Design, implementation and test of a fast impedance spectroscopy measurement system for biomedical applications

A degree Master Thesis Submitted to the faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Marcel Palet Brandi

Directed by

Ramon Bragós Bardia

Index

Acknowledgment

First, I would like to sincerely thank the Dr. Ramon Bragós Bardia for all the help and attention given along this project. He has not just been an exceptional tutor during this work but an exceptional professor during my four years of degree and two of master. Without any need to warm him up (he will probably not know that), I decided to enroll the Master in Electronics thanks to him so I wouldn't be where I am if I did not meet him. My more sincerely and honest thanks Ramon.

Also, I would like to thank the Biomedical and Electronic Instrumentation group of the Electronic Engineering Department of the UPC to give me the facilities and the opportunity to do this work.

And at last, but not least, I would really like to thank myself, for working really hard during these long six years, for waking up early on Sundays to study, for not even thinking of giving up although sometimes it has not been easy, for this and other thousands of unnumerable things, thank you Marcel.

Abstract

This project offers a detailed explanation on the design, implementation, and measures to characterize the hardware and firmware programmed for the acquisition of the electrical impedance spectra. It includes from the signal generation, which is based on the use of a multifrequency signal, to the acquisition and signal processing. All have been implemented using the development board RedPitaya.

Moreover, all the process of signal generation, signal acquisition and processing so as the results calculation is done inside the commented RedPitaya board. The communication between the board and the computer is handled through SSH via ethernet port. Finally, this project also includes a brief script implemented in MatLab which objective is just to represent the acquired results from the board.

To conclude, characterization measures are made to verify the system specifications.

List of figures

List of tables

1. Introduction

1.1. Objectives of the project

The objective of the project is to design and implement the firmware and software of a multifrequency impedance spectrum analyzer based on the development board RedPitaya. The impedance spectrum must be obtained using one measurement at multiple frequencies at the same time which can be achieved with the use of a multisine signal. In addition, the system must be capable of measuring frequencies up to 1MHz with enough precision to detect impedance changes in the range of 1Ω on a full scale of 1 kΩ.

All the needed code for signal generation, acquisition and processing must be running inside the RedPitaya board thus, using the computer just for results representation as all the data received is already processed. With this approach, the system becomes closed and compact as no raw data is transferred between the board and the computer.

1.2. Context of the project

In order to measure the impedance spectrum of biological materials it is required a set of measurement instruments. There is the possibility of buying these instruments but, for experimental research it is more regular to design them for the own benefits of the projects, as it is the case of the Electronic and Biomedical Instrumentation laboratory of DEE.

Until now, some of the implemented designs are based on directly programming an FPGA, using development tools as *Cadence* or *Quartus*, as it is the case of [1] where a multifrequency impedance analyzer is designed based on the development board DE3, in charge of the FFT calculation and processing along with an external DDR2 memory which is in charge of the embedded system. The system implemented in [1] can work at high-speed spectra per second and encompasses a frequency range from some mHz to 10MHz. Although it is a high-efficient system, it is very specific, expensive and not so compact solution, therefore it appeared the idea of studying the possibility to implement a complete multifrequency spectrum analyzer system inside a small development board, the RedPitaya.

An initial system using the above commented development board was implemented in [2]. In that work, the board is only dedicated to signal generation and acquisition so high spectra per second rate is achieved (60 spectrums/second). After all the data acquisition process, the raw data obtained is transferred to a computer which is the one in charge of the processing and representation of the results, which means transferring 0.6 GB of data for an acquisition of 12 seconds through an ethernet link, this makes difficult the real-time processing and visualization of data. Yet it is a very efficient and compact solution, the step beyond is what has been implemented in the present work. Why not using the RedPitaya board not just as a signal generator/acquisition but also as the processing tool, thus, the data transferred between the board and the pc is considerably reduced so as the computer is just in charge of representing the processed results. This is what has been implemented in this project but first, let's introduce some important key concepts for understanding the impedance of biological elements.

1.3. The electrical impedance

When characterizing biological systems as they could be tissues, organisms or even cellular suspensions, the measurement of electric bioimpedance becomes very useful. The concept of bioimpedance refers to the passive electrical characteristics of biological tissue when an electric

current travels through it. Consequently, it can be expressed as the relation between the voltage and current that flow through a determined load. Speaking in terms of steady state, impedance can be expressed as a complex number as $Re^{j\alpha}$, where the module relates the voltage and current amplitudes while the phase relates the phase shift between them. Moreover, this relation is not constant along a certain frequency sweep, thus the impedance will be expressed in terms of ω as:

$$Z(\omega) = \frac{V(\omega)}{I(\omega)} \tag{1}$$

1.3.1. Impedance of biological materials

For this work, the main interest is to study the impedance that biological materials present so as the physical phenomena to which it is associated. To further understand the behavior of impedance along biological tissues we can understand these tissues as an ordered dispersion of cells all suspended inside an ionic medium. The idea is that, at low frequencies, signals flow over the extracellular space, as the cell membrane acts as an isolator (see Figure 1). On the other hand, at high frequencies, the isolation feature of the cell membrane is lost so signals can flow all over the space (see Figure 1). Thus, impedance at low frequencies will be bigger (signals will have to travel bigger distances) than at high frequencies, where it will be lower.

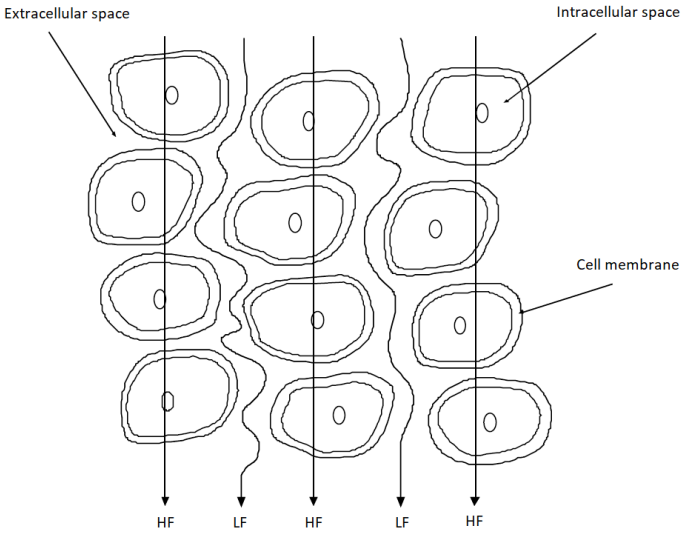


Figure 1. Simplified representation of current distribution at High and Low frequencies

The above commented effect is what is known as β dispersion which is allocated between some tens of kHz to tens of MHz. Furthermore, Schwan [3] described that there are three different frequency ranges in which the impedance of a material produces a relaxation, these where called the bio-impedance dispersions:

- **α:** The α dispersion goes from mHz to tens of kHz. This relaxation is associated to the dielectric losses of the tissue, its intracellular structure, and the ionic diffusion. This region is not of much interest regarding tissue recognition, moreover, the contact of electrodes itself introduces too much noise to the measurements making the analysis inside this margin non profitable.

- ***β*:** This is the dispersion recently commented in Figure 1. It encompasses the range between tens of kHz to tens of MHz and is directly related with the capacity of the cell membrane, the conductivity of the protein molecules that that penetrates this membrane and the intra and extra cellular ionic liquids. Thus, this band is the one of interest for extracting information about the cellular composition of the tissue.

- ***γ*:** Finally, γ represents the rest of the frequency band going to hundreds of GHz and is related with dipolar relaxation phenomena of water molecules and salts present in tissues. This frequency band is not relevant for tissue recognition, but its measurement is interesting and is done with microwave techniques and instrumentation.

