF

BLUE	RED	GREEN	Error
0	0	0	No Error
0	0	1	Error on Finger 1
0	1	0	Error on Finger 2
0	1	1	Error on Finger 3
1	0	0	Error on Finger 4
1	0	1	Error on Finger 5
1	1	0	Error in BackPalm Sensor

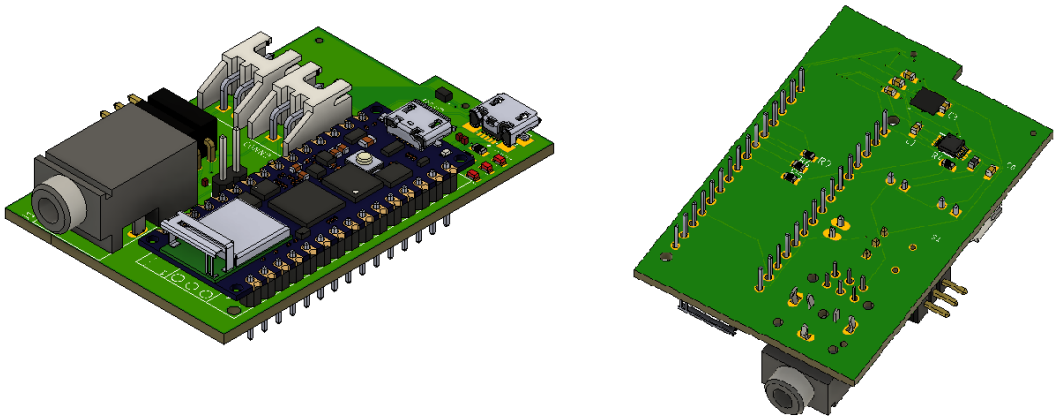
An ON/OFF switch was also included in the design, while for debugging purposes, two jumpers were included as well in order to bypass the charging circuit and connect the batteries directly to the main circuit if needed.

When the board design was finished, a model enclosure according to the size of the board (61mm x 45mm) was built and was tested on some of the candidates who were going to use the device, i.e., children between 3 and 11 years of age (Figure 5.16). Seeing the movement of the board on the hand, it was noted that the board was too big, and when the hand was twisted downwards at the wrist, the board (and hence the sensor for the back palm) was not in constant contact with the hand, thus for sure giving a wrong orientation reading of that particular sensor, and consequently of all the rest of the sensors, which use the palm sensor as the origin of their coordinate systems. Thus, it was decided to divide the control unit board in two. The main board, consisting of the Arduino, the charging circuit, batteries, and the voltage regulator would move up to the forearm of the child where it is easier to strap, while a new, smaller 'Connector Board', consisting of the I^2C Multiplexer, the IMU sensor circuit and six connectors (one for each finger and one connection to the Main board) would be placed on the back palm. This is described in the next section.

Figure 5.16: Testing the model of the main controller board. The board was too large so when the user made a fist or moved the wrist downwards, the board did not maintain constant contact with the back palm, meaning the motion sensor on it would not have given accurate readings

The final circuit diagram and model of the main Controller Board control unit board that goes on the forearm can be seen in Figures 5.17 and 5.18 respectively, with the final dimensions of the board being 42x50mm. This should fit well with all the children participating in the trial, whose average hand measurements (width of forearm) are of 41.3mm – 50mm at the wrist and 47.7mm – 72.3mm at the widest part of the forearm. The BOM can be found in Appendix 2.

Figure 5.17: Main Control Unit Board Schematic Diagram



Top **Bottom**

Figure 5.18: 3D Model of Main Control Unit board that is worn on the forearm

5.4 Connector Board

The aim of this board is to hold the sensor on the back palm, which sensor will be used as the origin coordinate for the hand model and all other sensors and to act as a connector board between the main control unit board and the sensor modules on the finger.

Thus, apart from the sensor module and the I^2C multiplexer, it contains six connectors. The input connector is a USB-B connector and provides the supply voltage to the sensors (3.3V) and the multiplexer (5V) as well as the I^2C bus to the sensor modules.

The five output connectors to the 5 fingers are JST connectors using ribbon cable, same as used and described in the Sensor Module section (Section 4.1).

A schematic diagram and 3D model of the Connector board are shown in Figures 5.19 and 5.20 respectively, with the board having dimensions of 20 x 24mm. Its BOM is tabulated in Appendix 3.

5.4.1 Hardware Connections

The sensors were connected with each other and to the Connector board by the use of 4-way flat ribbon cables [21] (figure 5.21). The 4 signals to be passed from the connector board to the sensors are the Power (3.3V), Ground, and the two I^2C signals, SDA and SDC. Although a bit intricate to connect the cable to the connectors, it allowed more freedom and space between the hands as the cable could be inserted directly inside the

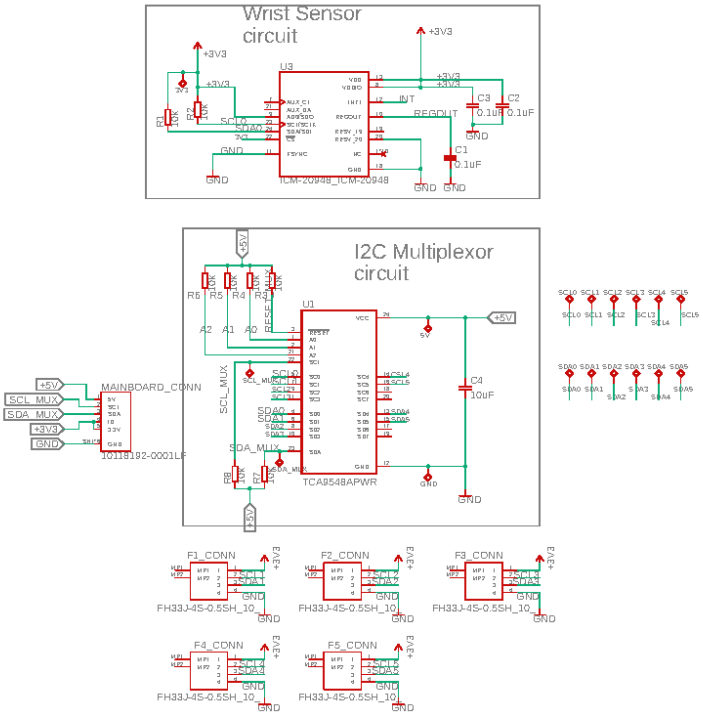


Figure 5.19: Connector Board Schematic Diagram

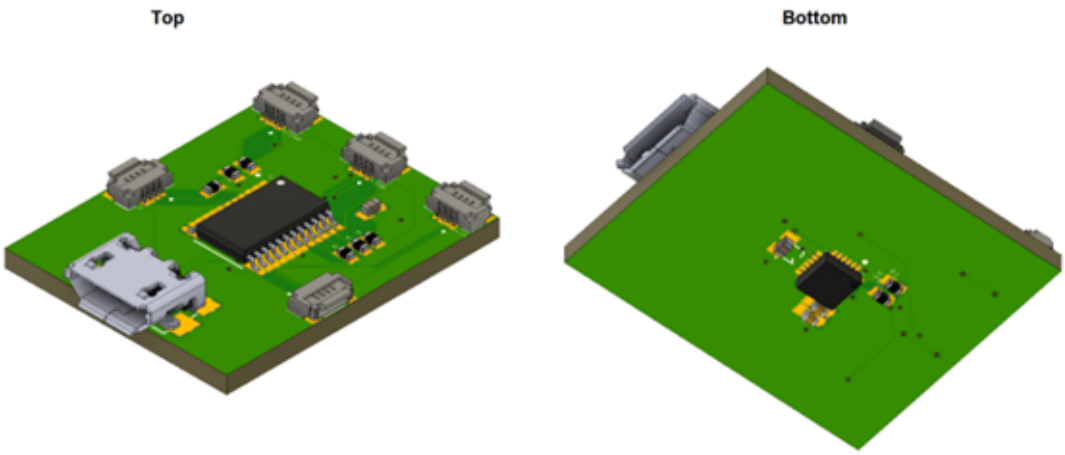


Figure 5.20: 3D Model of Connector Board that is worn on the back palm

enclosure and did not have to connect to a bulky socket extruding from the enclosure or connector (like a USB-B connector, for example).

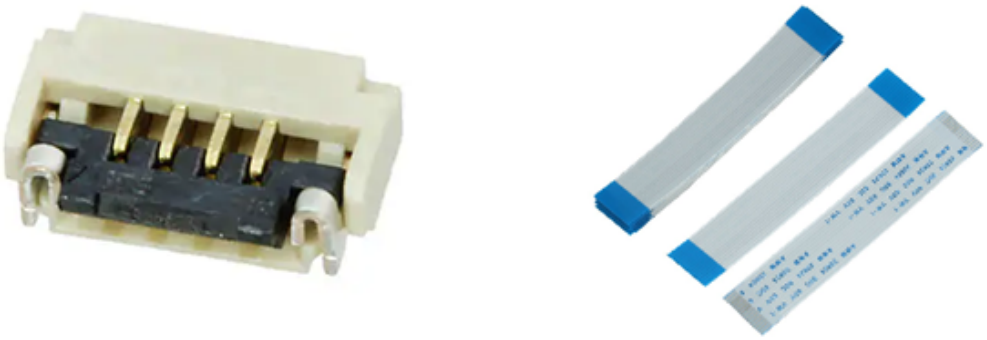


Figure 5.21: Connectors (left) and Ribbon cables (right) used to connct the connector board and the sensor boards

The final schematic of the hardware is shown in Figure 5.22 while the wearable device can be seen during testing in Figure 5.23

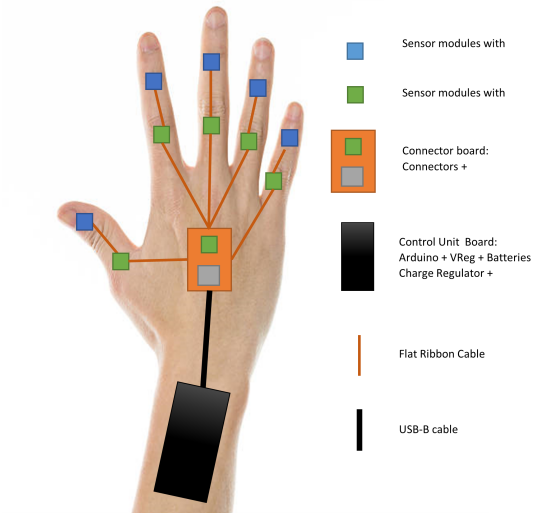


Figure 5.22: Final block diagram of finished product

Figure 5.23: The SMARTClap wearable device during testing

5.5 Conclusion

This chapter has outlined the hardware design of the SMARTCLAP wearable device, intended to capture the hand and finger movements of children suffering from CP.

Section 5.2 described how the technology was selected for every piece of the device, including the IMU sensors on each finger and the controller board to capture the motion data and send it to the main processing unit. The user needs to be able to make use of the device for at least 45 mins without interruption, and hence the capacity of the batteries was decided accordingly.

A detailed technical description of the electronic components used to bring the system together has also been laid out in Section 5.3.

There is ample space for improvement of the device. A lot of unused peripherals present on the Arduino board (e.g. an in-built IMU, UART and SPI ports, a USB connector et) were not used for this project. In order to decrease the footprint of the mainboard, the useful components of the board (e.g., the microcontroller, the Bluetooth adaptor etc.) can be designed directly on the controller board. However, this was deemed too time-consuming for the prototype's development process.

This chapter follows on and is based on the outcome of Chapter 4, where the mathematical framework of the Motion Capture Algorithm (MCA) was defined. In the next chapter, the MCA software and hardware are brought together to code and implement the device.

Chapter 6 – Coding and implementing the MCA

6.1 Introduction

Following on from Chapter 4, where the mathematical framework underlying the design of the MCA was presented, and Chapter 5, where the hardware design of the motion-capture wearable device was discussed, this chapter will focus on the coding, implementation, and testing of the MCA. The software was designed to program the hardware presented in Chapter 5 and consequently implement the algorithm described in Chapter 4.

The work of this chapter is, hence, divided in two parts:

- Programming the Arduino to capture the (quaternion) data from the Inertial Measurement Units (IMU) i.e., the sensor boards that are capturing the motion of the fingers on the hand on which the device is worn and sending it over via Bluetooth Low Energy (BLE) Communication to the game machine where the MCA is programmed.
- Implementing the MCA which creates a kinematic hand model, captures the quaternion data communicated from the Arduino, and merges the incoming data to allow for the simulation of the hand movements on the game screen. The raw data, outputted by the specific IMUs, does not describe the hand to be shown in Augmented Reality. The aim of the MCA is to fuse the IMUs' data to describe their position in relation to each other on the screen. For instance, indicating the position of the thumb on screen in relation to the wrist.

Once the algorithm was coded, it was tested for repeatability and consistency by comparing the output of the sensors to a VICON system, which is an industry-standard camera-based motion tracking system utilizing eight InfraRed cameras around a subject fitted with reflectors (refer to section 7.3.1 for more details). A special test rig is available at the University of Malta's Biomedical Engineering Laboratory specifically for this. Based upon this introduction, the rest of this chapter is structured as follows. Section 6.2 gives a brief description of the software used to program the Arduino to collect data from the sensors and send it to the game machine. Section 6.3 describes the software used to code the MCA. Finally, Section 6.4 describes the testing carried out to validate the algorithm, and the respective results.

6.2 Programming the Arduino

Using the Arduino IDE software, the Arduino 33 Nano BLE was programmed to collect the rotation and positioning data of the 11 sensors. The Arduino software is usually divided in two main functions: Setup and Loop. In the former, the one-time setup and initializations of the device (Arduino) peripherals are performed whereas in the latter, the continuous gathering of the data is coded. The Arduino library used to program and gather data from the ICM20948, was provided by Sparkfun Electronics [94]. In it all the classes used in the code are defined.

6.2.1 Setup

As previously mentioned, the setup function performs the initializations of the peripherals used by the Arduino i.e. I^2C, Serial, and Bluetooth Low Energy (BLE). Moreover, the initialization of the sensors, their DMP (Digital Motion Processor), and the I^2C multiplexor was included here.

Serial and I^2C Initialization

The serial port was initialized first. It was not used to send and receive positioning information from the device but only to output debugging information. The device (Arduino) sends and receives this data wirelessly through BLE. Next, the I^2C peripheral was initialized to run at 400kHz. The I^2C bus connects the Arduino to the I^2C Multiplexor (MUX) chip on address 0x77 (the TCA9548 can be addressed using one of 8 different addresses, 0x70 to 0x77 depending on the resistor configuration. See Section 5.3.2 for more details). Consequently, each channel of the MUX was selected sequentially to initialise the sensors on each finger (or back-palm). Thus, with the back palm (channel 0) and five fingers (channels 1-5), 6 channels of the 8 available on the MUX are used. Before initialization, the code checks how many sensors are connected to the device by scanning channels 0 to 5 of the MUX for any devices connected to addresses 0x68 and 0x69 of its I^2C bus, the addresses of the sensors connected on the proximal and intermediate phalanges respectively. The flowchart in Figure 6.1 shows an extract of the flow:

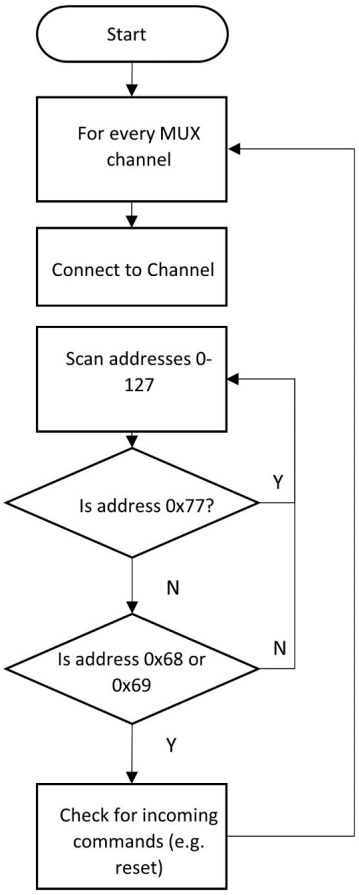


Figure 6.1: Check for available sensors

Sensor Initialization

After this step, sensors are initialized by connecting them to each channel of the MUX and sending the *begin ()* command, which polls one of the two I^2C sensor addresses (0x68 or 0x69) on each finger (or 1 address for the back palm) and retrieves the data from each one as shown in Figure 6.2.

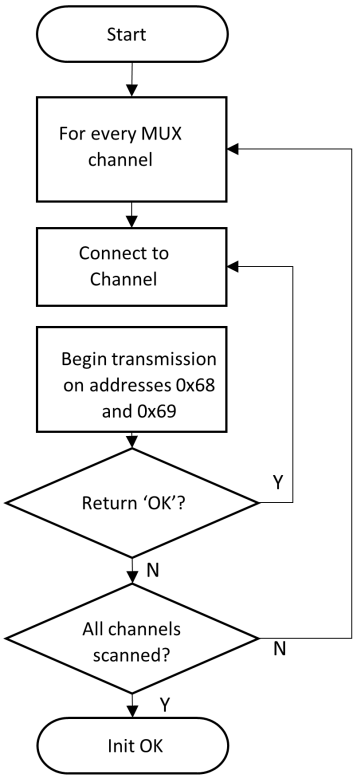


Figure 6.2: Sensor initialisation

Once all the sensors are initialized, the procedure is repeated, this time to initialize the Digital Motion Processor (DMP) of each sensor. From the sensor's (ICM20948) datasheet it is stated that the DMP [11]

"Offloads computation of motion processing algorithms from the host processor. The DMP can be used to minimize power, simplify timing, simplify the software architecture, and save valuable MIPS on the host processor for use in applications."

