

Test Apparatus for Analysis of Physical Therapy Rehabilitation Algorithms and Motion Sensor Accuracy

ISMAEL SILVA COSTA

julho de 2023

POLITÉCNICO DO PORTO
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Test Apparatus for Analysis of Physical Therapy Rehabilitation Algorithms and Motion Sensor Accuracy

Ismael Silva Costa

Master in Electrical and Computer Engineering
Specialization Area of Automation and Systems



DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA
Instituto Superior de Engenharia do Porto

July, 2023

*This dissertation partially satisfies the requirements of the
Thesis/Dissertation course of the program Master in Electrical and Computer
Engineering, Specialization Area of Automation and Systems.*

Candidate: Ismael Silva Costa, No. 1181174, 1181174@isep.ipp.pt

Scientific Guidance: Cláudio Maia, crr@isep.ipp.pt

Scientific Co-Guidance: Cecília Reis, cmr@isep.ipp.pt

Company: SWORD Health, S.A.

Advisor: Pedro Martins, p.martins@swordhealth.com



DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA
Instituto Superior de Engenharia do Porto
Rua Dr. António Bernardino de Almeida, 431, 4200-072 Porto

July, 2023

Acknowledgements

Obrigado,

A Deus.

Aos meus pais e ao meu irmão, pelo apoio incondicional, a paciência desmesurada, e os concelhos proveitosos.

À Mariana, por me suportar e apoiar no meu pior, superando sempre o meu melhor.

Ao Diogo, João, Fonseca, Stéphane M., Stéphane O., Manuel, e Ulisses, pela camaradagem durante este longo caminho.

Aos meus orientadores, Pedro Martins, Cláudio Maia, e Cecília Reis, pelo acompanhamento durante todo o processo.

À SWORD, por ceder o espaço e recursos necessários à elaboração desta tese.

À equipa de Research & Systems, pelo apoio e conselhos indispensáveis durante este projeto.

Ao Instituto Superior de Engenharia do Porto, por me acolher, guiar, e fazer crescer ao longo destes 5 anos.

A todos,

Obrigado!

Abstract

The use of technology to support traditional physical therapy in current times not only is a complement, but also an integral part of different types of rehabilitation. An example of this fusion between technology and physiotherapy is the Digital Physical Therapist developed by SWORD Health. It is used in physiotherapy sessions to provide immediate feedback to the user about the exercise being executed and improve the results of these sessions. SWORD has quality assurance processes in place for testing its products, but none of them physically stimulate the Motion Tracker (MT) repeatedly and accurately. A process like this eliminates circumstantial errors between multiple tests and allows the identification of systematic errors on the MT's.

To solve this problem, we decided to develop a human arm replica that stimulates the MT's by performing a specific subset of movements. We used two servo motors to replicate the shoulder joint and two more to replicate the elbow joint. These joints are connected using PVC pipes and 3D-printed pieces. The "arm" is attached to a fire extinguisher base. The motors provide feedback signals, which are used to monitor their position. To evaluate the values read by the MT's, we placed two Inertial Measurement Unit (IMU)'s on the apparatus to track its position. The apparatus is controlled using a MicroController Unit (MCU). This MCU has a shell interface that allows control of the apparatus without needing to change the firmware. This shell supports moving the apparatus and reading values from the IMU's, among other possible commands. A user interface that connects to the MT's the user wants to test is also provided, and offers multiple ways of moving the apparatus, such as manually selecting each angle, choosing from a list of presets, or even editing/adding/removing a preset from this list.

To test the behavior of this apparatus, functional tests were performed. Four different scenarios were added to test the MT's: checking if the acceleration module is equal to 1G when the MT's are resting, identifying if the axes of the MT's are correctly aligned, checking if the MT's read the correct acceleration values along all axes, and verifying the quaternions from the MT's. These tests were performed on a set that included functional and non-functional MT's. The apparatus proved to be able to identify the defective MT's in our test group, identifying problems like a wrong acceleration module, axes incorrectly aligned, wrong acceleration values, and wrong quaternions. These results confirmed that the apparatus is capable of being used to automate some of the manual work done by the Quality Assurance team.

Keywords: Robotics, biomechanics simulation, motion tracking, technology and physical therapy, hardware in the loop testing, testing apparatus.

Resumo

O uso de tecnologia como suporte da fisioterapia tradicional hoje é usada não só como um complemento, mas como uma parte integral de diferentes tipos de reabilitação. Um exemplo desta fusão entre tecnologia e fisioterapia é o *Digital Physical Therapist* desenvolvido pela SWORD Health. É usado em sessões de fisioterapia para providenciar feedback imediato ao utilizador sobre o exercício que está a fazer e melhorar os resultados destas sessões. A SWORD possui procedimentos para assegurar a qualidade dos seus produtos, mas nenhum destes procedimentos exercita fisicamente o MT repetidamente e com precisão. Um processo como este elimina erros circunstanciais entre múltiplos testes e permite a identificação de erros sistemáticos dos MTs.

Para resolver este problema, decidimos desenvolver uma réplica do braço humano que exercita os MTs utilizando um conjunto de movimentos. Usámos dois servo motores para replicar o ombro e mais dois motores para o cotovelo. Estas juntas são ligadas por tubos de PVC e peças impressas em 3D. O “braço” é preso a uma base de extintor. Os motores disponibilizam um sinal de *feedback*, que usamos para acompanhar a sua posição. Para verificar os valores lidos pelos MTs, colocámos dois IMUs no aparato para acompanhar a sua posição. O aparato é controlado utilizando um MCU. Este MCU tem uma interface *shell* que permite o controlo do aparato sem necessidade de alterar o *firmware*. Esta *shell* permite mover o aparato e ler os valores dos IMUs, entre outros comandos. Também é disponibilizada uma interface do utilizador que se liga aos MT’s a ser testados e oferece várias formas de mover o aparato, como escrever manualmente cada ângulo, escolher a partir de uma lista de *presets*, ou até editar/adicionar/remover *presets* desta lista.

Para testar o comportamento deste aparato foram efetuados testes funcionais. Foram adicionados quatro cenários de teste: verificar se o módulo de aceleração é igual a 1G quando os MTs estão em repouso, identificar se os eixos dos MTs estão alinhados corretamente, verificar se os MTs lêem os valores corretos de aceleração em todos os eixos, e verificar os quatérnios dos MTs. Estes testes foram feitos num grupo que incluía MTs funcionais e não funcionais. O aparato provou-se capaz de identificar os MTs com defeito neste grupo, ao identificar problemas como um módulo de aceleração errado, eixos alinhados incorretamente, valores de aceleração errados, e quatérnios errados. Estes resultados confirmaram que o aparato pode ser utilizado para automatizar algum do trabalho manual feito pela equipa de verificação de qualidade.

Palavras-Chave: Robótica, simulação de biomecânica, *motion tracking*, tecnologia e fisioterapia, testes *hardware in the loop*, aparato de teste.

Contents

List of Figures	vii
List of Tables	xi
1 Introduction	1
1.1 Motivation and Scope	2
1.2 Dissertation Goals	3
1.3 Timeline	4
1.4 Dissertation Structure	8
2 State of the Art	9
2.1 Biomechanics Simulation	10
2.1.1 Software Simulation	10
2.1.2 Mechanical Simulation	11
Articulated Robots	11
Humanoid Robots	12
Parallel Robots	13
2.2 Robotic Joints	15
2.2.1 Types of Human Joints	15
2.2.2 Robotic Joints	16
2.3 Motion Tracking	17
2.3.1 Sensors	18
2.3.2 Sensor Fusion	20
2.3.3 Data Presentation	22
2.4 Motion Sensor Testing	23
2.5 Non-Commercial Projects	26
3 Design Choices	31
3.1 Hardware	33
3.1.1 Motors	34
3.1.2 Inertial Measurement Unit	35
3.1.3 Communication between Computer and Microcontroller	37
3.1.4 Microcontroller	38
3.1.5 Power Delivery	39

3.1.6	Links, Joints and Base	40
3.2	Updated Architecture	41
3.3	Software	43
3.3.1	Use Cases	44
3.3.2	Control Algorithm	44
3.4	Bill of Materials	45
4	Implementation	47
4.1	Apparatus Hardware	48
4.1.1	Joints	48
4.1.2	Links	49
4.1.3	Base	51
4.2	Control of the Apparatus	56
4.2.1	Motor's Calibration	61
4.3	Position Tracking	64
4.4	Microcontroller Shell	67
4.5	User Interface	69
4.5.1	Manually Selecting the Angles	74
4.5.2	Preset Selection	75
4.5.3	Edit the Presets	76
4.6	Final Remarks	77
5	Tests and Validation	79
5.1	Limitations	79
5.1.1	Lack of Torque in the First Joint	79
5.1.2	Feedback Signal Noise	80
5.1.3	Drift Influencing the Sensor Fusion	82
5.2	Tests Performed	82
5.2.1	Functional Tests	83
5.2.2	Motion Tracker Tests	84
6	Conclusion	89
6.1	Summary of the Work Developed	89
6.2	Future Work	91
6.3	Main Outcomes	92
	References	93
	Appendix A Motor's calibration values	105
A.1	Graphs of Servo Motors	105
A.2	Values Gathered During Motors' Calibration	109
A.3	Motors' Calibration Script	110

List of Figures

1.1	Examples of ADL [Law, 2023].	2
1.2	Physical rehabilitation with the guidance of a robot [Wang et al., 2019].	2
1.3	SWORD Motion Trackers and mobile application [Sword Health, 2023].	3
1.4	First version of the Timeline for the dissertation.	5
1.5	Final version of the Timeline for the dissertation.	7
2.1	Diagram of an articulated robot [Renganathan, 2015].	12
2.2	Pyrène, a humanoid robot developed by PAL-Robotics [Stasse et al., 2017].	13
2.3	Parallel robots' diagrams.	14
2.4	Spherical active joint developed by [Mghames et al., 2019].	16
2.5	ABENICS, an active ball joint mechanism developed by [Abe et al., 2021].	17
2.6	Different types of motion tracking.	19
2.7	MTi-100, one of the IMU's part of the MTx line of sensors from Xsens [Farnell, 2023].	20
2.8	Diagram of a complementary filter [Kumar and Divakar, 2015].	21
2.9	Yaw, pitch and roll [Automatic Addison, 2020].	22
2.10	Diagram of a cartesian robot.	23
2.11	Motion sensor' testing apparatus, developed by L. Yavan and C. Tepe [Makalesi et al., 2021].	24
2.12	Pendulum apparatus for testing IMU [Chaulk, 2011].	25
2.13	Gesture replication in real-time [Mangate et al., 2022].	27
2.14	Apparatus used for mimicking arm and hand movement [Oliveira et al., 2019].	28
2.15	Project developed by R. Yenorkar and U. M. Chaskar [Yenorkar and Chaskar, 2018].	29
3.1	Human arm represented as a set of joints and links.	32
3.2	Functional diagram of this project.	33
3.3	DSS-M15S servo motor, from DFRobot [DFRobot, 2018].	35
3.4	MPU-6050 [HTF, 2023].	36
3.5	LSM6DSO Breakout, from SparkFun [SparkFun, 2023].	37

3.6	NUCLEO-G431RB development board [ST, 2023].	39
3.7	Components used for powering the apparatus.	40
3.8	Power distribution diagram.	40
3.9	Fire extinguisher base used as "torso" of the apparatus [Extintor, 2023].	41
3.10	Updated version of the Functional Diagram.	42
3.11	Electrical diagram of the apparatus.	43
3.12	Use cases of the apparatus.	44
3.13	High-level view of the control algorithm.	45
4.1	Servo brackets [ElectroFun, 2022].	48
4.2	Joints of the Apparatus.	49
4.3	Clamping Mount [ServoCity, 2023].	50
4.4	First version of the piece that attaches the joints of the apparatus to the PVC pipe.	50
4.5	Second and final version of the piece that attaches the joints of the apparatus to the PVC pipe.	51
4.6	Sketch of the apparatus.	51
4.7	Servo attachments for connecting the first joint to the fire extinguisher.	52
4.8	Attachments between the first joint of the apparatus and the fire extinguisher base.	53
4.9	Apparatus hardware assembled.	54
4.10	Shield to be placed on the MCU.	55
4.11	Flowchart of the motor sweep function.	58
4.12	Generic PID controller.	59
4.13	Diagram of the PID controller implementation in firmware.	60
4.14	Flowchart of the process of reading and processing the feedback signals.	61
4.15	Graph that shows the relationship between feedback value and angle of the first servo motor.	63
4.16	ROM of the human elbow [Luttgens and Hamilton, 1997].	64
4.17	Flowchart of the process of reading the IMU.	65
4.18	Flowchart of the procedure of processing the data from the IMU. . .	66
4.19	Flowchart of the firmware after implementing the shell.	69
4.20	nRF52840 USB dongle from Nordic [Nordic, 2023].	71
4.21	Reading quaternions from the MT.	71
4.22	Contents of the "presets.json" file.	72
4.24	User Interface prompt for selecting the MT's.	72
4.23	Starting sequence of the user interface.	73
4.25	User Interface prompt for selecting how the user wants to interact with the apparatus.	74
4.26	User Interface prompt for selecting the angles for each motor.	74

4.27	Quaternions presented in the user interface.	75
4.28	User Interface prompt for selecting one of the available presets. . . .	75
4.29	User Interface prompt for editing the available presets.	76
4.30	Final prototype of the apparatus.	78
5.1	Range of the first motor.	80
5.2	Updated control algorithm, now with the counter.	81
5.3	Axes of the MT's.	86
A.1	Relationship between feedback value and angle of the first servo motor.	106
A.2	Relationship between feedback value and angle of the second servo motor.	107
A.3	Relationship between feedback value and angle of the third servo motor.	108
A.4	Relationship between feedback value and angle of the fourth servo motor.	109

List of Tables

2.1	Advantages and disadvantages of different types of apparatus for physical biomechanical simulation.	15
3.1	Key characteristics of the DSS-M15S servo motor.	35
3.2	Ranges of the sensors inside the MPU-6050.	36
3.3	Ranges of the sensors inside the LSM6DSO.	36
3.4	Bill of Materials.	46
4.1	UART parameters.	57
4.2	MT's and IMU's parameters.	65
4.3	Registers of the IMU that contain the values of acceleration and rotation.	66
5.1	MCU shell functional tests.	83
5.2	MCU functional tests.	83
5.3	User interface functional tests.	84
5.4	Ranges of the motors.	84
5.5	Results of the four tests performed on the test group.	87
A.1	Raw values read by the motors's calibration script.	110

Chapter 1

Introduction

Physical therapy nowadays is deeply technological, but this has not always been the case. In the midst of the 20th century, various institutions began to standardize methods used in rehabilitation fields like occupational therapy, physiatry and physical therapy [Williamson, 2021]. With this standardization, there was a rise in the use of everyday activities as a rehabilitation technique, now known as Activities of Daily Living (ADL) [The Staff of the Benjamin Rose Hospital, 1959]. These techniques focus on rehabilitation using tasks considered “normal” in our daily lives, being described by Howard A. Rusk as tasks “the average healthy person does without giving them any thought. Things like brushing your teeth, combing your hair, going to the bathroom, putting on your shirt, or tying your shoes.” [Rusk, 1972]. Some examples of ADL are presented in Figure 1.1. An interesting fact about ADL is the noticeable parallel between them and the technology of their times. For example, at the New York Institute, the ADL test during the 1940s and 1950s included winding a wristwatch and operating a telephone and a typewriter [Williamson, 2021]. However, factors such as technological and sociological evolution have molded the methods and resulted in therapists incorporating new tools like fitness bands and other technologies into ADL. Thus, using technology as an aid to traditional physical therapy is not a new concept.

Nowadays, technology is used not only as a complement to ADL but as an integral component in many types of rehabilitation. There are all sorts of advancements in this field, such as the use of virtual reality therapy as a complement to traditional



Figure 1.1: Examples of ADL [Law, 2023].

physical therapy [Ma et al., 2019], and robots that assist patients during physical rehabilitation by analyzing motion and providing feedback [Wang et al., 2019] (Figure 1.2).

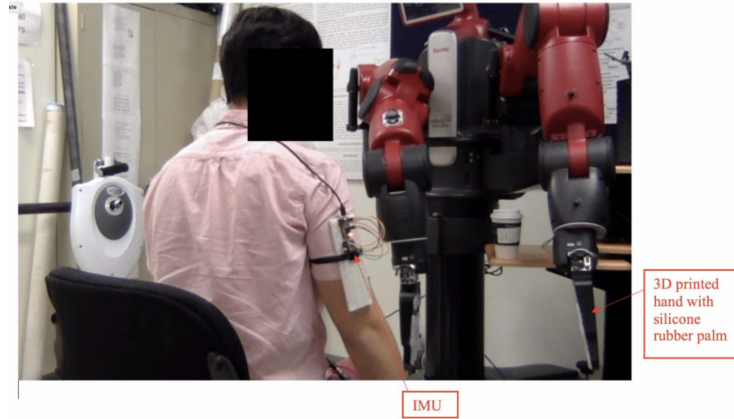


Figure 1.2: Physical rehabilitation with the guidance of a robot [Wang et al., 2019].

These advancements bring great advantages to modern rehabilitation, such as improved measurements [Levin et al., 2015], but also present new challenges in both adoption and development. These challenges are discussed in Section 1.1.

1.1 Motivation and Scope

SWORD Health (referred to as “SWORD” in this dissertation) develops a Motion Tracker (MT) used in physical therapy sessions, presented in Figure 1.3. SWORD develops not only the MT’s but also the algorithms to access the data captured by the MT’s and to infer whether patients are correctly doing a given set of exercises presented to them.

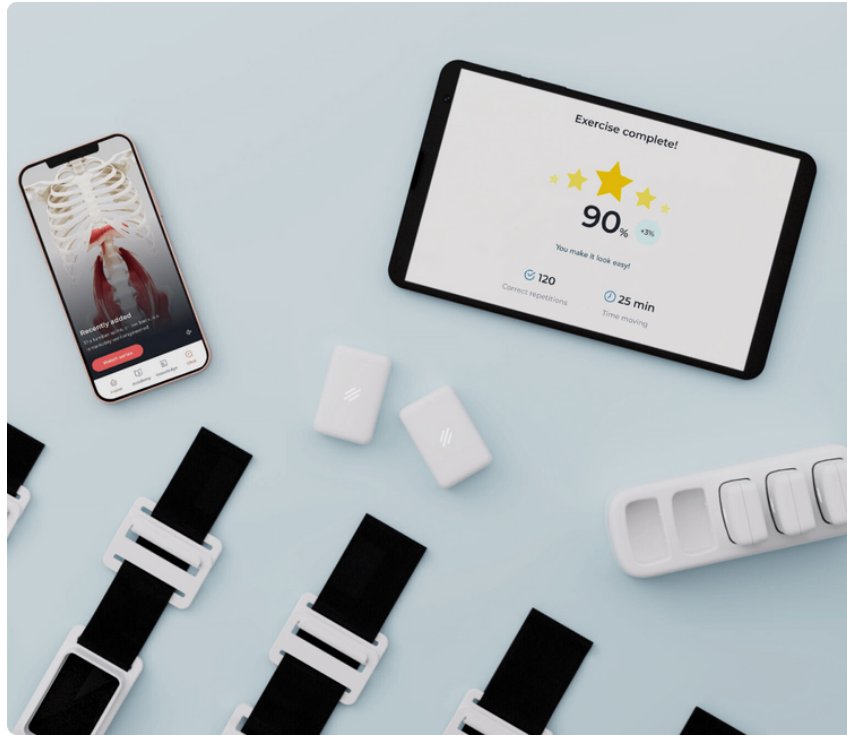


Figure 1.3: SWORD Motion Trackers and mobile application [Sword Health, 2023].

These MT's are placed in key spots on the body of the user to track movements during physical therapy sessions to obtain accurate feedback. The end goal is to continuously adjust the therapy to the evolution of the user in the rehabilitation recovery process. SWORD already has Quality Assurance (QA) procedures in place for executing various tests with MT's, but none of them can replicate human movements repeatedly with precision and with minimal human intervention. A solution with these characteristics would save SWORD time and money in testing procedures, by allowing a single operator to test various movements and verify the behavior of the MT's quickly and reliably. Developing a solution to this problem is the main goal of this dissertation.

1.2 Dissertation Goals

This dissertation proposes the creation of a Proof of Concept (POC) testing apparatus that can replicate the movements of the human arm as quickly and faithfully as possible. The decision to replicate only the human arm was made to help reduce the complexity of the final apparatus and its costs.

Besides replicating a subset of movements, the proposed solution must be able to read and process the data gathered by the MT's placed on the apparatus and

present this data and its accuracy in a simple but useful way. The apparatus should have the following characteristics:

- Repeatability and accuracy: the apparatus should be able to replicate any movement the operator desires (given the hardware constraints), as many times as needed, in a precise and accurate manner;
- Durability: as the final solution is to be used for testing procedures, the apparatus should be sturdy enough to repeat these tests multiple times;
- User-friendliness: the interface used for receiving the orders and outputting the results should be easy to interact with while providing various ways of interacting with the apparatus (e.g. list of preconfigured movements, multiple output formats);
- Cost: while not a decisive factor, the cost of the components should be taken into account when selecting them, to keep the cost of the solution as minimal as possible.

By the end of this dissertation, this testing apparatus should be fully functional and ready to be used as a complement to the development and testing procedures of the MT's developed by SWORD.

1.3 Timeline

The first version of the timeline with the scheduled activities for this dissertation is presented in Figure 1.4:

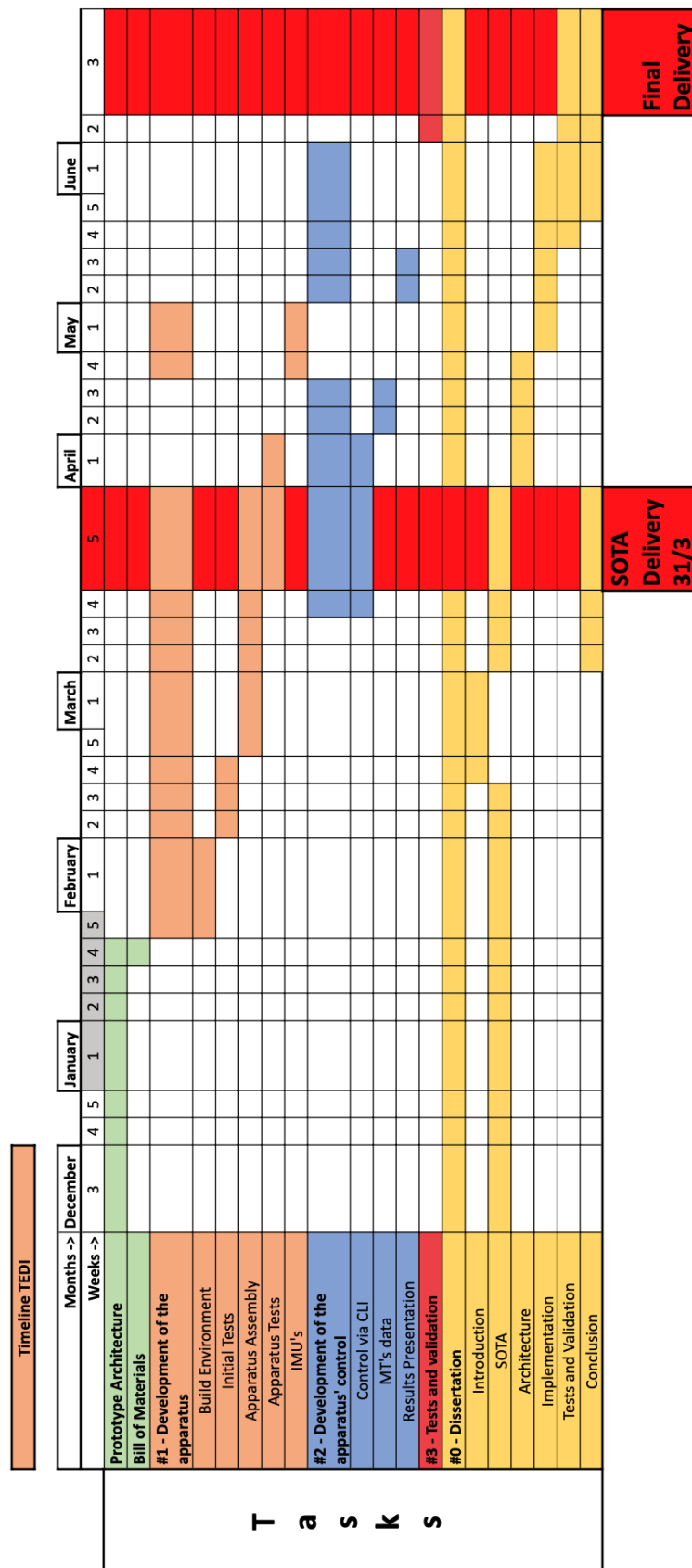
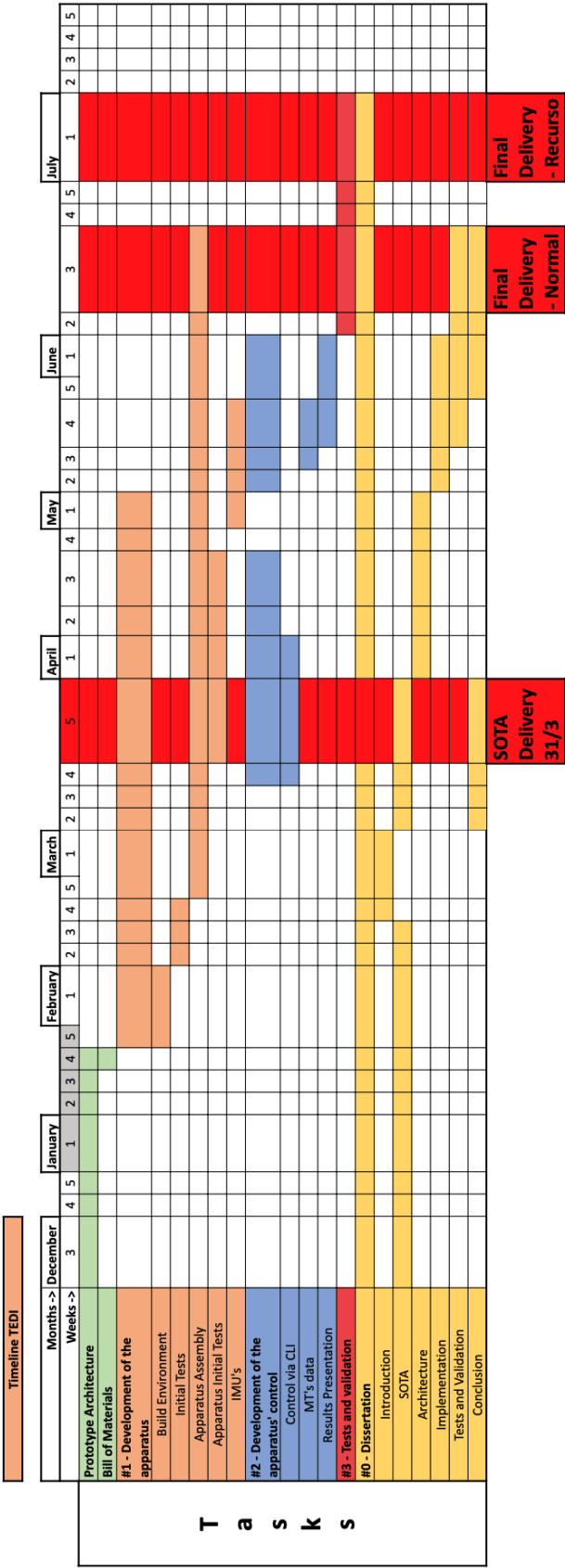


Figure 1.4: First version of the Timeline for the dissertation.

The first step was defining the problem at hand and the main characteristics that the solution should have. With this in mind, the architecture of the prototype and the respective Bill of Materials (BOM) were defined.

The step defined as “#1 - Development of the apparatus” in Figure 1.4 consists of building the apparatus and testing its behavior. The following step (“#2 - Development of the apparatus control”) entails the design of the control of the apparatus, and the display of the data gathered by the MT’s. Finally, the step called “#3 - Tests and validation” comprises all the final tests to be done to verify the behavior of the final solution. During all these steps, the whole process was documented and organized to be presented in this final document.

Despite having organized the work and allocated additional time to each task, there were some setbacks (e.g. lack of available parts at certain times) that forced us to restructure the timeline. Figure 1.5 shows the final version.



The main setback was during the assembly of the apparatus. Various iterations were needed to find the optimal solution, which implied dedicating more time to this process and having to delay the delivery of the final document.

