DESIGN, IMPLEMENTATION AND TEST OF A LOW-COST ELECTRICAL IMPEDANCE SPECTROSCOPY SYSTEM BASED ON AN AD5940

A Degree Thesis

Submitted to the Faculty of the

Technical School of Telecommunications Engineering of Barcelona

Universitat Politècnica de Catalunya

by

Maria Dolors Farré Romera

In partial fulfilment

of the requirements for the degree in

TELECOMMUNICATIONS TECHNOLOGIES AND SERVICES ENGINEERING

Advisor: Ramon Bragós Bardia

Barcelona, June 2021

Abstract

The bioengineering world is a non-stoppable evolving world, that is the main reason why this project has been carried out.

In this thesis, a low-cost electrical impedance spectroscopy (EIS) system based on the AD5940 board was studied checking all its functionalities. EIS can be used to characterize biological and electrochemical materials and systems. Since the beginning, the focus of the paper is to explore all its features, starting from the board itself deeply and then, creating a user interface that allows the potential users to check the results obtained with the board. The main strength of the board is that it measures impedance in the biological applications range (10Ω - 10kΩ; 1kHz – 150kHz) with very low noise, and this is demonstrated at the end of the document, where the experimental results of the board using the user interface are shown, including human measurements and inert biological object measurements.

Resum

El mon de la bioenginyeria està en constant desenvolupament, aquesta és la principal raó per la qual aquest projecte s'ha dut a terme.

A aquesta tesi, un espectroscopi d'impedància elèctrica de baix cost (EIS) basat en la placa AD5940 ha estat estudiat, per tal d'analitzar totes les seves funcionalitats. El sistema EIS serveix per caracteritzar materials i sistemes biològics i electroquímics. L'enfocament del document és explorar totes les seves característiques, començant des de la placa més profundament i més endavant, creant una interfície d'usuari que permet als possibles usuaris de la placa comprovar els resultats obtinguts. Una de les principals qualitats de la placa és que mesura la impedància en el rang definit per les aplicacions biològiques (10Ω - 10kΩ; 1kHz – 150kHz) amb molt poc soroll, tal i com es demostra al final del document, on es mostren els resultats experimentals utilitzant la interfície creada, incloent mesures humanes i mesures d'objectes biològics inerts.

Resumen

El mundo de la bioingeniería está en constante desarrollo, razón principal por la que este proyecto se ha llevado a cabo.

En esta tesis, un espectroscopio de impedancia eléctrica de bajo coste (EIS) basado en la placa AD5940 ha sido estudiado, para todas sus funcionalidades. Los sistemas EIS sirven para caracterizar materiales y sistemas biológicos y electroquímicos. El foco principal del documento es explorar todas sus características, empezando desde la placa más profundamente, y más adelante creando una interfaz de usuario que permita a los posibles usuarios de la placa comprobar los resultados obtenidos. Una de las principales cualidades del circuito es que mide la impedancia en el rango definido para las aplicaciones biológicas (10Ω - 10kΩ; 1kHz – 150kHz) con muy poco ruido, tal y como se demuestra al final del documento, donde se muestran los resultados experimentales utilizando la interfaz creada, incluyendo medidas humanas y medidas de objetos biológicos inertes.

To my family. They are who have trusted in me always and unconditionally, without asking, just supporting me every time I needed.

Apart of my family I have to say thanks to all the people I love; they are my second family. I am who I am thanks to them.

Acknowledgements

If I have to mention everyone who has helped me with this project, I would need more than one page, but there were some people specially involved.

I have to say thanks to my advisor Ramon Bragós, who is the main person that helped me with this. He helped me since day one and this project would not be possible without him (also thanks for not letting me panic when my PC almost got formatted a month before delivering the project).

The second person I want to say thanks is Isaac, who is the person that started the bases of this project in another project. Thanks to give me your support even when you were not working on this, I will never forget the day when we stayed locked inside the faculty trying to solve one issue of the project.

The last person to mention is Marc Mateu. He is a PDI of the University who helped me understanding how to connect the board with BLE, Matlab and Windows. Without him I think that this project could not have been possible.

Thanks to you and also to all of the people that helped me technical or emotional way because this thesis has been possible with all of your support.

Revision history and approval record

Revision	Date	Purpose
0	06/06/2021	Document creation
1	19/06/2021	Document revision
2	20/06/2021	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Maria Dolors Farré Romera	maria.dolors.farre@estudiantat.upc.edu
Ramon Bragós Bardia	ramon.bragos@upc.edu

Written by:		Reviewed and approved by:	
Date	17/06/2021	Date	19/06/2021
Name	Maria Dolors Farré	Name	Ramon Bragós
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables

1. Introduction

In this section, the thesis will be presented going through the objectives, requirements and specifications, the methods and procedures, the workplan and at the end the structure of the document is explained.

1.1. Objectives

The project is carried out at the Electronic Engineering Department in Technical University of Catalonia under the supervision of Ramon Bragós Bardia. The department is focused in teaching, researching and developing activities related with Semiconductors Devices and Microsystems, Industrial and Power Electronics, Integrated Circuits and Systems, and Measurement Systems and Biomedical Instrumentation. This thesis has been performed into this last research line.

This thesis deals with the design, implementation, test and validation through an application of a low cost electrical impedance spectroscopy. The system is based on the system-on-chip AD5940 evaluation board, an Arduino-compatible microcontroller board and a user interface developed in Matlab.

The project main goals are:

- Test the system-on-chip AD5940 under different circumstances and study its characteristics.
- Know how to implement the features found using the 4-wire bioimpedance measurement
- Design and develop a functional user interface which works selecting the preconditions of the system-on-chip.
- Put together all the features and ensure the functionality.

1.2. Requirements and specifications

Project requirements:

- Explore the features of the AD5940 system-on-chip for both the bioimpedance and the electrochemical applications.
- Adapt the existing microcontroller software to the Arduino platform, including a Bluetooth link with a computer.
- Develop a user interface in Matlab.
- Test and characterize the system and perform a measurement campaign.

Project specifications:

- The exploration of the AD5940 has to be fully completed
- The microcontroller should be integrated at 100% with an Arduino compatible board.
- The user interface is functional and user friendly in Matlab.
- The measurement campaign is completed and returns accurate and reliable results.

1.3. Background

This project is a continuation of a previous project made by Isaac Montsech in the Introduction to Research course of the master on Electronic Engineering:

- His project consisted of the study of the system-on-chip, the adaptation of the microcontroller firmware of the microcontroller of the evaluation board to an Arduino-compatible code, a first preliminary user interface and the first study of the system.
- The project was performed under the Electronic Engineering department at UPC.
- The initial ideas were shared by the supervisor to improve the work made by the first owner of the project.

1.4. Workplan – Gantt diagram

1.4.1. Work Packages

Project: Research and investigation	WP ref: (WP1)	
Major constituent:	Sheet 1 of 6	
Short description: Research and investigation about the main topic and the previous project that initiates this	Planned start date:15/02/2021 Planned end date:21/02/2021	
	Start event:15/02/2021 End event:21/02/2021	
Internal task T1: Read the previous project Internal task T2: Investigate about the topics Internal task T3: Prepare the pc to receive the software	Deliverables:	Dates:1

Project: Software installation	WP ref: (WP2)	
Major constituent: Software, hardware, simulation, programming	Sheet 2 of 6	
Short description: Reception of the software and hardware and their installation on the pc. Check that all works with the system-on-chip and the different applications.	Planned start date:22/02/2021 Planned end date:08/03/2021	
	Start event:22/02/2021 End event:24/03/2021	
Internal task T1: Receive the hardware Internal task T2: Install the software Internal task T3: Combine the hardware with the software	Deliverables:	Dates: 2

Project: Explore the features	WP ref: (WP3)
Major constituent: Hardware, programming, simulation	Sheet 3 of 6
Short description: Explore the system-on-chip AD5940 for both situations, bioimpedance and electrochemical applications	Planned start date:09/03/2021 Planned end date:31/03/2021

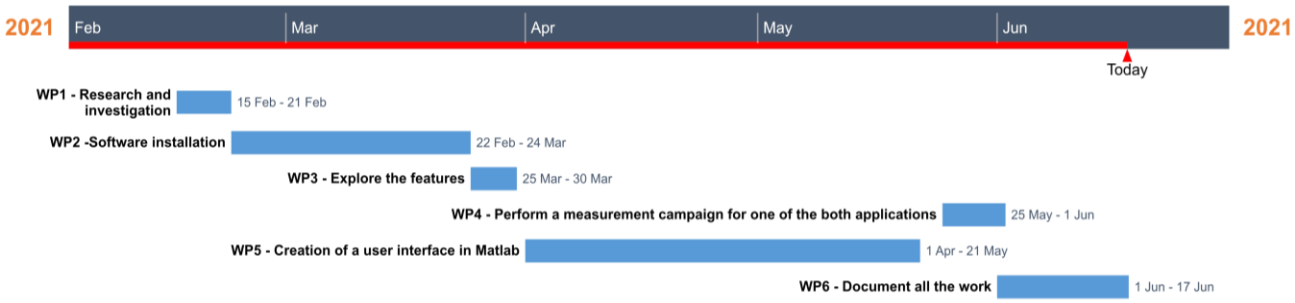
Retrieve some conclusions about the utility of the chip in both cases	Start event:25/03/2021 End event:30/03/2021	
Internal task T1: Explore bioimpedance part Internal task T2: Explore electrochemical application Internal task T3: Compare the results and retrieve the corresponding conclusions	Deliverables:	Dates: 4

Project: Perform a measurement campaign for one of the both applications	WP ref: (WP4)	
Major constituent: Hardware, software, simulation	Sheet 4 of 6	
Short description: Test the chosen application and explore deeply its usabilities and the scope of the measurements	Planned start date:01/04 Planned end date:02/05	
	Start event:25/05/2021 End event:01/06/2021	
Internal task T1: Test the chosen application Internal task T2: Get conclusions	Deliverables: Measurement Campaign	Dates: 7

Project: Creation of a user interface in Matlab	WP ref: (WP5)	
Major constituent: programming, software	Sheet 5 of 6	
Short description: Creation of a user interface to facilitate the usability of the system-on-chip. Through the interface the user is allowed to modify the parameters of the system and shows the signal received	Planned start date:03/05 Planned end date:31/05	
	Start event:01/04/2021 End event:25/05/2021	
Internal task T1: Code the user interface in Matlab Internal task T2: Test that works for both system-on-chip	Deliverables: User interface	Dates: 12

Project: Document all the work	WP ref: (WP6)	
Major constituent: documentation	Sheet 6 of 6	
Short description: Document all the work done	Planned start date: 01/06 Planned end date: 18/06	
	Start event:01/06/2021 End event:17/06/2021	
Internal task T1: Prepare the documentation	Deliverables: Final Report	Dates: 16

1.4.2. Gantt



Respect to the initial goals and workplan, the exploration of the electrochemical features of the AD5940 was not reached.

1.5. Structure of the document

The first part of the document after the introduction is the State of the art. In that section, the hardware of the project and how the measures have been taken is explained. The next section is the Project development which is the most extensive part because all the user interface and how is it related with BLE and Arduino is explained there.

Finally, the last part of the document is the Results one, where the experimental measures are shown in biological and non-biological materials.

After all of the document, there is the Budget of the project and Conclusions and in the end, the glossary. An annex with the code for the user interface is attached on another document.

2. State of the art of the technology used in this thesis

Before going deeply inside the project itself, a research among the existing bioimpedance measurement technologies has been done. In this section it is explained the basic concepts and the main features of the AD5940 evaluation board. Also, the main applications of the board are explained and why the bioimpedance one has been chosen.

2.1. Electrical Impedance Spectroscopy (EIS)

Electrical Impedance Spectroscopy also known in some application fields as Electrochemical Impedance Spectroscopy. It is widely used in materials characterization, and it is starting to be used in biomedical applications. One reason is that it allows separating the influences of different electrical conduction components which allows understanding the materials composition behaviour.

