POLITECNICO DI TORINO

DIPARTIMENTO DI INGEGNERIA MECCANICA E AEREOSPAZIALE
(DIMEAS)

Master Degree in Biomedical Engineering

Master Degree Thesis

Real-Time Embedded System for Event-Driven sEMG Acquisition and FES Control

Supervisor
Prof. Danilo Demarchi

Co-Supervisor:
Dr. Paolo Motto Ros

Candidate
Ricardo Maximiliano Rosales

TORINO, July 2019

Abstract

Nowadays, a larger number of people, suffering neuro-muscular disorders, are being treated with *Functional Electrical Stimulation (FES)* techniques as a result of its valuable advantages in the therapeutic and rehabilitative fields.

This project purposes a *real-time cross-platform embedded system* for controlling a multi-channel FES stimulator, whose regulation is performed based on an event-driven technique applied to the *Surface Electromyography (sEMG)*. This method, called *Average Threshold Crossing (ATC)*, allows a low power acquisition due to a reduction in the data size, and, as a consequence, in the entire circuitry complexity.

The system is composed of two digital goniometers, a FES apparatus and two types of wirelessly communicated sEMG acquisition structures, the individual channels and the four-channel acquisition board. As a main processing unit, several platforms can be used, however here, a minimalist single board computer, named Raspberry Pi 3b+, was implemented.

A multi-paradigm open-source programming language, referred as Python, was chosen for developing the software architecture through the employment of an *Object Oriented Programming* and a *Multithreading* approaches. Furthermore, a multi-screen Graphical User Interface (GUI), for allowing the user to control the entire application, was created, utilising Kivy, an open source portable pythonic library.

Several experiments were conducted in order to assess the system performance, firstly, the processing requirements are contained under the 73% of the Central Processing Unit (CPU) power and to around 90 MB of the Random Access Memory (RAM), when stimulating with four channels. In contrast, when employing a single channel, with and without the ATC data elaboration, a decrement of the 20 percent in only the CPU usage is registered by cause of the multithreading structure and not due to the channel data processing. Secondly, the mean and median latency for the stimulation current amplitude updating, in the worst case scenario, are equivalent to 11.8 ms and 4.8 ms, in 1400 assessed samples, which are suitable values for attaining the real-time aim. To conclude, *in-vivo* experiments have provided satisfactory outcomes, computing mean correlation values, between the therapist and patient joint angular deflections of 0.86 ± 0.07 and 0.95 ± 0.03, in the active Elbow Flexion and Knee Extension, respectively.

Summary

Disabling injuries and pathologies, such as the *Spinal Cord Injury (SCI)* and *Stroke* have a high occurrence rate, in our modern societies, and both likely conduce to a, temporal or permanent, reduction or loss of the physiological motion. The Functional Electrical Stimulation (FES) is a worldwide recognized technique for treating and recover, or even replace, those musculoskeletal impairments through the application of an electrical current. On the other hand, the Surface Electromyography (sEMG) refers to the measurement, from sensors at the skin surface, of the biopotentials originated in the muscle contraction. In literature, several works state that the combination of this two techniques could generate a more accurate control in the stimulation process.

The aim of this thesis is to reach that integration between those two techniques for performing a biomimetic superficial muscular excitement regulated by the electric signals generated by specific muscles. The hardware architecture is made up of distinct devices, such as goniometers, a stimulator and a Bluetooth Low Energy (BLE) acquisition subsystem, which communicate with the central workstation, where the information is received, processed and transmitted in real time, in order to allow the therapist, or even the patient, to control the stimulation by means of a single or several healthy muscles. Furthermore, the portable software implementation embeds these elements together and creates a Graphical User Interface (GUI) for enabling the user to perform the task easily, plotting feedback data as fast as the events occur and selecting suitable parameters for the stimulation, as well as for the acquisition procedure.

In order to provided a detailed description of this project the following nine chapters are presented:

1. **Introduction:** discusses some important concepts for the reader to have, in order to understand the whole work. Such as, an introduction to the muscular tissue 1.1, how its electrical signal is measured 1.2 and also, the description of a specific digitalization technique, the Average Threshold Crossing (ATC), to be applied on that signal 1.3. In addition, it presents the muscular FES and describes its features 1.4, and even how its action can be quantifiable 1.5. On the other hand, at the acquisition site, because the muscle data is delivered through a wireless link to the workstation, the Bluetooth Low

Energy (BLE) protocol is described 1.6. Finally, an introduction to the main topics concerning system architectures, where subsystems are implemented are explained in 1.7.

2. **State of Art:** it covers an overview of different modern projects which involve themes related with this thesis. At 2.1, an analysis of wireless protocols and biomedical requirements regarding to the data broadcasting speed is discussed, whilst in 2.2 some assessments, of how well the muscle activity is represented by the ATC, are reported. To conclude, two sections, the 2.3 and 2.4, describe how the Surface Electromyography and the Average Threshold Crossing data have been employed to control a stimulator, respectively.

3. **System Description:** this chapter details the components of the system. Firstly, an outline of the sEMG channels circuitry and the two types of acquisition approaches, the individual channels and the four channel board, utilised for this project are presented in 3.1, while a similar explanation of the functional electrical stimulator and the goniometer features are performed in 3.2 and 3.3. In the final section, as all the implementation is mainly conducted in a Raspberry Pi, in 3.4, its main capabilities are reported.

4. **Individual Channels Firmware:** the development of the acquisition channel firmware is illustrated here. Specifically, in 4.1, the context in which this programme needs to be placed is portrayed and in 4.2, the application it self is described deeply.

5. **Embedded Software Development:** at the beginning, an introduction to the Object-Oriented Programming (OOP) paradigm is reported, later, a brief application constitution is detailed in order to clarify the roles of some modules composing the work. Lastly, in the successive sections every class used is explained, outlining its main properties and behaviours.

6. **Graphical User Interface:** in this chapter the different screens and the library employed are described. In 6.1, the screen in charge of loading all the patient and therapist required information is shown, whereas the ones for executing all the calibration procedures and the main stimulation action are portrayed in 6.2 and 6.3, respectively. Finally, the last screen is shown in 6.4, where the employed application parameters can be checked, saved and even, in some cases, modified.

7. **Multithreading:** it details how the application task is dynamically performed in order to achieve a real time implementation, and also, presents an introduction to multitasking actions in Python.

8. **Results:** all the evaluation methods, executed for characterizing the system behaviour, and their outcomes are discussed in this chapter of the thesis. In 8.1

two different assessments regarding the resources consumption and the data processing latency in workstations with distinct operative systems, are presented. The first one permits to register the processing and memory conditions when the application is being executed, resulting the latter strongly steady to 90 MB while the former varies from 5% to 70% of the total computational power. Besides, a sharp influence of the multitasking structure implemented is prove as a result of the low variations in the Central Processing Unit (CPU) consumption when a data elaboration is and is not being performed with one or four channels. The second one demonstrates that the 130 ms real-time constraint is widely attained, as the mean and median latency in the current calculation are of 11.8 ms and 4.8 ms, respectively; therefore, both reveal that the implementation is widely acceptable for real time applications in different platforms, and even in one specific single board computer with poor quality resources, as the Raspberry Pi. Finally, in 8.2, two benchmark movements, the Elbow Flexion and Knee Extension, were studied, in different subjects, for determining how reliable the system is, in terms of therapist-patient biomimetic inducted movement similarity, collecting correlation values of 0.86 ± 0.07 and 0.95 ± 0.03, respectively.

9. **Conclusion:** in the last chapter a summarising of the whole work is done, highlighting the successful outcomes found in the experimental tests. Finally some possible future implementations, researches and experimentations are described in 9.1, in order to allow other people to improve the present project, e.g., how the frequency can be changed in real-time for mirroring the motor unit action and decrement the rapid muscular fatigue.

Acknowledgements

I would like to express my sincere gratitude to my supervisors Danilo Demarchi, Paolo Motto Ros and Fabio Rossi, for having guided and given me the opportunity to expand my knowledge with the development of this thesis. I also thank my colleagues in the laboratory, who have made me laugh and enjoy this time together.

Finally, I dedicate this thesis to my family and friends, from the bottom of my heart, because if it were not for them I would not have achieved even half of what I have attained until today.

Desidero esprimere la mia sincera gratitudine ai miei supervisori Danilo Demarchi, Paolo Motto Ros e Fabio Rossi, per avermi guidato e dato l'opportunità di ampliare le mie conoscenze con lo sviluppo di questa tesi. Ringrazio anche i miei colleghi del laboratorio, che sono riusciti a farmi divertire questo tempo insieme.

Infine, dedico questa tesi alla mia famiglia ed i miei amici, dal profondo del mio cuore, perché se non fosse per loro non avrei realizzato nemmeno la metà di quello che ho raggiunto fino ad oggi.

Querría expresar mi sincero agradecimiento a mis supervisores Danilo Demarchi, Paolo Motto Ros y Fabio Rossi, por haberme guiado y brindado la posibilidad de ampliar mis conocimientos con el desarrollo de esta tesis. Agradezco también a mis compañeros del laboratorio, que han sabido hacerme reír y disfrutar de este tiempo juntos.

Finalmente, esta tesis es dedicada a mi familia y amigos, desde el fondo de mi corazón, porque si no fuera por ellos yo no habría logrado ni siquiera la mitad de lo que he conseguido al día de hoy.

Contents

List of Figures

List of Tables

Acronyms

ADP	Adenosine Diphosphate.
API	Application Programming Interface.
AROM	Active Range of Motion.
ARV	Average Rectified Value.
ATC	Average Threshold Crossing.
ATP	Adenosine Triphosphate.
ATT	Attribute Protocol.
BLE	Bluetooth Low Energy.
CCCD	Client Characteristic Configuration Descriptor.
CCFES	Contralaterally Controlled Functional Electrical Stimulation.
CCLM	Continuous Channel List Mode.
CPU	Central Processing Unit.
CRC	Cyclic Redundancy Check.
CV	Conduction Velocity.
DAC	Digital to Analog Converter.
DD	Double Differential spatial filter configuration.
DRL	Driven Right Leg circuit.
ECG	Electrocardiography.
EMG	Electromyographic.
FES	Functional Electrical Stimulation.
GAP	Generic Access Profile.
GATT	Generic Attribute Profile.
GPU	Graphics Processing Unit.
GUI	Graphical User Interface.

HCI	Host Controller Interface.
INA	Instrumentation Amplifier.
L2CAP	Logical Link Control and Adaptation Protocol.
LL	Link Layer.
LpWiFi	Low Power Wi-Fi.
MDF	Median Frequency.
MHC	Main Hardware Configuration.
MISO	Master Input Slave Output.
MNF	Mean Frequency.
MOSI	Master Output Slave Input.
MU	Motor Unit.
MUAP	Motor Unit Action Potential.
MVC	Maximum Voluntary Contraction.
OOP	Object-Oriented Programming.
OSCLM	One Shot Channel List Mode.
PC	Personal Computer.
PCB	Printed Circuit Board.
PHY	Physical Layer.
PROM	Passive Range of Motion.
PSD	Power Spectral Density.
RAM	Random Access Memory.
Redox	Reduction and Oxidation chemical reactions.
RMS	Root Mean Square.
ROM	Range of Motion.
SA	Stimulation Artefact.
SBC	Single Board Computer.
SCI	Spinal Cord Injury.
SD	Single Differential spatial filter configuration.
SDK	Software Development Kit.
sEMG	Surface Electromyography.
SMP	Security Manager Protocol.
SNR	Signal-Noise Ratio.
SoC	System on Chip.
SP	Single Pulse.
SPI	Serial Peripheral Interface.

TC	Threshold Crossing.
UUID	Universal Unified Identifier.

Chapter 1

Introduction

According to the World Health Organization[1, 2], over a billion people worldwide are estimated to possess at least some form of disability. The *Stroke* is one of the leading causes of it, which is even more dramatically, because 1 in 6 people suffer this pathology at least once in their life. Furthermore, almost half million people have a *Spinal Cord Injury (SCI)*, every year, which can lead to soft or strong musculoskeletal conditions that deteriorate not only their physical health, but also their economical and social situations. One way to overcome some of those costs, is the access to suitable rehabilitation and therapeutic treatments and devices, as the ones purposed in this thesis.

This work presents a multi channel cross-platform real-time system which allows a biomimetic control of the *Functional Electrical Stimulation (FES)* in function of an specific reduced digital signal obtained from the *Surface Electromyography (sEMG)*, with an event-driven technique. The last method, called *Average Threshold Crossing (ATC)*, allows the implementation of a low power and wearable sEMG acquisition system, capable of broadcast data taking advantage of the *Bluetooth Low Energy (BLE)* protocol. The hardware, besides this acquisition subsystem, is completed by the presence of FES device, for applying the stimulation, and two digital-goniometers, for providing a feedback of the procedure effectiveness. A multi-screen *Graphical User Interface (GUI)* was created from open-source free cross-platform tools, in order to provide it with robustness and flexibility.

Moreover, a deep and detailed hardware and software design description is discussed for giving a valuable understanding of not only the structure, but also, of how and when the synergic behaviours of the elements, composing this embedded system, attain the intended goal.

In the following some of the fundamental concepts, that has been implemented here, are described, for giving a better and easier comprehension of the whole project.

1.1 Excitable Tissue: The Skeletal Muscle

The muscular tissue has a relevant role in different processes that our body do everyday, such as mobility, maintenance of the posture and stability, blood circulation, excretion, digestion, respiration and so on.

It is well-known that this tissue is made up of muscular cells which, as neurons, can be stimulated through chemical, electrical or mechanical ways to produce a perturbation called **action potential**. Once it has been generated, it flows along the cellular membrane and finishes on a contraction of the muscular cell.

In general, the muscles can be classified within the following three clusters: **skeletal, smooth** and **cardiac** muscles.

1.1.1 Basic Composition

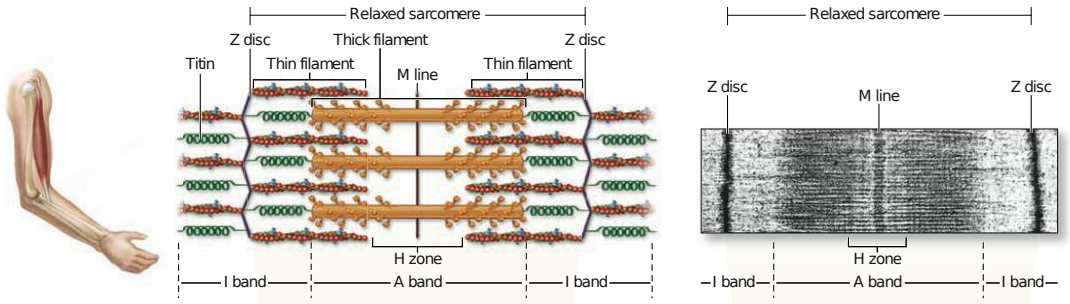
The skeletal muscle fibres [3] are innervated by myelinated nerves, which are born from big motoneurons in the anterior part of the spinal cord and are almost all the times under a voluntary control. This bond between muscular skeletal fibres and nerve endings is called **neuromuscular junction**.

Each muscular fibre is a multinucleated cell surrounded by a **sarcolemma**, which is a combination of collagen fibrils, polysaccharides and its own plasma membrane. This fibre, at the same time, is composed of several myofibrils that have many thousands of inner **myosin** and **actin** molecules (also known as thick and thin filaments, respectively). The distribution, that latter elements acquire in the space, generates different results once exposed to a polarized source of light, one of them is the presence of dark zones, where both filaments overlap, or bright ones where only actin is present. Those two, are the anisotropic *A band* and the isotropic *I band*, respectively.

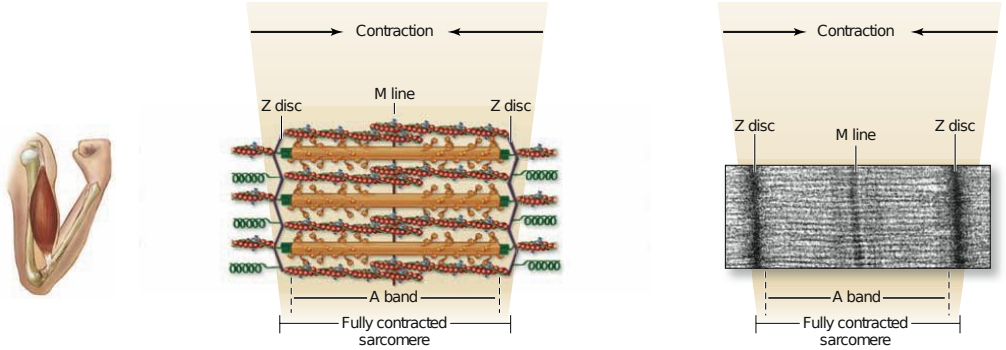
The actin molecules are bound to the Z discs, from which they prolong in both directions to reach myosin. Furthermore, these connect the myofibrils together to create the muscular fibre.

A **sarcomere** is the structure between two consecutive Z discs, this piece of microfibrils shrinks in the muscular contraction to allow the increment of the overlapping area between the two main filaments named before (as shown in Figure 1.1). Also some other elements can be found [4], such as the H zone, corresponding with a lighter portion of the A band with only the rod-like part of thick filament, and the M line, which is in the middle of the H zone and contains a protein for holding the myosin in its position, similar to the action of titin proteins that maintain the filaments in place allowing the contractile procedure to happen.

Besides, there are other components that make possible the muscular contraction. One of them is the specialized endoplasmic reticulum of these cells, called **sarcoplasmic reticulum**, which stores large amounts of Ca^{++} and allows or avoids this ion diffusion to the cellular cytoplasm according to the membrane stimulation.



(a) Relaxed skeletal muscle



(b) Fully contracted skeletal muscle

Figure 1.1: Structural modification of muscular filaments in contraction [4].

Also, two other proteins have a fundamental role, they are the *tropomyosin* and *troponin*, the former is attached with the actin filament and prevents the interaction with myosin at the resting state, whereas the latter consists in three subunits: troponin I (binds the entire molecule to the actin), troponin T (connects to the tropomyosin) and troponin C (has a high affinity for calcium ions)(see Figure 1.2).

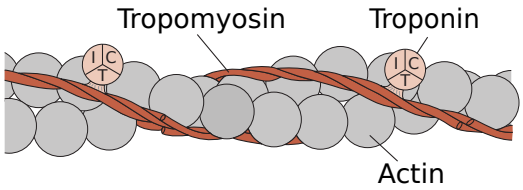


Figure 1.2: Actin, tropomyosin and troponin complex[6].

1.1.2 Muscular contraction mechanism

First of all [3, 7], the depolarization of a motoneuron is needed to make available the acetylcholine diffusion from the endings of different axon terminations to specific

areas, where some acetylcholine sensitive channels are present, in order to allow the flux of Na^+ to the cell inner. Afterwards, a muscular depolarization can generates a travelling action potential, when a certain threshold has been surpassed.

Secondly, the flowing action potential enables the Ca^{++} diffusion from inside of the sarcoplasmic reticulum, rising this ion concentration on the cytoplasm of the muscular fibre. Once this has been occurred, the troponin-tropomyosin complex, which avoids the main filaments interactions in the polarized state, reacts with the calcium exposing the active sites of the actin for the myosin heads. Then, an union occurs and the myosin head pulls the actin, shrinking the distance between the Z discs in the boundaries of the sarcomere, and generating the movement (as shown in Figure 1.3). The energy for the power stroke, that myosin heads need to make the interaction, is stored as an Adenosine Triphosphate (ATP) molecule that is hydrolysed into Adenosine Diphosphate (ADP) and phosphate.

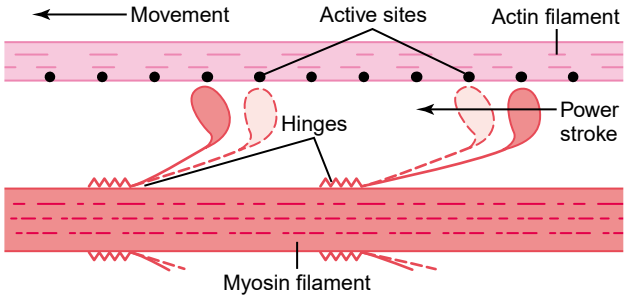


Figure 1.3: Tilting and subsequent pulling of the myosin heads to the actin active sites [5].

Finally, the depolarization ceases, the calcium ions return to the sarcoplasmic reticulum, and the tilting of myosin heads to generate the pulling of actin filaments enables the ADP and phosphate ion release from the myosin head. Then, as this site is empty a new ATP molecule binds and the myosin-actin detachment takes place, whilst the contraction reaches its end.

1.1.3 Sorts of Fibres

The mammals possesses hundreds of muscles[8], which are composed for different sorts of fibres types at the same time. In general, they can be classified into **fast-twitch** or **slow-twitch** fibres, changing from their biochemical composition to the way they looked.

The former are larger fibres recruited for short and strong contractions, they use a glycolytic metabolism allowing the faster energy acquisition, have less oxidative reactions, so blood supply and mitochondria numbers are lower, and finally the lack of myoglobin gives a white tone to these fibres. Conversely, the slow-twitch ones,

that are recruited on long and weak contractions, are also known as red fibres due to a myoglobin increment, which is a consequence of the oxidative metabolism preponderance that magnifies blood supply and the amount of mitochondria. Besides, the free calcium level is the double of its counterpart.

1.1.4 Sorts of Contractions

The most usual muscular exercises enable us to categorized the muscle contractions as **isotonic** or**isometric**.

Whether the muscular fibres obtain force by changing their length with a steady muscular tension, the former type of contraction is happening, but in contrast, whether the muscle length is blocked and also the muscular tension is fluctuating, it means that the latter contraction type is performed.

An isotonic contraction is associated with a dynamic movement, whilst an isometric with a static one, furthermore, it is often split into **concentric**, which permits the muscle to shorten while energy is changing, or the **eccentric** one, which contrary to its counterpart, allows an elongation of the muscular fibres when the high load that they are withstanding exceeds the force that the muscle can create [10] (this is shown in Figure 1.4).

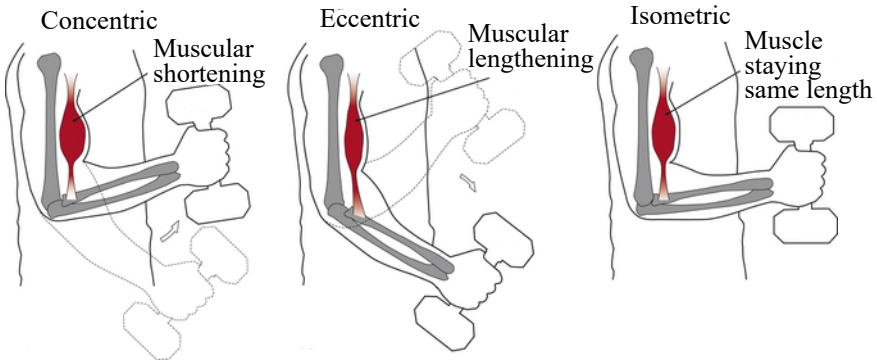


Figure 1.4: Types of muscular contractions [9].

1.1.5 Regulation of the Muscle Force

The **Motor Unit (MU)** is the functional unit of the muscular system and is composed by a motoneuron and the muscular fibres that it stimulates.

A single action potential triggers a short contraction followed by a relaxation, this muscular reaction is called **twitch**, and it happens 2 ms after the beginning of the depolarization, as shown in Figure 1.5.

There are two main processes that regulate the muscle force [11]:

- The first one involves the amount of motor units that have been stimulated, in other words, the **recruitment**. The *size principle* tells us that the motor

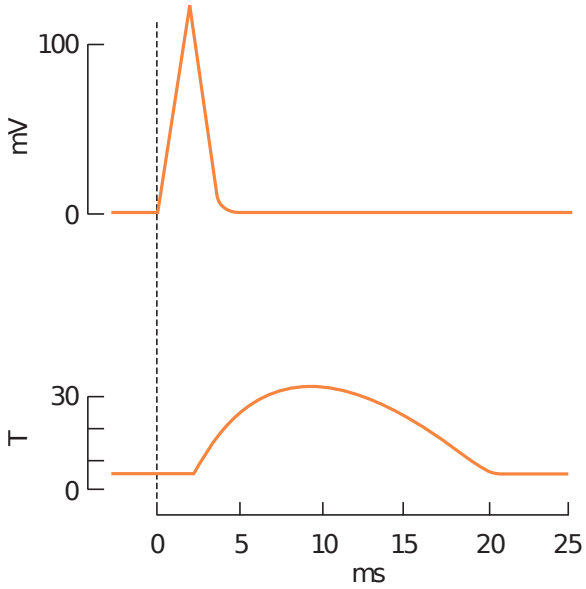


Figure 1.5: Electrical and mechanical response of one muscular fibre. Where T is a tension magnitude in arbitrary units [6].

units are recruited starting from the small ones to the big ones, causing an increment on force contraction. Therefore, there is a threshold distribution, where the specific one could be considered steady for each motor unit. This distribution is suggested to by exponential [19], in order to active a lot of little fibres first, and only a few large ones then. Besides, the percentage of **Maximum Voluntary Contraction (MVC)** indicates the range of motor units recruitment, i.e., which MVC percentage correspond to a total motor unit activation in a specific muscle.

- Conversely, a strength increment can be obtained due to a rise in the firing rate of pertinent motoneurons, when this phenomenon is perform at high frequencies the muscle fibres can not be completely relaxed when starting a new stimulation. This reaction that results in a fusion of single twitches into one continuous contraction is referred to, as **tetanus**.

1.2 Electromyography

The combination of action potentials originates a perturbation, which flows through our ionic medium and, specifically, if the depolarization waves come all from the muscle fibres belonging to the same MU, it is known as a **Motor Unit Action Potential (MUAP)**. So, in conclusion, the **Electromyographic (EMG)** refers to the measurement of the electrical perturbation created by all MUAPs that are in contraction at the moment [12].

1.2.1 EMG techniques

Two techniques can be used to collect this electrical signal:

1. **Surface Electromyography (sEMG)**, where the electrodes are put upon the skin (known as surface electrodes) without penetrating it.

2. **intramuscular EMG**, where the approach is to obtain the signal from the inner, inserting needle or wire electrodes [13].

Both procedures have their own benefits and drawbacks, as expected the sEMG is a non-invasive technique where the risk of infection and procedure complexity are outstandingly lower but, in spite of that, in some cases the second approach is preferred because it allows to measure a clean single MUAP, as can be evaluated in Figure 1.6.

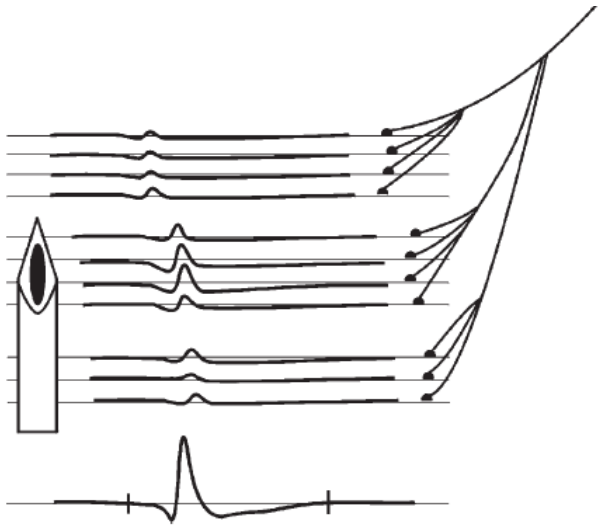


Figure 1.6: Motor Unit Action Potential (bottom), it can be seen that the four muscle fibres in the middle cause higher measurements due to, their proximity to the needle electrode [15].

Therefore, the invasive EMG can detect variations regarding the motor unit, such as its size, structure and performance. Besides, it allows us to realize of the presence of spontaneous contracting fibres that are a main symptom of denervation [16].

In addition, the non invasive procedure could lead to a difficult signal interpretation due to the different perturbation sources present in the body and in the environment [14], but using it correctly, is still useful for many scopes.

1.2.2 sEMG Signal

The sEMG [16], as said, does not permit us to study MUAPs and the background nervous signal, that originates them, but, in spite of that it gives us information of the muscles as a whole, in terms of their behaviour, patterns, or fatigue state. Because of that, this technique has a noticeable role in some areas where assessments have to be implemented several times. Such areas are, all sorts of sports, occupational medicine, rehabilitation and so on.

This signal is considered quasi-stochastic when no post-processing was applied to it, which means that when a muscle contracts the motor units recruited are not every time the same, causing the signal to be incredibly different from one burst to another [18]. In other words, when a contraction is recorded there is not a pattern in the electrical activity (see Figure 1.7).

However, it is possible to name some characteristic range of amplitude and frequency that are common to most of these signals when coming from a healthy subject. For instance, the sEMG signal amplitude [17] is usually between 1 to 10 mV (+5/-5 mV), being higher in athletes [18], and its frequency range lies from 6 to 500 Hz, being the 20 to 250 Hz the most significant range.

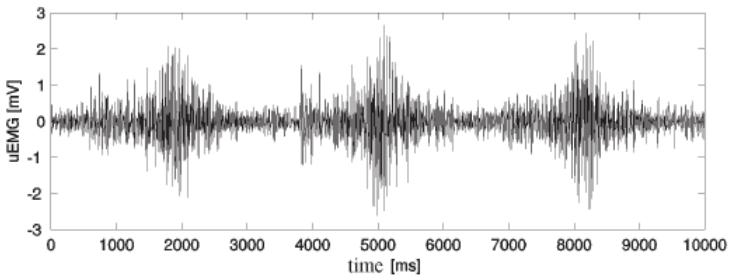


Figure 1.7: Raw sEMG signal [18].

1.2.3 sEMG perturbations

The signal fetched from the electrodes is barely similar to the inner one [16], this occurs because in the travel through tissues the signal suffers the action of different

perturbations, such as a low-pass filtering effect, crosstalk, and other sort of deep disturbs.

At the same time, the environment where measurement is taken place could be affected by noise, or simply, is our own acquisition system setting the one that generates disturbs, e.g., concerning to the electrode type, electrode placement and relative position between each other (inter-electrode distance).

It is possibly to list the following interference sources for the sEMG:

1. **Inherent noise in the equipment:** Every electronic device creates electrical alterations, the important fact is to understand how large is this going to be and whether it would affect my measurement. Some instruments can generate noise within the spectrum of the sEMG signal so choosing suitable isolation, and post-processing stages must be an important step. Moreover, as this interference can not be erased and the sEMG is really noise sensitive, the use of high quality electronic components is recommended for tending to reduce this perturbation [12, 10].

2. **Electrodes**: see 1.2.4.

3. **Environment noise**: our body is conductive and as antenna it could catch different, expanding in the ambient, electromagnetic disturbs that, sometimes, are three times greater in magnitude than sEMG signal. The biggest and common concern in this area, is the power line interference that pollutes the EMG spectrum at 50 or 60 Hz. This can be seen due to a lack of isolation in the devices that the subject is connected to, or because of a significant impedance difference between the electrodes, so off-line measuring is highly recommended.