In most research publications using IMU sensors, orientation data is extracted by fusing raw accelerometer, gyroscope and magnetometer into quaternion data using different kind of algorithms like Kalman Filtering [95] , complimentary filters [96] and extended variants of it [59], Inertial integration [97] and other methods. Most of these filters are used in order to reduce the effect of magnetic fields in indoor surroundings on these sensors. The DMP has its own internal fusion algorithm and it can output direct quaternions in a number of methods. Catarino [59] used the DMP output of a precursor of the ICM20948 (MPU-9250) as a comparison for his algorithm, and the output was very similar. Thus the sensor's internal DMP can be used to simplify the algorithm used for orientation in our device. This helped to save time and computational processing power by transferring these calculations onto the sensor's processor rather than executing them on the Arduino.

The procedure of initializing the DMP was provided by SparkFun's Arduino library

for the ICM20948 [94]. This provided several options for the DMP to output. The one used in this project was the 'Game Rotation Vector ', which outputs quaternions using 6 degrees of freedom, meaning it uses only the Accelerometer and the Gyroscope (and not the magnetometer) to calculate the quaternion data from the raw values of the former. In doing so, the quaternion data received is relative and not absolute. This means that, if the magnetometer data is included, the sensor would always use the magnetic north as an initial position, whereas without it, the initial position of the sensor can be anywhere the sensors are pointing at the time of initialization.

However, upon initialization, the sensors' output quaternion data is starting from [1,0,0,0], meaning that there is no rotation. The sensors then need time to stabilize to their orientation. This can be seen in Figure 6.3, which shows the raw quaternion data from the sensors on the back palm and the proximal phalanges of the fingers, just after initialization [1]. It can be seen that it takes some time for the values to stabilize. Note especially the value q2 of the proximal phalange of finger 1 (PP1), which stabilizes at a higher value than the others. This is because q2 represents the y-axis orientation of the sensor, which is attached to the side of the thumb (F1), rotated around its y-axes and not flat like the others (e.g. back palm).

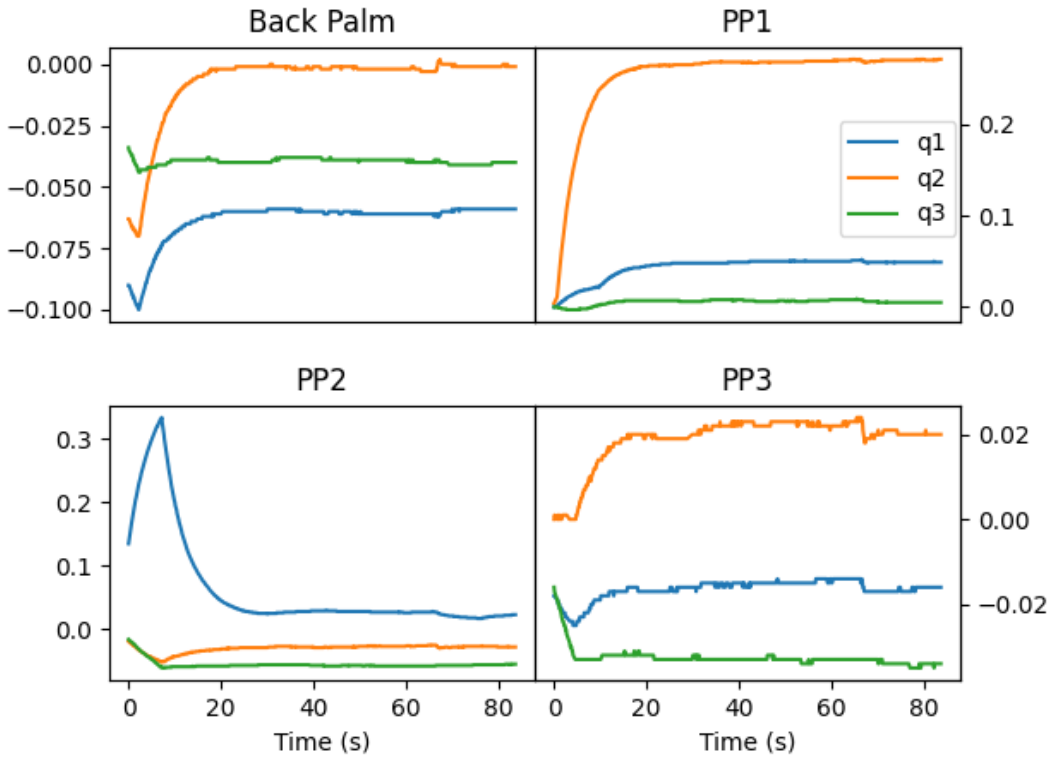


Figure 6.3: Stabilization time for the back palm sensors and the proximal phalanges (PP) of the three fingers. The y-axis represent the normalised values of the quaternions coming out from the sensors, ranging between -1 and 1

[1]Note that only q1, q2 and q3 are shown, since q0 is calculated from the other three, using equation (6.2).

Hence, after the initialization is done, there needs to be a waiting period to allow the sensors to stabilize. By experimentation, this time for the SmartClap device was set to 25 seconds [2]. During the stabilization period, the Arduino LED flashes continuously until, when done, it turns OFF.

In future improvements, this data can also be used to automatically detect on which hand the device is being worn.by detecting from which channel of the MUX chip the ”abnormal” stabilized value of q2 (abnormal in the sense that the thumb's q2 value stabilizes higher than in the other four fingers) is coming. If the abnormal value is on channel 1, than the device is being worn on the right hand, if it comes from channel 5, than the device is being worn on the left hand. Another option, which is only practical if not all five fingers are being used, is to monitor channels 1 and 5 of the MUX and detecting the hand on which the device is being worn by the lack of sensor data coming from one of them.

BLE Initialization

As the name suggests, Bluetooth Low Energy was designed to be used in applications where there is no need for large bandwidths and data transfers, like Smart home appliances and wearable devices. Thus, power consumption is extremely low, allowing for smaller battery capacity, size, and longer lifetime. Before the stabilization period of the sensor, the BLE adapter is initialized. This includes setting the service and the device name, adding the relevant service characteristics to be broadcast, and starting advertising (figure 6.4). Two read characteristics were advertised, the ‘quaternion data ’, which changes with every pass of the loop (see next section), the ‘number of sensors available ’, which is constant, while a write characteristic,‘reset ’is advertised so that the Arduino can be able to be reset over BLE (figure 6.5).

[2]after initialisation, before the start of the game, the user has to keep the hand stable for another 5 seconds for the calibration data to be taken.

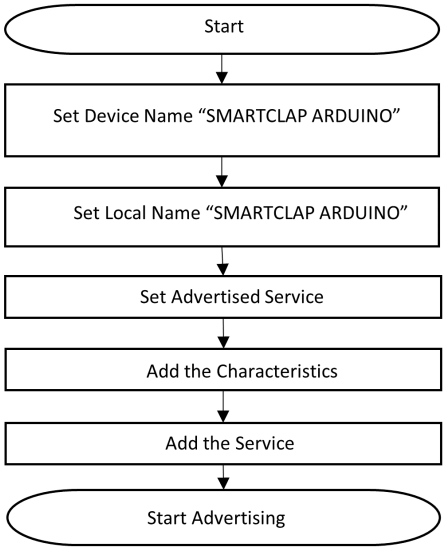


Figure 6.4: BLE setup

Figure 6.5: BLE characteristics coded (left) and advertised on a smartphone BLE application (right)

Thus, the whole setup function follows the steps depicted in the flowchart in Figure 6.6

6.2.2 Loop

The loop section of the Arduino code continuously gathers and processes the data from the sensors and sends it over Bluetooth BLE at a maximum frequency of 50Hz. The loop flowchart is shown in Figure 6.7

Retrieve Quaternions

As with the initialization process, for every IMU (ICM20948) pair of sensors on each finger (or single in the case of the back palm), the relevant MUX channel was selected.

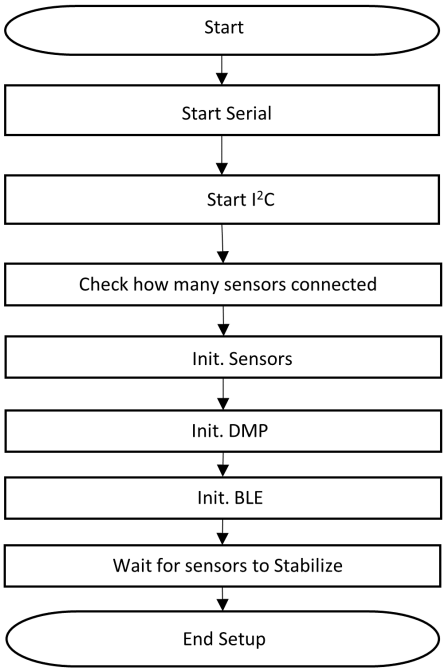


Figure 6.6: BLE characteristics coded (left) and advertised on a smartphone BLE application (right)

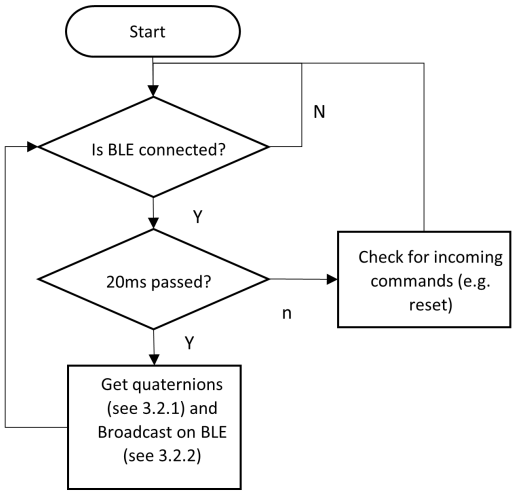


Figure 6.7: Arduino Loop flow

After checking that the data received from the sensor is in the correct format, the quaternion values of q_1, q_2 and q_3 are retrieved directly from the chip, from which the q_0 value is computed. In quaternion terms, a rotation can only be calculated correctly if the quaternion is a 'Unit Quaternion". A unit quaternion is one where:

$$\sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1 \tag{6.1}$$

Hence, after q_1, q_2 and q_3 are retrieved, q_0 is calculated by reordering Equation 6.1 to make this latter element as the subject of the formula.

$$q_0 = \sqrt{1 - (q_1^2 + q_2^2 + q_3^2)} \tag{6.2}$$

Before the quaternions are sent over BLE, a few checks are done to make sure that no corrupt values are sent. The three checks done arose from the output noticed from the sensors during experimental trials:

- Check that q_0 is a number (i.e. not NaN) – being the output of a square root of squares, if q_1, q_2 or q_3 have erroneous values such that the sum of their squares is higher than 1, then the value of q_0 will be a 'NaN '(Not a Number).
- Check the values received are between -1 and 1 because quaternion values are only valid if they are between -1 and 1. However sometimes, corrupted values are output from the sensor which need removing.
- Check that no value is exactly 0.000 – during trials, it was noted that sometimes, an erroneous value of exactly '0' was obtained. This is an error from the sensor.

If one of these checks fails, the data is discarded and the next set of quaternions is retrieved.

Once all the data from all the sensors is gathered, it is sent over BLE using the algorithm described in the next section. Figure 6.8 shows the flow of the data acquisition algorithm.

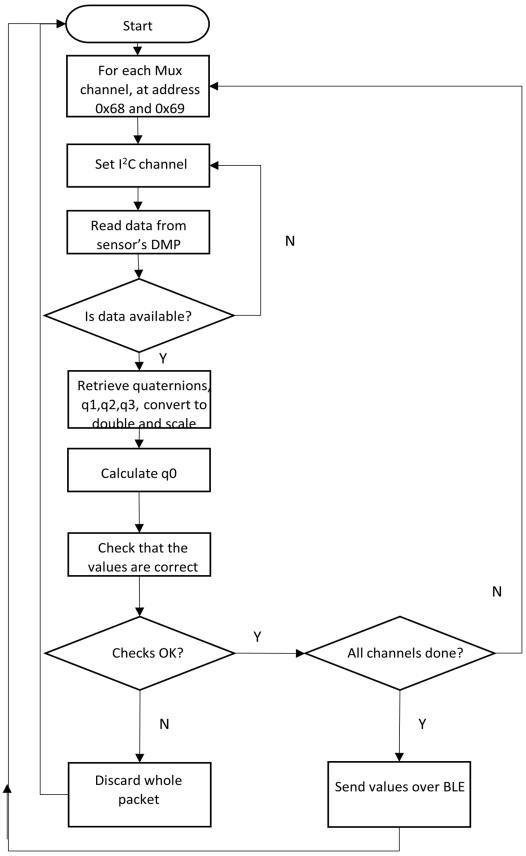


Figure 6.8: Data acquisition flow

Send over BLE

Data was sent over BLE by packaging all the values of all sensors in a char array and broadcasting as one BLE characteristic for every acquisition. The format of the value to be sent was limited to three decimal places and a space for the possible negative sign, "-0.xxx" (quaternion values always fall between -1 and 1), with a maximum of 6 characters (bytes) per value.

Since there are 44 values to be sent (4 quaternion elements/sensor for 11 sensors) with all values separated by a ',', the maximum packet size to be sent added up to (7 * 44)[3] = 308 bytes).

BLE communication is, however, limited to 244 bytes per transaction [98] [99]. To circumvent this limitation, the q_0 quaternion for each sensor was removed from the payload, to be calculated by the game-side processor instead. This allowed the maximum packet size to decrease to (3 * 11 *7) = 231 bytes, enough to send all necessary data over BLE.

As previously mentioned, a characteristic for the sensor count and a 'reset 'characteristic were also broadcast.

[3] six characters for the value + the one for the ',' = 7

6.3 Programming the Algorithm

Whereas the Arduino code in the previous section was inbuilt on the SMARTClap device and will be used in the final product, the Motion Capture Algorithm (MCA) part of the project will, in practice, be inbuilt within the AR game (installed on a PC or Tablet) that the end user (i.e. child with CP) would be playing. Thus, the MCA can be perceived as a black box within the whole system, that takes the input from the Arduino (i.e. the sensors) and outputs the positions to the game, which in turn projects them onto the screen. The steps involved in coding of the Algorithm are shown in Figure 6.9, and the rest of the chapter will more or less follow along its lines.

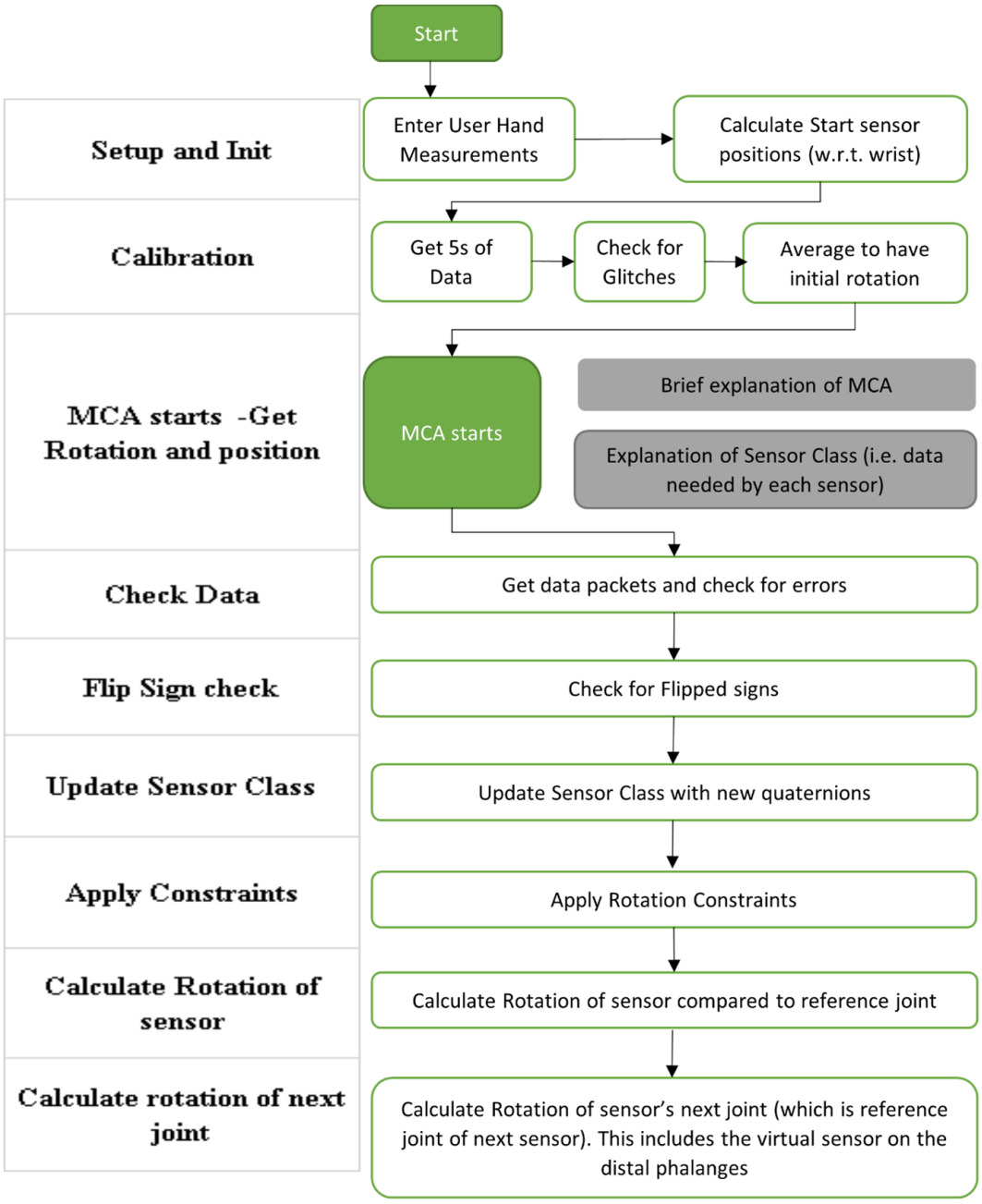


Figure 6.9: MCA flow

6.3.1 Software

The MCA was coded using Python within Visual Studio Code. Within Python itself, several libraries were used to facilitate the building of the algorithm. Visualisation of the sensors on the screen was done using the VPython library [100] even though this was not essential for the final product, reason being that the output positioning from the MCA was used by the AR game for visualisation. Visualisation libraries for AR are available for platforms like OpenAR platform and Microsoft Kinect, used in [59] or the Unity platform. However, since the code in this thesis was going to be implemented in Python the visualisation software was implemented in-house using VPython.