In Figure 2 it can be seen the effect of the above-described dispersions on the permittivity and conductivity:

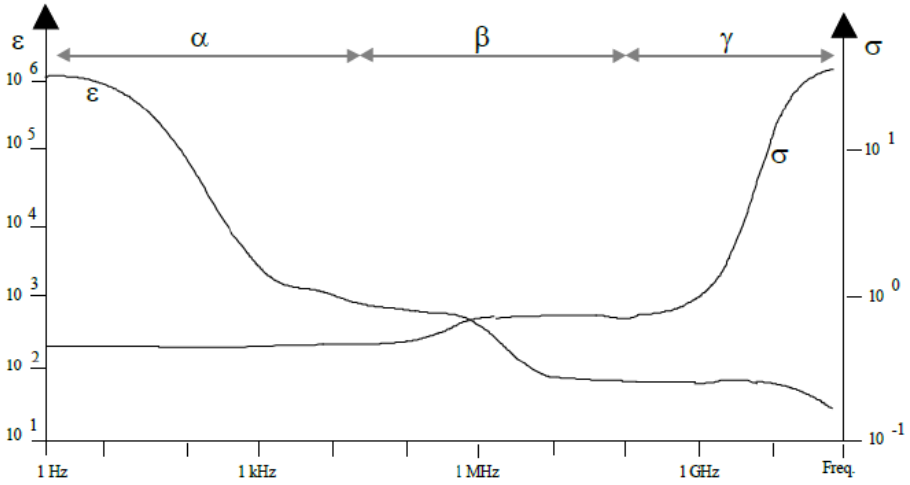


Figure 2. Dispersion of the module of σ and ε in biological materials *[4]*

1.3.2. Circuital models

The main cell component, in the case of this work, can be considered as the cell membrane the structure of which can be understood as a two-layer lipidic medium that acts as a dielectric interface which can be approximated to the one found between the layers of a capacitor. With the objective of analyzing individual cells, it is assumed that the cell is suspended in an environment that allows an ionic conductivity and that the intracellular liquid called cytoplasm also allows ionic conductivity, we can model a unique cell as a set of resistors and capacitors:

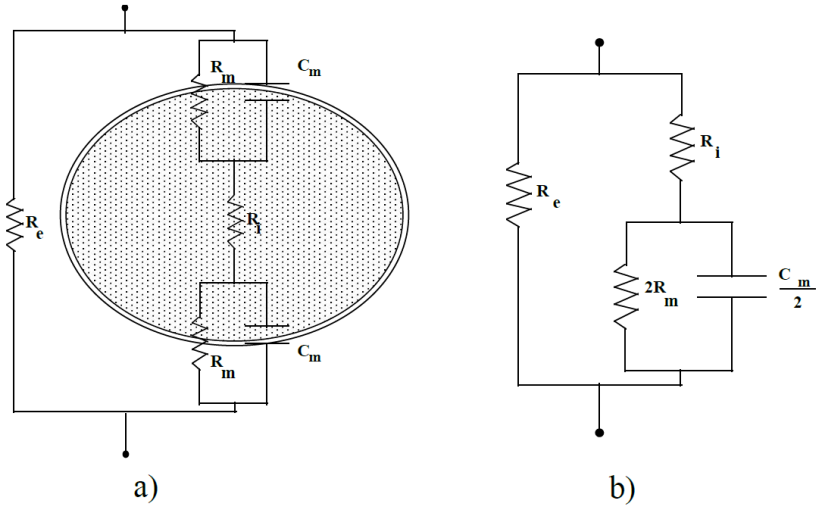


Figure 3. Model representing the main mechanisms that determine the impedance of a cell suspension (a) and its simplified model *[5]*

The model described in [5] is very interesting to finally understand how individual cells can be modelled. In Figure 3.a, they are represented the extracellular and intracellular mediums with the resistors R_e and R_I respectively and the capacity of the cell membrane with C_m along with its resistance R_m. Figure 3.b is the direct electrical simplification of the just commented model.

Furthermore, the behavior of the cell membrane is usually considered as fully capacitive, thus R_m can be neglected to ease a mathematical approach to the model. The resulting circuit can be represented as:

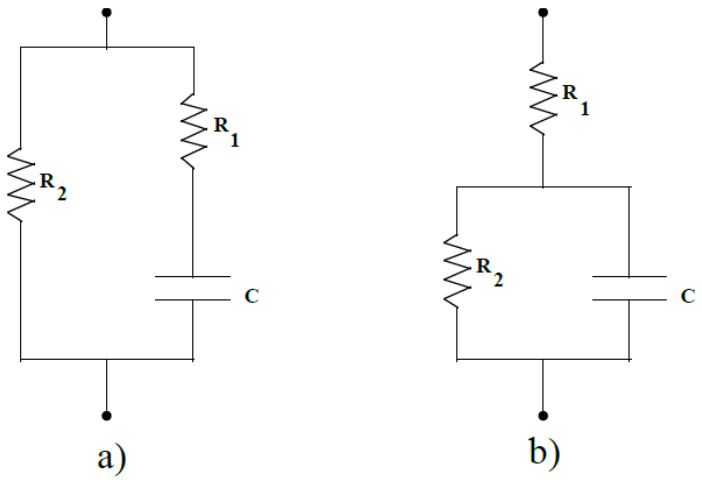


Figure 4. Circuital model of a cell neglecting the resistive component of the cell membrane R_m

By simple circuital analysis from Figure 4.b, impedance can be directly expressed as:

$$Z(\omega) = R_1 + \frac{R_2}{j\omega R_2 C + 1} \tag{2}$$

Using the following relations:

$$R_0 = R_1 + R_2; R_\infty = R_1; \omega_C = \frac{1}{R_2 C} \tag{3}$$

The impedance of the RC model from Figure 4 now can be expressed as what is known as Debye complex relaxation model:

$$Z(\omega) = R_\infty + \frac{R_0 - R_\infty}{j\frac{\omega}{\omega_c} + 1} \tag{4}$$

Concluding, the impedance of a cell in an ionic suspension is fully characterized with the three parameters R_0, R_∞ and ω_c of the Debye relaxation.

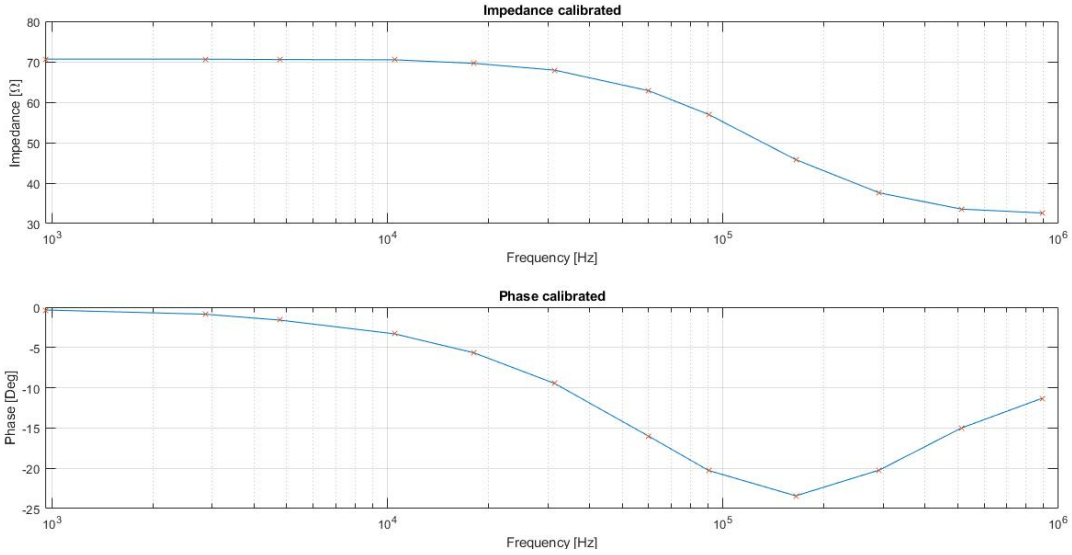


Figure 5. Module and phase representation of Debye relaxation with $R_0 = 70\ \Omega$, $R_\infty = 30\ \Omega$ and $\omega_c = 110\ Hz$

There are mainly two different approaches to obtain impedance spectrums depending on the requirements of the system used (velocity, accuracy…), the frequency sweep, which consists in obtaining the spectrum by using signals with one frequency component, thus, the final spectrum is the iteration over the desired frequency range with one iteration per frequency sample. On the other hand, there is the possibility to use what is known as multi-sine or multi-frequency signals which, as its name indicates, are signals with more than one frequency component. This second approach is what has been implemented on this work so, on the next chapter, it will be further discussed the advantages of using multi-frequency signals in this project.