Another reason is that it is a very sensitive to surface interfacial phenomena, which makes many changes visible, for example surface changes due to protein adsorption or penetration of corrosion protection layers. As a result, EIS is interesting for analytical electrochemistry, because molecules can be detected without a redox active marker.

While resistance is the ratio of voltage or potential and current for a DC (direct current) system, the impedance is the ratio of voltage or potential and current for AC (alternating current) systems. The wave nature makes it necessary to define the impedance with two parameters. One is the total impedance Z and the other one is the phase shift ϕ.

Considering the two periodic waves of current and voltage, they have the same frequency. As mentioned before, there is a phase shift ϕ that is a time shift between the two waves. Its units are degrees (º) because usually waves are represented as vectors[4].

The total impedance is the ratio of the amplitude of the potential and the amplitude of the current. The resulting impedance is a complex number, and it can be expressed in the complex plane in polar coordinates by using Z as the length of the vector and ϕ as the angle. The impedance can also be expressed as the real part of the impedance Z', which is the resistance and the imaginary part Z''.

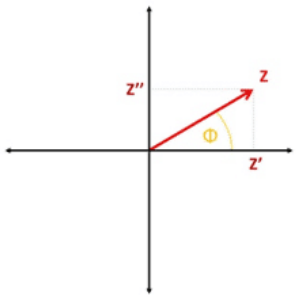


Figure 2.1: Impedance representation

2.2. Main applications

As seen before, EIS is a powerful analysis method, and it has applications in non-biological fields and (and most commonly) in biological scenarios. It is based on introducing a perturbation into the system by an instrument then acquiring and representing the impedance diagram. EIS has also applications across areas of science, including physical sciences, material science, biology and medicine[6].

2.2.1. Non-biological applications

In this type of scenarios, EIS is commonly used to study corrosion, because the current is affected by the interfaces between materials generated in the corrosion on the surface of the metal. It can be used for example to:

- Observing corrosion resistance of dental alloys in artificial saliva.
- Observing corrosion rates of biodegradable medical implants.
- Monitoring lithium-ion batteries and fuel cells.

2.2.2. Biological applications

EIS has wider applicability in biology than in non-biology. Some examples are:

- Study of cells and tissues:
  - Determination of the state of organs.
  - Differentiation of normal and malignant prostate tissue.
  - Study of malignant and normal breast tissue.
  - Characterization of neural tissue.
- Studying antibody-antigen binding.
- Determination of body composition.
- Detection of ischemia (lack of oxygen).
- Study of electron transfer mechanisms of horseradish peroxidase.

2.3. Measurement methods

Electrical bioimpedance is defined as the measurement of the electrical impedance of a biological material or system. This parameter is of minor importance, but it can reflect some interesting physiological conditions and events[10].

Schwan[1] defined three frequency regions for the dielectric properties of biological materials from the observed main dispersions of the conductivity and the permittivity:

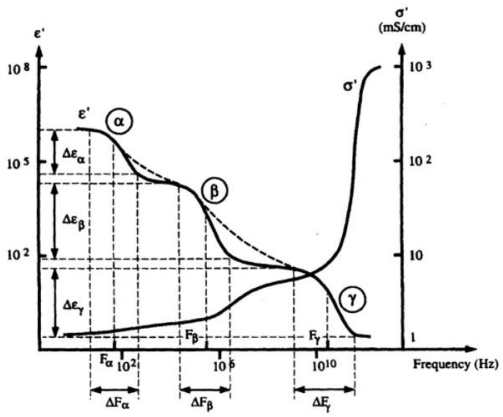


Figure 2.2: Three frequency regions of permittivity and conductivity dispersion

The large dielectric dispersions appearing between 10Hz and tenths of MHz (α and β dispersion regions) are generally considered to be associated with the diffusion processes of the ionic species (α dispersion) and the dielectric properties of the cell

[1] Herman Schwan (1915 - 2005) was a biomedical and biophysics engineer who developed his career in the University of Pennsylvania. He is recognized as one of the founding fathers of biomedical engineering.

membranes and their interactions with the extra and intra-cellular electrolytes (β dispersion). The dielectric properties at the γ region are mostly attributed by the aqueous content of the biological species and the presence of small molecules (Foster[2] and Schwan, 1989). Some authors also reference a fourth main dispersion called δ between the β and γ the dispersions, around 100MHz (Pethig, 1984) that would be caused by the dipolar moment of big molecules such as proteins. The more useful dispersion for the study of biological materials is the β dispersion, which provides structural information about the size, density and homogeneity of cells.

2.3.1. 4 electrodes measurement method

To measure the impedance in the β region the usual way is to inject a sinusoidal current at a given frequency and to detect the voltage drop in the sample. The frequency is swept by steps in a given range. To interface the measurement circuit with the biological material, it is needed to insert electrodes, which also have an impedance which can be higher than the impedance under test. Because of this, the four electrodes method is used to perform the measures: two of them take care of injecting current to the sample and the other two detect the voltage drop. With this method, the electrodes impedance does not affect on the tissue impedance calculation. The AD5940 uses this method[5].

2.4. SOC systems

A system on a chip, also known as SoC, is essentially an integrated circuit that takes a single platform and integrates an entire electronic or computer system onto it. It is exactly an entire system inside a chip[8]. The main components that a SoC generally looks to incorporate, include a central processing unit, input and output ports, internal memory as well as analog input and output blocks.

Depending on the kind of system that has been reduced to the size of a chip, it can perform a variety of functions including signal processing, wireless communication, artificial intelligence, etc.

2.4.1. AD5940

The SoC used for this project is the AD5940. It is specifically designed for bio-impedance. The SoC included in a board called EVAL-AD5940BIOZ[9]. It was designed for carrying out bio-electric measurements, which include:

- Electrodermal Activity (EDA)
- Body Impedance Analysis (4-wire) (BIA)
- Electrocardiogram (ECG)
- Body Impedance Analysis (2-wire) (BIOZ-2Wire)
- Electrochemical measurements.

[2] Kenneth R Foster is a professor of bioengineering – University of Pennsylvania

The platform has an Arduino UNO form factor board. It has the typical connectors of the Arduino Uno shield boards although the microcontroller board provided by Analog Devices and connected below the evaluation board is not Arduino Compatible and uses a microcontroller from the same manufacturer. This section describes the key features of the board and how to set it up to take measurements.

Figure 2.3: EVAL-AD5940BIOZ

There is a Micro USB connector on the EVAL-AD5940BIOZ board which has two functionalities:

- The provided ECG wires can be connected to a human body simulator which can be used to verify ECG (the board also includes an ECG SOC), body composition, etc[3].
- The provided AD5940 Z Test board can also be connected to the Micro USB connector. This board contains a number of banks of resistors and capacitors which can be used to model Body Impedance, Skin Impedance, Electrode Impedances and Contact Impedances. These resistor values are configured with the switches.

[3] Check section *4. Results* for further information

3. Project development

In this section, it will be explained the whole development of the project, the blockers found and how they affected on the project development and the functionality, starting from how BLE works with Matlab and at the end it will be presented the user interface and its characteristics.

3.1. Main barriers

3.1.1. Troubleshoot between BLE, Python and Windows

During the project development some problems were faced, especially trying to connect the microcontroller with the computer. Isaac Montsech created a basic interface in python for his Introduction to Research thesis and when running the interface for this thesis, that did not work. After checking the BLE configuration of both computers, the interface configuration and the Arduino configuration, the problem was that python uses the library *Bleak* to work with BLE which uses the driver *Generic Bluetooth Adapter.* This driver is working in Linux and Mac but is not present in all versions of Windows and that was the case of the computer used to develop this project. The interface was initially created in Linux and then when running it in Windows, was not working.

This issue blocked the thesis for several weeks because the problem seemed to be the configuration of the interface or the computer configuration. Finally, the problem was the operating system so with a specific Windows computer, the python interface cannot be used. Therefore it was decided to develop the user interface in Matlab.

3.1.2. Matlab and Arduino connection

Once the Matlab interface was initially created, some issues when trying to read data from AD5940 were found. The connection between Matlab and any BLE device is better explained in the next section 3.2, but it is not needed at all to understand what the main problem was. After connecting and reading data from the device with the help of Professor Marc Mateu, it was difficult to print the data received on the graphs because the format of the data received was configured in a specific way. With the help of Isaac, who developed the Python interface and printed the results in there, the interface configuration was converted to a format suitable for Arduino and sent to it.

After the data treatment inside Arduino, the data received in Matlab must be converted to known values to be displayed in graphs. In the next sections it is explained how the data is converted, but this conversion took a lot of time because of the lack of knowledge of the Arduino program for this board, and this caused problems when setting the correct data format and send it from Arduino.

3.1.3. Special microcontroller

The fact that the board, although being compatible with Arduino at connector level is not a common board, delayed the project for a while because of the need of learning from scratch the board behaviour.

The boards finally connected, are an Arduino Nano 33 BLE and the shield board of the development kit of the AD5940. Both were non-known boards, so some time was spend learning how they work.

The Arduino program was done to control the Arduino Nano 33 BLE but the code, developed by Isaac Montsech is so long and difficult and before start developing some time was taken to try to understand the code and its interface. The documentation for AD5940 was so extensive because the board has considerable applications, and every application has its configuration.

This is not a problem at all because learning how the hardware works is an important part of the interface development, but it is true that if the previous knowledge had been there, this step would have been more agile.

3.2. Bluetooth Low Energy (BLE)

Bluetooth has a special low energy feature which means that it can be used requiring less power from the devices connected. Bluetooth Low Energy or most used BLE, is the communication link chosen to be connected with the AD5940, so the interface must be developed accordingly to use BLE.

First, every Bluetooth device contains a table of data called *Attribute Table* and it can be accessed by other connected devices. That table of data and its accessibility belongs to the *Generic Attribute Profile* or *GATT*.

3.2.1. Generic Attribute Profile (GATT)

GATT defines the way that two BLE devices transfer data back and forth using concepts called *Services* and *Characteristics.* It makes use of a generic data protocol called the *Attribute Protocol* or *ATT,* which is used to store *Services*, *Characteristics* and related data in a table using 16-bit IDs for each entry.

The most important thing to keep in mind when using GATT so using BLE, is that connections are exclusive. What is meant is that a BLE peripheral can only be connected to one central device at a time. As soon as it is connected to a central device, it will stop advertising itself and other devices will no longer be able to see it or connect to it until the existing connection is broken. Establishing a connection is also the only way to allow two-way communication, where the central device can send meaningful data to the peripheral and peripheral to the central device. If data needs to be exchanged between two peripherals, a custom mailbox system will need to be implemented where all messages pass through the central device.[2]

Once a connection is established between a peripherals and central device, it can take place in both directions as seen in the image below.

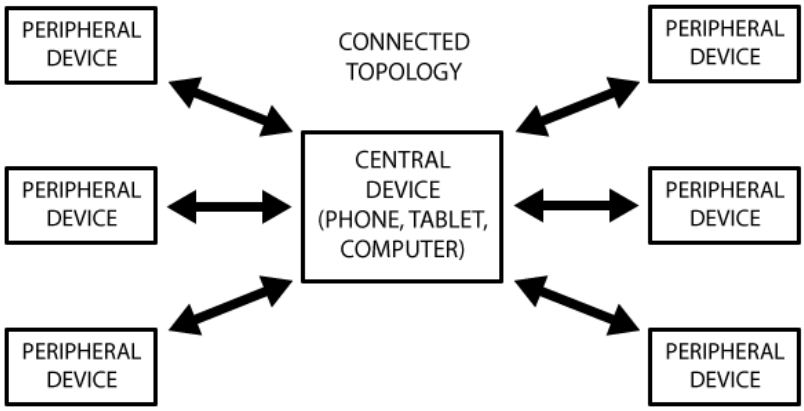


Figure 3.1: GATT connection

An important concept to understand related with GATT is the server/client relationship. On one side, the peripheral is known as the *GATT Server*, which holds the Attribute Table inside the Services and Characteristics definitions, and on the other side, the GATT Client (in this case, computer), is sending requests to the server (peripheral device). The GATT structure is composed by Services, Characteristics and Descriptors in this order and as seen previously, each group has its own Attribute Table.