1.4 Dissertation Structure

This dissertation is organized into the following sections:

- **Introduction:** The current chapter. Presents the scope and motivation of this dissertation as well as its main goals and the document organization;
- **State of the Art:** The research done by the authors about biomechanics simulation, robotic joints, motion tracking, motion sensor testing and non-commercial projects that fall within the scope of this dissertation;
- **Design Choices:** This Chapter presents the main elements of this project, the selected hardware, and the software developed. All the choices made during the idealization process are also presented, as well as the motives that led to them.
- **Implementation:** Guides the reader through the building process of the apparatus, using the hardware discussed in the previous Chapter. Also explains the development of the software needed, and how a user can interact with the apparatus.
- **Tests and Results:** presents the results of the functional tests done on the apparatus. Also contains the description of three testing scenarios implemented, as well as the results of these tests.
- **Conclusion:** The last chapter presents a summary of the work presented in this dissertation and the main outcomes. It also presents possible enhancements to be implemented in future versions of the prototype.

Chapter 2

State of the Art

The simulation of human movements is a wide-ranging topic that has multiple applications in many fields. The possibility of simulation without limitations regarding human safety contributes to the growth in the number of solutions presented by industry and academia. Because of the number of possible applications, there is an interest in developing and optimizing different simulation methods that contribute to the testing of possible solutions and allow, at the same time, the analysis of several sets of data produced within the context of simulation. In this chapter, we delve into the state of the art of the existing solutions and technologies that are used within the context of the simulation of human movements. In particular, this chapter covers the following:

- Biomechanics Simulation: definition of the concept, different classifications and multiple examples;
- Robotic Joints: types of human joints, and a selection of projects that focus on replicating them;
- Motion Tracking: definition of the concept, sensors used in this field, and the methods used for improving the accuracy of these sensors;
- Motion Sensor Testing: discussion about different methods for testing motion sensors, with a comparison between them in the end;
- Non-Comercial Projects: analysis of several projects with a focus on simulating human movement.

By the end of this chapter, the reader should have an understanding of the current state of biomechanics simulation, robotic joints, motion tracking, motion sensor testing, and projects in this field.

2.1 Biomechanics Simulation

According to [Hatze, 1974], biomechanics is defined as “the study of the structure and function of biological systems by means of the methods of mechanics”. One of the most useful applications of this study is biomechanics simulation: the simulation of human movements for studying the reaction of the human body to different mechanical stimuli, and the effects of these movements on the environment around it.

There are many potential applications for the simulation and analysis of human movement, and it has even become essential in fields like:

- Sports: biomechanical analysis and consequent improvement of athletes’ performance while decreasing the risk of injury. One example is the simulation of one-handed backhand strokes in tennis to investigate loading at the elbow [King, 2011];
- Health: simulation of Anterior Cruciate Ligament (ACL) damage in different movements [Fonseca, 2020] and knee biomechanics simulation for improvements of the surgery process [Erdemir, 2014];
- Automotive Industry: simulation of car crashes and consequent biomechanical reactions [Johannsen et al., 2015], or the enhancement of ergonomics in various sections of a car, such as the seats [Amortila et al., 2021];
- Ergonomic Studies: analysis of ergonomics for workspace design improvements [Golabchi et al., 2015].

This wide array of applications boosts the development of different methods for biomechanical simulation. As the work developed in this dissertation overlaps with many of the existing works found in the literature, it is important to understand their contribution and to analyze whether they can be used in the context of the current work. Therefore, within this section, works on the topics of software simulation and mechanical simulation are presented.

2.1.1 Software Simulation

Software biomechanics simulation aims to reproduce human movements and their consequences in the environment around them, using software to simulate the human body and its behavior in various situations [Machado, 2013]. Advantages over

physical tests come from the fact that the environment is one where every parameter is controlled, and there is no need for any real person or hardware. These characteristics allow for flexibility in the testing process and a virtually unlimited number of trials, either displaced into multiple sessions or performed over a short period (i.e. a Monte Carlo simulation [Santos and Valero-Cuevas, 2004]). As so, it is used the most when the costs of physically implementing the testing scenario would be excessive or human safety would be at risk [Nunes et al., 2015].

The software simulation of human movements and the interactions with the environment around them could be useful for prototyping and visually representing the desired movements of the prototype presented in this dissertation. While it would be an interesting topic to explore, this dissertation focuses on physically replicating a subset of movements performed by a person in a physical therapy session (as seen in Section 1.1). As so, the use of software simulation is beyond the scope of this dissertation.

2.1.2 Mechanical Simulation

As an alternative to software simulation, mechanical biomechanics simulation employs physical devices and mechanisms to accurately mimic subsets of human movements physically. This method is used often in later stages of development for testing prototypes (e.g. improvements in car safety) in a setting that is closer to reality. When the desired application is to test the performance of a sensor that measures physical characteristics such as temperature, sound, humidity, and movement (among others), this method is also ideal. With various characteristics and sensor attributes to be measured, as well as movements that need to be replicated, the mechanism developed for each application can greatly differ from one to the other. As such, there is a vast array of apparatuses developed with different characteristics, and studying these is mandatory to reach the solution that best accomplishes the problem presented in this dissertation. Some of the types of apparatuses that can be used to simulate biomechanics are presented in the following subsections.

Articulated Robots

Most commonly called “robotic arms”, these robots are mostly composed of rotational joints and, because they resemble the human arm, this configuration is also called “anthropomorphic configuration”. A diagram of this robot is presented in Figure 2.1.

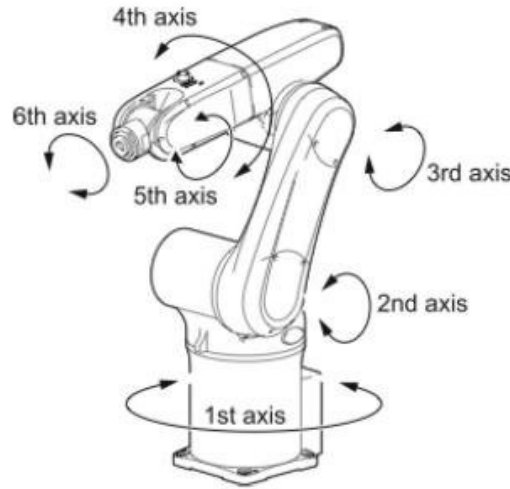


Figure 2.1: Diagram of an articulated robot [Renganathan, 2015].

These robots are typically used in industrial settings (e.g. object manipulation, welding, packaging), but have been used in other applications like the manipulation of boxes in pharmacies [ABB, 2020]. As mentioned before, these robots resemble a human arm so this characteristic would be advantageous for easily performing the subset of movements desired in this dissertation. Another benefit comes up when this type of robot has a six-axes configuration because it allows the end effector to be positioned in any orientation in a three-dimensional space (within the respective workspace). These characteristics make it one of the most used types of robot (particularly in industrial settings, having reached about 65% of the global industrial robot shipments in 2017 [Teulieres et al., 2019]) and could be a solution to the problem presented in this dissertation, but there are some caveats particular to the desired application. While the apparatus physically resembles a human arm, the set of movements it can perform is greater than the subset of movements desired in this particular dissertation. There has been some progress in trying to mimic the behavior of humans at a kinetic level, but there still are some challenges associated with this. The main one is finding a model that accurately replicates the smoothness and gracefulness of human motion [Gan and Dai, 2012]. Therefore, the complexity and cost associated with this solution increase greatly while not offering additional value in this scope.

Humanoid Robots

These are robots that are not only designed with a human appearance but that can sometimes also communicate with humans and take orders from its user. Used in healthcare [Yamada et al., 2019], education [Chin et al., 2011] [Chen and Chang, 2008], and entertainment [Fudal et al., 2013] to mollify people that interact with these robots [Choudhury et al., 2018], the human resemblance of its configuration

also allows the replication of human movements. An example of these robots is the Pyrène, developed as part of the TALOS series built by PAL-Robotics (seen in Figure 2.2).

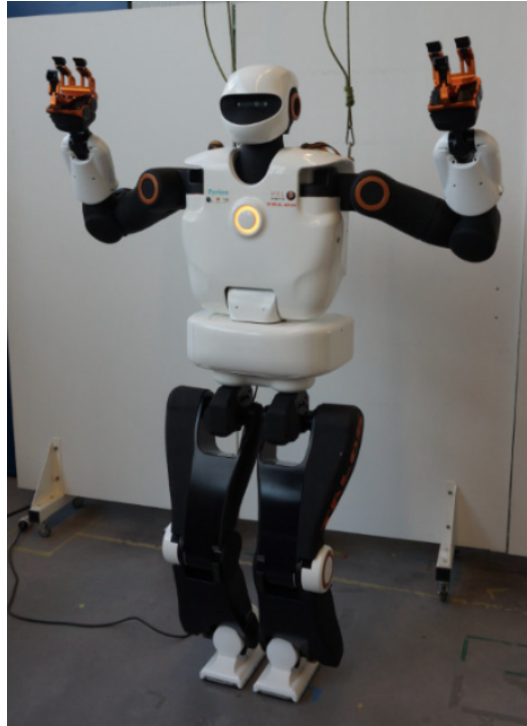
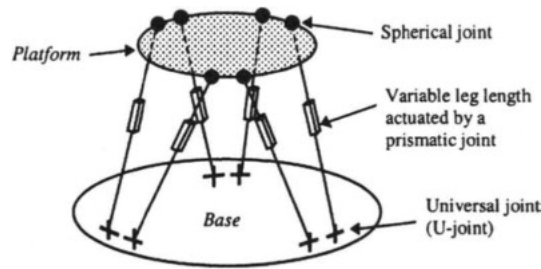


Figure 2.2: Pyrène, a humanoid robot developed by PAL-Robotics [Stasse et al., 2017].

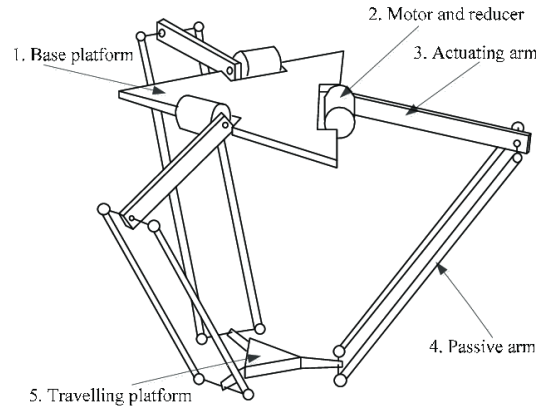
Similarly to articulated robots, their human resemblance is useful in this application because it allows for the physical simulation of the desired subset of movements. Building on top of this, this type of robot not only resembles a human, but its movements are also human-like. While an articulated robot might move from one point to the other using angles not possible for the human body (in this case specifically, an arm), a humanoid robot will only use movements similar to those found in biomechanics [Kapse and Kapse, 2013]. These are the type of movements that the developed solution should replicate (as per the requirements in Section 1.1), so this kind of apparatus has advantages for this project over the articulated robot. While these characteristics could be useful for this type of application though, the complexity of implementing a whole robot and the associated costs do not bring more value to this application. Therefore, this type of robot shown in Figure 2.2 will not be implemented as the testing apparatus, but the ability to perform human-like movements is useful and desired.

Parallel Robots

These robots can be defined as “a mobile platform connected to a fixed base by a set of identical parallel kinematic chains” [Khalil and Dombre, 2002]. These have mainly been used as motion simulators (e.g flight simulators, earthquake simulators such as the one present at the University of Minnesota [UMN, 2022]) by using a platform attached to multiple chains as seen in Figure 2.3a, but lately has seen a growth in industrial settings for performing mostly pick and place tasks using a gripper as the end effector instead of a platform, in what is usually called the “delta configuration” (represented in Figure 2.3b).



(a) Parallel robot with a platform as the end effector [Khalil and Dombre, 2002].



(b) Parallel robot in a delta configuration [Lu and Liu, 2016].

Figure 2.3: Parallel robots' diagrams.

For simulating human movement, there have been some implementations using a configuration similar to the one in Figure 2.3a. One of these implementations simulates heart muscle movement using such apparatus [Imbrie-Moore et al., 2020]. This type of device is ideal in the context of this dissertation because it can rotate in three axes (*yaw*, *pitch*, and *roll*) while not needing a big workspace.

These are some of the apparatuses used when physically simulating biomechanics, and understanding the main advantages and disadvantages of each is essential for developing the optimal solution to the problem presented in this dissertation. In the

context of this dissertation, the options presented here have the following advantages and disadvantages (Table 2.1):

Table 2.1: Advantages and disadvantages of different types of apparatus for physical biomechanical simulation.

	Advantages	Disadvantages
Articulated Robots	- Movement along six axes (x, y, z, yaw, $pitch$ and $roll$) - Resembles the human arm	- Movements can be dissimilar to those of the human arm - Complexity beyond the scope of this dissertation - Financial costs
Humanoid Robots	- Can be configured to move along six axes (same as the articulated robots) - Can resemble the human arm - Has human-like movement	- Complexity of a full body humanoid robot is beyond the scope of this dissertation - Financial costs
Parallel Robots	- Movement along three axes (yaw, $pitch$ and $roll$) - Small workspace needed	- Lack of linear movement

Researching these apparatuses provides useful information, but it does not remove the need to research the state of the art of robotic joints. Because one of the goals of this dissertation is replicating the movement of the human arm (Section 1.2), the study of robotic joints helps understand the challenges of simulating human movement.

2.2 Robotic Joints

The apparatuses shown in the last section (Subsection 2.1.2) can be used for performing human-like movements in appropriate settings, but because most are not geared towards human movement replication, the complexity of the components can be unnecessary for the scope of this work. Thus, because the main focus of this dissertation is to replicate a subset of movements performed in a physical therapy session, the developed apparatus should focus on replicating human movements as closely and faithfully as possible.

As discussed in Section 1.2, the movements to replicate are the movements of the human arm. As such, the process of choosing the components that simulate the shoulder/elbow joint should start with an analysis of the types of joints present in the human body.

2.2.1 Types of Human Joints

To faithfully recreate human movement, the joints used for the shoulder and elbow of the apparatus should mimic characteristics of their human counterpart, such as the stiffness or the Range of Motion (ROM), which have been researched before in [Ramanathan et al., 2000], [Romilly et al., 1994], [Magermans et al., 2005], and [Rosen et al., 2005]. As the main focus of this dissertation is analyzing the data gathered by MT's used in physical therapy sessions (as per Section 1.1), the ROM is particularly important because the final solution should have the same ROM as the human arm.

According to [Tortora et al., 2018], there are three types of joints present in the human body (all examples found in [Villanueva, 2016]):

- Fibrous joints, which are immovable (e.g. bones joined by strong fibrous tissue, such as cranial sutures between the bones of the skull);
- Cartilaginous joints, semi-movable (e.g. intervertebral discs);
- Synovial joints, that can move freely (e.g. shoulder and elbow).

From an engineering point of view, synovial joints can be approximated as spherical joints (according to [Wilhelms and Gelder, 2001]). An interesting remark is the fact that this type of joint is rarely found in current robotic and prosthetic designs [Mghames et al., 2019], (which could be because most industrial robots used in today's industry have been designed for specific applications, mainly material handling [Weyrich and Abdullah, 2013]), but various projects focus on replicating this type of joint. A selection is presented in the next subsection.

2.2.2 Robotic Joints

There have been various efforts to develop solutions that can faithfully simulate the behavior of human joints. Whether developing better prosthetic arms or dexterous robots, the answers to these problems vary in their approach. For example, the solution developed by [Lemerle et al., 2019] focuses on the variation in stiffness of a joint, for better replicating the ones found in the human body. As stated before, one of the most important factors in this project is the ROM. Because of this, the research done for this dissertation focused on projects inside this scope, such as the joint developed by [Mghames et al., 2019] (represented in Figure 2.4).



Figure 2.4: Spherical active joint developed by [Mghames et al., 2019].

The proposed prototype is a spherical joint that has almost-complete coverage of the workspace of a human shoulder, limited only by kinematic singularities [Mghames et al., 2019]. While this specific prototype was developed to mimic a shoulder, this type of joint could also be used to simulate other synovial joints such as the elbow.

Another example of a project that focuses on the ROM is the prototype developed by K. Abe et al., yet another advancement in the field of robotic joints. The developed prototype (represented in Figure 2.5) is an active ball joint mechanism that was inspired by the interactions between cross-spherical and monopole gears [Abe et al., 2021]. This type of joint could also be used to replicate the synovial joints presented previously, specifically the shoulder, as it provides three Degrees of Freedom (DoF).

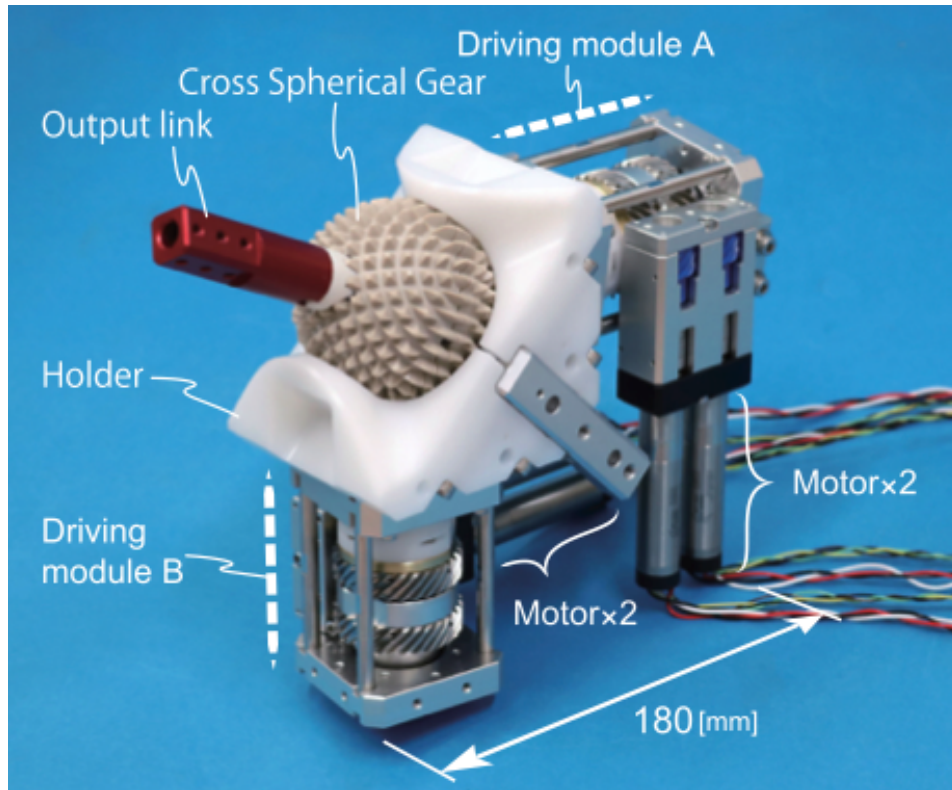


Figure 2.5: ABENICS, an active ball joint mechanism developed by [Abe et al., 2021].

This research in both biomechanics simulation and robotic joints (replications of human counterparts, specifically) is only part of the problem. When considering the focus of this dissertation (as presented in Section 1.1), the simulation of the movement of a human arm is important only if the data gathered by the motion trackers is useful. As so, the process of testing these and the methods employed today are of great interest in this scope.

2.3 Motion Tracking

According to Nilsson, motion tracking “describes the process of recording movement of objects” [Nilsson, 2012]. Motion tracking is also denominated as “motion capture” by most sources, despite the distinction by some authors (e.g. in filmmaking, “motion tracking” is associated with match moving [Rapp, 2014]). This process is used in applications like medical, military and entertainment, among others [VICON, 2023] [LEAP, 2023] [OptiTrack, 2023]. As explained by [Moeslund et al., 2006], applications of motion tracking can be classified in one of three ways:

- Surveillance: These are the solutions to problems related to monitoring locations with a large number of people, like airports and subways. Some examples

of the applications of these solutions are people counting, crowd flux, and congestion analysis. More recently, applications consist of queue and shopping behavior analysis, for example [Moeslund et al., 2006];

- Control: The data gathered is used to control something. Generally, these solutions are used as Human-Computer Interfaces (e.g. virtual reality headsets);
- Analysis: Applications like the diagnostics of medical patients or the analysis of athletes' performances (some examples provided in Section 2.1).

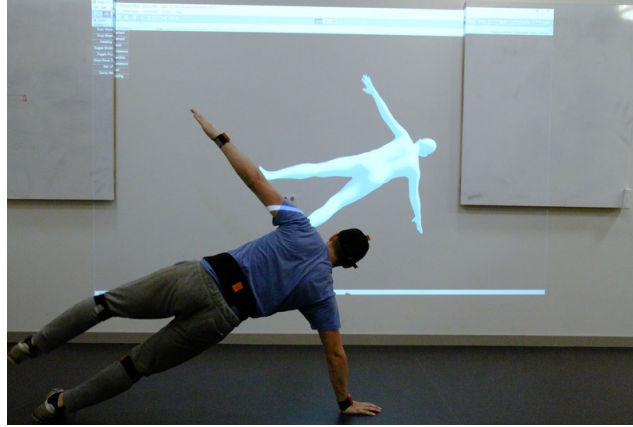
This wide array of applications implies different approaches, and the number of these approaches has grown over time [Moeslund et al., 2006]. This section aims to provide the reader with an introduction to the topic of motion tracking, the technology used, and the methods used for acquiring and processing the data.

2.3.1 Sensors

Motion sensors are an important component of many electronic devices today, as they allow the detection of movement and the device's response to it. These are typically used in several industry domains for tracking purposes, for instance in manufacturing, security (e.g. a space security system [Darmawan, 2019]), health industry [Bento, 2012], and fitness (e.g. an intelligent dumbbell designed for fitness activity monitoring [Liu et al., 2019]) domains. There are already examples of using Inertial Measurement Unit (IMU)'s for measuring human motion, such as [Bakhshi et al., 2011] using IMU's placed on the joints of patients to determine the angles of said joints during activities, or [Abhayasinghe and Murray, 2014] placing these sensors on the thigh for recognizing the activity of pedestrians (such as walking or sitting) in an indoor setting.

These sensors are used for tracking the motion of volumes. There are two main types of motion tracking:

- Inertial motion tracking: Based on the principle of inertia. It requires a sensor to be placed on the point to track, which will then locate the body in three-dimensional space. An example of the setup used for this type of tracking is presented in Figure 2.6a;
- Optical motion tracking: Requires cameras placed around the body to be tracked, and reflective markers placed on said body. Infrared light is then emitted on the body, and the cameras pick up the reflections from the markers [Lindequist and Lönnblom, 2004]. An example is presented in Figure 2.6b.



(a) Inertial motion tracking [Perceiving Systems, 2023].



(b) Motion capture setup [Vicon, 2023].

Figure 2.6: Different types of motion tracking.

Some authors organize these methods in different categories [Karimi, 2016], and others even write about other atypical ways of motion tracking (e.g. acoustical and potentiometer-based systems [Nogueira, 2011]), but the two types presented in this dissertation are the ones found in most literature.

Choosing which type of motion tracking to use is dependent on the characteristics of the application. For example, optical motion tracking is used in computer games, virtual clothing applications and virtual reality [Garcia-Garcia et al., 2020], while inertial motion tracking is seen on most mobile and wearable devices [Song et al., 2021]. This section will focus only on inertial motion tracking, as this is the type of tracking used by the SWORD MT's.

According to [Song et al., 2021], “the goal of motion tracking is to use the sensing information provided by these sensors to derive three-dimensional motion”. As discussed before, inertial motion tracking uses sensors based on the principle of inertia to determine this motion. The most used sensors are the accelerometer (which measures the acceleration of an object due to its motion in space) and the gyroscope (which measures angular velocity) [Dadafshar, 2014]. Since both sensors suffer from time-varying biases and noise, using only these sensors leads to a drift

in the estimation [Filippeschi et al., 2017]. As a way of reducing this drift, a third sensor called the magnetometer can be added (e.g. [Gallagher et al., 2004] [Gebre-Egziabher et al., 2000]). This sensor measures magnetic fields [Macintyre, 2014].

Pieces of equipment that combine these multiple sensors are most commonly called IMU's [Ribeiro and Santos, 2017], although some authors have used different designations when magnetometers are used, such as Magnetic/Inertial Measurement Unit (MIMU) sensor [Palermo et al., 2014] or even Magnetic Angular Rate and Gravity (MARG) sensor [Picerno et al., 2008]. An example of an IMU is the Xsens MTx line of sensors (Figure 2.7), which according to [Bai et al., 2011], are one of the most used IMU's for biomechanical measurements [Bai et al., 2011].



Figure 2.7: MTi-100, one of the IMU's part of the MTx line of sensors from Xsens [Farnell, 2023].

This IMU combines the data from an accelerometer, a gyroscope, a magnetometer and a barometer for providing accurate measurements of the 3D rate of turn, linear acceleration, and magnetic field [Movella, 2023].

As seen before, accelerometers, gyroscopes and magnetometers are used together to reduce the gap between the measurements and the real values. But this poses a new question: *how is the data from different sensors merged to produce one useful result?*. This topic is the subject discussed in the next Subsection (2.3.2).

2.3.2 Sensor Fusion

Terms like “data fusion”, “sensor fusion”, “information fusion” and “multi-sensor data fusion” are widely used in technical literature to refer to different methods, systems and technologies that use data derived from multiple sources [Elmenreich, 2002]. There have been several attempts at categorizing these different terms: in [Wald, 1998], Wald proposes the use of the term “data fusion” as the overall term

for fusion. Despite this effort, some authors have used this same term to specifically denote the fusion of raw data [Dasarathy, 1997]. To avoid this confusion, Dasarathy proposed the term “information fusion” as the term for any kind of fusion data [Dasarathy, 2001]. This makes it an all-encompassing term, and by defining a subset of it, “sensor fusion” is introduced as “the combining of sensory data or data derived from sensory data such that the resulting information is in some sense better than would be possible when these sources were used individually” [Elmenreich, 2002].

According to [Elmenreich, 2002], a physical sensor measurement can suffer the following problems (among others):

- Limited spatial coverage: An individual sensor only covers a restricted region. For example, the reading from a thermometer only estimates the temperature near the thermometer, which may differ from the average temperature of the whole body;
- Limited temporal coverage: The setup time to perform and transmit a measurement limits the maximum frequency of measurements;
- Imprecision: Measurements from individual sensors are limited to the precision of the sensing element.

In the same way that the human brain combines multiple sensory inputs, sensor fusion aims to overcome the downfalls of using a single sensor. As concluded by [Karimi, 2022], the fused sensor information is more accurate and reliable than data from an individual sensor, as it can provide benefits like extended spatial and temporal coverage, reduced ambiguity and uncertainty, and improved resolution.

As seen in Subsection 2.3.1, accelerometers and gyroscopes suffer from particular errors and noise. The most common is the gyroscope drift (where the output value deviates more and more from the real value over time) and the effect of orientation in the readings of the accelerometer (due to gravity) [Pupo, 2016]. The main motive for merging their data is exploiting the benefits of each (long-term compensation by the accelerometer, and short-term measurement by the gyroscope) and reducing the impact of their downfalls, which is the basic idea behind a complementary filter (Figure 2.8).

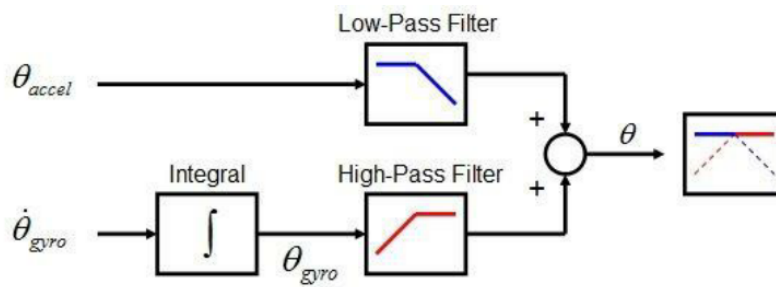


Figure 2.8: Diagram of a complementary filter [Kumar and Divakar, 2015].

A complementary filter is a method that uses an average of the measurements of the accelerometer and gyroscope. The accelerometer provides the orientation when the equipment is resting, and the gyroscope provides the angular rate of change. As proven in [Tseng et al., 2011], accelerometers typically work well in low-frequency motions and diverge in high frequency, while the gyroscope behaves the opposite way. This is the reason behind the filters seen in Figure 2.8: a low-pass filter is used for the accelerometer data to remove high-frequency noise, and a high-pass filter removes the drift from the gyroscope [Page et al., 2014].

Another method to fuse sensor data is the Kalman filter. Besides fusing the measurements from the sensors, this filter estimates the next measurement based on previous values [Islam et al., 2017]. By using this estimation, the error in the output is further reduced [Ribeiro, 2000].

The focus of this dissertation is to simulate human movement and read the data gathered by SWORD's MT's. As so, the mathematics behind these filters will not be discussed, but can be found in [Tseng et al., 2011], [Islam et al., 2017], [Page et al., 2014], and [Kumar and Divakar, 2015].

2.3.3 Data Presentation

There are various ways to present the orientation of a body in 3D space. This subsection presents the most common ones, as an introduction to familiarize the reader with the terms used in this topic. The discussed ways of representing orientation data are Euler angles, quaternions, and rotation matrices.

Euler angles represent the orientation in three rotations around three axes. These three angles are commonly called *yaw*, *pitch* and *roll* (Figure 2.9), and are used to describe the rotation of a body in a fixed coordinate system [Eberly, 1999].