The Bleak library [101] was used to connect the PC to the Arduino via Bluetooth while *pyquaternion* [98] and *dual_quaternion* [99] libraries from *Pypi* were used to ease quaternion operations.

It must be underlined that all code examples given in the following text are taken using the Index Finger since this incorporates the most complicated calculations compared to the Middle finger (where no angles are involved with respect to the wrist). The other fingers where similarly coded.

6.3.2 Setup and Initialization

Before capturing the incoming quaternion data from the sensors, which data describes the rotation of said sensors, the initial positions of the sensors on the hand were calculated. This gave an original position to the whole hand, in respect of which the incoming rotations were then calculated. Furthermore, with this initial data, each sensor 'knows 'its position with respect to the other sensors.

As described in the Section 4.5.1, the sensors are to be positioned at the centre of each phalange and on the back palm, with the origin [0 0 0] being at the wrist. The positions of the sensors were calculated by taking the lengths of the finger (from Metacarpal to fingertip, lf), the length between the origin and the fingertip, l, and the angle between the wrist and each fingertip with the hand in natural position, θ (these measurements were formulated both from literature [81][13], and from a small study carried out with the participation of the SmartClap Consortium members and a class of local students (Table 4.4) (Figure 4.9 from section 4.5 is replicated hereunder (figure 6.10) for reference purposes).

Once the three known measurements (l, lf and θ) were inserted for each finger, all the phalange lengths and sensor positions in the middle of the phalanges were calculated automatically using equations from literature [81, 13] and Table 4.3 (Table 6.1) . Table 6.2 shows how the finger measurements were calculated.

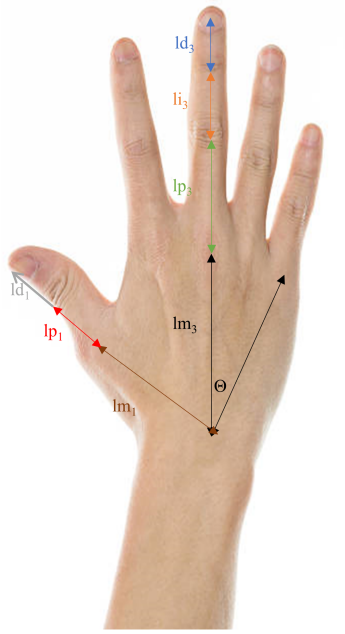


Figure 6.10: Hand measurements, calculated and measured.

Table 6.1: Finger Class Members

Finger class member	**Description**
lt	total length from wrist to tip
lf	length of finger
lm	length of metacarpal
lp	length of proximal phalange
li	length of intermediate phalange
ld	length of distal phalange
lpli	ratio of lp and li
lild	ratio li and ld
IMU1	distance of 1st IMU from MCP
IMU2	distance of 2nd IMU from MCP
angle_wrt_wrist	angle of finger w.r.t. wrist
IMU1pos	x,y,z position of IMU1
IMU2pos	x,y,z position of IMU2
IMU3pos	x,y,z position of IMU2
metacarpaljointpos#	MCP Joint Position
proxinjointpos	PIP Joint Position
distintjointpos	DIP Joint Position
fingertippos	Finger Tip Position

Table 6.2: Calculation of Finger measurements and sensor positioning. The sensor and joint positioning is given in the form of a vector [x,y,z].

Variable	Calculation
lt	measured
lf	measured
lm	lt - lf
lp	lpli * li
li	lild * ld
ld	finger_length/((lild*lpli) + lild + 1)
lpli	Table 4.3
lild	Table 4.3
angle_wrt_wrist (θ)	Table 4.4
IMU1pos (PP)	$-\cos(\theta) * (lm + (\frac{lp}{2}))$ 0 $-sin(\theta) * (lm + \frac{lp}{2})$
IMU2pos (IP)	$-\cos(\theta) * (lm + lp + (\frac{li}{2}))$ 0 $-sin(\theta) * (lm + lp + \frac{li}{2})$
IMU3pos (DP - virtual)	$-\cos(\theta) * (lm + lp + li + (\frac{ld}{2}))$ 0 $-sin(\theta) * (lm + lp + li + \frac{ld}{2})$
metacarpaljointpos (MCPj)	$-\cos(\theta) * (lm)$ 0 $-sin(\theta) * (lm)$
proxinjointpos (PIj)	$-\cos(\theta) * (lm + lp)$ 0 $-sin(\theta) * (lm + lp)$
indisttjointpos (IDj)	$-\cos(\theta) * (lm + lp + li)$ 0 $-sin(\theta) * (lm + lp + li)$
fingertippos (FT)	$-\cos(\theta) * (lm + lp + li + ld)$ 0 $-sin(\theta) * (lm + lp + li + ld)$

6.3.3 Initial Positioning data

Once the hand model was created, the initial sensor angles were calculated to give a reference value with respect to which all their movements will be calculated thereafter. As such, the initial hand position can be in any position as long as it remains stationary for the duration of the calibration data acquisition period, which, in this case, was taken for 5s. [4]). Within this period, an average of 35 readings are taken for each sensor, using

[4]A user needs to keep the hand stationary for several seconds so that the readings are taken. Asking a child with CP to keep hand stationary for a long time can prove to be difficult. Thus, it was decided (with the advice of the occupational therapists) that 5s was a suitable time to ask the children to keep

an average acquisition rate of 7Hz.

Each data packet (which includes three quaternion values for every sensor available, in string format and separated by commas) received is checked for errors, similar to the checks done on the Arduino side.

The first check makes sure the string of characters received is not empty, and if successful, the number of values available are counted to make sure that the total is three times the number of sensors. Finally, each value is checked for not being a 'NaN 'and is between -1 and 1.

A final check regards a 'glitch test '. This is to make sure that any spikes or wayward readings between 2 consecutive data values are removed. If any value is 1% bigger than the previous value, the whole data packet is discarded. This assumes that the first ever reading is correct. These checks are done as well for every acquisition of the MCA, as described later.

Once the 5s have passed, the correct data packets acquired are then averaged and the result is taken both as the reference data packet for the run and the previous data packet for the first packet of the game.

The reference data packet means that every data packet received during the game will calculate its angle of rotation (i.e., movement) with respect to this data packet.

The previous data packet means that the first data packet received when the game starts will be compared to this packet for glitches.

Once calibration is completed, this data packet is saved as the original data in the sensor module class (see section 6.3 for a description). Every 4 values are saved as one quaternion (for that specific sensor), each quaternion is turned into a dual quaternion and saved, and from every quaternion, the original Euler angles [5] – yaw (ψ), pitch (θ), and roll (φ) – compared to the initial hand model angles (thus, should ideally be 0,0,0) are extracted and saved. The conversion equations from quaternion q_0, q_1, q_2 and q_3 to Euler angles are already defined and described in equation (4.7) in Section 4.6.2

The next section will describe the programming data acquired by the algorithm, whose description can be found in Sections 4.6 and 4.6.3.

6.3.4 Sensor Classes

The read and calculated data for each acquisition from each sensor were all saved in a sensor ($Vsensor$) class, including the 'virtual 'data for the distal phalange sensors. All

still.

[5] Euler angles are used in the constraints of the finger movements. It is easier to work in Eulers than quaternions in this case.

the classes are then saved in a *sensor_object* list. The data saved in this list is tabulated in Table 6.3, with 'c 'indicating that the specific variable is a constant throughout the algorithm, and 'u 'indicating that the variable is one that is updated regularly.

Table 6.3: Data (acquired and calculated) saved for each sensor

Variables used by all sensors	
Finger Number	constant
Angle of Finger w.r.t. wrist	c
VPython module holding the positions of the specific sensor (only for development purposes)	updated
Translation Dual quaternion of sensor w.r.t. wrist	u
Translation Dual quaternion of sensor w.r.t. joint of rotation origin	u
Translation Dual quaternion of the Reference Joint (around which the sensor rotates)	u
Translation dual quaternion of next joint	u
Original Dual Quaternion of Sensor	c
Dual Quaternion of Reference Sensor	u
Dual Quaternion of Reference Joint	u
Dual Quaternion	u
Original quaternion qin	u
Sensor Original Angles	c
Sensor Euler Angles (always updating)	u
Variables which have same rotation as Sensor 0 (back palm), used only by sensors 1,3,5,7, and 9	
Translation quaternion of MCP1 Joint – which rotates with same quaternion as sensor 0 (Back Palm)	u
Translation quaternion of MCP2 Joint	u
Translation quaternion of MCP3 Joint	u
Translation quaternion of MCP4 Joint	u
Translation quaternion of MCP5 Joint	u
Variables only used by the Intermediate Phalange Sensors, i.e. sensors 2,4,6, and 8.	
Dual Quaternion of the Distal Module	u
Dual Quaternion of Fingertip	u

For better understanding of some variables, Figure 5.22 is being reproduced here (Figure 6.11) but with the sensors numbered accordingly. Sensor 0 is the reference sensor for the proximal phalange sensors of Fingers 2-5. The sensors on the proximal phalange (1,3,5,7,9) are the reference sensors of the intermediate phalange sensors (2,4,6,8,10) respectively. And these latter are the reference sensors for the virtual sensors on the distal phalanges (41,61,81,101) of fingers 2-5.

The thumb needed an extra virtual sensor (sensor 11) on its metacarpal, which rotates with the sensor on the thumb proximal phalange. In other studies ([13] [14], a sensor was always placed on the metacarpal of the thumb since this rotates freely around the carpo-metacarpal joint with 3 DOFs, unlike the other Metacarpals, which are fixed to the palm. However, for this work, a sensor could not be positioned in this point since

it was impractical to place it on a child with CP [6]. Hence a different model was built for the thumb, based on the fact that for CP therapy, the most important movement on the thumb is the opposition movement with the index and middle fingers.

In this model the rotation of the Thumb Metacarpal is based on the rotation of the proximal phalange sensor. An assumption was also made that the first 30 °yaw rotation from the initial position was a rotation - of both the metacarpal and the proximal phalange - around the carpometacarpal joint, whereas after this, the rotation was around the metacarpal joint (which remains stationary). In this way, the opposition movement stands, however, two DOF movements of the thumb were lost.

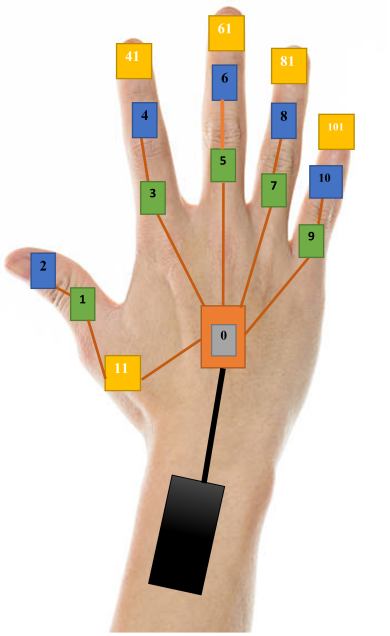


Figure 6.11: Hardware model with sensor numbers. Green boxes denote proximal phalange sensors, blue boxes denote intermediate phalane sensors (distal in case of the thumb) and yellow boxes denote virtual sensors

6.3.5 MCA – get position data

The main algorithm is comprised of seven steps, following the last steps in Figure 6.9:

- Check data packet for errors and glitches.
- Apply sign-flipping if necessary.
- Update the sensor class.

[6]It was advised by the CP therapists that sticky tape and/or a whole cloth glove like the ones used in the cited research are too rough for the sensitive skin of children suffering from CP. Furthermore, the way that 3D-printed SMARTClap glove was designed could not allow for a sensor on the Thumb Metacarpal. This is an issue which needs further research for improvement

- Apply constraints on the received rotation.
- Calculate the rotation of the sensor compared to its reference joint.
- Calculate the rotation of next joints.
- Calculate the rotation of the virtual sensors and fingertips.

Check Data for errors and glitches

This step was divided in 2 parts. The first step checks if the values themselves are correct (Figure 6.12) while the second step checks that there are no glitches between one packet an the previous one (Figure 6.13). A glitch is detected if the difference between two consecutive values is more than 1% (this value was established heuristically after conducting several tests).

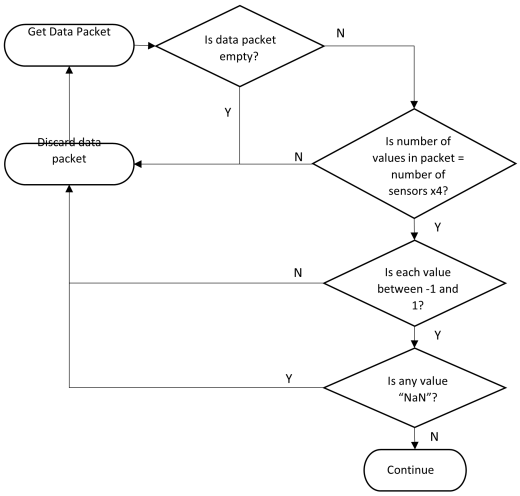


Figure 6.12: Data checking for correct values.

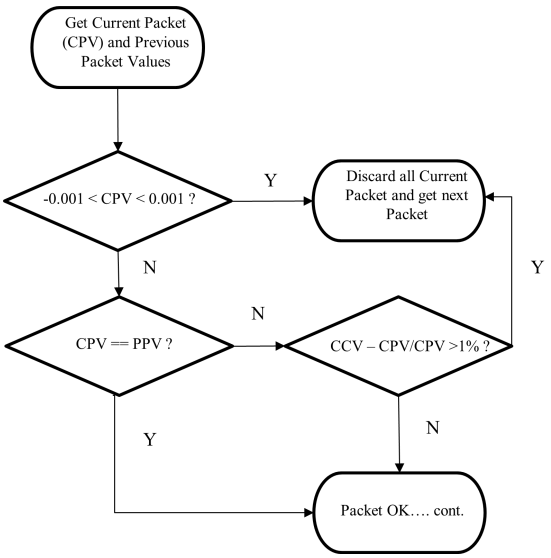


Figure 6.13: Check data for glitches and errors.

Flipsign

Depending on the rotation of the sensors, all four quaternions should have a value between -1 and 1. Since the sensors' fusion algorithm outputs only q_1, q_2 and q_3, the scalar term of the quaternion, q_0, is calculated post-capture (from Equation 6.1) by the equation:

$$q_0 = \sqrt{1 - (q_1^2 + q_2^2 + q_3^2)}$$

From Equation 6.2, q_0, which is the scalar term of the quaternion and is defined as $\cos\frac{\theta}{2}$, should have a range of values from 1 to -1. However, when plotting the raw quaternions coming out of the IMU, this was not the case, as can be seen in Figure 6.14, which plots the raw data of the quaternions while the sensor was rotated by 360°around its y-axis, and shows that q_0 (blue plot) never goes negative.

