BACHELOR'S DEGREE FINAL PROJECT

Double Degree in Biomedical and Industrial Electronics and Automatic Control Engineering.

FIRMWARE DESIGN OF A PORTABLE MEDICAL DEVICE TO MEASURE THE QUADRICEPS MUSCLE GROUP AFTER A TOTAL KNEE ARTHROPLASTY BY EMG, LBIA AND CLINICAL SCORE METHODS

Project Report and Annexes

Author: Arnau Diez Clos

Directors: Paco Bogónez Franco & Lexa Digna Nescolarde Selva

Convocation: June 2022

Resumen

El objetivo de este proyecto es el diseño del firmware de un dispositivo médico portátil para mediciones de EMG y LBIA, que se utilizará para la evaluación de pacientes de artroplastia total de rodilla, para estudiar la progresión de diferentes prótesis de rodilla (Medial-Pivot y Ultra-Congruente). En la tesis, se expone el conocimiento actual de los estudios y aplicaciones de EMG y LBIA, junto con los dispositivos comerciales utilizados actualmente. Además, se han estudiado e implementado las diferentes técnicas de filtrado y procesamiento digital para señales de EMG y LBIAs. Adicionalmente, se ha realizado un estudio estadístico preliminar con datos LBIA de 12 pacientes de artroplastia total de rodilla.

El diseño del firmware de esta tesis incluye: los procesos de adquisición de datos con el uso de diferentes ADCs (Conversor Analógico a Digital) (de la propia placa y externos, utilizando la interfaz SPI) y un DAC (Conversor Digital a Analógico), el correspondiente procesamiento de la señal y la extracción de sus características, la comunicación con un dispositivo externo utilizando un módulo BLE externo con interfaz UART, el proceso de encriptación de los datos médicos, la funcionalidad de manejo de errores y la aproximación del nivel de batería.

En esta tesis, todos los flujos de trabajo de los procesos se exponen y explican mediante diagramas de flujo, mientras que se justifica cada cálculo y configuración. Además, todo el código correspondiente se ha programado en lenguaje C y se expone en los anexos. También se ha revisado la normativa aplicable y se ha analizado tanto el impacto ambiental como el coste económico del producto. Por último, se proponen mejoras para futuros trabajos.

Resum

L'objectiu d'aquest projecte és el disseny del microprogramari d'un dispositiu mèdic portàtil per a mesures d'EMG i LBIA. L'aparell mèdic s'utilitzarà per a l'avaluació de pacients d'artroplàstia total de genoll per estudiar la progressió de dues pròtesis de genoll (Medial-Pivot i Ultra- Congruent). En el treball, s'exposa el coneixement actual dels estudis i aplicacions d'EMG i LBIA, juntament amb els dispositius comercials utilitzats actualment. A més, s'han estudiat i implementat les diferents tècniques de filtrat i processament digital dels senyals de EMG i LBIA. Addicionalment, s'ha fet un estudi estadístic preliminar amb dades de LBIA de 12 pacients amb artroplàstia total de genoll.

El disseny del microprogramari d'aquesta tesi inclou: els processos d'adquisició de dades fent ús de diferents ADCs (de la pròpia placa i externs, utilitzant la interfície SPI) i un DAC, el processament dels senyals i l'abstracció de les seves característiques, la comunicació amb un dispositiu extern utilitzant un mòdul BLE extern amb interfície UART, el procés d'encriptació de les dades mèdiques, la funcionalitat de l'avaluació d'errors i l'aproximació del nivell de bateria.

En aquest treball, totes les funcionalitats del dispositiu s'exposen i s'expliquen mitjançant diagrames de flux i es justifiquen els càlculs i configuracions corresponents. Tot el codi desenvolupat s'ha programat en llenguatge C i s'exposa als annexos. A més, s'ha revisat la normativa aplicable i s'ha analitzat tant l'impacte ambiental com el cost econòmic de l'aparell. Finalment, es proposen millores per a futurs desenvolupaments.

Abstract

The aim of this project is the firmware design for a portable medical device for EMG and LBIA measurements which will be used for the assessment of total knee arthroplasty patients to study the progression of different knee prostheses (Medial-Pivot and Ultra-Congruent). For its realization, the state of the art of the EMG and LBIA studies and applications are exposed, along with the currently used medical devices. In addition, the different digital filtering and processing techniques for these studies have been studied and implemented. Furthermore, a preliminary statistical study has been performed with LBIA data from 12 patients with total knee arthroplasty.

The firmware design of this thesis includes: the acquiring data processes with the use of different ADCs (from the actual board and external, using the SPI interface) and a DAC, the corresponding signal processing and feature abstraction, the communication with an external device using an external BLE module with UART interface, the medical data encrypting process, the error handling functionality, and the battery level approximation.

In this work, all the process workflows are exposed and explained using flowcharts, while every calculation and configuration is justified. In addition, all the corresponding code has been programmed using C language and exposed in the Annexes. Moreover, the applicable regulation has been reviewed, and both the environmental impact and economic cost of the product have been analyzed. Finally, improvements are proposed for future work.

Acknowledgments

I would like to thank both my tutors Francisco Bogónez and Lexa Nescolarde for guiding and inspiring me in the realization of this work and for their commitment as project directors. I would also like to acknowledge the investigators of the Hospital Germans Trias i Pujol for their help during this project. In addition, I would like to thank my friend and classmate Maxim Montero for his support during this project and the whole course of the double degree. Lastly, I want to express my gratitude to my parents for their unconditional support in each step of my studies.

Glossary

ACL	Anterior Cruciate Ligament
AD7398-4	Differential Input, Quad, Internal Reference, Simultaneous Sampling, 16-Bit SAR ADC
ADC	Analog to Digital Converter
Ag/AgCl	Silver chloride
AMP	Action Membrane Potential
AP	Access Point
BI	Bioimpedance
BLE	Bluetooth Low Energy
CH	Channel
CMSIS	Cortex Microcontroller Software Interface Standard
CTS	Clear to Send
DAC	Digital to Analog Converter
DC	Direct Current
DFT	Discrete Fourier Transform
DMA	Direct Memory Access
DNS	Domain Name System
DSP	Digital Signal Processing
EMG	Electromyogram
EMI	Electromagnetic Interference

EXTI	External Interrupt/Event
FFM	Fat Free Mass
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FM	Fat Mass
FT	Fourier Transform
GPIO	General Purpose Input/Output
HSI	High-Speed Internal Clock Source
HM-10	4.0 BLE Module
HOS	Higher Order Statistics
HSE	High-Speed External Clock source
KSS	Knee Society Score
LBIA	Localized Bioimpedance Analysis
LCL	Lateral Collateral Ligament
LED	Light Emitting Diode
LMS	Least Mean Squares
LPF	Low Pass Filter
MAV	Mean Amplitude Value
MCL	Medial Collateral Ligament
MCU	Microcontroller
MDF	Median Frequency

MISO	Master in Slave Out
MNF	Mean Frequency
MOSI	Master in Slave In
MOVAG	Moving Average
MP	Medial-Pivot Prosthesis
MU	Motor Units
MVC	Maximum Voluntary Contraction
NEMG	Needle Electromyogram
NVIC	Nested Vectored Interrupt Controller
PA	Phase Angle
PCA	Principal Component Analysis
PCL	Posterior Cruciate Ligament
PLL	Phase Locked Loop
PLN	Power Line Noise
R	Resistance
RAM	Random-Access Memory
RC	Resistor-Capacitor
RMP	Resting Membrane Potential
RMS	Root Mean Square
RX	Receive Line Pin
SBIA	Segmental Bioimpedance Analysis

SCLK	Signal Clock
SDO	Serial Data Output
SEMG	Surface Electromyogram
SPI	Serial Peripheral Interface
SRAM	Static Random-Access Memory
SS	Slave Select
SSID	Service Set Identifier
TBW	Total Body Water
TIM	STM32F4 Timer
TX	Transmit Line Pin
UART	Universal Asynchronous Receiver-Transmitter
UC	Ultra-Congruent Prosthesis
USART	Universal Synchronous Asynchronous Receiver Transmitter
Wi-Fi	Wireless Fidelity
WT	Wavelet Transform
Xc	Capacitance
Xl	Inductance
XOR	Exclusively-OR
Z	Impedance

Table of contents

Table of illustrations

1. Preface

1.1. Work origin

This project has been developed with an agreement with a PhD thesis developed by the investigators G. Pedemonte and F. Collado from the Department of Orthopaedic Surgery and Traumatology of the Hospital Germans Trias i Pujol from Badalona. Their thesis is called "Electrophysiological Comparative Study of the Neuromuscular Response of the Lower Extremities After the Implantation of an Ultra-Congruent vs Medial Pivot Total Knee Arthroplasty". The main objective of their study is to assess the evolution of total knee arthroplasty patients with two different prostheses, the Medial-Pivot and the Ultra-Congruent, focusing on the performance in semiflexion. In their work, they use two different medical devices for the measurements of LBIA and EMG. My contribution to the study is the firmware design of a portable Medical Device capable of optimize these data acquisitions, incorporating both LBIA and EMG measurement functionalities.

1.2. Motivation

During my Degree in *Biomedical Engineering*, I have developed a growing interest in electronics that led me to study a second Degree in *Industrial Electronics and Automatic Control Engineering*. During both these degrees I have studied a wide range of subjects which I have really enjoyed, but without a doubt, what interests me the most is the medical related electronics. In addition, I strongly believe that programming is an important skill to learn if you want to succeed as an electronic engineer. For this reason, when I was proposed to develop this final thesis project to end both my Degrees, I found it a great opportunity to expand my theoretical and practical knowledge in the field of electromedical devices. Furthermore, I consider it a big honor to be able to develop this project with an agreement with such a well-known and high-level hospital as Hospital Germans Trias i Pujol.

1.3. Scope

Although the agreement with the hospital comprehends the hardware and firmware development of a LBIA and EMG medical device, this thesis only encompasses the firmware design and the preliminary LBIA statistical analysis. The hardware design along with the corresponding specifications and component selection and the EMG statistical analysis are being developed in a parallel thesis project.

Furthermore, even though the communication protocol to establish a connection with an external device is done in this thesis, the application required to interact and communicate with the medical device is beyond the scope of this work. Additionally, since there is currently an international lack of electronic components, it is not intended to materialize the medical device design.

2. Introduction

2.1. Total Knee Arthroplasty

Total knee arthroplasty is a successful procedure that significantly improves the quality of life of patients by reducing pain and increasing their functional capacity. Metal and plastic-based parts are used to cap the ends of the bones that form the knee joint along with the kneecap. This process might be considered for individuals who suffer from severe arthritis or severe knee injury.

There are different types of arthritis that can affect the knee joint. The most relevant are: Osteoarthritis, a degenerative joint disease that has the most negative effects on middle-aged and older adults and can cause the breakdown of joint cartilage and the nearby bones; Rheumatoid arthritis, a disease that causes the inflammation of the synovial membrane resulting in the presence of excessive synovial fluid, causing pain and stiffness; and finally, traumatic arthritis, a type of arthritis caused by an injury that can damage the cartilage of the knee.

The main goal of the arthroplasty procedure is to resurface the damaged parts of the knee joint. Furthermore, this surgery can provide patients with pain relief and functional recovery. The combination of better surgical techniques together with new materials and designs has generated significant advances in the last years. With these advances, this technique is also focused on achieving the best possible imitation of the knee kinematics, however, to date there is not an ideal design that fully accomplishes this statement [1].

2.1.1. Knee anatomy

Joints are areas where 2 or more bones meet. These joints are mostly mobile which allows the bones to have movement. The knee is composed of two leg bones (the tibia and the femur) that are held together by a set of muscles, ligaments, and tendons. The end of these bones is covered with a layer of cartilage that absorbs shock and has a protection feature of the knee. The two bonds are in partial contact with the Patella which is the knee cap.

There are two principal groups of muscles involved in the knee, that include the quadriceps muscle group (formed by the Rectus Femoris, the Vastus Intermedius, the Vastus Lateralis and the Vastus Medialis) located on the front of the thighs, whose main function is to straighten the legs, and the hamstring muscles (composed by the Semitendinosus, the Biceps Femoris and the Semimembranosus) located on the back of the thighs, which bend the leg and the knee (Figure 2.1.1.1).

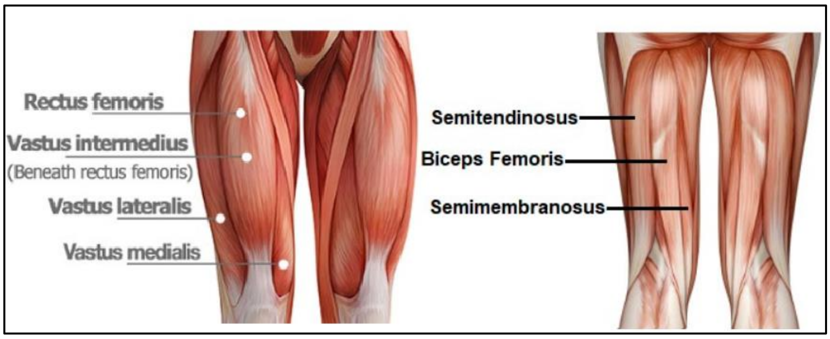


Figure 2.1.1.1.- *Muscle groups involved in the knee anatomy [2].*

Tendons are tough cords of connective tissue that attach muscles to bones. Ligaments, on the other hand, are elastic bands of tissue that connect a bone to another bone. The ligaments of the knee can be classified into two groups depending on their function. There are ligaments whose function is to provide the knee stability and protection of the joint, while other ligaments limit the movement of the tibia [2]. The main tendons involved are the Patellar tendon on the front of the knee which is part of the Quadriceps mechanism and other smaller tendons that surround the knee joint.

The four main ligaments in the knee (Figure 2.1.1.2) connect the femur to the tibia and include the following [3].

- Anterior cruciate ligament (ACL): Located in the knee centre, controls rotation and forward motion of the tibia.
- Posterior cruciate ligament (PCL): Located in the centre of the knee, controls the backward movement of the tibia.
- Medial collateral ligament (MCL): Provides stability to the knee.
- Lateral collateral ligament (LCL): Gives stability to the outer knee.

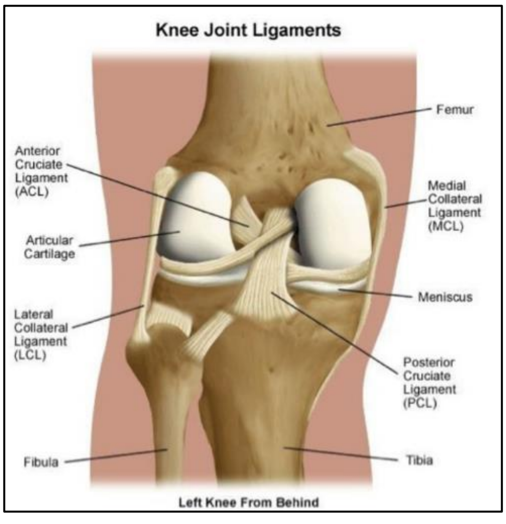


Figure 2.1.1.2.- *Joint ligaments involved in the knee anatomy [3].*

2.1.2. Clinical assessment of post-knee total arthroplasty patients focusing in semiflexion

This project is a part of a Ph.D. thesis developed by investigators of the *Hospital Germans Trias i Pujol* from Badalona with the main objective of assessing total knee arthroplasty patients with two different prostheses: the Medial-Pivot and the Ultra-Congruent, focusing on their performance in semiflexion. In this Ph.D. study, a statistic population of 80 patients (using Medial-Pivot and Ultra-congruent prosthesis) are evaluated three times, using EMG and LBIA methods to check the muscle activation (to assess the prosthesis stability) and the health of the muscle involved, respectively. A total of three evaluations are made, one prior to the operation, six months after the procedure, and finally, the patients are studied again one year after the intervention.

These two prothesis studied have functional and structural differences that will be exposed. In addition, it is worth mentioning that even though it is not required to dissect the posterior cruciate ligament of the knee if it is healthy, it has been removed in every intervention (independently prosthesis) to avoid possible future complications.

2.1.2.1. Medial-Pivot Prosthesis (MP)

The Medial Pivot knee design incorporates a cruciate retaining femoral component (however in this Ph.D. thesis the cruciate was removed in all cases as previously mentioned) and a highly congruent polyethylene liner. The medial compartment behaves as a ball and socket and pivot center, while the lateral compartment is less concave, allowing the lateral femoral condyle to roll posteriorly during flexion. In addition, the anterior lip on the polyethylene liner functionally acts to replace the PCL by limiting excessive posterior translation of the tibia [4]. Overall, the MP design is distinguished by a reduced lateral congruence with respect to the medial one. The purpose of this asymmetry is to reproduce the physiological tibiofemoral kinematics, which consists of a tibial internal rotation resulting from the coupled medial pivot and lateral femoral rollback movements during the knee flexion [5]. The commercial MP prosthesis studied in the thesis is the *Evolution* from Palex (Figure 2.1.2.1.1).

Figure 2.1.2.1.1.- *Medial Pivot Prosthesis (Evolution from Palex) [6].*

2.1.2.2. Ultra-Congruent Prosthesis (UC)

The UC bearing is a deep-dished polyethylene insert with a large anterior buildup which replaces the function of the post and thereby eliminates future complications. Stabilization is achieved with a tighter polyethylene, and they preserve bone preservation because the UC is used with a CR femoral component, eliminating the need to respect the intercondylar notch bone [7]. The UC has been designed to increase intra-operative sagittal translation and reduce posterior femoral rollback during knee flexion, and it achieves a large contact area and stability with a higher anterior than posterior lip [7].

The commercial UC prosthesis implanted in the present thesis is the *Columbus* from Braun (Figure 2.1.2.2.1).

Figure 2.1.2.2.1.- *Ultra-Congruent Prosthesis (Columbus from Braun) [8].*

2.2. Biosignals

A signal is a wave capable of transmitting information. There are many types of signals, but the present study will deal with the signals related to the biomedical environment. Therefore, will talk about bio signals or bioelectric signals [9].

2.2.1. Generation:

Cells have the ability to respond to certain stimuli due to their differential potential between their interior and exterior. Typically, the inside of the cell is more electrical negative in comparison with its exterior. That difference is called Membrane Potential (MP) and offers two different states to the cell: *resting membrane potential* and *action membrane potential.*

Resting membrane potential (RMP):

*R*esting membrane potential or RMP, can be found in those cells that are not stimulated, and its value is approximately -70 mV with respect to the exterior. The RMP depends on the intracellular and extracellular concentrations of different ions, like Sodium (Na^{+}), Potassium (K^{+}) and Chloride (Cl^{-}).

The RMP theory is represented by the equivalent circuit (Hodgkin-Huxley model) scheme (Figure 2.2.1.1). The individual equilibrium potentials produced by differences in ion concentrations act as a battery (each battery is shown as a rectangle) connected in series to a resistor. The net potential that results from connecting the three battery-resistance combinations in parallel appears across the cell membrane as a charge capacitor at -70 mV (gx is the ion channel conductance).

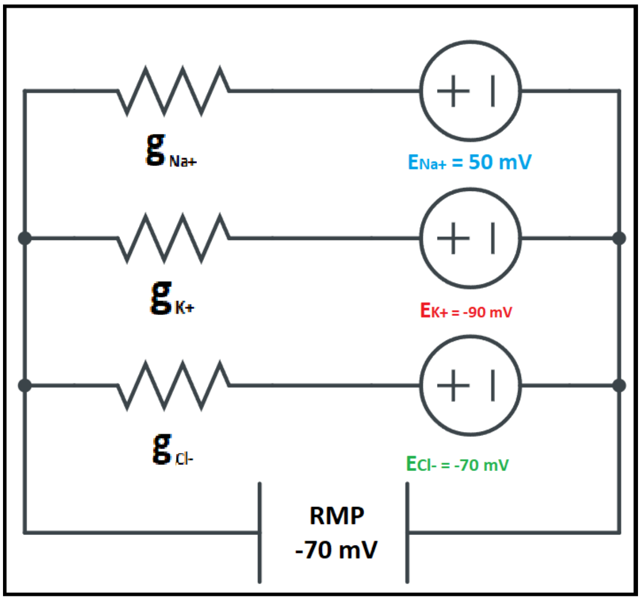


Figure 2.2.1.1.- *Electric schematic of the RMP. Adapted from [10].*

Action membrane potential (AMP):

Otherwise, whether the cell membrane is depolarized from -70 mV to about -40 mV, the cell responds with a short current impulse which changes the MP to about 25 mV and then it goes back to -75 mV (under the resting potential). That response is called Action Membrane Potential or AMP and it is a basic mechanism to allow the transport of information between cells. The process is represented in the following Figure:

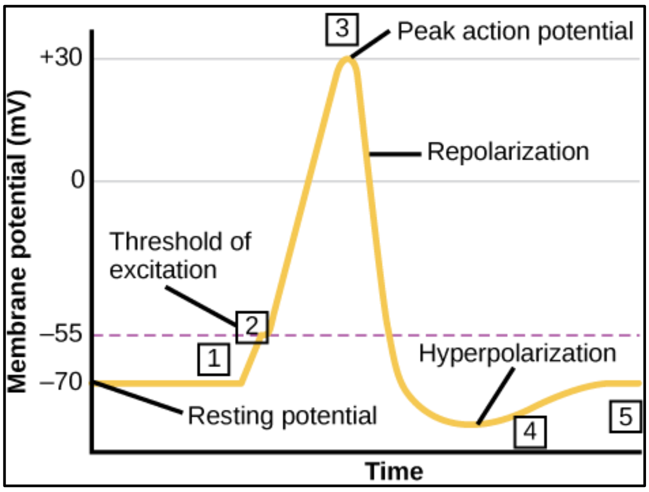


Figure 2.2.1.2.- *Schematic of an electrophysiological recording of an action potential showing the various phases that occur as the wave passes a point on a cell membrane [10].*

It is important to highlight that the residual permeability to potassium, together with the inactivation of sodium, produces a refractory period, which is characterized by being a short period in which the cell cannot be excited [10].

2.3. Bioimpedance

The electrical properties of the biologic tissues can be classified according to their response (active or passive). The active response (bioelectric signals) occurs when the tissue generates a potential difference due to the ionic activities inside the cells, such as in the case of the electromyogram (EMG) in the muscle.

The passive response occurs when the tissue is stimulated by an external electric current source. Bioimpedance (BI) can be simplistically described as the biological tissue capacity to oppose electrical current. The studies of Bioimpedance are based on the close relationship between the electrical properties of the human body, the body composition of the different tissues and the water content of the body [11].

2.3.1. Bioimpedance Basics

Impedance (Z) is defined as the electrical resistance that is generated in an electrical circuit when an alternating current tries to pass through it. Unlike resistance in direct current, impedance is expressed through complex numbers, that is, with a real part and an imaginary part. The real part corresponds to the Resistive component (R), whereas the imaginary part is made up of the reactance (X_C) associated with capacitors and the inductance (X_L) associated with coils. A phase angle relates to these parameters as can be seen in Figure 2.3.1.1.

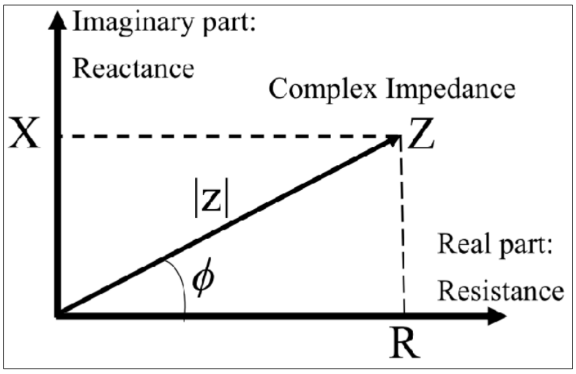


Figure 2.3.1.1.- *Components of electrical Impedance (Z) [12].*

Bioimpedance is the term to describe the safe, non-invasive measurement of the passive electrical characteristics of an organism after the introduction of a painless low-level alternating current to the body. This technique is commonly used for body composition measurement, muscle injury and clinical condition assessment [13].

The physical basis of the Bioimpedance (BI) method is the awareness that the human body is a network of resistors and capacitors. The physiological fluids perform as resistors and cell membranes behave like capacitors. Therefore, the body can be represented as a parallel resistor-capacitor (*RC*) circuit (Figure 2.3.1.2) in which two pathways can be differentiated (resistive and capacitive) when an alternating current is introduced [13].

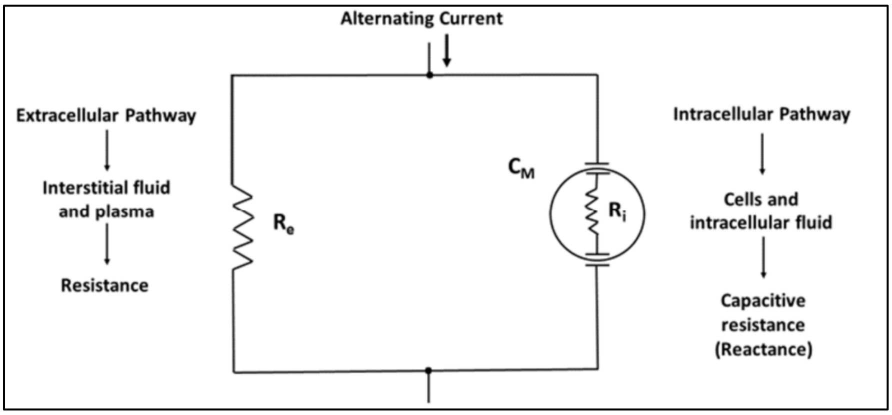


Figure 2.3.1.2.- *The body exposed as a network of resistors and capacitors in a RC circuit model.* C_M *is the membrane capacitance and* R_e *and* R_i *are the extracellular and intracellular resistance [13].*

At higher frequencies, current can pass through the cell membranes and the impedance values reflect the molecules both inside and outside the cells, while at lower frequencies, the current cannot penetrate to relatively non-conductive phospholipid bilayers and impedance values reflect molecules and structures outside the cell membrane.

The electrical properties of these heterogeneous biological tissues are not constant over the whole frequency spectrum. Some transition regions, known as dispersion regions, can be observed. Three frequency regions for the dielectric properties of the biological tissues in response to applied electric fields have been defined [11]:

1. **α dispersion** is seen at low frequencies (10 Hz - 10 kHz) and is caused by the ionic medium surrounding the cells.
2. **β dispersion** occurs at mid-range frequencies (10 kHz - 10 MHz) and is due to the capacitive charge of cell membranes.
3. **γ dispersion** occurs at higher frequencies (10 MHz - 10 GHz) and is due to the dielectric relaxation of water molecules.

2.3.2. Frequency selection

For BI measurements, different methods related to frequency can be used. There exist methods that use more than one frequency such as Multifrequency and Bioelectric Spectroscopy. However, for muscles assessment, a single frequency technique is commonly used. If the measurements are made at a single frequency, the most common is 50 kHz. The standard frequency at 50 kHz has optimal properties since it generates an impedance vector with a maximum phase angle in the frequency spectrum from 1 Hz to 1 MHz. With the obtained phase angle, the maximum value of reactance is reached. As Figure 2.3.2.1 illustrates, the frequency where the reactance reaches de maximum is called the characteristic frequency (f_c), even though 50 kHz is assumed to be the f_c, it can vary over a range of values for healthy individuals [14,15].

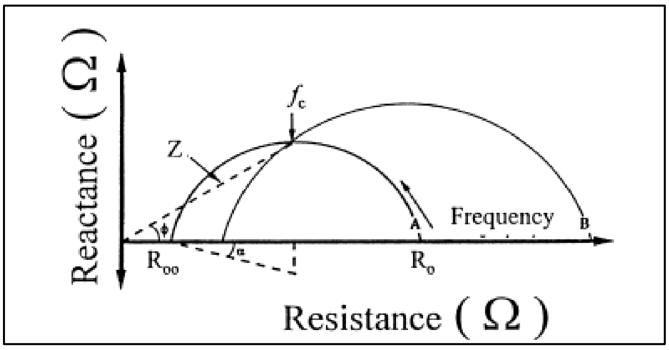


Figure 2.3.2.1.- *Plot of reactance versus resistance as a function of frequency Z impedance; f_c, characteristic frequency; R_0, resistance at zero frequency; R_∞, Resistance at infinite frequency [14].*

2.3.3. In Vivo Bioimpedance measurements

BI measurement is a method that is commonly used to measure the composition and hydration of the body and can therefore be used as part of routine health assessments, as well as to monitor the patient's nutritional status and health, hydration status in chronic or progressive disease. At present, there are two main techniques to obtain BI measurements: invasive and non-invasive methods.

The first procedure involves inserting electrodes (needles) into the tissues, applying a current, and measuring the impedance between pairs of electrodes. Although, non-invasive approaches involve connecting a series of surface electrodes to the skin, applying a current and measuring the impedance between pairs of successive electrodes [15].

2.3.3.1. Estimation of body composition

One of the most common applications for BI is to estimate body composition and body mass index. These measures have been used in a variety of medical applications, including assessing the body composition of healthy patients and monitoring nutritional status in various disease states and monitoring the athletes.

The human body, as a volume, is generally composed of the mass corresponding to fat (FM), which is considered a non-conductor of electric charge and is defined as the difference between body weight (Wt_{body}) and fat-free mass (FFM). This second type of mass is the one that acts as a conductive volume, which allows the passage of electric current due to the conductivity of electrolytes dissolved in body water. Studies show that water, identified as total body water (TBW), is the main compound of FFM and is equal to 73,2 % in subjects with normal hydration, as set out in the following equations [11].

$$\boldsymbol{FM = W_{t\,body} - FFM} \qquad \textbf{(Eq. 1)}$$

$$\boldsymbol{TBW = 0.73\,FFM} \qquad \textbf{(Eq. 2)}$$

When performing a bioimpedance measurement, the human body is divided into five segments (Figure 2.3.3.1.1): one segment for each upper extremity, two for the lower extremities, and one for the trunk. In this division into 5 compartments, the human body is composed of FM and FFM, consisting of bone minerals and body cell mass (BCM) and including protein and total body water (intracellular and extracellular fluid).

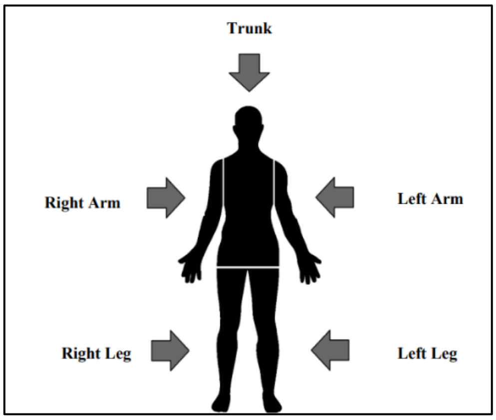


Figure 2.3.3.1.1.- *Main body segments and compartments [11].*

Most known prediction methods are based on the ratio of water volume to the ratio of squared length to resistance ($\frac{L^2}{R}$) [11].

2.3.3.2. Bioimpedance configurations

There are different configurations of bioimpedance measurements and therefore diverse options in how to locate the electrodes.

Whole body BIA:

In this configuration, the total impedance of a subject with normal hydration is determined by 50 % of the impedance of the lower limbs, 40 % of the impedance of the upper limbs, and by 10 % of the impedance of the trunk [4]. A pair of electrodes (an injector and a sensor) is placed dorsally on the hand (third metacarpophalangeal and carpal joint, respectively) and on the foot (third metatarsophalangeal joint, and tibio-tarsal) as illustrates Figure 2.3.3.2.1. The standard reference is the right side of the body since it is important to be consistent and use the same side on the test subject for repeatability [16].

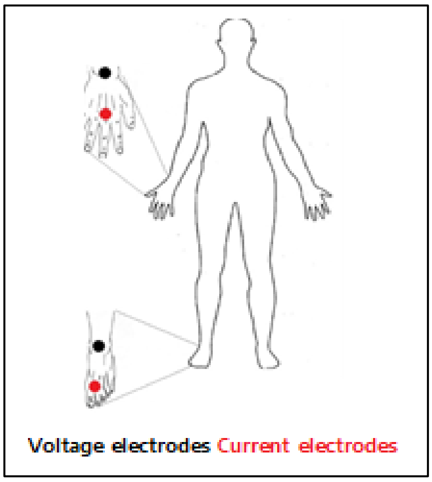


Figure 2.3.3.2.1.- *Whole body Bioimpedance measurement electrode position [11].*

Proximal BIA: To improve the estimation of the compartments of the conventional BIA, in particular of the fluids and the lean mass, different modalities of the positioning of the electrodes have been proposed (with the same considerations on cylindrical and isotropic conductors). Positioning the sensing electrodes on the antecubital fossa and in the popliteal fossa, a proximal BIA is achieved. The superiority of proximal BIA has not been confirmed over whole-body BIA in estimating compartments in healthy adults neither in single frequency nor multifrequency systems [16].

Segmental BIA: The whole-body Impedance measurements are sensitive to changes in regional fluid distribution and tend to underestimate fluid changes during ultrafiltration in hemodialysis patients. The aim of the segmental bioimpedance analysis (SBIA), is to obtain data not affected by changes in body position. A segmental BIA is achieved by placing electrodes at the extremities of the upper, lower limb, and trunk, according to various modalities (Figure 2.3.3.2.2). The technique, however, is not yet standardized and presents operational difficulties when identifying the reference points at the root of the limbs and on the trunk, especially in obese patients. The diffusion of the current in the tissues is not limited by the border of the body segments, which is the reason for the failure of the technique to discriminate different degrees of expansion of the fluids, both in the conventional analysis and using the direct measurements of R and X_C with vector analysis [16].

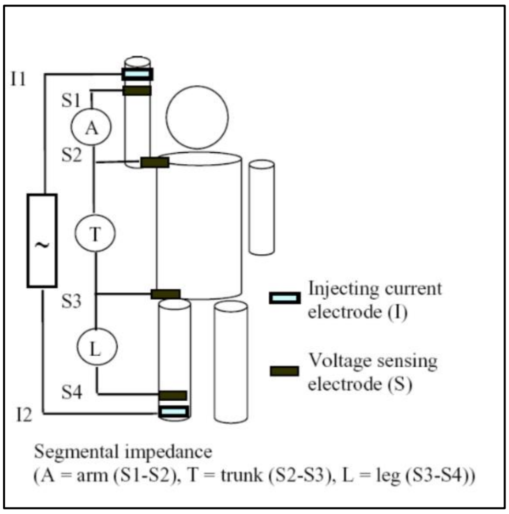


Figure 2.3.3.2.2.- *Segmental Bioimpedance Measurement electrode position [11]*

2.3.4. Localised Bioimpedance (LBIA) to assess muscle injury

The BI analysis can be used to assess and evaluate muscle injuries in the world of sports but also a clinical environment [9]. When an injury is produced to the muscle, causes marked reductions in *R*, X_C, and *Phase angle (PA)* and these changes are related to the severity of the injury. The LBIA method uses four surface adhesive contact electrodes (typically Ag/AgCl), placed in the injured muscle. The sensing pair of electrodes measuring the voltage drop (V) are placed 5 cm proximally and 5 cm distally from the centre of the injury and the source electrodes (that inject altern current at a frequency of 50 kHz), are placed next to the others. In the case the injury is anatomically deep (near the bone), the electrodes are placed 10 cm proximally and 10 cm distally to increase the depth of the current's penetration. The position of the electrodes is exposed in Figure 2.3.4.1 [17].

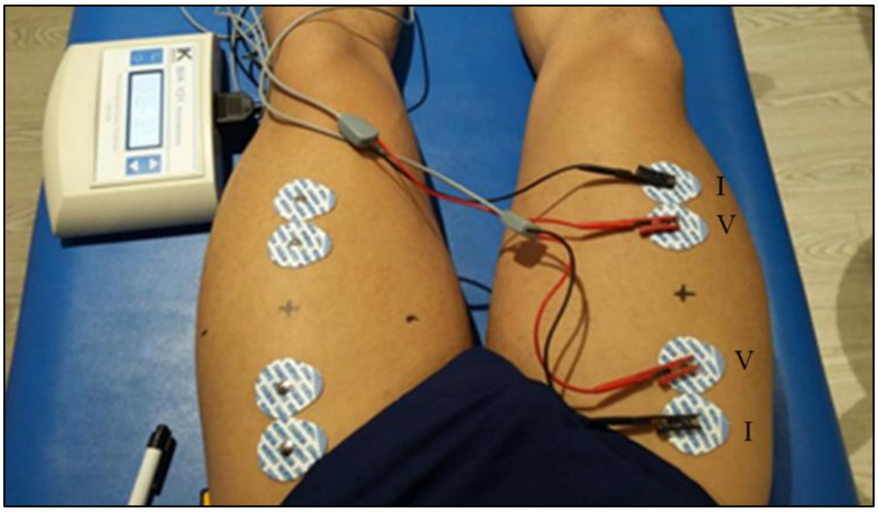


Figure 2.3.4.1.- *Tetra-polar localized BI electrode placement on the injury located in 1/3 medium quadriceps [17].*

Nescolarde et al. [17] made a serial tetra-polar, phase-sensitive 50 kHz localized BIA measurements of quadriceps, hamstring and calf muscles of three male football players before and after injury and during recovery until return-to-play, to determine changes in BIA variables (resistance (R), reactance (Xc) and phase angle (PA)) in different degrees of muscle injury.

As compared to non-injury status, the most severe muscle injury (grade III) was associated with a 23.1% decrease in R and a 27.6% decrease in PA [17]. Serial measurements during recovery showed a gradual return to near pre-injury Xc (20º to 17º) but only a partial restoration of PA (16.6º to 14.7º) values. The greatest impact of a grade III injury was the 45.1% reduction in Xc [17]. This effect, attributable to the injury-induced muscle cell damage, also elicited the 27.6% decrease in PA [17]. The near return to non-injury Xc and PA values suggests that slow incomplete healing of the residual intramuscular cavity and muscle cells associated with the muscle injury continued while function returned [17].

As compared to a grade III injury, relative decreases in BIA values were attenuated in grade II and I muscle injuries. These observations highlighted the sensitivity of this method to identify fluid accumulation and muscle cell disruption. The observed relative decrease in R was larger in the grade II compared to the grade I muscle injury (20.6% versus 11.9%) [17]. Localized fluid accumulation, an MRI-identified hematoma, explains the greater decrease in R in this grade II injury. The relative change in Xc was greater with more severe muscle injury (31.6 compared to 23.5% in grades II and I, respectively) [17].

These findings indicate that decreases in R reflect localized fluid accumulation, and reductions in Xc and PA highlight disruption of cellular membrane integrity and injury. Overall, Localized BIA measurements of muscle groups enable the practical detection of soft tissue injury and its severity [17].

2.3.5. Lock-in amplifiers

The obtention of the bioimpedance in an LBIA measurement relies on the use of the lock-in amplifier.

A lock-in amplifier is a topology of amplifier that is used to detect a signal of a known carrier frequency buried in noise. This can only be accomplished, however, if the signal of interest appears as amplitude modulation on a reference frequency. The ideal lock-in amplifier will then detect only the part of the input signal having the same frequency and phase as the reference signal. Depending on the dynamic range of the instrument, it is capable to detect signals up to one million times smaller than the present noise.

A Lock-in amplifier is based on the multiplication of two sine waves, one being the signal carrying the amplitude modulated information of interest, and the other being a reference signal with the chosen frequency and phase [18]. The result of this operation in the frequency domain are two components which are the sum and the difference between the two frequencies of both signals. An example is presented in Figure 2.3.5.1.

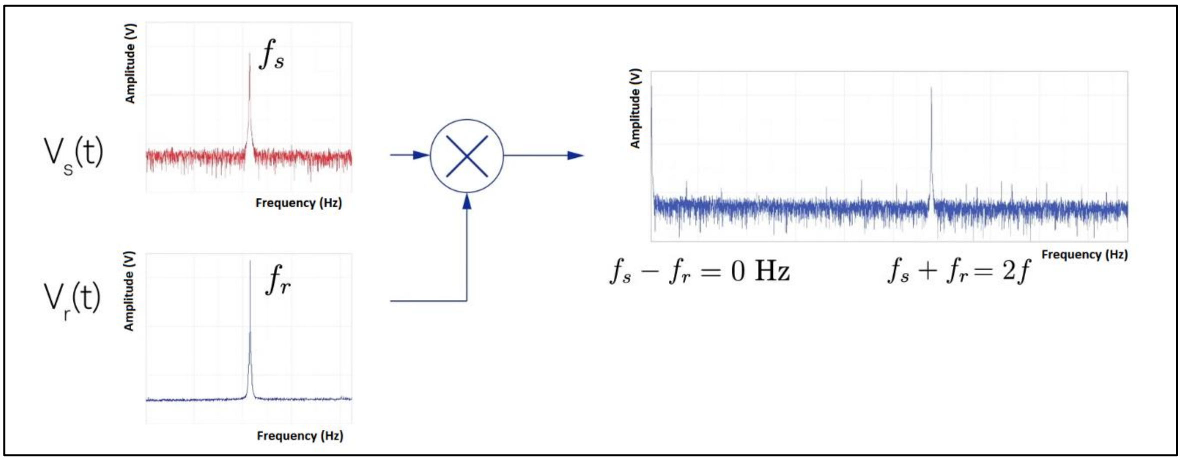


Figure 2.3.5.1.- *Frequency domain representation of the mixing of signals results [19].*

As it has been exposed before, the frequency of the reference signal is chosen to be the same as the input signal. The result in the frequency domain is illustrated in Figure 2.3.5.2, where the interest signal is the difference between the two frequencies. The sum of both frequencies must be removed using an adjustable low pass filter; therefore, the result will be a DC signal.

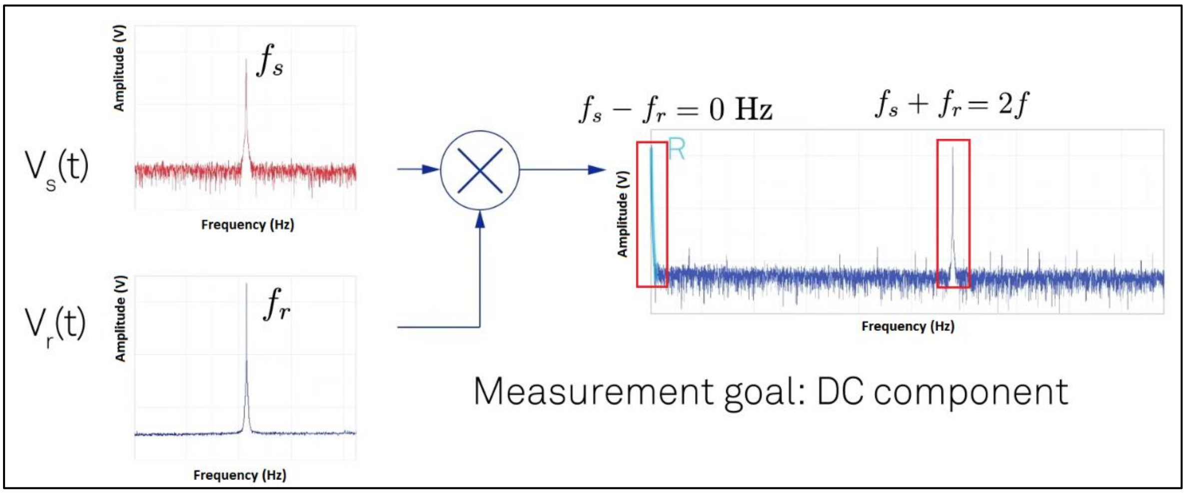


Figure 2.3.5.2.- *Frequency domain representation of the mixing of signals with the same frequency result [19].*

An adequate selection of the low pass filter is crucial. If the bandwidth of the filter is too wide, it will lead to possible measurement errors as the 2f component might be affecting the output signal. Moreover, due to the larger bandwidth, there can be the presence of more noise, resulting in a lower signal-to-noise ratio. On the other hand, choosing a filter bandwidth that is too narrow, will limit the time resolution, and will slow down the measurements.

The attenuation of the filter can also be adjusted by choosing its order. A higher order means a more ideal function transfer that blocks frequencies outside the filter bandwidth more efficiently but causes a phase delay since it takes more time to settle. A lower order filter has the advantage of causing less phase delay which helps when high-speed requirements need to be met.

After the low-pass filtering process is completed, the DC resulting signal follows the Equation 3 [18]:

$$V_o = \frac{V_i}{2} \cdot cos\varphi \qquad \textbf{(Eq. 3)}$$

This type of amplifier is a good choice when developing immittance measurements such as bioimpedance analysis. In this case, a pair of amplifiers could be useful, one with a reference signal identical or in phase with the excitation signal (alternating current) and with a reference signal 90 degrees out of phase. Using an alternating current as the source will produce a measured voltage that will be separated into two components, the in-phase component, and the quadrature component (corresponding to resistance and reactance respectively). There exist both, analogic and digital versions of these amplifiers. However, digital ones are more precise and flexible even though they are more power-consuming, and frequency limited.

2.3.5.1. Digital Lock-In Amplifiers

This typology of lock-in amplifier computes the digital multiplication of the input and reference signals after they are digitalized. To obtain the output signal from the multiplication product, a digital averaging (integration) process is used. This process is exposed in Figure 2.3.5.1.1.

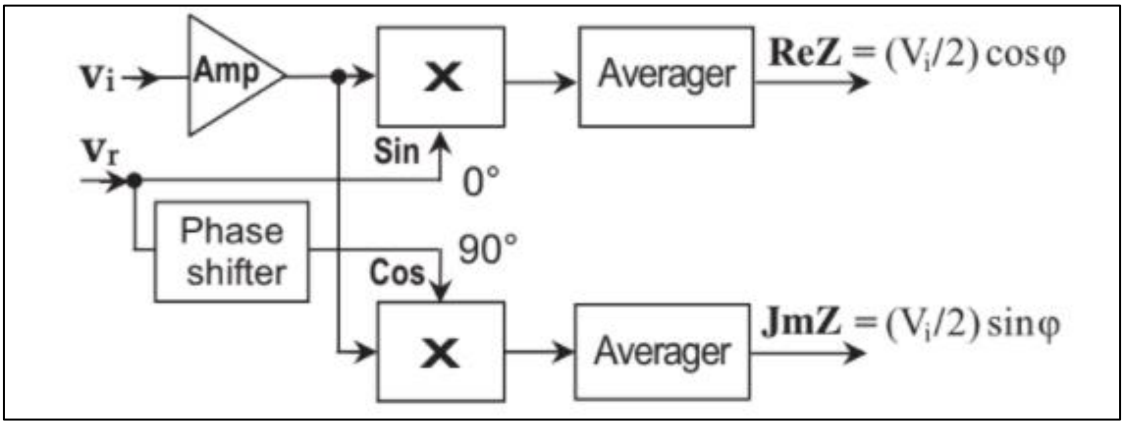


Figure 2.3.5.1.1.- *Two-phase (quadrature) lock-in amplifier [18].*

When the two components are obtained, the lock-in amplifier obtains the amplitude of the resistance and reactance which can be used to obtain the phase angle (θ).

2.4. EMG Physiology

The human motor system must cope with a great diversity of demands including movements, upright posture, and locomotion among others. To meet all the demands, the muscles of the body must contract and relax consciously and/or unconsciously by the subject. Thus, muscle contraction and relaxation are controlled by the central nervous system, which sends a signal through a motor neuron to a grouping of muscle cells, called fibers. That grouping of muscle cells, and the motor neuron that innervates them, is a Motor Unit (MU), the smallest functional part of muscle tissue [20].

The MU consists of an α-motoneuron (the final point of summation of all descending and reflex inputs) in the spinal cord and the muscle fibers it innervates, as illustrated in Figure 2.4.1. The contraction of the muscle is possible to the direct link between the cortex and the skeletal muscle. The number of MUs present in a muscle differs significantly across different muscles, as well as the force-generating capacity of these MUs. The number of MUs per muscle in humans may range from about 100 to 1000 or more for large limb muscles [21].

Moreover, Motor Units have therefore been categorized into three different categories based on chemical and contractile characteristics [21]: Fast-twitch, fatigue-resistant muscles, and slow-twitch fatigue-resistant muscles.

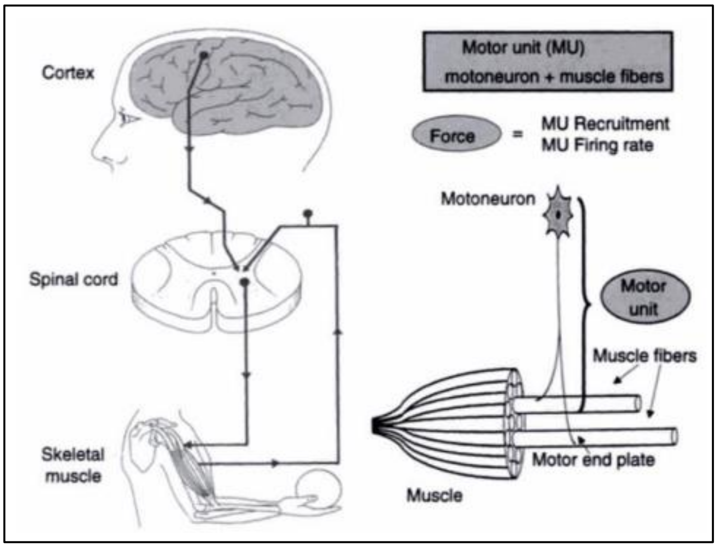


Figure 2.4.1.- *The underlying motor control mechanisms, motor units and their components [21].*

When the human body needs to generate movement, the cortex sends a biosignal through the neurons to the spinal cord, where an α-motoneuron activates muscle fibers. The number of fibers activated, or MU recruited, depends on the desired muscle strength and the firing rate of activation of each MU. Thereby, the greater the number of MUs recruited and their discharge frequency, the greater the force will be [21].

