Low-Cost Simultaneous and Proportional Myoelectric Control of Powered Upper Limb Exoskeletons

Francisco A. B. Sampaio

Master's Thesis

Supervisor: Professor Doutor Joaquim Gabriel Mendes

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Master in Mechanical Engineering

Porto, March 2021

Abstract

The work described in this thesis consisted of a study of powered upper limb exoskeletons and the posterior design, implementation and validation of a low-cost control system for a 1 DoF elbow joint exoskeleton.

The current state of the art on exoskeleton sensing, control and actuation strategies and technologies was explored, with a focus on intention estimation technologies, namely electromyography.

With these guidelines, a myoelectric simultaneous and proportional control system for the manipulation of a 1 DoF 3D printed robotic arm, actuated by a DC motor, was designed and implemented. The control system has three main components: an EMG data acquisition system, an intention estimation algorithm based on a random forest regressor and a motor controller. It also features a motion tracking system for measuring the current elbow joint flexion/extension angle, based on inertial magnetic sensors, an AHRS based on the Madgwick filter and a sensor-to-segment estimation algorithm.

The solution was validated in an experiment where the random forest model was trained using data from various trials, tuned with 20 iterations of Bayesian optimization and subsequently tested offline and online. System performance was evaluated and the offline and online performances compared, as well as performances before and after optimization. In offline testing, the model's coefficient of determination was $R^2 = 0.6499$ before optimization and $R^2 = 0.7134$ after Bayesian optimization. Under regular online testing circumstances, the lowest coefficient of determination obtained was $R^2 = 0.5393$. On average, the system was able to compute motion intention estimates as fast as every 202 [ms] before optimization, and 325 [ms] after optimization, both within the functional requirements early in the project.

Keywords: sEMG, simultaneous and proportional control, upper limb, exoskeleton, random forest, Bayesian optimization.

Resumo

O trabalho descrito nesta tese consistiu no estudo de exoesqueletos de membros superiores alimentados externamente e a subsequente conceção, implementação e validação de um sistema de controlo *low-cost* de um exoesqueleto de 1 grau de liberdade para a articulação do cotovelo.

O estado da arte das estratégias e tecnologias de sensorização, controlo e atuação de exoesqueletos foi explorado, com ym foco em tecnologias de estimação de intenção, nomeadamente a eletromiografia.

Foi projetado e implementado um sistema de controlo mioelétrico simultâneo e proporcional para a manipulação de um braço robótico de um grau de liberdade, impresso em 3D e atuado por um motor DC. O sistema de controlo tem três principais componentes: um sistema de aquisição de dados de EMG, um algoritmo de estimação de intenções baseado em florestas aleatórias e um controlador de motor. O sistema também incorpora um algoritmo de *motion tracking*, baseado em sensores inerciais magnéticos, um AHRS baseado no filtro de Madgwick e um algoritmo de estimação das orientações de sensores para segmentos, para medição do ângulo de flexão/extensão do cotovelo.

A solução foi validada num estudo experimental em que o modelo de floresta aleatória foi treinado com dados de ensaios, calibrado com 20 iterações de otimização Bayesiana e seguidamente testado *offline* e *online*. O desempenho do sistema foi avaliado e os seus desempenhos *offline* e *online*, assim como antes e após otimização, comparados. Nos testes *offline*, o coeficiente de determinação obtido para cada modelo foi $R^2 = 0.6499$ antes de otimização e $R^2 = 0.7134$ após otimização. Dentro de circunstâncias de teste *online* normais, o coeficiente de determinação mais baixo obtido foi $R^2 = 0.5393$. Em média, o sistema foi capaz de produzir estimativas de intenção de movimento a cada 202 [ms], antes de otimização, e 325 [ms], após otimização, encontrando-se ambos os valores dentro dos requisitos funcionais estabelecidos no início do trabalho.

Palavras-chave: sEMG, controlo proporcional e simultâneo, membros superiores, exoesqueleto, floresta aleatória, otimização Bayesiana.

Acknowledgements

First and foremost, I would like to express my extreme gratitude to my thesis advisor, Prof. Dr. Joaquim Gabriel Mendes, for giving me the great opportunity to work in such an exciting project for my master's thesis.

I would also like to thank my family, Inês, Luís, for all the patience and heartfelt love, but above all to my parents for giving me a much-needed sense of orientation and for helping me stay on track on what is the largest endeavour I have undertaken so far in this life.

In addition, and in no particular order, I would like to thank my friends João, Marta, Maria and Raquel for all the support they've given me. You're here for a reason.

Furthermore, I would like to express my gratitude to everyone else, for you helped me, one way or another, shape myself into who I am today and who I will be tomorrow.

Finally, I would also like to express my deepest gratitude to João Alexandre, whose teachings I hold on to very dearly.

"The purpose of life is finding the largest burden that you can bear and bearing it."

Jordan B. Peterson

Contents

Acronyms and Symbols

ADC	Analog-to-Digital Converter
AI	Artificial Intelligence
BO	Bayesian Optimization
CC	Cepstral Coefficients
CMRR	Common Mode Rejection Ratio
DC	Direct Current
EEG	Electroencephalogram
EMG	Electromyography
FSR	Force Sensitive Resistor
I^2C	Inter-Integrated Circuit
IMU	Inertial Measurement Unit
MAV	Mean Absolute Value
MSD	Musculoskeletal Disorder
MVC	Maximum Voluntary Contraction
PGA	Programmable Gain Amplifier
PWM	Pulse Width Modulation
RF	Random Forest
RMS	Root Mean Squared
SampEn	Sample Entropy
sEMG	Surface Electromyography
SENIAM	Surface Electromyography for Non-Invasive Assessment of Muscles
WL	Waveform Length
ZN	Ziegler-Nichols

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, context and motivation are provided for the present work, objectives are drawn and the document's structure is briefly detailed.

1.1 Context and Motivation

According to the World Health Organization, strokes and cardiovascular diseases take the lives of more than 17 million people worldwide every year (Organization, 2004). Of those who survive, more than 77% incur acute upper limb dysfunction and and about 60% of those will not achieve all the functional movements at 6 months post-stroke (Vieira, 2016).This comes with costs that affect their lives, their families, their communities and, of course, the economy (Rahman et al., 2015).

Fortunately, studies have found robot-aided therapy to significantly reduce arm impairment and improve motor function, to the point where the subject regains at least partial upper limb function of motion (Rahman et al., 2015).

In industries, exposure to postural loads, repeatable motions and tasks of very short duration increase the likelihood of workers to develop Musculoskeletal Disorders (MSDs), and the risk seems to be increasing (O'Sullivan et al., 2015; Sylla et al., 2014). Sylla et al. (2014) mentions the usage of robotic assistive devices as a relevant solution in preventing MSD and improving the quality of life of workers. The authors assert the use of collaborative robots, in particular, powered exoskeletons, as being of great interest to the industry.

Evidently, there is interest in powered exoskeletons in many fields, and several solutions have been presented thus far (Gopura et al., 2016). On the other hand, because these technologies are relatively new and definitely complex, prices tend to be very high (Pina, 2018). Beyond reducing costs, there is still a lot of room for improvement. Gull et al. (2020) acknowledge ergonomics and standardized design, modeling, compliant joints and actuation, performance assessment and adaptive control as areas that require further improvement.

1.2 Goals

With the previous considerations in mind, the main objective of this work is to design and implement a low-cost control system for an early prototype of a powered orthosis or exoskeleton. The device will simulate assistance to a 1 DoF human joint, more specifically, flexion/extension of the human elbow joint. The reasoning for the selection of the elbow joint is primarily due to how much simpler it is in comparison to other joints.

Designing and implementing a low-cost control system for a powered orthosis for the elbow joint is, still, an elaborate endeavour. It is an integrated automated system, complete with sensing, control and actuation. In addition, it has to account for the presence of a human, and not just any human - a patient in rehabilitation or recovery. The standards for safety, comfort and ergonomics are very high (O'Sullivan et al., 2015).

To embark on the development of an enterprise such as the one just presented, it is necessary to perform a careful review of the existing literature. Thereby, another, but in no way lesser than the first, objective is to study and assess, within the time constraints of the project, the current state-of-the-art in design and development of powered exoskeletons. This review can give guidance for the design and implementation of the prototype.

Finally, once the system has been designed and implemented, it is paramount to properly evaluate the solution. For this reason, an experiment must be appropriately devised and the results stringently analyzed.

In summation, there are three main objectives to the present work:

- To study and assess the current state-of-the-art on the design of upper limb powered exoskeletons. The focus should be on existing theory and most prevalent technologies and methods.
- To design and implement a low-cost control system for an early prototype of a powered 1 DoF elbow exoskeleton, based on the theory, technologies and techniques reviewed in the state-of-the-art.
- To conduct an experiment for validation of the system built, in an effort to highlight limitations of the solution adopted, as well as provide insight into future works.

1.3 Document Structure

This thesis is organized into seven chapters, the first being the present one.

In Chapter 2, a concise state-of-the-art of powered exoskeletons is presented, with a focus on upper body exoskeletons. It covers intention estimation technologies, typical control strategies and prevalent choices of actuation.

In Chapter 3, design goals and project requirements are established, thus enabling an overall system architecture to be designed. Based on the latter, component selection is presented, as well as a succinct wiring schematic.

In Chapter 4, the control system solution and the reasoning behind most design choices are presented in considerable detail. In brief, the control system constitutes sensor data acquisition, an intention estimation algorithm and a motor controller, all of which being thoroughly explained in the chapter.

In Chapter 5, implementation of the aforesaid design is presented. Component manufacture and system assembly are summaringly addressed. Codification of the control system is discussed, along with supplementary steps taken in implementing the system.

In Chapter 6, the design and execution of the experiment for validation of the solution are detailed. This involves experiment objectives, delineation of hypotheses, an experiment protocol and, at last, presentation and discussion of the experiment's results.

The last chapter, Chapter 7, is dedicated to the most important insights taken from this work. This chapter is followed by a bibliography and appendices to the present work.

Chapter 2

State of the Art

In this chapter, a brief overview of rehabilitative engineering is presented, with a focus on upper body exoskeletons. This is followed by a review of intention estimation technologies, of which surface electromyography is discussed in greater detail. A review on myoelectric control systems ensues. Finally, actuation strategies for exoskeletons are concisely addressed.

2.1 Rehabilitative Engineering and Powered Exoskeletons

The use of external aids to enable or facilitate movement is not, by any means, an innovative endeavour. When thinking of rehabilitative engineering, the first images to form in one's mind are those of healthcare professionals, such as occupational therapists, medical doctors and nurses, providing aid and guidance to a debilitated patient who is trying to recover from a certain ailment. With the help of engineers and specialists, they seek to restitute the person's mobility to that of times past.

The first prostheses in history were, as is known today, rudimentary in both methods and materials (Thurston, 2007). Since then, the number and variety of technologies and devices has grown quite substantially (Gopura et al., 2016; Gull et al., 2020).

Generally speaking, rehabilitation devices can be discerned depending of the degree of damage the patient's body has suffered (Gopura et al., 2016): prostheses and orthoses. Prostheses are devices that seek to restore function by substitution of a missing element of the patient's body, namely, amputated limbs or parts thereof. Orthoses, on the other hand, are complementary devices used when the body, fully present, is incapable of performing tasks up to a sufficient standard.

Figure 2.1: Examples of passive prostheses available for the arm and hand. Source: ottobock. (2020).

Additionally, rehabilitation devices can be classified according to the level of automation present in the developed solution (Gopura et al., 2016; Advanced Arm Dynamics, Inc., 2020):

- **Passive:** Calling a passive rehabilitation device a device could be seen as a bit misleading. Indeed, as their name implies, these solutions are not automated and tend to lack moving parts. When capable of motion, the purpose is customarily not to improve performance, but their realism. This is especially true for passive prostheses (Figure 2.1), where the concern is mostly cosmetic, such as with solid, static prosthetic arms, hands and fingers (ottobock., 2020). Passive prostheses with a greater functional focus do exist, however. When replacing the lower limbs, maintaining a healthy gait is crucial, hence one of the greatest difficulties in treating partial foot amputees is that their conditions imply, more often than not, some form of foot desensitization, which could lead to further foot breakdown if the prosthetic device isn't properly fitted and maintained (Kennedy and Meler, 2011). Undeniably, amputations alter the biomechanics of limbs, so proper compensation is due. Passive orthoses are also extremely common, predominantly as tools of correction of movement patterns and distribution of loads (Mnatsakanian et al., 2016).

- **Body-Powered:** Just about unique to prostheses, this flavour of the technology consists of utilizing, incidentally, the patient's own body as the source of power. By adapting the bionic mechanism to the underlying kinematics of the actuator (the patient's remaining body), an intuitive operating system of cables, harnesses, and occasionally, manual control, is elaborated (Advanced Arm Dynamics, Inc., 2020). In spite of missing electrical power and its collaborative bells and whistles, body-powered prostheses are among the best when it comes to mechanical performance and robustness, which makes them a reasonable choice for athletes (Figure 2.2) and people who are engaged in manual labour (Advanced Arm Dynamics, Inc., 2020).

- **Externally Powered:** Instead of utilizing the user's own body as a source of mechanical energy, these systems employ one or several of existing power technologies (Advanced Arm Dynamics, Inc., 2020), by and large of electrical nature, yet also hydraulic, pneumatic, etc. Motion estimation is mainly built on either electromyographic or force inputs, which are fed to a sophisticated control system that operates the actuators. Relative to body-powered devices, solutions rooted in external sources of power bring all sorts of novel problems to the table, like weight reduction of all the incumbent actuating paraphernalia, treatment and analysis of the input signals, cost management, etc. This comes with the potential of intelligent, responsive artificial limbs and externally powered orthoses (exoskeletons), the latter being the focus of this thesis.

- **Hybrid:** The last category is a combination of the ones before. Hybrid devices utilize both the user's body and external sources of power for actuation (Advanced Arm Dynamics, Inc., 2020). This comes as a necessity when there is extensive loss or damage of body parts, requiring an extremely complex structure that inevitably makes use of both approaches.

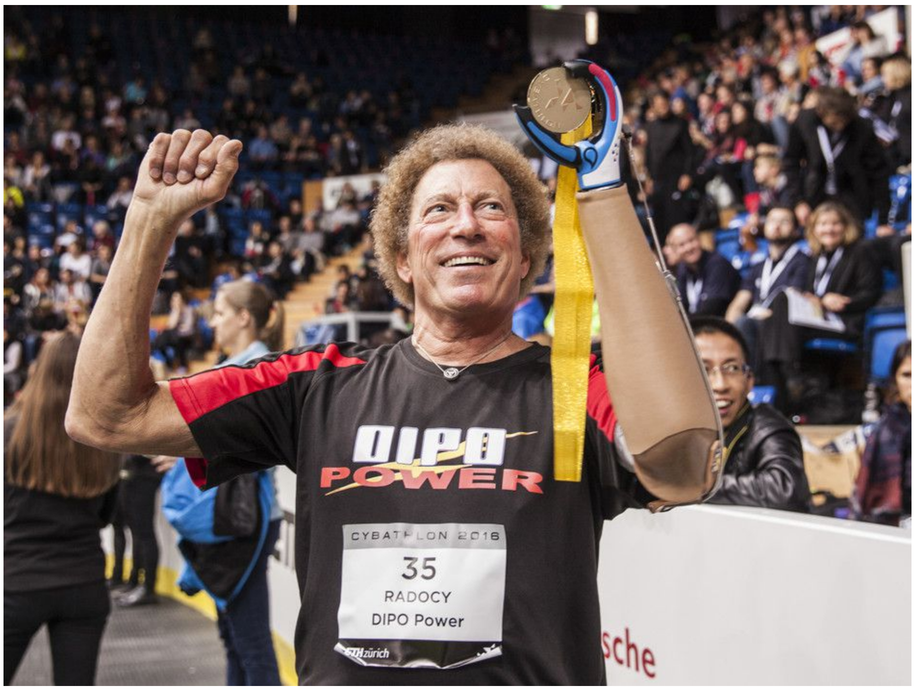

Figure 2.2: Bob Radocy, founder of TRS Prosthetics, after winning the 2016 Cybathlon in Zurich for body-powered prosthetics, wearing the Grip 5 Prehensor his team developed. Source: Caruso (2016).

2.1.1 The Choice for Powered Exoskeletons and Associated Challenges

Apropos orthoses, body-power is very rarely a viable solution when projecting assistive orthotic devices, for there are clear advantages to electing external power.

For starters, if the intent of orthoses is to assist movement, then taking as little energy as possible from the user is an absolute must. Consequently, external power should not be seen as an option, rather a necessity.

There are, of course, consequences to the use of external power. For instance, as it was previously mentioned, these actuators and power supplies come with an inevitable and substantial increase in weight, which leads, in addition, to significant changes in the conjugated system's inertia. This extraneous loading can severely impair or difficult the free motion of the user (Jin et al., 2017).

It's not clear the way in which assistive devices help in reducing the user's effort. When measuring the effect of an on-body ergonomic aid on oxygen consumption during a repetitive lifting task, Whitfield et al. (2014) found no statistically significant impact in oxygen consumption. Wearing the PLAD (the Personal Lifting Assistive Device, Figure 2.3) didn't seem to change the metabolic expenditure. Some sense could be made out of this finding, since the energy exchanges should be internal to the person-PLAD system, ergo the total energy demand is unchanged. The way in which energy demand is distributed throughout the body, nevertheless, can differ - some muscles will be assisted and others will have to work harder.

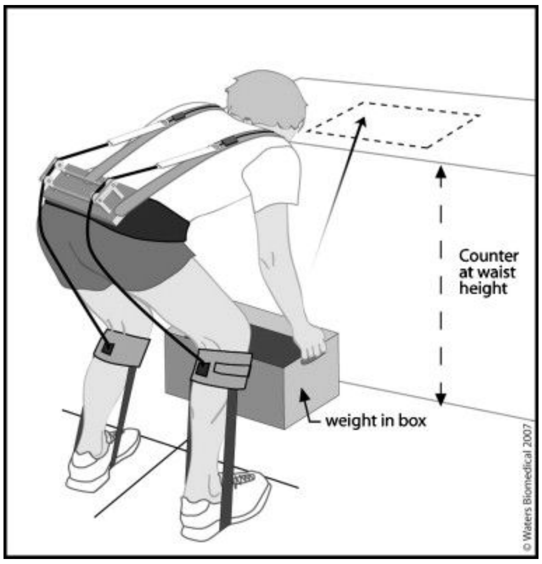

Figure 2.3: Depiction of a user performing a lift task, while wearing the PLAD (Personal Lifting Assistive Device) for assistance. Source: Whitfield et al. (2014).

Hence, knowing how to deal with the likely redistribution of loading becomes paramount. Given the kinematic complexity they introduce, it's not uncommon for exoskeletons to miss their pre-established goals or even work against them. When seeking to evaluate the impact of an upper limb exoskeleton for OHW (Overhead Work) on muscular activity of other muscles, as well as cardiovascular stress and kinematics, Theurel et al. (2018) found that perceived effort was higher when using the exoskeleton (relative to without it). Moreover, even though there were clear reductions in shoulder RPE (Rate of Perceived Exertion, a relative scale, from 0 to 10, rated by the user) and loading, not only was the lower back more loaded than without the assistance, but the test subjects' postures were also compromised.

An advantage of powered exoskeletons is that their control systems, by being more complex have greater conceptual potential. Assuming the array of required sensors can provide with enough information, the dream of accurate, seamless control of an adjustable exoskeleton is still very much alive (Whitton, 2019). Motion intention estimation is more accurate than ever (Artemiadis, 2012) and the expectation is for these technologies to become increasingly more precise and alike their biological counterpart: the human body.

Purely body-powered solutions also suffer from an intrinsic and ineradicable limitation: the power they provide can only go as far as the human body is capable. When human enhancement is taken in consideration, the only viable option is external power (Artemiadis, 2012). Superhuman physical prowess may not be a predominant market, howbeit there are several practical uses for technologies looking to explore this side of assistive devices. First responders, the military and the industry are all excellent examples of use cases for these technologies (Lockheed Martin Corporation, 2020).

Strikingly, the fact that powered exoskeletons have external, independent power sources gives rise to multiple safety concerns. The vast majority of assistive devices already have to take into

Table 2.1: Standards developers (TCs) identified by O'Sullivan et al. (2015) as potential target stakeholders for developing industrial exoskeleton specific standards. It should be emphasized how varied the standards developed by the aforementioned committees are, which highlights the multidisciplinary nature of powered exoskeletons.

Standards Technical Committees - evaluated as potential target stakeholders	
ISO/TC 184	Automation systems and integration
ISO/TC 159	Ergonomics
ISO/TC 94	Personal safety: Protective clothing and equipment
ISO/TC 173	Assistive products for persons with disability
ISO/TC 168	Prosthetics and orthotics
ISO/TC 199	Safety of machinery
ISO/TC 39	Machine tools
CEN/TC 122	Ergonomics
CEN/TC 310	Advanced manufacturing technologies

account critical issues like mechanical failure and ergonomics, and even so, exoskeletons require further supervision. Electrical power, human-robot interaction, numerous issues plague the standardization of these devices, cursed from the bureaucratic demands of being both robots and wearable machines. Some could see these problems as superfluous, but the truth is products that are manufactured in compliance with applicable standards adhere to minimum quality and safety design criteria, are comprised of acceptable materials and components, and have been tested and evaluated (O'Sullivan et al., 2015). Additionally, the application of standards can assist in meeting legal requirements, can minimize costs associated with material selection and production, and facilitate interoperability within the European and global markets. O'Sullivan et al. (2015) highlights some of the issues that could be key to the development of such standards (Table 2.1).

2.1.2 Upper Limb Exoskeletons

The present work pertains to upper limb exoskeletons, which are, essentially, a small subset of the umbrella category of powered exoskeletons. Similarly to their parent devices, upper limb exoskeletons are electromechanical systems designed to interact with the user for the purpose of power amplification, assistance, or substitution of motor function (Gull et al., 2020).

Even though most research and commercially available products concern lower limb exoskeletons, an extensively large variety can still be found in upper limb exoskeletons (Gopura et al., 2016). Exoskeletons purposed to in some way support the back are, technically speaking, upper limb exoskeletons and they are, too, abundant, they will not be, however, the subject of this work. That being said, a simple way to split existing upper limb exoskeletons into different groups is to separate them according to the joints that they assist. Having left the lower back out, the remaining major joints of interest would be the shoulder, the elbow and the wrist. Assistance is customarily delivered starting from the most distal point, and in consequence, to exemplify, some of the most common combinations are the hand and wrist, the hand, wrist and elbow, and the whole arm, shoulder joint included (Gull et al., 2020). To better understand the kinds of problems that arise from these different configurations, let us first briefly delve into the anatomy of the arm, pictured in Figure 2.4.

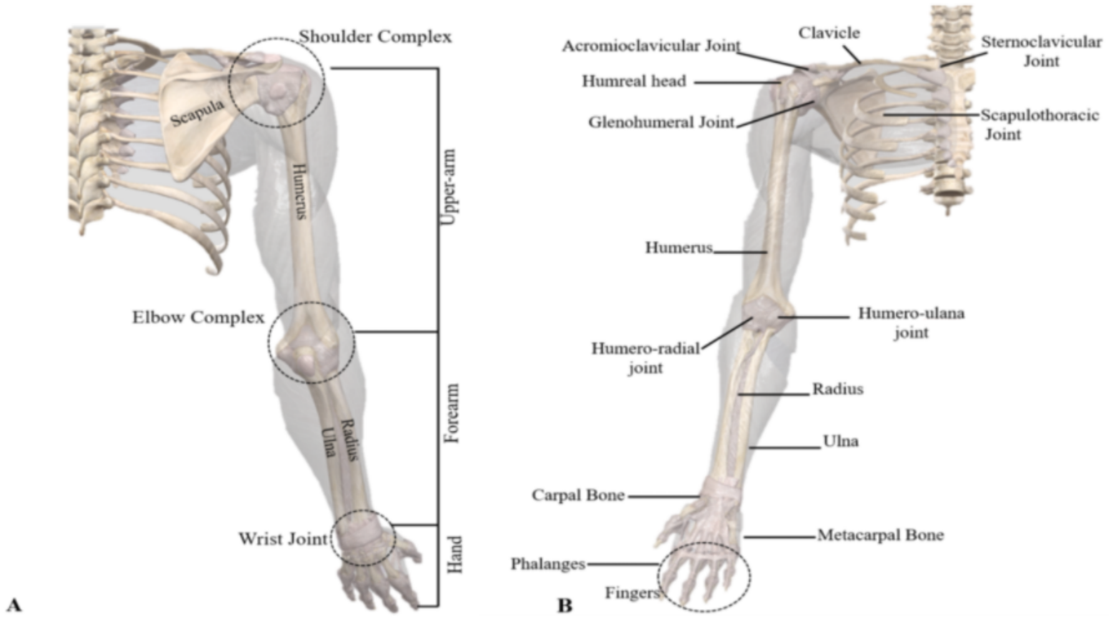

Figure 2.4: Simplified anatomy of the arm, displaying the shoulder complex, the elbow complex and the wrist joint. **A** shows the posterior (from the back) view of the arm, whereas **B** shows the anterior (from the front) view. The arm is an outstandingly complex structure, even in the simplest of its three joints, the elbow. Source: Gull et al. (2020).

At the top of the upper arm, the shoulder complex comprehends four articulations, each of them greatly influencing the kinematics of the arm. For proper ergonomic functioning of an exoskeleton, they have to, at the very least, be taken into account, since, for instance, the glenohumeral joint greatly impacts the position and orientation of the shoulder's centres of rotation.

Connecting the upper arm and the forearm, the elbow joint is a synovial compound joint consisting of two articulations, namely the humeroradial and the humeroulnar joints. The movement enabled by this joint is essentially a 2-DOF motion, but most researchers streamline it to a 1-DOF joint for flexion-extension.

The wrist joint connects the forearm to the hand. Despite not being completely static, most exoskeletons have assumed that the centers of rotation of the wrist joint are fixed. This allows for a very significant simplification in an already extremely complex structure. With several movable parts, the wrist and hand is frequently an isolated target in exoskeleton research.

As mentioned above, for all joints, it is commonplace to simplify the kinematics involved and use approximations or directly ignore degrees of freedom and axes of rotation (Gopura et al., 2016; Gull et al., 2020). These streamlined robots, whereas easier to build, calibrate and control, can suffer from serious joint misalignment and excessively bounded range of motion. Misalignment is a likely cause for discomfort and clunkiness of usability and can even lead to slippage between the human and the exoskeleton, which might translate to an undesired distribution of loads in the different human body parts, ultimately posing a safety and health risk. Again, one of the main difficulties felt with discomfort and misalignment is the current inexistence of standards that address them (Gull et al., 2020).

When the exoskeleton encompasses several joints, to ensure proper functionality and dynamics of the human-exoskeleton system, a comprehensive musculoskeletal model is recommended (Gull et al., 2020) to analytically study the effects at play in the upper limb exoskeleton system. This methodology leads to an accurate evaluation and exact optimization of the robot design. To this end, some popular modeling tools are the AnyBody Modeling System™(AnyBody Technology A/S, 2020), the Virtual Interactive Musculoskeletal System (VIMS) (Lin et al., 2005), OpenSim (OpenSim, 2020) and MB Dyn (MBDyn, 2020). There is not a unified standard biomechanical model for the human body, mostly due to an excessive complexity of the latter that forces specificity of design of the prior (Gull et al., 2020). In the present case, the exoskeleton will only assist the elbow joint and, as such, the aid provided by a musculoskeletal model did not justify the additional work.

2.2 Electromyography as a Study of Motion Intention

The estimation of motion intention is not, presently, an exact science, nevertheless there are sundry practical methods and technologies that can be applied in the deconstruction of this feature. The truth is, motion in the human body is characterized, like many processes, by a chain of events, in this case, the actualization of the self's will to move, the production of a nervous, electrical signal, and the journey of this signal, through the body, to the muscles of pertinence to the intended movement, and the subsequent firing of the muscles' motor unit action potential, ultimately realizing the intended motion (De Luca, 1997).

From here, it can become obvious that there are, in theory, multiple paths through which to estimate motion intention, assuming there is information to be extracted at each point of this chain of events. Since consciousness is still a fairly misunderstood concept, trying to extract anything from the expression of will (which is in itself an unproven abstraction) is currently impossible. Extracting information from the nerve signals coming from the brain is also exceedingly convoluted, knowing that several ends can come from the same common origin and traverse neighbouring nerves. In conclusion, there seems to be three viable alternatives: detection of the movement command at the brain, at the muscle or detection of the movement itself at the extremities.

When it comes to brain computer interface, the most popular technology for rehabilitative devices is, without a doubt, the EEG, or electroencephalogram (Nicolas-Alonso and Gomez-Gil, 2012), as is pictured in Figure 2.5. Brain cells, or neurons, communicate with each other through electrical pulses that travel accross the vast networks that make up the nervous system. The action potentials of this electric activity can be physically detected and measured in multiple ways

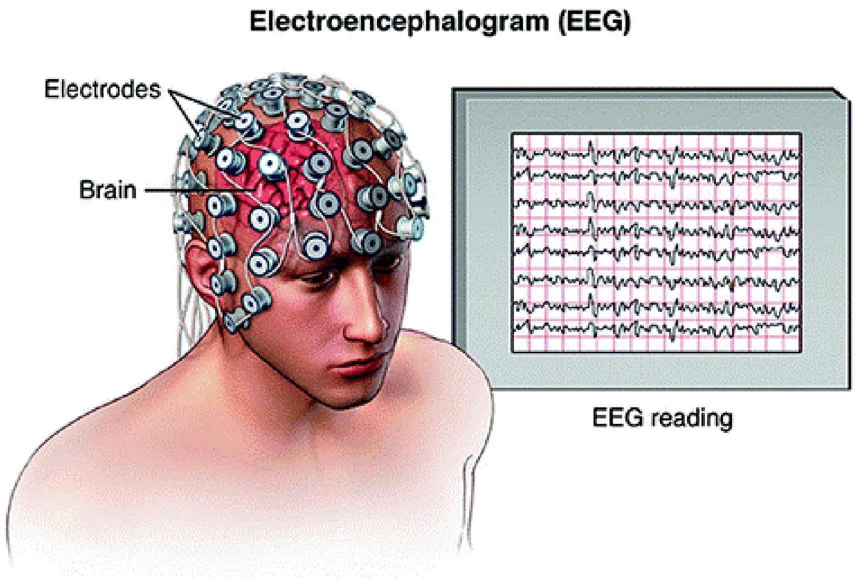

Figure 2.5: Illustration portraying a traditional electroencephalogram (EEG) and the respective signals that could be measured. Source: Siuly et al. (2016).

(Nicolas-Alonso and Gomez-Gil, 2012), one of them the basis for the electroencephalogram (Figure 2.5): small flat metal discs, called electrodes, are attached to the scalp, wired to an amplification and data acquisition system, tracking and recording brain wave patterns (Blocka, 2018). The study and analysis of electroencephalograms is critical for the diagnostic of certain neurological diseases and the treatment of others(Siuly et al., 2016).

As these electrical signals leave the brain, they run through the body, to the target muscles, firing the muscle's motor neurons, or motor units, that is, the nerve cells responsible for the activation and relaxation patterns of muscles (Moore, 2018). When attempting to measure the muscle motor unit's firing activity, EMG, or electromyogram, is the clear choice (De Luca, 1997). By placing electrodes in certain points of the muscles, this information is collected, amplified (due to the extremely low voltages of these signals) and sampled by data acquisition systems. This is called surface EMG data. Alternatively, EMG signals can be measured with the intramuscular EMG methodology (Figure 2.6), which employs monopolar needle electrodes (a highly conductive thin wire of great purity, to measure the motor unit's electrical potentials *in situ*. The resulting signals have much higher fidelity than their surface counterpart (which are more likely to be contaminated by surrounding noise), at the cost of a more invasive setup, requiring specific medical knowledge for the positioning of the electrodes, which, when done incorrectly, could be painful or even damaging (Moore, 2018). For powered exoskeletons, this variant is mostly useless, since it strongly restricts the user.

By opposition, the detection of movement can be achieved in multiple ways. Regardless, researchers frequently opt for FSR's, or Force Sensing Resistors (Gull et al., 2020). A Force Sensing Resistor (Figure 2.7) is a two-terminal passive piezoresistive device that suffers a change in its electrical resistance when stress is exerted over the Stress Sensitive Area (SSA). The SSA is a

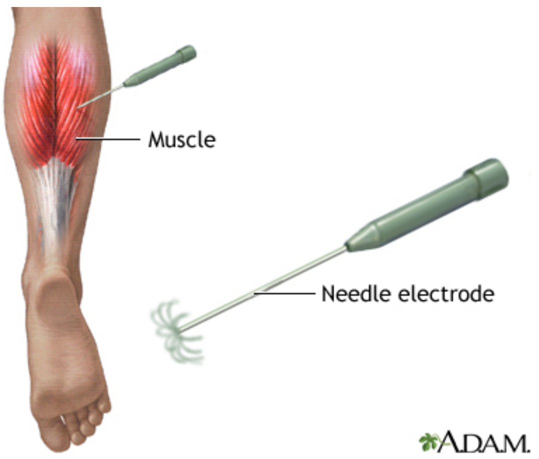

Figure 2.6: Intramuscular EMG illustrated. The needle electrode is a highly conductive thin wire that measures the motor units' action potentials. Source: A.D.A.M., Inc. (2018).

polymer composite of conductive particles dispersed in a non-conductive matrix that is sandwiched between two metal electrodes, the terminals for measuring voltage, thus forming a Traditional Sandwich Element (TSE) (Paredes-Madrid et al., 2017). The electrical resistance of the SSA changes because the spatial input causes the average inter-particle distance to change. Different topologies, configurations and materials are possible, making the FSR a truly flexible measuring unit, with plenty of design parameters and applications (Paredes-Madrid et al., 2017).

Each methodology has its own advantages and disadvantages in rehabilitative engineering. To begin with, the earlier in a chain of command a signal is detected, the earlier the control system responsible for the powered exoskeleton can issue an output command. For this reason, EEG and EMG could, in theory, provide with the fastest possible track to actuation, resulting in a responsive, intuitive device. The reality is that there is a considerable innate delay between the onset of the first electric signals in the brain and the realization of movement, so the fastest controller may not be, necessarily, the optimal one (Smith et al., 2011). On the other hand, an unreasonably slow controller is obviously not good either (Smith et al., 2011). Ergo, signal complexity must, too, be taken into account, and measured quantities that require a tremendous number of computations to be used are at odds with the end goal. By reason of the fact that force-based control measures the motion itself, the intention behind it should, in theory, be more closely related to the movement performed than any of the other methods. Other than trying to solely predict the intention behind the motion, force control follows the imminent motion, allowing for fast, accurate control (Gull et al., 2020).

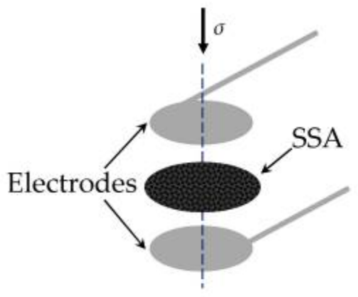

Figure 2.7: Diagram showcasing the components of Force Sensing Resistors, namely, a polymer matrix composite, the SSA, and two highly conductive electrodes. A stress σ exerted on the sensor leads to a change in its measured terminal resistance. Source: Paredes-Madrid et al. (2017).

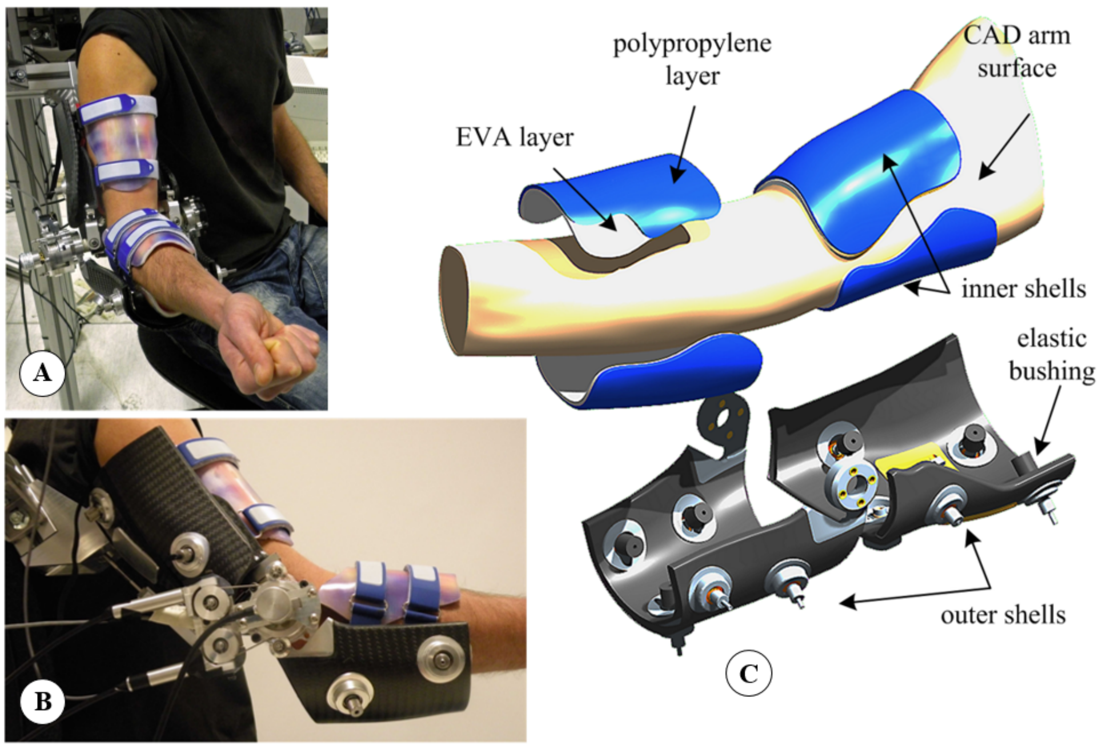

Figure 2.8: The NEUROExos exoskeleton. **A** is a front view, **B** is a lateral view and **C** is demonstrating the double-shelled link technology. Adapted: Vitiello et al. (2013).

EEG might be subject to a lot of noise compared to, say, force control, only it is relatively more accessible to use. Aside from being conceptually easier to grasp than, say, EEG, surface EMG can be implemented more simply, thanks to its electrode-based functionality. EEG is most used in research when patients suffer from severe spinal cord disabilities that make them unable to use their limbs and, consequently, have no EMG signals to be collected (Gull et al., 2020).

A downside of force control is that it is often more complex to integrate in an exoskeleton, mechanical design wise. Vitiello et al. (2013), for example, for the NEUROExos Powered Exoskeleton, shown in Figure 2.8, employed a sophisticated double-shelled links design to make the best use of the force control strategy and achieve a comfortable, ergonomic human-robot interaction.

Overall, sEMG stands out as the most commercially appropriate technology, thanks to how, in relation to the remaining technologies, practical, intuitive and cheap it is.

2.2.1 Recommended Setup for an Electromyogram and Factors Influencing the Output

Doing electromyographic analysis requires a number of conditions to be met. At the outset, the contact surface must have been properly conditioned to minimize chemical noise and interference. In other words, the skin has to be cleaned, disinfected and lightly abraded, to remove excess dead skin and improve electrical conductivity (De Luca, 1997). Depending on the subject and the muscles being studied, hair might have to be preemptively shaved (Xiao, 2019; Xiao et al., 2018a,b). The use of special conductive gels to improve conductivity is not uncommon (Ulrey and Fathallah, 2013a,b; Tang et al., 2014). In this case, the electrodes are said to be *wet*, as opposed to the regular *dry* electrodes, still preferred for their greater comfort and practicality (Bi et al., 2019).

As soon as the skin is ready for attaching the electrodes, the exact electrode positioning has to be established. In accordance with De Luca (1997), there are five main parameters for precisely defining the electrode positioning and configuration (Figure 2.9):

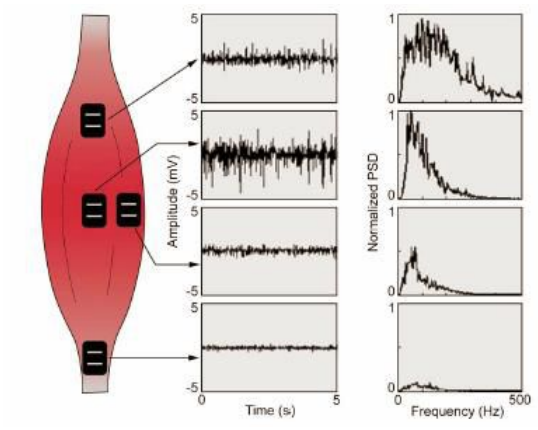

Figure 2.9: Influence of electrode positioning in the measured signal. On the left, four different placements, from top to bottom, over an innervation zone, between an innervation one and a myotendinous junction along the mid line of the muscle, between an innervation one and a myotendinous junction near the edge of the muscle, and over a myotendinous junction. The middle column shows the respective normalized amplitude responses of the electromyograms and the rightmost column shows the respective frequency responses of the electromyogram. It should be clear that the second configuration gives both the highest amplitude measurements and the most compact frequency spectrum. Source: De Luca et al. (2010).

- **Electrode configuration:** area and shape (which determine number of muscle motor units detected) and distance between electrodes (determines the bandwidth of the differential electrode configuration). Shape is usually circular with a diameter of 10 [mm]. The distance between electrodes should not be too small, otherwise the formation of sweat could shunt the signal, decreasing amplitude, signal to noise ratio and filtering out the higher frequency components. Similarly, sensors with electrodes spaced further apart detect sEMG signals having a more compressed frequency spectrum (Lindstrom, 1970). The conventional inter-electrode distance is 20 [mm], but different values are not that rare: Ulrey and Fathallah (2013a,b) use a distance of 40 [mm] and De Luca et al. (2010) uses a distance of 10 [mm].

- **Electrode location along the muscle:** distance relative to the myotendinous junction (the site of connection between tendon and muscle) affects amplitude and frequency characteristics of the signal. The preferred location is between the myotendinous junction and the nearest innervation zone, which tends to line up with the middle of the muscle.

- **Electrode location across the muscle:** proximity to the lateral edges of the muscle can lead to significant amounts of crosstalk from adjacent muscles. It is therefore recommended that the electrodes are placed along the mid line of the target muscle.
- **Orientation with respect to the muscle fibers:** affects the value of the measured conduction velocity of the action potentials and, consequently, amplitude and frequency. The recommended orientation is parallel to the muscle fibers.

These parameters are, as stated by De Luca (1997), extrinsic causative factors that influence the measured surface EMG amplitude and bandwidth. Causative factors have a basic or elemental effect on the signal and can be extrinsic or intrinsic. Factors, as defined by De Luca, are any and all variables that determine the output of an electromyogram. For this reason, the factors that influence the motor units' action potentials themselves are factors that should be taken into account and thoroughly understood. This is the case of intrinsic factors, the other subset of causative factors. To enumerate some intrinsic causative factors:

- **Number of active motor units at any particular time of the contraction**, which influences signal amplitude.
- **Fiber type composition of the muscle**, which changes interstitial fluid pH during a contraction.
- **Blood flow**. It influences the rate at which metabolites are removed during the contraction.
- **Fiber diameter**, which affects the amplitude and conduction velocity of the action potentials.
- **Depth and location of the fibers**. It determines spatial filtering and, consequently, impacts amplitude and frequency characteristics.
- **Volume of tissue between the muscle and the electrode**, determines spatial filtering and therefore affects amplitude and frequency of the signal.

De Luca (1997) defines, in addition, two more broad categories of factors affecting the measured EMG. Intermediate factors, that is, phenomena that are influenced by causative factors and, in turn, influence factors belonging to the third broad category, deterministic factors, which have a direct effect in the measured sEMG. Examples of intermediate factors are:

- **Band-pass filtering**, which is inherent to the differential electrode configuration.
- **Detection volume of the electrode**, which determines number of motor units detected.
- **Superposition of action potential signals**, which influences amplitude and frequency.
- **Conduction velocity of action potentials**. It influences amplitude and frequency.

- **Spatial filtering due to the electrode position and active muscle fibres**. This factor is a consequence of variations in relative position of the electrodes to the motor units as the muscle contracts and extends, while the skin remains mostly static. Because of this relative motion, the active motor units that are being detected can vary throughout the muscle contraction and consequently affect the signal's stability. A conclusion that follows from this is that only isometric contractions can provide the maximum stability.

- **Crosstalk from nearby muscles that contaminates the signal with noise**. De Luca and Merletti (1988) showed that, for example, in the leg, as much as 17% of electrical activity from nearby muscles may be detected on the surface of the muscle of interest. Crosstalk is a difficult problem to overcome since the tissues between and within the muscles are inhomogeneous and anisotropic, causing multiple diffractions of the electric field vectors at the discontinuities and generate multiple paths of differing impedances between the source and the detection sites, causing the detected signal to be filled with artifacts. De Luca (1997) considers using a double differential setup the optimal way for dealing with crosstalk. Crosstalk can also be detected by analyzing the bandwidth of the signal - whereas local EMG produces broad frequency spectrums, signals from more distant muscles that suffered spatial filtering during travel will come at more discrete frequencies.

Finally, the following are examples of deterministic factors:

- **The number of active motor units**.

- **Motor unit force-twitch**.

- **Mechanical interaction between muscle fibres**.

- **Motor unit firing rate**.

- **The number of detected motor units**.

- **Amplitude, duration and shape of the muscle unit action potentials (MUAP)**.

- **Recruitment stability of motor units**.

2.2.2 Acquisition, Preprocessing and Feature Extraction of the Raw EMG Signal

The raw, unprocessed signal of surface EMG, obtained immediately at the electrodes, is essentially useless (De Luca, 1997). It suffers from several problems that make it very hard to extract any sort of information, unless properly treated. The stages that precede motion estimation can be summarized as three (Bi et al., 2019): acquisition, amplification and sampling of the EMG signal, preprocessing of the sampled signal and, finally, feature extraction from the stream of EMG input data. Over the next few pages, the process required to transform the raw EMG signal into an extremely rich source of data will be comprehensively detailed.

Acquisition and Amplification

Firstly, the EMG signal has to be measured using a differential setup. This is due to its low amplitude in relation to other environmental signals that could otherwise easily interfere with it (De Luca, 1997). This differential arrangement acts as a comb band-pass filter or, if the spacing between electrodes is chosen so as to not alias the EMG signal, the differential electrode should behave as a high-pass filter, since the entire spectrum of the signal fits in the low end of the band-pass filter (De Luca, 1997). As mentioned before, the differential setup greatly helps, too, with avoiding crosstalk noise in the signal (De Luca, 1997).

The low voltage of the motor unit's action potentials has to subsequently be amplified and sampled, as recommended by De Luca (1997), with a CMRR (Common Mode Rejection Ratio) of at least 80 [dB]. A device's CMRR is the ratio between the differential voltage gain to the common mode gain (Fischer-Cripps, 2002). That is, if a signal was to be applied equally to both inputs of an operational amplifier (a common mode voltage), so that the differential input voltage was unchanged, the output should, in theory, be unaffected. In practice, there is a deviation in the output voltage that stems from this common mode voltage and is related to the differential voltage gain. This relationship is what the CMRR describes. A CMRR of at least 80 [dB] allows for the rejection of common mode noise with some degree of certainty.

According to the Nyquist sampling theorem, the rate at which a signal is sampled should be at least twice the highest frequency of interest in the signal. The brunt of the surface EMG signal is capped at around 500 [Hz] (De Luca, 1997) and the higher frequencies are largely explained by noise and other artifacts, as seen before. To prevent aliasing problems, the signal should, in theory, be sampled at a frequency of, at least, 1000 [Hz]. A sampling rate of about 1000 [Hz] does seem to be the norm across the literature, even though values of 2000 [Hz] (Theurel et al., 2018), 4000 [Hz] (Colebatch et al., 2016) and 5000 [Hz] (De Luca et al., 2010) were also found. In the cases where sampling rate was different from the norm, no explanation was found.

Sampling at rates below 1000 [Hz] is generally against convention. Nevertheless, the disadvantages of using higher sampling frequencies are clear - the components have to be more powerful and the control loop faster. By opposition, using a lower sampling frequency allows costs to be slashed, components to be smaller and the system response speed to be faster (Li et al., 2010). Li et al. (2010) tested the effect of sampling rate on the classification performance of an LDA (Linear Discriminant Analysis, refer to Section 2.3.1 for a study of intention estimation algorithms) classifier for 11 motion classes. The original sample, obtained at a sampling rate of 1000 [Hz] and band-pass filtered (5 to 400 [Hz]), was downsampled with sampling rates ranging from 1000 [Hz] to 100 [Hz], in increments of 20 [Hz]. The authors found a meaningful relationship between classification accuracy and sampling rate, as depicted in Figure 2.10.

However, the loss of classification accuracy between 1000 [Hz] and 400 [Hz] was not found to be statistically significant ($p = 0.15$, *t-test*), the second being 1.3% lower than the first. Oppositely, when downsampling to 300 [Hz], the classification accuracy drop of 2.3% was found to be statistically significant ($p = 0.02$, *t-test*). In short, the authors asserted that sampling rates as low

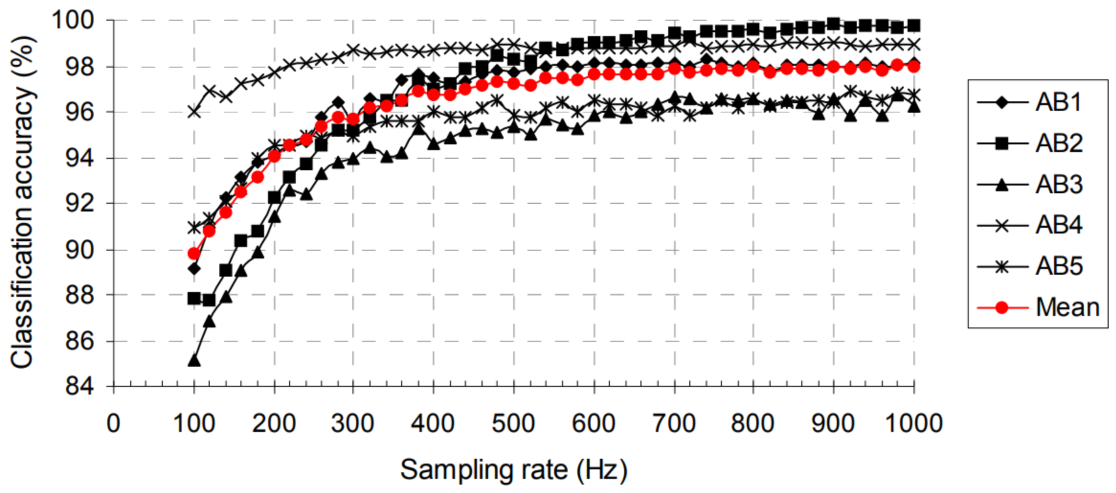

Figure 2.10: Classification accuracy of an LDA classifier versus sampling rate in five able-bodied subjects. For sampling rates above 400 [Hz], the loss of accuracy was found to be statistically insignificant. Source: Li et al. (2010).

as 400 to 500 [Hz] could be optimal when attempting to control a powered prosthesis in real time.

Noise Reduction and Filtering

Common mode rejection is just the tip of the iceberg when it comes to addressing noise in surface EMG data. Aside from mandating amplification, the low voltages at work in electromyography represent a massive vulnerability to noise. De Luca et al. (2010) distinguishes several factors as potential sources of noise, separating them into baseline noise and movement artifact noise, as shown in Figure 2.11. Baseline noise is attributed to both extrinsic noise sources and intrinsic noise sources. Extrinsic noise sources, which can be targeted with modern electronics and proper circuit design, are:

- Power line noise;
- Cable motion artifacts.

Intrinsic noise sources are generally static and their attenuation is thus easy to ensure. According to the authors, they are:

- Thermal noise originated in the electronics of the amplification system;
- Chemical noise originated at the skin-electrode surface.

Movement artifact noise, on the other hand, is due to the relative motion of the motor units to the electrode, more specifically:

- The relative motion between muscle and skin;

- External forces travelling through the body.

And is, therefore, highly time-variant with a lot of underlying dynamics. For this reason, it typically constitutes the brunt of the intellectual obstacle.

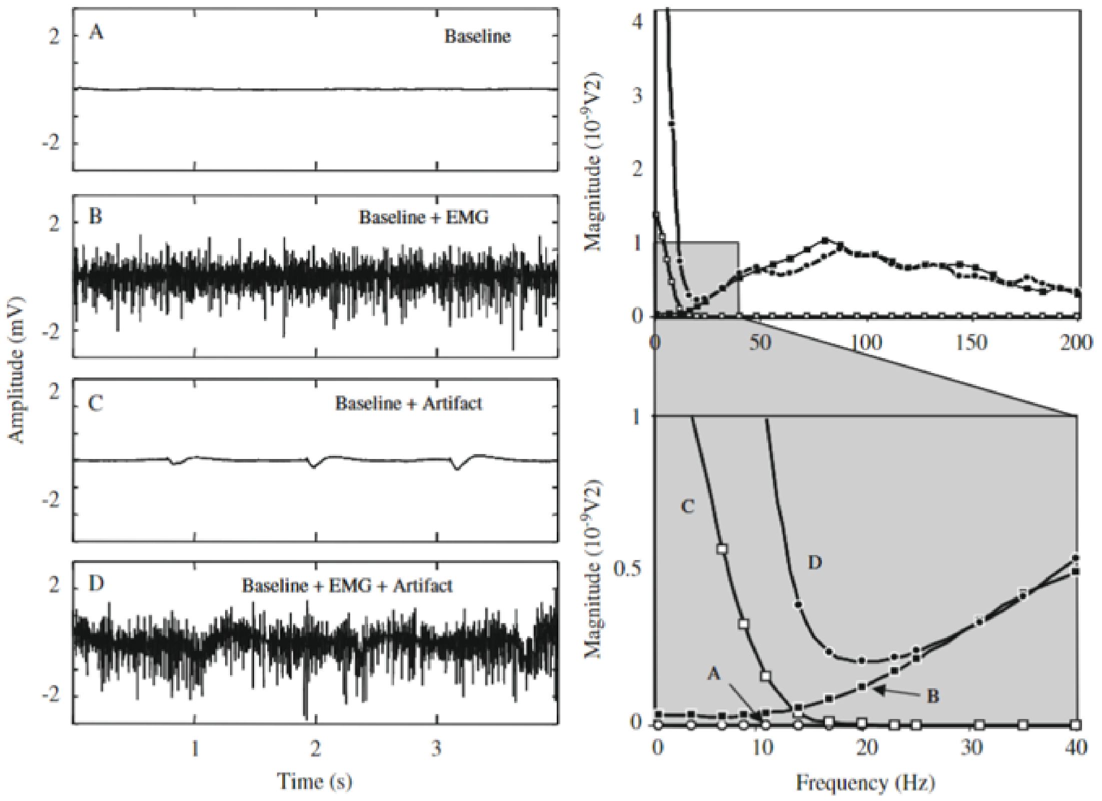

Figure 2.11: De Luca et al. (2010) studied the effect of baseline noise and an induced movement artifact noise in the sEMG signal and the performance of different high-pass filters when dealing with these inconveniences. The time domain and frequency domain characteristics of (**A**) the baseline noise (obtained during a 0% MVC), (**B**) the sEMG signal (naturally contaminated by baseline noise, obtained during a 50% MVC), (**C**) the induced movement artifact noise (naturally contaminated by baseline noise, during quiescent baseline conditions) and (**D**) the sEMG signal contaminated by baseline noise and movement artifacts (during a 50% MVC). The spectra of the individual signals are shown on the right, with the lower traces showing the expanded low-frequency portion. Source: De Luca et al. (2010).