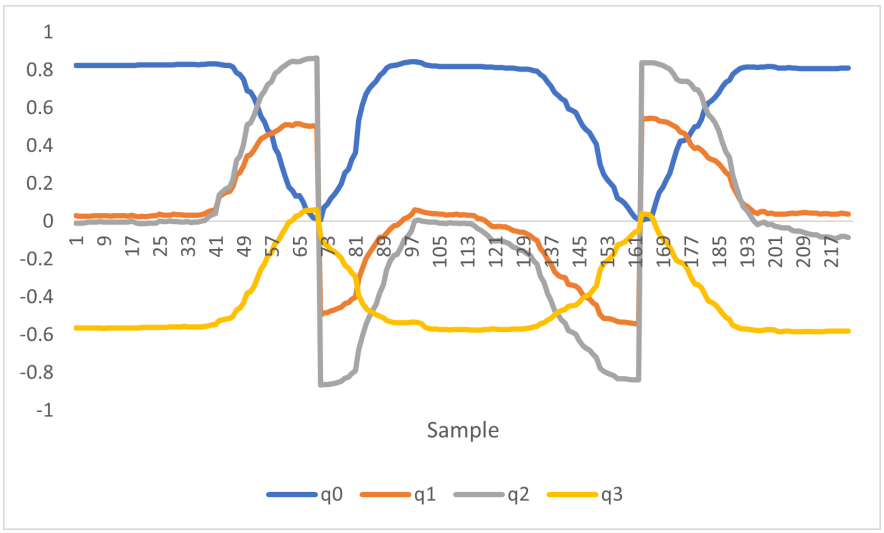


Figure 6.14: Raw Quaternion output from ICM20948 while turning it 360°around the y-axis

This was occuring because the calculation of equation 6.2 in software (both Arduino and Python) always provides a positive value of q_0.

Since a rotation can be represented by two quaternions (q and $-q$), it can be seen in Figure 6.15 that the correct result can be achieved by flipping the sign of q_0 when it reaches 0. This problem was resolved by comparing the magnitudes of all elements of the quaternion to the previous quaternion's elements. If the quaternion element with the biggest magnitude in the previous acquisition has changed sign, then the sign of all the other elements of the new quaternion is flipped. The result of the change from Figure 6.14 is shown in Figure 6.15, showing that q_0 is now varying from -1 to 1. The flowchart of the algorithm is shown in Figure 6.16.

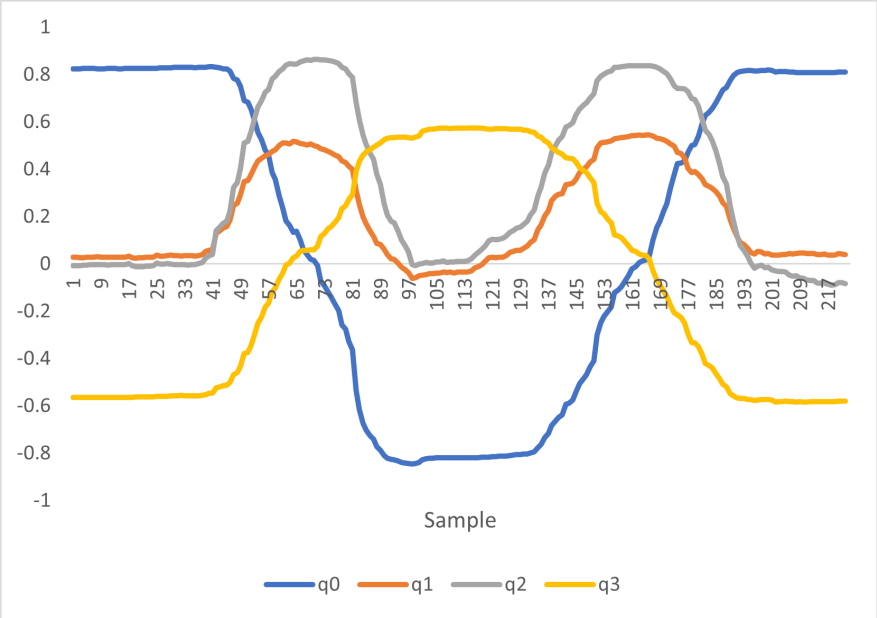


Figure 6.15: Quaternion output after flip-sign algorithm included of sensor turning 360°around y-axis

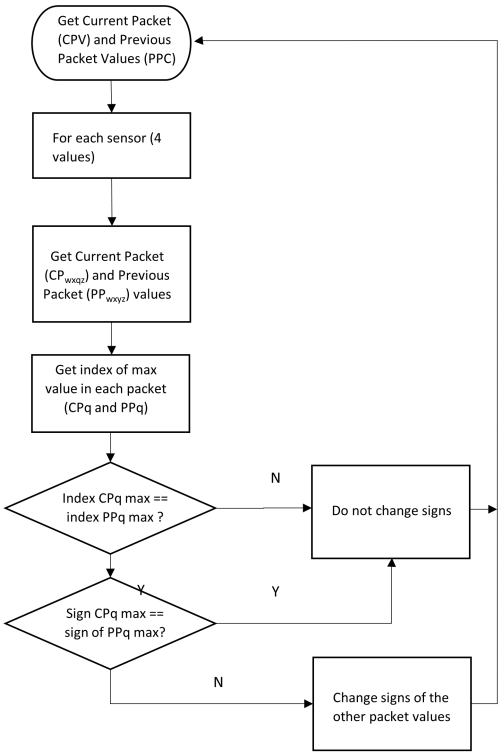


Figure 6.16: Flow to check if quaternion sign is to be switched or not

Update sensor class

Once the packet has been verified, it is saved as the current packet to be used in the algorithm, and eventually as the previous packet during the next iteration. The packet is saved in the sensors class described in Table 6.3 of Section 6.3.4. For every sensor, its values are saved as a rotation quaternion and a dual quaternion, and the respective Euler angles extracted and saved using Equation 4.7.

Applying constraints

Constraints were applied so that the rotation of the sensors is limited to the maximum angles that the hand and fingers can extend/flex or adduct/abduct. Constraints are applied according to [14] and Table 4.2 in section 4.4.2. The flow is shown in figure 6.17.

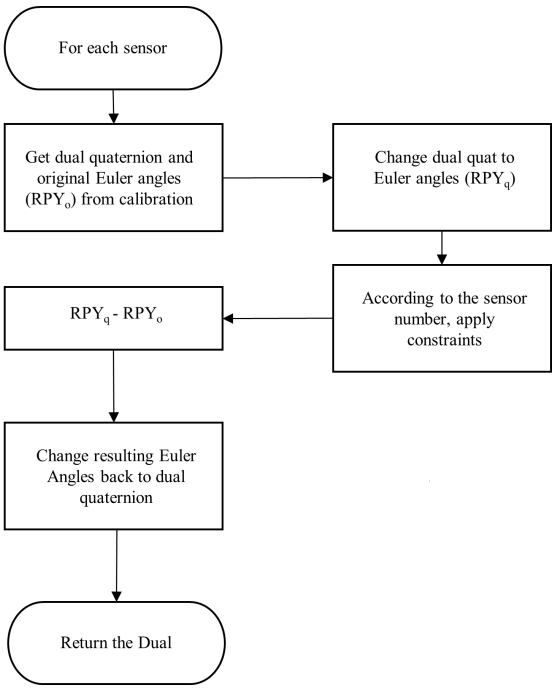


Figure 6.17: Flow to applying angle constraints.

Calculation of rotation of sensors with respect to the reference

The rotation of the sensors was calculated with reference to the joint around which it rotates, as shown in Figure 6.18

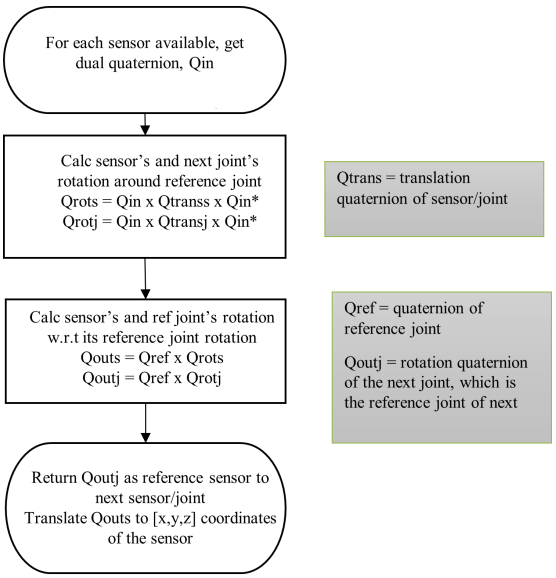


Figure 6.18: Flow to calculate rotation of incoming quaternions

Calculate next joints

The position of the next joint is calculated by applying the same rotation calculations of the sensor, with the difference being the translation from the reference joint. This

has already been shown in the previous section for the simple case of one finger (i.e. the metacarpal joint (MCPj) is the reference joint for proximal phalange sensor and proximal-intermediate phalange joint (PIPj), whereas the PIPj is the reference joint of intermediate phalange sensor and distal-intermediate phalange joint (DIPj), etc.). This is not so for the wrist joint, which is the reference joint of the five MCPjs. The five MCP joints all have the same reference joint as the Back Palm Sensor (Sensor 0), so once the position of the back palm sensor is calculated, there are five 'next joint 'positions to be calculated (Figure 6.19).

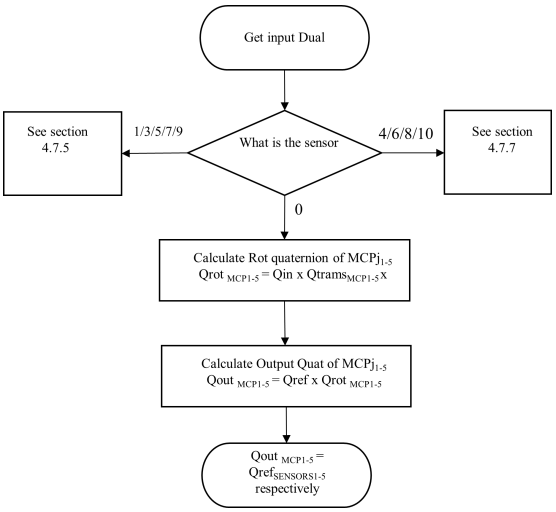


Figure 6.19: Flow to Calculate positioning of next joints

Calculate position of virtual sensor and Fingertip

Finally, to calculate the position of the fingertip, the position of the virtual sensor from the DIPj reference quaternion was needed. This holds for all fingertips apart from the thumb, since the fingertip position was calculated from sensor 2, positioned on the distal phalange. Rotation of the virtual sensors (numbered as: 41/61/81/101) and the respective fingertips was extrapolated from the rotation of the Intermediate Phalange sensors (4/6/8/10) by using equation 4.1 (Figure 6.20).

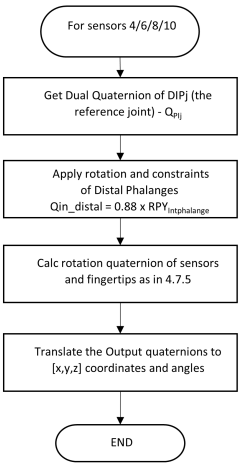


Figure 6.20: Flow showing calculation of virtual sensor and fingertip.

6.4 Conclusion

This chapter outlined how the Motion Capture Algorithm was coded and implemented.

The software used to program the SMARTClap device's hardware was introduced and explained in Section 6.2. It described how the Arduino code was divided in a setup section and a loop section to extract quaternion data from the sensors and send it over BLE to the AR game device.

In the longer and more detailed Section 6.3, the actual algorithm of how the quaternion data was manipulated to give the orientation and final positioning of the hand was described.

The next chapter will discuss the testing done to validate the algorithm and the results achieved.

Chapter 7 – Testing and Results

7.1 Introduction

In this chapter, the testing and validation procedures carried out to check that the Motion Capture Algorithm (MCA) is working correctly are discussed, as well as the results to show it. Testing and validation of the algorithm with the hardware was achieved in three parts. Firstly, during the development stage, Visual Python [1]and a simple inhouse-built rig were used to test and visualise the progress of the algorithm, one finger at a time. Once the algorithm was ready and the DigiClap glove was fully assembled, the motion tracking accuracy of the glove was tested by comparing it to the VICON system while the positioning accuracy of the fingertips was tested by grabbing different objects with known dimensions. Testing was carried out using only three of the five fingers (thumb, index and middle) since these are the most important during CP therapy sessions and also due to lack of electronic components at the time of testing. This also made testing easier, since for the ring and little fingers the algorithm is the same as for the index, with changes only to the finger measurements and angle with respect to the wrist.

7.2 Single finger testing

During the development stage of the MCA, Visual Python was used to visualise the movements of the sensors with respect to one another, even though it was known that there would be no need to visualise the hand in the final version of the game. To mimic the movement of the finger phalanges and the back palm, a simple test-rig with 3 moveable links was built, with a sensor attached to each link (Figure 7.1). Each sensor was then represented by a sphere in Visual Python (Figure 7.2).

These tests where verified visually, by making sure that motion of the linked arms with respect to one another was mirrored on VPython.

[1]As previously mentioned,Visualisation libraries for AR are available for platforms like OpenAR platform and others, however it was decided that visualisation was to be built using VPhython since the algorithm was to be built using python code, and these platforms use other languages

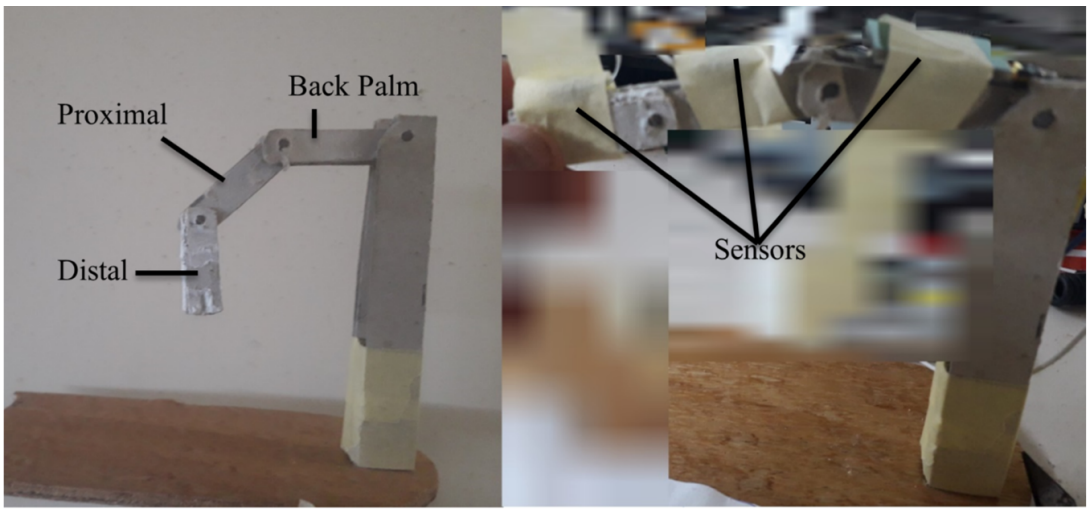


Figure 7.1: Test rig built for development (left) and with sensors attached (left)

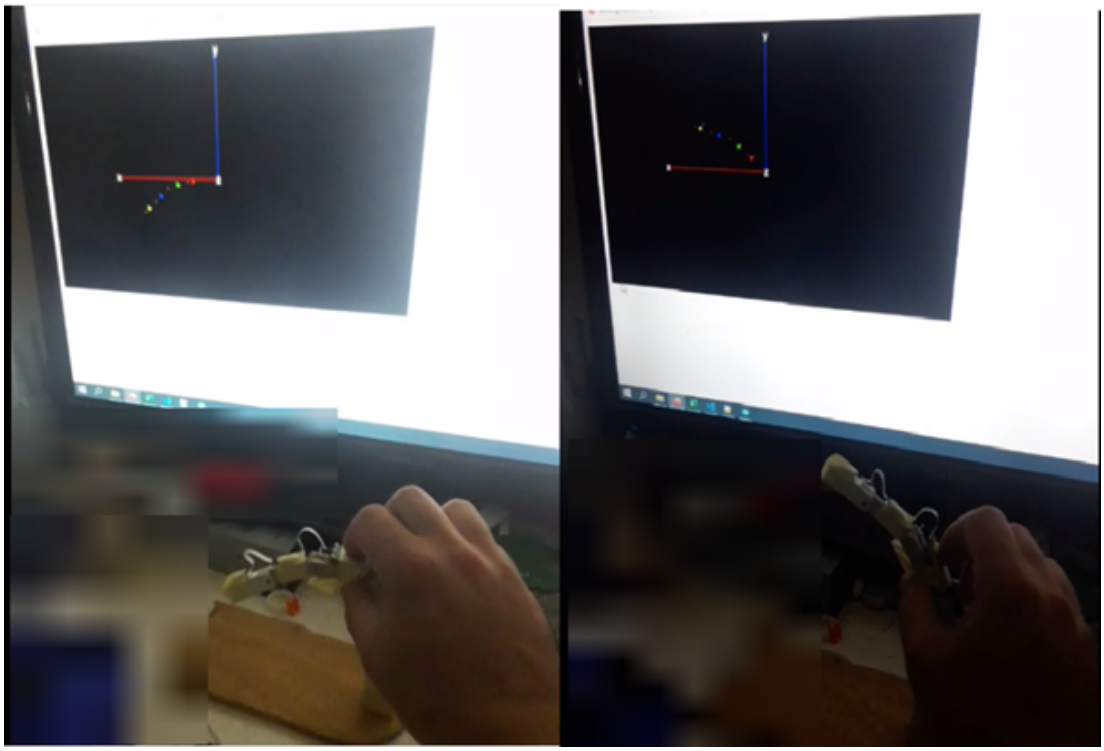


Figure 7.2: Test Rig and Visual Python. Red, green, blue, and yellow spheres represent the back palm, proximal, intermediate, and virtual distal sensors respectively..

Finally, once the algorithm was working for a single finger, the whole hand was included in the visualisation for testing, as shown in figure 7.3.