2.4.1. EMG Basics

To extract the relevant information from the MUs, it is essential to understand what an EMG is and what kind of applications exist.

First, an Electromyography (EMG) is the measurement of Action (bio)Potential signals transmitted by Motor Units (MUAP) that cause the contraction of muscles. In other words, the EMG is a representation of the electric potential field generated by the depolarization of the outer muscle fiber membrane. Its detection involves the use of intramuscular or surface electrodes.

There are two main techniques when measuring an EMG; using surface electrodes that overlay the muscle interest zone (sEMG) or intramuscular electrodes, which use needles or fine wires to detect specific and individual MUAPs.

As has been mentioned, sEMG employs surface electrodes to measure or monitor an interest muscle region (e.g., quadriceps muscles or biceps). The advantages of this approach are numerous: sEMG allows to study various aspects of behavior, for instance; Temporal patterns of activity, muscle fatigue, location of the innervation zone and length including orientation fibers. Additionally, it is highly accurate when evaluating the muscle-fiber conduction velocity.

Otherwise, unlike needle EMG, surface EMG cannot measure individual MUAPs because surface electrodes tend to record from much larger regions and MUAPs are not clearly visible as many MUs tend to be contracting at the same time. In addition, surface electrodes have limitations with recording dynamic muscle activity due to the concerns with crosstalk of adjacent muscles and the separation of the biological tissue between the source and the recording electrode that acts as a low-pass filter [21]. All noises and artifacts should be taken into account when processing the acquired EMG.

On the other hand, needle EMG (nEMG) is an invasive procedure that utilizes indwelling electrodes to detect directly inside the desired muscle. Thus, nEMG gives access to deep musculature, there is not as much noise as the sEMG technique and shows higher sensitivity and higher frequency and voltage ranges [22]. Besides, needle electrodes are suitable for detecting changes in MU size and internal structure, as well as revealing their abnormal function.

However, nEMG has several disadvantages owing to its invasive approach; the insertion of the needle into the muscle requires sterilization of the instrumentation and the environment, and needles are difficult to use on muscles located in sensitive areas, such as muscles in the tongue, lips, and face and there is minor muscle tissue damage. Although low, there is a risk of infection and must be supervised by a professional, meanwhile sEMG does not require medical supervision [23].

Overall, the two methods are better suited for a different span of applications and have their advantages and disadvantages and are therefore both currently used for EMG signal detection. The instrumentation required for this project is dedicated entirely to the noninvasive (sEMG) technique due to its easy and non-invasive application, low cost, and painless method.

2.4.2. Frequency content of sEMG

It is essential to understand the frequency content of MUAPs and the sources of noise that corrupt the signal to process EMGs correctly. Most sEMG signals have frequency content ranging from 0 to 500 Hz, with dominant energy between 50 to 150 Hz. However, content at up to 200 Hz may be useful [23] and the amplitude of the signal may vary from less than microvolts up to some millivolts. In literature, the sEMG spectrum includes signals with voltage peaks going from 50 μV_{PP} up to 2 mV_{PP}, 5 mV_{PP}, 6 mV_{PP} and even 10 mV_{PP} [24, 25]. Figure 2.4.2.1 illustrates the typical power spectrum of an EMG:

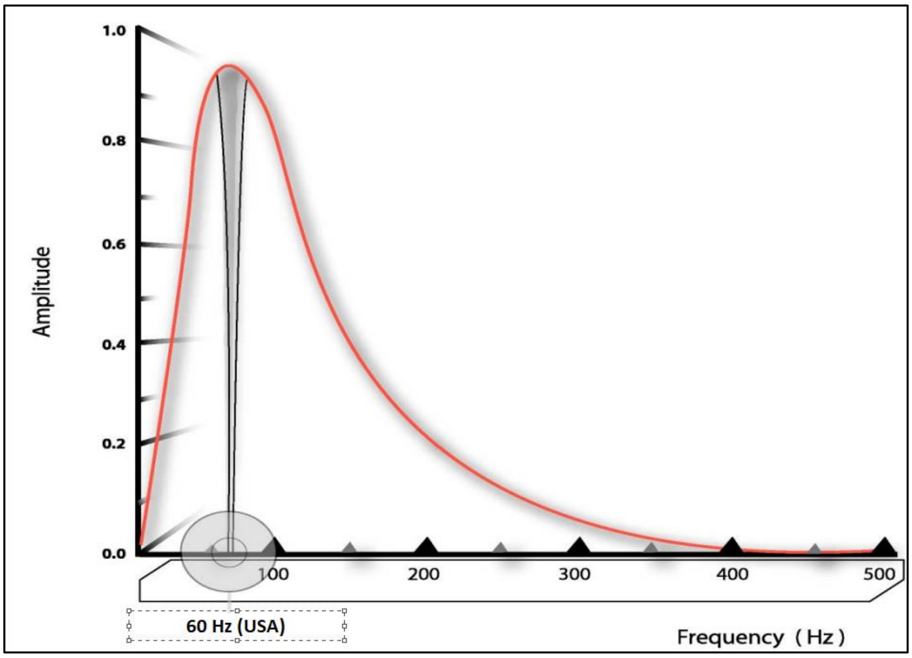


Figure 2.4.2.1.- *Schematic representation of a typical sEMG power spectrum. The shaded area indicates the signal lost when notch filtering is used, in this case to eliminate 60 Hz power source noise (USA). Adapted from [25] .*

There are numerous physiological and non–physiological factors that influence sEMG signal interpretation, such as electrode shape, size, inter-electrode distance, skin contact and location over the muscle, motor unit physiology, subject's muscles' properties and so on.

There are several types of electrical noise that affect a sEMG signal and understanding the frequencies of these sources allows for the removal and/or the reduction of them, obtaining a better-quality signal:

- Ambient noise*:* Ambient noise lies in a wide range of frequencies, but its dominant component is 50 Hz (Europe) which is the most common source of electrical noise in the EMG signal and corresponds to power line noise. Such noise results in a signal whose voltage can be larger than the EMG signal itself. Moreover, another electromagnetic interference (EMI) such as fluorescence or radio magnetic waves affects the human body, coupling through the skin in order to attenuate correctly.

  Some experts do not recommend the use of a 50 Hz notch filter as it partially removes frequency components adjacent to the unwanted ones. However, when skin preparation and quality equipment are not sufficient to attenuate ambient noise and if only a rough EMG signal amplitude is sought, a sharp notch filter may be acceptable. An important issue to keep in mind is that the power frequency harmonics can contain more power than the fundamental frequency, therefore repeated notch filtering (at the harmonic frequencies) may be necessary [26].
- Motion artifact: This type of noise is caused by the movements of electrode cables or at the interface between the muscle zone (skin) and the surface of the recording electrode.

Therefore, the power density of Motion artifacts is mostly below 20 Hz [24, 27], so it can be easily removed from the EMG signal after the integration of a high pass filter into the measurement instrumentation. On the other hand, to avoid loss of MUAPs signal power, the corner frequency of the filter must not be set higher than 20 Hz [26]. As has been already stated, skin preparation can reduce the electrode–skin impedance and helps to minimize motion artifacts.

- *Muscle crosstalk:* Crosstalk is the signal detected over a muscle but generated by another muscle close to the first one. This phenomenon is present exclusively in sEMG when the distance of the detection points from the source may be relevant and similar for the different sources [21]. Crosstalk is one of the most important sources of error when interpreting surface EMG signals because it can be confounded with the signal generated by the muscle, which thus may be considered active when indeed it is not.

  Crosstalk can be minimized by discriminating signals originating directly underneath the electrodes from those originating further away. This is possible since the characteristics of an EMG signal change as it travels through the body tissue, which acts as a low-pass filter. Since crosstalk signals must travel longer through the tissue to reach the recording zone than the signal of interest, the former will contain fewer high-frequency potentials. Due to that, crosstalk potentials will be more equally distributed over the recording surface than potentials originating nearby. Consequently, all the individual electrodes in the muscle area will record quite similar crosstalk potentials, but quite different near potentials. This makes it possible to improve the spatial resolution of the recording by common-mode rejection, which will attenuate the common crosstalk signal, but preserve the signal originating nearby [28].

  In addition, other options for reducing crosstalk include careful selection of electrode sizes and electrode spacings, as well as electrode placement; Smaller inter-electrode distances tend to lead to less crosstalk which is typically about 1 to 2 cm, but at the same time, overly small distances can result to electrode shorting (e.g., due to sweat). Besides, electrode alignment with the direction of muscle fibers increases the probability of detecting the same signal, thus will help to attenuate the common crosstalk [23].

- *DC offset potential:* Oil secretions, dead skin cells and skin impurity increase impedance on the outermost layer of the skin, which causes DC voltage potential up to 200 - 300 mV. This DC potential is common to all electrodes and can be minimized with a proper skin preparation. The quality of contact is typically reduced by at least a factor of 10 with proper preparation [21]. Usually, skin cleaning involves the use of special abrasive and conductive pastes to remove dead cells or fine sandpaper and alcohol swabs to clean the outer layer.

- Other noise sources*:* The noise and interferences covered above are the main sources of contamination in sEMG. However, there are other kinds of interferences that should be minimized as well. Any electronic equipment will generate noise up to thousands of Hertz and the electronic instrumentation used to amplify and filter the sEMG signal is not an exception. Although this type of noise cannot be eliminated, well-designed instrumentation tends to have noise less than 1,5 $mV_{(RMS)}$ (referred to as the input) over the band from 20–500 Hz [26]. Additionally, most sEMG recordings are processed via an analog-to-digital converter (A/D), which involves two issues: Sampling rate and sampling resolution.

The sampling rate theorem (Nyquist) states that the sampling frequency during a data acquisition should be at least twice the highest frequency contained in the signal in order to recover the complete information content [29]. If there are higher frequency components than one-half the sampling rate, ambiguities will arise, and information will be lost. This phenomenon is called aliasing and it is essential to know the bandwidth of the sampled signal so that the minimum sampling rate can be fixed, and antialiasing will be prevented. According to *SENIAM* (Surface Electromyography for the Non-Invasive Assessment of Muscles), the sampling frequency should be higher than 1000 samples per second in general applications [28].

As has been mentioned before, most of the signal power in sEMG is located below 500 Hz. Therefore, it is recommended to filtrate the analog signal using a low-pass filter with a sharp roll-off and a cut-off frequency at or below one-half the sampling rate.

On the other hand, A/D conversion resolution indicates the number of different and discrete values it can produce over the allowed range of analog input. For instance, a typically 16-bit ADC divides the input voltage range into 65536 discrete levels, meanwhile an 8-bit ADC converts into only 256 discrete levels. Since the analog scale is continuous, while the digital codes are discrete, there is a quantization process that introduces an error called *quantization error*, resulting in *quantization noise*. As the number of discrete codes increases, the quantization error gets smaller, and the ADC transfer function approaches an ideal straight line [30]. This noise is a broadband with a maximum magnitude of one half of a bit. *Kamen G et al.* [31] suggested using a 16-bit ADC in sEMG measurements because its resolution has an excellent range to match the maximum peak-to-peak amplitude of the recorded signal and it should not require the use of variable gain selection [31].

Summarizing, the EMG signal can be divided as illustrates Figure 2.4.2.2, from where is only interesting the signal labeled as *EMG signal*.

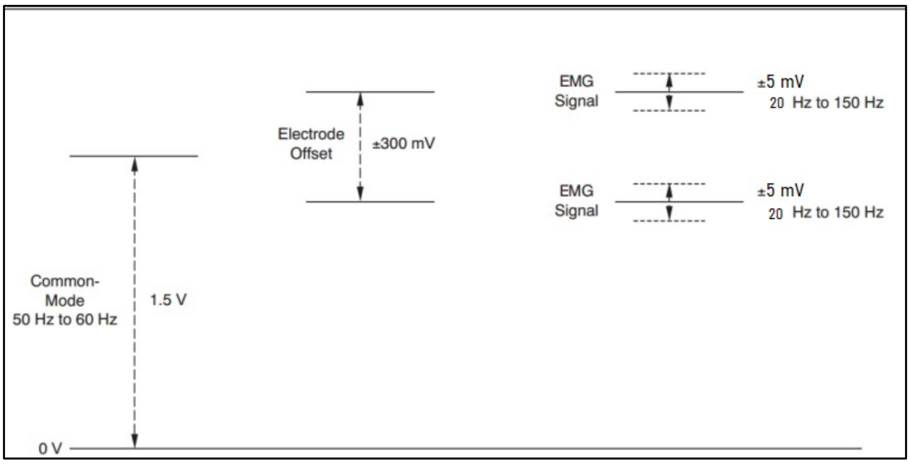


Figure 2.4.2.2.- *Raw EMG signal composition. Adapted from [32].*

Overall, Table 2.4.2.1 resumes all the requisites mentioned above and some SENIAM recommendations:

Table 2.4.2.1.- *Summary table of requisites and SENIAM recommendations [21].*

PARAMETER		**RECOMMENDED VALUE OR CONDITION**
ELECTRODES (BIPOLAR MONTAGE)	**Electrode size**	Diameter < 2 cm
	Electrode distance	< 2 cm or ¼ the muscle length (whichever is smaller)
	Reference electrode location	Wrist, ankle, or another inactive area
AMPLIFIER	**High-pass filter**	≈ 20 Hz (motion artifacts suppression)
	Low-pass filter	≈ 500 Hz (antialiasing suppression)
	Input Impedance	> 100 MΩ (for conventional electrodes)
	Gain	Suitable to bring the signal into the input range of the ADC
SAMPLER AND A/D CONVERTER	**Sampling frequency**	> 1000 samples/s (for general applications)
	N bits of A/D	12 bits (requires an amplifier with variable gain) 16 bits (fixed gain amplifiers may be used)

2.5. Electrode Basics:

To measure a biopotential and, hence, currents in the body, electrodes are designed to provide some interface between the body and the electronic measuring equipment. So, recording electrodes must therefore have the capability to conduct a current across the described interface.

It can be thought that the function of a bipotential electrode is relatively easy to achieve, but if the problem is considered in more detail, the electrode carries out a transducing function (transform one type of energy into another) [9], since electrical charges move across the cell membrane, not as electrons, but as charged ions across the plasma membrane. Therefore, electrodes must transduce the ionic current into an electronic current. Figure 2.5.1. shows the transduction function between electrode and electrolyte.

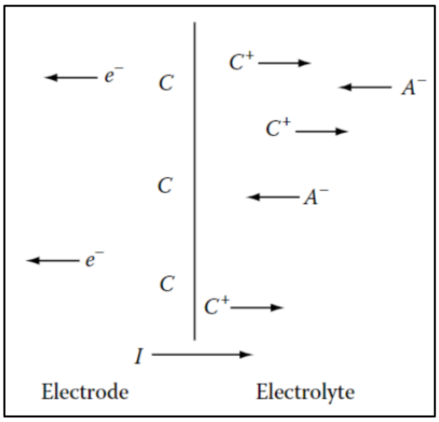


Figure 2.5.1.- *Electrode-Electrolyte interface with current left-to-right. The electrode is a metal with "metallic" atoms C and the electrolyte is a solution containing cations of the electrode metal C^+ and anions A^- [33].*

When the metal encounters the solution, the reaction begins, going either to the left or to the right depending on the concentration of ions and the equilibrium initially established. The local concentration of cations in the solution at the interface changes, resulting in a non-zero net transfer ratio. So, the electrode surrounding the metal is at a different electrical potential from the rest of the solution. This phenomenon is referred to as half-cell potential and it is important for understanding the behavior of biopotential electrodes [33, 34].

2.5.1. Polarizable and nonpolarizable electrodes

Theoretically, two types of electrodes are possible; those that are perfectly polarizable and those that are perfectly nonpolarizable; The first kind are those in which no actual charge crosses the electrode-electrolyte interface when a current is applied and behave as a capacitor. Alternatively, non-polarizable electrodes are those in which current passes freely across the electrode-electrolyte and there are no overpotentials [9]. The best example is the silver-silver chloride electrode (Ag/AgCl), which can be easily fabricated in the laboratory and is widely used in sEMG due to its approach to the nonpolarizable model.

2.5.2. Electrode's Electric Model

From the characteristics of the electrodes, it is possible to design an electrical model of the electrode such as shown in Figure 2.5.2.1.

To begin with, a voltage source represents the half-cell potential, which is usually omitted for nonpolarizable electrodes such as the silver-silver chloride electrode. Then, a parallel RC circuit is implemented where R_d and C_d represent the leakage resistance and the capacitance across the double layer of charge respectively. Finally, an R_s resistor is introduced to model interface effects and electrolyte resistance.

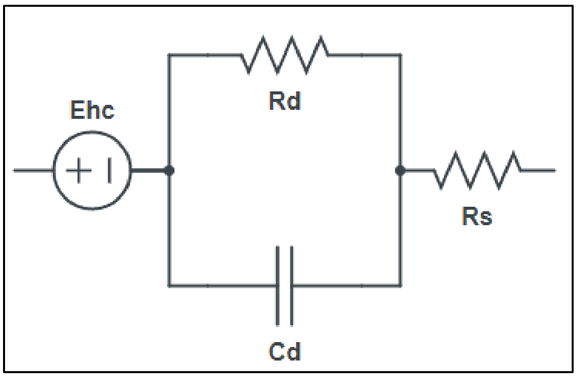


Figure 2.5.2.1.- *Equivalent circuit for a biopotential electrode [33].*

In summary, from the studied electrical circuit, the total impedance of the electrode-electrolyte interface will follow the Equation 4:

$$Z_{Total} = R_S + \frac{R_d}{1 + j\omega R_d C_d} \qquad \textbf{(Eq. 4)}$$

2.5.3. Electrode-Electrolyte Skin Interface

In order to measure biopotentials, some interface between the body and the electronic measuring circuit has to be applied. For the passage of current from the body to an electrode, and consequently, into an electronic circuit, it is required a charge transfer called *the electrode-electrolyte interface*. The electrolyte represents either the body fluid containing ions (sweat) or an electrolyte solution (gel material) applied between the electrode and the skin. The electrode-electrolyte interface occurs when a metal is placed or linked with an electrolyte solution, and an ion-electron exchange happens across the interface, passing from the electrode to the electrolyte.

So, when biopotentials are recorded from the surface of the skin, an additional interface must be considered; the interface between the electrode-electrolyte and the skin. Therefore, to understand how this interface behaves, it is vital to review the structure of the human skin.

The skin consists of three principal layers that surround the body to protect it from its environment [9]. The outermost layer, or epidermis, plays the most important role in the electrode skin interface. Moreover, this layer is constantly renewing itself, resulting in a large amount of dead material that has different electrical characteristics from living tissue. In addition, the deeper layers of the skin contain the vascular and nervous components as well as the sweat glands, sweat ducts and hair follicles. These layers are similar to other tissues in the body and do not bestow any unique electrical characteristics [33].

As a result of the dead material in the outer layer of the skin, the impedance at the surface is very high. Consequently, it is highly recommended to apply a conductive gel to reduce that impedance and also enhance the contact and conductivity between the skin and the surface of the electrode. This gel usually consists of a cream that contains chloride (Cl^-) ions.

The following figure (Figure 2.5.3.1) illustrates the electrode-electrolyte interface equivalent circuit.

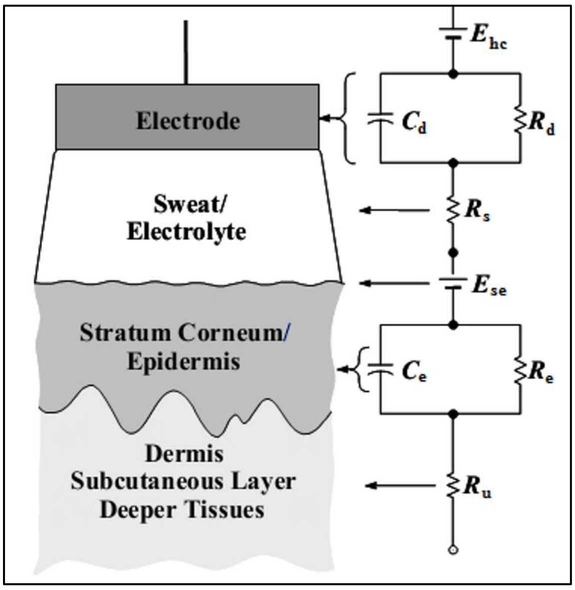


Figure 2.5.3.1.- *Total electrical equivalent circuit of electrode-electrolyte skin interface [34].*

The series resistance R_s is now defined as the effective resistance associated with the interface effects of the gel between the electrode and the skin. Moreover, the epidermis can be considered semipermeable membrane, so if there is a difference in ionic concentrations across it, there is a potential difference E_{se} (which can be obtained by Nernst equation). Additionally, the epidermal layer is also found to have an electric impedance that behaves as a parallel RC circuit, as shown in Figure 2.5.3.1. Finally, the lower layers (dermis and subcutaneous) behave in general as a pure resistor R_u [33, 34].

2.5.4. Accurate selection of electrodes

A key component of bioimpedance measurements is the use of contact surface electrodes. The most common type of electrodes used in this field is the Ag/AgCl surface electrodes which have been established for the non-invasive assessment of bioelectrical signals [35]. It is commonly believed that all electrodes of this typology should have the same intrinsic impedance half-cell potential. However, *Webster et al.* [9] stated that there are some differences among different commercial Ag/AgCl electrodes that can introduce errors in the bioelectrical measurements. *Shiwei et al*. [36] found a clear disparity with the measured impedance values among the same studied volunteers with different commercial electrodes for biomedical applications.

Consequently, *L. Nescolarde et al.* performed single frequency (50 kHz) bioimpedance measurements on 35 healthy volunteers between the ages of 35-50 (15 males and 25 females) using 9 types of commercial Ag/AgCl electrodes [35]. The obtained results had impedance values from *15.77 Ω* to *665.23 Ω.* The diversity in these values indicates that due to this fact, errors can be introduced in the measurement. For this reason, it is extremely important to use the electrodes stipulated by the fabricants of each measurement device. Furthermore, if it is not indicated, the electrodes chosen are needed to have a low impedance value.

3. Objectives

3.1. Main objectives

The main objective of this thesis is to develop the firmware design of a portable medical device to study the muscle groups affected in a total knee arthroplasty procedure by LBIA and EMG measurements.

3.2. Specific objectives

3.2.1. Biomedical specific objectives

- Study the state of the art of LBIA and EMG medical devices.
- Study and implementation of digital signal processing techniques of LBIA (using lock-in amplifiers) and EMG signals.
- Development of a preliminary LBIA statistical analysis.

3.2.2. Electronic specific objectives

- Programming all principal firmware processes (peripheral initialization, memory initialization, check of battery level, data acquisition, etc.).
- Implementation of different communication protocols of the medical device.
- Establish the communication protocol between the medical device and an external device.
- Develop an error handling functionality to assess the medical device performance.

4. State of the art of LBIA and EMG Medical Devices

In the latest decade, sEMG and LBIA have gained importance in monitoring and assessing muscle conditions. Due to this fact, new technology has been developed, resulting in new measuring techniques and commercial portable medical devices, which will be exposed and discussed.

4.1. EMG portable measuring devices

4.1.1. Mbody 3 Kit

Mbody 3 is a commercial product from *Myontec* designed as a complete wireless tool for analyzing muscle activity during training. It can be used for athletes, training centres and clinical research, related to sports such as cycling or athletics. The device allows to track which muscle groups are working as expected based on proper muscle activation (EMG studies) and whether there are imbalances that could be harmful or cause injury [37].

Myontec currently offers it on the market for 939 € and can be divided into three distinct parts: *MCell 3*, *Mbody 3* shorts and *Mbody Live 3* (Figure 4.1.1.1).

***Figure 4.1.1.1**.- Mbody 3 kit [37].*

4.1.1.1. MBody 3 Shorts

This element incorporates the electronic components and wiring integrated into a sports compression mesh. To measure the electromyogram signals of the different leg muscles during exercise (hamstrings, quadriceps, and glutes), textile electrodes arranged at specific points are used, as shown in Figure 4.1.1.1.

The use of textile electrodes in this device has certain advantages such as those already described above. *Mbody 3* Shorts offer great comfort and ease of placement, the possibility of being used regularly (can be washed), and allow the reduction of movement devices, obtaining a good signal and noise ratio in all exercises performed [37].

4.1.1.2. MCell 3

This component is intended to be used in conjunction with Mbody 3 shorts. It is a lightweight device that analyses EMG signals and inertial sensory data obtained. This analysis computes muscle overload warnings, training instructions, and postural corrections, and transmits this information to your cell phone wirelessly [37].

4.1.1.3. Mbody Live 3

Finally, a mobile app is included that transcribes the data provided by the MCell 3 into audio feedback (using headphones) to the athlete [37].

4.1.2. TeleMyo™

TeleMyo™ Direct Transmission System (DTS) (Figure 4.1.2.1) is a portable medical device from NORAXON company that directly transmits data from the electrode or sensor site to a belt-worn receiver [38]. This direct transmission concept greatly simplifies the arrangement of EMG measurements by eliminating the need to arrange cable connections between the EMG electrodes and EMG amplifier. The small light weight probes are also beneficial for small subjects like children and small animals. The belt Receiver can operate in 3 modes:

1. Direct connection to any PC via USB.
2. Wireless retransmission of signals in real-time to any NORAXON USB receiver.
3. Data logging via FLASH Memory card.

Figure 4.1.2.1.- *TeleMyo™ Direct Transmission System (DTS) product from Noraxon Company [38].*

The following table (Table 4.1.2.1) shows the main features of the *TeleMyo™* product:

Table 4.1.2.1.- *Main features of TeleMyo™ product form NORAXON [38].*

Key Features	- Free electrode type solution, fine wire included - 4 to 32 channel configuration (2 belts receivers) - Retransmission range up to 100 m - Compatibility existing NORAXON hardware and software - Optional fine wire amplifiers with the selectable band with
Power Requirements	- Replaceable Li-ion Rechargeable battery with an operation time of more than 8 hours when fully charged
Output and Transmission Frequency	- Up to 100 mW - DTS probe transmission range: 10 m - Up to 8 selectable radio channels (2412-2464 MHz)
EMG Sensor Data Acquisition System	- 16-bit resolution - Sample rate: 3000/1500 for 8/16 channels - Selectable LPF: 500, 1000, 1500 Hz
EMG Preamplifier Leads	- No notch (50/60 Hz) filters - 1st order HPF set to 10 Hz ± 10 % cut-offs - Baseline noise: < 1 mV_{Rms} - Input impedance > 100 MΩ - CMRR > 100 dB - Input Range ± 7 mV - Base gain 200 - Snap style terminal electrode connections

4.1.3. FREEEMG

FREEEMG (Figure 4.1.3.1) is a portable electromyography with wireless sensors for dynamic analysis of muscle activity, offered by BTS Bioengineering with at a price of 25.000$. The complete absence of cables, the lightness (13 g) and the *extremely* small size of the probes (41.5 × 24.8 x 14.0 mm) allow to carry out analysis of any type of movement, for each region of the body, without any way altering the motor gesture of the examined subject [39].

Figure 4.1.3.1.- *FREEEMG product from BTS Bioengineering [39].*

The system communicates with a PC via the provided USB receivers and can handle up to 20 sensors at the same time. In addition, each probe is equipped with an internal memory and a battery (with 6 hours of autonomy) to guarantee the continuity of the recording, even in the event of a momentary loss of connection, allowing acquisitions to be made over long distances and in the open field. The different probes attach directly to pre-gelled electrodes to collect the signal, with no additional hardware required. Table 4.1.3.1.1 shows the main characteristics of the product.

Additionally, the software processes the information captured for the return of the results in graphic form and allows an immediate comparison with the normal classes. Moreover, a wide selection of predefined analysis protocols is used to assess muscle activity during specific exercises.

In fact, this is the portable electromyograph used in Collado's thesis to measure the activity of the quadriceps muscles after a total knee arthroplasty.

Table 4.1.3.1.1.- *Main features of BTS FREEEMG product from BTS Bioengineering [40].*

Key Features	- Free electrode type solution, fine wire included - Retransmission range up to 20 m in free space (without obstacles) - Certification class IIa
Power Requirements	- Replaceable Li-ion Rechargeable battery with an operation time of more than 6 hours when fully charged
Memory	- Up to 1 hour and 40 minutes for systems with less than 6 EMG probes - Up to 2 hours for systems with more than 6 EMG probes - On board solid-state buffer memory system
Probes and Transmission Frequency	- Up to 20 EMG wireless probes - Wireless IEEE802.15.4 data transmission (probes - USB receiver) in Real-Time
EMG Sensor Data Acquisition System	- 16-bit resolution - Sample rate: 1 kHz
BTS EMG-Analyzer	- The most complete Software solution for sEMG analysis - Suitable for: - Predefined templates for any kind of protocol (jump, gait, fatigue analysis, isokinetic…) - Database for data storage. - Graphic interface to build new analysis protocol template (filtering, spectrum - computation, contact event measure, latencies, thresholds, fixed and mobile windows integration, RMS, interpolation, fatigue analysis…) - Tool for report customization in PDF format.
Size and weight	- 41.5 x 24.8 x 14.0 mm mother electrode - Ø 16 x 12 mm satellite electrode - 13 grams (battery included)

4.2. BIA portable measuring devices

4.2.1. BIA 101 ANNIVERSARY

The portable medical device *BIA 101 Anniversary* (Figure 4.2.1.1) *offered by* IK AKERN measures resistance (R) and reactance (Xc) of human tissue by a low-voltage and non-susceptible current [41]. In addition, the BIA 101 ANNIVERSARY is used in Pedemonte's thesis for measuring the quality of the quadriceps muscle after a total knee arthroplasty.

Figure 4.2.1.1.- *BIA 101 Anniversary from IK AKERN [41].*

Phase angle and body compartments are derived by the *Hydragram*, *Nutrigram* and *BIVA Analysis software* through medically validated algorithms. The bioimpedance results are based on a reading of bio-electrical data only. This clinically validated and meanwhile widely practiced method facilitates body analysis also under difficult conditions, e.g. in obesity, nephrology, oncology, and cardiology. The main characteristics can be seen in Table 4.2.1.1.

Table 4.1.1.31.- *Main features of BIA 101 Anniversary from IK AKERN [41].*

Key Features	- Backlit LCD Display - Parameter indicators: Resistance, Reactance, Phase Angle - Battery status indicator - CE 0051 Class II A according to EU standards 93/42/EEC
Battery	- Lithium-ion rechargeable - 6 hours of uninterrupted operation
Output and Transmission Frequency	- 0.5 mA constant at 50 kHz
LBIA Acquisition System	- Resistance (R): 0-999 ohms; resolution: 0.1 ohms - Reactance (Xc): 0-q00 ohms; resolution: 0.1 ohms

4.2.2. Quantum V

The Quantum V Segmental is a Bioelectrical Impedance Analysis (BIA) medical device from the industry originator RJL Systems (Figure 4.2.2.1).

Figure 4.1.1.31.- *Quantum V Segmental from RJL Systems [42].*

The Quantum V Segmental has been enhanced with the ability to perform segmental and localized body composition assessments on 13 zones of the human body. This Class II medical device provides fast and accurate segmental body composition assessments of fat and lean soft tissue (LST) [42].

The Quantum V Segmental uses an eight-lead, 12-channel multiplexer to quickly measure resistance and reactance values from each arm, each leg and the right and left torso, including the upper and lower regions of the human body. The repeatability and accuracy of the resistance and reactance measurements allow the smallest changes to be recorded with 0.1 ohms of resolution.

The main features of the device are exposed in Table 4.2.2.1.

Table 4.2.2.1. *Main features of Quantum V from RJL Systems [42].*

Key Features	- 8 electrodes can be used at the same time - The device provides results within 15 seconds - Bright OLED display is easy to read
Battery	- 9 hours on a single charge
Memory	- Memory can store more than 2000 records
Communication	- Bluetooth - USB port

5. Signal Processing

In this section, the main concepts of the required signal processing and parameter acquisition of both EMG and LBIA are exposed.

5.1. EMG Signal Processing

The obtained raw EMG signal contains valuable information that will serve as a first estimator of muscle activation. However, to obtain the desired quantitative amplitude and frequency parameters is required to apply some specific processing steps to increase de reliability and utility of the recordings. The most important digital processing procedures (as recommended by scientific associations like SENIAM [28]) will further be explained [43].

5.1.1. Full wave rectification

To obtain the standard amplitude parameters from the curve of the signal is necessary to start with a full wave rectification (since raw EMG has a mean value close to 0). In addition, this process results in a signal easier to read and evaluate. To perform this operation, different algorithms can be used to rectify all the negative values. The result of this process can be seen in Figure 5.1.1.1.

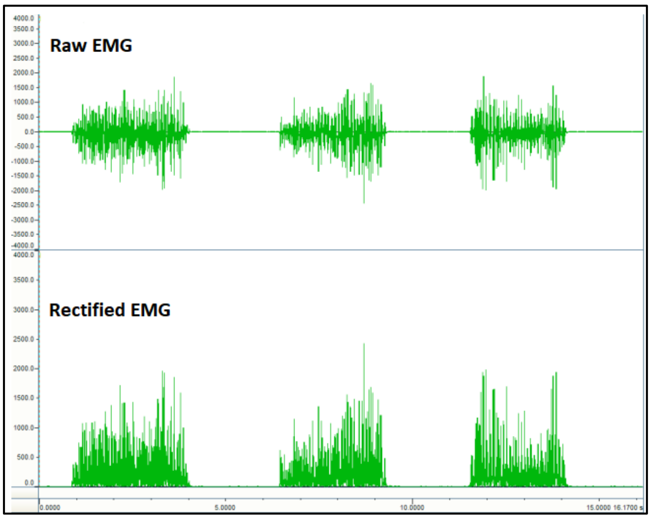


Figure 5.1.1.1.- *Raw EMG and Resulting EMG after a Full wave rectification [43].*

In the present thesis, the full wave rectification is implemented analogically to avoid overcharging the microcontroller with excessive processing load since it can be easily implemented with a full-wave precision rectifier.

5.1.2. Smoothing

As previously stated, the interference pattern of EMG is considered random since the number of recruited motor units changes constantly and the action potentials superpose in an arbitrary way. For this reason, EMG signals cannot be reproduced once again with the exact same shape. Besides this issue, there exist a variety of different methods based on digital smoothing algorithms to minimize the non-reproducible part, outlining the mean trend of the signal. Consequently, the present amplitude spikes are eliminated, leaving only the "linear envelope" of the signal [43]. In all these algorithms, is necessary to select a window to be used during the signal processing. This window is essentially characterized by two factors:

1. The number of points, known as the window length, corresponding to the length of the signal to be processed in each step [44].
2. The weight value attributed to each point in the window, known as the window type (such as Hamming, Hanning, Rectangular or Triangular) [44].

Once the window's length and type are defined, a representative value for each window can be calculated. Finally, the signal will then be represented by all the calculated values, forming the smooth envelope stated before.

The main established digital smoothing algorithms are the following (Figure 5.1.2.1):

Moving average (Movag): The moving average filter takes N samples of input at a time and takes the average of those to produce a single output point (Eq. 5).

$$MAV = \frac{\sum_{i=1}^{N} |EMG_i|}{N} \quad \textbf{(Eq. 5)}$$

As the length of the window selected increases, the smoothness of the output increases.

Root Mean Square (RMS): Based on the square root calculation, the RMS reflects the mean power of the signal (eq. 6) and is the preferred recommendation for smoothing [43].

$$RMS = \sqrt{\frac{\sum_{i=1}^{N} EMG^2}{N}} \quad \textbf{(Eq. 6)}$$

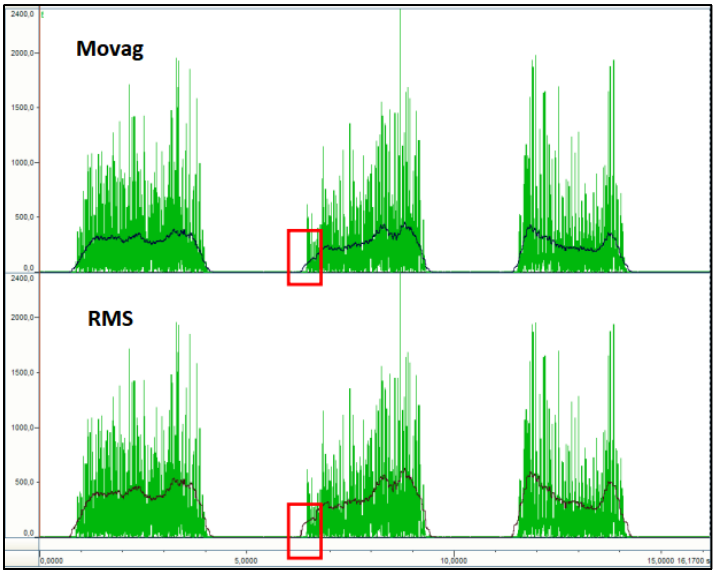


Figure 5.1.2.1.- *Comparison of two smoothing algorithms using the same window width [43].*

In kinesiological studies such as the ones performed in the knee prosthesis evaluation have a typical time window between 20 ms and 500 ms [43]. It is also important to take into account that with a higher time window, the chances of a phase shift with a steep signal increase (as outlined with the red rectangle in Figure 5.1.2.1) are bigger. Considering this fact, a value that provides the desired results in most conditions is 100 ms [43].

Another alternative to these methods is the application of a low-pass digital filter at 6 Hz (for example a Butterworth of 2nd order or higher) to create the desired linear envelope. One benefit of this choice is that higher order digital filters can be applied recursively, minimizing the phase shift phenomenon. Although this possibility exists, the smoothing method selected in this device is the RMS since it is the recommended technique in most of the literature [43], and its calculation will be also used for determining the amplitude of the signal in the time domain.

5.1.3. Digital filtering of the powerline noise

Since the required band pass filtering of the EMG signal is done analogically, it is not necessary to implement digital filters in most of the cases [43]. Moreover, scientific recommendations for research studies (SENIAM, ISEK) do not recommend implementing a notch filter to get rid of the powerline noise of 50/60 Hz because as previously explained, damages too much of the actual EMG signal power [43]. However, the power line interference can be detrimental to several methods of analyzing EMG. In the time domain, it can obscure the start of the contraction, whereas in the frequency domain, it can affect calculations of the mean and median frequencies [43]. In order to solve this problem, different alternatives have surged such as Spectrum Interpolation and Adaptive Filtering [45].

5.1.3.1. Spectrum Interpolation

The true power spectrum of an EMG signal corrupted with an additive sinusoidal interference (power line noise) is considered to be a continuous curve, with a superimposed peak at the interference frequency (ω_o) [46]. Then, the value of the true power spectrum at ω_o can be estimated by interpolation of the curve at ω_o. If this interpolation were to be performed on the discrete Fourier transform (DFT) of the signal, instead of the power spectrum, the inverse transform could then be taken to provide a signal with reduced interference [46]. Both the positive and negative frequency components of the interference must be interpolated so that a real-valued signal is returned. This process called spectrum interpolation, effectively implements a notch filter with limited attenuation instead of an infinite null. Because the DFT is complex-valued, not only its magnitude but also its phase could be interpolated. *T. Mewett et al*. [46] used a linear interpolation for computational simplicity in their experiments reported, although more sophisticated curve-fitting algorithms can also be applied [46].

5.1.3.2. Adaptative filtering

Adaptative filters are best used in cases where signal conditions or systems parameters are slowly changing, and the filter needs to be adjusted to compensate for this variation [47]. The basic adaptive structure for a noise cancellation application can be seen in Figure 5.1.3.2.1.

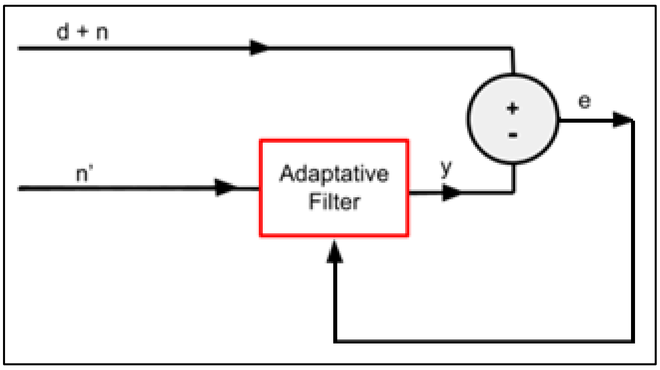


Figure 5.1.3.2.1.- *Basic adaptive Filter structure for noise cancellation. Own source.*

The desired signal *d* is corrupted by uncorrelated additive noise *n*. The input to the adaptive filter is a noise *n'* that is correlated with the noise *n*. The noise *n'* could come from the same source as *n* but modified by the environment. The adaptive filter's output *y* is adapted to the noise *n*. When this happens, the error signal (*e*) approaches the desired signal *d*. The overall output is this error signal and not the adaptive filter's output y [47].

When the main purpose of the filter is to eliminate the power line noise from an EMG signal, the least mean squares (LMS) criterion is a search algorithm that can be used to provide the strategy for adjusting the filter coefficients [47]. The LMS adaptive filter developed using digital functions is designed to remove the contaminating signal, as shown in Figure 5.1.3.2.2.

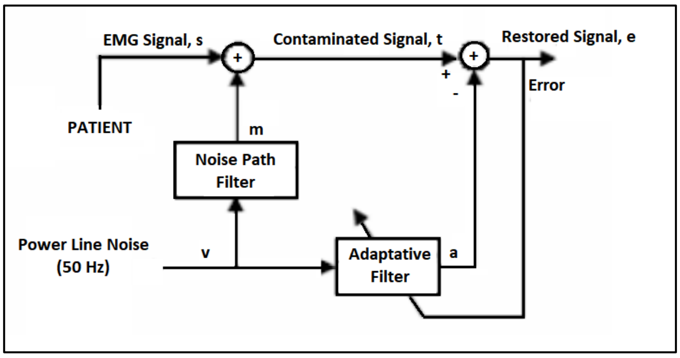


Figure 5.1.3.2.2.- *LMS Filter structure for noise cancellation. Own source.*

The EMG signal (*s*) is the original uncontaminated input signal. The desired output is the contaminated EMG signal *t*. The Adaptive Filter does its best to reproduce this contaminated signal, but it only knows about the original 50 Hz interference, *v* [47]. Due to that fact, it can only reproduce the part of *t* that is linearly correlated with *v*, which is *m*. In effect, the Adaptive Filter will attempt to mimic the noise path filter, so that the output of the filter *a* will be close to the contaminating noise *m*. In this way, the error *e* will be close to the original uncontaminated EMG signal *s*. The primary input is called *(s+m)* and the reference signal, *a*. Since the Adaptive Filter output is *a* and the error is *e* then the mean square error (MSE) is: [47].

$$Re^2 = ((s+m(-a)^2 = (s+m)^2 - 2(s+m)a + a^2 \quad \textbf{(Eq. 7)}$$

$$(m-a)^2 + s^2 + 2sm - 2sa \quad \textbf{(Eq. 8)}$$

Since the signal and the noise are not correlated, the MSE is:

$$E[e^2] = E[(m-a)^2] + E[s^2] \quad \textbf{(Eq. 9)}$$

Minimizing the MSE results in a filter error that is the best least-squares estimate of the signal s. The adaptive filter extracts the signal, or eliminates noise, by iteratively minimizing the MSE between the primary and the reference inputs [47]. *Behbahani et al.* [47] implemented this type of adaptative filter to remove the power line interference from an ECG measurement. Using this filter, the ECG signal was filtered without losing any relevant information and the same results would be expected for an EMG signal [47]. Even though these two options could be used to reduce the effect of the powerline noise, in this thesis none of them are used due to the high computational power they would require.

5.1.4. Amplitude Normalization

One big drawback of any analysis of an EMG measurement is that the amplitude data is strongly influenced by the detection condition. Its analysis can noticeably vary between electrode sides, subjects or even day-to-day measurements on the exact same muscle [43].

To overcome this uncertain character of the EMG parameters, the signal needs to be normalized to a reference value, for instance, the maximum voluntary contraction (MVC) [43]. The main idea is to calibrate the obtained voltage value to a unique calibration unit with physiological relevance, the "percent of maximum muscle activation" [43]. In this normalization method, the influence of the given detection condition is eliminated, and the data is rescaled from mV to a percent of the selected value. It is of high importance to mention that this process doesn't alter the shape of the EMG curves, only their vertical axis scaling [43]. The MVC normalization is the most popular normalization method, and it should be done prior to any measurements if it is not impeded by any injury of the patient (then other processing techniques should be considered) (Figure 5.1.4.1) [43].

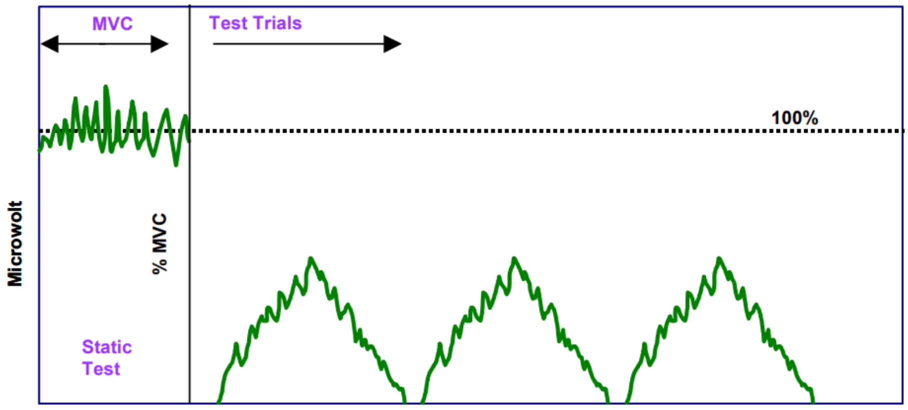


Figure 5.1.3.25.1.4.1.- *MVC normalization. Prior to the test/exercises a static MVC contraction is performed for each muscle. This MVC innervation level serves as reference level (=100%) for all forthcoming trials [43].*

This procedure is performed against static resistance and needs to be done for each investigated muscle of the quadriceps muscle group separately. By using the MVC normalization technique, the processing of the EMG data can provide more understanding of at what capacity level the muscles were activated. Furthermore, this allows a direct quantitative comparison of EMG findings between subjects since it eliminates any varying influence of local signal detection conditions [43]. Due to this fact, the normalization process using MVC will be required prior to any EMG measurement done with the designed medical device for this work.

5.1.5. Standard amplitude parameters

After the preliminary smoothing and normalization processes, the signal amplitude of EMG can be calculated with a wide variety of parameters, such as mean, peak, minimum value, area and slope [48]. However, the *International Society of Electrophysiology and Kinesiology (ISEK)* [44] states that the standard time parameters to analyse the EMG amplitude are the Mean Amplitude Value (MAV) and the Root Mean Square (RMS) (also used as a smoothing method). The MAV parameter is useful because it is less sensitive to duration differences in analysis intervals. It describes the muscle activation of a selected muscle for a given task and works best for comparison analysis [43]. Additionally, RMS is used to quantify the electric signal because it reflects the physiological activity during contraction [44].

5.1.6. Calculation of the frequency contents

To obtain the frequency domain parameters of the EMG is necessary to implement a mathematical transformation to the signal:

The Fourier transform (FT) converts a given signal f(t), into its frequency spectrum, which represents the signal in terms of infinite complex sinusoids of different frequency, *v*, and phase [49]:

$$F(v) = \int_{-\infty}^{+\infty} f(t)e^{-i2\pi vt}\,dt \qquad \textbf{(Eq. 10)}$$

The FT transforms the signal entirely between the time domain to the frequency domain. The spectrum provided by this transform represents the average frequency content of the signal, being ideal for stationary signals. The *Continuous Fourier Transform* can be calculated analytically according to Equation 10 but the computation of the FT for arbitrarily measured signals requires a discrete formulation. [49]. The *Discrete Fourier Transform (DFT)* is calculated on a discretely sampled finite signal and provides a discretely sampled finite spectrum [39].

The *Fast Fourier Transform (FFT)* utilizes a divide-and-conquer approach to calculate the DFT more efficiently [49]. Consequently, computing the FFT is much faster than the DFT, making it the preferred choice [49]. The FFT algorithm is described as the decomposition of the EMG signal to its underlying sinus components. The amplitude part of each frequency component is determined and assigned to the respective frequency in a graph (Figure 5.1.6.1). It is important to state that Fourier spectra are normally complex-valued and include both positive and negative frequencies whereas their absolute value is represented. If this analysis is done in a defined frequency range, it is called 'Total Power Spectrum' (Figure 5.1.6.1).

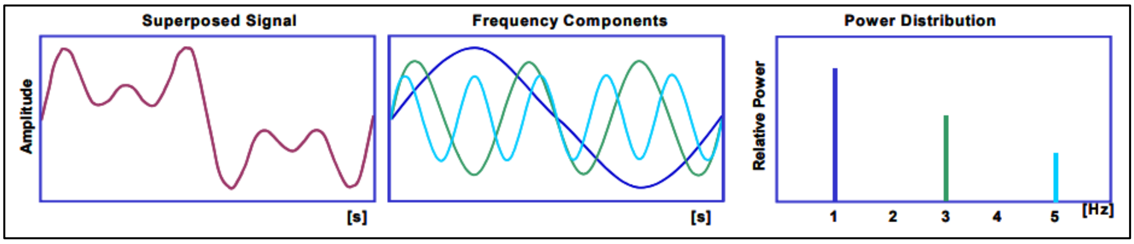


Figure 5.1.6.1.- *Model of frequency related signal decomposition based on FFT. The power distribution (right) indicates Power of different magnitude at three frequencies. [43].*

In addition, more complex and powerful technologies are used in order to obtain a time-frequency approach addressing the non-stationary nature of EMG signals such as the Wavelet Transform

(WT), Higher-Order Statistics (HOS) and the Empirical Mode Decomposition [50]. However, the implementation of these algorithms has been discarded due to the high time and computing power consumption they require, since providing a fast evaluation method is a key requisite of the medical device. Based on the exposed facts, the FFT is the technique chosen to obtain the frequency domain parameters of the EMG in this thesis.