To remove noise from a signal, knowing what the signal should ideally look like is vital. In the case of the electrical signals produced by the motor units, it is known that they occur in a bandwidth range somewhere between approximately 20 and 500 [Hz] (De Luca, 1997). Therefore, an evident step that should be taken is filtering anything that is out of this range. This can be accomplished in a plurality of ways. The most straightforward way would be to run the signal through a band pass filter, but this cannot be done on imprecise guesses of the cutoff frequencies and roll-off. De Luca (1997) recommends a roll-off of at least 12 [dB/oct].

Regarding the low-pass cut-off frequency, since EMG signals are less intense on the higher end of the spectrum, various authors use and recommend a value slightly lower than 500 [Hz].

SENIAM guidelines recommend a near 500 [Hz] low-pass cut-off (Stegeman and Hermens, 2007). Usually, 450 [Hz] filtering with a 4$^{\text{th}}$ order Butterworth (roll-off of 24 [dB/oct], rather sharp to ensure high frequency noise is promptly eliminated) is done (Whitfield et al., 2014; De Luca, 1997; De Luca et al., 2010).

With respect to the high-pass filtering, a lot more thought is given to the filter selection, by reason of movement artifact noise being more prevalent at the lowest frequencies. SENIAM guidelines recommend a 10-20 [Hz] high-pass cut-off (Stegeman and Hermens, 2007). De Luca et al. (2010) tested three different values for the high-pass cut-off frequency, at 10, 20 and 30 [Hz]. Exclusively attending to metrics such as signal-to-noise ratio was deemed inadequate, for noise was still very present at higher frequencies and its removal, while effective, would come at the expense of equally large amounts of surface EMG information. As a matter of fact, as the corner frequency was increased beyond 30 [Hz], loss of the EMG signal became more pronounced than that of movement artifacts. At 40 [Hz], the loss of sEMG was determined to be as high as 13%. The authors highlighted that this issue would be felt the most in less intense contractions of the target muscles, since the surface EMG signals tend to be the faintest there. Another problem could arise from the distortion of the frequency spectrum of the signal, which the authors warned as most pronounced when the corner frequency is the largest. The value recommended by the authors, which allowed for the highest noise reduction while preserving the most signal, was 20 [Hz].

As mentioned above, another potential source of noise is the power line noise (De Luca et al., 2010). In an electric circuit, power line noise exists in an extremely narrow interval of frequencies, ordinarily around a preset nominal value of 50 or 60 [Hz], depending on what the national standard is. The quintessential way of eliminating this noise is by utilizing some sort of band-reject filtering. This can itself be achieved in a variety of ways, with notch filters (Mewett et al., 2001), the regression-subtraction method (Mewett et al., 2001), the spectrum interpolation method (Mewett et al., 2001) or, surprisingly, state-of-the-art methods such as IIR notch filters with non-zero initial conditions (Piskorowski, 2013).

IIR filters, or Infinite Impulse Response filters, are a type of recursive filters, that is, filters that not only use past input values as a means to produce an output, but additionally consider past outputs in their computations (Smith, 2003).

These filters offer frequency selective gain, and can thus function as band-pass or band-reject filters. Figure 2.12 shows the frequency responses for the band-pass (a) and the band-reject (b) filters, as well as the band-reject step response (c). The band-reject variant, *id est*, an IIR notch filter, comes in extremely handy, for it is capable of filtering out very specific noise, in this case, the 50 [Hz] power line “hum”.

Rectification and Smoothing

The instantaneous value of sEMG is seldom useful due to its random nature. As De Luca (1997) notes, the instantaneous value of sEMG has a large stochastic component that is highly

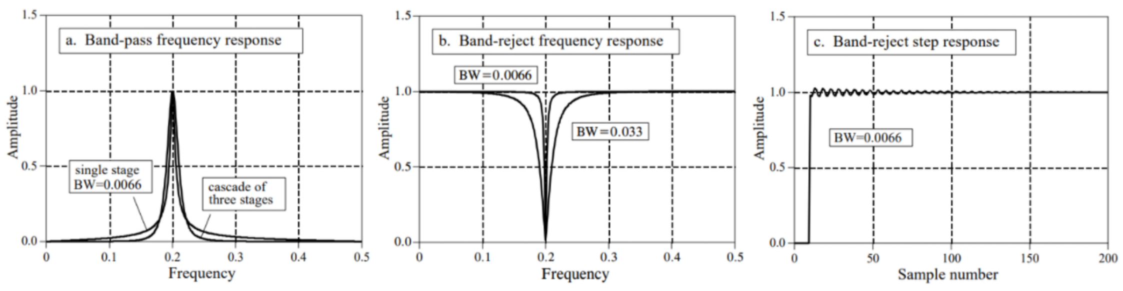

Figure 2.12: The IIR band-pass and band-reject filters. One the left, the frequency response for the IIR band-pass filter, in the middle, the frequency response for the IIR band-reject filter, or notch filter, and, on the right, the step response for the notch filter. There is some visible overshoot and ringing in the step response of the notch filter, but it should be noted that it is not expected to have a significant impact on the EMG signal. Source: Smith (2003).

variable around a mean value. On account of this, the signal has to be smoothed to extend the action potential spikes and improve the quality and fidelity of the sample.

Unfortunately, the bipolar nature of the EMG signal implies that the habitual smoothing techniques, like time-domain filters and high-selective acquisition systems, don't work, seeing that the average of a symmetric, bipolar signal should be, to some approximation, zero. What is more is that datasets with an average of zero are not particularly useful from a statistical analysis standpoint. If the measured signal doesn't average zero, correction for baseline offset can be performed, for instance, by subtracting the mean from all data points (Bi et al., 2019). There are two main methods for getting around this problem (Negro et al., 2015): rectification and Fourier transforms. The repercussions of the application of Fourier transforms are complex and its use fairly rare and will therefore not be broached here.

Rectification consists of taking the absolute value of EMG, achieved by either eliminating the negative portions (half-wave rectification) or changing the signal of the negative part (full-wave rectification) (Figure 2.13). It is an exceptionally trivial process with advantages that are quick and clear to understand. Its impact should not, however, be understated. At the end of the day, it consists of a non-linear transformation of the signal, which can have serious, unexpected consequences when attempting to extract features and apply regression algorithms (Negro et al., 2015). The truth is rectification acts, beyond intended, as a low-pass transfer of function and as such dampens the higher frequency components of the signal. This morphs the original frequency spectrum into something else, makes the end result more sensitive to the duration and magnitude of action potentials and can introduce peaks of coherence in the signal, that is, false patterns can appear in the rectified EMG that were otherwise nonexistent in the original electromyogram (Negro et al., 2015).

If certain data points are considered frivolous or extravagant, removal of outliers or application of thresholds can be considered. This should only be done after most data transformations, to prevent unnecessary warping of the data. With regards to removing outliers, the most common approach other than smoothing (which will be covered just next) would be to exclude all

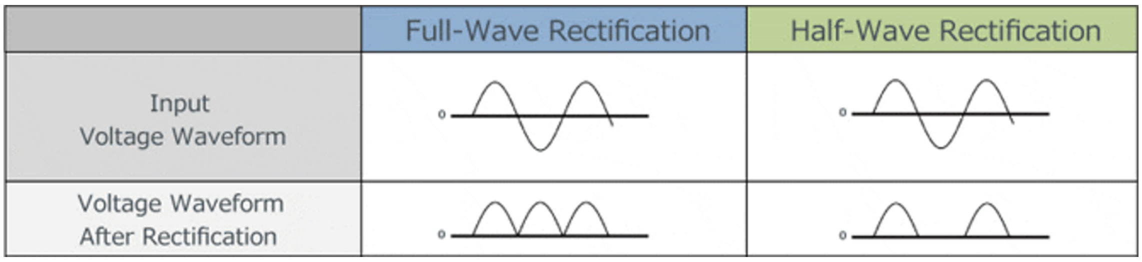

Figure 2.13: A rectifier's voltage output for an input sinusoid. Full-wave rectification returns the absolute value of the input, whereas half-wave rectification only returns the positive parts of the original sine wave. Source: ROHM Co., Ltd. (2020).

data points that are positioned a minimum number of standard deviations away from the mean (Whitfield et al., 2014). The usage of thresholds, on the other hand, can be slightly more nuanced. Roberts and Gabaldón (2008) recommends removing all data that is within one to two standard deviations of the signal. Naturally, removing data can have unintended consequences, so it shouldn't be done arbitrarily. Nowadays, thresholds are valued the most in feature extraction of unique characteristics for motion intention estimation.

At last, smoothing can be performed. The smoothed signal, as was previously mentioned, is a more reliable, more stable version of the original signal. For the most part, smoothing removes outliers and stochastic patterns in data, making it in total less noisy, at the potential cost of information. The smoothed signal boasts multiple treasured properties, though, that can make it better suited for regression algorithms and machine learning (Rocha et al., 2012), the prevalent means of interpretation and information extraction in the field.

The run-of-the-mill smoothing method, simple, effective and uncostly, is applying moving averages (rolling averages) in the time domain, effectively replacing every data point sampled to the moment, within a time window of length T_a (Figure 2.14), for the average of all data points in that same window, just like an uniform one-dimensional filter. Moving averages can, alternatively, be weighted, like the Hamming and Hanning windows, as well as triangular and rectangular windows (Rocha et al., 2012). Window lengths vary significantly across literature, more often than not between 10 and 500 milliseconds (Rocha et al., 2012). Nevertheless, the selection of the window length is not trivial.

According to Smith et al. (2011), the implications of moving averages are such that any given estimation at any point in time results from all data points sampled during the pertaining analysis window. This estimation consists of a computation that puts an algorithm into effect, ultimately introducing a delay, equal to the processing time, T_d (Figure 2.14), of a single step of said algorithm, into the control process. Simultaneously, the system is also generating new batches of samples, and the time needed to advance the analysis window is given by T_{inc} (Figure 2.14), that is, the length of the window increment. As T_d approaches T_{inc}, idle time converges to zero and the processing stream becomes as dense as possible. If T_d is bigger than T_{inc}, the system will end up skipping samples, and thus data is lost and the actual processing delay is increased, due to idle time. Since a movement command results not only from the samples from the latest increment,

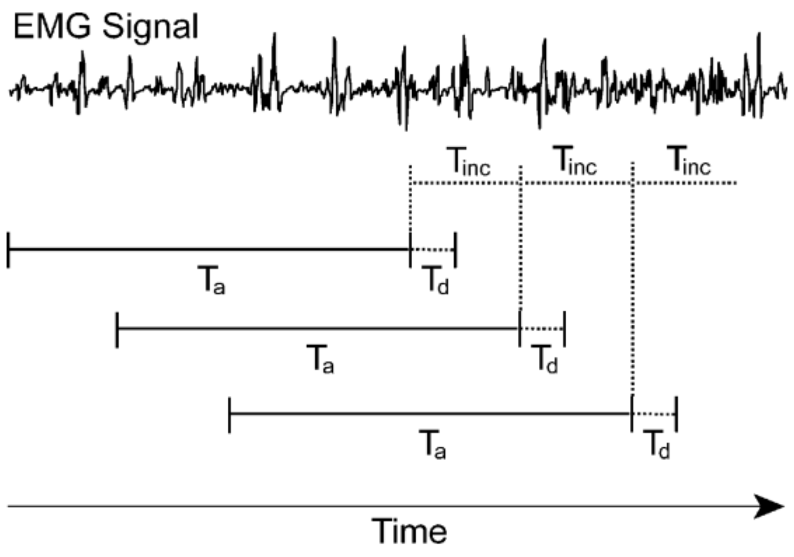

Figure 2.14: Relevant time frames of the EMG signal during processing. T_a, the length of the moving window, T_{inc}, the time increment needed for the next iteration, and T_d, the time needed to process the last window of data. Source: Smith et al. (2011).

rather from all the samples taken during the totality of the analysis window, delay is not only a function of the processing time, T_d, but also of T_a and T_{inc}.

This would make it tempting to significantly reduce the moving window average length, however, as Lock et al. (2005) showed, the average classification error increases with shorter window lengths. Smith et al. (2011) confirmed that lengthening the window led to higher accuracy in movement estimation, since the classifiers had more data to work with at each iteration, up until the point where the delay increased so much that the estimations caused overshooting. In addition, even before this point, increasing the delay had its disadvantages, as it generally led to a decrease in user-friendliness of the system, since users noticed the interference of excessive delay with their intents. Ultimately, the authors (Smith et al., 2011) determined a window length of 250 [ms] to be the best choice.

A more sophisticated alternative that has been explored a few times in the literature is to employ an adaptive window length. It has been shown that window length should be varied depending on certain factors, such as walking pace (Lobov et al., 2019). In principle, the dependence of output signal characteristics on the length of the smoothing window could, too, justify an adaptive window and a general rule could be formulated as such: as the EMG amplitude changes more rapidly, the window length should be decreased, and if the rate of changes decreases, the window should be widened. Regardless, studies are yet to show sufficiently conclusive results that legitimize adaptive transformation's increased computational costs (Lobov et al., 2019).

Normalization, EMG-Force Relationship and Feature Extraction

One of the greatest downsides of the EMG signal is that when rectified and sufficiently smoothed for use, it shows very little quantitative relationship to the torque about a joint, and even if the existence of a qualitative relationship is clear, the instantaneous EMG signal is mostly random (De Luca, 1997).

There are a variety of factors that impede such a straightforward quantitative approach to be used with EMG signals. As is well established, the ideal conditions for studying surface EMG sinals is during isometric contractions, which is of little interest when devising, say, a proportional myoelectric control system (refer to Section 2.3.1). The presence of movement artifacts in EMG, as previously seen, is well documented (De Luca et al., 2010), and surface EMG, specifically, suffers from significant nonstationarity of the signal caused by relative motion of the electrode to the motor units. When studying EMG signals in anisometric contractions, De Luca (1997) recommends splitting the analysis into multiple near-isometric epochs of the record and extrapolate from there. In applications where periodic movements are expected (for example, in gait), the epoch could be extended to a whole period.

Nonetheless, even when restricting sampling to short epochs, the EMG-force relationship is hardly rigid. For starters, the electrode only covers part of the muscle activity, so it is essentially a sample that can easily get biased if motor unit activation isn't completely uniform. This is why HD-EMG (High Density EMG), sensors that cover a muscle with tens or hundreds of electrodes, has recently been gaining noticeable traction (Bi et al., 2019). Unfortunately, while it is well-established that increasing the number of electrodes leads to performance improvements, Hahne et al. (2014) showed that diminishing returns could plague HD-EMG and lead to a quick saturation of performance.

In addition, as De Luca (1997) notes, the EMG-force relationship is highly nonlinear, showing saturation of force and different patterns of recruitment depending on muscle composition (not all muscles show the same relationships between developed force and its respective surface EMG signal, as shown in Figure 2.15), loading of the joints, patterns of muscle activation during an anisometric contraction (as the load, say, the limbs, moves through space, the required torque changes with time) and muscle length and geometry. In more complex, compound movements, compensatory strength is, evidently, another problem that can confound interpretation of the sEMG signal.

A common practice is to normalize the data with respect to the maximal isometric force a subject can generate at a monitored joint. Nonetheless, there is evidence that this is not without downsides (De Luca, 1997).

Unfortunately, there is no certainty that the so-called maximal isometric force is, indeed, maximal. There's a reasonable chance that the subject's contraction might have been sub-maximal. On the other hand, this force might not have originated in exactly the way the researchers assumed. Compensatory strength mechanisms might be at play. In order to estimate, to one's best ability, the maximal isometric force, the individual must be tightly restrained during the contraction, in an effort to minimize compensatory contributions of other muscles. This is obviously a convoluted,

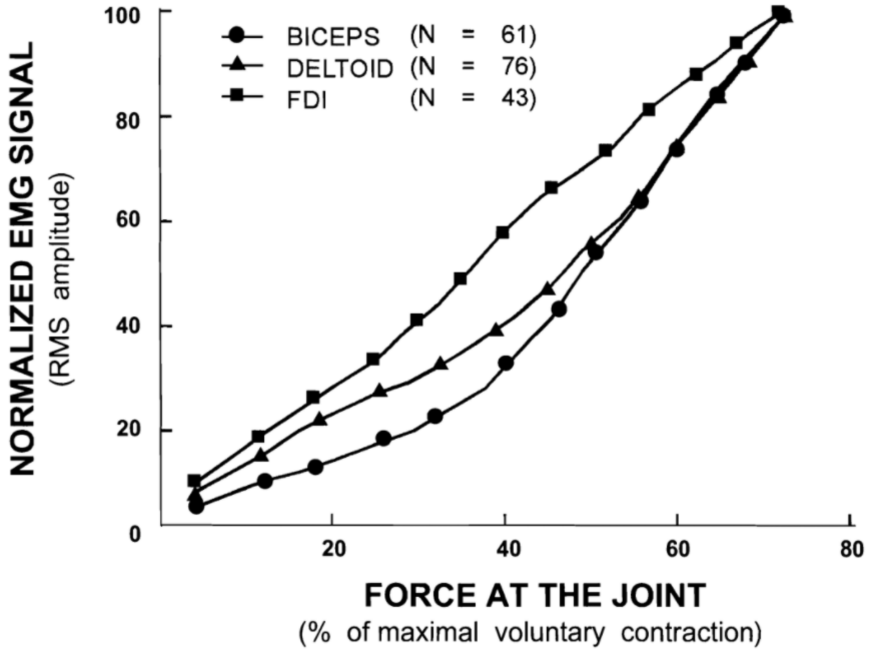

Figure 2.15: Normalized force-EMG relationships for three different muscles, with force charted as a percentage of the force developed during MVC. Source: De Luca (1997).

impractical task that requires extensive knowledge of biomechanics. De Luca (1997) suggests a strategy for determining the maximal isometric contraction EMG response. The author also suggests not using the MVC when normalizing the data and using about 80% of it, since the absolute maximum value would be too unstable and unreliable.

Consequently, when trying to estimate motion intention, the recurrent practice is to extract, besides the smoothed, rectified EMG signal, called the MAV (Mean Absolute Value), several other features. This is called feature extraction, and the result is a feature vector, a representation of features reduced from the raw EMG data that, in theory, better and more accurately describe the data that is being recorded by the system (Phinyomark et al., 2013). Features can also be selected as a mechanism of eliminating noise. This is done by selecting for features that are less sensitive to, say, white Gaussian noise (Phinyomark et al., 2008).

As a matter the fact, the inclusion of feature extraction in the EMG data preparation pipeline is so commonplace that a variety of features and even sets of features, like the fabled Hudgin's set (Hudgins et al., 1993), are, at present, well-defined.

sEMG features can be divided into three distinct categories: time-domain, frequency-domain and time-frequency or time-scale. Time-frequency features cannot be used by themselves and, on top of that, suffer from high dimensionality, thus requiring the application of dimensionality reduction techniques (Phinyomark et al., 2012, 2013). Ergo, they were not further investigated in the present thesis.

Conversely, time-domain features generally do not require transformations and are easy to use (Phinyomark et al., 2012, 2013). They are, however, most vulnerable to interference, especially when motion is dynamic, particularly those that are energy based. Elseways, frequency-domain features are most used for characterizing the fatigue of muscles and other spectrum-related phenomena (De Luca, 1997).

Phinyomark et al. (2012) divided time-domain features into four distinctive groups, according to the type of information provided by them:

- Energy and complexity features: measures of signal amplitude and energy, like the MAV (Mean Absolute Value), the WL (Waveform Length) and the SampEn (Sample Entropy) features.
- Frequency information features: like the ZC (Zero Crossings), the MYOP (Myopulse Percentage) and the WAMP (Willison Amplitude).
- Prediction model features: features that fit regressions to the signal, like the AR (Auto-Regressive Coefficients) and the CC (Cepstral Coefficients) features (Kang et al., 1995).
- Time-dependence features: measures of the signal's behaviour across time, such as MAVS (MAV Slope).

Regardless of final choice, the following remains true: not all features are created equal. Numerous studies have been conducted in an attempt to determine which features and sets of features provide the highest level of unique and valuable information for classification and proportional control purposes, the former being the most frequent motivation of the two.

Most research pertaining to myoelectric control systems (MSCs) is performed over discrete intervals of time, in relatively brief experiments. While some efforts have been done to study EMG and its classification algorithms across long periods of time, they're not common. Phinyomark et al. (2013) investigated the robustness of fifty time-domain and frequency-domain features to the consequences of recording fluctuating EMG across a period of 21 days, with five to six trials per day. The authors compared the performance of LDA (Linear Discriminant Analysis), a robust, low-cost, high performance classifier, to two other classification algorithms, RF (Random Forests, an ensemble of decision trees) and QDA (Quadratic Discriminant Analysis, a technique that generalizes LDA to quadratic surfaces, rather than linear). The classifiers were also tested for three overlapped window sizes (T_a of 125, 250 and 500 [ms]) and three increment interval sizes (T_{inc} of 62.5, 125 and 250 [ms]).

The best performing robust single feature was, across the board, SampEn (Zhang and Zhou, 2012), with a 93.37% accuracy. As expected, classification accuracy was found to increase with sets of more than one feature. The best performing two, three and four feature sets were, respectively, SampEn + CC, SampEn + CC + MAV2 (Modified Mean Absolute Value 2, Oskoei and Hu (2008)) and SampEn + CC + RMS (Root Mean Square, Kim et al. (2011)) + WL or AAC (Average Amplitude Change, Kim et al. (2011)), the last with a 97.75% accuracy. It should be noted that the CC feature represents, by itself, a four-dimensional vector, so the four feature set has, in actuality, a dimension of seven.

In addition, LDA yielded the best performance out of all three classifiers, as seen in Figure 2.16, not only in classification accuracy, but in variability and computational cost. The 500 [ms] window provided the highest performance, however the increase from 250 [ms] was barely noticeable, in contrast to the jump from 125 to 250 [ms]. The impact of T_{inc} was deemed negligible by the authors.

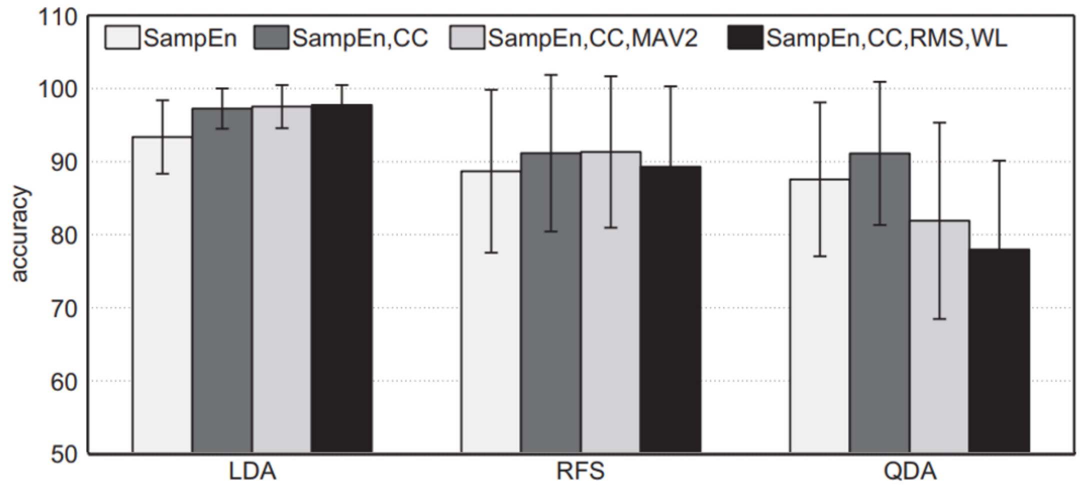

Figure 2.16: Average classifier test accuracy (%), as determined by Phinyomark et al. (2013), for the best single and multiple-feature sets, when trained in the first five trials using the LDA, RFS and QDA algorithms, with a window size of 250 [ms] and and increment interval of 125 [ms]. Error bars correspond to the first standard deviation for each bar. Source: Phinyomark et al. (2013).

In thermodynamics, entropy is a measure of the total number of possible microscopic states that a system can adopt, given its macroscopic state. When applied to other fields, the concept of entropy can provide information on the complexity, randomness and rate of information creation of a dynamic system. With regards to the EMG signal, a time series, entropy should, in theory, be capable of measuring alterations of state and detect the onset of muscle activation, as a burst of voluntary muscle activity would be accompanied by a similar increase in complexity, which should not be the case for spurious spikes of activity.

These spurious spikes of muscular activity are an unexpected downside of the advancement of signal acquisition technologies (Zhang and Zhou, 2012). As signal fidelity improves, the greater signal to noise ratio of surface EMG has the unintended consequence of exposing most traditional EMG features to spurious background spikes, caused by anomalous motor unit activity (all the more important in stroke and spinal cord injury patients), momentary electrode displacement, electrical interference and other parasitic sources, that they wouldn't, otherwise, be as sensitive to.

Zhang and Zhou (2012) compared the Sample Entropy feature with two other onset detection strategies, the AMP-based method (amplitude thresholding of rectified EMG) and a TKE-based method (a Teagar-Kaiser Energy operation on raw EMG). The latency of the SampEn detection threshold was found to depend on the sample's SNR. In spite of showing a middle-of-the-pack latency for low noise samples, SampEn vastly outperformed all other features for the noisy data group, with an averaged onset detection latency of a mere 20 [ms] and a comparably small variance at that.

Given a data series $N = \{x_1, x_2, ..., x_N\}$ with a constant time interval τ, SampEn is the negative natural logarithm of the probability that two similar sequences of equal length m remain similar at the next point $m+1$, by counting each vector over all the other vectors except on itself. The mathematical expression for the sample entropy, given a time series of length N with a constant time interval τ and a template vector of length m, is expressed in Equation 2.1 (Zhang and Zhou, 2012):

$$SampEn(m,r,N) = -\log\left[\frac{A^m(r)}{B^m(r)}\right] \tag{2.1}$$

where $A^m(r)$ and $B^m(r)$ are the number of template vectors at a distance, respectively, greater and shorter than r. r, computed according to a distance function (typically the Chebyshev distance), quantifies the similarity between the two template vectors. Phinyomark et al. (2013) recommends values of 0.2 and 2 for the variables r and m.

The cepstrum (the name derived by reversing the order of the first four letters of spectrum) of a signal is defined as the inverse Fourier transform of the logarithm of the magnitude of the power spectrum of the signal data (Zecca et al., 2002). The cepstrum of a signal provides information about periodic structures and the underlying dynamics in a signal's frequency spectrum - it is, in a sense, the spectrum's spectrum - and has multiple applications in signal processing.

Unfortunately, as discussed in Section 2.2.2, the instantaneous EMG signal is highly stochastic, nonstationary and nonlinear. In consequence, to obtain its power spectrum, an approximation is necessary. In a short time interval, it can be regarded as a stationary Gaussian process and x_k, the signal's magnitude at a set time k, modeled with an n^{th} order autoregressive (AR) model (Zecca et al., 2002), such as:

$$x_k = \sum_{i=1}^{n} a_i x_{k-1} + e_k \tag{2.2}$$

where a_i are the model's coefficients and e_k is the residual white noise. The coefficients can be obtained through an optimization algorithm.

Now that an approximation to the EMG signal is available, its cepstrum can be determined. The system function $H(z)$ for the AR model in the z-domain is:

$$H(z) = \frac{1}{\sum\limits_{l=1}^{n}(a_i z^{-1})} \tag{2.3}$$

As already mentioned, the signal's cepstrum is the inverse Fourier transform of the logarithm of the magnitude of the power spectrum:

$$C(z) = \ln(H(z)) = \sum_{l=1}^{\infty} c_i z^{-1} \tag{2.4}$$

Substituting Equation 2.3 in Equation 2.4, differentiating both sides with respect to z^{-1} and equating the coefficients of like powers of z^{-1}, the Cepstral coefficients can, at last, be computed directly from the AR model's coefficients (Kang et al., 1995):

$$\begin{aligned} c_1 &= -a_1 \\ c_i &= -a_i - \sum_{l=1}^{i-1} \left(1 - \frac{l}{i}\right) a_l \, c_{i-l}, \quad 1 < i \leq P \end{aligned} \tag{2.5}$$

where P is the AR model's order, i.e., it means that the approximation for the cepstrum should have an order, at most, as high as the AR model's. An order of 4 was chosen for both the AR model and the cepstral regression, with each coefficient being fed into the algorithm as a distinct feature.

The root mean square (RMS) has an extremely simple mathematical definition, being:

$$RMS_k = \frac{1}{N} \sum_{i=1}^{N} x_i^2, \quad t_i \in [t_{k-1}, t_k] \tag{2.6}$$

Its simplicity should not, nevertheless, detract from its worth. It is, still, one of the most widely used features for quantifying the magnitude of a sample and, in EMG analysis, it holds great descriptive power (De Luca, 1997).

A signal's waveform length is the cumulative length of the signal over an interval of time, that is, it is the sum of the absolute of all differences between every consecutive point in the time segment (Zecca et al., 2002):

$$WL_k = \sum_{i=1}^{N} |\Delta x_i|, \quad t_i \in [t_{k-1}, t_k] \tag{2.7}$$

Not only does it provide information regarding the signals amplitude, its sensitivity to frequency makes it a carrier of frequency and time information, too.

2.3 Control Strategies for Powered Exoskeletons

Exoskeletons are, essentially, human-robot collaborations (Bi et al., 2019). Unlike cooperation tasks, in which the human and the robot are unaware of what the other is doing and, instead, focus on their job, which is complete when both parts have performed their respective functions, collaboration tasks require a much higher level of interaction, such that both partners need to know, at all times, where their partner is, in order to ensure not only a swift performance but also the safety of both agents. To this end, human-robot collaborations require a vastly superior amount of information, gathered from sophisticated sensors, inferring power, present in the form of advanced control algorithms, and fast, effective feedback, to avoid, at best, conflict of interests. A diagrammatic representation of an exemplary control system behind a powered exoskeleton can be seen in Figure 2.17.

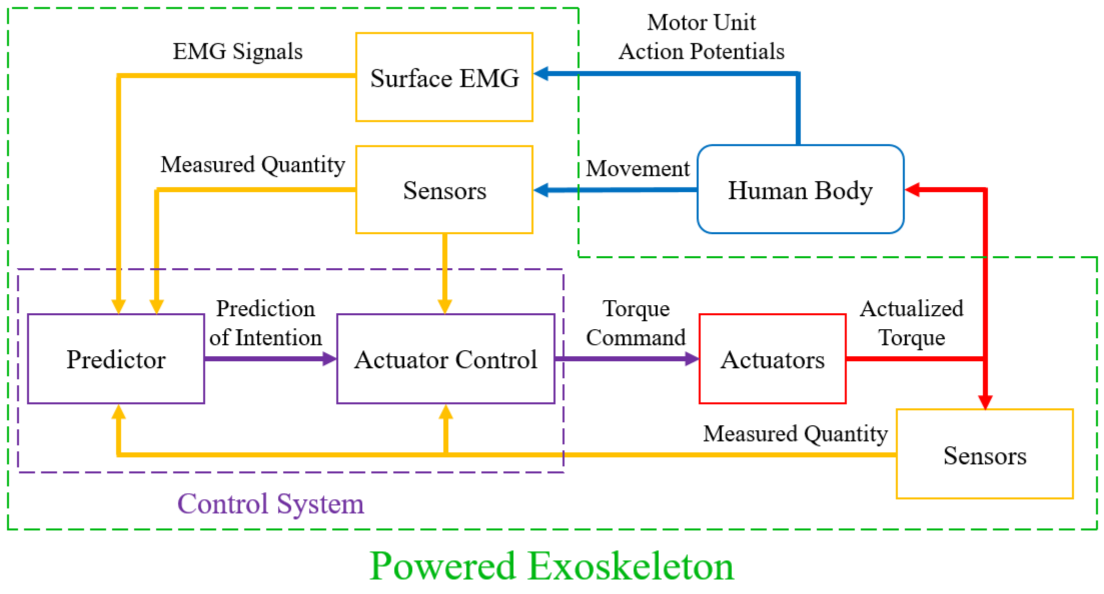

Figure 2.17: A possible control solution for a powered exoskeleton, with sensors, EMG-based intention estimation and torque control. The measured quantities could be angular positions, velocities, inertial measurements, etc.

This section pertains to the control strategies and modes used in powered exoskeletons.

2.3.1 Main Types of Control Strategies

Once all stages of EMG data treatment are complete, the newly acquired information is ready to be fed into the control algorithm. Generally speaking, there are two types of control methodologies in EMG-based intention estimation (Bi et al., 2019): model-based and model-free.

2.3.1.1 Model-Based Control

In model-based control, a relationship is established between variables of the problem, allowing for the development of a model that is grounded, to some degree, in reality. They can be kinematic, dynamic or musculoskeletal, and are the basis for the control system, that acts within the model constraints. Model parameters are adjusted until the model behaves as expected.

Kinematic Model

Kinematic models are simplified models that do not take into account inertial properties of the human bodies and the devices (Bi et al., 2019). Instead, they deal only with movements, trajectories, velocities, accelerations, and orientations. The human is thus modeled as a set of several rigid limbs that are connected through joints that have a determinate number of degrees of freedom, with the intent of replicating human kinematics. This results in a relatively concise model that does not require extensive anatomical knowledge, apart from basic functional and anatomical understanding of the human body.

When dealing with upper limb exoskeletons, the kinematic model (Bi et al., 2019) is frequently employed with a total of 7 degrees of freedom (if the upper limb model is unilateral), that is, three for the shoulder joint, two for the elbow joint and two for the wrist joint. Fingers are typically not considered in this approach.

Evidently, kinematic models suffer from egregious accuracy problems and are barely applicable when energetically dynamic motion is expected.

Dynamic Model

In essence, dynamic models are an extension of kinematic models. By considering, in addition to the baseline kinematics, the weight and mass distributions of limbs, they allow for a very accurate representation of the dynamics of the human body. This is oftentimes used in tandem with muscle force estimation from either EMG signals or FSRs (Bi et al., 2019).

When it comes to the implementation and parametrization of these models, the control algorithms can be quite varied. Both run-of-the-mill least squares optimization algorithms and state-of-the-art artifical neural networks can be found in the literature (Bi et al., 2019), to the point where choice becomes terrifically dependent on project objectives and restrictions.

Musculoskeletal Model

If dynamic models improve on kinematic models, musculoskeletal models go above and beyond to precisely mimic the properties, characteristic behaviours and dynamics of the limbs being studied, taking into account the stark differences between the tissues, organs and structures that make up each limb and body part, because, ultimately, they have an effect on the mechanical behaviour of the human-robot system.

The level of complexity can vary immensely, yet the predominant approach is the Hill muscle model, that reduces muscles to a contractile component (CC) in series with an elastic component (SEC) (Miller, 2018), in order to somewhat approximate the muscle behaviour, dynamics and underlying relationships between relevant variables. The redundancy of human kinematics becomes chiefly pertinent in this approach, posing a great challenge (Bi et al., 2019).

2.3.1.2 Model-Free Control

With model-free control, rather than grounding the control system's inferential capabilities in palpable analytical relationships, the curated EMG data is fed into a classification or regression algorithm that internally establishes black-box type relationships between the input variables and the intended output behaviours.

Classification and regression algorithms are used in EMG-based control function in identical ways. Put simply, they receive a set of inputs that are mapped to a set of outputs and the whole algorithm is optimized in an effort to achieve the best predictive performance (Bi et al., 2019).

When the algorithm is a classifier, the inputs are used to estimate if the user is enacting a certain pattern of movement or a specific static position, and the control system's performance hinges on the aggregate accuracy of all classifications. When the algorithm is a regressor, then a mathematical relationship is established between the inputs and the outputs, and the control system's performance is evaluated by a measure of the quantitative difference to the real values.

The complexity of such an algorithm depends on the researchers' goals and knowledge and is, fundamentally, a trade-off between computational cost and performance. As an example, an algorithm as simple as an ON/OFF threshold for EMG could be employed, but its performance would be poor at best. By contrast, an infinitely complex algorithm could make picture perfect predictions, but it would take eons to compute. For both classification and regression, the state-of-the-art approach is to train a machine learning algorithm (Gull et al., 2020).

Classification algorithms, however, even when extremely accurate, might not be the ideal control strategy, given that human motion is continuous and, therefore, the type of control that mimics humans best is continuous, proportional and simultaneous. As a matter of fact, most commercially available prostheses forego using classification algorithms altogether (Jiang et al., 2014). This is the reason why a much greater emphasis will be placed, throughout this work, on regression algorithms.

Classification Algorithms

When it comes to classification, the prominent technique is the SVM, or Support Vector Machines, machine learning algorithm (Gull et al., 2020; Roy et al., 2020). In the literature, a vast number of algorithms and machine learning-based control strategies can be found, like KNN (K-Nearest Neighbours), RF (Random Forests), ANN (Artificial Neural Network), NB (Naïve Bayes), etc. As (Roy et al., 2020) note, RF, ANN and SVM seem to, at present date, provide some of the best prediction accuracies.

Regression Algorithms

Regression algorithms, better known in the field as proportional myoelectric control algorithms (Xiao, 2019) or continuous EMG-based control (Bi et al., 2019), are not as widespread as classification algorithms. This is mainly due to the increased difficulty caused by the nonexistence of a clear, linear relationship between EMG and developed force in the muscles, as seen in Section 2.2.2.

This inherent nonlinearity of the EMG signal tends to favour nonlinear methods, which have recurrently been shown to obtain the best performances (Hahne et al., 2014). This is not to say linear methods are without any advantages. For one, linear algorithms are comparably inexpensive from a computational point of view and suffer from less overfitting issues. For a more detailed description of overfitting and the bias-variance trade-off, refer to Section 4.3.2.

In addition, Hahne et al. (2014) showed that transformations can be used to linearize the EMG data and subsequently improve the linear method's estimation performance. The authors also split the data into positive and negative displacements and attributed a different linear regressor to each set, which was proved a success.

There have already been several techniques and methods presented in a large number of papers (Gopura et al., 2016; Bi et al., 2019; Hahne et al., 2014). Weighted regressions, support vector machines, artificial neural networks and fuzzy control are some of the broad types of control archetypes found in the literature, usually present with smaller attempts at innovation. Other attempted algorithms include nonnegative matrix factorization (NMF) (Jiang et al., 2014), mixture of linear experts (ME) (Hahne et al., 2014), multilayer-perceptron (MLP) (Hahne et al., 2014) and kernel ridge regression (KRR) (Hahne et al., 2014), etc.

Xiao (2019), for instance, proposed a universal control strategy, as shown in Figure 2.18, for joint angle estimation based on SVM that, in tandem with a PID controller with feed-forward action, achieved proportional myoelectric control of a cable-conduit mechanism-driven upper limb exoskeleton. The control method was labelled universal, in view of the advantage of being independent of the model and, therefore, applicable to different joints and, even, multiple-joint motion. The feed-forward compensators were added to correct flat peaks in the output joint angle when the motion direction is changed.

The integral signal of sEMG was selected as the feature of sEMG in that work. Previously, RMS and the linear envelop of RMS had been used as features, still the integral signal of sEMG can be easily obtained with an integrated circuit, which allows for significantly faster feature extraction than when using a logical computation. In addition, time-delayed instances of these features were, too, fed to the machine learning algorithm. The authors hypothesized that time-delayed features would improve the predictive behaviour of the model, given the fact that these time-delayed feature signals of sEMG contain additional information that is crucial to the motion of the limbs and, conceivably, relates to the real electromechanical delay that exists between muscle activation and limb movements. The LSSVM model was optimized using a particle swarm optimization method and the value of RMSD was applied as the evaluation criteria.

The results in performance for the three methods were shown to be very similar and all three showed great improvements when using the time-delayed features. Overall, use of the time-delayed feature signals of sEMG and the feed-forward controllers was proved successful in improving estimation performance. The controller type, specifically, succeeded in showing that these additional control mechanisms can help with minimizing the hysteresis of cable-conduit exoskeletons, at a lesser computational cost than, say, an adaptive controller.

More papers have delved into the potential benefits of time-delayed features. Xiao et al. (2018a) used multiple feature signals and their time-delayed signals in an attempt to improve performance in motion estimation. Being a very high dimensional set of data, the authors opted for a random forests (RF) algorithm as the solution. Even though they were used here as regressors, for EMG-based intention estimation purposes, they are more often employed as classifiers. A more detailed description and explanation of random forests can be found in Section 4.3.4.

Xiao et al. (2018a) thus compared their RF algorithm with MTDF (Multiple Time-Delayed Features) with a BP (Back-Propagated) neural network, a RBF (Radial Basic Function) neural

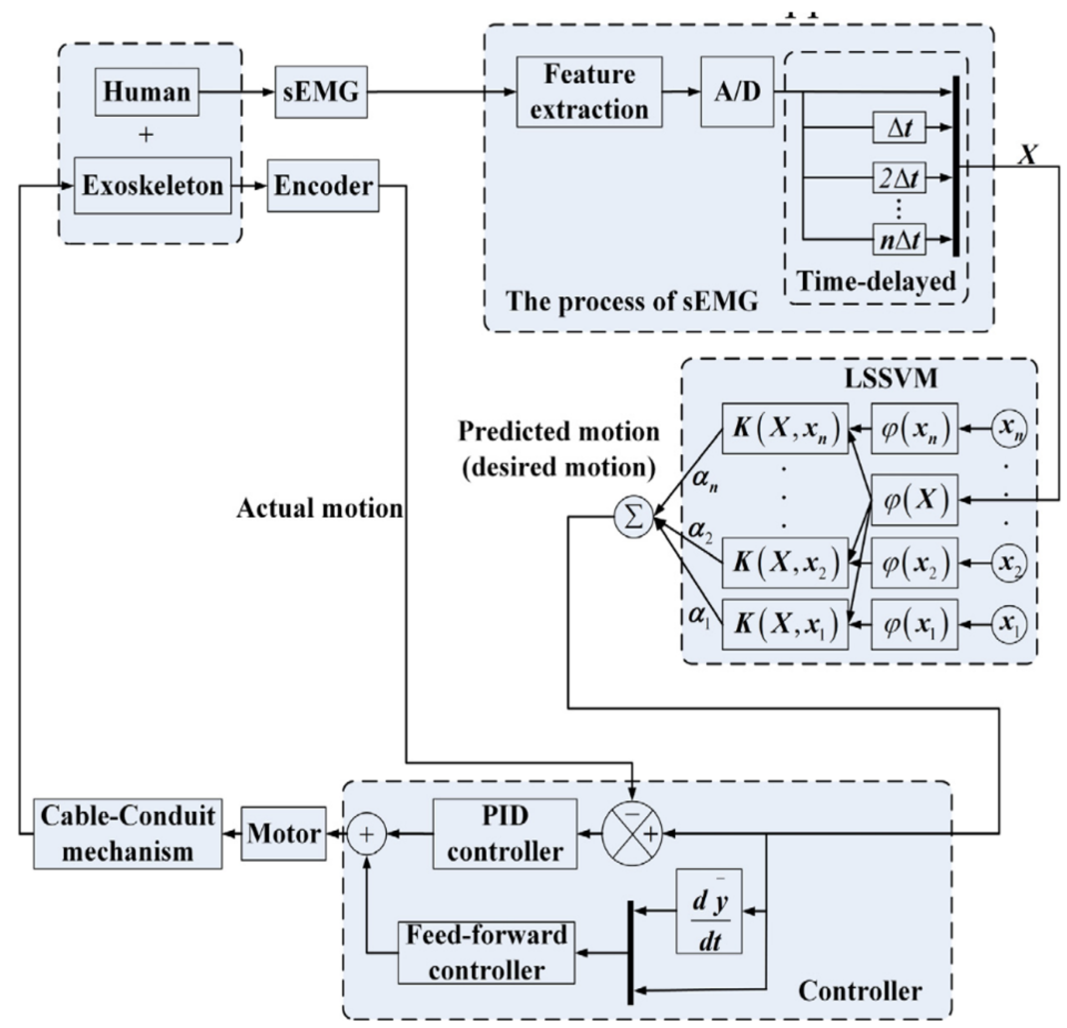

Figure 2.18: Diagram of the control system behind the time-delayed feature based LSSVM, with PID and feed-forward motor control. Source: Xiao (2019).

network and a LSSVM (least squares SVM, due to its simplicity and linearity of solving) with a Gaussian kernel function. The five hyperparameters of the RF were determined using a Genetic algorithm, and for the SVM parameters in common with the RF, the same values were used. All methods were applied with and without the time-delayed features, to investigate the impact of including them.

Five features were extracted from the EMG signal: MAV, WL, ZC, SSC (Number of Slope Changes) and DASDV (Difference Absolute Standard Deviation Value), all normalized using a MVC method. Performance of the estimation was evaluated with RMSD (Root Mean Square Difference) and RF with MTDF had the best predictive performance. Not only that, all algorithms performed better with the time-delayed features included.

Neural networks, albeit more complex, seem to be a promising technology. Tang et al. (2014), for example, used a back-propagation neural network for proportional myoelectric control of an upper limb power-assist exoskeleton. Selecting the RMS of the sEMG signal as the feature and evaluating prediction performance using RMSE, results obtained were satisfactory. Predictive performance was worst at the limits of the range of the movement, since acceleration was highest at those points, especially in the shortest interval motions. This goes in line with the theory - as the dynamics increase, the non-linearity of the sEMG single rises and prediction becomes much more difficult.

To combat that, a possible solution is to increase the amount of training data. When comparing the performance of various regression algorithms (LR, ME, KRR and MLP) when predicting hand movements of wrist flexion/extension and radial/ulnar deviation, the authors found that less training data considerably reduced performance across the board (Figure 2.19a). Surprisingly, when cross-validating the mixture of linear experts across training sets of different combinations of trajectories, the sensitivity of the algorithm to the number of training sets was found to be nearly independent of the trajectories used for training the algorithm, which showed no strong influence given enough samples (Figure 2.19b).

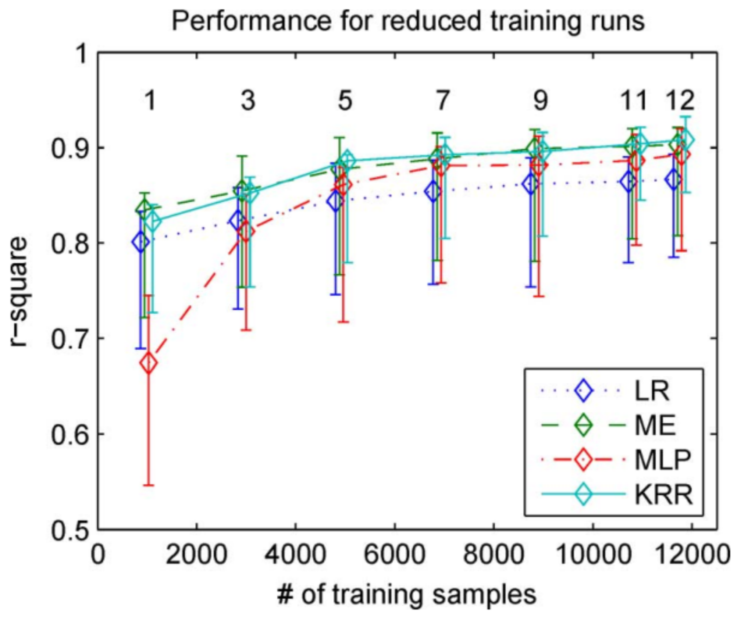

Figure 2.19: (a) Effect of number of samples for LR, ME, MLP and KRR algorithms on cross-validation performance (r-square). Curves indicate the median subject and whiskers show the quarter and three quarter percentiles. Above a minimum acceptable number of samples, the algorithms performed with consistence. (b) Effect of training trajectories for the ME algorithm on cross-validation performance (r-square). Shown is the median subject. Given enough samples, the algorithm showed some ability to generalise results to higher dimensional movements. Adapted: Hahne et al. (2014).

The authors concluded that the mixture of linear experts algorithm was able to generalize to regions of the dependent variable that weren't included in the training data set. This is an even more fascinating finding when taking into consideration the fact that such generalization happened with multiple degrees of freedom, that is, that small regions of coactivation were enough to generalize to the whole multi-DOF space.

In essence, intelligent, intuitive design seems to trump raw computational complexity. When studying the relationship between offline training and testing and online performance, Jiang et al. (2014) obtained results that indicate that the former does not necessarily provide a good predictive performance for the latter. This conclusion serves to highlight the subject's plasticity and capability for adaptation in real time. As long as the algorithm works with passable consistence, the users, who had no experience (naïve subjects) with these sorts of appliances, could always adapt and learn to work with the control system's shortcomings, provided enough feedback was returned to the user. Likewise, despite not being statistically significant, Smith et al. (2011) found a marked decrease in classification error from the first to the second sessions, which strongly suggests the users adapted to the system.

Another peculiar finding was that subjects could not tell, with much certainty, which algorithms performed best, nor why. According to the authors, there seems to be no dire need for "super machine intelligence". The active, though sometimes forgotten, role of "human embedded control" could be a pivotal contingency in designing successful interactions with imperfect machines.

These discoveries produce real insights into the nuclear traits that should, in practice, be pursued in proportional myoelectric control systems. Speed, efficiency, consistency and propensity to overshooting are a few of the core issues to optimize in these applications.

2.3.2 Control Modes

Powered exoskeletons, when used for assistance, can perform functions that, at heart, can be split in three groups, or control modes. These control modes help define exactly what kind of aid the robotic exoskeleton will provide. There are three main control modes for robotic assistive devices (Gull et al., 2020): assistive mode, corrective mode and resistance mode.

Assistive Mode

In an assistive control mode (Gull et al., 2020), the exoskeleton helps the user perform the movement by providing force to support the limbs, its own weight and perhaps even external loads. This can be passive or partially assistive control.

Passive control means that the user doesn't contribute to the movement - the impaired limb is taken through the trajectory by the robot. Passive control is usually done through passive trajectory tracking, nonetheless passive mirroring of the healthy limb is another technique that can be found. It is worth noting that mirroring doesn't mean symmetric EMG, so one limb cannot be used to train for the opposite limb (Roy et al., 2020).

Partially assistive control can take many forms, one of the most common being impedance/admittance control, a model-based force controller with position feedback. In partially assistive control, the exoskeleton cannot suppress any movement from the user. It is intended to aid the user in whatever motions they wish to perform.

This thesis regards an exoskeleton of partially assistive control mode, where an EMG-based control system estimates the user's intentions and provides motor aid to perform them.

Corrective Mode

In corrective mode (Gull et al., 2020), the exoskeleton only intervenes when the user makes errors when performing a set of predetermined movements that, typically, comprise a trajectory. Once a mistake is detected, the system then guides the user through a portion or the remainder of the trajectory, to help/force the user to learn the movement.

The most common technique in corrective mode control is tunneling (Gull et al., 2020). This technique devises a virtual trajectory that is enveloped in a channel, an error band that, when crossed, acts a force on the user in an attempt to bring the limb back to the trajectory. The force acts as though a spring impedance connects the limb to the center of the trajectory, increasing in magnitude if the user distances him/herself from the channel.

Resistance Mode

In resistance mode control, the robot resists the movements in certain directions to help correct or strengthen motion, and can even stop providing assistance or directly block movement. It does not, in spite of this, provide any further assistance or correct the movement - users have to complete it themselves. This can be also employed to reduce tremors (Gull et al., 2020).

2.4 Motion Tracking in Powered Exoskeletons

To evaluate the intention estimates, it is indispensable to employ means to record and measure the output, in this case, the resulting movement, as issued by the user's nervous system. A movement can, in generality, be described with one of three components: the trajectory, the velocity at all times and the acceleration at all times, provided the initial conditions are well-known. This is called motion tracking. In theory, any of these can be obtained from the others, by integration or differentiation of the measurements. There are other variables and factors that could be monitored in an EMG-based control system, but they are not essential to the process and will thus not be studied here.

The easiest way to track position and motion in an exoskeleton is to keep track of joint angles at all times. In similarity to what was aforesaid, this can be done with sensors for angle, angular velocity and angular acceleration. For this purpose, the most common solutions are potentiometers, goniometers, encoders, accelerometers, magnetometers, gyroscopes and IMUs, the likes of which will be addressed next. Finally, a state-of-the-art algorithm for joint angle estimation will be detailed.

2.4.0.1 Goniometers

In essence, a goniometer is a device that directly measures an angle or tracks an object's angular position in solidarity with it. They come in distinctive variants, such as universal goniometers, flexible goniometers and fluid goniometers (Figure 2.20). Interestingly, potentiometer-based goniometers have been called potentiometric goniometers in the literature (Tesio et al., 1995).

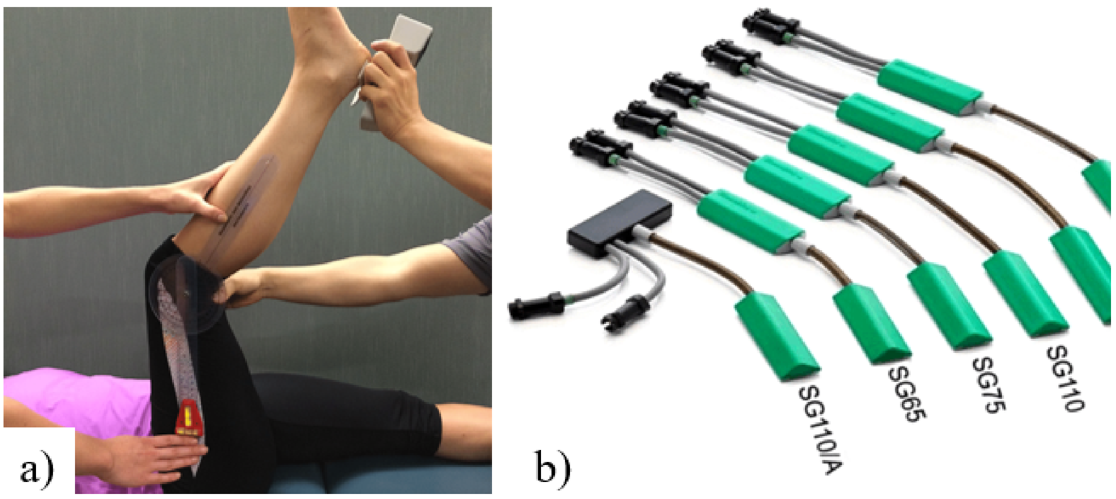

Figure 2.20: Some variants of goniometers, namely, (a) a universal goniometer being used for quantifying hamstring flexibility and different models of flexible or strain goniometers. Adapted: Williams and Welch (2015).

The universal goniometer is a manually operated device with two moving arms that measures the angle of one arm relative to the other, usually fixed. They are virtually useless in automated systems.

Flexible and fluid goniometers, also known as electrogoniometers, are extensively used in rehabilitative applications, especially the flexible goniometer. Originally proposed by Nicol (1987), this variation consists of a flexible measuring element, a strain-gauge steel strip, mounted between two plastic end blocks. One end is fixed and the other end is left free to glide, attached to a low-stiffness spring. The output is proportional to the angle of reciprocal orientation of the two bases in one or two planes (Tesio et al., 1995). Flexible goniometers are expensive, though reasonably priced - they are intuitive and precise. In spite of being physically incapable of measuring angles greater than 180º, this is not an inconvenience when applied to human joints, which have a limited range of motion.