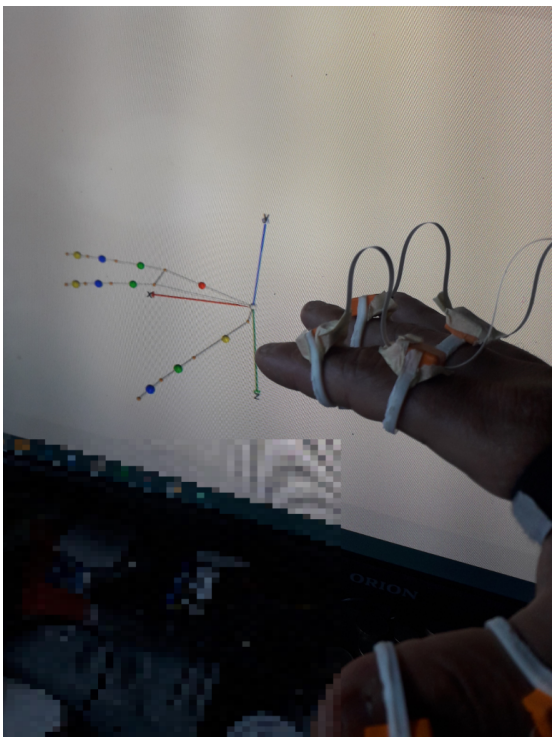


Figure 7.3: Visualisation of hand in Visual Python (Red, green, blue and yellow spheres represent the back palm, proximal, intermediate and virtual sensors respectvely. Small orange spheres represent the joints in between.)

7.3 Testing Systems

Once the algorithm was ready for testing, the whole device was assembled, with the electronics enclosed in their 3D-printed enclosures, designed as part of Work Package 3 of the SMARTClap project. Figure 7.4 shows the assembled device with six sensors (three fingers) attached to the back palm connector board. For testing purposes, and due to the lack of electronic components at the time of testing, only three fingers were used as this was deemed enough to validate the work done since these three fingers are the one used most by children suffering from CP.

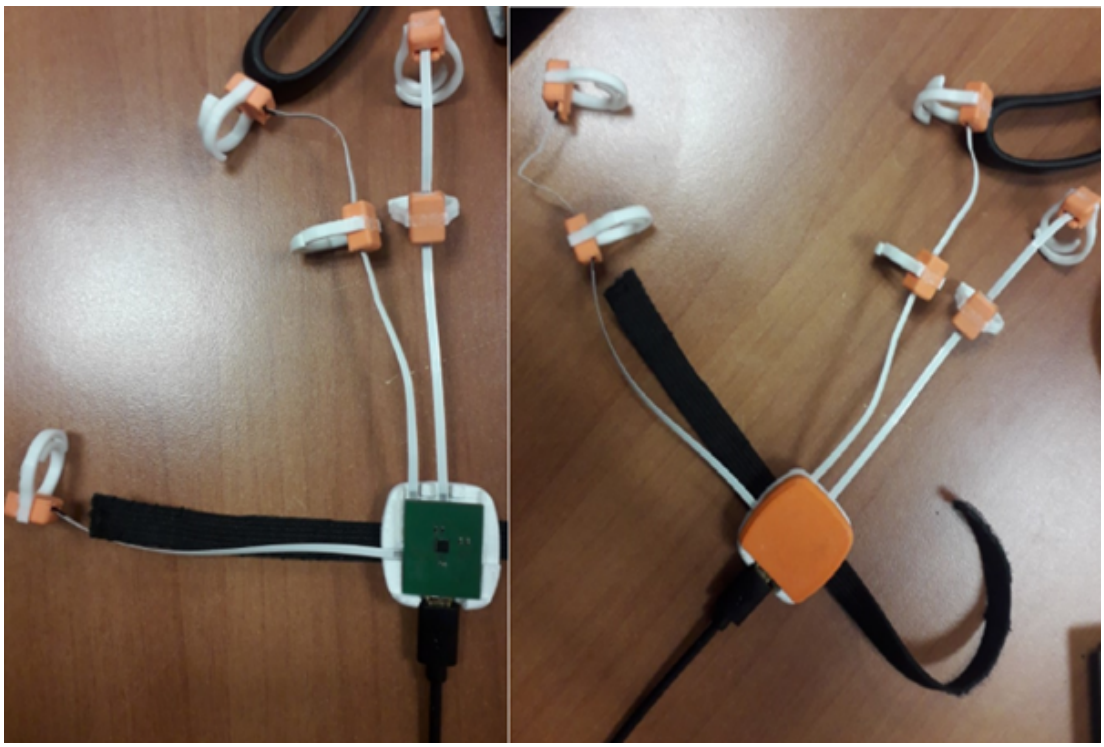


Figure 7.4: Assembled SMARTClap wearable device (Main Board enclosure not visible). The opened connector board containing the back palm sensor can be seen on the left picture.

To test the algorithm, two procedures were planned:

- A motion tracking test to see if the sensors move correctly in 3D space with respect to each other, comparing with the VICON system as reference.
- A measurement accuracy test, where opposition motion of the thumb with the index and middle finger was tested, as well as grabbing an object to measure the distance between fingertips. This is performed without the VICON reference system.

7.3.1 The VICON System

The motion tracking tests were carried out in the Biomedical Systems Laboratory at the Department of Systems and Control Engineering at the University of Malta. The VICON Motion Capture system includes a network of 8 cameras set up around a room, which can track the motion of a subject through IR signals reflected off special reflectors attached to said subject, like the one shown in Figure 7.5.

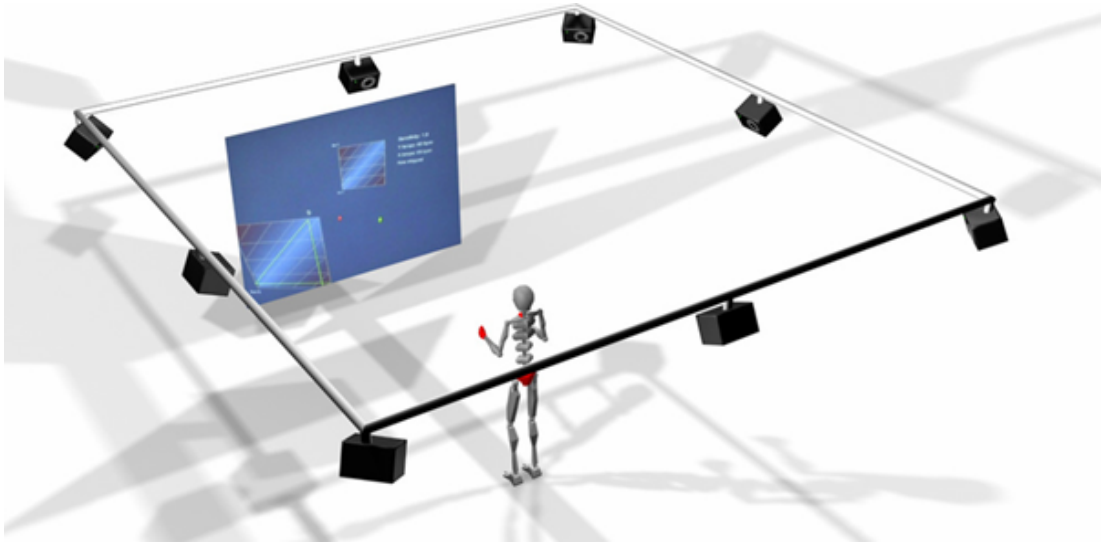


Figure 7.5: The Vicon™ Motion Capture- based system, with the reflectors shown in red [50]

To ensure as much similarity of the measurement as possible between the VICON system and the SMARTClap device, a VICON reflector was attached to each sensor enclosure of the device, as well as on the 4 distal phalanges of the hand, as shown in Figure 7.6. The offsets between the centre of the reflectors and the sensor boards, and the fingertip was noted for each distal reflector (including the thumb), so that both the rotation angle and the positioning of the finger phalanges can be calculated. The VICON system has an acquisition frequency of 100Hz.

Furthermore, as can be seen in Figure 7.6, a reflector was also placed on the wrist, which was used as the origin point for the VICON data gathered (also placed in the position where the SC algorithm assumes the origin point to be). This is because the VICON system is already calibrated for a [0,0,0] origin point somewhere in the lab, which is not the same point where the rig was placed. Thus, to make the wrist position the [0,0,0] point, during the post processing of the results and before comparison between the two systems (VICON and the SMARTClap device), all the data read from the hand and finger reflectors was first normalised in comparison to the data gathered from the wrist reflector, by subtracting by the [X,Y,Z] values of each reflector by the [X,YZ], values of the wrist reflector.

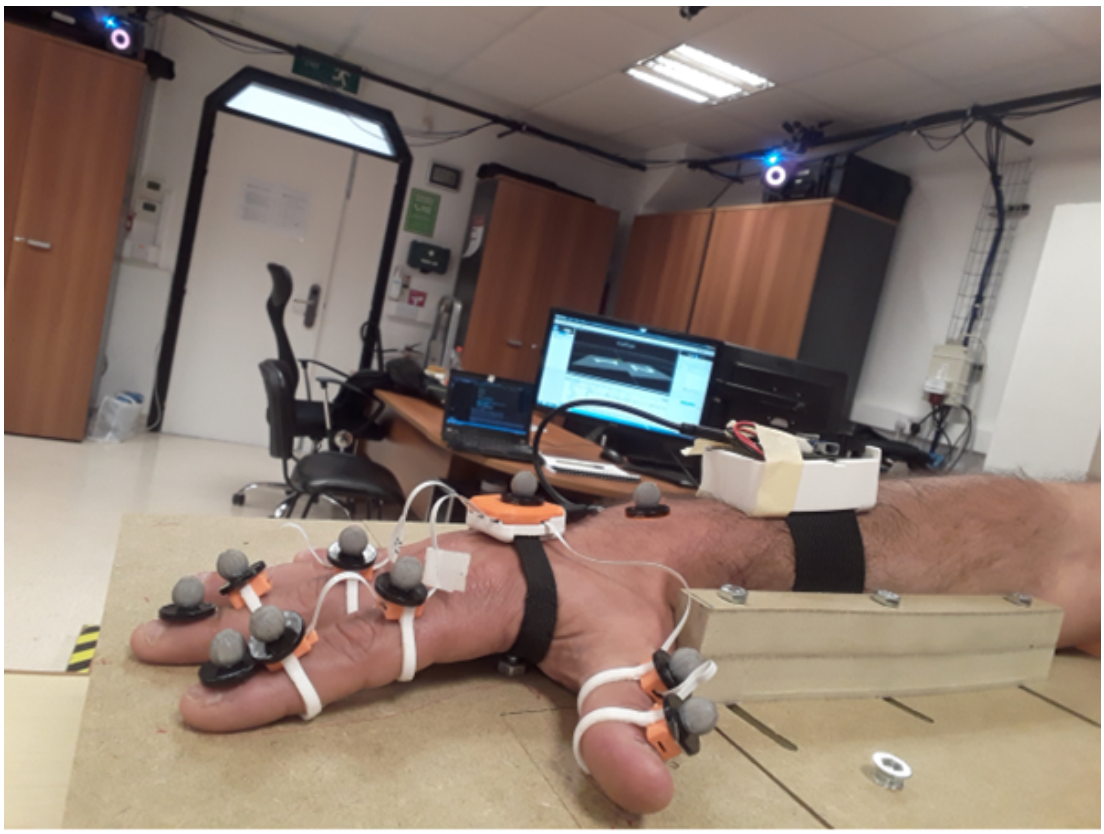


Figure 7.6: SMARTClap glove with spherical VICON reflectors attached. Two VICON IR cameras can be seen in the background (blue lights).

7.3.2 Rig Setup

To ensure consistency in readings taken, a test rig was built specifically for these tests (Figure 7.7). It was divided in 4 sections. The back part is fixed and is where the forearm sits. To make sure that the arm does not move during the different tests, 2 wooden planks were fitted to hold it in place. These planks were also designed to be movable to accommodate arms of different sizes. The front section was designed to be able to bend at an angle, so that the whole hand could move only from the wrist. This section was then divided in two other sections, which allowed for the movement of only the 4 fingers (index to pinkie) and the thumb. The angles at which these sections were bent was determined by stumps designed with different angles and attached to the bottom of the rig, as can be seen in figure 7.8, with an example of the thumb on the right picture.

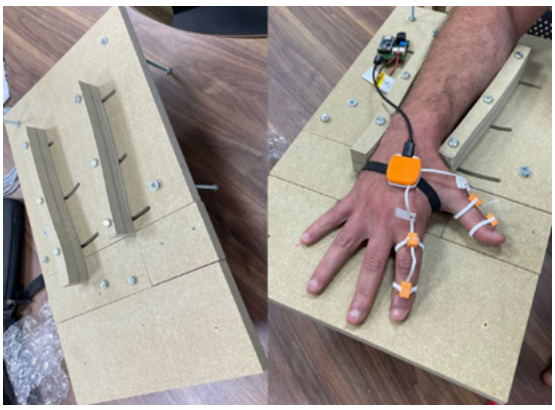


Figure 7.7: Test Rig

Figure 7.8: Stumps to set maximum bending angle (right) and an example with the thumb(left).

7.4 Testing Procedures

7.4.1 Motion Tracking

With both setups (VICON and SC) running, the hand was first rotated forward at a 30°angle around the wrist in a flexion movement, while trying to keep the wrist steady. The hand was kept stationary at this angle for 10 seconds. This movement was repeated three times per session, and three sessions were carried out in total by the same person.

The same set of experiments were done with a 60°angle.

Validation of the results was achieved by comparing the stationary angles over all the tests as well as verifying if the SC output tracks the motion of the VICON output correctly.

Due to the different sampling time between the VICON and SC systems (100Hz vs 8Hz), it was not possible to compare the raw data on a sample-by-sample basis. Hence, both datasets (VICON and SC) were re-sampled (using the Python *resample* command) so that the datasets had comparable sample numbers upon which comparison

could be made, as explained better in section 7.5.1. After this, the sampled data (over all the three sessions) was compared using three different methods, as shown in table 7.2.

Table 7.1: Tests and comparisons Performed

Test Name	Test Performed
M1	The difference in the average values between SC and VICON over the time when the rig is at an angle
M2	The difference in the average values between SC and VICON over the time when the rig is at its stationary position (i.e. ideally at 0°)
M3	The average error over the whole sampled dataset, including the movement periods

Thus, for tests M1 and M2, the average error between the SC and VICON readings was calculated, for every sensor, using equation:

$$E(t) = |AverageSC - AverageVICON| \tag{7.1}$$

where *AverageSC* are the average values of the SC system and $AverageVICON$ are the average values of the reference system.

In test M3, for every sensor, the difference of every sample between the SC and VICON datasets was taken and the total was averaged similar to (7.1). For all the tests, the standard deviation of the error STD(E) and the Root Mean Square Error (RMSE) (equation (7.2)) were calculated from the difference between the estimated values from SC and the values from VICON.

$$RMSE(t) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Average - AverageVICON)^2} \tag{7.2}$$

7.4.2 Dimension and Opposition Tests

The algorithm tested with the VICON system works fine for certain movements. However due to the way the VICON system is setup, movement of different fingers can be difficult to track due to occlusion of some of the markers. Such movement include pinching (finger opposition) and object grabbing, two movements which are critical for CP therapy. Thus, a further three tests were carried out without the VICON reference system, two pinching tests (an opposition between thumb and index, and the other between thumb and middle), and a dimensions test where a ball with known dimensions

was grabbed and released to see if the distance between the fingers matched. Each test was repeated 3 times. The tests performed are tabulated in Table 7.2 and shown in Figure 7.9.

Table 7.2: Tests Performed

Test Number	Test Performed
Test 1	Pinch test between thumb and index finger
Test 2	Pinch test between thumb and middle finger
Test 3	Grabbing a 5cm diameter ball

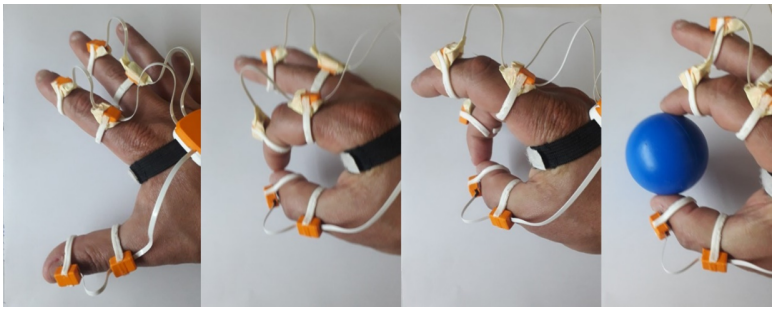


Figure 7.9: Dimensions and pinch/grab test movements

For each test, the XYZ coordinates of the distal phalange sensors of the thumb and the index/middle finger (the virtual sensors in this case) were noted. Consideration was taken for the distance between the sensors and the finger pad, and this distance was offset in the readings of the Y coordinate for the index and middle fingers and the X coordinate of the thumb.

For all three tests, the vector difference between the thumb (T) and the finger pad positions of the Index and middle fingers (F) was noted, and the magnitude of the vector (M) for each sample instance (t) calculated for according to equation (7.3) . The magnitude of the vector represents the distance between the two fingers.

$$M(t) = \sqrt{(X_F - X_T)^2 + (Y_F - Y_T)^2 + (Z_F - Z_T)^2} \tag{7.3}$$

where X_F, X_T, Y_F, Y_T, Z_F and Z_T are the X, Y and Z coordinates of the finger (index or middle) and the thumb respectively, at time t.