5.1.7. Frequency domain parameters

The total Power Spectrum can be analyzed by different frequency parameters (Figure 5.1.7.1):

- Mean frequency (MNF): Mathematical mean of the spectrum curve.
- Total Power: Integral under the spectrum curve
- Median Frequency (MDF): Parameter that divides the Total Power area into two parts of equal size.
- Peak Power: Maximum value of the spectrum.

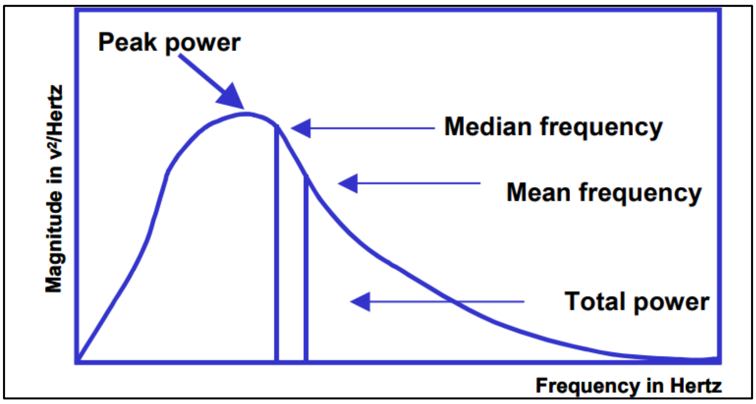


Figure 5.1.7.1.- *EMG standard frequency parameters based on FFT calculations. [43].*

The most important frequency parameters are the Median frequency and Mean frequency [43].

The MNF is the average frequency which is calculated as the sum of the product of the EMG power spectrum and the frequency divided by the total sum of the power spectrum [51]. The mathematical definition of MNF is given by:

$$MNF = \sum_{j=1}^{M} f_j P_j / \sum_{j=1}^{M} P_j \quad \textbf{(Eq. 11)}$$

Where f_j is the frequency value of EMG power spectrum at the frequency bin *j*, P_j is the EMG power spectrum at the frequency bin *j*, and *M* is the length of the frequency bin. In the analysis of the EMG signal, *M* is usually defined as the next power of 2 from the length of EMG data in the time domain [51].

On the other hand, the MDF is the frequency at which the EMG power spectrum is divided into two regions of equal amplitude [51]. MDF is also defined as half of the total power, or TTP (dividing the total power area into two equal parts). The mathematical definition of MDF is given by [51]:

$$\sum_{j=1}^{MDF} P_j = \sum_{j=MDF}^{M} P_j = \frac{1}{2}\sum_{j=1}^{M} P_j \quad \textbf{(Eq. 12)}$$

The behaviour of MNF and MDF is similar [51]. However, it should be noted that MNF is always slightly higher than MDF because of the asymmetrical shape of the EMG power spectrum [51]. In addition, the variance of MNF is typically lower than that of MDF, being the standard deviation of MDF higher than that of MNF by a factor of 1.253 [44]. On the contrary, the estimation of MDF is less affected by random noise, particularly in the case of noise located in the high-frequency band of EMG [51]. The notion that MPF and MDF decrease in effect to fatigue generation in muscle is well accepted in scientific society [51]. As fatigue develops, the EMG spectrum shifts towards lower frequency. In addition, is well established in the literature that the mean frequency of the power spectrum is proportional to propagation velocity [51]. Overall, MNF and MDF features extracted from the EMG signal are the optimal variables to identify muscle fatigue in a dynamic muscle contraction [51]. For these reasons. the MNF and the MDF will be the frequency parameters obtained from any EMG signal acquired by the medical device developed in this present thesis.

5.2. LBIA signal processing

The LBIA data acquired is processed in order to obtain the values of bioimpedance (Z). Both components of Z (resistance and reactance) are obtained by a digital demodulation process. Digital demodulation is very similar to analog demodulation. The main difference is that in the case of digital demodulation, the acquired signal is sampled, and demodulation is performed over several complete cycles of the signal period [52]. The sampled current and voltage are demodulated in order to obtain their amplitude and phase [52]. This process can be done with different digital systems such as:

- Double balanced modulator [52]
- I-Q demodulator [53]
- Peak and Phase demodulation [54]
- Phase Locked Loop demodulation [55]
- Quadrature amplitude demodulation [56]

One possible method to measure the amplitude and phase of a signal is to use a peak detector for the amplitude and a phase detector, based on zero crossings, for the phase. Nevertheless, this is not a good approach in the case of bioimpedance measurements where the amplitude of the injected current is very low (below 1 mA_{PEAK} in most cases [57]), and the environment is quite noisy [52]. An Inadequate sampling of data can result in inaccurate values for the locations and amplitudes of peaks, and the non-detection of valid peaks [52]. In addition, the appearance of high-frequency noise could result in the detection of many peaks, but only a few of these will be of interest [52]. Consequently, it is advisable to use demodulation to reject the noise or the interferences outside the frequency range of interest [52]. For this reason, a synchronous demodulation (quadrature amplitude demodulation) method is used in this thesis (Figure 5.2.1), based on a digital lock-in amplifier (explained in section 2.3.5.

This system involves a signal that is multiplied (mixed) by a reference signal with the same frequency that is both in-phase and 90 degrees out of phase (the quadrature component). This process results in the extraction of the signal in two components [58]. The two components are called I (V_R in this case) and Q (V_{Xc} in this case) and must be low-pass filtered to remove the 2f frequency component (as previously stated in section 2.3.5). This process causes the lock-in to focus on the signal exactly at the reference signal frequency and ignore the rest of the frequencies (Figure 5.2.1).

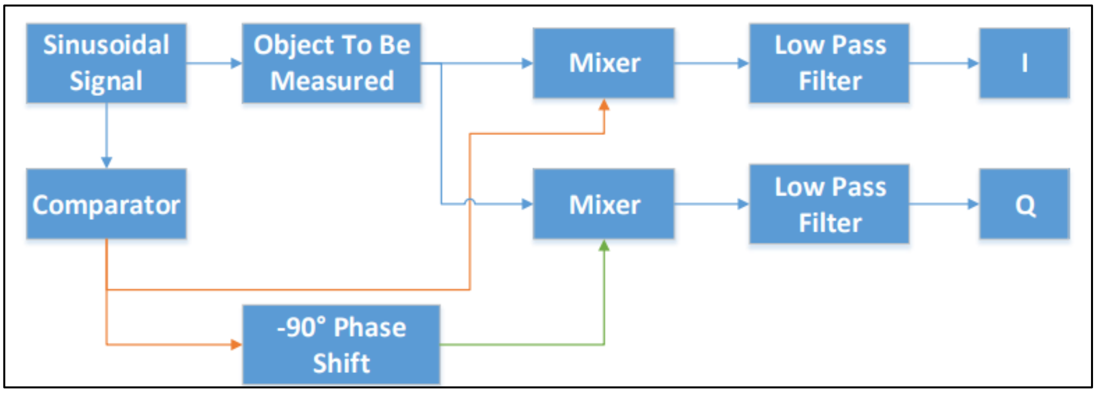


Figure 5.2.1.- *Synchronous demodulation block diagram. [58].*

If we have a signal S(t) and a square reference signal R(t) both with no DC and the reference signal has no phase offset and an amplitude of 1:

$$S(t) = A_s \, sin(\omega_s t + \theta) \quad \textbf{(Eq. 13)}$$

$$R(t) = sin(\omega_r t) \quad \textbf{(Eq. 14)}$$

When multiplying these two signals we get:

$$M(t) = S(t) \cdot R(t) = A_s \, sin(\omega_s t + \theta) \, sin(\omega_r t) \quad \textbf{(Eq. 15)}$$

$$M(t) = \frac{A_s}{2} cos[(\omega_s - \omega_r)t + \theta] - \cos\,[(\omega_s + \omega_r)t + \theta] \quad \textbf{(Eq. 16)}$$

Since the angular frequencies are equal, we get:

$$\omega = \omega_r = \omega_s \quad \text{(Eq. 17)}$$

$$M(t) = \frac{A_s}{2} cos\ (\theta) - cos\ (2\omega t + \theta) \quad \text{(Eq. 18)}$$

After the mixing of the interest signal and the reference signal, the result is a DC component and another component at twice the frequency of the original signals. If the component at twice the signal frequency is attenuated using a low-pass filter, M(t) is a DC signal whose amplitude only varies with the cosine of the phase difference between the two signals [58]. The in-phase and quadrature-phase signals are then:

$$V_R(t) = \frac{A_s}{2} cos\ (\theta) \quad \text{(Eq. 19)}$$

$$V_{Xc}(t) = \frac{A_s}{2} cos\left(\theta - 90^{\circ}\right) = \frac{A_s}{2} sin\ (\theta) \quad \text{(Eq. 20)}$$

After that, to obtain the respective $R(t)$ and $X_c(t)$ values, it is necessary to divide the voltage signals with the peak value of the injected current. To do that, these equations are followed:

$$R(t) = \frac{V_R(t)}{I_{peak}} \quad \text{(Eq. 21)}$$

$$X_c(t) = \frac{V_{Xc}(t)}{I_{peak}} \quad \text{(Eq. 22)}$$

Here R(t) is the real part and Xc(t) is the imaginary part of the bioimpedance phase signal. To obtain the corresponding R and X_c values, it is necessary to perform a mean operation on these signals. After the amplitude and the phase of the Bioimpedance vector can be calculated:

$$Z = \sqrt{R^2 + {X_c}^2} \quad \text{(Eq. 23)}$$

$$\varphi(t) = \arctan\left(\frac{X_c}{R}\right) \quad \text{(Eq. 24)}$$

6. Design specifications

To build a portable medical device that performs the acquisition of LBIA and EMG signals and the corresponding feature abstraction, it is necessary to previously stablish some specifications and requirements. To clearly understand what functionalities and characteristics were required for the device, a set of LBIA and EMG measurements were done in the Hospital Germans Trias i Pujol in 2021 and 2022.

6.1. Device requirements for its use in a clinical setting

Since the medical device of this present thesis is developed with the main objective of covering the needs of Pedemonte and Collado's thesis in respect of EMG and LBIA measurements, all the requirements imposed by them are exposed in this section.

6.1.1. General Requirements

The general requirements for the developed device are the following:

- External device to medical device connection and communication (to send commands and receive data back).
- Fast EMG/BIA parameters acquisition.
- Error handling functionalities to assess if the device is performing correctly.
- Visual indicator of the current state of the device (minimum of 4 LEDs).

6.1.2. LBIA Requirements

The LBIA measurement specific requirements are the following:

- LBIA Sampling frequency of 250 kS/s (5x the frequency of interest).
- LBIA acquisition of 10 periods of the signal.
- LBIA acquisition phase angle from 0 to 20º.

6.1.3. EMG Requirements

The EMG measurement specific requirements are the following:

- 3 Channels (with simultaneous acquisition).
- EMG and MVC sample rate of 5 kS/s for each channel (10x the highest frequency of interest).
- MVC acquisition with a duration of 6 s.
- EMG acquisition with a duration of 10 s.
- Frequency resolution below 1 Hz

6.2. Hardware specifications

Although the hardware design of this medical device is out of the scope for this present thesis, it is important to know some specifications of that design to better integrate the developed firmware design.

6.2.1. LBIA Hardware specifications

The LBIA hardware device has the characteristics exposed in Table 6.2.1.1.:

Table 6.2.1.1.- *Hardware design LBIA Measurement-related parameters.*

	PARAMETER	VALUE
VOLTAGE	**Signal amplification**	90 V/V
INJECTED CURRENT	**Output Amplitude**	250 μA_{RMS}
	Signal amplification	90 V/A

6.2.2. EMG Hardware specifications

The EMG hardware device has the characteristics exposed in Table 6.2.2.1.

Table 6.2.2.1.- *Hardware design EMG Measurement-related parameters.*

	PARAMETER	VALUE
GENERAL	**Input Amplitude**	50 µV to 5 mV
	Output Amplitude	2.5 V peak (Max)
	Bandwith	20 Hz to 500 Hz
AMPLIFICATION	**Gain**	500 V/V
FILTERING	**Suppression of Motion Artifacts and PLN**	At 20 Hz and 50 Hz, respectively
	HPF	20 Hz
	LPF	500 Hz
	Nyquist Frequency Attenuation	≥ 30 dB at 2500 Hz

6.3. Block diagram of the firmware design

The designed firmware of the medical device developed in this thesis (further explained in the following sections) can be represented as:

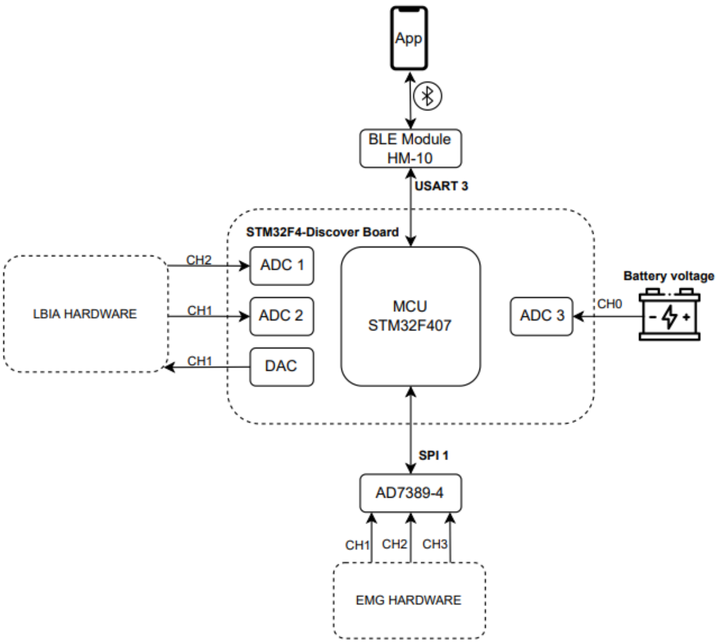


Figure 6.3.1.- *Firmware block diagram. Own source.*

As can be seen in the previous figure, the ADC1, ADC2, and DAC from the STM32F4 Board are used in the LBIA measurements (section 8.3.6). In addition, the board ADC3 is used to obtain the battery voltage value (section 8.3.3). On the other hand, external chips have been implemented, the AD7389-4 is used for EMG measurements (section 8.3.5) and the BLE module HM-10 is used to communicate with an external device (using a specific App) (section 8.3.4). However, it is important to state that the development of this mobile application has not been developed in the present thesis.

7. Microcontroller (MCU)

7.1. Microcontroller selection

To implement the firmware design of the medical device developed in this thesis, it is necessary to select a microcontroller that fulfills all the technical requirements and design specifications (stated in section 5.1). Due to the worldwide shortage of components, we will use a development board. At the moment, there are only a few boards that could be obtained. In this section, the main features of these boards will be exposed in order to select the more appropriate choice to be integrated into the design.

7.1.1. STM32F3-Discovery

This board is part of the Discovery Kits from *STMicroelectronics.* The STM32F3DISCOVERY allows users to develop applications with the STM32F3 Series mixed-signal Microcontroller STM32F303VCT6 based on the Arm Cortex-M4 (Figure 7.1.1.1) [59].

Figure 7.1.1.1.- *STM32F3-Discovery Board [59].*

7.1.2. STM32F4-Discovery

The STM32F4DISCOVERY Discovery kit from *STMicroelectronics* is built on the high-performance STM407VGT6 microcontroller based on the Arm Cortex-M4 (Figure 7.1.2.1) [60].

Figure 7.1.2.1.- *STM32F4-Discovery Board [59].*

7.1.3. ATMEL SMART SAM E70 Xplained

The SAM E70 Xplained evaluation kit uses an ATSAME70Q21 (Cortex®-M4 based) microcontroller from *Atmel (Microchip Technology)* (Figure 7.1.3.1) [61].

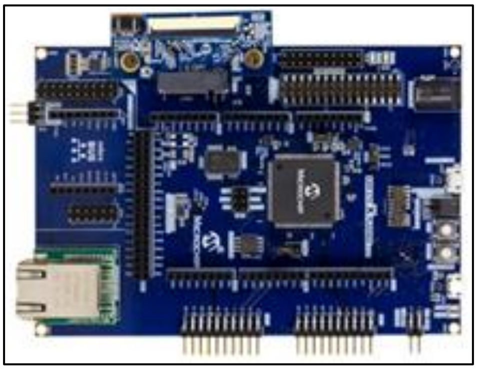


Figure 7.1.3.1.- *ATMEL SMART SAM E70 Xplained board [61].*

7.1.4. Board selection

One important criterion when choosing a development board is what facilities it provides when programming the code and how much related literature can be found. In this case, there is a considerably larger amount of useful bibliography about STM boards and projects based on them than the ATMEL board. Moreover, *STMicroelectronics* provides its users with the *STM32Cube MX,* a graphic tool that allows a very easy configuration of STM32 microcontrollers [62]. The software tool allows the selection of an existing board, the clock and peripherals configuration, the peripheral pin position, etc. Furthermore, it generates all the initialization code in C language to transfer the selected parameters to the board [62]. Consequently, it is much easier to develop the required firmware if it is based on one of these STM Boards. Additionally, the price of the ATMEL board is around 136 € [61], much higher than the STM alternatives (Table 7.1.4.1).

For all the exposed reasons, the ATMEL SMART SAM E70 is dismissed. To choose between the other two boards the following criteria are considered: Flash memory, SRAM, CAD, Peripherals (does the board provide the peripherals needed), available pins, programmer and debugger of the board, dimensions and price. This information is exposed in Table 7.1.4.1.

Table 7.1.4.1.- *Main features of the STM Boards 32F3-Discovery and 32F4-Discovery [60, 61].*

	STM32F3-Discovery	**STM32F4-Discovery**
Microcontroller	STM32F303VCT6	STM32F407VGT6
Flash memory	256 KB	1024 KB
SRAM	48 KB	192 KB
MCU max. frequency	72 MHz	168 MHz
Peripherals	- Motion sensor - 3-axis digital gyroscope - 8 LEDs (2 red, 2 blue, 2 orange and 2 green) - 4 ADCs (12 Bits) - 1 DAC (12 Bits) - 3 USARTS / 2 UARTS - 3 SPI - 10 Timers	- Digital microphone - 3-axis accelerometer - 4 LEDs (Red, Blue, Orange and Green) - 3 ADCs (12 Bits) - 1 DAC (12 Bits) - 4 USARTS / 2 UARTS - 3 SPI - 12 Timers
General Purpose pins	85 available GPIO pins	88 available GPIO pins
Programmer and debugger	On-board ST-LINK/V2 programmer and embedded debug tool	On-board ST-LINK/V2 programmer and embedded debug tool
Price	14,91 €/Unit	18,84 €/Unit
Dimensions	6,60 x 9,70 x 1,5 cm	6,60 x 9,70 x 1,5 cm

It can be seen in the previous table, that the dimensions and the programmer/debugger are the same. In addition, the number of available GPIO pins is similar. On the other hand, referring to the peripherals there are some differences. Since the digital microphone, motion sensor and gyroscopes are not intended to be used, these peripherals are not considered in this selection. In both cases, the ADCs provided by the boards are only 12 bits which will be used for bioimpedance and battery measuring processes (it will further be explained). Since the number of LEDs needed is four, the two discovery boards fulfill this requirement. Moreover, both have enough UART/USART, SPI and timers. Due to these facts, the board peripherals are not a crucial factor in this decision.

Another feature to consider is the memory of the board. All the acquired data will be stored as uint16_t rather than floats (since the space required for a uint16_t is 2 bytes, while a float needs 4 bytes of memory [63]). If the previously stated requirements are fulfilled, the following space will be needed during the measurement processes:

$$Mem.Space\ needed\ (MVC) = 3\ CH \cdot \frac{5\ kS}{s} \cdot 2Bytes \cdot 6s = 180.00\ kB \quad \textbf{(Eq. 25)}$$

$$Mem.Space\ needed\ (EMG) = 3\ CH \cdot \frac{5kS}{s} \cdot 2Bytes \cdot 10s = 300.00\ kB \quad \textbf{(Eq. 26)}$$

$$Mem.Space\ needed\ (LBIA) = 1\ CH \cdot \frac{250kS}{s} \cdot 2Bytes \cdot \frac{10\ periods}{\frac{50kH}{s}} = 0.10\ KB \quad \textbf{(Eq. 27)}$$

$$Total\ Mem.Space\ needed = 180\ kB + 300\ kB + 100\ B = 480.10\ kB \quad \textbf{(Eq. 28)}$$

As can be seen from the previous calculations, the amount of memory space needed only for the measurement processes is higher than the proportioned by the STM32F3. If this board was chosen, it would imply the necessity of integrating an external memory. However, the STM32F4 has 198 KB of SRAM and 1 MB of flash memory, which is enough. Furthermore, the STM32F4 maximum clock frequency is more than two times higher than its alternative. Since being able to store large series of data and process them with high efficiency and velocity, the SMT32F4 is a better choice. For this reason, even though the price is slightly higher, the STM32F4-Discovery board is the board chosen to be integrated into the developed medical device of this thesis.

7.2. STM32F4-Discovery Board (STM32F407VGT6)

Once the board and its microcontroller are selected, it is necessary to understand its basic features and structure prior to the actual design of the firmware. The block diagram of the STM32F4 Discovery board is exposed in Figure 7.2.1.

As previously exposed, the STM32F407 Discovery board uses the STM32F407VGT6 Microcontroller which has an ARM Cortex-M4F processor, and incorporates many peripherals such as GPIO ports, timers, ADCs and others. The processor and peripherals communicate via APBUS-Interface [64].

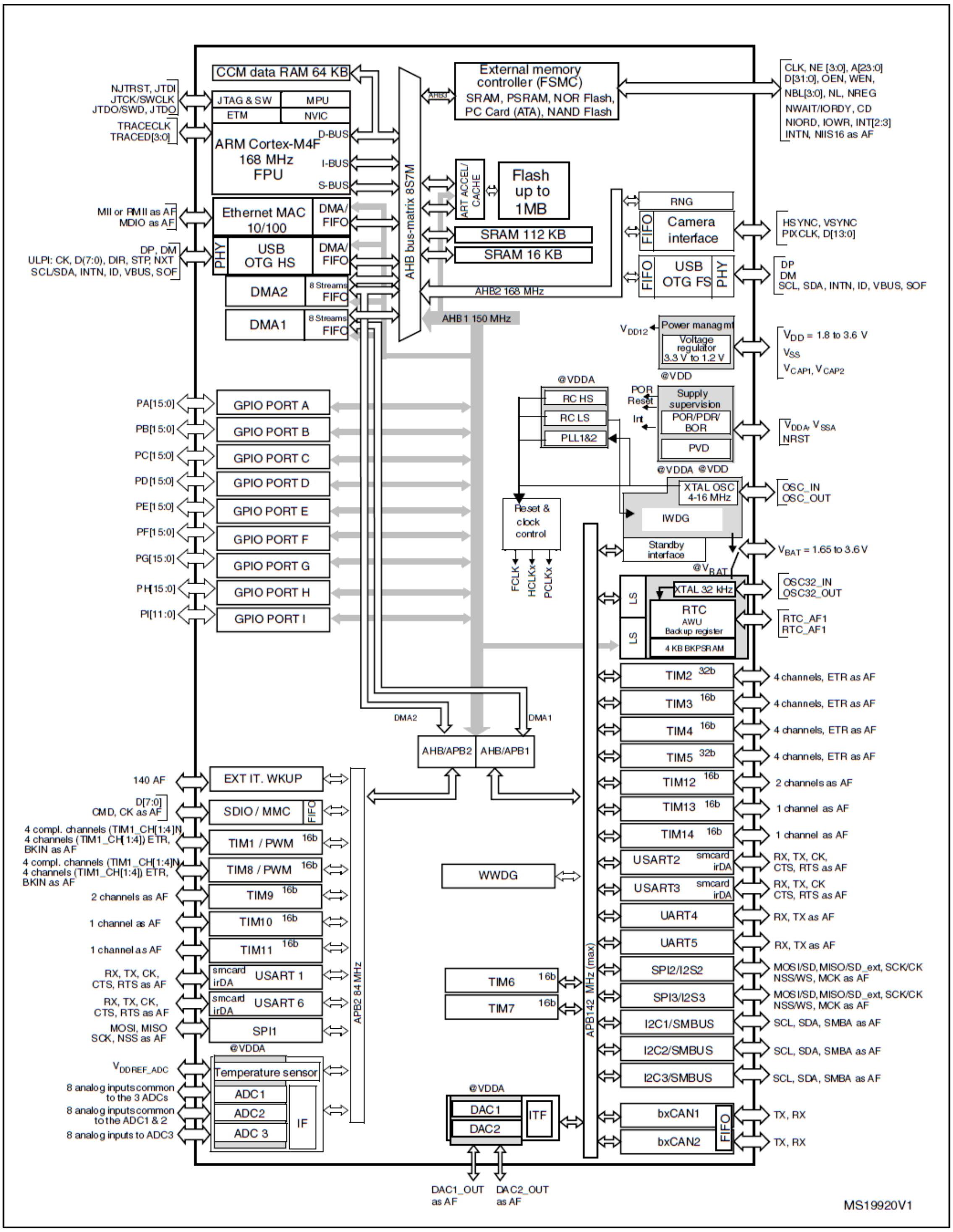


Figure 7.2.1.- *STM32F4 Discovery Board Block diagram [64].*

7.2.1. Bus Matrix

A schematic of the Bus Matrix can be seen in Figure 7.2.1.1.

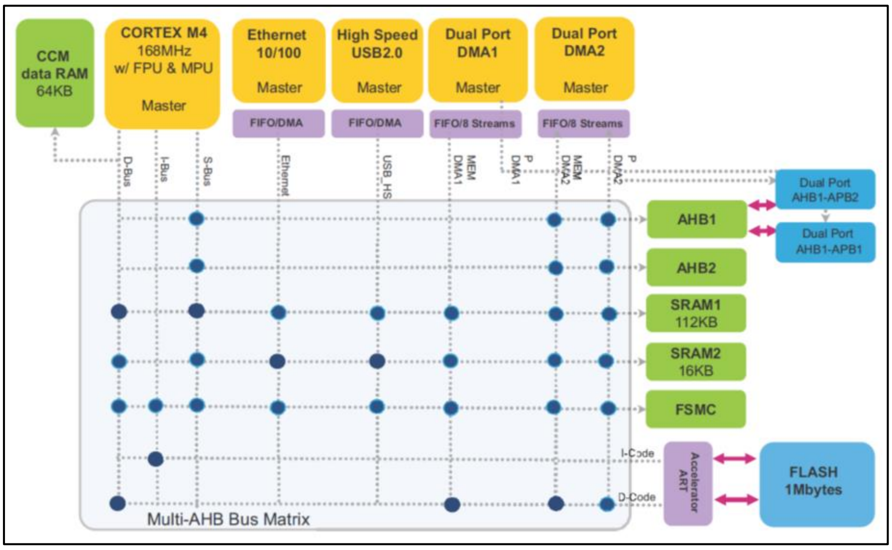


Figure 7.2.1.1.- *STM32F4 Bus Matrix [64].*

Referring to Figure 7.2.1.1, the yellow-colored blocks represent the masters whereas the green ones are slaves [64]. In the microcontroller, the communication between the processors and the peripherals is seen as communication between master and slave. In addition, each of the dots present in this picture indicates possible Master-slave communication [65].

7.2.2. Clock

STM32F407VGT6 Microcontroller has 3 main clock sources:

1. <u>Crystal Oscillator (HSE):</u> This High-Speed External clock source can be connected to MCU when it is required. If you intend to utilize the HSE as a system clock, an external clock with a frequency between 4 to 26 MHz must be connected. In this specific board, the manufacturer has connected an 8 MHz crystal [65].

2. <u>RC Oscillator (HIS):</u> The STM32F4 has an internal High-Speed RC oscillator (like the majority of MCUs). This oscillator with a frequency of 16 MHz is the selected clock of the microcontroller by default. Consequently, it will be activated if a reset is presented to provide a clock to the MCU [65].

3. <u>PLL (Phase Locked Loop):</u> Implemented internally in MCU, it uses low-frequency sources to generate a high-frequency clock (PLLCLK). PLL allows us to program different clock frequencies [65].

7.2.3. Vector Table

The vector table is a table that holds the specific addresses of exception handlers (Figure 7.2.3.1). Here system exceptions (MCU internally generated) and interrupts are collectively called as exceptions.

Position	Priority	Type of priority	Acronym	Description	Address
	-	-	-	Reserved	0x0000 0000
	-3	fixed	Reset	Reset	0x0000 0004
	-2	fixed	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000 0008
	-1	fixed	HardFault	All class of fault	0x0000 000C
	0	settable	MemManage	Memory management	0x0000 0010
	1	settable	BusFault	Pre-fetch fault, memory access fault	0x0000 0014
	2	settable	UsageFault	Undefined instruction or illegal state	0x0000 0018
	-	-	-	Reserved	0x0000 001C - 0x0000 002B
	3	settable	SVCall	System service call via SWI instruction	0x0000 002C
	4	settable	Debug Monitor	Debug Monitor	0x0000 0030
	-	-	-	Reserved	0x0000 0034
	5	settable	PendSV	Pendable request for system service	0x0000 0038
	6	settable	SysTick	System tick timer	0x0000 003C
0	7	settable	WWDG	Window Watchdog interrupt	0x0000 0040
1	8	settable	PVD	PVD through EXTI line detection interrupt	0x0000 0044
2	9	settable	TAMP_STAMP	Tamper and TimeStamp interrupts through the EXTI line	0x0000 0048
3	10	settable	RTC_WKUP	RTC Wakeup interrupt through the EXTI line	0x0000 004C

***Figure 7.2.3.1.-** Part of STM32F4 Vector Table [64].*

Each Exception handler can be classified by the following properties [65]:

- <u>Position</u>: These positions are with respect to NVIC (Nested vectored interrupt controller) which will further be explained.
- <u>Priority</u>: This column gives the priority to exceptions and interrupts.
- <u>Type of Priority</u>: This column tells us whether the priority of exceptions can be changed or not.
- <u>Address</u>: This column tells where exactly in the processor memory map you must keep the corresponding exception handler. A handler is just a C function that takes care of that exception.

7.2.4. NVIC (Nested Vectored Interrupt Controller)

Interrupts are a common feature supported by almost all microcontrollers. They are typically generated by hardware (peripherals or external input pins).

Figure 7.2.4.1 represents the Nested Vectored Interrupt Controller (NVIC) structure for interrupt handling. NVIC facilitates low-latency exceptions and interrupts handling, controls power management and implements System Control Registers.

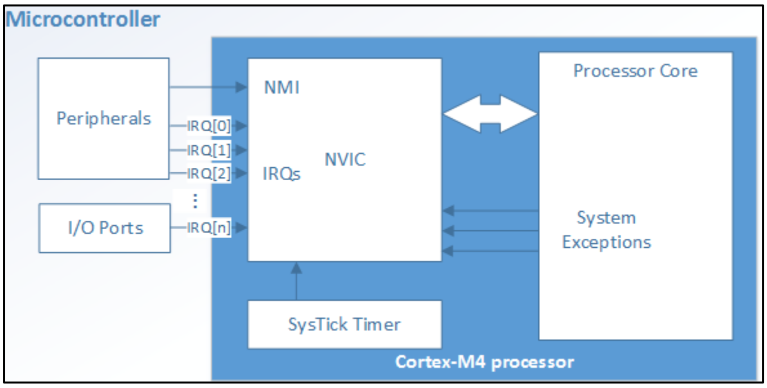


Figure 7.2.4.1.*- STM32F4 NVIC Structure [64].*

7.2.5. MCU Interrupt Design

We can observe that not all interrupts go directly to NVIC. Some peripherals deliver their interrupt to NVIC over the EXTI Lines, and some peripherals deliver their interrupts directly to NVIC (Figure 7.2.5.1) [64].

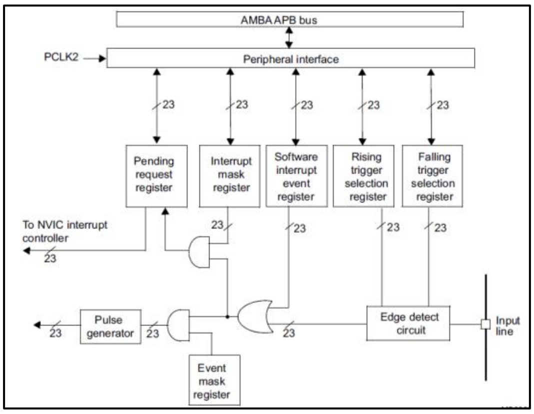


Figure 7.2.5.1.*- STM32F4 External interrupt/event controller block diagram [64].*

STM MCU has an engine called EXTI (External interrupt/event Controller)**.** Moreover, the engine is also connected to the NVIC Interrupt Controller. In EXTI it is important to know about the Pending Request Register [64]. This Register tells us on which EXTI line an interrupt is pending. When an interrupt event occurs, the corresponding bit in this register goes high [64]. Once the interrupt event is finished, it is the programmer's responsibility to clear this bit.

7.2.6. DMA

DMA (Direct Memory Access) is a hardware-controlled data transfer technique. Direct Memory Access can be abbreviated to DMA, which is a feature of computer systems [66]. It allows input/output (I/O) devices to access the main system memory (random-access memory), independent of the central processing unit (CPU), which speeds up memory operations. Direct Memory Access is useful whenever the CPU cannot keep up with the data transfer rate, or when the CPU needs to perform work while waiting for relatively slow I/O data transfers [66].

7.2.7. USART/UART

UART supports only asynchronous mode and USART supports both synchronous and asynchronous modes. In Asynchronous transmission, a clock is not sent along with the data, instead synchronous bits like start and stop bits are used whereas in synchronous transmission a separate clock is sent along with the data hence start and stop bits are not required [64].

7.2.7.1. UART Pins

The UART in duplex mode communication requires at least 2 pins, TX and RX**.** More control lines (RTS and CTS) are used to manage the communication. (Figure 7.2.7.1.1)

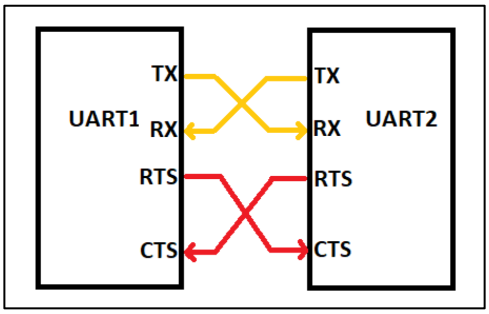


Figure 7.2.7.1.1.- *UART Interface structure. Own source.*

7.2.8. SPI

SPI is an interface bus commonly used to send data between the Microcontroller and small peripherals such as sensors, memory chips, etc. It uses separate clock and data lines along with a select line to choose the device it wants to communicate. The side that generates the clock is called the master and another side is called the slave [65]. There is always one master (i.e. MCU) and multiple slaves. SPI is a single master protocol; this means that only one master initiates communication with multiple slaves. A slave cannot be able to change its role from slave to master. SPI is a protocol of 4 lines (Figure 7 2.8.1), they are [65]:

- <u>SCLK (Clock Signal):</u> The Clock is sent from master to slave through this line, all the SPI signals are synchronous to this clock.

- Slave Select (SS): This line is used to select the slave device. Whenever the master wants to communicate to slaves, it pulls the corresponding slave select line to low.
- MOSI (Master Out Slave In): Master sends data to the slave over the MOSI line.
- MISO (Master in Slave out): Slave sends the data to master over the MISO line

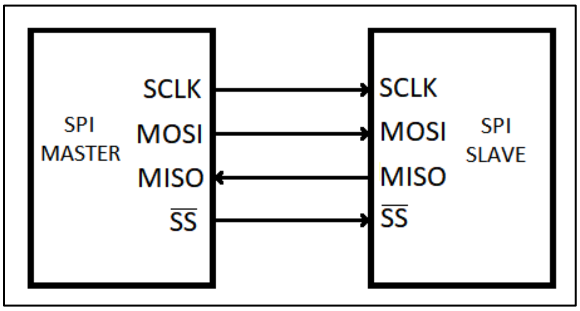


Figure 7.2.8.1.- *SPI interface structure. Own source.*

7.2.9. GPIO

GPIO stands for General Purpose Input/Output. The STM32F407VGT6 Microcontroller supports 9 GPIO ports (GPIO A to GPIOI). Each GPIO port is a group of 16 GPIO pins and has its own set of configuration registers. The MCU supports a total of 114 GPIO pins, but in the STM32F4 Discovery board there are only five ports available (GPIOA, GPIOB, GPIOC, GLIPIOD and GPIOE) with a total of 80 pins (Figure 7.2.9.1) [65].

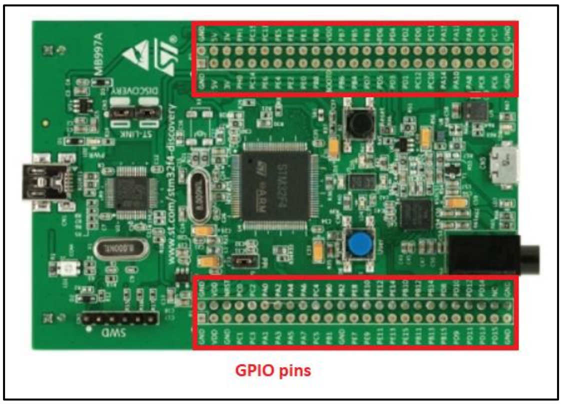


Figure 7.2.9.1.- *STM32F4 Discovery Board GPIO pins. Own source.*

7.2.10. Cortex Microcontroller Software Interface Standard (CMSIS)

The ARM microcontrollers are complex and writing macros to all peripheral memory registers, bitmasks and interrupt vectors is work that would have to be repeated by every developer. To prevent this ARM has a portable and vendor-independent hardware abstraction layer called CMSIS [67]. CMSIS works on all Cortex-based microcontrollers and provides a small, but important abstraction from hardware. This makes it easier to use the Cortex microcontroller, saves development time and increases standardization [67].

7.2.11. STM32 Libraries

For further implementation of the microcontroller code, different libraries are needed:

- CMSIS Digital Signal Processing Library
  CMSIS also contains a digital signal processing (DSP) library. This library adds support for complex numbers, PID regulation, filtering, matrix, statistics and Fast Fourier Transform (FFT). The code is written as a mix of C and assembly code optimized for the ARM Cortex instruction set [67]. This library will allow the device to perform the FFT required when obtaining the EMG frequency domain parameters (section 8.3.7.3).

- HAL Driver layer: It provides a simple, generic multi-instance set of APIs (application programming interfaces) to interact with the upper layer (application, libraries and stacks) [68]. The HAL driver APIs are split into two categories: generic APIs, which provide common and generic functions for all the STM32 series and APIs, which include specific and customized functions for a given line or part number. The HAL drivers include a complete set of ready-to-use APIs that simplify the user application implementation [68]. For example, the communication peripherals contain APIs to initialize and configure the peripheral, manage data transfers in polling mode, handle interrupts or DMA, and manage communication errors [68].

8. System Design: Software

8.1. Overview

8.1.1. Modular code

The microcontroller code has been developed with modularity in mind. Modular programming is a software design technique that applies strict guidelines to how code is organized. Modular code is separated into many files where each file is a part of the whole application with well-defined interfaces. This often reduces the number of dependencies for each file, makes it easier to navigate in the code, and is a good foundation for further development [58]. To implement this type of firmware structure, each module has been implemented with its corresponding source and header files (Annex A). For example, the code related with the LBIA acquisition and processing is written in a source file named "lbia.c", while all the functions and variables required for this code are defined in the header file named "lbia.h".

8.1.2. Used programming software

The code for this medical device is programmed using C language. The software tools utilized are mainly the following.

STM32CubeMX: This graphic tool, from *STMicroelectronics,* is used for the initial configuration of the STM32 microcontroller. It is also used to establish the parameters, the peripherals configuration, and the pin disposition [62]. After that, the code generated by this program is further extended and modified using Arm Keil MDK.

Arm Keil MDK*:* It is the most comprehensive software development environment for ARM and Cortex-M-based microcontrollers. MDK-ARM allows you to program, debug and optimize your code, while taking advantage of the Keil RTX operating system. Additionally, it supports hardware debugging and Flash programming based on ST-LINK2, the in-circuit debugger and programmer used in STM32 Microcontrollers [69].

8.2. System workflow

This section aims to make it easier to understand the implemented code of this thesis medical device (Annex A). In order to do that, the system workflow of each functionality of the medical device is exposed.

8.2.1. Main

When the device is turned on, it first starts the initializing process (section 8.2.2). Once it is done, the following step is to check the battery level (section 8.2.3). If the device battery level is below 15% of the maximum level, a visual warning (red LED) is turned on. In this case, an error flag is activated to inform about the low battery level, then, after 20 seconds, the device is turned off (section 8.2.7). On the other hand, if the battery level is above 15% but below 40%, the device turns on a visual warning (orange LED). After checking the battery level, the external ADC AD7389-4 is initialized and configured to prepare it for future EMG data acquisitions (section 8.2.4). Then, the medical device establishes a connection with an external device (to receive commands and sent the desired data back) (section 8.2.5). If it succeeds, a visual indicator (blue LED) is turned on.

After this first step, the device firmware is organized as a finite-state machine where only 5 states are allowed: Waiting Mode state, LBIA Measurement Mode state, EMG Measurement Mode state, Battery Reading Mode state and Error evaluation Mode state. The Waiting Mode state is the default one and is the state in which the device starts.

In the waiting mode, the device is continuously reading the battery level, turning ON the orange LED or shutting the device down if necessary (as previously explained). Additionally, the device waits one minute to receive a command (sent with the corresponding APP), if it is not received, the device will be shut down (section 8.2.9). The available commands and the corresponding functions, are the following:

- ‘LBIA’: This command sets the medical device to the “LBIA measurement” mode state (section 8.2.6) to proceed with the measurement and acquisition of the bioimpedance.
- ‘EMG’: This command sets the medical device to the “EMG measurement” mode state (section 8.2.7) to proceed with the measurement and acquisition of the electromyogram.
- ‘BAT’: This command sets the device to the “Reading Battery” mode state (section 8.2.3) to assess which is the state of charge of the battery.
- ‘ERROR’: This command sets the device to the “Error Evaluation” mode state where all the error flags from the BLE communication, AD7389-4, EMG and LBIA measurements are gathered to inform the user of the performing status (section 8.2.8). When this list of errors is sent to the external device, all the flags are cleared.

In each of the previous cases, when the desired parameters are obtained, a visual indicator (blinking green LED) is activated for 5 seconds and after that, the data is sent to the external device through the medical device APP. In addition, medical data (LBIA or EMG parameters) is encrypted prior to being sent (section 8.3.4.6).

This entire process can be seen in Figure 8.2.1.1.

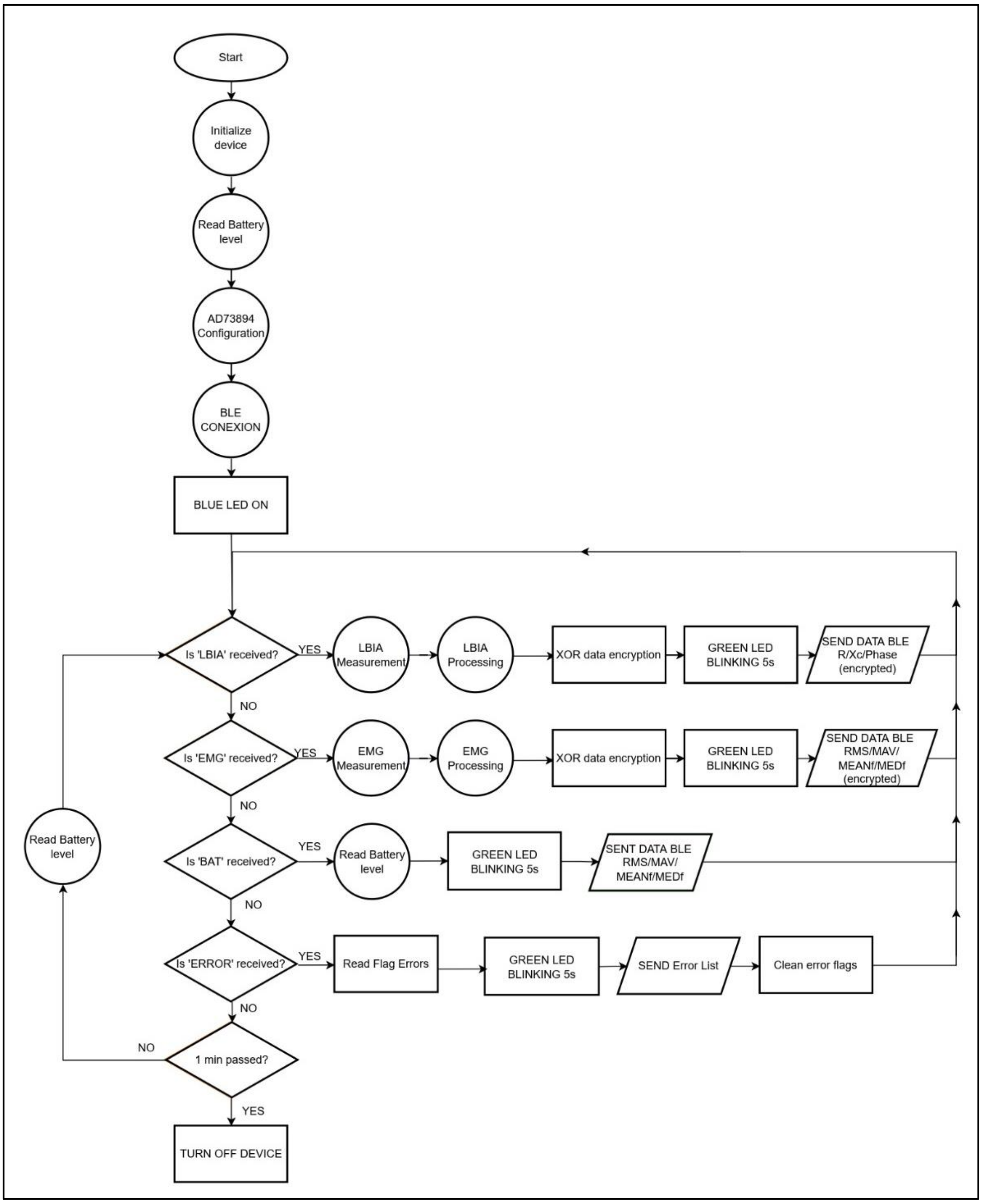


Figure 8.2.1.1.- *Main medical device workflow. Own source.*

8.2.2. Initialize device

When the device is initialized, the first step is to configure the MCU ports, the USART3 used for the BLE module, and the SPI1 used for the AD7389-4 (for the EMG acquisition), the board DAC and the ADC2 (for the LBIA acquisition) (Figure 8.2.2.1). After that, the corresponding timers and interruptions are also set up, followed by the DMA initialization, and charging of the Sine Waveform (that will be used by the DAC). Lastly, the MCU timer is set to 0 and the variables are initialized. This process is represented in Figure 8.2.2.1.