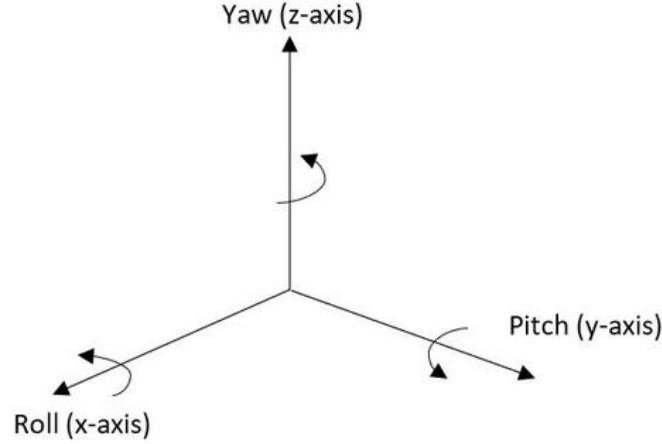


Figure 2.9: Yaw, pitch and roll [Automatic Addison, 2020].

Quaternions represent the orientation of a body as a four-dimensional vector: $q = (w, x, y, z)$, where w is called the “scalar part” and x, y and z are called the “vector part”. The vector part represents the axis of rotation, while the scalar part represents the amount of rotation around that axis. In other words, this quaternion “ q ” represents a rotation of angle θ around the axis defined by vector $v = (x, y, z)$, with $\theta = 2 \times \cos^{-1}(w)$ [Huerta, 2011].

A rotation matrix is a three-by-three matrix (in the case of a rotation in 3D space) that describes the rotation of a body in three axes. For example, the rotation matrix for a rotation of angle θ around the x axis is the following (Equation 2.1):

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \quad (2.1)$$

To use a rotation matrix, we multiply the matrix by a vector that represents the coordinates of a point in 3D space. The result of this multiplication is a vector that represents the coordinates of the point after rotation. The rotation matrices around x, y and z can also be combined to represent rotations around other axes [Wolfram, 2023].

2.4 Motion Sensor Testing

To ensure that a motion sensor is working properly, it is important to verify its behavior by testing it. One of the needed steps for testing is the stimulation of the sensor. This can be done using different solutions:

- Cartesian Robots: These robots are characterized by the use of only prismatic joints (representation in Figure 2.10).

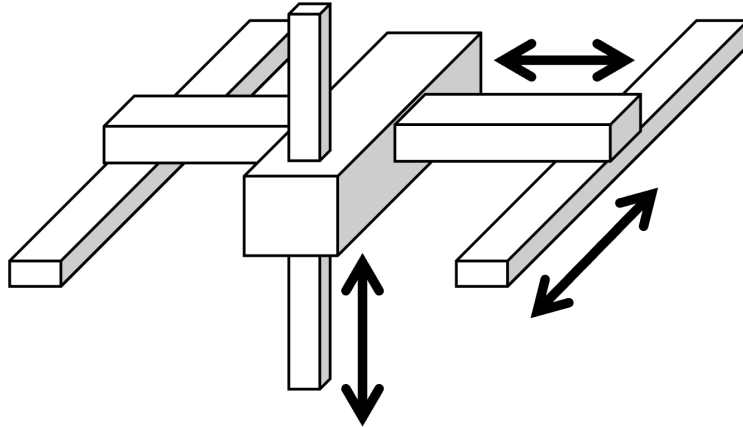


Figure 2.10: Diagram of a cartesian robot.

These robots make it possible to easily simulate linear movement in three axes and are cheap and easy to set up. There are some examples of using these to simulate an earthquake [Wolfe, 2021] [Rodríguez, 2009], which is useful for testing its effects on the data captured by the motion sensor. The apparatus developed by [Makalesi et al., 2021] (Figure 2.11), while not exactly a cartesian robot according to the definition presented previously, uses some of the same principles to move a sensor in one axis, while also being able to rotate it in another axis. The ability to move an object in these three axes is useful in the scope of this project, but because the main focus is mimicking human movements, this kind of apparatus cannot be used as-is in this project because it cannot rotate an object along the *yaw*, *pitch* and *roll* axes.



Figure 2.11: Motion sensor' testing apparatus, developed by L. Yavan and C. Tepe [Makalesi et al., 2021].

- **Uncontrolled Apparatus:** The main difference between this apparatus and the one presented before (apart from the hardware differences) is the level of control. In the previous apparatus, all movements must be calculated and described by the controller first to then be replicated by the apparatus. In this apparatus, the movement is not defined by a controller and, therefore, cannot be finely controlled. An example of this type of apparatus is shown in Figure 2.12.

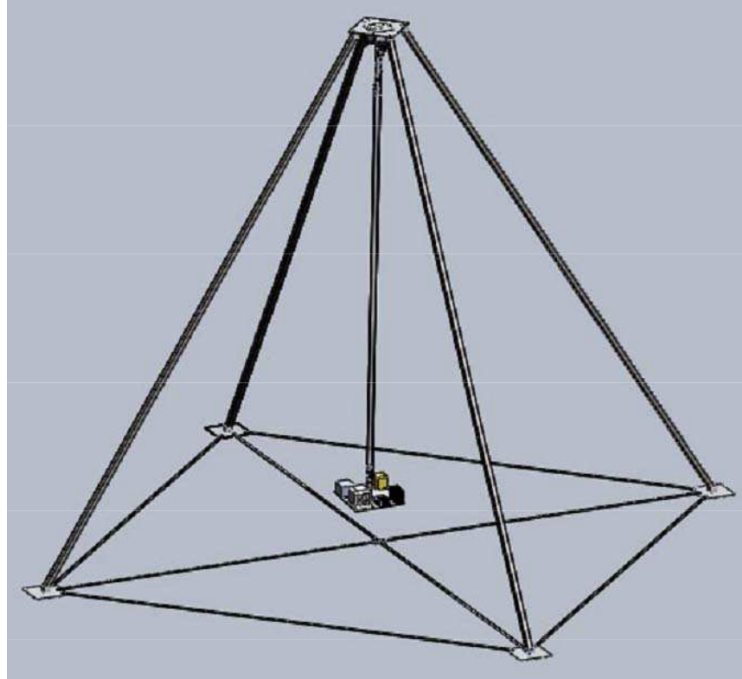


Figure 2.12: Pendulum apparatus for testing IMU [Chaulk, 2011].

This specific apparatus (Figure 2.12) was developed for testing different IMU's for their capabilities and accuracy. Only two operation states are possible: stationary and pendulum. The stationary state is used for performing what the author calls "static tests": these are held for a longer duration (between one and eight hours) and their primary purpose is to check the base drift rate of an IMU. The base drift rate is the rate at which a value returned by an IMU deviates over time, due to some internal error. The pendulum state is used to check if the frequency and amplitude of motion have any effect on the drift rate or the output signals which are produced by the IMU. The lack of power needs and the predictable movement are big advantages of this type of apparatus, but because there is no way to finely control them they cannot be used to accomplish the goals of this dissertation.

- **Motionless Testing:** This kind of testing is found only in Micro-ElectroMechanical System (MEMS) accelerometers, and manufacturers usually advertise it as a

feature called “self-test”. This feature allows the testing of the mechanical part of the sensor without applying an external motion to it. Using an electrostatic force, the mechanical sensing element is deflected and activated inside the sensor, simulating device movement [Janousek et al., 2019]. This allows for precise testing and calibration without the need to develop and build a physical apparatus, which could be expensive and take up extra time. This is mostly used for quickly testing if a motion sensor is working properly, but when the purpose of the test is to check the sensor’s behavior with a specific subset of movements, this kind of test is not practical.

As we can see, the devices presented here have advantages over one another. Cartesian robots are easy and fast to set up, the uncontrolled apparatus needs no power, and motionless testing requires no extra space, as the body of the sensor does not move. A common factor to all of them is the lack of human-like movement. These types of devices can stimulate the sensor in various ways, but none can replicate human arm movement.

In summary, motion sensor testing is an important part of the development of any device that employs it and there are several different ways of doing it, such as the ones presented in this section. While all of these have advantages over one another, choosing the most appropriate solution for this dissertation will always entail understanding the goals and limitations (such as costs and workspace available).

2.5 Non-Commercial Projects

There have been multiple solutions developed for physically replicating human movements (such as the apparatus developed by [Koenemann and Bennewitz, 2012], which replicates whole-body motion), and this section aims to present the projects that focus on mimicking the movement of a human arm. By studying and presenting their main advantages and disadvantages when compared to one another, the best solution for the problem at hand can be found. Even if the search scope is reduced to arm movement only, there are still many solutions that differ from one another because of their purpose. For example, if the main purpose is accurately simulating hand movement, the chosen apparatus will greatly differ from the one chosen with a focus on the movement of the arm and forearm.

Some of the published solutions are usually developed for mimicking a user’s movement in real-time. An example of this type of application is the apparatus developed by [Mangate et al., 2022] for simulating hand movement, found in Figure 2.13.

This project uses a camera to scan the user’s hand and replicates the user’s gestures by using five servo motors, each attached to one of the fingers. After using OpenCV (a computer vision library widely used by both industry and academia



Figure 2.13: Gesture replication in real-time [Mangate et al., 2022].

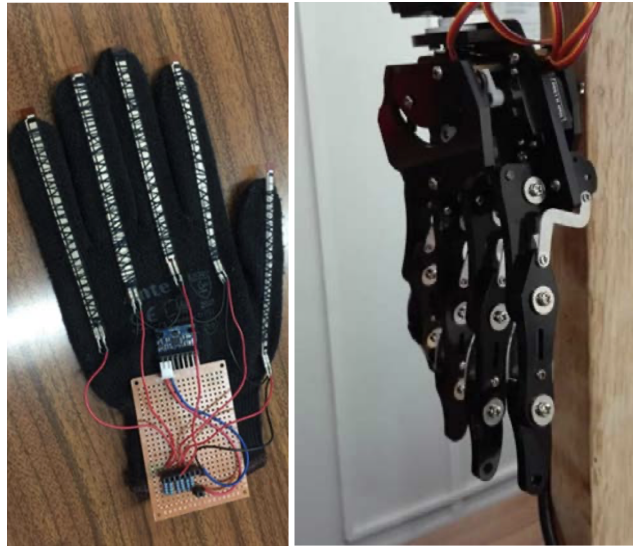
[OpenCV, 2023]) to analyze the image from the camera, the values are sent to an Arduino Uno which, in turn, commands the servo motors for replicating the user's gesture.

Another example of mimicking a user's movement in real-time is the glove and arm sleeve presented by [Oliveira et al., 2019]. The developed prototype is presented in Figure 2.14.

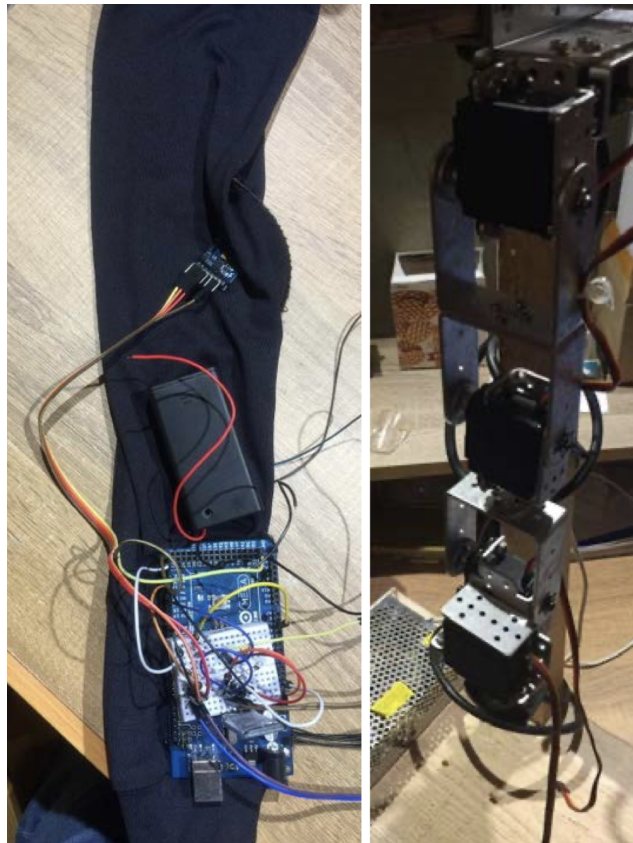
This solution not only replicates the movement of a user's hand but also the arm movement. While the project presented previously used a camera as the input method, this solution uses five flex sensors and one IMU in the user's glove, and one flex sensor and one IMU in the sleeve. These components are all connected to an Arduino MEGA that communicates with another Arduino via Bluetooth through an HM-10 module. The second Arduino is used for controlling both the robotic arm and the robotic hand.

Both of these apparatuses offer great adaptability because they can replicate most movements the user desires, while only requiring input from the user. This is an advantage in this context because the user can easily change the type of movement when needed, but in a testing scenario, it can also be considered a disadvantage. When the desired outcome is the repetition of a particular movement or subset of movements, the need for user input for every single movement delays the testing process and requires repetitive behavior from the user that could be avoided.

As so, the focus of this project should be implementing an apparatus that can replicate any movement the user desires while also requiring minimal input. An example of an apparatus that does just this is the one designed by [Jones and Kumar, 2018]. A robotic arm composed of five servo motors is controlled using

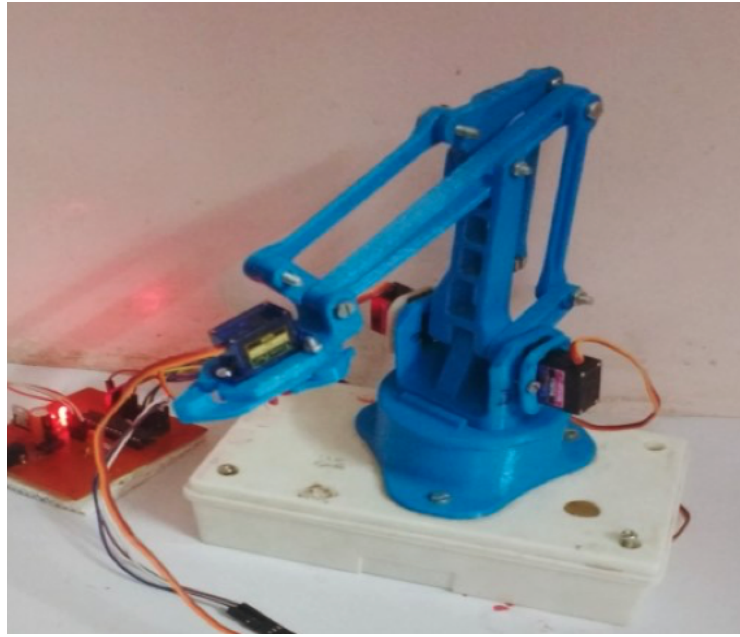


(a) Left: sensory glove; Right: robotic hand that replicates the movement.

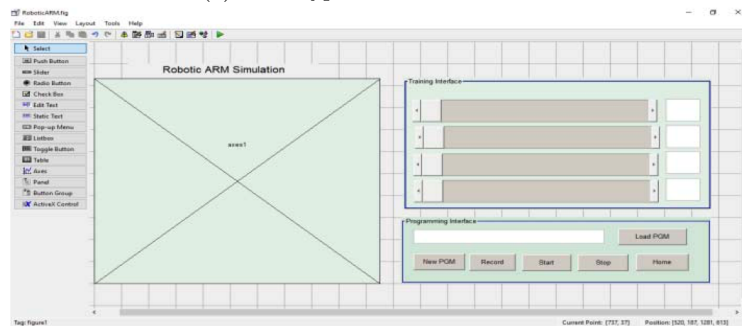


(b) Left: sensory arm sleeve; Right: robotic arm that replicates the movement.

Figure 2.14: Apparatus used for mimicking arm and hand movement [Oliveira et al., 2019].



(a) Prototype of the robotic arm.



(b) GUI developed for interacting with the robotic arm.

Figure 2.15: Project developed by R. Yenorkar and U. M. Chaskar [Yenorkar and Chaskar, 2018].

potentiometers. This way, the user can easily interact with the robotic arm and move it to any position without needing to physically replicate the move, like the projects presented previously. Another example is the apparatus presented by [Yenorkar and Chaskar, 2018] (Figure 2.15). While also using servo motors for moving a robotic arm, the authors developed a MATLAB Graphical User Interface (GUI) (presented in Figure 2.15b) for input instead of potentiometers. Both these projects make use of control methods better suited for a testing scenario like the one targeted in this project (see Section 1.2). This is particularly true when talking about the solution that uses a GUI, as it only requires a small number of clicks for moving the robotic arm to the desired position.

Analyzing the projects presented in this section, we can easily notice that most make use of robotic arms as the apparatus that replicates the user's movement. This should be expected, as the focus of the study in this section is the replication

of human arm movement. The distinctive factor between them is the control method. In [Mangate et al., 2022], a camera captures the movement of a user and a human hand model replicates it (see Figure 2.13); while in [Oliveira et al., 2019], the user’s movement is captured by an IMU, and replicated by a robotic arm (see Figure 2.14).

As discussed in this section, these control methods require too much effort from the user and are not adequate in a testing scenario. Because the focus of this dissertation is testing SWORD’s MT’s, the solution developed should be able to replicate the desired movement without requiring much effort from the user. This is why projects like [Jones and Kumar, 2018] and [Yenorkar and Chaskar, 2018] have been discussed in this section: to present two different control methods that do not require the user to physically perform the movement that the apparatus should replicate.

After the research presented in this section, these are the main remarks worth noting: First, the adopted apparatus in each project differs from other projects (e.g. differences between [Mangate et al., 2022] and [Oliveira et al., 2019]), but most apparatuses are not robotic arms, humanoid robots, or parallel robots. This is mainly due to the characteristics of each project, but could also be related to the added complexity of these apparatuses. And second, although the most accurate way to replicate a human shoulder and elbow is through the use of spherical joints [Wilhelms and Gelder, 2001], most projects make use of multiple rotational joints. This could be due to the complexity discussed in Subsection 2.2.1.

Chapter 3

Design Choices

As mentioned in Section 1.2, this dissertation aims to develop an apparatus that replicates a subset of movements of the human arm, with the goal of analyzing the data gathered by motion trackers used in physical therapy sessions and validating the associated algorithms. The various projects presented in Section 2.5 show that there is more than one way of replicating human movement. Three possible approaches were considered as possible candidates for the development of our apparatus:

- 3D Printer: This printer has the same behavior as a cartesian robot, and as such can move in three axes: X , Y and Z . This was the main reason for considering this option, but because it lacks movement along the *yaw*, *pitch* and *roll* axes, it was not chosen as the solution to our problem.
- Robotic Arm: This option provides movement along all six axes, and was considered as an option because it mitigates the shortcoming of the 3D printer. At first, this seemed like the best solution to our problem, but the problem that arose quite early in the development stage was the added complexity of this option. Because we are looking to replicate the movement of only the human arm, there is no need for six DoF. Therefore, developing a six DoF robotic arm is not required in the scope of this dissertation. Another caveat is the trajectory: unless a robotic arm is specifically built to mimic the human arm, the trajectory it takes from point A to point B will not always resemble the trajectory that a human arm would do. Because we want to mimic a subset of movements of the human arm, we looked for another solution.

- Human arm replica: This approach consists of replicating the exact joints and links found in the human arm. This way, there will be no unnecessary components, and the final solution will better resemble the trajectory/movement of the human arm. This was the chosen approach because it accomplishes the goals of this dissertation without overcomplicating the apparatus.

To create this replica, we must first define the number of links and joints needed. Figure 3.1 represents the joints and links that our apparatus should replicate.

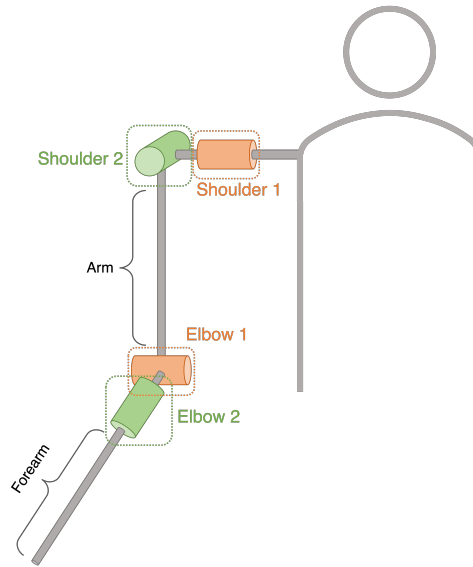


Figure 3.1: Human arm represented as a set of joints and links.

To replicate the human arm, we will need to use four motors (two for the shoulder, two for the elbow), and two links to replicate the human forearm and upper arm. This model does not exactly match the human arm movements, as the shoulder should be modeled with three rotational joints, not two. For the sake of this dissertation, this third joint will not be considered because the added complexity is not justified. Because we want to replicate only a subset of movements that can be performed using two motors, implementing the shoulder joint with three motors is beyond the scope of this dissertation.

To develop a prototype of this apparatus, there is a need to decide which materials and components to use. These include types and specific models of components, communication protocols, and software architecture. In this Chapter, the reader is presented with the various options considered during this process and guided through the analysis made to find the final choice for each component. Figure 3.2 presents a diagram of the main components of the apparatus.

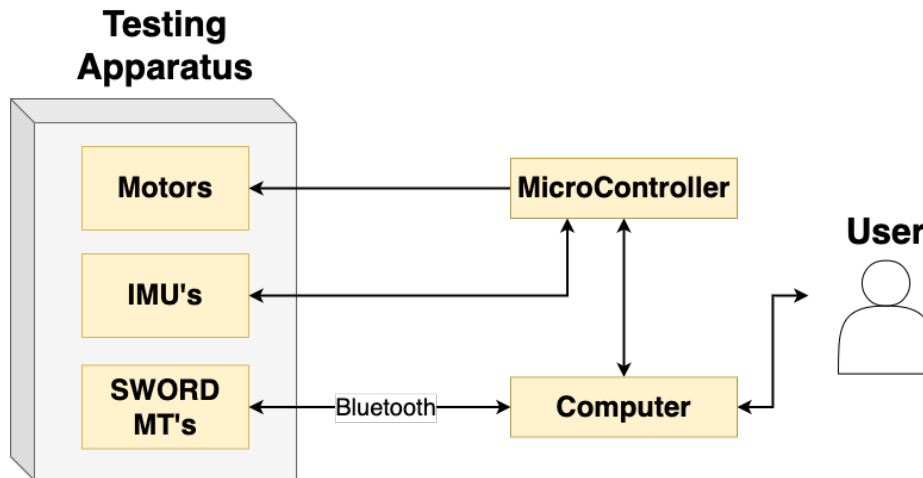


Figure 3.2: Functional diagram of this project.

Besides placing the MT's on the apparatus, the user will only interact with the computer. It will be used for selecting the desired position of the arm (e.g. by choosing from the available presets) or choosing the specific angles for each motor. The computer will send this data to the MicroController Unit (MCU), which will in turn control the movement of each motor. After the movement of the apparatus, the MCU reads the feedback sent by the IMU's and sends it back to the computer. The MT's placed on the apparatus also send their data to the computer via Bluetooth. Finally, the gathered data is displayed to the user on the computer.

The goal of the apparatus is testing the MT's, and the IMU's are used as a way of locating the apparatus in 3D space, and as a reference point for comparing the collected information with the one collected from the MT's.

This chapter is divided into the following sections:

- **Hardware:** This section contains the main physical components used for implementing the testing apparatus. This includes the MCU, the motors for moving the apparatus, the materials used for simulating the human arm, and also the protocol used for communication between the computer and the MCU.
- **Software:** This section presents the software requirements for controlling the apparatus, such as the inputs needed and the principles behind the control algorithm.

3.1 Hardware

The choices made concerning the hardware should reflect the main technical requirements of this project. The final apparatus should be able to replicate the movements

of the human arm accurately and precisely. This section presents the considered options for each component and the final choice for each, as well as the reasons for each choice.

3.1.1 Motors

The motors are the moving parts of the apparatus and are used as a simulation of the human shoulder and elbow joints. As such, they should be able to replicate the ROM of their human counterparts, while being able to be controlled accurately. The best option for replicating the human shoulder/elbow joints would be a spherical joint, as discussed in Subsection 2.2.2, but this type of joint is complex and expensive, at the time of writing. Because this complexity is not necessary for the scope of this thesis, the search for a proper solution will be focused on the most common types of motors.

In embedded applications, there are four main types of motors used - Direct Current (DC), brushless, stepper and servo - each having its own set of characteristics. Because we want to move the apparatus to specific angles, the continuous rotation that DC and brushless motors provide is not needed in this dissertation, so these will not be used. Both the stepper and servo motors can perform rotations to specific angles, but stepper motors are usually heavier, expensive, and demand a more complex type of control (when compared to the servo motor). They also have a fixed step, which means they are limited in the angles that they can reach. Because of all the points presented in this paragraph, the type of motor chosen for the apparatus was the servo motor.

After choosing the type of motor, the next step is choosing the specific model to be used. The final option should have the torque needed to move the apparatus while remaining inexpensive. It should also contain the ROM of the human shoulder and elbow joints. To accomplish these goals, the chosen motor is the DSS-M15S servo motor from DFRobot (presented in Figure 3.3).



Figure 3.3: DSS-M15S servo motor, from DFRobot [DFRobot, 2018].

Table 3.1 contains the main characteristics of this motor:

Table 3.1: Key characteristics of the DSS-M15S servo motor.

Characteristic	Value
Torque	15 Kg cm (when powered with 7V)
Rotation Range	0° to 270°
Feedback Signal	0V to 3.3V

A useful feature of this motor is that it can provide feedback to the MCU, using an analog signal that ranges from 0V to 3.3V. This signal can be used to correct the angle of the motor and make the apparatus more accurate overall. Another benefit is its low cost when compared to other servos with similar torque - its cost is below 20€ (at the time of writing).

As these motors are to be used to implement the human shoulder/elbow joints, we will need to use four of them (see Figure 3.1 for more details). Each motor is controlled using a Pulse Width Modulation (PWM) signal, which will be explored further in Section 4.2.

3.1.2 Inertial Measurement Unit

As discussed in the introduction of this Chapter (Chapter 3), the purpose of the IMU is locating the apparatus in 3D space, and measuring the acceleration and rotation that each joint is subjected to. This component will be used as a way of making sure that the movements desired by the user are being correctly replicated by the apparatus. This subsection presents the various options taken into account when selecting the IMU to be used in the final apparatus.

As seen in Subsection 2.3.2, sensor fusion can improve the accuracy and precision of the data gathered by these sensors. To implement this method in our apparatus, the chosen IMU should have at least two types of sensors available (selected among three possible types: accelerometer, gyroscope, magnetometer). The options presented in this subsection reflect this choice.

The first option considered was the MPU-60X0 range of IMU's from InvenSense (presented in Figure 3.4), used in many projects by both academia and maker community because of its low cost and features in a small package.

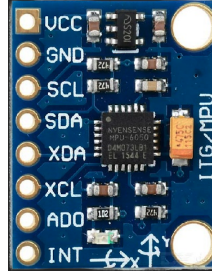


Figure 3.4: MPU-6050 [HTF, 2023].

It contains both an accelerometer and a gyroscope, which allow for motion tracking along six axes ($x, y, z, yaw, pitch$ and $roll$). Table 3.2 presents the ranges available for each sensor:

Table 3.2: Ranges of the sensors inside the MPU-6050.

Sensor	Available Ranges
Accelerometer	$\pm 2g, \pm 4g, \pm 8g$ and $\pm 16g$
Gyroscope	$\pm 250dps, \pm 500dps, \pm 1000dps$ and $\pm 2000dps$

Because of its wide adoption, there are already various examples of the use of this IMU in various applications (e.g. [Mahmood et al., 2020], [Aditya et al., 2022]). But despite these advantages, the manufacturer does not recommend them for new designs [InvenSense, 2023]. Because of this, this IMU will not be used in this project.

Another option is the LSM6DSO IMU from ST. This IMU also contains an accelerometer and a gyroscope. Table 3.3 presents the ranges available for each sensor:

Table 3.3: Ranges of the sensors inside the LSM6DSO.

Sensor	Available Ranges
Accelerometer	$\pm 2g, \pm 4g, \pm 8g$ and $\pm 16g$
Gyroscope	$\pm 125dps, \pm 250dps, \pm 500dps, \pm 1000dps$ and $\pm 2000dps$

Because it accomplishes the goals discussed, the chosen IMU for this dissertation is the LSM6DSO.

To ease the work of connecting the IMU to the MCU, a breakout package with this IMU will be used. The chosen breakout board is made by SparkFun, and presented in Figure 3.5.

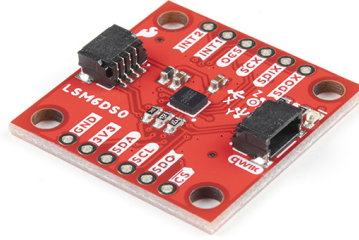


Figure 3.5: LSM6DSO Breakout, from SparkFun [SparkFun, 2023].