1.4. Limitations of frequency sweep methodology

The main drawback of using the frequency sweep methodology is clearly the time needed to make one impedance spectrum measurement, as it depends on the number of frequential steps where the spectrum wants to be represented and the cycles per frequency needed. This implies a major excitation time for lower frequencies than for higher ones not to say the huge amount of time needed for processing each iteration.

As it has been explained in 1.3.1, the frequency band of interest for characterizing biological tissues is the one defined by the β dispersion. A range from 1kHz to 1MHz provides enough information of the biological material in the case of animal cells and tissues. To have a global description of the material approximately between 15 and 30 logarithmically spaced frequential point must be considered. With the frequency sweep methodology around 10 to 20 seconds are needed to obtain the full spectrum. Depending on the application needed this is a good solution

but, for the current work, where dynamic biological systems as the heart or the lungs are involved, it is not acceptable as the time modulation of the impedance changes produced by them would be lost.

The most common technique to solve this problem is to use what is called multi-sine signals. The main advantage is that this type of signals gives information in all the desired frequency band. For instance, with just one period of this multi-sine signal a full impedance spectrum at the frequencies conforming the signals can be obtained.

1.4.1. Multi-sine signal

Multi-sine signals consist of the summation of different sines or cosines at different frequencies thus resulting in a signal with multiple frequency components [6]:

$$x(t) = \frac{1}{\sqrt{N}} \sum_{n=1}^{N} a_n \cdot \cos\left(\omega_n t + \varphi_n\right) \tag{5}$$

In equation (5), N represents the number of frequencies of the signal, a_n is the amplitude at each of these frequencies, ω_n stands for the angular frequency and φ_n is the phase. Understanding this, the spectrum of $x(t)$ can be obtained by applying the Fourier transformation to equation (5) resulting in:

$$X(f) = \frac{a_n}{2\sqrt{N}} \sum \delta(\omega - \omega_n) e^{-j\varphi_n} + \delta(\omega + \omega_n) e^{j\varphi_n} \tag{6}$$

An example of how the spectra of the multi-sine described in equation (6) looks like, is the one used along this work which has twelve frequency components:

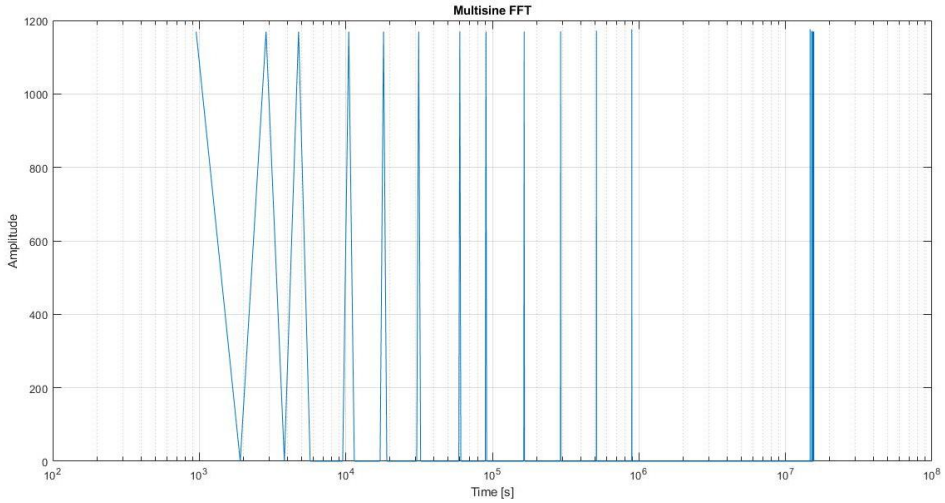


Figure 6. Spectrum of multi-sine signal

2. Architecture and design of the system

As it has been explained in previous sections, the idea of this work is to go a step beyond the work implemented in [2]. There, due to time development limitations and processing issues regarding the triggering of signals received, the RedPitaya development board was fully dedicated to signal generation and acquisition, thus, achieving very high spectrums per second rates. Although this approach is perfectly feasible, the present project tries to upgrade it by implementing all the signal processing inside the development tool, so no external hardware is needed for that, and the computing power required for transmitting all the data is considerably reduced.

In this chapter, a new architecture of the hole system will be explained, from the FPGA hardware modules that I have taken advantage of, to the software implemented to handle the generation and acquisition of data so as its processing. Finally, a discussion on the results will be done in order that validate the complete system.

2.1. Development tools

The complete overview system overview can be seen in Figure 7:

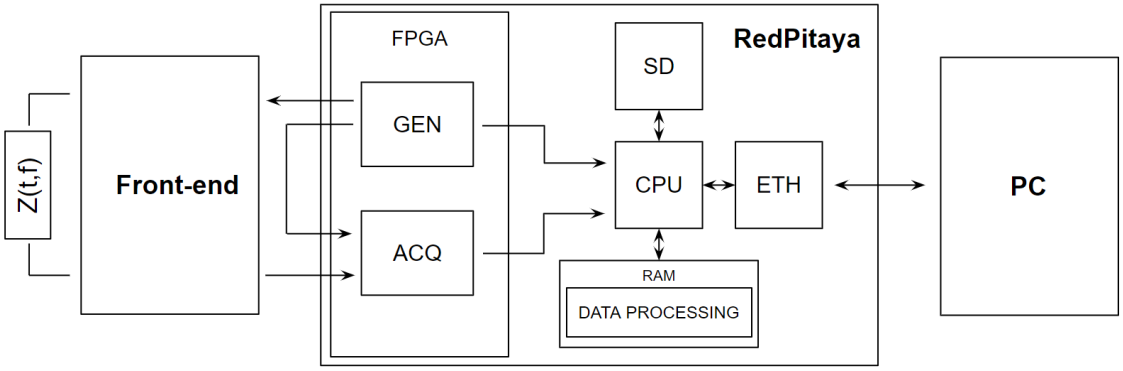


Figure 7. Bloc diagram of the implemented system

From Figure 7, it can be seen how the system is basically divided into three different blocs: Front-end, which is the hardware in charge of providing the system a tension and current measurements from a load so as that making a final impedance measurement can be done. The RedPitaya, the core of the system in charge of signal generation, signal acquisition, data processing and data transmission. At last, but not least, the PC or laptop which just works as a data representation tool in this work.

2.1.1. Front-end

The front end used in this work was developed in previous projects by the electronics instrumentation and biomedical group inside the Electronic Engineering department of the UPC. It is based on the use of a unipolar current source to transform the voltage signal coming from the FPGA of our device to current so as to perform the impedance calculation. Moreover, this Front-end also implements additional circuits to measure the differential voltage and current on the load, using differential amplifiers, buffers and a transimpedance amplifier.