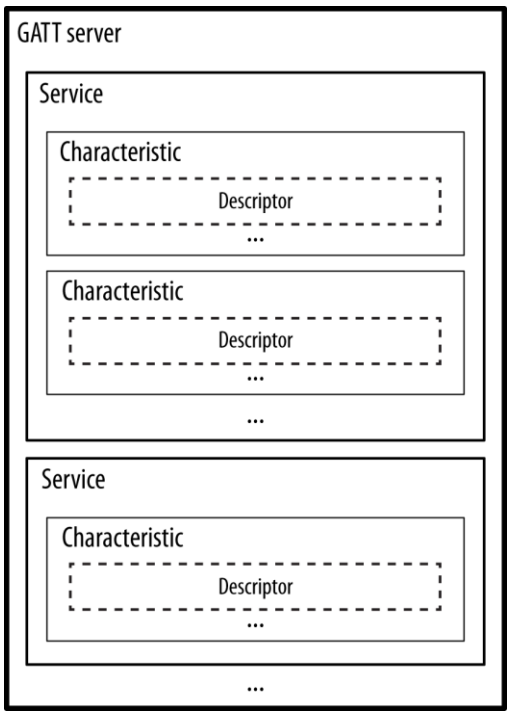


Figure 3.2: GATT Structure

All transactions are started by the Client, which receives response from the server. When establishing a connection, the peripheral will suggest a *Connection Interval* to the Client and the Client will try to reconnect in every Connection Interval to see if new data is available. This connection interval is just a suggestion, so the central devices maybe is not

able to honour the request because it is busy. The following diagram illustrates the process between a Server and a Client:

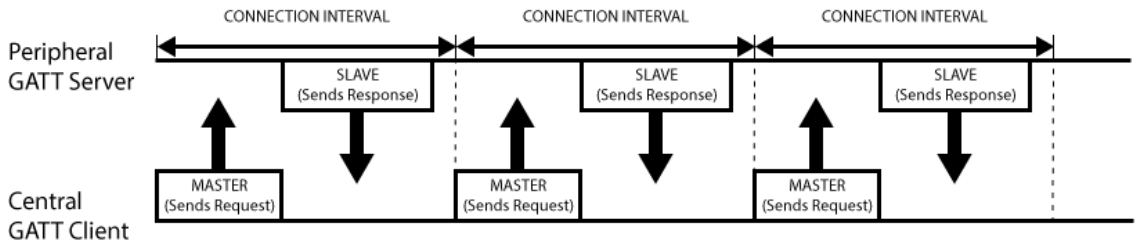


Figure 3.3: GATT transactions

It is worth mentioning that GATT roles are both completely independent and concurrently compatible with each other. That means that both a central device and a peripheral device can act as a GATT client or server, or even act as both at the same time.

3.2.1.1. Attributes

Attributes are the smallest data entity defined by GATT. They are addressable pieces of information that can contain relevant user data or metadata about the structure.

Conceptually, attributes are always located on the server and accessed by the client. Because attributes contain both static definitions of invariable nature and actual user data that is bound to change rapidly with time, attributes are usually stored in a mixture of non-volatile memory and RAM.[1]

Each attribute contains information about the attribute itself and then the actual data. The different attributes that are important for the correct understanding of this thesis are described below:

Type

The attribute type is the UUID[4]. This can be a 16-, 32- or 128-bit UUID taking up 2, 4 or 16 bytes respectively.

The type determines the kind of data present in the value of the attribute (values attributes are explained below), and mechanisms are available to discover attributes based exclusively on their type.

Although the attribute type is always a UUID, many kinds of UUID can be used to fill in the type. They can be standard UUIDs that determine the layout of the GATT server's attribute hierarchy

[4] Universally Unique Identifier

Permissions

Permissions specify which GATT operations can be executed on each attribute and with which security requirements.

GATT permissions:

- Access permissions: Determine whether the client can read, write or read and write an attribute value. Types of access permissions:
  - None: The attribute can neither be read nor written.
  - Readable: The attribute can be read.
  - Writable: The attribute can be written.
  - Readable and writable: The attribute can be both read and written.
- Encryption: Determines whether a certain level of encryption is required for this attribute to be accessed. Encryption permissions:
  - No encryption: The attribute is accessible, non-encrypted connection.
  - Unauthenticated encryption required: The connection must be encrypted to access this attribute, but the encryption keys do not need to be authenticated.
  - Authenticated encryption required: The connection must be encrypted with an authenticated key to access this attribute.
- Authorization: Determines whether user permission is required. Types:
  - No authorization required: Access to the attribute does not require authorization.
  - Authorization required: Access requires authorization.

All permissions are independent form each other and can be combined by the server which stores them.

Value

The attribute value holds the actual that of the attribute. There are no restrictions on the type of data contained but the maximum length is 512 bytes. Depending on the attribute type, the value can hold additional information about attributes themselves or other data. This part of the attribute is freely access for the client to both read and write.

The whole set of attributes can be seen as a table, with each row represents an attribute and a column represents the different parts of the attribute like in the following example:

Type	Permissions	Value
$UUID_1$ (16-bit)	Read only, no security	0x180A
$UUID_2$ (16-bit)	Read only, no security	0x2A29
$UUID_3$ (16-bit)	Read/write, authorization required	"a readable UTF-8 string"
$UUID_4$ (128-bit)	Write only, no security	{0xFF, 0xFF, 0x00, 0x00}
$UUID_5$ (128-bit)	Read/write, authenticated encryption required	36.43
$UUID_1$ (16-bit)	Read only, no security	0x1801

Table 3.1: Attributes

3.2.1.1.1. Services

A Service is a container for logically related Bluetooth data items. It can be thought of as the owner of the Characteristics inside it as seen in *Figure 3.2: GATT Structure*. Often, a Service represents a particular feature of a device. Each Service has its own ServiceName. The attributes of the service are always configurated this way:

- Type: always 0x2800 (Standard UUID for Service Declarations).
- Permissions: always Read Only without any authentication or authorization required.
- Value: an UUID defining what kind of service is.

In the case of Nano33BLE, the Services Table is the one shown below:

ServiceName	ServiceUUID
"Generic Access"	"1800"
"Generic Attribute"	"1801"
"Body Composition"	"181B"

Table 3.1: Nano33BLE Services

The table contains two columns: ServiceName corresponds to attribute *Type* and ServiceUUID corresponds to attribute *Value*.

3.2.1.1.2. Characteristics

The Characteristics declaration is similar to the Service declaration. They are items of data which relate to a particular internal state of the device[3]. In any of the cases, the way a device can expose data to other devices is by making them available as a Characteristic. Configuration of Characteristic attributes:

- Type: always 0x2803, the standard UUID for Characteristic Declarations.
- Permissions: always Read Only without any authentication or authorization required.
- Value: contains a handle, a UUID and properties. These elements describe the subsequent Characteristic Value Declaration.
    - Handle: points to the Characteristic Value Declaration in attribute table.
    - UUID: what type of information is expected to find in Value.
    - Properties: how the characteristic value can be interacted with. Different properties shown in the following table:

Properties	Value (Bit field)	Description (From BLE Core Specification v4.2)
Broadcast	0x01	If set, permits broadcasts of the Characteristic Value using Server Characteristic Configuration Descriptor. If set, the Server Characteristic Configuration Descriptor shall exist.
Read	0x02	If set, permits reads of the Characteristic Value
Write without response	0x04	If set, permit writes of the Characteristic Value without response
Write	0x08	If set, permits writes of the Characteristic Value with response
Notify	0x10	If set, permits notifications of a Characteristic Value without acknowledgement. If set, the Client Characteristic Configuration Descriptor shall exist.
Indicate	0x20	If set, permits indications of a Characteristic Value with acknowledgement. If set, the Client Characteristic Configuration Descriptor shall exist.
Authenticated Signed Writes	0x40	If set, permits signed writes to the Characteristic Value.
Extended Properties	0x80	If set, additional characteristic properties are defined in the Characteristic Extended Properties Descriptor. If set, the Characteristic Extended Properties Descriptor shall exist.

Table 3.2: Characteristic value properties

In the case of Nano33BLE board, the table of Characteristics is the one shown below divided in two parts:

ServiceName	ServiceUUID	CharacteristicName
"Generic Access"	"1800"	"Device Name"
"Generic Access"	"1800"	"Appearance"
"Generic Attribute"	"1801"	"Service Changed"
"Body Composition"	"181B"	"Body Composition Measurement"
"Body Composition"	"181B"	"Custom"

Table 3.3: Table of Characteristics I

CharacteristicUUID	Attributes
"2A00"	{["Read"]}
"2A01"	{["Read"]}
"2A05"	{["Indicate"]}
"2A9C"	{["Read" "Notify"]}
"C6B9F11B-F7BF-4F7C-84E0-FF810B38C248"	{["Read" "Write"]}

Table 3.4: Table of Characteristics II

The table contains five columns in total: ServiceName and ServiceUUID which come from the previous table of services, CharacteristicName (*Type* attribute), CharacteristicUUID (*Value UUID* attribute) and Attributes (*Value Properties* attribute).

When accessing a specific characteristic with Matlab (in this example the “2A9C”), the characteristic properties are displayed:

```
>> c = characteristic(b, "181B", "2A9C")

c =

  Characteristic with properties:

              Name: "Body Composition Measurement"
              UUID: "2A9C"
        Attributes: "Read" "Notify"
       Descriptors: [1x3 table]
  DataAvailableFcn: []
```

Figure 3.4: Characteristic properties

As is shown in the table, the descriptors of the characteristic appear for the first time in a table format. Also appear the DataAvailableFcn which is a property that calls the handle function whenever a new notification is received from the handle. This property will be explained deeply later.

3.2.1.1.3. Descriptors

Descriptors are mostly used to provide the client with additional information about the characteristic and its value. They are always placed within the characteristic definition and after the characteristic value attribute. Descriptors are always made of a single attribute, the characteristic descriptor declaration.

In the case of Nano33BLE, when accessing a descriptor, the following table is shown:

DescriptorName	DescriptorUUID	Attributes
"Client Characteristic Configuration"	"2902"	{["Read" "Write"]}

Table 3.5: Table of Descriptors

As in Characteristic Table, the DescriptorName is the *Type* attribute, the DescriptorUUID which is the *Value* attribute and the Attributes column that contains the *Property Value* attribute.

If going inside a Descriptor, the table displayed is the following:

```
Descriptor with properties:

        Name: "Client Characteristic Configuration"
        UUID: "2902"
  Attributes: ["Read"    "Write"]
```

Figure 3.5: Descriptor properties

The table contains the Name of the Descriptor, its UUID and its attributes, the same as seen in the previous Table of Descriptors. It exists two types of descriptors:

- GATT-defined descriptors: add metadata about the characteristic
- Profile or vendor-defined descriptors: can contain all types of data, including additional information that complements the reading itself.

3.2.2. Matlab connection

Once seen all the parameters of GATT and BLE software, the connection with Matlab must be done to start receiving some data. In this section, the connection between Matlab and the device is explained.

First, to connect the device and Matlab, the *ble* function is used where the address parameter can be the device name or the device address.

```
b = ble(address)
```

After the connection is established, Matlab has to subscribe to the characteristic which have to read data.

```
caracteristica = characteristic(b, "181B", "2A9C");
subscribe(caracteristica, "notification");
caracteristica.DataAvailableFcn = @plot_update;
c = characteristic(bio_device, "181B", "c6b9f11b-f7bf-4f7c-84e0-
ff810b38c248");
write(c, config_data);
```

Function characteristic() access to the characteristic to read, subscribe or write in there. It has three inputs, taking the first call inside the code:

- b: Name given to the ble object created previously
- “181B”: Service name or Service UUID of the characteristic where data has to be read.
- “2A9C”: Characteristic name or Characteristic UUID extracted from the Characteristics table explained in section *3.2.1.1.2*.

Function subscribe() picks up the data that comes from the characteristic. It has two input parameters:

- caracteristica: characteristic object created with function characteristic.
- “notification”: subscription type. This parameter is optional and it can be “notification” or “indication”.