Fluid goniometers use gravity's influence on a fluid to change the angular position of that fluid relative to a moving recipient (Rome and Cowieson, 1996). They consist of circular, 360º tubes half-filled with a coloured fluid that moves relative to a scale as the device is rotated. The position of the meniscus determines the position. They are only useful in mostly static or slow applications, where external accelerations beyond that of gravity have little to no effect in the fluid.

2.4.0.2 Inertial Measurement Units and Sensor Fusion

A sensible way to address the limitations of the foregoing sensors is to fuse multiple sensors that can counterbalance and complement each other. **Sensor fusion** is the process of using the information provided by more than one sensor, in an effort to aid the system in performing its intended task. When correctly applied, it provides the control system with a more accurate representation of the environment in which it operates. This can be done by taking advantage of certain design principles (Tzafestas, 2014):

- **Redundancy:** By using more than one sensor to measure the same quantity, the system becomes less vulnerable to problems that target a system in asymmetric ways. It is noteworthy that they are unable to provide any sort of protection against perturbations that affect the system evenly and completely.
- **Complementarity:** Sensors can complement each other and provide different information that helps create a more complete understanding of the world. This can be done with sensors that measure the same quantities in distinct pockets of space or sensors that measure disparate quantities.
- **Coordination:** When sensors complement each other not along spatial dimensions, rather across time, they are acting in coordination.

In motion tracking, one of the most common sensor fusion techniques is to combine an accelerometer and a gyroscope, the obligatory components of an Inertial Measurement Unit (IMU), also known as inertial sensors. The addition of a magnetometer results in what is known as an inertial/magnetic sensor, another type of IMU.

The pairing of an accelerometer and a magnetometer is also a sensible decision - while keeping the device still or barely moving, the accelerometer should detect almost exclusively the gravitational acceleration of the object and give a sense of its orientation relative to a vertical axis. With that knowledge, the magnetometer could then provide information on its orientation relative to the Earth's magnetic north and finish establishing the device's absolute orientation in space.

The latter works well in slow and stable movements. Be that as it may, it provides no solution to the problems that arise when dynamics become increasingly prevalent (Douglas, 2019). Angular velocities and even variable linear accelerations throw off the accelerometer's estimation of the direction of gravity. It becomes unusable. This is where the gyroscope finds its use. A sensor that measures angular velocity is, indeed, indispensable to the magnetometer and accelerometer couple. In addition, the magnetometer and accelerometer pairing give the gyroscope the initial absolute orientation that it needs. In essence, the inertial/magnetic sensor is an excellent example of fusing complementary sensors.

Finally, IMU's can help solve the problem of magnetic interference in magnetometer measurements (Douglas, 2019). Most activities of daily living (ADLs) take place in environments where magnetic disturbances transpire in great number and varying effects. The nature of these disturbances can be sorted according to the object causing them, and labeled as either hard or soft iron sources. Hard iron sources are agents that produce a magnetic field of their own, like magnets, coils and the Earth, generally resulting in offsets in measurements. Soft iron sources, on the other hand, are objects that interact with magnetic fields without being magnetically charged themselves. They tend to distort magnetic fields around them. These sources of parasitic magnetic activity can be, to a reasonable extent, be taken out of the equation through calibration, especially if they move in tandem with the sensor.

In powered exoskeletons, the process of calibration then becomes a necessity for ensuring proper and accurate measurements are performed by the IMU, otherwise the measured joint angles could be negatively affected (Laidig et al., 2017). It requires precise alignment and configuration of the measuring units relative to the body segments, such that the joint's degrees of freedom, their kinematic constraints and geometric irregularities in the body are accounted for and the sensors have a common reference Müller et al. (2017).

This process of alignment and calibration can be quite cumbersome. Laidig et al. (2017) proposed a method for sensor-to-segment orientation estimation of a pair of IMU's when used for measuring the angle of a two DoF joint, namely, the elbow. By taking into account the innate kinematic restrictions of the joint, online computation of the real joint angle is possible without any specific alignment procedures and calibration motions. The authors showed that their algorithm is reasonably efficient and can quickly converge to solutions that are in accordance with traditional optical motion tracking systems.

This technique is invaluable when attempting to develop a consumer friendly wearable powered device, since precise alignment and configuration can be cumbersome and, even, impractical. Accounting for the deviations caused by errors in sensor placement requires specific knowledge that users should not be expected to have. For this reason, this method was an obvious choice for the present work. In this section, a detailed explanation of the algorithm is given over the next few pages.

2.4.0.3 Attitude Estimation

In order to compute the relative orientation of the inertial units, sensor fusion is required *a priori* to estimate each sensor's orientation relative to a fixed frame (the World frame, from hereon denoted by the letter W).

A 9 DoF MARG sensor measures three 3-axis quantities: acceleration, angular velocity and magnetic activity. These 9 values have to be fed into a sensor fusion algorithm that extrapolates the orientation of the sensor's frame relative to the World frame. Several viable techniques exist for building what is commonly known as an AHRS (Attitude and Heading Reference System), since it provides a system with knowledge of its current roll, pitch and yaw, these being measures of a body's relative orientation to the World frame (as show in Figure 2.21).

Each algorithm has its own advantages, with some being direct improvements on older ones. The Complementary Filter and the Kalman Filter are two extremely popular techniques for building simple, yet very powerful motion tracking devices. Nowadays, the state of the art is composed mainly of improved versions of popular methods (e.g. the Extended Kalman Filter), recursive algorithms like the Recursive Optimal Linear Estimator of Quaternions (Zhou et al., 2018) and complex mathematical techniques like the QUEST algorithm, originally described by SHUSTER

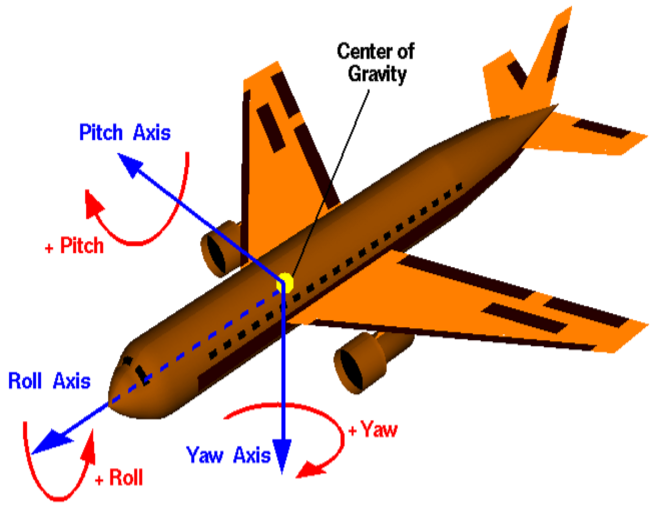

Figure 2.21: The three measures of an aerial body's orientation in space. Source: Glenn Research Center (2015).

and OH (1981). Most of them use purely inertial sensors, others include magnetometers and/or GPS, etc.

In 2010, Sebastian Madgwick proposed a novel orientation filter in Madgwick (2010), which later became known as the Madgwick algorithm. It uses a quaternion representation and Gradient Descent optimization (refer to Section 2.4.0.4 for a representative use case of the Gradient Descent algorithm) to compensate for gyroscope bias drift and magnetometer distortion, and compute the body's current orientation. The method is computationally efficient and extremely accurate. The mere fact that it compensates for magnetic distortion and is fairly fast to compute make it a stellar choice for a wearable device.

The output of the algorithm is the sensor's relative orientation in the World frame in the form of a quaternion. A quaternion is a 4-dimensional complex number of the general form:

$$a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k} \tag{2.8}$$

where a, b, c and d are real quantities and **i**, **j** and **k** are the complex fundamental quaternion units, in most use cases equivalent to the orthogonal unit-length vectors that represent the coordinate directions in a three-dimensional space.

Quaternions are commonplace in computer vision and motion tracking applications, given the low computational cost of operations involving quaternions and their invulnerability to gimbal lock (exemplified in Figure 2.22), a problem that plagues more traditional rotation coordinate systems such as the three-dimensional Euler angles.

The major downside of quaternions is their glaringly less intuitive understanding when compared to customary coordinate systems. On the grounds of clarity of the forthcoming joint angle estimation algorithm, quaternions can be converted to three-dimensional rotation matrices, which

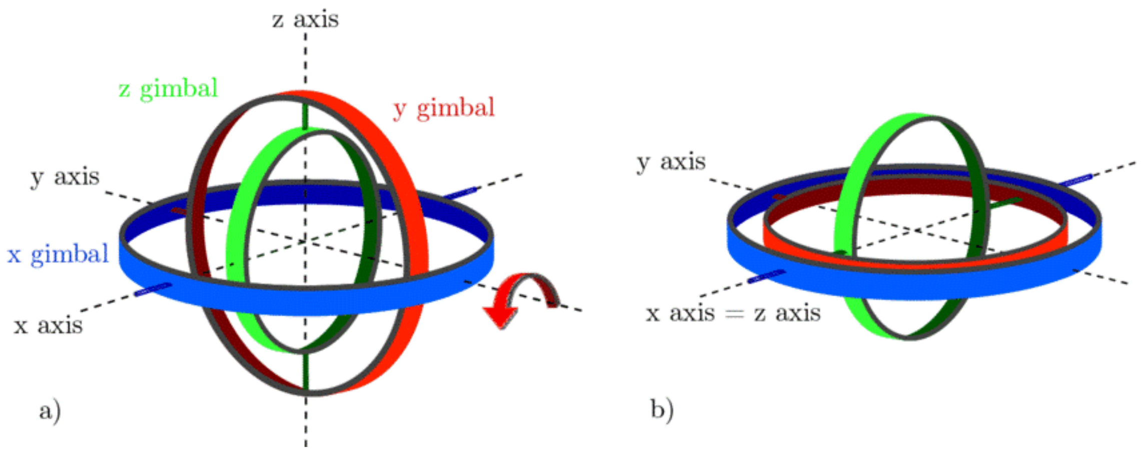

Figure 2.22: The gimbal lock phenomenon: when a body's rotations are (a) tracked with gimbals (either physical or abstract), (b) certain combinations of rotations can cause two gimbals to become superimposed, leading to the loss of an effective degree of freedom. In practice, this phenomenon alone can introduce singularities and other glitches in the interpolation of trajectories, as a body is taken through multiple positions and orientations. Source: Zeitlhöfler (2019).

are representations of a rotation belonging to $SO(3)$ (the 3D rotation group for all rotations about the origin in three-dimensional Euclidean space $\mathbb{R}^3$).

The sensor's orientation is, essentially, the rotation it suffered relative to the W frame, and as such the rotation of a body can be transformed from a quaternion representation to a rotation matrix representation (Kuipers, 2020):

$$\mathbf{R} = \begin{bmatrix} 2(a^2+b^2)-1 & 2(bc-ad) & 2(bd+ac) \\ 2(bc+ad) & 2(a^2+c^2)-1 & 2(cd-ab) \\ 2(bd-ac) & 2(cd+ab) & 2(a^2+d^2)-1 \end{bmatrix} \tag{2.9}$$

where a, b, c and d are the quaternion's real quantities and $\mathbf{R} \in SO(3)$.

The relative orientation $\mathbf{R}_A^B$ (rotation from the second to the first sensor's frame, as defined in Figure 2.23) can then be determined from each sensor's orientation relative to the World frame such that:

$$\mathbf{R}_A^B = \mathbf{R}_A^W \, \mathbf{R}_W^B = \mathbf{R}_A^W \, (\mathbf{R}_B^W)^{\mathrm{T}} \tag{2.10}$$

With $\mathbf{R}_1^W$ and $\mathbf{R}_2^W$ being obtained from the Madgwick filter based on each sensor's accelerometer, gyroscope and magnetometer measurements.

2.4.0.4 Alignment-Free Online Joint Angle Estimation

Müller et al. (2017) describe an algorithm for computing the joint angle without previously aligning the inertial sensors to their respective body segments or doing extensive calibration. The technique leverages the fact that the elbow joint is constrained to movement according to its two degrees of freedom, flexion/extension and pronation/supination (for a brief explanation of the anatomy of the human arm, refer to Section 2.1.2). Over the next few pages, the algorithm will be succintly explained. For a more complete understanding, refer to Laidig et al. (2017), Müller et al. (2017) and Lehmann et al. (2020).

To start with, it is necessary to determine ω_{AB}^{A}, the vector three-dimensional angular velocity of frame B relative to frame A, as seen on frame A. The upper index refers to the frame on which the angular velocity is mapped and the bottom two indexes indicate that the angular velocity pertains to the second sensor relative to the first. This angular velocity can be obtained by subtracting the angular velocities of both frames relative to the World frame W, all defined in the first frame:

$$\omega_{AB}^{A} = \omega_{WB}^{A} - \omega_{WA}^{A} = \mathbf{R}_{A}^{B}\,\omega_{WB}^{B} - \omega_{WA}^{A} \tag{2.11}$$

where $\mathbf{R}_{A}^{B}$ is the relative rotation matrix as previously defined and ω_{WA}^{A} and ω_{WB}^{B} are the angular velocities as output by each sensor's gyroscope.

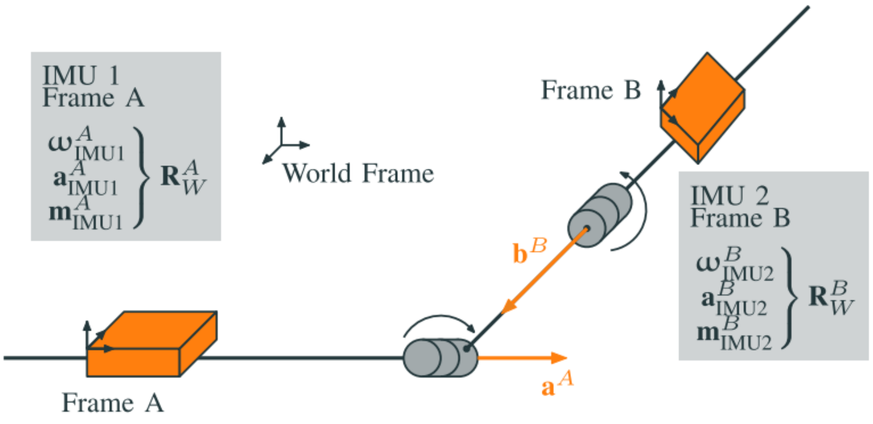
IMU 1
Frame A
$\boldsymbol{\omega}_{\mathrm{IMU1}}^{A}$
$\mathbf{a}_{\mathrm{IMU1}}^{A}$
$\mathbf{m}_{\mathrm{IMU1}}^{A}$
$\mathbf{R}_{W}^{A}$
World Frame
Frame B
IMU 2
Frame B
$\boldsymbol{\omega}_{\mathrm{IMU2}}^{B}$
$\mathbf{a}_{\mathrm{IMU2}}^{B}$
$\mathbf{m}_{\mathrm{IMU2}}^{B}$
$\mathbf{R}_{W}^{B}$
$\mathbf{b}^{B}$
$\mathbf{a}^{A}$
Frame A

Figure 2.23: Schematic of the elbow joint model: two segments connected by two revolute joints (the first, for flexion/extension, the second, for pronation/supination). Frames A and B are those of IMUs 1 and 2, respectively, which are attached to their corresponding segment in an arbitrary way. From accelerometer, gyroscope and magnetometer measurements, the IMUs estimate their frames' orientation relative to the World frame. By taking into account kinematic constraints, the directions of the two degrees of freedom can be obtained. Source: Müller et al. (2017).

The two angular velocities relative to the fixed frame can be described in totality by an axis-angle representation (represented in Figure 2.24), which consists of a unit-length vector representing the direction of the axis of rotation and a scalar quantity, the angle, in radians or degrees, that determines by how much the body is rotated around the axis. The relative angular velocity ω_{AB}^{A} can then be defined as:

$$\omega_{AB}^{A} = \alpha\,\hat{\mathbf{a}}^{A} + \beta\,\mathbf{R}_{A}^{B}\,\hat{\mathbf{b}}^{B} \tag{2.12}$$

where $\alpha\,\hat{\mathbf{a}}^{A}$ is the axis-angle representation of ω_{WA}^{A} and $\beta\,\hat{\mathbf{b}}^{B}$ is the axis-angle representation of ω_{WB}^{B}, with $\hat{\mathbf{a}}^{A}$ and $\hat{\mathbf{b}}^{B}$ being the directional unit-length vectors of the joint's degrees of freedom for

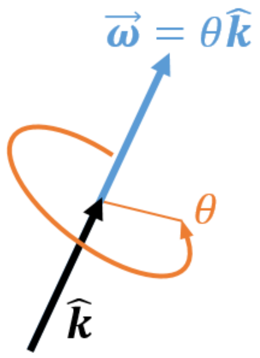

Figure 2.24: Axis-angle representation of a rotation $\vec{\omega}$. The unit-length vector $\hat{\mathbf{k}}$ provides direction of rotation and the scalar θ provides the quantity of rotation.

flexion/extension and pronation/supination, respectively, as defined in their respective frames.

These quantities can be determined from measurements, but may not represent the true rotation axes and angles, especially if alignment and calibration that would account for sensor-to-segment orientations were not performed in advance. For this reason, an error vector $\mathbf{e}$ is to be considered:

$$\omega_{AB}^{A} = \alpha\,\hat{\mathbf{a}}^{A} + \beta\,\mathbf{R}_{A}^{B}\,\hat{\mathbf{b}}^{B} + \mathbf{e} \tag{2.13}$$

The errors are not linear and instead vary in magnitude with the magnitude of the angular velocities at play. A normalized error vector $\mathbf{e}_n$ can be defined as:

$$\mathbf{e}_n = \begin{cases} \dfrac{\alpha\,\hat{\mathbf{a}}^{A} + \beta\,\mathbf{R}_{A}^{B}\,\hat{\mathbf{b}}^{B} - \omega_{AB}^{A}}{(\omega_{AB}^{A})^{T}\,\omega_{AB}^{A}} & \text{if } \left\|\omega_{AB}^{A}\right\|_2 > 0 \\ 0 & \text{if } \left\|\omega_{AB}^{A}\right\|_2 = 0 \end{cases} \tag{2.14}$$

where $(\omega_{AB}^{A})^{T}\,\omega_{AB}^{A}$ and $\left\|\omega_{AB}^{A}\right\|_2$ are two equivalent ways of computing the second order Euclidean norm of the relative angular velocity.

Finding the true axes and angles of rotation would then consist of optimizing Equation 2.13 such that the error term is as small as possible. This is equivalent to minimizing a loss function, namely, the quadratic cost function of the sum of the squares of the normalized errors:

$$J(\alpha,\beta,\hat{\mathbf{a}}^{A},\hat{\mathbf{b}}^{B}) = \frac{1}{N}\sum_{k=1}^{N}\mathbf{e}_{n,k}^{T}\,\mathbf{e}_{n,k} = \frac{1}{N}\sum_{k=1}^{N}\frac{\mathbf{e}_{k}^{T}\,\mathbf{e}_{k}}{(\omega_{AB}^{A})^{T}\,\omega_{AB}^{A}}, \quad \forall \left\|\omega_{AB}^{A}\right\|_2 > 0 \tag{2.15}$$

where measurements were taken for each interval k for a total of N times.

This problem requires the minimization of a cost function in an eight-dimensional space, that is, subject to the manipulation of eight different variables, namely, one angle and three coordinates for each axis of rotation. Nonetheless, given any pair of axes $\hat{\mathbf{a}}^{A}$ and $\hat{\mathbf{b}}^{B}$, provided $\hat{\mathbf{a}}^{A} \neq \pm\mathbf{R}_{A}^{B}\,\hat{\mathbf{b}}^{B}$ (that is, assuming the two degrees of freedom have non-parallel axes of rotation), finding α and β should be a linear and convex problem, which can be solved using the Moore-Penrose pseudoinverse:

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = (M^T M)^{-1} M^T w^A_{AB} \tag{2.16}$$

where $M = \begin{bmatrix} \hat{\mathbf{a}}^A \, \mathbf{R}^B_A \, \hat{\mathbf{b}}^B \end{bmatrix}$. The problem has thus been reduced to a six variable optimization.

In addition, knowing that the directional vectors are unit-length, they can be mapped to spherical coordinates, where unequivocally $r_a, r_b = 1$ and the dimensionality of the problem is further reduced to four:

$$\mathbf{a}^A = \begin{bmatrix} \sin\theta_a \cos\rho_a \\ \sin\theta_a \sin\rho_a \\ \cos\theta_a \end{bmatrix}, \quad \mathbf{b}^B = \begin{bmatrix} \sin\theta_b \cos\rho_b \\ \sin\theta_b \sin\rho_b \\ \cos\theta_b \end{bmatrix} \tag{2.17}$$

The optimization has henceforth been reduced to a minimization about four variables. The resulting vector of features is:

$$\phi = [\theta_a \;\; \rho_a \;\; \theta_b \;\; \rho_b]^T \tag{2.18}$$

The optimization problem can then be mathematically defined as:

$$\min_{\phi} J(\hat{\mathbf{a}}^A, \hat{\mathbf{b}}^B) = \frac{1}{N} \sum_{k=1}^{N} \frac{\mathbf{e}_k^T \, \mathbf{e}_k}{(\omega^A_{AB,k})^T \, \omega^A_{AB,k}}, \quad \forall \left\| \omega^A_{AB,k} \right\|_2 > 0 \tag{2.19}$$

A number of algorithms can be employed for finding the loss function's local minima. The Gradient Descent algorithm is a first-order iterative optimization algorithm for finding the local minimum of a differentiable function. Every iterative step, the variables are taken in the opposite direction of the loss function's gradient, which is known to be the direction of steepest descent in any surface. Figure 2.25 illustrates the algorithm for the case of a two-dimensional feature vector. In practice, the algorithm can be applied to feature vectors of higher dimensions.

In the present case, the Gradient Descent algorithm's feature update rule is:

$$\phi_{n+1} := \phi_n - \gamma \frac{\partial J(\hat{\mathbf{a}}^A, \hat{\mathbf{b}}^B)}{\partial \phi} \tag{2.20}$$

where γ is the step size. Müller et al. (2017) recommend a starting step size of 0.02 and progressively decreasing it as the algorithm converges.

The loss function is differentiable and its gradient can be simplified to (Müller et al., 2017):

$$\frac{\partial J(\hat{\mathbf{a}}^A, \hat{\mathbf{b}}^B)}{\partial \phi} = -\frac{2}{N} \sum_{k=1}^{N} \frac{\alpha_k \, \mathbf{e}_k^T \dfrac{\partial \hat{\mathbf{a}}^A}{\partial \phi} + \beta_k \, \mathbf{e}_k^T \, \mathbf{R}^B_{A,k} \dfrac{\partial \hat{\mathbf{b}}^B}{\partial \phi}}{(\omega^A_{AB,k})^T \, \omega^A_{AB,k}}, \quad \forall \left\| \omega^A_{AB,k} \right\|_2 > 0 \tag{2.21}$$

where $\mathbf{R}^B_{A,k}$, $\omega^A_{AB,k}$, $\mathbf{e}_k$, α_k and β_k can be respectively obtained from Equations 2.10, 2.11, 2.13 and 2.16 with the measurements at time k. The partial derivatives of $\hat{\mathbf{a}}^A$ and $\hat{\mathbf{b}}^B$ are given by:

$$\frac{\partial \mathbf{a}^A}{\partial \theta_a} = \begin{bmatrix} \cos\theta_a \cos\rho_a \\ \cos\theta_a \sin\rho_a \\ -\sin\theta_a \end{bmatrix}, \quad \frac{\partial \mathbf{a}^A}{\partial \rho_a} = \begin{bmatrix} -\sin\theta_a \sin\rho_a \\ \sin\theta_a \cos\rho_a \\ 0 \end{bmatrix}$$
$$\frac{\partial \mathbf{b}^B}{\partial \theta_b} = \begin{bmatrix} \cos\theta_b \cos\rho_b \\ \cos\theta_b \sin\rho_b \\ -\sin\theta_b \end{bmatrix}, \quad \frac{\partial \mathbf{b}^B}{\partial \rho_b} = \begin{bmatrix} -\sin\theta_b \sin\rho_b \\ \sin\theta_b \cos\rho_b \\ 0 \end{bmatrix} \tag{2.22}$$

and

$$\frac{\partial \mathbf{a}^A}{\partial \theta_b} = \frac{\partial \mathbf{a}^A}{\partial \rho_b} = \frac{\partial \mathbf{b}^B}{\partial \theta_a} = \frac{\partial \mathbf{b}^B}{\partial \rho_a} = 0 \tag{2.23}$$

The joint's DoF angles can subsequently be computed by comparing the relative orientation $\mathbf{R}^B_{A,k}$ at a time k compared to an initial relative orientation $\mathbf{R}^B_{A,0}$. The rotation matrices can then be decomposed into three separate axis-angle rotations, using the joint's axes of rotation $\hat{\mathbf{a}}^A$ and $\hat{\mathbf{b}}^B$.

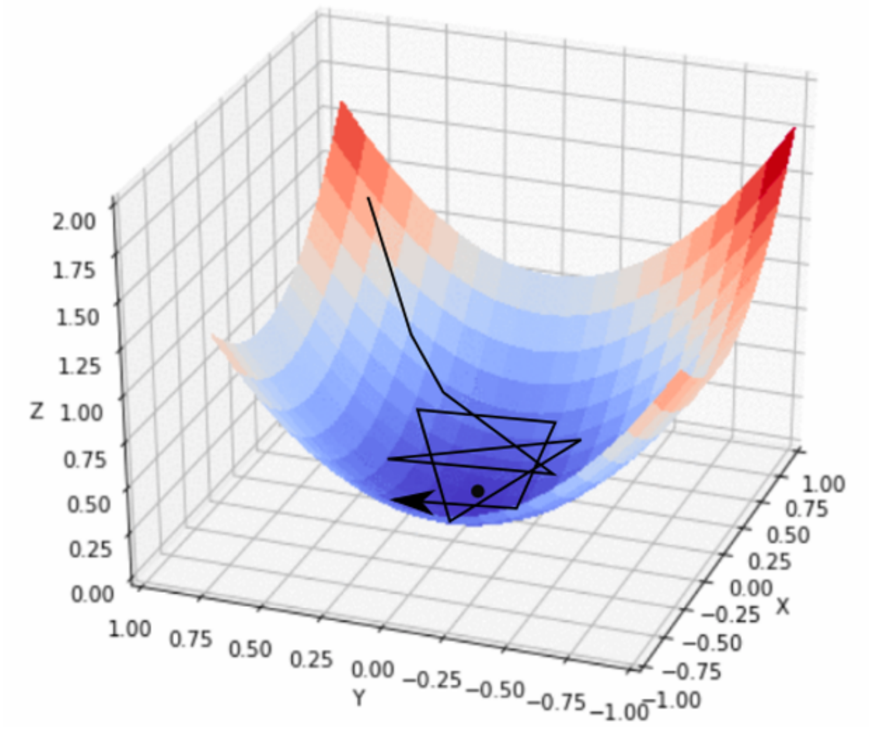

Figure 2.25: The Gradient Descent algorithm for a two-dimensional feature vector. X and Y are the feature axes and Z is the optimization function axis. In this case, the step size of the algorithm is too large and it fails to converge in a stable manner. Instead, it remains in proximity to the local optimum. Source: Kathuria (2018).

2.4.0.5 Computation of Joint Euler Angles

One technique for determining the angles is to compute the body's Euler angles for the rotation. When representing rotations via Euler angles, the axes of rotation move with the body as it rotates in space, changing the direction of the axes of rotation themselves. Decomposing a rotation into a set of Euler angles is called Euler decomposition, and a three-dimensional example is illustrated in Figure 2.26.

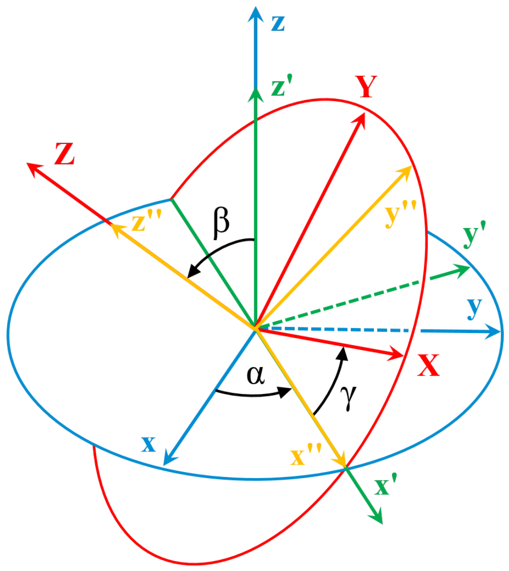

Figure 2.26: Decomposition of a rotation into three rotations about Euler angles. The rotation from xyz to XYZ can be decomposed into three successive rotations about z, x' and z'', with α, β and γ being the Euler angles.

Lamentably, the joint's axes of rotation are not orthogonal, due to the existence of a carrying angle between the arm and the forearm (Figure 2.27).

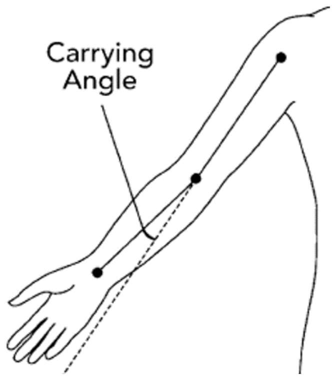

Figure 2.27: The human arm's carrying angle between the arm and forearm. Source: Cao et al. (2007).

Euler angles can be computed about non-orthogonal axes, as long as certain assumptions are met. A more complete proof and explanation of Euler decomposition about non-orthogonal axes can be found in Piovan and Bullo (2012).

Let $\hat{\mathbf{r}}_1$, $\hat{\mathbf{r}}_2$ and $\hat{\mathbf{r}}_3$ be three arbitrary unit-length vectors in $\mathbb{R}^3$. If $\hat{\mathbf{r}}_2$ is neither parallel to $\hat{\mathbf{r}}_1$ nor $\hat{\mathbf{r}}_3$, then the following condition:

$$\left|\hat{\mathbf{r}}_1^T(\mathbf{R}-\hat{\mathbf{r}}_2\hat{\mathbf{r}}_2^T)\hat{\mathbf{r}}_3\right| \leq \sqrt{1-(\hat{\mathbf{r}}_1^T\hat{\mathbf{r}}_2)^2}\sqrt{1-(\hat{\mathbf{r}}_3^T\hat{\mathbf{r}}_2)^2} \tag{2.24}$$

Is sufficient for a rotation matrix $\mathbf{R} \in \mathrm{SO}(3)$ to admit the Euler angles $\{\theta_1, \theta_2, \theta_3\} \in [-\pi, \pi[^3$ about $\{\hat{\mathbf{r}}_1, \hat{\mathbf{r}}_2, \hat{\mathbf{r}}_3\}$:

$$\exp(\theta_1[\hat{\mathbf{r}}_1])\exp(\theta_2[\hat{\mathbf{r}}_2])\exp(\theta_3[\hat{\mathbf{r}}_3]) = \mathbf{R} \tag{2.25}$$

where $\exp(\theta_i[\hat{\mathbf{r}}_\mathbf{i}])$ is the matrix exponential representation of the rotation matrix $\mathbf{R}_i$ that performs a rotation of θ_i around the unit-length vector $\hat{\mathbf{r}}_\mathbf{i}$, as defined in Rodrigues' rotation formula:

$$\vec{\mathbf{v}}_{rot} = \mathbf{R}(\theta[\hat{\mathbf{n}}])\vec{\mathbf{v}} \tag{2.26}$$

$$\mathbf{R}(\theta[\hat{\mathbf{n}}]) = \mathbf{I} + \sin\theta[\hat{\mathbf{n}}] + (1-\cos\theta)[\hat{\mathbf{n}}]^2 \tag{2.27}$$

where $\vec{\mathbf{v}}$ is an arbitrary vector in $\mathbb{R}^3$, $\vec{\mathbf{v}}_{rot}$ is that same vector rotated around a unit-length vector $\hat{\mathbf{n}}$ for an angle θ and $[\hat{\mathbf{n}}]$ is the cross-product matrix operator of unit-length vector $\hat{\mathbf{n}}$, such that:

$$\hat{\mathbf{n}} \times \vec{\mathbf{v}} = \begin{bmatrix} n_y v_z - n_z v_y \\ n_z v_x - n_x v_z \\ n_x v_y - n_y v_x \end{bmatrix} = \begin{bmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = [\hat{\mathbf{n}}]\vec{\mathbf{v}} \tag{2.28}$$

The Euler angles can be computed, starting with θ_2, which is one of the two solutions to:

$$(\theta_2)_{1,2} = \arctan_2(b, a) \pm \arctan_2(\sqrt{a^2+b^2-c^2}, c) \tag{2.29}$$

where $a = -\hat{\mathbf{r}}_1^T[\hat{\mathbf{r}}_2]^2\hat{\mathbf{r}}_3$, $b = \hat{\mathbf{r}}_1^T[\hat{\mathbf{r}}_2]\hat{\mathbf{r}}_3$ and $c = \hat{\mathbf{r}}_1^T(\mathbf{R}-\mathbf{I}-[\hat{\mathbf{r}}_2]^2)\hat{\mathbf{r}}_3$.

If $\mathbf{R}^T\hat{\mathbf{r}}_1 \neq \pm\hat{\mathbf{r}}_3$, θ_1 and θ_3 can, too, be uniquely determined:

$$\begin{aligned} \theta_1 &= \arctan_2(\hat{\mathbf{w}}_1^T\hat{\mathbf{r}}_1 \times \hat{\mathbf{v}}_1, \hat{\mathbf{v}}_1^T\hat{\mathbf{w}}_1 - (\hat{\mathbf{v}}_1^T\hat{\mathbf{r}}_1)(\hat{\mathbf{w}}_1^T\hat{\mathbf{r}}_1)) \\ \theta_3 &= -\arctan_2(\hat{\mathbf{w}}_3^T\hat{\mathbf{r}}_3 \times \hat{\mathbf{v}}_3, \hat{\mathbf{v}}_3^T\hat{\mathbf{w}}_3 - (\hat{\mathbf{v}}_3^T\hat{\mathbf{r}}_3)(\hat{\mathbf{w}}_3^T\hat{\mathbf{r}}_3)) \end{aligned} \tag{2.30}$$

where $\hat{\mathbf{v}}_1 = \exp(\theta_2[\hat{\mathbf{r}}_2])\hat{\mathbf{r}}_3$, $\hat{\mathbf{w}}_1 = \mathbf{R}\hat{\mathbf{r}}_3$, $\hat{\mathbf{v}}_3 = \exp(-\theta_2[\hat{\mathbf{r}}_2])\hat{\mathbf{r}}_1$ and $\hat{\mathbf{w}}_3 = \mathbf{R}\hat{\mathbf{r}}_1$.

The rotation sequence consists of the flexion/extension, followed by the fixed carrying angle rotation and, to finish, pronation/supination. In summary, the joint DoF angles at a given time k can be computed as Euler angles by considering $\hat{\mathbf{r}}_1 = \hat{\mathbf{a}}^A$, $\hat{\mathbf{r}}_2 = \frac{\hat{\mathbf{a}}^A \times \hat{\mathbf{b}}^B}{\left\|\hat{\mathbf{a}}^A \times \hat{\mathbf{b}}^B\right\|_2}$, $\hat{\mathbf{r}}_3 = \hat{\mathbf{b}}^B$ and $\mathbf{R} = \mathbf{R}_{A,k}^B$.

2.5 Actuation in Powered Exoskeletons

To put the foregoing intention estimation techniques into practice, systems capable of converting algorithmic commands into mechanical power are required. This is achieved through actuators, usually paired with complementary devices, such as power supplies.

A wearable robotic assistive device has to fulfill a wide range of requirements. First and foremost, the system's torque-to-weight and torque-to-volume ratios have to be as high as possible (Pina, 2018), so as to not overload the already strict weight budget or impede natural movement trajectories. This can become very problematic and difficult to overcome when the power supply is considered. For example, the most popular actuator (Figure 2.28), the DC motor, is often powered by batteries which, although mobile, come with a sizeable increase in weight and volume.

Noise and other comfort-related concerns can, too, impose constraints in the actuator selection. Pneumatic compressors, for example, are too noisy and too heavy for quotidian use (Pina, 2018).

Perhaps the single most critical factor when designing a wearable robotic device is safety (Pina, 2018). Be it imprecise movement, limited range of motion, added weight, uneven loading, leakage of harmful substances, overheating, risk of explosion, the potential for danger is prevalent, pervasive and unpredictable.

Hence, these characteristics will be delved into to a reasonable, though shallow, extent, over the next few pages.

2.5.1 Types of Actuators in Robotic Exoskeletons

There are three main forms of actuation in exoskeletons (Gopura et al., 2016):

- **Electric actuation:** moderately high in power-to-weight proportions, one of the main criticisms to this mechanism of actuation is the excessive impedance when considering rehabilitative applications. Additionally, their energy sources (batteries) tend to be large and heavy, thus cumbersome.

- **Hydraulic actuation:** Hydraulic actuators are extremely power dense and can produce fantastic torques. They tend to be, however, heavy, large and complex, requiring the solution of problems such as leakage and user safety.

- **Pneumatic actuation:** despite being less expensive than their hydraulic counterpart, pneumatic actuators require heavy systems for air treatment, storage and conditioning. They are less vulnerable in hazardous environments, they have less impedance and are lighter than electric motor systems, but, above all, they lack precision and accuracy.

Beyond these three main categories, other solutions that can be found in the literature are, essentially, either hybrid combinations of technologies or experimental and prototypical in nature. A few examples of these alternative actuation technologies are Electro-Active Polymers (EAPs), Shape Memory Alloys (SMAs) and Series Elastic Actuators (SEAs). In the upper-limb segment

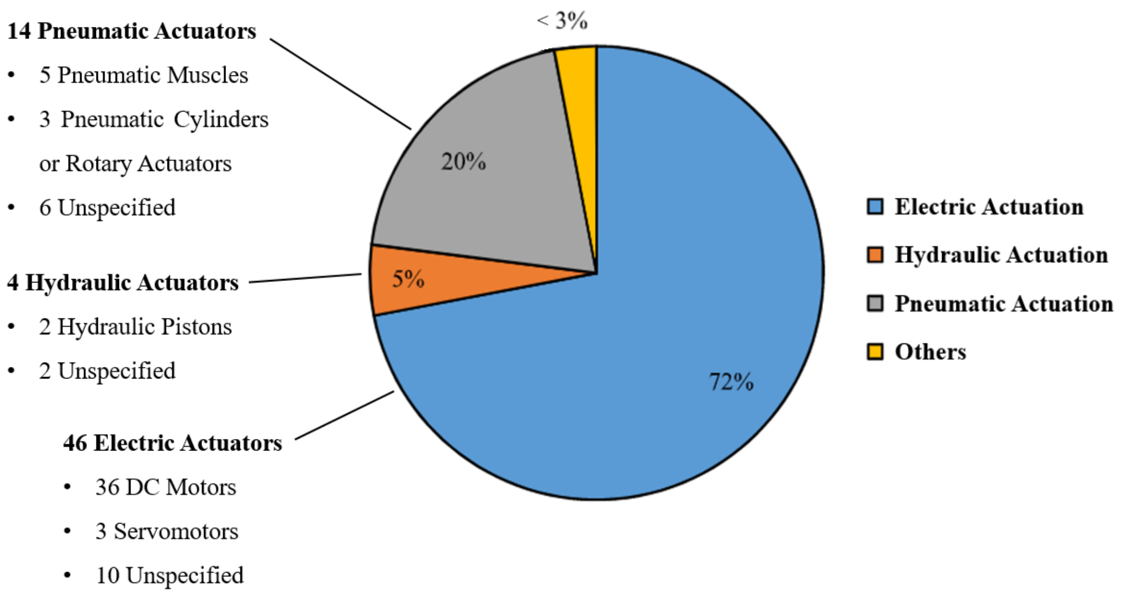

Figure 2.28: Estimated distribution of actuation technologies in upper limb exoskeletons, based on 67 different robotic assistive devices found in the literature by Gopura et al. (2016). Adapted: Gopura et al. (2016).

of the field of exoskeleton technology, EAPs and SMAs are not as common as other types of actuation.

Regardless, hybrid and alternative solutions together make up only a nugatory fraction of the robotic assistive devices found in the literature (Figure 2.28). The choice traditionally boils down to the electric, pneumatic and hydraulic actuation.

2.5.1.1 Electric Actuation

The vast majority of rehabilitative robotic devices use electric actuation as their primary form of actuation, as shown in Figure 2.28. This is understandable, considering the wide array of electric motors readily available, such as servomotors, stepper motors and the conventional DC motor.

Of these, the DC motor is, by far, the most prevalent in the literature. This can be largely attributed to its near-unbeatable power-to-weight ratio at a surprisingly low cost. Servomotors and stepper motors come respectively with an increment in price or weight, which only exacerbate the exoskeleton's main design obstacles. As mentioned before, increasing the exoskeleton's weight will result in a heavier loading of the user and more stringent mechanical design necessities.

Whereas DC motors can be selected for countless combinations of power, speed and torque, servomotors and stepper motors are often designed for applications where torques are higher and speeds are lower. DC motors, on the other hand, are, in general, built for high speed and low torque output, which can be easily converted, in lieu of the mechanical design requirements, into a lower speed and higher torque output through power transmission mechanisms, such as gear reduction, at the cost of added weight.

Servomotors customarily come with integrated sensors, which can be both a relief or a hassle, depending on the application. When it comes to the design of wearable exoskeletons, the budgets for weight and accessories are outstandingly tight, and so the choice frequently gravitates towards a custom build which typically involves a conventional DC motor. Servomotors also require more sophisticated controllers, which, given the time it already takes to run the control algorithms (refer to Section 2.3.1), is not practical.

The same happens with stepper motors. Their bulk, weight (they incorporate several magnets for turning in steps), and inflexible design (found, for instance, in the number of steps per turn) are seldom appropriate for an application where every kilogram is, at the outset, off-budget. As shown in Figure 2.28, of all 46 reviewed robotic assistive devices that used electric actuation, only 10 did not explicitly use DC motors. Of these 10, only 3 authors specified servomotors, 6 used unspecified "Electric motors", and not even one, perhaps to circumvent opprobrium, mentioned using a stepper motor. All in all, the choice of a DC motor in exoskeletons is essentially uncontested.

The working principle of a DC motor is on display in Figure 2.29. A current is run through a coil winded around a rotor (rotating part of a motor), creating a strong magnetic field. Magnets in the stator (static part of a motor) induce electromagnetic forces in the rotor, causing it to turn, thus producing mechanical power from an electric current.

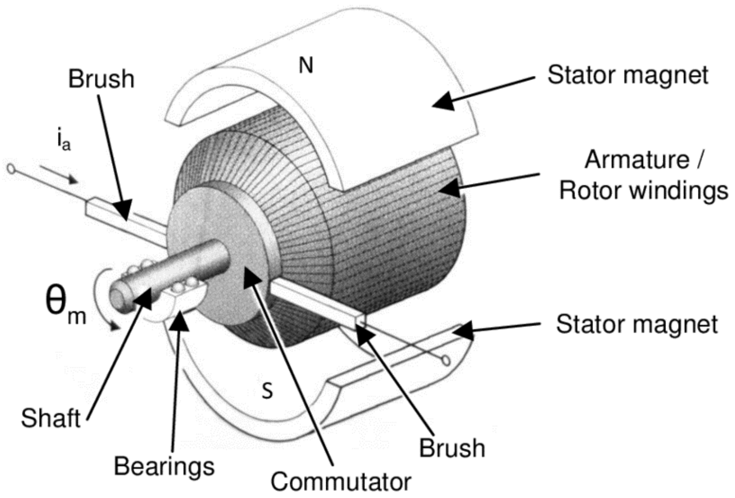

Figure 2.29: Principle of operation and most important components of a brushed DC motor. A current i_a is fed through the rotor's armature, crossing the stator magnet's electromagnetic field inducing an electromagnetic force, causing the armature to rotate. The shaft attached to the rotor can then be used to transmit mechanical power to an application. Source: Pedersen (2013).

The motor shown is a brushed DC motor, due to the presence of contacts named brushes, which act as electric commutators. Certain motors, suitably called brushless, lack these elements. Brushes are in constant relative motion to the rotor, under risk of severe wear from the resulting friction. By changing the configuration of the motor and switching the armature current electronically, friction is avoided and the longevity of the motor is increased dramatically. The efficiency of the motor is increased and the absence of the contact commutators makes them considerably more silent.

Barring their differences, both of these DC motor variants are reliable, inexpensive actuators, admitting of precise control and wide-ranging speeds. They offer infinite range of motion, which is wasted in a device that seeks to mimic human kinematics. They require mechanical or logical stops that only add to the device's complexity.

Their biggest flaw, nevertheless, is having a strikingly low torque-to-energy ratio, consequently requiring speed reduction in their transmission. The generality of speed reduction mechanisms come with a conspicuous weight increase, which reduces the overall power-to-weight ratio of the actuation system.

These limitations have caused researchers to look for solutions elsewhere. By far, the second most prolific actuation archetype in robotic assistive devices is pneumatic power.

2.5.1.2 Pneumatic Actuation

The number one pneumatic actuator found in robotic assistive devices is the McKibben actuator, also known as Pneumatic Artificial Muscle or Braided Muscle (Pina, 2018).

A McKibben actuator is a biomimetic actuator that contracts and expands similarly to a human muscle by inflating a rubber bladder, causing it to expand and shorten, and moving its ends closer together, as shown in Figure 2.30.

Pneumatic muscles have a great deal of exceptional advantages over DC motors, such as an appreciably high torque-to-weight ratio, being fairly lightweight and having an exceptionally convenient range of motion at a high torque (Pina, 2018).

Being gas powered (most of the times, air) makes them safe, and their actuation speeds and intrinsic histeresis (caused by mechanical friction), albeit a little too slow for certain applications, only curtail the already low risk they present to users (Pina, 2018).

Their biggest flaw is their nonlinear actuation and they are, in similar fashion to other pneumatic actuators, substandard in their position control. They, too, require compressed air supplies, which tend to be both bulky and heavy, and are markedly more expensive and complex than electric actuators Pina (2018).

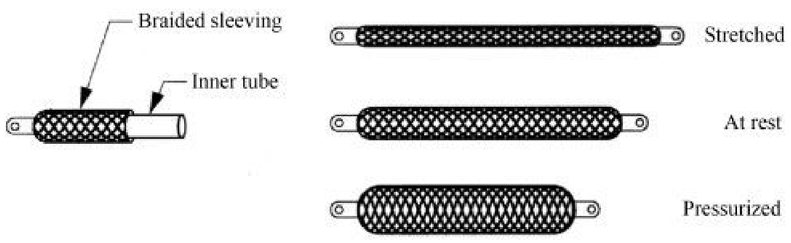

Figure 2.30: Overview of the basic components of a pneumatic muscle and its function. As the pressure inside the inner recipient increases, it expands sideways while contracting in length, mimicking a human muscle. Source: Johnson and Sensinger (2019).

Second to pneumatic muscles are traditional pneumatic cylinders and rotary actuators. The first, although simple, suffer from the same major problem that haunts the pneumatic muscle: unsuitability for precise control. Rotary actuators, while capable of slightly more meticulous movement, are heavier and bulkier.

2.5.1.3 Hydraulic Actuation

Hydraulic actuators are very rarely used by themselves. In spite of boasting sensational power-to-weight ratios, hydraulic cylinders pose larger safety risks to the user. Not only are their working pressures several orders of magnitude higher than those of pneumatic actuators, the working fluids can be hazardous (Pina, 2018).

The near incompressibility of liquids allow for proportional control and modestly sophisticated logic (Pina, 2018). Sadly, they are vastly more expensive than their pneumatic counterparts and, even further, when compared to the DC motor.

When used as the sole actuation technology, they come with large volume and weight burdens (Gopura et al., 2016), necessitating mechanically strong components (usually heavy) and fluid transmission and storage systems, such as pipes, conduits and recipients, as well as filters and lubrication equipment.

2.6 Conclusions

Briefly to conclude, rehabilitation engineering is field that has seen substantial growth over the last few years (Gopura et al., 2016; Gull et al., 2020). Advancements in several fields have contributed to the realization of powered exoskeletons, a technology that shows great promise when compared to the more traditional passive and body-powered assistive devices.

Leading the charge is the study of intention estimation, which seeks to act directly from the user's will. The most prominent technology in intention estimation applied to powered exoskeletons is EMG, that is, electromyography (a technique for the diagnostic of electrical activity in skeletal muscles). Researchers around the world have found breathtaking success in estimating subject intention from EMG signals. Their findings have now been widely employed to achieve proportional control of powered exoskeletons and numerous other devices.

Central to these control systems are machine learning algorithms. The new AI trend was adeptly adopted by the academia and to great results. When it comes to regression algorithms, neural networks seem to hold the most potential, just not exclusively. Random forest regressors, support vector machines, the list of performant algorithms grows every day.

In order to make their use possible, the EMG signal has to first be adequately conditioned. Pre-processing steps and feature extraction are a few of the cares that help produce a source rich in information for the learning algorithms to take advantage of.

The other crucial element of a powered exoskeleton is the actuation system. A myriad of actuation technologies have been incorporated in exoskeletons. Regardless, the DC motor remains the quintessential choice. It boasts high power-to-weight ratio, ease of use and is very accessible.

In this project, the objective is to design and implement a proportionally and simultaneously controlled one DoF robotic device using myoelectric signals from the flexor and extensor muscles of the arm. It serves as a proof of concept system to demonstrate, as accurately as possible, the challenges involved in the creation of a competitively priced powered exoskeleton, using run-of-the-mill components.

At the heart of the control system is a random forest regressor. As shown by studies, this ensemble algorithm has been used to achieve accurate intention estimation and fine control of exoskeletons. It possesses various properties that suit the EMG signal very well, such as not requiring normality of data or *a priori* transformations and being robust to noise. Random forests will be delved into in Chapter 4, Section 4.3, where an extensive introduction to the algorithm is presented. Perhaps it would have been more sensible to implement a more powerful algorithm, say, a neural network. However, correctly implementing a neural network would require far too much knowledge and experience than was possible given the author's lack of previous experience with machine learning algorithms and the project's time constraints. Regardless, to recompense for this limitation, the author ensured modularity was a core feature of the system, so that, in future work, the machine learning algorithm at the nucleus of the control system could be dexterously replaced.

Actuation is done by a DC motor. The preference for this actuation technology was motivated, quite straightforwardly, by economic factors. The DC motor is, indisputably, the most cost effective choice, while still providing excellent results.

In closing, supported by the literature review undertaken, the foregoing outlines were paramount in actualizing the present project, whose design, implementation and validation will be detailed over the next few chapters.

Chapter 3

Project Goals, System Architecture and Component Selection

In this chapter, much needed direction is established for this work. Assumptions are made and goals and requirements are defined. Then, a system architecture for the solution is discussed. Finally, components are selected and the wiring schematic is presented.

3.1 Design Assumptions, Goals and Project Requirements

As was previously mentioned, the primary objective of this work is to design and test a low-cost control system for an early prototype of a 1 DoF robotic assistive device for the elbow joint, based on simultaneous and proportional myoelectric control.

In accordance to the literature review (Chapter 2), the system will feature all customary components of a system based on EMG, namely, acquisition and processing of EMG signals, intention estimation based on a random forest algorithm and actuation, in this case, by a DC motor.

At the current stage, the device will not be implemented directly on the user, instead this thesis' work will serve as a proof of concept, a test bed for future *in situ* implementations. The main motive behind this decision is that a wearable, externally powered robotic device poses an uncomfortably high number of safety risks to a potential user, which, paired with the complex nature of the EMG technology, would require more time than was otherwise available to reduce the underlying uncertainties to a negligible level.

The implementation will be done in a laboratory bench or workstation, where vital testing equipment is readily available. The device itself will be 3D printed and serve as a rudimentary prototype.

There are many simplifications that not building the prototype for on-person testing enables.

For starters, the necessity for a portable power supply is essentially nullified. This will inevitably allow for a relaxation of the device's mechanical requirements, since most energy supplies come with a substantial increase in weight and volume. Moreover, the electrical complexity of the system will be lessened. In place of meticulous management of voltage levels, separate bench power supplies can be configured individually for devices that have disparate energetic demands.

Likewise, not having to assist a human arm in moving means there is no loading from the limbs themselves. This could be seen as negligible. In reality, the forearm and hand together can account for up to 2% of the total body mass (de Leva, 1996), which is not negligible. Not having to factor in this excess loading makes mechanical design of the robotic arm more manageable, if at all restrictive.

Finally, not implementing *in vivo* immediately eliminates considerations of ergonomics and comfort, which can almost entirely be left out for a future implementation.

To come to the point, the project has four main practical objectives:

- Design and implement a system for EMG data acquisition.
- Design a control system that processes the acquired EMG signals, extracts features and predicts motion intention. The motion intention, in this case, will be portrayed by the elbow joint's angle of flexion, defined according to the International Society of Biomechanics (Wu et al., 2005): positive for flexion and negative for extension, as illustrated in Figure 3.1.
- Use the estimated motion intention to simultaneously and proportionally control a 3D printed 1 DoF robotic device.
- Conduct experiments in such a way that the results can be evaluated and reproduced in future, more realistic and complete conditions.

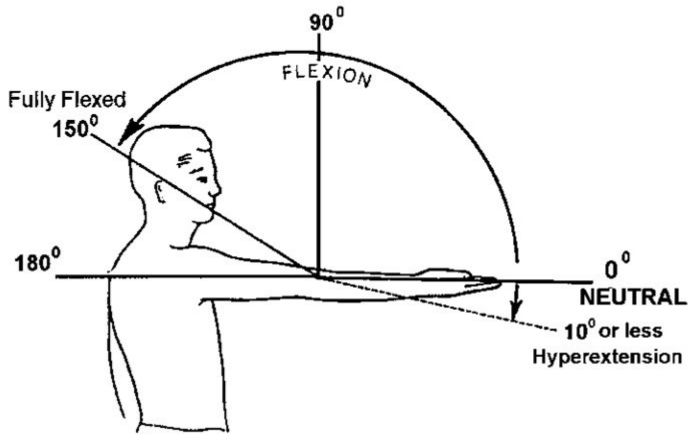

Figure 3.1: Convention for measurement of the human elbow joint's angle of flexion. Source: n.d. (2011).

For the purpose of achieving these objectives, certain nonspecific tasks have to be performed. They are:

- Establish overarching requirements for component selection and select them accordingly.
- Design a circuit schematic for wiring and assembly.

- Program the components according to intended functionality and underlying logic.
- Design and assemble a 3D printed 1 DoF mechanism for online testing and evaluation.
- Test portions of the system whenever necessary, to ensure operability as a whole.

Furthermore, design and implementation are subject to certain non-functional project requirements, to be specific:

- The system has to be as inexpensive as possible. The objective is to build it with ordinary components.
- The experiment has to be reproducible. Being reproducible means that the system can be successfully copied and tested for similar and consistent results.
- In order to be reproducible, the control system has to be implemented in a clear and modular way (for example, with regards to codification), so that the final solution is understandable and intuitive. The system has to be testable.
- Modularity will give the system some necessitous adaptability and flexibility of design. These qualities are crucial for iteratively improving the design in future works.
- Being reproducible is not enough. The design will have to display an objective level of similarity to an *in vivo* implementation, that is, a wearable exoskeleton, if conclusions are to ever be generalized.