At a structural level, the front-end has an input port where the reference signal of about 2Vpp is introduced, an output measuring the tension drop on the load and a second output proportional to the current injected on the load. Everything must be supplied at a constant 5Vdc.

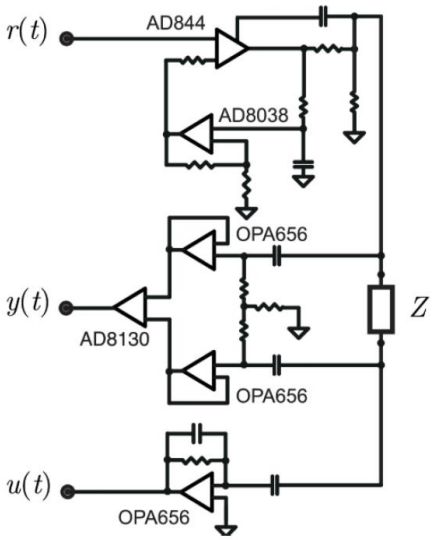


Figure 8. Circuit implementing the Front-end *[7]*

2.1.2. RedPitaya development board

RedPitaya is an open-source development board that incorporates two DAC and two ADC both of 14 bits able to convert and acquire at 125Ms/s and four more low speed DAC/ADC of 12 bits each. The board also has an FPGA and an ARM Cortex A9 microprocessor with two cores. It also incorporates different peripherals for digital communication as they are Ethernet, USB, I2C, SPI, UART and some general input-output GPIO pins. All the platform works with the Linux operating system.

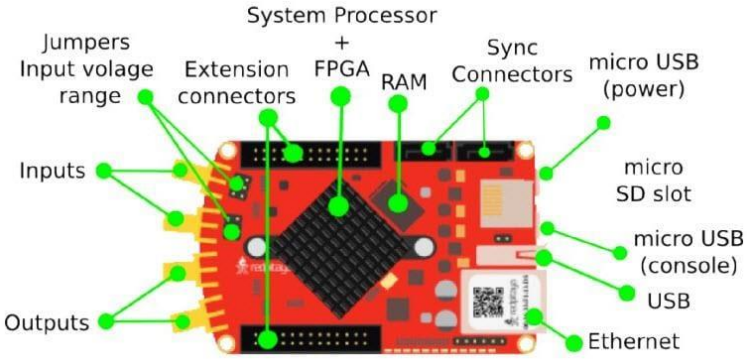


Figure 9. RedPitaya board characteristics

The device is based on a SoC that already integrates the ARM cortex along with the FPGA from the family of Zynq 7000 of Xilinx. This has been in fact one of the main attractions of using this board instead of using a Xilinx board based on the Zynq 7000, the RedPitaya already comes with all the hardware needed for signal generation and acquisition, so no need of external hardware is needed. At the same time this could be considered a drawback because, even though all hardware modules implemented on the FPGA are said to be open source there is a very limited documentation on how to access or reprogram them. Nevertheless, it is a very good

starting point just assuming all FPGA modules as they come just focusing on develop the software in charge of handling an efficient way to collect and process the data.

2.2. Hardware architecture for signal generation and acquisition

As all the hardware modules for signal handling are already programmed inside the FPGA of the board, no FPGA programming has been implemented. Even though, some considerations and calibrations has been done to adjust the system to the requirements. A basic model of how the generation block works can be seen in Figure 10:

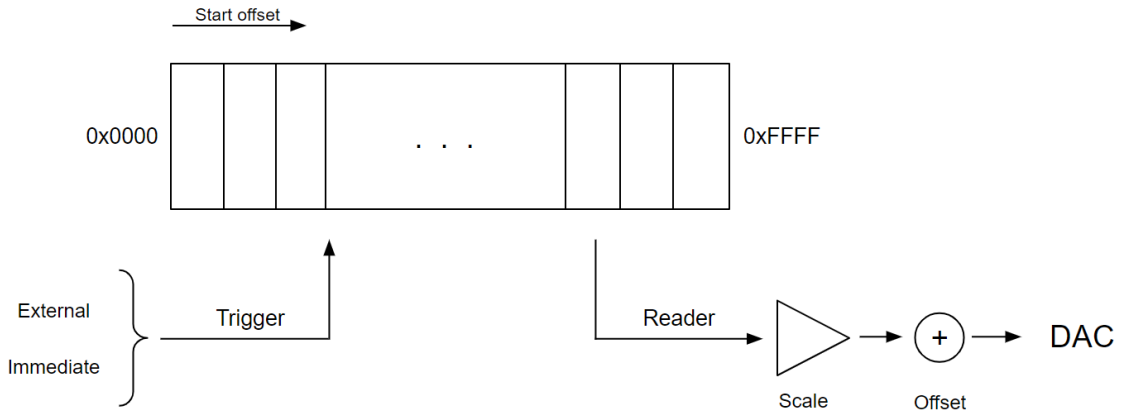


Figure 10. Generation module of the RedPitaya

The already implemented bloc in the FPGA allows to load arbitrary waveforms to the 16384 samples buffer, this is a really great feature for this project were a multi-sine signal, that practically seems white noise, must be generated. Moreover, the system also allows to set an initial offset to start filling the buffer, a scale factor and an offset before the sample is sent to the DAC.

One of the greatest features is that it also implements a digital counter, handled by the application with a pointer, which keeps track on the position of the last sample sent. By software, the increment of this counter can be adjusted relatively to the clock frequency of the DAC, this way the waveform stored inside the buffer can be modulated to the desired frequency. The multi-sine signal used in this project is 16384 samples length with one period of the fundamental frequency, which is 1kHz, thus the frequency shall be adjusted to 1kHz.

On the other hand, the acquisition block is very similar to the generation:

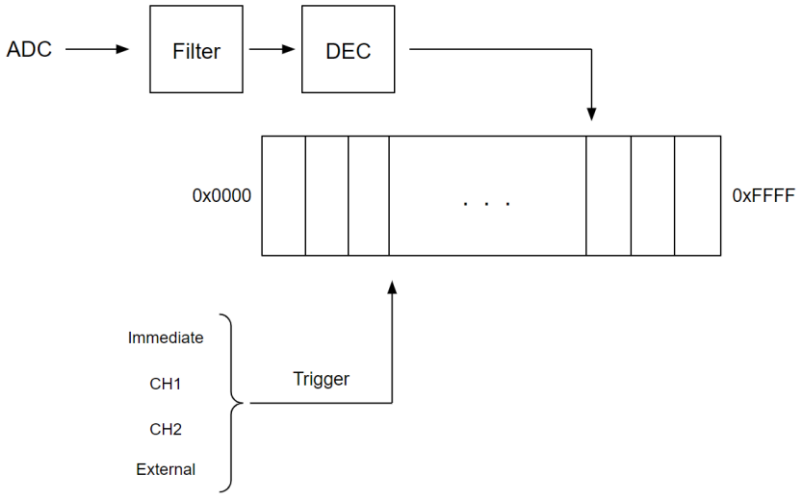


Figure 11. RedPitaya acquisition module

As it can be seen in Figure 11, the acquisition block also implements a 16384-length buffer with a pointer indicating the position being read at every iteration. It is of high importance to notice the stage previous to writing the buffer, the decimation. Differently to the generation part, the counter cannot be configured with the same freedom as before as it directly depends on that decimation stage. There are a set of values, predefined by the RedPitaya platform, so the correct one must be chosen:

Table 1. Decimation factor relation with sampling frequency and temporal buffer capacity