4. **Internal noise**: the amount of fat, subcutaneous and other tissues between the electrodes and the muscle of interest, cause a smoothing effect on the signal, which reduces its high frequency content.

5. **Motion artefact**: The movement in a muscle contraction cause the electrodes to move, while this occurs their half-cell potential varies and generates irregularities in the measurement. Furthermore, the electrode cable is also another source of perturbation when the subject is moving. The spectrum of this noise is from 1 to 10 Hz.

6. **Crosstalk**: when performing a superficial measurement, a muscle or group of muscles that are not desired to be evaluate, can be taken into account due to this phenomenon. Crosstalk is the contamination of the signal of interest, because of the coupling with signals that come from other sources, as can be observed in Figure 1.10, for example, from deeper muscles or even MUAPs [12].

7. **Electrocardiographic artefact**: this problem exists when the sEMG measurement takes place in a area near to the heart, like in the trunk muscles. Besides, due to the spectra overlap of these two signals, the Electrocardiography (ECG) signal removal is deeply complicated.

1.2.4 Electrodes

In the field of bio-signal measuring, always, appears the need of obtain quantitative data for assessing a phenomenon, and for that scope sensors have to be used. In the case of bioelectrical signals, these sensors are called electrodes.

The selection of those sensors is vital to reach reliable results in the sEMG measurements, therefore, is important to know they add interference to the signal, which must be taken into account.

Redox, or reduction and oxidation chemical reactions, occur in the electrolyte-electrode interface [20]. In this application, the former arises when the metal of the electrode releases some electrons and remains positively charged, and the latter, in contrast, when those cations recover their electrons.

The skin can be considered as an electrolyte solution, as ions diffuse in the organic medium and create ionic currents [19], so, Redox reactions phenomenon is repeated on it when in contact with an electrode. It can be observed that, when these two elements are attached, the redox interaction, between the skin ions and the electrode electrons, produces a variation of ions concentration near the electrode surface, generating a different electrical potential, referred as **half-cell potential** [20]. This value depends on the material and structure composition of the interface and it is fixed when not current is crossing.

Polarization is the change, at the solution, of ions distribution and it allows to classify the electrodes in two ideal types:

* **Perfectly polarizable electrodes:** allow a current to cross through the interface by changing the ion distribution, so it is a virtual current.

* **Perfectly non-polarizable electrodes:** on the other hand, they allow the charges to move through the skin-electrode interface with total freedom.

Those electrodes exist only in the theoretical world, neither truly exist, but some of the real electrodes belong more to one of these groups than to another.

The most common one for recording Surface Electromyography (sEMG) and other bioelectric signals is the silver/silver chloride (Ag/Ag Cl) [19], which is a non-polarizable electrode. Its surface remains stable and its minimum polarization reduces instabilities (see **Motion artefact** at the previous subsection). In addition, it inserts a minimum electrical disturb to the signal, specially at low frequencies.

Under certain assumptions, of low potential and current, it is possible to represent the biopotential electrode with the electrical model shown in Figure 1.8, where

the pair R_d and C_d represent the interface polarization and impedance, while the R_s element is associated to the real resistance, of the electrode materials. As previously commented, a potential difference appears at the contact zone, which is represent as the E_{hc} battery in the model [20].

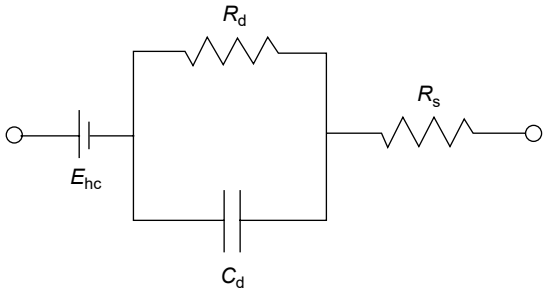


Figure 1.8: Electrode model when low potential and current are measured [20].

Furthermore, its impedance graph in function whit the frequency illustrates the low pass filter action of the electrode, and with simple laboratory tests the elements values can be found for a complete quantitative electrode characterization (see Figure 1.9). Finally is a good practice for decrement the interface impedance using gel with the electrode, and if it is possible, to shave, clean and rub the skin, to acquire a reliable measurement [19].

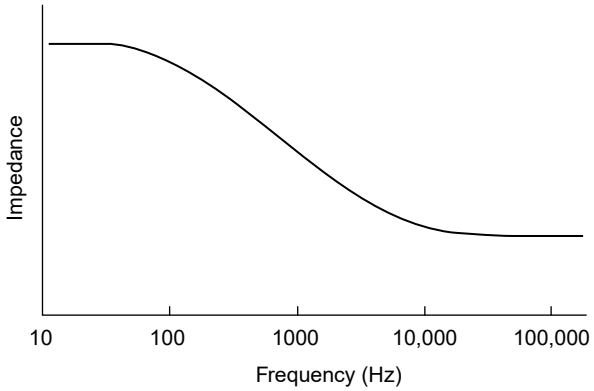


Figure 1.9: Electrode Transfer Function [20].

1.2.5 Spatial Filters

A single electrode, or **monopolar** configuration, where one of the two electrodes is placed in a neutral site, like a bony surface, is never used to detect the sEMG signal due to its bad performance [19]. Instead of that, a **spatial filter**, which implements the linear combination of the signals arriving to every electrode of the filter, is utilised.

The two more common filters are the **Single Differential SD** or **bipolar** configuration, that calculates the signal as the arithmetic difference between the two electrodes placed in the muscle, and the **Double Differential DD**, which obtains the sEMG signal as a linear combination of the potential in its three electrodes, with coefficients +1, +1 and -2. Both of them have an extra electrode for reference.

The distance between electrodes regulates how deep are the muscle fibres which are going to be taken into account for the sEMG signal calculation, it is a fact, that when inter-electrode distance is increased, the volume from which the signals are collected is also larger. Hence, the DD architecture should be used when crosstalk occurs, because of the lower inter-electrode distance that it has. Conversely, the SD one is less noisy than the DD configuration [22, 19].

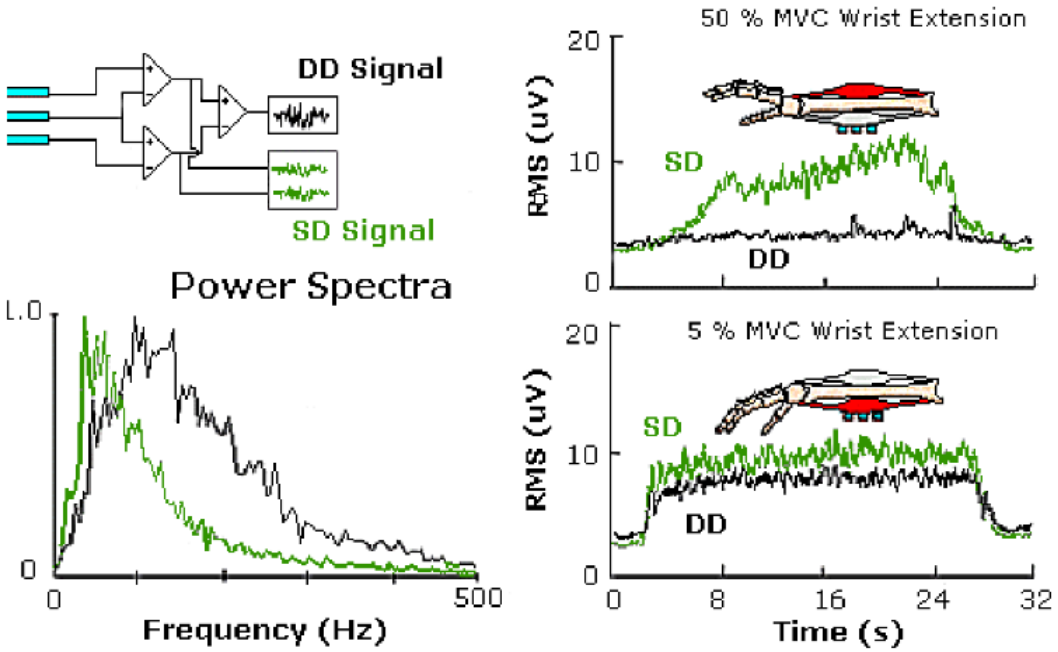


Figure 1.10: Crosstalk perturbation at a sEMG measurement [21]. The bottom-left panel correspond to the power spectra of the wrist extensor muscles (**green**) and of the Flexor Carpi Ulnaris (**black**).

In the figure 1.10, it is observed how the electrodes positioned at the *Flexor Carpi Ulnaris*, using a SD filter, detect the signal generated for the wrist extensor muscles when only a half of the MVC has occurred. On the other hand, with a DD configuration there is not any signal when no contraction of the focused muscle is present, and when it does, both techniques are successful (bottom-right panel). At the power spectra graph, it could be analysed how the signal that travels more through the tissues, until it arrives to the electrodes, has suffered a low-pass filtering effect, as mentioned before.

1.2.6 Amplification and Signal Conditioning

Usually some amplification steps are perform to render the signal analysis easier. The requirements for them are:

- low noise, achieved whit a differential amplifier.
- high input impedance, to decrement the effects of the electrode-electrolyte impedance.
- high gain, because the sEMG signal has a few mV of amplitude.
- a specific bandwidth, which allows to erase the interference being out of this range, using high and low pass filters.
- high depletion of common mode perturbation capacity, some specific circuit architectures can help with this scope like the Driven Right Leg (DRL) one.

1.2.7 sEMG Parameters

There are several indexes than can be calculated from the sEMG signal to test muscular features, like the strength or fatigue level. The most common ones are defined below [19, 23, 24].

- **Conduction Velocity (CV):** when two sEMG signals are being measured from two points along the muscle, the CV is the distance between these two points divided by the delay of one signal regarding to the other. This value tends to be lower with fatigue.
- **Frequency based:**
  - **Mean Frequency (MNF):** is the mean value of the sEMG signal Power Spectral Density (PSD).

$$MNF = \frac{\sum_{j=1}^{M} f_j PSD_j}{\sum_{j=1}^{M} PSD_j}$$

  - **Median Frequency (MDF):** is the value that corresponds to a vertical line which cuts the sEMG signal PSD in two equal parts.

$$MDF = \frac{1}{2} \sum_{j=1}^{M} PSD_j$$

  Where $PSDj$ is PSD value at the j-th frequency f_j, and M is the maximum frequency value. Both are decremented with muscle tiredness due to a rise in the low frequencies composition in the PSD of a fatigued muscle.

- **Amplitude based:**
  - **Average Rectified Value (ARV):** is the mean value of the modulus of the sEMG signal.

$$ARV = \frac{1}{N}\sum_{n=0}^{N}|x(n)|$$

  - **Root Mean Square (RMS):** it measures the mean power of the signal.

$$RMS = \sqrt{\frac{1}{N}\sum_{n=0}^{N}|x(n)|^2}$$

Where $x(n)$ is a sEMG signal value at the n-th sample n, and N is the total amount of samples. Both are amplitude estimators which increase their values when higher muscular fatigue and strength levels take place.

1.3 Average Threshold Crossing

It is usual to sample a signal periodically using an analog to digital converter (ADC) in order to digitalize the signal for post processing. However, there is another option, which has more benefits and is also amazingly easy to implement, it is known as **Event Driven** sampling. This technique, in comparison with the periodic sampling, only collects data when a specific and significant event has happened, like, for example, when a signal changes its value [25]. Moreover, this is widely used in biological systems.

The **Average Threshold Crossing (ATC)** is an event based sampling in which, the significant event is when the sEMG signal surpasses an specific threshold value, this is called *Threshold Crossing (TC)* event. This approach, as almost every event-based one, reduces the data amount in order to relieve the post-processing systems and in consequence, requires less electronics for that, making possible the decrement of the silicon chip area needed to the acquisition. So, in general, a circuit complexity minimisation is accomplished, which it is a fundamental constrain in any wearable device that is intended to work with batteries for allowing the user movement [26, 27].

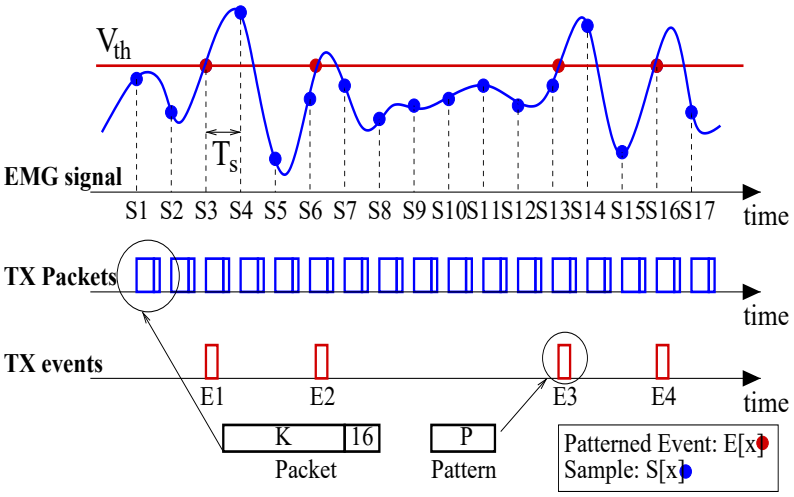


Figure 1.11: Comparison between a periodic sampling and the threshold crossing event-based method [27].

In the Figure 1.11, the packet transmission reduction with the event based sampling method, is demonstrated. For the time window selected, the EMG signal is sampled 17 times every fixed time interval (Ts) with the periodic approach, whereas TC event occurs only 4 times generating less data and shorter packets than the traditional method, where extra bits are required. As a consequence, low power

wireless communication protocols can take advantage of this approach in order to transmitted the data, and also more simple processing and minor workstation loads are needed, saving energy and earning simplicity.

In spite of all these advantages before declared, there are some drawbacks that have to be kept in mind [25, 29]. As it can be evaluated, this method achieves its accomplishments dropping information, that is not essential for certain applications but it can be highly meaningful for others where almost the total amount of the signal is demanded, like at medical diagnosis. In addition, one of the most important facts is that the theory, for some applications regarding this method, is not as large as its suggested potential.

1.4 Functional Electrical Stimulation

As discussed previously, our muscles are stimulated by signals coming from the nervous system. There are several pathologies that affect the performance of this neuromuscular system. The use of an electrical current to stimulated safely the excitable cells of our body in order to supplement or restore a lost of functionality, is referred as **Functional Electrical Stimulation (FES)** [33].

1.4.1 Classification

There are different techniques with their own disadvantages and advantages, and can be classified regarding to their access site, stimulation mode and its control system [30].

1. **Access site:**
    - **Peripheral motor nerve stimulation:** in this situation both, muscle surface and peripheral nerve are stimulated, but in general as the nerve threshold is lower than the muscular one, the stimulation is generated from the neuron, whether the entire neuromuscular junction and muscle structure are healthy. It also allows a wide and selective stimulation relying on the inter-electrode distance [33].
    - **Motor root stimulation:** the motoneurons in the spinal cord are stimulated with inner electrodes, a main concerning is the tiny space for electrodes placement and the low spread current requirement to avoid different root stimulation.
    - **Peripheral sensory nerve stimulation:** subcutaneous electrodes are implicated to obtain a sensorial response, as pain relief in cases where the nerve can be easily targeted.
    - **Implanted artificial sensors.**
2. **Stimulation mode:**
    - **Regular pulse trains:** commonly, current pulses are used to trigger our cells, with frequencies between 10 to 100 Hz and a pulse width range from 5 to 500 µs. Other parameters, such direct or capacitively coupled, mono or biphasic signals, and steady current or voltage can be chosen permitting us to control harmful effects on our cells. For reaching that objective, capacitive coupled and biphasic pulses give better charge balance for avoid tissue damage (see Figure 1.12). One main disadvantage is that real nerves do not fire steadily, they generate a burst first and after that reduce its firing rate, otherwise the fatigue arrives faster.

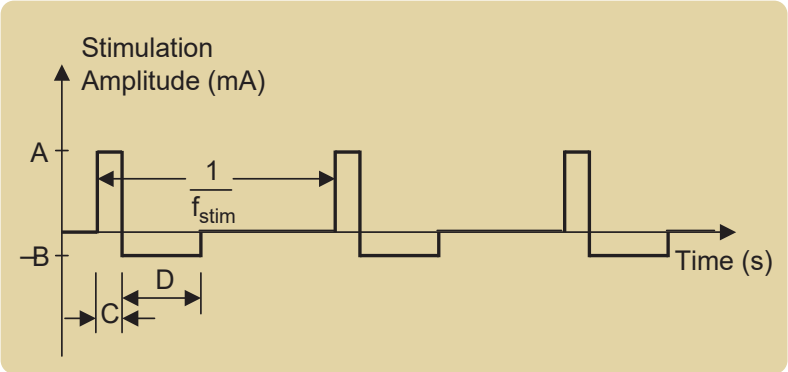


Figure 1.12: Example of a common stimulation pulse train [32], where the parameters to set are the positive and negative current amplitudes (A and B , respectively), the width of each pulse (C for the positive and D for the negative) and the stimulation frequency.

- **Patterned:** it consists in modify the firing rate dynamically inserting one (doublet) or two (triplet) more pulses before the next arrives, so, basically change the frequency in a single interval of the stimulation.
- **Anodal block:** in this situation an anode blocks a nerve fibre conduction hyperpolarizing it.

3. **Control Systems:** the two more used settings, in clinical applications [32], are:

   - **Open-loop:** the activation is set by the user, and continuously controlled by him.
   - **Finite-State:** is also an open loop configuration, but when a certain condition is fulfilled a stimulation sequence is started.

1.4.2 Traditional Stimulation

One specific application of FES is the stimulation through biphasic pulses, of the nerves to generate a muscle contraction in a impaired patient or also in some cases in which there is a muscular weakness after a surgery or injury, for rehabilitation [31]. It permits to prevent atrophy because of the utilization lack, pain due to body parts being steadily subdued to pressure (like a patient in a wheelchair), and also retrain the motor physiology to compensate motor flaws. Nevertheless, some limitations has to be discussed, such as the simultaneous recruitment of motor units, starting for the ones which are nearer to the muscle surface, whether the electrodes are placed in the skin.

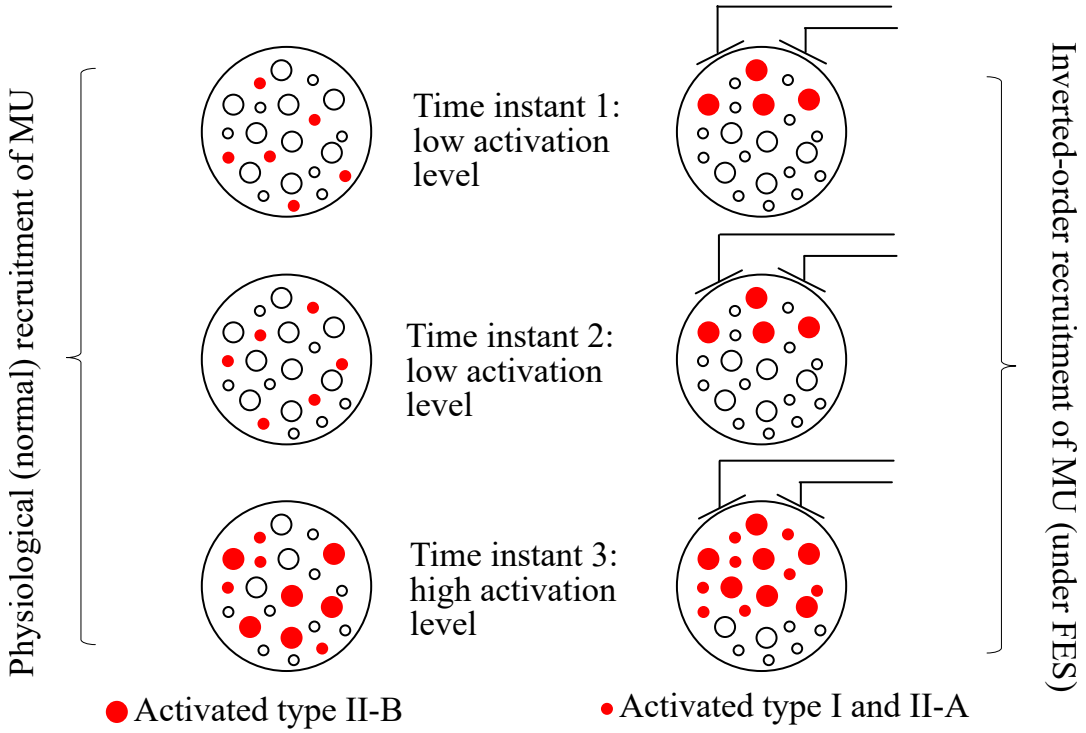


Figure 1.13: Comparison between a normal (healthy nervous system performed) and a FES stimulation [31].

Figure 1.13 shows how the nearer low threshold fibres are activated first and how, when incrementing the activation level, almost the entire bunch of fibres is triggered simultaneously, leaving inactivated the further ones. Furthermore, another superficial electrical stimulation drawback is the fact that, with the same setting, the fibres activated in each stimulus are the same, reaching the fatigue faster, whilst the normal activation is a regulated stimulation to avoid a sudden fatigue state.

Then, is important to know what consequences can occur regarding to the variation of stimulation frequencies [23].

- **Low frequency fatigue:** when applying current pulses in the range of 10 to 20 Hz, the reduction of the muscular force appears. This physiological procedure is due to a calcium diminution after a long stimulation period.
- **High frequency fatigue:** at the rates of 20 to 100 Hz a non physiological muscle debilitation occurs, due to failures at neurotransmitter diffusion and action potential nerve propagation.
- **Junction fatigue:** the neurotransmitter diminution takes place faster than in the past case, generating a coordination loss between stimulus and muscle contraction when using 100 to 1000 Hz frequencies.

- **Electrical Block:** at higher rates the fibres totally block after a short time.

To sum up, for this specific application, two main aspects separate the superficial FES stimulation from the natural one. Firstly, the constant activation of the same adjacent fibres, and secondly, the steady stimulation frequency. Both are the reason of an earlier muscle exhaustion.

1.5 Range of Motion

Motion is a central function for the human beings, and movement loss or efficiency reduction are a widely spread problem in the nowadays population [40].

In order to assess the movement that our body performs a parameter known as **Range of Motion (ROM)** can be used. It is the total motion of the body segments relative to one reference (see Figure 1.14) and it could be angular or linear. Those segments can constraint this range of motion, in some cases this can be physiological, but in others, pathological, where the movement need larger and faster actions and whit that, bigger ROMs that are being blocked for some reason [11].

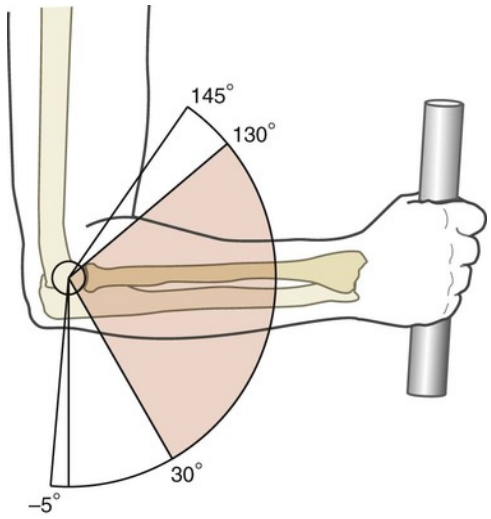


Figure 1.14: Elbow Flexion Angular Range of Motion. The normal and functional arcs are highlighted in red and white, respectively [39].

There are some many variables to be aware of, when measuring ROM, such as [40]:

- *Age:* the ROM values are considerably diverse at different stages of our lives. For instance, it is well-know that these values decrement for elderlies.
- *Gender:* at the same age, women tend to have more flexibility than men.
- *Body Mass Index:* for people with an excess of fat the ROM values are lower due to joint movement decrement.
- *Diseases:* like the ones that spoil or create inflammation at the joint.
- *Habituation:* when a person does an specific movement repeatedly the flexibility can be increased and with it the ROM values for that movement.

A principle related with ROM declares that when specific, slow and weak movements are perform the short ROMs are preferred, whilst larger ROMs are more fruitful in cases where the movements are faster and stronger, in other words in cases with bigger effort levels [11].

1.5.1 Sorts of Range of Motion

There are different techniques to evaluate the ROM in a patient and they are classify according to whether the movement execution is done entirely by the patient (active) or it needs some type of assistance whereas he is in a state of total relaxation (passive).

Hence, in the initial steps of an evaluation the *Active Range of Motion (AROM)* is assessed. This technique allows to observe movement limiting factors, such as ache or structural disturbs, which can be helpful to determine the patient condition. Then, when some motion limitation are discovered a *Passive Range of Motion (PROM)* can be performed in order to analysed the joint and muscle integrity, the joint excursion capacity and the degree of inflammation [11, 29].

1.5.2 A Measuring Device

In order to measure the joint capability we can use different kind of devices, one of the common ones is the **goniometer**, which is the most widely known gadget for performing PROM and AROM measurements. In general, it is made up of two mobile arms, which are attached in one end and have a scale to measure the angular difference between each other.

This device is easy to use, cheap and simple, but it has some drawbacks as well, all regarding with a loss of accuracy that could lead to misinterpretations in the medical analysis. This problem is cause due to inaccurate visual placement and unstable device holding, while the movement is being performed.

It is also possible to use a digital goniometer that can overcome some of the anterior problems, like the placement one, because it can set the zero degrees in any plane of interest. In addition, it could present the information in a screen making more suitable and accurate the data collection [42].

Another concern is the performance that these devices can reach in terms of how near is the measured value from the real one and how stable is the measurement. For this in [41], a goniometer performance, has been analysed, in contrast with an electromagnetic tracking system, which can also obtain measurements in the three spatial dimensions. It has been studied that, for certain movement configurations, like hip abduction and internal rotation both devices have similar outcomes, otherwise it is not the same when flexion, adduction or external rotation of the hip are performed.

In conclusion, it is important to recognise when the situation requires a more

sophisticated method, but in general the manual goniometer is good enough to remain as the first option for measuring ROMs at different joints.

1.6 Bluetooth Low Energy

Nowadays one of the most popular wireless communication techniques is the **Bluetooth**, which has been changed along the years to follow the technological advances. In 2010, the *Bluetooth Special Interest Group (SIG)* introduced variations in the **Protocol Stack**, a cluster of protocols which permits the data broadcast, creating the new **Bluetooth Low Energy (BLE)**. This standard is available since the version 4.0 of Bluetooth and a device having only Bluetooth Low Energy (BLE) support, can not communicate with an old Bluetooth device with a version prior to 4. Today, this feature enables a classification of current devices into two groups: the *Single Mode or Smart*, that possesses Bluetooth Low Energy (BLE) support but not the traditional one, and the *Dual Mode or Smart Ready*, which can use both standards [44, 43].

The Bluetooth Low Energy (BLE) Protocol Stack, is a combination of different **protocols**, that use **profiles** to communicate among them and, as shown in Figure 1.15, it is made up of three main blocks [45]:

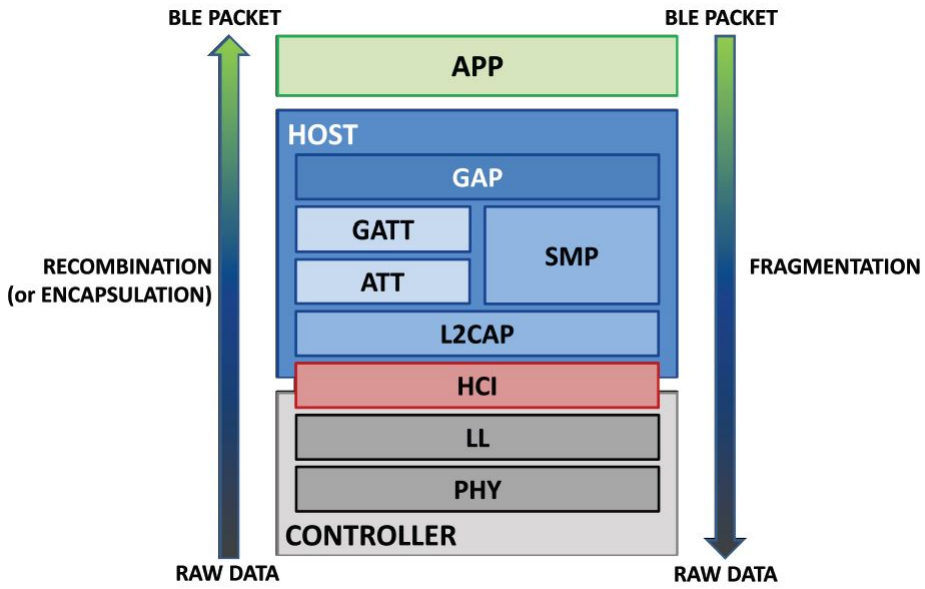


Figure 1.15: Bluetooth Low Energy Protocol Stack for a Single Mode device [45].

1. **The Application:** the superficial block defines the user interface and profiles, giving the capability to perform an intercommunication with Bluetooth Low Energy (BLE) devices from different manufacturers, and also it establishes some other custom-made profiles for specific implementations.

2. **The Host:** consists in a multilayer block, where every layer can be a protocol or a profile:

- *Generic Access Profile (GAP)*
- *Generic Attribute Profile (GATT)*
- *Logical Link Control and Adaptation Protocol (L2CAP)*
- *Attribute Protocol (ATT)*
- *Security Manager Protocol (SMP)*
- The Host side of the *Host Controller Interface (HCI)*

3. **The Controller:** as the Host is a collection of layers:
   - The Controller side of the *Host Controller Interface (HCI)*
   - *Link Layer (LL)*
   - *Physical Layer (PHY)*

The data caught for the antenna is encapsulated into BLE packets in order to be interpreted for the application block and also for the user, whereas the data that the user wants to communicate has to be fragmented in raw data again to be transmitted.

1.6.1 Physical Layer

It is the layer in which all the analog or raw data received, for the device, is converted to digital entities and obviously, all the electronics required for this scope, is contained in it.

It also manages to use the 2.4 GHz band, the Industrial, Scientific and Medical one, so it sends electromagnetic waves in the range of the radio-frequencies. In Figure 1.16, it is seen that this band is split in 40 channels where every channel is 2 MHz wide, those are divided into two groups: the three *advertising channels*, which are not consecutive in frequency, but they are in nomenclature (37, 38 and 39), and the second group of the other 37 *connection channels*, which are used once the link has been settled.

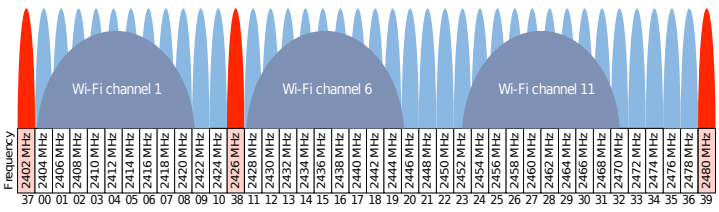


Figure 1.16: Bluetooth Low Energy communications channels [43], the advertising ones can be observed in red, while in blue it is shown the overlapping between the connection channels of the BLE and some channels for the Wi-Fi communications.