With these requirements in mind, the following functionality is to be expected:

- The 3D printed device must feature a 1 DoF joint with a range of motion of at least 0 to 135 degrees, similar to that of a human elbow.
- The device must be capable of reproducing the elbow flexion/extension angle with a delay in the same order of magnitude as that of the length of a sampling window of EMG data.
- The data acquisition system must be easy and intuitive to fit on the user, which makes experimental procedures more time-efficient and user-friendly.
- The user interface must be intuitive and allow the user to save data from trials and experiments, for posterior treatment of it.

3.2 System Architecture

Predicated on the previous directions, a system architecture can be specified, so that hardware components can be selected and the system design furthered. This architecture can be seen in Figure 3.2. Over the next few paragraphs, a macroscopic inspection of this solution will be discussed.

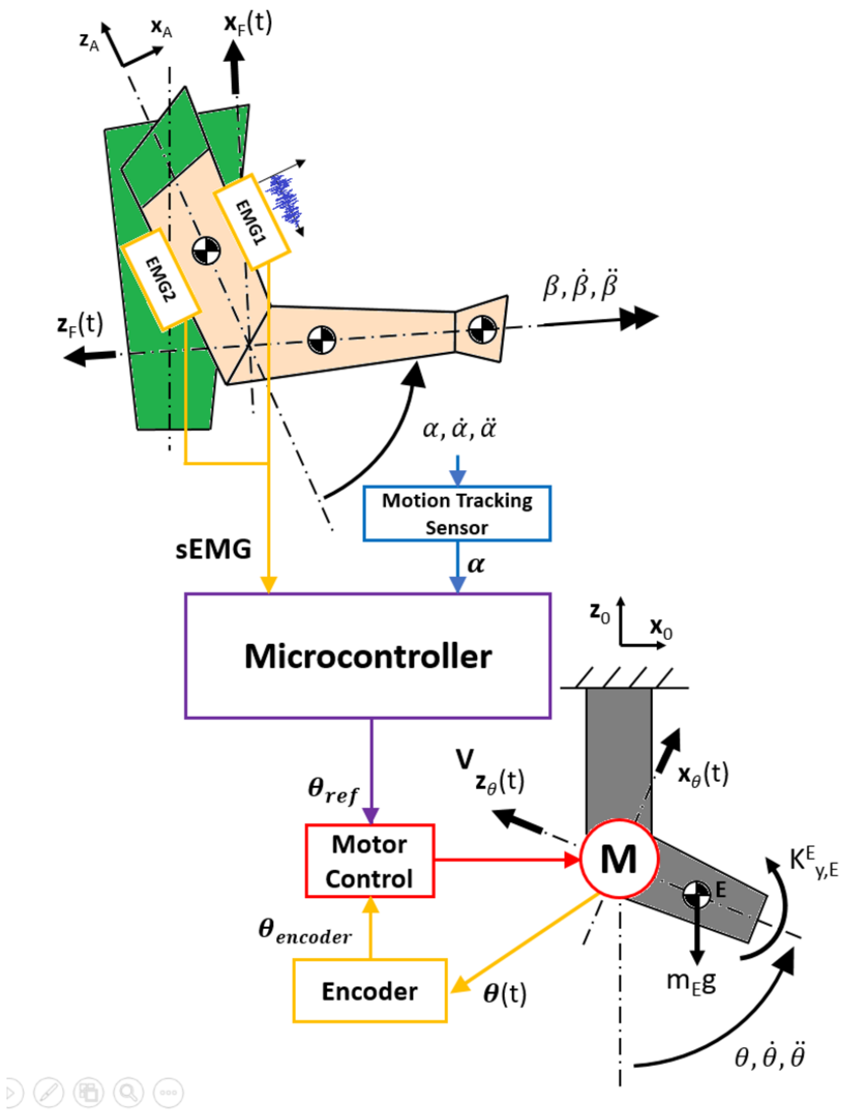

Figure 3.2: Illustration of the system architecture. The surface EMG sensors acquire the electric signals from the muscles during movement and send them to the microcontroller. Simultaneously, a motion tracking sensor produces information on the elbow joint's angle of flexion α. The microcontroller uses this information to build and train a model that can take an sEMG input and produce a positional command, an angular reference θ_{ref}, to the motor controller. The motor controller ensures, to its best ability, that the DC motor performs the intended movement. To do so, it receives data regarding the motor's angular position $\theta(t)$, say, from an encoder, which digitizes the quantity and produces a $\theta_{encoder}$ angle.

Firstly, in pursuance of motion intention estimation, two variables will always be necessary - the input variable and the response variable. In an EMG-based control system, the input variable is, unmistakably, the EMG data. Ergo, the response variable will be information about the elbow joint's current configuration, specifically, its angle of flexion/extension, which typically ranges from roughly 0 to about 145 degrees. Having this data available is imperative because it will then be used to train an algorithm that is capable of estimating motion intention based on electromyogram activity. It is the model's dependent variable.

The EMG signal can be swiftly acquired with a dedicated EMG sensor, in the case of Figure 3.2, a pair of sensors, given that both the biceps brachii and triceps brachii muscles have a fundamental role in elbow flexion/extension. For a brief explanation of the anatomy of the human arm, refer to Section 2.1.2.

Conversely, motion tracking can be achieved in a multitude of ways. A brief discussion of sensors available for motion tracking can be found in Section 2.4.

Once all necessary information is measured and recorded, a processing unit has to analyze the data and extrapolate a prediction of where the user intends to move. This result will thenceforth be fed to a controller for the actuation that ensures that movement is performed as intended.

The last component needed is an encoder or motion sensor for the actuator. This is the biggest complication due to not designing the prototype for *in situ* assistance. Evidently, the explanation is that by detaching the robotic hardware from the human arm, the movement is no longer physically coupled and another set of motion tracking sensors is required. This is represented in Figure 3.2 by the Encoder block, a ubiquitous solution to the problem.

The system could run without any feedback, except open-loop control systems are frequently unreliable and imprecise. For the problem at hands, closed-loop control is the indisputable choice.

Here, another discussion can be had on whether to decentralise computing or not. The control system has to perform four distinct tasks:

- EMG signal acquisition;
- Motion tracking;
- Intention estimation;
- DC motor control.

On one hand, separating them into individual processing units (presumably microcontrollers) creates a modular system, where information flow is kept neatly fragmented and whose flexibility and versatility make future enhancements much easier to implement. Furthermore, pathways for fault are no longer concentrated on a single component, which makes the system more robust and maintenance easier.

On the other hand, in a system where only one degree of freedom is being actuated upon, decentralisation is somewhat redundant. The truth is information flow is expected to be almost entirely linear and sequential, with tasks being undertaken in a cyclic way (refer to Section 2.2.2:

Rectification and Smoothing for a brief discussion of window-based analysis of the EMG data) and as frequently as possible, for the purpose of achieving seamless, responsive control.

However, the intention estimation algorithm alone is expected to demand the most computational resources, so it is reasonable to assume that it is the one that will run at the lowest frequency.

Halting all other tasks with each and every new step in the control loop creates a situation where every task has to wait for the previous one, as it produces an entirely new window increment T_inc of data (for context, refer to Section 2.2.2 - Rectification and Smoothing). Assuming T_inc and the processing time T_d are of similar magnitude, the control system's delay is essentially doubled, which is outright unacceptable.

Acquisition of new sensor data should instead be scheduled to happen asynchronously and at specific intervals, so that a new batch of information is ready just in time for a new cycle of intention estimation. Similarly, the motor controller should be running asynchronously and at a higher frequency, to ensure a stable, rapidly converging output. Most operations involved in these two tasks (data acquisition and motor control) are decidedly uncostly in computation, so splitting them across individual units of processing would increase the complexity of the system in exchange for a virtually inconsequential benefit.

For that matter, there is no real reason why a processing unit capable of online simultaneous and proportional intention estimation, which demands serious computational power, would be solemnly handicapped by a few data acquisition and control tasks. In the exclusive case of a single joint powered orthosis, a centralised control system might ultimately be the least expensive solution.

Thus it is clear that these tasks will have to be performed in parallel or, at the very least, concurrently (for a more extensive discussion of multiprocessing, refer to Section 5.3.2). With the aim of centralising the control system, the microcontroller will have to be capable of parallel and asynchronous computing.

From here, selecting the components should be a relatively straightforward task, which is detailed in the next section.

3.3 Component Selection

The realization of the solution presented in the previous section can be achieved with numerous combinations of both hardware and software options. Based on the system architecture, the components necessary to implementing it are:

- EMG sensors and/or acquisition systems.
- Motion tracking sensors.
- A powerful microcontroller suited for learning algorithms and capable of parallel processing.
- Actuator(s).

- Encoder or another sensor that can provide position feedback.

In an attempt to narrow down the component selection, and keeping in line with the project's requirements, the following general criteria were adopted:

- The components should be as inexpensive as possible and, when not possible, should contribute to a cost reduction of the whole system;
- The components should be compatible with each other and facilitate, to the maximum extent, implementation and improve modularity;
- The components should not be too heavy, too large or require too much power;
- Last but not least, the components should undoubtedly satisfy the aforementioned project requirements with sufficient reliability and quality of function.

3.3.1 Microcontroller

While selecting the microcontroller was a relatively expedite process, the variety of adequate affordable options should not be underestimated. Currently, there is a vast assortment of low cost boards specifically designed for IoT and prototyping needs (a fragment of those being visible in Figure 3.3) and, for the most part, they all satisfy the project's design requirements, offering superb computational prowess in a small credit-card-sized package. A rudimentary comparison of some of the boards discussed hereinafter is shown in Figure 3.4.

In spite of the available array of microcontrollers, the Raspberry Pi 4 Model B (Figure 3.3a) stood out as a chiefly competent choice, due, for the most part, to its cost, user friendliness and robust community. The latter, especially, when considering the forcefully fleet-footed nature of the project, took unprecedented priority in the selection.

The Raspberry Pi 4, starting at around 40€ (Raspberry Pi Foundation, 2020), delivers significant upgrades in performance from the Raspberry Pi 3 Model B+, at similar levels of energy and power consumption. Whereas the Pi 3 is barely useful in runtime machine learning applications, the new Raspberry is up to most machine learning requirements.

Its modern CPU, the Quad-Core Cortex-A72 (an ARM architecture CPU) is capable of achieving speeds of up to 1.5 [GHz]. The default OS is the 64-bit Raspberry Pi OS. Depending on the model, the Pi 4 can provide with up to 8 [GB] of DDR4 RAM memory. Perhaps most importantly, the Raspberry Pi 4 is compatible with most Python libraries and packages, which paired with its 40 pin GPIO allows for exceptional flexibility in IoT applications. The workflow when using a Raspberry Pi is characterized by speed and accessibility, thanks to the extensively large and beginner friendly community.

In sum, the Raspberry Pi 4 Model B is the latest in a line of affordable high performance and low power multi-purpose microcontrollers. This is not to say that other microcontrollers might not have been slightly more appropriate for a wearable automation project, though none come at the cost and size or with the amount of support that the Raspberry community provides.

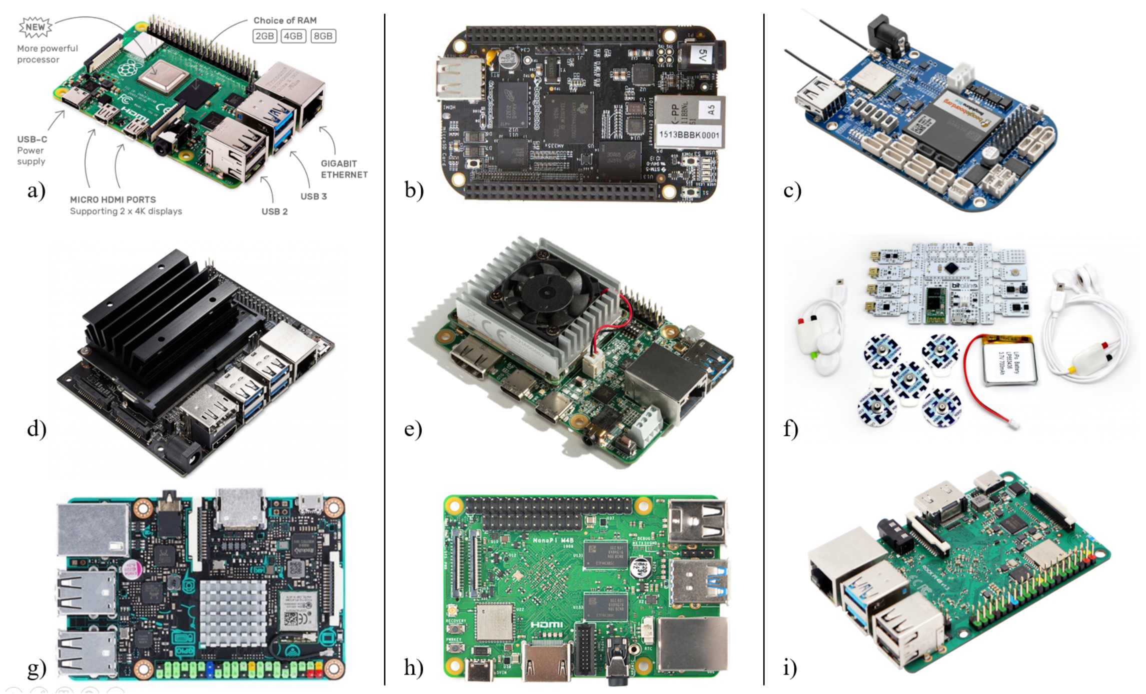

Figure 3.3: Affordable microcontrollers for IoT and prototyping applications. (a) Raspberry Pi 4 Model B (Source: Raspberry Pi Foundation (2020)). (b) BeagleBone Black (Source: BeagleBoard.org Foundation (2020)). (c) BeagleBone Blue (Source: BeagleBoard.Blue (2020)). (d) NVIDIA Jetson Nano Board (Source: NVIDIA Corporation (2020)). (e) Coral Dev Board (Source: Google, LLC (2020)). (f) BITalino (r)evolution Board Kit (Source: PLUX Wireless Biosignals S.A. (2020a)). (g) ASUS Tinker Board S (Source: ASUSTek Computer Inc. (2020)). (h) NanoPi M4B (Source: FriendlyElec (2020b)). (i) ROCK Pi 4 (Source: Radxa Limited. (2020)).

Taking exclusively technical prowess into account, a popular alternative could be one of the boards from the BeagleBoard.org Foundation, a non-profit corporation dedicated to producing open-source software and hardware. Their staple board is the Beablebone Black (BeagleBoard.org Foundation, 2020) (Figure 3.3b), with a AM335x 1GHz ARM® Cortex-A8 processor. It's a native Linux board that comes with a lot of flexibility, thanks to its two separate 46 pin expansion headers. This is an excess in GPIO capabilities for the project at hands, and the Beaglebone Black's firepower is not enough to justify its price of about 72€[1].

This is not the only Beagle available - a popular alternative to the Black is the Beaglebone Blue (BeagleBoard.Blue, 2020). Besides similar performance, the Beaglebone Blue (Figure 3.3c) provides extra peripherals specialized for robotics, such as H-bridges and discrete connectors for DC motors and encoders. It can be found at a heftier price tag of 75€[2], which makes the selection of the Pi that much easier. Furthermore, it should be noted that the Beagles are not, by any means, beginner friendly. They are best suited for programming in C, which is not an accessible language for those who are not software engineers.

[1]https://www.ptrobotics.com/beagleboard/2742-beaglebone-black-rev-c.html, 07/03/2021.

[2]https://pt.mouser.com/ProductDetail/BeagleBoard/BBBLUE?qs=MoCBKJu1Jj2UWBPau58zsg%3D%3D, 07/03/2021.

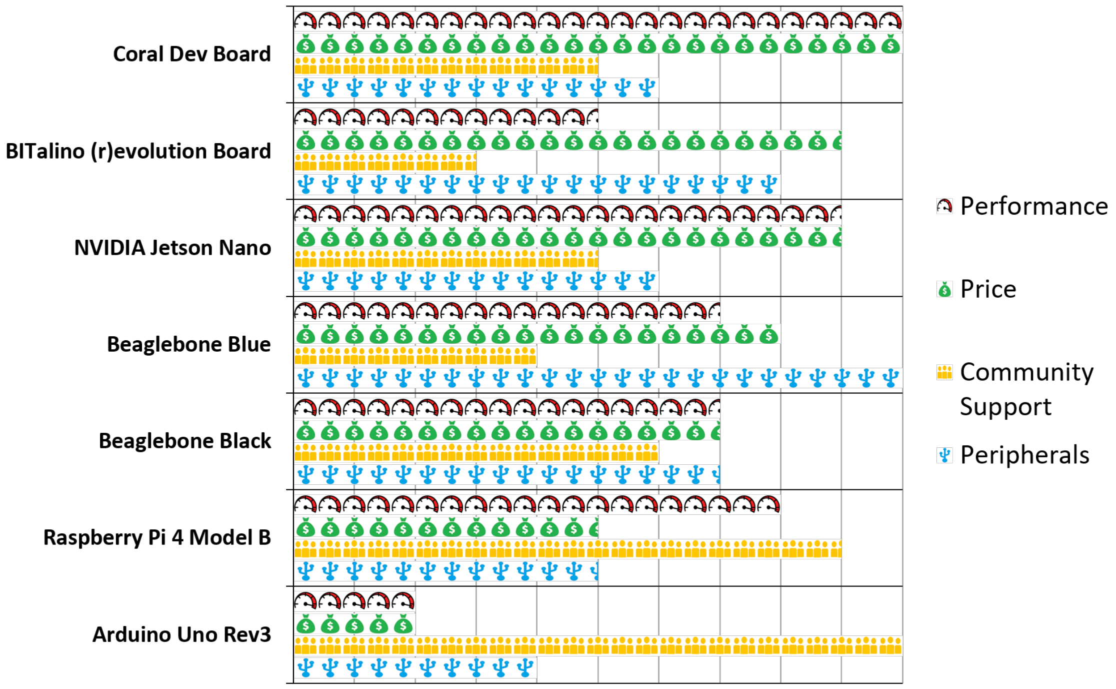

Figure 3.4: Pictorial comparison between several different microcontroller boards.

If ease of use is of such precedence, then an Arduino board, like the Arduino Uno Rev 3 (Arduino, 2020), could be of relevance. Unfortunately, the Arduino boards are not nearly as powerful as the others being presented here and their specifications are nowhere near enough for most machine learning applications. Aggravating this shortcoming is the fact that the Arduino cannot be trivially programmed for multiprocessing, though some protothreading and asynchronous programming can be achieved. Of course, the Arduino boards do come at a significantly lesser, though of no inconceivable difference, price tag, starting at approximately 20€.

Considering a machine learning algorithm was the intended approach for intention estimation, then boards that are more targeted for these techniques could be relevant. When it comes to learning algorithms, the NVIDIA® Jetson Nano™ Developer Kit (NVIDIA Corporation, 2020) and the Coral Dev Board (Google, LLC, 2020) are extremely powerful alternatives (Figure 3.3d and e). Despite having an earlier CPU in the Quad-Core Cortex-A53, the Coral Dev Board is decisively one of the best technical performers in this list, thanks to its machine learning Accelerator, the Google Edge TPU coprocessor. It features an integrated GPU and a 1 or 4 [GB], nonetheless is limited to the Tensorflow Lite set of machine learning packages, which, though diverse in itself, can be seen as a frustrating constraint. That and its hefty starting price of 109.99€[3] make it an unlikely substitute for the Pi 4. The NVIDIA® Jetson Nano™, on the other hand, is compatible with a larger breadth of ML packages and tools. Its video and graphics processing capabilities, some of the best in this segment of the market, are of little use in a myoelectric control system.

[3]https://coral.ai/products/dev-board/, 07/03/2021.

When taking into consideration its starting price of 105€[4], it loses vital appeal.

There are few solutions as well-suited for biosignal-based control as the BITalino (r)evolution Board kits (PLUX Wireless Biosignals S.A., 2020a) (Figure 3.3f). The brand offers different products, the cheapest coming in at 149.00€. It includes EMG, EEG, ECG (electrocardiogram) and EDA (electrodermal activity) modules that can be connected for biosignal acquisition. It already comes with pre-gelled electrodes, an accelerometer and a number of accessories that complement any bio-inspired system, comfortably justifying the kit's price tag. The main flaw of these kits is the inclusion of sensors that, for this particular work, were futile. Moreover, the BITalino (r)evolution Core is an Atmel ATMega328p, whose computational power is insufficient for the application.

In reality, the microcontrollers presented so far represent a minute portion of all IoT development boards that would suit the requirements. A few honourable mentions include the ASUS Tinker Board S (ASUSTek Computer Inc., 2020) (Figure 3.3g), the Odroid boards (Hardkernel Co, Ltd., 2020), the Banana Pi boards (Sinovoip, 2020), the NanoPi boards (FriendlyElec, 2020a) (Figure 3.3h), the ROCK Pi 4 (Radxa Limited., 2020) (Figure 3.3i), etc. All these pieces of impressive hardware fall short, relative to the Raspberry Pi, in the same, identical way - they lack the phenomenal and truly unparalleled community support that the Pi has, and for this simple reason, they were not objects of further consideration.

3.3.2 Surface EMG Sensors

The vast majority of scientific literature concerning EMG analysis reports on the use of EMG acquisition and sampling systems of tremendous refinement and appositely extravagant prices. A few examples of such upmarket devices for which quotes were available are shown in Table 3.1. Clearly, when attempting to design and actualize a low-cost system, befittingly affordable sensors have to be selected.

The most widely available, low cost sEMG sensor available in the market is the MyoWare Muscle Sensor (Advancer Technologies, LLC, 2020), the successor to Advancer Technologies' Muscle Sensor v3. The Muscle Sensor can be found at a price around 38€. A sensible alternative would be BITalino's MuscleBIT BT EMG data acquisition system (PLUX Wireless Biosignals S.A., 2020b), though at a considerably higher price of 275.00€. The two are shown in Figure 3.5.

Fuentes et al. (2019) studied the adequacy of a low-cost EMG system to conduct electromyographic analysis and intention estimation. The low-cost system was, pertinently, composed of one Myoware Muscle Sensor and an Arduino Mega. The authors found that there was excellent agreement between the low-cost and the custom, high performing system. This was not without flaws. The low-cost system produced a signal more strongly contaminated by noise and with a slightly larger delay.

The Myoware Muscle Sensor was thus selected for this work. It features:

[4]https://pt.mouser.com/Embedded-Solutions/Engineering-Tools/Embedded-Development-Tools/Embedded-Processor-Development-Kits/Development-Boards-Kits-ARM/$_/N - cvw9o?P = 1y8pymk, 07/03/2021$.

Table 3.1: Sample of devices for EMG analysis, found in the literature, for which quotes were publicly available.

Device	**Price (Estimate)**	**Description**	**Reference**
Cometa Wave Plus	$11,900.00	EMG system, with wireless sensors and up to 32-channel acquisition and amplification system.	Theurel et al. (2018)
D360 Amplifier, Digitimer Co	£10,995.00	Computer-controlled 8-channel patient-isolated AC-coupled biological amplifier and analogue filter system.	Colebatch et al. (2016)
CED Power1401	$6,675.00	Acquisition system with up to 16 input channels and multiple programmable gains.	Colebatch et al. (2016)
FlexComp Infiniti 10 Channel System	$6,500.00	Data acquisition and sampling system for Thought Technology devices.	Tang et al. (2014), Xiao et al. (2018a)
Myoscan sEMG Sensor for FlexComp/Myotrac	$295.00	Pre-amplified surface electromyography sensor compatible with other Thought Technology data acquisition and sampling systems.	Tang et al. (2014), Xiao et al. (2018a)
BlueSensor N-00-S	$22.20 (per unit)	Pre-gelled Ag/AgCl single-use electrode.	Theurel et al. (2018)

- Differential EMG acquisition with a third reference electrode to filter out common mode noise.
- Two output modes: The output is an analog signal, which can be raw or pre-processed EMG. The pre-processed signal is obtained after the raw signal is put through amplification, rectification and integration steps. The raw EMG is not entirely "raw" as the name suggests. Due to the incredibly low voltages at play in the electromyogram, the signal is amplified 201x before sending out.
- Adjustable gain (when using the pre-processed signal) and a CMRR of 110 [db].

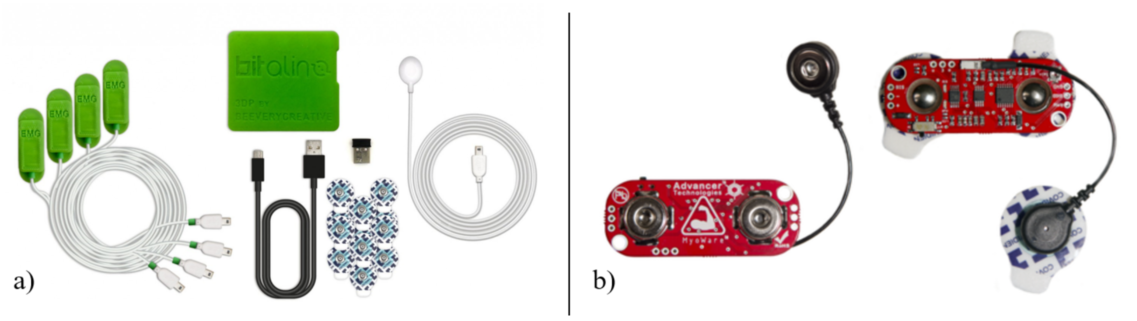

Figure 3.5: Two low cost sEMG sensors: (a) Advancer Technologies' MyoWare Muscle Sensor (Source: Advancer Technologies, LLC (2020)) and (b) BITalino's MuscleBIT BT (Source: PLUX Wireless Biosignals S.A. (2020b)).

The sensor requires a power supply from 3.3 to 5 [V] at a current of 9 [mA]. The sensor can additionally be expanded via shields (accessories that can be mounted directly on the board). A schematic of the sensor's layout can be seen in Figure 3.6.

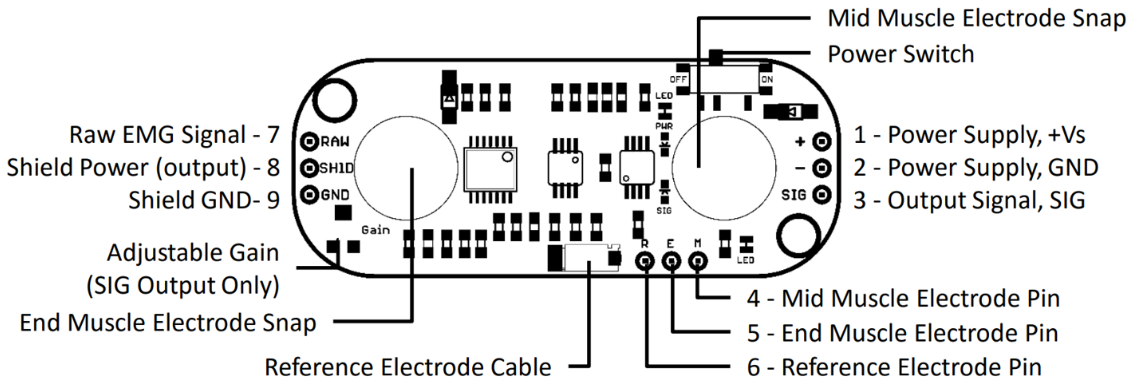
Mid Muscle Electrode Snap
Power Switch
Raw EMG Signal - 7
Shield Power (output) - 8
Shield GND- 9
1 - Power Supply, +Vs
2 - Power Supply, GND
3 - Output Signal, SIG
Adjustable Gain
(SIG Output Only)
End Muscle Electrode Snap
Reference Electrode Cable
4 - Mid Muscle Electrode Pin
5 - End Muscle Electrode Pin
6 - Reference Electrode Pin
RAW
SHID
GND
Gain
OFF
ON
LED
PWR
SIG
R
E
M

Figure 3.6: Schematic of the Myoware Muscle Sensor's layout. Adapted: Advancer Technologies, LLC (2020).

3.3.3 Analog to Digital Converter

The Myoware Muscle Sensor outputs an analog signal. Considering the Raspberry Pi solely supports digital inputs, an analog to digital converter is required. The choice was not particularly elaborated, seeing that there are plenty of low-cost alternatives in the market. Adafruit's ADS1115, illustrated in Figure 3.7, is a tried and true board that has been solidified as the go-to high-precision ADC board for DIY IoT applications. It has significant community support and an appreciable amount of documentation.

The ADS 1115 is priced at around $15 (Adafruit, 2020). It features 16-bit (enough for almost any sEMG application (Stegeman and Hermens, 2007)) analog to digital conversion in 4 single channels or 2 differential channels at an output rate of up to 860 samples per second over I^{2}C communication. This sampling ratio is a little under the literature norm, which is usually 1000 [Hz] (refer to Section 2.2.2). The board supports, in addition, inputs of different voltage ranges, thanks to its integrated PGA (Programmable Gain Amplifier).

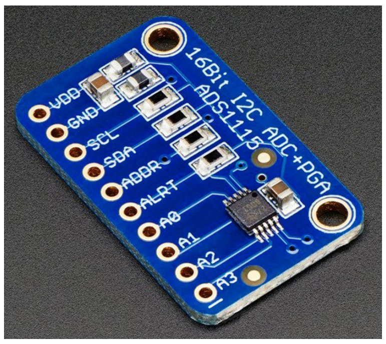

Figure 3.7: Adafruit's ADS1115 16-Bit ADC board. Source: Adafruit (2020).

3.3.4 Motion Tracking Sensors

As discussed in Section 2.4, the choice of sensors for motion tracking can be notably flexible. At a relatively low cost, most kinds of inertial sensors are excellent options for a wearable device that demands judiciously high performance.

Invensense's ICM-20948 is one of the world's lowest power low-cost 9-axis motion tracking devices (Invensense, 2020) that perfectly suits wearable technology and, in this case, exoskeletons. The 9 axis correspond to a 3-axis accelerometer with four selectable ranges, a 3-axis Hall-effect magnetic sensor with magnetic concentrator and an FSR (Full Scale Range) of ± 4900 [μT] and a 3-axis gyroscope, too, with four selectable ranges, all incorporated within the sensor's millimetric CMOS-MEMS package. It features on-chip 16-bit ADCs, a Digital Motion Processor™ that offloads the computation of motion sensing algorithms from the detectors, run-time calibration firmware and accelerometer and gyroscope ranges.

Coming at a measly price of about $17, an implementation of this sensor is the SparkFun 9DoF IMU Breakout board (SparkFun Electronics®, 2020), shown in Figure 3.8, which incorporates the Invensense's ICM-20948 into a Qwiic-enabled (Qwiic-enabled boards use a common 1mm pitch, 4-pin JST connector), that makes prototyping easier and less prone to human error, reducing the amount of required PCB space) breakout board complete with broken out GPIO pins. Matching the sensor, SparkFun's board supports both I2C and SPI communication at up to 100 [kHz] on standard mode and 400 [kHz] on fast mode. The ICM-20948 solely has two possible I^{2}C addresses, which, in the present case, should not be a problem, since only two inertial sensors will be utilized anyway.

SparkFun's board requires a voltage supply between 1.8 and 5.5 [V] and features a built-in regulator that allows the digital IO supply to be in the same range. The ICM-20948 has a maximum current consumption of 200 [μA].

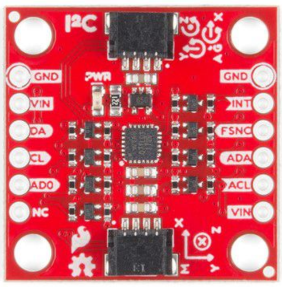

Figure 3.8: The SparkFun 9DoF IMU Breakout board (Source: SparkFun Electronics® (2020)).

3.3.5 DC Motor

For actuation purposes, a DC motor was chosen. This choice ties in with the review done in Section 2.5, where the DC motor stood out as an effective, efficient, lightweight, sufficiently powerful and, above all, inexpensive actuator for a wearable robotic device.

Nevertheless, unlike most other components, selecting a DC motor requires careful consideration of the pivotal parameters that govern its activity. Determining the ideal set of parameters of a DC motor for any given application first requires a solid understanding of the problem at hands. Both an oversized and an undersized DC motor can be quite costly mistakes.

In practical terms, providing robotic assistance to a human limb can be streamlined to a problem where an irregular load has to be lifted and this load is dependent on gravity and the moment arm's orientation in space. This analogy is detailed in Figure 3.9. The **E** (Exoskeleton), **F** (Forearm) and **H** (Hand) loads together constitute a body that has to be lifted about a point **O** that is travelling in space, but is fixed in the coordinate systems $(\mathbf{x}_A, \mathbf{z}_A)$ (of the arm) and $(\mathbf{x}_F, \mathbf{z}_F)$ (of the forearm).

The torque that has to be developed in order to rotate the exoskeleton, forearm and hand relative to the arm (this orientation given by the angle α) is denoted by $T^O_{y,M}$ and can be computed by applying Newton's Second Law of Motion for rotation about **O**:

$$T^O_{y,M} = -(r_{OE}\sin\theta)\cdot m_E\, g - r_{OF}(\sin\theta)\cdot m_F\, g - (r_{OH}\sin\theta)\cdot m_H\, g + K^O_{y,E} + K^O_{y,F} + K^O_{y,H} \qquad (3.1)$$

where r_{OP} is the distance from point **O** to **P**, m_X is the mass of body X, g is the gravitational constant and $K^P_{y,X}$ is body X's net torque about point **P**.

$K^O_{y,X}$ can be estimated from $K^X_{y,X} = J^X_X \cdot \ddot{\theta}$ by using the parallel axis theorem:

$$K^O_{y,X} = K^X_{y,X} + m_X (r_{OX})^2\,\ddot{\theta} = \left(J^X_X + m_X (r_{OX})^2\right)\ddot{\theta} \qquad (3.2)$$

Though this requires knowledge of each segment's mass and inertial properties, which vary a lot from person to person.

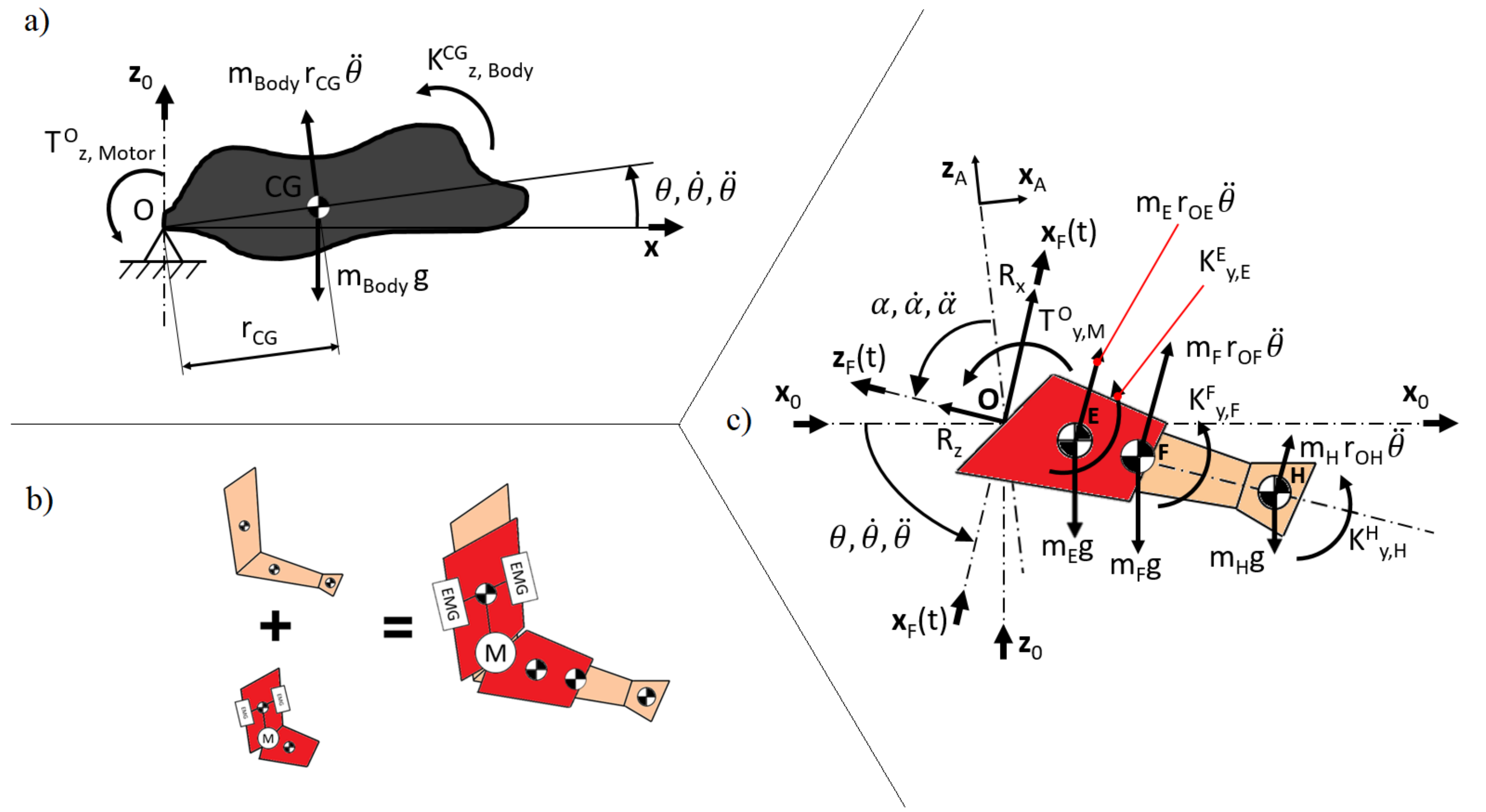

Figure 3.9: Free body diagram of a system composed of a powered exoskeleton and a limb. (a) Free body diagram of a body being lifted about one of its ends, a simplified analogy to the powered exoskeleton loading problem. m_{Body} is the body's mass, $T^O_{z,Motor}$ is a motor torque exerted about **O** on the body, $K^{CG}_{z,Body}$ is the body's net torque about its center of gravity **CG**. (b) Separation of the arm and wearable device into segments. Highlighted are potential placements of EMG sensors and actuator. (c) Free body diagram of the loading problem in a powered exoskeleton and limb, taking into consideration the segmentation defined in (b). $(\mathbf{x}_0, \mathbf{z}_0)$ is a fixed coordinate system (the World frame), $(\mathbf{x}_A, \mathbf{z}_A)$ is the upper arm's (not shown in the figure) coordinate system and $(\mathbf{x}_F, \mathbf{z}_F)$ is the forearm's coordinate system. Nomenclature is the same as the one adopted in (a).

A method for obtaining these properties in a slightly more expedite way than direct measurements is to use approximate values, like the ones provided by de Leva (1996) and shown in Table 3.2, which are dependent on other, more accessible measurements, such as the subject's total body weight and length of the pertaining segment.

Table 3.2: A segment's mass, center of mass and sagittal gyradius as a percentage of body weight and segment length. Sagittal pertains to the sagittal plane of human anatomy. Adapted: de Leva (1996).

Segment	**Mass** (% of Body Weight)		**Center of Mass** (% of Segment Length)		**Sagittal Gyradius** (% of Segment Length)	
	Female	Male	Female	Male	Female	Male
Upper Arm	2.55	2.71	57.54	57.72	27.8	28.5
Forearm	1.38	1.62	45.59	45.74	26.1	27.6
Hand	0.56	0.61	74.74	79.00	53.1	52.8

A segment's inertia can then be computed from its sagittal plane gyradius with:

$$J_X^X = m_X (r_{G,X})^2 \tag{3.3}$$

Finally, the resulting equation of motion for the system would be:

$$\begin{aligned}
T_{y,M}^O &= -(r_{OE}\, m_E + r_{OF}\, m_F + r_{OH}\, m_H)\, g \sin\theta + K_{y,E}^O + K_{y,F}^O + K_{y,H}^O \\
&= -(r_{OE}\, m_E + r_{OF}\, m_F + r_{OH}\, m_H)\, g \sin\theta + \\
&\quad + \left(J_E^E + m_E (r_{OE})^2 + J_F^F + m_F (r_{OF})^2 + J_H^H + m_H (r_{OH})^2\right) \ddot{\theta} \\
&= -(r_{OE}\, m_E + r_{OF}\, m_F + r_{OH}\, m_H)\, g \sin\theta + \\
&\quad + \left(J_E^E + m_E (r_{OE})^2 + m_F (r_{G,F})^2 + m_F (r_{OF})^2 + m_H (r_{G,H})^2 + m_H (r_{OH})^2\right) \ddot{\theta} \\
&= -(r_{OE}\, m_E + r_{OF}\, m_F + r_{OH}\, m_H)\, g \sin\theta + \\
&\quad + \left(J_E^E + m_E (r_{OE})^2 + m_F \left[(r_{G,F})^2 + (r_{OF})^2\right] + m_H \left[(r_{G,H})^2 + (r_{OH})^2\right]\right) \ddot{\theta}
\end{aligned} \tag{3.4}$$

which can be solved if the desired working speeds have been established. This can be done by first defining what the device's speed curve would look like starting from rest and accelerating up to the maximum speed. A common choice is to consider a trapezoidal speed curve, as in Figure 3.10. From here, the maximum acceleration can be stipulated as preferred, which will determine the curve's maximum slope. In a robotic assistive device, maximum speed and acceleration should be carefully selected - they should be close to the user's typical maximum speed, after considering safety factors. With these variables set, the maximum working torque follows from a direct application of the system's equation of motion (Equation 3.4).

Based on the maximum working torque and maximum working speed, a point can be charted on the motor's torque speed characteristic. A DC motor's torque speed characteristic is the governing equation that limits the possible combinations of torque and speed at which a motor can operate. An example of such a curve can be seen in Figure 3.11.

Subsequently, a DC motor would be selected such that the target working point would fit inside the motor's characteristic. The characteristic is often not explicitly shown in a DC motor manufacturer's catalogues, instead the stall torque and the no-load speed are given, also represented in Figure 3.11. It is worth noting that most DC motor are built for high speeds and low torques. This means that either the working point or the motor characteristic have to be modified to account for the use of gear reduction.

In practice, a 12 [V] leftover DC motor from another application was chosen. Based on an informal analysis of the market, it was priced at 5€. The selected DC motor, shown in Figure 3.12, had unknown specifications and parameters, the determination of which can be found in Appendix A.

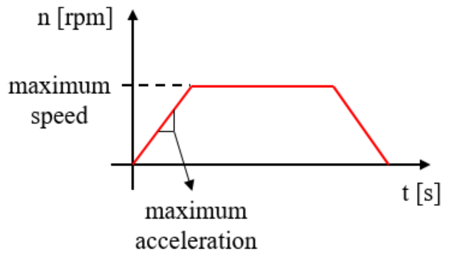

Figure 3.10: Ideal trapezoidal speed curve of a motor. The motor accelerates at a maximum constant value and stops accelerating once the maximum speed is hit. In reality, there can be overshoot and the acceleration is neither attained nor stopped instantly.

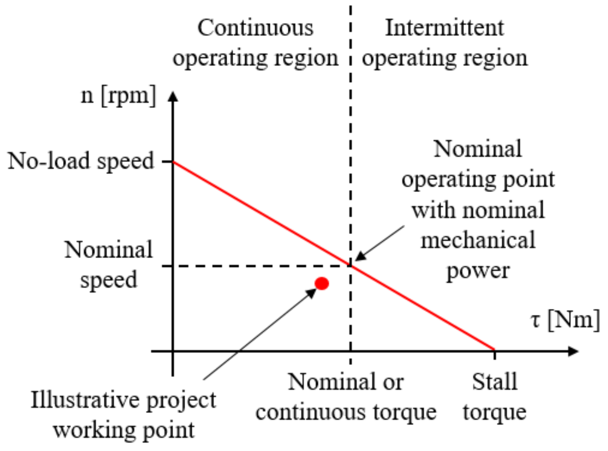

Figure 3.11: The torque speed characteristic of a DC motor. Operation is expected to be carried out in the continuous operating region. Consequently, the target working point should fit inside this region.

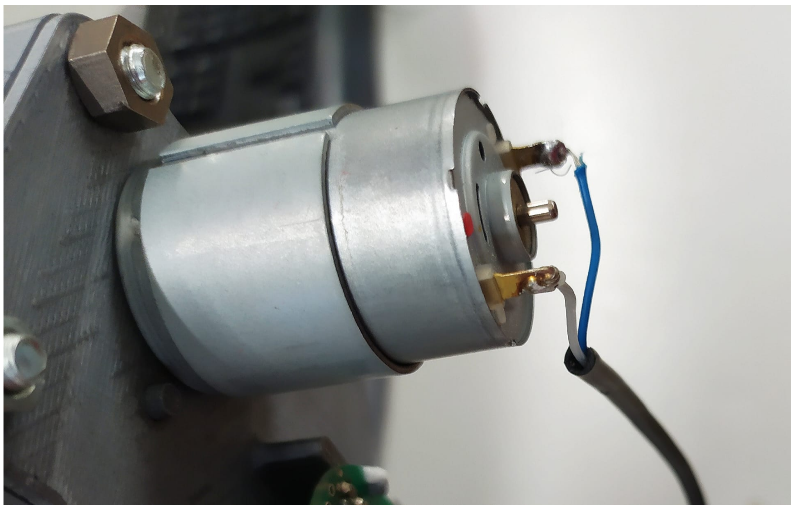
Figure 3.12: The selected DC motor, already attached to the gearbox.

3.3.6 H-Bridge

A microcontroller could be directly connected to a DC motor, although there are numerous reasons not to, including:

- The DC motor might operate at different voltages: if the motor in question requires a 12 [V] supply, no traditional microcontroller could ever provide that kind of voltage. The motor would end up with an insufficient voltage supply, which limits its maximum thresholds.
- A DC motor typically requires a large current: a microcontroller, on the other hand, can only provide very small currents at its outputs.
- Loading oscillations could damage the microcontroller: without getting into great detail, due to the formation of back EMF (refer to Section 2.5 and Appendix A for more information about a DC motor's principles of operation), immeasurably large current spikes could form in the circuit as mechanical loading of the motor oscillates and even forces its braking, potentially damaging the board.

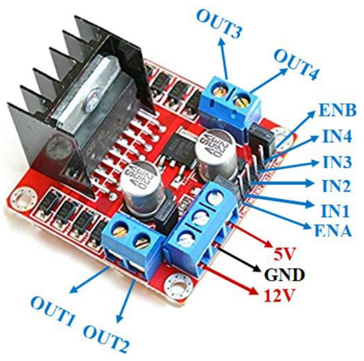

Figure 3.13: L298N Motor Driver Module Pinout. Source: Han.

For these and other reasons, a DC motor is always led by a Motor Driver, a circuit that fixes all the aforementioned issues and accommodates additional functionality. A popular choice is the L298N Dual H-Bridge Motor Driver, brought to view on Figure 3.13.

It is a moderately high power motor driver module (capable of supplying up to 25 [W] to the motor), consisting of a L298 motor driver IC and a 78M05 5 [V] voltage regulator (Han). It can control up to four motors unidirectionally or two motors bidirectionally. It supports voltage control and PWM driving (Pulse Width Modulation, Figure 3.14), and changing the voltage levels on its logic input pins can instantly demand the motor to revert direction or brake.

PWM is a technique for digitally supplying a set of electrical pulses at a constant voltage (Vikhe et al., 2014). This results in an effective supply voltage that is equal to the pulse voltage times the duty cycle (the percentage of time, for a given periodic segment of the PWM signal, that the voltage is set to high).

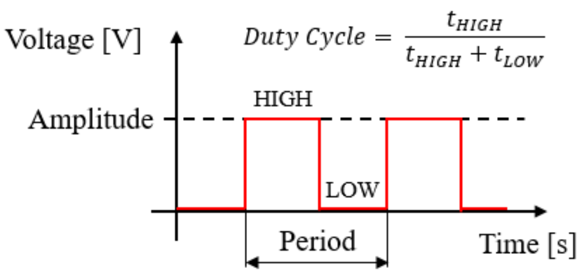

Figure 3.14: Representation of a PWM signal. In general, the variable that can be manipulated in order to change the output is the duty cycle. The HIGH voltage is kept constant, instead the signal is kept at high for a larger or smaller fraction of the period.

This is especially useful in applications where a low effective voltage is needed, but effectively providing a low voltage signal is not feasible. In DC motors, specifically, the use of low supply voltages would lead to reduced torque, even if the effective voltage of the PWM is the exact same (Vikhe et al., 2014). In rehabilitative engineering, low speeds are advised for the sake of the users. In a sense, with PWM, it becomes possible to supply a low effective voltage (and thus keep the motor working at low speeds) at a constant high voltage level (without compromising performance).

The main inconvenience with PWM is that the system being supplied with the signal must have sufficiently slow dynamics, to ensure the innate on-off dynamic of the PWM pulses (regulated by the PWM frequency) is not picked up on by the receiving device (Vikhe et al., 2014). In the case of a DC motor, sufficiently high frequencies can easily be achieved for the PWM signal, which explains why this technique is so widely used over basic voltage control.

The driver sports accessorial features. For instance, when the connected DC motors are being powered by less than 12 [V] from the bridge, the 5 [V] can be used as a 5 [V] supply, which can power other components in the system, namely sensors and even the microcontroller.

A spare part was used, though this component is not at all expensive (Han).

3.3.7 Magnetic Encoder

The encoder selected for tracking the DC motor's position was the RMK3B Evaluation Board, shown in Figure 3.15, a sensor based on the AM8192B, a compact, contactless angular magnetic sensor chip. Based on Hall sensor technology, the integrated circuit senses the angular position of a permanent magnet strategically placed above or under the chip, whose relative angular movement causes the magnetic field to move relatively to the sensor.

The board is capable of producing a number of different outputs, the primary a 13 bit (8192 counts per revolution, that is, a resolution of 0.77×10^{-3} radians per step or 44×10^{-3} degrees per step) absolute encoder, and others, such as incremental A QUAD B encoder signals, sine and cosine encoding and analogue output.

The sensor is extremely compact and reasonably precise, making it an excellent choice for lightweight wearable robotic devices. The component was already present in the laboratory, but from an informal analysis of the market it was estimated to be priced at around 35€.

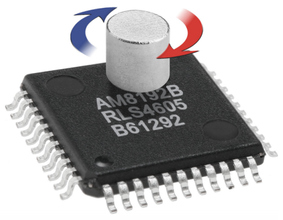

Figure 3.15: The RMK3B Evaluation Board with the AM8192B IC. The board comes with a small, cylindrical magnet, that can be attached to the object whose angular position is to be measured. Source: RLS d. o. o. (2021).

3.4 Wiring Schematic

With all components selected, the system's wiring and electrical circuitry were designed. In practice, this process was iterative and cohesively integrated with the design of the control system (refer to Chapter 4). An illustration of the final circuit is shown in Figure 3.16.

Numerous features of the final solution are worth being discussed. In addition, differences between the schematic and the original system architecture can be detected. These small adjustments were necessary to accommodate certain characteristics of the components selected, in an effort to meet project goals and requirements.

The most noticeable aspect is the use of the Muscle Sensor's RAW output pin instead of the more customary SIG output. This is because SIG's signal is not the raw sEMG signal, rather its filtered, integrated envelope. This could be useful, however it prevents custom pre-processing from being applied. Moreover, feature extraction is very limited since only a few features can be extracted from the integral of the sEMG signal. To learn which pre-processing stages were adopted, refer to Section 4.2. To learn about pre-processing of EMG signals in general, refer to Section 2.2.2.

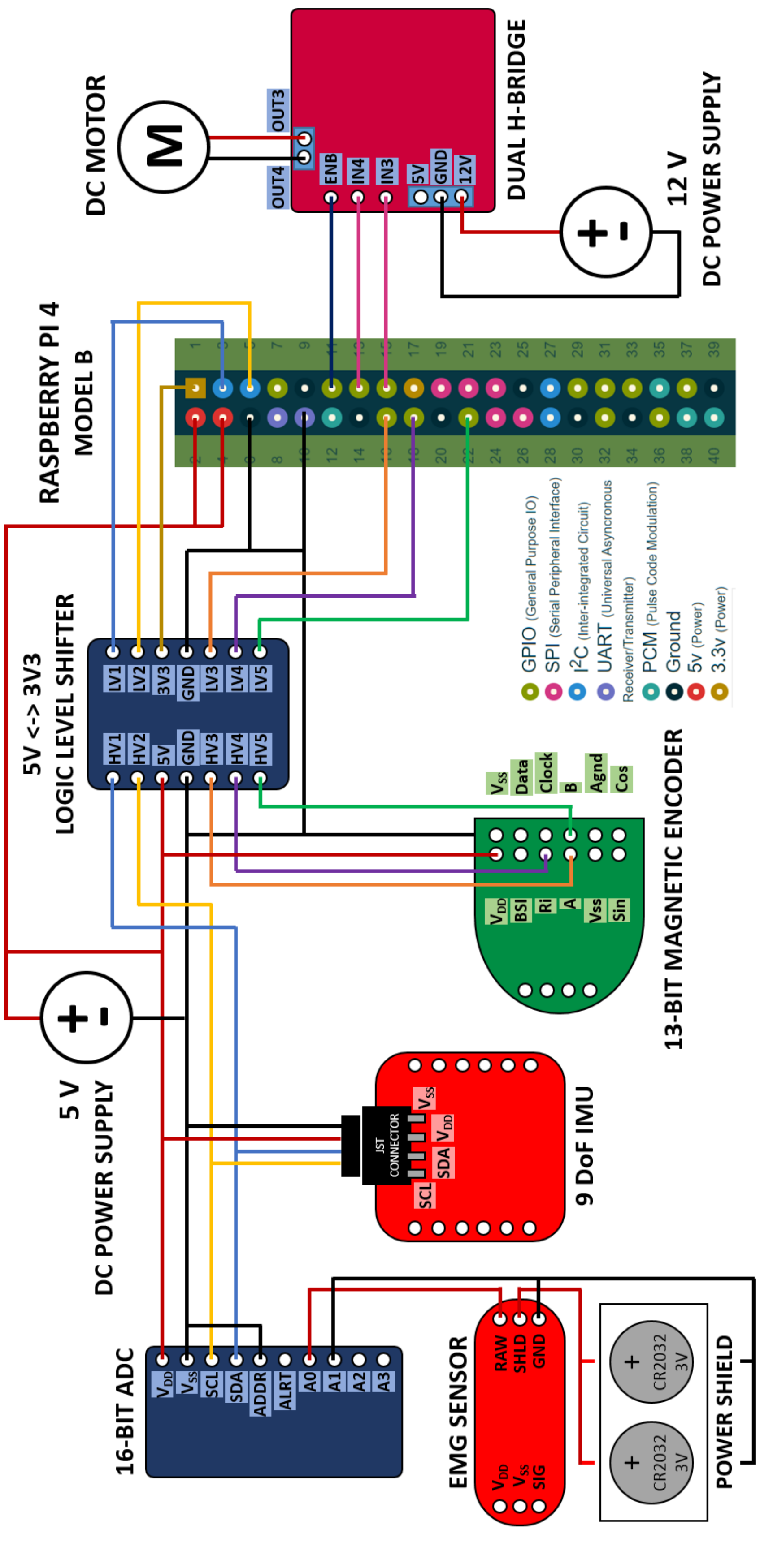

Figure 3.16: The system's wiring schematic. In reality, there are two 9 DoF IMUs and two 16-bit ADCs (each for a different EMG Sensor). Adapted: https://pinout.xyz/.