Decimation	Sampling frequency	Temporal buffer capacity
1	125 Ms/s	131 µs
8	15.6 Ms/s	1.048 ms
64	1.953 Ms/s	8.388 ms

For ease of the reader, in Table 1 a set of the three first decimation factors have been included. For this work, the decimation factor that better fits our needs is by 8 as, considering that f_{min} will be determined by the sufficient time to capture one period of the fundamental signal. On the other hand, being f_{max} determined by the Nyquist condition, the range of frequencies that our acquisition block would handle is determined by:

$$f_{min} = \frac{1}{1.048\ ms} \tag{7}$$

$$f_{max} \leq \frac{f_s}{2} = 1.812\ MHz \tag{8}$$

Within the acquisition stage there is one last aspect which results of key importance, the trigger. As it can be seen in Figure 11, the acquisition bloc also offers some different possibilities to handle this. Nevertheless, none of these options has been used and a new triggering protocol has been implemented. This has been so because for measuring the impedance, the generation and acquisition of signals must be synchronous, if this is not accomplished then it is impossible to have a repeatable phase measurement as all the readings will be taken at different times. One feasible option to make signals synchronous is to use on of the slow input-outputs of the board to generate a pulse train which can be seen as a “clock”. It has been discarded as it implies the use of external hardware, concretely a level-shifter, as the slow outputs of the board

generate signals of 3.3V amplitude and the fast inputs are limited to 1V. As one of the objectives was to implement a fully software-controlled system and the use of level shifters was used in previous works [2], this approach has been discarded.

2.3. Software architecture

Before the final solution, many approaches have been implemented with modifications regarding the signal handling, triggering method… so an initial overview of them will be discussed to better understand how it has been arrived to the final one.

2.3.1. Different software solutions

As commented in the Hardware architecture for signal generation and acquisition section, the generation module already implements a 16384-samples buffer for arbitrary signal generations, as it is the case of this project. Concretely, the signal used along this work is a multifrequency sinusoid with twelve frequency components ranging from 1 kHz to 939 kHz, its shape is the following:

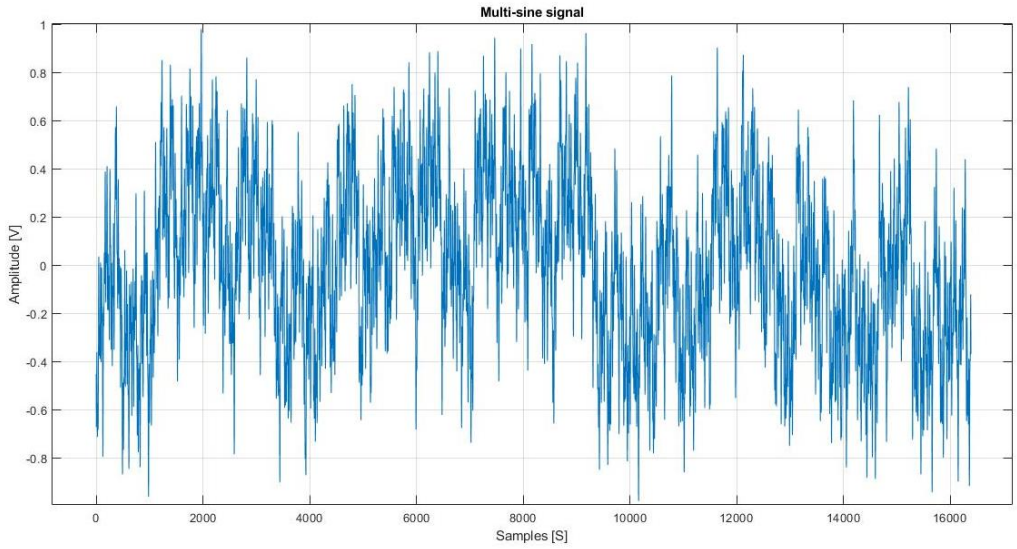


Figure 12. Multi-sine signal used for impedance measurement

As the reader may notice, setting a trigger for a continuous generation of the signal in Figure 12 is quite complex as initially it may seem white noise. This is the main reason why the first and more direct solution is to generate pulses of this multi-sine signal, this way the acquisition process is directly eased as, when the acquisition thread receives something different than zero then it starts saving data to the buffer. Although this solution may work for a first prototype it has some important drawbacks that make it a no feasible approach:

- Regarding the FPGA block in charge of generating signals, it must be configured each time an arbitrary signal is sent. This recurrent configuration considerably decreases the spectrums per second rate which is one of the key points for achieving a live recording system.

- The second and most important drawback is related to medical care. One of the measurements that the system must handle is arm-to-arm impedance measuring and even direct measurements in the heart using a catheter. A minimum rate of spectrums per second is required in order to obtain the lungs behavior along with the heart rate,

concretely five spectrums per second. In this work, it means that the multi-sine signal of 1.047 ms must be generated every 200 ms thus obtaining the 5 spectrums per second rate desired. From the software point of view, this is completely feasible to program but due to medical care it is not possible for a final version as, the body of the subject will receive the envelope of the pulsed arbitrary signal generated additionally to the signal itself, thus, deriving in a low frequency signal very dangerous for the subject heart.

To conclude, a simple flowchart of the spectroscopy analyzer using this approach would be the following:

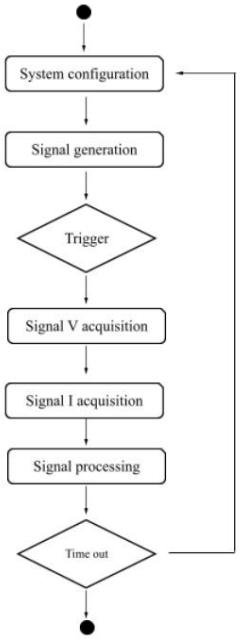


Figure 13. Flowchart of spectroscopy analyzer using pulsed arbitrary signal generation

Figure 13 is very illustrative as the before commented drawback can be clearly seen, at each iteration the generation bloc must be reconfigured to create the needed arbitrary signal, thus, deriving in a non-acceptable amount of time.

Owing to all the above considerations, the signal generation must then be continuous. Moreover, by generating a continuous arbitrary signal, the two commented drawbacks are solved as the generation bloc must be configured once, so no time is lost configuring it in each iteration. On the other hand, here is where one of the major problems of this system appear, the implementation of a trigger by software becomes very difficult as there is no way to stablish a signal level to be triggered in a continuous multi-sine wave. Therefore, considering that the system is ruled by the same clock and that the processes have fixed timings, it is tried to eliminate the triggering. With this, there is no control on the part of the signal that is being captured but all the captures shall be the same. To sum up, the updates flowchart is the following:

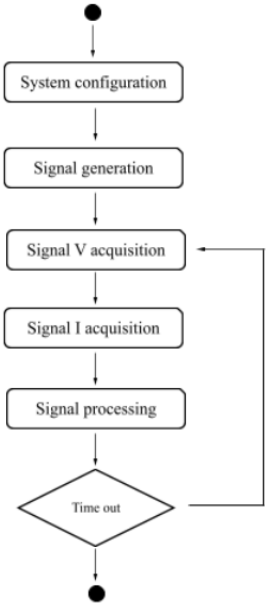


Figure 14. Flowchart of spectroscopy analyzer using a continuous signal generation without trigger

From Figure 14 it can be seen how the configuration process is done before entering the loop, this way there is no loss of time reconfiguring the generation bloc. In addition, as there is no triggering process the overall system timing is decreased.