7.5 Results

7.5.1 Post Processing

Figure 7.10 shows a typical plot of the raw data of the motion tracking tests, using the Back Palm sensor (flexion of the wrist at 30°x 3 times, for 10s each time) as an example. The top plot shows the VICON results while the bottom shows the SMART-Clap (SC) device results. The SC data that was taken is the raw data outputted from the algorithm. However, the VICON data is taken is the raw data from each reflector (placed on each sensor) with respect to the data gathered from the reflector placed on the wrist origin.

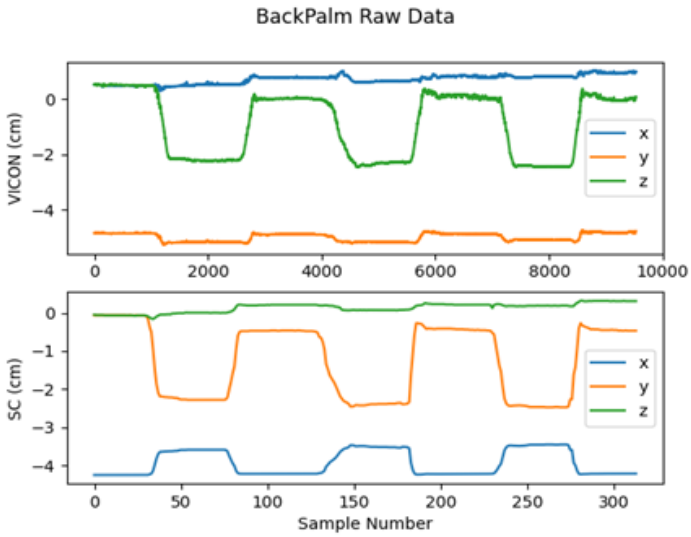
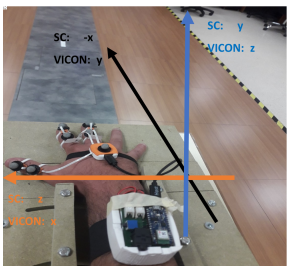


Figure 7.10: Raw data output for the back palm sensor for the 3 axes , for VICON (above) and SC device (below). Note the difference in axis labels in the legend, which is explained later in Figure 7.11

It is to be noticed that the legends of the two plots in figure 7.10 (VICON and SC) reference different readings. E.g. the blue x-axis in the VICON plot matches the green z-axis in the SC plot, similarly for the y/x-axis and z/y-axes plots for VICON/SC respectively. This difference in annotation is due to the difference in orientation of the axes between the ICM- 20948 IMU and the VICON system, as shown in Figure 7.11. In this dissertation, the SC notation is used.

From the same figure 7.10, it is also noted that the blue plot of the x-axes (in the SC plot) and the corresponding axes in the VICON orange plot (i.e., the y-axis) move in opposite directions, i.e. while the SC plot goes from trough to peak, the VICON plot goes from peak to trough. This is shown better in the normalised plots of figure 7.11 and is due to the difference in orientation between the ICM-20948 IMU (x) and the VPython visualisation (-x). Hence to correctly visualise the hand on VPython, the x-axes had to be oriented in the negative direction.



SC		VICON
-X	⇨	Y
Y	⇨	Z
Z	⇨	X

Figure 7.11: Difference in orientation between VICON and SC systems

From figure 7.10, also note the offset at the starting point of the X-axes compared to the other two axes in the SC plot. This is marked on the plots of Figure 7.10 and it occurs because the back palm sensor is positioned in a location where the x coordinate is at an offset of 4.2cm away from the origin point at [0,0,0] while the y and z coordinates are sitting in a position which is not offset from the origin point.

Furthermore, it is noted that there is another offset of the VICON data (z and x-axes) compared to the SC data (z and y-axes). While the SC data starts at 0, the VICON data starts at an offset of around 0.5cm. This was assumed to be due to the different positionings of the back palm sensor and the wrist sensor and the fact that, as explained previously in section 5.3, the [X, Y, Z] values of each reflector were normalised (i.e. subtracted from) with respect to the [X, Y, Z] values of the wrist reflector, to make this latter point the [0,0,0] origin. Thus, for example, if the back palm sensor was sitting at 4.5cm away from the origin, its position vector should read [4.5,0,0], with an offset of 4.5cm on the x- axes, (going away from the origin in direction of the fingertips, using SC notation) and 0cm offset on the other directions. However, as can be seen in figure 7.6, the back palm reflector and the wrist reflector are not at exactly the same level in the y- axis direction (upwards direction, using SC notation), because the back palm reflector is sitting on the back palm housing of the SC device, and is hence positioned slightly higher than the wrist sensor. This explains the offset seen in figure 7.10 for the y- axis.

In spite of the anomalies explained, the two plots do follow the same trends. However, due to the different sampling rate between the two systems (the VICON sampling rate is 100Hz while the SC samples at 8Hz), the two datasets were first re-sampled to analyse them better with the same sample number and timescale, then normalised to remove the offsets so that the first point is at 0, as shown in figure 7.12.

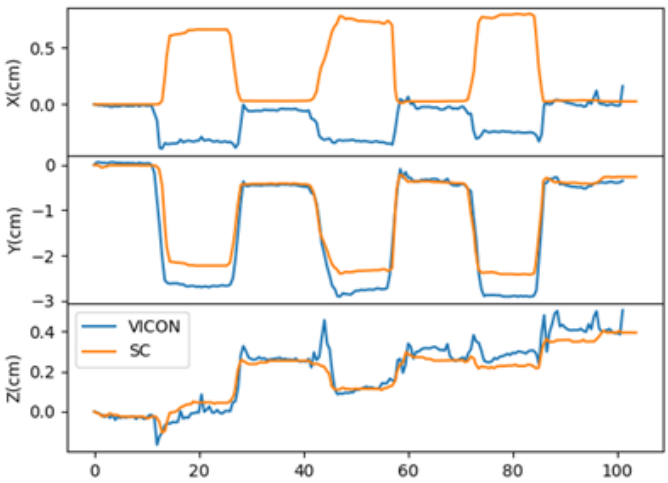


Figure 7.12: Displacement in the 3D space of back palm sensor while doing a 30°rotation around the Z- axis of the wrist. The displacement axes are according to the ones shown in Figure 7.11. E.g. Y(cm) denotes the amount of displacement done along the y- axis (i.e. vertical) using the SC coordinate frame.

Figure 7.12 shows that the SC data tracks the movement of the VICON system well, albeit the offset from the VICON data in the X and Y axis which was noted throughout the datasets acquired. It is not known what creates this consistent offset, however it has been recorded in other cases in literature [102].

The raw data coming out of the VICON and the SC denotes the positioning of the finger sensors on the phalanges, in XYZ space. For a better understanding of the motion, this data was transformed into rotation data (angles in degrees), in a similar way as in [102]. Hence, every output position in terms of x, y and z was assumed as a 3D vector whose origin is the reference joint (the wrist at position [0,0,0] in the example above), and the angle rotated around the reference joint is the angle between the current 3D vector and the original vector, assumed to be a straight line from the origin to the sensor, on the X- Z plane. The algorithm uses the $atan2$ function between the cross and dot product of each point compared to the original to find the angle between the two, as shown in equation (7.4).

$$[Yaw, Pitch, Roll](t) = atan2[O \times V(t), O \cdot V(t)] \tag{7.4}$$

where O and V are the original and current vectors (in form of [x,y,z]) respectively. The O vector in this case is the 1st vector of the dataset, whereas the V is every other vector in the same dataset. An example of the output is shown in Figure 7.13.

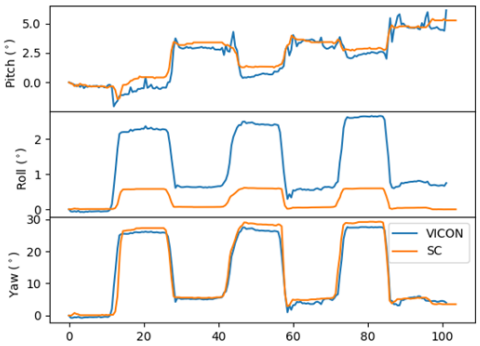


Figure 7.13: Angles derived from the output positions of the two devices for the Back Palm sensor. Roll is rotation around the X- axis, Yaw is the rotation around the Z- axis, Pitch is the rotation around the Y- axis

In the example shown, the hand has been rotated up and down, around the Z- axis, thus, as expected, it is the yaw angle that showed the biggest rotation, the rotation which is of interest to us in this specific case. The discrepancies and offsets shown in the three rotations are consistent across all other readings, as shown in the example for the Yaw, in Figure 7.14.

There is more discrepancy on the Thumb sensors. This is mainly due to the thumb lacking a degree of freedom in its motion as explained previously in Section 6.3.4. Another reason is due to the way the thumb sensors are connected, laterally on the thumb rather than flat on top like the other fingers. Since the movement shown is of the whole hand rotating around the z- axis, while the other sensors are flat and rotate only around their z- axis, the thumb sensors rotate around the other axis as well (due to their positioning), which might not affect the yaw as much as the two other rotations (pitch and roll).

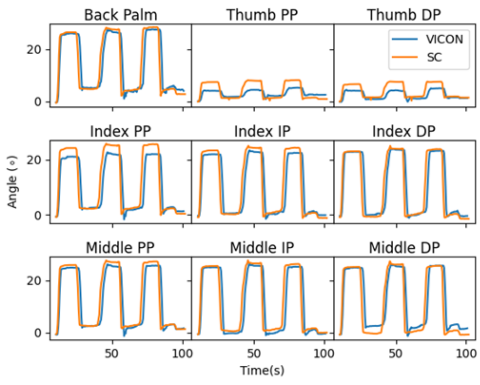


Figure 7.14: Angles derived from the output positions of the two devices for the Back Palm sensor. Roll is rotation around the X- axis, Yaw is the rotation around the Z- axis, Pitch is the rotation around the Y- axis

7.5.2 Motion Tracking Results

Data from the three repetitive tests carried out was gathered and the results were divided in three sections: the difference between SC and VICON in the peaks (Test M1 i.e. when the hand was at an angle e.g. 30°), the difference in the troughs (Test M2 i.e. when the hand was at stationary level i.e. 0°) and the overall difference, between the VICON and the SC system data (Test M3).

Tables 5, 6 and 7 show the results of tests M1, M2 and M3 respectively. The Mean and Root Mean Square Error (RMSE) were calculated from equation (7.1) and equation (7.2).

Table 7.3: Statistical Results of Test M1 (Peaks)

Sensor Number	**Average** (°)		**Mean Error** (°)	**STD of the Error** (°)	**RMSE** (°)
	VICON	SC			
1- Back palm	24.98	26.62	1.64	2.53	3.02
2 - Thumb PP	5.13	7.36	2.23	1.03	2.45
3 - Thumb DP	5.33	6.9	1.57	1.31	2.05
4 - Index PP	21.59	25.14	3.54	2.58	4.38
5 - Index IP	23.25	25.1	1.84	2.48	3.09
6 - Index DP	24.24	25.18	0.93	2.47	2.64
7 - Middle PP	25.76	26.82	1.06	2.87	3.06
8 - Middle IP	25.97	26.81	0.84	2.71	2.84
9 - Middle DP	25.76	26.55	0.79	3.4	3.49

Table 7.4: Statistical results of Test M2 (Troughs))

Sensor Number	**Average** (°)		**Mean Error** (°)	**STD of the Error** (°)	**RMSE** (°)
	VICON	SC			
1- Back palm	5.27	7.53	2.26	1.36	2.64
2- Thumb PP	1.93	2.15	0.22	0.92	0.95
3- Thumb DP	1.05	2.53	1.48	0.29	1.51
4- Index PP	3.3	5.62	2.33	0.77	2.45
5- Index IP	4.54	4.45	1.47	2.52	2.92
6- Index DP	3.04	4.08	2.16	1.1	2.43
7- Middle PP	4.31	6.2	1.9	0.87	2.09
8- Middle IP	3.63	4.88	1.25	3.91	4.11
9- Middle DP	4.31	4.07	0.23	1.19	1.21

Table 7.5: Statistical results of test M3 (whole sample average)

Sensor Number	**Average** (°)		**Mean Error** (°)	**STD of the Error** (°)	**RMSE** (°)
	VICON	SC			
1- Back palm	13.73	13.65	1.47	0.46	2.72
2- Thumb PP	2.79	3.86	1.74	0.67	2.26
3- Thumb DP	2.24	3.8	1.59	0.76	2.01
4- Index PP	10.42	11.59	2.19	0.89	3.17
5- Index IP	9.63	10.41	1.8	0.89	2.9
6- Index DP	9.89	9.95	1.37	0.83	2.79
7- Middle PP	12.13	12.53	1.5	0.73	2.74
8- Middle IP	10.98	11.57	1.68	0.57	3
9- Middle DP	12.13	11.02	1.97	0.24	3.16

The results in all tables show a good correlation between the SC and the VICON systems. The mean error falls between 0.20° and 3.54°, with an overall average error of 1.43° , which is both well within the discrimination threshold of the just noticeable differences experienced in the joints (between 2.5° and -2.7°) [77]. This is the threshold angle which humans can distinguish as a joint movement. Also, these results are very comparable with similar works, with a mean of the estimated error of 3.3°in [14], and a maximum error of 2.6°in [13].

7.5.3 Opposition and Dimension Results

Figure 7.15 shows a representative series of Test 1 (from Table 7.2), i.e., the magnitude of the vector between the finger pads of the index and the thumb, calculated from equation 7.3 in section 7.4 . The index and thumb were opened and closed four times, with the dashed line showing the 0-magnitude mark, which would be the ideal position since it means that the distance between the two fingers is 0cm, and hence they are touching, as shown by a screenshot taken during one of the tests in Figure 7.16

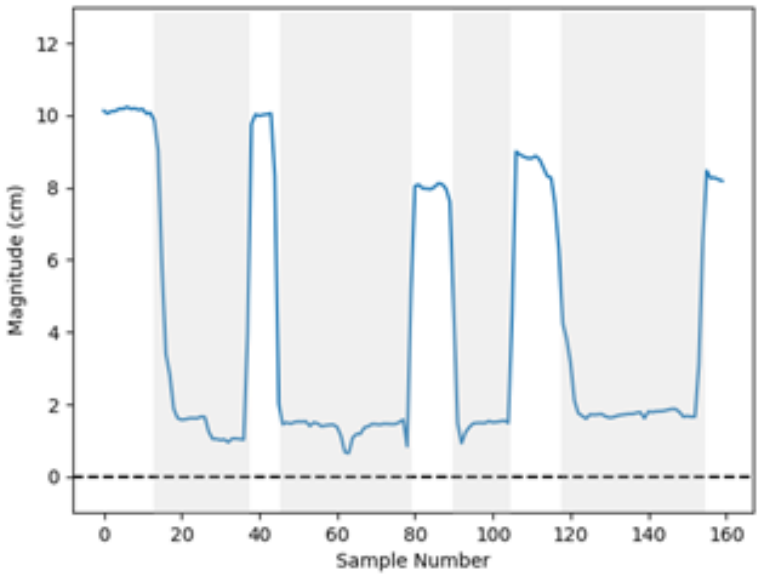


Figure 7.15: Magnitude of distance vector for opposition (pinch) between thumb and index. The dashed line shows the ideal position when pinching occurs, i.e. 0cm. Grey areas denote moments when the thumb and index were in opposition.

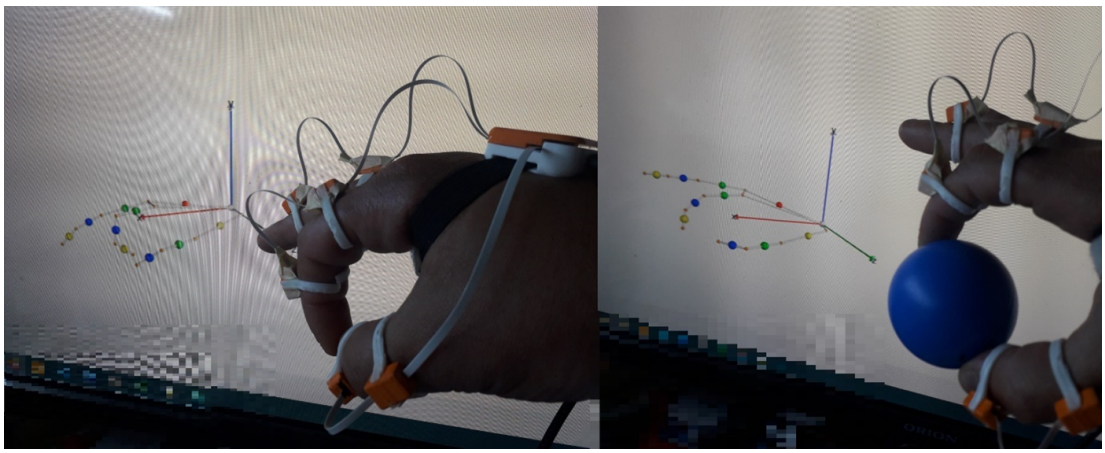


Figure 7.16: Visualisation of pinching (left) and dimension (right) positions with VPython on the PC. The red sphere represents the back palm sensor. Green spheres represent the proximal phalange sensors. Blue spheres represent the intermediate phalange (distal for the thumb) . Yellow spheres represent the virtual sensors.