The Wi-Fi and the traditional Bluetooth communications employ bands overlapping this one, therefore, in order to avoid interference, this layer has to switch in a pseudo-random manner the communication channel, that is being utilised, through a standard called *frequency hopping spread spectrum* [44, 45].

1.6.2 Link Layer

A device in a BLE connection can be a **Master/Central** peer, which seeks for advertising packets in the air and it establishes a connection when an appropriate one is found, in addition it controls the timing and data transmission in the communication. On the other hand, a **Slave/Peripheral** device starts sending data packets with information regarding to the device and the linking modes, called advertising packets. The latter decides if a specific master can connect with it, and then, whether this slave device is handled for that master.

The Link Layer is communicated with the PHY layer and is in charge of defining the device role, that could be *advertiser*, *scanner*, *master* or *slave*.

A Basic Communication

The whole process starts when the advertising packets are sent at time intervals from 20 ms to 10.24 s, as can be perfectly intuit, while higher the packet frequency emission is, the power consumption is going to be more elevated too.

The scanner sets a *scan interval* and a *scan window*, which establish the rate and the actual period of time in which the scanner is going to search for advertising packets (see Figure 1.17). It is easy to realize that those parameters affect the power consumption because it is directly related with the amount of time that radio waves are working.

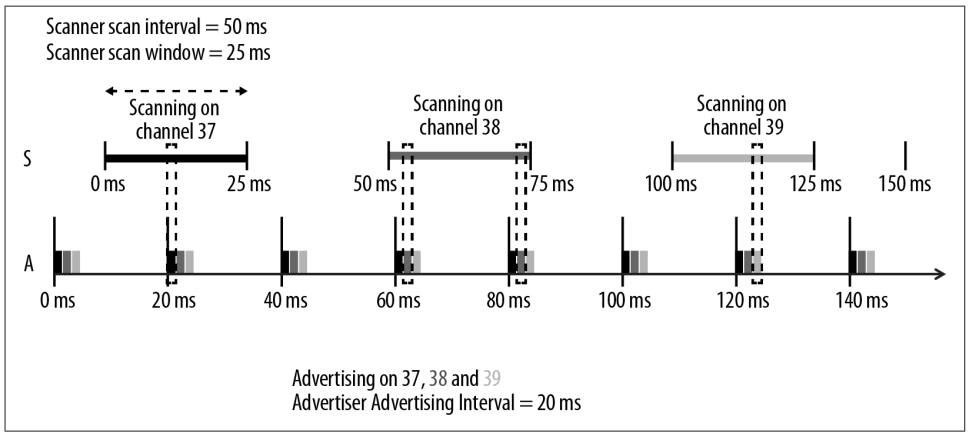


Figure 1.17: Bluetooth Low Energy scanning-advertising procedures at advertising channels, where A stands for advertiser, and S for scanner [44].

Once an advertising packet has been found, the scanner tries to initiate a connection, relying on the information that the answer packet from the advertiser has. This received data is related to the configuration of the future, or not, slave and its answer to the connection packet request, sent from the master; then, if the link is allowed it is established. The link permits the diffusion, through the air, of information in both directions, this defines a **connection event** which is the period of time in which the radio is on and a variable number of packets are being transmitted. Later, when the packet exchange is finished, the radio enters to an **idle state** for reaching a power consumption decrement (see Figure 1.18).

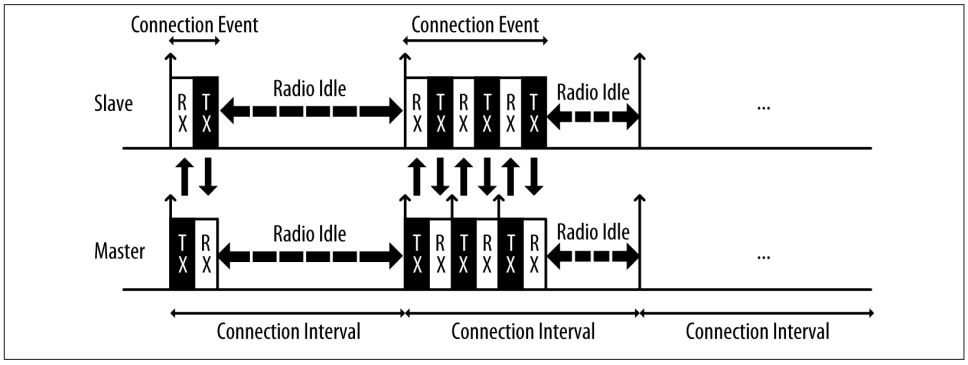


Figure 1.18: Bluetooth Low Energy, packet transmission when connected [44].

There are three important parameters that affect the connection and can be set by the LL:

1. **Connection interval:** it is the elapsed time from a connection event to the next one and it could be a value in the range of 7.5 ms to 4 s.

2. **Slave Latency:** it is the amount of connection events that the slave can not take into account without generate a link loss.

3. **Connection Supervision Timeout:** it is the maximum time that can elapse between two valid data reception without consider that a disconnection has happened.

Another feature that this layer can use is the **white list**, which is a list storing addresses of different devices of interest, this aids to any bluetooth device to filter the possible peers whit which they are going to interact. However, a limitation for the use of this is that previous knowledge of device addresses has to be provided.

1.6.3 Host Controller Interface

The HCI is a protocol in charge of the communication between the host and the controller, therefore, it defines some commands and events that allow this interaction.

1.6.4 Logical Link Control and Adaptation Protocol

This multiplexer protocol mainly occupation is to perform the fragmentation and recombination of the BLE packets and raw data, respectively. In other words it takes long packets and cut them in snippets of 27 bytes, which is the maximum length of BLE data packets, or takes these fragmented data to recombine it in larger packets. It also puts a header on every BLE packet that is 4 bytes long, shrinking the true data length in the packet to 23 bytes (in case the maximum has been used)

Moreover, it controls the ATT and SMP protocols that are discussed below.

1.6.5 Attribute Protocol

This protocol controls the communication between devices defining them as clients, servers or both, this classification is independent from their master or slave roles settled in the Link Layer (LL). Every device has a set of **attributes**, which are the smallest addressable data structures storing information and are define with the following elements:

- *Handle:* is a 16-bit number and it is used to address the attribute.
- *UUID:* is the universal unified identifier and gives information about the stored data and can be also utilised to handle the attribute. It is a 128-bit number possibly represented with a 16-bit (4 byte) one, in order to shrink data pay-load at the LL and increase the efficiency.
- *Permissions:* they establish what sorts of operations can be performed whit this piece of information and with which level of security.
- *Value:* is the actual value of the attribute.

Specifically the server device that, again can be master or slave, has information in its attributes that can be requested by the client. This one can implement mainly two operations which are reading or writing server attributes but this depends on the permissions that them own [44, 46].

1.6.6 Security Manager Protocol

This is a protocol that employs different algorithms to protect the data, that is being shared between the devices, from corruption and to generate a secure communication through an encrypted link. There are three procedures that are followed to comply this:

- *Pairing:* a temporary key is shared between the device to originate the secure link.
- *Bonding:* once the before link is set, the creation of a permanent key can occur to allow the secure connection every time the devices will be connected, saving time at the encrypted link generation.
- *Encryption Re-establishment:* here, the keys are stored in both devices and whit this, the use of them to re-create this encrypted link is defined.

It is possible to only set a pairing procedure between the devices without doing the other steps.

1.6.7 Generic Access Profile

It enables different peers to interact with each other defining these aspects: roles, modes, procedures and security.

Roles

The roles that a GAP can manage in a BLE device are four:

- *Broadcaster:* is a device that sends data through the advertising channels continuously and does not connect with other peers. The LL role is the advertiser.
- *Observer:* is complementary to the broadcaster and it is a device that is, permanently, ready to receive the advertising packets from a broadcaster. The LL is configured as a scanner in this case.
- *Peripheral:* corresponds to a LL set as a slave and, as said uses advertising packets to present themselves to central devices and once the connection has occurred it answers the central requirements.
- *Central:* is complementary to the peripheral and corresponds to the master definition in the LL. It is able to connect with different peripherals and is the procedure initiator because it starts the connection if it finds advertising packets coming from a properly settled peripheral.

Modes and Procedures

The roles can operated in different modes using distinct procedures to achieve a final outcome. For instance, there are some **discoverability modes**, that allow the peripheral to be totally or partially invisible to central devices, or the most common one, that is the *general discoverable mode*, which allows the peripheral to be found by central peers performing the counterpart mode. Also other modes can change the information sent at the advertising packets to avoid the connection, or allow a direct or an undirected link.

In contrast the procedures explain the sequence of actions performed to obtain these modes, one example can be the *general discovery procedure* in a central device, that starts the scanning freely without taking into account the addresses in the white list and evaluates every advertising packet found.

Security

The GAP defines some modes and procedures that are related with the SMP. Those express levels of security and, in addition, the GAP, can implement other features to increase the security of the communication, such as address types handling, authentication, among others.

1.6.8 Generic Attribute Profile

This profile is in charge of driving the actual data transmission procedures and formats, for that, it uses the ATT attributes where this information is stored. The GATT also defines roles, but this roles are independent from the ones assigned by the GAP.

The two roles are: as a **client** or as a **server**. The former does not know about the attribute content of the latter so it has to perform discoveries by sending requests. The server, instead of that, organizes the information and responds to client originated received packets.

The GATT uses an hierarchy, as show in Figure 1.19, to organize the attributes and with them the information. The **service** is a container of other smaller groups of attributes called **characteristics**, a GATT server can have several services and these can hold more than one characteristic, as well. Furthermore, these groups of information are related with two concepts to take into account, the *Declaration*, which is the single first attribute of the cluster (i.e., service or characteristic) and presents the metadata of it. The second concept is the *Definition*, which is the joint of the declaration with the extra attributes at the cluster [44, 45].

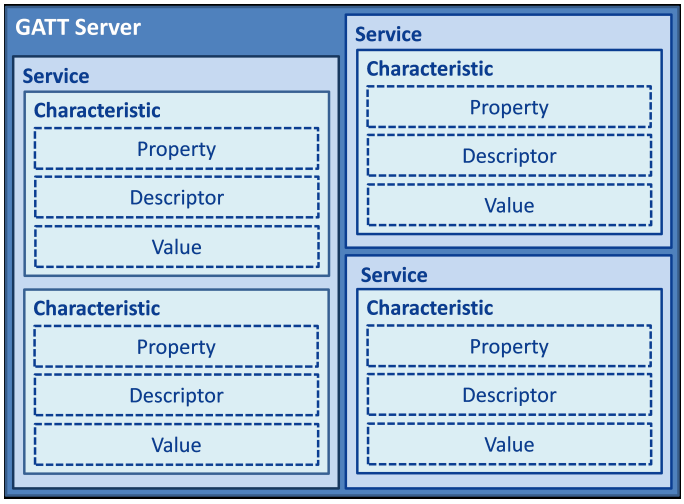


Figure 1.19: GATT Server Hierarchy example, it contents three services, one with two characteristics and the others with only one. The property attribute refers to the characteristic declaration, which contains metadata regarding to the characteristic permissions [45].

In the specific case of characteristics, their definition can also have another attribute which adds information to the one stored at the declaration. This single attribute is called *Descriptor* and there are different types of it, such as user description descriptor, extended properties descriptor or an outstanding one known as *Client Characteristic Configuration Descriptor (CCCD)*. The CCCD imitates a switch that enables or disables the server functions through which it communicates characteristic updates to the client. This means that it can turn on the transmission of *notifications* or *indications*, which differ in that the second ones require a packet reception confirmation from the client in order to continue the transmission, whilst when the notification is allowed, in a characteristic property and its CCCD is correctly set to 1, the modification of the attribute value leads to an update of it that is sent to the client, without any other intermediating packets [44, 46].

1.7 Embedded Systems

A computational embedded system is a combination of elements which interact among each other, belonging these to the hardware, software or the environment. The function of an embedded system is separated from traditional computers or supercomputers in its non computational scope. So, in other words, it can be thought as a computer system with an specific not computational task, which needs of the synergy of different other subsystems to fulfil that task [34].

Also the word embedded tells us that, in spite of the not computing scope, this system has inside a processing unit which controls it and, most of the time, the user does not realize about this computational presence.

These systems are characterized for having a physical boundary which can be divided in reaction and execution constraints. The former is related whit the real world interactions that are going to unleash some processes in the system, and, the latter are in relation whit the physical platform where the system works [36].

Embedded systems are presented in our dairy life (as can be seen in Table 1.1), and over the years they are evolving quickly.

Table 1.1: Examples of embedded systems in different markets [35].

Market	***Embedded Device***
Consumer Electronics	Television Automobiles Telephones, Cellphones
Industrial Control	Control Systems
Medical	Infusion Pumps Dialysis Machines Cardiac Monitors
Office Automation	Fax Machine Printers Monitors
Networking	Routers Gateways Hubs

According to [35], an embedded system can be represented with a modular model as shown in Figure 1.20. The hardware section, which involves all the physical components, is always present, whilst the other two, that have all the software to control the hardware, and do whatever is chased, in most of the cases, are present as well.

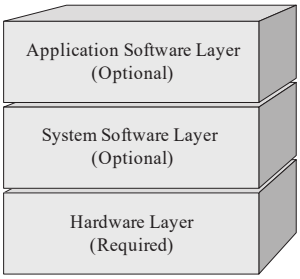


Figure 1.20: Embedded system model [35].

1.7.1 Common Features

Typically, as priorly mentioned, the distinct embedded systems use software and hardware elements, and also they share, in general, other characteristics, such as [34]:

- *Constrained Specification:* regarding to their design, most of the time they need to be small, with a low cost and in some of them, the reaction time due to environment fluctuations has to be fast enough to accomplish real-time tasks.
- *Reliability:* they have to work in an excellent way, without issues, in order to maintain a good performance in situations where a fault can be catastrophic, like in some life-support applications.
- *Network Processors:* multi-core utilization for controlling individual subsystems. Obviously this is need in complex cases, where also connection between the processes for sharing information is required, to work as an unified system.
- *Internet enabled:* for networking different systems, as it is pursued in some applications, as home appliances, office systems, factory control, medical, communications and automotive systems, among others.

1.7.2 Design

There are four sorts of design approaches that one person or team can follow when working whit an embedded system [35]:

1. **Big-bang model:** whether no plans or processes are consider.
2. **Code-and-fix model:** when the objectives are clear but no processes to follow are set before the beginning of the design step.
3. **Waterfall model:** the system is done in steps, and the outcome of those steps are related with the others.

4. **Spiral model:** is implemented as a process divided into steps, as before, and also information of each one is used as a feedback for the process.

The best are the last two that usually are used as a combination in successful projects.

Some characteristics allow us to measure and assess this design procedure, the most important are the following [34]:

- *Safety:* an unsafe system is an useless one, they can not damage users, the environment or even other systems. This is the most important metric.
- *Cost:* it includes the cost from the design process until the production for selling the product.
- *Time-to-market:* the entire time, from the development beginning, until the market availability of the product.
- *Size:* for example, in wearable elements where the size and weight are important, otherwise, it is going to generate discomfort in the user. This can be related to the software size that is measured in bytes and number of instructions, or to the hardware where the number of transistors could be an indicator.
- *Performance:* it is related with the time that system spends to finish a task, it could be indicated as elapsed time from the start to the finish of a task, or as instructions per unit of time regarding to the processor performance.
- *Power consumption:* it indicates the lifetime, and heating hazards. Therefore is an indicator of the reliability.
- *Design flexibility:* when a system can be changed partially to arrive to another prototype, with a little amount of resources, a good design flexibility is obtained.

In conclusion, the life-cycle of an embedded system generation can be split into four stages: architecture creation, implementation, testing and maintenance [35].

Hence, it is important to define the **architecture**, as an abstract map that does not have information about the implementation. It shows in a general way, how the different software or hardware elements work, changing their behaviour or interacting among them.

Another important concept is the **standard**, which establishes how elements have to be created and what other components to add, in order to reach a functional integration. There are thousands of standards, that can be classified as *market-specific*, *general purpose* or standards that belong to both categories, even the programming languages (like Java, C) are standards inside the general purpose class.

1.7.3 Challenges

The embedded systems started as systems less powerful than normal computers, the hardware was as poor as the software, which began being binary code and after assembler language. Also they were single task performing systems and were built from the scratch. Nevertheless, they have changed and now are actual complex systems where multi-processes are being implemented and the modelling with subsystems is used due to their increasing complexity. Also, these subsystems are communicating among each other with different standards which rises the understanding among them.

Summarising, some of the challenges in this area now and in the near future are the necessity of design them from the subsystems and the correct modelling, due to their recent complexity enhancement. The modelling has to be based in the requirements and the environmental and user-driven context. Moreover, the utilization of object-oriented approaches, to increment the product quality and give flexibility to the system, as well as, hardware tools, for debugging and performance evaluation, shall be considered priceless tools. Finally, the integrated circuit providers should include software tools and intellectual property blocks for making the system developer life easier, in other words, they have to become subsystem manufacturers, instead of, only silicon chips providers [37, 38].

Chapter 2

State of Art

This thesis project is a complex system that deals with distinct technologies. In order to give an overview of how these, or similar ones, have been implemented in different other projects, a separation in topics has been made in this chapter.

2.1 Wireless Real-Time Communication

Today an incredible development in biomedical devices is occurring, which in general need specific information that comes from sensors attached to, or near, the patient. In these health care applications this data can be delivered to everywhere with the aid of wireless technology. Nevertheless, some issues, such as the *throughput* and *latency*, can unleash data misinterpretations, that is why the IEEE (Istitute of Electrical and Electronics Engineers) has settled some parameters to be fulfilled (as can be seen in the Table 2.1). Also, protocols with a low energy consumption are required in certain wearable devices and, as the radio transmission is critical on this, is vital to assess accurately every existing option.

Table 2.1: IEEE latency and data rate required for biomedical signal wireless transmission [47].

Bio-medical Signal	***Data Rate***	***Latency***
Heart Rate	80-800 bit/s	<1 s
Blood Pressure	80-800 bit/s	<1 s
Respiration	50-120 bit/s	<300 ms
Electrocardiography	4 kbit/s per channel	<250 ms per channel
Electromyography	64 kbit/s per channel	<15.6 ms per channel
Electroencephalography	3 kbit/s per channel	<360 ms per channel

2.1.1 sEMG Data Transmission

Some different wireless tools have been tested to accomplish tasks similar to this project. In [47] and [48] the over the air communication of sEMG data has been assess through different protocols.

In the first one, the Table 2.2 shares information about different characteristics of the assessed protocols, being the BLE and the IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) the ones that waste less energy but, at the same time the ones which need more proximity between the devices, whereas the traditional Wi-Fi consumes so much more energy but allows data transmissions at longer distances. These values are theoretical, and obviously, they depend on the specific practical application, which in [47] was an 8 channel EMG system that requires a high bandwidth to transmit real-time data at rates of 24 kB/s for each channel. Then, due to this constraint in the throughput, in order to have an appropriated signal reconstruction the selected solution was the Wi-Fi method.

Table 2.2: Protocols characteristics [47].

Protocol	***6LoWPAN/Zigbee***	***Wi-Fi***	***BLE***
Radio Frequency	2.4 GHz	2.4 GHz	2.4 GHz
Bandwidth	250 kbit/s	100 Mbit/s	1 Mbit/s
Range (meters)	1-75	1-100 (typical)	1-100 (typical)
Peak Current	<15 mA	<300 mA	<15 mA
Standby Current	0.003 mA	20 mA	0.2 mA

On the other hand, in [48], an evaluation of the wireless protocols was performed keeping in mind the reduction of the throughput due to the specific application, so an efficiency value was calculated as the ratio between practical throughput and the theoretical one. As a result of this, BLE and Low Power Wi-Fi (LpWiFi) were selected to posterior tests because of the higher bandwidth of the latter and the excellent capacity of low power functionality and transmission feature of the former. Moreover, the wide compatibility of them with different platforms and single chip implementations in the market, were others characteristics for the selection. This application uses up to 32 EMG channels and the two wireless techniques were assessed with different BLE and LpWiFi modules, the conclusion were that the throughput-consumption trade-off were similar between the two approaches because the BLE unleashes lower consumption but also lower rates of data transmission whilst the LpWifi technique is totally the opposite, with a wider range of bandwidths but energy losses seven times bigger than BLE, so the decision mainly relies on the utilised system.

2.1.2 ATC Data Transmission

As has been said this project use the Average Threshold Crossing (ATC) approach and its wireless transmission implementation can vary from the prior ones, when the sEMG signal was transmitted, due to the data reduction and energy consumption that this technique creates.

It can be observed as in [26], a wireless transmission of ATC data was performed using asynchronous Impulse-Radio Ultra Wide Band (IR-UWB), the communication channel was evaluated with a single frequency signal. They tested different distances between transmitter and receiver with distinct frequencies, obtaining that an almost flawless transmission with less than a 0.1 percent of relative standard deviation was achieved, for a distance and frequencies under 27 cm and 300 kHz, respectively. Besides, for rates smaller than 3 MHz the relative standard deviation was still lower than 1%, for the same range of distances. However, for increasing the distance to approximately 2 m, a rise in the power consumption will be required. It is important to remember that the sEMG signal has a spectrum compressed within the range of a few Hz to 1 kHz so this is the range of interest in this application. Finally, the entire system consisting in transmission and reception consumes 15.49 mW.

Another case using BLE technology to link a sEMG system with a receiver was demonstrated in [49]. Two BLE commercial modules were used, the RN4020 BLE as transmitter and the CC2540-BLE-USB dongle as receiver (see Figure 2.1). Here, the chosen time window, in this event-driven method, was set to 130 ms to analyse 4 sEMG channels. It could be analysed that the power consumption for the complete 4 channel system was reduced by 14 percent, in comparison with the sEMG transmission of 1 sample, using the ATC instead of the sEMG. This is a consequence of the data reduction that allows to pass from a 1 kB/s rate in the sEMG case to a 28 B/s one in the ATC approach. In other words, a power consumption decrement can be shown when using an approximate rate of 7 Hz to send 4 channels ATC data represented in 1 B each, instead of sending sEMG samples from 1 channel stored in 1 B as well, but with a rate of 1 kHz.

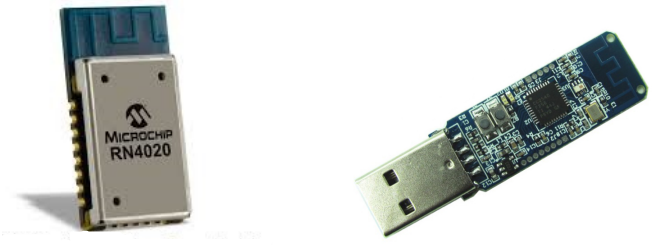


Figure 2.1: RN4020 BLE transmitter module (left) and CC2540 receiver dongle (right) [49].

2.2 ATC utilization on sEMG

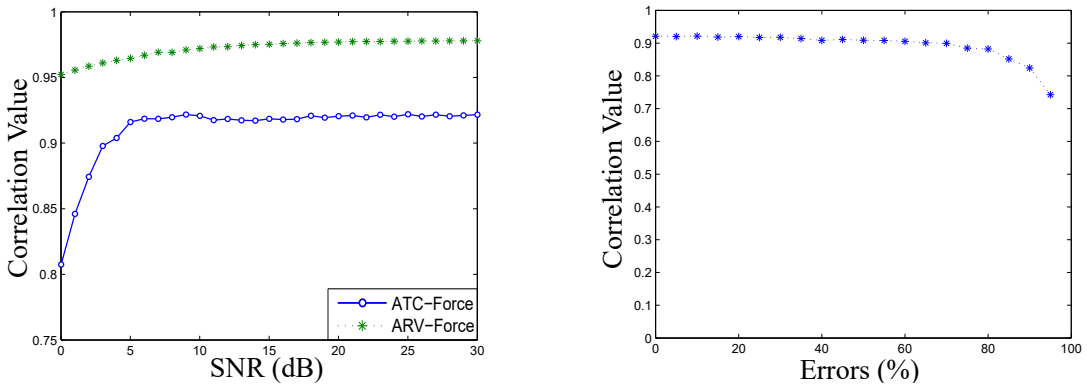
The surprising ATC technique, has been employed in some works for evaluating it at the sEMG field.

It has been demonstrated, in [27], the robustness of the ATC technique. Three methods were used in order to test that, where the ATC and the Average Rectified Value (ARV) correlations with the actual grasp force were compared. The reference for those test was measured using a dynamometer.

1. **Signal-Noise Ratio (SNR):** adding noise and generating different SNR has shown that with values higher than 5 dB the correlation of the ATC force is already good whereas the ARV value has a lower correlation and it is even more noise sensitive than ATC (as shown in Figure 2.2a).

2. **Amplifiers effect:** different grades of non-linearities were also tested and the method does not show a throughput detriment regarding to amplifiers distortions.

3. **Event Losses:** some ATC events were deliberately dropped to simulate packet losses, the correlation does not suffer big decrements, until 7 of 10 packets were lost (see Figure 2.2b).

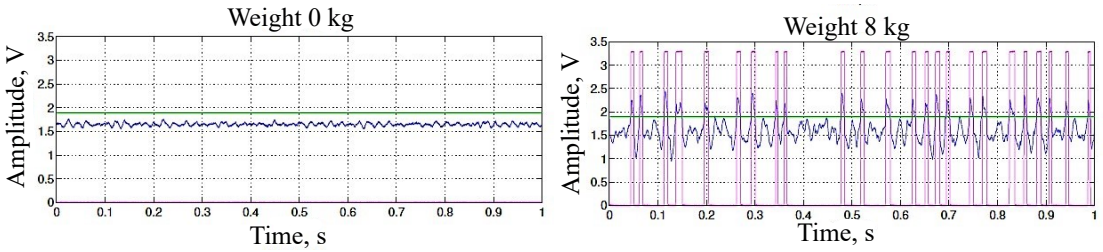


(a) (b)

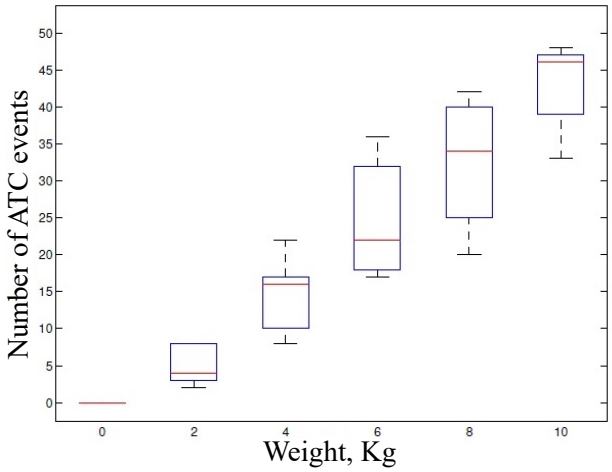
Figure 2.2: ATC force correlation related to SNR (a) and event losses (b) [27].

Also, in [28] the muscle force correlation with the ATC was also evaluated through isometric contractions using distinct weights. In this experimentation, the *Biceps Brachii* and the *Vastus Lateralis* were loaded with 5 different weights, the ATC events were counted and compared with the estimations of other parameters, such as the MNF, MDF, RMS, the total power and the peak to peak amplitude.

In Figure 2.3 it could be observed how the ATC events increase with the weight load, also the ATC median correlation, in comparison with the others previously presented indexes, has been shown as a better parameter to measure muscle force.



(a) The following signals are present: Threshold value (**green**), sEMG signal (**blue**) and ATC events (**purple**).



(b) Boxplots of the values obtained in the different force levels are presented.

Figure 2.3: The *Biceps Brachii* ATC events increment, in an isometric contraction, can be seen while the weight also rises [28].

In spite of all the advantages before declared, there are some limitations that have to be considered when using ATC with sEMG signals [28, 26]. One of them, is that it can not separate the five levels of force, flawlessly, although it can discriminate strength stages that have a difference of 6 kg at least, without any error. In order to improve that, the threshold can be modified but the technique has a trade-off regarding to its selection, because when this is set to a lower value more TC events are acquired in response to a contraction, but also, this rise at the events collection, can be a consequence of out-of-band or noisy disturbs. On the other hand, rising the threshold value will lead to the opposite situation where noisy events are reduced, but also, are the in-band events, obtaining a performance decrement in the strength judgement.

2.3 FES controlled by sEMG signal

As previously said, it has been proved that the FES technique can be used at rehabilitation of mainly, two pathologies, the stroke and the SCI, or more in general, to give a partial or complete support in a big amount of cases in which the muscle physiology is deteriorated.

Its control has been changed along the years, starting being manual, through the action of pushing a button to activate the electrical stimulation of the lower limb to support the standing. Another impractical approach, is the use of signals from the neurons to perform the stimulation [51]. The most effective and accurate control is based on the sEMG signal (see Figure 2.4), which can be perform in three different ways, according to [31]:

- *Biomimetic:* the FES patterns are regulated by the sEMG signal taken from a healthy subject.

- *EMG-triggered FES:* when the sEMG from the patient crosses an specific value a certain FES pattern is started.

- *EMG-driven FES:* the sEMG signal controls the FES stimulation changing its parameters, the most common action is the modification of the stimulation intensity in order to support the muscular contraction.

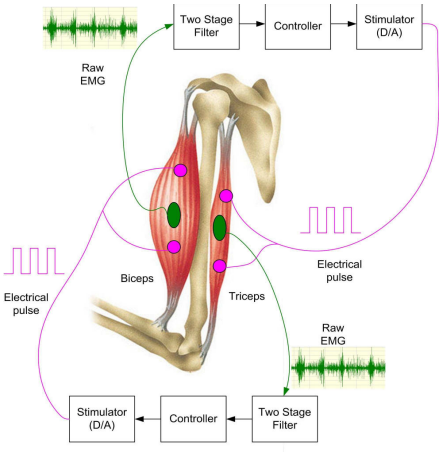


Figure 2.4: FES stimulation controlled by sEMG signal from the antagonist muscle. A double filtering is performed, the first is for removing sEMG artefacts generated because of the FES and the second stage is for removing a pathological tremor (as propose in [50]).

2.3.1 Triggered/Driven Implementation

In the FES triggered/driven by the EMG, some issues are generated due to the fact that the sEMG and FES electrodes are contemporarily placed in the same muscle. One of them is the **M-wave Artefact** generation, which is an EMG signal thousand times bigger than the actual and healthy one, caused by a synchronic activation of a bigger number of motor units [52]. There is also a **Stimulation Artefact (SA)**, that disturbs the sEMG measurements, created by the direct action of the stimulation pulses that generate electrical fields in the tissue, these perturbations can spoil the sEMG channel amplifiers or simply saturate the measurement.