Although this solution is valid for magnitude recording, it has huge affectations on the phase. Let's take the impedance phase spectrum of a 1 kΩ resistance over 100 iterations to illustrate the phase issue:

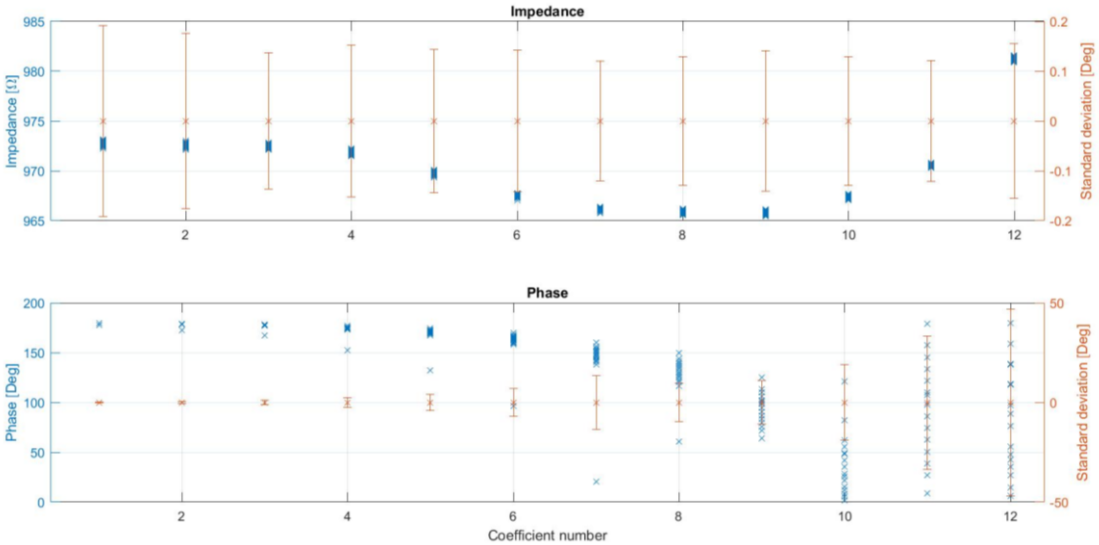


Figure 15. Phase coefficient dispersion over 100 iterations

It can be seen how, even though the system has a common clock and fixed timings, there is an exponential increase on the precision of the impedance phase coefficients as frequency increases. This is a consequence of the small error present in the acquisition process of the system, as Figure 13 and Figure 14 illustrate how acquisition of tension and current signals is made sequentially accessing to the their respective buffers, this clearly affects the phase shift between the acquired signals as the current will never be taken at the same time as voltage thus,

adding a small random error on the samples acquired. At lower frequencies this error does not represent a huge affection on the variation of the samples but at higher ones it is not acceptable. Let's analyze then how an error of a few samples delay would affect the two extreme frequencies, 954 Hz and 939 kHz.

Just one period of the lowest frequency is present in the multi-sine signal, it would correspond to the fundamental frequency. Moreover, as it has been explained in the Hardware architecture for signal generation and acquisition section, the sampling frequency of the acquisition bloc is decimated to 15.6 Ms/s so the total amount of samples per period of the fundamental frequency component are:

$$\frac{Samples}{period} = \frac{1}{954\ Hz} \cdot 15{,}6\frac{Ms}{s} \cong 16352\ samples \tag{9}$$

While the number of samples per frequency at the top frequency component of 939 kHz are:

$$\frac{Samples}{period} = \frac{1}{939\ kHz} \cdot 15{,}6\frac{Ms}{s} \cong 16\ samples \tag{10}$$

Comparing equations (9) and (10) it can be seen that, at low frequencies, practically all buffer samples are dedicated to representing the signal while, as frequency increases, less samples per period are used. Thus, deviations of small samples at each iteration of the acquisition algorithm are insignificant for the lower frequencies but can represent errors of 180º in phase at the higher ones.

2.3.2. Final software implementation

Until now, some of the most important programmed solutions have been overviewed so as the principal problems that have been appearing along the project. The last step is then to discuss the final solution implemented, how all the technical considerations have been solved and how each principal bloc of the tool work. The final flowchart of the system can be seen in Figure 16.

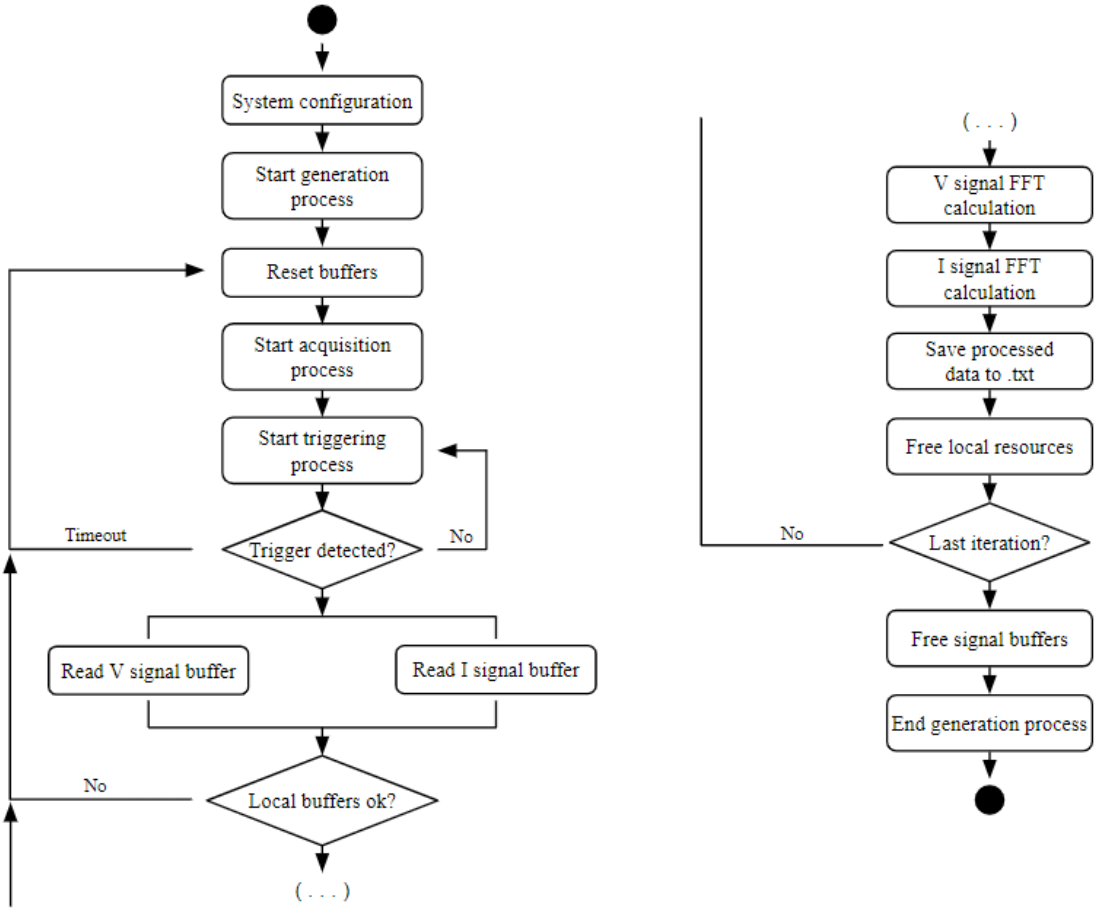


Figure 16. Final flowchart of the implemented system

Making a quick look at Figure 16, the trigger stage is again present, and the acquisition process has been divided into two different threads, thus, solving the before commented synchronization problems.