Since we need to track the position of the apparatus in 3D space and compare it with the position read by the MT's, we will use two IMU's (one placed on the "forearm", the other placed on the "upper arm").

This IMU offers the possibility of using either Inter-Integrated Circuit (I2C) or Serial Peripheral Interface (SPI) communication protocols. This project does not require the difference in transmission speed that SPI offers over I2C (10MHz clock frequency with SPI and 400KHz clock frequency with I2C, according to the LSM6DSO datasheet [ST, 2019]). On top of this, I2C only requires two cables, while SPI requires at least four. As we are building a mechanical apparatus, more conductors result in more failure points. Because we will be using two IMU's, and I2C requires fewer conductors to interact with them, that will be the chosen protocol.

3.1.3 Communication between Computer and Microcontroller

The MCU must be able to communicate with the computer, transmit the orders from the user to the apparatus, and present results back to the user. Various options were considered for this connection:

- **Wireless:** Because the apparatus has moving parts, the lack of conductors that wireless technologies like Bluetooth and ANT provide is a great advantage. But these options would require the purchase of a module for the microcontroller and would add complexity beyond the scope of this dissertation. As such, it will not be used for communication between the computer and the microcontroller.
- **Universal Serial Bus (USB):** As a widely adopted protocol, there are many documented examples of using USB for communication between computer and

microcontroller. But this protocol is more complex than others discussed in this list as it supports features like device identification, enumeration, and different network topologies. These features are not required, as the goals of this dissertation can be accomplished without this extra complexity. Because of this, this protocol was not used.

- Universal Asynchronous Receiver-Transmitter (UART): This is the simplest protocol in this list, in terms of complexity. As the transmission speed is not a top priority and there is a need of exchanging simple strings, UART was the chosen protocol for communication between the computer and the microcontroller.

3.1.4 Microcontroller

The MCU is responsible for controlling the motors on the apparatus, configuring and reading the values from the IMU, and transmitting information between the apparatus and the computer. The main requirements for the MCU are the output of four PWM signals (for driving the servo motors), and the availability of I2C/SPI communication peripherals (for communication with the IMU's). It should also be able to communicate with the computer using UART. Because this list of requirements is not too restrictive, there is a large range of MCU that will suffice. As such, the differentiating factors for the final choice are availability and cost.

In the current status of the industry, a solid choice would be an Advanced RISC Machine (ARM) based device. When talking about ARM Cortex processors, these can be split into three main categories (according to [SiLabs, 2023]):

- Cortex-A: Application processor cores for performance-intensive systems.
- Cortex-R: High-performance cores for real-time applications.
- Cortex-M: Microcontroller cores used in a wide range of embedded applications.

Because we are looking to implement a prototype as a proof of concept, we looked into the Cortex-M family of devices. While it is not required, choosing an MCU that has a Floating-Point Unit (FPU) is desirable since we want to manipulate four PWM signals in real-time, and perform sensor fusion with the IMU's. Because we have worked with their products in the past, we started looking into the product ranges from ST.

An option that meets all these requirements, while also being available and inexpensive at the time of writing, is the STM32G431RB MCU from ST. This device is based on the Arm Cortex-M4 processor and offers I2C, SPI, UART, and PWM output on up to eight simultaneous outputs. As a way of easing the prototyping

process, the NUCLEO-G431RB development board will be used. This development board is presented in Figure 3.6.

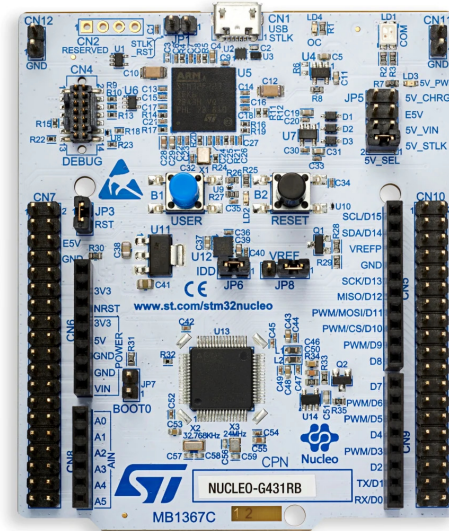


Figure 3.6: NUCLEO-G431RB development board [ST, 2023].

3.1.5 Power Delivery

The components presented in this section have different power requirements, which should be taken into account when designing the power delivery of this solution. The MCU can be powered using the USB cable used for the UART communication, and the IMU's can be powered by the MCU (using the 3.3V it outputs). The biggest outlier when it comes to power requirements is the DSS-M15S servo motor. It has an operating voltage of 4.8V to 7.2V while requiring a maximum current value of 2A when in stall (according to the motor's datasheet [DFRobot, 2018]). Because of these values, the apparatus cannot be powered solely by the MCU, as its maximum voltage output is 3.6V and the maximum output current is 150mA (according to the datasheet [ST, 2021]). Because of this limitation, an external power source must be used to power these motors.

As discussed in the introduction of this Chapter, we will need to power four motors. As each motor requires a maximum of 7.2V and 2A, the selected power supply was the Mean Well LRS-100-12. This power supply has a rated current of 8.5A and outputs 12V of DC voltage, which translates into a rated power of 102W. To step down the 12V into the 7V that the motors require, four DFR0379 buck converters will be used. This buck converter is rated for an input voltage of 0V to 40V, output voltage of 1.25V to 37V, and output current of 3A. The LRS-100-12 power supply and DFR0379 buck converter are presented in Figure 3.7.

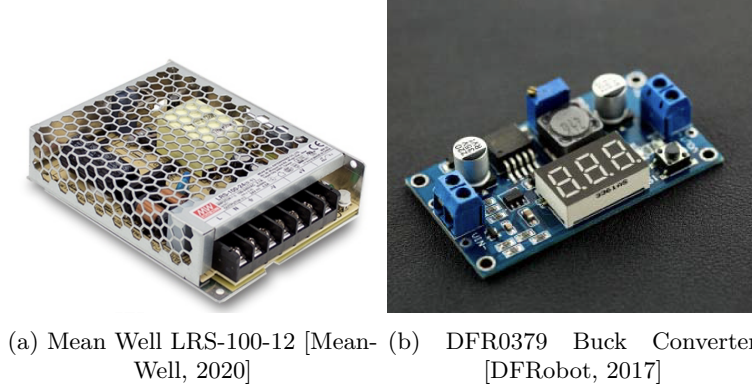


Figure 3.7: Components used for powering the apparatus.

Figure 3.8 presents the power distribution diagram, to verify that these components would work for this dissertation.

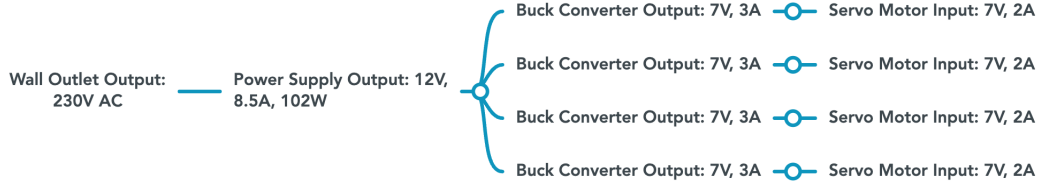


Figure 3.8: Power distribution diagram.

3.1.6 Links, Joints and Base

All the electronic components have been discussed in the previous subsections. The next step is choosing the hardware that will connect all these electronics. There are three main hardware components left to discuss: links, joints, and base.

The links simulate the human arm and forearm, connecting pairs of joints between them. The material chosen for this element should have the necessary rigidity to withstand the movement of the apparatus, while still being lightweight enough to be moved by the servo motors. To implement these links, the best solution found was a PVC pipe. It achieves the requirements discussed here and is also easy to work with using the tools available.

The joints serve as a replica of the human shoulder and elbow. The aspects discussed in this paragraph refer to the hardware needed to replicate the human joints, other than the motors. Because we have chosen servo motors for these joints, there will be a need to use multiple motors to simulate a single human joint. As such, we need to find the optimal way of connecting these motors, to faithfully replicate the human joints. This scenario is rather specific, and there are no commercially available components that fit our architecture (none that are easily available at the time of writing, at least). Because of this, the best solution is designing and 3D printing models adapted to our specific needs.



Figure 3.9: Fire extinguisher base used as "torso" of the apparatus [Extintor, 2023].

The base will serve as the "torso" of our apparatus. The chosen solution for this component was a fire extinguisher base, such as the one in Figure 3.9. This was the chosen solution because it is tall enough to hold the rest of the hardware. It is also sturdy enough to remain stable when the other components are moving.

All these components will be discussed extensively in Chapter 4, regarding the development process and various iterations.

3.2 Updated Architecture

After establishing the components that will be used, this project's architecture can be updated. This version is presented in Figure 3.10.

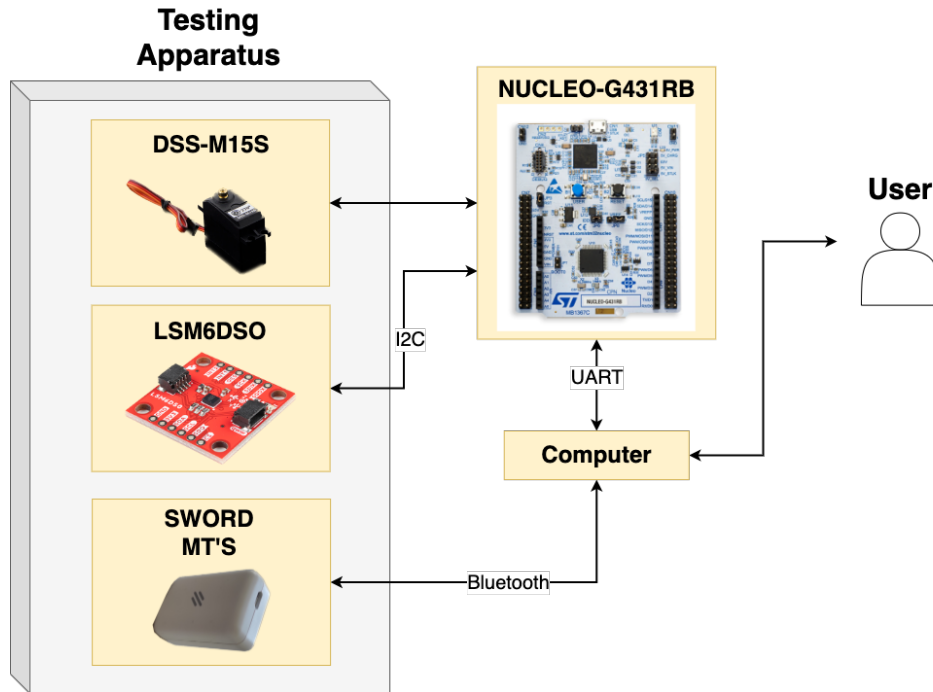


Figure 3.10: Updated version of the Functional Diagram.

There are three main components in this project. Using the figure above as a reference, we can now detail specifically what each component does:

- **Computer:** Serves as the interface between the user and the apparatus. Will be used for communicating to the apparatus what the desired position is, and showing the results of the test to the user.
- **Microcontroller:** Controls the apparatus by outputting PWM signals to the motors, reads the feedback from the motors and data from the IMU's, and changes the PWM signals until the desired setpoint is reached. In our apparatus, the chosen MCU is the STM32G431RB, in the NUCLEO-G431RB development board.
- **Testing Apparatus:** The element that replicates the human arm. It is built using the DSS-M15S servo motors, the LSM6DSO IMU's, and the hardware discussed in Subsection 3.1.6 that connects everything. This is also where the SWORD's MT's are placed. Because they already have velcro on them, they will be placed on velcro strips in the apparatus.

Figure 3.11 presents the electrical diagram of the components used in the apparatus.

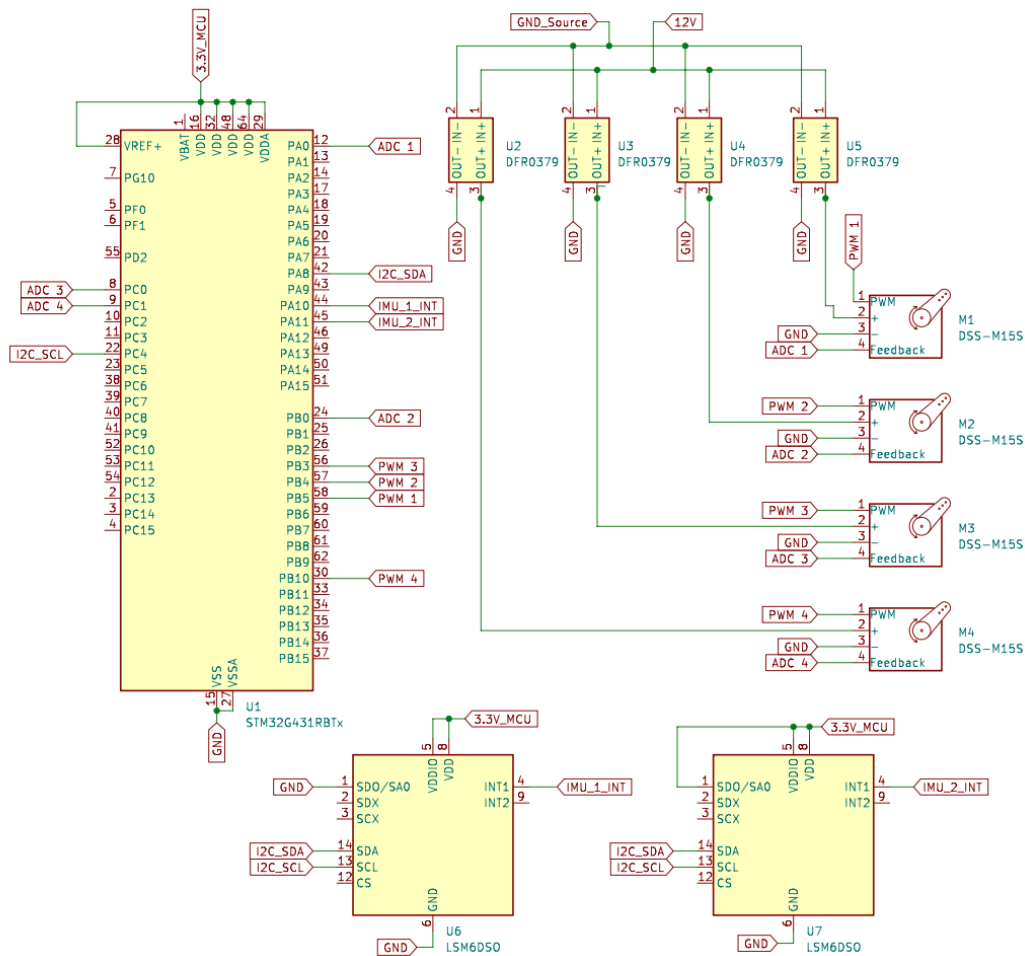


Figure 3.11: Electrical diagram of the apparatus.

The MCU is powered through the USB port on the development board, and this board generates the 3.3V used to power the IMU's. The 12V and GND_Source are generated by the MeanWell power supply. The pull-up resistors required for the I2C bus are implemented using the pull-ups on the General-Purpose Input/Output (GPIO) pins of the MCU.

This is the hardware needed to implement the testing apparatus. The following section covers the software developed for this project.

3.3 Software

After discussing all the physical components and their roles in the apparatus, the following step is to develop the software that will tie all components together. This section presents the software architecture. The discussed topics are the use cases, and the algorithm used for controlling the apparatus.

3.3.1 Use Cases

The main goal of this dissertation is to test SWORD's MT's for their accuracy (as discussed in Subsection 1.2), which requires the user to be able to interact with the apparatus. This interaction is the main use case in this dissertation, therefore the software should be developed with this in mind. Figure 3.12 presents the use cases.

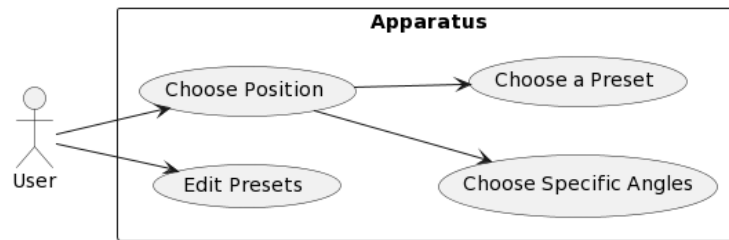


Figure 3.12: Use cases of the apparatus.

The user must be able to choose the position of the apparatus, either by selecting the angles of each motor, or choosing a preset. This list of presets can be edited by the user, which is the other use case. The implementation of these use cases will be discussed in Chapter 4.

3.3.2 Control Algorithm

The purpose of the control algorithm is to configure the hardware and move the apparatus according to the user's orders. This consists in sending the PWM signals, reading the feedback from the motors and IMU's, and adjusting the position accordingly. If this control algorithm is working properly, the apparatus should be able to move to any setpoint the user desires and adjust its movement when the desired setpoint is not reached. The diagram in Figure 3.13 shows a high-level vision of how this control algorithm should work.

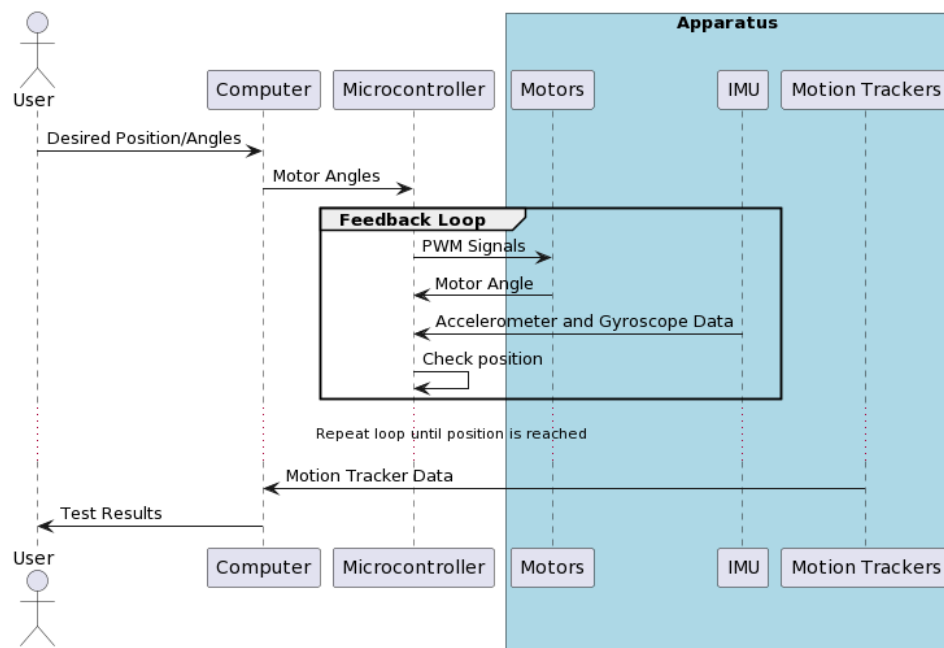


Figure 3.13: High-level view of the control algorithm.

The feedback loop occurs until the desired position is reached, or the loop goes through a predetermined number of iterations (so that if the angle is never reached, the apparatus will not keep moving).

3.4 Bill of Materials

After choosing all the components, the resulting bill of materials was created and can be consulted in Table 3.4.

Table 3.4: Bill of Materials.

Description	Model	Quantity	Unitary Cost (€)	Total Cost (€)
Servo Motor (w/ encoder)	DSS-M15S	4	14.95	59.80
Servo Horns	-	6	2.85	17.1
Heat Sink	-	1	2.28	2.28
Microcontroller	NUCLEO- -G431RB	1	14.66	14.66
Servo Brackets	-	3	8.84	26.52
PVC Pipe	-	1	5.29	5.29
Fire Extinguisher Base	-	1	17.71	17.71
Buck Converter	DFR0379	4	4.61	18.44
Power Supply	MeanWell LRS-100-12	1	20.33	20.33
IMU's	LSM6DSO	2	12.20	22.40
Total				204.53

After defining the architecture of this project, the next step was building the prototype and implementing the control algorithm. This is the main topic of Chapter 4.

Chapter 4

Implementation

After defining the main components of this project and deciding on the solution for each, we started assembling the apparatus. This chapter focuses on guiding the reader through the choices made during the assembling process, explaining the firmware developed for the MCU, and presenting the interface that allows the interaction between the user and the apparatus.

This chapter is divided into the following sections:

- **Apparatus Hardware:** This section presents the process of assembling the hardware of the apparatus, and the method developed for connecting the apparatus to the MCU.
- **Control of the Apparatus:** The development of the firmware for the MCU is detailed, and the choices made during this process are discussed. The calibration of the servo motors is also presented in this section.
- **Position Tracking:** This is where the process of locating the apparatus in 3D space is explained. It goes over the characteristics of the IMU's, and how we used sensor fusion to get more precise results.
- **Microcontroller shell:** An interface built for interacting directly with the MCU. This section discusses why it was necessary, its advantages and disadvantages.
- **User Interface:** This section details the process of developing the interface used to interact with the apparatus and its features.

- Final Remarks: Some comments to complement the topics discussed in the previous sections.

4.1 Apparatus Hardware

The physical apparatus can be divided into three main components: joints, links, and base. To assemble all of these components, we needed to find a way of connecting them. Each connection is a different problem because these components have different dimensions and geometries. Therefore, each connection was studied to find the best solution for each.

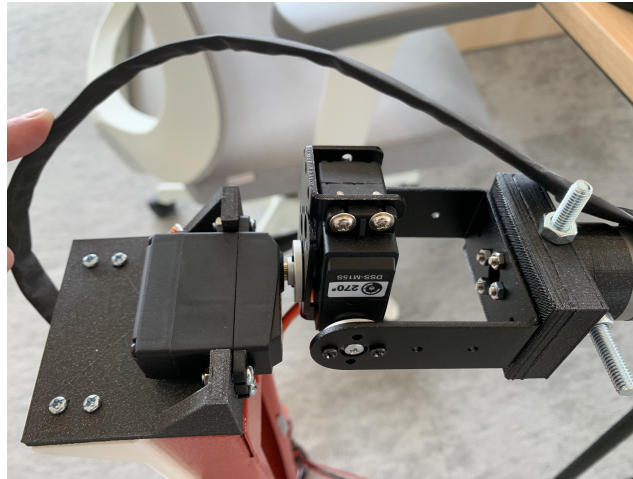
4.1.1 Joints

As discussed previously in Subsection 3.1.6, we needed to connect pairs of motors to replicate each human joint (shoulder and elbow). The first question that arose during the assembling process was: *how can we attach these motors?*. While exploring the available solutions, the most common method found for attaching two servos was using servo brackets. An example of these brackets is presented in Figure 4.1.

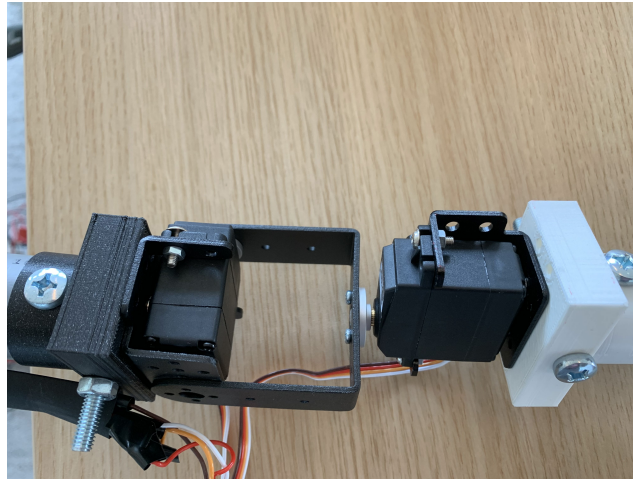


Figure 4.1: Servo brackets [ElectroFun, 2022].

These brackets allow us to connect two servo motors. These pairs are referred to simply as “joints” for the remainder of this dissertation. According to Figure 3.1, these joints are not exact copies of one another, as the movements that each should perform are different. Because of this, the joints in this project are not identical. Figure 4.2 presents them side by side.



(a) “Shoulder” joint



(b) “Elbow” joint

Figure 4.2: Joints of the Apparatus.

The main differences come from the relationship between servos when it comes to their orientation. These pairs can be used to replicate the human shoulder and elbow joints, although with some limitations. These limitations will be discussed in Chapter 6.

4.1.2 Links

After deciding on and implementing the joints, the following stage is connecting the joints. This connection is referred to as “links” in this chapter, and as discussed before, a PVC pipe will be used to implement them. This creates a new problem: *how can we connect the joints to the PVC pipe?* The solution found in most projects for attaching a servo motor to a PVC pipe is a clamping mount. Figure 4.3 shows an example:



Figure 4.3: Clamping Mount [ServoCity, 2023].

This clamp could be attached to a PVC pipe on one end and to a servo bracket on the other. The problem with this option is that it would not allow us to properly replicate the appearance of the human arm, and at the time of writing, there are no other commercially available solutions that fit our architecture. Therefore, a different solution had to be found. Because a 3D printer was available, the chosen approach was creating 3D models from the ground up. With this solution, these pieces could be designed with the characteristics needed for our particular architecture.

The first version of the 3D model of this piece is shown in Figure 4.4.

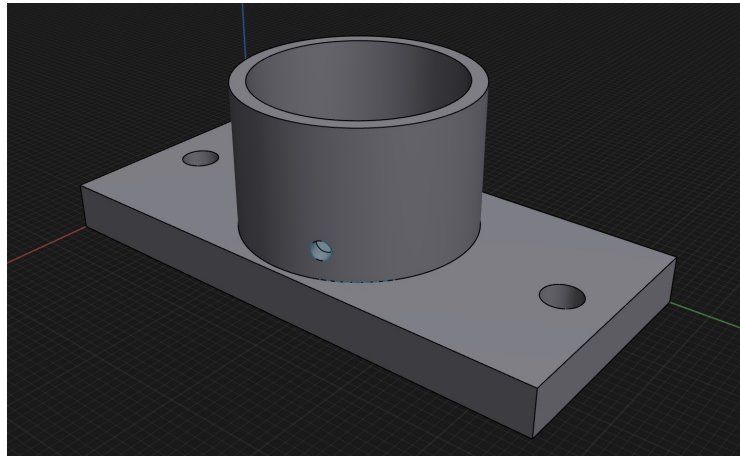


Figure 4.4: First version of the piece that attaches the joints of the apparatus to the PVC pipe.

The base allows this piece to be attached to the joints using screws, and the cylindrical wall on top of the base wraps around the PVC pipe. This pipe is fixed in place using screws that go from one point of the wall to the other and through the PVC pipe. Because the wall is only 3mm wide, there were some concerns about the

rigidity of this piece. As such, a new version of this piece was made. This second version is presented in Figure 4.5.

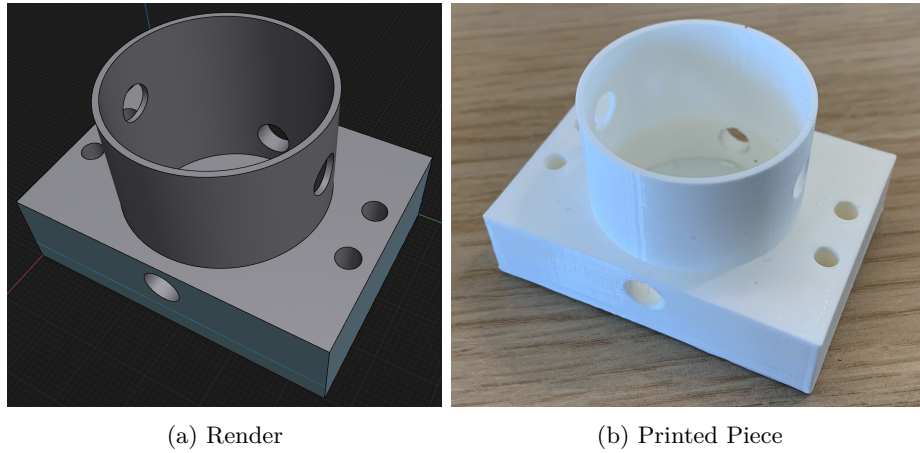


Figure 4.5: Second and final version of the piece that attaches the joints of the apparatus to the PVC pipe.

The added height of the base helps to support the PVC pipe and allows the addition of another screw to secure the pipe in place. This version also adds two extra screw holes on the edge of the base to attach this piece to the joints, as well as another set of screw holes in the center of the wall. This extra set lets this piece be attached to either the elbow or the shoulder joints.

4.1.3 Base

After discussing the joints and the design of these pieces, it is easier to grasp how the apparatus will look. Figure 4.6 presents a rough sketch of its appearance.