Function DataAvailableFcn calls the callback, in this case the function plot_update() when new data is read from the previous characteristic.

Function write() writes the passed as an input parameter inside the characteristic. Two inputs:

- c: The characteristic where the data has to be written
- config_data: The data that has to be written. It is important that the data is passed to the function in the correct format defined in Arduino because otherwise Arduino will not be able to write the data in the correct characteristic. In this case, according to Arduino, the data has to be send with this format so the conversion is done before sending the data to the Nano33BLE:

```
/* Configuration command structure */
struct ConfigData {
  float BiaODR;
  int32_t NumOfData;
  float SinFreq;
  BoolFlag SweepEn;
  float SweepStart;
  float SweepStop;
  uint32_t SweepPoints;
  BoolFlag SweepLog;
};
```

Figure 3.6: Arduino configuration

Inside function plot_update() a reading to the characteristic is done and then all the data picked up is printed in the interface explained in the next section.

3.3. User Interface – Matlab

In this section the whole user interface created to manage the data sent to the Nano33BLE board is explained.

The interface is divided in three screens, each screen has its own utility, and they are described below. The sections are: generate signal, visualize signal and model adjusting.

All the code used to create the interface is attached in the Appendix I of the document.

3.3.1. Main page

The first screen of the user interface is the Main Page. In this section the user can choose which functionality of the program want to use. When clicking in the desired functionality, another screen appears.

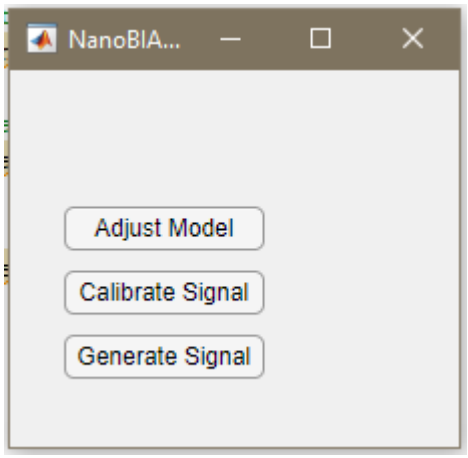


Figure 3.7: Main Page

3.3.2. Generate Signal

The first screen is the Generate Signal one. It appears when clicking in “Generate Signal” button of the Main Page.

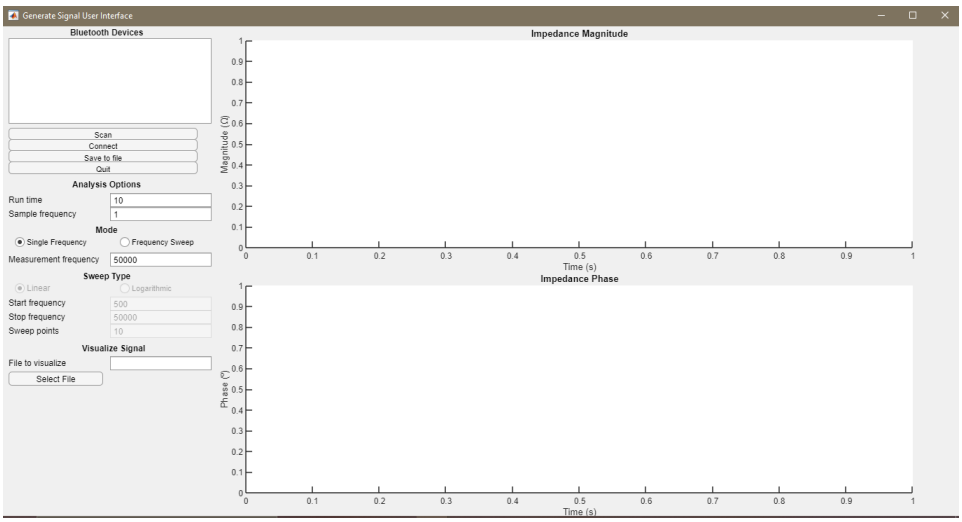


Figure 3.8: Generate signal screen

In this section the user can set up the parameters of the data send to the board, then see the impedance signal displayed in the graphs and in the end, save the signal to a file. It also has the functionality to visualize an acquisiton done before.

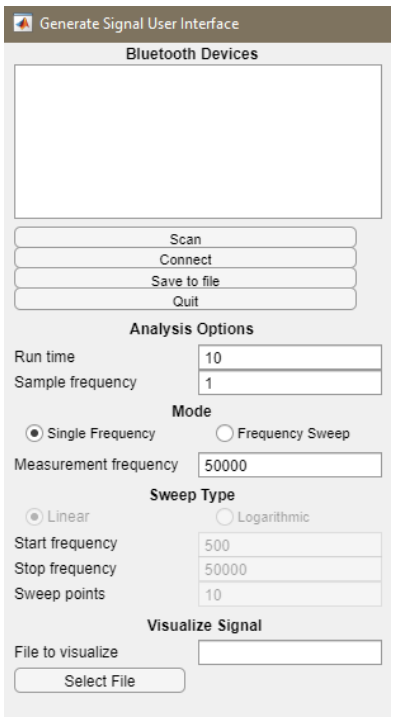


Figure 3.9: Left panel of generate signal screen

Looking more deeply at the left panel, a Bluetooth Devices list can be seen. This blank list is filled with all the BLE Devices available after clicking *Scan* button. *Scan* button checks all the BLE devices found by the computer and then this list is passed to the text area to be printed there.

After clicking Scan, the configuration parameters can be settled with the *Analysis Options* menu, the *Mode* menu and the *Sweep Type* menu if needed. The functionality of this subsections of the interface is explained in the following parts of the document more deeply.

Once the configuration is set, the next step is to print the signal in the graphs. This action is done by the *Connect* button, which checks if some BLE Device is picked from the Bluetooth Devices list and then, it connects the Matlab with the Nano33BLE and send the parameters configured before. If no parameters are settled, connect function send the default ones.

After sending the parameters, the data received is printed in the graphs as seen on the following example:

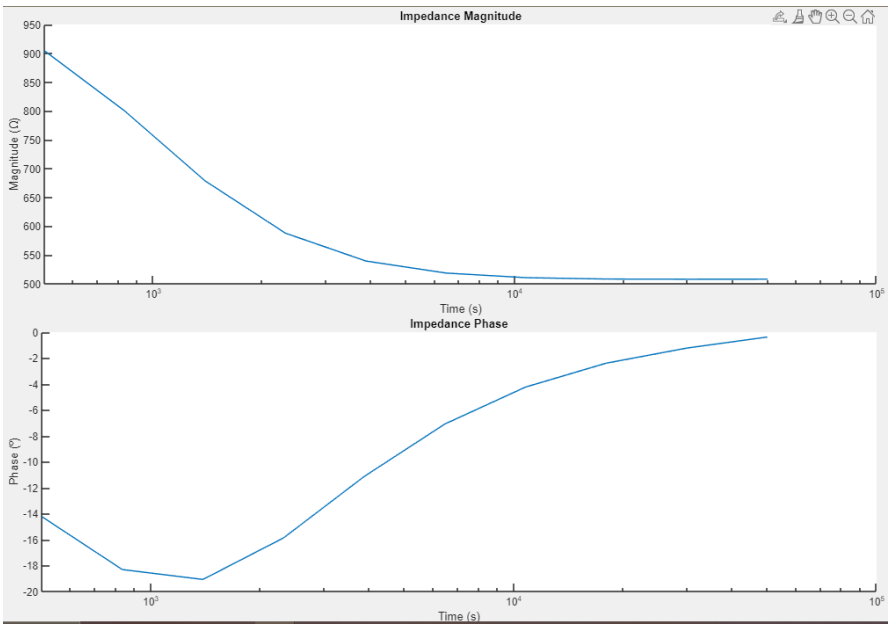


Figure 3.10: Example of graphs showing generated signal

The first graph (the top one) is showing how the magnitude of the impedance measured varies and the second graph (at the bottom) shows the variation of the impedance phase for a logarithmic frequency sweep.

Once the measure is done, the user can choose between saving the results to a file or exiting the interface. If the decision is saving results to a file, the user has to select the button *Save to file* and then a pop-up saying that the results are correctly saved will appear.

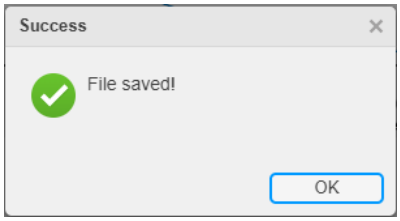


Figure 3.11: Pop-up file saved

The results are saved in a file located inside the Matlab folder and the name of the file is *Results.txt.* The results are saved in a specific format, the file saved contains a structure like the following:

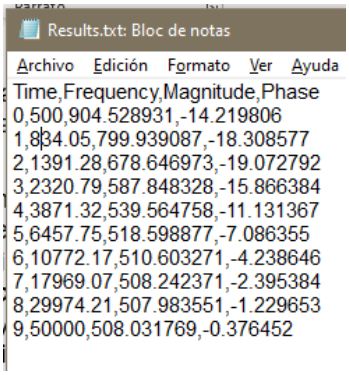


Figure 3.12: Example of file saved

The file contains four columns, the first one contains the values for time, the second column, the ones for frequency and the third and the fourth the values for magnitude and phase, respectively. These values will be explained in the following sections. The format of the file is important to allow the program read data a posteriori. If the file is already created, the new results will be attached following the existing ones.

If the other action is taken and the user decides to exit the interface, button *Quit* must be selected. After clicking it the following pop up will appear:

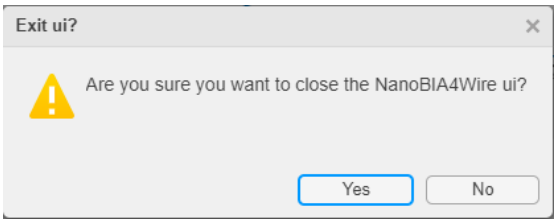


Figure 3.13: Pop-up to close the ui

Yes option will close the interface and *No* option will close the pop-up and will maintain the interface opened as before.

3.3.2.1. Single frequency

Inside the left menu of the previous screen (Generate Signal) two options can be selected, in this section the first one will be explained: Single Frequency.

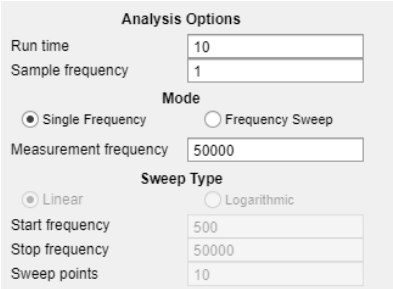


Figure 3.14: Single Frequency configuration

The panel disables all of Sweep Type menu because as seen in next section, this menu only applies to Frequency Sweep mode.

Single frequency mode works only with one frequency where the program will read all the data. The frequency chosen is the one written by the user in *Measurement frequency* field. *Run time* indicates the lapse of time when the program is reading data from the board. *Sample Frequency* is the number of samples read per second.

Summarizing, this mode allows the user to read data from the board during a period of time at a particular frequency, then the results are printed on the graphs. The axles will be Time(s) and Magnitude(Ω) for the x and y axles of the top graph and Time(s) and Phase(º) for the x and y axles of the bottom graph.

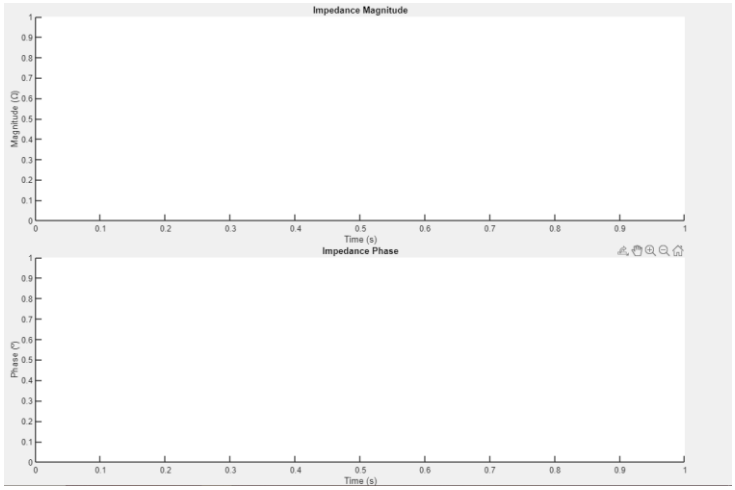


Figure 3.15: Axles for Single frequency mode

3.3.2.2. Frequency sweep

The other option in the screen is the Frequency Sweep one. It has a lot of functionalities because it works with more than one frequency, making a sweep between them.