The results of the three tests are tabulated in Table 7.6, showing the average values for the 3 tests conducted (over a number of values when the fingers were in opposition for Test 1 and Test 2, and during the time the ball was being grabbed in Test 3), their standard deviation and the Root Mean Square Error between the obtained values and the expected values. In the case of Tests 1 and 2, the expected values where 0cm (there should be no distance between fingers for the pinching movement) and 5cm (distance between the fingers when grabbing the 5cm diameter ball) for Test 3.

Table 7.6: Statistical results of the dimension and pinch motion tests from the SC device only. RMSE is taken compared to the expected values (0cm for test 1 and test 2, and 5cm for Test 3).

	Average (cm)	**STD**	**RMSE**
Test 1 – pinch between thumb and index	1.57	0.77	1.75
Test 2 – pinch between thumb and middle	3.75	0.73	3.8
Test 3 – grabbing a 5cm-diameter ball	5.79	1.47	1.84

The pinching results of test 1 (pinching between thumb and index finger) shows an estimated average error of 1.7cm, which matches well with other studies [13] (1.5cm).

The result of test 2 (pinch between the thumb and the middle finger) results in a bigger estimated error. While the results were constant and consistent, this error was attributed to the limited rotation of the thumb in the algorithm that resulted from the assumptions taken for its rotation due to a lack of sensor on the Metacarpal and the consequent loss of 2 DoFs of its rotation. Physiologically, to reach the middle finger, the thumb

must stretch a bit more from the Metacarpal bone (the simple rotation of the distal phalange is not enough in this case). However, due to the limited rotation, this is not currently possible in the algorithm as it stands, because the thumb is not allowed that much freedom of movement. Allowing such freedom resulted in distorted movements. This is something that needs to be worked upon in the future.

Test 3 resulted in a 15% error from the expected result of 5cm. Like previous tests, this error was consistent over the three repetitions of the tests, and again was attributed to the lack of ideal movement of the thumb due to its missing DoFs.

7.6 Conclusion

This chapter has described the testing procedures carried out in order to test the MCA described in Chapter 6, which was used for the SMARTClap project.

Section 7.2 described the setup used during the development of the algorithm.

The systems to be used for testing were introduced in Section 7.3 and the procedures carried out using these mentioned systems where explained in Section 7.4. This included the comparison of the SMARTClap device to the VICON reference system as well as measuring the width of a ball with known dimensions, and the distance between fingers during pinching movements.

The results, in section 7.5, show good correlation between the movement captured by the SMARTClap device and the VICON reference system. The error margins where similar to other systems in literature, with a maximum error of 3.54°achieved when compared to the 3.3°and 2.6°reported by [13] and [14] respectively.

The pinching tests showed a a bigger error when there is opposition motion between the thumb and middle finger than between the thumb and the index finger. This was attributed to the missing sensor on the Metacarpal (MCP) of the thumb (when compared to other similar studies [13, 14]) and consequently on the assumptions taken because of this, namely, that the motion of the MCP of the thumb was based on the motion of the proximal phalange, which deprived sensing of the thumb's two degrees of freedom of the Carpo-Metacarpal joint. The same reason was attributed to the differences seen in the dimension test results.

The results show a need of improvement in the hardware in order to obtain better sensing of the motion of the thumb, specifically that the motion of the thumb MCP needs to be captured by a sensor. This was not possible in the current set-up due to restrictions with the wearable device's 3D design. Future works and modifications could address these issues.

Chapter 8 – Conclusion

This chapter will conclude and summarise the work carried out in this dissertation. The aims and results of the work are outlined, as well as the achieved outcomes. Finally, limitations and future improvements are discussed

8.1 Research Aims and Results

8.1.1 Aims

This research formed part of the SMARTClap project, which aims to design and manufacture a 3-D printed, user-centred wearable device to be used by children suffering from Cerebral Palsy (CP) during their rehabilitation, while they interact with a serious Augmented Reality (AR) game. CP is one of the most common disabilities in childhood, and research shows that interacting with Robot-Aided Gait Therapy (RAGT) can improve the child's motivation to relearn the motor functions lost due to the disease.

The aim of this work was divided in two parts:

- To design sensory hardware for the wearable device that is small and light enough to fit into 3D-printed housing designed to sit on a 4-11 year-old child's fingers and hand that can capture the motion of the hand and fingers. This device should also be able to send wireless data to an AR-based central processing unit (e.g. PC or tablet)
- Using the data retrieved from the sensory hardware, a Motion Capture Algorithm (MCA) needed to be developed and implemented to track the real-time motion of the hand an fingers.

8.1.2 Results

Results show that the main aims of the research as highlighted previously were reached. The hardware, based on IMUs on each finger phalange, correctly captured the motion of the fingers and successfully sent this data via Bluetooth to the main processor. Using the visualisation platform built specifically for the development of the MCA, it was shown that finger motion was correctly captured and tracked by the MCA.

From the tests carried out to validate the MCA, motion tracking of the rotation of the fingers and hand when compared to an industry standard video-based motion capture reference systems (the VICON system) shows an overall average variation of 1.38°, well within the discrimination threshold experienced by the joints.

Opposition motion tests between the thumb and index, which is a crucial motion to use for children during therapy, resulted in an average error of 1.7 cm which is comparable with other studies in literature. The same function test between the thumb and the middle finger experienced a bigger error, which was attributed to a lack of motion DoFs of the thumbs, as explained in Section 8.3.

Grabbing tests showed quite a large error of 15% when grabbing a 5cm ball. This, again was attributed to the lack of DoF tracking of the thumb, since the results were consistent over a number of repetitions.

8.2 Research Outcome

This work brought together research from other studies [13, 14] that effectively used dual quaternions with a minimum number of sensors to capture the free movement of the fingers and the hand. It provides a finished solution of a chargeable, battery-powered, motion capture wearable device, capable of running for longer than the 45 minutes that CP therapy sessions typically take.

The MCA and hardware were designed to have a short calibration process compared to how IMUs are normally calibrated using a ‘figure of 8 ’technique [85]. This was needed because children suffering from CP might have difficulty moving their arms and hand. Thus the calibration process was designed to take 5 seconds before the game starts. There is also a 25 second period when the sensors need to be stationary in order to stabilise. This latter period is however only necessary when the device is switched on. Another design outcome is the flexibility to be able to use a different number of fingers depending on the therapy session, apart from the thumb which has to be always present.

Furthermore, the hardware is designed and programmed so that there is a possibility for the device to be worn on both hands. This feature is not yes functional, and is described furtherl in the future improvements section.

The MCA is designed in a way that, unlike similar studies [13, 14], a sensor is not positioned on the thumb’s metacarpal but only on the thumb’s proximal phalange. This was a limitation imposed by the 3D-printed design structure adopted, arising from the difficulty of attaching a sensor to the MCP of a child with sensitive skin. As the results show, this was not an ideal solution (due to the lack of complete sensing of all the DoFs of the thumb) and further work is needed in this regard. However it proved to be good enough to be used by children with CP since the angles of rotation of the joints, outputted from the sensors, are still captured and can be used by the therapists.

Finally, the device and MCA can be used in other fields as well (e.g. stroke rehabilitation, generic gaming), and is not necessarily constrained to CP therapy.

8.3 Limitations, improvements and further work

The main drawback of the MCA proposed in this work, as previously described, is the lack sensing of the free movement in 2 DoFs of the thumb due to no sensor being placed on its Metacarpal. This can be improved by further research on the assumptions made in the algorithm itself, or by adding a sensor which captures the 3DoFs of the Metacarpal bone, and on which the movement of the proximal phalange sensor is based.

The Main Controller board (described in section 5.3.2) which captures the measurements from the sensors and sends them to the main processor via Bluetooth can be further reduced in size and weight by replacing the Arduino Nano on the board with a Bluetooth processor chip (e.g. [103]). This would not only reduce the PCB's footprint (and hence, the 3D-printed housing) but also would allow for a reduction of one of the batteries used to power the device, since it can be powered from 3.3V rather than from 7.2 V like the Arduino Nano.

Improvements and further work on the Arduino Nano software includes adding further functionality to the RGB LED on the board so that this can give light feedback during game play (e.g. if there are sensor faults, which sensor or which finger is the fault on). It currently only lights up during the initialization process of the sensors.

Further improvements would include the functionality to automatically detect the hand in which the device is worn using the data in figure 6.3 as described in section 6.2.1. This would be done by monitoring the quaternion stabilization data coming from the thumb sensor, which is different from the other four fingers.

Appendix A – Sensor Board BOM

Table A.1: Finger Class Members

Parts	Qty	Value	Description
C1, C2, C3	3	0.1uF	Capacitor
R1, R2	2	10k	Resistor
SJ1	1		SMD solder jumper
J1, J2	2	FH33J - 4S - 0.5SH10	FFC & FPC Connectors
U1	1	ICM- 20948	IMU

Appendix B – Main Board BOM

Parts	Qty	Value	Description
BAT1, BAT2	2		Battery
C1, C4, C5	3	10uF	Capacitor
C2, C3	2	1uF	Capacitor
C6	1	0.1uF	Capacitor
CONNBOARD_ CONN	2	10118192 0001LF	Micro USB 2.0 Connector to connect to Connector board
J1	1	SJ3589ANG	Audio Jack Connector Switch
JP1	1	Jumper Pins	Pin Headers
JP1_BATT1, JP2_BATT2	2	BM02B SRSS TB	Battery Connectors
LED1	1	Blue	LED
LED2	1	Green	LED
LED3	1	RED	LED
R1, R2, R3	3	220	Resistor
R4	1	50	Resistor
R5	1	330	Resistor
RGB_LED	1	ASCB JTC2 0A308	Standard LEDs SMD 4 LED
RT1_THERM	1	NCP03XH103J05RL	Thermistor 10K
S1	1	TG39W000050	PCB Slide Switch
U1	1	TPS51103DRCR	Voltage Regulator
U2	1	Arduino Nano	Arduino Nano V3.0 Module
U3	1	MCP73833	Battery Charger Regulator

Appendix C – Connector Board BOM

Parts	Qty	Value	Description
C1, C2, C3	3	0.1uF	Capacitor
C4	1	10uF	Capacitor
F1_CONN, F2_CONN, F3_CONN, F4_CONN, F5_CONN	5	FH33J 4S 0.5SH_10	FFC & FPC Connectors
MAINBOARD_ CONN	1	10118192 0001LF	Micro B USB 2.0 Connector to connect to main board
R1, R2, R3, R4, R5, R6, R7, R8	8	10k	RESISTOR
U1	1	TCA9548APWR	I2C Multiplexor
U3	1	ICM 20948	Sensor

Bibliography

[1] NHS, "Gait therapy," 2023. [Online]. Available: https://www.smart.scot.nhs.uk/service/gait-lab/

[2] NHS, "Continuous glucose monitoring," 2023. [Online]. Available: https://www.niddk.nih.gov/health-information/diabetes/overview/managing-diabetes/continuous-glucose-monitoring

[3] M. Clinic, "Cerebral palsy," 11 2023. [Online]. Available: https://www.mayoclinic.org/diseases-conditions/cerebral-palsy/symptoms-causes/syc-20353999

[4] C. Arnould, M. Penta, A. Renders, and J.-L. Thonnard, "Abilhand-kids: A measure of manual ability in children with cerebral palsy," *Neurology*, vol. 63, pp. 1045–52, 09 2004.

[5] A.-C. Eliasson, L. Krumlinde-Sundholm, B. Rösblad, E. Beckung, M. Arner, A.-M. Ohrvall, and P. Rosenbaum, "The manual ability classification system (macs) for children with cerebral palsy: scale development and evidence of validity and reliability," *Developmental medicine and child neurology*, vol. 48, pp. 549–54, 08 2006.

[6] B. E, *The multiply handicapped child*. CambridgeUniversity Press, 1997.

[7] C. Bayon, R. Raya, S. L. Lara, O. Ramirez, J. Serrano, and E. Rocon, "Robotic therapies for children with cerebral palsy: a systematic review," *Transl Biomed*, vol. 7, no. 1, p. 44, 2016.

[8] A. Meyer-Heim and H. J. A. van Hedel, "Robot-assisted and computer-enhanced therapies for children with cerebral palsy: current state and clinical implementation," vol. 20, p. 139–145, 2013.

[9] FaceBook, "Inside facebook reality labs: Wrist-based interaction for the next computing platform," Mar. 2021. [Online]. Available: https://tech.fb.com/inside-facebook-reality-labs-wrist-based-interaction-for-the-next-computing-platform/

[10] Bosch, "Smart sensor: Bno055," 2023. [Online]. Available: https://www.bosch-sensortec.com/products/smart-sensors/bno055/

[11] Invensense-TDK, "Icm20948 datasheet," 2023. [Online]. Available: https://invensense.tdk.com/wp-content/uploads/2016/06/DS-000189-ICM-20948-v1.3.pdf

[12] G. Wu, F. C. van der Helm, H. (DirkJan) Veeger, M. Makhsous, P. Van Roy, C. Anglin, J. Nagels, A. R. Karduna, K. McQuade, X. Wang, F. W. Werner, and B. Buchholz, "Isb recommendation on definitions of joint coordinate systems of various joints for the reporting of human joint motion—part ii: shoulder, elbow, wrist and hand," *Journal of Biomechanics*, vol. 38, no. 5, pp. 981–992, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S002192900400301X

[13] C. Salchow-Hömmen, L. Callies, D. Laidig, M. Valtin, T. Schauer, and T. Seel, "A tangible solution for hand motion tracking in clinical applications," *Sensors (Basel, Switzerland)*, vol. 19, no. 30626130, p. 208, 01 2019. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6339214/

[14] T. L. Baldi, S. Scheggi, L. Meli, M. Mohammadi, and D. Prattichizzo, "Gesto: A glove for enhanced sensing and touching based on inertial and magnetic sensors for hand tracking and cutaneous feedback," *IEEE Transactions on Human-Machine Systems*, vol. 47, no. 6, pp. 1066–1076, 2017.

[15] S. E. Fasoli, B. Ladenheim, J. Mast, and H. I. Krebs, "New horizons for robot-assisted therapy in pediatrics," *American journal of physical medicine & rehabilitation*, vol. 91, no. 11, pp. S280–S289, 2012.

[16] M. Sandlund, S. McDonough, and C. . Hager-Ross, "Interactive computer play in rehabilitation of children with sensorimotor disorders: a systematic review," *Developmental Medicine & Child Neurology*, vol. 51, no. 3, pp. 173–179, 03 2009. [Online]. Available: https://doi.org/10.1111/j.1469-8749.2008.03184.x

[17] K. D. Jennings, R. E. Connors, and C. E. Stegman, "Does a physical handicap alter the development of mastery motivation during the preschool years?" *Journal of the American Academy of Child and Adolescent Psychiatry*, vol. 27, no. 3, pp. 312 – 317, 1988. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0890856709655829

[18] H. Sveistrup, "Motor rehabilitation using virtual reality," *Journal of NeuroEngineering and Rehabilitation*, vol. 1, no. 1, p. 10, 2004. [Online]. Available: https://doi.org/10.1186/1743-0003-1-10

[19] S. H. You, S. H. Jang, Y.-H. Kim, Y.-H. Kwon, I. Barrow, and M. Hallett, "Cortical reorganization induced by virtual reality therapy in a child with hemiparetic cerebral palsy," *Developmental Medicine & Child Neurology*, vol. 47, no. 9, pp. 628–635, 09 2005. [Online]. Available: https://doi.org/10.1111/j.1469-8749.2005.tb01216.x

[20] A. F. Leal, T. D. da Silva, P. B. Lopes, S. Bahadori, L. V. de Araújo, M. V. B. da Costa, A. P. de Moraes, R. H. Marques, T. B. Crocetta, L. C.

de Abreu, and C. B. d. M. Monteiro, "The use of a task through virtual reality in cerebral palsy using two different interaction devices (concrete and abstract) - a cross-sectional randomized study," *Journal of NeuroEngineering and Rehabilitation*, vol. 17, no. 1, p. 59, 2020. [Online]. Available: https://doi.org/10.1186/s12984-020-00689-z

[21] M. Bonello, "Upper limb taxonomy chart," *SMARTClap Project*, 2021.

[22] S. Ahn and S. Hwang, "Virtual rehabilitation of upper extremity function and independence for stoke: a meta-analysis," *J Exerc Rehabil*, vol. 15, no. 3, pp. 358–369, 06 2019. [Online]. Available: http://www.e-jer.org/journal/view.php?number=2013600688

[23] Y. Chen, L.-J. Kang, T.-Y. Chuang, J.-L. Doong, S.-J. Lee, M. Tsai, S.-F. Jeng, and W.-H. Sung, "Use of virtual reality to improve upper-extremity control in children with cerebral palsy: A single-subject design," *Physical therapy*, vol. 87, pp. 1441–57, 12 2007.

[24] D. T. Reid, "The use of virtual reality to improve upper-extremity efficiency skills in children with cerebral palsy: A pilot study," *Technology and Disability*, vol. 14, no. 2, pp. 53–61, 2002.