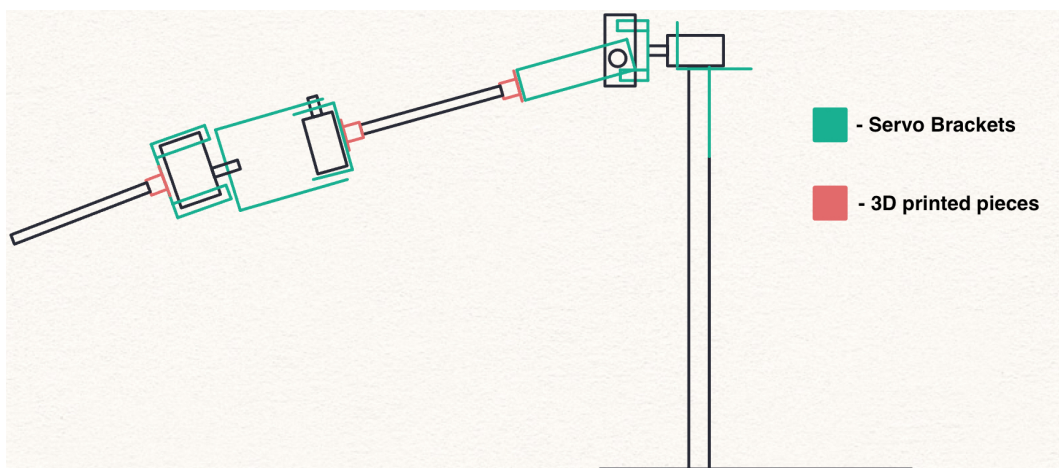


Figure 4.6: Sketch of the apparatus.

The components drawn in green are the servo brackets, and the ones in red are the pieces we designed and 3D printed. Almost all components in this figure have

been discussed, but there is still one component left: the connection between the base and the first joint.

As discussed before, the base of the apparatus is a fire extinguisher support. As such, we needed to find a way to attach the first pair of motors to this base. Because we had already used servo brackets to create the joints, a similar solution was considered. These brackets are presented in Figure 4.7.

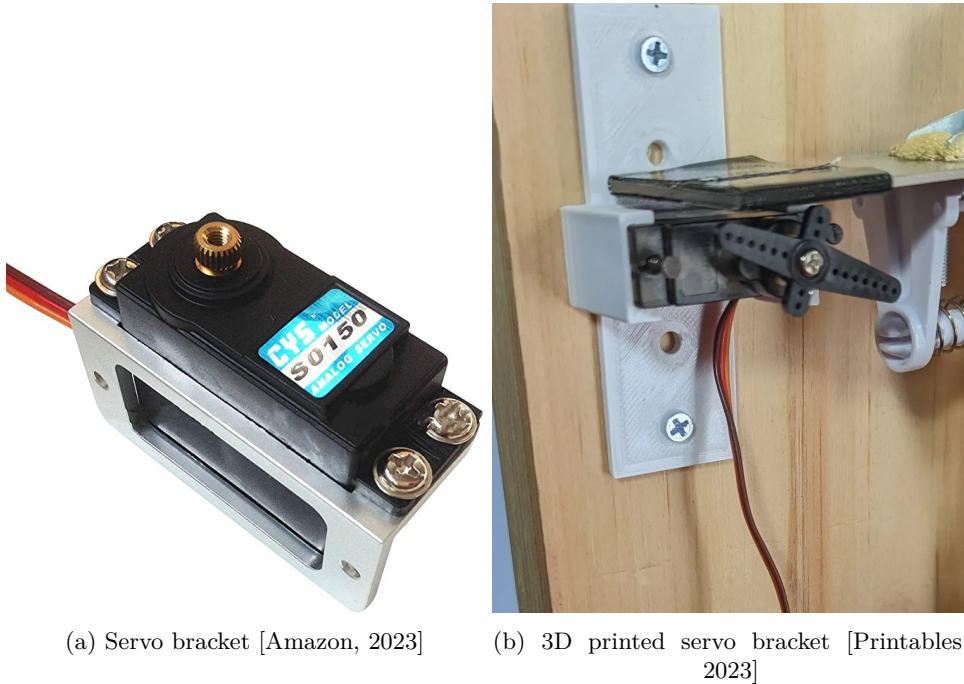
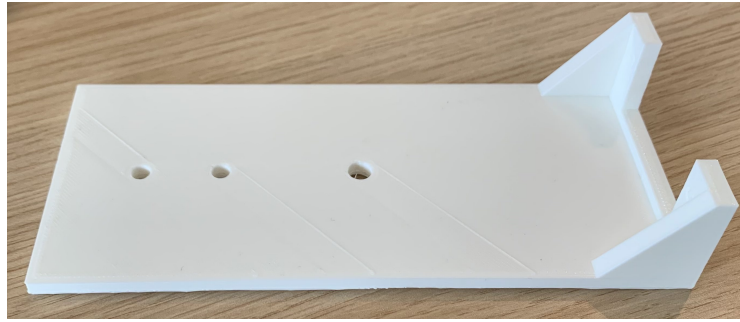


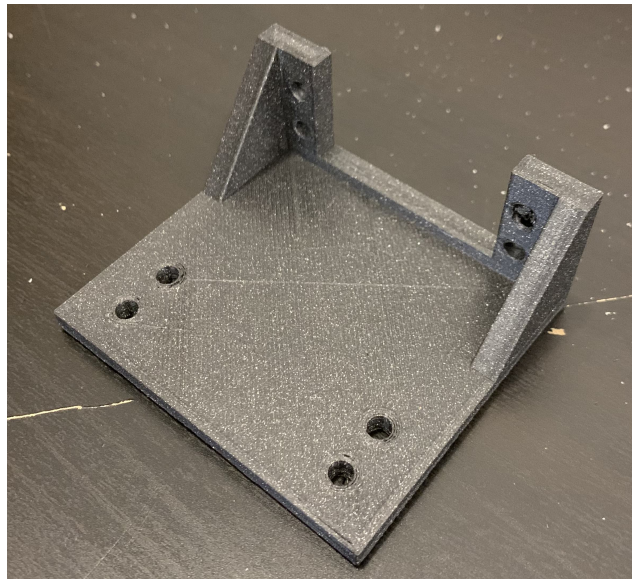
Figure 4.7: Servo attachments for connecting the first joint to the fire extinguisher.

By drilling holes in the fire extinguisher support, these brackets could be attached to it and would secure the first joint. But these brackets were not available for purchase during this dissertation, so a different solution had to be found.

Having already designed and printed a 3D component in this project, this was the next best solution. Figure 4.8 shows the pieces created for this purpose.



(a) First “base” piece



(b) Second “base” piece

Figure 4.8: Attachments between the first joint of the apparatus and the fire extinguisher base.

The piece on the top is attached to the fire extinguisher support using the screw holes already present on the support. The piece on the bottom is screwed on top of the aforementioned piece and allows the first servo motor to be placed in the correct orientation.

These pieces could have been designed as a single piece, but there are two reasons for designing them separately:

- **Different Orientations:** As discussed before in the introduction of Chapter 3, a replica of the human arm that uses servo motors would need to use three motors for the “shoulder”. Because we have decided to use only two motors, there is a set of human arm movements that this apparatus cannot replicate. If the user needs to simulate this set of movements, the black piece can be taken out, and the first servo can be placed directly on the white piece.
- **Human Leg:** This apparatus focuses on replicating the human arm, but could be adapted to simulate a human leg. By placing the first servo on the white

piece instead of the black one, the user could replicate human leg movements.

These options are not currently implemented because they are outside the scope of this dissertation, but could be further explored in future implementations.

These are the main components that make up the hardware of the apparatus. Figure 4.9 shows the first stage of the apparatus, after assembling the components discussed in this section.



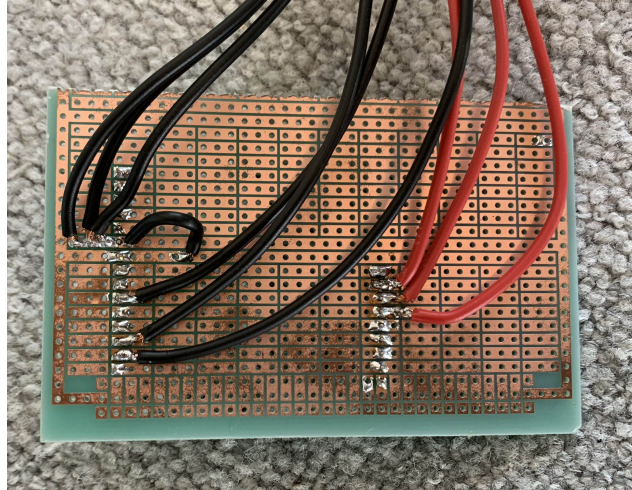
Figure 4.9: Apparatus hardware assembled.

The hardware responsible for delivering power - power supply and buck converters - was placed near the base of the fire extinguisher support. The MT's and the IMU's will be placed on the PVC pipes, but their exact placement will be discussed in Section 4.3.

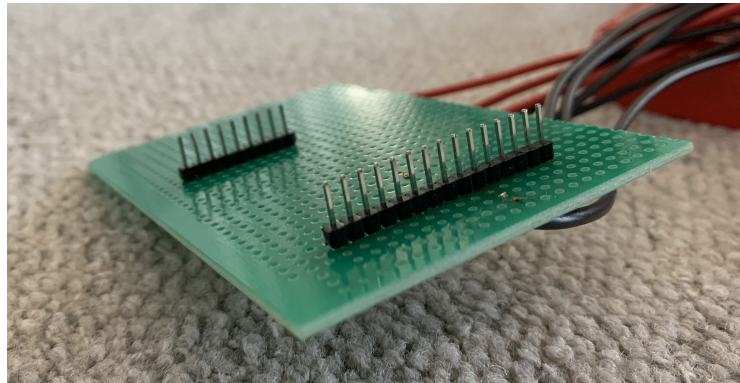
Finally, we need to connect the motors and the IMU's to the MCU. Soldering all the conductors directly to the development board could work, but this is a solution that semi-permanently attaches the MCU to the apparatus. If we needed to upgrade or change the MCU, this would require us to desolder and resolder all the conductors,

which is a problem that could be avoided altogether. Even if we want to change the firmware, this would require moving the entire apparatus.

As a way of eliminating these troubles, the selected solution was building a shield that could be placed on top of the development board. This shield is presented in Figure 4.10.



(a) Top view



(b) Side view

Figure 4.10: Shield to be placed on the MCU.

As a means of improving the modularity of this project, and because the development board allows it, the pins chosen for interacting with the MCU are all located on top of the board. This allows us to use any MCU development board that supports the ARDUINO Uno V3 connectivity standard, as the headers on the shield follow this same standard.

After assembling these components, the following step is developing the control of the apparatus. This development process is discussed in the next section.

4.2 Control of the Apparatus

The physical components of the apparatus have been assembled, but at this stage, the apparatus does not move and cannot be controlled. To control the servo motors, a PWM signal with a specific duty cycle has to be sent to each motor. This is one of the roles of the MCU and was the first feature developed.

According to the manufacturer's datasheet [DFRobot, 2018], the PWM signal that controls the servos must have a frequency between 50Hz and 330Hz. Because it fits inside this range, the selected frequency for the PWM signals in this project was 320Hz. The next step was finding a way of converting the angle we want into the duty cycle of the PWM. To better understand it, this conversion can be divided into three steps:

- Servo angle into servo percentage: Starting with the angle we want, we can convert it into a percentage, using Equation 4.1:

$$Angle(\%) = \frac{Angle \times 100}{270} \quad (4.1)$$

The angle is limited to the $[0, 270]^\circ$ range.

- Servo Percentage into duty cycle: The servos datasheet says that the full range of the motors can be achieved by driving them with a PWM signal that is high during $[500, 2500]$ ms. On top of this limitation, we decided to only use the PWM driver from 20% to 80%, so that the 0° angle would not use a 0ms PWM, and so that the control algorithm still has room to increase the PWM if the signal is already at the equivalent of 270° . Using the percentage we got in the last step, we can convert it into the duty cycle using Equation 4.2:

$$DutyCycle(\%) = Angle(\%) \times 0.6 + 20 \quad (4.2)$$

- Duty cycle into pulse value: We already have the duty cycle we need, now we need to convert it into the value to be written into the Capture Compare Register (CCR). This is the register that dictates the duty cycle of our PWM signal, and its value is referred to as “pulse” in this dissertation. This conversion is done with Equation 4.3:

$$Pulse = \frac{DutyCycle(\%) \times Period}{100} \quad (4.3)$$

The “Period” value dictates the period of the PWM signal in milliseconds.

Using these conversions, we can have the servo motors move to the angle we want. Because we know the range of these motors ($[0, 270]^\circ$), the firmware also

contains a function that makes sure that the duty cycle of the PWM signal always corresponds to angles inside this range.

To change these angles without having to change them directly in the code and recompile the whole project, the UART protocol was also added to the firmware at this stage. This type of communication in this dissertation does not require the use of very specific parameters. As such, the UART was configured using the most common values. Table 4.1 presents these values:

Table 4.1: UART parameters.

Parameter	Value
Baud rate	115200 bit/s
Word Length	8 bits
Parity	None
Stop Bits	1 bit

With the algorithm “as-is”, if we use these pulse values directly, the motors move at their maximum speed (about 0.18 seconds to rotate 60° with no load, according to the datasheet [DFRobot, 2018]). Because this apparatus was assembled using low-cost components, some factors that impact its behavior and performance. One of their impacts is residual motion. This is the motion of the apparatus after the motors have stopped. While it is caused by the inertia of the apparatus, it could be reduced by using components with lower manufacturing tolerances. But because these tend to be expensive, a different solution had to be found. One of the factors that can reduce this residual motion is the moving speed of the apparatus, so this was the chosen approach. Figure 4.11 shows the flowchart of this algorithm.

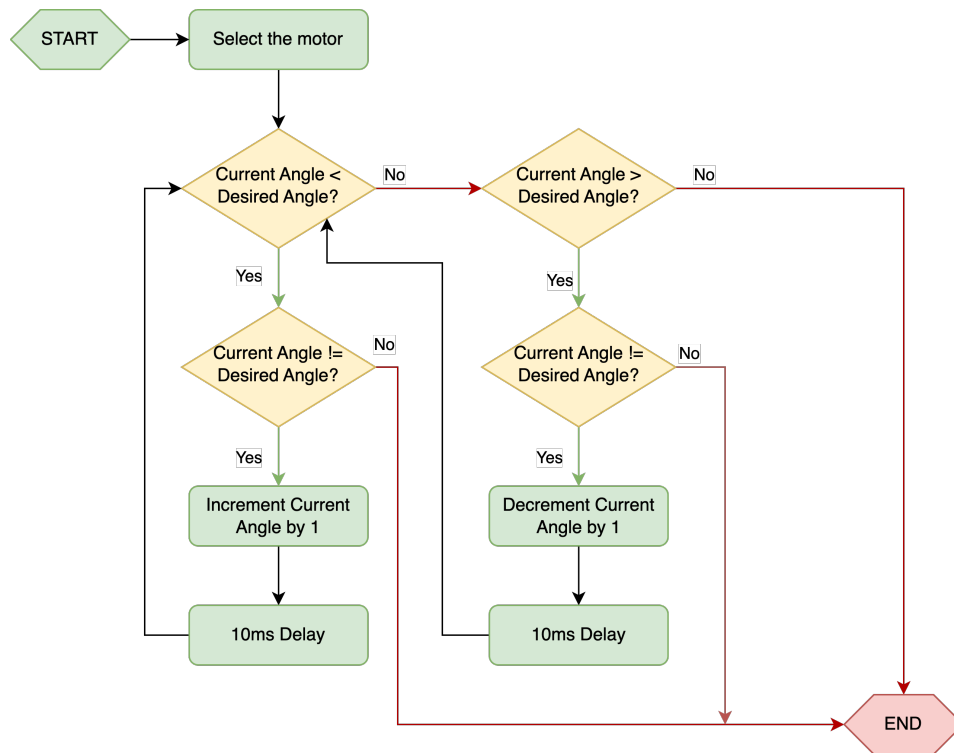


Figure 4.11: Flowchart of the motor sweep function.

Because we cannot control the speed of the motor directly, our solution consists in moving the motors to angles in between the current and the desired angles, with a delay between movements. This process is referred to as the “servo sweep” in this dissertation. Using this process, we can reduce the speed of the apparatus, and reduce the residual motion when it stops.

So far, we have discussed how we can move the servo motors to the desired angles. But as it stands, there is still no way of verifying if the apparatus has reached those angles. Because the servo motors we chose for this project have a feedback signal, we can use it to create a closed control loop that adjusts the PWM signal based on the “real” angle of the motor. This helps close the gap between the value we want and the “real” angle of the servos.

The solution that first came to mind was to create a closed control loop that uses the values read by the IMU’s to adjust the position of the apparatus. But developing a control algorithm that considers the setpoint of four servo motors, each feedback signal, the data from two IMU’s and the biomechanical limitations of the human body/apparatus is too complex and beyond the scope of this dissertation.

The chosen solution was to implement a Proportional-Integral-Derivative (PID) controller that continuously checks the feedback signal from the motors and corrects their duty cycle accordingly. Figure 4.12 shows a generic diagram block representation of a PID controller:

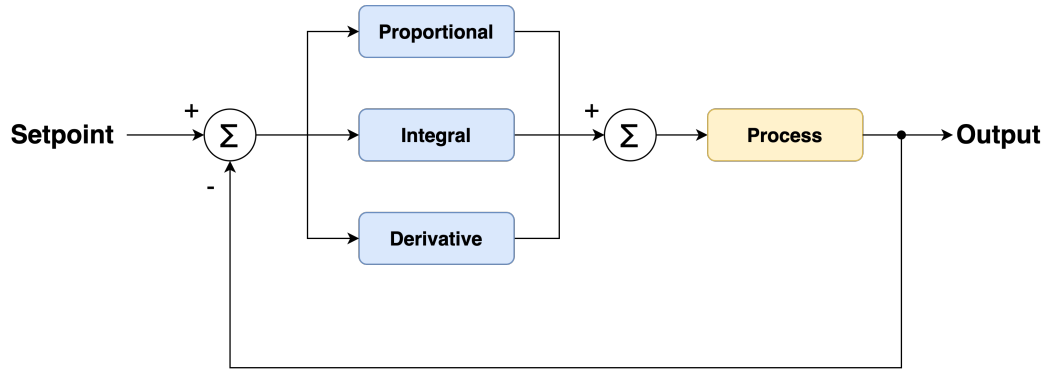


Figure 4.12: Generic PID controller.

We can use the following equation (Equation 4.4) to calculate the control signal:

$$\text{Control Signal} = P + I + D \quad (4.4)$$

With P (Equation 4.5), I (Equation 4.6) and D (Equation 4.7) calculated with the following equations:

$$P = K_p \times \text{Error} \quad (4.5)$$

$$I = K_i \times \text{Integral Sum} \quad (4.6)$$

$$D = K_d \times \frac{(\text{Error} - \text{Previous Error})}{\Delta t} \quad (4.7)$$

The “*Error*” is the difference between the desired and current angles ($\text{Error} = \text{Desired} - \text{Current}$) and the “*Previous Error*” is the “*Error*” from when this process was last run. “ Δt ” represents the time elapsed since that last run. The “*Integral Sum*” is calculated by multiplying the “*Error*” by “ Δt ” and adding to the previous value of “*Integral Sum*”, like this: $\text{New Integral Sum} = \text{Previous Integral Sum} + (\Delta t \times \text{Error})$. To prevent the “*Integral Sum*” from growing too large, a limit is implemented. The values until now are dynamically calculated in the firmware in each iteration. Figure 4.13 shows how a diagram of how this process is implemented in the firmware.

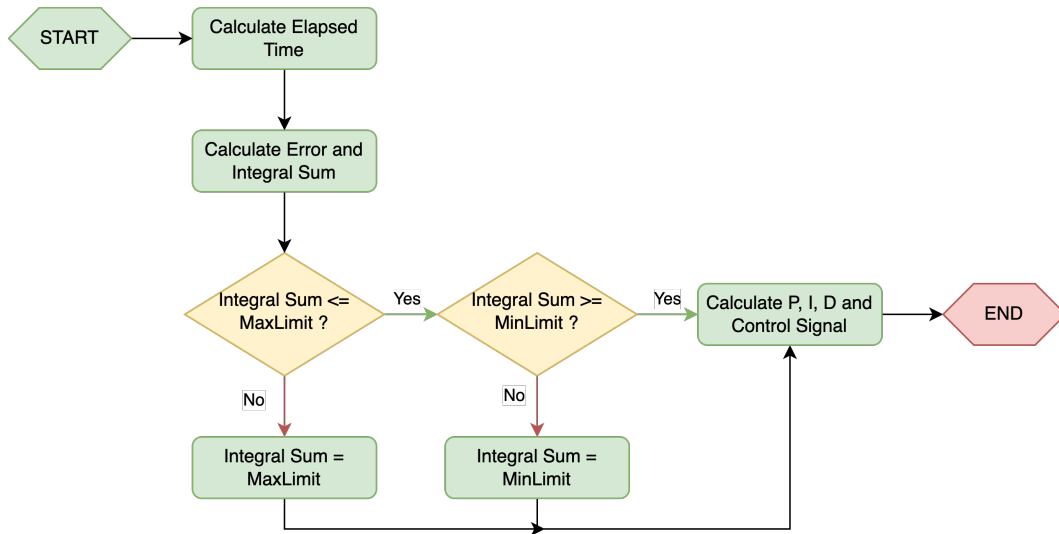


Figure 4.13: Diagram of the PID controller implementation in firmware.

There are three variables left to discuss: K_p , K_i and K_d . These gains are determined by us once and stay unchanged, and their value influences the behavior of the controller. Selecting the optimal values for each application is commonly referred to as “tuning the controller”, and was the next step of implementing this controller. We started by testing each component isolated (P , I and D) and tweaking each gain (K_p , K_i and K_d) while observing the behavior of the apparatus with each change. With this manual process, we found gains that have the PID controller perform as expected.

We have discussed how we can use the feedback from the motors to continuously adjust the duty cycle of the motors, but we still have to discuss how we read and process this feedback signal. As presented in Subsection 3.1.1, these feedback signals range from 0V to 3.3V. To be able to read them, the MCU has to convert these analog signals into digital data. For this, we have used the Analog-to-Digital Converter (ADC) available on our MCU. While the ideal scenario in this project is to have the apparatus move to the desired angle and stop, external factors like added weight could cause the apparatus to move undesirably. Because of this, it is desirable to periodically read the feedback signal from the motors. For accomplishing this, the ADC that reads the feedback signals is triggered using a hardware timer from the MCU. As a final improvement of this process, this data transfer is performed using Direct Memory Access (DMA). This way, this process can be autonomous and independent of the main processor. The complexity of this project does not require the use of DMA, but it was used as a learning opportunity. Figure 4.14 shows how these feedback signals are read and processed:

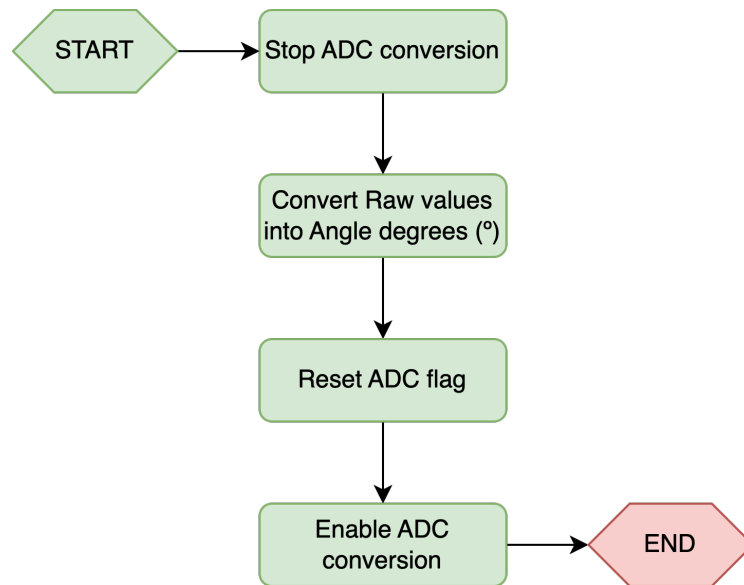


Figure 4.14: Flowchart of the process of reading and processing the feedback signals.

Whenever the ADC reads the feedback signals, this function is triggered using an “ADC flag”. This flag works like a synchronization mechanism that guarantees that this operation is only performed once for each value received. When the function starts, the ADC conversion is stopped so that the current values are not changed while in the function. After converting the raw values, the “ADC flag” that was used to trigger the function is reset, and the ADC conversion is re-enabled.

The used ADC has a resolution of 12 bits, which means it will read a value between 0 and 4095. But this value is meaningless if we cannot convert it into an angle. In Subsection 4.2.1, it is discussed how the raw ADC measurements are converted to an angle.

4.2.1 Motor’s Calibration

To know the “real” angle that each servo is at, we need to read the feedback signals they provide. As discussed in this section (Section 4.2), the value that the ADC reads ranges from 0 to 4095, but we still need to convert this value into degrees to compare it to the desired angle. The motors’ datasheet already provides a formula for calculating the angle using the analog value provided as feedback but advises against using it when the application requires precise control [DFRobot, 2018]. Because of the characteristics of this dissertation, we decided to find the conversion formula for our motors, to be more accurate with our readings.

The first approach was assuming that the relationship between these two variables was a linear function. The voltage that the ADC reads ranges from 0V to 3.3V, and the feedback signal from the motors have this same range. Because the

angle of the motor ranges from 0° to 270° and the feedback value ranges from 0 to 4095, the linear function that relates these two variables is presented in Equation 4.8:

$$Angle = \frac{270}{4095} \times Feedback \quad (4.8)$$

But upon testing this function, we noticed that it only works when the angle is between 100° and 120° . As such, this function cannot be used to reliably convert the feedback value into degrees.

The second approach was moving the motors to different angles, measuring the corresponding feedback value, and plotting these values to find a function. We decided to move the motors through their whole range in steps of 10° and record the feedback values for these angles, repeating this process for each of the four motors. But this process would be tedious if done manually and would take a long time. To expedite it, it was automated using Python.

This consisted in developing a script that moves the motors to a specific angle, reads the feedback values, and stores them in a .csv file. The script communicates with the MCU using the UART protocol, which was accomplished using the *pyserial* library [PySerial, 2023]. After reading the feedback signals, it stores all the data using *pandas*, a Python data analysis library [Pandas, 2023]. Because we have four motors, we can use the script to find the equation of each motor. The script takes this into account and performs the process described in this paragraph for each of the motors simultaneously. The user only needs to power the four servos, connect them to the MCU, connect the MCU to the computer, and run the script. It selects whichever port the MCU is connected to, moves the motors to each angle, and records all the feedback values. This script can be found in Appendix A.3.

After running this script, the data was plotted in graphs for each motor. The graph of one of the first servo motors is shown in Figure 4.15 as an example:



Figure 4.15: Graph that shows the relationship between feedback value and angle of the first servo motor.

Using linear regression, we found the equation that represents the relationship between the two variables. The data gathered by this script and the resulting graphs for each motor can be found in Appendix A.

There is one main characteristic that should be noticed: the first three angles (0° , 10° and 20°) have the same feedback value. This is a characteristic present in the four motors, and it could be a problem in this dissertation, as we need to be accurate with the angles we want to replicate. This hardware limitation poses the question: *how can we circumvent this problem?*

As this dissertation focuses on replicating a subset of human arm movement, the motors need to move within the ROM of the human arm. This ROM is different for every human joint but typically has a lower ROM than the servo motors (0° to 270°). For example, according to [Luttgens and Hamilton, 1997], the ROM of the elbow is about 140° , as seen in Figure 4.16:

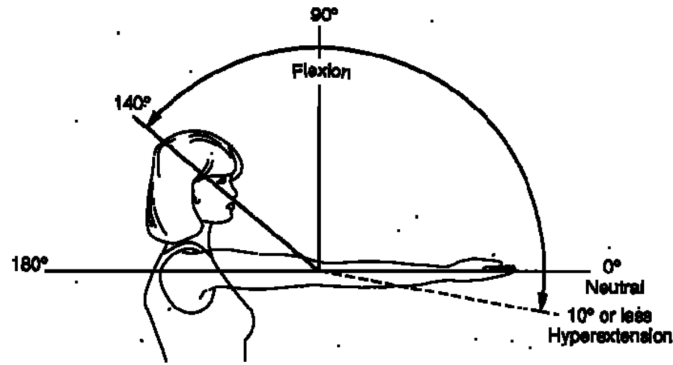


Figure 4.16: ROM of the human elbow [Luttgens and Hamilton, 1997].

Because the human joints' ROM are smaller than the servos', we can place the servos in such a way that they never need to reach 0° , 10° or 20° . This way, we avoid running into the lack of feedback accuracy when at those angles.

After getting the equations for each motor, we can process the feedback data and figure out what the “real” angle of each motor is. With the PWM's from the MCU, and the feedback signals from the servos, we can now control the apparatus and have it move to the desired angles.

4.3 Position Tracking

The feedback values of the servos are useful in this project because they allow us to create a closed control loop of their position. This results in the motors moving closer to the desired setpoint. But although it is possible to use only these values to find the spatial orientation of the apparatus, this process would be too complex. Using these angles to find the position of the apparatus in 3D space goes beyond the scope of this dissertation, so a different solution was used.