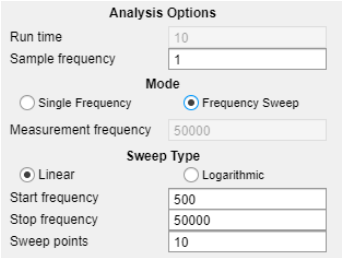


Figure 3.16: Sweep Type configuration

First of all, when the option is selected, *Run time* field is disabled and not editable anymore, same happens with *Measurement frequency.* In the other side, the fields blocked in Single frequency mode are editable now, *Start frequency, Stop frequency* and *Sweep points.* This happens because as mentioned before, the mode works with more than one frequency, doing a swept between two frequencies indicated by the fields *Start frequency* and *Stop frequency.* The parameter *Sweep Points* indicates the number of points picked from the frequency range evaluated.

In the case of the image, the board will return ten values between 500Hz and 50kHz.

As seen in the image, it also exists two types of swept: Linear and Logarithmic. The first one (Linear) returns the data in linearly spaced frequencies, so in the image example, it will return the following frequency values: 500Hz, 5450Hz, 10400Hz, 15350Hz … With a space of 4950Hz between each value.

The second type of swept is Logarithmic, which returns the data in a range of frequencies logarithmically spaced. In the image example, if Logarithmic was selected, the data returned for frequencies will correspond to approximately these values: 500Hz, 834Hz, 1391Hz, 2320Hz … They are spaced 334Hz, 557Hz, 929Hz which are growing interfrequency distances.

After the data is collected, it is displayed in the graphs, and if Logarithmic is selected, the axles of the graphs are configurated to show a logarithmic scale. In this case, the axles of the graphs are: Frequency(Hz) for x axis and Magnitude(Ω) and Phase (º) for the y axis in the top graph and bottom graph respectively as seen in the following image:

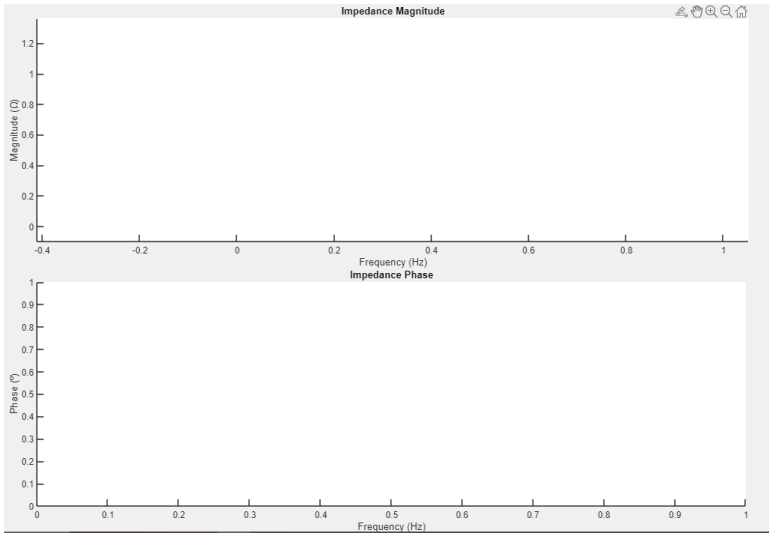


Figure 3.17: Axles for frequency sweep mode

All this parameters, frequency sweep and single frequency ones, are send it to the board and then the board returns the values for Magnitude and Phase and Matlab print them in the graphs.

3.3.2.3. Visualize signal

The last option inside Generate Signal screen is the visualization of a previously generated signal, explained in this section.

This option allows the user to visualize a Signal that has been generated as explained before through the interface located at the bottom of the left panel.

Figure 3.18: Visualize Signal section

In this menu, the user can select the file where the results of the signal generated are saved. When pressing the button, a pop up will appear allowing the user to select the correct file. This must be a txt text with the same pattern as the one in *Figure 3.12.: Example of file saved.* If the user uploads a file that does not correspond with that format, the interface will throw an exception.

By the other way, if the file uploaded is correct, the path is attached to the text area next to *File to visualize* field (see *Figure 3.18: Visualize Signal section*) and then the signal is printed into the graphs but in this case with the results read from the file.

This functionality is only available for results coming from a logarithmical frequency sweep because is the more used configuration for this functionality.

3.3.3. Calibrate signal

The second screen of the interface is the one used to calibrate a signal. The interface looks like the following image. As seen here, the graphs have frequency in x-axis

because as same as in visualization signal, the logarithmical frequency sweep is the most used configuration for this functionality.

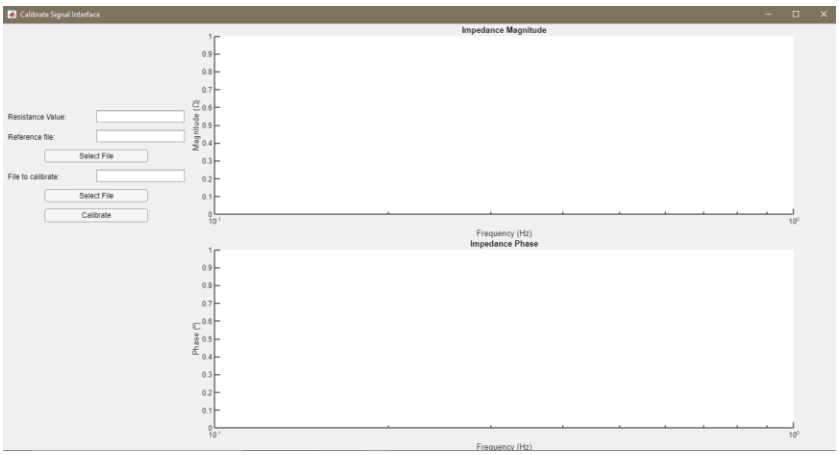


Figure 3.19: Calibrate signal interface

Looking in the left panel of the screen, these parameters are found. The file the user wants to calibrate is uploaded in the field *File to calibrate* with the correct format explained before. When clicking *Select File* button, a file browser appears to select it. After this, the path is attached in the text area located next to *File to calibrate*.

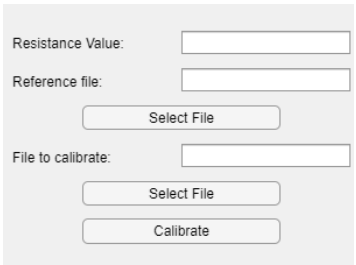


Figure 3.20: Configuration menu of calibrate signal screen

The next step is to upload a file with the results calculated for the calibration impedance. These results have to be calculated with the same configuration as the results to calibrate. For example, if the reference impedance is 100Ω, the file has to contain the measurements obtained with a 100Ω resistance.

$$|Z_c| = |Z_m| * \frac{R_{teo}}{|R_{cal}|} \qquad \varphi_c = \varphi_m - \varphi_R$$

This file is uploaded selecting the top *Select File* button and following the same procedure as for the file to calibrate, the reference file will be saved. The field *Resistance Value* has to contain the impedance at which the reference file results are taken. In the previous example, that field will be filled by 100Ω.

The last step when everything is configured is click *Calibrate* button and then, both signals will appear printed on the graphs as seen in the following image:

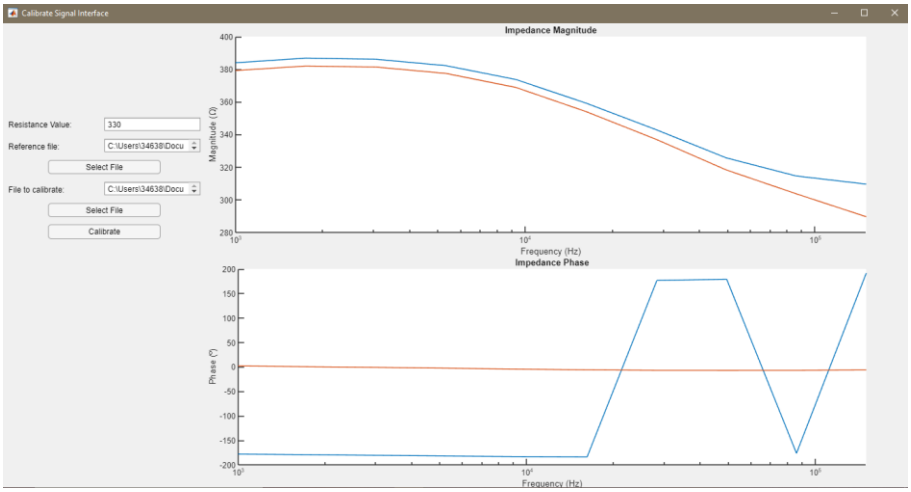


Figure 3.21: Calibrate signal example

In the image the two signals are shown. The blue one is the reference signal, and the red one is the signal to calibrate. Then, the calibrated signal is saved in a file with the same format as in *Generate Signal* screen.

3.3.4. Model adjusting

The last screen of the interface is the model adjusting one. This interface shows a comparison between the results of the signal generated before and the Cole-Cole Arc function[5]. The screen looks like the following image:

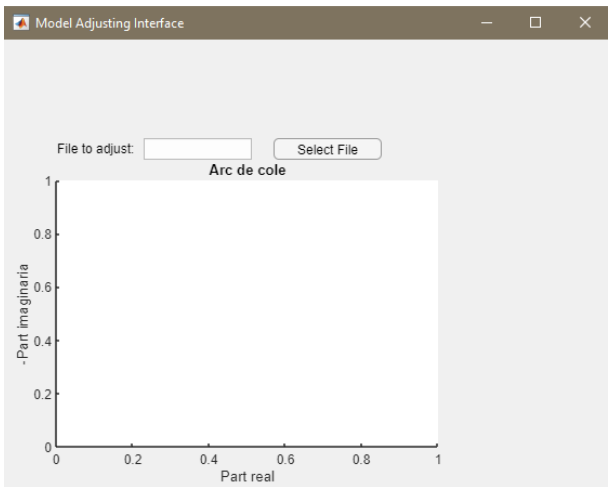


Figure 3.22: Model adjusting screen

The file can be uploaded with the *Select File* button. This button opens a prompt where the user can select the desired file to be compared with the Cole-Cole function. After selecting it, the path of the file is shown in the text area next to *File to adjust* and the graph will print the file uploaded and the Cole-Cole function in the same graph to compare them.

The graph axles show the real part and the (negative) imaginary part of the impedance calculated in x and y axles, respectively. In the representation of the Cole-Cole model, the imaginary axis is usually represented with the negative sign because all

[5] See section *3.3.4.1. Cole-Cole Arc* for further information

biological reactances are capacitive and, in that way, they fit in the 1st quadrant. The file uploaded must be configurated with the correct format, in which the impedance is given by module and phase so Matlab will transform the values into binomial mode.

An example of the final result of this functionality is the one seen in the following image, the blue graph is the Cole-Cole arc and the red points are the ones generated with the file selected previously:

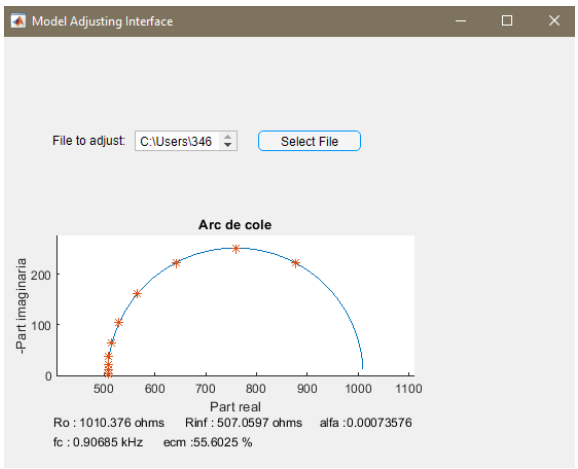


Figure 3.23: Model adjusting example

As seen in the image, five parameters are printed below the graphs. These points are generated inside the Cole-Cole function, and they are explained in the next section.