Some techniques have been used in order to minimize this two perturbations, in [50], the SA is eliminated through a software technique called blocking (blanking) window and for the M-wave a type of Infinite Impulse Response (IIR) filter, referred as "comb filter" has been used. Also in order to deal with the SA the sEMG electrodes can not be the same as the FES, and the muscular electrical signal measurement can be stopped whenever a pulse is induced in the pertinent muscle [31].

2.3.2 Biomimetic Implementation

Totally different approaches exist, where the before named problems do not exist because the supporting FES is evoked by sEMG signals which are not from the same muscle. In [51], lower limb muscles are controlled by the sEMG signal from the ones of the upper-trunk, because it has been observed that some of them, such as the pectoralis major, trapezius and deltoids, suffer a contraction pattern while standing and sitting that is totally different from the patterns generated in other movements.

Another particular case, of importance for this project, is when the muscle is stimulated by the sEMG signal arriving from the contralateral healthy muscle, this is called **Contralaterally Controlled Functional Electrical Stimulation (CCFES)**. As can be intuited this method is useful in patient with hemiplegic stroke, where one of the two limbs is undamaged. According to [53], such a system has to use more sensors to provide some inertial information, which added to the sEMG data is going to improve the decision making process for the stimulation pattern generation.

Moreover, a therapist can afford the sEMG signals which are going to regulated the FES to aid in the rehabilitation process (as shown in Figure 2.5), for instance in [54], ideal EMG patterns are taken from a multi-channel device connected to the therapist. These signals are sent as an input to a machine learning software which evaluates them and the joint movement, to decide the right electrical stimulation pattern. Here, the transmission of information has been verified.

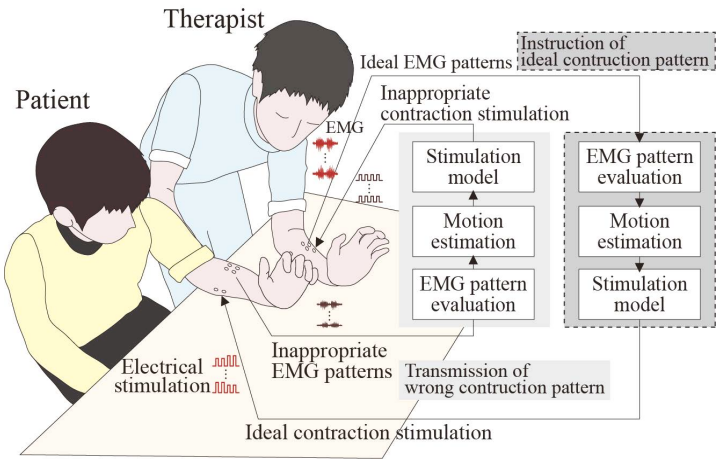


Figure 2.5: Example of direct rehabilitation using sEMG signal coming from the therapist in order to stimulate the patient [54].

2.4 FES managed by ATC signal

In a precedent work [23], using the ATC signal of sEMG channels, an electrical stimulator was commanded. The interfacing between those two elements was reached with the help of MATLAB® and Simulink®, both very complex and robust software.

For the FES control, a Simulink®model, provided by the manufactured, was implemented. This allows to combined together, that and the rest of the system required functionalities, coded in Matlab. Once the stimulation is running, the FES model has an inner control that pauses the stimulation during a certain amount of time, in order to catch and evaluate the new received ATC data, which might unleash a modification of the stimulation parameters, specifically, the current amplitude. In addition, this idle time, ought be employed to send data for plotting and give a feedback to the user. In conclusion, there is a clock that controls the stimulation parameters update rate.

Some concepts need to be clarified for understanding the outcomes of this project:

1. *Total processing time:* is the elapsed time between the initiation of a stimulation to the beginning of a next one using new ATC data, i.e., the sum of the parameters update period (T_{pu}), in which the stimulation is happening, and the time for processing the new fetched information, known as *Elaboration time* (see Figure 2.6).

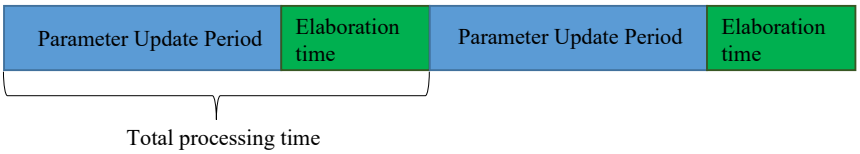


Figure 2.6: Main processes performed at the stimulation in [23].

2. *True Active Stimulation (TAS):* is the ratio between the mean elaboration time and T_{pu}. This value should be at least much lower than 0.5, meaning that the system owns a fast data elaboration, resulting in short stimulation stops for parameter updates.

$$TAS = \frac{\bar{T}_e}{T_{pu}}$$

Also the parameters update period, in that project is define as:

$$T_{pu} = TIA * n_{packet}$$

Where n_{packet} is the processing required number of ATC packets and TIA is the *Time Interval Acquisition*, which is a constant of 130 ms.

As it is possibly to see, several ATC packets might be utilised for generating a new current value, this occurs because the elaboration time might be widely variable relying on the number of active channels, as a consequence, for achieving a lower TAS, the system waits for 3 packets, T_{pu} equals to 390 ms, and based on that information calculates the new current value. The limitation of this technique is that a higher idle state destroys the real time application, making the current wave less accurate when rapid changes in the muscular strength occurred, as in motion, due to a slow modification of the FES parameters that causes a roughly variation of the amplitude stimulation value.

As a real time application is pursued, several tests were performed to assess the system efficiency using a collecting data time window of 3 min and a computer owning a dual-core 2.50 GHz i7-6500U CPU (with a 3.1 GHz overclocking maximum), 8 GB RAM and Windows 10 as operative system.

Table 2.3: Results of ATC data elaboration using Matlab and Simulink [23]. Pulse resolution refers at how many stimulation pulses are sent with a past configuration before a new one can be settled.

Test	Period (ms)	Elab. time (ms)	Tot. time (ms)	TAS[1]	Pr[2,3]
n.1 - 1 Ch	130 (1 ITA)	40(mean) 160(max)	170(mean) 290(max)	0.19	3-4
n.2 - 2 Ch	390 (3 ITA)	160(mean) 310(max)	550(mean) 700(max)	0.41	11-12
n.3 - 3 Ch	390 (3 ITA)	160(mean) 300(max)	550(mean) 690(max)	0.40	11-12

[1]True Active Stimulation [2]Pulse resolution [3]Stimulation frequency:30 Hz

The Table 2.3 compiles the three main tests. The first one, with one active channel can utilise a parameter update period of 130 ms, which is the minimum, and allows, as mentioned, a better current modulation. On the other hand, the application of several channels deteriorates the data elaboration time, then a higher T_{pu} shall be used and the application acquires a delay at the total processing time, which arrives till 10 s after 1 min stimulation.

Chapter 3

System Description

The system is basically composed for 3 main parts, as can been observed in Figure 3.1.

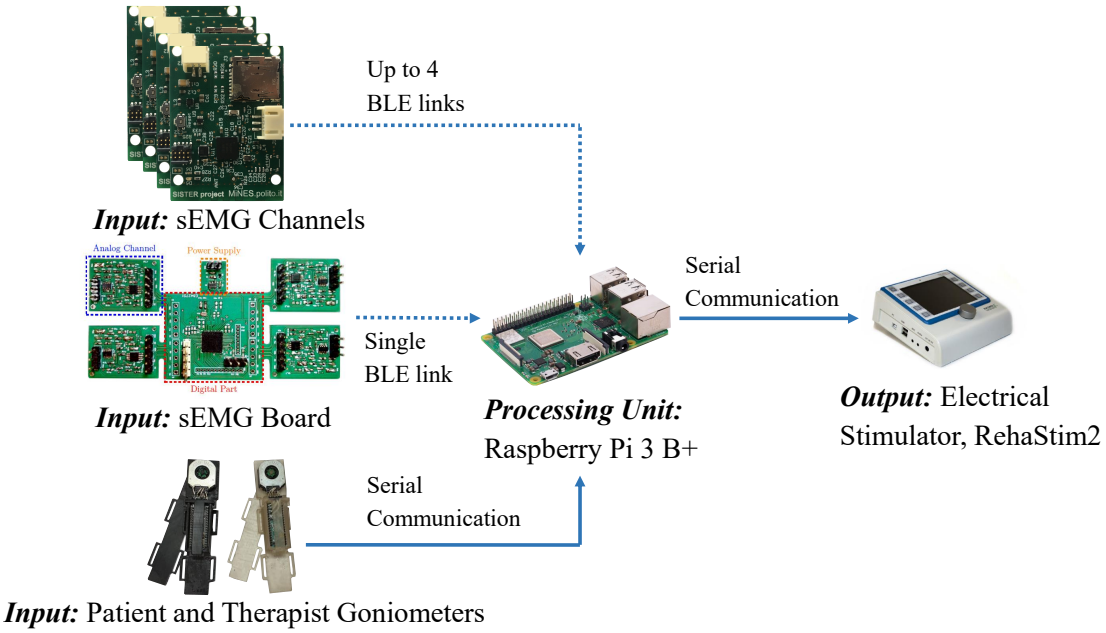


Figure 3.1: Diagram of the entire system.

- **Input:** it collects information from the environment, about the patient and therapist ROMs, and therapist , or even patient healthy, muscular strength. This is accomplished by the **Goniometers** and the **sEMG Channels**, respectively.
- **Processing Unit:** it receives and analyses the data from the inputs, in order to generate a respond, which is going to be sent as an output. This unit is intended to work in real time, be simple and low-cost, due to that a **Raspberry**

Pi was selected to perform this task.

- **Output:** currently it is only made up of an **Electrical Stimulator**, which acts only upon the patient and in function of information coming from the processing unit.

This system was thought to have two different configurations, the first one which employs CCFES, where the patient uses in a healthy limb, the acquisition channels and one goniometer to stimulate, from the data provided by them, the altered contralateral limb. So the stimulation electrodes and the second goniometer are placed in the pathological zone. On the other hand, it is also common to stimulate the patient based on a muscular pattern arriving from the therapist, so the sEMG channels and one goniometer are this time, attached to the professional.

In the following sections of this chapter a brief introduction, for each device this system uses, is made.

3.1 sEMG acquisition system

In this thesis two main approaches were used in order to collect the muscles data, the first one was a sEMG acquisition board previously developed in two precedent works [55, 23], and a second one, which is the result of a Master Thesis that is being done, contemporaneously at this one, for the Sister Project.

Both were implemented with almost equals acquisition channels, meaning this, with a similar circuital architecture for performing data acquisition. As can be seen in Figure 3.2, the significant elements of a single channel are [55]:

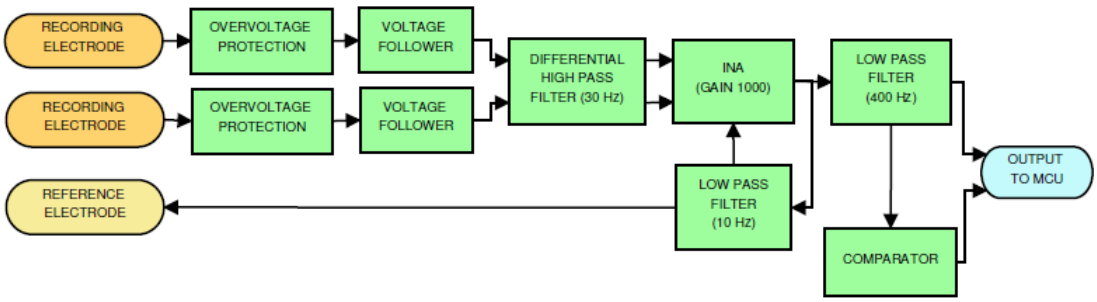


Figure 3.2: Single channel block diagram [55].

- **Over-voltage Protections:** the FES can generate extremely out of range tensions or even negative ones, which could damage the entire circuit. In order to avoid the potential hazards of this application every channel is made up of an over-voltage protection. A combination of common and Zener diodes, and ferrites were used to fulfil this task.
- **Voltage Followers:** this is a typical circuit, implemented using operational amplifiers in a closed-loop configuration, with a gain equals to 1. It provides isolation between the two section of the circuit, this element is in contact with, in other words, the high input and low output impedances decouple the operational amplifier output and input between them.
- **Differential High Pass Filter:** it is an specific model, which is implemented with passive components only (resistors and capacitors, in this case). It eliminates contributions under 33.86 Hz, which is its cutoff frequency. In general, at that low side of the spectrum motion artefacts occur, as previously stated in Section 1.2.3.
- **Instrumentation Amplifier (INA):** as said before, the signal needs to be amplified, for making it more suitable for the assessment and processing. This is performed by the INA, which is a combination of operational amplifiers with passive components. It is characterised by its high input and low output

impedance and a steady gain. The latter has been settled to 922 V/V for obtaining signals in the scale of V, instead of only mV.

- **Low Pass Filter (10 Hz):** the filtering of almost all the signal coming from the INA is done, leaving almost only the direct current. This filter gives a reference to the INA, as a negative feedback, for avoiding low frequency noise caused by it.

- **Low Pass Filter (400 Hz):** this, as the prior one, is an active filter, meaning that it uses active components in its configuration. It deletes all the contributions outside the upper limit of the sEMG spectrum. Moreover, this element has a gain that is set to 1 choosing the correct resistance values, surrounded it.

- **Voltage Comparator:** as this application needs to perform a digitalization, through the TC event-driven method, of the data, a comparator is required. It compares its two inputs continuously and changes its output to a high or low voltage level when the sEMG signal is greater or lower than the threshold, respectively. Then, this new digital signal is sent to the board microcontroller to be processed.

In the following a brief explanation of the two approaches are going to be presented, taking into account mainly, the distinguished differences between the two architectures.

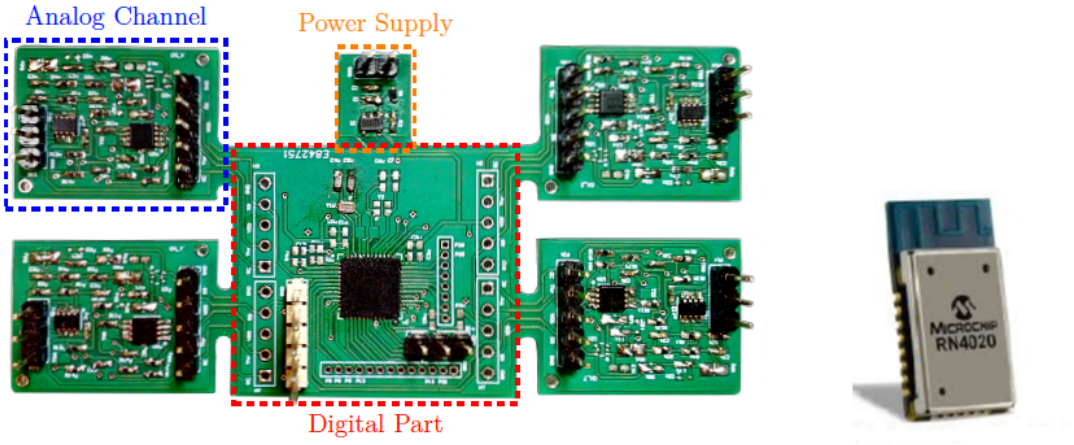
3.1.1 Acquisition Board

This system, at the moment, is made up of four main parts, the first three are all include in the same Printed Circuit Board (PCB), which are the power supply and the analog and digital parts, being the latter connected to the fourth one, a Bluetooth low energy module (see Figure 3.3).

The analog section contains four channels, with the architecture antecedently shown, each of them can be bounded or not to the body of the PCB.

The digital part consists, mainly, in the MSP430FR5969 microcontroller manufactured by Texas Instrument, which performs the threshold setting, the control of the BLE module and the acquisition of sEMG and ATC data. It is important to emphasise, that this implementation only allows to obtain a single threshold value for all the channels, at the same time.

The power supply has a voltage linear regulator in order to enable the utilisation of a commercial battery of 3.6 V, to power the different components in the PCB that requires a 3.3 V input. The battery isolates the circuit from 50/60 Hz noise, a relevant fact knowing that the channels do not contain a specific filter (Notch Filter) to clean interferences in that range, because these frequencies are inside the



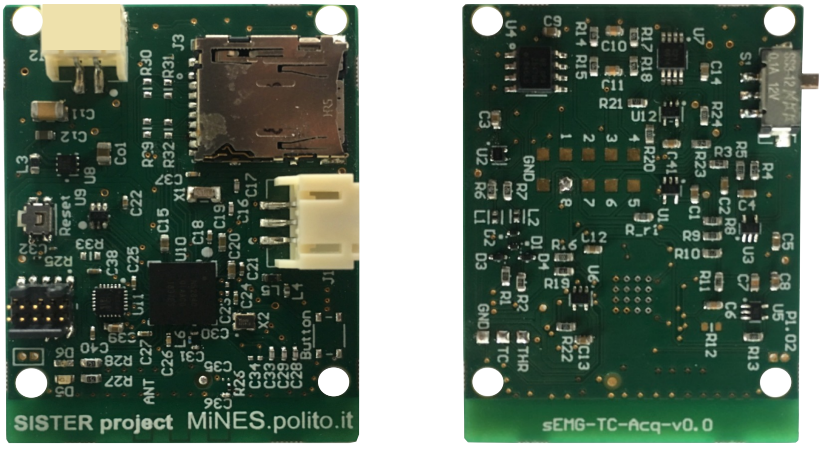
(a) Analog, digital and power supply sections. (b) BLE Server.

Figure 3.3: Four-Channel Acquisition Board [23, 49].

EMG signal spectrum. Furthermore, the regulator provides a steady voltage for feeding the entire circuit.

The BLE module implemented as a server/slave, to communicate data from the board, is the RN4020 produced by Microchip. This device has been configured in order to be capable of transmitting data packets with the four ATC values together, or information of one sEMG signal at the time [23].

3.1.2 Individual Channels



(a) Top Layer. (b) Bottom Layer.

Figure 3.4: Single Acquisition Channel.

Another approach is the latest implementation of single channels, each one with their own microcontroller as shown in Figure 3.4.

In this PCB the bottom layer consists in the same sEMG channel presented priorly, whereas the top layer implements other different features, being the most important the new technology of system on chip microcontroller, from Nordic Semiconductors, the NRF52840. This technology supports the use of different wireless communication protocols, such as BLE, Thread, ANT, among others.

The main difference with the previous approach is that every channel can implement an individual threshold and manage its own communication as a slave/server peer. This achievement is reached by the presence of the NRF52840 and also, a Digital to Analog Converter (DAC) for each channel, because the lack of it within the microcontroller. The DAC is handled by the latter through the Serial Peripheral Interface (SPI), allowing the generation of a fixed voltage level that works as the channel threshold.

Besides, some other hardware components have been added, such as two diode leds, manipulated by the microcontroller, and some other elements that do not have relevance in this application, such as an accelerometer and a SD card.

The firmware of this channel was developed in this thesis an it is explained at the chapter 4.

3.2 Functional Electrical Stimulator

One of the critical stages of this system is the electrical stimulation, where a certificated medical device has been chosen, the **RehaStim2** which is presented in Figure 3.5. This device is manufactured by HASOMED GmbH, a German company which has certified the product following the EN 60601-1 and EN 60601-2-10 international standards.

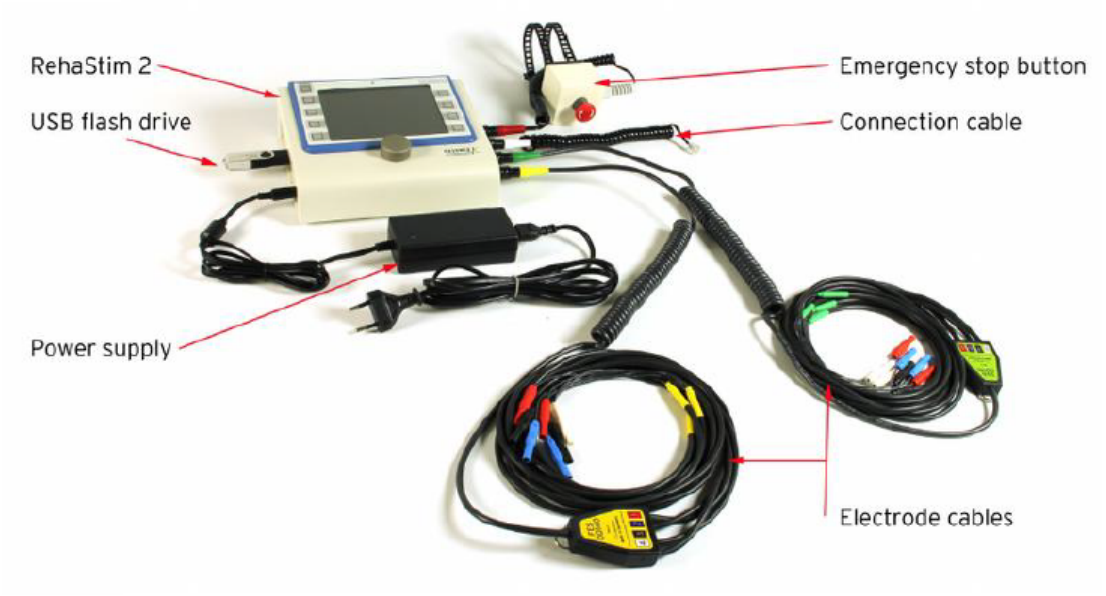


Figure 3.5: RehaStim device with all its accessories [56].

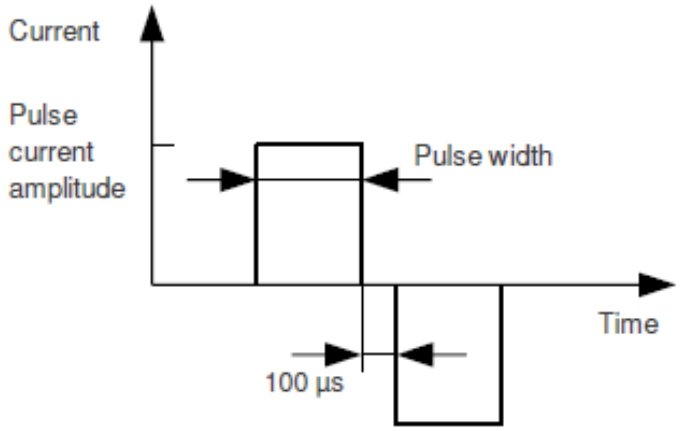
This is a portable stimulator that can reach until 8 contemporaneously stimulating channels, using surface electrodes, and also can be control externally to generate complex patterns of stimulation, this is the reason why it is perfect for research projects [57]. Besides, in this work the ScienceMode2 communication protocol has been utilised, having this three stimulation modes that can be performed:

1. **Continuous Channel List Mode (CCLM):** in this mode the enabled channels are generating the current stimulation steadily (an example is shown in Figure 3.6b).

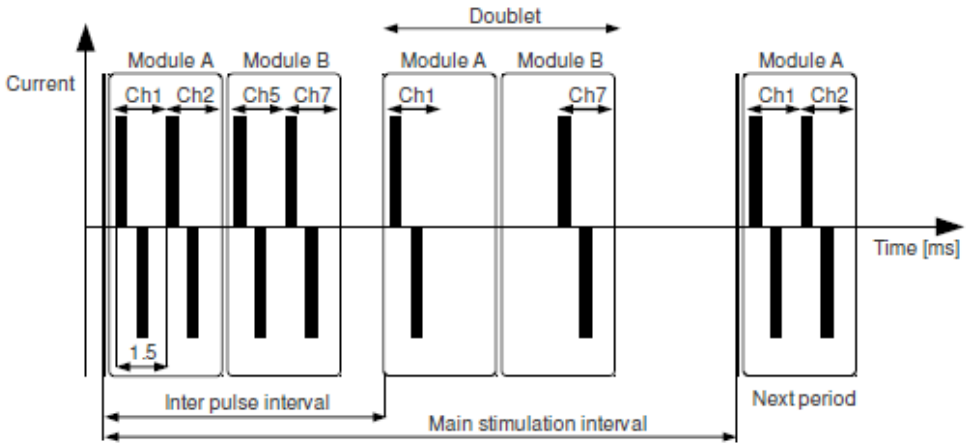
2. **One Shot Channel List Mode (OSCLM):** on the other hand, in this mode the channels evoke the pattern only once and waits to another start command to be received.

3. **Single Pulse (SP):** with this setting the stimulator only creates one single current stimulus in an unique selected channel.

There are some parameters that have to be taken into account in order to set a correct stimulation pattern [58], and others that are fixed characteristics of the apparatus, such as (see Figure 3.6):



(a) Pulse parameters in the RehaStim.



(b) Example of CCLM stimulation, where channels 1 and 7 have a settled impulse mode with doublets, in contrast with channels 2 and 5 that use only single pulses. In addition, Module A and B refers to the two internal modules that generate the current in the first and the second group of four channels, respectively.

Figure 3.6: Device signal features [58].

- **Current:** is the rectangular pulse amplitude and can be an integer value from 0 to 130 mA.
- **Pulse Width:** is the time at which the signal has an amplitude that matches with the selected current value, and it is in the range of [0, 500] µs.

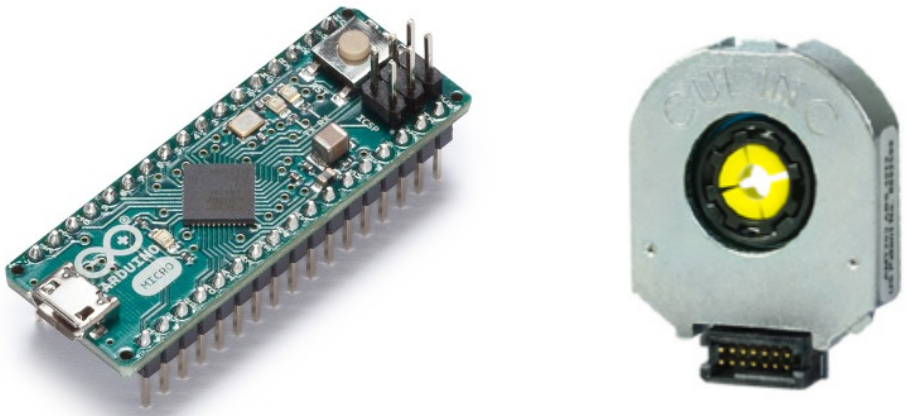
- **Inter-phase pause:** there is a zero current period, between the anodic and cathodic pulses of the wave, of 100 µs.
- **Wave shape:** is a signal composed of biphasic rectangular impulses, which allow the current balance at the tissue.
- **Mode:** it is possible to choose three impulse mode, the simplest one is the *single pulse* (this is not related with the SP of the stimulation mode), where every biphasic signal has a period defined by the main stimulation interval. Then, the *doublet* in which the pulse is repeated one time and in consequence the frequency is duplicated. Finally in the *triplet*, three pulses are performed during the main stimulation interval, unleashing a threefold frequency increment. It is important to keep in mind, that these modes are individual for each selected channel and have sense in the CCLM and OSCLM stimulation modes, while in the SP they are not applied.
- **Inter Pulse interval (t_2):** is the period of time, elapsed from one pulse in a channel to another in the same channel, and it is fixed for all the eight channels. The range for this parameter is from 8 ms to 129 ms in the ScienceMode2 protocol.
- **Main stimulation interval (t_1):** is the main period of the stimulation, in which all the pulses for all the channels are performed, so as a rule the $t_1 \geqslant t_2 * n$, where n is the maximum of the impulse modes in the chosen channels, being 1 for the single pulse, 2 for the doublet and 3 for the triplet. For example if channel 1, 2 and 7 are being used and the 2 uses triplets and the other two only single pulse, then n is equal to 3.
- **Low Frequency Factor:** it refers to the times that the stimulation is skipped, it is a fixed value for the entire amount of channels settled as low frequency in each continuous stimulation. Moreover, it can vary from 0 to 7.
- **Low Frequency Channels:** it defines the channels that are going to work with a lower frequency, relying on the low frequency factor.
- **Channel Execution:** it defines the fixed time of 1.5 ms in which the pulses are implemented.

3.3 Goniometers

The system currently possesses two articular goniometers, built previously in [29], one is intended to be used for the therapist and the other for the patient, in order to evaluated a correlation between ROM values arriving from both subjects.

The devices are the same, they are made up of an encoder, to measure the angle between the fixed segment and the mobile one, and an Arduino micro, which is in charge of data processing for this subsystem (see Figure 3.7).

In addition, they have some noteworthy features to notice, as their lightness, which is enough to be wearable, its resolution of 0.2°, due to the 12 bit encoder, and also, its high data sensing rate. Finally, they set their zero value wherever the initial relative position between the two axes is.



(a) Arduino Micro (b) ATM electromechanical encoder.

Figure 3.7: Goniometers Hardware [59, 29].

3.4 Raspberry Pi

The Raspberry Pi is an affordable *Single Board Computer (SBC)*, as can be observed in Figure 3.8, developed by the Raspberry Pi Foundation. The SBC classification, means that it has a microprocessor, memory and input/output hardware all together in an unique Printed Circuit Board (PCB). In this thesis the model 3 B+ was used, being this, at the moment, one of the newest ones. Some main features of it are enumerate below [60].

1. It possesses a Broadcom BCM2837B0 Cortex-A53 (ARMv8) 64 bit processor with a 1.4 GHz of clock frequency.
2. RAM memory of 1 GB.
3. Bluetooth 4.2, which Bluetooth Low Energy capabilities.
4. 4 USB 2.0 ports and full size HDMI.
5. Extended 40-pin GPIO header and more.

Moreover, it can be plugged to the rest of the hardware needed to obtain an inexpensive normal computer or any other desired system. It is important to notice that, in comparison with the other regular personal computers the Raspberry Pi has less resources in terms of the processor power and dynamic memory, but when those are accurately managed, they should be enough for a wide range on applications.

Regarding to the software, it could support different operating systems, for instance, Ubuntu Mate, Windows 10 IoT and so on. However the official one is Raspbian, a free operating system based on Debian, which has been developed to leverage the hardware of the board [61].

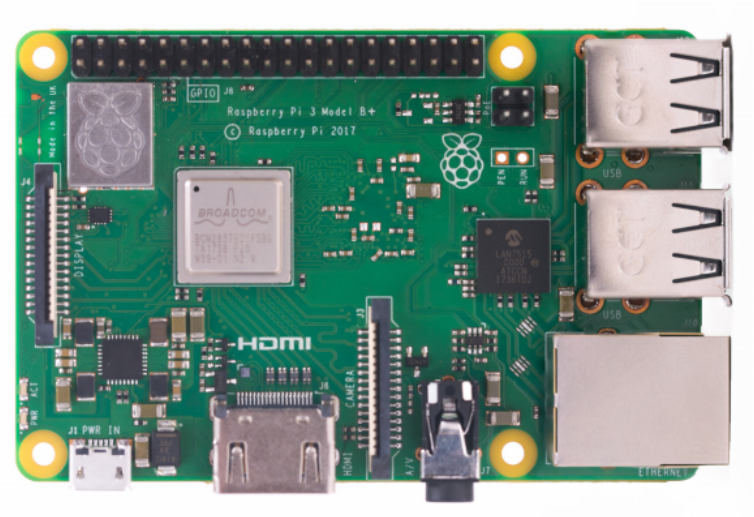


Figure 3.8: Raspberry Pi top view [60].