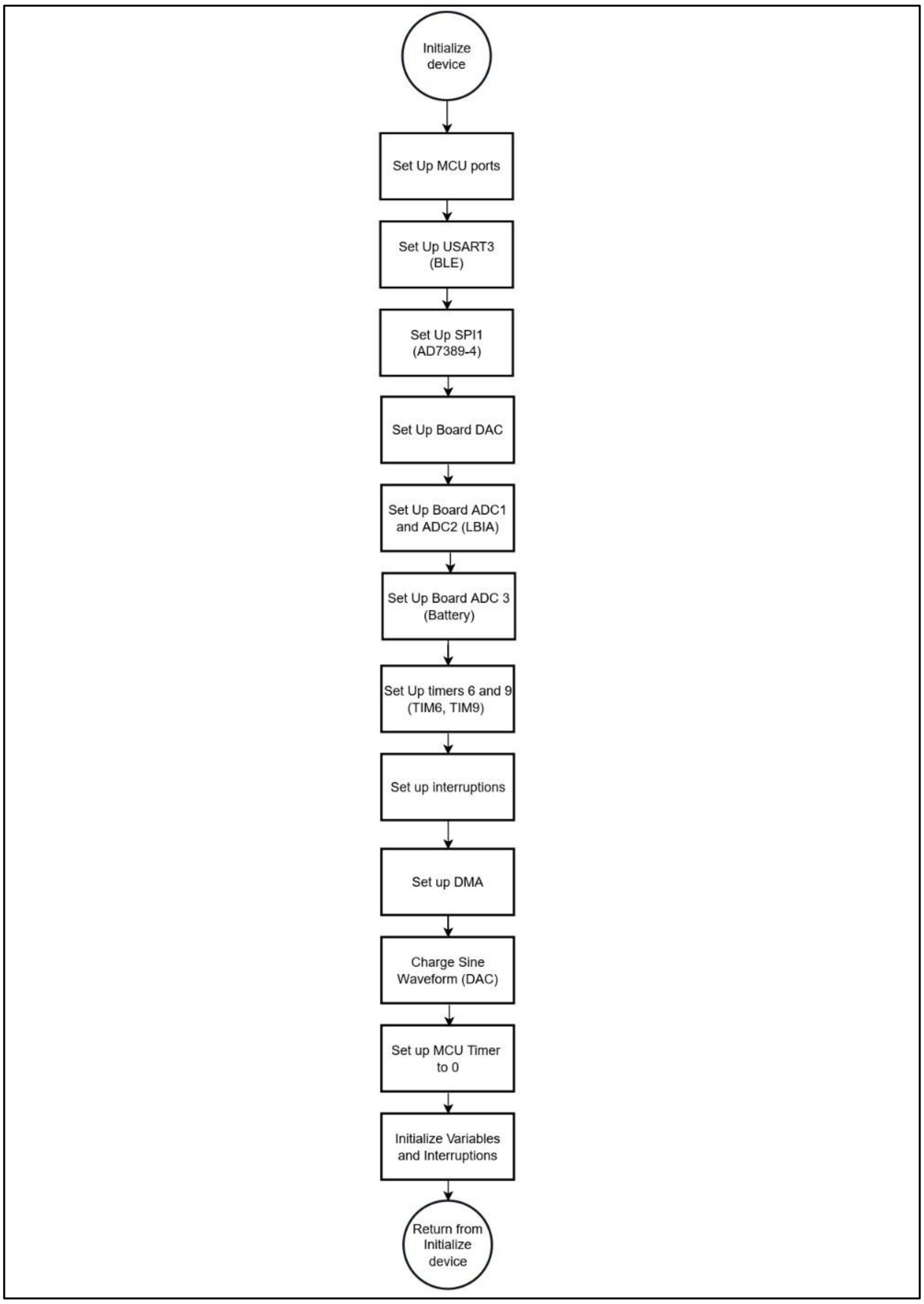


Figure 8.2.2.1.- *Workflow of the Initialize process. Own source.*

8.2.3. Read battery level

In this process, the level of the battery of the device is checked. Channel 0 of the ADC 3 from the STM32F4 board is used to obtain the battery voltage in a single conversion. After that, the level of battery left is estimated. Then different actions take place depending on this value (as seen in section 8.2.1) (Figure 8.2.3.1).

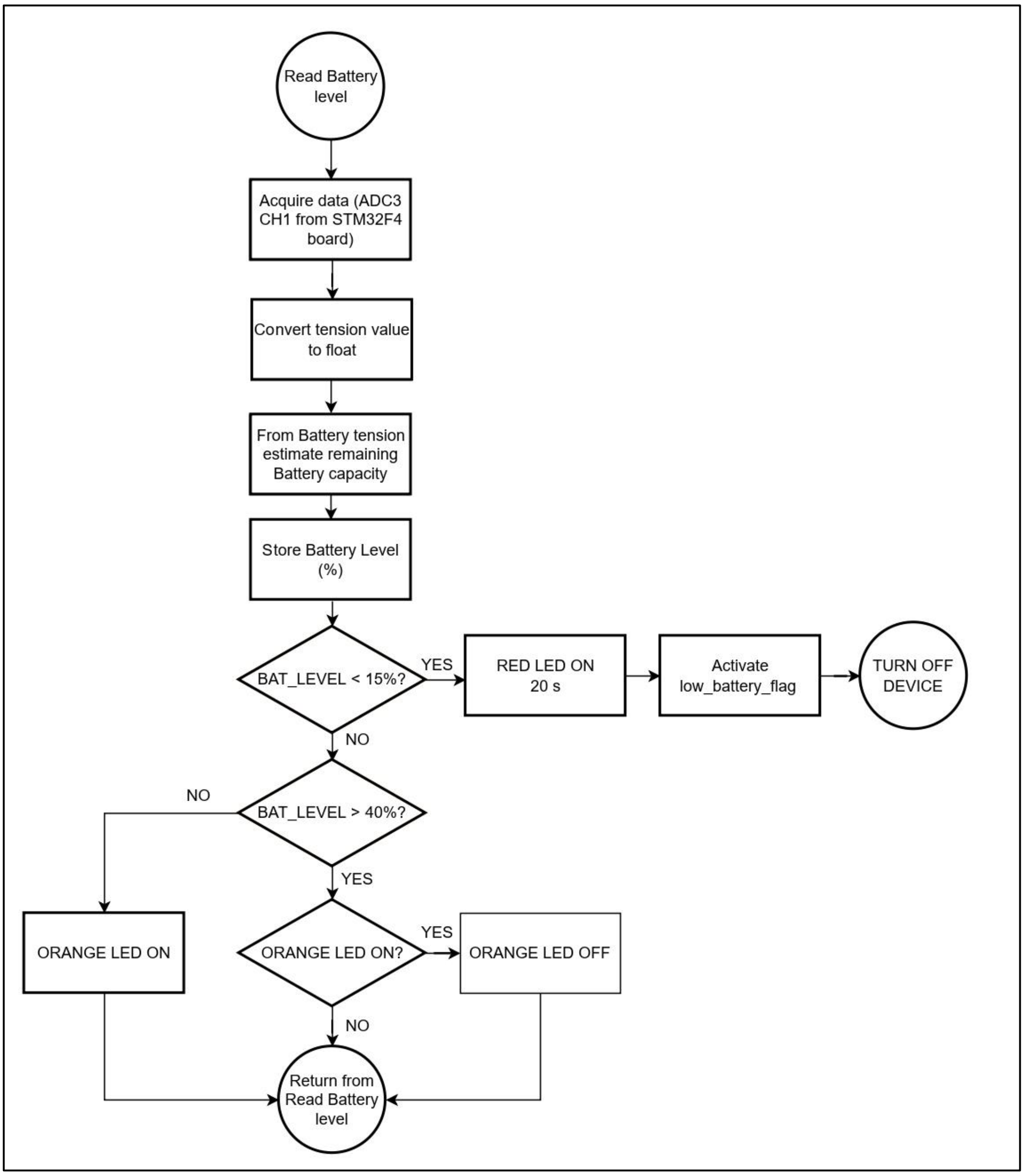


Figure 8.2.3.1.- *Workflow of the Battery reading process. Own source.*

8.2.4. AD7389-4

In this step the AD8739-4 is initialized and configurated by writing to the corresponding registers via SPI. This external ADC has two main configuration registers that need to be written in order to correctly acquire data. First, the ADC is configured *to* normal sampling mode (explained in section 8.3.5). Then the ADC is configured to set the serial data output set (SDOA) to *1-wire* (explained in section 8.3.5). In case the *configuration is not done* correctly, the corresponding error flag is activated along with a visual warning (blinking red LED) for 5 seconds, leading to the device shutting down (section 8.2.9).

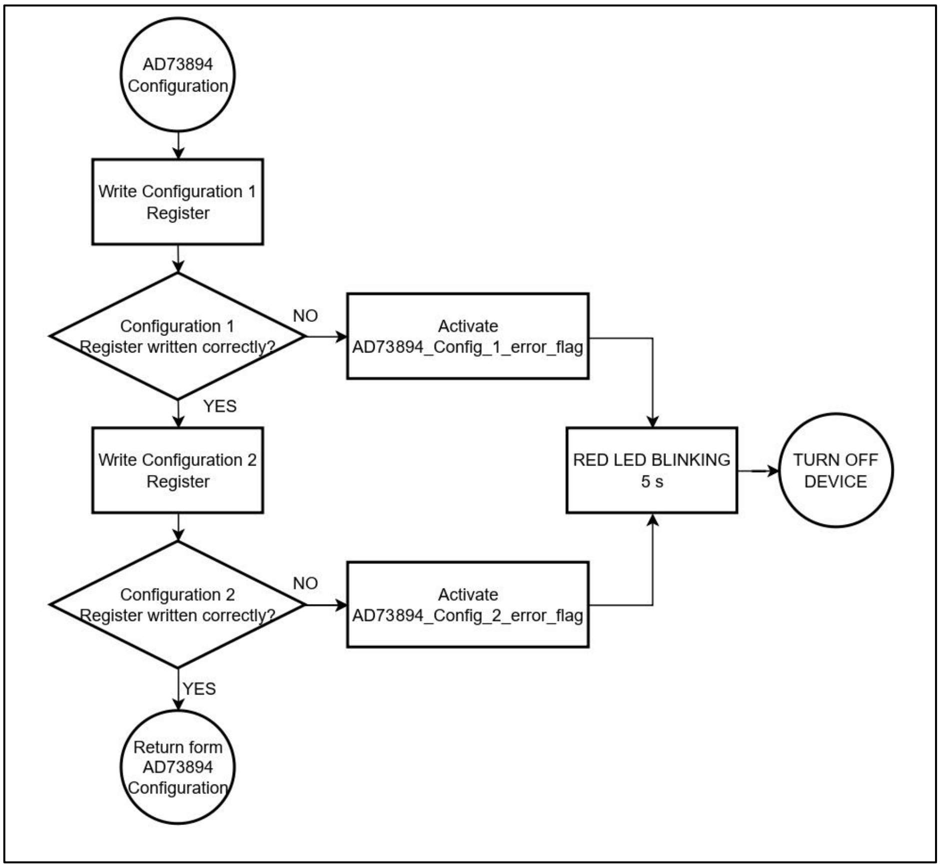


Figure 8.2.4.1.- *Workflow of the AD7389-4 Initialization and configuration process. Own source.*

8.2.5. BLE connection

In this stage, the medical device tries to stablish connection with an external device that is using the corresponding medical device App (Figure 8.2.5.1). To ensure that the connected device is an authorized user (it is using the application), a two-factor authentication protocol is used. The application will need to provide a 6-digit PIN and a 10-character password to establish the connection. To configure and evaluate different parameters of the HM-10, the AT Commands from the device are used [70].

First, it is necessary to check if the communication between the MCU and the BLE module through USART3 is well stablished. In other to do that, the command "AT" is sent to the BLE module, and the command "OK" is expected to be received back. If this step is accomplished, the communication between the HM-10 and the MUC through USART3 has been proved to be successful. After that, the device name is set to *TFG_ARNAU_DIEZ* using the "AT+NAMETFG_ARNAU_DIEZ" command. If the device name has been properly stablished, the command "OK+Set[TFG_ARNAU_DIEZ]" is received.

Then, the 6-digit PIN required to connect with the BLE module is set to *482731* using the "AT+PASS[482731] command. If this PIN has been stablished correctly, the command "Ok+Set[482731]" is received. Following this step, the bond mode of the BLE module is set to *PIN Authentication* using the command "AT+TYPE[2]" (to connect to the medical device, the previously defined PIN will need to be introduced). If the command "OK+Set[2]" is received back, the process has gone correctly.

The following step is to do the "Advertising" process which makes the medical device noticeable to the external devices, allowing their connection. To configure the advertising type to the "Advertising Scan Response, Connectable" option (which will allow the advertising and the posterior connection), the command "AT+ATDY[0]" is sent. In case, "OK+Set:[0]" is received, the advertising type was set correctly. Then, it is necessary to set the Work Mode of the device to "Transmission mode", which will allow the device to be visible to other devices and to stablish communication with them. To do that, the command "AT+MODE[0]" is sent, and the device waits to receive "OK+Set:[0]" to verify that the mode was stablished correctly.

After that, an external device (for instance a smartphone) can connect to the medical device. In this stage, the medical device waits one minute for an external device to connect with it. If this happens, the BLE State pin of the board is set to a high state indicating that a connection has been made. At this stage, the application has 2 seconds to send the correct 10-character password, which in this case is "h325bc092A". If this password is received, the BLE connection function is completed.

However, during each one of the previous steps, an error can occur, and the communication can be lost. If this happens, an error flag will be activated accordingly, a visual warning (red LED blinking) is activated for 5 s and the device is turned off (section 8.2.9) (Figure 8.2.5.1).

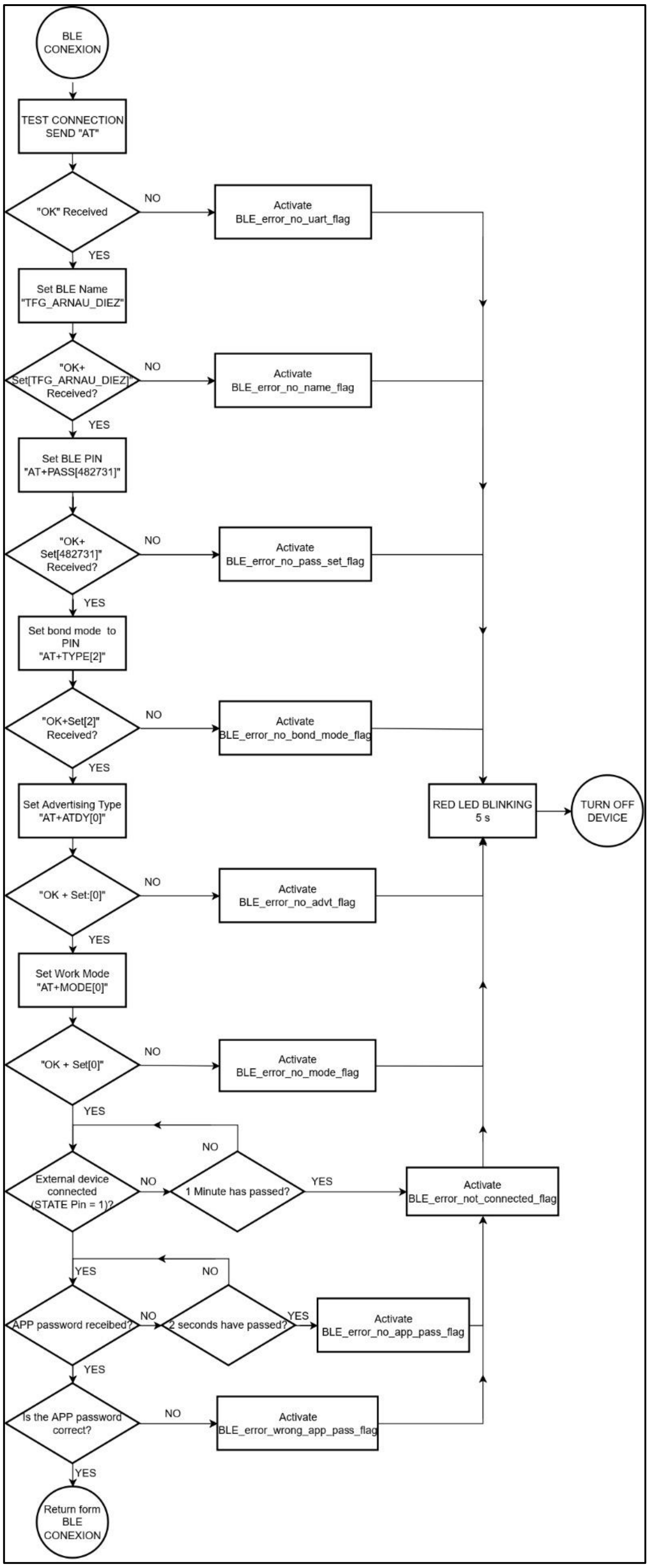


Figure 8.2.5.1.- *Workflow of the BLE connection process. Own source.*

8.2.6. LBIA measurement

When the LBIA measurement mode state is set, it allows the user to perform three types of LBIA measurements. The first one is where the R and Xc components are obtained, along with the corresponding phase angle, the second choice will only obtain the capacitive component, Xc whereas the third one will only obtain the real component, R. These three types of acquisition will start when the corresponding command is received ("LBIAS", "LBIAXcS" or "LBIARS" respectively). It is important to state that whether one command is received or another, will only affect the LBIA Processing process (explained in section 8.2.6.1), while the acquisition will remain the same in each case. If none of these commands is received for one minute, the medical device will be turned off (section 8.2.9). On the other hand, if any of these commands is received, the acquisition starts (figure 8.2.6.1).

First, a visual indicator (green LED) is activated to inform the user that the measurement is being performed. In this stage, the board DAC is activated and a sinusoid signal with a frequency of 50 kHz is generated (which will be converted to current with the use of hardware systems). At the same time, the ADC1 and ADC2 from the MCU acquire data simultaneously with a sampling frequency of 250 kS/s. The ADC1 acquires the voltage, while the ADC2 acquires the injected current, allowing more accurate processing. During this process, different errors can occur. For instance, the samples can be saturated, obtaining then, erroneous results. To prevent this from happening, every sample acquired is checked. If the current or voltage samples are saturated, an error flag and a visual warning (blinking red LED for 5 s) are activated, and the measurement needs to be repeated. This process is done for the two ADCs. If the acquisition has been correctly completed, the DAC is stopped, and the green LED is now turned off. After that, the LBIA processing process is performed (Figure 8.2.6.1).

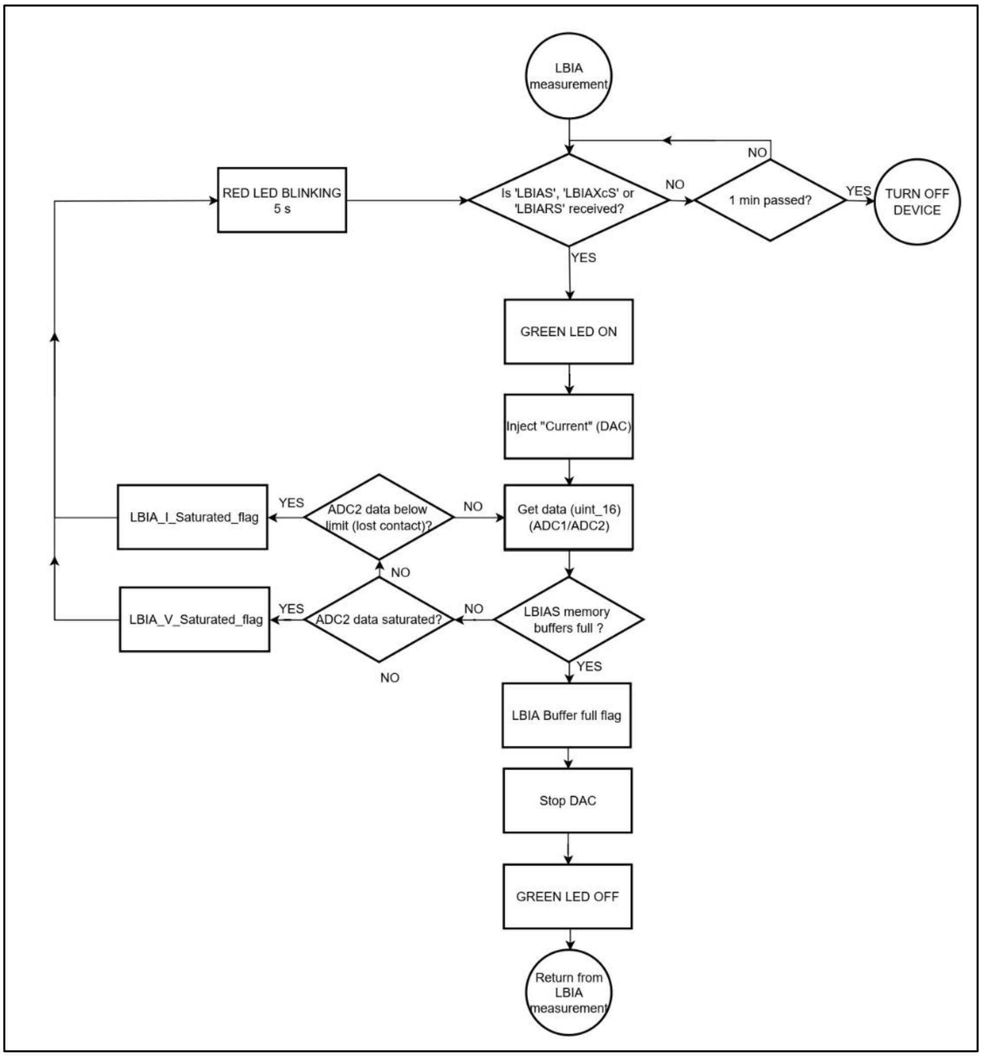


Figure 8.2.6.1.- *Workflow of the LBIA acquisition process. Own source.*

8.2.6.1. LBIA processing

First, in order to check if the current electrodes lost contact with the patient's skin, the Intensity peak of the injected current is compared with the expected current Peak value. If this value is below 80% of the expected value, the acquisition is considered erroneous, activating an error flag and a visual warning (blinking red LED for 5s) and the whole acquisition will need to be repeated (Figure 8.2.6.1.1).

In case the acquisition was successful, the obtained voltage will be multiplied by a sine signal, and a sine signal with a phase of 90º with the same frequency. After that, the resulting data is filtered using a 4th order FIR Low-pass filter (which will further be explained in section 7.3.6.2) to obtain the DC component. Then, the R and Xc components are obtained and stored by dividing the corresponding DC voltage component by the peak current value obtained before. If the starting command is "LBIAXcS" or "LBIARS", only the capacitive or resistive components are stored respectively. However, if the starting command was "LBIAS", both components are stored. In addition, if that is the case, the bioimpedance phase angle will be also obtained and stored. Since this angle cannot be negative (there is no inductive component) and should be below 20º (which is above the maximum phase angle accepted), the measure is considered erroneous if it is not inside this interval. If this happens, the device proceeds to activate an error flag and a visual warning (blinking red LED for 5 s). In this case, the whole acquisition and processing will need to be repeated. On the other hand, if the phase angle is considered acceptable, the process is finished (Figure 8.2.6.1.1).

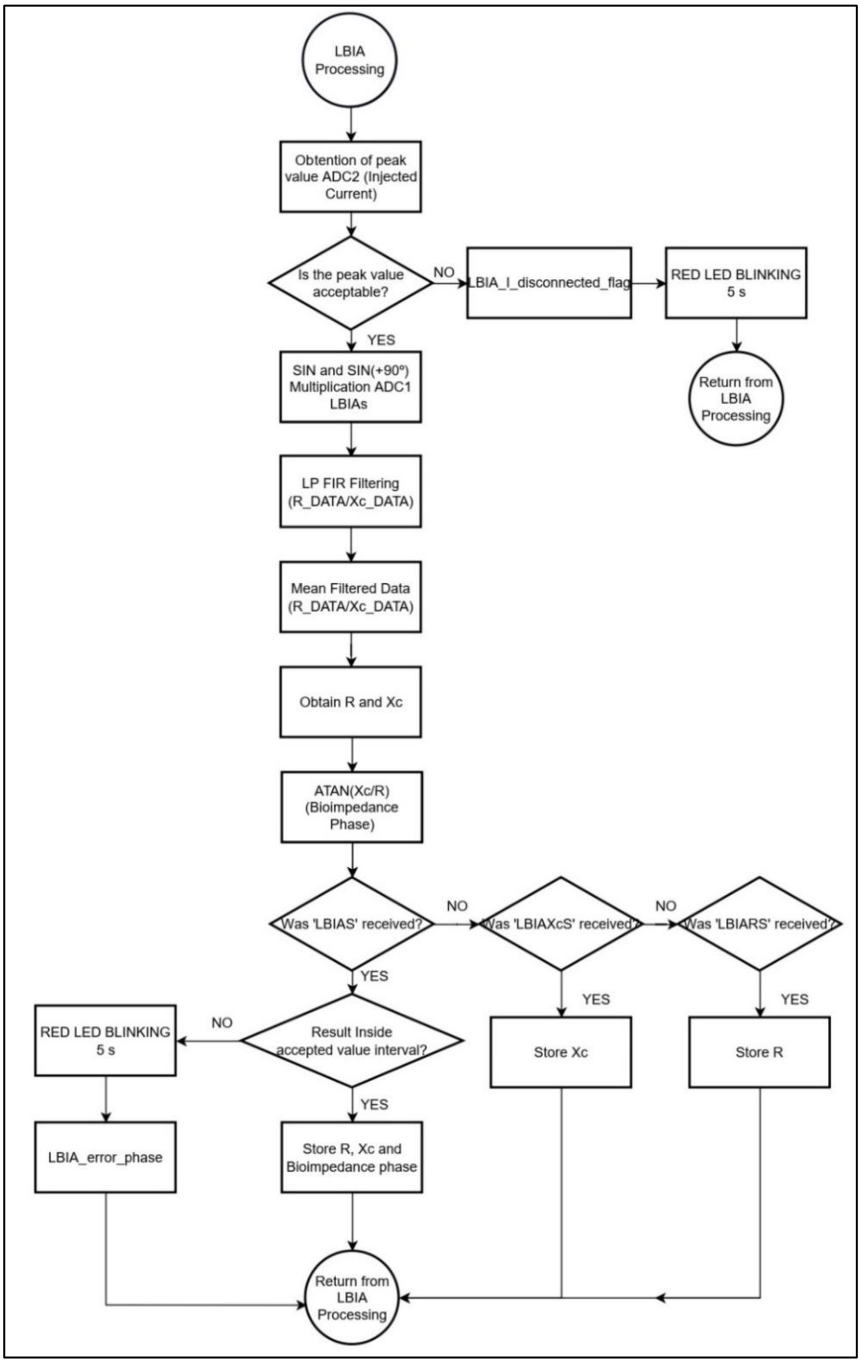


Figure 8.2.6.1.1.- *Workflow of the LBIA Processing process. Own source.*

8.2.7. EMG measurement

When the EMG measurement mode state is selected, it allows the user to perform two consecutive acquisitions of data. As previously explained, it is necessary to normalize the EMG data using MVC. For this reason, prior to the actual EMG measurement, an acquisition is performed to collect the required data for the normalization process (Figure 8.2.7.1).

In this mode, the device waits for a command to start one of the two types of acquisition related to the EMG. If this command is not received in one minute, the device is turned off (section 8.2.9). The possible commands are the following:

- ‘MVCS’: When this command is received, the data acquisition process begins. First, a visual indicator (green LED) is activated to inform the user that the measurement is being performed. In this stage, the device acquires data from 3 ADCs (ADC1, ADC2 and ADC3) with a sampling frequency of 5 kS/s for 6 seconds. During this process, different measuring errors can occur (for instance, the loss of contact between the electrode and the patient), leading to the obtention of saturated data. To prevent this from happening, every sample acquired is checked. If it is saturated, the counter of saturated samples in a row increases, whereas if it is not, the counter is restarted. If this counter reaches 2000 samples (0.4 seconds of acquisition), an error flag and a visual warning (blinking red LED for 5 s) are activated, and the measurement needs to be repeated. If there weren’t any problems, and the respective memory buffers are full, the green LED is turned off (to indicate the measuring is done) (Figure 8.2.7.1). After that, a mean operation is performed on the acquired data, obtaining and storing the corresponding MVC values for each channel. Finally, an auxiliary indicator is activated to inform the device that the MVC measurement has been performed (Figure 8.2.7.1).
  If necessary, this measure can be repeated if the same command is sent again.

  ‘EMGS’: If the MVC is already performed (the auxiliary indicator is checked), this command starts the EMG acquisition process. This process is identical to the one explained before, but in this case the acquisition lasts 10 seconds. If the measurement was performed correctly, the data is then processed (section 8.2.7.1). If necessary, this measure can be repeated if the same command is sent again (Figure 8.2.7.1).

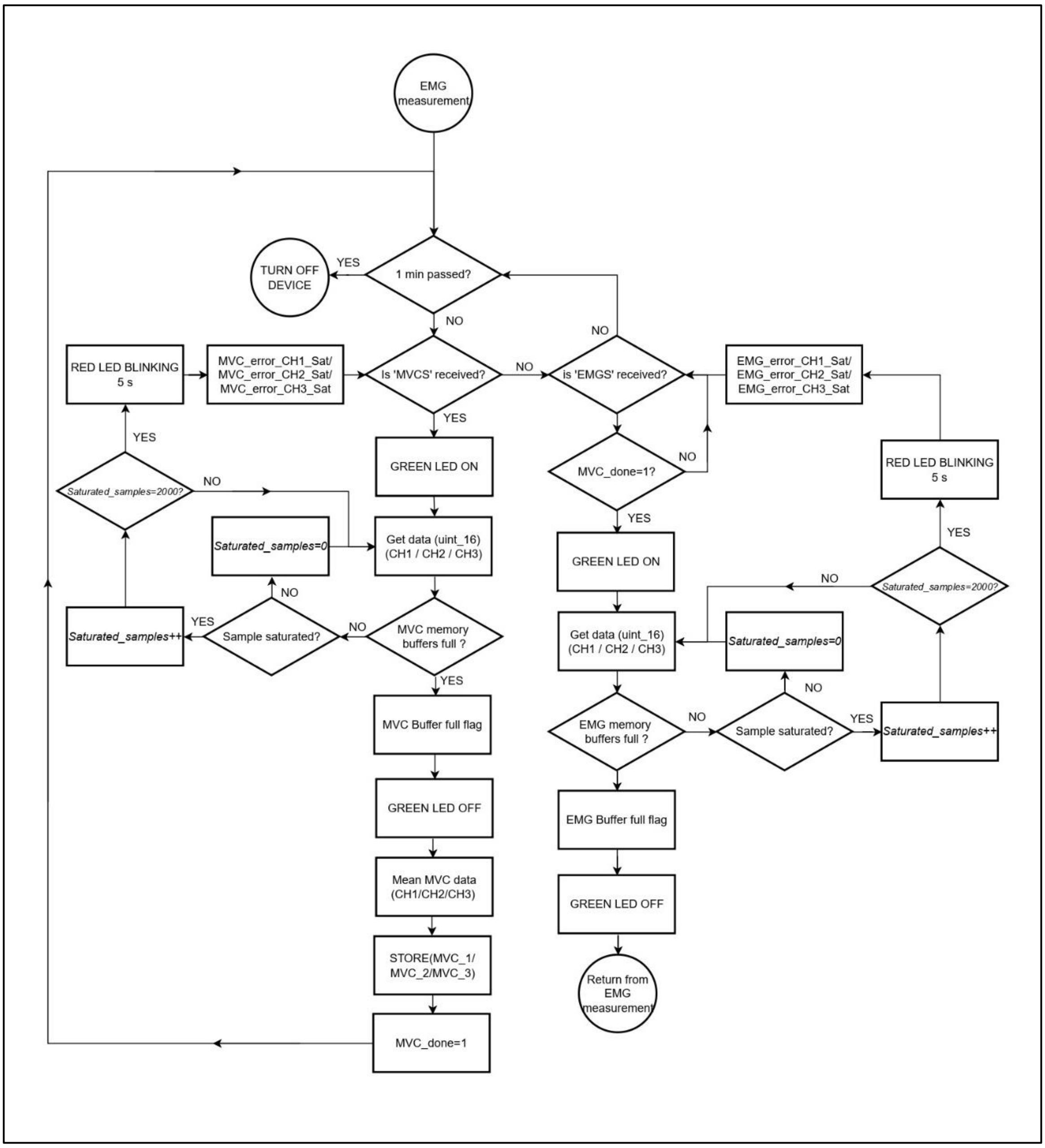


Figure 8.2.7.1.- *Workflow of the EMG acquisition process. Own source.*

8.2.7.1. EMG processing

In this step, the EMG data from each one of the three channels is already obtained and processed. First, the standard amplitude parameters are obtained, followed by the frequency domain ones. The first step to obtain the amplitude parameters is the performing of the RMS envelope, followed by the normalization procedure (involving the previously obtained MVC value). After that, the MAV and the RMS calculations are done, storing the resulting data (Figure 8.2.7.1.1).

Following this procedure, the frequency domain parameters are obtained. An FFT is performed on the original EMG data from each channel. After that, the mean and median frequencies of each channel are stored (Figure 8.2.7.1.1).

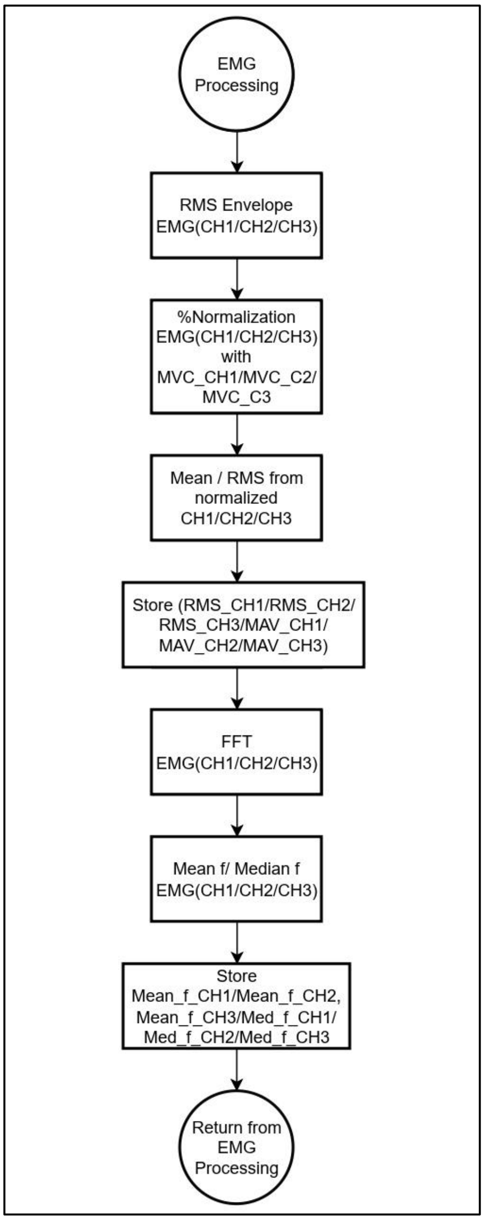


Figure 8.2.7.1.1.- *Workflow of the EMG Processing process. Own source.*

8.2.8. Error handling

As previously explained, when the device mode is set to error handling, all the errors that have been produced during the functioning of the device are gathered and sent to the external device for debugging purposes. All the possible errors that this device can detect and communicate to the external device are exposed in Table 8.2.8.1.

Table 8.2.8.1.- *Error flags of the medical device used in the Error Handling Mode state.*

Error Type	Flag Name	Message received (BLE)	Error description
Battery	Low_battery_flag	“LOW BATTERY”	The battery level is below 15% (The device has been shut down)
AD7389-4 Configuration	AD73894_Config_1_error_flag	“AD73984 CONF1. ERROR”	The Configuration 1 register from the AD7394 ADC was not correctly written.
	AD73894_Config_1_error_flag	“AD73984 CONF2. ERROR”	The Configuration 2 register from the AD7394 ADC was not correctly written.
BLE	BLE_error_no_uart_flag	"BLE CON. ERROR NO UART"	The “OK” was not received indicating the USART3 communication between the BLE module and the board is not stablished correctly.
	BLE_error_no_name_flag	"BLE CON. ERROR NO NAME"	The “OK+Set[TFG_ARNAU_DIEZ]” was not received, indicating that the Name setting operation of the device did not succeed.
	BLE_error_no_pass_set_flag	“BLE NOT PIN. ERROR"	The “OK+Set[482731]” was not received, indicating that the authentication PIN was not correctly set.
	BLE_error_no_bond_mode_flag	"BLE NOT BOND. ERROR"	The “OK+Set[2]” was not received, indicating that the bond mode setting operation of the device did not succeed.
	BLE_error_no_advt_flag	"BLE CON. ERROR NO ADT"	The “OK+Set[0]” was not received, indicating that the advertising type was not set adequately.
	BLE_error_no_mode_flag	"BLE CON. ERROR NO MODE"	The “OK+Set[0]” was not received, indicating that the working mode was not set adequately. Consequently, the discover operation, was not performed.
	BLE_error_not_connected_flag	"BLE NOT CON. ERROR"	The BLE State Pin was not set to s high state for 1 minute, indicating that no external device was connected to the medical device.
	BLE_error_no_app_pass_flag	"BLE CON. ERROR NO PASS"	The required password was not sent from the connected external device.
	BLE_error_wrong_app_pass_flag	"BLE CON. ERROR WRONG PASS"	The password sent by the external device was not correct.

LBIA	LBIA_I_Saturated_flag	"LBIA I SAT"	The data acquired from the injected current during the LBIA measurement is saturated.
	LBIA_V_Saturated_flag	"LBIA V SAT"	The data acquired of the voltage during the LBIA measurement is saturated.
	LBIA_I_disconnected_flag	"LBIA I NO CONTACT"	The injected peak current values are below the expected levels, indicating that the electrodes didn’t have good contact with the patient’s skin.
	LBIA_error_phase_angle_I	"Z PHASE WRONG"	The obtained Bioimpedance Phase Angle is not between the expected values.
MVC	MVC_error_CH1_Sat	"MVC CH1 SAT."	The data acquired from channel 1 during the MVC measurement is saturated.
	MVC_error_CH2_Sat	"MVC CH2 SAT."	The data acquired from channel 2 during the MVC measurement is saturated.
	MVC_error_CH3_Sat	"MVC CH3 SAT."	The data acquired from the channel 3 during the MVC measurement is saturated
EMG	EMG_error_CH1_Sat	"EMG CH1 SAT."	The data acquired from channel 1 during the EMG measurement is saturated.
	EMG_error_CH2_Sat	"EMG CH2 SAT."	The data acquired from channel 2 during the EMG measurement is saturated.
	EMG_error_CH3_Sat	"EMG CH3 SAT."	The data acquired from channel 3 during the EMG measurement is saturated.

8.2.9. Device Off

In the previous sections, there have been described different situations where the device is turned off. In this case, the device will stablish all the used peripherals to sleep mode (which heavily reduces the power consumption) and then, the MCU will also be put to sleep mode (Figure 8.2.9.1). To "wake up" from this state, the board START push button will need to be pressed.

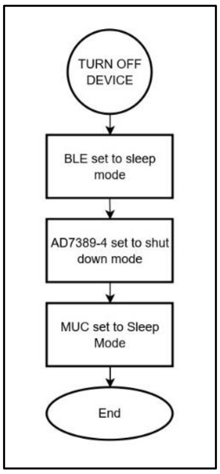


Figure 8.2.9.1.- *Workflow of the Device Turning OFF process. Own source.*

8.3. Firmware design justification

After exposing the actual workflow of all the functions and processes involved in the firmware design of this medical device, it is necessary to explain all the calculations, configurations and code design specifications that make the correct functioning of the device possible.

8.3.1. Clock configuration

As previously discussed in section 7.2.2, there are different possibilities when choosing the system clock (SYSCLK) configuration of the board. The clock chosen to be used during the main processes of the device is the HIS of 16 MHz. To decide whether boost this frequency using the PLL or not, the STM32CubeMX is used to simulate the consumption it would entail (considering all the board peripherals used). The results of this simulation can be seen in Table 8.3.1.1.

As exposed in Table 8.3.1.1, the step consumption is almost 5 times bigger when using PLL. Therefore, in this thesis, the PLL is not used.

Table 8.3.1.1- *Results of consumption simulation developed in STM32CubeMX.*

	PLL HIS	HIS
Step consumption (mA)	49.43	10.15
Consumption without peripherals (mA)	40.00	5.00
Peripherals consumption (mA)	9.43	5.15

8.3.2. Command Protocol

As can be seen in section 8.2, to control the device, different commands will be sent to it via BLE. The data received by the BLE is a string called *RX_DATO* and based on the "word" received, one command or other will is implemented. The commands and their respective effect on the device are exposed in Table 8.3.2.1.

Table 8.3.2.1- *Command Protocol of the medical device.*

RX_DATO	Command
"LBIA"	The device mode is set to LBIA Measurement.
"LBIAS"	If the device mode is *LBIA Measurement*, start the LBIA acquisition and the extraction of R, Xc and Phase angle bioimpedance parameters. After that, it sends them via BLE to the external device
"LBIAXcS"	If the device mode is *LBIA Measurement*, start the LBIA acquisition and the extraction of the Xc bioimpedance parameter. After that, it sends the value via BLE to the external device
"LBIARS"	If the device mode is *LBIA Measurement*, start the LBIA acquisition and the extraction of the R bioimpedance parameter. After that, it sends the value via BLE to the external device
"EMG"	The device mode is set to EMG Measurement.
"MVCS"	If the device mode is *EMG Measurement*, start the MVC acquisition and calculation of the MVC value.
"EMGS"	If the device mode is *EMG Measurement*, and the *MVC process* has already been done, start the EMG acquisition and the extraction of the Standard Amplitude and Frequency domain parameters. After that, it sends them via BLE to the external device.
"BAT"	It acquires the battery voltage and estimates the Battery level left. After that, it sends the value via BLE to the external device
"ERROR"	It sends a list of the errors to the external device via BLE.

8.3.3. Battery level estimation

As explained in section 8.2.3, the battery level left is estimated from the battery voltage level. In this thesis the battery chosen to be implemented in the medical device is the *1/LPP 503562 S* from VARTA AG [71]. The approach followed to read the battery level is to use channel 0 of the ADC3 from the STM32F4 board to obtain the battery voltage with a single value acquisition and then estimate the actual capacity.

The discharge curve that exposes the relationship between the battery voltage and the discharge capacity can be seen in Figure 8.3.3.1.

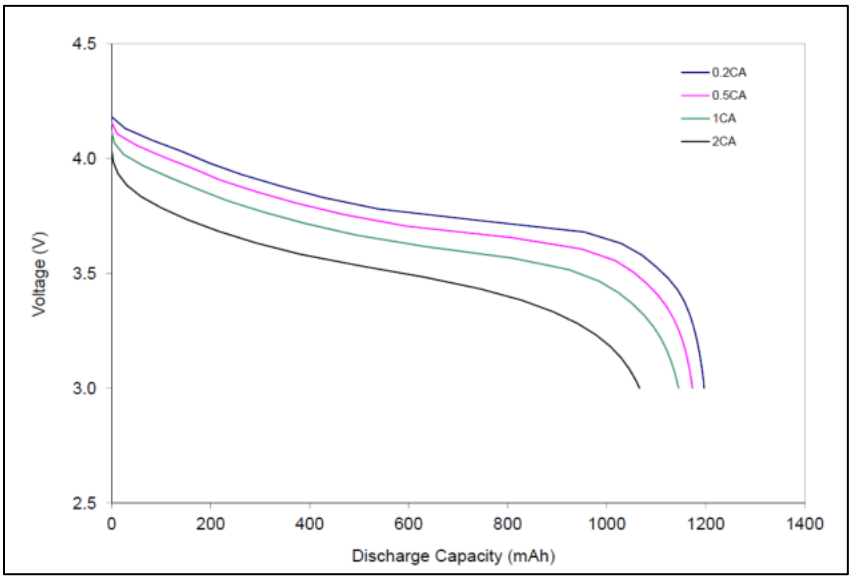


Figure 8.3.3.1.- *Discharge curves with C rates as a parameter. [71].*

In the previous figure, the different curves state different rated capacities, however, in this thesis, a full consumption study is not available to correctly select the respective discharge curve (since it would depend on all the additional hardware components). For this reason, the curve that is followed to estimate the battery level is the 0.2C (blue color line) since it represents the discharge at 20% of the battery capacity [71].

To obtain a better approximation of the curve, the graph is divided into 6 different zones, where a linear regression is performed in each one of them (Figure 8.3.3.2).

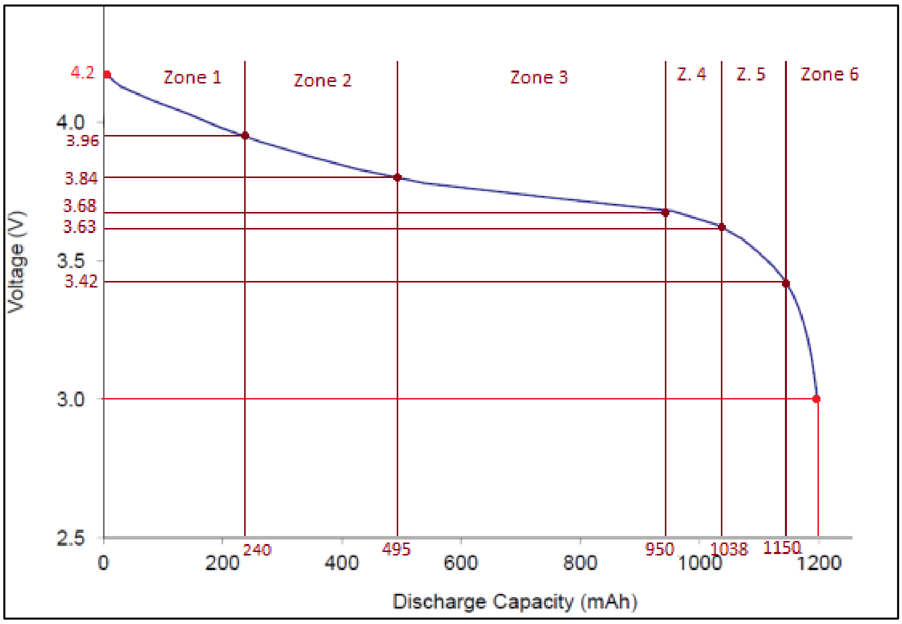


Figure 8.3.3.2.- *Discharge curve with different delimited zones. Own source.*

The equations that relate the voltage (V) and the discharge capacity (D.C) for each zone are exposed in the following table.

Table 8.3.3.1.- *Corresponding linear equations for each zone of the battery curves.*

Zone	Voltage Interval (V)	Linear equation
1	$4.20 \geq V > 3.96$	$D.C = \frac{-(V-4.20)}{0.00120}$
2	$3.96 \geq V > 3.84$	$D.C = \frac{-(V-3.96)}{0.00047} + 240$
3	$3.84 \geq V > 3.68$	$D.C = \frac{-(V-3.84)}{0.00035} + 495$
4	$3.68 \geq V > 3.63$	$D.C = \frac{-(V-3.68)}{0.00057} + 950$
5	$3.63 \geq V > 3.42$	$D.C = \frac{-(V-3.63)}{0.00190} + 1038$
6	$3.42 \geq V > 3.00$	$D.C = \frac{-(V-3.42)}{0.0084} + 1150$

Since the Vref voltage of the ADC3 is 3.3 V [72], a hardware voltage divider circuit is used to bring down the battery voltage to that level (for instance, if the voltage is 4.2 V, the ADC will only get 3.3 V). Consequently, to obtain the actual voltage value from the uint16_t data obtained with the ADC3, the Equation 29 is used:

$$Battery\ voltage\ (V) = \frac{3.3V}{2^{12}-1} \cdot \frac{4.2V}{3.3V} \qquad \textbf{(Eq. 29)}$$

With the battery voltage value, the discharge capacity is obtained following one of the equations of the previous table (depending on which interval the voltage value fits). For instance, if the value is between 4.2 V and 3.96 V, the first equation will be the one used to obtain the corresponding discharge capacity.

After the discharge capacity value is obtained, the following operation is performed to obtain the percentage of battery left:

$$Battery\ left\ (\%) = \frac{1200 - Disc.Capacity}{1200} \cdot 100 \qquad \textbf{(Eq. 30)}$$

8.3.4. Data communication

As can be seen in section 8.2.5, the device can receive external device data as order commands like "EMG measurement" and send the results and obtained parameters back to this device. This communication is done using the BLE module HM-10. This communication protocol selection Is justified in this section.

8.3.4.1. Bluetooth (BT)

It is a Personal Area Network (PAN) that allows the interconnection of two different devices through a radio frequency link in the 2.4 GHz band, referred to as the frequency band for the use of industrial, scientific, and medical devices (ICM) [73].

Bluetooth Low Energy (BLE)

BLE is an emerging wireless technology for short-range communication. In comparison with previous Bluetooth protocols, BLE has been designed as a low-power solution for control and monitoring applications [74]. Just like the classic Bluetooth protocol, in BLE there is the *Controller* and the *Host*, constituents of the main protocols along with the Host Controller Interface (HCI). The controller consists of the physic layer and link layer which are both commonly integrated into one unique chip [74]. The host is the responsible between both devices and includes the different protocol layers [74].

The main features of BLE are the following:

1. To reduce the power consumption, a BLE device is kept in sleep mode most of the time. When an event occurs, the device wakes, and a short message is transferred to a gateway, PC or smartphone. Consequently, its highest power consumption peak is less than 15 mA

and the average is around 1 µA [75]. These values represent a tenth of the energy consumption of the classical Bluetooth [75].

2. BLE technology uses the same adaptive frequency hopping (AFH) technology as classic Bluetooth technology. This enables BLE to achieve robust transmission in the 'noisy' RF environments found in the home, industrial, and medical applications. To minimize the cost and energy consumption of using AFH, BLE technology has reduced the number of channels to 40 2-MHz wide channels instead of the 79 1-MHz wide channels used with classic Bluetooth technology [75].

3. The BLE modulation offers a range up to 300m with a 10 dBm radio chipset [75].

4. BLE communication is typically based on a master connected to several slaves. A device is either a master or a slave, but never both. The master controls how often the slaves are allowed to communicate, and the slave only communicates by request from the master [75].

8.3.4.2. Wireless Fidelity (Wi-Fi)

Wi-fi is a high-speed wireless connection based on the IEEE 802.11 suite of standards that uses radio frequencies (RF), in concrete 2,4 GHz to 5 GHz [76]. In each Wi-Fi communication, we have a wireless adapter in the selected device which translates the signal to the correspondent radiofrequency and sends it with the help of an antenna and a router (also wireless). After that, the router decodifies the signal and then sends the information to other servers (This process is bidirectional) [76].

Access Point (AP)

An access point is an area with wireless connectivity through this technology. This area or access point creates a local wireless network (WLAN) to which we can connect from other devices. Some of the items required for the configuration of an AP are exposed in continuation [77].

- SSID: Specify the name of the wireless network(s) (for example, WLAN). This is the name that is advertised to other devices.
- Encryption: Specify the security encryption in the communications (for instance WPA-2 Enterprise (Preferred), or WPA-Enterprise).
- Wireless AP IP address (static): Configure a unique static IP.
- Subnet mask: Configure this to match the subnet mask settings of the LLAN to which the wireless AP is connected.
- DNS: Some wireless APs can be configured with a DNS name. The DNS service on the network can resolve DNS names to an IP address.