3.3.4.1. Cole-Cole Arc

The model used to fit the data is the Cole-Cole impedance function, which in a Nyquist plane (Im(z) vs Re(Z)) traces the arc of a circumference. A Matlab function fits the impedance data to the equation of a circumference and calculates the values of the real and imaginary parts of the impedance model at the frequencies swept in by the file uploaded. These values are displayed forming an arc which starts at R_∞ (left side of the arc) and ends in R_0 (right side of the arc). The middle of the arc (the top) corresponds to the central frequency of the impedance relaxation, which corresponds to the maximum of the imaginary part.

Finally, the α parameter corresponds to the angle with the horizontal axis in which the central point of a circumference that can contain the arc would be found and which is related with the cell size and shape dispersion. The Cole-Cole equation of the impedance function is the following[7]:

$$Z_{\text{Cole}}(\omega) = R_\infty + \frac{R_0 - R_\infty}{1 + \left(j\frac{\omega}{\omega_c}\right)^\alpha}$$

These parameters are the ones that appear below the graph in previous section. Just one value is missing that is *ecm* but it is not used to print the values in the graph. This parameter is informative, and it indicates the mean square error of the values calculated to give an estimation about the reliability of the results obtained.

4. Results

In this section the results obtained with the BLE connection between the Arduino NANO-AD5940 board and the developed Matlab interface are shown. These results have been obtained through different experiments that show the functionality of the UI described.

4.1. Non-biological measures

4.1.1. Linearity – Range

The first experiment was done measuring different resistances with a reference multimeter (orange line) and with the NANO33BLE/AD5940 board (blue line). Both measures have been taken 50kHz.

As seen in the image and the table, the experimental measure behaves linearly until 20kΩ, where the measured impedance displays a saturation.

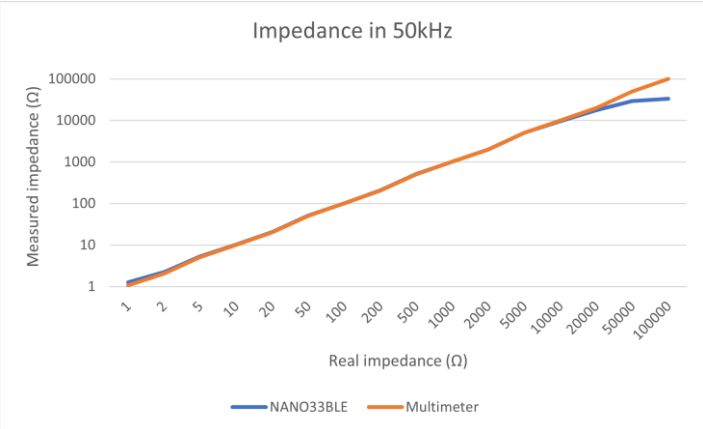


Figure 4.1: 50kHz measurements

	Measured impedance(Ω)		
Real impedance (Ω)	NANO33BLE	Multimeter	Absolute error(Ω)
1	1,2592	1,067	0,1922
2	2,224	2,042	0,182
5	5,29	5,08	0,21
10	10,235	9,97	0,265
20	20,48	20,05	0,43
50	50,895	50,05	0,845
100	102,11	100,4	1,712
200	205,61	202,3	3,31
500	514,12	506,4	7,72
1000	1019,54	1004,8	14,74
2000	2006,58	1981,7	24,88
5000	5088,9	5079	9,9
10000	9618,5	9890,8	272,3
20000	17628	20109	2481
50000	29240	50218	20978
100000	33340	99946	66606

Table 4.1: Graph data

Following this example, another experiment was to measure the linear subset of the impedances in three different frequencies set by the UI. The frequencies bands are 1kHz, 50kHz and 150kHz. As seen in the figure, there is a slight difference between the

measurements at the different frequencies, more visible at 150kHz, which means that the calibration coefficients have to be obtained at all frequencies.

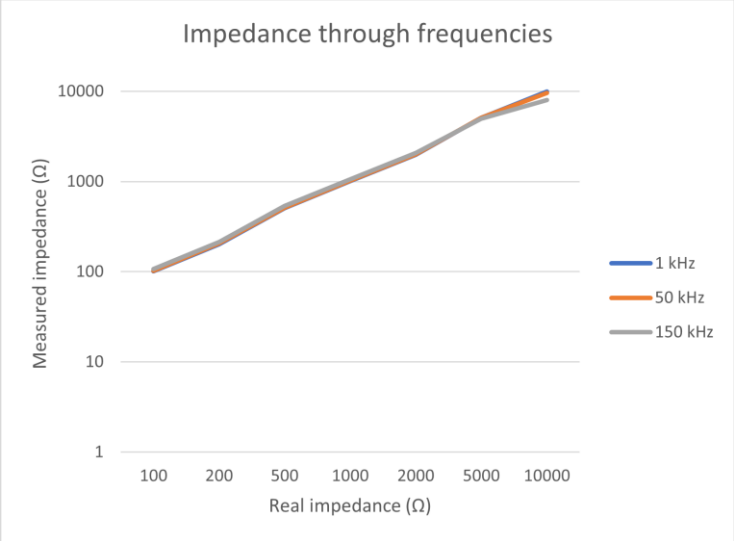


Figure 4.2: Impedance in different frequencies

	Measured impedance(Ω)			
	Multimeter	NANO33BLE		
Real impedance (Ω)		1kHz	50kHz	150kHz
100	100,4	100,94	102,11	106,415
200	202,3	203,15	205,61	214,7
500	506,4	508,4	514,12	537,56
1000	1004,8	1009	1019,54	1062,05
2000	1981,7	1989	2006,58	2071,45
5000	5079	5099	5088,9	4944,5
10000	9890,8	9926	9618,5	8041

Table 4.2: Graph data

4.1.2. Noise – Dispersion

The second experiment consists in measuring the noise and dispersion of a measure. To do this, an impedance of 200Ω has been used, because it is representative of several bioimpedance applications. The UI was running during 50 seconds with 2 samples per second, giving a total of 100 measures, the ones represented in the image. Regarding it, it can be observed that some noise appeared during the working time, but checking the y-axis, the total variation of the impedance magnitude is approximately 0.03Ω.

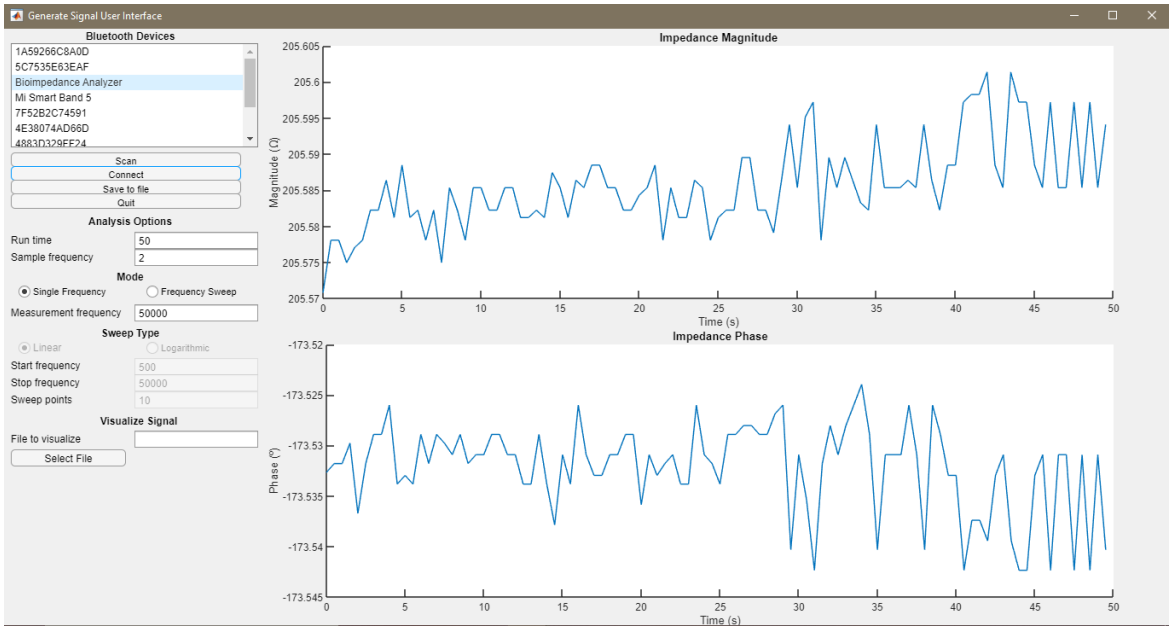


Figure 4.3: Noise-Dispersion experiment

To check numerically the behaviour, the average of the measures has been done and the result was 205,59Ω which is pretty accurate to the real impedance. Also, the variation and 2.6*standard deviation, which would include the 99% of the measurements, were calculated, and they gave minimum results for both.

	Magnitude	Phase
Average	205,5863657	-173,527426
σ	9,1881E-05	0,00229193
2,6*σ	0,000238891	0,00595903

Table 4.3: Statistical values for noise-dispersion experiment

4.1.3. RC net – Model adjusting

The next experiment has been performed with a RC net built with two circuits connected in parallel. The first circuit is constituted by a resistor of 1008Ω and the second one is another resistor of 1008Ω in series with a capacitor of 99,7nF (measured values).

The impedance is shown in the next figure, and it has been generated with a swept between 500Hz and 50kHz and it gave values from 900Ω to 500Ω approximately.

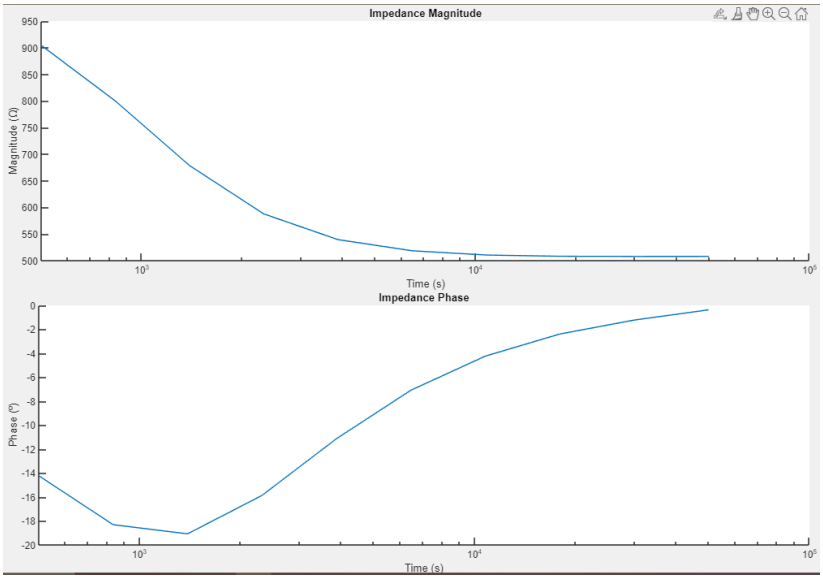


Figure 4.4: RC net signal

When checking the Cole-Cole function fitting with the signal generated, it can be seen that the graphs are apparently the same, and the R_0 value is 1010Ω which is almost 1008Ω that is the real value of the impedance while R_∞ is 507Ω (should be 504Ω).