Chapter 4

Individual Channels Firmware

In the approach using individual channels, the firmware of the NRF52840 was coded in C language with Segger Embedded Studio (SES), an Integrated Development Environment (IDE) that supports Nordic Semiconductor®devices.

As the older approach, the sEMG signal is compared with a threshold generated by the microcontroller and, the total amount of TC events occurred in a specific time window, is the ATC value to be sent to the central device, in order to process it and execute a desired response.

There are many wireless protocols but the BLE is widely used in different fields, like at fitness applications. In this case, transmitted data is long only **2** B and the required transmission frequency is not higher than **8** Hz because of the digitalisation, therefore this specific low energy communication protocol fulfils greatly the requirements for this application.

4.1 BLE Implementation

Nordic Semiconductors®provides different software layers that allows the developer to create an application more easily, as show in Figure 4.1, the *Software Development Kit (SDK)*, *Drivers* and the *SoftDevice* are utilised to interface the developer application and the physical **System on Chip (SoC)**.

The SDK is a tool providing examples and firmware modules to assist in the interfacing of the SoftDevice and drivers with the user application, whilst the SoftDevice, basically, is a group of pre-compiled binaries in which the Bluetooth Protocol Stack is already implemented with all its layers, such as PHY, L2CAP, and so on [62]. Using this SoC the employed SoftDevice is the s140 for BLE Protocol.

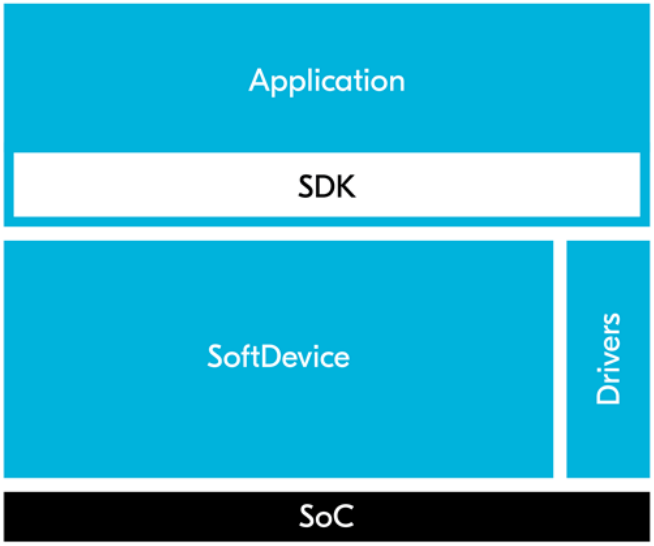


Figure 4.1: Interface between the developed application and the actual Nordic microcontroller [62].

4.2 Application

For the real application implementation, several steps are done. In the following sequential initial configurations applied to the SoC are listed:

1. **Serial Peripheral Interface initialization:** the SPI is enabled and also, the pins through which the SoC communicates with the DAC in order to provide a non zero threshold. These four pins are the ones that handle the four signals of the SPI [1]:

    - The clock is assigned to the pin 0.14.
    - The slave select is assigned to the pin 0.15.
    - The Master Output Slave Input (MOSI) is assigned to the pin 0.13.
    - As the SoC does not receive any message from the DAC the Master Input Slave Output (MISO) is not assigned to any pin.

[1]The Serial Peripheral Interface is a synchronised serial communication among devices where there is one master and from one to several slaves. In general, four wires are connecting two devices: the slave select, for enabling the slave, the clock, that synchronises the communication, and the other two are for the specific data travelling from the master to the slave and vice versa. In this system, this interface operates in every sEMG channel and at the goniometers.

For a pin assignment layout see the Figure 4.2.

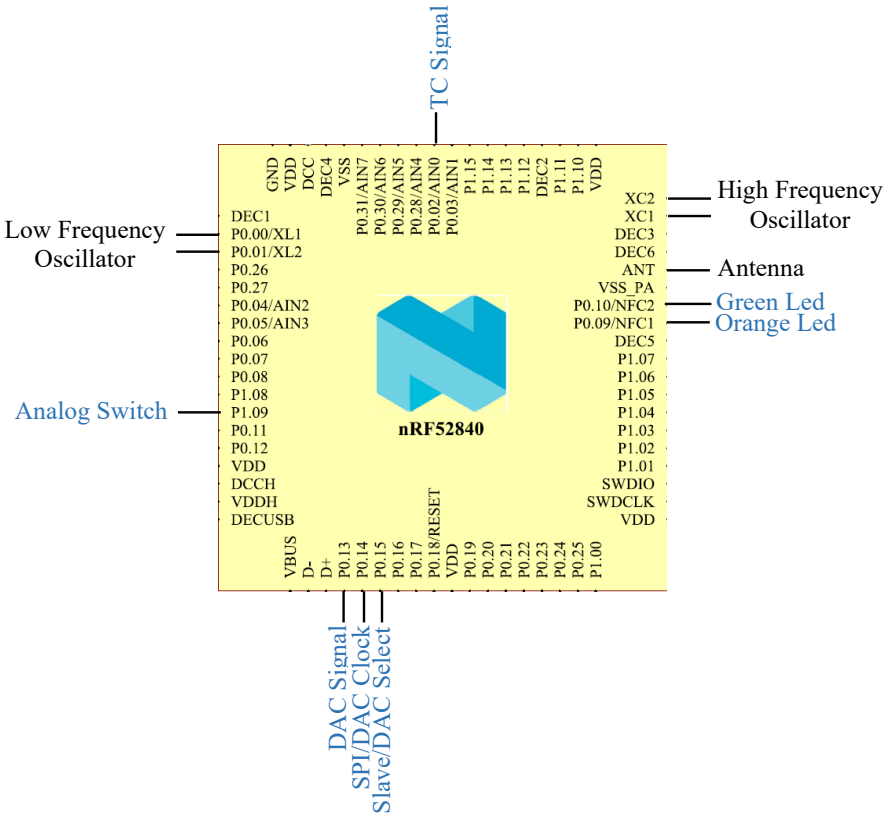


Figure 4.2: SoC Pin Assignment. The most relevant pin assignments are shown, whereas others were omit for clarity. Specific pins for this application are in **blue**, whereas other typical circuit connections are presented in **black**.

2. **Input and Output pin setting:** the pin 0.2 is employed as an input, for receiving the TC signal, and the pin 0.10 performs an output for toggling the green led state every time a single threshold crossing event takes place.

3. **INA shortcut:** the reference and the output of the instrumentation amplifier are shortcut briefly using an analog switch for avoiding an INA saturation issue due to initial conditions. This switch is controlled by the pin 1.9.

4. **Timers initialization:** an specific timer is created but not started, it is in charge of unleashing an interruption every 130 ms.

5. **BLE stack start:** the SoftDevice is initialised and the interruption due to a BLE event is enabled. Then, the GAP parameters are configured, such as device name, appearance, among others. Afterwards, the GATT module is started.

6. **Service and Characteristic creation:** here, the user application becomes meaningful, a service is generated and, with it, two characteristics for handling the threshold new value and the ATC notifying process. The association of the service with two characteristics is not the only task performed at this point, besides, all the attributes of the three of them are set, such as the UUIDs, handles, properties, metadata, permissions, and so on.

   The application main service is shown at Figure 4.3, the first characteristic contains the ATC current value and it possesses a declaration, the actual value and a CCCD Descriptor. The latter allows notifying the value. The second characteristic, involves the threshold value, this one can be read or written but the notifying property is not allowed, that is the reason why there is not a CCCD descriptor.

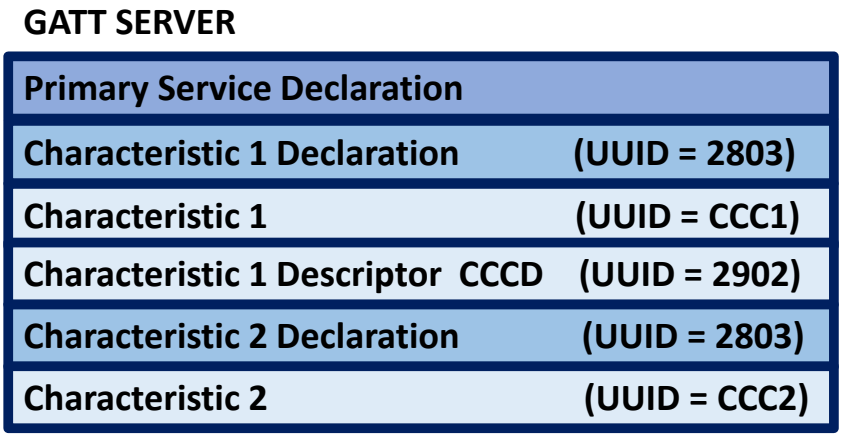


Figure 4.3: Service implemented at the GATT of every channel.

7. **Advertising and idle state:** the connection parameters are selected and finally, the advertising begins whilst the main loop enters in an idle state in which it waits for the next event.

After the peripheral device is ready and it begins the advertising, some events have to be triggered by the central peer in order to accomplish its entire function.

A flowchart, describing BLE application actuations and states, is shown in Figure 4.4:

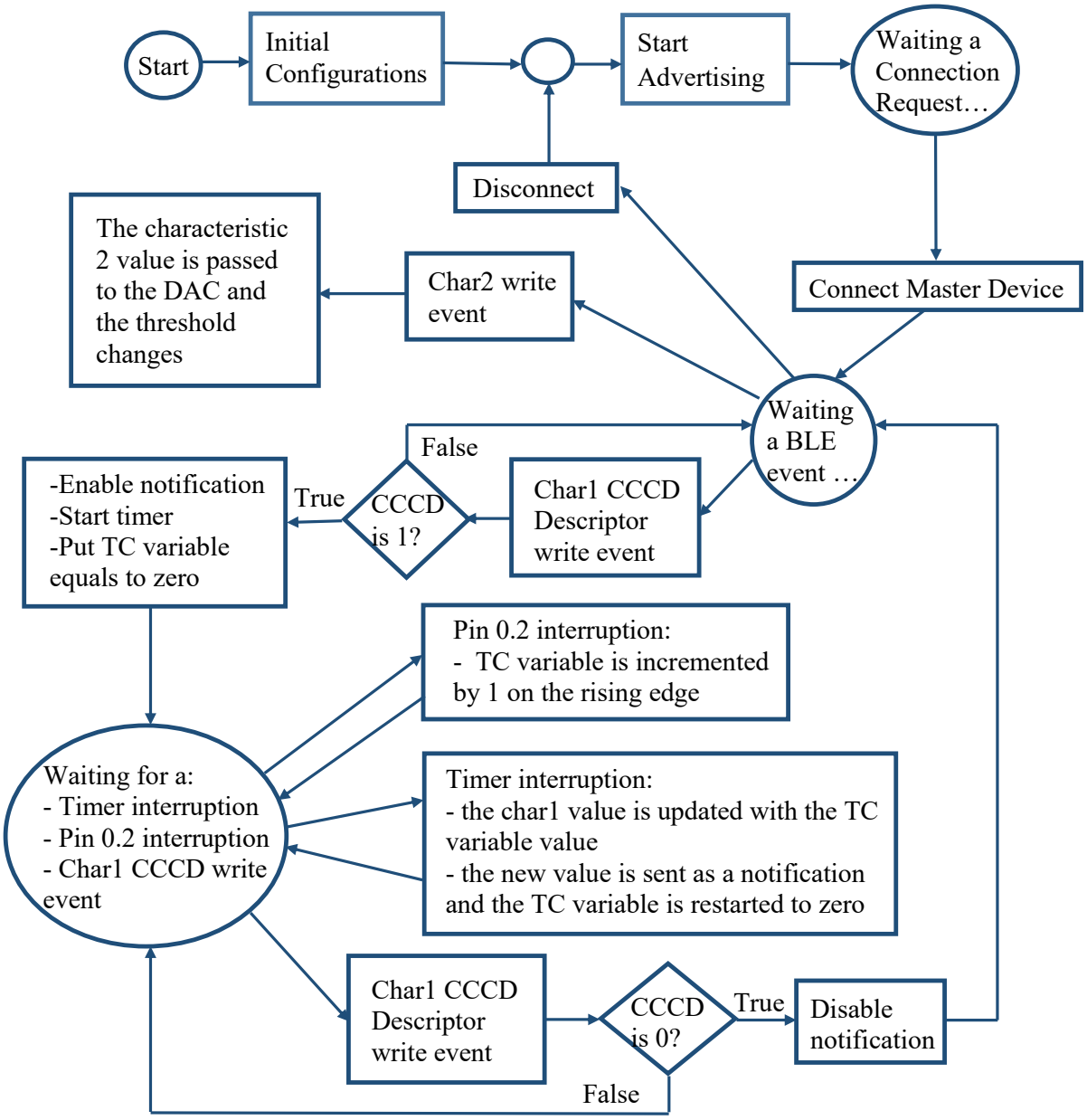


Figure 4.4: Flowchart of channel firmware.

It can be observed from the chart that, after the link has been established the channel halts connected for distinct events, and they could be more than only the three named in the scheme, for example, there is also a reading event for both characteristics, but those are not as essential in the channel functionality.

In this waiting state a write event on the second characteristic value triggers the DAC output modification, as a function of the value itself.

In addition, whether the characteristic one descriptor is written and the value is

a two byte hexadecimal number equivalent to 1, the notifications from this channel to the central device are enabled. From here, mainly, other three events can arise, two of them are in charge of counting the amount of threshold crossing events that occur in the fixed 130 ms time window. Finally, if the descriptor is written again with the same structure value, but this time equivalent to 0, the notifications are terminated.

Chapter 5

Embedded Software Development

The entire system, as expressed before, is composed by several sort of devices, so an **Object-Oriented Programming (OOP)** supported language ought to be utilised in order to obtain more robustness and flexibility than with other programming paradigms[1], such as the procedural one.

Therefore, **Python** is the chosen language for developing this programme. It owns a readable syntax, simpler than other OOP languages, due to it is **dynamically-typed**[2] and **garbage collected**[3]. In addition, it supports different paradigms and is available for diverse operative systems. Nevertheless it is not the most powerful programming language, it is more than enough for this application.

In the OOP approach, the modelling of complex systems is implemented by using models of the real world objects, named in the same way. Software *Objects* are defined by *Classes*, which contain all their specific information. This data can be under the name of *Attributes*, whether it represents objects characteristics or properties, or *Methods* if it symbolizes their actions or behaviours [63]. The collection of methods and attributes define the class **Interface**.

In this paradigm, some associations among objects happen and is in the synergy, created from their interactions, that the power of this paradigm lies. The most

[1]The Programming Paradigm is a characteristic of the programming languages, used to define them based on its features, which sometimes, involve the syntax and grammar, while others, the code organization or execution.

[2]It means that at runtime, it performs actions that other static languages make during the compilation, as C++. This actions can be adding new code or modifying the type system, among others.

[3]The Garbage Collection is a feature where an object called collector, automatically, retrieves memory that is been used to store objects no longer called by the application.

important ones are:

- **Inheritance:** it is possible to define a child class which shares the parent methods and attributes, this is useful when there are objects with several properties or behaviours in common.

- **Aggregation:** is a type of association between objects, where an object is added to another one but both are independent from each other, i.e., is not senseless to think in the existence of one without the other.

- **Composition:** this is similar to the prior one, but here, whether an object, which is made up of others, disappears, the remaining ones do not have reason to exist, so they are destroyed too.

5.1 Code Hierarchy

The entire software is divided in 10 principal modules, which store the different classes, functions, imported libraries and so on. The *main* one is **GUI_tesis** because it sets the software framework and starts the application, in addition, it establishes the actions and only a part of the visual aspect of the interface, the rest of it is decided in the *Main.kv* file.

The second more important module is the **System** one, which defines the equally called class and incorporates the secondary modules for characterising the system components, such as the goniometers, the BLE subsystem and the stimulator. Any of them, at the same time, import even more required sections, for instance, the **Exceptions Management** one, in charge of defining the custom exceptions and errors of the entire programme.

Furthermore, the **FES_module** section only specifies the stimulator's class, but it also, includes a **CRCcalculation** library which contains the function for calculating the checksum in the transmitted serial packets (described in details at 5.2.3). On the other hand, three different classes were developed in order to manage the acquisition architectures, the **BLE_channels_module** is the one containing the class selected for composing the system when individual channels are handle with the Raspberry Pi BLE hardware, while the **BLE_dongle_channels** and the **BLE_oldboard_module** enclose the acquisition configuration where the individual channels and the four channel board are linked to the workstation through the master dongle.

Finally all the modules own its logger object as a global variable, for implementing a logging procedure wherever in the application this is desired.

5.2 Classes Description

In Figure 5.1 a summarised class diagram can be observed. The *main* class **System** is composed by all the needed objects, which are instantiated from their respective classes. Therefore, the object system interfaces the user action, aided by the GUI (see Section 6), with the distinct elements and, also, embeds them together.

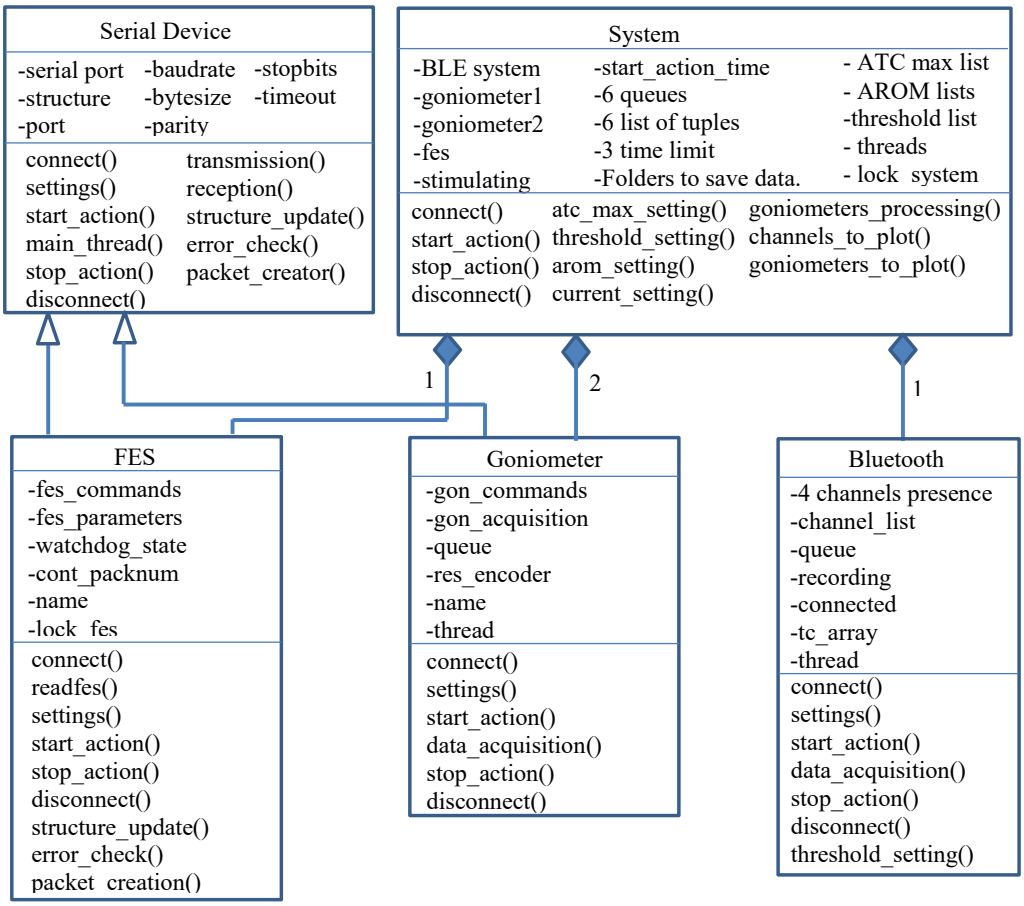


Figure 5.1: System class diagram according the Unified Modelling Language (UML). Only the relationship among the principal classes is presented for clarity.

5.2.1 Serial Device

This is an *abstract class*, so its scope is not to be instantiated but provide an interface that serial devices in the system must respect.

It does not inherit from the Python Class Serial, however it is composed by one serial element, in order to give an **encapsulation** of the Serial Class, meaning that an object from this class does not know how is the serial class implemented but it is enabled to used it.

The stimulator and goniometers inherit from this class and, that is why, they share the same interface. Obviously these devices are not identical, as a consequence, some of those parent class methods should be overwritten for the specific implementations of the two elements previously named.

Class Interface Overview

As it can be observed at the Figure 5.1, this abstract class defines as attributes the serial port with all its properties, like the *name*, *baudrate*, *bytesize*, and so on. Whilst the most relevant methods are the **connect** and **disconnect**, that allow to open and close the serial port, and the **transmission** and **reception**, which permit the communication through data packets with the serial device. Finally, the rest of behaviours are implemented afterwards, in each child class.

5.2.2 Goniometers

As serial devices, they are defined by a child class of Serial Device. As previously shown in Figure 5.1 the system involves the composition of two goniometers.

Class Interface Overview

An instance of this class has specific attributes, and also inherits the serial port configuration, the actual values can be seen in the Table 5.1.

Table 5.1: Goniometer attributes. *Threads are discussed on Section 7.

Attribute	***Type***	***Value***	***Attribute***	***Type***	***Value***
serial	Object	-	Start_Command	String	'1'
port	String	According OS	Stop_Command	String	'0'
baudrate	Integer	9600	gon_acquisition	Boolean	False
bytesize	Integer	8	queue	Object	-
parity	None	-	encoder_resolution	Float	0.087
Stopbits	Integer	1	name	String	-
Timeout	Float	100(ms)	thread*	None	-

As can be analysed from the table, half of the attributes are for configuring the serial communication and the other parameters are used in the information processing.

When the *Start_Command* is sent to the Arduino controlling the encoder, the *gon_acquisition* flag is set to *True* and a loop is initialised. There, the *serial* object reads the port buffer waiting for a 2 bytes long hexadecimal number containing the angle information. This value arrives every 13 ms and is processed using the *encoder_resolution*. Then, the real angle value with its corresponding occurrence time are sent through a *queue* to the entire system.

There are methods and an attribute unused here, because the goniometers do not answer back after a packet has been sent from the Raspberry Pi, so an **error_check** and a *structure* property which saves feedbacks from devices, are useless, as well as, the **packet_creator** and the **structure_update**.

5.2.3 Functional Electrical Stimulator

This class also inherits from the Serial Device one, so as before, some properties and behaviours of the stimulator could be common, and the specific ones need to be overwritten.

Attributes

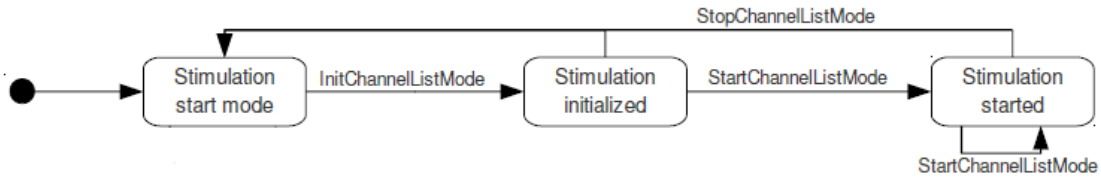
Serial Port setting The functional stimulator uses the ScienceMode2 software, a serial communication protocol previously mentioned, so the computer port shall be configured to achieve this interaction between both. One outstanding parameter is the timeout, which is not established as default (see Table 5.2), this was chosen in this manner as a consequence of the initial steady data sending from the device to the workstation. Later, when the port has been opened and the connection set, the timeout is changed to 800 ms.

Table 5.2: FES serial communication setting values.

Attribute	***Type***	***Value***
Serial	Object	-
Port	String	According OS
Baudrate	Integer	460800
Bytesize	Integer	8
Parity	String	'Even'
Stopbits	Integer	1
Timeout	Float	None

Modes In Section 3.2, the various stimulation modes that the FES supports were enumerate, the employed here is the CCLM.

The FES switches among states, as shown in Figure 5.2a, these are reached through the serial data packets communication with the workstation. The hexadecimal number travelling between the those apparatus has a defined structure, that, at the same time, consists in a fixed part and a variable one in function of the specific packet.



(a)

Start byte 1 Byte	Checksum 2 Byte	Data length 2 Byte	Packet number 1 Byte	Command 1 Byte	Command data < 61 Byte	Stop byte 1 Byte
			<— Checksum —>			
			<— Data length —>			

(b)

Figure 5.2: Stimulator working modes utilised in this application (a), and its packet structure (b) [58].

The packet structure can be split in different sections, as follows:

- The fixed part of the packet is made up of three constants: the **Start**, **Stop** and **Stuffing** bytes. The latter is accompanying the real checksum and the data length value, that is why those two are consider as two byte-long numbers, see Figure 5.2b for reference.

- The **Checksum** is a one byte-long hexadecimal number that comes from a Cyclic Redundancy Check (CRC) algorithm, specifically the CRC-8-CCITT. This computation is made in function of the non-steady hexadecimal number composed by the *packet number*, the *command* and *command data*. It performs a polynomial division between the latter and the specific algorithm's polynomial, in this case, $x^8 + x^2 + x + 1$. Afterwards, the remainder is modified through an XOR operation with the *stuffing key*. The outcome permits, the FES, to check raw data integrity by recomputing the value and assessing equality between the received and the calculated. Be aware that stuffing key is not the same as stuffing byte.

- The **Data Length** is obtained, as the checksum, from the data formed from the three pieces before named. As the command data can vary its length, it is not necessarily a fixed number.

- The programme implements a counter to enumerate every single packet sent to the stimulator, it is important to consider, that every time this value is equal to one of the three constants always placed in the packet, it shall be transformed by an XOR computation with the stuffing key and that value is

stuffed with the stuffing byte, so in these cases the **Packet Number** becomes two byte long.

- **Command** corresponds to an unique hexadecimal number indicating a device modality or state, in other words, is an identification of the FES operation. The values used in the class as attributes, required for the application, are shown at the Table 5.3.

- Every single transmitted command need to set some parameters, that are included in the packets, at the called **Command Data** section. These settings are variable, as a consequence, the command data is at least shorter than 61 bytes, and not always the same. In the next section, a description of parameters values, chosen for this project, is presented.

Table 5.3: FES commands and constant values.

Attribute	***Type***	***Value***	***Attribute***	***Type***	***Value***
Start_byte	String	’f0’	Stop_byte	String	’0f’
Stuffing_key	String	’55’	Stuffing_byte	String	’81’
Init	String	’01’	Init_Ack	String	’02’
InitChannelList	String	’1e’	InitChannelList_Ack	String	’1f’
Start_Update	String	’20’	Start_Update_Ack	String	’21’
Stop	String	’22’	Stop_Ack	String	’23’
Stim_error	String	’26’	Unknown	String	’03’
Watchdog	String	’04’			

Parameters As mentioned at the Section 3.2, the FES allows the user to change many parameters to set an accurate stimulation. At this point three types of attributes can be observed at the Table 5.4:

1. The **InitChannelListMode**: these are the parameters set before the beginning of the stimulation, they are part of the packet with the same name.

   The *main stimulation interval* is chosen as 25 ms (40 Hz), because this value is contained in every recommended stimulation range for each sort of movement this project performs [64]. Moreover, it is a value in common for all the channels and the *interpulse interval* value must be contemplated to respect the formula before shown, that is the reason why that value is configured to be fixed at 8 ms, in this manner even whether a Triplet is selected, the single stimulus repetitions are compressed within the main stimulation interval.

Other two values needed to initialise the stimulation mode are the *low frequency factor (low_ff)* and the *low frequency active channels (Active_LFchannel)*, here any channel is handled in that way, so they do not configured any channel as a low frequency one.

Finally, *channel execution* only has one option for this FES software release, the fixed interval.

2. The **StartChannelListMode**: These parameters are the ones that define values included at the start stimulation packet. The *mode_list* and the *pulsewidth_list* are the actual values that are going to be sent but, the *maximum_current_list* values constrain the highest possible amplitude, whereas the real current value is calculated at the system class and passed to the FES instance.

3. The **Specific** ones: these attributes are used for the object in order to achieve its scope. The *cont_packnum* stores the packet number and it is sent within the packet, while the *watchdog_state* is a flag indicating the connection state of the FES with the workstation. Finally, the *lock* works in the synchronization of threads (see Section 7 for clarity).

Table 5.4: FES attributes.

Attribute	***Type***	***Value***
main_interval (ms)	Integer	25
interpulse_interval (ms)	Integer	8
low_ff	Integer	0
Active_LFchannel	String	'00'
Channel_execution	String	'00'
mode_list	List of Strings	['SP', 'SP', 'SP', 'SP']
pulsewidth_list (µs)	List of Integers	[20, 20, 20, 20]
maximum_current_list (mA)	List of Integers	[0, 0, 0, 0]
Channel_Init	List of Strings	['0', '0', '0', '0']
cont_packnum	Integer	0
watchdog_state	Boolean	False
name	String	'FES'
thread	None	-
lock_fes	Lock Object	-

Methods

The behaviours of the device follow the Serial Device class template but the implementation varies a lot from the one done within the goniometer class.

In the **connect** method, the workstation reads packets from the FES and if the correct protocol information is codified within the packet a connection is started. As soon as that occurs, the Raspberry Pi/workstation starts to continuously halt for data on the serial port and every 800 ms it maintain the bond by transmitting a watchdog packet to the stimulator.

Afterwards, every time an stimulation is desired, the FES is configured with the updated parameters, before explained, and enters first, the Stimulation Initialized state, and later, the Stimulation Started one, using the **settings** and **start_action** behaviours, respectively (see Figure 5.2a). Once at the latter, the stimulation StartChannelListMode parameters can be modified or, simply, the stimulation could be stopped.

Other behaviour to noticed, is that in the **structure_update** and **error_check** functions, the workstation assesses every single response for every single emitted command and classifies it as Successful, Correct Protocol or an Error, like Electrode Error, Wrong Mode Error, Stimulation Module Error, Emergency Switch/not connected Error, Transfer or Parameter Error.

To conclude, the FES instance uses its **packet_creation** method to generate the correct data to be sent when required.

5.2.4 BLE system

This subsystem controls the acquisition boards to gather all the data coming from them. Two approaches were previously mentioned for the physic acquisition system, the 4-channel board and the individual channels.

Furthermore, for the code developing with the former, the CC2540 BLE dongle was employed for the communication, whereas with the second option, two cases were assessed: as it was intended to leverage the workstation own BLE hardware and, due to, handle that on a Windows Machine is not an easy task, it was decided to do this approach only on Linux, where **BlueZ**, a Linux Bluetooth stack that offers support for the Bluetooth layers and protocols [65], is available. On the other hand, when running the application on Windows the BLE dongle has to employed.