[25] D. Reid and K. Campbell, "The use of virtual reality with children with cerebral palsy: A pilot randomized trial," *Therapeutic Recreation Journal*, vol. 40, no. 4, pp. 255–68, 2006.

[26] Mindflux, "P5 glove," 2006. [Online]. Available: http://www.mindflux.com.au/products/essentialreality/p5glove.html

[27] C. G. Systems, "Cyber glove iii," 2017. [Online]. Available: http://www.cyberglovesystems.com/cyberglove-iii/

[28] E. Carmeli, S. Peleg, G. Bartur, E. Elbo, and J.-J. Vatine, "Handtutortm enhanced hand rehabilitation after stroke – a pilot study," *Physiother. Res. Int.*, vol. 16, no. 4, pp. 191–200, 12 2011. [Online]. Available: https://doi.org/10.1002/pri.485

[29] D. D. Wille, K. Eng, L. Holper, E. Chevrier, Y. Hauser, D. Kiper, P. Pyk, S. Schlegel, and A. Meyer-Heim, "Virtual reality-based paediatric interactive therapy system (pits) for improvement of arm and hand function in children with motor impairment—a pilot study," *Developmental Neurorehabilitation*, vol. 12, no. 1, pp. 44–52, 2009. [Online]. Available: https://doi.org/10.1080/17518420902773117

[30] K. Van Hedel, Wick and K. Eng, “Improving dexterity in children with cerebral palsy preliminary results of a randomized trial evaluating a glove based vr-system,” 2011.

[31] Kiper, Verra, Bernhard, and Shuster, “Effectiveness of the yougrabber system using virtual reali-ty in stroke rehabilitation- study protocol of a single blind-ed randomised controlled multi-centre tria,” *PLOS ONE*, 2014. [Online]. Available: https://journals.plos.org/plosone/article/file?type=supplementary&id=info:doi/10.1371/journal.pone.0204455.s002

[32] C. Schuster-Amft, A. Henneke, B. Hartog-Keisker, L. Holper, E. Siekierka, E. Chevrier, P. Pyk, S. Kollias, D. Kiper, and K. Eng, “Intensive virtual reality-based training for upper limb motor function in chronic stroke: a feasibility study using a single case experimental design and fmri,” *Disability and Rehabilitation: Assistive Technology*, vol. 10, no. 5, pp. 385–392, 2015. [Online]. Available: https://doi.org/10.3109/17483107.2014.908963

[33] STMicroelectronics, “Flexible smart glove,” no. PCTUS1021483, 2012. [Online]. Available: https://patents.google.com/patent/US9529433B2/en

[34] J.-H. Shin, M.-Y. Kim, J.-Y. Lee, Y.-J. Jeon, S. Kim, S. Lee, B. Seo, and Y. Choi, “Effects of virtual reality-based rehabilitation on distal upper extremity function and health-related quality of life: a single-blinded, randomized controlled trial,” *Journal of neuroengineering and rehabilitation*, vol. 13, no. 26911438, pp. 17–17, 02 2016. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4765099/

[35] H.-T. Jung, H. Kim, J. Jeong, B. Jeon, T. Ryu, and Y. Kim, “Feasibility of using the rapael smart glove in upper limb physical therapy for patients after stroke: A randomized controlled trial,” in *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2017, pp. 3856–3859.

[36] IDRT, “Accelleglove,” 2012. [Online]. Available: https://www.idrt.com/GestureRecognition.php

[37] G. Pradhan, N. Engineer, M. Nadin, and B. Prabhakaran, “Integration of motion capture and emg data for classifying the human motions,” *2007 IEEE 23rd International Conference on Data Engineering Workshop*, pp. 56–63, 04 2007.

[38] L. K. Simone and D. G. Kamper, “Design considerations for a wearable monitor to measure finger posture,” *Journal of NeuroEngineering and Rehabilitation*, vol. 2, no. 1, p. 5, 2005. [Online]. Available: https://doi.org/10.1186/1743-0003-2-5

[39] L. K. Simone, N. Sundarrajan, X. Luo, Y. Jia, and D. G. Kamper, "A low cost instrumented glove for extended monitoring and functional hand assessment," *Journal of Neuroscience Methods*, vol. 160, no. 2, pp. 335–348, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165027006004754

[40] R. Gentner and J. Classen, "Development and evaluation of a low-cost sensor glove for assessment of human finger movements in neurophysiological settings," *Journal of Neuroscience Methods*, vol. 178, no. 1, pp. 138–147, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165027008006559

[41] S. Bakhshi and M. H. Mahoor, "Development of a wearable sensor system for measuring body joint flexion," pp. 35–40, 05 2011.

[42] L. Dunne, P. Walsh, B. Smyth, and B. Caulfield, "Design and evaluation of a wearable optical sensor for monitoring seated spinal posture," *Proceedings - International Symposium on Wearable Computers, ISWC*, pp. 65–68, 10 2006.

[43] L. Dipietro, A. M. Sabatini, and P. Dario, "A survey of glove-based systems and their applications," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 4, pp. 461–482, 07 2008.

[44] G. Saggio, "Mechanical model of flex sensors used to sense finger movements," *Sensors and Actuators A: Physical*, vol. 185, pp. 53 – 58, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0924424712004621

[45] N. Tongrod, S. Lokavee, N. Watthanawisuth, A. Tuantranont, and T. Kerdcharoen, "Design and development of data glove based on printed polymeric sensors and zigbee networks for human-computer interface," *null*, vol. 8, no. 2, pp. 115–120, 03 2013. [Online]. Available: https://doi.org/10.3109/17483107.2012.737540

[46] Shinya Kusuda, Satoshi Sawano, and Satoshi Konishi, "Fluid-resistive bending sensor having perfect compatibility with flexible pneumatic balloon actuator," pp. 615–618, 01 2007.

[47] Y. Mengüç, Y.-L. Park, H. Pei, D. Vogt, P. M. Aubin, E. Winchell, L. Fluke, L. Stirling, R. J. Wood, and C. J. Walsh, "Wearable soft sensing suit for human gait measurement," *The International Journal of Robotics Research*, vol. 33, no. 14, pp. 1748–1764, 11 2014. [Online]. Available: https://doi.org/10.1177/0278364914543793

[48] D. M. Vogt, Y. Park, and R. J. Wood, "Design and characterization of a soft multi-axis force sensor using embedded microfluidic channels," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 4056–4064, 10 2013.

[49] Z. Shen, J. Yi, X. Li, M. H. P. Lo, M. Z. Q. Chen, Y. Hu, and Z. Wang, "A soft stretchable bending sensor and data glove applications," *Robotics and biomimetics*, vol. 3, no. 28003951, pp. 22–22, 2016. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5133288/

[50] D. Robertson, "Vicon motion capture," 07 2013. [Online]. Available: https://www.researchgate.net/publication/249963535_Vicon_Motion_Capture

[51] S. Harris, "Inertial measurement unit (imu) – an introduction," *Online*, 2023. [Online]. Available: https://www.advancednavigation.com/tech-articles/inertial-measurement-unit-imu-an-introduction/

[52] Yoon Sang Kim, Byung Seok Soh, and Sang-Goog Lee, "A new wearable input device: Scurry," *IEEE Transactions on Industrial Electronics*, vol. 52, no. 6, pp. 1490–1499, 12 2005.

[53] H. G. Kortier, V. I. Sluiter, D. Roetenberg, and P. H. Veltink, "Assessment of hand kinematics using inertial and magnetic sensors," *Journal of NeuroEngineering and Rehabilitation*, vol. 11, no. 1, p. 70, 2014. [Online]. Available: https://doi.org/10.1186/1743-0003-11-70

[54] H. Luinge, P. Veltink, and C. Baten, "Ambulatory measurement of arm orientation," *Journal of Biomechanics*, vol. 40, no. 1, pp. 78–85, 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021929005005282

[55] S. Cobos, M. Ferre, M. A. Sánchez-Urán, and J. Ortego, "Constraints for realistic hand manipulation," 2007.

[56] C.-E. Hrabia, K. Wolf, and M. Wilhelm, "Whole hand modeling using 8 wearable sensors: Biomechanics for hand pose prediction," pp. 21–28, 2013.

[57] X. Yun, E. R. Bachmann, and R. B. McGhee, "A simplified quaternion-based algorithm for orientation estimation from earth gravity and magnetic field measurements," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 3, pp. 638–650, 03 2008.

[58] P. Chen, Y. Kuang, J. Li, and N. Donato, "Human motion capture algorithm based on inertial sensors," *Journal of Sensors*, vol. 2016, p. 4343797, 2016. [Online]. Available: https://doi.org/10.1155/2016/4343797

[59] T. F. R. Catarino, "Development of hand-tracker: Wireless solutation based on inertial sensors," 2016. [Online]. Available: http://hdl.handle.net/10316/81666

[60] G. Welch and E. Foxlin, "Motion tracking: No silver bullet, but a respectable arsenal," *Computer Graphics and Applications, IEEE*, vol. 22, pp. 24 – 38, 12 2002.

[61] B. Kenwright, "A beginners guide to dual-quaternions: What they are, how they work, and how to use them for 3d character hierarchies," pp. 1–10, 2012, 20th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2012, WSCG 2012 ; Conference date: 26-06-2012 Through 28-06-2012.

[62] J. Zeitlhöfler, "Nominal and observation-based attitude realization for precise orbit determination of the jason satellites," 06 2019.

[63] W. Clifford, *Mathematical Papers*. Macmillan and Company, 1882. [Online]. Available: https://books.google.com.mt/books?id=zzNwzgEACAAJ

[64] G. Leclercq, P. Lefèvre, and G. Blohm, "3d kinematics using dual quaternions: theory and applications in neuroscience," *Frontiers in behavioral neuroscience*, vol. 7, no. 23443667, pp. 7–7, 02 2013. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3576712/

[65] A. Musiolik, M. M. Maia, P. T. Silva, and F. O. de Seabra Pereira, "Multibody model of the human hand for the dynamic analysis of a hand rehabilitation device," 2008.

[66] G. ElKoura and K. Singh, "Handrix: animating the human hand," pp. 110–119, 2003.

[67] J. Lee and T. Kunii, "Model-based analysis of hand posture," *IEEE Computer Graphics and Applications*, vol. 15, no. 5, pp. 77–86, 1995.

[68] M. M. Rahman, T. T. Choudhury, S. N. Sidek, and A. Awang, "Mathematical modeling and trajectory planning of hand finger movements," *2014 First Conference on Systems Informatics, Modelling and Simulation*, pp. 43–47, 2014.

[69] A. Z. Suárez, A. Osorio, R. Lopez, S. Salazar, and R. Lozano, "Mathematical model and simulation of finger movement with electromyographic signals," *2016 20th International Conference on System Theory, Control and Computing (ICSTCC)*, pp. 771–775, 2016.

[70] O. A. Van Nierop, A. van der Helm, K. J. Overbeeke, and T. J. P. Djajadiningrat, "A natural human hand model," *The Visual Computer*, vol. 24, no. 1, pp. 31–44, 2008.

[71] K.-S. Lee and M.-C. Jung, "Three-dimensional finger joint angles by hand posture and object properties," *Ergonomics*, vol. 59, no. 7, pp. 890–900, 2016.

[72] V. Gracia-Ibáñez, M. Vergara, and J.-L. Sancho-Bru, "Interdependency of the maximum range of flexion-extension of hand metacarpophalangeal joints," *Computer methods in biomechanics and biomedical engineering*, vol. 19, p. 1800–1807, 2016.

[73] K.-S. Lee and M.-C. Jung, "Flexion and extension angles of resting fingers and wrist," *International journal of occupational safety and ergonomics*, vol. 20, p. 91–101, 2014.

[74] W. S. Yu, H. van Duinen, and S. C. Gandevia, "Limits to the control of the human thumb and fingers in flexion and extension," *Journal of Neurophysiology*, vol. 103, p. 278–289, 2010. [Online]. Available: https://journals.physiology.org/doi/abs/10.1152/jn.00797.2009

[75] J. M. Daniels, E. G. Zook, and J. M. Lynch, "Hand and wrist injuries: Part i. nonemergent evaluation," *American family physician*, vol. 69, p. 1941–1948, 2004.

[76] J. Bach, B. Draslov, and B. Jørgensen, "Positioning, splinting and pressure management of the burned hand: A method," *Scandinavian journal of plastic and reconstructive surgery*, vol. 18, p. 145–147, 1984.

[77] H. Tan, M. Srinivasan, C. Reed, and N. Durlach, "Discrimination and identification of finger joint-angle position using active motion," *TAP*, vol. 4, 07 2007.

[78] K. O. Taams, G. J. Ash, and S. Johannes, "Maintaining the safe position in a palmar splint: The "double-t" plaster splint," *Journal of Hand Surgery*, vol. 21, p. 396–399, 1996.

[79] J. M. Bednar, "The treatment of hand fractures by the application of casts and splints," *Operative Techniques in Orthopaedics*, vol. 7, p. 93–95, 1997.

[80] D. C. Clark, "Common acute hand infections," *American family physician*, vol. 68, p. 2167–2176, 2003.

[81] A. Buryanov and V. Kotiuk, "Proportions of hand segments," *Int. J. Morphol*, pp. 755–758, 2010.

[82] R. Hamilton and R. A. Dunsmuir, "Radiographic assessment of the relative lengths of the bones of the fingers of the human hand," *Journal of Hand Surgery*, vol. 27, no. 6, pp. 546–548, 2002.

[83] STMicroelectonics, "Computing tilt measurement and tilt-compensated ecompass," 2021. [Online]. Available: https://www.st.com/resource/en/design_tip/dt0058-computing-tilt-measurement-and-tiltcompensated-ecompass-stmicroelectronics.pdf

[84] Q. Wang, Y. Li, and X. Niu, "Thermal calibration procedure and thermal characterisation of low-cost inertial measurement units," *Journal of Navigation*, vol. -1, pp. 1–18, 07 2015.

[85] P. McWorter, "9-axis imu lesson 5: Calibrating the bno055 9-axis in," 2019. [Online]. Available: https://www.youtube.com/watch?v=yPfQK75dZbU

[86] S. Althaf and C. W. Babbitt, "Disruption risks to material supply chains in the electronics sector," *Resources, Conservation and Recycling*, vol. 167, p. 105248, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921344920305632

[87] M. Meotto, "Implications of covid-19 on fast-moving consumer goods and electronics supply chains : a systematic review of secondary materials," 2021. [Online]. Available: https://www.politesi.polimi.it/handle/10589/173721

[88] Hirose, "Hirose connector mouser," 2023. [Online]. Available: https://eu.mouser.com/ProductDetail/Hirose-Connector/FH33J-4S-05SH10

[89] Texas-Instruments, "Tca9548a," 2023. [Online]. Available: https://www.ti.com/product/TCA9548A

[90] JST, "Header jst," 2023. [Online]. Available: https://uk.farnell.com/jst-japan-solderless-terminals/bm02b-srss-tb-lf-sn/header-top-entry-2way/dp/1679127

[91] Microchip, "Battery charger," 2023. [Online]. Available: https://www.microchip.com/en-us/product/MCP73833

[92] digikey, "A deep dive into audio jack switches and configurations," 2019. [Online]. Available: https://www.digikey.com/en/articles/a-deep-dive-into-audio-jack-switches-and-configurations

[93] Texas-Instruments, "Tps51103," 2023. [Online]. Available: https://www.ti.com/product/TPS51103

[94] Sparkfun, "Sparkfun icm20948 arduino library," 2019. [Online]. Available: https://github.com/sparkfun/SparkFun_ICM-20948_ArduinoLibrary

[95] W. T. Ang, P. Khosla, and C. Riviere, "Kalman filtering for real-time orientation tracking of handheld microsurgical instrument," *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, pp. 2574–2580 vol.3, 04 2004.

[96] P. Mi, Q. Du, L. Ye, and W. Zou, "An adaptive fast quaternion-based human motion tracking algorithm with inertial/magnetic technology," *IEEE 10th International Conference On Signal Processing Proceedings*, pp. 1252–1258, 2010.

[97] T. Seel and S. Ruppin, "Eliminating the effect of magnetic disturbances on the inclination estimates of inertial sensors," *IFAC-PapersOnLine*, vol. 50, pp. 8798–8803, 07 2017.

[98] K. Wynn, "Python quaternion library," 2016. [Online]. Available: https://pypi.org/project/pyquaternion/

[99] A. Verheye, "Dual quaternion python library," 2019. [Online]. Available: https://pypi.org/project/dual-quaternions/

[100] GlowScript, "Vpython," 2023. [Online]. Available: https://vpython.org/

[101] MIT, "Bleak," 2020. [Online]. Available: https://bleak.readthedocs.io/en/latest/index.html

[102] C. Novo, M. Abser, O. Adegbite, S. Alharbi, B. Atoufi, F. Popa, A. Radmand, R. Razak, and C. Vatsa, "Investigation of the accuracy of the goniometer in measuring the knee angle using the vicon system as reference," 04 2011. [Online]. Available: https://www.researchgate.net/publication/278647068_Investigation_of_the_Accuracy_of_the_Goniometer_in_Measuring_the_Knee_Angle_using_the_Vicon_system_as_reference

[103] uBlox, "Nina-b30 series (open cpu)," 2023. [Online]. Available: https://www.u-blox.com/en/product/nina-b30-series-open-cpu-0