8.3.4.3. Selected Wireless Communication system

Bluetooth devices have much less power consumption than Wi-fi communication systems. This fact is due to the higher reach that Wi-fi communication provides, being 10 times larger than when using BLE. Consequently, Wi-Fi requires ten times more power even performing the same tasks, consuming about 500µW for ten messages per day, while BLE consumes only 50µW [77]. In addition, BLEs are less costly, self-sufficient and can run on a single battery for years, depending on usage with almost no configuration required while Wi-Fi needs a wide range of settings prior to the start of the communication [77].

In this case, the acquisition of the medical data is performed in the same place as the evaluation of the parameters, consequently, there is not a need for a high signal reach. Accordingly, the communications between the medical device and the external device to interact with (Smartphone), are implemented with BLE for the previously stated advantages.

To find the suitable BLE module the following criteria were used: Low Power consumption, UART Interface, economical cost, BLE Version, dimensions, and useful programming resources to ease its implementations. According to these requisites, different modules were found:

Table 8.3.4.3.1.- *Summary table of requisites [78, 79, 80, 81].*

LE Module	HM-10	Nrf51822	Cypress PSoC 4
Manufacturer	Jinan Huamao Technology Co	Nordic Semiconductor	Cypress
Average Power Consumption (Active)	0.4 – 1.5 mA	1.0 mA	16.4 mA
UART Interface	Yes	Yes	Yes
Price	3,95€ / Unit	4,26 € / Unit	6,41 €/unit
BLE Version	Bluetooth 4.0 / Ibeacon support	Bluetooth 4.1	Bluetooth 4.2
Dimensions	3,04 x 1,52 x 0,25 cm	1,85 x 0,92 x 0,20 cm	1,54 x 0,95 x 0,02 cm
Useful Literature with STM32F4 Board	Yes	No	No

As can be seen in the previous table, *Cypress PSoc4* is the most power-consuming and expensive option, and it doesn't have STM32F4 communication-related bibliography, for these three reasons it has not been considered. Between the HM-10 and Nrf51822, the Power consumption is similar. However, the Nrf51822 is smaller which is something to consider when designing a portable device. Moreover, the BLE version of this device is more up to date that the case of the HM-10. On the other hand, HM-10 provides *ibeacon* support which will allow Apple phones to interact with the device, is less expensive than its alternative and has well-documented bibliography about the implementation of this module in the discovery-F4 board [73]. For these reasons, the HM-10 module is the selected option.

8.3.4.4. BLE communication Configuration

As previously exposed, the main purpose of the BLE communication is to receive commands and send data to an external device. The first step to enable this communication is to set the corresponding pins for USART3 communication (TX and RX). The pin PB10 is used for TX and the pin PB11 for RX. Next, the baud rate must be set to 9600 Bits/s as it is the baud rate of the HM-10 module (as previously stated, both baud rates must be the same) [70]. Lastly, the DMA is activated for the USART3 in order to be able to receive commands of more than one character (for instance, "LBIA").

Additionally, the HM-10 BLE module disposes of a State *Pin* [70] whose output is set to HIGH when an external device is connected, otherwise is in LOW state. In order to implement this pin to the STM32F4 board, a GPIO Pin (pin PE15) is set as GPIO_Input and renamed *BLE_State.*

8.3.4.5. BLE module initialization

As seen in the system workflow (section 8.2.5), the HM-10 module must be set up prior to the communication starts. As previously seen, the HM-10 parameters can be configurated with the AT COMMANDS via UART before the actual connection with the external device [70].

8.3.4.6. Encrypting data

As can be seen in section 8.2, the data acquired (EMG or LBIA) is encrypted prior to sending it to the external device (section 9.2). Nowadays, numerous advanced encryption algorithms exist, like the Advanced Encryption Standard (AES), Blowfish or Twofish [82]. However, since this version of the STM32F4 board does not support any encryption protocol, and these advanced security algorithms would entail a high computing cost, a custom-made XOR cipher is used in this thesis [83].

The XOR cipher is a type of additive cipher that operates according to these principles:

$$A \oplus 0 = A \qquad \textbf{(Eq. 31)}$$

$$A \oplus A = 0 \qquad \textbf{(Eq. 32)}$$

$$(A \oplus B) \oplus C = A \oplus (B \oplus C) \quad \textbf{(Eq. 33)}$$

$$(B \oplus A) \oplus A = B \oplus 0 = B \quad \textbf{(Eq. 34)}$$

Where $\oplus$ denotes the exclusive disjunction (XOR) operation. With this logic, a string of text can be encrypted by applying the bitwise XOR operator to every character using a given key to decrypt the output, merely reapplying the XOR function with the key will remove the cipher [83].

To make this a secure protocol, the following requirements need to be met: The key is random generated, has the same size or longer than the actual data and is never re-used [83]. To achieve these conditions, two encryption keys are used in this thesis:

- Random key: Key used to encrypt the medical data that will be sent. This key is 23 characters long (since the maximum length of the sent messages is 22). To generate this key, a pseudo-random number generation function Is implemented. This function uses the C library function *rand()* to fill the 23 positions of the random key with one of the following possible characters each time: "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789,.-#'?!". Since one character is chosen randomly each time, there are no restrictions in repetition (one character can be used more than once).

- Personal key: This key will be used to encrypt the random key to inform the user which one it is, before sending the actual data. This key has a length of 24 characters, and it is known by the user. The key used in this thesis for this purpose is "aBhd7w?4Ysn3c#ap2x5zq03c".

Therefore, the steps this protocol follows are:

1. Generate random key.
2. Encrypt random key using the personal key.
3. Send the encrypted random key to the external device.
4. Encrypt the medical data using the randomly generated key.
5. Send the medical data encrypted with the previously generated random key.

It is important to state that, even though using a pseudo-random generator function increases the protocol security, to make it even more secure the random key should be obtained from a truly random source [84].

8.3.5. AD7389-4 initialization

Once the SPI communication between the AD7389-4 and the STM32F4 board is stablished, it is necessary to write to the corresponding configuration registers to stablish the operation mode we want for the ADC. The register's structure for the microchip is exposed in Figure 8.3.5.1.

Reg	Name	Bits	Bit 15 Bit 7	Bit 14 Bit 6	Bit 13 Bit 5	Bit 12 Bit 4	Bit 11 Bit 3	Bit 10 Bit 2	Bit 9 Bit 1	Bit 8 Bit 0	Reset	RW
0x1	Configuration 1	[15:8]	ADDRESSING				RESERVED		OS_MODE	OSR, Bit 2	0x0000	R/W
		[7:0]	OSR, Bits[1:0]		CRC_W	CRC_R	ALERT_EN	RES	RESERVED	PMODE		
0x2	Configuration 2	[15:8]	ADDRESSING				RESERVED		SDO		0x0000	R/W
		[7:0]	RESET									
0x3	Alert indication	[15:8]	ADDRESSING				RESERVED		CRCW_F	SETUP_F	0x0000	R
		[7:0]	AL_D_HIGH	AL_D_LOW	AL_C_HIGH	AL_C_LOW	AL_B_HIGH	AL_B_LOW	AL_A_HIGH	AL_A_LOW		
0x4	Alert low threshold	[15:8]	ADDRESSING				ALERT_LOW, Bits[11:8]				0x0800	R/W
		[7:0]	ALERT_LOW, Bits[7:0]									
0x5	Alert high threshold	[15:8]	ADDRESSING				ALERT_HIGH, Bits[11:8]				0x07FF	R/W
		[7:0]	ALERT_HIGH, Bits[7:0]									

Figure 8.3.5.1 *AD7389-4 Register description. [85]*

As can be seen in the previous figure, the registers are either read/write (R/W) or read only (R). The last two registers (Alert low threshold and Alert high threshold) are used for Alert Mode [85]. The alert functionality is an out-of-range indicator and can be used as an early indicator of an out of bounds conversion result [85]. However, this feature is not needed since error detection algorithms are already implemented in the corresponding EMG acquisition code. For these reasons, the only registers that are needed to be written are the *configuration 1 Register* and the *Configuration 2 Register.*

First, the *Configuration 1 Register* is written. To correctly initialize the AD7389-4, the ADC is set to normal mode (although it would be set to shutdown mode when the entire device is put to sleep to put the ADC to a lower consuming mode (section 8.2.9). In addition, the oversampling, CRC and alerting functions are deactivated.

If the register was correctly written, the next step is the writing of the *configuration 2 register* (as seen in section 8.2.4). After the simultaneous acquisition of the data from the three channels, the data can be sent with a 1-wire, 2-wire or 4-wire implementation. The STM32F4 board does not support a Dual or Quad SPI which would allow the 2-wire or 4-wire options, respectively. If any of those were implemented, significant software and hardware modifications would need to be done [86]. For this reason, the SDO value is set to 1-wire SDO.

If the *Configuration 2 Register* was written correctly, the AD7398-4 is ready for the previously explained EMG acquisitions.

8.3.6. LBIA measurement

8.3.6.1. LBIA Acquisition

As previously explained and shown in section 8.2.6, the LBIA measurement requires two different ADCs (ADC1 and ADC2). In order to correctly perform the LBIA measurement, ten periods of both signals are stored. Considering a signal frequency of 50 kHz and a sample frequency of 250 KS/s, the required length of the two memory buffers is the following:

$$Buffers\ Length = 250\frac{kS}{s} \cdot 10\frac{1}{50\ kHz} = 50\ \text{uint16\_t} \qquad \textbf{(Eq. 35)}$$

On the other hand, all the considerations needed for the acquisition are exposed:

DAC (Generating of the injected current)

First, to generate the required voltage sine signal with a frequency of 50 kHz (which is later converted to a current signal through a hardware system), the board DAC is used. In order to correctly implement this component, channel one of the DAC1 (pin PA4 of the bard) is activated in STM32CubeMX.

After that, it is necessary to activate the DMA with the circular mode in *DMA Settings,* used for the sine waveform generation process. This mode allows us to configure the number of data items to transfer once, and automatically restart the transfer after a Transfer Complete event, which is very convenient to support continuous transfers such as this one [87].

As described in [88], a sine wave pattern needs to be prepared according to the following formula, where n_s is the number of samples and 0xFFF is 4095 (the resolution of the DAC is 12 bits):

$$Y_{SineSignal}(x) = (\sin\,(2\pi \cdot \frac{x}{n_S}) + 1) \cdot \frac{0xFFF + 1}{2} \qquad \textbf{(Eq. 36)}$$

The digital inputs are converted to output voltages between 0 and V_{REF+}. Then, the analog output voltage of the used DAC channel pin is determined as [88]:

$$DAC_{Output} = V_{REF+} \cdot \frac{DOR(DAC\ Output\ Register)}{DAC\_MaxDigitalValue} \qquad \textbf{(Eq. 37)}$$

So, the analog sine waveform can be determined by the Equation 38:

$$Y_{SineAnalog}(x) = (3.3\ \mathrm{V}) \cdot Y_{SineDigital}(x)/0xFFF \qquad \textbf{(Eq. 38)}$$

To stablish a generated signal frequency of 50 kHz, it is necessary to set a correct frequency of the timer trigger input, in this case the timer 6 (TIM6). The frequency of the produced sine wave is the following:

$$f_{Sinewave} = f_{Timer6}/n_s \qquad \textbf{(Eq. 39)}$$

The self-rechargeable timer 6 (TIM6) timer depends on the APB1 Timer clock with a frequency of 16 MHz [64]. To obtain a stable signal frequency, without jitter on every sampling period (if the pre-scaler value was for instance 10.5, one cycle would count to ten and the next one, to 11), is necessary to follow Equation 40 (where both the n_s and the TIM6 Counter need to be integers):

$$50\ kHz = \frac{16\ MHz}{TIM6\ Counter}/n_s \qquad \textbf{(Eq. 40)}$$

Therefore, the relation between these two variables is the following:

$$TIM6Counter = \frac{320}{n_s} \quad \text{(Eq. 41)}$$

To obtain the best waveform with the highest resolution possible, the TIM6Counter is set to the lowest possible integer value, 1. Consequently, the DAC sine waveform will be formed of 320 samples. For this reason, at the start of the program, the sinewave will be generated using the following formula:

$$Y_{SineSignal}(x) = (\sin{(2\pi \cdot \frac{x}{320})} + 1) \cdot 2048 \quad \text{(Eq. 42)}$$

The sinewave obtained using this formula is saved in a memory buffer and transferred by DMA when the DAC is being activated. The transfer is triggered by the same timer that triggers the DAC.

ADC (Acquiring the current and voltage)

As explained in section 7.2, two ADCs of the board are used in this step. The ADC1 will be implemented to acquire the required voltage from the patient to later obtain the bioimpedance. In addition, the ADC2 will be used to obtain the injected current amplitude (to check what is the real injected value, since it can vary due to different circumstances) in order to avoid measuring errors. In this stage, both ADCs need to proceed with the data acquisition process simultaneously.

As previously stated, both ADCs need to be set with a sampling rate of 250 kS/s. To obtain this frequency, they are controlled by the timer 12 (TIM12). This self-rechargeable timer depends on the APB1 Timer clock with a frequency of 16 MHz [64]. To obtain the desired frequency of the ADCs, the TIM12 is set to count to the following pre-scaler value:

$$TIM12\ Prescaler = \frac{16\ MHz}{250\ KHz} = 64 \quad \text{(Eq. 43)}$$

The acquisition and consequently the TIM12 action is stopped once all the corresponding buffers are full or an error has occurred.

8.3.6.2. LBIA Processing

Extraction of the DC voltage component

As explained in section 8.2.6.1, a 4^{th} order FIR filter is used to obtain the DC component of the signals. To choose this filter order and its specifications, a set of MATLAB simulations were performed. First, a sine signal with an amplitude of 1 mV, a phase of 10º (value below the maximum of 20º stated in the requirements). Even though this signal will be conditioned and filtered with hardware systems, it makes sense to suppose some noise will remain in the signal. Therefore, White Gaussian noise with a SNR of 20 V/V was added to the signal.

After that, the previously explained synchronous demodulation process occurs where a FIR low-pass filter is used. Then, the real and capacitive components of the Bioimpedance signal (the previously generated sine signal) is obtained (as previously explained in section 5.2). After that, the phase is calculated along with its relative error as can be seen in equation 44.

$$Error(\%) = \frac{10 - obtained\ phase}{10} * 100 \quad \textbf{(Eq. 44)}$$

The low-pass filter implemented is a Butterworth Hanning window FIR filter. This type of filter is used due to its simplicity and efficiency (the window method is the most commonly used method for designing digital filters) [89]. The implemented filter had a cutoff frequency of 10 Hz (as previously explained, the frequency of interest is the DC component of 0 Hz). To better stablish the order of the filter, different simulations were done to obtain the corresponding relative errors. These values are exposed in Table 8.3.6.2.1. Since one of the requirements of the medical device is to be the as accurate and as fast as possible while obtaining and processing the data, it is important to achieve a commitment between accuracy and the required computational power this process will require.

Table 8.3.6.2.1- *Relative errors for each filter order.*

Filter Order N	Relative error (%)
1	4.60
2	22.03
3	2.05
4	0.31
5	0.04
6	0.01

As can be seen in the previous table, when the 4th order low-pass filter was implemented, the relative error was below 1%. This value is considered adequate for the developed device. In addition, a higher order filter would suppose more computational power, and slower functioning of the device. For these reasons, it is the order chosen for the implemented filter. The magnitude and impulse response of the filter can be seen in Figure 8.3.6.2.1.

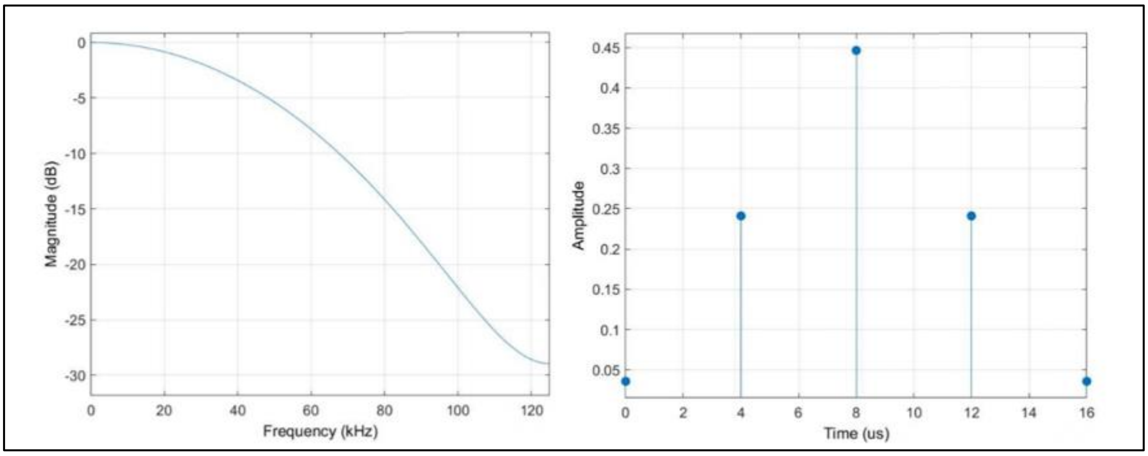


***Figure 8.3.6.2.1.-** Magnitude and Impulse response of the FIR Low-Pass filter (Figures obtained from Matlab Simultations). Own Source.*

To apply this low-pass filter to the corresponding signals, a convolution algorithm has been used in the code.

Obtaining the peak current value

As explained previously, the ADC2 is used to acquire the injected current to obtain its peak value. In this way, the real value of the injected current is known (it could have deviations from the expected value) and can be used to obtain the bioimpedance. Even though the injected current is previously conditioned using hardware components, to remove additional noise, it is band-pass filtered using a digital filter. The filter order and its specifications were obtained by means of a set of MATLAB simulations (Annex A).

In the simulations, the injected current was a sine wave signal of 250 μV_{RMS} (as mentioned in section 6.2.1.) and a frequency of 50 kHz was generated. After that, white Gaussian noise with a SNR of 20 V/V was added to the injected current. A Butterworth Hanning window FIR filter was used. The cut-off frequencies were set to 45 kHz and 55 kHz (since the frequency of the interest is 50 kHz). The range of accepted frequencies couldn't be reduced if keeping a low order was a requisite like in this case.

After applying such filter, the maximum value of the filtered signal was found (peak value) and compared to the expected value:

$$Peak\ value = 250\mu V \cdot \sqrt{2} = 353.55\ \mu V \quad \textbf{(Eq. 45)}$$

Then a relative error value was obtained with the Equation 46:

$$Error(\%) = \frac{353.55\mu V - obtained\ Peak}{353.55\mu V} * 100 \quad \textbf{(Eq. 46)}$$

To better stablish the order of the filter, 5 simulations were made, and the mean value was obtained. These values are exposed in Table 8.3.6.2.2. Since one of the requirements of the medical device is to be as accurate and as fast as possible while obtaining and processing the data, it is important to achieve a commitment between accuracy and the required computational power this process will require.

Table 8.3.6.2.2- *Relative errors for each filter order.*

Filter Order N (For the LP + HP)	Relative error (%)
1	28.47
2	10.82
3	4.82
4	1.12
5	0.89

As can be seen in the previous table, when the 4th order filter was implemented (4th order Low-pass + 4th order High Pass Filter), the relative error was below 1%. This value is considered adequate for the developed device (considering the current signal will be previously filtered using hardware components. In addition, a higher order filter would suppose more computational power, slower functioning of the device and higher current consumption. For these reasons, the order chosen for the implemented filter Is 4th. The magnitude and impulse response of the filter can be seen in Figure 8.3.6.2.2.

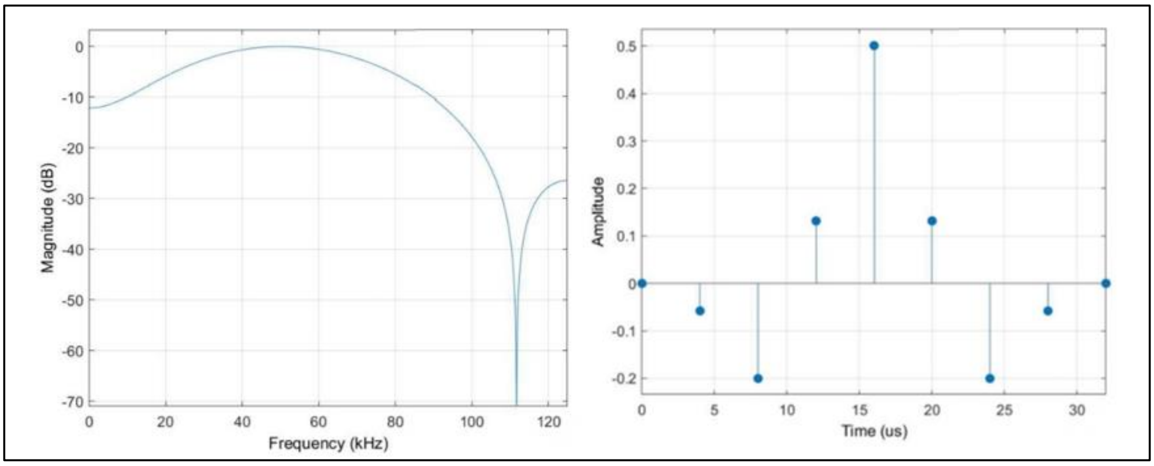


Figure 8.3.6.2.2.- *Magnitude and Impulse response of the FIR Band-Pass filter (Figures obtained from Matlab Simulations). Own Source.*

To apply this band pass filter to the corresponding signal, a convolution algorithm has been used in the code.

8.3.6.3. LBIA parameters (Conversion to float)

The real and capacitive components of the bioimpedance voltage, along with the injected peak current value are converted to floats. As previously mentioned, during the LBIA measurement process, the data is stored and processed as uint_16 variables. Considering that the Vref voltage of both ADC1 and ADC2 is 3.3V and that the hardware voltage and current amplification processes have gains of 90 V/V and 90 V/A respectively, the equations followed to convert the uint16_t parameters to floats are the following:

$$Tension\ float\ value\ (V) = \frac{3.3\ V}{2^{12}-1} \cdot \frac{uint16\_t\ value}{90} \quad \textbf{(Eq. 47)}$$

$$Current\ float\ value\ (I) = \frac{3.3\ V}{2^{12}-1} \cdot \frac{uint16\_t\ value}{90} \quad \textbf{(Eq. 48)}$$

8.3.7. EMG measurement

As previously explained and shown in section 7.2.7, the EMG measurement requires two different ADC acquisitions based on the 16-bit resolution AD7389-4. The first ADC conversion process is a 6 second acquisition to obtain the MVC value, whereas the second one is the corresponding to the 10 second EMG acquisition. For each of the two conversions, 3 channels (CH1, CH2 and CH3) will be measured simultaneously at a sample rate of 5 kS/s. Therefore, the length needed for the memory buffers for the 6 seconds acquisition is:

$$MVC\ Buffers\ Length = 5\frac{kS}{s} \cdot 6s = 30000\ uint16\_t \quad \textbf{(Eq. 49)}$$

$$EMG\ Buffers\ Length = 5\frac{kS}{s} \cdot 10s = 50000\ uint16\_t \quad \textbf{(Eq. 50)}$$

These buffers are defined with arrays of uint16_t data (since in every single conversion, 16 bits of data are obtained).

8.3.7.1. EMG Acquisition frequency

To stablish an EMG acquisition frequency of 5 kS/s, one of the board timers is required. In this case the timer 9 (TIM9) is chosen. This self-rechargeable timer depends on the APB2 Timer clock with a frequency of 16 MHz [64]. To obtain the desired frequency of the ADC the timer is set to count to the following pre-scaler value:

$$TIM9\ Prescaler = \frac{16\ MHz}{5\ KHz} = 3200 \quad \textbf{(Eq. 51)}$$

When the timer 9 reaches this value, a new data acquisition is performed, and after that, this value is brought back to 0, to start the counting again. The acquisition and consequently the TIM9 action is stopped once all the corresponding buffers are full, or an error has occurred.

8.3.7.2. SPI Configuration

Prior to the corresponding data acquisitions, it is necessary to initialize and configure the AD7389-4. To stablish a connection with this chip, the SPI1 peripheral is used. The baud rate needs to be below the maximum read frequency of the AD7389-4. Since the SCKL period of the AD7389-4 is 12.5 ns [85], the maximum frequency is:

$$Max\ Frequency = \frac{1}{12{,}5ns} = 80\ MHz \qquad \textbf{(Eq. 52)}$$

As can be seen, the baud rate should be below 80 MHz. The baud rate of the SPI1 interface depends on the APB2 Peripheral Clock with a frequency of 8 MHz [64]. Furthermore, it is necessary to impose an order 2 pre-scaler to it. Since we want the maximum transmission frequency possible to avoid communication delay errors, the pre-scaler is set to the minimum value, 2. Consequently, the baud rate for this communication is set to 4 MB/s, value below the maximum frequency allowed. On the other hand, the data Size is set to 8 bits since most of the microchips use 1 byte of data size when using the SPI interface, making it a more suitable solution in case a future change is needed.

8.3.7.3. EMG processing

To process the EMG signal (exposed in section 8.2.7.1), it is needed to choose a window size for the RMS envelope and FFT processes.

RMS envelope

The RMS envelope window size is set to 100 ms, which results in an adequate smoothing process in this type of studies (as explained in section 5.1.2). Since the sampling frequency of the EMG acquisition is 5 kS/s, the actual size of the RMS envelope buffer is the following:

$$RMS\ Buffer\ size = \frac{5kS}{s} * 0.1s = 500\ uint16\_t \qquad \textbf{(Eq. 53)}$$

FFT

In order to perform the FFT, the previously mentioned CMSIS DSP library is used [67]. In this case the "arm_fft_bin_example_f32.c" file was chosen to further develop the corresponding FFT algorithm used for this thesis. The FFT window can have different sizes, all power of 2 (256, 1024, 4096,…) [90]. To correctly chose this value, it is important to consider that a wide window gives better frequency resolution but poor time resolution, whereas a narrower window gives good time resolution but poor frequency resolution [90]. The resolution of the FFT process can be determined by the Equation 54 (where T is the duration of the FFT):

$$Resolution = \frac{1}{T} \quad \textbf{(Eq. 54)}$$

Considering that the sampling frequency of the EMG acquisition is 5 kS/s, the previous equation translates to the Equation 55:

$$Resolution = \frac{5000}{FFT\ window\ size} \quad \textbf{(Eq. 55)}$$

The frequential resolutions that would be obtained with different FFT window sizes are exposed in Table 8.3.7.1.

Table 8.3.7.1- *Frequential resolutions with different FFT window sizes.*

FFT Window Size	Frequential resolution (Hz)
1024	4.88 Hz
2048	2.44 Hz
4096	1.22 Hz
8192	0.61 Hz
16384	0.30 Hz

As can be seen in the previous table, the first window size which entails a frequential resolution below 1 Hz (which is an acceptable frequential resolution as stated in section 6.2.2) is 8192. Even though, the frequential resolution is prioritized in this case, it is intended to maintain a compromise between both the time and frequency resolution. For this reason, 8192 is the window size chosen, instead of going for a higher number.

8.3.7.4. EMG standard amplitude parameters (Conversion to float)

During the EMG Processing process, the standard amplitude parameters are converted to float values. As previously mentioned, EMG data acquisition is stored and processed as uint_16 variables. Considering that the Vref voltage imposed for the AD7893-4 is 2.5V and the hardware EMG amplification process has a gain of 500 V/V, the Equation 56 is used to convert the uint16_t parameters to floats is the following:

$$Float\ value\ (V) = \frac{2.5\ V}{2^{16} - 1} \cdot \frac{uint16\_t\ value}{500} \quad \textbf{(Eq. 56)}$$

9. Preliminary LBIA Statistical Analysis

As previously stated in section 6, a set of LBIA and EMG measurements were done to clearly understand what functionalities and characteristics were required for the device. In this thesis, a preliminary statistical analysis has been performed using the LBIA data. Further data will be obtained from a larger cohort of patients with total knee arthroplasty.

9.1. Patients Sample

The sample size was 12 knee arthroplasty patients (31.4 ± 5.3 kgm-2; 69 ± 8 yr.) In addition, epidemiological variables such as age, weight, height, and BMI were analysed.

All patients underwent a pre - and post-operative clinical assessment at 6 and 12 months of their evolution using clinical KSS (Knee Society Score) scales. The KSS knee scale gives an overall assessment of the knee by completing a questionnaire with 7 variables [91]. A section assessing functional parameters (3 items) is then added to the original score. Both sections are scored from 0 to 100; lower scores indicate poorer functional ability of the patient whereas higher KSS knee scores indicate better functional outcomes [91].

9.2. Bioimpedance Measurement Materials

To obtain the bioimpedance data, the previously mentioned BIA 101 Anniversary was used. The electrodes chosen for this measurement were the contact electrode Ag/AgCl (COVIDIEN Ref. 31050522, COVIDIEN llc, Mansfield, IL, USA) with R and Xc intrinsic values of 10.89 Ω and 0.30 Ω respectively [35].

9.3. Localized bioimpedance electrode placement (L-BIA)

Patients were set in the supine decubitus position when the electrodes (tetra-polar single-frequency at 50 kHz) were placed in the main muscle groups of the lower limbs. The specific placement was the following:

- Rectus femoris: 5 cm distally from anterior inferior iliac spine; and 10 cm proximally from the superior pole of the patella.

- Vastus medialis: 2 cm medial from the proximal rectus femoris electrode (Inferior portion of intertrochanteric line); and 2 cm proximal and medial to the medial border of the patella.

- Vastus lateralis: 2 cm lateral from the proximal rectus femoris electrode and at the confluence with 5 cm inferior to the anterior aspect of the greater trochanter; and 2 cm proximal and lateral to the lateral border of the patella.

9.4. Data analysis

When the data corresponding to the 13 patients has been obtained, it is statistically processed with the statistical software IBM® SPSS® version 28.0 (Armonk, NY: IBM Corp, USA) [92]. The categorical values to correctly identify the statistical population of this study are exposed in Table 9.4.1.

As can be seen in Table 9.4.1, the categorical values used in this thesis discern: the gender of the patient, the intervention side (left or right leg), if a previous knee operation was previously performed or a total knee replacement (TKA) was performed, if the patient suffers from painful contralateral arthrosis, if a total hip replacement was performed (THR) and the type of knee prosthesis implanted (TKR).

Table 9.4.1.- *Frequency Table of the Bioimpedance statistical analysis population.*

			Frequency	Percent (%)	Valid Percent (%)	Cumulative Percent (%)
GENDER	**Valid**	**Female**	9	75	75	75
		Male	3	25	25	100
		Total	12	100	100	
SIDE	**Valid**	**Left**	2	16.7	16.7	17.6
		Right	10	83.3	83.3	100.0
		Total	12	100.0	100.0	
PREVIOUS SURG. KNEE	**Valid**	**Yes**	3	25.0	25.0	25.0
		No	9	75.0	75.0	100.0
		Total	12	100.0	100.0	
TKR CONTRALATERAL	**Valid**	**Yes**	3	25.0	25.0	25.0
		No	9	75.0	75.0	100.0
		Total	12	100.0	100.0	
PAINFULL CONTRALATERAL ARTHROSIS	**Valid**	**Yes**	3	25.0	33.3	33.0
		No	6	50.0	66.7	100.0
		Total	9	75.0	100.0	
	Missing	**System**	3	25.0		
	Total		12	100.0		
THR	**Valid**	**No**	12	100.0	100.0	100.0
TKR TYPE	**Valid**	**MP**	6	50.0	50.0	50.0
		UC	6	50.0	50.0	100.0
		Total	12	100.0	100.0	

Once the characteristics of the studied population are identified, the data processing process is developed.

First, the normality of distribution in the variables is determined by the Shapiro-Wilk test (n=12). The variables normally distributed are shown as mean ± SD while that non-normally distributed data are shown as the median and interquartile range (IQR). After that, the repeated T-test serves to determine the effect of total knee arthroplasty (TKA) on LBIA measured one-week pre-IQ (pre-surgical intervention) and six months post-IQ_TKA (post-total knee arthroplasty surgical intervention) (figure 9.4.1).

Paired Samples Test

		Paired Differences							Significance	
					95% Confidence Interval of the Difference					
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper	t	df	One-Sided p	Two-Sided p
Pair 1	R_Left_RF_Ohm_1 - R_Left_RF_Ohm_2	-5.85000	20.07846	5.79615	-18.60724	6.90724	-1.009	11	.167	.335
Pair 2	Xc_Left_RF_Ohm_1 - Xc_Left_RF_Ohm_2	-.70000	2.24297	.64749	-2.12511	.72511	-1.081	11	.151	.303
Pair 3	R_Right_RF_Ohm_1 - R_Right_RF_Ohm_2	11.44167	24.93047	7.19681	-4.39840	27.28173	1.590	11	.070	.140
Pair 4	Xc_Right_RF_Ohm_1 - Xc_Right_RF_Ohm_2	.60000	1.89928	.54828	-.60675	1.80675	1.094	11	.149	.297
Pair 5	R_Left_VM_Ohm_1 - R_Left_VM_Ohm_2	-6.85833	24.22694	6.99371	-22.25139	8.53473	-.981	11	.174	.348
Pair 6	Xc_Left_VM_Ohm_1 - Xc_Left_VM_Ohm_2	-.48333	2.53694	.73235	-2.09523	1.12856	-.660	11	.261	.523
Pair 7	R_Right_VM_Ohm_1 - R_Right_VM_Ohm_2	6.60000	26.47737	7.64336	-10.22292	23.42292	.863	11	.203	.406
Pair 8	Xc_Right_VM_Ohm_1 - Xc_Right_VM_Ohm_2	1.18333	1.79232	.51740	.04455	2.32212	2.287	11	.021	.043
Pair 9	R_Left_VL_Ohm_1 - R_Left_VL_Ohm_2	-7.36667	27.67215	7.98826	-24.94871	10.21538	-.922	11	.188	.376
Pair 10	Xc_Left_VL_Ohm_1 - Xc_Left_VL_Ohm_2	-.95000	1.82782	.52764	-2.11134	.21134	-1.800	11	.050	.099
Pair 11	R_Right_VL_Ohm_1 - R_Right_VL_Ohm_2	9.78333	21.70316	6.26516	-4.00619	23.57286	1.562	11	.073	.147
Pair 12	Xc_Right_VL_Ohm_1 - Xc_Right_VL_Ohm_2	1.67500	1.59381	.46009	.66234	2.68766	3.641	11	.002	.004

***Figure 9.4.1.-** Paired Samples Test from the IBM SPSS software, with most statistically relevant data highlighted in red. Own source.*

As can be seen in the previous figure, the data with more statistical relevance (the data that has changed the most drastically in the previously defined period) is the corresponding to the Xc values from the VL and VM muscles. In addition, it can be seen that the right leg values have more relevance than the corresponding to the left leg (right VM and VL have one and two-sided p significance while the left VL Xc data is only one-sided p significant). This result is coherent with the patient sample used since as can be seen in table 6.1.4.1, 10 of the 12 patients had the surgical intervention in their right leg (83.3%).

Then, a PCA (Principal Components Analysis) is done between Knee Society Score (KSS) and LBIA parameters which shows statistical significance (Annex B). The level of statistical significance is set at $P < 0.05$. The results of the PCA test are shown in Figure 6.1.4.2.

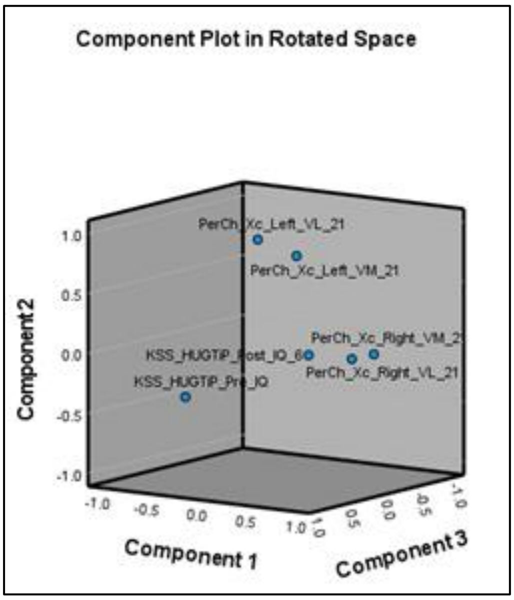


Figure 9.4.2.- *PCA Component plot in rotated space from the IBM SPSS software. Own source.*

As can be seen in the previous figure, the more statistically relevant parameters are the Xc values from the right VL and VM. Since the obtained result corresponds with the previously shown with the paired samples test from figure 9.4.2, we can conclude that effectively, these are the more relevant parameters. However, it is important to state that all the other parameters are also needed to be able to fully study the patient evolution.

10. Environmental impact analysis

This thesis has been mainly focused on the firmware and software design of the present medical device, consequently, a computer has been used for the entire programming process. This use entails an electricity consumption that involves a corresponding CO_2 footprint.

To estimate it, first it is needed to know how much electricity the used computer requires. The laptop used in this thesis is a Dell G15 with the specifications stated in [93]. Considering it has been used a total of 1200 hours with purposes related with this thesis, approximately 372 kWh have been consumed [94]. This amount of electricity corresponds to 93.00 kg of CO_2 [95].

11. Applicable regulations

In this section all the regulations that would apply to the medical device developed in this thesis as a whole (not only the firmware corresponding regulations) are gathered.

11.1. General Medical Device regulations

According to the *UNE-EN 60601-1:2018* [96] standard for electromedical equipment, electromedical equipment is the equipment that:

- Has an applied part, which is the part of the equipment that, in its normal use, necessarily comes into physical contact with the patient in order to perform its function.
- Transfers energy to or from the patient or senses such energy transfer to or from the patient.
- The manufacturer's intended use is related to the diagnosis, treatment or monitoring of the patient or to alleviate or compensate for a disability, illness or injury.

Following this definition, the device presented in this project can be defined as an electromedical equipment. Therefore, it must follow the *UNE-EN 60601-1:2018* [96] standard approved by CENELEC on 12 September 2006 concerning general requirements for basic safety and essential performance. As previously explained, one basic step of the bioimpedance measurement performed by the present medical device is the injection of current directly to the patient's tissue. In order to avoid any possible harmful effect, this device follows the safety standards specified in this rule. For instance, the maximum peak value of this injected current is below 1 mA.

In addition to this general standard, several collateral standards apply to the device presented in this project:

- UNE-EN 60601-1-2:2015, Electromagnetic disturbances: Requirements and tests: This standard specifies electromagnetic compatibility requirements for medical systems or devices. Electromagnetic compatibility is the ability of equipment to function correctly in the electromagnetic environment for which it is intended to be used. Electromagnetic disturbances in the environment must not affect the operation of the equipment, nor must the equipment emit disturbances that may affect devices in its environment. The device presented in this project has BLE communications with external devices, so this standard must be applied [97].

- UNE-EN 60601-1-6:2010, Usability: This standard specifies a process for the manufacturer to analyze, specify, design, verify and validate fitness for use. This engineering process evaluates and eliminates the risks caused by fitness-for-use problems associated with the correct use and associated with correct usage and usage errors [98].

Additionally, the normative *UNE-EN 61000-4* is based on different parts related to EMC compatibility tests that apply to the device:

- UNE-EN 61000-4-2:2010. Electromagnetic compatibility. Part 4-2. Test and measurement techniques. Electrostatic discharge immunity test: The purpose of this standard is to establish a common basis for assessing the performance characteristics of electrical and electronic equipment when subjected to electrostatic discharge [99].

- UNE-EN 61000-4-3:2007. Electromagnetic compatibility. Part 4-3: Test and measurement techniques. Testing for immunity to electromagnetic, radiated and radio frequency fields: This part specifies the immunity tests to be carried out to ensure the protection of equipment against electromagnetic fields from any Source [100].

- UNE-EN 61000-4-8:2011. Electromagnetic compatibility. Part 4-8: Test and measurement techniques. Testing for immunity to magnetic fields at industrial frequency: This standard addresses the immunity of equipment under operating conditions to magnetic disturbances at 50 Hz and 60 Hz frequencies related to commercial, and residential premises, industrial installations, power stations and high and medium voltage substations [101].

It is also applicable the standard *UNE-EN 62366-1:2015* [102]. *Medical devices - Part 1: Application of usability engineering to medical devices (Endorsed by Asociación Española de Normalización in September of 2020).* The objective of this standard is to detail a process for the manufacturer to analyze, specify, develop, and evaluate the usability of the medical device in relation to Safety, to mitigate the risks associated with misuse of the device. In addition, for patient safety, a risk analysis according to *UNE-EN ISO 14971:2019* [103]. is required. This standard aims to describe a risk management process to help the manufacturer to identify hazards, estimate and evaluate risks, control, or correct these risks and monitor the effectiveness of controls. For the quality management of medical devices, the *ISO 13485:año* [104] also is applicable.

11.2. Firmware oriented regulations

Since the scope of this thesis is the firmware and software design of this medical device, the international standard *IEC 62304:2007 Medical device Software* [105] has been considered and followed in every step of this project. Additionally, as seen in this thesis, the developed medical device acquires and sends medical data from the patients. Therefore, the GDPR (General Data Protection Regulation) [106] is also applicable.

11.2.1. IEC 62304:2007

This standard is applicable to the development and maintenance of the medical device software when:

- The software is by itself a medical device.
- The software is used in the production of a medical device.
- The software is an embedded part of the final medical device.

As a foundation, the standard specifies that medical device software must be developed and maintained within a quality management system (section 9.2.1) and a risk management system (9.2.2). As well as the designation of a software safety classification [105].

11.2.1.1. Quality management system

Manufacturers of medical device software shall demonstrate the ability to deliver a product that meets customer expectations and needs as well as regulatory requirements.

The capability mentioned in this requirement can be demonstrated using a quality management system that is compliant with *ISO 13485:año* (previously mentioned) [107]. But a generally accepted national quality management standard, or an accepted quality management system required by national regulation, can also be applied.

11.2.1.2. Risk management system

In terms of risk management, *UNE-EN 62304:año* is very precise [107]. Only the implementation of a risk management system in accordance with *ISO 14971* (previously mentioned) is accepted [107].

11.2.1.3. Software evaluation

Beyond the standard class system, the FDA evaluates software as a medical device based on its scope. The clinical evaluation puts the software into one of four categories [107]. As the category number rises, the software's impact on patients does too:

- <u>Type I:</u> This type includes informative programs that deal with no-serious data. For example, the software might collect, store, and transmit patient data such as blood pressure readings and heart rate statistics
- <u>Type II:</u> The software informs and drives more pressing matters. This type of medical software often analyzes heart rates, predicts the risk of disease, or recommends diagnosis.
- <u>Type III:</u> The software is more active and can drive critical processes and treat severe conditions. The software might detect breathing irregularities or monitor a disease.
- <u>Type IV:</u> This medical software type can treat or diagnose critical issues. For example, the software can provide treatment solutions for stroke victims.

Since the device designed in this thesis will be used in a study to evaluate the patient progression using a knee prosthesis (it does not inform about critical matters), it's software / firmware is a type I medical device [107].

11.2.2. General Data Protection Regulation (GDPD)

Health-related data such as medical records are considered sensitive information and are therefore particularly protected. Any personal data collected or processed must be protected. To do so, appropriate technological and organizational measures must be implemented to ensure a level of security appropriate to the level of risk [106]. The used medical device must also encrypt personal data at rest or in transit, unless data are otherwise protected through pseudonymization, and individuals cannot be identified from their data [106].

Conclusions

During the development of this present thesis, I have applied all the acquired knowledge of both my bachelor's degrees, *Biomedical Engineering* and *Industrial Electronics and Automatic Control Engineering*. In relation to my whole biomedical studies, I have explained and studied the main characteristics, and specifications of the EMG and LBIA measurements, along with the state of the art of the corresponding medical devices, and the digital signal processing techniques. In addition, I have done a preliminary statistical analysis obtaining the first results.

On the other hand, I have been able to apply what I have learned in my degree in electronic engineering when developing the corresponding firmware design. However, since in this degree I only attended one unit related to microcontroller programming (Industrial Computer Science), there has been a steep learning curve to be able to develop this design. To overcome this difficulty, I have done wide bibliography research using different academic searchers.

The whole firmware design has been developed with all the corresponding processes, implementing two different types of data communication (USART and SPI), BLE communication with an external device, an error handling functionality, and the corresponding digital signal processing techniques for both LBIA (using lock-in amplifiers) and EMG measurements.

It is necessary to state that although this thesis has completely accomplished all the objectives (principal and specific), to be able to use this device in a medical setting, future work will be done. After the presentation of this thesis, it is intended to implement the designed firmware with its corresponding hardware components to construct the present medical device. Consequently, it will be possible to check the structure and functions of the code in a practical environment. If the device performance is not adequate, different solutions will be implemented in order to solve the problem in each case. Additionally, the mobile application required to communicate and interact with the medical device will also be implemented.

After that, different tests and functionality studies will be performed as regularly as when manufacturing a medical device, to ensure it fulfills all the applicable normative. Furthermore, the next step of the project would be to statistically analyze and compare the results obtained with this device with the results obtained with the well-stablished commercial devices mentioned in this work. If these results are positive, this device will be further used in the studies such as the one done in the Hospital Germans Trias I Pujol.

Economic Analysis

In this section, the economic cost of this thesis is discussed. Although the development of the actual medical device would be more expensive due to all the components implemented for the corresponding hardware design, this part has not been considered for the budget since it is out of the scope of this thesis.

Labour

In this section are included the cost of designing and coding the medical device. The established price for hours for extracurricular internships (according to the EEBE normative) is 8 € [108]. The amount of hours required for credit according to the EEBE normative is 25. Consequently, since this thesis is worth 48 ECTS, the total of hours inverted has been of 1200 hours.

Electricity cost

As previously stated in the environmental impact section, during this thesis, a total of approximately 372 kWh has been consumed. Since the mean price of electricity in Spain in this year 2022 has been 0.3124 €/kWh [109], the cost related to this factor has been approximately 116.21 €.

The budget of this thesis can be seen in the following table.

Table Economic Analysis.1.- *Thesis total Budget*

	Unitary Price (€)	Units	Total Price (€)
Labour	8.00	1200	9600.00
Electricity	0.31	372	116.21
TOTAL			9716.21

As exposed in the previous table, the total price is as high as 9716.21 €.

Bibliography

[1] Anatomy of the knee. *Knee replacement surgery* [online]. United States of America: Johns Hopkins Medicine, 2019. Available from: https://www.hopkinsmedicine.org/health/treatment-tests-and-therapies/knee-replacement-surgery-procedure

[2] Quadriceps. *Knee Muscles* [online]. Chloe Wilson. KPE Medical Review Board, 2018. Available from: https://www.knee-pain-explained.com/kneemuscles.html

[3] Types of Knee Ligaments. *Ligaments in the knee* [online]. Stanford Health Care. Available from: https://stanfordhealthcare.org/medical-conditions/bones-joints-and-muscles/knee-ligament-injury/types.html

[4] Is Medial Pivot Design a Major Advancement in Total Knee Arthroplasty. International Congress for Joint Reconstruction [online]. Dionisio Ortiz III, MD; Akash K. Shah, MD; James Slover, MD, MSc; Ran Schwarzkopf, MD, MSc, 2021. Available from: https://icjr.net/articles/is-medial-pivot-design-a-major-advancement-in-total-knee-arthroplasty

[5] PUTAME, Giovanni, et al. Kinematics and kinetics comparison of ultra-congruent versus medial-pivot designs for total knee arthroplasty by multibody analysis. Scientific Reports, 2022, vol. 12, no 1, p. 1-11.

[6] Artroplastia de rodilla. Cirurgía ortopédica y traumatología [online]. Palex Constant Improvement. Available from: https://www.palexmedical.com/es/group.cfm?id=artroplastia%2Drodilla#.YjtSTefMJD8

[7] SONG, Eun-Kyoo, et al. Total knee arthroplasty using ultra-congruent inserts can provide similar stability and function compared with cruciate-retaining total knee arthroplasty. Knee Surgery, Sports Traumatology, Arthroscopy, 2017, vol. 25, no 11, p. 3530-3535.

[8] Sistema Total de rodilla. Columbus, sistema total de rodilla [online]. Brau Sharing Expertise. Available from: https://www.bbraun.es/es/products/b/columbus-sistematotalderodilla.html

[9] Chapter 5.1. Webster, J.G, “The Electrode-Electrolyte Interface, J. Wiley (Ed), Medical Instrumentation, Application and Design,” 2010, p. 189-192.

[10] Sección VIII. Netter, F.” Fisiología y neuroanatomía funcional, Salvat (Ed), Sistema nervioso, anatomía y fisiología”, 1987. p. 157-213.

[11] KHALIL, Sami F.; MOHKTAR, Mas S.; IBRAHIM, Fatimah. The theory and fundamentals of bioimpedance analysis in clinical status monitoring and diagnosis of diseases. Sensors, 2014, vol. 14, no 6, p. 10895-10928. DOI: 10.3390/s140610895.

[12] KHODADAD, Davood, et al. The value of phase angle in electrical impedance tomography breath detection. En *2018 Progress in Electromagnetics Research Symposium (PIERS-Toyama)*. IEEE, 2018. p. 1040-1043.

[13] LUKASKI, Henry C., et al. Classification of hydration in clinical conditions: indirect and direct approaches using bioimpedance. *Nutrients*, 2019, 11.4: 809.

[14] ELLIS, Kenneth J. Human body composition: in vivo methods. *Physiological reviews*, 2000, 80.2: 649-680.

[15] COFFMAN, Frederick D.; COHEN, Stanley. Impedance measurements in the biomedical sciences. Stud Health Technol Inform, 2013, vol. 185, p. 185-205. DOI: 10.3233/ACP-2012-0070.