Raspberry Pi and individual channels

This is going to be referred as Main Hardware Configuration (MHC). In order to take advantage of Raspberry BLE hardware an Application Programming Interface (API) for controlling it from Python was sought. The chosen library is **bluepy**, that was almost entirely developed in a Raspberry Pi and is specific for Low Energy [66].

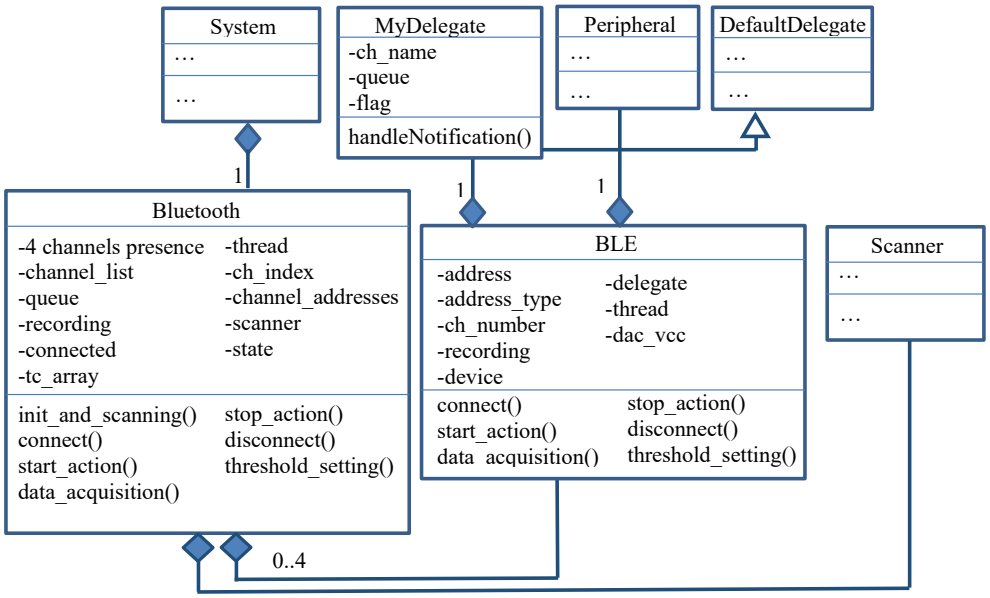


Figure 5.3: BLE subsystem class diagram to communicate the channels with the BLE device within the workstation.

As can be seen in the Figure 5.3, the Bluetooth system is composed by an user selected number of BLE connections, from 0 to 4, in addition, each single

connection owns a Delegate and a Peripheral. The former is the object in charge of asynchronous data manipulation, as notification handling, and the latter is a bluepy defined instance that encapsulates a connection to a BLE peripheral through BlueZ, which in this case is obviously a single sEMG channel. Besides, this instance will have a scanner element that seeks for advertising Low Energy (LE) devices.

Interface Overview At the Table 5.5 a summarized list of the principal three classes properties, implemented in this approach, are detailed.

Table 5.5: Main properties of the Raspberry Pi and Individual channels approach. The attributes correspond to: the BLE system class in **green**, the BLE connection in **blue** and to MyDelegate in **red**.

Attribute	*Type*	*Value*	*Attribute*	*Type*	*Value*
ch_1	Boolean	False	**ch_2**	Boolean	False
ch_3	Boolean	False	**ch_4**	Boolean	False
channel_list	List	Empty	**tc_array**	Column Array	[0,0,0,0]
recording	Boolean	False	**queue**	Object	-
thread	None	-	**connected**	Boolean	False
scanner	Object	-	**ch_index**	Integer	-
state	String	'All'	**ch_ads**	List of Strings	["","","",""]
address	String	-	**address_type**	String	'random'
ch_number	Integer	-	**device**	None	-
thread	None	-	**recording**	Boolean	False
delegate	Object	-	**dac_vcc**	Float	-
dac_resolution	Integer	4096	**ch_name**	Float	-
queue	Object	-	**flag_first**	Boolean	True

The system involves 4 *ch* variables, these are boolean and indicate whether the user would like to connect to that channel or not.

When connecting, it adds every link instance to the *channel_list* and calls every **connection** method of the selected peripherals. After the link was created using the *address* and *address_type*, the *delegate* is associated to it, for performing notifications processing when enabled.

Once the BLE system **start_action** method is called, the notification initialization is triggered for one or all the devices connected, relying on the *state* attribute. This technique was implemented for calling only one channel specifically when it is needed, or simply, for calling all the linked ones. At the first case the *ch_index* gives a reference to the desired channel. For starting this procedure, the system writes the CCCD descriptor of the first characteristic and sets to *True* its attribute

recording, then waits for a new measurement in all the *queues* that link the BLE connection delegate of each activated channel with itself. Afterwards, it builds a column from that data, called *tc_array*, with zeros for the channels that are not in use, and implements its own *queue* for transmitting to the general system. These data packets are made in this manner for being compatible with the others acquisition systems implemented in this thesis.

On the other hand, for **stopping** the notifications flow, the system writes the characteristic descriptor again and modifies the *recording* flag to exit every loop running.

Finally, the **threshold_setting** behaviour employs one channel at the time, and the specific link uses the Digital to Analog Converter (DAC) voltage supplied value, referred as *dac_vcc*, and the *dac_resolution* for computing the 2 byte-long hexadecimal value, required to achieve the wanted threshold at the channel by writing the second Characteristic value.

The *connected* system attribute serves to evaluate the condition of the BLE and avoid a double connection attempt. Moreover, the delegate *flag_first* is in charge of dropping the first received notification value because it does not corresponds to an ATC value.

BLE Dongle and individual channels

The general system can decide, based on the operative system and the parameters established at the configuration file, which classes of these three it should instance. That is why, this class might integrates the **System** class. Besides, it can be observed at the class diagram shown in the Figure 5.4, that when using the dongle, which communicates through a serial port with the workstation, the inheritance from the **Serial Device** class shall be done. Hence, the BLE instance becomes a serial device element of the system, like the FES and goniometers.

This class defines how the CC2540 dongle is going to work for interacting with the 4 individual channels. This implementation is compatible with any platform due to the employment of extra hardware. A summary is presented in the following.

Interface Overview The interface used here is similar to the one employed in the first case. The main differences are the presence of the *BLE_commands* that are sent trough the serial port for interfacing with the dongle and the extra method called **handle_notification**, which is in charge of handling all the packets arriving to the workstation when the notifying behaviour is activated, independently whether the packets arrive from one or different peripherals. When this is executed the received packets are classify, according to their own identifying pieces, as a write response or as a notifying packet. The latter should be associated to a channel so link's handles are saved when the **connection** occurs.

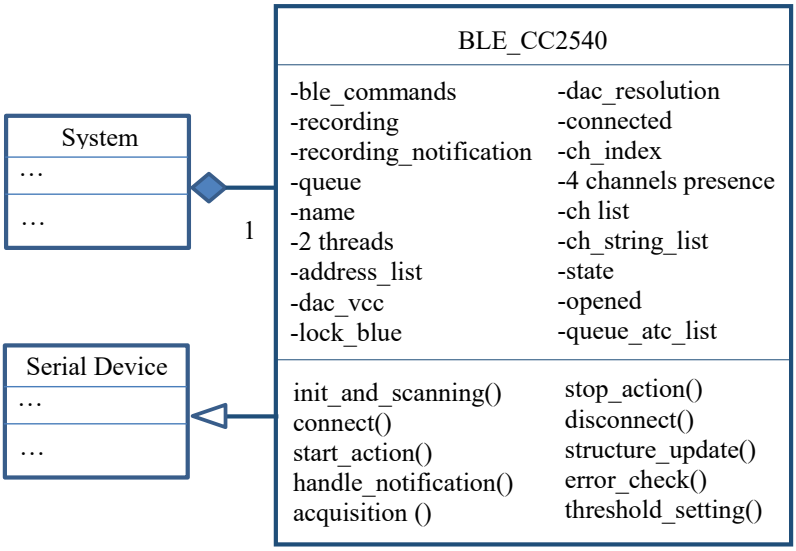


Figure 5.4: BLE subsystem class diagram to communicate the channels with the dongle CC2540.

Furthermore, as seen for the stimulator, the **error_check** and **structure_update** methods, are overwritten, in order to be functional for this acquisition approach.

BLE Dongle and 4 channel board

Finally, the 4 channel board described at the section 3.1.1 was also implemented using the same dongle. The diagram class shown in the Figure 5.5, is almost identical to the one presented before.

Before describing the interface, two significant aspect of the implementation of this board must be clarify. The first one is that the firmware originates a distinct GATT Server at the peripheral, and as a consequence, the *BLE_commands* and other properties are not the same as in the previous case. Secondly, the microcontroller in this board collects the information and send ATC packets already organized, with all the four values in the same packet and, moreover, as only one unique threshold is created, the implementation of the interface is, in general, a little bit different.

Interface Overview The biggest alterations are, firstly, in the data acquisition, here, the four channel data is already gathered and the board can also sent a digital sEMG signal wirelessly, then it is not enough to only write the CCCD descriptor, also other characteristic value must be written, for enabling the notifications of the ATC values, specifically. As a consequence of the data pre-organization at the microcontroller, the received packet is directly sent through a *queue* to the system,

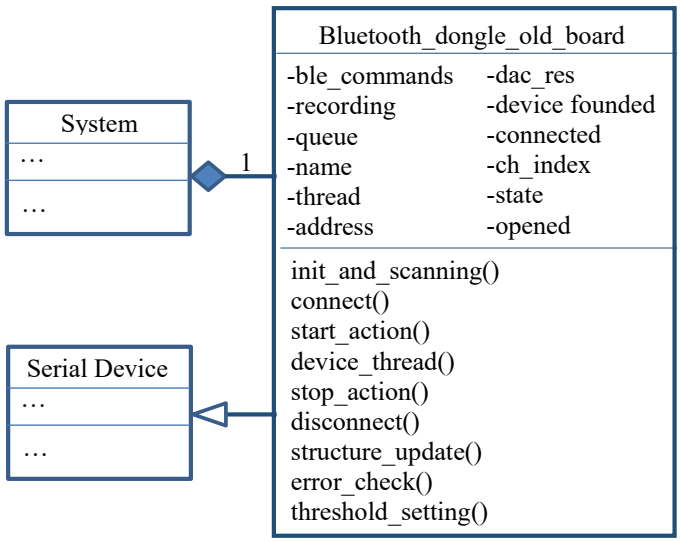


Figure 5.5: BLE subsystem class diagram to gather information from the 4 channel acquisition board with the dongle CC2540.

for being processed, and it is not necessary to classify packets by its source device, due to, there is simply only one connection.

Secondly, the initial BLE parameters are configured at the central peer, so a connection interval, a slave latency and a timeout are established by the dongle. Finally, the **threshold_setting** is performed once for all the channels but the system must deals with the way in which this value is selected and which channels are considered for that.

5.2.5 System

This object, as previously mentioned, is made up of multiple elements. It is in charge of processing all the information coming from the inputs, such as the goniometers and channels, and for triggering responses at the output, the stimulator.

Interface Overview

Obviously, the system uses the four instances that manage the 4 physical devices, otherwise it involves many other properties for accomplishing its task. Some of them can be seen in the Table 5.6.

Table 5.6: System attributes.

Attribute	***Type***	***Value***
fes	Object	-
blue	Object	-
gon1	Object	-
gon2	Object	-
stimulating	Boolean	False
start_time	Integer	0
data_matrix	Array	Null Matrix 4x4
6 queues	Object	-
4 current	List of tuples	[(0, 0)]
2 angle	List of tuples	[(0, 0)]
3 limit_times	Integer	10
ATC_max	List	[10, 10, 10, 10]
therapist_data	Dictionary	-
patient_data	Dictionary	-
current_set_param	Dictionary	-
movement_t	List	[]
movement_p	List	[]
lock_system	Object	-
AROM_p	List	[40, 40, 40, 40]
AROM_t	List	[40, 40, 40, 40]
threshold	List	[1.9, 1.9, 1.9, 1.9]
2 threads	None	-
6 data_folder	None	-
connected	Boolean	False

The utility of this properties is better seen inside the behaviours or actions that this object can unleash.

- **connect:** this method simply links all the four devices and changes the value of the flag *connected* to True.
- **start_action:** is the steady stimulation in function of the ATC signal, the system analyses all the data coming from the sEMG acquisition system.

  At the beginning all the devices are set in order to be ready for the stimulation and the current vector, for each channel, is computed. These array values start at 0 and finish at the specific *max_current* element, also, the step is equal to the ratio between the relevant values from the *max_current* and *ATC max* properties, see Figure 5.6.

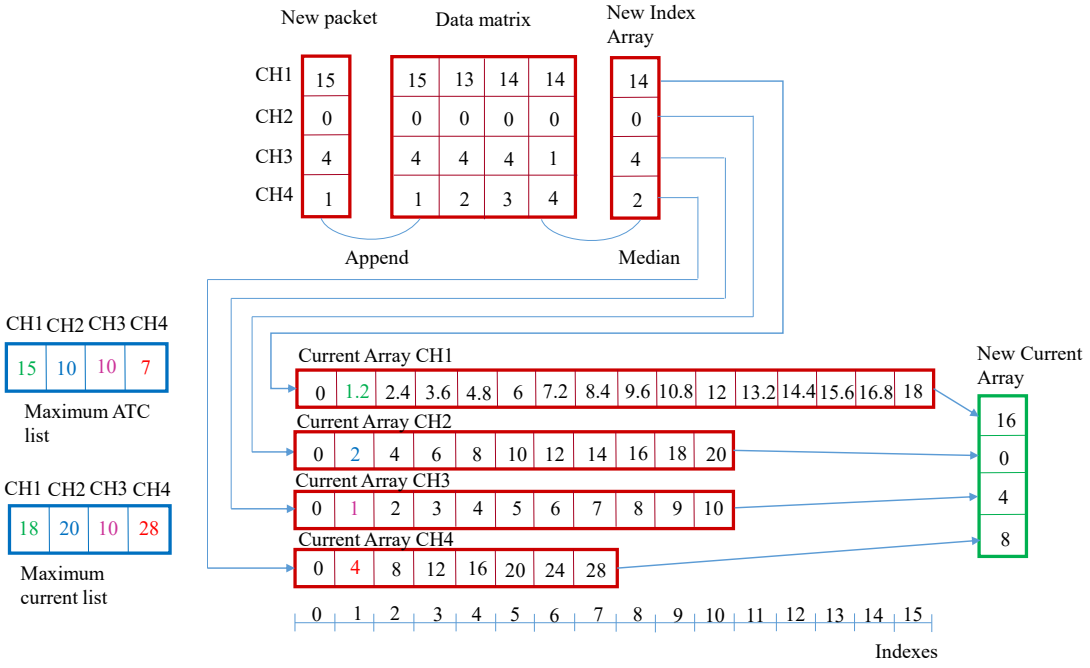


Figure 5.6: Numeric example of the current calculation. The steps, which are the result of the ratio between the Maximum ATC and current, are shown at the index 1 of each array. Furthermore, for obtaining the New Current Array, the values are picked in basis of the New Index Array, for instance, the channel 1 new index value is 14, pointing to the value 16.8, as the FES only manage integer current values, the number is rounded to the minimum, 16.

The packet received is an one dimensional array with length equivalent to four, then, every of the three priorly seen classes deliver the data in that defined shape. As shown in Figure 5.6, the ATC packet is gathered with the other precedent three arrays, creating the *data_matrix*. Moreover, the median among this four values, in the row axis, is computed. Afterwards, those values,

that are going to be between 0 and the *ATC_max* calculation, are employed as indexes in the relevant current array created for every channel.

Once this packet is processed, the system needs to pass information for two different scopes. In the first one, it sends the computed current and time value to the **channels_to_plot** method. The time value is found as the difference between the present time and the *start_time*, which involves the time when the stimulation was started. In addition, for the second scope, it shall use the *fes* object to perform the execution of the stimulation.

- **stop_action:** here, the *stimulating* flag is modified to False and the stimulation halts, afterwards, it stops all the other loops that are acquiring data and exits the *fes* started stimulation mode.

- **disconnect:** All the serial ports are closed and the wireless links are ended.

- **threshold_setting:** it tries to set a minimum threshold for maximise the number of ATC events, and, in that manner, achieve the maximum current at the patient, with less effort from the therapist.

  Mainly, it sets a desired threshold and starts the Bluetooth acquisition for one and a half seconds, whether, at least, one or more ATC values different from zero are not observed it decrements the threshold value in 10 mV and repeats the task. When finally a non zero ATC value is caught, the threshold is set to that value plus 30 mV.

  Relying on the acquisition system, the channels can be evaluated one at the time, or even, multiple channels can be assessed when the 4 channel board is utilised. In that situation, only the channels that have been actually selected for the user are checked, hence, the unique threshold will be the maximum value among all the minimum chosen channel thresholds.

  Finally, here the *threshold_list* attribute stores the outcomes.

- **atc_max_setting:** at the beginning, the acquisition is initiated and the ATC data start to arrive. This packets are always received as arrays with 4 measurements, for any acquisition technique, but only one of the 4 values is assessed at the time.

  Every time the ATC value is higher that 1 for more than two consecutive times the algorithm considers that a movement has happened, so it stores all the values between that moment and the one in which the value fall to 0 again, from this data the maximum is computed and saved.

  Finally when the user decides to stop this calibration procedure, the maximums of all the motion repetitions are available and the ATC max is the median of this data array.

Obviously, the more repetitions the therapist makes, the more accurate the calculation will be but, performing too many muscle contractions derives to fatigue, so a trade off should be found, that is the reason why the recommended number of repetitions is 5. As in the threshold setting, the ATC maximum values are written at the *ATC_max* system property.

- **arom_setting:** the system receives data coming from a particular goniometer. It evaluates when, these values surpass 10°, and when, they return to be lower than this fixed angular threshold, in order to computed the maximum of all the data gathered between this two events.

  This calculation is utilised as a reference for finding the minimum current that provides a significant angular motion.

- **current_setting:** this method is in charge of generating a stepped wave for stimulating the patient, here the *current_set_param* attribute owns the different parameters for this wave, such as the start minimum and maximum current. The originated wave is symmetrical, meaning that the stimulation is performed in ascending and descending ways.

  The wave possesses a peak, the maximum current, which is maintained for 400 ms, whereas the other levels remain only for 180 ms, see Figure 5.7.

Figure 5.7: Stepped Wave generated for current amplitude evaluation.

Once all the pattern has been done a rest period of 4 s is set and a new maximum value, equals to the present highest number plus two, is added to the pattern and the operation is repeated.

In addition it calls the method **goniometers_processing** for the angular data analysis.

- **goniometers_processing:** this method can execute two sorts of actions, when it is called from the **start_action** one, it fetches the goniometers data and generates pieces composed by 10 values, in order to synchronise the goniometers and channels information, for plotting, at 130 ms.

The other task occurs when this method is triggered from the **current_setting** one. At this situation, the angular data is compared with the percentage, indicated at *current_set_param*, of the equivalent AROM measurement for the patient, located at *AROM_p*. When the angular information exceeds, in magnitude that value, the stimulation is stopped and the maximum amplitude of the stimulation wave is established as the 110 percent of the present current value [64].

- **channels_to_plot:** it fills the four *current* list of tuples and sends them to the plotting function, the MyPlotScreen method called **get_value** (see 6). As the *limit_time* is fixed in 10 s, whenever the arrival time calculated for the measurements exceeds that value, it saves the list of tuples in the *folders*, as appended chunks, and empties them, in order to only represent the data that is actually placed in the time window printed by the graph. If that is not the case, the amount of data to show, grows rapidly and the frames updating decrements its frequency, delaying the data plotting.

- **goniometers_to_plot:** here, as above, the same technique for reducing the data length of the arrays, is employed, but this time with the angular measurements.

Chapter 6

Graphical User Interface

In order to allow the user manipulation of the entire system a *Graphical User Interface (GUI)* was developed using **Kivy**. This is a Python library, for creating modern interfaces, which works in distinct devices and on several operative systems, such as Windows, Linux, Android, OS X and, more interesting, on Raspberry Pi. Also of been completely free, Kivy performances are incremented because the graphics engine implements a new and high speed graphics pipeline [67, 68].

At Figure 6.3 the GUI class diagram can be observed, and some considerations to keep in mind could be declared:

1. The GUI is composed by 5 screens (see Figure 6.1), which are the Start, Login, Plot, Calibration and Parameters. Kivy uses an object **Screen Manager** in order to handle them and decide the transitions among them, for that, it aggregates a single screen object for each one.

2. All those screens are created from classes which inherit from the Kivy Screen class but two of them contain graphs, so, in order to take advantage of the similarities, those two are defined first by a **MyPlotScreen** Class. This class possesses some Kivy objects needed for plotting the different signals that this application manages, such as six plot objects, one for every signal, two graphs, one for the angles and another for the current or ATC value plottings. Furthermore, it includes lists for selecting which signals are going to be actually plotted by adding them to the related graph. Its **start** method initializes the graphs and starts a **clock**, which is going to call the method **get_value** every 70 ms for updating the plots and the graphs contemporaneously.

3. The system instance is *aggregated* in every main screen, in where, the user can unleash system actions through the widgets[1] that these screens involves.

[1]Widget, in informatics, refers to a graphic component that is part of a programme User Interface (UI), it is intended for facilitating the user interaction with the programme.

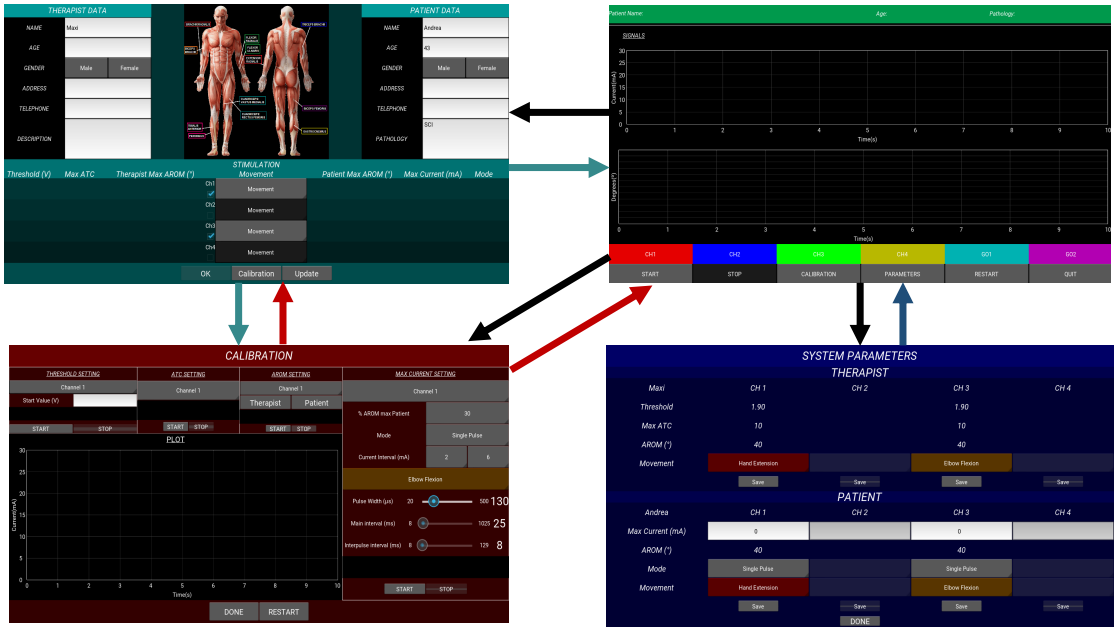


Figure 6.1: Application screens and possible transitions among them, at the top left corner the Login screen is presented while the Plot one is next to it. In addition, the bottom right corner is occupied by the Parameters screen which has the Calibration one to its left. The Start screen is not shown for clarity.

4. Some screen can be, many times, the current one set by the Screen Manager, whereas the GUI is running, so a reset of useful attributes (including the system variables) is implemented at the **on_pre_enter** method that every screen object owns, for avoiding inconsistencies.

5. Besides, as shown in Figure 6.2, a custom spinner class inherits from the Kivy's one and overwrites the method *change_on_drop_down_list* in order to change its colour, relying on the user selected movement.

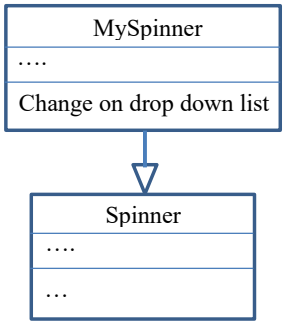


Figure 6.2: Custom Spinner class diagram.

6. Also in order to execute this GUI with the Raspberry Pi 3B+, the RAM memory assigned to the Graphics Processing Unit (GPU) shall be change from 64 to 256 MB.

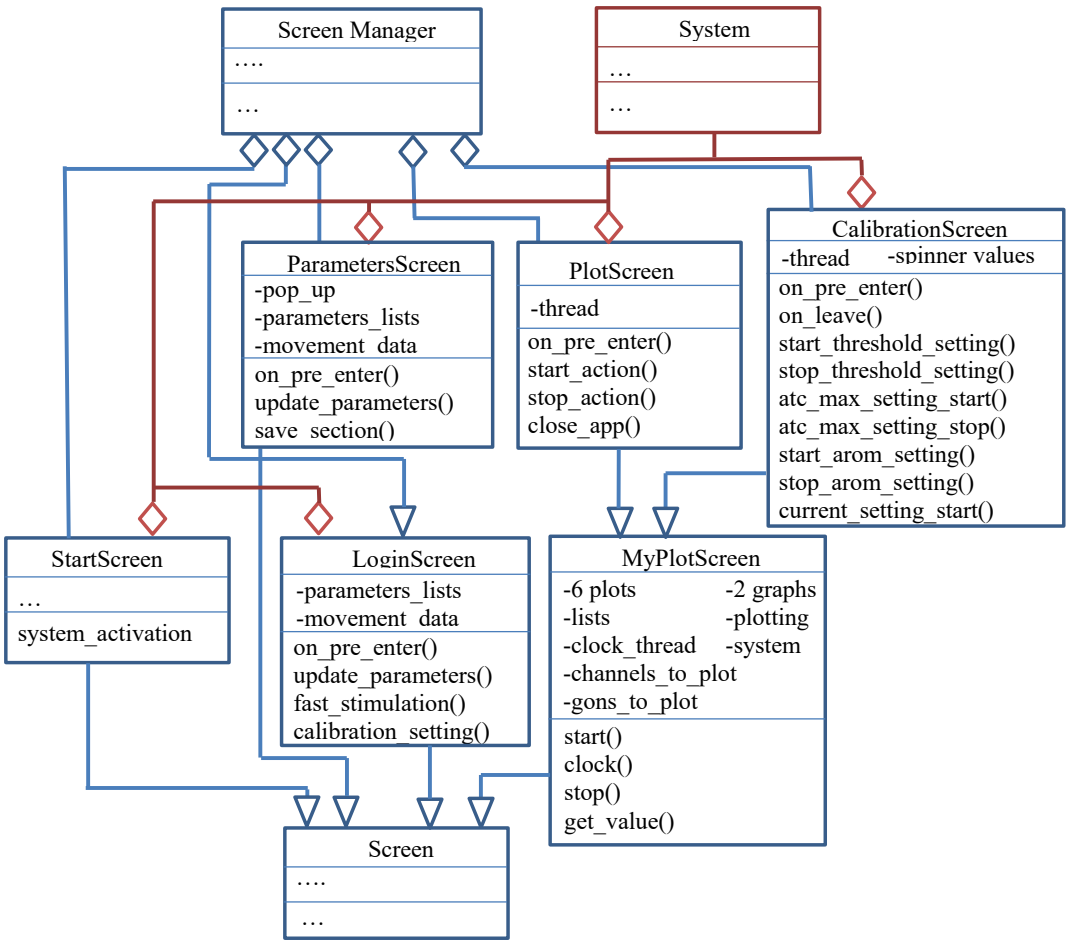


Figure 6.3: GUI class diagram including Kivy classes.

In the following the most relevant screens are described, in terms of their design and interface.

6.1 Login Screen

This is the second screen, which allows the user to configure all the parameters required to perform an sEMG controlled stimulation.

6.1.1 Interface Overview

This instance owns several lists as attributes, each of them stores the four widget references that are utilised for showing the parameter values. It is possible to see that those properties are: lists for the four *maximum currents*, the stimulation *modes*, the *movements*, the *thresholds* and the *channels layout* that involves a *Checkbox* and a *Spinner*, objects for enabling or not the channel selection and the movement performed with it, whether it was found advertising or not (see Figure 6.4). In addition, it has the system instance and a special variable with the corresponding *pulse width* for each permitted motion.

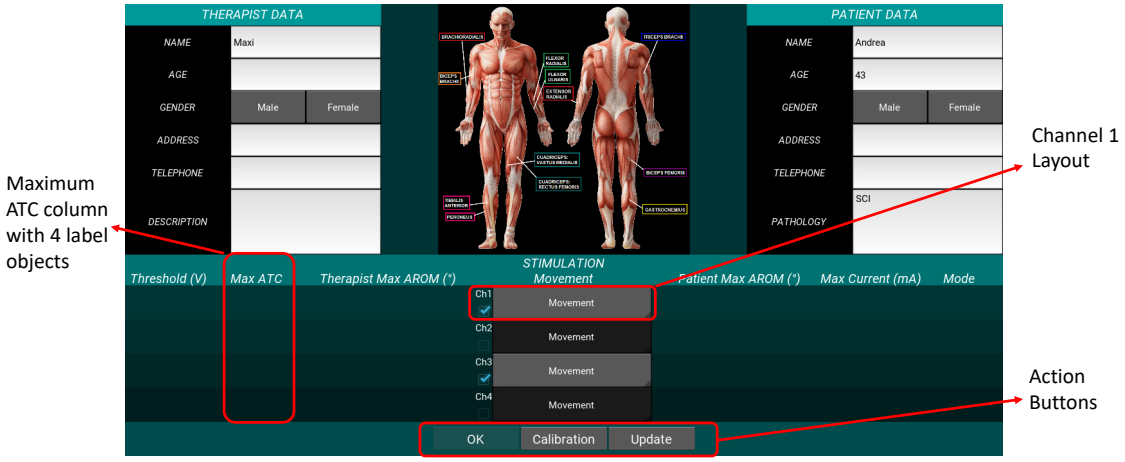


Figure 6.4: Login Screen.

The screen possesses five different behaviours, associated with widgets include at the screen:

1. **on_pre_enter:** it is executed before the screen is shown but once it has been called. Specifically here, it cleans all the variables used in the application, in other words, it erases, in every label, all the previously set messages, and redefines the system variables to their default value. Besides that, the *Channel_Layout* is disabled and then a scanning procedure is begun, by the current Bluetooth instance composing the system.