The biggest drawback of using the RAW signal is that it requires amplification, since it has a very small amplitude (about 300 to 500 [mV]) when compared to the SIG signal (± 3.3 [v]). Fortunately, the 16-bit ADC comes with a PGA that can amplify the input signal and compensate, to some extent, for the low voltage. Of course, this does not perfectly accommodate the signal's full-scale range, and as such some digitization errors are expected. Ideally, an amplification step would be integrated in the circuit, for example, with precision amplifiers that are robust to noise. Instead, the ADC was programmed for a PGA gain of 2.

In the wiring schematic, it can also be seen that the Muscle Sensor was connected to the ADC in differential mode, i.e., by connecting both the RAW and GND pins. The ADS1115's differential measurement setup can eliminate a lot of common noise in the signal, which is instrumental in achieving a high SNR.

A component found in Figure 3.16 that was not previously mentioned is a 5 to 3.3 [V] level shifter. It is necessary since the Raspberry Pi can only take in 3.3 [V] logic inputs, whereas the magnetic encoder, for instance, is run on 5 [V].

The DC motor's position will be tracked by the magnetic encoder, which will be outputting the A QUAD B encoder signals, as well as a reference signal Ri.

Chapter 4

Design of the Control System

With a solid understanding of the system architecture, specific design solutions can be addressed for each fragment of the system. As reviewed in Section 2.2, a typical myoelectric control system is composed of three fundamental steps: sEMG data acquisition and sampling, processing of the sEMG signal and feature extraction, and last, and not by any means least, intelligent intention estimation.

The crux of this thesis, the intention estimation algorithm, has a clear, albeit complex, purpose - to extract information from the muscles' myoelectric activity, transform that into knowledge and extrapolate estimates of the subject's intentions. The work of this thesis contemplates, moreover, the control of an electric actuator, that is, a simple DC motor, to be controlled based on the output of the intention estimation algorithm. In this chapter, the design of the entire control system will be comprehensively detailed.

4.1 Surface EMG Data Acquisition and Sampling

Evidently, no myoelectric control system is feasible without first sampling the surface electrical activity of the muscles.

When attempting to measure the EMG signals of the biceps and triceps muscles with a pair of sensors each, the SENIAM guidelines suggests that, for the former, electrodes need to be placed on the line between the medial acromion and the fossa cubit at one third from the fossa cubit (SENIAM, 2006a), and, for the later, electrodes need to be placed at 50% on the line between the posterior crista of the acromion and the olecranon at 2 finger widths medial to the line (SENIAM, 2006b). Both reference electrodes should be on the wrist. These placements can be visualized in Figure 4.1.

Finally, as the studies show (refer to Section 2.2.2), the data is to be acquired at a rate of at least 400 [Hz].

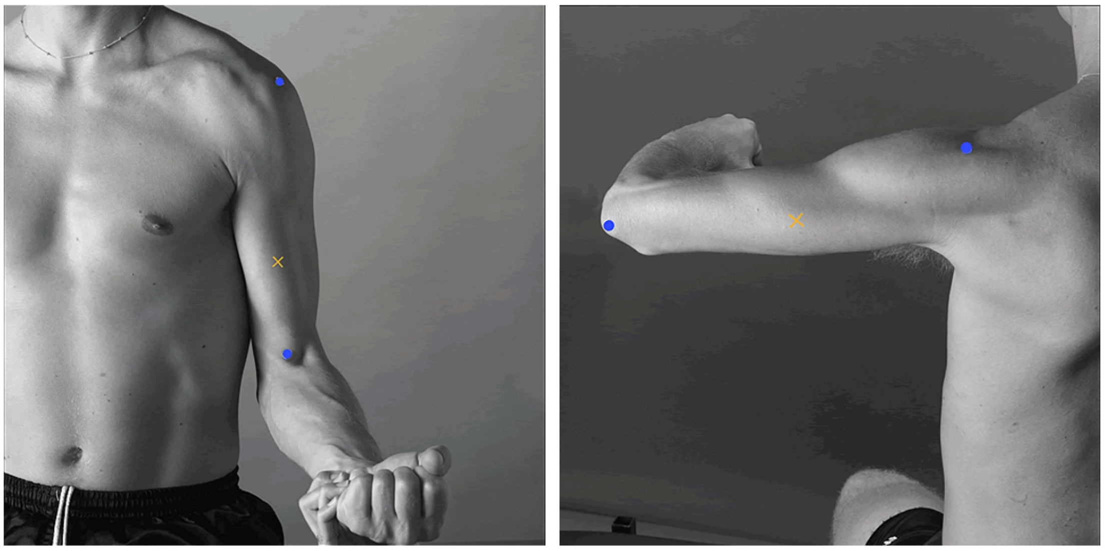

Figure 4.1: EMG electrode placements recommended by the SENIAM guidelines, for (on the left) the biceps brachii (Source: SENIAM (2006a)) and (on the right) the triceps brachii (Source: SENIAM (2006b)). Blue points are location references and the yellow crosses are the recommended sensor placement.

4.2 Processing of the sEMG Signal and Feature Extraction

Once the EMG signal has been acquired and sampled, a multi-stage process is employed to ensure the best possible inputs are fed into the control algorithm. All pre-processing steps were established with reference to the literature, based on the findings from Section 2.2.2. Moreover, all steps were performed using digital tools (refer to Section 5.3.1), after the signal is split into 250 [ms] windows of five 50 [ms] increments.

4.2.1 Pre-Processing

The sampled signal is band-pass filtered using a 20 [Hz] second order Butterworth low pass filter with a roll-off of 12 [db/oct] and a 450 [Hz] fourth order Butterworth high pass filter with a roll-off of 24 [db/oct], to remove noise outside the EMG's natural frequency spectrum.

To minimize power line noise, a 50 [Hz] second order IIR notch filter with a notch frequency of 50 [Hz] and lower and upper frequencies of 49.5 and 50.5 [Hz], respectively, are applied.

For both filtering stages, the normalized frequencies $f_n = f/f_s$ (f_s being the sampling frequency) were computed. However, according to Nyquist-Shannon theorem, the sampling frequency has to be at least two times the frequency f, and thus each above-mentioned frequency was divided by half the sampling rate, as shows Equation 4.1.

$$f_n = \frac{f}{f_s/2} \tag{4.1}$$

For the notch filter, in addition, the quality factor Q had to be computed. The quality factor determines how narrow or wide the rejection band is and can, as Nguyen (2017) explains, be found with greater precision by considering the effect of the sampling frequency (Equation 4.2):

$$Q = \frac{\tan(\pi f_N/f_s)}{\tan(\pi f_u/f_s) - \tan(\pi f_l/f_s)} \tag{4.2}$$

where f_N is the notch frequency (not the centre frequency, which is itself a geometric average of the upper and lower frequencies), f_u is the upper frequency, F_l is the lower frequency and F_s is the sampling frequency.

Finally, the signal is full-wave rectified.

4.2.2 Feature Extraction

Once pre-processing is complete, a number of features can be extracted. Following the findings of Phinyomark et al. (2013) (refer to Section 2.2.2), a set of four features was selected, namely, the Sample Entropy (SampEn) the Root Mean Square (RMS) the first four Cepstrum Coefficients (CC) and the Waveform Length (WL).

4.2.3 Motion Tracking

The motion tracking method described in Section 2.4.0.4 was used for this work, based on attitude estimation with Madgwick filters (refer to Section 2.4.0.3).

4.3 Simultaneous and Proportional Myoelectric Control

The control system was designed with a traditional random forest algorithm at its center. As the literature shows (refer to Section 2.3.1), RF is one of the algorithms being used for EMG-based simultaneous and proportional control. In this section, the random forest algorithm will be explained to a fair degree of detail. To do so, a brief introduction to decision trees is necessary, followed by a discussion of the bias-variance problem and a brief discussion of the suitability of random forests to myoelectric control are necessary. An explanation of Breiman's Random Forest algorithm follows and, to conclude, a discussion of hyperparameter tuning for random forests.

4.3.1 The Decision Tree Algorithm

Random forests, as first introduced by Breiman (2001), are decision tree based ensemble algorithms. Decision trees are simple classifiers where data is successively tested for certain conditions, starting at a root, until all examples are sorted into one of many leaf nodes. An example of a decision tree classifier can be seen in Figure 4.2.

The driving force behind decision trees is the minimization of node impurity - at each node, a rule (a split) is applied in an effort to separate a set of data points into two sets of data points that are less alike. All purity functions describe, in one way or another, the degree to which a

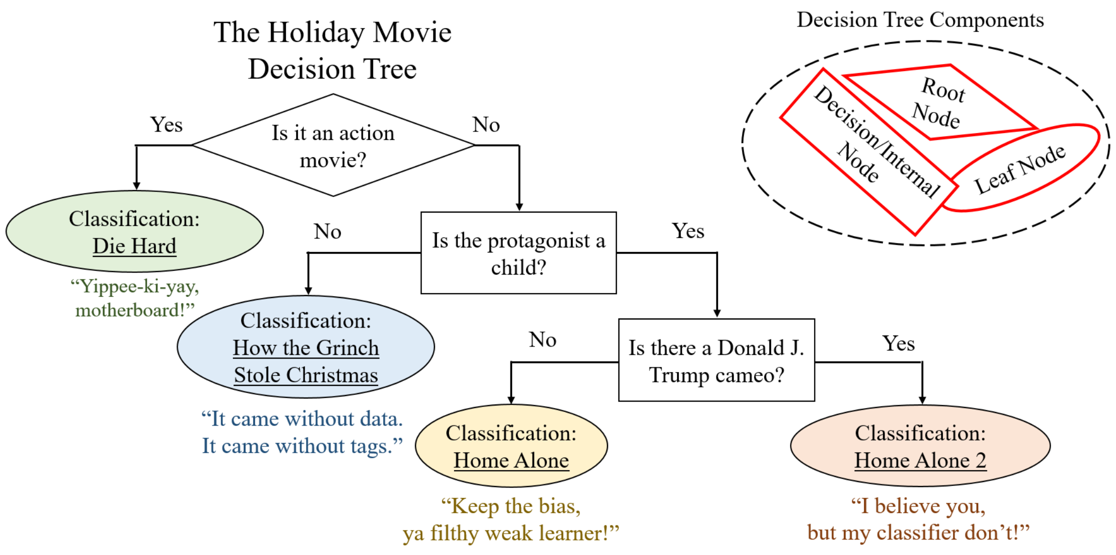
The Holiday Movie Decision Tree
Is it an action movie?
Yes
No
Classification: Die Hard
"Yippee-ki-yay, motherboard!"
Is the protagonist a child?
No
Yes
Classification: How the Grinch Stole Christmas
"It came without data. It came without tags."
Is there a Donald J. Trump cameo?
No
Yes
Classification: Home Alone
"Keep the bias, ya filthy weak learner!"
Classification: Home Alone 2
"I believe you, but my classifier don't!"
Decision Tree Components
Root Node
Decision/Internal Node
Leaf Node

Figure 4.2: An example of a decision tree classifier and its basic components. In this decision tree, the classification is entirely categorical, both in decisions and labels, though regression trees can exist.

group of data points has a homogeneous probabilistic landscape, that is, how close the group is to a perfectly impure set of data, which would have equal probabilities for all possible classification labels. There are several techniques for assessing purity at a node, the most popular being Gini impurity (Biau and Scornet, 2015).

For a regression problem, the criterion has to be distinct. Instead of splitting nodes with categorical variables, cuts are performed along a variable's axis, at a certain coordinate. There is a handful of viable procedures, the most common being the CART split criterion (Breiman, 2001), which stands for Classification and Regression Trees split criterion.

In CART splits, the quality of a cut is directly evaluated for its impact in group variance, the objective being to maximize the reduction from the parent node to each of the child nodes. The function in the CART split criterion for any given split on a tree n, along a variable j at the position z along that coordinate is (Breiman, 2001):

$$L_n(j,z) = \frac{1}{N_n(A)} \left[\sum_{i=1}^{N_n(A)} (Y_i - \bar{Y}_A)^2 - \sum_{i=1}^{N_n(A_L)} (Y_i - \bar{Y}_{A_L}) - \sum_{i=1}^{N_n(A_R)} (Y_i - \bar{Y}_{A_R}) \right], \quad (j,z) \in \mathbb{C}_A \tag{4.3}$$

with A being the set of data points in the node in question, A_L and A_R being, respectively, the "left" and "right" nodes resulting from the cut, $N_n(S)$ the number of data points in a generic data cell S, $\mathbb{C}_A$ being the set of all possible cuts in A, Y_i being the label of a certain data point $X_i \in A$ and $\bar{Y}_S$ being the expected label value in any given set of data points S.

In essence, the CART split criterion computes the difference between the parent node's variance and a weighted average of the variances of the two nodes born from the cut:

$$\begin{aligned} L_n(j,z) &= \frac{1}{N_n(A)} \left[\sum_{i=1}^{N_n(A)} (Y_i - \bar{Y}_A)^2 - \sum_{i=1}^{N_n(A_L)} (Y_i - \bar{Y}_{A_L}) - \sum_{i=1}^{N_n(A_R)} (Y_i - \bar{Y}_{A_R}) \right] = \\ &= \frac{1}{N_n(A)} \left[N_n(A) \cdot \sigma^2 - N_n(A_L) \cdot \sigma_L^2 - N_n(A_R) \cdot \sigma_R^2 \right] = \\ &= \sigma^2 - \frac{N_n(A_L)}{N_n(A)} \cdot \sigma_L^2 - \frac{N_n(A_R)}{N_n(A)} \cdot \sigma_R^2 \end{aligned} \tag{4.4}$$

where σ is the cell's standard deviation for variable j and σ_L and σ_R are the standard deviations for, respectively, the left and right child nodes.

4.3.2 The Bias-Variance Trade-Off and Ensemble Algorithms

Once a splitting criterion has been selected, the decision tree can be parametrized for optimal performance. In machine learning, meta parameters that determine mechanisms such as the learning process are traditionally called hyperparameters (Probst et al., 2019). In spite of not being a direct component of the classification or regression algorithms, they constrict the algorithm in specific ways, analogous to, say, the order of a polynomial regression. In the case of decision trees, there are a number of hyperparameters that require tuning in order to achieve the best possible performance. They can be tree depth, maximum and minimum node size, etc.

Notwithstanding, decision trees are weak learners (a weak learner is a learner that can always do, at least, slightly better than random guessing). Their biggest problem lies in the fact that they are mortally prone to overfitting. What this means is that when tuning a weak learner, there is a strong tendency for that weak learner to become a high accuracy predictor, though at the cost of becoming overtly specific. To put it simply, a decision tree can be near endlessly improved to become a powerful predictor for one specific sample, at the cost of becoming virtually useless for any data that slightly deviates from the former. This is what is often called the bias-variance trade-off and can be found illustrated in Figure 4.3.

In the case of decision trees, the implications could be, for example, that increasing tree depth until every single data point in the training data set is correctly classified causes the algorithm to lose accuracy if it were generalized to other data sets, even if belonging to similar (or the same) populations. That is to say, the model has a variance problem and, besides a few excellent predictions, the algorithm's performance will vary wildly. This is why it is called a trade-off - at a certain point it is virtually impossible to improve on a weak classifier's performance without making it less generalizable. On the other hand, keeping its performance below this break point means it will be a consistently unremarkable performer, and thus a high bias classifier - it is permanently distant from reality.

A tried-and-true solution to this problem of weak learners with costly bias-variance trade-offs, specifically learners with high variance, is a technique called bagging, generally found in ensemble algorithms.

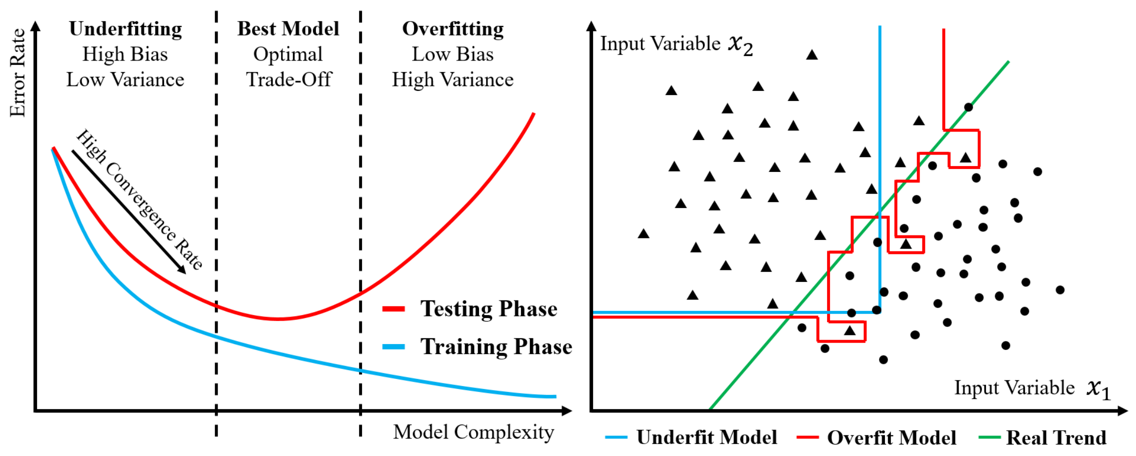

Figure 4.3: The bias-variance trade-off. On the left, the relationship between model complexity and the bias-variance trade-off. After an initial speedy convergence to a local solution, an increase in model complexity can have different outcomes, improving the fit on the training set and worsening the fit on the testing set and, consequently, any generic sample from the population. On the right, the classification of data based on two input variables, according to different models. Both an underfit model and an overfit model fail to capture the real underlying trend of the data set.

To better explain the reasoning behind ensemble algorithms, let us assume that we are in possession of a decision tree that has been trained on one unique data set, D, and promptly optimized for performance on that very same data set. As stated prior, this would likely be a blindly trained, high variance estimator, which we will call $h_D(\mathbf{x})$, $\mathbf{x}$ being any input vector and the index D telling of the data set on which the decision tree was trained. Let us assume we are also in possession of the expected estimator, $\bar{h}(\mathbf{x})$, that is, an estimator that, for any given input vector, computes the expected true label. The expected squared error (any other loss function would fit here) of the weak learner would then be given by (Biau and Scornet, 2015):

$$E_{\mathbf{x}}\left[(h_D(\mathbf{x}) - \bar{h}(\mathbf{x}))^2\right] \tag{4.5}$$

To further the previous hypothetical scenario, let us accept that in place of one unique data set, there were numerous unique data sets, and that each data set was used to train a separate decision tree, perhaps to the point of overfitting. Assuming these data sets were independent, by the weak law of large numbers, aggregating the estimates of all learners and computing the mean estimate should, in theory, converge to the expected estimator's prediction (Biau and Scornet, 2015):

$$\lim_{m\to\infty} E_{\mathbf{x},D}\left[\frac{1}{m}\sum_{j=1}^{m} h_{D_j}(\mathbf{x})\right] = \bar{h}_{\mathbf{x},D}\,(\mathbf{x}) \tag{4.6}$$

In other words, as the size of the ensemble of weak learners grows, the ensemble's fitness improves and variance is kept in check.

This set of estimators being aggregated would then be called an ensemble of weak learners. In the case of a random forest, the ensemble would be an analogous "forest" of multiple trees, hence the name.

Unfortunately, a faithful implementation of any ensemble algorithm would be utterly impractical. The requisite for multiple data sets is simply far too great.

In a random forest algorithm, a single data set is obtained and split into multiple bootstrapped (sampled, usually, with repetition) sample data sets. Then, multiple decision trees are trained with the data, constructing a single predictor from each bootstrapped sample. To improve performance, the bootstrapped set is not used in full at each node of every tree. Alternatively, a subset of variables are randomly selected as candidate variables and cuts are performed by taking into account exclusively the candidate variables. Finally, test data is classified using the aggregate, average estimation of all trees in the "forest". This averaging of decision trees is depicted in Figure 4.4.

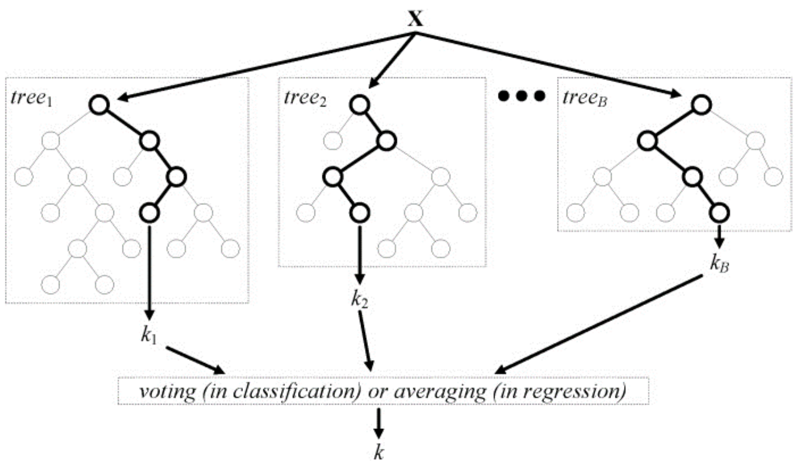

Figure 4.4: The classification process in random forests. Ensembles of Decision Trees provide each a prediction and the results are averaged. Source: Verikas et al. (2016).

This sampling technique of bootstrapping and aggregating is very appropriately called bagging. In this case, the weak law of large numbers is technically being violated because the bootstrapped sets are not independent, though the method still provides excellent approximations. As a matter of fact, it is one of the most effective computationally intensive procedures to improve on unstable estimates, especially for large, high-dimensional data sets, where finding a good model in one step is impossible because of the complexity and scale of the problem Biau and Scornet (2015).

4.3.3 Myoelectric Control and Decision-Tree Based Ensemble Algorithms

When it comes to simultaneous and proportional intention estimation, decision-tree based ensemble algorithms boast characteristics that are extremely valuable when using surface EMG data. More specifically, they are:

- Non-parametric and, as such, do not have a normality distribution assumption restricting its use (Nikulski, 2020). EMG data is frequently biased.

- Very robust to multicollinearity between features (Nikulski, 2020). Considering all features are being extracted from the same EMG signal or signals from potentially adjacent muscles, subject to identical mathematical treatment, correlations are bound to be heavily present.
- Robust to overfitting, noise and outliers (Nikulski, 2020). EMG data, as previously mentioned, is filled with artifacts, noise from surrounding muscle activity and other undesired electrical interference.
- Not sensitive to data magnitude (Nikulski, 2020) and, therefore, do not require the data points, in the case of EMG data, to be normalized, thus saving time.

Nowadays, there are several variants to the popular random forest algorithm that leverage the aforementioned idiosyncrasies to contrastive extents.

All these alternative algorithms, however, are more vulnerable to noise and data artifacts than the random forest algorithm. They tend to significantly outperform random forest in terms of training speed and efficiency, but when it comes to classification or regression performance, they have a hard time coming out ahead (Nikulski, 2020). Given that the task at hands involved a calibration and training phase separate from online testing, the choice ultimately resided on Breiman's Random Forest algorithm.

4.3.4 Breiman's Random Forest Algorithm

Breiman's Random Forest algorithm, akin to most machine learning algorithms, can be split into two different stages: training and testing.

For a random forest, training consists of the bootstrapping process and subsequent growing of trees from each data set. Testing, on the other hand, consists of classifying an input data point with every decision tree grown in the previous step and averaging the result.

Considering a step zero to be the separation of an original data set into two samples, one for training and the other for testing, the pseudocode for the algorithm of the training and testing stages of the random forest can be detailed as follows:

```
# Define the RF's hyperparameters
D: Original Data Set
m: Number of Trees
n: Size of Bootstrapped Data Sets
mtry: Number of candidate variables
nodesize: Limit node size for splitting it any further

(train_set, test_set) = Split D into a training and a testing set

# Build the RF model
Random_Forest = Training_Phase(train_set, m, n, mtry, nodesize)
```

```
# Classification / Regression of the testing data set
Predictions = Test_Phase(Random_Forest, test_set)

model Training_Phase(train_set, m, n, mtry, nodesize)
    # For every tree
    for i = 1, ..., m
        # Bootstrapping
        A = randomly selected n data points with replacement from the
        training data set train_set

        # Grow the tree with recursion
        branch = Grow_Tree(A)
    end for

    recursive function Grow_Tree(parent_node)
        # Check if node size is not too small
        if size(parent_node) < nodesize
            # Don't grow this tree branch any further, instead return
            the leaf node
            leaf_node = parent_node

            Return leaf_node
        else
            # Keep growing tree
            # Start by selecting candidate split variables
            candidate_variables = randomly selected mtry candidate
            variables

            # And finding the best split
            for j = 1, ..., mtry
                (j, z) = best split for candidate_variable(j) at
                coordinate z

                # Every new iteration, check if the new best split is
                the best over all variables
                Update or keep (j, z) accordingly
            end for

            # Perform the split
            left_node, right_node = CART_split(parent_node)

            # Grow the tree recursively
            left_branch = Grow_Tree(left_node)
            right_branch = Grow_Tree(right_node)
        end if
    end recursive function
end model

estimate Testing_Phase(model, test_set)
    # Classify every data point in the test set
    for data_point in test_set
        estimate(data_point) = 0

        # Compute the estimate for every tree
        for 1, ..., m
            estimate(data_point) = estimate(data_point) +
            + model.predict(data_point)
        end for

        # Compute the aggregate estimation
        estimate(data_point) = estimate(data_point) / m
    end for
end estimate
```

It should be noted that the underlying algorithm of this pseudocode is inefficient and computationally exorbitant. It could be further optimized. The implementation involved the use of an already optimized open-source package (refer to Section **??**).

In addition to the existing functionality, evaluation metrics could be added to the program and the algorithm's hyperparameters, such as `m` and `mtry`, could be tuned. Random forest hyperparameter tuning will be discussed in the following sections.

4.3.5 Random Forest Tuning: Empirical Rules

Like most machine learning algorithms, the performance of RF can be improved by adjusting some of its hyperparameters. As was aforestated, the hyperparameters of a learning algorithm are variables that remain unchanged during the learning process, yet can be used to tweak the learning process. Whereas model parameters are obtained from the training itself, model hyperparameters are derived from empirical knowledge, mathematical derivations or some form of model optimization.

Changing these meta variables in an effort to improve model performance is called tuning. The first step when attempting to tune an algorithm is, evidently, to identify the hyperparameters available. In random forests, the most common are (Probst et al., 2019):

- Number of candidate variables: When growing a given tree, splits have to be performed at each node. The best split can be found by looking at all possible splits across all variables, but this is very rarely the choice. The traditional procedure is to sub-sample the bootstrapped data set and select among a restricted number of candidate variables.
- Sample size: The bootstrapped data sets can be sampled with any size, and the choice is not trivial.
- Whether the sampling is done with or without replacement.
- Node size: By limiting the number of data points allowed to remain in a node before or after a cut, the learning process of individual trees can be manipulated.
- Number of trees: The number of weak learners in the ensemble algorithm.

Breiman's random forest's complex and abstract foundations mean that a purely analytical determination of the best set of hyperparameters remains to be derived, despite some existing efforts to progress the mathematical understanding of RF (Biau and Scornet, 2015). As a modest substitute, empirical rules have been established and amply studied in the scientific literature. Numerous times, these heuristics have been shown to be limited and highly data dependent (Probst et al., 2019). Nevertheless, they provide invaluable knowledge on the inner workings of this convoluted algorithm, which is why they will be succinctly discussed over the next few pages.

4.3.5.1 Number of Candidate Variables - `mtry`

The idea of picking a high number of candidate variables for each split is well financed in intellectual resources. It is prodigiously clear that, when attempting to optimize any given split at any given node in any given decision tree, a number of candidate variables close to the total of all available variables would result in a higher probability of selecting for the best candidate and, thus, growing the best possible predictor (Biau and Scornet, 2015).

Diversely, there is a remarkably good case, albeit less evident, for picking only a small fraction of all the available variables (Probst et al., 2019). For starters, selecting the strongest candidate variable is not necessarily an advantage - it makes individual trees more predictable and more strongly correlated with both the bootstrapped data set and the remaining trees in the forest, and obfuscates the chance of selecting lesser variables that could be bearing pieces of pivotal information. Not only that, picking a low number of candidate variables vastly improves on computational cost, given that the costliest step is, traditionally, the splitting procedure.

All in all, picking a high number of candidate variables increases the likelihood of picking the strongest variables, which in turn increases the risk of biasing the predictor too heavily, making it less stable. It is, summarily, an inevitable trade-off between accuracy and stability.

The default recommendation, in most software packages and empirical studies, is to use, for classification purposes, the square root, and for regression purposes, a third of the total number of variables in the model (Probst et al., 2019).

Nonetheless, even this value can be misleading - not all data sets have the same distribution of importance among existing variables, which should technically be taken into account when picking the candidates. The greater the count of highly relevant features, the lower the pool of candidates should be to give weaker variables a better shot at making the cut.

4.3.5.2 Bootstrapped Sample Size - `p` and `n`

`p` is the number of variables (dimensions) included in the bootstrapped sample and `n` is the number of data points in each sample.

Similarly to the number of candidate variables, smaller samples produce diverse, less correlated trees, at the cost of performance and accuracy at the individual tree level. Contrary to the last hyperparameter, however, the optimal sample size seems more problem dependent. Therefore, there are no widespread empirical tricks (Probst et al., 2019). In any case, it is well understood that decreasing the bootstrapped sample size improves computational efficiency.

4.3.5.3 Sampling With or Without Replacement

There seems to be little or no evidence of substantial impact of replacement in the model's performance, especially when the remaining hyperparameters have already been adequately tuned, though certain authors claim that sampling with replacement in problems with several categorical features should not be a given (Probst et al., 2019).

4.3.5.4 Node Size - `nodesize`

In most software packages, the default node sizes are 1 for classification and 5 for regression, which seem to garner good results (Probst et al., 2019). As the overall sample size and the number of noisy, low content features increases, though, node size should be incremented, otherwise a lot of resources will be wasted on relatively worthless splits. As a matter of fact, the computational expenditure seems to increase exponentially as the minimum node size for splitting requirement is lessened.

4.3.5.5 Number of Trees - `m`

According to Biau and Scornet (2015), the random forest's aggregate estimate can be defined, in a simplified manner, as:

$$h_{m,D}(\mathbf{x}) = \frac{1}{m}\sum_{j=1}^{m} h_{D_j}(\mathbf{x}) \tag{4.7}$$

where m is the number of trees, $\boldsymbol{x}$ is the input vector, the original data set is D and the bootstrapped data set for the jth tree is D_j.

Then, if we recall the reasoning presented in Section 4.3.2, by the weak law of large numbers, as we let m tend to infinity, the average of all trees in the forest will tend to the expected value of the classifier for the original data set (Biau and Scornet, 2015):

$$h_{\infty,D}(\mathbf{x}) = \bar{h}_D(\mathbf{x}) \tag{4.8}$$

where $\bar{h}_D$ is the predictor's expected value.

The variance of the classifier will fall without overfitting to the bootstrapped sets, that is, it will become independent of each bootstrapped set and will solely depend on the original sample (Biau and Scornet, 2015):

$$\lim_{m \to \infty} E[h_{m,D}(\mathbf{x}) - \bar{h}_D(\mathbf{x})] = E[h_{\infty,D}(\mathbf{x}) - \bar{h}_D(\mathbf{x})] \tag{4.9}$$

The bootstrapped data sets are not, of course, independent from the original data set. Under ideal conditions, i.e., assuming the original sample is a good descriptor of the real population, the random forest should, in theory, provide an excellent approximation to the real estimation problem. If the training set is limited, a large number of trees can lead to noticeable overfitting.

This gives valuable insight into why this algorithm can be really powerful in EMG-based intention prediction, since each trial represents, to a reasonable extent, whilst noisy, the true relationship between the EMG signal and motion intention.

Returning to the matter of hyperparameter tuning, this realization harbors both a positive and a negative side to it: on one hand, it becomes conspicuously apparent that the number of trees should be as high as possible. On the other hand, it is also clear that increasing the number of trees leads to a palpable linear increment in computational load. In a sense, the problem is not exactly one of tuning, but one of managing a trade-off.

The elementary approach to this trade-off is to establish what the convergence rate for this hyperparameter is and grow trees until the marginal utility gain is close to nil. Probst et al. (2019) hypothesizes that when tuning the other hyperparameters for stability and diversity among trees, a larger number of trees should be chosen to ensure every observation is correctly predicted.

4.3.6 Random Forest Tuning: Bayesian Optimization

Alternatively, the random forest's hyperparameters could be tuned using an optimization algorithm. Unfortunately, as previously mentioned, the random forest algorithm suffers from numerous limitations that make it unsuitable for most traditional optimization algorithms (Probst et al., 2019).

The biggest one is the nonexistence of proper mathematical formulation (Biau and Scornet, 2015). The greatest consequence of this is that an algorithm that relies on knowing completely the objective function is no longer applicable. An example is gradient descent optimization, which requires knowledge of the gradient.

Secondly, for a random forest regressor, similarly to most machine learning algorithms, a single iteration of training and evaluating the model is very expensive (Probst et al., 2019), which quickly becomes impractical if the optimization algorithm is very greedy.

Another problem of tuning random forests with optimization algorithms is that, like certain machine learning algorithms, they are rarely convex, that is to say they rarely converge on a single optimum, on which the latter frequently rely. A lot of algorithms can get stuck on local optima.

Bayesian optimization has surfaced as one solution to design problems where the breadth of explicit decisions to be taken is large. The goal of Bayesian optimization is to produce an optimal classifier that is both faster and more accurate (Shahriari et al., 2016).

Put simply, Bayesian optimization works by constructing a posterior distribution of Gaussian processes in an n-dimensional space, where n is the number of hyperparameters. The posterior results from fitting a Gaussian process to the observations obtained so far. An observation consists of a score (response variable) obtained by a model built with a certain set of hyperparameters (input variables). In this case, an observation could be the accuracy scores of the random forest algorithm on the available data set, given a certain set of hyperparameters.

As the number of observations grows, the posterior distribution improves, while the algorithm keeps track of which regions of the hyperparameter domain have been explored and to how much success. After every iteration, the algorithm optimizes an acquisition function, which takes into consideration the uncertainty of unexplored space (exploration of the hyperparameter space) and the scores of neighbouring observations (exploitation of the objective function). The optimized acquisition function outputs a target point in the hyperparameter space, which is used to obtain the next observation.

In essence, Bayesian optimization uses a proxy optimization problem - finding the maximum of the acquisition function - and progressively maps the hyperparameter space with every model found so far. This could be seen as redundant. In reality, provided the acquisition function and the exploration strategy are appropriately selected, Bayesian optimization can, in very few steps, find a combination of parameters that are close to the optimal combination (Shahriari et al., 2016). This is illustrated in Figure 4.5.

The increase in computational effort would be too large in traditional optimization procedures. However, when taking into account how costly a single observation can be, the algorithm ends up paying dividends. Therefore, Bayesian optimization is most adequate for situations where sampling the function to be optimized is a very expensive endeavor, which is the case for random forests (Probst et al., 2019).

Summaringly, a Bayesian optimization algorithm applied to a random forest regressor consists of the following steps (Shahriari et al., 2016):

1. Establish the domain of the hyperparameter space that is to be studied.
2. Select an initial arbitrary point in the hyperparameter space or one provided by the user. If this is not the initial point, select the next point to probe in the hyperparameter space by optimizing the acquisition function.

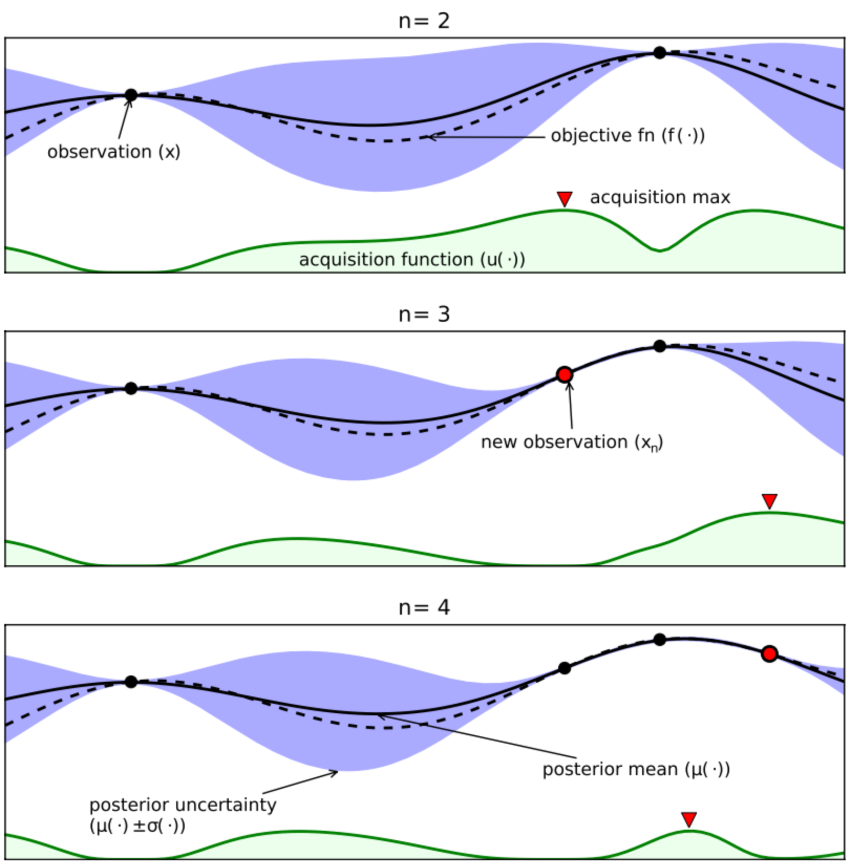

Figure 4.5: Illustration of the Bayesian optimization procedure over three iterations. The dashed line represents the real objective function, which is unknown. The dark line is the posterior, which is estimated by the Gaussian process from the available observations. Based on the uncertainty surrounding each observation (the blue shaded areas represent one standard deviation from the mean), the maximum of the acquisition function (the dark green line) is computed. The acquisition function balances exploration and exploitation, that is, it considers both unexplored areas of the hyperparameter domain and the potential to yield high scores of the objective function, when determining the next best candidate observation. n is the number of observations. Source: Shahriari et al. (2016).

3. Train a random forest using that set of hyperparameters.

4. Evaluate the objective function. This can be accomplished by splitting the data set into folds, performing cross-validation, and using the average score as the result.

5. Add the latest result to the set of observations and update the surrogate function (and its corresponding posterior distribution) by fitting Gaussian process priors to the data.

6. Repeat from (2) until predefined criteria are met.

An implementation of Bayesian optimization was done to tune the random forest. Scores of observations were computed through k-fold cross-validation of different trials, using the mean squared error as the measure. A light discussion of the measures employed for validation of the random forest models can be found in Sections 6.4.

Expected Improvement was selected as the acquisition function. There are numerous alternatives to this, whose treatment is beyond the scope of this work. A simple, traditional choice is the Probability of Improvement (Snoek et al., 2012), which, as the name suggests, takes into account how likely a set of hyperparameters is to yield an observation that results in a new best observation. Expected Improvement takes this one step further by also taking into consideration how large of an improvement each of these points would provide over the current best observation.

Whereas the Probability of Improvement acquisition function uses the area of the prior to estimate a probability of improving the performance of the objective function, the Expected Improvement takes into account the difference between the mean of a point's respective prior distribution and the mean of the current best observation (Shahriari et al., 2016):

$$\mathrm{EI}(\mathbf{x}) = \begin{cases} (\mu(\mathbf{x}) - \mu^+ - \xi)\,\Phi(Z) + \sigma(\mathbf{x})\,\phi(Z) & \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & \text{if } \sigma(\mathbf{x}) = 0 \end{cases} \tag{4.10}$$

where $\mathbf{x}$ is any set of hyperparameters in the hyperparameter space, μ^+ is the current best observation, $\mu(\mathbf{x})$ is the mean function of $\mathbf{x}$, ξ is a noise or error term, $\sigma(\mathbf{x})$ is the deviation function of $\mathbf{x}$ and $\phi(\cdot)$ and $\Phi(\cdot)$ are the PDF and CDF of the standard normal distribution.

The associated acquisition function can given by (Snoek et al., 2012):

$$\alpha_{EI}(\mathbf{x}; \{x_n, y_n\}) = \sigma(\mathbf{x}; \{x_n, y_n\})\,(\phi(\mathbf{x})\,\Phi(\phi(\mathbf{x})) + \mathcal{N}(\phi(\mathbf{x}); \nu)) \tag{4.11}$$

where $\{x_n, y_n\}$ is the current set of observations and $\mathcal{N}(\phi(\mathbf{x}); \nu)$ is a source of noise. Snoek et al. (2012) recommends $\nu = 0.1$, which was the value adopted in this work.

In addition to choosing an acquisition function, designing a Bayesian optimization procedure requires the selection of a covariance kernel function for the Gaussian prior. Again, discussion of kernel functions is outside the scope of this thesis. In short, a kernel function is a measure of similarity between data points that can be used as a computationally cheaper and algorithmically more effective alternative to the covariance matrix of the multivariate standard normal distribution.

As recommended by Snoek et al. (2012), the ARD Matérn 5/2 kernel was used:

$$K_{M52}(\mathbf{x}, \mathbf{x'}) = \theta_0 \left(1 + \sqrt{5r^2(\mathbf{x}, \mathbf{x'})} + \frac{5}{3} r^2(\mathbf{x}, \mathbf{x'})\right) \exp\left\{-\sqrt{5r^2(\mathbf{x}, \mathbf{x'})}\right\} \tag{4.12}$$

4.4 DC Motor Control

Once estimation of the angle by the random forest algorithm is complete, the device should be commanded to move to the target position. This is achieved by feeding a controller with that same information, so that it produces the correct output for the motor.

There are many types of controllers used in DC motor control, such as lead and lag compensators, LQRs (linear quadratic regulators), PID and sliding-mode controllers (Sabir and Khan, 2014). A standard solution is to use a PI or PID controller with negative feedback, and this choice is definitely not uncommon, for its simplicity and effectiveness make it a flexible tool (Somefun et al., 2020). When controlling DC motors, PI is sometimes preferred over PID, due to being simpler and still boasting acceptable performances. Regardless, the following discussion will be done with regards to a PID controller, since it applies very appositely to PI control.

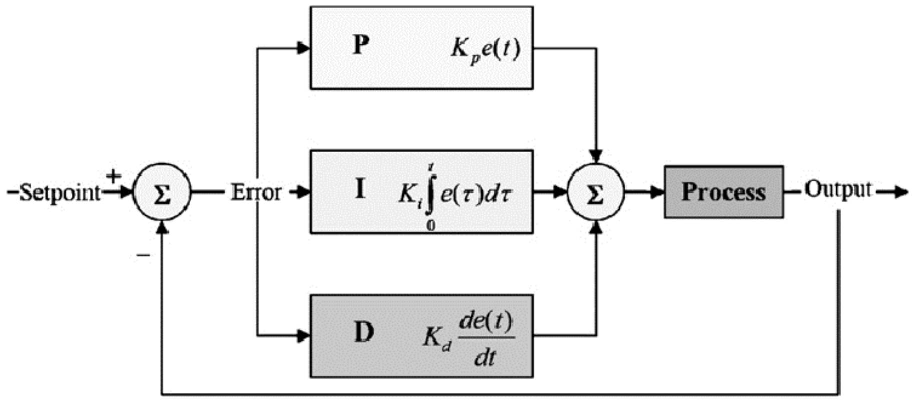

Figure 4.6: Block diagram of a PID controller. The output is compared to the reference and their difference is used to compute proportional (P), integral (I) and derivative (D) control actions, which are summed and their output fed to the process. Source: Biswas et al. (2009).

PID control can be interpreted as a collective action based on present errors (proportional control), past errors (integral control) and future errors (derivative control) (Ali et al., 2019), the last being based on current rate of change. The block diagram of a PID controller is illustrated in Figure 4.6. The time-domain control action of the PID controller is defined as follows Ali et al. (2019):

$$u(t) = K_p e(t) + K_I \int_0^t e(t) + K_d \frac{de(t)}{dt} \tag{4.13}$$

where $e(t)$ is the error at time t and K_p, K_I and K_d are the proportional, integral and derivative gains. Applying the Laplace transform gives:

$$U(s) = K_p E(s) + K_I \frac{E(s)}{s} + K_d E(s)\, s \tag{4.14}$$

The controller's open-loop s-domain transfer function would then be given by:

$$C(s) = \frac{U(s)}{E(s)} = K_p + K_I \frac{1}{s} + K_d\, s \tag{4.15}$$

The PID controller's open-loop transfer function *C(s)* can, additionally, be represented in a time constant formula, as the ISA standard defines (Vandoren, 2003):

$$C(s) = K_p\,(1 + \frac{1}{T_i s} + T_d\, s) \tag{4.16}$$

where all terms of the control action are defined as depending on the proportional gain. T_i is the integral time and T_d is the derivative time.

The integral time of a PID controller is the time it would take the controller's integral action to reach the same magnitude as the controller's proportional action, in an hypothetical scenario where the input error instantly increases from zero to a constant value. A PID controller with a large integral time constant is more heavily weighted towards proportional control, since its integral action dynamics are slower.

By contrast, the derivative time of a PID controller refers to the time it would take for the proportional action of the controller to catch up with the derivative action, in an hypothetical scenario where the error increases at a constant rate, starting at zero. Consequently, a PID controller with a large derivative time has a more aggressive derivative control action, in comparison to its proportional action.

It is worth bearing in mind that the controller will be implemented digitally. Consequently, the control action $u_k(t)$ at a step k will instead be given by:

$$u_k(t) = K_p \left(e_k(t) + \frac{1}{T_i} \sum_{i=0}^{k} e_i(t)\Delta t + T_d \frac{e_k(t) - e_{k-1}(t)}{\Delta t} \right) \tag{4.17}$$

4.4.1 PID Tuning

A large portion of mechatronic systems rely, at the most basic level, on the control of a DC motor (Somefun et al., 2020). This low level task, therefore, has to be adequately taken care of. As soon as a controller has been selected, the next step is to tune it. Tuning the controller consists of changing the controller's parameters in an effort to perfect how it reacts to the difference between the closed-loop system's output behaviour and the desired one (Vandoren, 2006). The biggest drawback of a PID controller is its reliance on precise, sometimes even manual, fine-tuning (Somefun et al., 2020). If the plant is overly sensitive to input changes, then adjusting the control action to be more conservative might be the right *modus operandi*. Ultimately, the controller is parametrized in the most aggressive possible manner, without letting the system become unstable.

It should be clear that despite having explicit, intelligible goals, tuning is not a straightforward task. When adjusting the control action, it isn't always obvious that the problematic outputs are indeed caused by disturbances, rather than measurement errors or defective control action.

This is why there is a wide range of tuning techniques, which can be, for instance, manual (graphical pole placement, heuristics, manual tweaking), automatic (e.g. with dedicated software) and performed online (while the plant is running) or offline (tuning based on a model).

Offline tuning boasts many advantages, most in the realm of safety - by testing a model *a priori*, there is a higher chance that online testing has a positive and expected outcome. On the other hand, as the common aphorism goes, all models are wrong. The only way to fine-tune a controller is to do it online. Usually, a combination of the two methods is employed, and the present case is no different.

The first step is to build, as accurately as possible, a model of the system. This necessitates previous knowledge of the system's parameters, in this case, the DC motor's transfer function, which can, in turn, be approximated from the motor's specifications. As was cautioned in Section 3.3.5, the parameters of the DC motor were mostly unknown at the time of motor selection. Fortunately, they can and were determined experimentally, and the DC motor's transfer function was promptly established. Discussion of this procedure for characterization of the motor and its transfer function can be found in Appendix A.

The controller can then be tuned by defining certain design requirements that pertain to the control response - these can be related to overshoot, speed, stability, robustness, etc.

Given the application, that is, a robotic assistive device, one characteristic of the control system stands out as critical to the success of any implementation *in vivo*: stability. Insufficient stability could precipitously undermine any effort to rehabilitate and reinstruct users to properly and efficiently use their limbs. Instability would pose an even greater risk in patients with tremors or neurodegenerative disorders (Gull et al., 2020), say Parkinson's disease. Another cause of concern is overshoot - if the system were to overshoot the reference angles, the device could put the user's limb in uncomfortable, even dangerous, positions, which is a prevalent risk among patients that are in recovery and rehabilitation - the target consumer of a powered orthosis. These concerns with stability and accuracy could even overshadow speed - for the average user of powered exoskeletons, a slow and steady pace is commended.

Tuning analytically with respect to specific parameters, like overshoot, is tempting. Nevertheless, this would be completely frivolous - as will become clearer going forward, the motor transfer function is, in actuality, grossly incorrect.

Alternatively, an initial optimal controller can be estimated and used as a starting point from which to conduct tuning via a heuristic tool. The controller gains yielded by the heuristic technique can then be manually fine-tuned.

There are numerous techniques and heuristics that make manual tuning significantly easier and more accessible. A widely accepted one is the Ziegler-Nichols (ZN) tuning method, which has two variants: open-loop tuning and closed-loop tuning (Vandoren, 2003).

In the ZN open-loop tuning method, the system's response to a step input is analyzed while keeping feedback shut off and the controller disabled. Evidently, this is not applicable to position control of a DC motor where the end-effector has a limited range of motion, where a constant control action would cause the robot arm to rotate until it stops at one of the ends.

The closed-loop version consists of progressively increasing the controller's proportional gain (while keeping the integral and derivative actions turned off) until sustained oscillations of period T_u (the ultimate period) start occurring. This gain threshold is called the ultimate gain K_u, and the optimal parameters for the controller can be promptly computed with the ZN parameters (Table 4.1).

The biggest drawback with this method is that it requires the process to be put in a state of oscillation, which could be undesirable, even if momentarily. In addition, ZN closed-loop tuning has a number of limitations and assumptions that make it hard to be generalized. Having said that,

Table 4.1: Ziegler-Nichol's parameters for P, PI, PD and PID controllers.

Controller	K_p	T_i	T_d
P	0.50 K_u	-	-
PI	0.45 K_u	0.80 T_u	-
PD	0.80 K_u	-	0.125 T_u
PID	0.60 K_u	0.50 T_u	0.125 T_u

the method is applicable to tuning the controller of a DC motor (Cinar et al., 2020), and was, in consequence, the choice for this work.

Chapter 5

Implementation

With the most important design decisions having been carefully defined, components were manufactured, the system assembled and the program coded. This chapter provides some insight into the steps, as well as criteria, taken in successfully implementing the system.

5.1 Design and Assembly of 3D Printed Device

To properly demonstrate the control system, a joint assistive device was designed and later 3D printed. The structure serves as a very coarse prototype of what could be a real world concretization of an externally powered elbow exoskeleton. For the results of this work to translate, as best as possible, to an *in vivo* implementation, the following design requirements were established:

- The device had to be anatomically compatible with the human elbow joint. It should feature appendages for attaching firmly to the human arm and forearm.
- It should have one DoF, as per the elbow joint's flexion/extension.
- This DoF should have a range of motion of at least 0 to 125 degrees.
- Weight and volume should be kept to a minimum, to prevent unnecessary loading of the subject or unwieldiness.
- It had to harbor a gearbox for power transmission with speed reduction, from the DC motor's shaft to the mechanical arm.

A more thorough explanation of design and assembly of the device can be found in Appendix C. Designing the gearbox encompassed characterization of the gears and pinions, as well as the gear train and computation of the necessary speed reduction steps. A description of this process can be found in Appendix B. Furthermore, a custom support was built for displaying the device. The assembled device and its custom support can be seen in Figure 5.1. As can be seen in the Figure, a "mannequin" hand with some weights was attached to the device, to broadly simulate the loading that the device would be subject to in a real life implementation.

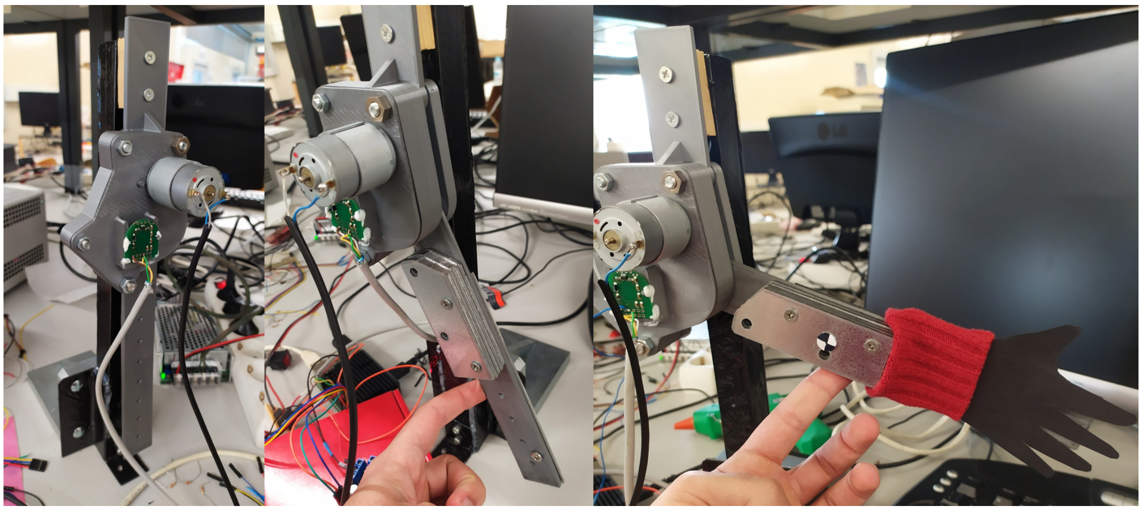

Figure 5.1: Assembled device with arm (left), weights (middle) and fake hand (right). The weights consist of a long steel plate (the length of the forearm) and several short steel plates. The long plate gives the arm more inertia, whereas the short plates give it weight, to help counterbalance the very high starting inertia due to friction.

5.2 Wiring and Assembly

The complete system can be seen in Figure 5.2. All wiring was done in accordance to the schematic of Section 3.4.

A difference worth highlighting is the existence of two ADC's, despite the ADS1115 board allowing for up to two differential channels. This is due to a limitation of the open source Python library for the ADS1115 Adafruit board[1], which, when requested by the I^{2}C master to switch channels, performs a few configuration steps that effectively reduce the sampling rate by up to (according to some initial testing) ten times. For this reason, a single ADC must be employed for each EMG channel, exclusively.

Finally, the fact that the entire data acquisition system is connected to the microcontroller via a single cable makes this part of the system very easy to don/doff, through the exclusive use of pre-gelled electrodes and medical adhesive tape thus satisfying one of the design requirements.

5.3 Codification

Programming was done in Python 3.7.3 and open-source packages were used whenever suitable. An initial version of the code was implemented in a Raspberry Pi OS (a Debian-based operating system for Raspberry Pi) virtual machine and subsequently tested on board with developer JetBrains' PyCharm Community 2020.3.3[2].

[1]https://github.com/adafruit/Adafruit_CircuitPython_ADS1x15, 09/03/2021.
[2]https://www.jetbrains.com/pycharm/, 07/03/2021.