[16] PICCOLI, A.; NESCOLARDE, L. D.; ROSELL, J. Análisis convencional y vectorial de bioimpedancia en la práctica clínica. *Nefrología*, 2002, vol. 22, no 3, p. 228-238.

[17] NESCOLARDE, L., et al. Effects of muscle injury severity on localized bioimpedance measurements. *Physiological measurement*, 2014, vol. 36, no 1, p. 27.

[18] GRIMNES, Sverre; MARTINSEN, Orjan G. *Bioimpedance and bioelectricity basics*. Academic press, 2011.

[19] Lock-In Amplifiers. *Principle of Lock-in Amplifiers* [online]. Zurich Instruments, 2020. Available from: https://www.zhinst.com/europe/en/resources/principles-of-lock-in-detection

[20] (Martini, Frederic H. "Chapter 10. Muscle Tissue." Anatomy & Physiology. San Francisco: Benjamin Cummings, 2005.)

[21] MERLETTI, Roberto; PARKER, Philip J. (ed.). *Electromyography: physiology, engineering, and non-invasive applications*. John Wiley & Sons, 2004.

[22] BUCHTHAL, Fritz; GULD, Christian; ROSENFALCK, Poul. Multielectrode study of the territory of a motor unit. *Acta Physiologica Scandinavica*, 1957, vol. 39, no 1, p. 83-104.

[23] CLANCY, Edward A. Design of a High–Resolution Surface Electromyogram (EMG) Conditioning Circuit. *Worcester Polytechnic Institute*, 2012.

[24] MUÑOZ FABRA, Elena. *Diseño e implementación de un sistema electrónico de captación de señal de electromiografía para el estudio de la fatiga muscular en condiciones estáticas y dinámicas*. 2021. Tesis Doctoral. Universitat Politècnica de València.

[25] Day, Dr. Scott: Important Factors in Surface EMG Measurement. Bortec Biomedical, 2002.

[26] CLANCY, Edward A.; MORIN, Evelyn L.; MERLETTI, Roberto. Sampling, noise-reduction and amplitude estimation issues in surface electromyography. *Journal of electromyography and kinesiology*, 2002, vol. 12, no 1, p. 1-16.

[27] PALLAS-ARENY, Rarnon. Interference-rejection characteristics of biopotential amplifiers: a comparative analysis. *IEEE Transactions on Biomedical Engineering*, 1988, vol. 35, no 11, p. 953-959.

[28] VAN VUGT, J. P. P.; VAN DIJK, J. G. A convenient method to reduce crosstalk in surface EMG. *Clinical Neurophysiology*, 2001, vol. 112, no 4, p. 583-592.

[29] LÉVESQUE, L. Revisiting the Nyquist criterion and aliasing in data analysis. *European Journal of Physics*, 2001, vol. 22, no 2, p. 127.

[30] Application Report, *Understanding Data Converters* [Online]. Texas instruments. Available from: https://www.ti.com/lit/an/slaa013/slaa013.pdf

[31] KAMEN, Gary; CALDWELL, Graham E. Physiology and interpretation of the electromyogram. *Journal of Clinical Neurophysiology*, 1996, vol. 13, no 5, p. 366-384.

[32] Soundarapandian, Karthik Mark Berarducci: Analog Front-End Design for ECG Systems Using Delta-Sigma ADCs. Texas Instruments, 2009.

[33] TOGAWA, Tatsuo; TAMURA, Toshiyo; ÖBERG, P. Åke. Biomedical sensors and instruments. 2011.

[34] GUERREIRO, José. *A biosignal embedded system for physiological computing*. 2013. Tesis Doctoral. Instituto Superior de Engenharia de Lisboa.

[35] NESCOLARDE, L., et al. Different displacement of bioimpedance vector due to Ag/AgCl electrode effect. *European journal of clinical nutrition*, 2016, vol. 70, no 12, p. 1401-1407.

[36] Shiwei X, Dai X, Meng X, Canhua X, Chaoshuang Ch, Mengxing T et al. Performance evaluation of five types of Ag/AgCl bio-electrodes for cerebral electrical impedance tomography. Ann Biomed Eng 2011; 39: 2059–2067.

[37] Mbody 3 kit, *Myontec* [online], 2020. Available from: https://www.myontec.com/product-page/mbody-3

[38] TeleMyo (DTS) EMG [online], NORAXON, 2020. Available from: http://www.noraxon.com/wp-content/uploads/2014/12/telemyo_dts_belt_4-v1-3.pdf.

[39] FREEEMG 1000 [online], BTS Bioengineering, 2021. Available from: https://www.btsbioengineering.com/products/freeemg-clinical-functional-protocols/

[40] Datasheet [online], FUNCTIONAL EVALUATION, BTS Bioengineering, 2021.

[41] BIA 101 Anniversary, *Bioelectric Impedance Analysis* [online]. AKERN. Available from: https://realmetinstitute.com/product/bia-101-anniversary-sport-full/?lang=en

[42] Quantum V Segmental BIA, *Whole and Segmental Body Composition Analysers* [Online]. RJL Systems. Available from: https://www.rjlsystems.com/products/quantum-v-segmental/#section=description

[43] KONRAD, Peter. The abc of emg. *A practical introduction to kinesiological electromyography*, 2005, vol. 1, no 2005, p. 30-5.

[44] SANTOS, Artur Bonezi; SOARES, Denise Paschoal; CANDOTTI, Cláudia Tarrago. Smoothing EMG signals: Implications on delay calculation. *RPCD*, 2012, 12.1: 60-72.

[45] GOEN, Anjana; TIWARI, D. C. Review of surface electromyogram signals: its analysis and applications. *International Journal of Electrical, Electronics, Communication, Energy Science and Engineering*, 2013, 7.11: 965-973.

[46] D. T. Mewett, K.J. Reynolds, and H. Nazeran, “Reducing power line interference in digitized electromyogram recordings by spectrum interpolation”, Medical and Biological Engineering and Computing, vol. 42, pp. 524-531, 2004.

[47] BEHBAHANI, Soroor. Investigation of Adaptive Filtering for Noise Cancellation inECG signals. In: *Second International Multi-Symposiums on Computer and Computational Sciences (IMSCCS 2007)*. IEEE, 2007. p. 144-149.

[48] MERLETTI, Roberto; DI TORINO, PJJEK. Standards for reporting EMG data. *J Electromyogr Kinesiol*, 1999, vol. 9, no 1, p. 3-4.

[49] BROWN, Robert A.; LAUZON, M. Louis; FRAYNE, Richard. Developments in Time-Frequency Analysis of Biomedical Signals and Images Using a Generalized Fourier Synthesis. 2009.

[50] KARHEILY, Somar, et al. Time-frequency Features for sEMG Signals Classification. En *BIOSIGNALS*. 2020. p. 244-249.

[51] The Usefulness of Mean and Median Frequencies in Electromyography Analysis, *Computational Intelligence in Electromyography Analysis* [online]. Available from: https://www.intechopen.com/chapters/40123

[52] GÓMEZ ABAD, Daniel. *Development of a capacitive bioimpedance measurement system*. 2009. Tesis de Maestría. Universitat Politècnica de Catalunya.

[53] KIRKHORN, Johan. Introduction to IQ-demodulation of RF-data. *IFBT, NTNU*, 1999, vol. 15.

[54] Sistemas, Universidad Politécnica de M. Señales y Comunicaciones d. Señnales y Comunicaciones C. Señales y Comunicaciones: Electronica y comunicaciones. http://www.gr.ssr.upm.es/elcm/actual/ pdf. Version: 2007

[55] Pindado, R: Phase Locked-Loop (PLL): Fundamento y aplicaciones. / Departament d'Enginyeria Electrònica. Universitat PolitÈcnica de Catalunya. E.U.E.T.I.T.– Forschungsbericht

[56] HERNÁEZ RIOJA, I. Procesado de senal en comunicaciones: modulación QAM. 2007.

[57] NARANJO-HERNÁNDEZ, David; REINA-TOSINA, Javier; MIN, Mart. Fundamentals, recent advances, and future challenges in bioimpedance devices for healthcare applications. *Journal of Sensors*, 2019, vol. 2019.

[58] BRATAAS, Patrick Hisni. *Wireless embedded microcontroller system for bioimpedance measurements*. 2014. Tesis de Maestría.

[59] STM32F3 Discovery KIT, *Discovery kit with STM32F303vc MCU* [online]. STMicroelectronics. Available from: https://www.st.com/en/evaluationtools/stm32f3discovery.html#:~:text=might%20also...-,Description,users%20to%20get%20started%20quickly.

[60] STM32F4 Discovery KIT, *Discovery kit with STM32F407VG MCU* [online]. STMicroelectronics. Available from: https://www.st.com/en/evaluation-tools/stm32f4discovery.html

[61] SAM E70 Xplained Evaluation Kit, *Microchip Developer Help* [online]. Available from: https://microchipdeveloper.com/boards:sam-e70-xpro

[62] STM32Cube Initialization code generator *STM32CubeMX* [online]. STMicroelectronics. Available from: https://www.st.com/en/development-tools/stm32cubemx.html

[63] Data Types and Sizes, *Oracle.* Available from: https://docs.oracle.com/cd/E19253-01/817-6223/chp-typeopexpr-2/index.html

[64] STM32F407 Manual, *RM0090 Reference manual* [online]. STMicroelectronics. Available from: https://www.st.com/resource/en/reference_manual/dm00031020-stm32f405-415-stm32f407-417-stm32f427-437-and-stm32f429-439-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf

[65] Exploring STM32F407 Discovery Board. *ARM cortex Based STM32F407 Discovery board* [online]. STMicroelectronics. Available from: https://github.com/SharathN25/STM32F407-Discovery

[66] What is Direct Memory Access (DMA) and How Does it Work [online]. MiniTool. Available from: https://www.minitool.com/lib/direct-memory-access.html

[67] CMsIS DSP Software library, *CMSIS* [online]. Available from: https://www.keil.com/pack/doc/CMSIS/DSP/html/index.html

[68] Description of STM32F4 HAL and low-layer drivers, *UM1725* [online]. STMicroelectronics. Available from: https://www.st.com/resource/en/user_manual/dm00105879-description-of-stm32f4-hal-and-ll-drivers-stmicroelectronics.pdf

[69] MDK-ARM Version 5, *ARMKEIL Microcontroller Tools* [online]. Digi-key. Available from: https://www.digikey.es/es/product-highlight/k/keil/mdk-arm-version-5

[70] Bluetooth 4.0 BLE module [online]. Jinan Huamao Technology CO. Available from: https://www.rhydolabz.com/documents/37/datasheet%20HM-10.pdf

[71] CellPac 503562, *VARTA* [online]. Available from: https://www.varta-ag.com/en/industry/product-solutions/lithium-ion-battery-packs/cellpac-lite/1/lpp-503562-s

[72] *STM32F407xx Production data,* [online]. STMicroelectronics. Available from: https://www.st.com/resource/en/datasheet/dm00037051.pdf

[73] OVSIYENKO, Denys. *Diseño de un sistema Bluetooth Low Energy para la medida de fuerza*. 2020. Tesis de Licenciatura. Universitat Politècnica de Catalunya.

[74] Bluetooth Low Energy, *Bluetooth low energy (BLE) Fundamentals* [online]. Available from: https://www.embedded.com/bluetooth-low-energy-ble-fundamentals/

[75] Que es el wi-fi, *Que es el wi-fi y cómo funciona* [online]. Available from: https://www.adslzone.net/reportajes/tecnologia/que-es-wifi-como-funciona/

[76] Configure Wireless APs. *Wireless Acces Deployment* [online], Microsoft. Available from: https://docs.microsoft.com/en-us/windows-server/networking/core-network-guide/cncg/wireless/e-wireless-access-deployment

[77] BLE vs Wi-Fi. *BLE vs Wi-Fi: Which is better for IoT* [online], Cabot. Available from: https://www.cabotsolutions.com/ble-vs-wi-fi-which-is-better-for-iot-product-development#:~:text=The%20proximity%20data%20provided%20by,the%20connectivity%20through%20external%20antennas.

[78] Módulo Bluetooth 4.0 BLE HM-10 [online]. Available from: https://naylampmechatronics.com/inalambrico/133-modulo-bluetooth-40-ble-hm-10.html

[79] Módulo Bñuetooth Nrf51822, *BLE modules* [online]. Nordic Semiconductor Available from https://infocenter.nordicsemi.com/index.jsp?topic=%2Fstruct_nrf51%2Fstruct%2Fnrf51822.html

[80] Nrf51822, how to minimize current consumption for BLE application using nRF51822. Available from: https://devzone.nordicsemi.com/f/nordic-q-a/1657/how-to-minimize-current-consumption-for-ble-application-on-nrf51822#:~:text=Low%20Power%20UART%20When%20you,of%20current%2C%20~1mA%20constantly.

[81] Cypress Semiconductor PSoC 4 BLE Module [online]. Cypress. Available from: https://www.mouser.es/new/cypress-semiconductor/cypress-cyble-224116-01-ble-module/

[82] Encryption protocols, *What is Data encryption: Types, Algorithms, Techniques and Methods* [online]. Available from: https://www.simplilearn.com/data-encryption-methods-article

[83] What is the XOR Cypher, *The XOR Cypher.* Defend the web [online]. Available from: https://defendtheweb.net/article/the-xor-cipher

[84] Random values in c, *Generating random values in C.* Randomization [online]. Available from: https://www.cs.yale.edu/homes/aspnes/pinewiki/C(2f)Randomization.html

[85] AD7389-4 Datasheet, *Differential Input, Quad, Internal Reference Simultaneous Sampling, 16-Bit SAR ADC* [online]. Available from: https://www.analog.com/media/en/technical-documentation/data-sheets/ad7389-4.pdf

[86] Quad-SPI interface on SMT32 microcontrollers and microprocessors, AN5760 [online]. STMicroelectronics. Available from: https://www.st.com/resource/en/application_note/an4760-quadspi-interface-on-stm32-microcontrollers-and-microprocessors--stmicroelectronics.pdf

[87] DMA, *STM32GO Direct memory access controller (DMA)* [online]. STMicroelectronics. Available from: https://www.st.com/content/ccc/resource/training/technical/product_training/group0/10/5f/2c/5e/70/3e/49/8c/STM32G0-System-Direct-memory-access-controller-DMA/files/STM32G0-System-Direct-memory-access-controller-DMA.pdf/jcr:content/translations/en.STM32G0-System-Direct-memory-access-controller-DMA.pdf

[88] Extending the DAC performance of STM32 microcontrollers,*AN4566 Application note* [online]. STMicroelectronics. Available from: https://www.st.com/resource/en/application_note/dm00129215-extending-the-dac-performance-of-stm32-microcontrollers-stmicroelectronics.pdf

[89] Finite impulse (FIR) filter design methods. EBOOKS [online]. Available from: https://www.mikroe.com/ebooks/digital-filter-design/finite-impulse-response-fir-filter-design-methods

[90] Time and Frequency resolution of the FFT, *Choosing the right window size* [online]. Available from:https://support.ircam.fr/docs/AudioSculpt/3.0/co/The%20Right%20FFT.html#:~:text=A%20wide%20window%20gives%20better,frequency%20definition%20of%20the%20analysis.

[91] SCUDERI, Giles R., et al. The new knee society knee scoring system. *Clinical Orthopaedics and Related Research®*, 2012, vol. 470, no 1, p. 3-19.

[92] Software UBM SPSS [Online]. IBM. Available from: https://www.ibm.com/es-es/analytics/spss-statistics-software

[93] Portatil para juegos G15 [online]. DELL. Available from: https://www.dell.com/es-es/shop/laptops/396-cm-15/spd/g-series-15-5520-laptop/cn55121sc

[94] OuterVision Power supply calculator [online]. Extreme Outer vision. Available from: https://outervision.com/power-supply-calculator

[95] Calcula y compensa tus emisiones de CO2 [online]. CeroCO2. Available from: https://www.ceroco2.org/calculadoras/

[96] UNE-EN 60601-1:2018. *Medical electrical equipment -- Part 1-8: General requirements for basic safety and essential performance - Collateral Standard: General requirements, tests and guidance for alarm systems in medical electrical equipment and medical electrical systems* [online]. UNE Normalización Española. Available from: https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma?c=N0041618

[97] UNE-EN 60601-1-2:2015. *Medical electrical equipment - Part 1-2: General requirements for basic safety and essential performance - Collateral Standard: Electromagnetic disturbances - Requirements and tests (Endorsed by AENOR in November of 2015)* [online]. NE Normalización Española. Available from: https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma/?c=N0055535

[98] UNE-EN 60601-1-6:2010. *Medical electrical equipment -- Part 1-6: General requirements for basic safety and essential performance - Collateral standard: Usability* [online]. UNE Normalización Española. Available from: https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma/?c=norma-une-en-60601-1-6-2010-n0046353#:~:text=1%2D6%3A2010-,Equipos%20electrom%C3%A9dicos.,Norma%20colateral%3A%20Aptitud%20de%20uso.

[99] UNE-EN 61000-4-2:2010. *Electromagnetic compatibility (EMC) -- Part 4-2: Testing and measurement techniques - Electrostatic discharge immunity test* [online]. UNE Normalización Española. Available from: https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma?c=N0046324

[100] UNE-EN 61000-4-3:2007. *Electromagnetic compatibility (EMC) -- Part 4-3: Testing and measurement techniques - Radiated, radio-frequency, electromagnetic field immunity test (IEC 61000-4-3:2006)* [online]. UNE Normalización Española. Available from: https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma/?c=N0039175

[101] UNE-EN 61000-4-8:2011. *Electromagnetic compatibility (EMC) -- Part 4-8: Testing and measurement techniques - Power frequency magnetic field immunity test* [online]. UNE Normalización Española. Available from: https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma?c=N0046802

[102] UNE-EN 62366-1:2015/A1:2020. *Medical devices - Part 1: Application of usability engineering to medical devices (Endorsed by Asociación Española de Normalización in September of 2020.)* [online]. UNE normalización Española. Available from: https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma/?c=N0064517#:~:text=%3A2020%20(Ratificada)-,Productos%20sanitarios.,AENOR%20en%20junio%20de%202015.)

[103] ISO 14971:2019. *Medical devices. Application of risk management to Medical devices*. ISO (International Organization for Standarization) [online]. Available from: https://tienda.aenor.com/norma-iso-14971-2019-072704?_gl=1*14t3kq0*_up*MQ..&gclid=CjwKCAjw4ayUBhA4EiwATWyBrh0_cCFb7xtYswMbjEmq_FHM7P1GGJhyRlE7X7hGkJjnmKCSAdyIOxoCXq4QAvD_BwE&gclsrc=aw.ds

[104] UNE-EN ISO 13485:2016. *Medical devices - Quality management systems - Requirements for regulatory purposes (ISO 13485:2016)* UNE [online]. ISO (International Organization for Standardization). Available from: https://tienda.aenor.com/norma-une-en-iso-13485-2016-n0056668?gclid=CjwKCAjw4ayUBhA4EiwATWyBrh0_cCFb7xtYswMbjEmq_FHM7P1GGJhyRlE7X7hGkJjnmKCSAdyIOxoCXq4QAvD_BwE&gclsrc=aw.ds

[105] UNE-EN 62304:2007. Medical device software - Software life-cycle processes (IEC 62304:2006). Software de dispositivos médicos. Procesos del ciclo de vida del software [online]. UNE Normalización Española. Available from: https://tienda.aenor.com/norma-une-en-62304-2007-n0038684?_gl=1*sk4qhk*_up*MQ..&gclid=CjwKCAjw4ayUBhA4EiwATWyBrh0_cCFb7xtYswMbjEmq_FHM7P1GGJhyRlE7X7hGkJjnmKCSAdyIOxoCXq4QAvD_BwE&gclsrc=aw.ds

[106] Medical software applications. *HIPAA Compliance for Medical Software Applications* [online]. HIPAA Journal . Available from: https://www.hipaajournal.com/how-does-gdpr-apply-to-medical-devices/

[107] Software in a Medical Device. *From basic to lifesaving: when software becomes a medical device.* Voler Systems. Available from: https://www.volersystems.com/blog/from-basic-to-lifesaving-when-software-becomes-a-medical-device

[108] Régimen económico. Precio orientativo, *Prácticas académicas externas* [online]. Universitat Politècnica de Catalunya Barcelonatech. Available from: https://www.upc.edu/cce/es/estudiantes/practicas-academicas-externas

[109] Evolución precio de la luz en España [online]. Selectra. Available from: kWhhttps://selectra.es/energia/info/que-es/precio-kw

Annex A. Code

In this annex, the code developed in this thesis is exposed. This code has been programmed following section 7. It is important to state that only the code designed in this thesis is included, therefore, the code generated by STM32CubeMX to initialize all the board peripherals is not exposed. Additionally, the code corresponding to the different libraries used (previously explained) is also left out.

C Source files code

Main

```c
/*
This is the main file of a final degree thesis project called:
"FIRMWARE DESIGN OF A PORTABLE MEDICAL DEVICE TO MEASURE
THE QUADRICEPS MUSCLE GROUP AFTER A TOTAL KNEE ARTHROPLASTY BY EMG,
LBIA AND CLINICAL SCORE METHODS."

Arnau Diez Clos
EEBE, 2022
*/

/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "dma.h"
#include "spi.h"
#include "tim.h"
#include "usart.h"
#include "gpio.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "stdio.h"  //This is for snprintf function
#include <string.h> //This is required for strlen function
#include "math.h"   //This is required for the AMG and LBIA processing
#include "emg.h"    //This is required for the EMG processing
#include "lbias.h"  //This is required for the LBIA processing
#include "Battery.h"//This is required for the reading of the battery
#include "BLE.h"    //This is required for the BLE connections
#include "LEDS.h"   //This is required for the LEDs control
#include "ad73894.h"//This is required for thr AD73894

/* Definitions ---------------------------------------------------------------*/
//Device modes
#define WAITING_MODE    0
#define LBIA_MEASUREMENT 1
#define LBIA_START      2
#define LBIA_R_START    3
#define LBIA_Xc_START   4
#define EMG_MEASUREMENT 5
#define MVC_START       6
#define EMG_START       7
#define READ_BATTERY    8
#define READ_ERRORS     9

//LEDS (To turn ON and OFF the LEDs)
#define GREEN_LED_ON HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_SET)
#define GREEN_LED_OFF HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_RESET)
#define ORANGE_LED_ON HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_SET)
#define ORANGE_LED_OFF HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_RESET)
#define RED_LED_ON HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_SET)
#define RED_LED_OFF HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET)
#define BLUE_LED_ON HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_SET)
#define BLUE_LED_OFF HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_RESET)

//Acquisition functions
#define START_LBIA_ACQUISITION HAL_TIM_Base_Start(&htim12);
#define STOP_LBIA_ACQUISITION HAL_TIM_Base_Stop(&htim12);
#define START_MVC_EMG_ACQUISITION HAL_TIM_Base_Start(&htim9);
#define STOP_MVC_EMG_ACQUISITION HAL_TIM_Base_Stop(&htim9);
#define START_LBIA_DAC HAL_TIM_Base_Start(&htim6);
#define STOP_LBIA_DAC HAL_TIM_Base_Stop(&htim6);

//Memory Buffers lengths
#define LBIAS_LENGTH 50            //LBIA memory buffer length 10 PERIODS (>8 required periods)
#define MVC_LENGTH 30000           //MVC memory buffer length
#define EMG_LENGTH 50000           //EMG memory buffer length
#define uart3maxlen 30             //Buffer Size of Uart3str from handling uart3
#define SINEWAVE_LENGTH 480        //SineWave lenght


//Constants
#define BATT_Max_Discharge_Cap 1200   //Max Battery Discharge capacity
#define Vref 3.33                     //ADC Vref (TO CHANGE)
#define Minute 60000                  //Minute in ms
#define s 1000                        //Second in ms


```

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

```c
/* Variables ---------------------------------------------------------------*/
//GENERAL USE VARIABLES
uint8_t device_mode;                        //Indicates in which state the device is
uint8_t MVC_done=0;                         //Indicates if the MVC has been done already
double Battery_level=0;                     //Indicates the current battery level

//ERROR VARIABLES
uint8_t BLE_error_no_uart_flag;             //Flag for the BLE connection error no uart
uint8_t BLE_error_no_name_flag;             //Flag for the BLE connection error no name
uint8_t BLE_error_no_advt_flag;             //Flag for the BLE connection error no advertising
uint8_t BLE_error_no_mode_flag;             //Flag for the BLE connection error no mode
uint8_t BLE_error_not_connected_flag = 0;   //Flag for the BLE external device connection error


uint8_t LBIA_error_SAT_V=0;                 //Flag error to inform that the LBIA V samples were saturated
uint8_t LBIA_error_SAT_I=0;                 //Flag error to inform that the LBIA I samples were saturated
uint8_t LBIA_error_I_electrode_off = 0;     //Flag error to inform that the LBIA I electrodes have lost contact
uint8_t LBIA_error_phase_angle_I=0;         //Flag error to inform that the obtained phase angle does not make sense

uint8_t EMG_error_CH1_Saturated_flag = 0;   //Flag to inform that the EMG acquisition went wrong with ch1
uint8_t EMG_error_CH2_Saturated_flag = 0;   //Flag to inform that the EMG acquisition went wrong with ch2
uint8_t EMG_error_CH3_Saturated_flag = 0;   //Flag to inform that the EMG acquisition went wrong with ch3

uint8_t MVC_error_CH1_Saturated_flag = 0;   //Flag to inform that the MVC acquisition went wrong with ch1
uint8_t MVC_error_CH2_Saturated_flag = 0;   //Flag to inform that the MVC acquisition went wrong with ch2
uint8_t MVC_error_CH3_Saturated_flag = 0;   //Flag to inform that the MVC acquisition went wrong with ch3

uint8_t AD73894_Config_1_error_flag=0;      //Flag to inform that the configure 1 register of AD73894 was not writen correctly
uint8_t AD73894_Config_2_error_flag=0;      //Flag to inform that the configure 1 register of AD73894 was not writen correctly

uint8_t low_battery_flag=0;                 //Flag to inform that the device battery was below 10%


//Auxiliar Flags
uint8_t LBIA_V_Saturated_flag=0;            //The tension acquisition is saturated flag
uint8_t LBIA_I_Saturated_flag=0;            //The current acquisition is saturated flag
uint8_t LBIA_I_disconnected_flag=0;         //Current electrodes lost contact
uint8_t LBIA_ADC_wrong_phase_flag = 0;      //The obtained phase angle does not make sense
uint8_t device_connected_flag=0;            //Flag to inform that a device is found

uint8_t AD73894_conf1_failed_flag=0;        //Flag to inform that the configure 1 register of AD73894 was not writen correctly
uint8_t AD73894_conf2_failed_flag=0;        //Flag to inform that the configure 1 register of AD73894 was not writen correctly
uint8_t AD73894_Init_OK_flag=0;             //Flag initializing process of the AD73894 was done correctly.

//BLE CONNECTION
uint8_t RX_DATO;                            //Auxiliar variable for the BLE data receiving process
char uart3str[uart3maxlen];                 //Buffer definitions

//ADC CONVERSION VARIABLES (EMG)
uint32_t adc_counter = 0;                   //Counter for the Buffer lenght
uint8_t MVC_acquisition_done_flag=0;        //Indicates if the MVC has been done already
uint8_t EMG_acquisition_done_flag=0;        //Indicates if the EMG has been done already
uint8_t EMG_CH1_Saturated_flag = 0;         //Flag to inform that the acquisition went wrong with ch1
uint8_t EMG_CH2_Saturated_flag = 0;         //Flag to inform that the acquisition went wrong with ch2
uint8_t EMG_CH3_Saturated_flag = 0;         //Flag to inform that the acquisition went wrong with ch3
uint16_t saturated_samples_1 = 0;           //Counter of the saturated samples for CH1
uint16_t saturated_samples_2 = 0;           //Counter of the saturated samples for CH2
uint16_t saturated_samples_3 = 0;           //Counter of the saturated samples for CH3

//Battery related variables
uint16_t ADC3_data=0;                       //Variable where the ADC3 value will be stored
```

```
double Battery_tension = 0;                  //Double to store the battery tension level

//***************************************************************************
//EMG PROCESSING FUNCTION
//MVC means for CH1/CH2/CH3
float MVC_mean_1=0;
float MVC_mean_2=0;
float MVC_mean_3=0;

//Standart amplitude parameters
//RMS for CH1/CH2/CH3
float EMG_RMS_1=0;
float EMG_RMS_2=0;
float EMG_RMS_3=0;

//MAV for CH1/CH2/CH3
float EMG_MAV_1=0;
float EMG_MAV_2=0;
float EMG_MAV_3=0;

//Frequency domain parameters
//Median f for CH1/CH2/CH3
float EMG_Med_f_1=0;
float EMG_Med_f_2=0;
float EMG_Med_f_3=0;

//Mean f for CH1/CH2/CH3
float EMG_Mean_f_1=0;
float EMG_Mean_f_2=0;
float EMG_Mean_f_3=0;

//***************************************************************************
//FFT Related
//External Input and Output buffer Declarations for FFT Bin Example

static float32_t fftOutput[EMG_LENGTH/2];

//Global variables for FFT Bin
uint32_t fftSize = 8192;
uint32_t ifftFlag = 0;
uint32_t doBitReverse = 1;
arm_cfft_instance_f32 varInstCfftF32;

float median_frq=0;                           //Where the median f is stored
float mean_frq=0;                             //Where the mean f is stored

//***************************************************************************
//LBIA ACQUISITION AND PROCESSING FUNCTION
uint8_t LBIA_ADC_buff_full_flag=0;            //It informs that the acquisiton has been completed
float LBIAS_R = 0;                            //Real component of the bioimpedance
float LBIAS_Xc = 0;                           //Capacitive component of the bioimpedance
float LBIAS_phase = 0;                        //Bioimpedance phase
uint8_t Xc_var=0;                             //Used to differenciate between sig0 and sig90


int adcChannelCount = sizeof(LBIA_Acquisition_Value)/sizeof(LBIA_Acquisition_Value[0]);
uint8_t LBIA_Buff_index=0;                    //Used to indicate the current position of the Buffer
double LBIA_Current=0;                        //LBIA DC component of the injected current

uint8_t LBIA_acquisition_done_flag=0;         //Indicates if the LBIA has been done already
uint8_t LBIA_acq_started=0;                   //Used to check if the TIM12 has already started

//DAC sine waveform
uint32_t sine_wave_array[SINEWAVE_LENGTH];

//Filter coeficients
float filt_coef[FT_ORDER] = {0.0357142844990487,  0.241071427982274,  0.446428575037355,
0.241071427982274, 0.0357142844990487};       //Low Pass Filter
float bandpass_filt_coef[BP_ORDER] = {0,  -0.0579204230134285,  -0.200390444267026,
0.131704474544405,  0.500646395337873,  0.131704474544405,  -0.200390444267026,
-0.0579204230134285,  0};

/* Memory Buffers -----------------------------------------------------------------*/
//LBIA (1 channel)
uint16_t LBIAs_CH1_data[LBIAS_LENGTH];

//MVC (3 channels)
```

```c
uint16_t MVC_CH1_data[MVC_LENGTH];
uint16_t MVC_CH2_data[MVC_LENGTH];
uint16_t MVC_CH3_data[MVC_LENGTH];

//EMG (3 channels)
uint16_t EMG_CH1_data[EMG_LENGTH];
uint16_t EMG_CH2_data[EMG_LENGTH];
uint16_t EMG_CH3_data[EMG_LENGTH];

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */
void TURN_DEVICE_OFF(void);                              //Turns device to STOP mode


/******************************************************************************/

/*Main -------------------------------------------------------*/
int main_algorithm_process(void)
{
//****************************************************************************
  //INITIALIZE UART
  HAL_UART_Receive_DMA (&huart3, (uint8_t *)&uart3str, 10);// this is required to get ready to new fame


//GENERATES AND STORES THE SINE WAVE USED BY THE DAC
  dac_sinewave();


//****************************************************************************
  //INITIALIZE EXTERNAL ADC
  init_AD7389();                                         //The adc is initialized

  if (AD73894_conf1_failed_flag==1)                      //If the configuration 1 was not writen properly
    {
      AD73894_conf1_failed_flag=0;                       //Resets the flag
      AD73894_Config_1_error_flag=1;                     //The correspondent error flag is activated
      RED_led_blinking();                                //Blink the red light and put the device to sleep
      TURN_DEVICE_OFF();                                 //The device is shut down
    }
  if (AD73894_conf2_failed_flag==1)                      //If the configuration 2 was not writen properly
    {
      AD73894_conf2_failed_flag=0;                       //Resets the flag
      AD73894_Config_2_error_flag=1;                     //The correspondent error flag is activated
      RED_led_blinking();                                //Blink the red light and put the device to sleep
      TURN_DEVICE_OFF();                                 //The device is shut down
    }

//****************************************************************************
  //BLE IS INITIALIZED
  uint8_t BLE_Init_OK = BLE_init();
  if (BLE_Init_OK != 1)                                  //If it is not initialized correctly
  {
    RED_led_blinking();                                  //Blink the red light and put the device to sleep
    TURN_DEVICE_OFF();                                   //The device is shut down
  }

//****************************************************************************
  //EXTERNAL DEVICE IS CONNECTED TO THE MEDICAL DEVICE
  uint8_t device_connected=wait_for_BLE_connection(Minute); //Wait for one minute to the connection of the external device
  if(device_connected==0)                                //If no device is connected correctly
  {
    RED_led_blinking();                                  //Blink the red light and put the device to sleep
    TURN_DEVICE_OFF();                                   //The device is shut down
  }
  else
```

```c
  {
    BLUE_LED_ON;                                        //Turns ON the blue LED
  }

//****************************************************************************
  //CHECK THE BATTERY LEVEL

  Read_battery_level();                                 //It reads the battery level
  if (Battery_level < 15)                               //If the battery is below 15%
  {
    low_battery_flag=1;                                 //It activates the low battery flag
    RED_LED_ON;                                         //Turns ON the RED LED
    HAL_Delay(20*s);                                    //The red light is ON for 20s
    TURN_DEVICE_OFF();                                  //Turns OFF the device
  }
  else if ((Battery_level  >= 15) && (Battery_level < 40))  //If the battery is between 14 an 40%
  {
    ORANGE_LED_ON;                                      //Turns ON the ORANGE LED
  }


//****************************************************************************
  //THE DEVICE IS SET IN WAITING MODE
  uint16_t time = 0;                                    //The time starts at 0
  device_mode = WAITING_MODE;                           //It starts the device as waiting mode

//****************************************************************************
//DIFFERENT DEVICE MODES
//If none of the modes is activated (no command is received, put the device in stop mode)
  while (1)
  {
    time = 0;                                           //It restarts the time variable

    //****************************************************************************
    //LBIA MEASUREMENT MODE
    //It waits one minute for the start command or it turns off the device
    if (device_mode==LBIA_MEASUREMENT) //If the current device mode is LBIA measurement
    {
      while (time < Minute) //For one minute
      {
        if (device_mode == LBIA_START||device_mode == LBIA_R_START||device_mode == LBIA_Xc_START)
        {
          if (LBIA_acq_started==0)                      //IF the DAC and ADC has not started
          {
            GREEN_LED_ON;                               //Turns ON the GREEN LED
            START_LBIA_DAC;                             //Start DAC
            START_LBIA_ACQUISITION;                     //Performs the LBIA acquisition
            LBIA_acq_started=1;                         //It indicates that the LBIA ADC/DAC were iniciated
          }
          if (LBIA_V_Saturated_flag==1||LBIA_V_Saturated_flag==1)
          {
            STOP_LBIA_ACQUISITION;                      //Stops the LBIA acquisition
            STOP_LBIA_DAC                               //Stops the DAC signal generation
            GREEN_LED_OFF;                              //Turns OFF the GREEN LED
            RED_led_blinking();                         //The red LED blinks to incate error
            LBIA_acq_started=0;                         //Resets the indicator
            if (LBIA_V_Saturated_flag==1)               //If the Voltage ADC failed
            {
              LBIA_V_Saturated_flag=0;                  //Resets the flag
              LBIA_error_SAT_V=1;                       //Flag error to inform that the LBIA V samples were saturated
              device_mode=LBIA_MEASUREMENT;             //The acquisition needs to be done again
            }
            else                                        //If the Current ADC failed
            {
              LBIA_I_Saturated_flag=0;                  //Resets the flag
              LBIA_error_SAT_I=1;                       //Flag error to inform that the LBIA I samples were saturated
              device_mode=LBIA_MEASUREMENT;             //The acquisition needs to be done again
            }
          }
          if (LBIA_ADC_buff_full_flag==1)               //If the conversion is done
          {
            STOP_LBIA_ACQUISITION;                      //Stops the LBIA acquisition
            STOP_LBIA_DAC                               //Stops the DAC signal generation
            LBIA_acq_started=0;                         //Resets the indicator
```

```
            LBIA_ADC_buff_full_flag=0;                  //It resets the buffer full flag
            GREEN_LED_OFF;                              //Turns OFF the GREEN LED
            LBIAS_demodulation();                       //It does the demodulation
            if (LBIA_I_disconnected_flag==1)            //Current electrode has lost contact
            {
              LBIA_I_disconnected_flag=0;               //Resets the flag
              LBIA_error_I_electrode_off=1;             //Flag error to inform that the LBIA I electrodes have lost contact
              RED_led_blinking();                       //Red LED Blinks for 5s
              device_mode=LBIA_MEASUREMENT;             //The acquisition needs to be done again
            }

            else if (LBIA_ADC_wrong_phase_flag == 1)    //If the obtained phase angle is erroneus
            {
              LBIA_ADC_wrong_phase_flag = 0;            //Resets the flag
              LBIA_error_phase_angle_I=1;               //Activates the error flag
              RED_led_blinking();                       //Red LED Blinks for 5s
              device_mode=LBIA_MEASUREMENT;             //The acquisition needs to be done again
            }
            else
            {
              GREEN_led_blinking();                     //Green LED blinks for 5s
              BLE_Send_results();                       //Sends the LBIA results to the external device
              device_mode = WAITING_MODE;               //It sets the device mode to waiting mode
              time=Minute;                              //It gets out of the loop
            }
          }
        }
        else
        {
          HAL_Delay(0.1*s);                             //0.1s delay
          time++;                                       //It increases it by one
        }
      }
      if (device_mode !=WAITING_MODE)
      {
        TURN_DEVICE_OFF();                              //Turns off the device
      }
    }

    //****************************************************************************
    //EMG MEASUREMENT MODE
    //It waits one minut for the start command or it turns off the device
    else if (device_mode == EMG_MEASUREMENT) //If the current device mode is EMG measurement
    {
      while (time <Minute) //For one minute
      {
        if (device_mode == MVC_START)                   //If an start MVC command has been received
        {
          GREEN_LED_ON;                                 //Turns ON the GREEN LED
          START_MVC_EMG_ACQUISITION;                    //It starts the MVC acquisition
          if (EMG_CH1_Saturated_flag == 1||EMG_CH2_Saturated_flag==1||EMG_CH3_Saturated_flag==1) //If the samples saturated
          {
            STOP_MVC_EMG_ACQUISITION;                   //The acquisition is stoped
            GREEN_LED_OFF;                              //Turns OFF the GREEN LED
            RED_led_blinking();                         //The RED LED blinks for 5s
            if (EMG_CH1_Saturated_flag==1)              //If the CH1 samples are saturated
            {
              EMG_CH1_Saturated_flag=0;                 //The flag is reseted
              MVC_error_CH1_Saturated_flag=1;           //The error flag is activated
            }
            else if (EMG_CH2_Saturated_flag==1)         //If the CH2 samples are saturated
            {
              EMG_CH2_Saturated_flag=0;                 //The flag is reseted
              MVC_error_CH2_Saturated_flag=0;           //The error flag is activated
            }
            else                                        //If the CH3 samples are saturated
            {
              EMG_CH3_Saturated_flag=0;                 //The flag is reseted
              MVC_error_CH3_Saturated_flag=1;           //The error flag is activated
            }
            time = 0;                                   //Restarts the counter
            break;                                      //Waits again for a command
          }
```

```
          else if (MVC_acquisition_done_flag == 1)          //If the acquisition has been completed
          {
            mean_MVC();                                     //It does the mean of the MVC values
            GREEN_LED_OFF;                                  //Turns OFF the GREEN LED
            MVC_done = 1;                                   //It informs that the MVC has been done
            time = 0;                                       //It restarts the minute timer
          }
        }
        else if (device_mode == EMG_START)  //If an start EMG command has been received
        {
          GREEN_LED_ON;                                     //Turns ON the GREEN LED
          START_MVC_EMG_ACQUISITION;                        //It starts the MVC acquisition
          if (EMG_CH1_Saturated_flag == 1||EMG_CH1_Saturated_flag==1||EMG_CH1_Saturated_flag==1) //If the samples saturated
          {
              STOP_MVC_EMG_ACQUISITION;                     //The acquisition is stoped
              GREEN_LED_OFF;                                //Turns OFF the GREEN LED
              RED_led_blinking();                           //The RED LED blinks for 5s
              if (EMG_CH1_Saturated_flag==1)                //If the CH1 samples are saturated
              {
                EMG_CH1_Saturated_flag=0;                   //The flag is reseted
                EMG_error_CH1_Saturated_flag=1;             //The error flag is activated
              }
              else if (EMG_CH2_Saturated_flag==1)           //If the CH2 samples are saturated
              {
                EMG_CH2_Saturated_flag=0;                   //The flag is reseted
                EMG_error_CH2_Saturated_flag=1;             //The error flag is activated
              }
              else                                          //If the CH3 samples are saturated
              {
                EMG_CH3_Saturated_flag=0;                   //The flag is reseted
                EMG_error_CH3_Saturated_flag=1;             //The error flag is activated
              }
              time = 0;                                     //Restarts the counter
              break;                                        //Waits again for a command
          }
          else if (EMG_acquisition_done_flag == 1)            //If the acquisition has been completed
          {
            STOP_MVC_EMG_ACQUISITION;                       //The acquisition is stoped
            GREEN_LED_OFF;                                  //Turns OFF the GREEN LED
            EMG_processing();                               //The data is processed
            GREEN_led_blinking();                           //GREEN LED blinks for 5s
            BLE_Send_results();                             //Sends the EMG results to the external device
            device_mode= WAITING_MODE;                        //It sets the device mode to Waiting mode
            time = Minute;                                  //Get out of the while loop
          }
        }
        else
        {
          HAL_Delay(0.1*s);                                 //0.1s delay
          time++;                                           //It increases it by one
        }
      }

      if (device_mode !=WAITING_MODE)
      {
        TURN_DEVICE_OFF();                                  //Turns off the device
      }
    }

    //*****************************************************************************
    //READ BATTERY MODE
    //Sends the battery level value to the external device
    else if (device_mode == READ_BATTERY)                   //If the current device mode is read_battery
    {
      Read_battery_level();                                 //It reads the current battery
      GREEN_led_blinking();                                 //GREEN LED blinks for 5s
      BLE_Send_results();                                   //Sends the Battery level to the external device
      device_mode= WAITING_MODE;                            //It sets the device mode to Waiting mode
    }


    //*****************************************************************************
```


UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

```
    //READ FLAG ERRORS MODE
    //Sends the battery level value to the external device
    else if (device_mode==ERROR)                            //if the device is in error mode
    {
      GREEN_led_blinking();                                 //GREEN LED blinks for 5s
      BLE_Send_results();                                   //Sends the Error flags to the external device
      device_mode= WAITING_MODE;                            //It sets the device mode to Waiting mode
    }


    //****************************************************************************
    //WAITING MODE
    //This mode acts as a countdown to turn off the device if no command is received
    // And checks the battery
    else if (device_mode == WAITING_MODE) //If the mode selected is the WAITING mode
    {
      while (time<Minute)
      {
        if (device_mode == WAITING_MODE)                    //If it is not changed
        {

          Read_battery_level();                             //It reads the battery level
          if (Battery_level < 15)
          {
            low_battery_flag=1;                             //It activates the low battery flag
            RED_LED_ON;                                     //Turns ON the RED LED
            HAL_Delay(20*s);                                //The red light is ON for 20s
            TURN_DEVICE_OFF();                              //Turns OFF the device
          }
          else if ((Battery_level  >= 15) && (Battery_level < 40))  //If the battery is between 14 an 40%
          {
            ORANGE_LED_ON;                                  //Turns ON the ORANGE LED
          }

          else
          {
            ORANGE_LED_OFF;                                 //Turns OFF the ORANGE LED if it was ON
          }
          HAL_Delay(0.1*s);
          time++;
        }
        else
        {
          time=Minute;                                      //get out of the while loop
        }
      }
      if (device_mode == WAITING_MODE)                      //If a command has not been received and the time has passed
      {
        TURN_DEVICE_OFF();                                  //Turns off the device
      }
    }
  }
}


/*Auxiliar Functions ----------------------------------------------------------*/
//****************************************************************************
//TURNING OFF DEVICE (STOP MODE)
//This function puts the MUC to STOP mode and all the peripherals to sleep mode
void TURN_DEVICE_OFF(void)
{
  //Putting the BLE to SLEEP mode
  char Str[50];                                             //Buffer to send the command to the BLE
  snprintf(Str,sizeof(Str),"AT+SLEEP");                     //It sends to the BLE module the SLEEP mode comand
  BLE_Send_function((uint8_t*)Str,strlen(Str));             // Sends the string as uint_8
  HAL_Delay(10);                                            // some delay for the process time of BLE chip

  //Putting the external ADC to sleep mode
  shutdown_AD7389();                                        //The ADC is put to shut down mode

  //Putting the MUC to SLEEP mode
  HAL_PWR_EnterSLEEPMode(PWR_LOWPOWERREGULATOR_ON, PWR_STOPENTRY_WFE); //It enters the Sleep Mode
```

```
}
//*****************************************************************************
//TIMERS FOR DAC and ADCS Conversion

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    //Timer 9 (TIM9) interruption used for the EMG ADC acquisition
    if (htim->Instance == TIM9)
    {
     // 12khz
      TIMER9IntHandler();
    }
    //Timer 12 (TIM12) interruption used for the LBIA ADC acquisition
    else if (htim->Instance == TIM12)
    {
     // 250 kHz
      TIMER12IntHandler();
    }
    //Timer 6 (TIM6) interruption used for the LBIA DAC generation
    else if (htim->Instance == TIM6)
    {
     // 50 kHz
     TIMER6IntHandler();
    }
}

//*****************************************************************************
//UART interruption service routine (BLE Data is received)
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    if(huart->Instance==USART3)                               //Only if its UART3
  {
    if (device_connected_flag==1)                             //If the device has been connected
    {
      CommandEvalFunc(uart3str);                              //Evaluate the received command
      ble_uart_clear();                                       //Clear
    }
    HAL_UART_Receive_DMA(&huart3, (uint8_t*) uart3str, 6);  //Needed for the following received data
    /*The external device should send the command like this: "EMG000"*/
  }
}
```

Battery level estimation

```

//Include

#include "Battery.h"

//READ BATTERY
//It does an estimation of the Battery level with the Battery tension
//The Battery model in which this algorithm is based is the LPP 503562 S
void Read_battery_level(void)                                   //This function reads the battery value (gives back the proportion of battery left in %)
{
  double discharge_capacity=0;                                  //Double to store the correspondant discharge capacity of each tension
  Battery_ADC_Acquisiton();                                     //It calls the function to do the conversion
  ADC3tofloat();                                                //It converts the acquired data to float

  //Zone 1
  if (Battery_tension > 3.96)                                   //If it is in zone 1 of Voltage/Discharge capacity graph of the battery
  {
    discharge_capacity=(-(Battery_tension-4.2)/0.0012);  //Corresponding zone 1 equation
    Battery_level=(BATT_Max_Discharge_Cap-discharge_capacity/BATT_Max_Discharge_Cap)*100;   //It gets the precentage of battery left
  }
  //Zone 2
  else if (3.96>=Battery_tension > 3.84)                        //If it is in zone 2 of Voltage/Discharge capacity graph of the battery
  {
    discharge_capacity=(-(Battery_tension-3.96)/0.0012)+240;  //Corresponding zone 2 equation
    Battery_level=(BATT_Max_Discharge_Cap-discharge_capacity/BATT_Max_Discharge_Cap)*100;   //It gets the precentage of battery left
  }
  //Zone 3
  else if (3.84>=Battery_tension > 3.68)                        //If it is in zone 3 of Voltage/Discharge capacity graph of the battery
  {
    discharge_capacity=(-(Battery_tension-3.84)/0.00035)+495;  //Corresponding zone 3 equation
    Battery_level=(BATT_Max_Discharge_Cap-discharge_capacity/BATT_Max_Discharge_Cap)*100;   //It gets the precentage of battery left
  }
  //Zone 4
  else if (3.68>=Battery_tension > 3.63)                        //If it is in zone 4 of Voltage/Discharge capacity graph of the battery
  {
    discharge_capacity=(-(Battery_tension-3.68)/0.00057)+950;  //Corresponding zone 4 equation
    Battery_level=(BATT_Max_Discharge_Cap-discharge_capacity/BATT_Max_Discharge_Cap)*100;   //It gets the precentage of battery left
  }
  //Zone 5
  else if (3.63>=Battery_tension > 3.42)                        //If it is in zone 5 of Voltage/Discharge capacity graph of the battery
  {
    discharge_capacity=(-(Battery_tension-3.63)/0.00190)+1038;  //Corresponding zone 5 equation
    Battery_level=(BATT_Max_Discharge_Cap-discharge_capacity/BATT_Max_Discharge_Cap)*100;   //It gets the precentage of battery left
  }
  //Zone 6
  else                                                          //If it is in zone 5 of Voltage/Discharge capacity graph of the battery
  {
    discharge_capacity=(-(Battery_tension-3.42)/0.0084)+1150;  //Corresponding zone 6 equation
    Battery_level=(BATT_Max_Discharge_Cap-discharge_capacity/BATT_Max_Discharge_Cap)*100;   //It gets the precentage of battery left
  }
}


//ADC Conversion
void Battery_ADC_Acquisiton(void)
```

```
{
  //Read the ADC3 value
  HAL_ADC_Start(&hadc3);                                        //It starts the conversion
  if (HAL_ADC_PollForConversion(&hadc3, 5) == HAL_OK)           //When the conversion is done
  {
    ADC3_data = HAL_ADC_GetValue(&hadc3);                       //It stores the tension of the battery to this variable
  }
  HAL_ADC_Stop(&hadc3);                                         //After that it stops the conversion
  //100 ms delay
  HAL_Delay(100);                                               //Delay
}

//Converts the value to float
void ADC3tofloat(void)
{
  //It defines the variable to store the float value
  Battery_tension = ADC3_data*(Vref_ADC3/4095)*(4.2/Vref_ADC3); //Converts the acquired ADC3 data to float
}