2. **update_available_channels:** it is called by pressing the *Update* button

and executes a scanning of the available channels, activating the layouts of the founded ones.

3. **update_parameters:** here, every time a new movement is chosen, and if the therapist or patient names correspond with pre-saved data, their parameter values are printed at the pertinent *Label* objects.

4. **fast_stimulation:** when a channel has been checked and the therapist and patient parameter values are loaded, the pressing of the *OK* button releases this behaviour, which is in charge of updating every useful value for performing a stimulation. Firstly, it evaluates which are the chosen channels for the stimulation and changes, to their pre-loaded values, the attributes in all the relevant devices composing the system. For instance, it updates the system *ATC_max* and *threshold_list* attributes, also, the FES *Channel_Init* and *max_current_list* properties. Secondly, it connects the system and updates the channels threshold or thresholds, relying on the utilised BLE acquisition system and switches to the Plot screen.

5. **calibration_setting:** on the other hand, pressing the *Calibration* button executes this simple method in which the selected channels are checked and stored, then the system is connected and the Calibration screen turns up.

6.2 Calibration Screen

It can be called from the Plot or Login screens and implements the four calibration methods described before and available in [29].

This screen inherits form the abstract class MyPlotScreen, but some specific features and behaviours are provided here.

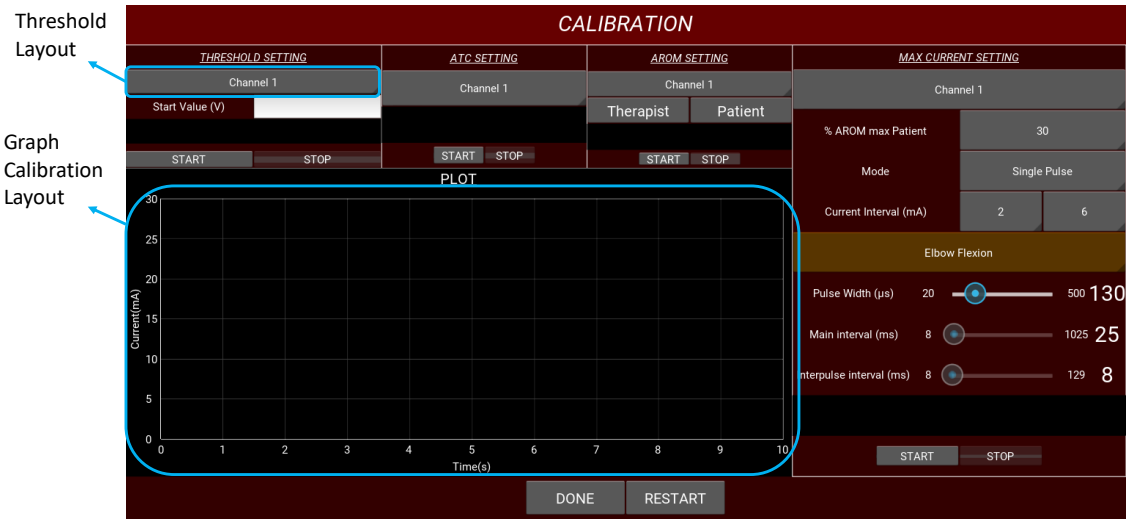


Figure 6.5: Calibration Screen.

6.2.1 Interface Overview

The most important properties are the *channels_values_for_spinners*, the *channel_threshold* and the *old_board_layout*. In the first one, the elected channels are saved, while the second and third one are the two widgets that can be added to the layout called *threshold_layout*, shown in Figure 6.5. This depends on the acquisition approach, from which the data is arriving, due to the distinct manners of handling the threshold setting in both of them.

It could trigger different methods:

- **on_pre_enter:** as in the screen before, cleans all the attributes to their default value and, using the FES *Channel_Init* attribute, it controls what are the picked channels. As a consequence, it modifies its own variable *channels_values_for_spinners*, in order to, later, update the spinner of each calibration technique with those available channels. In addition, the *threshold_layout* is added with one of the two possible widgets according to the used acquisition architecture.

- **on_leave:** in contrast, with the prior one, it is executed once the screen is not shown any more and removes the added widget for avoiding an accumulation of them at the *threshold_layout*. Moreover, it redefines the FES attribute that determines the picked channels, due to a possible alteration on it that could be made by the **current_setting** calibration technique, which only invokes one channel at the time. This shall be done like this because the *Channel_Init* attribute is employed by the FES instance for communicating to the device which are going to be the implemented channels. As in the fourth calibration step is desired to set the maximum current for one channel/muscle at the time, this variable must be reinitialised to the user initial configuration.

In addition, every time a *Start* button is pressed, in any of the calibration sections, the method with the same name is triggered, and the same thing occurs with the *Stop* button.

In general, when any method is started all the buttons are disabled except for the *stop* one of the same calibration procedure, also, the variables for selecting the channels that will be actually plotted are modified, the correct graph or graphs are added to the *graph_calibration_layout* and the **start** of the plotting is called (method inherited from MyPlotScreen). Finally, the system equivalent method is invoked, being this object the one that actually processed the information.

The *Done* and *Restart* buttons switch to the Plot and Login screen, respectively.

6.3 Plot Screen

As the previous screen, it inherits from the MyPlotScreen class, so it shares the same methods and attributes.

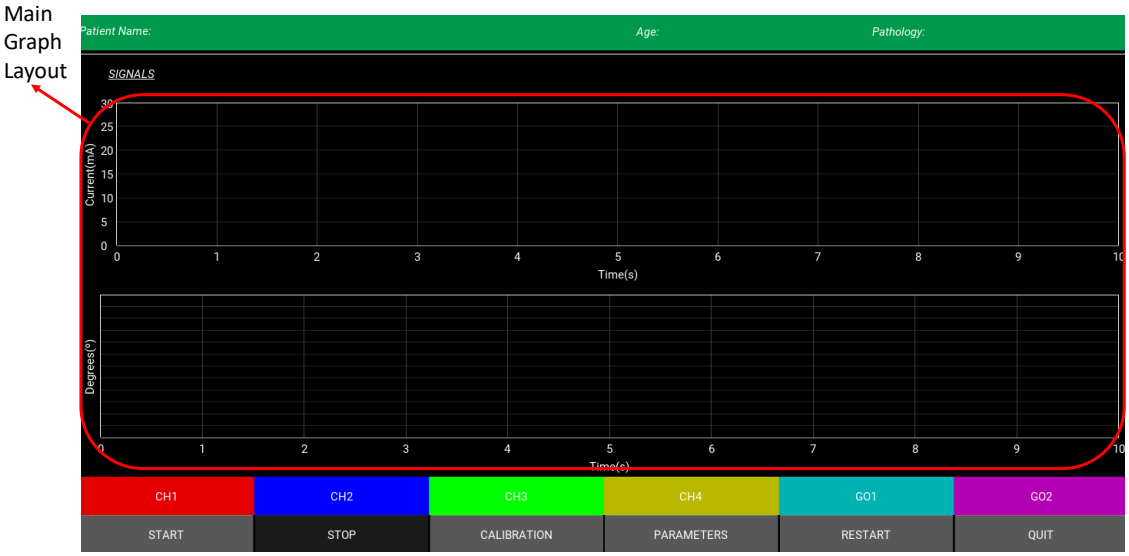


Figure 6.6: Plot Screen.

6.3.1 Interface Overview

It does not involve outstandingly properties to be discussed here, but its interface, aggregates some new behaviours to the abstract class.

First of all, the **on_pre_enter** one is implemented in the way that Patient relevant information is printed on the top of the screen and graphs are added to the *main_graph_layout* (see Figure 6.6), whereas the **on_leave** behaviour is in charge of clearing them from this layout. This procedure of adding widgets before showing the screen and removing them once the application is not more there, applied here and in different other screens like with the graphs in the calibration one, reduces RAM load.

Moreover, there are several buttons that execute the following actions:

1. **Start:** initiates the **start_action_method** which disables the other buttons and enables the stop one, modifies the inherited attributes for plotting the selected signals, begins the plotting and triggers the system counterpart.

2. **Stop:** enabled only when the stimulation is occurring, it ends the plotting and the system processing, reinstating the initial condition in the screen.

3. **Parameters:** this button makes the Screen Manager to switch from this screen to the Parameters one.

4. **Calibration:** the Calibration screen can be called by pressing this button in order to adjust one or several system parameters.

5. **Restart:** the entire process can be reinitialised by coming back to the Login screen, for changing subjects, channels or even movements.

6. **Quit:** this runs the **close_app** method which disconnects the system, closing the serial ports and the link or links established with the acquisition system.

6.4 Parameters Screen

In this case, it is in charge of summarizing all the parameter values that every channel is using at the moment and allows to alter some of them, such as, the maximum current applied to the patient, and the pulse width through the chosen movement. The screen can be observed at the Figure 6.7.

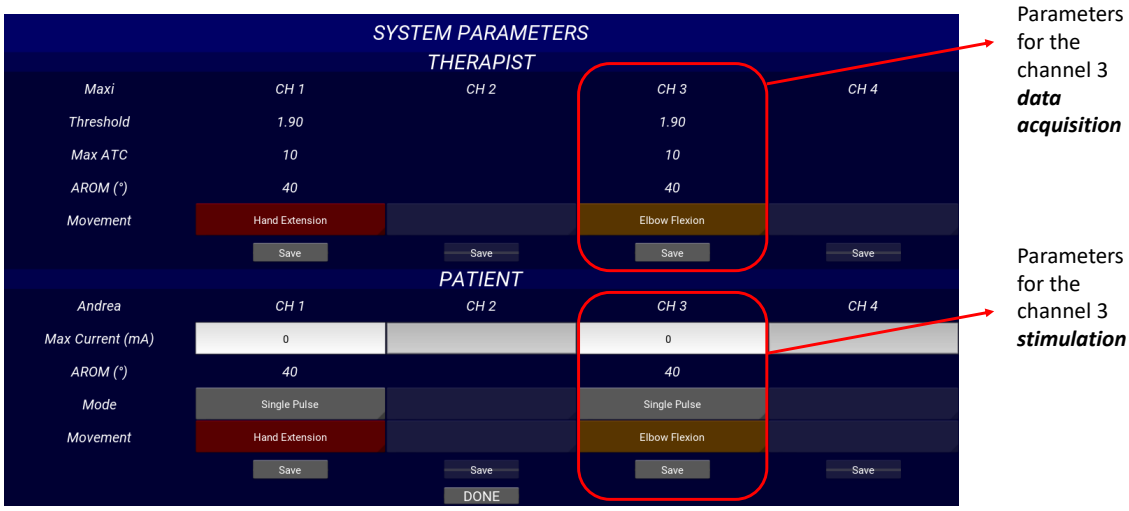


Figure 6.7: Parameters Screen.

6.4.1 Interface Overview

It involves almost the same attributes that the Login screen because it contains lists for storing references to the instances that print or permit, in some cases, to select the parameter values of every channel.

The methods are, the **on_pre_enter** one that cleans all the widgets that could be used in previous entries to the screen and update the new values, the **save_section** which is called every time a *Save* button is pressed and conserves the specific subject parameter values in the configuration file. Finally, the *Done* button returns to the Plot Screen modifying the parameter values with the new ones using the **update_parameters** behaviour.

Chapter 7

Multithreading

This project is mainly based on dealing with different devices, that interface with each other, and where the elapsed time for the information processing is required to be as low as possible. Hence, the resources need to be highly leveraged, for this, some techniques are offered, the most relevant ones are what in informatics is called as **Multithreading** and **Multiprocessing** [69].

Nowadays, the central processing units (CPUs) of computers, cellphones, and so on, are essentially made up of several processing cores integrated in a single package. It means that computers are more powerful, because they can execute real parallel independent tasks. Each *process* running in one core, can have many *threads*, being these ones the shortest piece of programmed instructions that are handled individually for the operative system, so the idea for increasing the performance is to split the program in different threads that, at the same time, can be contained in one or more processes, i.e., in one or more cores.

One central differentiation between threads and processes is that threads share the same memory, so when a thread changes a variable, it modifies the same variable that other ones might be using as well, releasing incoherences. Therefore, *thread-safe* programmes handle locks for synchronisation, in order to avoid problems. The Python interpreter is not the exception, that is why it implements a tool called **Global Interpreter Lock (GIL)** for solving this issue, but, when using the multithreading library, it boundaries the execution of one thread at the time , so in conclusion, concurrency is obtained that seems to be parallelism, but, frankly, it is not.

In contrast, the multiprocessing python library, permits the total execution of the program in different cores, taking advantage of the modern CPU architectures, like the Raspberry 3B+ has. Nevertheless, the problem emerges when a process needs to utilise a variable from another one, due to the fact that processes do not employ physically the same memory zones.

At this point, it has been decided that multithreading is more accurate than

the other approach, because in this system many of threads that are created require information contained in others to work and most of them are almost all the time in a sleeping state waiting for an event to happen, hence, the multithreading performance is enough and does not cause an unnecessary programme complexity increment, as the multiprocessing approach.

At the next Figure 7.1 a diagram illustrating the application dynamics is shown.

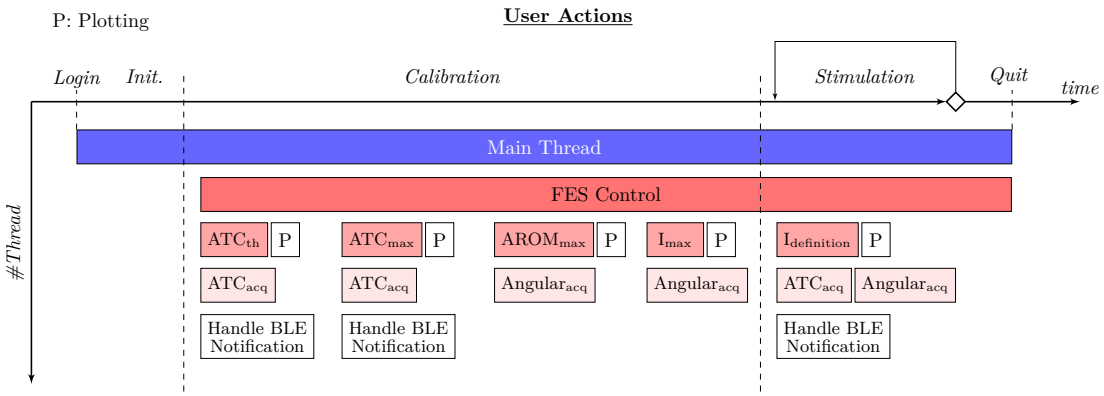


Figure 7.1: Simplified multithreading scheme of the application.

As it is possible to evaluate, the number of threads running varies relying on the procedures that the application is performing. In general, the *Main Thread* is the one waiting for user actions, whilst the *FES control* implements the reception of serial packets arriving from the stimulator and the link maintenance by sending watchdog packets steadily. There are also 5 special ones that compute the four calibration procedures and the main stimulation, they are the ATC_{th}, ATC_{max}, $AROM_{max}$, I_{max} and $I_{definition}$, respectively. In addition, the plotting thread is unleashed every time a signal needs to be plotted, like when the ATC and Angular acquisition threads are activated to receive their specific data.

The ATC acquisition thread can be distinct according the chosen acquisition structure:

- **Individual channels and Bluepy or MHC**: every single channel object requires a *Handle BLE notification* thread, in which they wait independently for a value to come, when arrives, it is sent through a queue to the BLE subsystem where another thread is gathering the four channel information in order to transmit these data packets, that are equivalent to the other implementations, to the system thread.

- **Individual channels and CC2540**: here, the situation changes, because the data flows through one serial port but the packets from different peripherals are received. Therefore, two threads are started, the *Handle BLE notification* analyses the hexadecimal numbers entering at the serial port to classify the

values relying on its origin peripheral, whereas the ATC_{acq}, creates the 4 long-value array for sending it to the post-processing.

- **Four channel board**: here, the *Handle BLE notification* is not required as the respective ATC_{acq} thread, already collects the organized data packet from the four channels.

There are other extra threads that are launched by the APIs, as Kivy or Bluepy, but those are ignore here. Then, the total maximum amount of threads defined for the script are:

1. **Nine** when the 4 channel board is implemented.
2. **Ten** for the Individual channels in combination with the Texas Instrument's dongle approach.
3. From **Ten** to **Thirteen** employing the Raspberry Pi own BLE hardware as a central device, in combination with one to four channels, respectively, as peripherals.

Finally, for a more detailed description of the application dynamics see appendix A.

Chapter 8

Results and Discussion

Two sort of analysis were done in order to assess the system performance, the first one is related with the computation loads and data latency, whereas the second evaluates rehabilitative motion benchmarks in healthy subjects.

8.1 Computational performance

8.1.1 Workstation resources usage

First of all, employing the **htop** Linux tool, the Random Access Memory (RAM) and Central Processing Unit (CPU) consumptions were tested for providing information about the employment of resources. These evaluations were performed in the worst case scenario (utilising the four individual channels and workstation own BLE hardware) and without the Raspbian GUI activated, so, only the application and all its dependencies (such as BLE and kivy ones) were running.

Table 8.1: Application maximum load values when using MHC.

Stages	***CPU (%)***	***RAM (***MB***)***
Login	20	84
Initialization	21	85
Threshold calibration	24,1	88,9
ATC maximum calibration	26	87
AROM calibration	32	89
Maximum current calibration	45,7	89
Stimulation	73,2	87,8
Parameters	15	88,3

As can be observed at the Table 8.1, the most challenging CPU performance is reached in the main stimulation procedure, where the highest amount of threads are alive. In addition, employing one single channel causes a processing usage reduction equivalent to, approximately, the 20 percent at this section of the programme. Then, it was decided to delete the ATC processing, for the stimulation current updating, occurring at the $I_{definition}$ thread; discovering equivalent values for both parameters with 1 and 4 channels, respectively (see Table 8.2).

Table 8.2: Application maximum load values when performing the $I_{definition}$ thread with 4 and 1 individual channels. *Test 1:* current new values are being computed from the ATC packets, in contrast, *Test 2:* refers to a direct equivalence between ATC/current values, i.e., without channel data processing.

	Test 1		Test 2	
Resources	Ch1	C4h	Ch1	Ch4
CPU (%)	53,8	73,2	53	74,4
*RAM (*MB*)*	87,7	87,8	91	92

Therefore, this decrement to the 50% of computational power consumption with a single sEMG channel is not related with the ATC/Current computing, but with a fewer threads implementation than with the 4 channel experiment. As an extra proof of this, on the rest of application stages, whit the single and four channel implementations, the CPU usage does not change dramatically as the number of running threads match.

On the other hand, the dynamic memory suffers just low variations in the entire four channel test due to the different amount and types of widgets the GUI owns (as shown at Table 8.1). A small difference can be detected when comparing the single and four channel memory utilization because the main consumers of it, in this application, are the graphics that kivy implements. Hence, as there are not different GUIs for a single or a four channel test, the RAM consumption stays almost unalterable (see Table 8.2).

8.1.2 Latency assessment

As mentioned before the system can adopt different architectures depending on which sEMG acquisition device it uses, as a consequence five system structures exist:

- **Case 1:** when individual channels are connect with Raspberry Pi through the

BLE master dongle[1].

- **Case 2:** the Raspberry Pi communicates with the individual channels employing its own Bluetooth hardware (up to four connections), this was also called Main Hardware Configuration (MHC).
- **Case 3:** when the four channel acquisition board is linked with the Raspberry Pi through the dongle.
- **Case 4:** when a regular Personal Computer (PC) uses the master dongle to communicate with the individual channels.
- **Case 5:** similar to the priorly mentioned case, but here, the PC is linked with the acquisition board, using the dongle.

In order to evaluated the above cited cases, a three minute stimulation employing the maximum possible number of channels, for each case, was followed. Moreover, a Toshiba Satellite L830-14J PC, owning an Intel Core i3-3227U with 1.9 GHz of clock frequency, 4 GB of RAM and Windows 10 as operative system, has been used as a workstation, at the last two cases.

In the following an analysis of two crucial sections of the application is presented.

Current definition

The time profiling of this thread can be observed at Figure 8.1, it corresponds to the second case when 4 channels have been linked to the workstation, but this scheme is repeated in any architecture because the code does not change. It is possible to see that the time is mainly spent at the *queue* object, almost 91 percent, where the script is waiting for the data to arrive. In contrast, the other called subfunctions, indicated in blue, only represent a small percentage, therefore the programme works the way it is supposed to do.

In Figure 8.2 the processing time is shown, this is the elapsed time between a new ATC packet arrival and the FES current update. As can be seen, for all the cases, the values are mostly under the 130 ms threshold whose limitation, has to be supported by the application, in order to avoid a delay between the movement and the stimulation. It is true that, the first three cases using a Raspberry Pi own a worse performance than the ones employing a standard personal computer, but, they achieve an outstandingly effectiveness, because more than the 99 percent of the data is under the before named limit, hence, any of these system architectures fulfil the real-time multi channel application.

[1]Remember that the CC2540 master dongle, utilised in this application, only manages up to three connections contemporaneously.

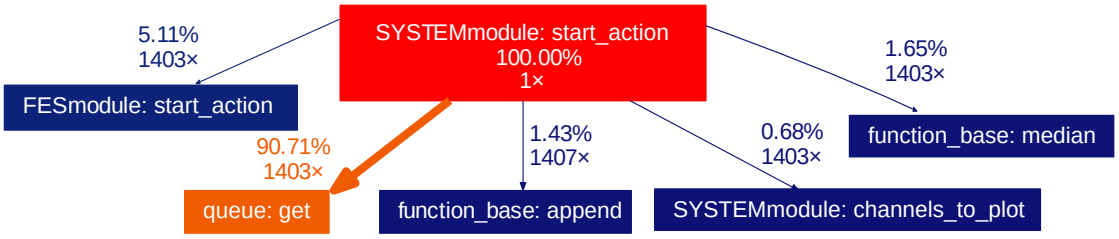
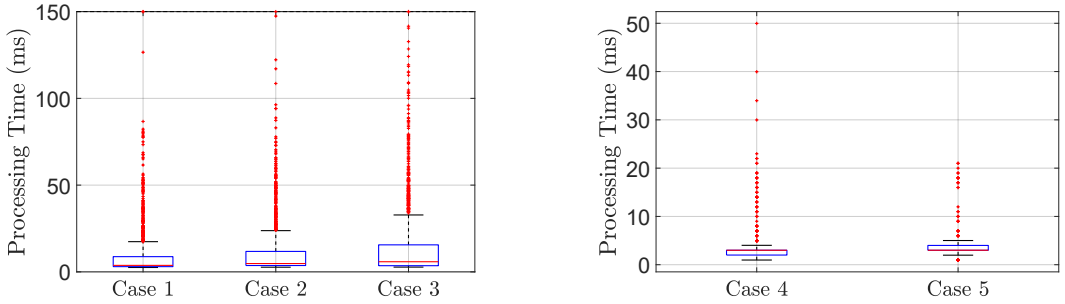


Figure 8.1: Time profiling of the Current definition thread. The function and subfunctions that this thread runs are shown with their number of calls and time execution percentages, where a 100% corresponds to 182,859 s.



(a) (b)

Figure 8.2: Data processing latency for the five possible system architectures employing the maximum number of channels for each case. The cases that have the Raspberry Pi and the regular PC as a workstation are presented in (a) and (b), respectively.

Lastly, in a previous work 2.4, the maximum amount of channels implemented was 3, having a mean elaboration time equals to 160 ms. Here, using a low resource computer and 4 individual channels, a value of 11,8 ms was accomplished.

Plotting

As for the current definition thread, the plotting one is assessed. It consists in the **clock** method, which is in charge of sleep 70 ms and afterwards, obtain the signals new value for plotting, through the **get_value** method. In Figure 8.3, a correct time profiling of this thread can be seen, because in comparison with the sleeping state, the graphs updating procedure is insignificant.

Furthermore an analysis of the **get_value** method was done, measuring the processing time occupied for the plotting, in other words, the time that it takes to the thread to plot the new six (4 currents and 2 angular) values on the two graphs at the *Plot Screen*. After three minutes of collecting data, the last two cases

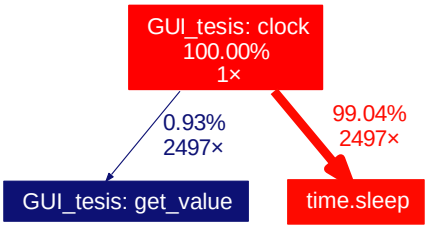


Figure 8.3: Time profiling of the Plotting thread. The function and subfunctions that this thread runs are shown with their number of calls and time execution percentages, where a 100% corresponds to 182,657 s, for the second hardware case. The thread can be sleeping or updating the graphs.

generate mean values lower than 20 µs. On the other hand, the first three achieve plotting values of 625±418 µs, 657±418 µs and 672±535 µs, respectively, giving evidence of the real-time plotting capability that this system possesses.

8.2 In vivo experiments

As the system is intended for rehabilitative sessions, some tests in order to prove this characteristic were conducted. The therapist-patient approach in which one person stimulates the homologous muscle of another, was implemented; for this, two movements were selected the **Elbow Flexion** and the **Knee Extension**, while the digital-goniometer signals acquired from both subjects were compared with the normalised cross-correlation using MATLAB®:

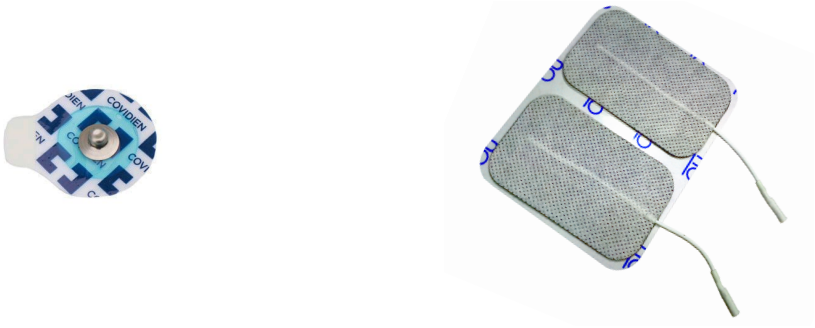
$$\hat{R}_{x,y,coeff}(m) = \frac{1}{\sqrt{\hat{R}_{x,x}(0)\hat{R}_{y,y}(0)}}\hat{R}_{x,y}(m)$$

Where m is the lag and, with the normalization, the autocorrelation with 0 lag is equal to 1. In that way, values from -1 to 1 indicate the linear similarity between the two discrete signals, being 1 or -1 for signals perfectly correlated in a direct or an inverse way. The maximum of those values, computed at different lags, is the one reported as correlation value in this project.

Electrodes and Skin preparation

Two different kind of electrodes were employed, the H124G manufactured by Convidien, for the acquisition, and the RehaTrode produced by Hasomed®, for the electrical stimulation (shown in Figure 8.4). The formers are round-shaped pre-gelled electrodes, made up of a polymer Ag-AgCl coated sensor and a stainless steel connector, and also, with a diameter of 24 mm. The latter is a rectangular-shaped electrode with a 5x9 cm^2 size, it also presents a pre-gelled side for increasing the

adhesion and the conductivity. A main difference between those two is the working area, having the acquisition ones a higher spatial resolution than the larger stimulation electrodes. This is the reason why an excitation of a single small muscle is not possible.



(a) (b)

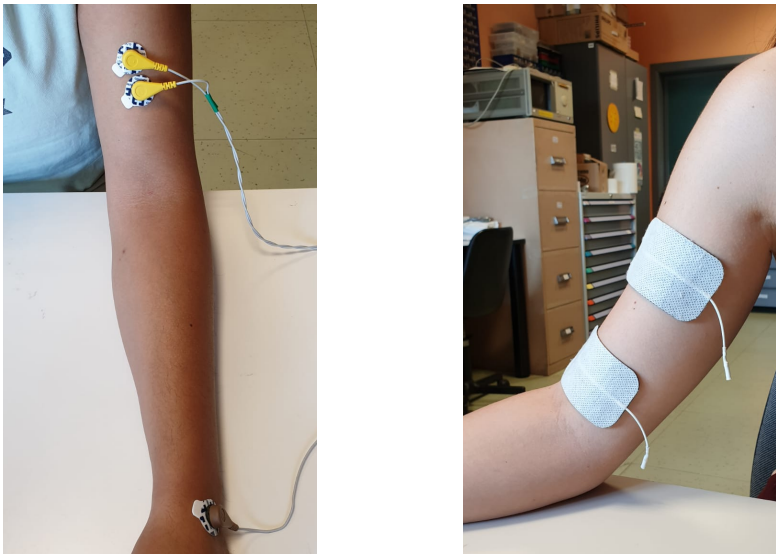
Figure 8.4: (a) Acquisition and (b) Stimulation Electrodes.

As mentioned before, the electrode-skin interface generates some perturbations, so, in order to avoid them, the skin-electrode conductivity was improved even more by using alcohol for cleaning the fat, dust and dead cells of the skin surface at the electrodes placement zones.

8.2.1 Elbow Flexion

This movement was tested on 10 healthy subjects (5 male and 5 female), with ages from 24 to 27. Twelve repetitions were performed, leaving a time interval between each one of 3/4 s.

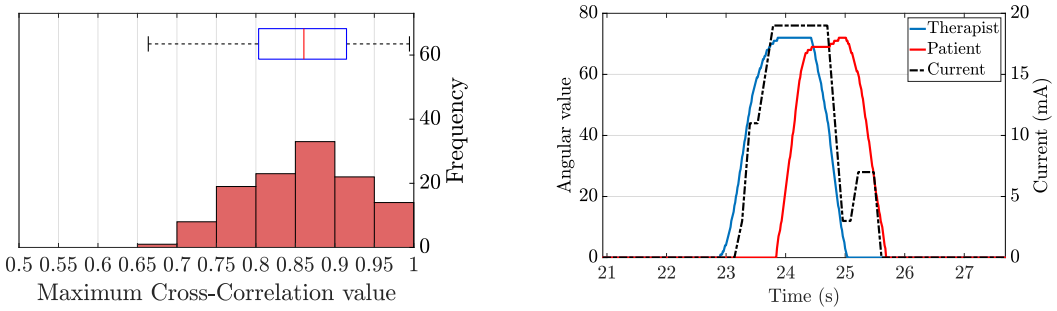
The muscles taking part in the Elbow flexion are: the *Biceps Brachii*, the *Brachialis* and *Brachio-radialis*. For fast movements the three of them are used, in spite of that, here only the main elbow flexor, the *Biceps Brachii*, was stimulated, which is enough for the test. The acquisition electrodes were placed in correspondence with the line linking the acromion and the cubital fossa, at 1/3 from the latter, whereas, the reference electrode was located on the wrist [70]. In contrast, the stimulation ones were attached close to the crease of the elbow and on the muscle belly [71]. For a graphic explanation see Figure 8.5. Furthermore, the subjects position is set with both sitting upright, with their forearms and hands completely lean against the table, and forming a 90°angle with the upper arm.