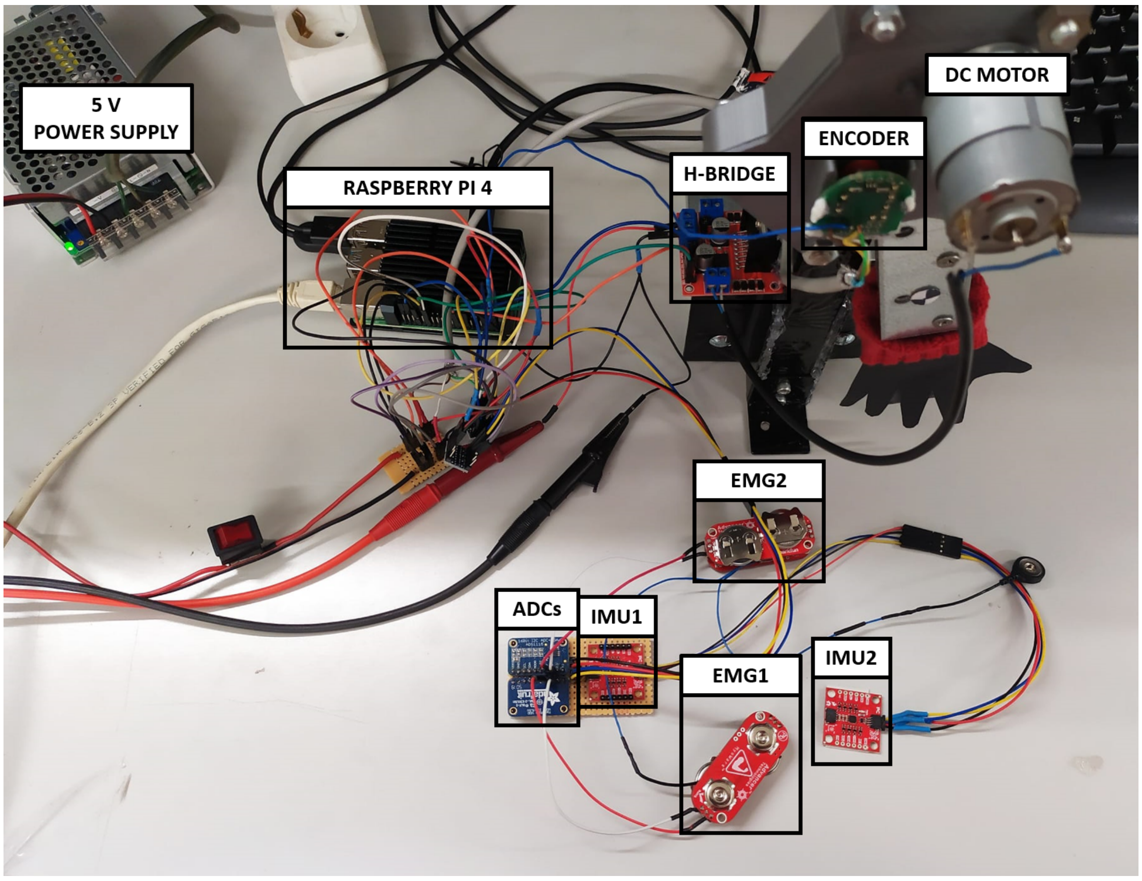

Figure 5.2: Complete system, with all components wired together.

5.3.1 External Resources

To save time, numerous tasks were performed using pre-existing publicly available open-source packages and modules. The most important ones are here mentioned.

The filter design parameters and filtering itself were accomplished with the SciPy (Virtanen et al., 2020) signal processing (`scipy.signal`) and multidimensional image processing (`scipy.ndimage`) packages.

Full-wave rectification was achieved with ease by merely taking the absolute of every point in the data set, which can be done with a variety of tools, in this case, the NumPy library's `numpy.absolute` (Harris et al., 2020).

For motion tracking, the authors of the Madgwick filter provide open-source modules (Madgwick, 2010) for a number of programming languages, namely Python, which can be found in the `ahrs` package, under `ahrs.filters.Madgwick`. The algorithm is simple to use, requiring only the creation of a `Madgwick` class object and the `Madgwick.updateMARG()` (in the case of inertial sensors that also incorporate a magnetometer) method to be called for run time iterative estimation.

For building and training the random forest, a popular open-source Python package was used: scikit-learn's `RandomForestRegressor` model (Pedregosa et al., 2011).

Bayesian optimization of the random forest model was performed with the `bayes_opt` package (Nogueira, 2014), a pure Python implementation of Bayesian global optimization with Gaussian processes.

5.3.2 Program Structure

As discussed in Section 3.2, the control system is distributed across four distinct tasks running in parallel:

1. EMG sensor data acquisition;
2. Motion tracking;
3. Intention estimation regression algorithm;
4. DC motor control.

When decentralizing the computational load, it is crucial to not forget the project at hand and the design requirements, functional and non-functional (refer to Section 3.1). Otherwise, the program might be inadequately tuned to the problem in question, as other less pressing matters take precedence.

To start with, it was defined that the robotic device would have to be as responsive as possible, to feel comfortable and intuitive to the user if it were to be implemented *in vivo*. In practical terms, this means, for example, minimizing computational dead time. Instead of having certain processes wait for the completion of others, they should function, whenever appropriate, asynchronously. This is not always feasible, which is the case for nearly all processing steps of incoming sensor data, which have to be chained sequentially. Here, intelligent distribution and fragmentation of processing steps is key.

Oppositely, reducing delay should not overshadow accuracy. In accordance with the findings of Section 2.2.2, input data has to span over a large enough interval of time to achieve sufficient precision. This does imply some computational dead time, as the processing unit waits for new data to be compiled. This undesirable necessity can be taken advantage of, by doing other lesser tasks concurrently. This turned out to be the case of the first two tasks: sensor data acquisition and motion tracking, which were split into two concurrent threads.

Ultimately, the Raspberry Pi was put in charge of three parallel processes, as illustrated in Figures 5.3 and 5.4.

Each process runs in parallel. This does not mean there is no synchrony. As a matter of fact, certain procedures ensure that the flow of data is sequential whenever necessary. An example of such a procedure is the use of locks. A lock is a parallel processing class that secures the safety of shared variables when used by more than one process. In this control system, the shared variables susceptible to instabilities caused by asynchronous programming are the batches of muscle sensor and motion tracking data, since they are the ones being accessed by different ongoing processes.

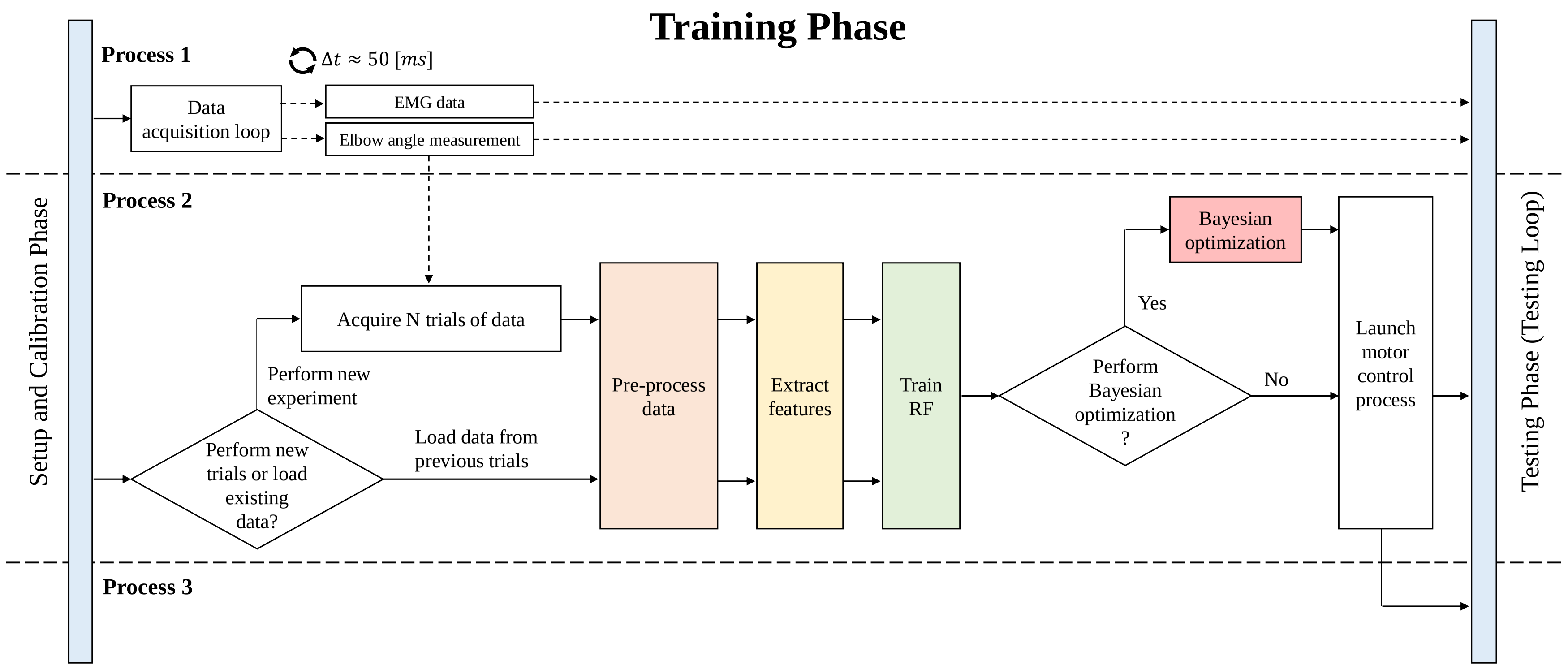
Training Phase
Setup and Calibration Phase
Process 1
$\Delta t \approx 50\ [ms]$
Data acquisition loop
EMG data
Elbow angle measurement
Process 2
Perform new trials or load existing data?
Perform new experiment
Acquire N trials of data
Load data from previous trials
Pre-process data
Extract features
Train RF
Perform Bayesian optimization ?
Yes
Bayesian optimization
No
Launch motor control process
Process 3
Testing Phase (Testing Loop)

Figure 5.3: Training phase flowchart. The user has the option to perform new trials (incoming data from the data acquisition process is stored) or load data from previous trials. The EMG data is pre-processed and relevant features extracted. Then, the random forest regressor is trained using default hyperparameters. Finally, before entering the testing phase, the user has the option to perform Bayesian optimization on the random forest.

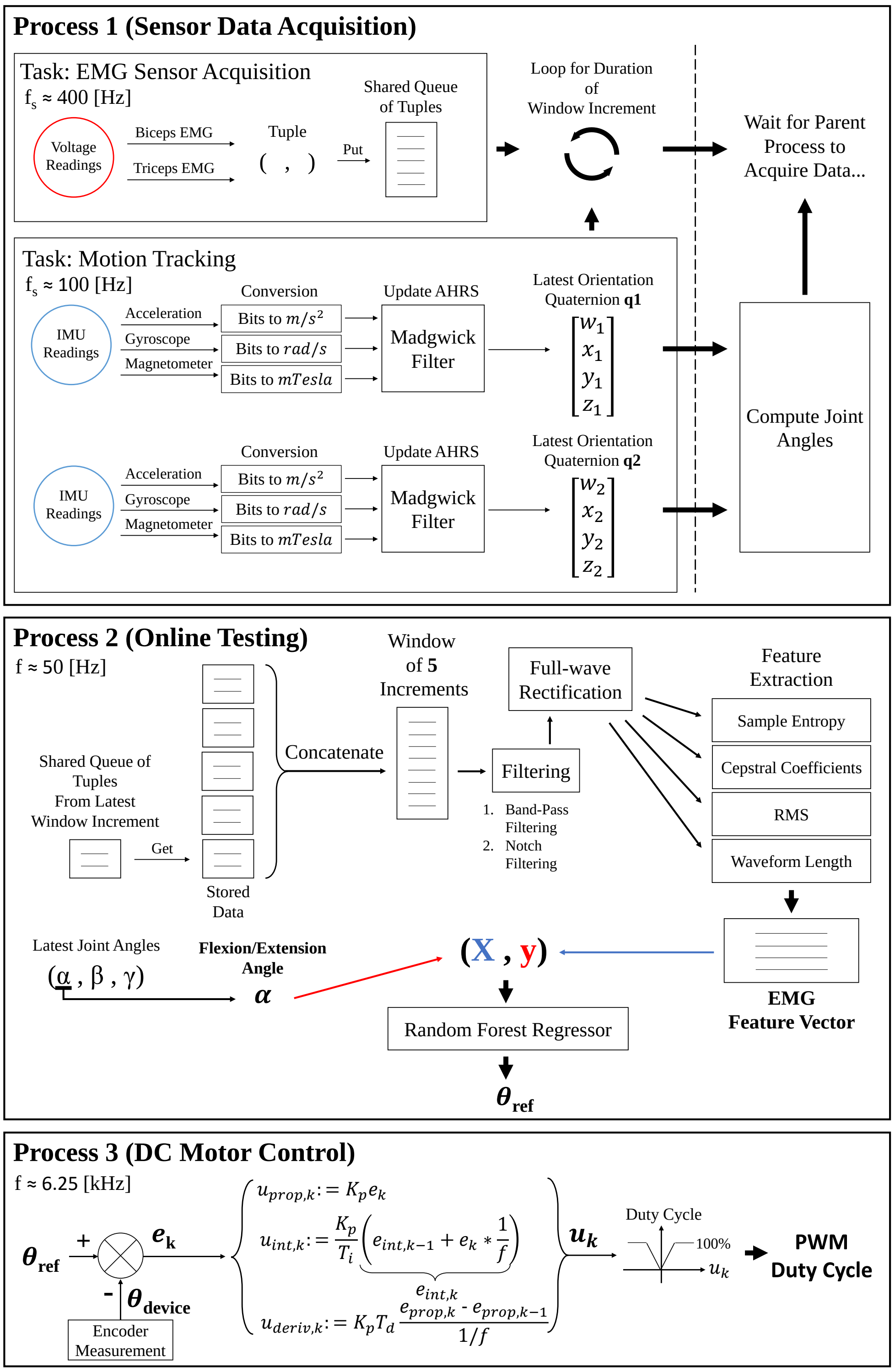

Figure 5.4: The three parallel processes' loops in the online testing phase. The first process deals with data acquisition: EMG data is collected and the orientation quaternions are updated, in a loop, for the duration of a sampling window increment. Then, the joint angles are computed and data is shared as soon as the parent process can collect the data. The second process joins the latest increment of data to previous increments, therefore creating a full window. EMG data is processed and features are extracted. A testing example is built and a reference angle produced. The reference angle is taken in by the third process, which feeds it to the PID controller. A control action is produced and mapped to the correct duty cycle scale (0 to 100%), which is used to update the Raspberry's PWM output pin.

The first process is responsible for running two concurrent threads: a sensor data acquisition and compilation thread and a motion tracking thread. A lock is acquired before any thread is started. The first consists of periodically (at a sampling frequency of $f_{s,EMG} = 400$ [Hz]) asking each 16-bit ADC for new EMG sensor data and saving it in memory. In reality, the sampling frequencies achieved were significantly lower (refer to Section 6.6). The data is aggregated in windows of length T_a (refer to Section 2.2.2 for a background on window averages) and discarded whenever too old to be held. The second thread consists of acquiring the IMU sensor's accelerometer, gyroscope and magnetometer measurements, applying the Madgwick filter and taking a step of the elbow angle's Gradient descent algorithm (refer to Section 2.4.0.4). Once enough muscular activity and kinematic data is recorded (that is, after a duration equal to a window increment T_{inc} has passed), the lock is released for the second process to retrieve the new data.

The second process starts by setting up the entire control system. Then, it performs different roles depending on the phase of the program. In total, it features three phases: setup, training and online testing. Figure 5.5 illustrates the calibration steps of the setup phase, Figure 5.3 is a flowchart of the training phase, and Figure 5.4 details the online testing phase.

In the setup phase, the first step is to import much needed libraries and modules. Next, shared variables are created, the first process is launched and calibration is performed. Shared boolean flags are used to coordinate the calibration steps. This calibration is mandatory for at least four reasons. For starters, the inertial magnetic sensors have to be calibrated for magnetic field warping and gyroscope bias drift. The first would put the magnetometer's accuracy at risk, the second would introduce a small, yet ever-increasing integration error, and both would severely impact the performance of the system. Consequently, the user is asked to remain in a static, stable position for 20 seconds, as shown in Figure 5.5, allowing the algorithm to compute the expected values of these errors and later use them to correct incoming data.

The second reason to calibrate is to allow the Madgwick filters to converge. This is because, as mentioned in Section 2.4.0.3, the Madgwick filter uses a Gradient Descent optimization algorithm to converge on the correct sensor orientation. To this end, the user remains in the same position as the previous calibration step and simply waits until convergence of the algorithm, which takes only a few seconds. Convergence is quantified by computing the dot product between the last two consecutive quaternions. If they are sufficiently similar, the Madgwick filter is said to have converged.

Next, to determine the sensor-to-segment orientations, the user has to perform, over the course of 45 seconds, periodic repetitive drills that consist of very simple elbow flexion/extension and forearm supination/pronation movements, again illustrated in Figure 5.5. After this period, during which the Madgwick filter's quaternion orientations and the IMU's gyroscope measurements are stored, the algorithm converges on a pair of sensor-to-segment orientation via gradient descent optimization. The joint angle estimation algorithm is explained in-depth in Section 2.4.0.4. An implementation based on quaternions was used. For more details refer to Laidig et al. (2017).

At last, the relative quaternion orientation can be decomposed in Euler angles. However, the joint angles cannot be computed at once, because of an arbitrary initial offset due to the arbitrary

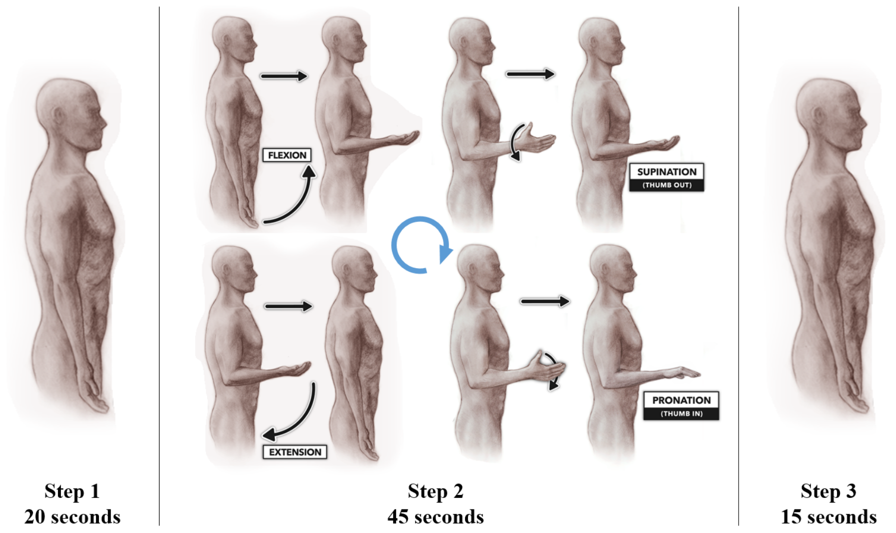

Figure 5.5: The three steps of calibration. **Step 1:** The subject keeps the arm in a straight, relaxed, vertical position for 20 seconds. **Step 2:** The subject repetitively performs flexion of the elbow, extension of the elbow, supination of the wrist and pronation of the wrist. These movements can be sequenced in any order. Each movement should take about 3 seconds. **Step 3:** The subject keeps the arm in a straight, extended, vertical position for 15 seconds. Adapted: CrossFit, LLC. (2019).

sensor positioning and orientation. The user is instead asked to remain in a known position for a brief instant (Figure 5.5), the Euler angle offsets are computed and compared to a reference. This concludes the setup phase, which is followed by the training phase.

In the training phase, as the name implies, the goal is to train the random forest intention estimation algorithm. Two alternatives are presented to the user - either new data can be collected, or pre-existing data sets can be used to train the algorithm. When asked to collect new data, the second process merely collects all window increments of EMG and elbow angle data received from the first process and aggregates it into multiple windows of length T_a. Every time data is fetched, the lock is acquired and later released. Once data acquisition from a user-defined number of trials is complete or a recorded set of data is loaded, the data is treated and relevant features are extracted.

The data is split into training, testing and validation sets. The user is then provided with two options: to train the random forest regressor using a pre-existing set of hyperparameters or to perform Bayesian optimization on the training and testing sets.

In the online testing phase, the third process is launched and the third process enters an endless loop, acquiring the lock every time a new window increment of data is available, collecting it and releasing the lock. Then, it pre-processes the EMG data, extracts relevant features and computes a random forest regression estimate. The last step in the loop is updating a shared variable, the target joint angle.

The target joint angle is used by the third process. This process is responsible for controlling the DC motor. Once launched (at the beginning of the online testing phase), it starts by setting up control variables and the Raspberry's GPIO peripherals, then entering a loop, the motor control loop. There, a step is taken in the controller's algorithm (compares the encoder's measured angular position to the target joint angle, computes errors and produces a control action) and updates the output variables - the PWM duty cycle and the motor driver's directional logic pins. This control loop runs at a significantly higher frequency than the intention estimation loop, which ensures that the controller acts upon the latest, most up-to-date data. This is illustrated in Figure 5.4.

By now, it should be clear that the "second" process, as defined in Figure 5.4, is, in actuality, the first one to be launched. It is, in fact, the main process and the parent process of both the first and third processes. There are numerous reasons for this.

Firstly, the intention estimation algorithm both requires inputs (the EMG and motion data) that are provided by the "first" process and produces an output that is needed to control the motor (the reference angle) and used by the "third" process. Consequently, from a programming perspective, it is much easier to have this as the main process, launching two other auxiliary processes that it shares key variables with.

Indeed, logically speaking, the random forest algorithm is the crux of this control system and even, perhaps, this thesis. It is undeniably the most important component in the control system and should thus take a central role, even in the code's architecture. As it happens, it is good coding practice to keep the code aesthetically tidy and well-organized.

Thirdly, the complete feature extraction and regression algorithm is not a repetitive task. This is because implementation of the regression algorithm demands separate training and testing phases. There are too many unique tasks that go into each of these two very distinct phases that would make the algorithm lengthier than necessary, and not at all iterable. This is not even considering Bayesian optimization, which only emphasizes the sequential nature of a machine learning algorithm. The "first" and "third" processes, by contrast, deal with tasks that can, after an initial setup, be run in endless loops: the data acquisition loop and the motor control loop.

In addition, the time it takes to extract features from incoming data and compute a random forest regression prediction is the common measure that coordinates the entire control system. Everything happens, as previously mentioned, with respect to the windows of EMG data. Given that the feature extraction and random forest regression is the longest, most computationally expensive step in the entire program, it serves as the least common denominator for all synchronous activities.

Finally, certain resources, functions, variables and methods are easier to access in the main process.

5.3.3 Modularity

Great effort was put in making the program as modular as possible. This was a very important goal because experimentation with the system has a large number of small decisions that can be taken. For instance:

- The desired processing steps of the sEMG data could be changed, to reflect advances in the field, changes in the components used, etc.
- The intention estimation algorithm could be changed. This is to be expected, given how many machine learning algorithms can be used (refer to Section 2.3.1 of the literature review).
- The dependent variable could change - instead of an arm angle, for example, force estimation could be desired.

These are just a few of the numerous minute variants that could be desired. The program was coded with this flexibility in mind, and these changes and many more are possible with relative ease.

5.4 Tuning the Motor Controller

As discussed in Section 4.4.1, before operation begins, the motor controller needs to be tuned.

The characterization of the DC motor in Appendix A culminated in the obtainment of the following transfer function for the DC motor (from Equation A.11):

$$G(s) = \frac{0.036}{2.788 \times 10^{-4} s^2 + 0.8143 s + 0.0708} \tag{5.1}$$

Combining this with the controller's transfer function and taking into account the speed reduction stage from the gearbox to which the motor is attached (as mentioned above, the system features a gearbox for speed reduction - design of the gearbox can be found in Appendices B and C), the actuation control loop can be established in a more concrete manner. The resulting control loop is represented in the MATLAB®Simulink block diagrams of Figure 5.6.

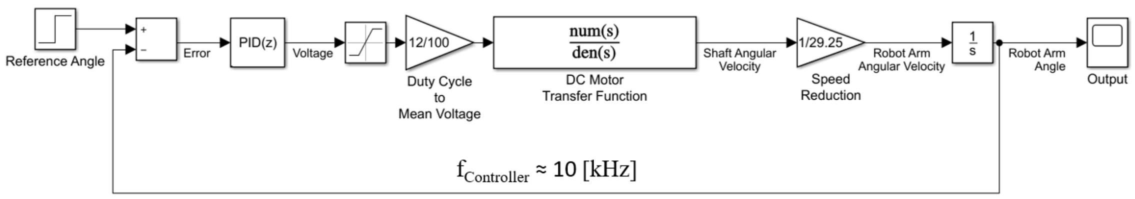

Figure 5.6: The DC motor actuation control loop. Here, the PID controller is represented by a discrete controller. Discrete since, technically speaking, it will be implemented digitally. In practice, nevertheless, the motor controller is so fast (nearly 10 [kHz]) when compared to the device's dynamics and the rest of the algorithm, that it essentially behaves as a continuous controller.

An initial estimate for the Proportional Gain K_p was computed using MATLAB®'s PID automatic tuning tool, applied to the block diagram of Figure 5.6, with the PID controller block being replaced by a simple P controller.

For a Proportional Gain of a little over 100, the system started to overshoot. This already represented a significant departure from reality, given that the real system was found to occasionally overshoot for as little as $K_p = 10$. This, of course, only served to demonstrate how imprecise the model was. To simply tune the controller based on this simulation would be risky, irresponsible, even, on the grounds that it is a definite departure from reality.

Instead, the Ziegler-Nichols test was performed with an initial conservative K_p of 10. This should give the reader a grasp of how little confidence there was in this initial estimate. This was, of course, justified, based on earlier informal testing.

Recorded data of the closed-loop ZN-tuning experiment is plotted in Figure 5.7. After the system achieved a somewhat steady state, the proportional gain was progressively incremented until sustained oscillations could be detected. As the gain increased, the device would sometimes get stuck, which paired with the innately high starting torque would cause it to immediately overshoot and then correct itself - this phenomenon explains the spurious spikes of Figure 5.7 top.

Eventually, at a K_p of 86, the system showed the first manifestation of a sustained oscillation (shown amplified in Figure 5.7 bottom), which lasted near one and a half full seconds. The proportional gain was increased, to ensure the phenomenon hadn't been anomalous. The system continued to show brief periods of sustained oscillations, which increased both in frequency and intensity as the gain grew larger. The largest showed an amplitude of nearly 3 degrees, justifying the interruption of the experiment.

From Figure 5.7, the ultimate gain and period, K_u and T_u, were computed, with values of 86 and 0.05 seconds, respectively.

Using Table 4.1, three alternatives were selected for further consideration: PI, PD and PID. After some experimentation, the PID controller was selected. The integral term helped correct small steady state errors, whereas the derivative term proved successful in avoiding most instances of overshoot. The gains were further fine-tuned manually and the resulting controller is also shown in Table 5.1. The resulting K_p is the lowest of all to ensure a smooth trajectory and the resulting T_d is the highest because a more aggressive derivative action was necessary due to high starting friction. Its response to a step reference joint angle of 50 [deg] is shown in Figure 5.8.

Table 5.1: Ziegler-Nichol's parameters for P, PI, PD and PID controllers, given the values obtained experimentally for K_u and T_u.

Controller	K_p	T_i	T_d
P	43	-	-
PI	39	0.04	-
PD	69	-	0.006
PID	52	0.025	0.006
Final Controller	30	0.02	0.5

This serves as an excellent example of deficiencies with modelling and offline testing. Firstly, the model revolved around a plant (the DC motor) transfer function that was obtained with an unreasonably high degree of uncertainty. Not only was the equivalent circuit used to approximate the

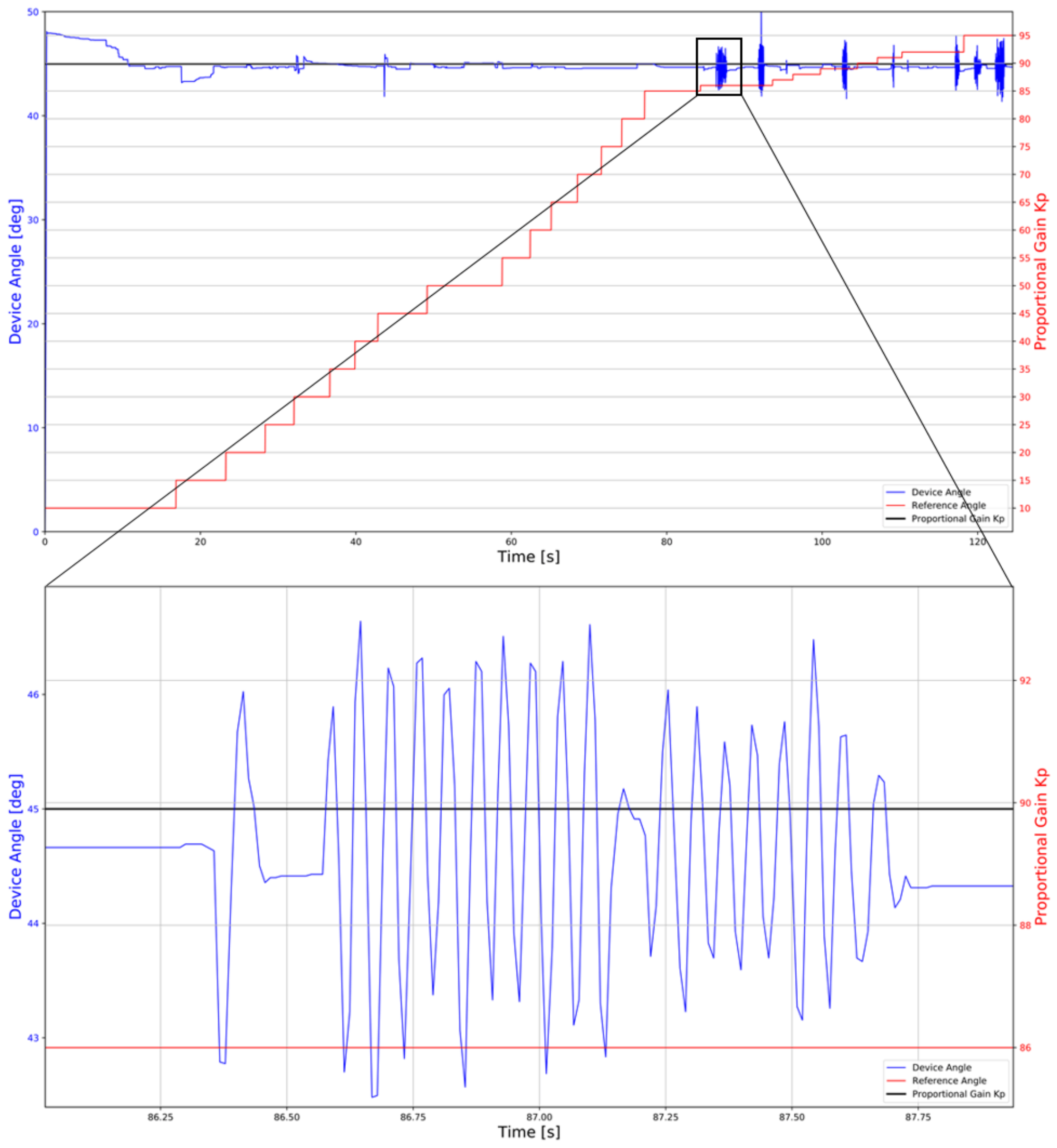

Figure 5.7: Recorded data of the closed-loop ZN-tuning experiment. At a K_p of 86, the system showed the first sustained oscillation, which is shown amplified in the bottom figure.

motor, by itself, limited, but the parameters were selected through grossly imperfect experiments and based on fallible criteria.

Furthermore, a multitude of differences exist between the control loop of Figure 5.6 and the real thing. For example, energy losses were not factored in, which can become a very gross oversight when dealing with a 3D printed device with 3D printed gears and other 3D printed moving parts.

This could be an explanation for the exceptionally high starting torques found. What would otherwise be slight tracking corrections, could become unreasonably large overshoots of several degrees. In fact, the motor was found to completely stall at any duty cycle under about 33%, that is, for an equivalent mean voltage of 4 V. Evidently, this introduced severe non-linearities in the plant which further invalidated the model.

On second thought, in spite of all its limitations, the model was an invaluable tool in revealing the inefficiencies of the real system. Once again, as the common aphorism goes, all models are wrong, but some are useful.

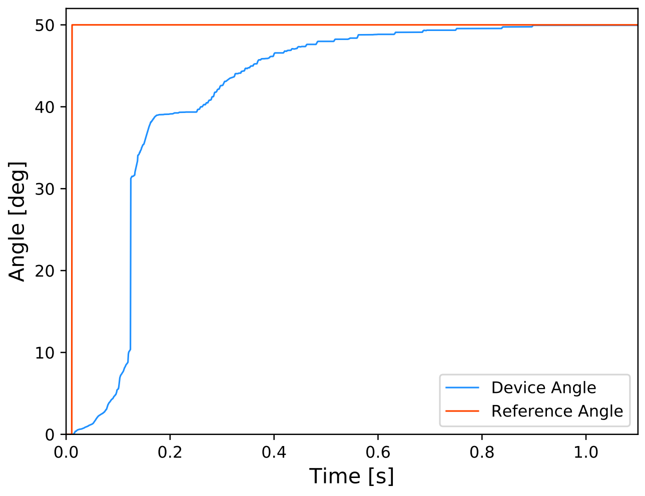

Figure 5.8: Response of the DC motor to a step input of 50 [deg]. The fine-tuned PID controller allowed for a fast, yet smooth, response and consistent minimal or no overshoot.

Chapter 6

Validation

This chapter pertains to the experiment designed to evaluate the implemented system. First, objectives for the experiment were set. From these, research questions and their associated hypotheses were drawn. Based on the measures being used, statistical tests were selected to test their respective statistical hypotheses. Results of the experiment are presented and discussed.

6.1 Objectives

The main objective of the experiment was to validate the quality of the solution, that is, the system that was designed and implemented.

To this end, both the intention estimation algorithm and the complete system should be analyzed, not just the latter. On one hand, the random forest regressor was designed so as to accurately describe the relationship between the sEMG signal acquired and the measured elbow joint flexion/extension angle. This presumes testing the algorithm offline after training it on acquired data.

On the other hand, if the remainder of the system is incapable of accurately replicating the reference angle computed by the random forest regressor, then the system does not perform to an acceptable level.

From here, three research questions were established:

RQ1: What is the level of performance of the system? This pertains to both the intention estimation algorithm by itself and the complete system in online testing.

RQ2: Do the intention estimation algorithm in offline validation and the complete system in online testing perform significantly better when the random forest model is tuned with Bayesian optimization, relative to using predefined hyperparameters?

RQ3: Are there differences between the performances of the intention estimation algorithm in offline validation and the complete system in online testing?

RQ3 follows the findings of Chapter 2. When trying to transition from an offline environment to online testing *in vivo*, studies have shown that offline testing results are not necessarily a good indication of online performance (Jiang et al., 2014). Thus, it is relevant to assess the difference between online and offline performance.

6.2 Hypotheses

Each of the aforementioned research questions were expressed in more concrete terms, by means of a set of null hypotheses and their respective alternative hypotheses.

RQ1: Having been considered a mostly descriptive research question, no hypotheses were established for this question.

RQ2: A simple but effective way of comparing the performance of the algorithm and the system overall when using default hyperparameters relative to optimized hyperparameters is to compare their accuracies when tested offline and online.

Consequently, the following hypotheses were established for RQ2:

$H1_0$ Null Hypothesis: The intention estimation algorithm and the robotic device show no significant differences in accuracy when the random forest regressor is trained using optimized hyperparameters relative to default hyperparameters.

$H1_a$ Alternative Hypothesis: The intention estimation algorithm and the robotic device have significantly better predictive accuracy when the random forest regressor is trained using optimized hyperparameters relative to using default hyperparameters.

RQ3: This question can be considered descriptive and as such no hypotheses were drawn for it.

6.3 Subjects

The experiment was performed on a single subject, a 23 year old male with 70 kg and self-reported sub 12% body fat. The subject had previous knowledge of sEMG, despite no practical experience. The subject's carrying angle was measured to be approximately 15º.

6.4 Measures

The system's accuracy can be measured using the errors between its output and the measured output, the outputs here being angles of flexion/extension of the subjects' elbow joint.

The joint angles were measured in real time using the motion tracking solution discussed in Section 2.4, which combines inertial magnetic sensors, attitude estimation and the joint angle algorithm detailed in Section 2.4.0.4. The precision of these measurements are dependent on the IMUs' resolution, quantization errors, spurious errors, etc.

When analyzing the intention estimation algorithm, the output is directly an estimated angle that the random forest regressor computes based on an input feature vector. These inputs, of course, are obtained from the EMG sensors, whose signal is fed to a 16-bit ADC and only then received by the microcontroller. The sEMG signals, as discussed in Section 2.2, have numerous flaws, such as being very noisy.

The complete system's output is the motion of the robotic device. In this case, the relevant output variable is the angular position of the arm, which is measured with a 13-bit magnetic encoder. This sensor, of course, is subject to magnetic interference from the environment and other sources of errors.

On these accounts, the accuracy can be quantified in numerous ways. When using a machine learning algorithm, it is common to use the coefficient of determination, R^2, which measures the amount of variance in the dependent variable that is explained by the independent variable(s). A general definition of the coefficient of determination is:

$$R^2 = 1 - \frac{\sum_i (y_i - f_i)^2}{\sum_i (y_i - \bar{y})^2} \tag{6.1}$$

where y_i is the value of the dependent variable of observation i, $\bar{y}$ is the mean of the data set and f_i is the model's estimate.

Evidently, this is a very simple way of comparing models, since the R^2 of each variant can be compared and thus provide some insight into the goodness of fit of a model or system.

The two biggest disadvantages of the coefficient of determination are its dependence on the number of independent variables relative the sample size and its dependence on the number of independent variables included in the model (Alshibly, 2018). Fortunately, the number of variables in the random forest model does not change throughout the experiment. The manipulation of hyperparameters in Bayesian optimization could be seen as an increase in the number of independent variables, but this effect was not considered.

Therefore, the values to compare in the context of RQ2 and RQ3 will be the R^2 for the following four different scenarios:

1. R^2 computed from the intention estimation algorithm's joint angle estimates, when tested offline with default hyperparameters.
2. R^2 computed from the intention estimation algorithm's joint angle estimates, when tested offline with optimized hyperparameters.
3. R^2 computed from the intention estimation algorithm's joint angle estimates, when tested online with default hyperparameters.
4. R^2 computed from the intention estimation algorithm's joint angle estimates, when tested online with optimized hyperparameters.

6.5 Preparation and Execution

The experiment followed the steps described in the Experiment Protocol of Appendix D. In this appendix, some pictures of the experiment can be found, such as electrode and sensor attachment to the subject, the muscle sensors' output on an oscilloscope, side-by-side comparison of the subject and the device during online testing, etc.

6.6 Results and Discussion

Data acquired during the trials can be found in Appendix D. Overall, sEMG data acquisition and motion tracking seem to have performed up to expectations. Perhaps most noticeable is the higher amplitude of the biceps' signal relative to that of the triceps. This could be related to faulty electrode placement or the nature of the movement not requiring as much effort from the triceps (which are working in favour of gravity). There is a difference between the biceps' signal during the first three trials and the last three. This makes sense from an anatomical perspective, considering that the brachialis muscle shares more load during this part of the movement when the wrist is neutral, as was the case in the last three trials.

Sampling frequency during the trials was worse than expected, ranging from 200 to 300 [Hz] instead of the expected 400 [Hz]. The explanation for such a low sampling frequency is the lengthy I^2C routine for requesting data from the ADS1115, which does not seem to be optimized, as was already mentioned in Section 5.2. As reviewed in Section 2.2.2, lower sampling frequencies, especially those below 450 [Hz], are expected to incur in greater losses of signal information.

Data was collected and split across 6850 windows of 250 [ms] in 50 [ms] increments. Training the random forest with the default hyperparameters took less than 10 seconds. This is very impressive considering pre-processing the entire 6850 samples took approx. 75 seconds and feature extraction took approx. 440 seconds. Taking into consideration the length of the calibration procedure (recorded at approx. 3 minutes) and the entire trial duration (over 6 minutes), it could be expected for the entire system to be ready in less than 20 minutes.

Figure 6.1 displays the offline estimation of the RF algorithm, for both the default and optimized cases. Coefficients of determination for the two tests with the test trial (trial number 4, selected at random) were $R^2 = 0.6499$ for the RF with default hyperparameters and $R^2 = 0.7134$ for the RF after 20 iterations of Bayesian optimization. It can be affirmed that the optimized model performed better. This is also clear in the figure - with the same amount of data, the Bayesian optimized random forest managed to much better describe the measured angles, especially the peaks of the movement, though the two models show some spurious spikes that were not otherwise present in the reference. Contrastingly, it should be noted that building the model with Bayesian optimization took over 40x longer, clocking in at around 440 seconds.

Concerning the Bayesian optimization process itself, the feature optimizations most worth noting were the number of trees and the maximum number of data points per bootstrapped sample. The estimated objective function after 20 iterations is shown for the two features in Figure 6.2.

With regards to the first, the number of trees in the forest, there seems to be a very anomalous low at the exact value of the default number of estimators, 200. This could of course be related to some unforeseen interaction between the default values of the features. At around 400 the

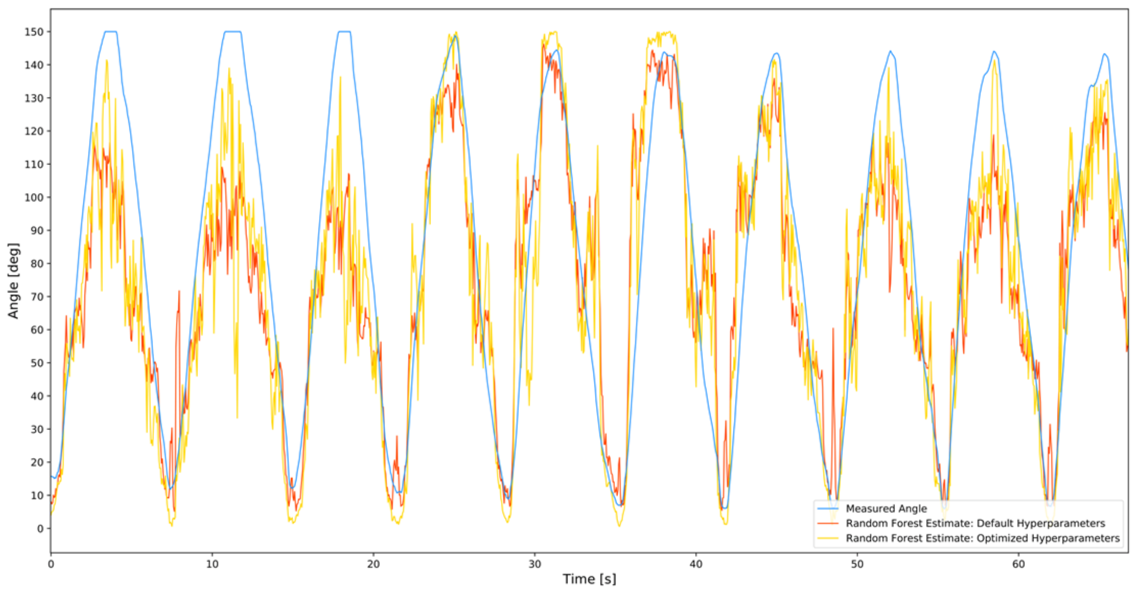

Figure 6.1: Offline intention estimation of the test data.

objective function scored the highest, which could imply there is some saturation to the added benefit of increasing trees, after a few hundreds of trees have been grown.

With respect to the max bootstrapped sample size, the performance of the algorithm also seems to saturate, in this case, when each bootstrapped sample reaches about 20% of the complete data set. Both of these findings seem to suggest that there is little performance improvement for the large computational cost of progressively increasing the complexity of the random forest.

Variable importance for the default RF and the optimized RF was computed using MDI (mean decrease impurity). Results are shown in Appendix D, Table D.1. MDI computes the weighted decrease of impurity corresponding to splits along a feature and averages this quantity across all trees (Biau and Scornet, 2015). It is worth noting that MDI has been shown to behave poorly when correlation between features increases (Biau and Scornet, 2015), which definitely is the case for EMG since all the features are extracted from the same raw signals.

In both models, WL and RMS scored very high (only RMS_{tri} is under a 10% contribution), which is not at all unexpected when considering the literature. They are both known to be very simple and effective features (Phinyomark et al., 2012). Noticeable is also the first cepstral coefficient for both muscles. This is impressive, but not that surprising, considering they come with large computational costs compared to other features (Phinyomark et al., 2012). Naturally, however, as the number of coefficients increase they become less and less important, therefore as expected. Perhaps most surprising is the fact that sample entropy scored so low, which does not seem to agree with the findings of Phinyomark et al. (2013), reviewed in Chapter 2.

When considering the online performance of the algorithms, shown in Figures 6.3 and 6.4, their degraded performance relative to that of the offline tests is visible. For starters, the algorithms do not seem to be able to reproduce any peaks whatsoever, and even their performance at the lowest points of the movement seem to be inferior. For the random forest with default parameters, the

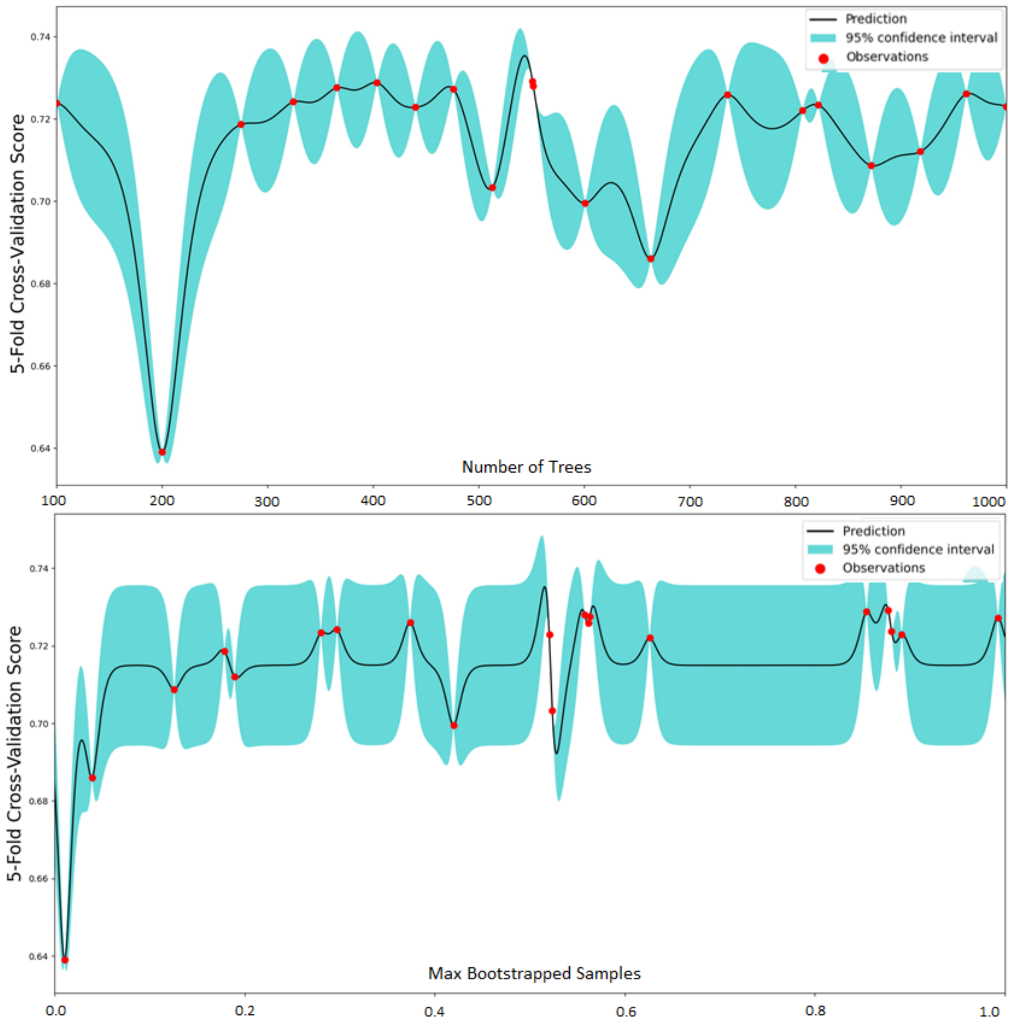

Figure 6.2: Bayesian optimization's estimated objective function after 20 iterations, for the number of trees and the maximum size of bootstrapped samples.

device's arm barely even moved below 45º.

For the online tests, two different values of R^2 could be considered. The first, in the same manner as the offline tests, can be computed using the measured joint angle and the algorithm's estimated angle. Oppositely, the performance of the system as a whole could be considered. Thus, R^2 would be computed using the measured joint angle and the device's measured angle. Both provide valuable information. Their values were $R^2_{estimate} = 0.5393$ and $R^2_{device} = 0.5546$ for the system with default hyperparameters and $R^2_{estimate} = 0.0736$ and $R^2_{device} = -0.1345$.

For the RF with default hyperparameters, both values are clearly worse than those obtained offline, with the device R^2 being slightly higher, though this difference is very small.

Surprisingly, for the optimized RF, both values were much worse than all R^2's computed thus far. The main reason for the regressor's poor performance was the fact that the subject moved faster than for two periods of the movement, as can be seen in the Figure. Nevertheless, the relevance of the rest of the test should not be discarded. Regarding the device, the poor estimates were

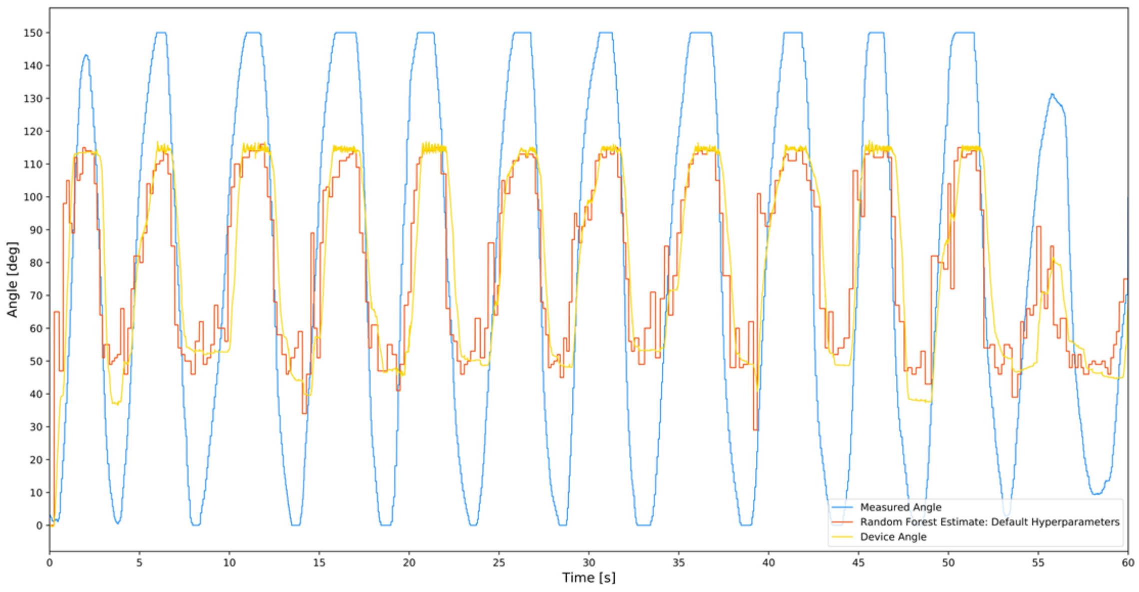

Figure 6.3: Online test of the system with the RF model trained using default hyperparameters.

exacerbated by what seems to have been significant gear wear (the online tests were performed after weeks of testing of the device), until the moment where the gearbox seems to have broken and collapsed, causing the arm to get completely stuck. This happened at around 25 seconds of testing, at which point the test was stopped and the remainder of the data scrapped.

The latency of the estimate also seems to be much larger, which is to be expected, considering the added computational delay of online intention estimation. Despite noticeable, however, the computational latency was within the established objectives, that is, in the same order of magnitude of the sampling windows (of approx. 250 [ms] of length), at a mean 202 [ms] for the default model and 325 [ms] for the optimized one. This stark increase is to be expected, since the Bayesian optimization did not take into account model complexity when selecting for the best performer. Most of the mean time was spent computing the random forest estimates, at 129 [ms] for the default RF and 253 [ms] for the optimized RF. For both, the mean time spent pre-processing was approx. 9 [ms] and the mean time spent extracting features was approx. 61 [ms].

The standard deviation of the time spent by the regressor was 33 for the default RF and 50 [ms] for the optimized RF. Neither seem to be very consistent. Considering the human tactile reaction time[1] to be around 150 [ms], this might be too inconsistent for a user.

To conclude, with respect to **RQ1**, the system performance has been characterized using measures of accuracy and time. When compared to the literature, such as Xiao et al. (2018a), who used MTDF-RF to obtain a mean joint angle estimation error of 0.0543 ± 0.0071, the system's offline performance, with a highest $R^2 = 0.7134$, does not particularly stand out. When being tested online with default hyperparameters, the lowest value obtained for the coefficient of determination under regular conditions was $R^2 = 0.5393$. In online testing, the longest mean time it

[1]https://backyardbrains.com/experiments/reactiontime, 09/03/2021.

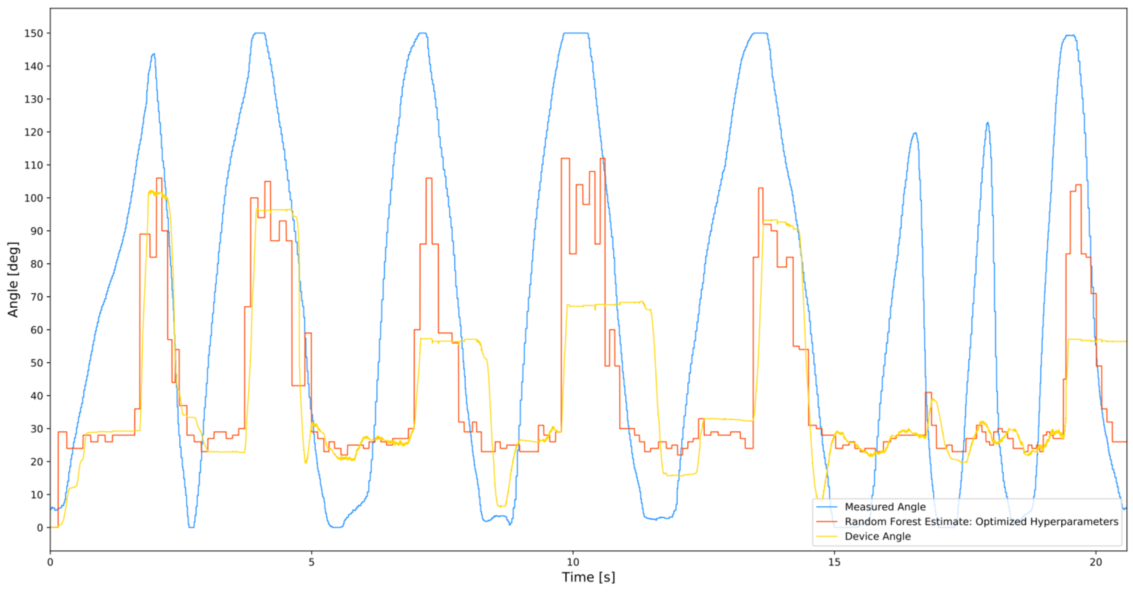

Figure 6.4: Online test of the system with model trained using optimized hyperparameters.

took to react to a sample was less than 400 [ms] (after considering the window increment time, the computational dead time and one period of the motor control loop), for the online testing system.