```

AD7389-4

```c
/*
 * ad73894.c
 */

#include "ad73894.h"

//Abreviations for the code
ConfigurationRegister1 ConfReg1;
ConfigurationRegister2 ConfReg2;
REGISTERS AdRegister;
addressing address;


void init_AD7389(void)
{

  //This function initializes the AD7389 prior convertion
  //Configuration 1 Register
  address.wr=1;                         //WR to '1' to write in the register
  address.regaddress=Configuration_1;   //Adress  set to configuration 1 register

  ConfReg1.ADDRESSING=address;
  ConfReg1.OS_MODE=0;                   // Sampling set to normal mode (No need for oversampling)
  ConfReg1.OSR=0;                       // Disable over sampling
  ConfReg1.CRC_W=0;                     // No crc check for writing
  ConfReg1.CRC_R=0;                     // No crc check for reading
  ConfReg1.ALERT_EN=0;                  // Alert disabled
  ConfReg1.PMODE=0;                     // Normal mode

  uint8_t retStatus=adWriteRegister((uint8_t*)&ConfReg1,2); //The parameters are writen to the configuration 1 register

  if (retStatus==0)                     //If it has not been writen correctly to the configuration 1 register
  {
    AD73894_conf1_failed_flag=1;        //The corresponding flag is activated
  }

  if (AD73894_conf1_failed_flag==0)     //If the first register was writen correctly
  {
    //Configuration 2 Register
    address.wr=1;                       //WR to '1' to write in the register
    address.regaddress=Configuration_2; //Adress  set to configuration 1 register

    ConfReg2.ADDRESSING=address;
    ConfReg2.SDO= 1;                    //01: Conversion Results Serial Data Output set to 1-wire (SDOA only)

    retStatus=adWriteRegister((uint8_t*)&ConfReg2,2); //The parameters are writen to the configuration 1 register

    if (retStatus==0)                   //If it has not been writen correctly to the configuration 1 register
    {
      AD73894_conf2_failed_flag=1;      //The corresponding flag is activated
    }
    else                                //If the two configuration registers have been set correctly (Init done)
    {
      AD73894_Init_OK_flag=1;           //The corresponding flag is activated
    }
  }

}
void shutdown_AD7389(void)
{
  //This function puts sets the AD7389 to the shutdown mode
  //Configuration 1 Register
  address.wr=1;                         //WR to '1' to write in the register
  address.regaddress=Configuration_1;   //Adress  set to configuration 1 register

  ConfReg1.ADDRESSING=address;
```

```
  ConfReg1.OS_MODE=0;                      // Sampling set to normal mode (No need for oversampling)
  ConfReg1.OSR=0;                          // Disable over sampling
  ConfReg1.CRC_W=0;                        // No crc check for writing
  ConfReg1.CRC_R=0;                        // No crc check for reading
  ConfReg1.ALERT_EN=0;                     // Alert disabled
  ConfReg1.PMODE=1;                        // Shutdown mode.

  uint8_t retStatus=adWriteRegister((uint8_t*)&ConfReg1,2);

}


void TIMER9IntHandler(void)
{
  //ISR routine for the TIM9 (ADC Acquistion with 5kS/s of sample frequency
  uint16_t AD73894_ch1, AD73894_ch2, AD73894_ch3;                  //Channels
  Read_ADC_values(&AD73894_ch1, &AD73894_ch2, &AD73894_ch3);       //Reads the value from the ch1,ch2,ch3 and ch4


  //AVOIDING SAMPLE SATURATION FOR THE THREE CHANNELS
  //Chanel 1 saturation samples evaluation
  if (AD73894_ch1 >= EMG_Saturated)                    //If the channel 1 is saturated)
  {
    saturated_samples_1++;                             //The counter of saturated samples increases by one
    if (saturated_samples_1==Max_Saturated_samples)    //If the limit of saturated samples in a row has been met
    {
      EMG_CH1_Saturated_flag=1;                        //The corresponding flag is activated
    }
  }
  else
  {
    saturated_samples_1=0;                             //The counter is reseted
  }

  //Chanel 2 saturation samples evaluation
  if (AD73894_ch2 >= EMG_Saturated)                    //If the channel 2 is saturated)
  {
    saturated_samples_2++;                             //The counter of saturated samples increases by one
    if (saturated_samples_2==Max_Saturated_samples)    //If the limit of saturated samples in a row has been met
    {
      EMG_CH2_Saturated_flag=1;                        //The corresponding flag is activated
    }
  }
  else
  {
    saturated_samples_2=0;                             //The counter is reseted
  }

  //Chanel 3 saturation samples evaluation
  if (AD73894_ch3 >= EMG_Saturated)                    //If the channel 3 is saturated)
  {
    saturated_samples_3++;                             //The counter of saturated samples increases by one
    if (saturated_samples_3==Max_Saturated_samples)    //If the limit of saturated samples in a row has been met
    {
      EMG_CH3_Saturated_flag=1;                        //The corresponding flag is activated
    }
  }
  else
  {
    saturated_samples_3=0;                             //The counter is reseted
  }


  //FULLING THE CORRESPONDING BUFFERS (MVC OR EMG)

  //For the MVC acquisition
  if (device_mode==MVC_START)
  {
```

```
      //If the samples were not saturated they are stored in the buffers
      if  (EMG_CH1_Saturated_flag==0||EMG_CH2_Saturated_flag==0||EMG_CH3_Saturated_flag==0)
      {
        MVC_CH1_data[adc_counter]=AD73894_ch1;
        MVC_CH2_data[adc_counter]=AD73894_ch2;
        MVC_CH3_data[adc_counter]=AD73894_ch3;
      }
      adc_counter++;
      if(adc_counter>=MVC_LENGTH)                        //If the buffer is full
      {
        adc_counter=0;                                   //reset buffer pointer
        MVC_acquisition_done_flag=1;                     //The corresponding flag is activated
      }
    }

    //FULLING THE CORRESPONDING BUFFERS (MVC OR EMG)
    else if (device_mode==EMG_START)
    {
      //If the samples were not saturated they are stored in the buffers
      if  (EMG_CH1_Saturated_flag==0||EMG_CH2_Saturated_flag==0||EMG_CH3_Saturated_flag==0)
      {
        MVC_CH1_data[adc_counter]=AD73894_ch1;
        MVC_CH2_data[adc_counter]=AD73894_ch2;
        MVC_CH3_data[adc_counter]=AD73894_ch3;
      }
      adc_counter++;
      if(adc_counter>=MVC_LENGTH)                        //If the buffer is full
      {
        adc_counter=0;                                   //reset buffer pointer
        EMG_acquisition_done_flag=1;                     //The corresponding flag is activated
      }
    }
  }

  void Read_ADC_values(uint16_t *ch1,uint16_t *ch2,uint16_t *ch3)
  {
    //This function reads the data from the 4 channels
    uint8_t  RxData[6];                                  //Variable needed for the ADCSpiSendReceiveArray function (2 bytes/chan · 3 chan = 6 bytes)
    uint8_t  TxData[6];                                  //Variable needed for the ADCSpiSendReceiveArray function (2 bytes/chan · 3 chan = 6 bytes)
    AdcSpiSendReceiveArray(TxData,RxData,6);             //It calls the function to get the data (2 bytes/chan · 3 chan = 6 bytes)

    //The data from the two bits (of each channel) is converted to uin16_t
    // 0 is the LSB and the 1 IS MSB => uint16_t type
    *ch1=(uint16_t)RxData[0]>>8 | (uint16_t)RxData[1];
    *ch2=(uint16_t)RxData[2]>>8 | (uint16_t)RxData[3];
    *ch3=(uint16_t)RxData[4]>>8 | (uint16_t)RxData[5];

  }

  //Function to write to the register
  uint8_t adWriteRegister(uint8_t *dataTx,uint16_t Size)
  {
    //This function writes to the register via SPI and checks if it was done correctly
    HAL_StatusTypeDef hal_ret;

    hal_ret = HAL_SPI_Transmit(&hspi1, dataTx,  Size, 10) ; // 10ms timeout

    //If it was writen correctly, the function returns 1, if not the function returns 0
    if(hal_ret==HAL_OK)
      return 1;
    else
      return 0;

  }

  //SPI FUNCTIONS
  //*******************************************************************************
  uint8_t AdcSpiSendReceiveByte(uint8_t dataTx)
  {
```

```c
    //Sends SPI byte on MOSI pin and captures MISO return byte value.

    AD_CSL;                                 //Chif select to low

    HAL_SPI_Receive(&hspi1, &dataTx,1,10); //with 10ms timeout

    uint8_t dataRx=dataTx;

    AD_CSH;                                 //Chif select to high

    return dataRx;
}


void AdcSpiSendReceiveArray(uint8_t dataTx[], uint8_t dataRx[], uint8_t byteLength)
{
// Sends SPI byte array on MOSI pin and captures MISO data to a byte array.
    int i;
    for (i = 0; i < byteLength; i++)
    {
        dataRx[i] = AdcSpiSendReceiveByte(dataTx[i]);
    }
}
```

BLE connection

```

//Include

#include "BLE.h"


//Function to clear the buffer
void ble_uart_clear(void)
{
  memset(uart3str,0,uart3maxlen);                                     // clearing the BLE Buffer
}

//Function to send data to the BLE module
void BLE_Send_function(uint8_t *txData, uint8_t txDataSize)
{
  HAL_UART_Transmit_IT(&huart3,txData,txDataSize);
}


//Initiate the BLE device
int8_t BLE_init(void)
{
  char Str[50];

  GPIO_PinState ret;
  ret=HAL_GPIO_ReadPin(GPIOE,GPIO_PIN_15);                  //STATE PIN of the BLE device to see if an external device connected

  if(ret==GPIO_PIN_RESET)                                   //Only if not device is connected
  {
    //Testingg the connection
    ble_uart_clear();
    snprintf(Str,sizeof(Str),"AT");                         //Sends AT to test the connection
    BLE_Send_function((uint8_t*)Str,strlen(Str));
    HAL_Delay(10);
    if(!strcmp(uart3str,"OK"))                              //If it receives OK, the connection is stablished correctly
    {
      ble_uart_clear();                                     //Clear
    }
    else
    {
      BLE_error_no_uart_flag=1;                             //If not, it activates the error flag
      return 0;
    }
    //Set Module name
    snprintf(Str,sizeof(Str),"AT+NAMETFG_ARNAU_DIEZ");      //It sets the device name as TFG_ARNAU_DIEZ
    BLE_Send_function((uint8_t*)Str,strlen(Str));
    HAL_Delay(10);
    if(!strcmp(uart3str,"OK+Set[TFG_ARNAU_DIEZ]"))          //If this message is received, the Name is stablished correctly
    {
      ble_uart_clear();                                     //Clear
    }
    else
    {
      BLE_error_no_name_flag=1;                             //If not, it activates the error flag
      return 0;
    }

    //Set Module PIN
    snprintf(Str,sizeof(Str),"AT+PASS[482731]");              //It sets the connection as 482731
    BLE_Send_function((uint8_t*)Str,strlen(Str));
    HAL_Delay(10);
    if(!strcmp(uart3str,"OK+Set[482731]"))                  //If this message is received, the PIN is correctly stablished
      ble_uart_clear();                                     //Clear
    }
    else
    {
      BLE_error_no_pass_set_flag=1;                         //If not, it activates the error flag
      return 0;
```

```
    }

    //Set Module Bond mode for PIN
    snprintf(Str,sizeof(Str),"AT+TYPE[2]");                    //It sets the module bond mode to "Bond with Pin"
    BLE_Send_function((uint8_t*)Str,strlen(Str));
    HAL_Delay(10);
    if(!strcmp(uart3str,"OK+Set[2]"))                        //If this message is received, the Name is stablished correctly
    {
      ble_uart_clear();                                      //Clear
    }
    else
    {
      BLE_error_no_bond_mode_flag=1;                               //If not, it activates the error flag
      return 0;
    }

    //Set Advertising Type
    snprintf(Str,sizeof(Str),"AT+ADTY[0]");                  //Sets the advertising type to Advertising, ScanResponse, Connectable
    BLE_Send_function((uint8_t*)Str,strlen(Str));
    HAL_Delay(10);
    if(!strcmp(uart3str,"OK+Set:[0]"))                       //If this message is received, the Advertising type is stablished correctly
    {
      ble_uart_clear();                                      //Clear
    }
    else
    {
      BLE_error_no_advt_flag=1;                              //If not, it activates the error flag
      return 0;
    }
    //Set Module work mode
    snprintf(Str,sizeof(Str),"AT+MODE[0]");                  //It sets the mode to Trabsmission
    BLE_Send_function((uint8_t*)Str,strlen(Str));
    HAL_Delay(10);
    if(!strcmp(uart3str,"OK+Set:[0]"))                       //If this message is received, the Mode is stablished correctly
    {
      ble_uart_clear();
    }
    else
    {
      BLE_error_no_mode_flag=1;                              //If not, it activates the error flag
      return 0;
    }
  return 1;                                                  //All initialise is OK successfull-y
}

uint8_t CheckApplicationPass(uint32_t timeout_ms)
{ //Function to check if the app has sent the correct password

  uint32_t currentTick = HAL_GetTick();                      //gets the internal time of the MUC in ms
  uint32_t timeout=currentTick+timeout_ms;                   //Target MCU time
  ble_uart_clear();
    while(1)
  {
    if( HAL_GetTick()>=timeout)
    {
      BLE_error_no_app_pass_flag=1;                          // timeout is done, error flag is activated
    }

    if(strlen(uart3str)>0)                                   //If something is received
    {
      HAL_Delay(10);
      if(!strcmp(uart3str,apppassword))                      //If the app pasword is received
      {
        return 1; // success
      }

```


UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

```
      else
      {
        BLE_error_wrong_app_pass_flag=1;                  // wrong password, error flag is activated
        return 0; // wrong password
      }
    }

  }
}

//Wait for the BLE connection
uint8_t wait_for_BLE_connection(uint32_t timeout_ms)            //timeout with milisecond
{

  GPIO_PinState ret;
  uint32_t currentTick = HAL_GetTick();                         //gets the internal time of the MUC in ms
  uint32_t timeout=currentTick+timeout_ms;                      //Target MCU time
  while(1)
  {
    if( HAL_GetTick()>=timeout) BLE_error_not_connected_flag=1; //timeout is done, error flag is
activated
    ret=HAL_GPIO_ReadPin(GPIOE,GPIO_PIN_15);                    //Reads the State pin of the BLE
    if(ret==GPIO_PIN_SET)                                       //If it is activated
    {
      int authstatus=CheckApplicationPass(2*s);                 //It checks if the app has sent the
password

      if(authstatus==1)                                         //If it was sent correctly, the device
is connected
      {
          return 1;                                             // Activates flag
      }

    }
  }
}


//Function to send results to the external device
void BLE_Send_results(void)
{
  char Str[50];                                                               //Buffer to send commands
over the BLE
  char* encrypted_rdm_key;                                                    //Char* where the
encrypted random key is stored
  char* encrypted_data;                                                       //Char* where the data is
stored
  //EMG MODE
  if(device_mode==EMG_START)                                                  //If the EMG results
needs to be sended to the external device
  {
    //RMS

    snprintf(Str,sizeof(Str),"RMS CH1: %.2f", EMG_RMS_1);                     //Sends the RMS value of
CH1 encrypted
    encrypted_rdm_key = rdm_key_gen(23);                                      //Generates a
pseudo-random key and also encrypts it with the user-known key
    encrypted_data=XORCipher(Str,Pseudo_random_key);                          //It encrypts the data
with the newly generated pseudo-random key
    BLE_Send_function((uint8_t*)encrypted_rdm_key,strlen(encrypted_rdm_key)); //It sends the
pseudo-random key, encrypted with the user-known key
    BLE_Send_function((uint8_t*)encrypted_data,strlen(encrypted_data));       //It sends the encrypted
data
    HAL_Delay(10);

    snprintf(Str,sizeof(Str),"RMS CH2: %.2f", EMG_RMS_2);                     //Sends the RMS value of
CH2 encrypted
    encrypted_rdm_key = rdm_key_gen(23);                                      //Generates a
pseudo-random key and also encrypts it with the user-known key
    encrypted_data=XORCipher(Str,Pseudo_random_key);                          //It encrypts the data
with the newly generated pseudo-random key
    BLE_Send_function((uint8_t*)encrypted_rdm_key,strlen(encrypted_rdm_key)); //It sends the
```

```
pseudo-random key, encrypted with the user-known key
    BLE_Send_function((uint8_t*)encrypted_data,strlen(encrypted_data));       //It sends the encrypted data
    HAL_Delay(10);

    snprintf(Str,sizeof(Str),"RMS CH3: %.2f", EMG_RMS_3);                    //Sends the RMS value of CH3 encrypted
    encrypted_rdm_key = rdm_key_gen(23);                                      //Generates a pseudo-random key and also encrypts it with the user-known key
    encrypted_data=XORCipher(Str,Pseudo_random_key);                          //It encrypts the data with the newly generated pseudo-random key
    BLE_Send_function((uint8_t*)encrypted_rdm_key,strlen(encrypted_rdm_key)); //It sends the pseudo-random key, encrypted with the user-known key
    BLE_Send_function((uint8_t*)encrypted_data,strlen(encrypted_data));       //It sends the encrypted data
    HAL_Delay(10);

    //MAV
    snprintf(Str,sizeof(Str),"MAV CH1: %.2f", EMG_MAV_1);                    //Sends the MAV value of CH1 encrypted
    encrypted_rdm_key = rdm_key_gen(23);                                      //Generates a pseudo-random key and also encrypts it with the user-known key
    encrypted_data=XORCipher(Str,Pseudo_random_key);                          //It encrypts the data with the newly generated pseudo-random key
    BLE_Send_function((uint8_t*)encrypted_rdm_key,strlen(encrypted_rdm_key)); //It sends the pseudo-random key, encrypted with the user-known key
    BLE_Send_function((uint8_t*)encrypted_data,strlen(encrypted_data));       //It sends the encrypted data
    HAL_Delay(10);

    snprintf(Str,sizeof(Str),"MAV CH2: %.2f", EMG_MAV_2);                    //Sends the MAV value of CH2 encrypted
    encrypted_rdm_key = rdm_key_gen(23);                                      //Generates a pseudo-random key and also encrypts it with the user-known key
    encrypted_data=XORCipher(Str,Pseudo_random_key);                          //It encrypts the data with the newly generated pseudo-random key
    BLE_Send_function((uint8_t*)encrypted_rdm_key,strlen(encrypted_rdm_key)); //It sends the pseudo-random key, encrypted with the user-known key
    BLE_Send_function((uint8_t*)encrypted_data,strlen(encrypted_data));       //It sends the encrypted data
    HAL_Delay(10);

    snprintf(Str,sizeof(Str),"MAV CH3: %.2f", EMG_MAV_3);                    //Sends the MAV value of CH3 encrypted
    encrypted_rdm_key = rdm_key_gen(23);                                      //Generates a pseudo-random key and also encrypts it with the user-known key
    encrypted_data=XORCipher(Str,Pseudo_random_key);                          //It encrypts the data with the newly generated pseudo-random key
    BLE_Send_function((uint8_t*)encrypted_rdm_key,strlen(encrypted_rdm_key)); //It sends the pseudo-random key, encrypted with the user-known key
    BLE_Send_function((uint8_t*)encrypted_data,strlen(encrypted_data));       //It sends the encrypted data
    HAL_Delay(10);      ;

    //MEAN FREQUENCY
    snprintf(Str,sizeof(Str),"Mean f CH1: %.2f", EMG_Mean_f_1);              //Sends the mean f value of CH1 encrypted
    encrypted_rdm_key = rdm_key_gen(23);                                      //Generates a pseudo-random key and also encrypts it with the user-known key
    encrypted_data=XORCipher(Str,Pseudo_random_key);                          //It encrypts the data with the newly generated pseudo-random key
    BLE_Send_function((uint8_t*)encrypted_rdm_key,strlen(encrypted_rdm_key)); //It sends the pseudo-random key, encrypted with the user-known key
    BLE_Send_function((uint8_t*)encrypted_data,strlen(encrypted_data));       //It sends the encrypted data
    HAL_Delay(10);

    snprintf(Str,sizeof(Str),"Mean f CH2: %.2f", EMG_Mean_f_2);              //Sends the mean f value of CH2 encrypted
    encrypted_rdm_key = rdm_key_gen(23);                                      //Generates a pseudo-random key and also encrypts it with the user-known key
    encrypted_data=XORCipher(Str,Pseudo_random_key);                          //It encrypts the data
```

```
with the newly generated pseudo-random key
    BLE_Send_function((uint8_t*)encrypted_rdm_key,strlen(encrypted_rdm_key)); //It sends the
pseudo-random key, encrypted with the user-known key
    BLE_Send_function((uint8_t*)encrypted_data,strlen(encrypted_data));       //It sends the encrypted
data
    HAL_Delay(10);

    snprintf(Str,sizeof(Str),"Mean f CH3: %.2f", EMG_Mean_f_3);               //Sends the mean f value
of CH3 encrypted
    encrypted_rdm_key = rdm_key_gen(23);                                      //Generates a
pseudo-random key and also encrypts it with the user-known key
    encrypted_data=XORCipher(Str,Pseudo_random_key);                          //It encrypts the data
with the newly generated pseudo-random key
    BLE_Send_function((uint8_t*)encrypted_rdm_key,strlen(encrypted_rdm_key)); //It sends the
pseudo-random key, encrypted with the user-known key
    BLE_Send_function((uint8_t*)encrypted_data,strlen(encrypted_data));       //It sends the encrypted
data
    HAL_Delay(10);

    //MEDIAN FREQUENCY
    snprintf(Str,sizeof(Str),"Median f CH1: %.2f", EMG_Med_f_1);              //Sends the median f
value of CH1 encrypted
    encrypted_rdm_key = rdm_key_gen(23);                                      //Generates a
pseudo-random key and also encrypts it with the user-known key
    encrypted_data=XORCipher(Str,Pseudo_random_key);                          //It encrypts the data
with the newly generated pseudo-random key
    BLE_Send_function((uint8_t*)encrypted_rdm_key,strlen(encrypted_rdm_key)); //It sends the
pseudo-random key, encrypted with the user-known key
    BLE_Send_function((uint8_t*)encrypted_data,strlen(encrypted_data));       //It sends the encrypted
data
    HAL_Delay(10);

    snprintf(Str,sizeof(Str),"Median f CH2: %.2f", EMG_Med_f_2);              //Sends the median f
value of CH2 encrypted
    encrypted_rdm_key = rdm_key_gen(23);                                      //Generates a
pseudo-random key and also encrypts it with the user-known key
    encrypted_data=XORCipher(Str,Pseudo_random_key);                          //It encrypts the data
with the newly generated pseudo-random key
    BLE_Send_function((uint8_t*)encrypted_rdm_key,strlen(encrypted_rdm_key)); //It sends the
pseudo-random key, encrypted with the user-known key
    BLE_Send_function((uint8_t*)encrypted_data,strlen(encrypted_data));       //It sends the encrypted
data
    HAL_Delay(10);

    snprintf(Str,sizeof(Str),"Median f CH3: %.2f", EMG_Med_f_3);              //Sends the median f
value of CH3 encrypted
    encrypted_rdm_key = rdm_key_gen(23);                                      //Generates a
pseudo-random key and also encrypts it with the user-known key
    encrypted_data=XORCipher(Str,Pseudo_random_key);                          //It encrypts the data
with the newly generated pseudo-random key
    BLE_Send_function((uint8_t*)encrypted_rdm_key,strlen(encrypted_rdm_key)); //It sends the
pseudo-random key, encrypted with the user-known key
    BLE_Send_function((uint8_t*)encrypted_data,strlen(encrypted_data));       //It sends the encrypted
data
    HAL_Delay(10);
  }

  //LBIA MODE
  else if(device_mode == LBIA_START)                                          //If the LBIA results
needs to be sended to the external device
  {
    snprintf(Str,sizeof(Str),"R Value: %.2f", LBIAS_R);                       //Sends R value of
bioimpedance encrypted
    encrypted_rdm_key = rdm_key_gen(23);                                      //Generates a
pseudo-random key and also encrypts it with the user-known key
    encrypted_data=XORCipher(Str,Pseudo_random_key);                          //It encrypts the data
with the newly generated pseudo-random key
    BLE_Send_function((uint8_t*)encrypted_rdm_key,strlen(encrypted_rdm_key)); //It sends the
pseudo-random key, encrypted with the user-known key
    BLE_Send_function((uint8_t*)encrypted_data,strlen(encrypted_data));       //It sends the encrypted
data
    HAL_Delay(10);
```

```

    snprintf(Str,sizeof(Str),"Xc Value: %.2f", LBIAS_Xc);                    //Sends Xc value of bioimpedance encrypted
    encrypted_rdm_key = rdm_key_gen(23);                                     //Generates a pseudo-random key and also encrypts it with the user-known key
    encrypted_data=XORCipher(Str,Pseudo_random_key);                         //It encrypts the data with the newly generated pseudo-random key
    BLE_Send_function((uint8_t*)encrypted_rdm_key,strlen(encrypted_rdm_key)); //It sends the pseudo-random key, encrypted with the user-known key
    BLE_Send_function((uint8_t*)encrypted_data,strlen(encrypted_data));       //It sends the encrypted data
    HAL_Delay(10);

    snprintf(Str,sizeof(Str),"PHASE Value: %.2f", LBIAS_PHASE);              //Sends the Phase value of bioimpedance encrypted
    encrypted_rdm_key = rdm_key_gen(23);                                     //Generates a pseudo-random key and also encrypts it with the user-known key
    encrypted_data=XORCipher(Str,Pseudo_random_key);                         //It encrypts the data with the newly generated pseudo-random key
    BLE_Send_function((uint8_t*)encrypted_rdm_key,strlen(encrypted_rdm_key)); //It sends the pseudo-random key, encrypted with the user-known key
    BLE_Send_function((uint8_t*)encrypted_data,strlen(encrypted_data));       //It sends the encrypted data
    HAL_Delay(10);     ;
  }

  else if(device_mode == LBIA_R_START)                                       //If the LBIA results needs to be sended to the external device
  {
    snprintf(Str,sizeof(Str),"R Value: %.2f", LBIAS_R);                      //Sends R value of bioimpedance encrypted
    encrypted_rdm_key = rdm_key_gen(23);                                     //Generates a pseudo-random key and also encrypts it with the user-known key
    encrypted_data=XORCipher(Str,Pseudo_random_key);                         //It encrypts the data with the newly generated pseudo-random key
    BLE_Send_function((uint8_t*)encrypted_rdm_key,strlen(encrypted_rdm_key)); //It sends the pseudo-random key, encrypted with the user-known key
    BLE_Send_function((uint8_t*)encrypted_data,strlen(encrypted_data));       //It sends the encrypted data
    HAL_Delay(10);
  }

  else if(device_mode == LBIA_Xc_START)                                      //If the LBIA results needs to be sended to the external device
  {
    snprintf(Str,sizeof(Str),"Xc Value: %.2f", LBIAS_Xc);                    //Sends Xc value of bioimpedance encrypted
    encrypted_rdm_key = rdm_key_gen(23);                                     //Generates a pseudo-random key and also encrypts it with the user-known key
    encrypted_data=XORCipher(Str,Pseudo_random_key);                         //It encrypts the data with the newly generated pseudo-random key
    BLE_Send_function((uint8_t*)encrypted_rdm_key,strlen(encrypted_rdm_key)); //It sends the pseudo-random key, encrypted with the user-known key
    BLE_Send_function((uint8_t*)encrypted_data,strlen(encrypted_data));       //It sends the encrypted data
    HAL_Delay(10);
  }

  //BATTERY MODE
  else if(device_mode == READ_BATTERY)                                       //If the Battery level needs to be sended to the external device
  {
    snprintf(Str,sizeof(Str),"BATTERY LEVEL: %.2f", Battery_level);          //Sends the Battery left percentage to the external device
    BLE_Send_function((uint8_t*)Str,strlen(Str));
    HAL_Delay(10);
  }
  //ERROR MODE
  else if(device_mode == ERROR)                                              //If the errors need to be sended to the external device
  {

```

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

```
    //AD73894 errors
    if (AD73894_Config_1_error_flag == 1)                          //If the Flag for the AD73894 configuration 1 is activated
    {
      snprintf(Str,sizeof(Str),"AD73984 CONF1. ERROR");            //The error is send to the external device
      BLE_Send_function((uint8_t*)Str,strlen(Str));
      HAL_Delay(10);
      AD73894_Config_1_error_flag = 0;                             //The flag is desactivated
    }
    if (AD73894_Config_2_error_flag == 1)                          //If the Flag for the AD73894 configuration 2 is activated
    {
      snprintf(Str,sizeof(Str),"AD73984 CONF2. ERROR");            //The error is send to the external device
      BLE_Send_function((uint8_t*)Str,strlen(Str));
      HAL_Delay(10);
      AD73894_Config_2_error_flag = 0;                             //The flag is desactivated
    }


    //BLE Errors
    if (BLE_error_no_uart_flag == 1)                               //If the Flag for the BLE connection error no UART is activated
    {
      snprintf(Str,sizeof(Str),"BLE CON. ERROR NO UART");          //The error is send to the external device
      BLE_Send_function((uint8_t*)Str,strlen(Str));
      HAL_Delay(10);
      BLE_error_no_uart_flag = 0;                                  //The flag is desactivated
    }
    if (BLE_error_no_name_flag == 1)                               //If the Flag for the BLE connection error no name is activated
    {
      snprintf(Str,sizeof(Str),"BLE CON. ERROR NO NAME");          //The error is send to the external device
      BLE_Send_function((uint8_t*)Str,strlen(Str));
      HAL_Delay(10);
      BLE_error_no_name_flag = 0;                                  //The flag is desactivated
    }

    if (BLE_error_no_pass_set_flag == 1)                           //If the Flag for PIN not set is activated
    {
      snprintf(Str,sizeof(Str),"BLE NOT PIN. ERROR");              //The error is send to the external device
      BLE_Send_function((uint8_t*)Str,strlen(Str));
      HAL_Delay(10);
      BLE_error_no_pass_set_flag = 0;                              //The flag is desactivated
    }

    if (BLE_error_no_bond_mode_flag == 1)                          //If the Flag for bond mode not set is activated
    {
      snprintf(Str,sizeof(Str),"BLE NOT BOND. ERROR");             //The error is send to the external device
      BLE_Send_function((uint8_t*)Str,strlen(Str));
      HAL_Delay(10);
      BLE_error_no_bond_mode_flag = 0;                             //The flag is desactivated
    }

    if (BLE_error_no_advt_flag == 1)                               //If the Flag for the BLE advt error mp advertising is activated
    {
      snprintf(Str,sizeof(Str),"BLE CON. ERROR NO ADT");           //The error is send to the external device
```

```
        BLE_Send_function((uint8_t*)Str,strlen(Str));
        HAL_Delay(10);
        BLE_error_no_advt_flag = 0;                              //The flag is
desactivated
      }
      if (BLE_error_no_mode_flag == 1)                          //If the Flag for the BLE
connection error no mode is activated
      {
        snprintf(Str,sizeof(Str),"BLE CON. ERROR NO MODE");     //The error is send to
the external device
        BLE_Send_function((uint8_t*)Str,strlen(Str));
        HAL_Delay(10);
        BLE_error_no_mode_flag = 0;                              //The flag is
desactivated
      }

      if (BLE_error_no_app_pass_flag == 1)                      //If the Flag for the BLE
no app password recieved is activated
      {
        snprintf(Str,sizeof(Str),"BLE CON. ERROR NO PASS");     //The error is send to
the external device
        BLE_Send_function((uint8_t*)Str,strlen(Str));
        HAL_Delay(10);
        BLE_error_no_app_pass_flag = 0;                          //The flag is
desactivated
      }

      if (BLE_error_wrong_app_pass_flag == 1)                   //If the Flag for the BLE
no app password recieved is activated
      {
        snprintf(Str,sizeof(Str),"BLE CON. ERROR WRONG PASS");  //The error is send to
the external device
        BLE_Send_function((uint8_t*)Str,strlen(Str));
        HAL_Delay(10);
        BLE_error_wrong_app_pass_flag = 0;                       //The flag is
desactivated
      }

      if (BLE_error_no_mode_flag == 1)                          //If the Flag for the BLE
connection error no mode is activated
      {
        snprintf(Str,sizeof(Str),"BLE CON. ERROR NO MODE");     //The error is send to
the external device
        BLE_Send_function((uint8_t*)Str,strlen(Str));
        HAL_Delay(10);
        BLE_error_no_mode_flag = 0;                              //The flag is
desactivated
      }


      if (BLE_error_not_connected_flag == 1)                    //If the Flag for device
not connected is activated
      {
        snprintf(Str,sizeof(Str),"BLE NOT CON. ERROR");         //The error is send to
the external device
        BLE_Send_function((uint8_t*)Str,strlen(Str));
        HAL_Delay(10);
        BLE_error_not_connected_flag = 0;                        //The flag is
desactivated
      }





      //MVC and EMG errors

      if (MVC_error_CH1_Saturated_flag == 1)                    //If the Flag MVC CH1
saturated samples
      {
        snprintf(Str,sizeof(Str),"MVC CH1 SAT.");               //The error is send to
the external device
```

```
        BLE_Send_function((uint8_t*)Str,strlen(Str));
        HAL_Delay(10);
        MVC_error_CH1_Saturated_flag = 0;                    //The flag is desactivated
      }

      if (MVC_error_CH2_Saturated_flag == 1)                 //If the Flag MVC CH2 saturated samples
      {
        snprintf(Str,sizeof(Str),"MVC CH2 SAT.");            //The error is send to the external device
        BLE_Send_function((uint8_t*)Str,strlen(Str));
        HAL_Delay(10);
        MVC_error_CH2_Saturated_flag = 0;                    //The flag is desactivated
      }

      if (MVC_error_CH3_Saturated_flag == 1)                 //If the Flag MVC CH3 saturated samples
      {
        snprintf(Str,sizeof(Str),"MVC CH3 SAT.");            //The error is send to the external device
        BLE_Send_function((uint8_t*)Str,strlen(Str));
        HAL_Delay(10);
        MVC_error_CH3_Saturated_flag = 0;                    //The flag is desactivated
      }

      if (EMG_error_CH1_Saturated_flag == 1)                 //If the Flag MVC CH1 saturated samples
      {
        snprintf(Str,sizeof(Str),"EMG CH1 SAT.");            //The error is send to the external device
        BLE_Send_function((uint8_t*)Str,strlen(Str));
        HAL_Delay(10);
        EMG_error_CH1_Saturated_flag = 0;                    //The flag is desactivated
      }

      if (EMG_error_CH2_Saturated_flag == 1)                 //If the Flag EMG CH2 saturated samples
      {
        snprintf(Str,sizeof(Str),"EMG CH2 SAT.");            //The error is send to the external device
        BLE_Send_function((uint8_t*)Str,strlen(Str));
        HAL_Delay(10);
        EMG_error_CH2_Saturated_flag = 0;                    //The flag is desactivated
      }

      if (EMG_error_CH3_Saturated_flag == 1)                 //If the Flag EMG CH3 saturated samples
      {
        snprintf(Str,sizeof(Str),"EMG CH3 SAT.");            //The error is send to the external device
        BLE_Send_function((uint8_t*)Str,strlen(Str));
        HAL_Delay(10);
        EMG_error_CH3_Saturated_flag = 0;                    //The flag is desactivated
      }


      //LBIA
      if (LBIA_error_SAT_V==1)                               //if the voltage saturated error flag is activated
      {
        snprintf(Str,sizeof(Str),"LBIA V SAT");              //The error is send to the external device
        BLE_Send_function((uint8_t*)Str,strlen(Str));
        HAL_Delay(10);
        LBIA_error_SAT_V = 0;                                //The flag is desactivated
```

```
    }

    if (LBIA_error_SAT_I==1)                                    //if the current saturated error flag is activated
    {
      snprintf(Str,sizeof(Str),"LBIA I SAT");                   //The error is send to the external device
      BLE_Send_function((uint8_t*)Str,strlen(Str));
      HAL_Delay(10);
      LBIA_error_SAT_I = 0;                                     //The flag is desactivated
    }

      if (LBIA_error_I_electrode_off==1)                        //if the current electrodes lost contact error flag is activated
    {
      snprintf(Str,sizeof(Str),"LBIA I NO CONTACT");            //The error is send to the external device
      BLE_Send_function((uint8_t*)Str,strlen(Str));
      HAL_Delay(10);
      LBIA_error_I_electrode_off = 0;                           //The flag is desactivated
    }

    if (LBIA_error_phase_angle_I==1)                            //if the impedance phase error flag is activated
    {
      snprintf(Str,sizeof(Str),"Z PHASE WRONG");                //The error is send to the external device
      BLE_Send_function((uint8_t*)Str,strlen(Str));
      HAL_Delay(10);
      LBIA_error_phase_angle_I = 0;                             //The flag is desactivated
    }

    //BATTERY
    if (low_battery_flag==1)                                    //if the battery was below 15%
    {
      snprintf(Str,sizeof(Str),"LOW BATTERY");                  //The error is send to the external device
      BLE_Send_function((uint8_t*)Str,strlen(Str));
      HAL_Delay(10);
      low_battery_flag = 0;                                     //The flag is desactivated
    }
  }
}

//COMMAND RECEIVED EVALUATION
//Function that evaluate the received command from an external device
//and activates one of the device modes
void CommandEvalFunc(char * RX_DATO )
{
  //LBIA
  if(!strcmp(RX_DATO,"LBIA"))                         // If the LBIA measurement mode command is received
  {
      device_mode= LBIA_MEASUREMENT;                  // Current device mode is set to LBIA Measurement
  }
  else if(!strcmp(RX_DATO,"LBIAS"))                   //If the Start LBIA command is received
  {
    if (device_mode == LBIA_MEASUREMENT)              // If the device is in LBIA measurement mode, it can start its acquisition
    {
      device_mode = LBIA_START;                       // Current device mode is set to start the LBIA acquisition process
    }
  }
  else if(!strcmp(RX_DATO,"LBIAXcS"))                 //If the Start LBIA (Imaginary part) is received
```

```
  {
    if (device_mode == LBIA_MEASUREMENT)                // If the device is in LBIA measurement mode, it can start its acquisition of the imaginay part
    {
      device_mode = LBIA_R_START;                       // Current device mode is set to start the LBIA acquisition process
    }
  }
  else if(!strcmp(RX_DATO,"LBIARS"))                    //If the Start LBIA (Real part) is received
  {
    if (device_mode == LBIA_MEASUREMENT)                // If the device is in LBIA measurement mode, it can start its acquisition
    {
      device_mode = LBIA_Xc_START;                      // Current device mode is set to start the LBIA acquisition process
    }
  }

  //EMG
  else if(!strcmp(RX_DATO,"EMG"))                       // If the EMG measurement mode command is received
  {
    device_mode = EMG_MEASUREMENT;                      //Current device mode is set to EMG measurement
  }

  else if(!strcmp(RX_DATO,"MVCS"))                      //If the start MVC command is received
  {
    if (device_mode == EMG_MEASUREMENT)                 // If the device is in EMG measurement mode, it can start its acquisition
    {
      device_mode = MVC_START;                          // Current device mode is set to start the MVC acquisition process
    }
  }
  else if (!strcmp(RX_DATO,"EMGS"))                     //If the start EMG command is received
  {
    if ((device_mode==EMG_MEASUREMENT) && (MVC_done==1))  //If the device is in EMG mode and the MVC is done
    {
      device_mode = EMG_START;                          //Current device mode is set to start the EMG acquisition process
    }
  }

  //BATTERY
  else if (!strcmp(RX_DATO,"BAT"))                      //If the read battery command is received
  {
    device_mode = READ_BATTERY;                         //Current device mode is set to start the battery reading process
  }

  //ERRORS
  else if (!strcmp(RX_DATO,"ERROR"))                    //If the read errors command is received
  {
    device_mode = READ_ERRORS;                          //Current device mode is set to check the existing errors
  }
}



//Function to generate a pseudo-random key
//This function generates a pseudo-random key to be used to encrypt the medical data
//and encrypts this key with the user-defined key
char *rdm_key_gen(size_t length)
{ // const size_t length

//XOR Cypher User Known key
char* XOR_user_known_key = "aBhd7w?4Ysn3c#ap2x5zq03c";              // 23 Characters encrypting key for XOR

// All the posible characters used for the key are defined below
```

```c
static char charset[] = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789,.-#'?!";
char *randomString;                                  //string where the data will be stored
char *randomkeyencrypted;                            //string where the encrypted pseudo-random key will be stored

if (length) {
    randomString = malloc(length +1);                // It reserves a blocck of storage of size bytes

    if (randomString)
      {
        int key;                                     //Random character chosen
        int n=0;
        for(n = 0;n < length;n++)                    //For each one of the positions of the key
        {
            key = rand() % 70;                       //It generates a pseudo-random number between 0 and 69 (number of characters)
            randomString[n] = charset[key];          //The value is inserted to the pseudo-random key
        }
      }
}
Pseudo_random_key=randomString;                      //It updates the pseudo-random key with the new generated key
randomkeyencrypted=XORCipher(Pseudo_random_key, XOR_user_known_key); //Encrypts the new pseudo-random key with the user-known key
return randomkeyencrypted;                           //It returns the result of encrypting the pseudo random key with the user-known key
}

//Function to encrypt the data before sending it (XOR Cypher)/
/*
With this algorithm, a string of text can be encrypted by applying the bitwise
XOR operator to every character using a given key.
To decrypt the output, merely reapplying the XOR function with the key will remove the cipher.
*/
  char* XORCipher(char* data, char* key)
  {
    int XOR_Key_length=strlen(key);
    int dataLen=strlen(data);
    char* output = (char*)malloc(sizeof(char) * dataLen);
    for (int i = 0; i < dataLen; ++i)
    {
      output[i] = data[i] ^ key[i % XOR_Key_length];
    }
  return output;
  }
```

LEDs blinking visual warning

```
//Include
#include "main.h"
#include "LEDS.h"

//BLINKING RED LED
//It blinks the red LED for 5s
void RED_led_blinking(void)
{
  int i = 0;
  while (i<5)                //Blinks the red LED for 5s
  {
    RED_LED_ON;              //Turns ON the RED LED
    HAL_Delay(0.5*s);        //The LED is ON 0.5s
    RED_LED_OFF;             //Turns OFF the RED LED
    HAL_Delay(0.5*s);        //The LED is off 0.5s
    i++;
  }
}

//***************************************************************************
//BLINKING GREEN LED
//It blinks the green LED for 5s
void GREEN_led_blinking(void)
{
  int i = 0;
  while (i<5)               //Blinks the green LED for 5s
  {
    GREEN_LED_ON;           //Turns ON the GREEN LED
    HAL_Delay(0.5*s);       //The LED is ON 0.5s
    GREEN_LED_OFF;          //Turns OFF the GREEN LED
    HAL_Delay(0.5*s);       //The LED is off 0.5s
    i++;
  }
}
```

LBIA Measurement

```c

//Include

#include "lbias.h"

//************************************************************************************
//CURRENT AND VOLTAGE ACQUISITION
//***********************************************************************************
//ISR For the LBIA ADC acquisition (TIM12)
void TIMER12IntHandler(void)
{
  // Clear the timer interrupt

__HAL_TIM_CLEAR_IT(&htim12,TIM_IT_UPDATE);


  //Read the ADC1 and ADC2 (Actual Voltage and the injected current)
  //In simultaneous multiple ADC conversion the ADC1 acts as master and ADC2 as a slave

  HAL_ADC_Start_DMA(&hadc1,(uint32_t*)LBIA_Acquisition_Value,adcChannelCount);      //It starts the conversion
  while (LBIA_ADC_conversion_done_flag==0)                                           //While it is converting
  {
  }
  LBIA_ADC_conversion_done_flag=0;                                                   //When the ADC acquisiton is done


  if (LBIA_Acquisition_Value[0]>=LBIA_SATURATED)                                     //If the value is saturated
  {
    {
      LBIA_V_Saturated_flag=1;                                                       //The flag for LBIA measurements is activated
    }
  }

  if (LBIA_Acquisition_Value[1]>=LBIA_SATURATED)                                     //If the value is saturated
  {
      LBIA_I_Saturated_flag=1;                                                       //The flag for LBIA measurements is activated
  }

  if (LBIA_V_Saturated_flag==0&&LBIA_I_Saturated_flag==0) //If there hasn't been a saturation error during the acquisition
  {
    LBIA_Buff_index++;                                                               //It increases by one the position inside the buffer
    if (LBIA_Buff_index>= LBIAS_LENGTH)                                              //If the buffer is full
    {
      //Buffer full
      LBIA_Buff_index=0;
      HAL_TIM_Base_Stop(&htim12);                                                    //It stops the timer
      LBIA_ADC_buff_full_flag=1;                                                     //It informs that the acquisiton has been completed
    }
    else
    {
      LBIAs_CH1_data[LBIA_Buff_index]=LBIA_Acquisition_Value[0];                     //It puts the Voltage value to the buffer
      LBIAs_Injected_current[LBIA_Buff_index]=LBIA_Acquisition_Value[1];             //It puts the current value to the buffer
    }
  }
}

//Change the definition of this HAL function so the flag is activated when the conversion is done
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef*hadc)
{
  LBIA_ADC_conversion_done_flag=1;
}

//************************************************************************************
```

```
//DAC FUNCTION GENERATION OF SINE SIGNAL OF 50 KHZ
//**************************************************************************************

//ISR For the LBIA DAC signal generation (TIM6)
void TIMER6IntHandler(void)
{
  __HAL_TIM_CLEAR_IT(&htim6,TIM_IT_UPDATE);
  HAL_DAC_Start(&hdac,DAC_CHANNEL_1);
  HAL_DAC_Start_DMA(&hdac, DAC_CHANNEL_1, (uint32_t*)sine_wave_array, SINEWAVE_LENGTH, DAC_ALIGN_12B_R);
}


//***************************************************************************************
//MAIN SYNCHRONOUS DEMODULATION FUNCTION
//**************************************************************************************

void LBIAS_demodulation(void)
{

    LBIA_Current=injected_current_peak();     //It computes the peak value of the injected current
    if (LBIA_Current< LBIA_Adequate_I_Vallue) //If the value is not accepted
    {
      LBIA_I_disconnected_flag=1;             //The flag is activated
    }
    else                                      //The injected value was accepted
    {
      sig_product();                          //The multiplication with the sines fucntions occurs
      //First, the real component is obtained
      firfiltering();                         //The signal is filtered
      LBIAS_R=mean_func();                    //The real parameter is obtained using mean function (Voltage)
      LBIAS_R=LBIAS_R/LBIA_Current;           //The real parameter is obtained using mean function (Resistence)

      //Then, the capacitive component is obtained
      Xc_var=1;                               //To indicate we are doing the capacitive part now
      firfiltering();                         //The signal is filtered
      LBIAS_Xc=mean_func();                   //The real parameter is obtained using mean function
      Xc_var=0;                               //Restart the value of Xc
      LBIAS_Xc=LBIAS_Xc/LBIA_Current;         //The imaginary parameter is obtained using mean function (Capacitance)

      //Finaly the bioimpedance phase is obtained
      LBIAS_phase = atan(LBIAS_Xc/LBIAS_R);   //The arcangent is performed to obtain the phase angle
      if (LBIAS_phase<=Min_LBIA_Phase || LBIAS_phase>=Max_LBIA_Phase) //Accepted phase intervals
      {
        LBIA_ADC_wrong_phase_flag=1;          //If it is not an accepted value, it activates this flag error
      }
    }
}
//***************************************************************************************
//Peak of the injected current
//***************************************************************************************
//This function filters the current with a BP filter and then gets the max value of the signal (Peak)
float injected_current_peak(void)
{
  uint8_t j = 0;
  uint8_t i = 0;
  uint16_t max_value=0;                     //Auxiliar variable to obtain the peak value
  float current_peak;                       //Auxiliar variable to store the peak value of the current

  for (i=0; i<LBIAS_LENGTH+BP_ORDER; i++)   //In a convolution the result array has the lengths added
  {
      LBIAS_CURRENT_POST_FILTER[i] = 0;     //Initialization of the destination array
  }
  for(i=0;i<LBIAS_LENGTH;i++)
  {
    for(j=0;j<BP_ORDER; j++)
    {
        LBIAS_CURRENT_POST_FILTER[i+j] = LBIAS_CURRENT_POST_FILTER[i+j]+LBIAs_Injected_current[i]*bandpass_filt_coef[j];
    }
  }
  for (i=0; i<LBIAS_LENGTH+BP_ORDER; i++)
  {
```

```c
    if (LBIAS_CURRENT_POST_FILTER[i]>max_value)
    {
      max_value=LBIAS_CURRENT_POST_FILTER[i];
    }
  }
  current_peak=LBIA_I_tofloat(max_value);    //It converts the maximum value to a float


  return current_peak;                       //The peak is returned by the function
}
//************************************************************************************
//FIR FILTER CONVOLUTION (ORDER = 4, fS = 250000 Hz, Fc = 10 Hz, Window: Hamming)
//************************************************************************************
//Convolution and storing in a new array
void firfiltering(void) //Implements the convolution between the filter components and the signal
{
  uint8_t j = 0;
  uint8_t i = 0;

  for (i=0; i<LBIAS_LENGTH+FT_ORDER; i++)    //In a convolution the result array has the lengths added
  {
      LBIAS_POST_FILTER[i] = 0;              //Initialization of the destination array
  }
  for(i=0;i<LBIAS_LENGTH;i++)
  {
    for(j=0;j<FT_ORDER; j++)
    {
      if (Xc_var==1)
      {
        LBIAS_POST_FILTER[i+j] = LBIAS_POST_FILTER[i+j]+LBIAs_CH1_data_sig90[i]*filt_coef[j];
      }
      else
      {
        LBIAS_POST_FILTER[i+j] = LBIAS_POST_FILTER[i+j]+LBIAs_CH1_data_sig0[i]*filt_coef[j];
      }
    }
  }
}

//************************************************************************************
//SIN AND COS MULTIPLICATION
//************************************************************************************
//Create pick -up square signal and its +90 degree phase shifted version
//And multiply it by data
void sig_product(void)
{
  int i = 0;
  for (i=0;i<LBIAS_LENGTH;i++)
    {
      if (device_mode==LBIA_START)           //If both the real part and the imaginary part is needed
      {
        LBIAs_CH1_data_sig90[i] = LBIAs_CH1_data[i]*sin(2*PI*LBIAS_FREQUENCY*i/LBIAS_LENGTH);         //Multiplies the data with a sine of the same frequency
        LBIAs_CH1_data_sig0[i] = LBIAs_CH1_data[i]*sin(2*PI*LBIAS_FREQUENCY*i/LBIAS_LENGTH+(PI/2));   //Multiplies the data with a signe with the phase shifted
      }
      else if (device_mode==LBIA_R_START)    //If only the real part is needed
      {
        LBIAs_CH1_data_sig90[i] = LBIAs_CH1_data[i]*sin(2*PI*LBIAS_FREQUENCY*i/LBIAS_LENGTH);         //Multiplies the data with a sine of the same frequency
      }
      else                                   //If only the imaginary part is needed
      {
        LBIAs_CH1_data_sig0[i] = LBIAs_CH1_data[i]*sin(2*PI*LBIAS_FREQUENCY*i/LBIAS_LENGTH+(PI/2));   //Multiplies the data with a signe with the phase shifted
      }
    }
}

//************************************************************************************
//MEAN OF THE FILTERED SIGNAL
//************************************************************************************
//Definition of the mean function
```

```
float mean_func(void)
{
  int i = 0;
  float mean=0, sum = 0;
  for(i=0;i<LBIAS_LENGTH+FT_ORDER;i++)
  {
    sum=sum + LBIA_V_tofloat(LBIAS_POST_FILTER[i]);
  }
mean = sum/LBIAS_LENGTH;
return(mean);
}

//*************************************************************************************
//CONVERSION TO FLOAT
//*************************************************************************************
float LBIA_V_tofloat(uint16_t adcdata)
{
  float float_value_after_conv = 0;                                        //Variable where the
uint16_t will be stored after conversion to float
  //It defines the variable to store the float value
  float_value_after_conv = adcdata*(LBIA_vref/4095)/LBIA_V_Hardware_gain;   //Converts the acquired
data to float
  return float_value_after_conv;
}

float LBIA_I_tofloat(uint16_t adcdata)
{
  float float_value_after_conv = 0;                                        //Variable where the
uint16_t will be stored after conversion to float
  //It defines the variable to store the float value
  float_value_after_conv = adcdata*(LBIA_vref/4096)/LBIA_I_Hardware_gain;   //Converts the acquired
data to float
  return float_value_after_conv;
}

//*************************************************************************************
//DAC SINEWAVE GENERATION
//*************************************************************************************
//Creates a sinewave for the DAC use
void dac_sinewave(void)
{
  int i = 0;
  for (i = 0; i<SINEWAVE_LENGTH; i++)
  {
    sine_wave_array[i]=((sin(i*2*PI/SINEWAVE_LENGTH)+1)*(4096/2));
  }
}
```