(a) (b)

Figure 8.5: Elbow Flexion electrodes placement. In (a) and (b) the acquisition and stimulation electrodes positions are shown.

The results can be observed at the Figure 8.6, where an example epoch with all the signals of interested can be observed, moreover, cross correlation values are represented in a histogram and a boxplot, which display an acceptable behaviour, because the majority of the 120 epochs present more than a 0,8 linear similarity value. In addition, the high, 0,86, mean value and the low, 0,07, standard deviation fortifies the satisfactory outcome of the system.



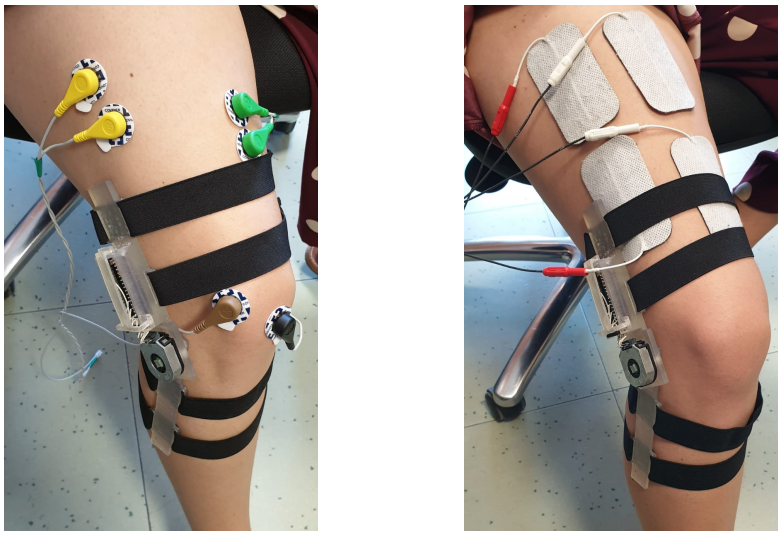
(a) (b)

Figure 8.6: Elbow Flexion Results. In (a) the combination of a boxplot and a histogram, containing the maximum cross correlation data, is shown, whilst in (b) an example epoch, with a single stimulation and two angular deflection signals, is presented.

8.2.2 Knee Extension

Finally in order to prove the multichannel performance, one healthy female subject performs 13 times a therapist-induced *Knee Extension* movement. Besides, as before the correlation between the two angular signals, measured with the digital-goniometers, was calculated.

This movement is mainly induced for the *Quadriceps Femoris*, a four-headed muscle, which occupies the front and sides of the femur. It is composed by the *Rectus Femoris*, the*Vastus Medialis*, the *Vastus Lateralis* and the *Vastus Intermedius*, so here, the acquisition electrodes were situated on the *Vastus Medialis* and *Lateralis*. In the first case, they were placed at the 80% of the line from the anterior superior iliac spine and the medial side of the patella, while in the second case, at 2/3 of the line from the anterior superior iliac spine with the superior lateral side of the patella. Both reference electrodes were located on the patella. On the other hand, the stimulation electrodes are located along the muscles bellies, in order to create a surface including both muscles and the *Rectus Femoris*, as shown in Figure 8.7. To conclude, the subjects position consists in a 135 °angle between the thigh and the calf.



(a) (b)

Figure 8.7: Knee Extension electrodes placement with goniometers in situ. In (a) and (b) the acquisition and stimulation electrodes positions are shown, respectively.

As can be assessed in Figure 8.8, also here there is a reproduction of therapist movement.

As a consequence of the low amount of data collected for this movement the boxplot and histogram are not significant here. Nevertheless an important linear similarity was initially proved owing to a high mean of maximum cross-correlation

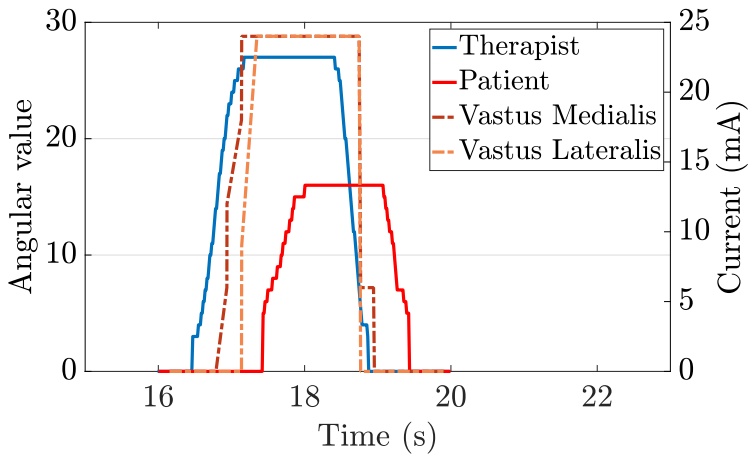


Figure 8.8: Epoch with the signals of interested at a Knee Extension. Therapist an Patient angular data is represented with continuous lines and the current signals, stimulating both vastus, with dotted ones.

values, equivalent to 0.95±0.03, was founded. In addition, the reduced amplitude of the patient angular deflection can be originated by a higher angle in the initial position, which shrinks the stimulated movement, or due to the employment of a not high enough maximum current amplitude.

Chapter 9

Conclusion

An embedded real-time cross-platform system, for commanding a multichannel FES stimulator from data acquired with an event-driven technique, called ATC, was successfully created in this thesis.

The implementation of *multithreading* and *OOP* techniques in Python, leverage the workstation for improving the real time performance without wasting computational and graphical resources. Besides, Kivy, in combination with Python, allows to fulfil the cross-platform aim, thanks to the capability of both of them to work in different operative systems.

This project achieves a proper functionality with several hardware structures, one of them is the utilisation of individual sEMG/ATC acquisition channels, whose firmware was coded in this work in order to communicate with the workstation, set an individual threshold and compute the ATC values from the sEMG signal. Another option is to measure the muscular signal from a four channel acquisition board, which can broadcast data, using the BLE protocol, with different types of workstations in combination with a master BLE dongle or, in some cases where Linux is the chosen platform, employing the workstation own Bluetooth capabilities. In addition, the GUI, created for the user to control the stimulation and calibration procedures, is robust enough for self-adapting to the hardware configuration being used.

The resources usage test proves that the MHC, using a Raspberry Pi 3B+, is suitable for carry system requirements out, because an steady and contained RAM consumption, lower than 100 MB, and a regular CPU load, were found. Nevertheless, a variation in the number of measuring channels lead to a rise in the number of threads running contemporaneously, which causes peaks, near the 70 percent, of computational resources utilisation.

Furthermore, the latency of the current update for the five hardware architecture cases were evaluated on three minutes of stimulation, finding that all of them own a median and mean value quite under the ATC packet arrival time interval, of 130 ms. Specifically an important improvement in comparison with [23], where the fifth

case with the processing of three channels was implemented, was attained; here, employing even one more channel and a modulation resolution of 1 ATC packet[1], a reduction of the 92,5 % of the mean elaboration time is accomplished.

On the other hand, two movements were selected for studying the therapist movement imitation by the patient:

1. *Elbow Flexion:* the *Biceps Brachii* was stimulated electrically on 5 male and 5 female healthy subjects, obtaining a mean maximum correlation value of 0.86±0.07 which is a valuable linear correlation between the therapist and patient movement.

2. *Knee Extension:* the three more superficial heads of the *Quadriceps Femoris* were contracted 13 times for one single female subject in order to assess the multichannel performance of the system. Lower AROM values were registered for the pseudo-patient, this might be a consequence of a wider initial relative position which reduces the movement angular path. In spite of that, extremely satisfactory values of similarity were obtained, being the mean and the standard deviation equal to 0.95 and 0.03, respectively.

To sum up a valuable embedded system was developed, which has the following main characteristics:

1. *real-time performance*, with a mean current update period of approximately 130 ms due to the use of a low power acquisition system based on the ATC technique.

2. *cross-platform*, achieved with the aid of portable programming languages and libraries.

3. *multichannel performance*, accomplished with multithreading and object oriented programming approaches.

4. And lastly, the system was proved to be suitable for a *minimalist single board computer* and for *therapist-patient active stimulation* method.

9.1 Future Perspective

Despite of all this acceptable and positive outcomes, future analysis and research need to be conducted.

[1]In [23] using three channels constraints the current update to halt for 3 ATC packets, being the current modulation resolution based on these ATC values.

First of all, more robust goniometers, with wireless communication, need to be implemented in order to measure complex joint movements, such as the ones from the foot and wrist, and to complete the four channel system.

Secondly, a biomechanical study of the FES is required for understanding deeply which are the better calibration procedures for setting the stimulation and acquisition parameters in a better way. In addition, this can be used for adding some other parameters to the real time control in function of the ATC packets or even of the digital-goniometer signals. For instance, is well-known that our neurons fired action potentials to the muscles at higher frequencies in the beginning of the movement (rapid contraction onset) and lower after, for avoiding fatigue [30], then a variation of the *Mode* parameter of the FES can follow the ATC values as shown in Figure 9.1, where every time a value is new, it is set as a *Triplet*, while whether this value is repeated one or two times, it is configured as a *Doublet* or a *Single*, respectively.

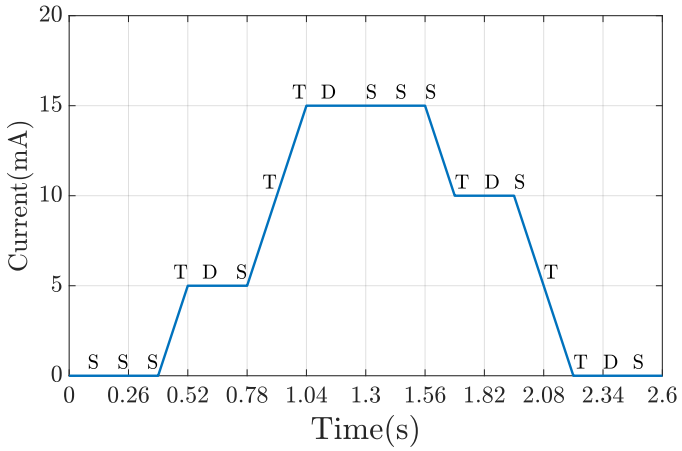


Figure 9.1: FES frequency modulation using the *Mode* FES parameter. It is modified in function of the current values repetition, where T is triplet, D is doublet and S is Single.

Finally further tests with more sorts of movements and a wider group of subjects with distinct characteristics (such as weight and age) need to be done, also, a real evaluation of the therapeutic and rehabilitative benefits in people with neuromuscular issues is required, in order to describe the actual medical power of this work.

Appendix A

Detailed Application Flowchart

In the Chapter 7, the multithreading performance was described, here, a complete flowchart is presented in order to add more information about each thread specific function.

The Figure A.1 shows how the application flows through the initial screens and threads in function of the user actions, making clear the correlation between the GUI and the multitasking approach. The *Main thread* is in blue while the others are coloured.

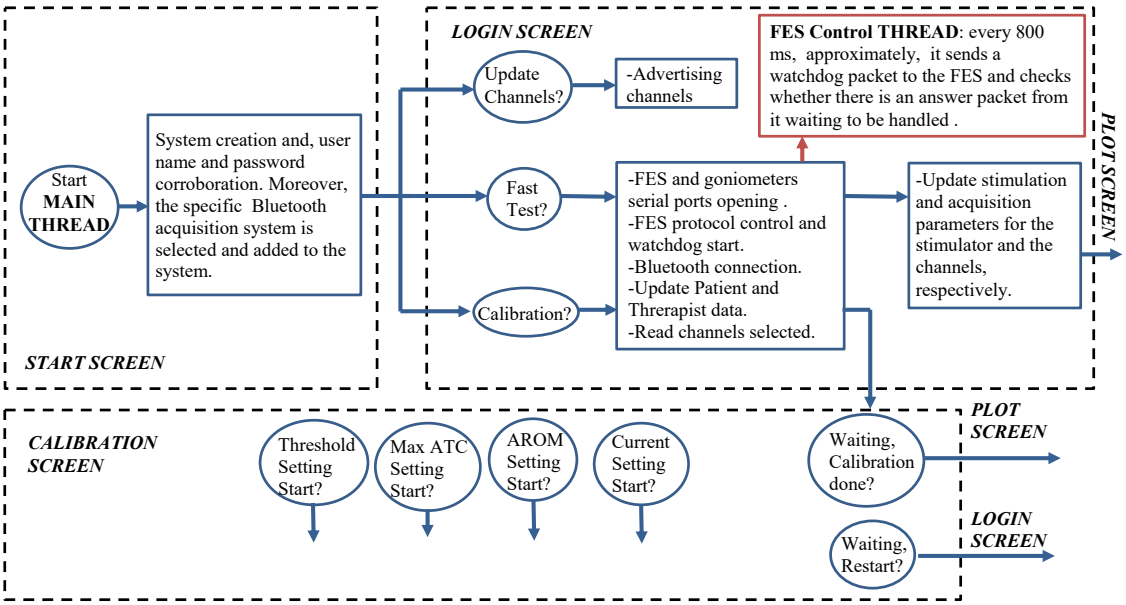


Figure A.1: Detailed Flowchart of the application's beginning. Dynamics of three screens can be observed. The four calibration procedures are not shown for clarity.

On the other hand, in the Figure A.2, the flowchart of the Plot screen is presented with the specific actions that the application executes. Every thread is shown with a different colour except the two *Angular processing* ones, at the **System** class, that are both in red.

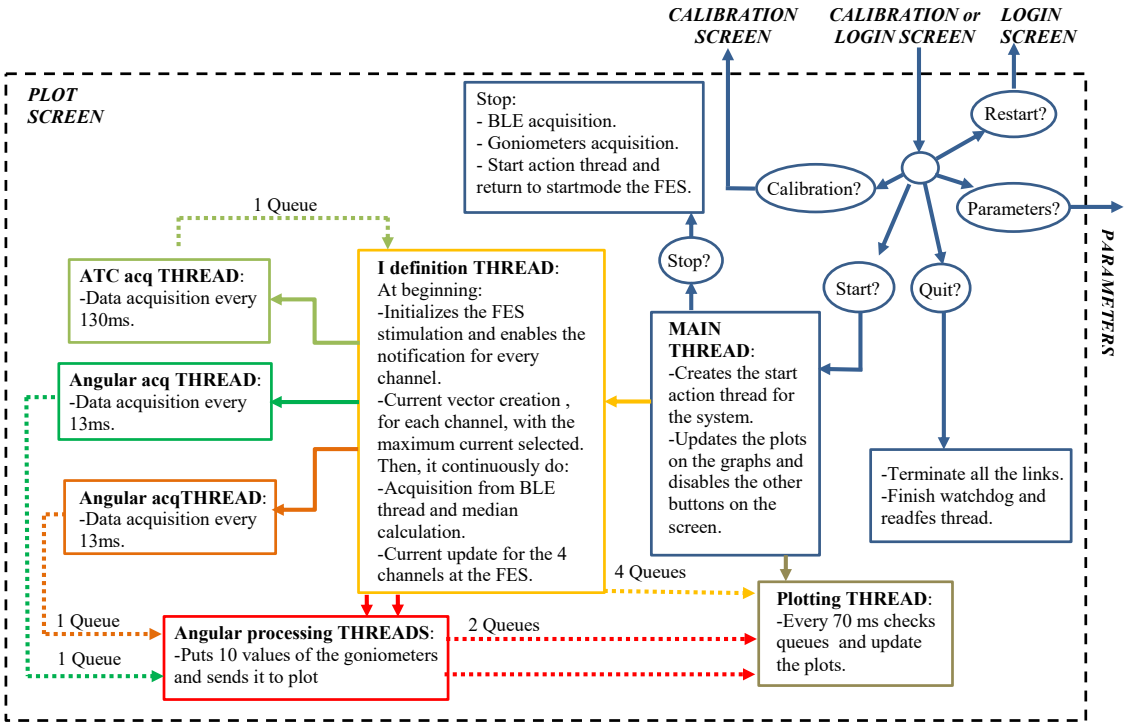


Figure A.2: Detailed Flowchart of the Main application task. The Parameters screen is neglected because it does not involve any extra thread.

Bibliography

[1] World Stroke Organization, *Facts and Figures about Stroke*, from https://www.world-stroke.org/component/content/article/16-forpatients/84-facts-and-figures-about-stroke, Accessed: 29 Jun 2019.

[2] World Health Organization, *Fact Sheets: Spinal Cord Injury*, from https://www.who.int/news-room/fact-sheets/detail/spinal-cord-injury, Accessed: 29 Jun 2019.

[3] John E. Hall and Arthur C. Guyton, *Textbook of Medical Physiology*, 12th ed., Saunders Elsevier, pp. 71-78, 2011.

[4] Anthony L. Mescher, *Junqueira's Basic Histology*, 14th ed., McGraw-Hill Education, 2016.

[5] Arthur C. Guyton and John E. Hall, *Textbook of Medical Physiology*, 12th ed., Saunders Elsevier, 2011.

[6] Kim E. Barrett, Scott Boitano, Susan M. Barman and Heddwen L. Brooks, *Ganong's Review of Medical Physiology*, 23th ed., McGraw-Hill Medical, Chapter 5, 2010.

[7] Matthew C. Gash and Matthew Varacallo, *Physiology, Muscle Contraction*, Treasure Island (FL): StatPearls Publishing; Dec 2018.

[8] Ivana Y. Kuo and Barbara E. Ehrlich, *Signaling in Muscle Contraction*, Cold Spring Harb Perspect Biol., 2015 Feb, doi: 10.1101/cshperspect.a006023.

[9] Gavin Tucker, *Different Lifting Motions*, Weight Lifting, from https://rawpress760425330.wordpress.com/, Accessed: 30 Jun 2019.

[10] Nurhazimah Nazmi, Mohd A. Rahman, Shin-Ichiroh Yamamoto, Siti A. Ahmad and Saiful A. Mazlan, *A Review of Classification Techniques of EMG Signals during Isotonic and Isometric Contractions*, Sensors (Basel), 2016 Aug; 16(8): 1304, doi: 10.3390/s16081304.

[11] Duane Knudson, *Fundamentals of Biomechanics*, 2nd ed., Springer Science, 2007.

[12] Rubana H.Chowdhury, Mamun B. I.Reaz, Mohd A. B. M.Ali, Ashrif A. A.Bakar, Kalaivani Chellappan and Tae G.Chang, *Surface Electromyography Signal Processing and Classification Techniques*, Sensors(Basel), 2013 Sep, doi: 10.3390/s130912431.

[13] Roberto Merletti and Dario Farina, *Analysis of intramuscular electromyogram*

signals, The Royal Society Publishing, 2008 Nov.

[14] Carlo J. De Luca, Alexander Adam, Robert Wotiz, L. Donald Gilmore and S. Hamid Nawad, *Decomposition of Surface EMG signals*, J Neurophysiol 96, 2006 May, doi:10.1152/jn.00009.2006.

[15] Roberto Merletti and Philip A. Parker, *Electromyography (Physiology, Engineering, and Noninvasive Applications)*, Jonh Wiley and Sons, Chapter 2, 2004 IEEE.

[16] Roberto Merletti and Philip A. Parker, *Electromyography (Physiology, Engineering, and Noninvasive Applications)*, Jonh Wiley and Sons, pp. 27-30, 2004 IEEE.

[17] Kasun Samarawickrama, Sadun Ranasinghe, Yasoja Wickramasinghe and Wageesha Mallehevidana, *Surface EMG Signal Acquisition, Analysis and Classification for the Operation of a Prosthetic Limb*, International Journal of Bioscience, Biochemistry and Bioinformatics, vol. 8, no.1, pp. 32-41, 2018.

[18] Vladimir Medved and Mario Cifrek, *Kinesiological Electromyography*, 2018 Sep, doi: 10.5772/21282.

[19] Luca Mesin, *Introduction to Biomedical Signal Processing*, 2nd ed., GEDI, Chapter 7 and 8, 2017.

[20] Joseph D. Bronzino, *The Biomedical Engineering Handbook - Medical Devices and Systems*, 3rd ed., CRC Taylor and Francis, Chapter 47 , 2006.

[21] Carlo J. de Luca, *The Use of Surface Electromyography in Biomechanics*, The International Society for Biomechanics, 2002.

[22] D. Gordon E. Robertson, Graham E. Caldwell, Joseph Hamill, Gary Kamen and Saunders N. Whittlesey, *Research Methods in Biomechanics*, Human Kinetics, pp 166-168, 2004.

[23] Fabio Rossi, *Low Power System for Event-Driven Control of Functional Electrical Stimulation*, Master Thesis - Politecnico di Torino, 2017 Jul.

[24] Ivan Furfaro, *Integration and validation of Average Threshold Crossing (ATC) applied to surface Electromyography (sEMG)*, Master Thesis - Politecnico di Torino, 2015 Oct.

[25] Karl J. Aström, *Event Based Control*, in Analysis and Design of Nonlinear Control Systems, Springer, 2008.

[26] Stefano Sapienza, Marco Crepaldi, Paolo Motto Ros, Alberto Bonano and Danilo Demarchi, *On Integration and Validation of a Very Low Complexity ATC UWB System for Muscle Force Transmission*, IEEE Transactions on Biomedical Circuits and Systems, Vol. 10, no.2, Apr 2016.

[27] Paolo Motto Ros, Marco Paleari, Nicoló Celadon, Alessandro Sanginario, Alberto Bonanno, Marco Crepaldi, Paolo Ariano and Danilo Demarchi, *A Wireless Address-Event Representation System for ATC-Based Multi-Channel Force Wireless Transmission*, IEEE, 2013, doi: 10.1109/IWASI.2013.6576061.

[28] David A. F. Guzman, Stefano Sapienza, Bianca Sereni and Paolo Motto Ros

Very Low Power Event-Based Surface EMG Acquisition System with Off-the-shelf Components, IEEE Biomedical Circuits and Systems Conference (BioCAS), 2017, doi:10.1109/biocas.2017.8325152.

[29] Sofia Cecchini, *Average Threshold Crossing Validation for Functional Electrical Stimulation applied to surface ElectroMyoGraphic signals*, Master Thesis - Politecnico di Torino, 2018 Jul.

[30] D. N. Rushton, *Functional Electrical Stimulation*, Physiological Measurement, Vol. 18, no.4, pp. 241-275, 1997, doi: 10.1088/0967-3334/18/4/001.

[31] Raafat E. Shalaby, *Development of an Electromyography Detection System for the Control of Functional Electrical Stimulation*, Master Thesis, 2011 Jul.

[32] Cheryl L. Lynch and Milos R. Popovic, *Functional Electrical Stimulation, Closed-Loop Control of Induced Muscle Contractions*, IEEE Control Systems Magazine, 2008 Apr.

[33] P. Hunter Peckham and Jayme S. Knutson, *Functional Electrical Stimulation for Neuromuscular Applications*, Annual Reviews Biomed. Eng., 2005, doi: 10.1146/annurev.bioeng.6.040803.140103.

[34] Ahmad M. Ibrahim, *Fuzzy Logic for Embedded Systems Applications*, Elsevier Science, Chapter 1, 2004.

[35] Tammy Noergaard, *Embedded Systems Architecture, A Comprehensive Guide for Engineers and Programmers*, Elsevier Science, Section 1 and 2, 2005.

[36] Thomas A. Hezinger and Joseph Sifakis, *The Embedded Systems Design Challenge*, Springer, 2006.

[37] Martin Grant, *Unified Modelling Language for Embedded Systems Specification and Design: Motivation and Overview* , IEEE Proceedings 2002 Design, Automation and Test in Europe Conference and Exhibition, 2002 March, doi: 10.1109/DATE.2002.998386.

[38] Alberto Sangiovanni-Vicentelli and Martin Grant, *Platform-Based Design and Software Design Methodology for Embedded Systems*, IEEE Design and Test for Computers, Vol. 18, no. 6, pp. 23-33, 2001 Nov, doi: 10.1109/54.970421.

[39] Muskuloskeletal Key, *Structure and Function of the Elbow and Forearm Complex*, Kinematics, from https://musculoskeletalkey.com/structure-and-function-of-the-elbow-and-forearm-complex/, Accessed: 30 Jun 2019.

[40] Stacie J. Fruth, *Fundamentals of the Physical Therapy Examination*, Jones and Barlett Learning, Chapter 9, 2018.

[41] Silvio Nussbaumer, Michael Leunig, Julia F. Glatthorn, Simone Sauffacher, Hans Gerber, Nicola A. Maffiuletti, *Validity and test-retest reliability of manual goniometers for measuring passive hip range of motion in femoroacetabular impingement patients*, BioMed Central, 2010.

[42] Michael J. Mullaney, Malachy P. McHugh, Christopher P. Johnson and Timothy F. Tyler, *Reliability of shoulder range of motion comparing a goniometer to a digital level*, Physiotherapy Theory and Practice, Vol. 26, no. 5, pp 327-333,

doi: 10.3109/09593980903094230, 2010 Jun.

[43] Ali Nikoukar, Mansour Abboud, Borna Samadi, Mesut Günes and Behnam Dezfouli, *Empirical Analysis and Modeling of Bluetooth Low-Energy (BLE) Advertisement Channels*, IEEE 2018 17th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net), pp 1-6, doi: 10.23919/MedHocNet.2018.8407089, 2018 Jun.

[44] Kevin Townsend, Carles Cufí, Akiba and Robert Davidson, *Getting Started with Bluetooth Low Energy*, O'Really Media, 2014.

[45] Jacopo Tosi, Fabrizio Taffoni, Marco Santacatterina, Roberto Sannino and Domenico Formica, *Performance Evaluation of Bluetooth Low Energy: A Systematic Review*, Sensors, Vol. 17, no. 12, 2017 Dec.

[46] Carles Gomez, Joaquin Oller and Josep Paradells, *Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology*, Sensors, Vol. 12, no. 19, doi:10.3390/s120911734 , 2012 Aug.

[47] Mingzhe Jiang, Tuan Nguyen gia, Arman Anzanpour, Amir M. Rahmani, Tomi Wedterlund, Sanna Salantera, Pasi Liljeberg and Hannu Tenhunen, *IoT-based Remote Facial Expression Monitoring System with sEMG Signal*, IEEE, doi:10.1109/SAS.2016.7479847 , 2016 Apr.

[48] Davide Brunelli, Elisabetta Farella, Davide Giovanelli, Bojan Milosevic and Ivan Minakov, *Design Considerations for Wireless Acquisition of Multichannel sEMG Signals in Prosthetic Hand Control*, IEEE Sensors Journal, Vol. 16, no. 23, pp 8338-8347, doi:10.1109/JSEN.2016.2596712 , 2016 Dec.

[49] Stefano Sapienza, Paolo Motto Ros, Fabio Rossi, Rossana Terracciano, Elisa Cordedda and Danilo Demarchi, *On-Line Event-Driven Hand Gesture Recognition Based on Surface Electromyographic Signals*, IEEE International Symposium on Circuits and Systems (ISCAS), pp 1-5, doi:10.1109/ISCAS.2018.8351065 , 2018 May.

[50] Dingguo Zhang and Wei T. Ang, *Reciprocal EMG Controlled FES for Pathological Tremor Suppression of Forearm*, Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp 4810-4813, doi: 10.1109/IEMBS.2007.4353416, 2007 Aug.

[51] Daniel Graupe, *EMG Pattern Analysis for Patient-Responsive Control of FES in Paraplegics for Walker-Supported Walking*, IEEE Transactions on Biomedical Engineering, Vol. 36, no. 7, 1989 Jul.

[52] Markus Valtin, Kristian Kociemba, Carsten Behling, Björn Kuberski, Sebastian Becker and Thomas Schauer, *RehaMovePro: A versatile mobile stimulation system for transcutaneous FES applications*, European Journal of translational myology, Vol. 26, no. 3, doi: 10.4081/ejtm.2016.6076, 2016 Jun.

[53] Andrés F. Ruiz-Olaya, *On the use of wearable sensors to enhance motion intention detection for a contralaterally controlled FES system*, IEEE 13th International Conference on Wearable and Implantable Body Sensor Networks (BSN), pp 324-328, doi: 10.1109/BSN.2016.7516282, 2016 Jun.

[54] Keisuke Shima and Koji Shimatani, *A New Approach to Direct Rehabilitation Based on Functional Electrical Stimulation and EMG Classification*, International Symposium on Micro-NanoMechatronics and Human Science (MHS), pp 1-6, doi: 10.1109/MHS.2016.7824200, 2016 Nov.

[55] Bianca Sereni, *Design and development of a low-power wearable device for the acquisition of surface electromyography (sEMG) signals with average threshold crossing (ATC)*, Master Thesis - Politecnico di Torino, 2016 Oct.

[56] HASOMED, *Operation Manual RehaStim2 and RehaMove2*, HASOMED GmbH, version 1.4, 2012 Sep.

[57] HASOMED, *RehaMove, FES Cycling, Functional Electrical Stimulation, Templates, External Trigger, Studies, Science Mode, Sequence Training FES Walking, Sit to Stand and EMG*, HASOMED GmbH, 2016.

[58] Bjoern Kuberski, *Science Mode 2, RehaStim2 Stimulation Device, Description and Protocol*, HASOMED GmbH, version 1.24, 2012 Dec.

[59] Arduino, *Arduino Micro*, from https://store.arduino.cc/arduino-micro, Accessed: 31 March 2019.

[60] Raspberry Pi Foundation, *Raspberry Pi 3 B+ Specifications*, from https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/, Accessed: 30 March 2019.

[61] Raspberry Pi Foundation, *Raspbian*, from https://www.raspberrypi.org/documentation/raspbian/, Accessed: 30 March 2019.

[62] Nordic Semiconductor, *Software and Tools*, from https://www.nordicsemi.com/Software-and-Tools/Software, Accessed: 1 May 2019.

[63] Dusty Philips, *Python 3 Object-oriented Programming*, 2nd Edition, Packt Publishing, 2015 Aug.

[64] HASOMED, *Functional Electrical Stimulation Applications*, HASOMED GmbH, version 2015-06, 2015.

[65] BlueZ, *BlueZ: Official Linux Bluetooth protocol stack*, from http://www.bluez.org/about/, Accessed: 8 May 2019.

[66] Ian Harvey, *Bluepy: Python interface to Bluetooth LE on Linux*, from https://github.com/IanHarvey/bluepy, Accessed: 8 May 2019.

[67] Kivy, *Kivy: Home*, from https://kivy.org/#home, Accessed: 11 May 2019.

[68] Roberto Ulloa, *Kivy - Interactive Applications and Games in Python*, Packt Publishing, 2015 Jun.

[69] Steven Lott, *Functional Python Programming*, Packt Publishing, Chapter 12, 2015 Jan.

[70] H.J. Hermens and B Freriks, *Surface ElectroMyoGraphy for the Non-Invasive Assessment of Muscles (SENIAM - Sensors Locations)*, from http://seniam.org/sensor_location.htm, Accessed: 27 Jun 2019.

[71] AXELGAARD MANUFACTURING CO. LTD, *Education: Electrode Placement*, from https://www.axelgaard.com/Education, Accessed: 27 Jun 2019.