As discussed in Chapter 3, the position of the apparatus can be found using the IMU's placed on it. The selected models (LSM6DSO) contain both an accelerometer and a gyroscope. As discussed in Subsection 2.3.2, these sensors can be used together to improve the accuracy of the measurements. To use them together, some method of sensor fusion must be employed. Because analyzing the pros and cons of multiple sensor fusion algorithms is beyond the scope of this dissertation, the method used for implementing sensor fusion in this project was using a library.

Because the chosen MCU was developed by STM, the MotionFX library by the same company was chosen [STM, 2023a]. This library allows the use of multiple filters, such as complementary, Madgwick, Mahony, and Kalman, which is the one used in this dissertation. Using the acceleration and rotation values read by the IMU's, the MotionFX library outputs the fused data in various formats. One of

these formats is the quaternion, which is useful because we want to compare these values to the ones read by the MT's, represented also by quaternions.

But we still need to have the MCU read the values from the IMU's. As discussed in Subsection 3.1.2, these IMU's will be connected to the MCU using the I2C protocol. With this protocol, we need to know the IMU's address to communicate with them. The LSM6DSO offers two distinct addresses, which allows us to assign the address "0x6A" to the first LSM6DSO (placed on the PVC pipe between the "shoulder" and "elbow" joints) and the address "0x6B" to the second one (placed on the other PVC pipe).

After connecting the IMU's to the MCU via I2C, we needed to configure the accelerometer and gyroscope. Table 4.2 compares the characteristics of the MT's and the IMU's.

Table 4.2: MT's and IMU's parameters.

Sensor	Range	Output Data Rate (ODR) (Hz)
MT's Accelerometer	$\pm 2g$	225
MT's Gyroscope	$\pm 2000dps$	225
IMU's Accelerometers	$\pm 4g$	416
IMU's Gyroscope	$\pm 2000dps$	416

As a way of measuring extremes that these MT's cannot measure, we decided on configuring the accelerometer on the IMU to measure up to $\pm 4g$. This reduces the resolution, but the lack of these last digits will be taken into account when comparing the values from the IMU's and the MT's.

After setting up the sensors, we can read the acceleration and rotation measured by the IMU. Figure 4.17 contains a flowchart that represents the process of reading these values:

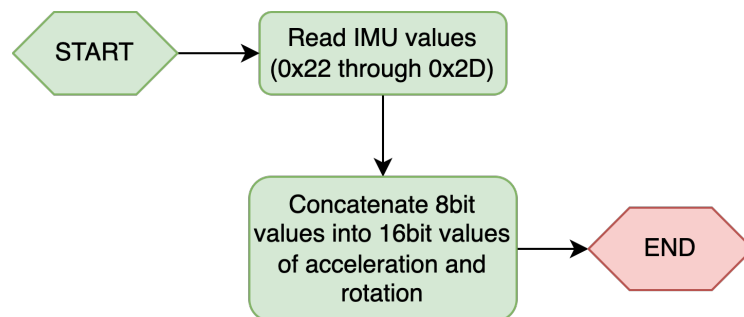


Figure 4.17: Flowchart of the process of reading the IMU.

The IMU's represent acceleration and rotation as 16bit values in two's complement, divided into two 8bit registers each. Table 4.3 shows the registers where the IMU's store this data.

Table 4.3: Registers of the IMU that contain the values of acceleration and rotation.

Description	Address	Description	Address
Accel_L_X	0x28	Gyro_L_X	0x22
Accel_H_X	0x29	Gyro_H_X	0x23
Accel_L_Y	0x2A	Gyro_L_Y	0x24
Accel_H_Y	0x2B	Gyro_H_Y	0x25
Accel_L_Z	0x2C	Gyro_L_Z	0x26
Accel_H_Z	0x2D	Gyro_H_Z	0x27

To test the configuration of these IMU's, the first version of the firmware had an algorithm that periodically read the registers. This option worked for testing purposes, but this pooling method would not work if the registers were updated while the MCU was reading them. To prevent this, the interrupt pins in the LSM6DSO breakout board were used. When the accelerometer has new data ready, the INT1 pin goes from low to high, which triggers the function in the MCU to read the registers. This way, because the function that reads the registers is faster than the Output Data Rate (ODR) of the sensors, the values inside the registers will not change while the MCU is reading them.

After reading these registers, the MCU must concatenate these 8bit values into 16bit to get the acceleration and gyroscope readings. Now, to use the MotionFX library, we need to convert these raw values into G (for the acceleration) and Degrees per Second (dps) (for the rotation). Figure 4.18 shows a flowchart that explains this process:

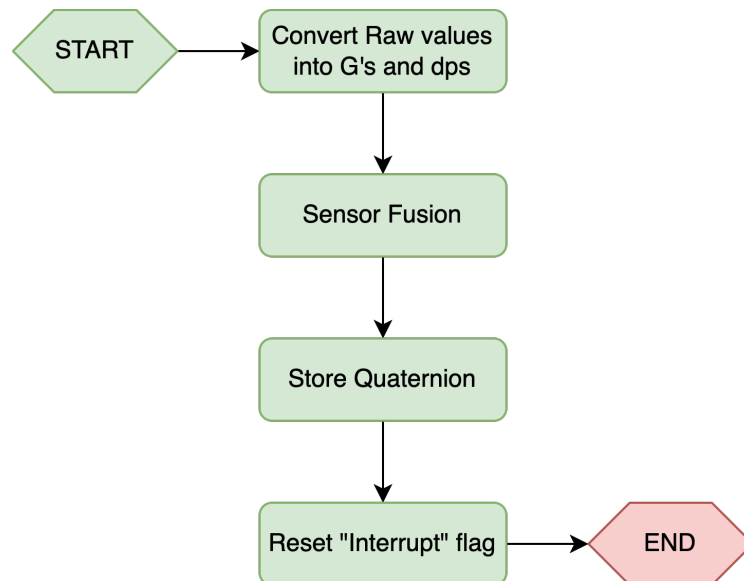


Figure 4.18: Flowchart of the procedure of processing the data from the IMU.

As we are using a range of $\pm 4g$ for the accelerometer, we can convert the raw value into G using Equation 4.9:

$$Acceleration(G) = \frac{Raw \times 4}{2^{(16-1)} - 1} \quad (4.9)$$

As for rotation, because we configured the gyroscope with a rotation range of $\pm 2000dps$, we can turn the raw values into dps using Equation 4.10:

$$Rotation(dps) = \frac{Raw \times 2000}{2^{(16-1)} - 1} \quad (4.10)$$

After these conversions, we can use the MotionFX library to fuse this data and output a quaternion that represents the spatial orientation and rotation of the IMU's and, by extension, of the apparatus.

At this point, we can already move the apparatus to the desired setpoint and read the values of the IMU's. These IMU's were placed on the PVC pipes using velcro strips, and their position was chosen to coincide with the position of the MT's. The "Interrupt" flag represented in Figure 4.18 is used to signal when the process function must be triggered, so it should be reset when the function ends.

For the user to change the angles of each motor, we considered developing a set of functions that can be triggered using specific commands. The next section discusses how this feature was implemented in this project.

4.4 Microcontroller Shell

There are various ways of implementing a communication channel between a user and the MCU, one of them being an embedded shell. This method consists of implementing a way for the user to send a command to the MCU, parse said command, and trigger the corresponding function. By implementing this shell, the user can do tasks like change the duty cycle of a PWM signal, check the feedback value of the motors, or even change properties like the frequency of the ADC. As long as the appropriate function and respective command are implemented, the user can change virtually any aspect of the firmware without needing to update and recompile the whole project.

This shell was implemented using the "*cli_embedded*" tool developed by [Farrelly, 2022]. This allowed us to start using this shell by importing only one header file. After setting it up, we only need to create a function, assign a command to it, and start using it. A user can interact with it using any serial monitor software, as the communication channel is implemented using UART.

For this project, five functions were added to this shell:

- *help*: This command displays all available commands, as well as a brief description of each.
- *info*: Prints the values read by the ADC on the MCU, and the “counter” value for each PWM signal.
- *imu_quat*: Prints the quaternion from the IMU’s. It has the following syntax: “imu_quat <x>”, where “<x>” is the number of the IMU that the user wants to read.
- *imu_accel*: Prints the acceleration values from the IMU’s. It has the following syntax: “imu_accel <x>”, where “<x>” is the number of the IMU that the user wants to read.
- *motors*: Changes the angles of the apparatus. The user must specify the desired motor and angle, using the following syntax: “motor <x> <y>”, where “<x>” is the motor and “<y>” is the angle. Each motor is represented by a number from “1” to “4”, and these numbers were assigned starting with the first motor near the fire extinguisher base.

Using this shell, we can provide the user with an Application Programming Interface (API) for the MCU. If the user needs to interact with the MCU in a way that is not implemented yet, the firmware can be updated to include it.

This shell allowed us to test multiple parameters easily, without needing to update the firmware. But there are some downfalls when it comes to using it as the user interface for the whole project:

- *Middleware*: To use this shell, the user must search and choose some middleware that allows communication via the serial port. This requires extra work on the side of the user, as well as knowledge about the hardware (e.g. UART parameters).
- *Motion Trackers*: This topic will be further discussed in Section 4.5, but the added complexity of reading the data of the MT’s and presenting it specifically on the shell goes beyond the scope of this dissertation. Because we need to read these values and compare them to the IMU’s, using only this shell is not a solution.

Figure 4.19 presents the sequence of the firmware after integrating this shell:

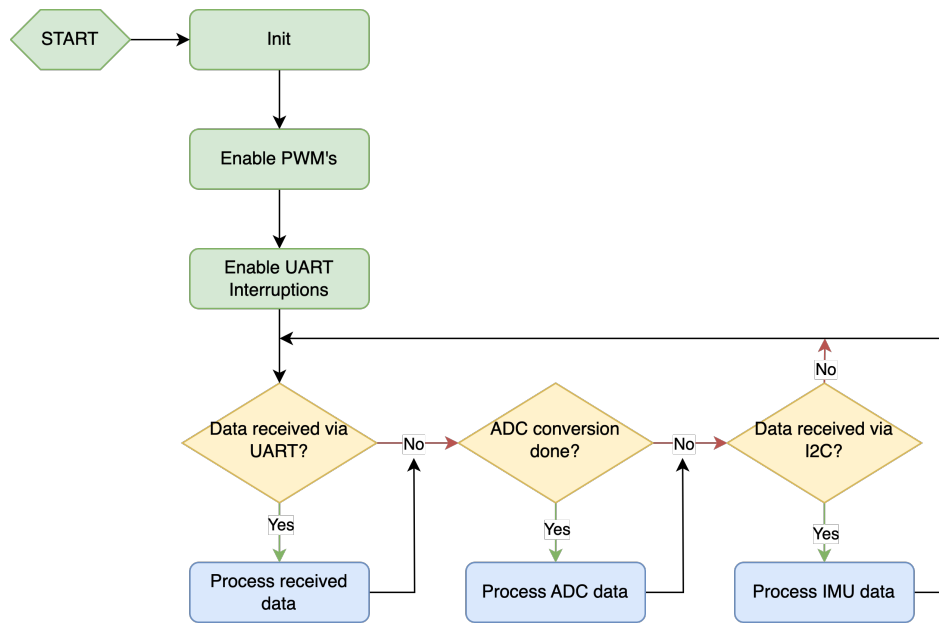


Figure 4.19: Flowchart of the firmware after implementing the shell.

After initializing all the components used (on the MCU and the peripherals), the firmware enters a loop where it checks if a new command was received through the shell, if the ADC read new data, and if the IMU's sent new data.

This shell is a valuable tool for testing the behavior of the apparatus and changing some parameters, and this is what led to it being implemented and kept in this project. But it cannot be used as the only interface between the user and the apparatus. How can we overcome these limitations, and improve the user experience? How can this other solution allow the user to check the values read not only by the IMU's but also the MT's, and compare them? Both of these questions are answered in the next section.

4.5 User Interface

On top of solving the limitations mentioned in the previous section, a user interface helps abstract the user from the underlying complexity of the firmware. As user-friendliness is one of the main goals of this dissertation (see Section 1.2), developing a user interface can help us accomplish this goal.

This user interface must perform the following actions:

- Move the apparatus: This interface should be the only one needed to interact with the apparatus. As such, we must implement a method for moving the apparatus.

- Read the IMU's: The user has to be able to compare the values of the IMU's and the MT's. To do this, the interface must be able to access the values read by the MCU.
- Read information from the MTs': For the same reasons presented for the previous item, the interface must be able to read the values of the MT's.

As we have already implemented the MCU shell, and the functions needed to move the apparatus and read the IMU's, we can use those commands to perform the first two items of this list. To interact with the shell, we must establish a connection between the interface and the MCU using UART. There are multiple ways of implementing this channel, but the first option considered in this dissertation was using Python.

As discussed in Subsection 4.2.1, there is a Python library called *pyserial* [Py-Serial, 2023] that allows us to establish this type of connection. Using this library, the interface can process the inputs given by the user and output the corresponding commands for the shell. Another advantage of using Python is the MT Interface, a tool developed by SWORD that is written in Python. It allows us to interact with their MT's and read various parameters (e.g. spatial orientation, battery). There are other advantages of using Python (e.g. effortless string parsing), but the two reasons presented here are the ones that led to choosing Python for developing the user interface.

After choosing the programming language, we started developing the needed features. The first step was establishing communication between the interface and the MCU. Using the *pyserial* library, the interface automatically finds the port where the MCU is connected to, using its Vendor ID and Product ID. These are parameters used by the USB protocol, which we can use to identify which device is connected to the computer, and in which port. After finding this port, the interface connects to the MCU and is ready to send commands to the shell. Using the shell, the interface can move the apparatus to the desired setpoint and also read the values from the IMU's. The next step was connecting the interface to the MT's.

As presented in Chapter 3, the MT's communicate using Bluetooth. To communicate with them, the MT Interface from SWORD (from now on, referred to simply as "MT Interface") requires the use of the nRF52840 USB dongle from Nordic. This component is represented in Figure 4.20:



Figure 4.20: nRF52840 USB dongle from Nordic [Nordic, 2023].

The use of this dongle allows the MT Interface to communicate with the MT's on any computer, without needing to deal with various Bluetooth configurations.

The first step in setting up the MT interface is to find the port that the dongle is connected to, using the same process used for finding the MCU. After finding the dongle, we can set it as a central for the Bluetooth communication.

The MT Interface reads the data from the MT's using a subscription model. Figure 4.21 shows the process needed to read quaternions from the MT's:

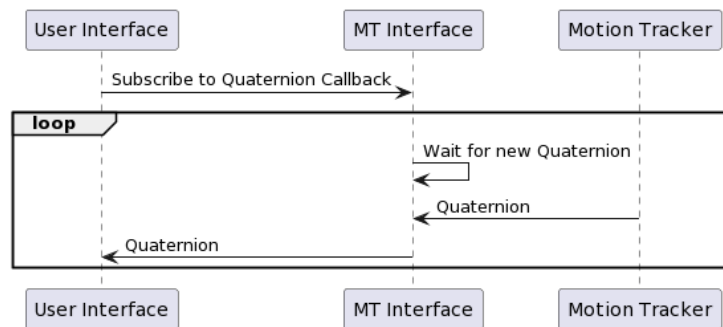


Figure 4.21: Reading quaternions from the MT.

Until the interface unsubscribes this callback, the MT Interface will keep sending the quaternions from the MT's. We can process these values as needed, on the interface side. The callbacks used by the MT Interface run on a separate thread, which makes them a non-blocking process. This means that the program can execute other tasks while the MT Interface waits for new data.

After adding support for the MT Interface, the user interface can read quaternions from both the MT's and the IMU's, and move the apparatus. As it stands, the user can only move the apparatus by typing the angles for each motor individually. As discussed in Subsection 3.3.1, one of the interactions that the user should be able to perform is to select a preset that contains predetermined angles for all motors. This feature was implemented by storing all the presets in a text file using

the JavaScript Object Notation (JSON) data format. Figure 4.22 shows an example of the contents of this file, called “presets.json”:

```
{ } presets.json > ...
1  {
2  "1": {
3      "1": 135,
4      "2": 135,
5      "3": 135,
6      "4": 135
7  },
8  "2": {
9      "1": 135,
10     "2": 135,
11     "3": 250,
12     "4": 135
13  }
14 }
```

Figure 4.22: Contents of the “presets.json” file.

Each object represents a different preset, and the angles for each motor are assigned to a number that stands for each motor. Whenever the user wants to edit the list of presets available, the user interface edits this file. The examples shown in Figure 4.22 only have numbers as names, but the interface allows the user to assign any name to these presets. The interface also supports creating new presets, editing the ones already on the file, or even deleting them.

Using the features discussed until now, the final step was assembling them into one user interface. Figure 4.23 presents the starting sequence.

After connecting the USB dongle and the MCU to the computer, the user can start the interface. After it finds the USB dongle and the MCU, the interface scans available Bluetooth devices for two seconds, and presents a list of the devices it found to the user. Figure 4.24 contains the text shown to the user at this point:

```
Device 1: SH00REB1TKA
Device 2: SH00REB8H1H
Name of the device(s) to use (if multiple, separate with spaces):
```

Figure 4.24: User Interface prompt for selecting the MT’s.

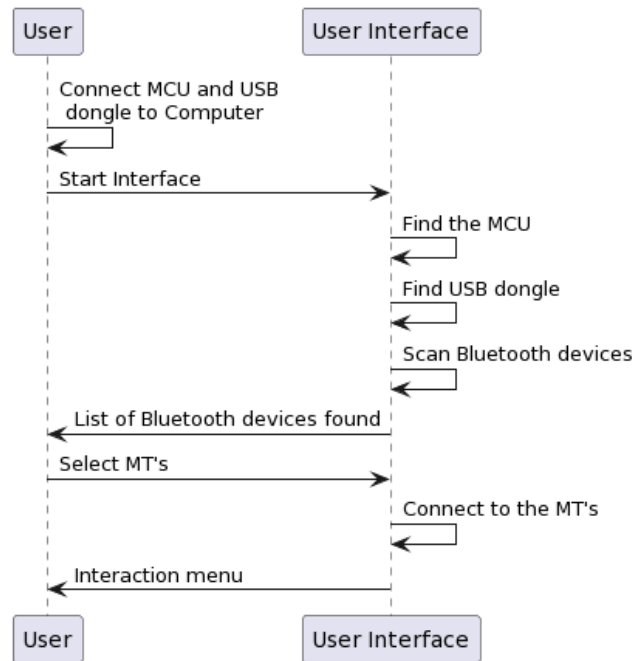


Figure 4.23: Starting sequence of the user interface.

When the user gets this prompt, they can select any MT from the list. This dissertation focuses on testing two MT's at the same time because this is the scenario found in a physical therapy session, but the interface allows the user to test only one MT if this is desired. If no MT's are found, the user is prompted to start a new Bluetooth search. When selecting the MT's, the user should first write the name of the MT placed on the "upper arm" of the apparatus, and then the name of the MT placed on the "forearm". The first MT is assigned the number "1", and the second one is assigned the number "2".

After selecting the MT's to be tested, the user can choose one of the five options from the menu:

- Manually select the angles: This option should be selected when the user wants to select the angle of each motor individually.
- Select a preset: If the movement that the user desires is already stored as a preset, this option can be used for selecting one of them.
- Edit the presets: If the user wants to edit, add or remove presets from the list, this is the option that should be selected.
- Run Tests: When the user just wants to repeat the tests, this is the option that should be selected.

- Quit: If the user wants to quit the program, this option disconnects the MT's from the USB dongle. This way the MT's are no longer bound to the user interface and can be used for other purposes.

Figure 4.25 shows how these options are presented to the user:

```
[Name of the device(s) to use (if multiple, separate with spaces): SH00VEKT1HC SH00VEK3A56
*****
Available actions:
1 - Manually select the angles
2 - Select a preset
3 - Edit the presets
4 - Run Tests
5 - Quit
***** Which action would you like to perform? (Type the number): █
```

Figure 4.25: User Interface prompt for selecting how the user wants to interact with the apparatus.

4.5.1 Manually Selecting the Angles

If the first option is selected, the user can insert the angles desired for each motor, or leave them unchanged. Figure 4.26 presents the prompt shown to the user at this stage:

```
[Which action would you like to perform? (Type the number): 1
Type the desired angle (If you don't want to change an angle, press <Enter>).

[Angle of motor 1: 120
[Angle of motor 2: 30
[Angle of motor 3:
[Angle of motor 4: 270
```

Figure 4.26: User Interface prompt for selecting the angles for each motor.

The interface checks if the angles inserted by the user are valid, and if they are inside the range of the servos. After selecting the angles, the interface sends these values to the MCU shell using the “*motors*” command, to change the angle of each motor. After the apparatus moves, the interface sends the command “*imu_quat*” to the shell to get the quaternions from both IMU's. These values are stored by the interface, which then uses the MT Interface to subscribe to the quaternion callback for each MT. The MT's are read for one second, and the interface calculates the average of all these values and stores it as just one quaternion, one for each MT. These quaternions are presented to the user, as well as the quaternions from the IMU's. Figure 4.27 shows how these quaternions are presented by the interface:

```

MT1 Quat: [0.46503668989277175, 0.0006482694472503684, 0.00829045225092082, 0.8852521820583482]
MT2 Quat: [0.729310156777697, -0.012500856342369105, 0.6840651300819809, 0.002271766111498939]

MCU Quaternions:
IMU 1:
    Quat: [-0.311985 0.846767 -0.259110 -0.344259]

IMU 2:
    Quat: [0.682317 0.597187 0.256583 0.334630]

```

Figure 4.27: Quaternions presented in the user interface.

4.5.2 Preset Selection

After this process, the user is sent back to the prompt presented in Figure 4.25. If the user selected the second option, a list of the available presets would be displayed. Figure 4.28 shows the options presented to the user when this option is selected:

```

Which action would you like to perform? (Type the number): 2

Preset 1:
    Angle 1: 135
    Angle 2: 135
    Angle 3: 135
    Angle 4: 135

Preset 2:
    Angle 1: 0
    Angle 2: 0
    Angle 3: 0
    Angle 4: 0

Select one preset (Press <Enter> to go back): █

```

Figure 4.28: User Interface prompt for selecting one of the available presets.

The user can select one of the presets available on the list by typing the number associated. When the user selects a preset, the interface identifies the angles of each motor and uses the “*motors*” command of the MCU shell to move the apparatus. After the apparatus reaches the setpoint and stops moving, the process of reading the MT’s and the IMU’s is the same as the one used when the angles are chosen manually.

4.5.3 Edit the Presets

The third option that the user can select is “Edit the presets”. If the user selects this option, this is the screen that is shown (Figure 4.29):

```
Preset Editor:

Preset 1:
    Angle 1: 135
    Angle 2: 135
    Angle 3: 135
    Angle 4: 135
Preset 2:
    Angle 1: 0
    Angle 2: 0
    Angle 3: 0
    Angle 4: 0
Preset 3:
    Angle 1: 100
    Angle 2: 100
    Angle 3: 100
    Angle 4: 100
Preset 4:
    Angle 1: 200
    Angle 2: 200
    Angle 3: 200
    Angle 4: 20

Other Commands:
    'new': Create a new preset
    'remove': Delete an existing preset
    <Enter> key: Go back
Select one preset to edit (or a command): █
```

Figure 4.29: User Interface prompt for editing the available presets.

There are four options inside this submenu:

- *edit*: By typing the number (or name) of a preset, the user will be able to edit the angles of that preset.
- *new*: This option allows the user to add a new preset to the list.
- *remove*: If the user wants to remove one of the presets from the list, they should select this option.
- *<Enter>*: This option takes the user back to the main menu.

After using one of the first three options, the “presets.json” file is edited to either update one of the presets, add a new one, or remove an existing one.

These are the three use cases of the user interface that allow the user to move the apparatus. As it stands, this interface allows the user to select the MT's to be tested and the position of the apparatus. After selecting these parameters, the user interface takes care of sending commands to the MCU for moving the motors and reading the quaternions from the IMU's. The interface is also capable of reading the quaternions from the MT's and presenting all these values to the user. Using this feature, we can test the MT's. This option will be discussed in Chapter 5.

4.6 Final Remarks

The design choices discussed in Chapter 3 proved to be adequate for building a prototype of the apparatus. The parts available for purchase were integrated into the apparatus without much fiddling, and the 3D pieces were developed specifically to fit this project.

The use of an electrical interface similar to an Arduino shield for connecting the MCU to the apparatus proved to be a valuable feature, as it made the process of updating the firmware easier. This was especially true during the development of this project, as it allowed us to develop and test certain components of the firmware without having to carry the whole apparatus. Figure 4.30 presents the final look of our prototype.

As for the development of this firmware, the main Integrated Development Environment (IDE) used was the STM32CubeIDE, developed by STM [STM, 2023b]. The choice of this IDE is remarked here because it allowed us to use the CubeMX framework, which takes care of the configuration of the peripherals used in this project [STM, 2023c].

The development of a MCU shell was also helpful, as it allows the creation of commands that let a user change the parameters of the MCU without having to update and recompile the firmware. This allows the creation of a test environment tailored specifically for the MCU, that can be used to test any parameter the user desires (as long as the respective commands are implemented).

Finally, we managed to simplify the user interface from the user's perspective so that they are guided through all the necessary inputs while abstracting the complexity running underneath. This creates a user-friendly interface, while still retaining all the desired features (see Section 1.2).

We have already chosen the necessary components and assembled the apparatus, as well as developed the firmware and the necessary tools. The next step is testing this solution, and measuring its behavior. This is the topic discussed in Chapter 5.



Figure 4.30: Final prototype of the apparatus.

Chapter 5

Tests and Validation

The apparatus has been assembled, an MCU shell has been developed for interaction with the MCU, and the user interface has been implemented. Now that we have discussed all these components, how they work, and how they interact with each other, there is just one topic left to discuss before using the apparatus.

5.1 Limitations

Although the appropriate measures were taken during the development phase, there are still limitations that impair the performance of the apparatus. The apparatus mostly performs as expected, and meets nearly all of the goals discussed in Section 1.2, but some caveats must be considered when discussing the tests performed and the results they produced. This section presents these limitations to the reader, as well as some solutions that minimize their impact.

5.1.1 Lack of Torque in the First Joint

The DSS-M15S servo motor was selected for this dissertation because of its high torque at its cost range (see Subsection 3.1.1 for the discussion about this topic). During individual testing, these motors seemed to be an adequate choice because their torque was more than enough for moving the servo brackets and even the PVC pipes. But after assembling the whole apparatus, we noticed that the first and second motors (the “shoulder” joint) were not able to lift the “elbow” joint beyond a specific setpoint without external help (e.g. the user helping the “arm” move past

that setpoint). Figure 5.1 illustrates the discrepancy between the expected and real ranges of the first motor:

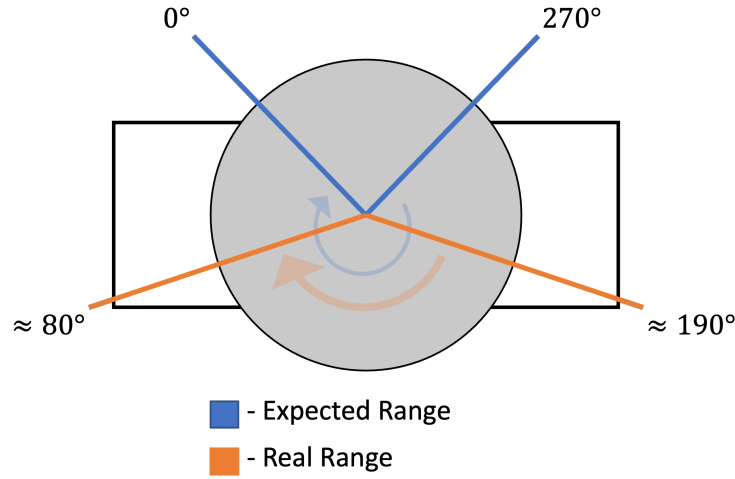


Figure 5.1: Range of the first motor.

Despite the high torque of these motors, the distance between them and the “elbow” joint proved to be too great for the first and second motors. This limits the ROM of these motors and reduces the subset of possible movements.

To circumvent this limitation, the feedback signal that these motors provide can be used to alert the user to this limit. For accomplishing this, if the feedback signal of the motors does not match the desired setpoint in a certain amount of time, the user is alerted. This way, if the desired angle is outside the range of the motors, or if something went wrong when moving the apparatus, the user can understand why the desired setpoint is not being reached.

5.1.2 Feedback Signal Noise

As discussed in Subsection 3.1.1, the servo motors also offer a feedback signal that allows the MCU to know the actual position of the motor. This signal is read by the MCU using its ADC, but this signal is only converted from analog to digital at the MCU, which means that this signal is analog from the motors to the MCU. Some of the cables that carry this signal have lengths of over one meter, and there are twenty-six cables in the apparatus placed side by side. On top of this, some of these cables carry power and the PWM control signal (which has very energetic transitions) to the motors. This results in the deterioration of the analog signal, caused by Electromagnetic Interference (EMI). Because of this interference, the values read by the MCU are noisy and can vary even when the apparatus is not moving.