EMG measurement

```

//Include

#include "emg.h"
//*************************************************************************************
//MAIN EMG FEATURE EXTRACTION FUNCTION
//This function gets a filtered and rectified EMG signal to further process it
//************************************************************************************
void EMG_processing(void)
{
  //First, we do the RMS envelope
  rms_envelope();
  //Then, we normalize its values with the MVC
  normalize();
  //Then, we get the Standart Amplitue values (RMS/MAV)
  RMS_MAV_EMG();
  //Then, we perform the FFT and get the Frequency domain parameters (Median/Mean)

  //For CH1
  DoFFT((float*)EMG_CH1_data);
  EMG_Med_f_1=median_frq;
  EMG_Mean_f_1=mean_frq;

  //For CH2
  DoFFT((float*)EMG_CH2_data);
  EMG_Med_f_2=median_frq;
  EMG_Mean_f_2=mean_frq;
  //For CH3
  DoFFT((float*)EMG_CH3_data);
  EMG_Med_f_3=median_frq;
  EMG_Mean_f_3=mean_frq;
}

//*************************************************************************************
//FFT
//*************************************************************************************
//Different functions for FFT calcullations

// function to sort the array in ascending order
void Array_sort(float32_t *array , int n)
{
    // declare some local variables
    int i=0 , j=0 , temp=0;

    for(i=0 ; i<n ; i++)
    {
        for(j=0 ; j<n-1 ; j++)
        {
            if(array[j]>array[j+1])
            {
                temp        = array[j];
                array[j]    = array[j+1];
                array[j+1]  = temp;
            }
        }
    }
}

// function to calculate the median of the array
float Find_median(float32_t array[] , int n)
{
    float median=0;

    // if number of elements are even
    if(n%2 == 0)
        median = (array[(n-1)/2] + array[n/2])/2;
    // if number of elements are odd
    else
        median = array[n/2];

    return median;
}

//Main FFT function
void DoFFT(float32_t * CH_DATA) {

   median_frq=0;
```

```c
    mean_frq=0;

    doBitReverse = 1;
    ifftFlag = 0;

     /* Process the data through the CFFT/CIFFT module */
     arm_cfft_f32(&varInstCfftF32, CH_DATA, ifftFlag, doBitReverse);

     /* Process the data through the Complex Magnitude Module for
     calculating the magnitude at each bin */
     arm_cmplx_mag_f32(CH_DATA, fftOutput, fftSize);


    //Calculates the median frequency

    // Sort the array in ascending order
    Array_sort(fftOutput , fftSize);

    // Now pass the sorted array to calculate
    // the median of the array.
    median_frq = Find_median(fftOutput , fftSize);


    //Calculates the mean frequency
    for(int i=0;i<fftSize;i++)
    {
       mean_frq+=fftOutput[i];
    }
    mean_frq/=fftSize;
}


//*************************************************************************************
//RMS and MEAN PARAMETER CALCULATION
//*************************************************************************************
void RMS_MAV_EMG(void)
{
  int i = 0;
  int chan = 0;
  float sum_1_mean=0, sum_1_rms, sum_2_mean=0, sum_2_rms, sum_3_mean=0, sum_3_rms;
  for(chan=0;chan<2;chan++)
  {
    for (i=0; i<EMG_LENGHT_RMS;i++) //It performs the Mean and RMS (previously converting the variables to float)
    {
      if (chan == 0)
      {
        sum_1_mean=sum_1_mean+EMGtofloat(EMG_CH1_pros[i]);
        sum_1_rms=sum_1_rms+(EMGtofloat(EMG_CH1_pros[i])*EMGtofloat(EMG_CH1_pros[i]));
      }
      else if (chan == 1)
      {
        sum_2_mean=sum_2_mean+EMGtofloat(EMG_CH2_pros[i]);
        sum_2_rms=sum_2_rms+(EMGtofloat(EMG_CH2_pros[i])*EMGtofloat(EMG_CH2_pros[i]));
      }
      else
      {
        sum_3_mean=sum_3_mean+EMGtofloat(EMG_CH3_pros[i]);
        sum_3_rms=sum_3_rms+(EMGtofloat(EMG_CH3_pros[i])*EMGtofloat(EMG_CH3_pros[i]));
      }
    }
  }

  //MEAN
  EMG_MAV_1 = sum_1_mean/EMG_LENGHT_RMS;                        //The mean is performed for channel 1
  EMG_MAV_2 = sum_2_mean/EMG_LENGHT_RMS;                        //The mean is performed for channel 2
  EMG_MAV_3 = sum_3_mean/EMG_LENGHT_RMS;                        //The mean is performed for channel 3

  //RMS
  EMG_RMS_1 = sqrt(sum_1_rms/EMG_LENGHT_RMS);                   //The RMS is performed for channel 1
  EMG_RMS_2 = sqrt(sum_2_rms/EMG_LENGHT_RMS);                   //The RMS is performed for channel 2
  EMG_RMS_3 = sqrt(sum_3_rms/EMG_LENGHT_RMS);                   //The RMS is performed for channel 3

}
```

```
//*************************************************************************************
//NORMALIZATION OF THE SIGNAL
//*************************************************************************************
void normalize(void)
{
  int i = 0;
  int chan =0;
  for (chan=0; chan<2; chan++)                                  //For each channel
  {
    for (i=0;i<EMG_LENGHT_RMS;i++)                              //The normalization is done every point is a % respecte the MVC value
    {
      if (chan == 0)
      {
        EMG_CH1_pros[i] = (EMG_CH1_pros[i]/MVC_mean_1)*100;
      }
      else if (chan == 1)
      {
        EMG_CH2_pros[i] = (EMG_CH2_pros[i]/MVC_mean_2)*100;
      }
      else
      {
        EMG_CH3_pros[i] = (EMG_CH3_pros[i]/MVC_mean_3)*100;
      }
    }
  }

}

//*************************************************************************************
//RMS ENVELOPE OF THE SIGNAL
//*************************************************************************************
void rms_envelope(void)
{
  int i=0;
  int j=0;
  int k=0;
  int chan=0;
  float sum_1=0, sum_2=0, sum_3=0;

  for (chan=0;chan<2;chan++)                                     //For each channel
  {
    for (i=RMS_wind_size/2;i<EMG_LENGTH-(RMS_wind_size/2);i++)    //All the point of the signal to be considered(avoiding indexing errors)
    {
      for (j=i-RMS_wind_size/2; j<i+RMS_wind_size/2;j++)          //For all the points inisde the window the square is added to sum
      {
        if (chan==0)                                             //Channel 1
        {
          sum_1 = sum_1+(EMG_CH1_data[j]*EMG_CH1_data[j]);        //Square of the digit
        }
        else if(chan==1)
        {
          sum_2 = sum_2+(EMG_CH2_data[j]*EMG_CH2_data[j]);        //Channel 2
        }
        else
        {
          sum_3 = sum_3+(EMG_CH3_data[j]*EMG_CH3_data[j]);        //Channeñ 3
        }
      }
    EMG_CH1_pros[k] = sqrt(sum_1/RMS_wind_size);                  //The square root is performed --> RMS envelope CH1
    EMG_CH2_pros[k] = sqrt(sum_2/RMS_wind_size);                  //The square root is performed --> RMS envelope CH2
    EMG_CH3_pros[k] = sqrt(sum_3/RMS_wind_size);                  //The square root is performed --> RMS envelope CH3
    k++;                                                          //The position is updated
    }

  }
}
//*************************************************************************************
//MEAN OF THE MVC
//*************************************************************************************
//Definition of the mean function
```

```

void mean_MVC(void)
{
  int j = 0;
  int chan = 0;
  float sum_1 = 0, sum_2 = 0, sum_3 = 0;        //Auxiliar variables
  if (MVC_done == 0)                             //MVC is not yet done
  {
    for (chan=0; chan<2; chan++)                 //For each channel (1,2,3)
    {
      for(j=0;j<MVC_LENGTH;j++)
      {
        if (chan==0)
        {
          sum_1=sum_1 + MVC_CH1_data[j];
          MVC_mean_1=sum_1/MVC_LENGTH;          //Mean of channel 1 (MVC)
        }
        else if (chan==1)
        {
          sum_2=sum_2 + MVC_CH2_data[j];        //Mean of channel 2 (MVC)
          MVC_mean_2=sum_2/MVC_LENGTH;
        }
        else
        {
          sum_3=sum_3 + MVC_CH3_data[j];        //Mean of channel 3 (MVC)
          MVC_mean_3=sum_3/MVC_LENGTH;
        }
      }

    }
  }
}

//***************************************************************************************
//CONVERSION TO FLOAT
//***************************************************************************************
float EMGtofloat(uint16_t adcdata)
{
  float float_value_after_conv = 0;                     //Variable where the uint16_t will be stored after conversion to float
  //It defines the variable to store the float value
  float_value_after_conv = adcdata*(Vref_AD78934/65535)/EMG_Hardware_gain;        //Converts the acquired data to float
  return float_value_after_conv;
}

```

C Header files code

Main

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.h
  * @brief          : Header for main.c file.
  *                   This file contains the common defines of the application.
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2022 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
  ******************************************************************************
  */
/* USER CODE END Header */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx_hal.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#define ARM_MATH_CM4            //Needed for CMSIS
#include "arm_math.h"
#include "arm_const_structs.h"
/* USER CODE END Includes */

/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */

/* USER CODE END ET */

/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */

/* USER CODE END EC */

/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */

/* USER CODE END EM */

/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);

/* USER CODE BEGIN EFP */

/* USER CODE END EFP */

/* Private defines -----------------------------------------------------------*/
#define BLE_State_Pin GPIO_PIN_15
#define BLE_State_GPIO_Port GPIOE
#define GREEN_LED_Pin GPIO_PIN_12
#define GREEN_LED_GPIO_Port GPIOD
#define ORANGE_LED_Pin GPIO_PIN_13
#define ORANGE_LED_GPIO_Port GPIOD
#define RED_LED_Pin GPIO_PIN_14
#define RED_LED_GPIO_Port GPIOD
#define BLUE_LED_Pin GPIO_PIN_15
#define BLUE_LED_GPIO_Port GPIOD
#define SP1_CS_Pin GPIO_PIN_6
```

```
#define SP1_CS_GPIO_Port GPIOB
/* USER CODE BEGIN Private defines */
extern SPI_HandleTypeDef hspi1;
extern TIM_HandleTypeDef htim9;
/* USER CODE END Private defines */

#ifdef __cplusplus
}
#endif

#endif /* __MAIN_H */

```

Battery voltage estimation

```
//It is used for the Battery calculation

#include "main.h"
#include "adc.h"
//Definitions

#define BATT_Max_Discharge_Cap 1200                 //Max Battery Discharge capacity
#define Vref_ADC3 3.3                               //Vref voltage value of the ADC3


//Variables
extern double Battery_level;                        //Indicates the current battery level
extern double Battery_tension;                      //Double to store the battery tension level
extern uint16_t ADC3_data;                          //Variable where the ADC3 value will be stored
//Used functions
void Read_battery_level(void);                      //It reads the battery level left
void Battery_ADC_Acquisiton(void);                  //ADC Conversion for ADC3 of the board
void ADC3tofloat(void);                             //uint16_t (ADC3 data) to float

```

AD7389-4

```c
/*
  ad73894.h
*/

// Standard libraries
#include <assert.h>
#include <stdint.h>
#include <stdbool.h>
#include "main.h"

#include "emg.h"

/* Chip Select Pin for AD7389*/
#define AD_CSL    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, GPIO_PIN_RESET)
#define AD_CSH    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, GPIO_PIN_SET)

//Register adresses
#define Configuration_1 0x1
#define Configuration_2 0x2
#define Alert_Indication 0x3
#define Alert_Low_Trh 0x4
#define Alert_High_Trh 0x5

//Auxiliar definitions
#define EMG_Saturated 65535
#define Max_Saturated_samples 2000

//Device modes
#define WAITING_MODE    0
#define LBIA_MEASUREMENT 1
#define LBIA_START      2
#define LBIA_R_START    3
#define LBIA_Xc_START   4
#define EMG_MEASUREMENT 5
#define MVC_START       6
#define EMG_START       7
#define READ_BATTERY    8
#define READ_ERRORS     9

//Required Functions
void init_AD7389(void);                                                    //Function to initialize the AD7389
void shutdown_AD7389(void);                                                //Function to put the AD7389 to shutdown mode

uint8_t adWriteRegister(uint8_t *dataTx,uint16_t Size);                    //Function needed to write to a register
uint8_t AdcSpiSendReceiveByte(uint8_t dataTx);                             //Function used to
void AdcSpiSendReceiveArray(uint8_t dataTx[], uint8_t dataRx[], uint8_t byteLength); //Function to receive and send data through the spi
void Read_ADC_values(uint16_t *ch1,uint16_t *ch2,uint16_t *ch3);           //Function to read the adc values
void TIMER9IntHandler(void);                                               //ISR for the timer 9

//Auxiliar vairable
extern uint32_t adc_counter;                                               //Counter for the Buffer position
extern uint16_t saturated_samples_1;                                       //Counter of the saturated samples for CH1
extern uint16_t saturated_samples_2;                                       //Counter of the saturated samples for CH2
extern uint16_t saturated_samples_3;                                       //Counter of the saturated samples for CH3
extern uint8_t device_mode;                                                //Indicates in which state the device is
//Definition of REGISTERS Structure
typedef struct
{
  uint16_t ConfReg1;        //Configuration register 1 (Used)
  uint16_t ConfReg2;        //Configuration register 2 (Used)
  uint16_t AlertIndication; //Alert indication register (not used)
```

```c
  uint16_t AlertLowTrh;      //Alert low threshold (not used)
  uint16_t AlertHighTrh;     //Alert high threshold (not used)

} REGISTERS;

//Adressing used in the register structure
typedef struct
{
  uint8_t wr: 1;             //1 bit for WR (1 to write to the specified register)
  uint8_t regaddress: 3;     //3 bits to specifie the regsiter address
} addressing;

//Configuration 1 register
typedef struct
{
  uint8_t PMODE: 1;          //1 Bit for PMODE (normal mode/shutdown mode)
  uint8_t RESERVED: 1;       //1 Bit reserved
  uint8_t RES: 1;            //1 Bit for RES (Resolution)
  uint8_t ALERT_EN: 1;       //1 Bit for ALERT_EN (Enable Alert Indicator Function)
  uint8_t CRC_R : 1;         //1 Bit for CRC_R (CRC Read. Controls the CRC functionality for the SDOx interface)
  uint8_t CRC_W : 1;         //1 Bit for CRC_W (CRC Write. Controls the CRC functionality for the SDI interface)
  uint8_t OSR: 2;            //2 Bits for OSR (Oversampling Ratio)
  uint8_t OSR2: 1;           //1 Bit for OSR2
  uint8_t OS_MODE : 1;       //1 Bit for OS_MODE (Oversampling Mode)
  uint8_t RESERVED2: 2;      //2 Bits reserved
  addressing ADDRESSING;     //4 Bits for the previously defined Adressing

} ConfigurationRegister1;


//Configutation 2 register
typedef struct
{
  uint8_t RESET: 8;          //1 Byte for RESET (Soft reset or Hard reset)
  uint8_t SDO: 2;            //2 Bits for SDO (Conversion Results Serial Data Output)
  uint8_t RESERVED: 2;       //2 Bits reserved
  addressing ADDRESSING;     //4 Bits for the previously defined Adressing

} ConfigurationRegister2;


//Definition of the code abreviations
extern ConfigurationRegister1 ConfReg1;   //Configuration 1 Register
extern ConfigurationRegister2 ConfReg2;   //Configuration 2 Register
extern REGISTERS AdRegister;              //Registers group
extern addressing address;                //Adress

//Used flags
extern uint8_t AD73894_conf1_failed_flag;    //Flag to inform that the configure 1 register of AD73894 was not writen correctly
extern uint8_t AD73894_conf2_failed_flag;    //Flag to inform that the configure 1 register of AD73894 was not writen correctly
extern uint8_t AD73894_Init_OK_flag;         //Flag initializing process of the AD73894 was done correctly.
extern uint8_t EMG_CH1_Saturated_flag;       //Flag to inform that the EMG acquisition went wrong with ch1
extern uint8_t EMG_CH2_Saturated_flag;       //Flag to inform that the EMG acquisition went wrong with ch2
extern uint8_t EMG_CH3_Saturated_flag;       //Flag to inform that the EMG acquisition went wrong with ch3
extern uint8_t MVC_acquisition_done_flag;    //Indicates if the MVC has been done already
extern uint8_t EMG_acquisition_done_flag;    //Indicates if the EMG has been done already

```

BLE connection

```
//It is used for the BLE communication

#include "main.h"
#include "usart.h"
#include "dma.h"
#include "stdio.h" // this is for snprintf function
#include <string.h> // this is required for strlen function
#include <stdlib.h> //For the malloc function


/* Definitions ------------------------------------------------------------------*/
//Device modes
#define WAITING_MODE    0
#define LBIA_MEASUREMENT 1
#define LBIA_START      2
#define LBIA_R_START    3
#define LBIA_Xc_START   4
#define EMG_MEASUREMENT 5
#define MVC_START       6
#define EMG_START       7
#define READ_BATTERY    8
#define READ_ERRORS     9


#define uart3maxlen 30                                      //Buffer Size of Uart3str from handling uart3
#define apppassword "h325bc092A"                              //String that represents the app pasword
#define s 1000                                              //One second (in ms)

/*******************************************************/
//GENERAL USE VARIABLES
extern uint8_t device_mode;                                 //Indicates in which stte the device is (LBIAs measurement or EMG measurement) (Will be initialized later)
extern uint8_t MVC_done;                                    //Indicates if the MVC has been done already
extern double Battery_level;                                //Indicates the current battery level

//Variables
extern uint8_t device_connected_flag;
extern uint8_t RX_DATO;
extern char uart3str[uart3maxlen];

//EMG PROCESSING FUNCTION
//MVC means
extern float MVC_mean_1;
extern float MVC_mean_2;
extern float MVC_mean_3;

//Standart amplitude parameters
//RMS
extern float EMG_RMS_1;
extern float EMG_RMS_2;
extern float EMG_RMS_3;

//MAV
extern float EMG_MAV_1;
extern float EMG_MAV_2;
extern float EMG_MAV_3;

//Frequency domain parameters
//Median f
extern float EMG_Med_f_1;
extern float EMG_Med_f_2;
extern float EMG_Med_f_3;

//Mean f
extern float EMG_Mean_f_1;
extern float EMG_Mean_f_2;
extern float EMG_Mean_f_3;

//LBIA PARAMETERS
extern float LBIAS_Xc;
```

```
extern float LBIAS_R;
extern float LBIAS_PHASE;

//ERRORS
extern uint8_t BLE_error_no_uart_flag;                        //Flag for the BLE connection error no uart
extern uint8_t BLE_error_no_name_flag;                        //Flag for the BLE connection error no name
extern uint8_t BLE_error_no_advt_flag;                        //Flag for the BLE connection error no
advertising
extern uint8_t BLE_error_no_mode_flag;                        //Flag for the BLE connection error no mode
extern uint8_t BLE_error_no_pass_set_flag;                    //Flag for the BLE connection error no
PIN set
extern uint8_t BLE_error_no_bond_mode_flag;                   //Flag for the BLE connection error no
bond type set
extern uint8_t BLE_error_not_connected_flag;                  //Flag for the BLE external device
connection error
extern uint8_t BLE_error_no_app_pass_flag;                    //Flag for the BLE external device
connection no app password error
extern uint8_t BLE_error_wrong_app_pass_flag;                 //Flag for the BLE external device
connection wrong app password error

extern uint8_t low_battery_flag;                              //Flag error to inform that the battery
was below 15%

extern uint8_t LBIA_error_SAT_V;                              //Flag error to inform that the LBIA V
samples were saturated
extern uint8_t LBIA_error_SAT_I;                              //Flag error to inform that the LBIA I
samples were saturated
extern uint8_t LBIA_error_I_electrode_off;                    //Flag error to inform that the LBIA I
electrodes have lost contact
extern uint8_t LBIA_error_phase_angle_I;                      //Flag error to inform that the obtained
phase angle does not make sense

extern uint8_t EMG_error_CH1_Saturated_flag;                  //Flag to inform that the EMG acquisition
went wrong with ch1
extern uint8_t EMG_error_CH2_Saturated_flag;                  //Flag to inform that the EMG acquisition
went wrong with ch2
extern uint8_t EMG_error_CH3_Saturated_flag;                  //Flag to inform that the EMG acquisition
went wrong with ch3
extern uint8_t MVC_error_CH1_Saturated_flag;                  //Flag to inform that the MVC acquisition
went wrong with ch1
extern uint8_t MVC_error_CH2_Saturated_flag;                  //Flag to inform that the MVC acquisition
went wrong with ch2
extern uint8_t MVC_error_CH3_Saturated_flag;                  //Flag to inform that the MVC acquisition
went wrong with ch3

extern uint8_t AD73894_Config_1_error_flag;                   //Flag to inform that he config1 register
of the AD73894 was not well implemented
extern uint8_t AD73894_Config_2_error_flag;                   //Flag to inform that he config2 register
of the AD73894 was not well implemented

//Pseudo-random generated key encrypted with the user-know key
extern char*Pseudo_random_key;                                // Pseudo-random generated key

//Used functions
void ble_uart_clear(void);                                    //Clear function for the BLE
char* XORCipher(char* data, char* key);                       //Function to encrypt the data before
sending it
char *rdm_key_gen(size_t length);                             //Function to generate a pseudo-random key
void BLE_Send_function(uint8_t *txData, uint8_t txDataSize);  //Function to send the data to the
external device
void BLE_Send_results(void);                                  //It sends the results to the external
device
void encrypt_data(void);                                      //It encrypts the data using a XOR
algorithm
int8_t BLE_init(void);                                        //It initializes the BLE module
uint8_t wait_for_BLE_connection(uint32_t timeout_ms);         //Waits for an external device to be
connected by BLE
void CommandEvalFunc(char * RX_DATO );                        //It evaluates the received command from
the external device
```

LEDS

```
//It is used for the control of the LEDs

#include "main.h"


//Definition
//LEDS (To turn ON and OFF the LEDs)
#define GREEN_LED_ON HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_SET)
#define GREEN_LED_OFF HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_RESET)
#define ORANGE_LED_ON HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_SET)
#define ORANGE_LED_OFF HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_RESET)
#define RED_LED_ON HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_SET)
#define RED_LED_OFF HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET)
#define BLUE_LED_ON HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_SET)
#define BLUE_LED_OFF HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_RESET)

//1 second
#define s 1000                              //Second in ms

//Functions
void RED_led_blinking(void);            //Function that blinks the RED LED for 5 seconds
void GREEN_led_blinking(void);            //Function that blinks the Green LED for 5 seconds

```

LBIA measurement

```c
/*
*Performs the LBIA processing
*/

#include "main.h"
#include "tim.h"
#include "adc.h"
#include "dac.h"

//******************************************************************************
//
// Required signal processing Libraries
//
//******************************************************************************
#include <math.h>
#include <stdlib.h>
#include <time.h>
//******************************************************************************

/* Definitions ------------------------------------------------------------------*/
//Device modes
#define WAITING_MODE    0
#define LBIA_MEASUREMENT 1
#define LBIA_START      2
#define LBIA_R_START    3
#define LBIA_Xc_START   4
#define EMG_MEASUREMENT 5
#define MVC_START       6
#define EMG_START       7
#define READ_BATTERY    8
#define READ_ERRORS     9


//Uint16 to float definitions
#define LBIA_vref 3.3
#define LBIA_I_Hardware_gain 10                          //TO BE MODIFIED!!!!!!!!!!!!!!
#define LBIA_V_Hardware_gain 10                          //TO BE MODIFIED!!!!!!!!!!!!!!



//LBIA Phase adequate interval values
#define Min_LBIA_Phase  0                                //Minimum impedance phase value to get accepted
#define Max_LBIA_Phase  20                               //Maximum impedance phase value to get accepted

//Injected Current Normal Value
#define LBIA_Adequate_I_Vallue 0.0002828                 //Minimum peak value to be accepted

#define LBIAS_FREQUENCY 50000                            //Frequency of the imput signal
#define FT_ORDER 5                                       //This is the number of filter coefficients plus one
#define BP_ORDER 9                                       //This is the number of filter coefficients plus one
#define LBIA_SATURATED 4095                              //Value for a saturated data from LBIA acquisition

//LBIAs data buffer
#define LBIAS_LENGTH 50                                  //10 periods --> 50x2 (uint16)
#define SINEWAVE_LENGTH 480                              //SineWave lenght

extern uint16_t LBIAs_Injected_current[LBIAS_LENGTH];     //It stores the  "Intensity generated"

extern uint16_t LBIAs_CH1_data[LBIAS_LENGTH];             //It imports the data of LBIAs
extern uint16_t LBIAs_CH1_data_sig0[LBIAS_LENGTH];        //It creates a buffer to store the sin
extern uint16_t LBIAs_CH1_data_sig90[LBIAS_LENGTH];       //It creates a buffer to store the sin




//Auxiliar variables
extern volatile uint16_t LBIA_Acquisition_Value[2];       //Where the acquired intensity (generated by the DAC) and the Voltage is stored
extern int adcChannelCount;                               //Used for the acqusition process of multiple
```

```
ADC channels
extern volatile uint8_t LBIA_ADC_conversion_done_flag;    //Flag to inform that the conversion has been completed
extern uint8_t LBIA_ADC_buff_full_flag;                   //Flag to inform that the buffer is full (The acquisition has finished)
extern uint8_t LBIA_Buff_index;                           //Used to indicate the current position of the Buffer
extern uint8_t Xc_var;                                    //Used to differenciate between sig0 and sig90
extern uint8_t device_mode;                               //To identify the current device mode
extern uint8_t LBIA_V_Saturated_flag;                     //The tension acquisition is saturated flag
extern uint8_t LBIA_I_Saturated_flag;                     //The current acquisition is saturated flag
extern uint8_t LBIA_I_disconnected_flag;                  //The current electrodes has been desconected
extern uint8_t LBIA_ADC_wrong_phase_flag;                 //The obtained phase angle does not make sense
extern double LBIA_Current;                               //LBIA DC component of the injected current
//Results
extern float LBIAS_R;                                     //Real component of the bioimpedance
extern float LBIAS_Xc;                                    //Capacitive component of the bioimpedance
extern float LBIAS_phase;                                 //Bioimpedance phase
extern uint32_t sine_wave_array[SINEWAVE_LENGTH];               //Sine array waveform
extern float LBIAS_POST_FILTER[LBIAS_LENGTH+FT_ORDER];    //it defines the array where the data will be stored after the filtering
extern float LBIAS_CURRENT_POST_FILTER[LBIAS_LENGTH+FT_ORDER];  //it defines the array where the data will be stored after the filtering
//These arrays store all the filter coefficients
extern float filt_coef[FT_ORDER];
extern float bandpass_filt_coef[BP_ORDER];
//Required functions
void LBIAS_demodulation (void);                           //Main processing fucntion
float mean_func(void);                                    //It performs the mean operation
void sig_product(void);                                   //Generating and multiplying with a sine and a sine with phase
void firfiltering(void);                                  //Low Pass filtering
void TIMER12IntHandler(void);                             //Timer for the ADC 2 of the board
void TIMER6IntHandler(void);                              //Timer for the DAC
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef*hadc);    //Used for the acquisition
float LBIA_V_tofloat(uint16_t adcdata);                   //To convert the voltage uint16_t to float value
float LBIA_I_tofloat(uint16_t adcdata);                   //To convert the I uint16_t to float value
float injected_current_peak(void);                        //To get the DC component of the injected current
void dac_sinewave(void);                                  //Function used to generate the DAC sinewave
```

EMG measurements

```c
/*
*Performs the EMG processing
*/

#include "main.h"


//******************************************************************************
//
// Required signal processing Libraries
//
//******************************************************************************
#include <math.h>
#include <stdlib.h>
#include <time.h>

//******************************************************************************
//Definitions
#define MVC_LENGTH 30000
#define EMG_LENGTH 50000
#define RMS_wind_size 500   //100 ms (5000/100) //A canviar
#define EMG_LENGHT_RMS EMG_LENGTH-RMS_wind_size
#define Vref_AD78934 2.5
#define EMG_Hardware_gain 500

//******************************************************************************
//Auxiliar variables
extern uint8_t MVC_done;
//******************************************************************************
//Desired Parameters

//MVC Means
extern float MVC_mean_1;
extern float MVC_mean_2;
extern float MVC_mean_3;

//Standart amplitude parameters
//RMS
extern float EMG_RMS_1;
extern float EMG_RMS_2;
extern float EMG_RMS_3;

//MAV
extern float EMG_MAV_1;
extern float EMG_MAV_2;
extern float EMG_MAV_3;

//Frequency domain parameters
//Median f
extern float EMG_Med_f_1;
extern float EMG_Med_f_2;
extern float EMG_Med_f_3;

//Mean f
extern float EMG_Mean_f_1;
extern float EMG_Mean_f_2;
extern float EMG_Mean_f_3;

//******************************************************************************
//MVC Buffers
extern uint16_t MVC_CH1_data[MVC_LENGTH];
extern uint16_t MVC_CH2_data[MVC_LENGTH];
extern uint16_t MVC_CH3_data[MVC_LENGTH];

//******************************************************************************
//EMG Buffers
extern uint16_t EMG_CH1_data[EMG_LENGTH];
extern uint16_t EMG_CH2_data[EMG_LENGTH];
extern uint16_t EMG_CH3_data[EMG_LENGTH];

//EMG (RMS/NORMALIZATION) Buffers
extern uint16_t EMG_CH1_pros[EMG_LENGHT_RMS];
```

```
extern uint16_t EMG_CH2_pros[EMG_LENGHT_RMS];
extern uint16_t EMG_CH3_pros[EMG_LENGHT_RMS];

//****************************************************************************
//FFT Related
//External Input and Output buffer Declarations for FFT Bin Example

static float32_t fftOutput[EMG_LENGTH/2];
//Global variables for FFT Bin Example

extern uint32_t fftSize;
extern uint32_t ifftFlag;
extern uint32_t doBitReverse;
extern arm_cfft_instance_f32 varInstCfftF32;

extern float median_frq;
extern float mean_frq;

//****************************************************************************
//Required functions
void rms_envelope(void);                        //It performs the RMS envelope to the data
void mean_MVC(void);                            //It performs the mean to the MVC data
void normalize(void);                           //It normalizez the EMG data with the MVC value
void RMS_MAV_EMG(void);                         //It obtains the Standart Amplitude Parameters
(MAV/RMS)
void DoFFT(float32_t * CH_DATA);                //It performs the FFT
void Array_sort(float32_t *array , int n);      //It sorts the array for the FFT
float Find_median(float32_t array[] , int n);   //It finds the median for the FFT
void EMG_processing(void);                      //Main function for the processing of the EMG data
float EMGtofloat(uint16_t adcdata);             //It converts the uin16_t data to float
```

Matlab simulations code

LBIA Injected current peak value simulation

```
%%LBIA Injected Current signal Analysis
%Arnau Diez Clos

clear all;
clc;
%% Parameters of the simulated Bia Voltage (Sine)
%This sine signal represents the acquired and conditioned monofrequency (50kHz)
%LBIA signal from a random patient.

%Signal Frequency
f = 50000;

%Sampling Rate
Fs = 250000;

% Make time vector (Period time * Number of periods) (1s of acquisition)
t = 0:1/Fs:(1/ f)*50000;

% Acquired sine signal with an amplitude of 250µA rms
LBIA_I_SIGNAL = 0.0003535*sin(2*pi*f*t);

%White Gaussian Noise is added to the signal with a SNR of 10
NOISY_LBIA_I_SIGNAL = awgn(LBIA_I_SIGNAL,20,'measured','linear');

%% Low Pass Filter

% All frequency values are in Hz.
Fs = 250000;  % Sampling Frequency

N    = 8;         % Order
Fc1  = 44000;     % First Cutoff Frequency
Fc2  = 55000;     % Second Cutoff Frequency
flag = 'scale';   % Sampling Flag
% Create the window vector for the design algorithm.
win = hamming(N+1);

% Calculate the coefficients using the FIR1 function.
b  = fir1(N, [Fc1 Fc2]/(Fs/2), 'bandpass', win, flag);
Hd = dfilt.dffir(b);
LP_I_BIA = dfilt.dffir(b);

%Before getting the peak value, the signal is filtered with a fir low pass
Filtered_signal_1 = filter ( LP_I_BIA , NOISY_LBIA_I_SIGNAL);


%% Find LBIA I peak
max_value=0;

for i=1:length(Filtered_signal_1)
    if Filtered_signal_1(i)>max_value
        max_value=Filtered_signal_1(i)

    end
end

%% Relative error
Relative_error = abs(((0.0003535 - max_value)/0.0003535)*100)

```

LBIA Injected synchronous demodulation simulation

Simulation Script

```
%%LBIA demodulation Analysis
%Arnau Diez Clos

clear all;
clc;
%% Parameters of the simulated Bia Voltage (Sine)
%This sine signal represents the acquired and conditioned monofrequency (50kHz)
%LBIA signal from a random patient.

%Signal Frequency
f = 50000;

%Sampling Rate
Fs = 250000;

%The phase is normally between 8° and 15° in monofrequency studies.
%Phase degrees
phaseInDegrees = 10;

% Make time vector (Period time * Number of periods) (1s of acquisition)
t = 0:1/Fs:(1/ f)*50000;

% Calculate phase in degrees to radians
phaseInRad = deg2rad ( phaseInDegrees );

%Creation of the signal
LBIA_V_SIGNAL = 0.001*sin(2*pi*f*t+phaseInRad);

%White Gaussian Noise is added to the signal with a SNR of 10
NOISY_LBIA_V_SIGNAL = awgn(LBIA_V_SIGNAL,20,'measured','linear');

%% Low Pass Filter
%Filter specifications (All frequency values in Hz)
% All frequency values are in Hz.
Fs = 250000;  % Sampling Frequency

N    = 4;        % Order
Fc   = 10;       % Cutoff Frequency
flag = 'scale';  % Sampling Flag

% Create the window vector for the design algorithm.
win = hamming(N+1);

% Calculate the coefficients using the FIR1 function.
b  = fir1(N, Fc/(Fs/2), 'low', win, flag);
LP_BIAS = dfilt.dffir(b);

%% Demodulation

[R_LBIA, Xc_LBIA, PHASE_LBIAS] = LBIAS_demodulation(NOISY_LBIA_V_SIGNAL, f, t, LP_BIAS)
Relative_error = abs(((phaseInDegrees - PHASE_LBIAS)/phaseInDegrees)*100)
```

LBIA demodulation function

```
function [ Ravg , Xcavg, phase ] = LBIA_demodulation (LBIA_SIGNAL , f, t, filterObject)
% Does IQ demodulation of a sine and square product signal and returns the
% average.

% input :  filterObject ’ is the provided filter object to apply on the sine and square p

% Create pick -up sine signal and its +90 degree phase shifted version
sqr = sin(2* pi*f*t);
sqr90 = sin(2* pi*f*t + (pi/2));

% Multiply sine with the sine signal and with the phase shifted sine signal separatly .
sig0 = LBIA_SIGNAL .* sqr ;
sig90 = LBIA_SIGNAL .* sqr90 ;


% Apply the designed filter
R = filter ( filterObject , sig0 );
Xc = filter ( filterObject , sig90 );

% Average R and Xc arrays
Ravg = mean (R);
Xcavg = mean (Xc);
phase = atand(Xc/R);
end
```

Annex B. SPSS PCA Output

In this annex, the additional documentation generated by the software SPSS when developing the preliminary LBIA statistical analysis is exposed.

Factor Analysis

Descriptive Statistics

	Mean	Std. Deviation	Analysis N
PerCh_Xc_Left_VM_21	.3609	24.56806	11
PerCh_Xc_Right_VM_21	-16.9718	23.39369	11
PerCh_Xc_Left_VL_21	7.5327	18.72709	11
PerCh_Xc_Right_VL_21	-21.7418	14.85248	11
KSS_HUGTiP_Post_IQ_6	163.09	24.643	11
KSS_HUGTiP_Pre_IQ	92.55	20.206	11

Covariance Matrix

	PerCh_Xc_Left_VM_21	PerCh_Xc_Right_VM_21	PerCh_Xc_Left_VL_21	PerCh_Xc_Right_VL_21
PerCh_Xc_Left_VM_21	603.589	156.071	231.587	67.293
PerCh_Xc_Right_VM_21	156.071	547.265	-32.833	289.985
PerCh_Xc_Left_VL_21	231.587	-32.833	350.704	5.246
PerCh_Xc_Right_VL_21	67.293	289.985	5.246	220.596
KSS_HUGTiP_Post_IQ_6	-49.011	-119.957	-126.961	-134.041
KSS_HUGTiP_Pre_IQ	-89.685	-208.131	-102.877	-83.125

Covariance Matrix

	KSS_HUGTiP_Post_IQ_6	KSS_HUGTiP_Pre_IQ
PerCh_Xc_Left_VM_21	-49.011	-89.685
PerCh_Xc_Right_VM_21	-119.957	-208.131
PerCh_Xc_Left_VL_21	-126.961	-102.877
PerCh_Xc_Right_VL_21	-134.041	-83.125
KSS_HUGTiP_Post_IQ_6	607.291	-51.855
KSS_HUGTiP_Pre_IQ	-51.855	408.273

Correlation Matrix[a]

		PerCh_Xc_Left_VM_21	PerCh_Xc_Right_VM_21	PerCh_Xc_Left_VL_21
Correlation	PerCh_Xc_Left_VM_21	1.000	.272	.503
	PerCh_Xc_Right_VM_21	.272	1.000	-.075
	PerCh_Xc_Left_VL_21	.503	-.075	1.000
	PerCh_Xc_Right_VL_21	.184	.835	.019
	KSS_HUGTiP_Post_IQ_6	-.081	-.208	-.275
	KSS_HUGTiP_Pre_IQ	-.181	-.440	-.272
Sig. (1-tailed)	PerCh_Xc_Left_VM_21		.210	.057
	PerCh_Xc_Right_VM_21	.210		.413
	PerCh_Xc_Left_VL_21	.057	.413	
	PerCh_Xc_Right_VL_21	.294	.001	.478
	KSS_HUGTiP_Post_IQ_6	.406	.270	.206
	KSS_HUGTiP_Pre_IQ	.298	.088	.209

Correlation Matrix[a]

		PerCh_Xc_Right_VL_21	KSS_HUGTiP_Post_IQ_6	KSS_HUGTiP_Pre_IQ
Correlation	PerCh_Xc_Left_VM_21	.184	-.081	-.181
	PerCh_Xc_Right_VM_21	.835	-.208	-.440
	PerCh_Xc_Left_VL_21	.019	-.275	-.272
	PerCh_Xc_Right_VL_21	1.000	-.366	-.277
	KSS_HUGTiP_Post_IQ_6	-.366	1.000	-.104
	KSS_HUGTiP_Pre_IQ	-.277	-.104	1.000
Sig. (1-tailed)	PerCh_Xc_Left_VM_21	.294	.406	.298
	PerCh_Xc_Right_VM_21	.001	.270	.088
	PerCh_Xc_Left_VL_21	.478	.206	.209
	PerCh_Xc_Right_VL_21		.134	.205
	KSS_HUGTiP_Post_IQ_6	.134		.380
	KSS_HUGTiP_Pre_IQ	.205	.380	

a. Determinant = .080

KMO and Bartlett's Test

Kaiser-Meyer-Olkin Measure of Sampling Adequacy.		.415
Bartlett's Test of Sphericity	Approx. Chi-Square	18.088
	df	15
	Sig.	.258

Communalities

	Initial	Extraction
PerCh_Xc_Left_VM_21	1.000	.643
PerCh_Xc_Right_VM_21	1.000	.926
PerCh_Xc_Left_VL_21	1.000	.858
PerCh_Xc_Right_VL_21	1.000	.892
KSS_HUGTiP_Post_IQ_6	1.000	.864
KSS_HUGTiP_Pre_IQ	1.000	.734

Extraction Method: Principal Component Analysis.

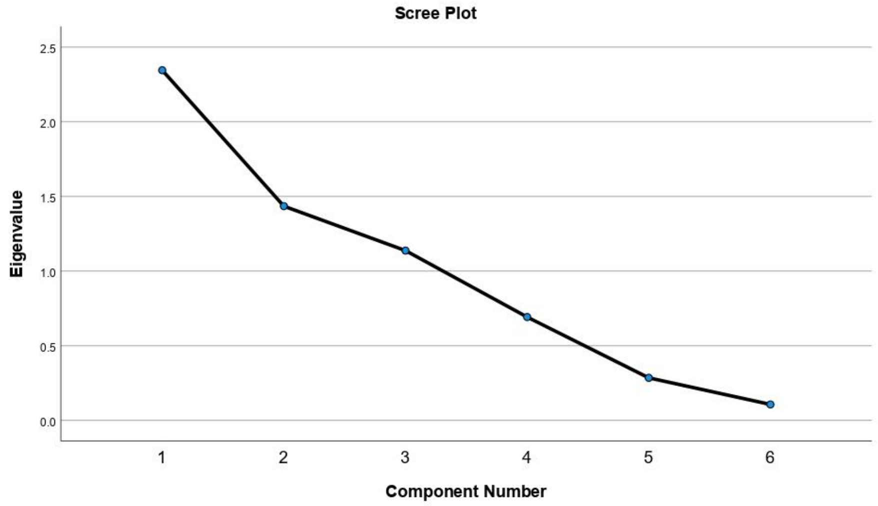
Total Variance Explained

Component	Initial Eigenvalues			Extraction Sums of Squared Loadings		
	Total	% of Variance	Cumulative %	Total	% of Variance	Cumulative %
1	2.345	39.086	39.086	2.345	39.086	39.086
2	1.435	23.915	63.001	1.435	23.915	63.001
3	1.137	18.951	81.951	1.137	18.951	81.951
4	.692	11.532	93.484			
5	.285	4.743	98.227			
6	.106	1.773	100.000			

Total Variance Explained

Component	Rotation Sums of Squared Loadings		
	Total	% of Variance	Cumulative %
1	2.136	35.592	35.592
2	1.633	27.223	62.815
3	1.148	19.136	81.951
4			
5			
6			

Extraction Method: Principal Component Analysis.



Component Matrix[a]

	Component		
	1	2	3
PerCh_Xc_Left_VM_21	.525	.597	.106
PerCh_Xc_Right_VM_21	.858	-.424	.100
PerCh_Xc_Left_VL_21	.361	.851	
PerCh_Xc_Right_VL_21	.841	-.401	-.157
KSS_HUGTiP_Post_IQ_6	-.423		.822
KSS_HUGTiP_Pre_IQ	-.564		-.641

Extraction Method: Principal Component Analysis.

a. 3 components extracted.

Rotated Component Matrix[a]

	Component		
	1	2	3
PerCh_Xc_Left_VM_21	.185	.780	
PerCh_Xc_Right_VM_21	.961		
PerCh_Xc_Left_VL_21		.913	.131
PerCh_Xc_Right_VL_21	.918		.220
KSS_HUGTiP_Post_IQ_6	-.275	-.215	-.861
KSS_HUGTiP_Pre_IQ	-.503	-.379	.581

Extraction Method: Principal Component Analysis.
Rotation Method: Quartimax with Kaiser Normalization.[a]

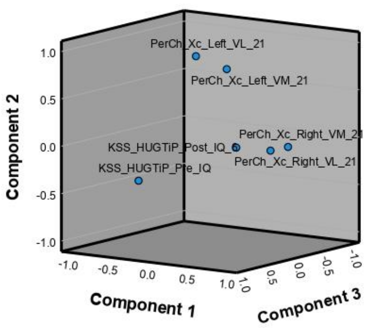
a. Rotation converged in 4 iterations.

Component Transformation Matrix

Component	1	2	3
1	.878	.469	.094
2	-.474	.880	.038
3	.065	.078	-.995

Extraction Method: Principal Component Analysis.
Rotation Method: Quartimax with Kaiser Normalization.

Component Plot in Rotated Space



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

Component Score Coefficient Matrix

	Component		
	1	2	3
PerCh_Xc_Left_VM_21	.005	.478	-.056
PerCh_Xc_Right_VM_21	.467	-.082	-.065
PerCh_Xc_Left_VL_21	-.150	.589	.094
PerCh_Xc_Right_VL_21	.438	-.088	.161
KSS_HUGTiP_Post_IQ_6	-.081	-.084	-.739
KSS_HUGTiP_Pre_IQ	-.224	-.202	.536

Extraction Method: Principal Component Analysis.
Rotation Method: Quartimax with Kaiser Normalization.
Component Scores.

Component Score Covariance Matrix

Component	1	2	3
1	1.000	.000	.000
2	.000	1.000	.000
3	.000	.000	1.000

Extraction Method: Principal Component Analysis.
Rotation Method: Quartimax with Kaiser Normalization.
Component Scores.