It is hard to say whether the alternative hypothesis has been confirmed for **RQ2**. Indeed, the offline performance after Bayesian optimization seems to have been consistently better than the offline performance using default hyperparameters, as expected. However, the irregularities of the online test for the optimized system led to very contrasting results in the online tests. The online test with default hyperparameters also showed no such difference. Consequently, it is hard to say that the null hypothesis has been rejected.

Regarding **RQ3**, the results of the experiment corroborate very strongly with the assertion that offline training and testing bears limited similarity to online testing. Not only were the accuracies of the models considerably lower during online testing, the behaviour of the regressor itself seemed very different in the two scenarios. To add to this, a slight deviation from the expected movement led to complete inability of the intention estimation algorithm to properly estimate.

Chapter 7

Conclusion

In this chapter, the most important conclusions and limitations of this work are presented. In addition, some insights into future works are provided.

7.1 Objectives Achieved

The main objective of this work, to design and implement a low-cost control system for an early prototype of a powered 1 DoF elbow exoskeleton, was achieved. This was done based on the findings of a review of the literature, which provided crucial insights into the theory and technologies behind most modern powered upper limb exoskeletons, without which the project would not have been possible.

Designing the entire solution involved the design and implementation of a system for EMG data acquisition, the design of an intricate and layered control system, based on a random forest regressor, for estimating the user's motion intentions and simultaneously and proportionally control a 3D printed 1 DoF robotic device. This system was implemented using affordable, off-the-shelf components, which were studied, wired and programmed to their intended functionality. The 3D printed device was designed and assembled successfully. Incremental testing was employed to increase the likelihood of operability of the system as a whole. This was only possible thanks to the modular structure of the system that conveyed it some amount of testability.

Once complete, the solution was evaluated in a carefully designed experiment, with clear goals and challenges, and conducted in an orderly fashion, in an effort to ensure consistent, reproducible results. Flexibility of the design also made the system permeable to future improvements.

Most design requirements were met. The 3D printed device was built with a 0 to 145º range of motion, surpassing the requirement for 0 to 125º of flexion. The online testing delay was of the same order of magnitude as that of the length of a sampling window of EMG data. The EMG data acquisition system, together with the motion tracking sensors, was designed in an easy-to-don/doff way, involving only the attachment of medical adhesive tape and pre-gelled electrodes, making it time efficient, thus building a foundation for a user friendly device.

7.2 Limitations and Future Works

In spite of having fulfilled its objectives, a number of limitations to the design, implementation and validation steps were present in this work.

To start with, the number of participants was extremely low, with just one. In addition, the subject already had knowledge of sEMG, despite no practical experience with devices based on myoelectric control. It is evident that this limitation greatly restricts the generalizability of the experiment and its results. Any future works should envision the recruitment of more participants.

Secondly, despite performing largely up to expectations, the system is too slow to react and too inconsistent in its performance for an *in-vivo* implementation. In order to make the system faster, all three steps of intention estimation should be improved. They are pre-processing, feature extraction and the regression model. Pre-processing can easily be hastened by doing all pre-processing steps electronically. Ideally, the components would be created specifically for the effect. When doing so, amplification, analog-to-digital conversion and sampling could be performed directly on the sEMG sensors, minimizing computational dead time and preventing issues like unnecessarily long setup between the microcontroller and the sensors.

Making feature extraction faster could be achieved by eliminating features that represent too large a computational cost relative to the performance improvement they grant. Feature dimensionality reduction techniques can be useful for this purpose. Compared to the literature, the intention estimation algorithm performed worse. Xiao et al. (2018a) obtained a mean root mean squared angle estimation error of 0.0543 ± 0.0071 when using MTDF random forests. The modularity of the system could be taken advantage of to try and implement different algorithms. Instead of random forests, algorithms such as temporal convolutional networks with adaptive reinforcements (Betthauser et al., 2020) could be explored for improvements in both accuracy and speed.

Faster predictions could be achieved by reducing the length of the window required, provided sEMG signals are being sampled from more muscles. These could also account for movements where the main agonist and antagonist muscles participate less, such as compound movements. To do so, more movements should be integrated in the training trials, to improve robustness of the algorithm. This, of course, comes with the cost of more setup time.

According to the literature reviewed, the sampling frequency should also have been higher, up to two times as fast. This could be implemented by using components that are specialized for speed. Of course, this would have to be done without incurring in large cost increases. Motion tracking speed could, too, be improved if attitude estimation was decentralized and allocated to the sensors themselves.

Another limitation of the solution devised is its mechanical characteristics. After all, the 3D printed gears did succumb to wear during the experiment. For this reason, other materials and manufacturing processes could be explored in future works.

Ultimately, once all the foregoing considerations, as well as many others, are addressed, solved and settled, designing the device for a higher number of joints would be the next logical step.

Bibliography

Adafruit. Ads1115 16-bit adc - 4 channel with programmable gain amplifier, 2020. URL `https://www.adafruit.com/product/1085`.

A.D.A.M., Inc. Electromyography, 4 2018. URL `https://www.rohm.com/electronics-basics/ac-dc/rectification`.

Advanced Arm Dynamics, Inc. Prosthetic options, 2020. URL `https://www.armdynamics.com/our-care/prosthetic-options`.

Advancer Technologies, LLC. Myoware muscle sensor, 2020. URL `http://www.advancertechnologies.com/p/myoware.html`.

Ali Ali, Ali Alhelli, and Hassan Alwan. Tuning pid controllers for dc motor by using microcomputer. *International Journal of Applied Engineering Research*, 14:202–206, 01 2019.

Haitham Alshibly. How to statistically compare the coefficient of determination (r2) among multiple simple linear models?, 01 2018.

AnyBody Technology A/S. The anybody modeling system, 2020. URL `https://www.anybodytech.com/software/ams/`.

Arduino. Arduino uno rev 3, 2020. URL `https://store.arduino.cc/arduino-uno-rev3`.

Panagiotis Artemiadis. Emg-based robot control interfaces: Past, present and future. *Advances in Robotics & Automation*, 01, 01 2012. doi: 10.4172/2168-9695.1000e107.

ASUSTek Computer Inc. Asus tinker board, 2020. URL `https://www.asus.com/us/Single-Board-Computer/Tinker-Board/`.

BeagleBoard.Blue. Beaglebone blue, 2020. URL `http://beagleboard.org/blue`.

BeagleBoard.org Foundation. Beaglebone black, 2020. URL `http://beagleboard.org/black`.

J. L. Betthauser, J. T. Krall, S. G. Bannowsky, G. Lévay, R. R. Kaliki, M. S. Fifer, and N. V. Thakor. Stable responsive emg sequence prediction and adaptive reinforcement with temporal convolutional networks. *IEEE Transactions on Biomedical Engineering*, 67(6):1707–1717, 2020. doi: 10.1109/TBME.2019.2943309.

Luzheng Bi, Aberham >Genetu Feleke, and Cuntai Guan. A review on emg-based motor intention prediction of continuous human upper limb motion for human-robot collaboration. *Biomedical Signal Processing and Control*, 51:113 – 127, 2019. ISSN 1746-8094. doi: https://doi.org/10.1016/j.bspc.2019.02.011. URL `http://www.sciencedirect.com/science/article/pii/S1746809419300473`.

Gérard Biau and Erwan Scornet. A random forest guided tour. *TEST*, 25, 11 2015. doi: 10.1007/s11749-016-0481-7.

Arijit Biswas, Swagatam Das, Ajith Abraham, and Sambarta Dasgupta. Design of fractional-order pi l d m controllers with an improved differential evolution. *Eng. Appl. of AI*, 22, 01 2009.

Karla Blocka. Eeg (electroencephalogram), 09 2018. URL `https://www.healthline.com/health/eeg`.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Lili Cao, Tadashi Masuda, and Sadao Morita. Compensation for the effect of soft tissue artefact on humeral axial rotation angle. *Journal of medical and dental sciences*, 54:1–7, 03 2007.

Catherine Caruso. Cybathlon winner: Bob radocy, 10 2016. URL `https://www.trsprosthetics.com/cybathlon-winner-bob-radocy/`.

Said Cinar, Zekeriya Balci, and İsmail Yabanova. Performing speed control of a dc motor with auto-tuning pid. 19:690–696, 01 2020. doi: 10.35414/akufemubid.593609.

James G Colebatch, Sendhil Govender, and Danielle L Dennis. Postural responses to anterior and posterior perturbations applied to the upper trunk of standing human subjects. *Experimental brain research*, 234(2):367–376, 2016.

CrossFit, LLC. Movement about joints, part 2: The elbow, 3 2019. URL `https://www.crossfit.com/essentials/movement-about-joints-part-2-the-elbow`.

Paolo de Leva. Adjustments to zatsiorsky-seluyanov's segment inertia parameters. *Journal of Biomechanics*, 29(9):1223 – 1230, 1996. ISSN 0021-9290. doi: https://doi.org/10.1016/0021-9290(95)00178-6. URL `http://www.sciencedirect.com/science/article/pii/0021929095001786`.

Carlo J. De Luca. The use of surface electromyography in biomechanics. *Journal of Applied Biomechanics*, 13:135–163, 05 1997. ISSN 1543-2688. doi: 10.1123/jab.13.2.135. URL `https://www.delucafoundation.org/download/bibliography/de-luca/078.pdf`.

Carlo J De Luca and Roberto Merletti. Surface myoelectric signal cross-talk among muscles of the leg. *Electroencephalography and clinical neurophysiology*, 69(6):568–575, 1988.

Carlo J. De Luca, L. Donald Gilmore, Mikhail Kuznetsov, and Serge H. Roy. Filtering the surface emg signal: Movement artifact and baseline noise contamination. *Journal of Biomechanics*, 43(8):1573 – 1579, 2010. ISSN 0021-9290. doi: https://doi.org/10.1016/j.jbiomech.2010.01.027. URL `http://www.sciencedirect.com/science/article/pii/S0021929010000631`.

Brian Douglas. Matlab tech talks: Understanding sensor fusion and tracking, part 2: Fusing a mag, accel, & gyro estimate, 10 2019. URL `https://www.youtube.com/watch?v=0rlvvYgmTvI`.

MsC. Alejandra Garcia Toll Dr. Gonzalo Gonzalez Rey, Eng. Christian Irving Enrique Rodriguez Gonzalez. A procedure to determine the unknown geometry of cylindrical gears, 2 2016. URL `https://gearsolutions.com/features/a-procedure-to-determine-the-unknown-geometry-of-external-cylindrical-gears/`.

A.C. Fischer-Cripps. 3.3 - instrumentation amplifier. In A.C. Fischer-Cripps, editor, *Newnes Interfacing Companion*, pages 246 – 260. Newnes, Oxford, 2002. ISBN 978-0-7506-5720-4. doi: https://doi.org/10.1016/B978-075065720-4/50119-4. URL `http://www.sciencedirect.com/science/article/pii/B9780750657204501194`.

FriendlyElec. Friendlyelec, 2020a. URL `https://www.friendlyarm.com/`.

FriendlyElec. Nanopi m4b, 2020b. URL `https://www.friendlyarm.com/index.php?route=product/product&path=69&product_id=275`.

Sergio Fuentes, Yuyang Wei, Ester Olmeda, Lei Ren, Guowu Wei, and Vicente Díaz. Validation of a low-cost electromyography (emg) system via a commercial and accurate emg device: Pilot study. *Sensors*, 19:5214, 11 2019. doi: 10.3390/s19235214.

Glenn Research Center. Aircraft rotations, 05 2015. URL `https://www.grc.nasa.gov/www/k-12/airplane/rotations.html`.

Google, LLC. Coral dev board, 2020. URL `https://coral.ai/products/dev-board/`.

R.A.R.C. Gopura, D.S.V. Bandara, Kazuo Kiguchi, and G.K.I. Mann. Developments in hardware systems of active upper-limb exoskeleton robots: A review. *Robotics and Autonomous Systems*, 75:203 – 220, 2016. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2015.10.001. URL `http://www.sciencedirect.com/science/article/pii/S0921889015002274`.

Muhammad Gull, Shaoping Bai, and Thomas Bak. A review on design of upper limb exoskeletons. *Robotics*, 9:16, 03 2020. doi: 10.3390/robotics9010016.

Janne Hahne, F. Biebmann, Ning Jiang, Hubertus Rehbaum, Dario Farina, Frank Meinecke, Klaus-Robert Müller, and Lucas Parra. Linear and nonlinear regression techniques for simultaneous and proportional myoelectric control. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 22:269–279, 03 2014. doi: 10.1109/TNSRE.2014.2305520.

L298N Dual H-Bridge Motor Driver. Handson Technology. URL `https://components101.com/sites/default/files/component_datasheet/L298N-Motor-Driver-Datasheet.pdf`.

Hardkernel Co, Ltd. Hardkernel, 2020. URL `https://www.hardkernel.com/`.

Charles R. Harris, K. Jarrod Millman, St'efan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fern'andez del R'ıo, Mark Wiebe, Pearu Peterson, Pierre G'erard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL `https://doi.org/10.1038/s41586-020-2649-2`.

B. Hudgins, P. Parker, and R. N. Scott. A new strategy for multifunction myoelectric control. *IEEE Transactions on Biomedical Engineering*, 40(1):82–94, 1993. doi: 10.1109/10.204774.

Invensense. Icm-20948, 2020. URL `https://invensense.tdk.com/products/motion-tracking/9-axis/icm-20948/`.

ISO 21771:2007(E). Gears — Cylindrical involute gears and gear pairs — Concepts and geometry. Standard, International Organization for Standardization, Geneva, CH, 9 2007.

N. Jiang, I. Vujaklija, H. Rehbaum, B. Graimann, and D. Farina. Is accurate mapping of emg signals on kinematics needed for precise online myoelectric control? *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 22(3):549–558, 2014. doi: 10.1109/TNSRE.2013.2287383.

Xin Jin, Yusheng Cai, Antonio Prado, and Sunil Agrawal. Effects of exoskeleton weight and inertia on human walking. pages 1772–1777, 05 2017. doi: 10.1109/ICRA.2017.7989210.

Reva E. Johnson and Jonathon W. Sensinger. Chapter two - actuator technologies. In Jacob Segil, editor, *Handbook of Biomechatronics*, pages 31 – 59. Academic Press, 2019. ISBN 978-0-12-812539-7. doi: https://doi.org/10.1016/B978-0-12-812539-7.00002-7. URL `http://www.sciencedirect.com/science/article/pii/B9780128125397000027`.

Wen-Juh Kang, Jiue-Rou Shiu, Cheng-Kung Cheng, Jin-Shin Lai, Hen-Wai Tsao, and Te-Son Kuo. The application of cepstral coefficients and maximum likelihood method in emg pattern recognition. *IEEE transactions on bio-medical engineering*, 42:777–85, 09 1995. doi: 10.1109/10.398638.

Ayoosh Kathuria. Intro to optimization in deep learning: Gradient descent, 6 2018. URL `https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/`.

Séamus Kennedy and Robert Meler. A dynamic approach to managing partial foot amputees. *The O&P EDGE*, 01 2011. doi: 10.1016/j.apergo.2017.10.008.

Kang Soo Kim, Heung Ho Choi, Chang Soo Moon, and Chi Woong Mun. Comparison of k-nearest neighbor, quadratic discriminant and linear discriminant analysis in classification of electromyogram signals based on the wrist-motion directions. *Current Applied Physics*, 11 (3):740 – 745, 2011. ISSN 1567-1739. doi: https://doi.org/10.1016/j.cap.2010.11.051. URL `http://www.sciencedirect.com/science/article/pii/S1567173910004153`.

J.B. Kuipers. *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality*. Princeton University Press, 2020. ISBN 9780691211701. URL `https://books.google.pt/books?id=p\T1\textbackslash_3RDwAAQBAJ`.

Daniel Laidig, Philipp Müller, and Thomas Seel. Automatic anatomical calibration for imu-based elbow angle measurement in disturbed magnetic fields. *Current Directions in Biomedical Engineering*, 3(2):167 – 170, 09 2017. doi: https://doi.org/10.1515/cdbme-2017-0035. URL `https://www.degruyter.com/view/journals/cdbme/3/2/article-p167.xml`.

Dustin Lehmann, Daniel Laidig, Raphael Deimel, and Thomas Seel. Magnetometer-free inertial motion tracking of arbitrary joints with range of motion constraints. 02 2020.

G. Li, Y. Li, Z. Zhang, Y. Geng, and R. Zhou. Selection of sampling rate for emg pattern recognition based prosthesis control. pages 5058–5061, 2010. doi: 10.1109/IEMBS.2010.5626224.

Hwai-Ting Lin, Yasuo Nakamura, Fong-Chin Su, Jun Hashimoto, Katsuya Nobuhara, and Edmund Y. S. Chao. Use of Virtual, Interactive, Musculoskeletal System (VIMS) in Modeling and Analysis of Shoulder Throwing Activity. *Journal of Biomechanical Engineering*, 127(3):525–530, 01 2005. ISSN 0148-0731. doi: 10.1115/1.1894387. URL `https://doi.org/10.1115/1.1894387`.

Lars Lindstrom. Muscular fatigue and action potential conduction velocity changes studied with frequency analysis of emg signals. *Electromyography*, 10(4):341–356, 1970.

Sergey Lobov, N. Krylova, A. Anisimova, Vasily Mironov, and Victor Kazantsev. Optimizing the speed and accuracy of an emg interface in practical applications. *Human Physiology*, 45: 145–151, 03 2019. doi: 10.1134/S0362119719010109.

Blair Lock, Kevin Englehart, and Bernard Hudgins. Real-time myoelectric control in a virtual environment to relate usability vs. accuracy. *MyoElectric Controls/Powered Prosthetics Symposium, Fredericton*, 01 2005.

Lockheed Martin Corporation. Exoskeleton technologies, 2020. URL https://www.ottobockus.com/prosthetics/upper-limb-prosthetics/solution-overview/passive-arm-prostheses/.

Sebastian O.H. Madgwick. An efficient orientation filter for inertial and inertial/magnetic sensor arrays, 04 2010. URL http://www.x-io.co.uk/res/doc/madgwick_internal_report.pdf.

MBDyn. Mbdyn - homepage, 2020. URL https://www.mbdyn.org/?Homepage.

D.T. Mewett, Homayoun Nazeran, and Karen Reynolds. Removing power line noise from recorded emg. volume 3, pages 2190 – 2193 vol.3, 02 2001. ISBN 0-7803-7211-5. doi: 10.1109/IEMBS.2001.1017205.

Ross H. Miller. *Hill-Based Muscle Modeling*, pages 1–22. Springer International Publishing, Cham, 2018. ISBN 978-3-319-30808-1. doi: 10.1007/978-3-319-30808-1_203-2. URL https://doi.org/10.1007/978-3-319-30808-1_203-2.

Ani Mnatsakanian, John Kissel, Philip Terry, and Wendy King. One clinic's experience with carbon fiber orthoses in neuromuscular disease. *Muscle & Nerve*, 55, 06 2016. doi: 10.1002/mus.25233.

Danielle Moore. Electromyography (emg), 03 2018. URL https://www.healthline.com/health/emg.

P. Müller, M. Bégin, T. Schauer, and T. Seel. Alignment-free, self-calibrating elbow angles measurement using inertial sensors. *IEEE Journal of Biomedical and Health Informatics*, 21(2): 312–319, 2017. doi: 10.1109/JBHI.2016.2639537.

n.d. How bad is elbow flexion for a good serve?, 3 2011. URL https://tt.tennis-warehouse.com/index.php?threads/how-bad-is-elbow-flexion-for-a-good-serve.585977/.

Francesco Negro, Kevin Keenan, and Dario Farina. Power spectrum of the rectified EMG: when and why is rectification beneficial for identifying neural connectivity? *Journal of Neural Engineering*, 12(3):036008, 04 2015. doi: 10.1088/1741-2560/12/3/036008. URL https://doi.org/10.1088%2F1741-2560%2F12%2F3%2F036008.

Phuoc Nguyen. How the q-factor of a notch filter is determined?, 04 2017.

AC Nicol. A new flexible goniometer with widespread applications. *Biomechanics X-B*, pages 1029 – 1033, 1987. URL https://jglobal.jst.go.jp/en/detail?JGLOBAL_ID=200902029124405092.

Luis Fernando Nicolas-Alonso and Jaime Gomez-Gil. Brain computer interfaces, a review. *Sensors*, 12(2):1211–1279, 2012. ISSN 1424-8220. doi: 10.3390/s120201211. URL https://www.mdpi.com/1424-8220/12/2/1211.

Julia Nikulski. The ultimate guide to adaboost, random forests and xgboost, 2020. URL https://towardsdatascience.com/the-ultimate-guide-to-adaboost-random-forests-and-xgboost-7f9327061c4f.

Fernando Nogueira. Bayesian Optimization: Open source constrained global optimization tool for Python, 2014. URL https://github.com/fmfn/BayesianOptimization.

NVIDIA Corporation. Nvidia jetson nano developer kit, 2020. URL https://developer.nvidia.com/embedded/jetson-nano-developer-kit.

OpenSim. Opensim - home, 2020. URL https://opensim.stanford.edu/.

World Health Organization. The atlas of heart disease and stroke / judith mackay and george mensah ; with shanthi mendis and kurt greenland, 2004.

M. A. Oskoei and H. Hu. Support vector machine-based classification scheme for myoelectric control applied to upper limb. *IEEE Transactions on Biomedical Engineering*, 55(8):1956–1965, 2008. doi: 10.1109/TBME.2008.919734.

Leonard O'Sullivan, Rachel Nugent, and Johan van der Vorm. Standards for the safety of exoskeletons used by industrial workers performing manual handling activities: A contribution from the robo-mate project to their future development. *Procedia Manufacturing*, 3:1418 – 1425, 2015. ISSN 2351-9789. doi: https://doi.org/10.1016/j.promfg.2015.07.306. URL http://www.sciencedirect.com/science/article/pii/S2351978915003078. 6th International Conference on Applied Human Factors and Ergonomics (AHFE 2015) and the Affiliated Conferences, AHFE 2015.

ottobock. Passive arm prostheses, 2020. URL https://www.ottobockus.com/prosthetics/upper-limb-prosthetics/solution-overview/passive-arm-prostheses/.

Leonel Paredes-Madrid, Carlos A Palacio, Arnaldo Matute, and Carlos A Parra Vargas. Underlying physics of conductive polymer composites and force sensing resistors (fsrs) under static loading conditions. *Sensors (Basel, Switzerland)*, 17(9), 09 2017. ISSN 1424-8220. doi: 10.3390/s17092108. URL https://europepmc.org/articles/PMC5621037.

Jacob Pedersen. *Model Based and Robust Control Techniques for Internal Combustion Engine Throttle Valves*. PhD thesis, 12 2013.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Angkoon Phinyomark, Chusak Limsakul, and P. Phukpattaranont. Emg feature extraction for tolerance of white gaussian noise. 12 2008.

Angkoon Phinyomark, Pornchai Phukpattaranont, and Chusak Limsakul. Feature reduction and selection for emg signal classification. *Expert Systems with Applications*, 39(8):7420 – 7431, 2012. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2012.01.102. URL http://www.sciencedirect.com/science/article/pii/S0957417412001200.

Angkoon Phinyomark, Franck Quaine, Sylvie Charbonnier, Christine Serviere, Franck Tarpin-Bernard, and Yann Laurillau. Emg feature evaluation for improving myoelectric pattern recognition robustness. *Expert Systems with Applications*, 40(12):4832 – 4840, 2013. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2013.02.023. URL http://www.sciencedirect.com/science/article/pii/S0957417413001395.

Daniel Francisco Barros Marinho Esteves Sá Pina. *Development of a methodology to design a lower limb assistive exoskeleton*. PhD thesis, 07 2018. URL https://hdl.handle.net/10216/114414.

G. Piovan and F. Bullo. On coordinate-free rotation decomposition: Euler angles about arbitrary axes. *IEEE Transactions on Robotics*, 28(3):728–733, 2012. doi: 10.1109/TRO.2012.2184951.

Jacek Piskorowski. Time-efficient removal of power-line noise from emg signals using iir notch filters with non-zero initial conditions. *Biocybernetics and Biomedical Engineering*, 33(3):171 – 178, 2013. ISSN 0208-5216. doi: https://doi.org/10.1016/j.bbe.2013.07.006. URL http://www.sciencedirect.com/science/article/pii/S0208521613000211.

PLUX Wireless Biosignals S.A. Bitalino (r)evolution board kit, 2020a. URL https://plux.info/kits/33-bitalino-revolution-board-bt-810121001.html.

PLUX Wireless Biosignals S.A. Musclebit bt, 2020b. URL https://plux.info/bundles/426-bitalino-revolution-musclebit-bt.html.

Philipp Probst, Marvin N. Wright, and Anne-Laure Boulesteix. Hyperparameters and tuning strategies for random forest. *WIREs Data Mining and Knowledge Discovery*, 9(3):e1301, 2019. doi: https://doi.org/10.1002/widm.1301. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1301.

Radxa Limited. Rock pi 4, 2020. URL https://rockpi.org/rockpi4.

M. H. Rahman, M. J. Rahman, O. L. Cristobal, M. Saad, J. P. Kenné, and P. S. Archambault. Development of a whole arm wearable robotic exoskeleton for rehabilitation and to assist upper limb movements. *Robotica*, 33(1):19–39, 2015. doi: 10.1017/S0263574714000034.

Raspberry Pi Foundation. Raspberry pi 4 model b, 2020. URL https://www.raspberrypi.org/products/raspberry-pi-4-model-b/?resellerType=home.

RLS d. o. o. Rmk3b evaluation board with am8192b, 2021. URL https://www.rls.si/eng/rmk3b-evaluation-board.

Thomas Roberts and Annette Gabaldón. Interpreting muscle function from emg: Lessons learned from direct measurements of muscle force. *Integrative and comparative biology*, 48:312–20, 08 2008. doi: 10.1093/icb/icn056.

Everton Rocha, Débora Cantergi, Artur Bonezi, Denise Soares, Cláudia Candotti, and Jefferson Loss. Smoothing emg signals: Implications on delay calculation. *Revista Portuguesa de Ciências do Desporto*, 12:60, 01 2012. doi: 10.5628/rpcd.12.01.60.

ROHM Co., Ltd. Full-wave rectification and half-wave rectification, 2020. URL `https://www.rohm.com/electronics-basics/ac-dc/rectification`.

Keith Rome and Fiona Cowieson. A reliability study of the universal goniometer, fluid goniometer, and electrogoniometer for the measurement of ankle dorsiflexion. *Foot & ankle international / American Orthopaedic Foot and Ankle Society [and] Swiss Foot and Ankle Society*, 17:28–32, 02 1996. doi: 10.1177/107110079601700106.

Joy Roy, Md. Asraf Ali, Md. Razu Ahmed, and Kenneth Sundaraj. *Machine Learning Techniques for Predicting Surface EMG Activities on Upper Limb Muscle: A Systematic Review*, pages 330–339. 07 2020. ISBN 978-3-030-52855-3. doi: 10.1007/978-3-030-52856-0_26.

Mirza Muhammad Sabir and Junaid Ali Khan. Optimal design of pid controller for the speed control of dc motor by using metaheuristic techniques. *Advances in artificial neural systems*, 2014, 2014.

Dave Seaton. Brushed dc motor control: Parameter characterization, open loop and pi controller simulation. 2010.

SENIAM. Recommendations for sensor locations in arm or hand muscles, 2006a. URL `http://seniam.org/bicepsbrachii.html`.

SENIAM. Recommendations for sensor locations in arm or hand muscles, 2006b. URL `http://seniam.org/tricepsbrachiilonghead.html`.

B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016. doi: 10.1109/JPROC.2015.2494218.

M. D. SHUSTER and S. D. OH. Three-axis attitude determination from vector observations. *Journal of Guidance and Control*, 4(1):70–77, 1981. doi: 10.2514/3.19717. URL `https://doi.org/10.2514/3.19717`.

Sinovoip. Banana pi, 2020. URL `http://www.banana-pi.org/bpi-products.html`.

Siuly Siuly, Yan Li, and Yanchun Zhang. *Electroencephalogram (EEG) and Its Background*, pages 3–21. Springer International Publishing, Cham, 2016. ISBN 978-3-319-47653-7. doi: 10.1007/978-3-319-47653-7_1. URL `https://doi.org/10.1007/978-3-319-47653-7_1`.

L. H. Smith, L. J. Hargrove, B. A. Lock, and T. A. Kuiken. Determining the optimal window length for pattern recognition-based myoelectric control: Balancing the competing effects of classification error and controller delay. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 19(2):186–192, 2011. doi: 10.1109/TNSRE.2010.2100828.

Steven W. Smith. Chapter 19 - recursive filters. In Steven W. Smith, editor, *Digital Signal Processing*, pages 319 – 332. Newnes, Boston, 2003. ISBN 978-0-7506-7444-7. doi: https://doi.org/10.1016/B978-0-7506-7444-7/50056-X. URL `http://www.sciencedirect.com/science/article/pii/B978075067444750056X`.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL `https://proceedings.neurips.cc/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf`.

Oluwasegun Somefun, Kayode Akingbade, and Folasade Dahunsi. Speed control of dc motors: Optimal closed pid-loop model predictive control. *International Journal of Control and Automation*, 8:9–21, 03 2020. doi: 10.13189/ujca.2020.080102.

WL Soong. Pm machines: Parameter measurement and performance prediction. *Power Engineering Briefing Note Series*, 2008.

SparkFun Electronics®. Sparkfun 9dof imu breakout - icm-20948 (qwiic), 2020. URL `https://www.sparkfun.com/products/15335`.

Dr. Hermann J. Stadtfeld. The basics of gear theory, 6 2015. URL `https://www.geartechnology.com/articles/0615/The_Basics_of_Gear_Theory/`.

Dick Stegeman and Hermie Hermens. Standards for suface electromyography: The european project surface emg for non-invasive assessment of muscles (seniam). 1, 01 2007.

Nahema Sylla, Vincent Bonnet, Frédéric Colledani, and Philippe Fraisse. Ergonomic contribution of able exoskeleton in automotive industry. *International Journal of Industrial Ergonomics*, 44(4):475–481, 2014. ISSN 0169-8141. doi: https://doi.org/10.1016/j.ergon.2014.03.008. URL `https://www.sciencedirect.com/science/article/pii/S0169814114000833`.

Zhichuan Tang, Kejun Zhang, Shouqian Sun, Zenggui Gao, Lekai Zhang, and Zhongliang Yang. An upper-limb power-assist exoskeleton using proportional myoelectric control. *Sensors*, 14 (4):6677–6694, 2014.

L Tesio, M Monzani, R Gatti, and F Franchignoni. Flexible electrogoniometers: kinesiological advantages with respect to potentiometric goniometers. *Clinical Biomechanics*, 10(5):275 –

277, 1995. ISSN 0268-0033. doi: https://doi.org/10.1016/0268-0033(95)00017-F. URL http://www.sciencedirect.com/science/article/pii/026800339500017F.

Jean Theurel, Kevin Desbrosses, Terence Roux, and Adriana Savescu. Physiological consequences of using an upper limb exoskeleton during manual handling tasks. *Applied Ergonomics*, 67: 211–217, 02 2018. doi: 10.1016/j.apergo.2017.10.008.

Alan J. Thurston. ParÉ and prosthetics: The early history of artificial limbs. *ANZ Journal of Surgery*, 77(12):1114–1119, 2007. doi: 10.1111/j.1445-2197.2007.04330.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1445-2197.2007.04330.x.

Spyros G. Tzafestas. 12 - mobile robot localization and mapping. In Spyros G. Tzafestas, editor, *Introduction to Mobile Robot Control*, pages 479 – 531. Elsevier, Oxford, 2014. ISBN 978-0-12-417049-0. doi: https://doi.org/10.1016/B978-0-12-417049-0.00012-2. URL http://www.sciencedirect.com/science/article/pii/B9780124170490000122.

Brent L. Ulrey and Fadi A. Fathallah. Effect of a personal weight transfer device on muscle activities and joint flexions in the stooped posture. *Journal of Electromyography and Kinesiology*, 23(1):195 – 205, 2013a. ISSN 1050-6411. doi: https://doi.org/10.1016/j.jelekin.2012.08.014. URL http://www.sciencedirect.com/science/article/pii/S1050641112001502.

Brent L. Ulrey and Fadi A. Fathallah. Subject-specific, whole-body models of the stooped posture with a personal weight transfer device. *Journal of Electromyography and Kinesiology*, 23(1): 206 – 215, 2013b. ISSN 1050-6411. doi: https://doi.org/10.1016/j.jelekin.2012.08.016. URL http://www.sciencedirect.com/science/article/pii/S1050641112001526.

Vance J. Vandoren. Loop tuning fundamentals, 07 2003. URL https://www.controleng.com/articles/loop-tuning-fundamentals/.

Vance J. Vandoren. Auto-tuning control using ziegler-nichols, 10 2006. URL https://www.controleng.com/articles/auto-tuning-control-using-ziegler-nichols/.

Antanas Verikas, Evaldas Vaiciukynas, Adas Gelzinis, James Parker, and M. Charlotte Olsson. Electromyographic patterns during golf swing: Activation sequence profiling and prediction of shot effectiveness. *Sensors*, 16:592, 04 2016. doi: 10.3390/s16040592.

André Vieira. Clinical reabilitation of upper limb in chronic stroke in portugal-a cross sectional survey. *International Journal of Physiotherapy*, 3, 02 2016. doi: 10.15621/ijphy/2016/v3i1/88927.

Pratap Vikhe, Neelam Punjabi, and C B Kadu. Real time dc motor speed control using pid controller in labview. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 10 2014.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Nicola Vitiello, Tommaso Lenzi, S. Roccella, Stefano Marco Maria De Rossi, Emanuele Cattin, Francesco Giovacchini, Fabrizio Vecchi, and Maria Chiara Carrozza. Neuroexos: A powered elbow exoskeleton for physical rehabilitation. *IEEE Transactions on Robotics*, 29:220–235, 02 2013. doi: 10.1109/TRO.2012.2211492.

Brett H. Whitfield, Patrick A. Costigan, Joan M. Stevenson, and Catherine L. Smallman. Effect of an on-body ergonomic aid on oxygen consumption during a repetitive lifting task. *International Journal of Industrial Ergonomics*, 44(1):39 – 44, 2014. ISSN 0169-8141. doi: https://doi.org/10.1016/j.ergon.2013.10.002. URL `http://www.sciencedirect.com/science/article/pii/S0169814113001121`.

Rian Whitton. An exciting future for exoskeletons, 05 2019. URL `https://www.abiresearch.com/blogs/2019/05/13/exciting-future-exoskeletons/`.

Dorsey Williams and Lee Welch. Male and female runners demonstrate different sagittal plane mechanics as a function of static hamstring flexibility. *Brazilian Journal of Physical Therapy*, 19, 10 2015. doi: 10.1590/bjpt-rbf.2014.0123.

Ge Wu, Frans van der Helm, Dirkjan Veeger, Mohsen Makhsous, Peter Roy, Carolyn Anglin, Jochem Nagels, Andrew Karduna, Kevin McQuade, Xuguang Wang, Frederick Werner, and Bryan Buchholz. Isb recommendation on definitions of joint coordinate systems of various joints for the reporting of human joint motion - part ii: Shoulder, elbow, wrist and hand. *Journal of biomechanics*, 38:981–992, 06 2005. doi: 10.1016/j.jbiomech.2004.05.042.

Feiyun Xiao. Proportional myoelectric and compensating control of a cable-conduit mechanism-driven upper limb exoskeleton. *ISA Transactions*, 89:245 – 255, 2019. ISSN 0019-0578. doi: https://doi.org/10.1016/j.isatra.2018.12.028. URL `http://www.sciencedirect.com/science/article/pii/S0019057818305238`.

Feiyun Xiao, Yong Wang, Yongsheng Gao, Yanhe Zhu, and Jie Zhao. Continuous estimation of joint angle from electromyography using multiple time-delayed features and random forests. *Biomedical Signal Processing and Control*, 39:303 – 311, 2018a. ISSN 1746-8094. doi: https://doi.org/10.1016/j.bspc.2017.08.015. URL `http://www.sciencedirect.com/science/article/pii/S1746809417301775`.

Feiyun Xiao, Gao Yongsheng, Yong Wang, Yanhe Zhu, and Jie Zhao. Design and evaluation of a 7-dof cable-driven upper limb exoskeleton. *Journal of Mechanical Science and Technology*, 32:855–864, 02 2018b. doi: 10.1007/s12206-018-0136-y.

Massimiliano Zecca, Silvestro Micera, M.C. Carrozza, and Paolo Dario. Control of multifunctional prosthetic hands by processing the electromyographic signal. *Critical reviews in biomedical engineering*, 30:459–85, 02 2002. doi: 10.1615/CritRevBiomedEng.v30.i456.80.

Julian Zeitlhöfler. *Nominal and observation-based attitude realization for precise orbit determination of the Jason satellites*. PhD thesis, 06 2019.

Xu Zhang and Ping Zhou. Sample entropy analysis of surface emg for improved muscle activity onset detection against spurious background spikes. *Journal of Electromyography and Kinesiology*, 22(6):901 – 907, 2012. ISSN 1050-6411. doi: https://doi.org/10.1016/j.jelekin.2012.06.005. URL `http://www.sciencedirect.com/science/article/pii/S1050641112001150`.

Zebo Zhou, Jin Wu, Jinling Wang, and Hassen Fourati. Optimal, recursive and sub-optimal linear solutions to attitude determination from vector observations for gnss/accelerometer/magnetometer orientation measurement. *Remote Sensing*, 10, 02 2018. doi: 10.3390/rs10030377.

Anexo A

DC Motor Model and Experimental Characterization of its Parameters

Finding the DC motor's transfer function is an instrumental step in achieving fine control of the DC motor, when the control action is model-based. Over the next few pages, the process of modelling the DC motor and empirically determining its parameters is comprehensively detailed.

A.1 DC Motor Model

Producing the correct output is dependent on the quality of the design of the controller, and properly designing a DC motor controller first requires a ground level understanding of this actuator.

A DC motor can be represented by its equivalent circuit, that is, an electrical circuit that mimics the electrical behaviour of a DC motor when it is connected to an electrical power supply. An illustration of its equivalent circuit can be seen in Figure A.1, where E_a is the voltage at the motor's terminals, R_a and L_a are the motor's equivalent series resistance and inductance, and i_a is the current being ran through the motor. The index a refers to the motor's armature, which composes the overwhelming brunt of the motor's electrical circuitry. E_b is the motor's back EMF voltage (back EMF is the the counter electromotive force that opposes the change in current that induced it). As illustrated, the rotor and the load are rotating at an angular velocity and acceleration of $\dot{\theta}$ and $\ddot{\theta}$, respectively, with an associated total inertia of J.

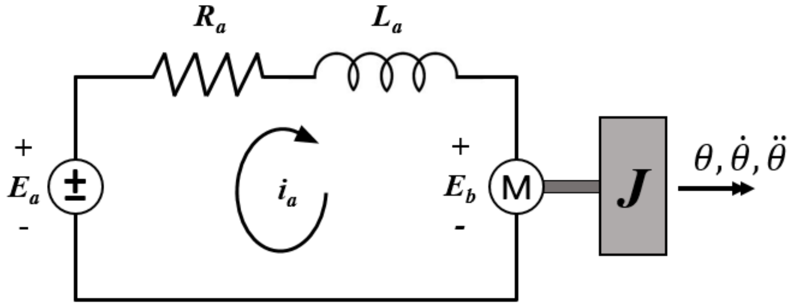

Figure A.1: Equivalent circuit of the DC motor. A voltage E_a applied at the motor's terminals induces a current through the machine's armature, causing the inertial mass J to move.

Applying Kirchoff's voltage law, the following equality arises:

$$e_a(t) = i_a(t)\,R_a + L_a \frac{di_a(t)}{dt} + e_b(t) \tag{A.1}$$

where the difference of potential across the armature inductance depends on the first-order derivative of the current and the back EMF voltage depends on the loading that the motor is being subjected to and can be given by (Somefun et al., 2020):

$$e_b(t) = K_b\,\dot{\theta}_m(t) \tag{A.2}$$

where K_b is the motor's back EMF constant and $\dot{\theta}_m(t)$ is the motor's angular velocity at time t.

The motor's rotor has the governing differential equation of motion (Somefun et al., 2020):

$$J\,\ddot{\theta}_m(t) + c\,\dot{\theta}_m(t) = \tau_m(t) \tag{A.3}$$

where J is the system's moment of inertia, c is the motor's friction constant and $\tau_m(t)$ is the load being applied to the motor at time t.

The two aforestated equations govern the DC motor's dynamics. Controlling the DC motor is equivalent to manipulating these dynamics in real time for speed or position control. This can be achieved in a multitude of ways, such as basic supply voltage regulation, armature current control and field current control, to name a few (Ali et al., 2019).

For armature current control, the torque developed by the motor is considered linearly dependent on the armature current, in such a way that:

$$\tau_m(t) = K_m\,i_a(t) \tag{A.4}$$

where K_m is the motor torque constant.

This DC motor model ignores certain nonlinearities like backlash and dead zones. This is appropriate given the relatively minute contribution some of these dynamics have in the motor's transfer function (Sabir and Khan, 2014). With this new relationship, the motor's open-loop transfer function can be found. Applying the Laplace transform to Equation A.5 results in:

$$\begin{aligned} E_a(s) &= I_a(s)\,R_a + L_a s \cdot I_a(s) + K_b\,s \cdot \Theta_m(s) = \\ &= I_a(s)\,(R_a + L_a\,s) + K_b\,s \cdot \Theta_m(s) \end{aligned} \tag{A.5}$$

The motor's open-loop tranfer function $G(s)$, as determined by the output angular velocity divided by the terminal voltage, is given by:

$$\begin{aligned} G(s) = \frac{\Omega_m(s)}{E_a(s)} &= \frac{\Omega_m(s)}{I_a(s)\,(R_a + L_a\,s) + K_b\,s \cdot \Theta_m(s)} = \\ &= \frac{s \cdot \Theta_m(s)}{I_a(s)\,(R_a + L_a\,s) + K_b\,s \cdot \Theta_m(s)} \end{aligned} \tag{A.6}$$

where $s \cdot \Theta_m(s)$ can be substituted in from Equation A.3:

$$
\begin{aligned}
G(s) &= \frac{\dfrac{K_m I_a(s)}{J\,s+c}}{I_a(s)\,(R_a+L_a\,s)+K_b\,\dfrac{K_m I_a(s)}{J\,s+c}} = \\
&= \frac{K_m}{(R_a+L_a\,s)(J\,s+c)+K_b\,K_m} = \\
&= \frac{K_m}{L_a\,J\,s^2+(R_a\,J+L_a\,c)\,s+(R_a\,c+K_m^2)}
\end{aligned}
\tag{A.7}
$$

where the back EMF and torque constants, K_b and K_m are equal provided there is conservation of energy (Ali et al., 2019). In reality, there are many losses in a motor, which cause the two constants to be different, though this nuance will be deemed negligible here.

A.2 Experimental Motor Parameter Characterization

To compute the transfer function, now all that is required is the motor's parameters. Lamentably, as mentioned in Section 3.3.5, the parameters for the selected DC motor are completely unknown. Thankfully, they can be estimated with experimental procedures.

The armature resistance R_a can be found in two ways. The first and simplest is to directly measure, using a multimeter, the motor's resistance at its terminals, when it is completely disconnected from power. The multimeter should also be checked for any resistance offset, by connecting both ends of the multimeter together (Soong, 2008).

The second method, regarded as more accurate (Soong, 2008), is to supply the motor with a known current and measure the voltage drop, at its terminals, when the rotor is forced into a stall. A representation of the method is shown in Figure A.2. The current should be as close as possible to the motor's nominal working current (which implies that knowledge of this information already exists, nonexistent here) and the test should be performed as quickly as possible, to avoid heating (which invariably influences the measurement). The armature resistance can then be determined using Ohm's law. It is suggested that, when measuring the armature resistance, the test should be performed multiple times for different combinations of voltage and current (Seaton, 2010).

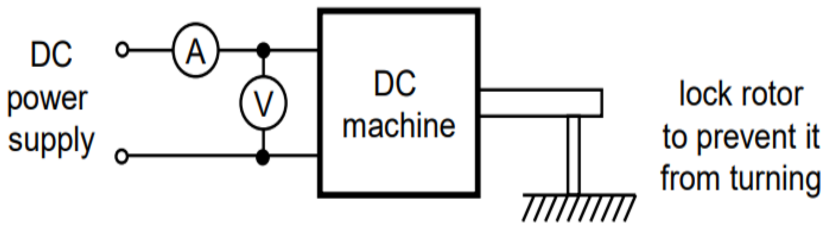

Figure A.2: Empirical method for determination of a motor's armature resistance. The motor is fed a known pair of voltage and current, while its shaft is brought to a stall. Source: Soong (2008).

Both methods were performed and the results can be found in Tables A.1 and A.2.

Finding the motor's series inductance L_a can be extremely simple, too, if an LCR meter is available (Seaton, 2010). Then, measuring the series inductance at the motor's terminals will promptly give the intended value. Results are shown in Table A.3.

To determine the motor torque constant K_m, the current and voltage have to be tracked under no-load conditions, as well as the rotor's speed (sensor feedback is required). Then, knowing the

Table A.1: Experimental results of the first method for determining the DC motor's armature resistance R_a.

Trial	Multimeter Offset Δ [Ω]	Measurement R [Ω]
1	0.2	18.0
2	0.2	17.9
3	0.3	17.8
Mean	0.233	17.9
Armature Resistance $R_a = \bar{R} - \bar{\Delta}$ [Ω]		17.7

armature resistance R_a, the motor's speed-voltage relationship allows computation of K_m (Seaton, 2010):

$$w = \frac{E_a - E_b}{K_m} = \frac{E_a - i_a\,R_a}{K_m} K_m = \frac{E_a - i_a\,R_a}{w} \tag{A.8}$$

where, once again, E_a and i_a are the motor's terminal voltage and current, respectively, E_b is the back EMF voltage and w is the rotor's angular velocity.

This test was performed for a range of values of voltage and current, with results visible in Table A.4.

Three parameters remain: K_b, J and c. We already established that $K_b = K_m = 0.036$ [V·s/rad].

The values of the inertia and friction coefficient are difficult to determine - they depend on working and loading conditions. Not only that, but their magnitudes, when compared to the previously determined parameters of the motor's transfer function, are rather small. In addition, in an *in vivo* implementation of the system, they could vary greatly, depending on the application the user would be doing.

The system's inertia was estimated based on the considerations of Section 3.3.5, such that:

$$J = J_E^E + m_E (r_{OE})^2 + m_F \left[(r_{G,F})^2 + (r_{OF})^2\right] + m_H \left[(r_{G,H})^2 + (r_{OH})^2\right] \tag{A.9}$$

where r_{OP} is the distance from point **O** (the center of the elbow) to any point **P**, m_X is the mass

Table A.2: Experimental results of the second method for determining the DC motor's armature resistance R_a.

Armature Resistance $R_a = \frac{V_{drop}}{I_S}$ [Ω]		Supply Voltage V_S [V] 4	5	6
	0.100	$\frac{1.03}{0.097} = 10.62$	$\frac{0.91}{0.100} = 9.10$	$\frac{0.97}{0.098} = 9.90$
Supply Current	0.150	$\frac{1.35}{0.149} = 9.06$	$\frac{1.43}{0.148} = 9.66$	$\frac{1.43}{0.149} = 10.27$
I_S [A]	0.200	$\frac{1.93}{0.199} = 9.70$	$\frac{1.86}{0.200} = 9.30$	$\frac{2.23}{0.202} = 11.04$
	0.250	$\frac{2.45}{0.249} = 9.84$	$\frac{2.54}{0.250} = 10.16$	$\frac{2.61}{0.249} = 10.48$

Mean Armature Resistance $\bar{R}_a = 9.93$ [Ω]

Table A.3: Experimental LCR measurements of the DC motor's series inductance L_a.

Trial	Series Inductance Measurement L_a $[mH]$
1	3.440
2	3.436
3	3.423
Mean	3.433

of body X and J_X^P is body X's net torque about point **P**. In this case, **E** denotes the Exoskeleton, **F** denotes the Forearm and **H** the Hand.

The Forearm Piece of the 3D printed device (refer to Appendix C) can be approximated to a thin rectangular plate (illustrated in Figure A.3) with a weight of approximately 27 [g] and dimensions of . Its inertia can thus be given by:

$$J_E^E = \frac{m_E}{12}(a^2+b^2) = \frac{0.027}{12}(0.235^2+0.030^2) = 1.26\times10^{-4}\ [kg\cdot m^2]$$

For a 65 [kg] man with a forearm length of 25 [cm] and a hand length of 19 [cm], the system has a total inertia of:

$$J = 1.26\times10^{-4} + 0.027\left(\frac{0.235}{2}\right)^2 + 1.053\left[(0.069)^2+(0.114)^2\right] + \\ +0.397\left[(0.100)^2+(0.235+0.150)^2\right] = 0.0820\ [kg\cdot m^2] \tag{A.10}$$

When it comes to the friction coefficient, with knowledge of the other parameters, considering the amount of uncertainty already surrounding the DC motor being used, it is recommended to simply resort to the parameters of a similarly-sized motor. For a similarly sized 12 [V] motor, Seaton (2010) recommends $c = 0.00724$ [N·m·s / rad], and as such a value of $c = 0.007$ [N·m·s / rad] will be considered.

At last, the resulting DC motor transfer function is:

Table A.4: Experimental results of motor constant characterization.

Supply Voltage	Current Drawn	Rotor Angular Velocity		Motor Constant
[V]	[A]	[rad/s]	[rpm]	K_m [V·s/rad]
5.00	0.150	82.1	784	0.043
5.00	0.153	83.5	797	0.042
7.49	0.182	179.0	1709	0.032
7.49	0.193	176.8	1688	0.032
9.99	0.244	223.6	2135	0.034
9.98	0.242	217.1	2073	0.035
			Mean	**0.036**

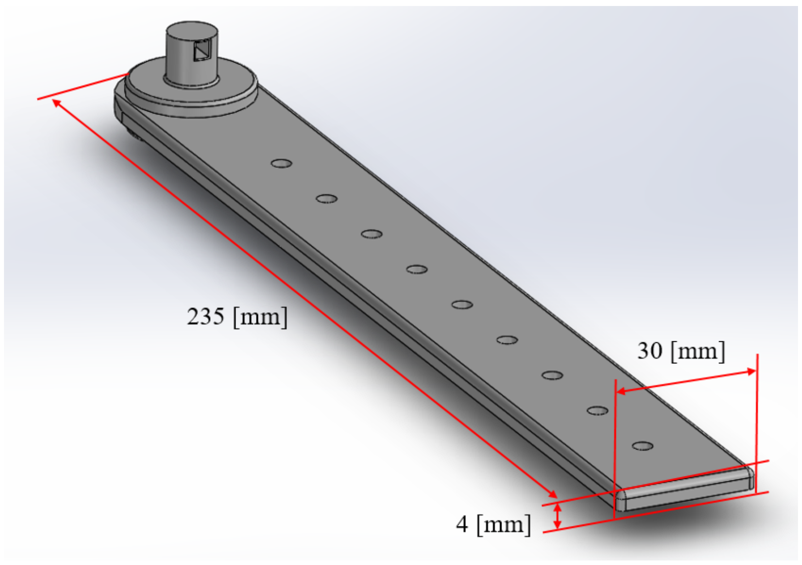

Figure A.3: Dimensions of the Forearm Piece of the 3D printed device.

$$
\begin{aligned}
G(s) &= \frac{K_m}{L_a J\, s^2 + (R_a J + L_a c)\, s + (R_a c + K_m^2)} \\
&= \frac{0.036}{0.003433 \cdot 0.0820\, s^2 + (9.93 \cdot 0.0820 + 0.003433 \cdot 0.007)\, s + (9.93 \cdot 0.007 + 0.036^2)}
\end{aligned}
$$

which can be simplified, by approximation, to:

$$
G(s) = \frac{0.036}{2.788 \times 10^{-4}\, s^2 + 0.8143\, s + 0.0708} \tag{A.11}
$$

Anexo B

Gear Transmission with Speed Reduction: Gear Train and Gear Geometry Specification

The robotic device's gear transmission constitutes a power transmission system with speed reduction, the latter being required due to the high nominal speeds and low nominal torques that the DC motor is able to produce.

B.1 Gear Train

The first step in designing the gear transmission is defining exactly what the total speed reduction ratios are, and in what way they the speed reduction function is distributed throughout the gear train.

The objective was to build a gear train with a speed reduction of at least 30:1. Next, each gear reduction step had to be established.

A problem arose in this very step - the selected DC motor already had a helical pinion attached to its shaft. Besides forcibly removing the motor's pinion, two options remained: fitting a new helical gear to the pinion or preserving the existing gear pair. On one hand, helical gears, in light of their greater geometric complexity, are harder to characterize empirically than spur gears. On the other hand, preserving the existing gear pair would mean being forced to incorporate their respective gear reduction step into the gear train. In addition, the gear in question has a second pinion directly attached to it, so fitting a gear to it would still be necessary.

The motor's pinion has 16 teeth and the associated gear has 52 teeth. This constitutes a speed reduction step of 3.25:1. Being rather large, and considering the added complexity to fitting a new helical gear to this pinion, the decision was to incorporate the pair into the gear train.

To achieve a speed reduction of roughly 30, given the initial step's reduction ratio, the resulting reduction is $30/3.25 \simeq 9.23$. This is very close to 9, a perfect square number, with squared root 3. Thus, it is only logical to distribute the remaining speed reduction across two steps of about the same magnitude of speed reduction. The pinion that is physically connected to the aforementioned 52 teeth gear has 39 teeth. A gear reduction of exactly 3:1 would imply a matching gear of 117 teeth.

The remaining speed reduction would then be $9.23/3 \simeq 3.08$. To make the gearbox as compact as possible, the next pinion was given 15 teeth and the gear 45 teeth, making for a 3:1 reduction.

In sum, the complete gear train reduction is 3.25:1, then 3:1, then 3:1, for a total speed reduction of 29.25:1. The gear train is illustrated in Figure B.1.