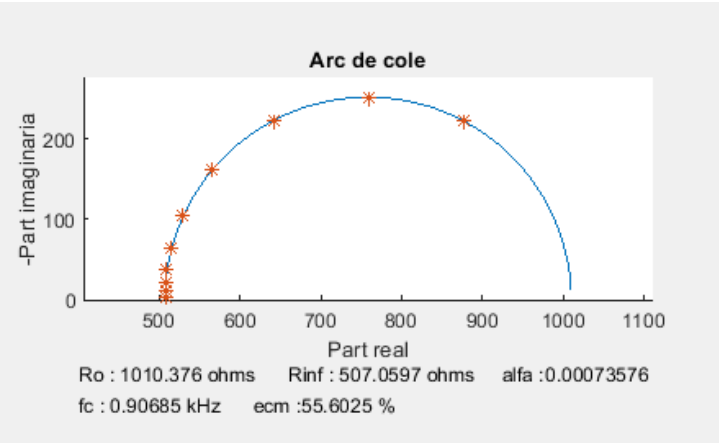


Figure 4.5: Cole-Cole arc with RC net

4.2. Biological measures

After the non-biological measures, some biological measures have been done to check the correct behaviour with a real environment.

4.2.1. Inert object – Apple

The first experiment has been carried out connecting the board to an apple through 4 stainless-steel rod electrodes, that represents an inert biological object as shown in the following image.

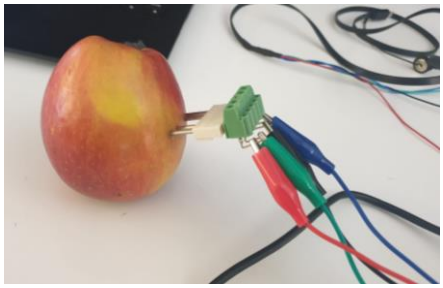


Figure 4.6: Apple connected to the board

The impedance signal generated is this one:

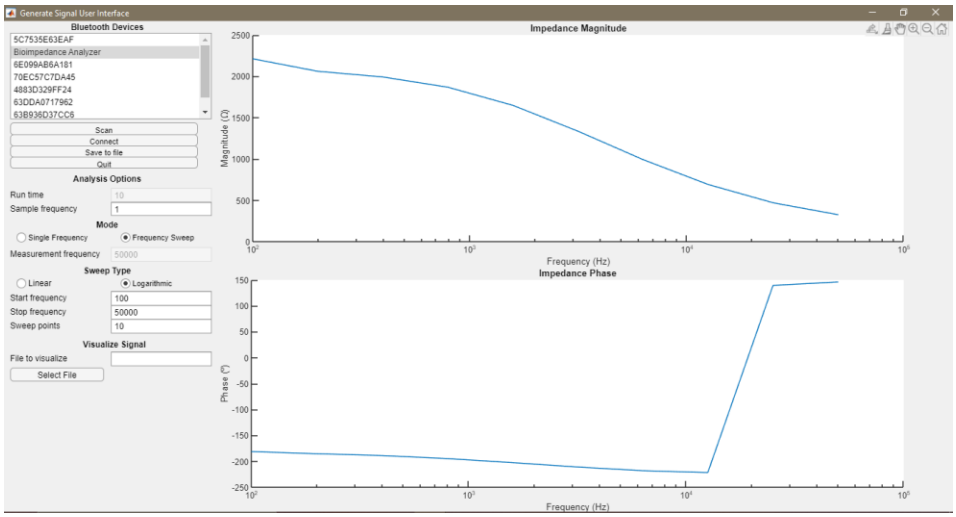


Figure 4.7: Signal coming from the apple

There is a 360º leap in the phase determination, which will be corrected in the calibration process.

In the image it can be seen that the mean impedance received from the apple is nearly 2000Ω so the calibration will be done with a resistor of 2kΩ as shown in the following figure:

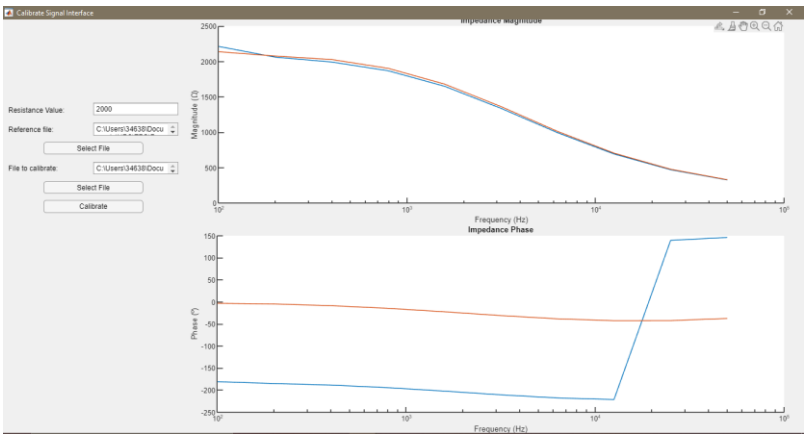


Figure 4.8: Calibration of the signal coming from the apple

Comparing the signal with the Cole-Cole function, the first experimental deviations can be seen due to the residual effect of the electrodes at low frequency. Also, the curve of the semi circumference becomes more flattened, with an α parameter of 2.0 due to the dispersion of sizes and shapes of the cells. The model fitting parameters provide information about the cellular structure of the apple.

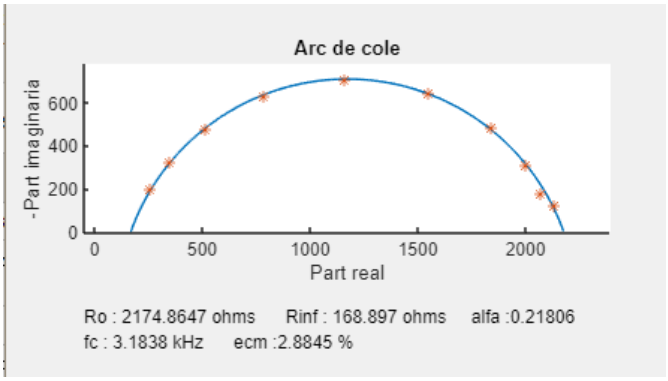


Figure 4.9: Cole-Cole function applied to an apple

4.2.2. Body measures

Inside the biological measures, there is a subsection that is the body measures explained in this section. The measurements are performed on a volunteer (the thesis supervisor). The circuit contains series protection capacitors and it is battery operated.

4.2.2.1. Static body

The first experiment executed was the evaluation of the signal received from a relaxed body. The electrodes were placed as in the image on the right, two in one arm and the other two in the same side leg (right side impedance).

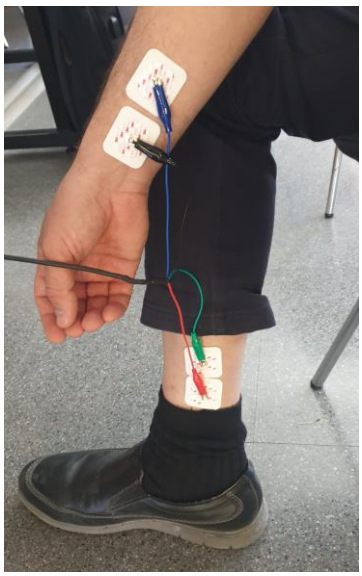


Figure 4.10: Static body - Electrodes placement

With this configuration the signal received can be calibrated and adjusted to the cole-cole model and the body mass index can be retrieved.

First, the signal received from a swept of 10 points between 1kΩ and 150kΩ is the one in *Figure 4.12*.

As seen in it, the highest impedance measured is approximately 390Ω and the lowest is 310Ω so a good calibration impedance can be 330Ω.

The fact that the impedance is changing between -180º and 180º is solved with the calibration, this effect is caused by how Matlab interprets the data.

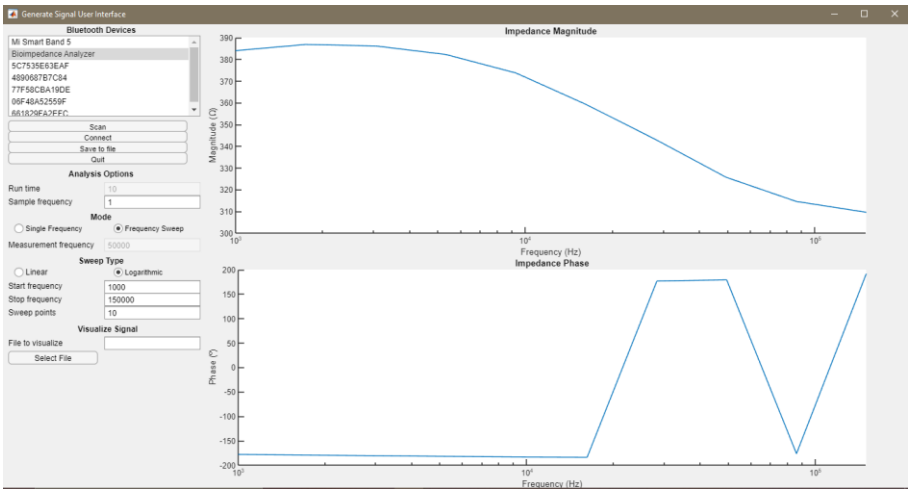


Figure 4.11: Static body - Signal generation

As explained before, the signal can be well calibrated with a 330Ω impedance and the phase error will be avoided:

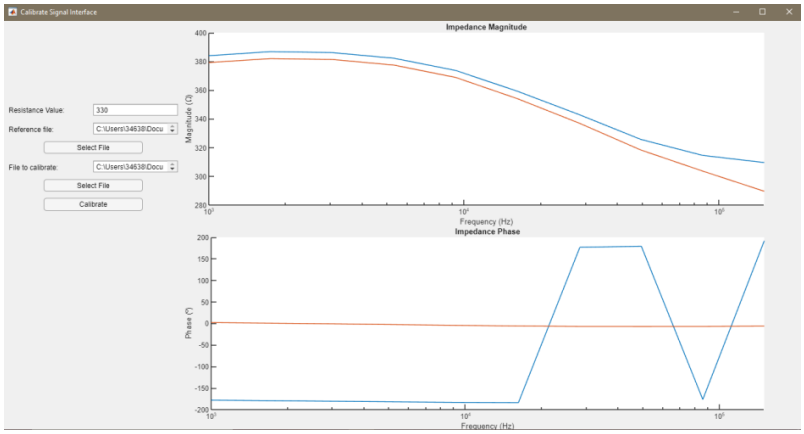


Figure 4.12: Static body – Calibration

Finally, comparing the signal after the calibration with the cole-cole arc function, it is noticed that the measurement does not exactly fit an arc.

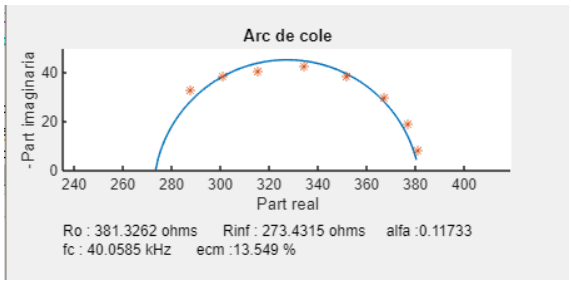


Figure 4.13: Static body - Model adjusting

After these measurements, the total body water can be calculated with the following expression taking into account that: W is the weight of the human (in this case 85kg), H is the height (170cm) and the parameters of the effective resistivity of the electrical fluid,

ρ_∞ = 104.3Ω·cm, dimensionless body shape factor k_b = 4.3 and the body density D_b = 1.05kg/L. The value of R_∞ is the one extracted from cole-cole function[11].

$$TBW = \frac{1}{100}\left(\frac{\rho_\infty k_b H^2 \sqrt{W}}{R_\infty \sqrt{D_b}}\right)^{\frac{2}{3}} = 56.66\ L$$

Now, assuming the value of the hydration constant k_h = 0.732, the fat free mass can be obtained:

$$FFM = \frac{TBW}{k_h} = 77.4\ kg$$

And subtracting the fat free mass from the weight:

$$FM = W - FFM = 7.59\ kg$$

These results give a body mass index of:

$$BMI = \frac{FM}{W} = 0.089 \rightarrow 8.93\%$$

4.2.2.2. Arm contraction

The next experiment is realized putting the electrodes in the arm as seen in the following image:

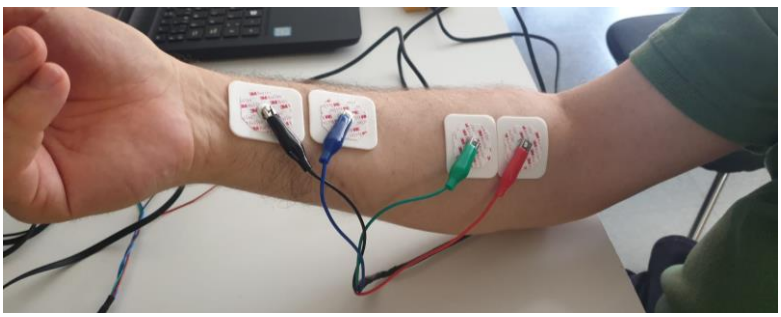


Figure 4.14: Arm contraction - Set up

The signal received during 10 seconds at 50kHz from the arm is the one seen in the Figure 4.16. The impulse (2% rise) between 2s and 9s happens because of tension exerted with the arm, that is why the impedance changed in this lapse of time due to the contraction (change in size and shape) of the muscle cells.