The control algorithm uses this feedback signal as a way of checking the current position of the motors. Because this feedback is noisy, the control algorithm is always moving the motors to correct their position. This results in the motors

always moving, even when the user wants them to be still. Because some of the tests that we want to perform require the apparatus to be still, this jitter has to be eliminated somehow.

One of the solutions we tried to implement was using the “oversampling” function made available by our MCU. This function has the ADC read multiple values, calculate the average, and use this average as if it was the only value read by the ADC. However, even with this solution, the signal still fluctuates too much to be reliable. This fluctuation made the PID controller unviable for this project, and so a different solution had to be found.

The chosen solution involves using a counter inside the control algorithm. Figure 5.2 shows the updated control algorithm.

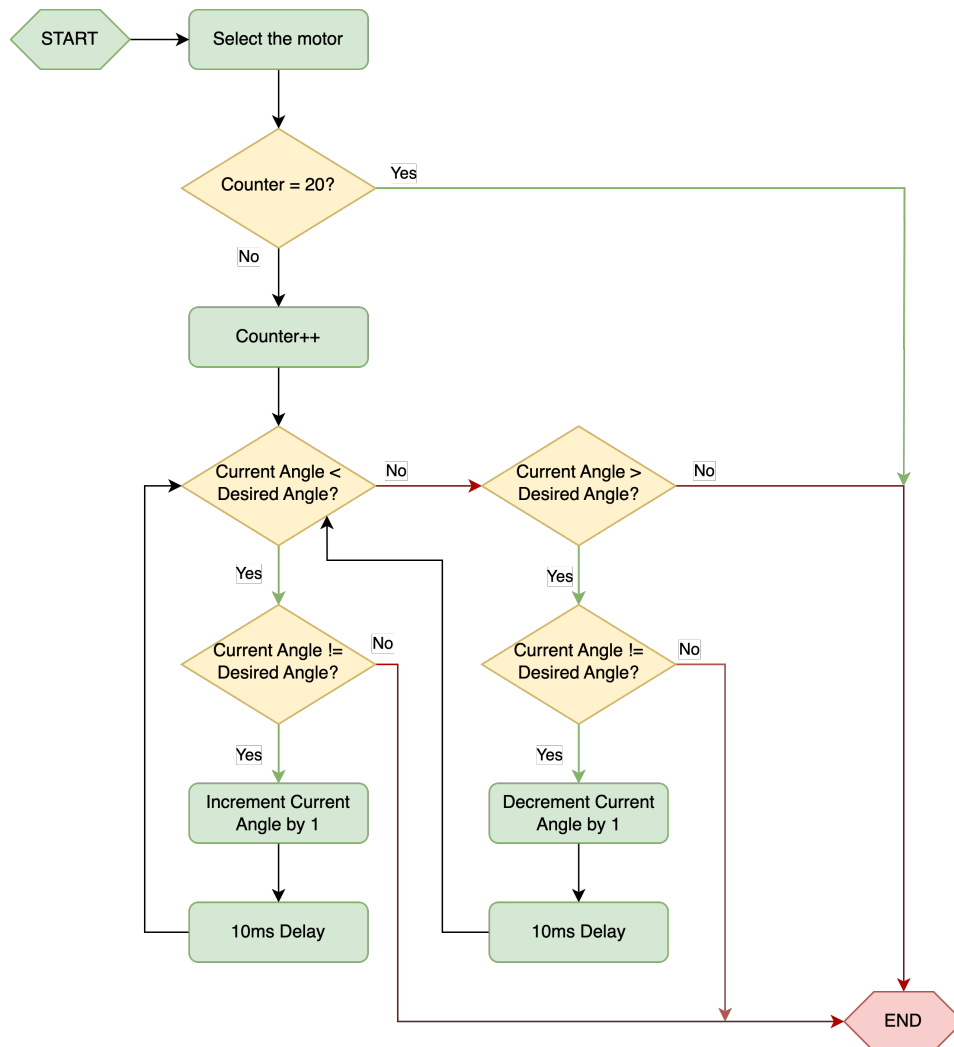


Figure 5.2: Updated control algorithm, now with the counter.

This counter is used to keep track of the number of iterations of this function. After a set number of iterations, this control algorithm will stop running. After

testing various values, we decided to settle on “20” because it is a good compromise between time to settle and angle accuracy. This helps make sure that the motors do not keep jittering as a result of the noisy feedback signal. However, this means that this is no longer a closed control loop because it does not run continuously.

There are other solutions that allow us to “clean” this noisy feedback signal and keep on using a closed control loop. These solutions are discussed in Section 6.2.

5.1.3 Drift Influencing the Sensor Fusion

To fuse the data from the accelerometer and gyroscope of the LSM6DSO IMU’s placed on the apparatus, we used the MotionFX library because it was developed by STM, the same company that produces the MCU used in this dissertation (see Section 4.3 for the discussion on this topic). The main goal for this fusion was getting quaternions from the IMU’s that could be compared to the MT’s to verify their behavior.

Despite being able to output these quaternions, we noticed that the values tended to drift as time went on because of gyroscope drift (see Subsection 2.3.1 for more on this topic). This meant that the quaternions produced by the first conversions were accurate, but the next conversions produced values that did not match the real position of the apparatus. Because of this, we could not use these quaternions to verify the behavior of the MT’s. This limits the type of tests that can be performed with our apparatus. We wanted to use this apparatus to check if the quaternions presented by the MT’s were correct, and so we implemented a different method for getting quaternions that represent the position of our apparatus.

The chosen solution consists in using MT’s that have been already verified to be working properly. Before performing tests with the target MT’s, we perform the tests with the functional MT’s and record their quaternions. After this, we repeat the tests with the target MT’s, read their quaternions, and compare them with the correct quaternions. This test is further discussed in Subsection 5.2.2.

5.2 Tests Performed

After identifying and discussing the limitations of this apparatus, we started testing its behavior. The first step in this testing process was checking if the components were working correctly by themselves, and if the communication between them was well implemented. Subsection 5.2.1 discusses the first tests performed and their results.

5.2.1 Functional Tests

Before delving into the tests with the MT's, we needed to make sure that the apparatus worked as intended. This also included the MCU shell and the user interface that was developed for interacting with the apparatus.

The method developed for interacting with the MCU was the shell discussed in Section 4.4, and so it follows that before testing anything else, we had to make sure that this shell was working properly. Table 5.1 shows the tests performed:

Table 5.1: MCU shell functional tests.

Function Tested	Working Status
"help"	✓
"info"	✓
"imu_accel"	✓
"imu_quat"	✓
"motors"	✓

The description of all these functions can be found in Section 4.4. To communicate with the shell and test these commands, we used a serial communication application on our host computer. To test the "help" and "info" commands, we sent these commands and verified that the output matched our expectations (i.e. function descriptions and the motors' feedback). To test the "imu_accel" and "imu_quat" commands, we manually compared the output to the expected and found that these also matched our expectations. The "motors" command was tested by observing if the motors reached the desired setpoint after the command was sent. As presented in Table 5.1, all these functions are accomplishing their intended goals.

Using this shell, we started testing if the MCU was able to interact with the peripherals. Table 5.2 shows the result of these tests.

Table 5.2: MCU functional tests.

Function Tested	Working Status
Configure the IMU's	✓
Read values from the IMU's	✓
Move the motors	✓
Read the feedback signal	✓

As presented in Table 5.2, the MCU can interact with the peripherals as intended. We tested the IMU configuration by validating the values they presented; this process

allowed us to verify that the first two functions from the Table above were functioning correctly. Using the shell, we verified that we could also move the motors. Finally, we were able to read the feedback signal sent by each motor.

After making sure that the MCU shell was working properly and the MCU was able to control the apparatus, we began testing the features available in the user interface. Table 5.3 shows the result of these tests:

Table 5.3: User interface functional tests.

Function Tested	Working Status
Move the motors	✓
Manually select the angles	✓
Choose an available preset	✓
Edit an existing preset	✓
Create a new preset	✓
Read values from the IMU's	✓
Read values from the MT's	✓
Quit	✓

All the basic operations from the user interface work as expected. Having tested the MCU, the MCU shell, and the user interface, we could properly define the actual range of each motor. Table 5.4 presents these ranges:

Table 5.4: Ranges of the motors.

Motor	Range (°)
1	$[\approx 80, \approx 190]$
2	$[135, \approx 220]$
3	$[0, 270]$
4	$[0, 270]$

The range of motor “2” ends at $\approx 220^\circ$ because of the limitation discussed in Subsection 5.1.1, but the 135° limit is imposed by the firmware. This was done to prevent the “arm” from bumping into the fire extinguisher base.

At this point, we have verified that the components of the apparatus work as intended (given its limitations), as well as the MCU shell and the user interface. We are now ready to start performing tests on the MT's.

5.2.2 Motion Tracker Tests

Now that the apparatus is working as intended, we can start using it to accomplish one of the goals set in Section 1.2: testing the MT's and the respective algorithms. To assess if these tests can correctly identify any problems, they were performed

on eleven MT's in total: two completely functional MT's, and nine MT's with different problems. This Subsection discusses what tests were implemented, the motive behind them, and their results.

The first test implemented for the MT's is done even before the apparatus starts moving. One of the characteristics that should be checked on the MT's is the acceleration module when at rest. This module can be calculated using Equation 5.1:

$$\text{Acceleration Module} = \sqrt{x^2 + y^2 + z^2} \quad (5.1)$$

If the MT is well configured, this value should always be $\approx 1g$ when the MT is at rest. After the user chooses which MT's to test, the user interface connects to them and performs this test. The interface can present four different outcomes to this test:

- “ERROR: Accel_module $\neq 1g$ on both MT's when resting, therefore these MT's are defective.”
- “ERROR: Accel_module $\neq 1g$ on the first MT when resting, therefore this MT's is defective.”
- “ERROR: Accel_module $\neq 1g$ on the second MT when resting, therefore this MT's is defective.”
- “SUCCESS: Accel_module = $1g$ on both MT's. All MT's are functional.”

After this, we need to verify if the MT axes are correctly aligned. For the second test, we want to align one axis of the MT with Earth's gravity and check if the acceleration read by the MT on that axis has the correct sign. Figure 5.3 shows how the MT axes should be oriented:

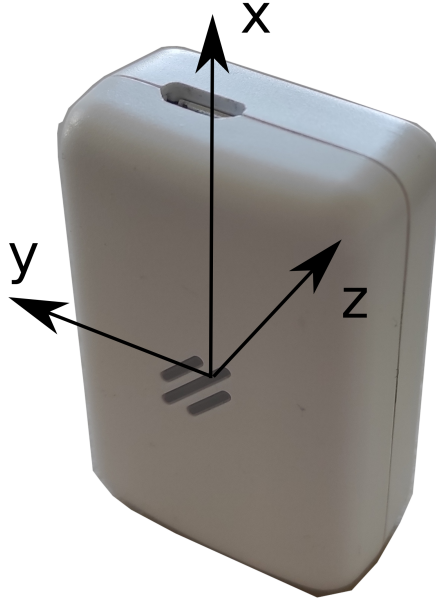


Figure 5.3: Axes of the MT's.

Because of the way that the MT's are placed on the apparatus, we can accomplish this by having the apparatus at its “resting position” (see Figure 4.9). In this position, the MT's must read an acceleration of $\approx -1g$ in the X axis. To test the other axes, we have to move the apparatus to positions that align each axis with Earth's gravity. The user interface can present these four outcomes to the user (“ $\langle x \rangle$ ” identifies which MT is being tested):

- “The X axis of MT $\langle x \rangle$ is not correctly aligned.”
- “The Y axis of MT $\langle x \rangle$ is not correctly aligned.”
- “The Z axis of MT $\langle x \rangle$ is not correctly aligned.”
- “All axes of MT $\langle x \rangle$ are correctly aligned.”

The third test consists in moving the apparatus to a position that decomposes gravity along the three axes of the MT's. After the apparatus is at this position, the acceleration values from the MT's are read and compared with the IMU's. If they match (within a certain threshold), the test is considered a success.

The fourth and final test checks if the quaternions presented by the MT's are correct. The apparatus moves to the same position used in the third test, and the quaternions are checked using the method described in Subsection 5.1.3. We tested each target MT to check if they presented an accurate quaternion.

Table 5.5 shows the result of these tests, performed on all eleven MT's:

Table 5.5: Results of the four tests performed on the test group.

MT Identifier	Test 1 Result	Test 2 Result	Test 3 Result	Test 4 Result	Affected Data	Observations
1	Pass	Pass	Pass	Pass	-	Functional MT
2	Pass	Pass	Pass	Pass	-	Functional MT
3	Fail	Pass	Pass	Fail	Acceleration and Quaternion	As expected
4	Pass	Pass	Fail	Fail	Acceleration and Quaternion	As expected
5	Fail	Pass	Fail	Fail	Acceleration and Quaternion	As expected
6	Pass	Fail	Fail	Fail	Acceleration and Quaternion	As expected
7	Pass	Fail	Fail	Fail	Acceleration and Quaternion	As expected
8	Pass	Pass	Pass	Fail	Quaternion	As expected
9	Pass	Pass	Pass	Fail	Quaternion	As expected
10	Pass	Pass	Pass	Fail	Quaternion	As expected
11	Pass	Pass	Pass	Fail	Quaternion	As expected

The first and second MT's are the ones that had been previously verified. After performing the tests on the remaining MT's, the results confirmed that our apparatus can be used to evaluate their behavior when it comes to their acceleration module, the alignment of the axes, the acceleration values read by each MT, and their quaternions.

After performing the functional tests, we can be sure that the apparatus is working as expected (given its limitations). The tests performed on the MT's also showed that this apparatus can be used to effectively test different characteristics of the MT's. Despite only having tested four different scenarios, more could be added to the user interface. This will be discussed in Section 6.2.

Chapter 6

Conclusion

We have now tested the functionalities of all the components that make up the prototype of this dissertation: the physical apparatus, the MCU shell, and the user interface. We have also implemented three different testing scenarios that show the value this apparatus brings to the QA procedures that test the behavior of SWORD's MT's and the respective algorithms. Looking at the four main goals presented in Section 1.2 and the apparatus that is discussed in this dissertation, we can confirm that these four goals were successfully achieved.

6.1 Summary of the Work Developed

We started this dissertation by defining the objectives we desired to accomplish. These objectives were related to different aspects of this dissertation, ranging from the desired features to the cost of the materials. But the main desired outcome was developing a method that would be able to replicate human movement, to test SWORD's MT's.

To get a better understanding of this problem and the available solutions, we looked into the state of the art in the field of biomechanics. Our research showed us that no commercially available solution would be capable of accomplishing all our goals (especially at a low cost), so we started developing our apparatus.

The first step in the development process was selecting the components that would best suit our goals. This entailed selecting the IMU's, the motors, the MCU, the power supply, and all the physical components that make up the apparatus. Not

all these components were available for purchase during this process, so some were designed and printed using a 3D printer.

After acquiring all the needed components, the apparatus was assembled. To interact with the apparatus, an MCU shell was created. This shell allows the user to communicate with the MCU using any software that allows serial communication between a computer and, in this case, the MCU. Multiple functions were implemented in this shell, allowing the user to move the motors or read the values from the IMU's. This shell is useful in a scenario where the user only needs to move the apparatus, but to talk with the MT's and improve the user experience, we needed to create another solution.

This was the motive behind the creation of the user interface. Written in Python, this interface can connect to the MT's using the MT Interface developed by SWORD, and allows the user to not only move the motors but also use a preset system. This system allows the user to select a preset with predetermined angles for each motor, and the list of presets can be edited by the user to include whichever presets are needed.

After implementing the apparatus, its functionality was verified by performing functional tests on its various features. Using these tests, we were able to show that every feature implemented on the MCU shell works as intended, as well as the options implemented in the user interface. We were also able to define what the ROM of the apparatus is.

At this point, the only step left was using the apparatus to test multiple MT's. Four different test scenarios were created to test two functional MT's and nine defective MT's:

- Acceleration module: When at rest, the acceleration module of the MT's (and any device, for that matter) should be $\approx 1g$. To test which MT's had this problem, the user interface reads the acceleration values read by the MT's, calculates their acceleration module, and tells the user which MT's are defective.
- Axes alignment: The MT's should have their axes aligned in a specific orientation. To test their alignment, the apparatus moves to specific positions that have the axes of the MT's aligned with Earth's gravity. At these positions, the apparatus reads the acceleration on the three axes of the MT's and determines if they match the correct values for each position.
- Three axes rotation: This test consists of moving the apparatus to a position that rotates the MT's along all three axes. After the apparatus moves and stops, the acceleration is read on the three axes and compared to the IMU's. If the values match (within a certain threshold), the interface considers that the MT's have passed the test.

- Quaternion: The final test checks if the quaternions read by the target MT's match the quaternions recorded previously with functional MT's. After moving the apparatus from the resting position to the same position used on the third test, the quaternions are compared.

But despite having accomplished these goals, the apparatus that was developed during this dissertation can be improved. These future improvements are discussed in Section 6.2.

6.2 Future Work

After developing the prototype presented in this dissertation, these were the main improvements that we would like to add in future implementations:

- Reduce the signal noise: We ended Subsection 5.1.2 referring that there are other ways to “clean” the feedback signal from the motors. One of these methods is placing ADC's closer to the motors. This way, the feedback signal transmitted through the long conductors would be digital instead of analog, which would make it less prone to being corrupted by EMI. If this signal reached the MCU uncorrupted, we could implement a closed control loop for the servo motors (e.g. PID controller).
- Different servo motors: Another limitation discussed in Section 5.1 was that the torque of the DSS-M15S servo motors was not enough to move the “elbow” joint through the whole ROM of these motors. In a future implementation, these motors will be upgraded to servos with more torque. This implies an increase in total weight, which is already the source of this limitation. This balance needs to be studied further to find the optimal solution, but will likely imply higher costs.
- Improvements in sensor fusion: The final limitation discussed in Section 5.1 was the drift in the quaternions from the IMU's. In future implementations, one of the first new features is compensating the gyroscope drift and using these quaternions to compare directly with the quaternions from the MT's.
- Add more tests: Four different tests were implemented in this dissertation and discussed in Section 5.2.2. More tests can be added - for example, testing if the MT's saturate at any point during the motion of the apparatus.
- Movement fidelity: The testing scenarios we have implemented and discussed in Section 5.2.2 are to be performed once or twice, but this apparatus can also be used to repeat the same tests multiple times. For example, we can have the apparatus perform the same movement multiple times and register if the MT's always register the same value.

- Simulate other subsets of movements: As discussed in Section 4.1, the pieces developed for connecting the “shoulder” joint to the fire extinguisher base can be adapted to simulate more human arm movements or even the movements of the human leg. This requires the modification of the user interface and of the firmware on the MCU, to present these new options to the user. It also requires modifying the position of the fire extinguisher base to better resemble the human leg.
- Improve user-friendliness: The user interface developed in this dissertation is fully functional and easy to interact with. But it can become more user-friendly if a GUI is developed. This can also be done in Python using the *PySimpleGUI* library, for example [Mike, 2023].
- End-to-End solution: Having verified that this apparatus can simulate human movements, we can build more apparatuses and connect them all with software. This allows the simulation of not only arm/leg movements, but also a subset of movements done by the whole body. If robust enough, this can evolve into an End-to-End solution that will be used to test the Digital Physical Therapist [Sword, 2023].

6.3 Main Outcomes

After all the work discussed in this dissertation, there are two main outcomes:

- A mechanical apparatus that can replicate human movements. While we chose to replicate a subset of movements of the human arm, this apparatus could be adapted to replicate a wider set or even movements from the human leg.
- Software that can be used for MT motion data validation. While it was created with our apparatus in mind, the interface and the MCU shell could be adapted to be used with future iterations, or even different apparatuses.

This dissertation focuses on presenting the use cases that we considered during the development of this prototype, but these two outcomes could be used to implement additional use cases and different testing scenarios.

References

- ABB (2020). Robots revolutionize drug dispensing at shanghai seventh people’s hospital. <https://new.abb.com/news/detail/70025/robots-revolutionize-drug-dispensing-at-shanghai-seventh-peoples-hospital>. (Last accessed in 12/12/2022). [Cited on page 12]
- Abe, K., Tadakuma, K., and Tadakuma, R. (2021). Abenics: Active ball joint mechanism with three-dof based on spherical gear meshings. *IEEE Transactions on Robotics*, 37(5):1806–1825. [Cited on pages vii and 17]
- Abhayasinghe, N. and Murray, I. (2014). Human gait phase recognition based on thigh movement computed using imus. pages 1–4. [Cited on page 18]
- Aditya, H. N., Mardiaty, R., Kamelia, L., and Rohandy, F. (2022). Two-axis balancing system for ship-table based on the proportional integral derivative controller (pid) methods. In *2022 16th International Conference on Telecommunication Systems, Services, and Applications (TSSA)*, pages 1–4. [Cited on page 36]
- Amazon (2023). Servo bracket. https://www.amazon.com/-/pt/dp/B08F4XX5CG/ref=sr_1_3?keywords=servo+bracket&qid=1675177979&sr=8-3. [Cited on page 52]
- Amortila, V., Mereuta, E., Novetschi, M., and Rus, M. (2021). Modeling and simulation of the driver’s biomechanical system using adams software. 336:461–353. [Cited on page 10]
- Automatic Addison (2020). Automatic addison - yaw, pitch, and roll diagrams using 2d coordinate systems. <https://automaticaddison.com/yaw-pitch-and-roll-diagrams-using-2d-coordinate-systems/>. (Accessed 16/3/2023). [Cited on pages vii and 22]
- Bai, L., Pepper, M. G., Yan, Y., Spurgeon, S. K., Sakel, M., and Phillips, M. (2011). A multi-parameter assessment tool for upper limb motion in neurorehabilitation. In *2011 IEEE International Instrumentation and Measurement Technology Conference*, pages 1–4. [Cited on page 20]
- Bakhshi, S., Mahoor, M. H., and Davidson, B. S. (2011). Development of a body joint angle measurement system using imu sensors. *2011 Annual International*

- Conference of the IEEE Engineering in Medicine and Biology Society*, pages 6923–6926. [Cited on page 18]
- Bento, V. A. F. (2012). Sword: an intelligent vibratory wearable device to improve rehabilitation in stroke patients. [Cited on page 18]
- Chaulk, M. (2011). Inertial measurement unit (imu) testing procedure. [Cited on pages vii and 25]
- Chen, G.-D. and Chang, C.-W. (2008). Using humanoid robots as instructional media in elementary language education. In *2008 Second IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning*, pages 201–202. [Cited on page 12]
- Chin, K.-Y., Wu, C.-H., and Hong, Z.-W. (2011). A humanoid robot as a teaching assistant for primary education. In *2011 Fifth International Conference on Genetic and Evolutionary Computing*, pages 21–24. [Cited on page 12]
- Choudhury, A., Li, H., Greene, C. M., and Perumalla, S. (2018). Humanoid robot-application and influence. *Archives of Clinical and Biomedical Research*, 02. [Cited on page 12]
- Dadafshar, M. (2014). Accelerometer and gyroscopes sensors: Operation, sensing, and applications senior member of technical staff (field application engineering). [Cited on page 19]
- Darmawan, E. (2019). Space security system using motion sensor and notification of short message service with arduino-based fuzzy logic algorithm. [Cited on page 18]
- Dasarathy, B. (1997). Sensor fusion potential exploitation-innovative architectures and illustrative applications. *Proceedings of the IEEE*, 85(1):24–38. [Cited on page 20]
- Dasarathy, B. V. (2001). Information fusion – what, where, why, when, and how? *Information Fusion - INFFUS*, 2:75–76. [Cited on page 20]
- DFRobot (2017). 20w adjustable dc-dc buck converter with digital display. [Cited on page 40]
- DFRobot (2018). Dss-m15s 270° 15kg df metal servo with analog feedback sku: Ser0044. [Cited on pages vii, 35, 39, 56, 57, and 61]
- Eberly, D. (1999). Euler angle formulas. [Cited on page 22]

- ElectroFun (2022). Kit suporte em alumínio para servo-motor. <https://www.electrofun.pt/robotica/kit-suporte-aluminio-servo-motor-mg995-mg996>. [Cited on pages viii and 48]
- Elmenreich, W. (2002). An introduction to sensor fusion. [Cited on pages 20 and 21]
- Erdemir, A. (2014). Open knee: Open source modeling and simulation in knee biomechanics. *Journal of Knee Surgery*, 29:107–116. [Cited on page 10]
- Extintor, L. (2023). Suporte de pé amovível. <https://alojadoextintor.pt/produto/suporte-de-pe/>. [Cited on pages vii and 41]
- Farnell (2023). Farnell - mti-100-2a8g4. <https://pt.farnell.com/xsens/mti-100-2a8g4/mems-module-4-5v-to-34v/dp/3514472>. (Accessed 16/3/2023). [Cited on pages vii and 20]
- Farrelly, S. (2022). cli-embedded. <https://github.com/FARLY7/embedded-cli>. [Cited on page 67]
- Filippeschi, A., Schmitz, N., Miezal, M., Bleser, G., Ruffaldi, E., and Stricker, D. (2017). Survey of motion tracking methods based on inertial sensors: A focus on upper limb human motion. *Sensors (Basel, Switzerland)*, 17. [Cited on page 19]
- Fonseca, M. C. (2020). Biomechanical simulation of the damage on the acl in injury related movements. [Cited on page 10]
- Fudal, P., Gimbert, H., Gondry, L., Hofer, L., Ly, O., and Passault, G. (2013). An experiment of low cost entertainment robotics. In *2013 IEEE RO-MAN*, pages 820–825. [Cited on page 12]
- Gallagher, A., Matsuoka, Y., and Ang, W.-T. (2004). An efficient real-time human posture tracking algorithm using low-cost inertial and magnetic sensors. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2967–2972. IEEE. [Cited on page 19]
- Gan, Y. and Dai, X. (2012). Human-like manipulation planning for articulated manipulator. *Journal of Bionic Engineering*, 9:434–445. [Cited on page 12]
- Garcia-Garcia, B., Bouwmans, T., and Silva, A. J. R. (2020). Background subtraction in real applications: Challenges, current models and future directions. *Computer Science Review*, 35. [Cited on page 19]

- Gebre-Egziabher, D., Elkaim, G. H., Powell, J., and Parkinson, B. W. (2000). A gyro-free quaternion-based attitude determination system suitable for implementation using low cost sensors. In *IEEE 2000. Position Location and Navigation Symposium (Cat. No. 00CH37062)*, pages 185–192. IEEE. [Cited on page 19]
- Golabchi, A., Han, S., Seo, J., Han, S., Lee, S., and Al-Hussein, M. (2015). An automated biomechanical simulation approach to ergonomic job analysis for workplace design. *Journal of Construction Engineering and Management*, 141:04015020. [Cited on page 10]
- Hatze, H. (1974). The meaning of the term "biomechanics". *Journal of biomechanics*, 7:189–190. [Cited on page 10]
- HTF (2023). Mpu6050-6 as-gyroscope accelerometer sensor. <https://www.htfelectronics.nl/en/mpu6050-6-as-gyroscope-accelerometer-sensor.html>. [Cited on pages vii and 36]
- Huerta, J. (2011). Introducing the quaternions. [Cited on page 23]
- Imbrie-Moore, A. M., Park, M. H., Paulsen, M. J., Sellke, M., Kulkarni, R., Wang, H., Zhu, Y., Farry, J. M., Bourdillon, A. T., Callinan, C., Lucian, H. J., Hironaka, C. E., Deschamps, D., and Woo, Y. J. (2020). Biomimetic six-axis robots replicate human cardiac papillary muscle motion: Pioneering the next generation of biomechanical heart simulator technology. *Journal of the Royal Society Interface*, 17. [Cited on page 14]
- InvenSense (2023). Mpu-6050 six-axis (gyro + accelerometer) mems motiontracking™ devices. <https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050/>. [Cited on page 36]
- Islam, T., Islam, M. S., Shajid-Ul-Mahmud, M., and Hossam-E-Haider, M. (2017). Comparison of complementary and kalman filter based data fusion for attitude heading reference system. *AIP Conference Proceedings*, 1919(1):020002. [Cited on page 22]
- Janousek, V., Jirankova, Z., and Stukjunger, P. (2019). Dt0127 design tip st’s mems accelerometer sensor testing & self-test. [Cited on page 25]
- Johannsen, H., Stein, M., Otte, D., Hannover, U., Wichert-Allee, K., Hannover, D., and Berlin, T. U. (2015). Crash simulation for biomechanical research. [Cited on page 10]
- Jones, K. S. and Kumar, R. U. (2018). Robotic arm with real time human arm motion replication using iot. [Cited on pages 27 and 30]