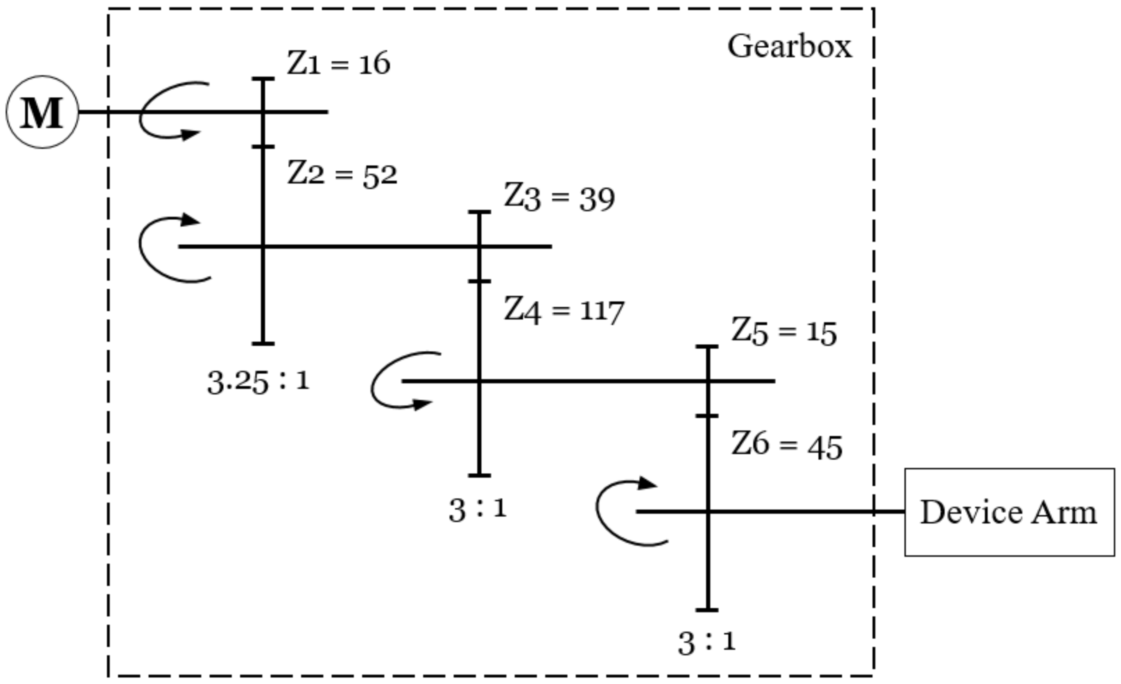

Figure B.1: Complete gear train for the gearbox. The three speed reduction steps amount to a total speed reduction of 29.25:1.

B.2 Gear Geometry

Once the gear train is delineated, the ensuing step is to compute and define each gear's geometric parameters. The most important proportions of a spur gear are visible in Figure B.2.

No gears (with the exception of the first pair, already present) were designed with helical geometry. This significantly eases the machine design of the gearbox, and the absence of tangential transmission in the gears is to a large extent insignificant, considering the exiguous demands of the mechanical loading.

Accordingly, the gear parameters that had to be defined were the number of teeth z (already known), the module m, the pressure angle α, the profile shift coefficient x, the tooth tip clearance coefficient c, the contact backlash j_{bn} and the gear thickness b. Most of these parameters are designed into the gear through a reference rack profile Figure B.3.

Selecting these parameters can be rather trivial, though in this case, as was laid out in the foregoing section, the pinion for the second gear pair was kept from the DC motor's original application. Therefore, before further ado, the existing pinion had to be experimentally characterized.

There are numerous ways for finding a gear's geometric parameters, each with its own advantages. The author opted for a procedure based on span (base tangent length W_k) and addendum diameter, d_a, measurements.

Mean measurements of the foregoing geometric quantities, for both the pinion and its matching gear (from the pinion's original application) are displayed in Table B.1.

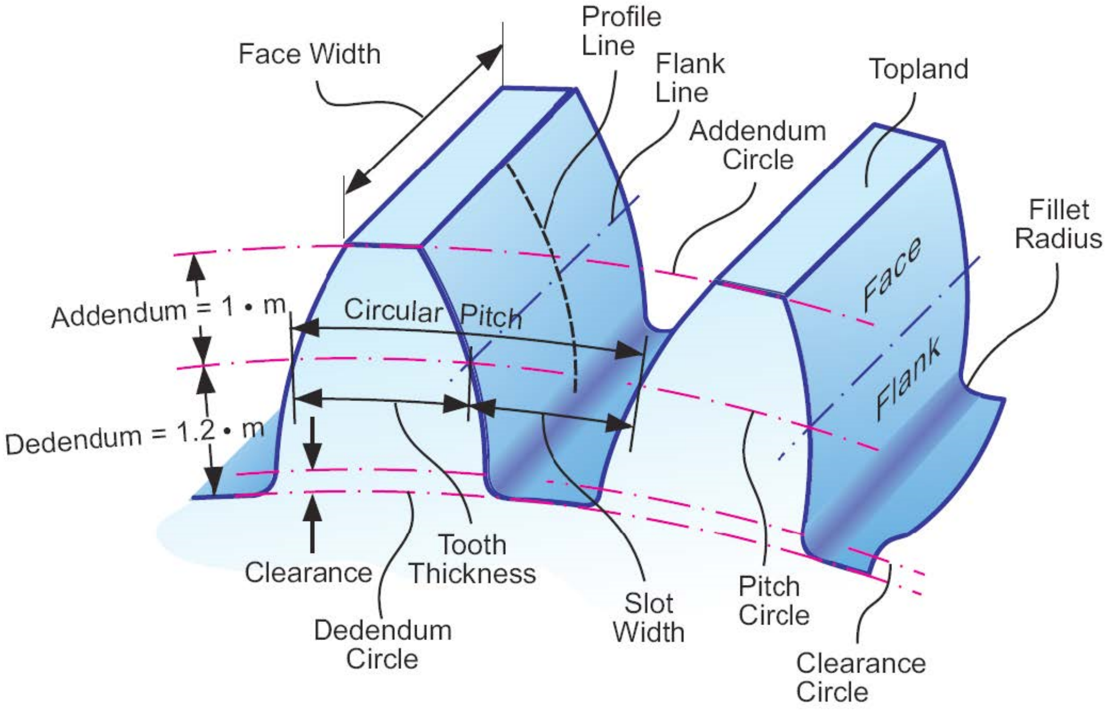

Figure B.2: The most important proportions of a standard spur gear. Nomenclature can vary based on standard. The Figure specifies the addendum and dedendum heights relative to the module m, but it should be warned that these values are rarely true. Not only can gear tooth geometry standards vary, their geometric parameters are often adjusted to improve on certain features of the gear, with the addendum and dedendum circles being frequent targets of such changes. Source: Stadtfeld (2015).

If the pinion was constructed in a standard way and without profile shift, then its addendum diameter should be that of a standard gear, and thus related to the pitch circle's diameter d by:

$$d_a = d + 2m \tag{B.1}$$

where d can be computed from the number of teeth and the module:

$$d = zm \tag{B.2}$$

From the addendum diameter the module can then be estimated with a high degree of confidence by combining the last two equations:

Table B.1: Mean measurements of the pinion and its mating gear. k was 5 for the pinion and 12 for the gear, based on the recommendations of Dr. Gonzalo Gonzalez Rey (2016).

	Nº of Teeth	Addendum Diameter	Base Tangent Length for k Teeth [mm]		
	z	$\bar{d}$ [mm]	$\bar{W}_{k-1}$	$\bar{W}_k$	$\bar{W}_{k+1}$
Pinion	39	16.50	6.70	5.50	4.29
Mating Gear	108	43.50	15.33	14.12	12.89

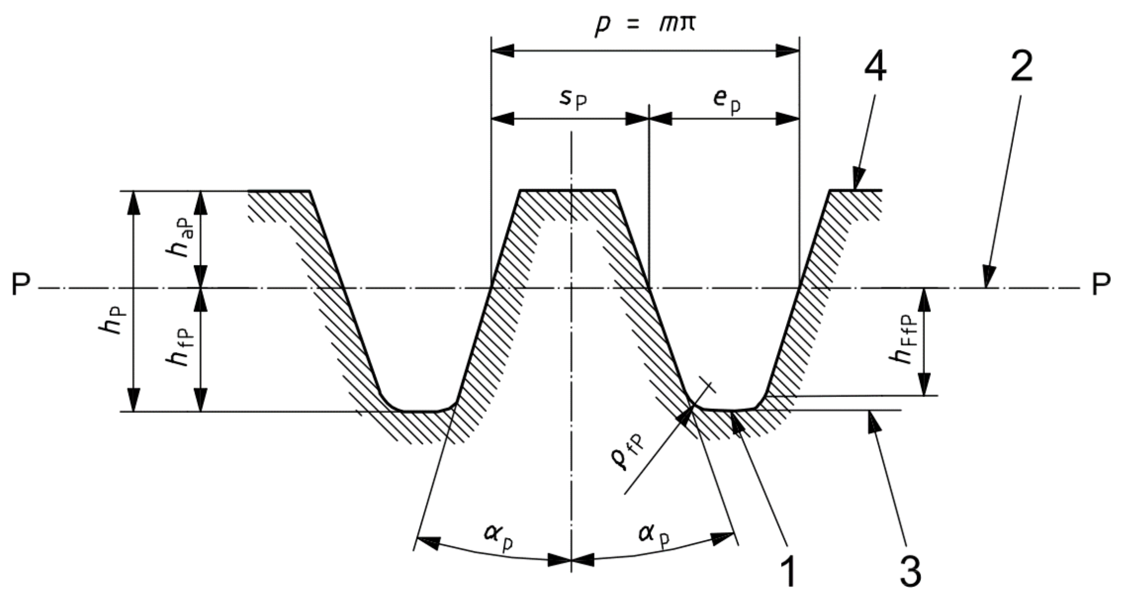

Figure B.3: Basic rack tooth profile for involute gear teeth, where p is the pitch on the reference cylinder, m is the module, s_P is the tooth thickness, e_P is the space width, α_P is the pressure angle, ρ_{fP} is the root radius, h_P is the tooth depth, h_{aP} is the addendum height, h_{fP} is the dedendum height and h_{FfP} is the depth of dedendum form. The subscript P specifies quantities of the standard basic rack tooth profile. 1 is the basic rack profile, 2 is the datum line, 3 is the root line and 4 is the tip line. Source: ISO 21771:2007 (E).

$$d_a = zm + 2m = (z+2)m \Rightarrow m = \frac{d_a}{z+2} \tag{B.3}$$

The results were in accordance with each other: $m = 0.402$ [mm] for the pinion and $m = 0.395$ [mm] for the gear, from which it can be confidently asserted that the module must be 0.4 [mm].

The next step is to estimate the pressure angle α. This can be reliably estimated with knowledge of the base tangent length W_k for consecutive values of k. Dr. Gonzalo Gonzalez Rey (2016) recommends $k = 5$ for spur gears of 36 to 44 teeth and $k = 12$ for spur gears of 100 to 110 teeth. Then, α can be estimated using (ISO 21771:2007, E):

$$W_{k+1} - W_k = \pi m \cos\alpha \tag{B.4}$$

Results are shown in Table B.2. A pressure angle of 16.5 degrees was selected going forward.

Table B.2: Estimated pressure angles for the existing pinion and gear. The mean difference is the consecutive average of two span measurement differences. Once estimated, the pressure angle was established to be 16.5 degrees.

	Mean Difference [mm]	Computed Pressure Angle [deg]
	$\Delta\bar{W}_{k,k+1} = \frac{\Delta W_{k+1} + \Delta W_k}{2} = \frac{W_{k+1} - W_{k-1}}{2}$	$\alpha = \arccos\left(\frac{\Delta\bar{W}_k}{\pi m}\right)$
Pinion	1.205	16.48
Mating Gear	1.217	14.49

Finally, for a gear that has a non-zero profile shift, the base tangent length is known to be given by (ISO 21771:2007, E):

$$W_{k,\text{ with profile shift}} = W_{k,\text{ without profile shift}} + 2x\, m_n \sin\alpha_n \tag{B.5}$$
$$= m_n \cos\alpha_n \left[\pi(k-0.5) + z\,\text{inv}\alpha_t\right] + 2x\, m_n \sin\alpha_n \tag{B.6}$$

where $m_n = m$ and $\alpha_n = \alpha$, given that the gear's helix angle β is zero (it is a spur gear).

With this relationship, the selected values of m and α can be compared to the span measurements and a profile shift coefficient x can be estimated. Results are shown in Table B.3.

Table B.3: Estimated profile shift for the existing pinion and gear.

	Module	Pressure Angle	Base Tangent Length, W_k [mm]		Profile Shift
	m [mm]	α [deg]	Theory	Measured	x
Pinion	0.4	16.5	5.55	5.5	-0.127
Matching Gear	0.4	16.5	14.20	14.1	-0.207

Both pinion and gear were estimated to have negative profile shift coefficients, at $x_{pinion} = -0.127$ and $x_{gear} = -0.207$, which comes as a moderate surprise. For instance, if the goal were to balance the gear's specific sliding, the two values should be symmetric. Excluding the obvious existence of systematic and random measurement errors, there is also a good chance that the span measurements were affected by other design changes to the gears' geometries, that caused them to depart from the absolute standard. Backlash compensation and imposition of working center distance are just two of many possible design concerns that could have affected their tooth profiles.

At last, the two remaining gears could be specified. A little prudence is advised. Parameters that regulate the gear's size should be kept to a minimum. This goes in line with the effort to make the gearbox as compact as possible. The main parameters that impact the gear's volume are the gear thickness, its module and the number of teeth.

On the other hand, it is advisable to increase the module, relative to that of the previous gears. An increase in transmission torque that comes as a reflection of the speed reduction (for a somewhat constant power transmission throughout the gear train) places greater mechanical demands on the gears - if their module is too small, their teeth may not be able to endure the increasing loads.

Moreover, the gears, much like the gearbox (refer to Appendix C), were designed for 3D printing. The printer has physical limitations insofar as the size of the geometries it can manufacture. Early prototypes of the gears helped establish that the aforestated module of 0.4 [mm] is dangerously close to the absolute limit of what the printer could reliably produce. Consequently, for the last gear pair, a module of 0.7 [mm] was set. Given the minute scale of the components involved, what might seem like a modest increase in module is, in fact, a 75% increase in gear geometry size, which should be more than enough for handling the paltry torques at play.

Unlike the other gears, a more standard pressure angle of 20 degrees was chosen.

Lastly, the gears' profiles were shifted in an effort to impost a center distance of 21.3 [mm]. The resulting profile shift coefficients were $x_{pinion} = 0.488$ and $x_{gear} = -0.037$.

To put it concisely, all relevant geometric parameters of all elements of the gear train are presented in Table B.4.

Table B.4: Relevant geometric parameters of all elements of the gear train. For the first pinion and gear (both helical), only the module was estimated.

Gear Nº	**1**	**2**	**3**	**4**	**5**	**6**
Nº of Teeth z	16	52	39	117	15	45
Module m [mm]	0.45 *(estimated)*	0.5 *(estimated)*	0.4 *(estimated)*	0.4	0.7	0.7
Pressure Angle α [deg]	-	-	16.5	16.5	20	20
Profile Shift Coefficient x	-	-	-0.127	-0.290	0.489	-0.037
Backlash Coefficient j_{bn}	-	-	0.04	0.05	0.04	0.04
Clearance Coefficient c/m	-	-	0.25	0.5	0.5	0.5

Anexo C

Design and Construction of the Assistive Device

This device is composed of four basic structural elements and additional transmission components (gears, shafts and a key). A complete overview of all the 3D printed components can be seen in Figure C.1 and an exploded view is shown in Figure C.2.

The joint's range of motion consists of 145º of rotation, all the way from a fully extended arm (0º of flexion) to 145º of flexion.

Gears and shafts were printed as one solid piece whenever possible, to ensure maximum mechanical properties, which could be an issue given the printing material (polymer).

All components were 3D printed in generic PLA. Most structural components were printed with relatively low densities of internal infill (around 15%), whereas functional pieces, such as gears, shafts and key, were printed with higher densities of internal infill (around 25%).

Most edges were filleted to prevent sharp edges from coming into contact with users. In addition, certain cares were necessary due to the manufacturing process. Volume contractions and expansions due to the stark temperature gradients - the printer's nozzle operated at temperatures of up to 215 [ºC], contrasting greatly with the environmental temperatures of as low as 10 [ºC]. Such low temperatures, caused by a harsh ongoing winter and the necessity to maintain windows open in the context of a COVID-19 lockdown, were occasionally felt despite efforts to externally heat the printing area.

To counteract this, the models were adjusted, in an effort to reduce sharp angles and acute geometric features, which greatly improved the prints' quality. This was not put in practice whenever it undermined adhesion to the printing surface. In defiance of all these efforts, certain components were still printed with slight warping. Fortunately, these deformations did not impair the functionality of the system.

Other 3D printing specific changes were the separation of certain large, uneven components into smaller parts (shown in Figure C.3), which helped reduce unnecessary consumption of printing material and save time.

In addition, a small structure was included on the outer side of the joint, to mount the encoder board, which is illustrated in Figure C.4.

Finally, extruded cuts were placed for the holes for the screws that hold the entire structure together.

After 3D printing, some components had to be slightly amended, to make up for printing inaccuracies (most of them caused by the thermal expansion/contraction of the PLA material) and occasional oversights. This was particularly important in adjacent components that move relative

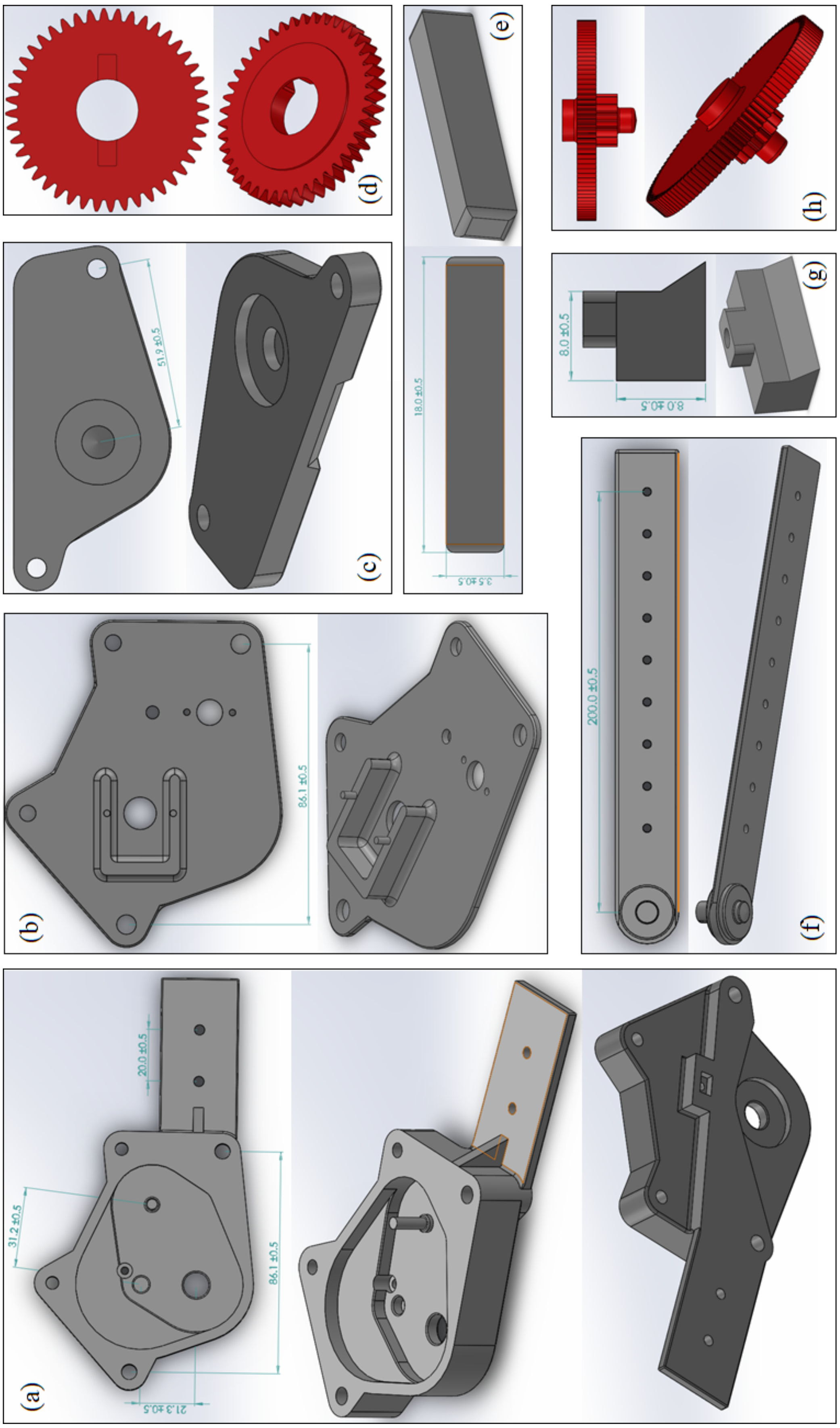

Figure C.1: All 3D printed components of the assistive device. (a) The Gearbox Piece, (b) the Top Piece, (c) the Bottom Piece, (d) the Gear 6 Piece, (e) the Key Piece, (f) the Forearm Piece, (g) the Foot Piece, (h) the Gears 4-5 Piece.

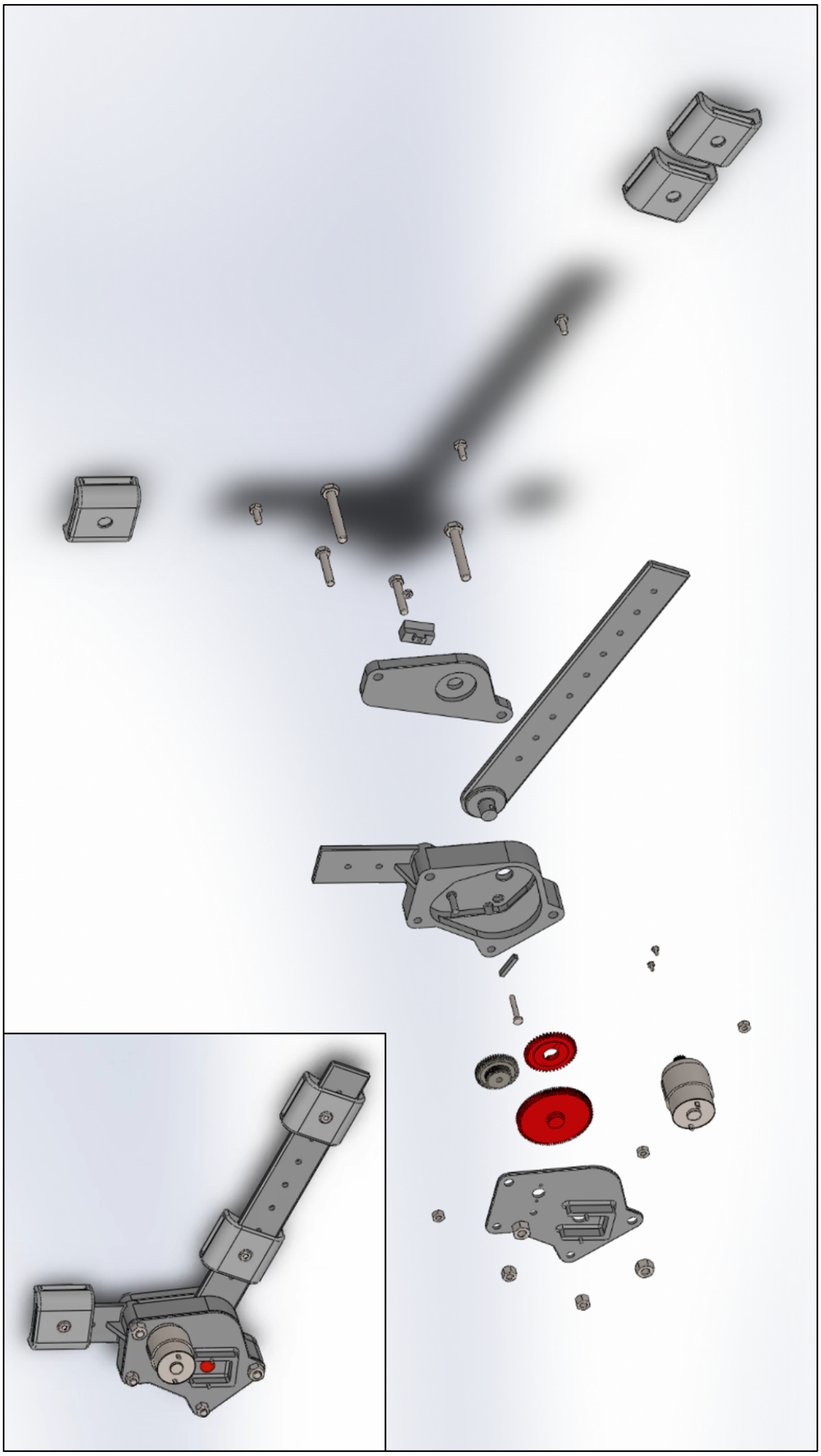

Figure C.2: Exploded and (top left) collapsed views of the 3D printed device. In addition to the components of Figure C.1, the DC motor, the Gears 2-3 Piece and buckles for in-person fixation, as well as screws and nuts can be seen.

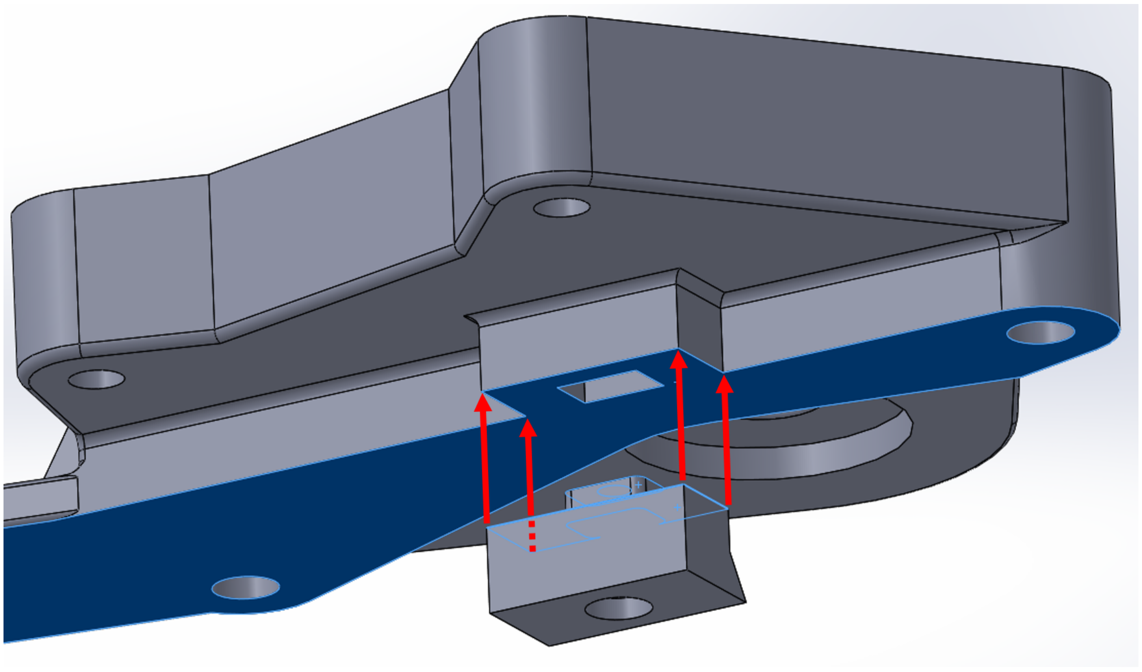

Figure C.3: The separation of certain components into distinct, smaller, complementary pieces was necessary to improve the manufacturing conditions. In the Figure's case, prior to the separation, choosing a printing ground surface was notoriously difficult, with most choices requiring a surplus of support material. After separation, the surface highlighted in blue is an excellent candidate for the printing ground surface, only entailing a few changes to the original design.

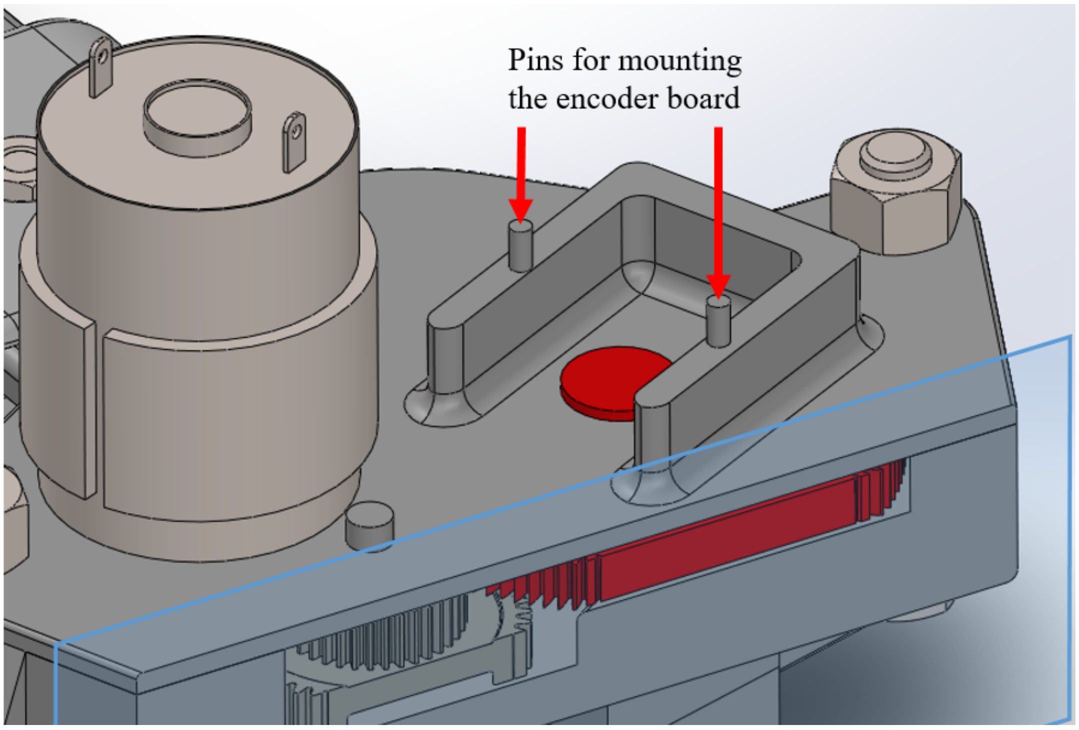

Figure C.4: Small structure for mounting the encoder board. A section view shows the gear to which the encoder's magnet will be attached. The DC motor can be seen on the left.

to each other, since this movement had to be as smooth and linear as possible, to prevent from adding unnecessary complexity and loading to the motor.

Once all pieces were ready, the device was assembled, which encompassed a number of steps steps. Some consisted of lubricating the moving pieces. When dealing with polymeric materials, solvents and other reactive lubricants are, in general, to be avoided, since they can easily react with the polymeric chains and debilitate the 3D printed scaffolds. To this end, a silicone oil was used. The exact composition is unknown (the recipient does not specify it), though the author presumes it to be PDMS (polydimethylsiloxane), the most widely used silicone-based organic polymer. It boasts high thermal stability and rheological properties that make it an excellent choice for a lubricant. The recipient does specify, however, its viscosity: V100, that is, a kinematic viscosity of 100 cSt (centi-Stoke), that is, 100 [mm^2/s].

At last, following the nomenclature of Figure C.1 and the configuration of Figure C.2, the assembling procedure was:

1. Apply the PDMS lubricant to the Forearm Piece's joint structure and mount it in the Gearbox, from below.
2. Fit the Key and mount Gear 6.
3. Mount the Foot Piece, securing it with the M3 screw, which should be inserted from the Gearbox's upper side. Tighten the screw with an M3 nut in the Foot Piece.
4. Mount the Gear 2 - Pinion 3 piece in the Gearbox, while previously applying lubricant to the shaft.
5. Lubricate the Gear 2 - Pinion 3 piece, the Gear 4 - Pinion 5 piece and Gear 6.
6. Mount the Gear 4 - Pinion 5 piece in the Gearbox, carefully so as to mate the gears' teeth with those of Gear 6 and Pinion 3.
7. Mount the DC motor in the Top Piece, using the M2 screws.
8. Lubricate the DC motor's pinion (Pinion 1) and mount the Top Piece over the Gearbox.
9. Mount the Bottom Piece under the Gearbox, holding it still with help from the Foot Piece and securing the Forearm Piece with lubricant in-between.
10. Check if the transmission mechanism moves as expected and place the M5 and M6 screws, from under the Gearbox. The M5 screws should go in the shortest holes, whereas the M6 screws should go in the longest.
11. Tighten the screws in place with their respective (M5 and M6) nuts.

Once completely assembled, buckles could, in theory, be inserted in the device's arms, in order to attach the whole device to a user. The complete assembly of the device together with the buckles is shown in Figure C.2.

Anexo D

Experiment Protocol

Execution of the experiment was comprised of the following steps:

1. Mark the positioning for the electrodes. One pair for the biceps brachii, the other for the triceps brachii. The electrodes on the biceps muscles will be placed on the line between the medial acromion and the fossa cubit at one third from the fossa cubit. The electrodes on the triceps muscles will be placed at 50% on the line between the posterior crista of the acromion and the olecranon at 2 finger widths medial to the line. The reference electrode will be placed on the bursa of the olecranon, in the boniest neighbouring spot.

2. Prepare the subject's skin for electrode attachment. The first step is to remove any excess hairs. The second step is to clean, disinfect and lightly abrade the skin with rubbing alcohol, until it displays a light red color.

3. Attach the pre-gelled electrodes.

4. Attach the inertial sensors. The upper arm IMU should be approximately between the two sEMG sensors. The forearm IMU should be close to the wrist, on the dorsal side. Precise positioning is not necessary.

5. Check the sEMG signals being acquired by the sensors with an oscilloscope.

6. The subject must be standing besides the system, carefully to not obstruct the robotic device.

7. Start the Python script. It will ask the researcher, in succession, for the desired sampling window length, increment length and the joint angle estimation algorithm's gradient descent step size.

8. Perform calibration. The program will wait for permission between each calibration step. The first step requires the subject to stay still, with the arm in a vertical, relaxed and fully extended position, with the wrist in a neutral position, for a total duration of 30 seconds. The second step requires the subject to repetitively perform elbow flexion into wrist supination into wrist pronation into elbow extension, for a total duration of 45 seconds. The third step requires the subject to return to the first position and rest there for a few seconds.

9. Perform 6 trials of repetitive elbow flexion/extension movements, while keeping the wrist in a neutral position. Each trial has a duration of 60 seconds. Rest period is 60 seconds. Frequency of the complete movement is 20 [Hz]. A metronome is used to help the subject maintain a consistent pace. For trials 1, 2 and 3 (sorted at random), the participant is asked

to hold his wrist in a supinated position, whereas for trials 4, 5 and 6 the participant is asked to hold his wrist in a neutral position.

10. Allow the program to save all trial data.
11. Split the data into 5 train/validation samples and 1 test sample. The test sample is held off until the models are trained, for offline testing.
12. Train the random forest with default settings.
13. Perform an online test. The test consists of a single 60 second trial, during which the subject performs repetitive elbow flexion/extension movements, while keeping the wrist in a neutral position. Frequency of the movements is 20 [Hz]. A metronome is used to help the subject maintain a consistent pace.
14. Restart the program, skip trial data acquisition and instead load the previous data. Perform Bayesian optimization with 20 iterations. The subject can rest during this period, without removing any sensors.
15. Perform an online test. The test consists of a single 60 second trial, during which the subject performs repetitive elbow flexion/extension movements, while keeping the wrist in a neutral position. Frequency of the movements is 20 [Hz]. A metronome is used to help the subject maintain a consistent pace.

Figures D.1 to D.6 showcase the execution of the experiment. Figure D.7 shows the data acquired during the six trials. Figure D.8 compares the biceps' EMG data raw, pre-processed and three features (SampEn, WL, RMS).

Figure D.1: The pre-gelled electrodes used in the experiment.

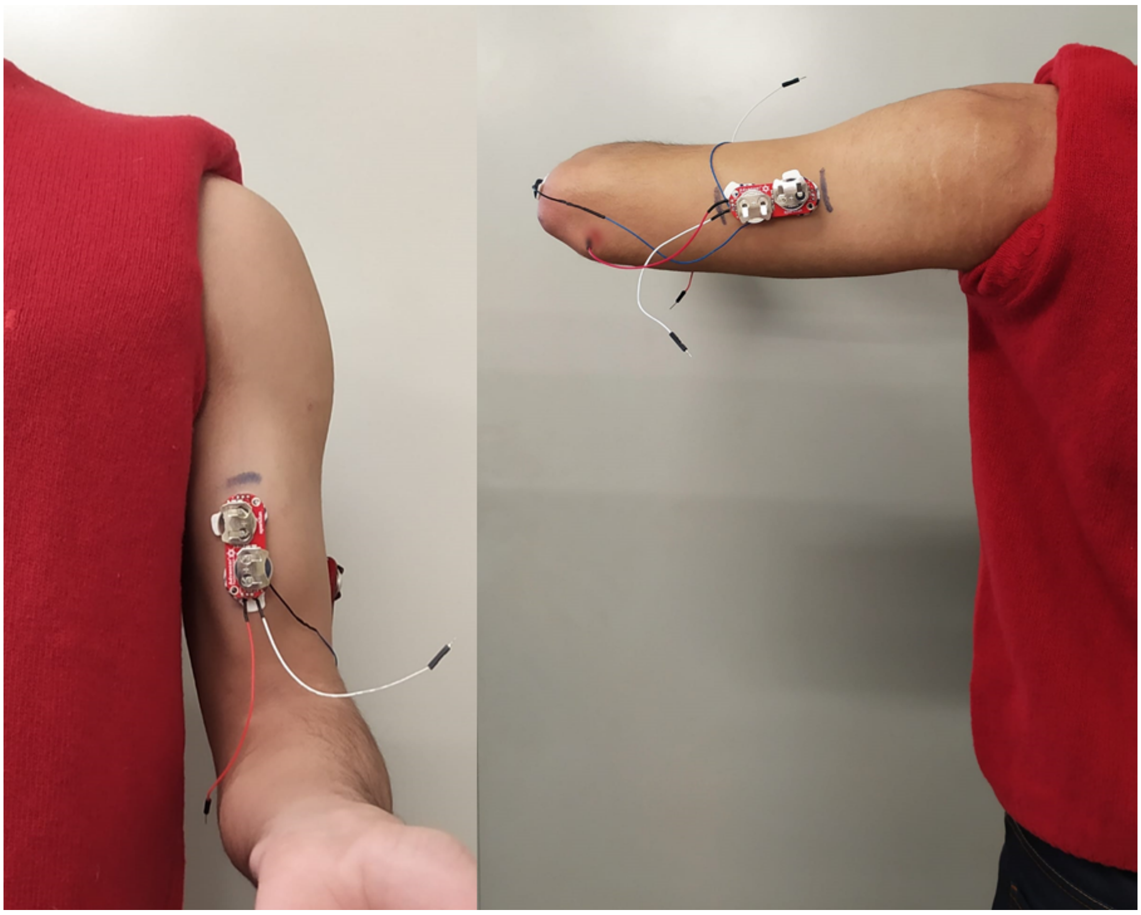

Figure D.2: Placement of the electrodes (and sEMG sensors) on a subject during the experiment.

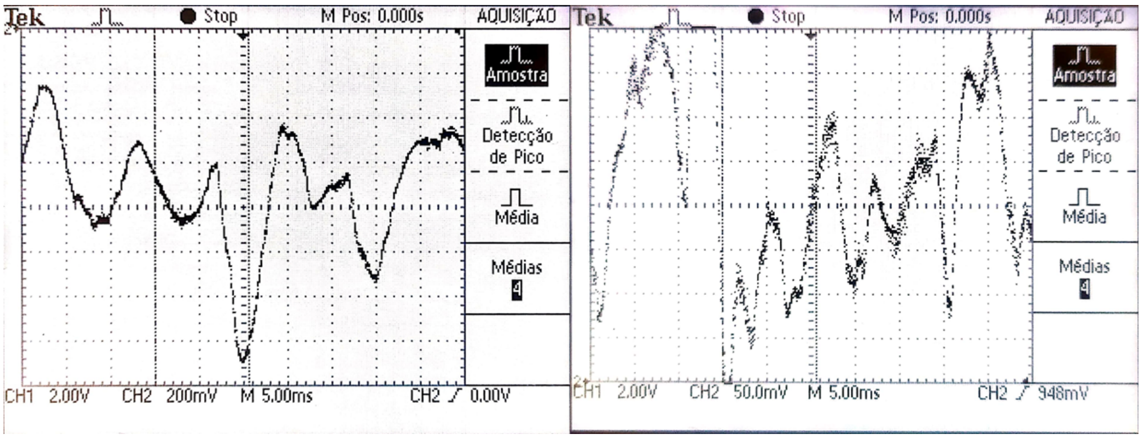

Figure D.3: Oscilloscope readings of the sEMG signals for the subject's biceps (left) and triceps (right) muscles.

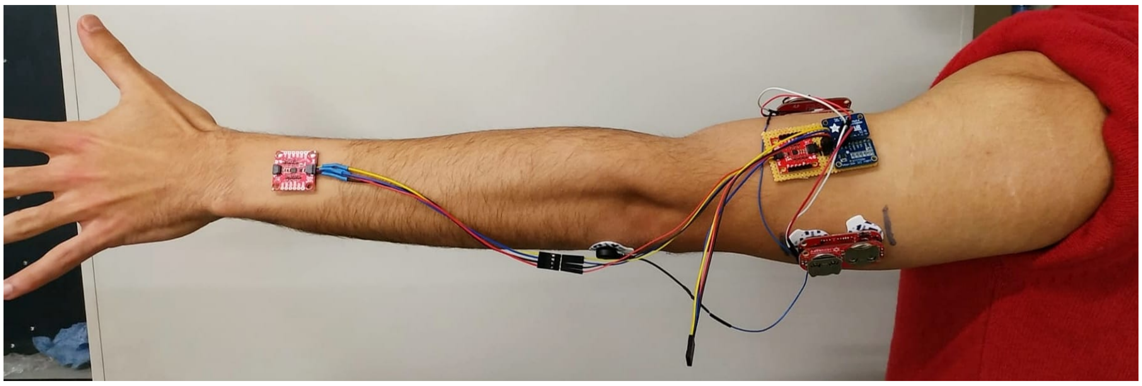
Figure D.4: Mobile subset of the system attached to the user.

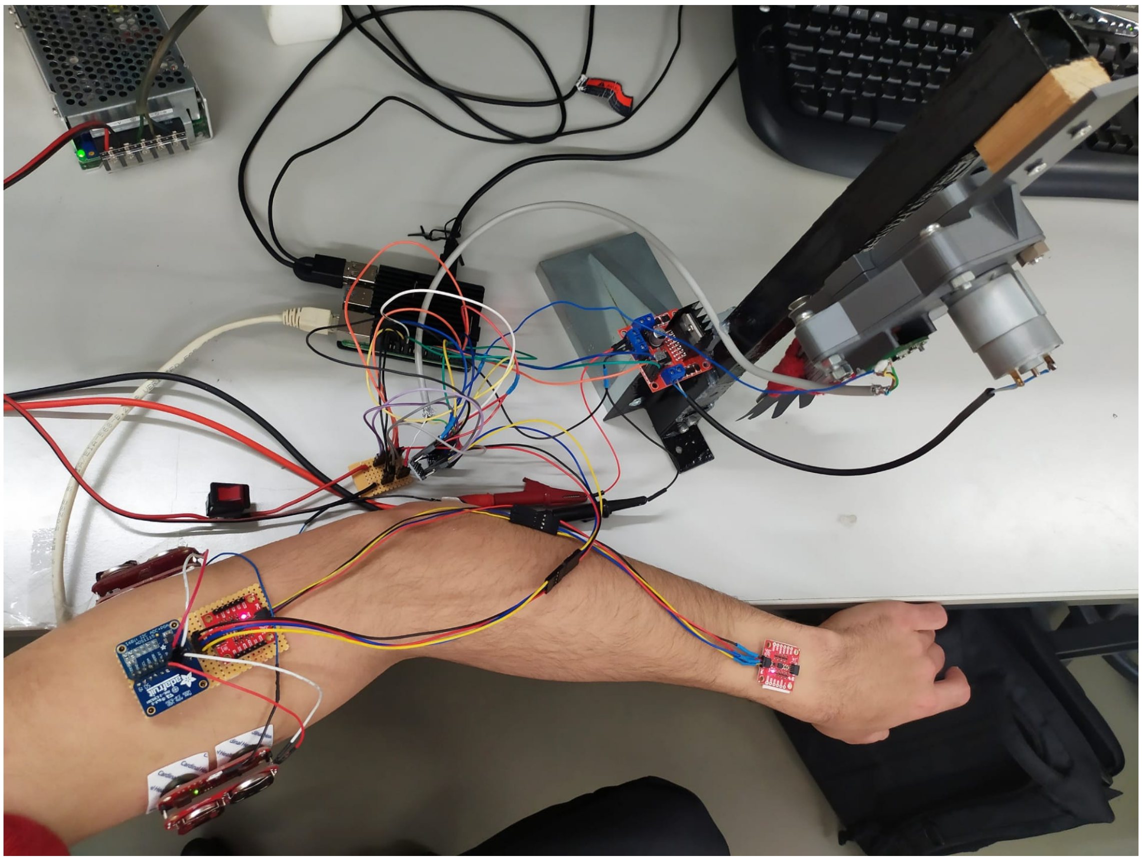
Figure D.5: Complete system mounted and wired to the subject.

Table D.1: Variable importance computed using MDI, for both the model trained using default hyperparameters and the model trained after Bayesian optimization.

Feature	**Random Forest Default Hyperparameters**	**Random Forest Optimized Hyperparameters**	**Improvement**
Total (R^2)	0.6499	0.7134	0.0635
WL_{bi}	0.2105	0.1992	**-0.0113**
$CC1_{bi}$	0.1925	0.1864	**-0.0061**
RMS_{bi}	0.1776	0.1571	**-0.0205**
WL_{tri}	0.0952	0.1022	0.0070
RMS_{tri}	0.0940	0.1196	0.0256
$CC1_{tri}$	0.0601	0.0652	0.0051
$SampEn_{tri}$	0.0332	0.0326	**-0.0006**
$SampEn_{bi}$	0.0293	0.0394	0.0101
$CC2_{tri}$	0.0211	0.0187	**-0.0024**
$CC3_{tri}$	0.0191	0.0144	**-0.0048**
$CC3_{bi}$	0.0186	0.0163	**-0.0023**
$CC4_{bi}$	0.0177	0.0163	**-0.0015**
$CC4_{tri}$	0.0169	0.0159	**-0.0009**
$CC2_{bi}$	0.0142	0.0166	0.0024

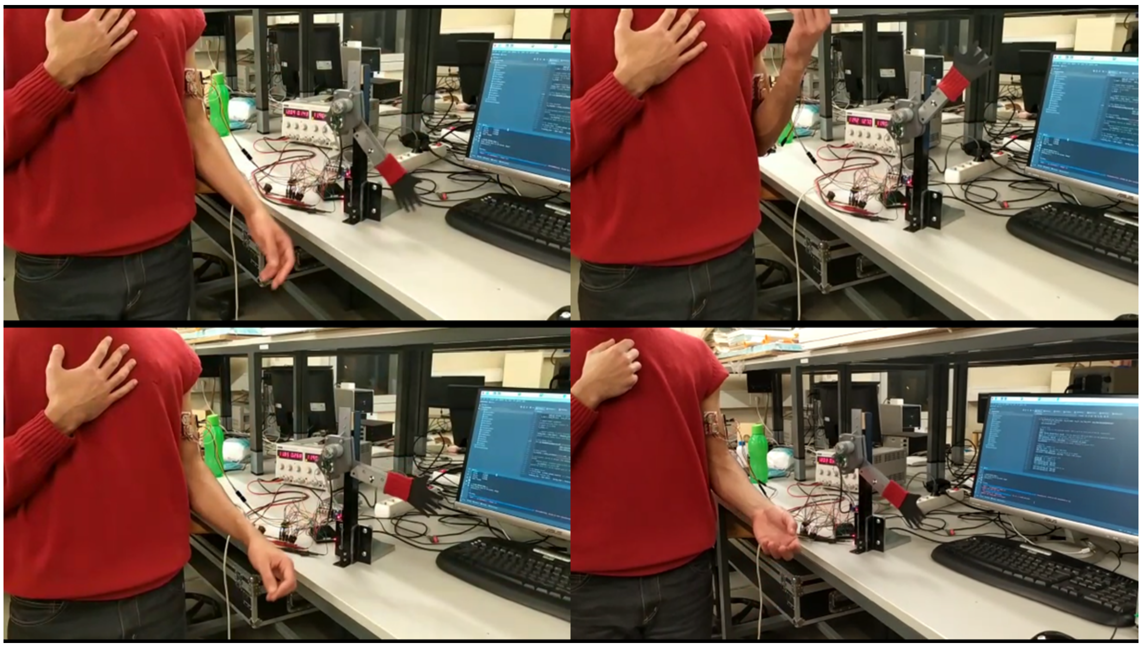

Figure D.6: Side-by-side comparison of the subject and the robotic device during online testing.

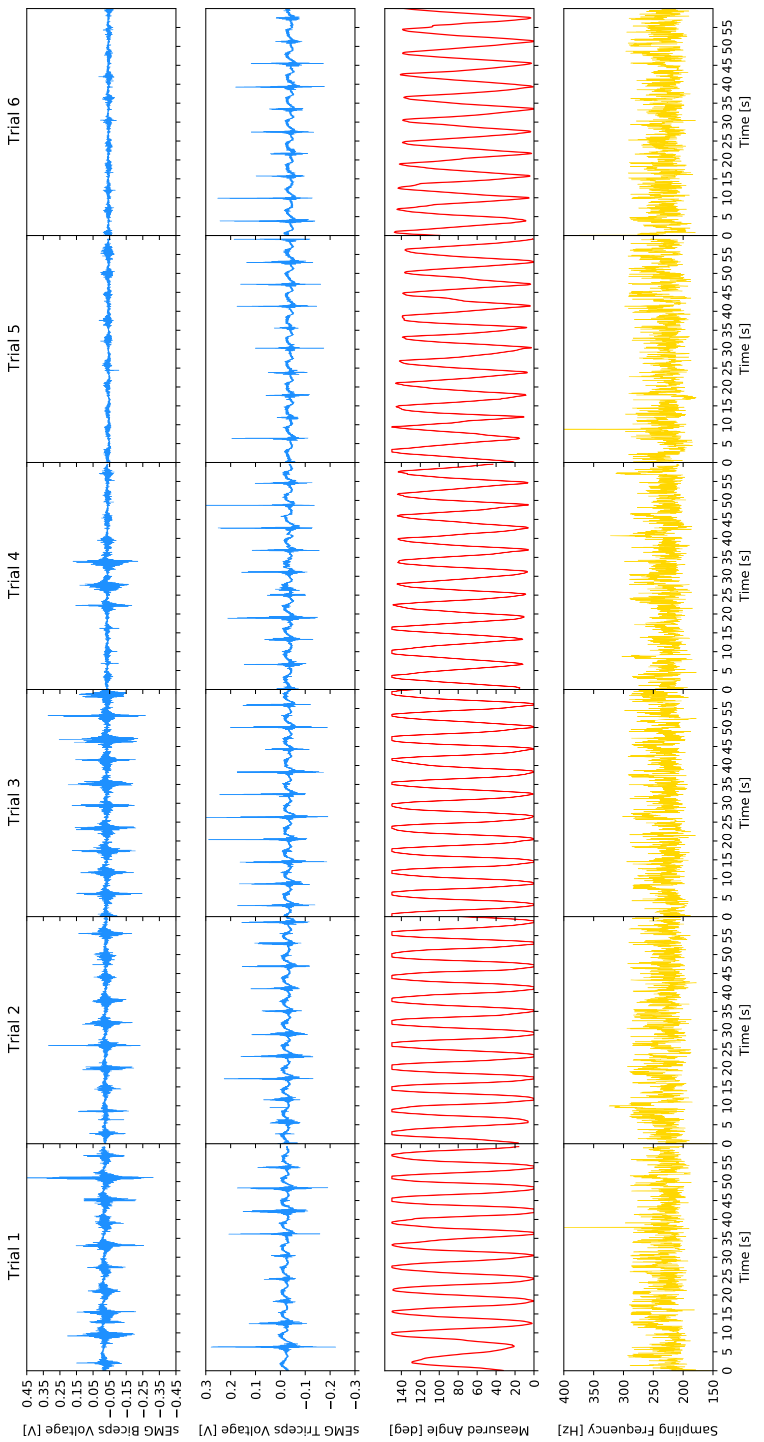

Figure D.7: Data acquired during the experiment's six trials.

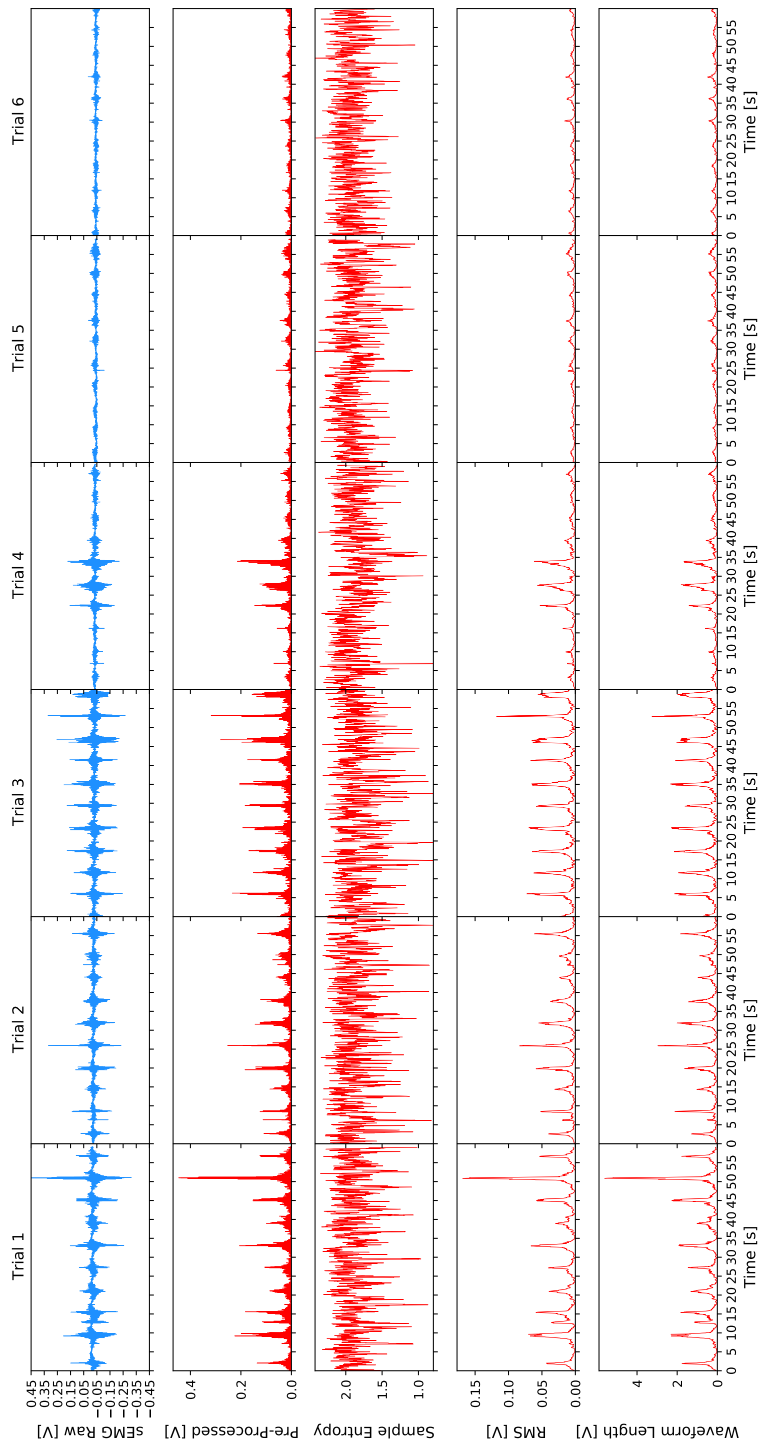

Figure D.8: Comparison between raw EMG data acquired from the biceps, the same EMG data after pre-processing and three features extracted (SampEn, WL, RMS).