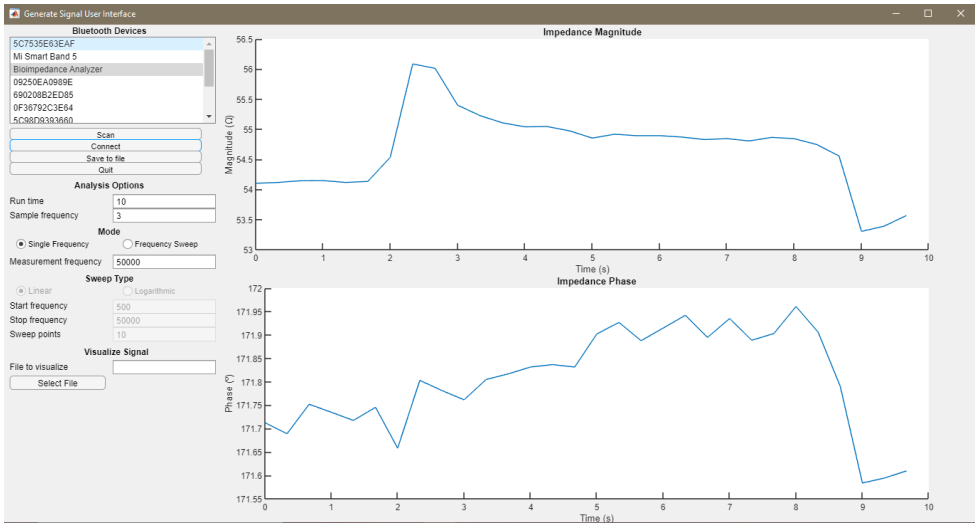


Figure 4.15: Arm contraction - Signal received

4.2.2.3. Breathing

This measurement is carried out with four electrodes connected to the arms of the person, both in each arm. It is useful to study the breathing of the individual, given that the lungs are located in the current path.

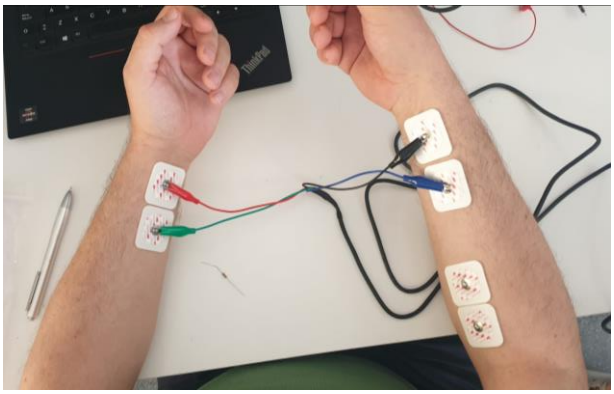


Figure 4.16: Breathing - Set up

The signal received from there is the one shown in the following image. During a period of 15 seconds with 3 samples per second at 50kHz which is the standard frequency for bioimpedance measurement. The peaks of the signal appear when the person is inhaling, and the valleys appear when he is exhaling, due to the low conductivity of the air. The phase angle at this frequency does not display the same sensitivity to the ventilation but is able to show the effect of the blood perfusion.

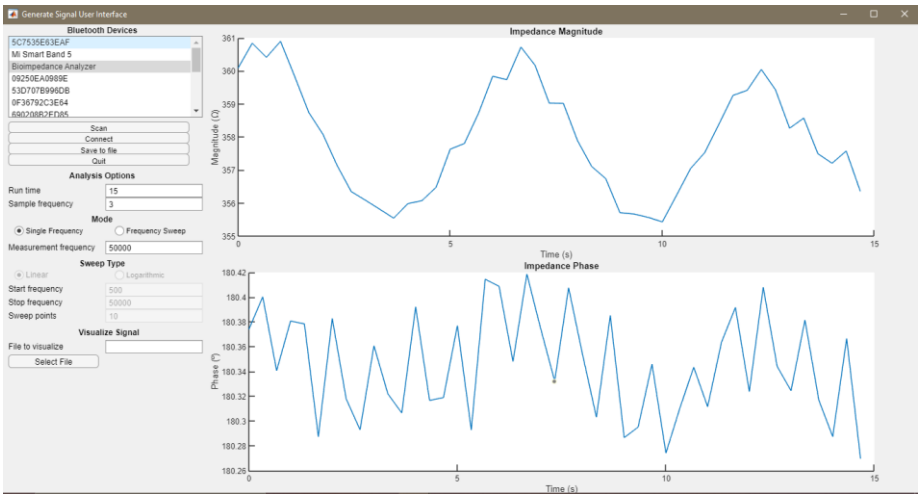


Figure 4.17: Breathing - Signal received

4.2.2.4. Apnea

The last experiment performed is similar to Breathing one. The set up is the same but in this case the individual has to maintain the air inside the lungs for a while, so the modulation induced by the ventilation should not be there and only the blood perfusion (at heart beat rhythm) should be detected. It can be seen both in the magnitude and phase angle although with 3 measurements per second, we are near the limit of the Nyquist sampling frequency.

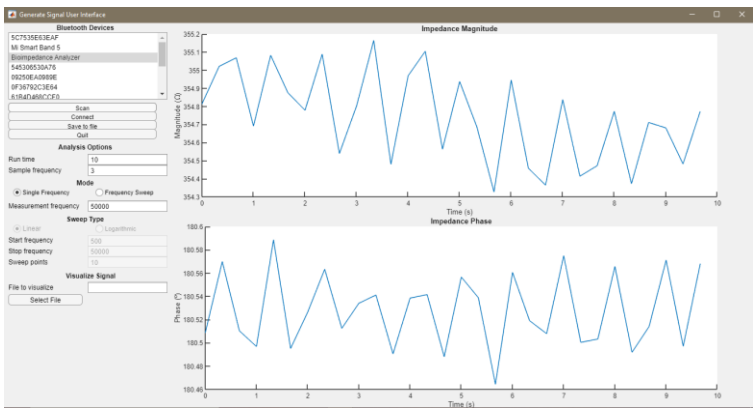


Figure 4.18: Apnea - Signal received

5. Budget

Material

Component	Prize (€)
NanoBLE33	22
EVAL-AD5940BIOZ	300
Electrodes	20
Wires	10
Total	352

Table 5.1: Material costs

Software

Software	Prize (€)
Matlab	800 (annual) → 270€ (4 months)
Total	270

Table 5.2: Software costs

Human resources

Resources	Prize (€)
Junior Engineer	10€/hour * 450hours = 4500€
Total	4500

Table 5.3: Human resources costs

Material	352€
Software	270€
Human resources	4500€
Total	5122€

Table 5.4: Total costs

6. Conclusions and future development:

Now, having the project finished, an assessment based on the results obtained can be done and new features can be proposed.

This project shows the design, implementation and test of a low-cost electrical impedance spectroscopy system based on an AD5940 and a Nano 33 BLE microcontroller board. A user interface to see the results obtained from the board is also developed to make the board more practical and materialize the results.

The experimental measurements have been done to check the functionality in a way closer to reality because the board has biological features that can be used in hospital or medical centres and the measures must be accurate to ensure the patients health.

After checking the results, it is seen that the user interface has limitations because the maximum number of Sample Frequency that can be set in the UI is 3 and the maximum numbers of sweep points is 10. Other things to improve of the board is that represents the measures in real time, but it has a delay of seconds between the measure is received by Arduino and it is represented in Matlab.

However, the board gives accurate results because these limitations do not affect the correct behaviour if the board since it works well taking this into account. The board extract the impedance values with minimum noise, and this is a key point in biological environment because it gives the measures accurate, and it is important for patients health when detecting some kind of diseases.

The fact that the user interface has the option of calibrate the signal can solve the problem of the non-exact accuracy because it will fix the practical errors and give the signal without noise. Also, the other option, which is model adjusting, has a lot of potential in biological aspects because for example, the flattening of the cole-cole semi circumference gives reliable information for human measures, another aspect of the cole-cole function is the way that the signal measured differs of cole-cole arc which also gives quality information for biological measures.

The next steps and improvements for this board can be correct the main limitations of the user interface, making it functional for a number of sweep points greater than 10 and for a sample frequency greater than 3.

Another way of improvement is to make the interface more faster when printing the signal received from Arduino but this is more related with how Matlab treats the data received from an external program.

Bibliography:

[1] M. Woolley. “A Developer’s Guide to Bluetooth”. [Online] Available: https://www.bluetooth.com/blog/a-developers-guide-to-bluetooth/ [Accessed: 05 June 2021]

[2] Martin BL. “Bluetooth Low Energy Characteristics, a beginner’s tutorial”. [Online] Available: https://devzone.nordicsemi.com/nordic/short-range-guides/b/bluetooth-low-energy/posts/ble-characteristics-a-beginners-tutorial [Accessed: 05 June 2021]

[3] K. Townsend. “Chapter 4. GATT (Services and Characteristics)”. [Online] Available: https://www.oreilly.com/library/view/getting-started-with/9781491900550/ch04.html [Accessed: 06 June 2021]

[4] Anonymous. “Electrochemical Impedance Spectroscopy (EIS)”. [Online] Available: https://www.palmsens.com/electrochemical-impedance-spectroscopy-eis/ [Accessed: 14 June 2021]

[5] L. Lax. “Disseny d’un sistema d’espectroscòpia d’impedància elèctrica basat en una placa de Red Pitaya”. UPC, Barcelona, 2019

[6] Dr. C. Shaffer. “Electrochemical Impedance Spectroscopy Applications”. [Online] Available: https://www.news-medical.net/life-sciences/Electrochemical-Impedance-Spectroscopy-Applications.aspx [Accessed: 14 June 2021]

[7] M. Isaac. “Design, implementation, and test of a low-cost, low-power, electrical impedance spectroscopy system for biomedical applications”. UPC, Barcelona, 2021

[8] Anonymus. “What is a System on Chip (SoC)?”. [Online] Available: https://anysilicon.com/what-is-a-system-on-chip-soc/ [Accessed: 16 June 2021]

[9] L. Micheál “AD5940 Bio-Electric Shield User Guide”. [Online] Available: https://wiki.analog.com/resources/eval/user-guides/eval-ad5940/hardware/eval-ad5940bioz [Accessed: 16 June 2021

[10] I. Antoni “Bioimpedance Monitoring for physicians: an overview”. [Online] Available: https://www.researchgate.net/publication/253563215_Bioimpedance_Monitoring_for_physicians_an_overview [Accessed: 15 June 2021]

[11] B. Sanchez, A. L. P. Aroul, E. Bartolome, K. Soundarapandian and R. Bragos. Propagation of Measurement Errors Through Body Composition Equations for Body Impedance Analysis, IEEE Transactions on Instrumentation and Measurement, vol. 63,no. 6, pp. 1535-1544, June 2014.

Glossary

A list of all acronyms and the meaning they stand for.

- GATT: Generic Attribute Profile
- ATT: Attribute Protocol
- UUID: Universally Unique Identifier
- UI: User Interface
- ECM: Mean Square Error (in Spanish)
- EIS: Electronic Information Services
- BLA: Biological License Application
- SoC: System on Chip
- EDA: Electronic Design Automation
- ECG: Electrocardiogram