- Kapse, S. S. and Kapse, S. K. (2013). Modular design and control of humanoid robot arm. *International Journal of Scientific & Engineering Research*, 4. [Cited on page 13]
- Karimi, K. (2022). The role of sensor fusion and remote emotive computing (rec) in the internet of things. [Cited on page 21]
- Karimi, M. (2016). Master thesis motion tracking: Using multiple leap motion sensors in assembly workplace in smart factory. [Cited on page 19]
- Khalil, W. and Dombre, E. (2002). Introduction to geometric and kinematic modeling of parallel robots. *Modeling, Identification and Control of Robots*, pages 171–190. [Cited on pages 13 and 14]
- King, M. (2011). Computer simulation modelling in sports biomechanics. *Biomechanics in Sports 29 Portuguese Journal of Sport Sciences*, 11:2011. [Cited on page 10]
- Koenemann, J. and Bennewitz, M. (2012). Whole-body imitation of human motions with a nao humanoid. pages 425–425. [Cited on page 26]
- Kumar, D. and Divakar, V. (2015). Spy robot using self-balancing algorithm with pan and tilt control of camera. [Cited on pages vii, 21, and 22]
- Law, E. N. (2023). Activities of daily living - elder needs law. <https://www.elderneedslaw.com/blog/activities-of-daily-living-and-medicaid-eligibility>. (Accessed 7/3/2023). [Cited on pages vii and 2]
- LEAP (2023). Leap motion tracker system. <https://www.ultraleap.com/product/leap-motion-controller/>. (Accessed 16/3/2023). [Cited on page 17]
- Lemerle, S., Grioli, G., Bicchi, A., and Catalano, M. G. (2019). A variable stiffness elbow joint for upper limb prosthesis. pages 7327–7334. [Cited on page 16]
- Levin, M. F., Weiss, P. L., Keshner, E. A., Levin, M. F., Weiss, P. L., and Keshner, E. A. (2015). Emergence of virtual reality as a tool for upper limb rehabilitation: Incorporation of motor control and motor learning principles. *Physical Therapy*, 95:415–425. [Cited on page 2]
- Lindequist, J. and Lönnblom, D. (2004). Construction of a motion capture system. [Cited on page 18]
- Liu, Y., Hsu, C.-L., and Liu, Y. (2019). An intelligent dumbbell designed for fitness activity monitoring. In *2019 IEEE International Conference on Computation, Communication and Engineering (ICCCE)*, pages 29–32. [Cited on page 18]

- Lu, X. and Liu, M. (2016). Optimal design and tuning of pid-type interval type-2 fuzzy logic controllers for delta parallel robots. *International Journal of Advanced Robotic Systems*, 13:1. [Cited on page 14]
- Luttgens, K. and Hamilton, N. (1997). *Kinesiology: Scientific Basis of Human Motion*. Brown & Benchmark. [Cited on pages viii, 63, and 64]
- Ma, L., Tornetta, F., Egan, W., and Wright, W. G. (2019). Virtual reality therapy as adjunct to traditional physical therapy for a tbi patient who suffered a gunshot wound to the head: Case report. pages 1–5. [Cited on page 1]
- Machado, M. M. F. (2013). A multibody approach to the contact dynamics: a knee joint application. <https://repositorium.sdum.uminho.pt/handle/1822/24564>. [Cited on page 10]
- Macintyre, S. A. (2014). Magnetic field measurement. <https://web.archive.org/web/20150319010220/http://engineering.dartmouth.edu/dartmag/docs/macintyre.pdf>. (Accessed 16/3/2023). [Cited on page 20]
- Magermans, D. J., Chadwick, E. K., Veeger, H. E., and Helm, F. C. V. D. (2005). Requirements for upper extremity motions during activities of daily living. *Clinical Biomechanics*, 20:591–599. [Cited on page 15]
- Mahmood, M., Rizwan, M. F., Sultana, M., Habib, M., and Imam, M. H. (2020). Design of a low-cost hand gesture controlled automated wheelchair. In *2020 IEEE Region 10 Symposium (TENSYP)*, pages 1379–1382. [Cited on page 36]
- Makalesi, A., Yavan, L., and Tepe, C. (2021). Test experiment design for imu-based angle measurement systems. *European Journal of Science and Technology Special Issue*, 26:180–184. [Cited on pages vii and 24]
- Mangate, L., Kadu, S., Kale, D., Agrawal, K., Kandurke, P., Kasliwal, T., and Pune, V. (2022). Gesture replication robo-arm. 10. [Cited on pages vii, 26, 27, and 30]
- MeanWell (2020). Mean well lrs-100-x power supply - datasheet. [Cited on page 40]
- Mghames, S., Catalano, M. G., Bicchi, A., and Grioli, G. (2019). A spherical active joint for humanoids and humans. *IEEE Robotics and Automation Letters*, 4(2):838–845. [Cited on pages vii and 16]
- Mike (2023). Pysimplegui. <https://www.pysimplegui.org/en/latest/>. [Cited on page 92]
- Moeslund, T. B., Hilton, A., and Krüger, V. (2006). A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104:90–126. [Cited on pages 17 and 18]

- Movella (2023). Movella - mti-100 imu. <https://www.movella.com/products/sensor-modules/xsens-mti-100-imu>. (Accessed 16/3/2023). [Cited on page 20]
- Nilsson, M. (2012). Evaluating the accuracy of motion tracking algorithms for determining the position of finger tips. [Cited on page 17]
- Nogueira, P. (2011). Motion capture fundamentals a critical and comparative analysis on real-world applications. [Cited on page 19]
- Nordic (2023). nrf52840 dongle. <https://www.nordicsemi.com/Products/Development-hardware/nrf52840-dongle>. [Cited on pages viii and 71]
- Nunes, J. F., Moreira, P. M., and Tavares, J. M. R. (2015). Human motion analysis and simulation tools: A survey. *Handbook of Research on Computational Simulation and Modeling in Engineering*, pages 359–387. [Cited on page 11]
- Oliveira, R., Resende, R., Soares, F., Calado, A., and Leite, P. (2019). Mimicking human movement with robots: Control of an anthropomorphic robotic arm using a glove-based system as an educational tool. *2019 6th International Conference on Control, Decision and Information Technologies, CoDIT 2019*, pages 79–84. [Cited on pages vii, 26, 28, and 30]
- OpenCV (2023). About opencv. <https://opencv.org/about/>. (Accessed 23/1/2023). [Cited on page 26]
- OptiTrack (2023). Optitrack. <https://www.optitrack.com/>. (Accessed 16/3/2023). [Cited on page 17]
- Page, Shah, M., Bhargava, S., Mr, S., and Kandhasamy, S. (2014). Mission planning and waypoint navigation of a micro quad copter by selectable gps co-ordinates. *International Journal of Advanced Research in Computer Science and Software Engineering*. [Cited on page 22]
- Palermo, E., Rossi, S., Marini, F., Patanè, F., and Cappa, P. (2014). Experimental evaluation of accuracy and repeatability of a novel body-to-sensor calibration procedure for inertial sensor-based gait analysis. *Measurements*, 52:145–155. [Cited on page 20]
- Pandas (2023). pandas. <https://pandas.pydata.org/>. [Cited on page 62]
- Perceiving Systems (2023). Perceiving systems - max planck institute for intelligent systems: Inertial motion capture. <https://ps.is.mpg.de/pages/inertial-motion-capture>. (Accessed 14/3/2023). [Cited on page 19]
- Picerno, P., Cereatti, A., and Cappozzo, A. (2008). Joint kinematics estimate using wearable inertial and magnetic sensing modules. *Gait & Posture*, 28:588–595. [Cited on page 20]

- Printables (2023). Servo bracket. <https://www.printables.com/model/172925-servo-mount>. [Cited on page 52]
- Pupo, L. B. (2016). Characterization of errors and noises in mems inertial sensors using allan variance method. [Cited on page 21]
- PySerial (2023). Pyserial. <https://pypi.org/project/pyserial/>. [Cited on pages 62 and 70]
- Ramanathan, R., Eberhardt, S., Rahman, T., Sample, W., Seliktar, R., and Alexander, M. (2000). Analysis of arm trajectories of everyday tasks for the development of an upper-limb orthosis. *IEEE Transactions on Rehabilitation Engineering*, 8(1):60–70. [Cited on page 15]
- Rapp, I. (2014). Motion capture actors: Body movement tells the story. https://web.archive.org/web/20140703113656/http://www.nycastings.com/dmxreadyv2/blogmanager/v3_blogmanager.asp?post=motioncaptureactors. (Accessed 9/3/2023). [Cited on page 17]
- Renganathan, G. (2015). Design and control of 3-dof articulated robotic arm using labview and ni-myrio. *INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING*, 3:5. [Cited on pages vii and 12]
- Ribeiro, M. I. (2000). Introduction to kalman filtering a set of two lectures. [Cited on page 22]
- Ribeiro, N. F. and Santos, C. P. (2017). Inertial measurement units: A brief state of the art on gait analysis. [Cited on page 20]
- Rodríguez, P. (2009). Optical motion tracking in earthquake-simulation shake table testing: Preliminary results. pages 371–375. [Cited on page 24]
- Romilly, D. P., Anglin, C., Gosine, R. G., Hershler, C., and Raschke, S. U. (1994). A functional task analysis and motion simulation for the development of a powered upper-limb orthosis. *IEEE Transactions on Rehabilitation Engineering*, 2:119–129. [Cited on page 15]
- Rosen, J., Perry, J. C., Manning, N., Burns, S., and Hannaford, B. (2005). The human arm kinematics and dynamics during daily activities - toward a 7 dof upper limb powered exoskeleton. *2005 International Conference on Advanced Robotics, ICAR '05, Proceedings*, 2005:532–539. [Cited on page 15]
- Rusk, H. A. (1972). *A World to Care For: The Autobiography of Howard A. Rusk, M.D.* Random House. [Cited on page 1]

- Santos, V. and Valero-Cuevas, F. (2004). A bayesian approach to biomechanical modeling to optimize over large parameter spaces while considering anatomical variability. In *The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 2, pages 4626–4629. [Cited on page 11]
- ServoCity (2023). 1" bore bottom tapped clamping mount. <https://www.servocity.com/1-bore-bottom-tapped-clamping-mount/>. [Cited on pages viii and 50]
- SiLabs (2023). Which arm cortex core is right for your application: A, r or m? [Cited on page 38]
- Song, Z., Cao, Z., Li, Z., Wang, J., and Liu, Y. (2021). Inertial motion tracking on mobile and wearable devices: Recent advancements and challenges. 26. [Cited on page 19]
- SparkFun (2023). Sparkfun 6 degrees of freedom breakout - lsm6dso (qwiic). <https://www.sparkfun.com/products/18020>. [Cited on pages vii and 37]
- ST (2019). Datasheet - lsm6dso. [Cited on page 37]
- ST (2021). Stm32g431x6 stm32g431x8 stm32g431xb datasheet. [Cited on page 39]
- ST (2023). Stm32 nucleo-64 development board with stm32g431rb mcu. <https://www.st.com/en/evaluation-tools/nucleo-g431rb.html>. [Cited on pages vii and 39]
- Stasse, O., Flayols, T., Budhiraja, R., Giraud-Esclasse, K., Carpentier, J., Mirabel, J., Prete, A. D., Souères, P., Mansard, N., Lamiriaux, F., Stasse, O., Flayols, T., Budhiraja, R., Giraud-Esclasse, K., Carpentier, J., Mirabel, J., Prete, A. D., Eres, P. S., Mansard, N., Lamiriaux, F., Laumond, J.-P., Marchionni, L., Tome, H., and Ferro, F. (2017). Talos: A new humanoid research platform targeted for industrial applications. *Humanoid Robotics*. [Cited on pages vii and 13]
- STM (2023a). Sensor and motion algorithm software expansion for stm32cube. <https://www.st.com/en/embedded-software/x-cube-mems1.html>. [Cited on page 64]
- STM (2023b). Stm32 cube ide. <https://www.st.com/en/development-tools/stm32cubeide.html>. [Cited on page 77]
- STM (2023c). Stm32 cubemx. <https://www.st.com/en/development-tools/stm32cubemx.html>. [Cited on page 77]

- Sword (2023). Digital physical therapy. <https://swordhealth.com/solutions/digital-therapy>. [Cited on page 92]
- Sword Health (2023). Sword health digital msk: Freeing the world from pain. <https://swordhealth.com/>. (Accessed 7/3/2023). [Cited on pages vii and 3]
- Teulieres, M., Tilley, J., Bolz, L., Ludwig-Dehm, P. M., and Wagner, S. (2019). Industrial robotics: Insights into the sector’s future growth dynamics. [Cited on page 12]
- The Staff of the Benjamin Rose Hospital (1959). Multidisciplinary studies of illness in aged persons: Ii. a new classification of functional status in activities of daily living. *Journal of Chronic Diseases*, 9:55–62. [Cited on page 1]
- Tortora, G. J., Grabowski, S. R., and Wiiey, J. (2018). *PRINCIPLES OF ANATOMY AND PHYSIOLOGY Tenth Edition Volume 2 Support and Movement of the Human Body*. 15th edition. [Cited on page 15]
- Tseng, S. P., Li, W.-L., Sheng, C.-Y., Hsu, J.-W., and Chen, C.-S. (2011). Motion and attitude estimation using inertial measurements with complementary filter. In *2011 8th Asian Control Conference (ASCC)*, pages 863–868. [Cited on page 22]
- UMN (2022). Mast lab multi-axial sub-assembly testing facility. https://mastlab.umn.edu/sites/mastlab.umn.edu/files/files/media/mast_brochure.pdf. (Last accessed in 12/12/2022). [Cited on page 13]
- Vicon (2023). 100 years of motion-capture technology. https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.engadget.com%2F2018-05-25-motion-capture-history-video-vicon-siren.html&psig=A0vVaw3tUw9d1wuTm8QkV_L438MR&ust=1678911747097000&source=images&cd=vfe&ved=0CBAQjRxqFwoTC0jKjOSf3POCFQAAAAAAdAAAAABAE. (Accessed 14/3/2023). [Cited on page 19]
- VICON (2023). Vicon motion capture system. <https://www.vicon.com/>. (Accessed 16/3/2023). [Cited on page 17]
- Villanueva, J. P. d. (2016). Universidad de navarra - anatomy i - bones & joints 2016
2017. <https://dadun.unav.edu/bitstream/10171/43018/1/BONES%20%26%20JOINTS%2016-17.pdf>. (Last accessed in 2/2/2023). [Cited on page 15]
- Wald, L. (1998). A european proposal for terms of reference in data fusion. international archives of photogrammetry and remote sensing. *XXXII*, 7:651–654. [Cited on page 20]

- Wang, W. S., Mendonca, R., Kording, K., Avery, M., and Johnson, M. J. (2019). Towards data-driven autonomous robot-assisted physical rehabilitation therapy. In *2019 IEEE 16th International Conference on Rehabilitation Robotics (ICORR)*, pages 34–39. [Cited on pages vii, 1, and 2]
- Weyrich, M. and Abdullah, M. W. (2013). Concept of a three d.o.f spherical-joint gripper for industrial robots. In *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*, pages 1–4. [Cited on page 16]
- Wilhelms, J. and Gelder, A. V. (2001). Fast and easy reach-cone joint limits. *Journal of Graphics Tools*, 6(2):27–41. [Cited on pages 15 and 30]
- Williamson, B. (2021). Physical therapy, design, and technology in a changing world. *Physical Therapy*, 101. [Cited on page 1]
- Wolfe, S. (2021). Simulating an earthquake with a 3d printer. <https://www.youtube.com/watch?v=fzvmHa8yw7o>. (Accessed 2/1/2023). [Cited on page 24]
- Wolfram (2023). Wolfram mathworld - rotation matrix. <https://mathworld.wolfram.com/RotationMatrix.html>. (Accessed 16/3/2023). [Cited on page 23]
- Yamada, S., Nomura, T., and Kanda, T. (2019). Healthcare support by a humanoid robot. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 1–2. [Cited on page 12]
- Yenorkar, R. and Chaskar, U. M. (2018). Gui based pick and place robotic arm for multipurpose industrial applications. pages 200–203. [Cited on pages vii, 27, 29, and 30]

Appendix A

Motor's calibration values

A.1 Graphs of Servo Motors

This section contains the graphs obtained using the process described in Subsection 4.2.1.

Feedback --> Angle: Motor1

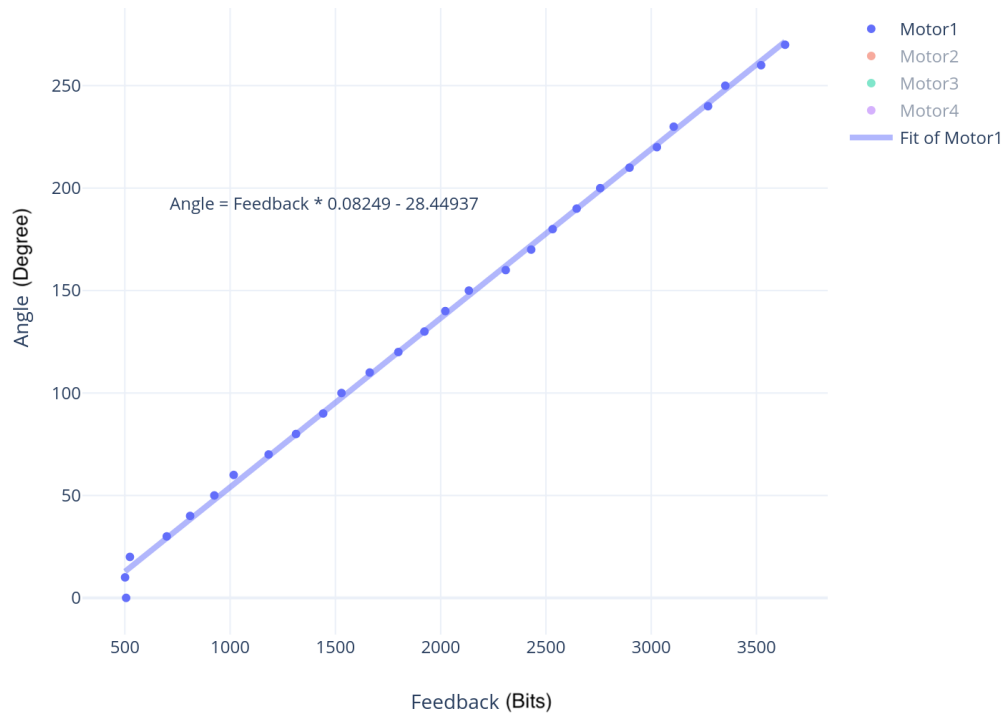


Figure A.1: Relationship between feedback value and angle of the first servo motor.

Feedback --> Angle: Motor2

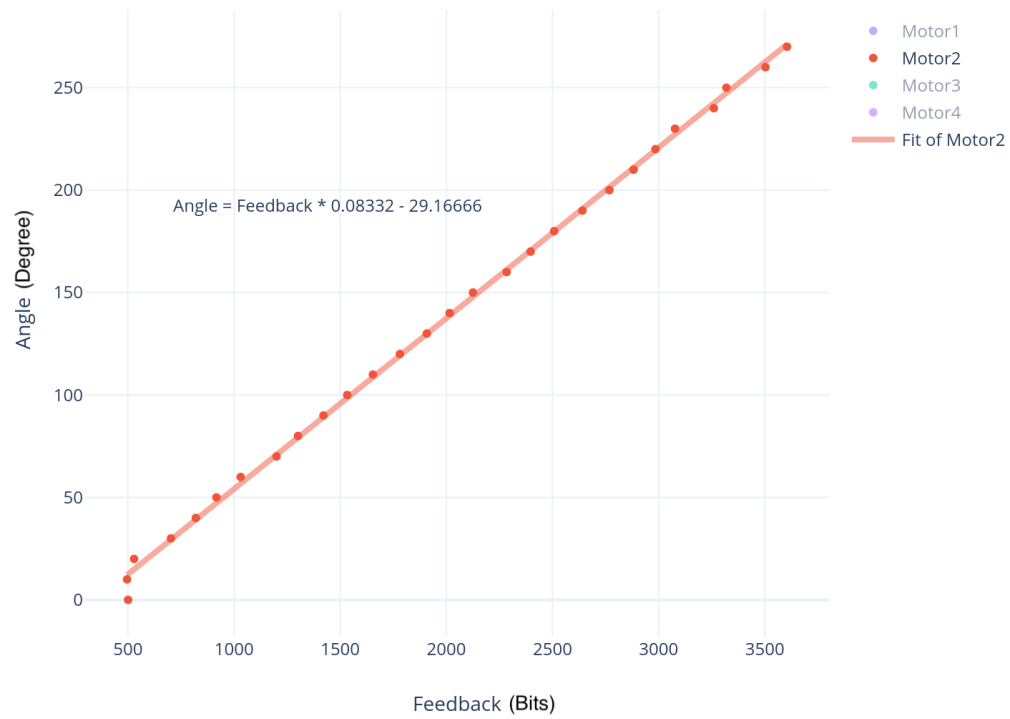


Figure A.2: Relationship between feedback value and angle of the second servo motor.

Feedback --> Angle: Motor3

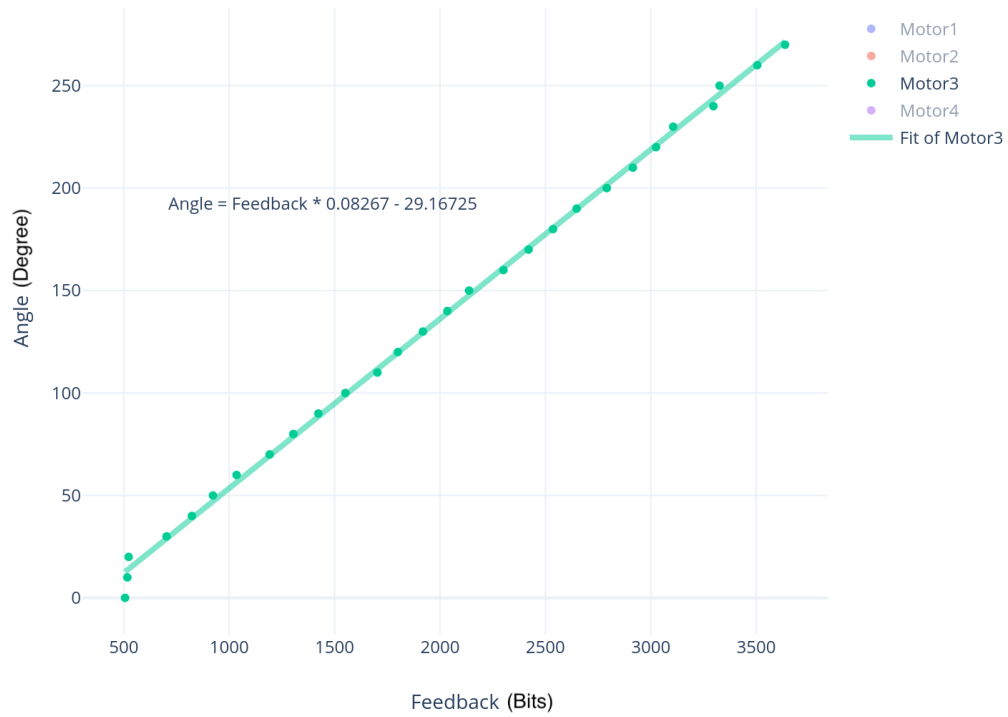


Figure A.3: Relationship between feedback value and angle of the third servo motor.

Feedback --> Angle: Motor4

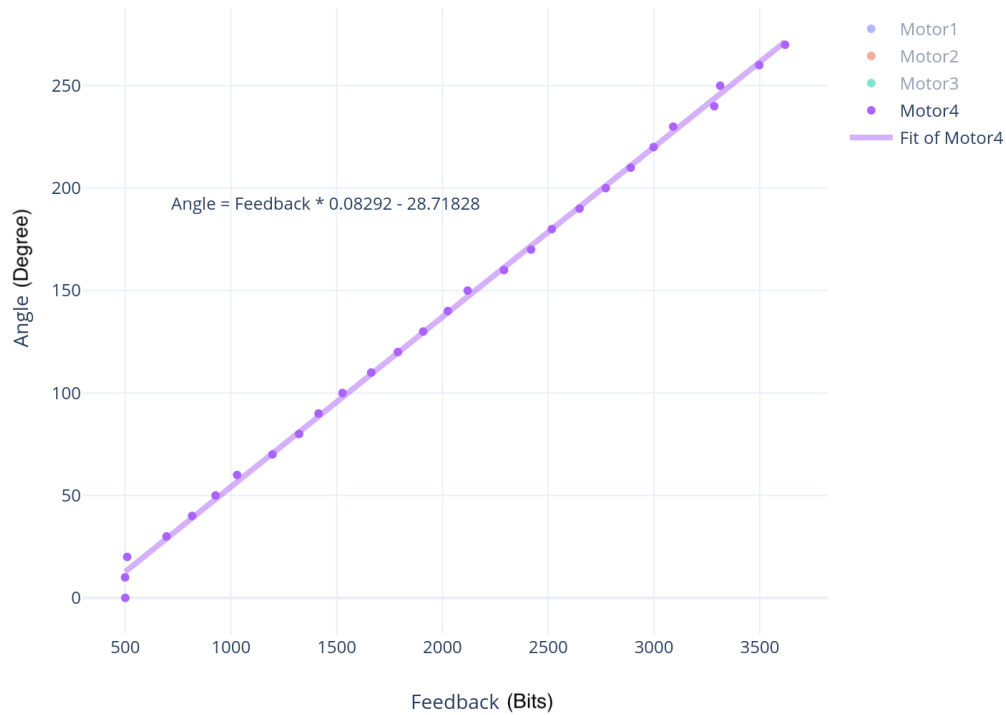


Figure A.4: Relationship between feedback value and angle of the fourth servo motor.

A.2 Values Gathered During Motors' Calibration

This section presents the values gathered using the script described in Subsection 4.2.1.

Table A.1: Raw values read by the motors's calibration script.

Angle (°)	Motor 1	Motor 2	Motor 3	Motor 4
0	506	501	506	500
10	501	496	517	499
20	524	529	523	509
30	699	703	703	695
40	810	820	823	816
50	925	917	923	927
60	1017	1031	1035	1029
70	1183	1200	1192	1196
80	1313	1301	1304	1322
90	1442	1421	1423	1414
100	1529	1533	1551	1527
110	1663	1654	1703	1663
120	1799	1781	1800	1789
130	1923	1908	1919	1909
140	2022	2015	2035	2025
150	2134	2125	2138	2119
160	2309	2284	2301	2291
170	2430	2397	2420	2419
180	2532	2508	2536	2517
190	2646	2641	2648	2648
200	2758	2768	2791	2772
210	2897	2882	2915	2890
220	3027	2985	3025	2998
230	3107	3077	3106	3091
240	3270	3260	3297	3285
250	3352	3319	3326	3313
260	3522	3503	3505	3498
270	3636	3605	3637	3620

A.3 Motors' Calibration Script

This section presents the script used to gather the values presented in Appendix A.2.

```

1 import time
2 import serial
3 from serial.tools.list_ports import comports

```

```
4
5 import pandas as pd
6 import openpyxl
7
8 UART_BAUD = 115200
9 UART_TIMEOUT = 0.05
10 UART_SERIAL_PORT = ''
11
12 columns_excel = [0] * 28
13 feedbacks_motor1 = [0] * 28
14 feedbacks_motor2 = [0] * 28
15 feedbacks_motor3 = [0] * 28
16 feedbacks_motor4 = [0] * 28
17
18 # Select NUCLEO-Board
19 for port in comports():
20     if port.vid == 1155: # STM VendorID: 1155
21         UART_SERIAL_PORT = port.device
22
23 # Open Serial Port
24 ser = serial.Serial(port=UART_SERIAL_PORT, baudrate=UART_BAUD,
25                     timeout=UART_TIMEOUT)
26
27 # Excel Data
28 for i in range(28):
29     columns_excel[i] = i*10
30
31     ser.write('d'.encode())
32     time.sleep(0.6)
33     feedbacks_motor1[i] = ser.readline().decode()
34     ser.flushInput()
35
36     ser.write('f'.encode())
37     time.sleep(0.6)
38     feedbacks_motor2[i] = ser.readline().decode()
39
40     ser.write('g'.encode())
41     time.sleep(0.6)
42     feedbacks_motor3[i] = ser.readline().decode()
43
44     ser.write('h'.encode())
45     time.sleep(0.6)
46     feedbacks_motor4[i] = ser.readline().decode()
47
48     print("Li os motores " + str(i+1) + " vezes")
49
50
51 # Format Excel data
```

```
52 df = pd.DataFrame(  
53     [  
54         feedbacks_motor1,  
55         feedbacks_motor2,  
56         feedbacks_motor3,  
57         feedbacks_motor4  
58     ],  
59     index=['Motor1', 'Motor2', 'Motor3', 'Motor4'],  
60     columns=columns_excel  
61 )  
62 # Create Excel file  
63 df.to_excel('calibrar_motores.xlsx', engine='xlsxwriter')
```

Listing A.1: Motors' calibration script.