Initially, a *"System configuration"* is done by means of initializing all the required variables and loading the pre-defined multi-sine signal, which is directly load to a local buffer from an external file loaded to the RedPitaya. In addition, the generation and acquisition modules of the FPGA must be configured, that means choosing the channels that are going to be used for both generation and acquisition. In this work, Channel 1 of the generation block is used, Channel 1 and Channel 2 of acquisition are used for tension and current measurement respectively. In the case of the generation, it is needed to load the arbitrary signal (which is now stored in a local buffer) to the generator buffer of the FPGA and set the desired generation frequency. Regarding the acquisition bloc, the decimation value chosen is set (please go to Hardware architecture for signal generation and acquisition).

Now that the two blocks are configured the main loop begins. A *"Reset buffers"* initial check is done, saving the required memory space for each of the needed local buffers. The implemented system uses eight local buffers of 16384-samples length for calculating the impedance, these are:

- Signal buffers: four buffers for signal storing, two per channel. One is used for directly storing the received signal and the other, called *"worker buffer"*, is a temporal buffer reserved for calculations and processing.

- FFT buffers: again, four buffers are used to store the calculated FFT coefficients, two per channel. Two buffers have been used for each channel so as to store the module and phase coefficients of the complete signal separately, otherwise the length of the required buffers would have been too long and not so easy to work with.

At this point, the acquisition process is started, and the designed triggering methodology begins. As it has been explained in the previous section, the trigger method is not trivial at the time of precise recording the phase of the multi-sine, the generated and both voltage and current signals must be as synchronized as possible to reduce the phase error. Thus, the first improvement has been a very simple but consistent idea which consists of basically downscaling the generated arbitrary signal, after this any of the samples confirming the signal reaches the 1V of maximum amplitude. Once this downscaling is done, four samples fixed at 1V of amplitude are added to the beginning of the arbitrary signal. Thanks to this idea, the initial trigger based on amplitude can be implemented as there is a clear flag inserted in each period of the generated arbitrary signal. The new adapted signal has the following aspect:

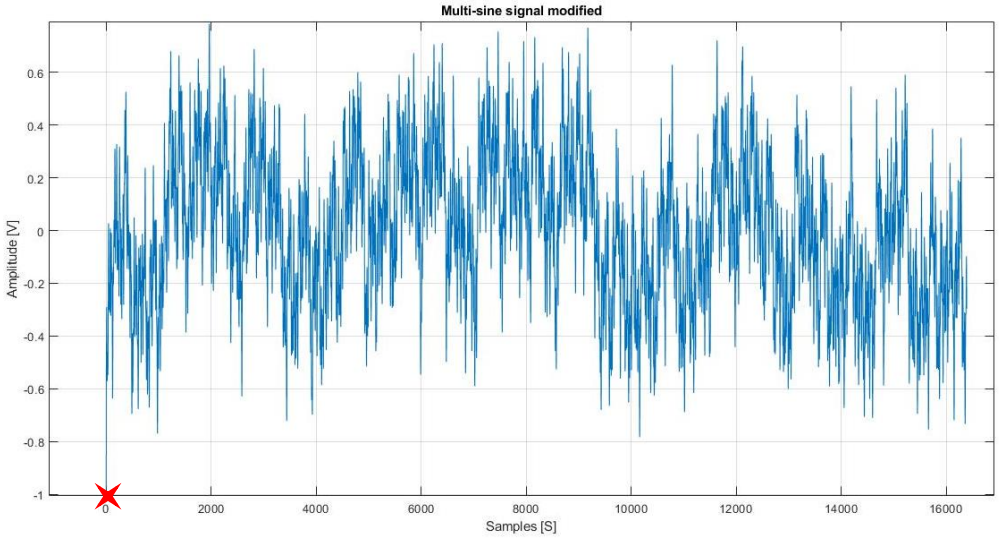


Figure 17. Modified multi-signal for handling trigger by software

The reader may thing that by altering the multi-sine, the impedance measurement can be affected, but please consider that the initial four samples of a 16384-sample length signal just represent a 0.02% of the whole signal, so comparing pros and cons it is perfectly assumable. Apart from this, looking at Figure 17, a second improvement has been done consisting of the parallel acquisition (instead of sequentially) of both signals, thus reducing a lot the acquisition time and considerably increasing the accuracy of the overall system. To implement this part some of the internal controllers of the RedPitaya have been modified so as to allow this fast acquisition. Finally, to check that the new triggering methodology works, three different captures of the tension acquired signal, in three different iterations (1, 20 and 50), to validate that effectively they are the same:

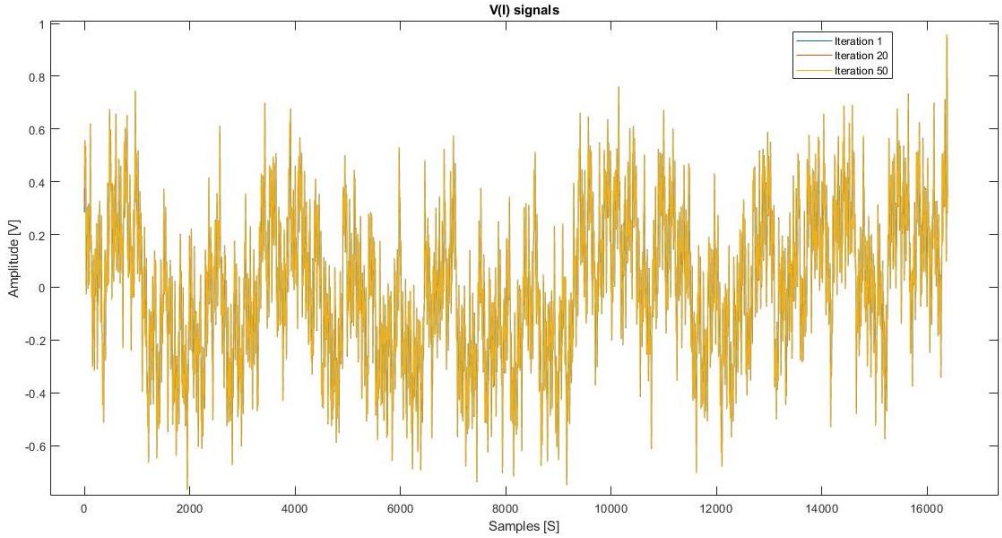


Figure 18. Acquired multi-sine current component at iterations 1, 20 and 50<

In Figure 18 it can be clearly seen how the three acquired signals are completely superposed; it is always captured the same fragment of the multi-sine. To conclude, let's look again at the phase spectrum of a 1 kΩ resistor over 100 iterations of the algorithm:

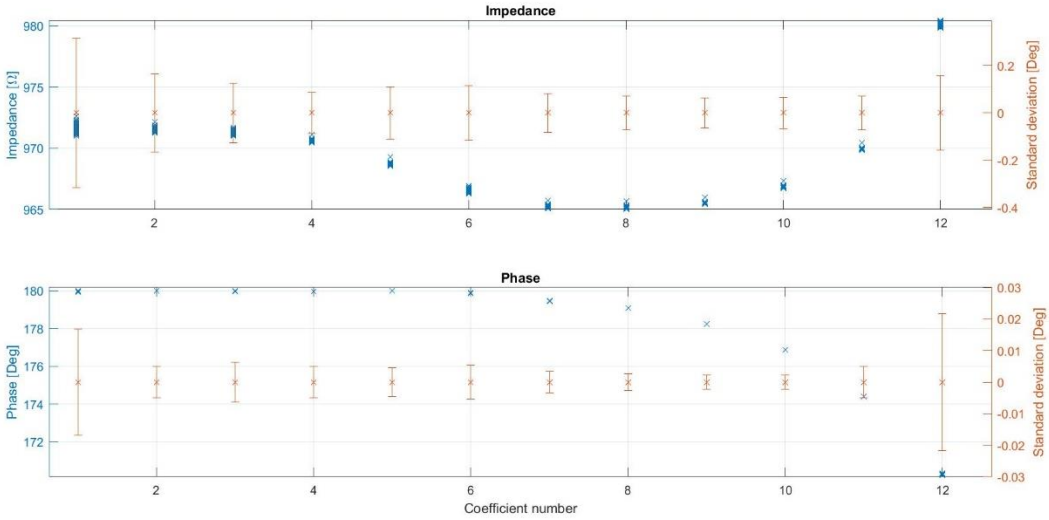


Figure 19. Phase coefficients dispersion over 100 iterations

Again, it is finally demonstrated how the precision of the phase coefficients has considerably improved compared with the one in Figure 15. In the case of Figure 19, taking a typical frequency for measurements as it would be 50 kHz and considering a confidence level of 99%, the precision can be easily calculated as $\pm\, 2.6\sigma$. In the case under analysis, regarding the module we obtain a standard deviation of $0.096\ \Omega$ which corresponds to a precision of $0.251\ \Omega$. Regarding the phase, considering that the deviation is $0.0034\ Deg$ at this frequency, the system is even more accurate with a precision of $0.008\ Deg$.

Now that both tension and current signals have been stored, it is checked that effectively the local buffer are correctly saved, and it proceeds to the FFT calculation for extracting the needed coefficients. The extraction of the voltage and current coefficients is done sequentially by taking the twelve samples of interest (corresponding to the twelve frequency components), inside each of the four FFT buffers. This can be directly done as these samples corresponds to the FFT maximum positions, which are fixed and known. For the FFT calculation the library

"kiss_fft" has been used, it is an already optimized library that calculates the Fourier transform of a set of input samples and can be configured to return a struct consisting of the module and phase of the calculated transformation. Finally, the achieved spectra per second rate with the implemented algorithm is 30 spectra/second, which is more than enough considering that the needed rate for real time acquisition is 5 spectra/second.

Once the coefficients are extracted, they are saved to two different .txt files (one for voltage and one for current) with the following comma separated structure:

```
-247.729608,-411.107753,1
-203.293029,-314.067764,3
-228.408947,272.823325,5
293.624626,-156.411617,11
-289.789095,-134.251334,19
-267.600232,136.887073,33
-277.194958,-36.905275,63
142.913743,-225.798201,95
-103.012047,230.876631,173
-173.165986,165.415317,307
204.719473,-89.286578,539
40.274821,181.558247,939
```

Figure 20. Example of one set of spectrum coefficients captured data

Table 2. Data structure of the .txt files

Module	Phase	Frequency component
%lf	%lf	%d

It has been decided to transfer separately the tension and current coefficients, with module and phase distinction, to the computer to allow the user further processing with the received coefficients.

Now that the voltage and current coefficients of this iteration have been saved, all the local buffer memory spaces are released, and a *"last iteration"* check is done, if so, all the FPGA modules are released and the algorithm ends.

3. Validation with experimental results

The last part of the work is to validate the implemented system with a set of experimental results demonstrating that all the initial requirements have been accomplished. Nevertheless, before the experiments an initial calibration must be done.

3.1. Calibration process

The calibration methodology followed has been the one implemented in [8] which starts from the assumption of a physical model of 4 ports of the measuring system:

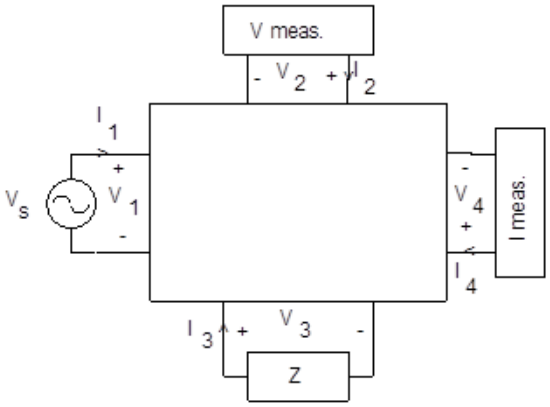


Figure 21. Physical model of an impedance measuring system

Assuming all the current applied by the generator is measured ($i_1 \approx i_4$) and that the tension meter has high impedance ($i_2 \approx 0$), the following equations are obtained:

$$\begin{pmatrix} v_1 \\ \vdots \\ v_4 \end{pmatrix} = \begin{pmatrix} Z_{11} + Z_{14} & Z_{13} \\ \vdots & \vdots \\ Z_{41} + Z_{44} & Z_{43} \end{pmatrix} \cdot \begin{pmatrix} i_1 \\ i_2 \end{pmatrix} \tag{11}$$

Developing the equation (11), it can be expressed the measured impedance as a function of the real one as:

$$z = -\frac{v_3}{i_3}; z_m = \frac{v_2}{i_4} = \frac{v_2}{i_1} \rightarrow$$
$$z_m = \frac{z(Z_{21} + Z_{24}) + Z_{33}(Z_{21} + Z_{24}) - Z_{23}(Z_{31} + Z_{34})}{z + Z_{33}} \tag{12}$$

The Z_{ij} values are not known, but the equation (12) can be re-written simplifying in terms of a_i to finally isolate z as a function of z_m:

$$z = \frac{a_1 z + a_2}{z + a_3} \rightarrow z_m = \frac{a_2 - a_3 z_m}{z_m - a_1} \tag{13}$$

In order to fins the coefficients a_i present in equation (13), three different known impedances should be measured to solve the following system:

$$\begin{pmatrix} z_{c1} & 1 & -z_{mc1} \\ z_{c2} & 1 & -z_{mc2} \\ z_{c3} & 1 & -z_{mc3} \end{pmatrix} \cdot \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} z_{c1} z_{mc1} \\ z_{c2} z_{mc2} \\ z_{c3} z_{mc3} \end{pmatrix} \tag{14}$$

Finally, by isolating the a_i coefficients from (14), we would obtain the three final calibration coefficients:

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} z_{c1} & 1 & -z_{mc1} \\ z_{c2} & 1 & -z_{mc2} \\ z_{c3} & 1 & -z_{mc3} \end{pmatrix}^{-1} \cdot \begin{pmatrix} z_{c1}z_{mc1} \\ z_{c2}z_{mc2} \\ z_{c3}z_{mc3} \end{pmatrix} \tag{15}$$

Now that the system is calibrated, three different experiments are done to validate the system.

3.2. Linearity and dispersion

The first experiment tries to characterize the implemented multi-frequency system. For this purpose, a set of seven resistors have been measured in the frequency range of 952 Hz to 894 kHz. In particular, the resistances used are 10 Ω, 20 Ω, 51 Ω, 100 Ω, 200 Ω, 510 Ω and 1kΩ.

To visualize the linearity of the system, a linear regression has been applied to the set of seven measurements of each resistance over each particular frequency value, resulting in a total of twelve lines. Each of these lines is characterized by the parameters β_1 and β_0 as:

$$y = \beta_0 + \beta_1 x \tag{16}$$

In equation (16), y corresponds to the measured resistance value while x corresponds to the nominal resistance value. The β_0 parameter corresponds to the vertical intercept at $x = 0$ and the β_1 to the slope of the linear regression line. Concluding, the parameters of each line can be seen in Table 3:

Table 3. Linearity parameters for each frequency

	Parameter	
Frequency	$\boldsymbol{\beta_0}$	$\boldsymbol{\beta_1}$
952 Hz	2.749	1.006
2.856 kHz	2.748	1.006
4.760 kHz	2.729	1.006
10.47 kHz	2.62	1.006
18.09 kHz	2.575	1.006
31.42 kHz	2.428	1.006
59.98 kHz	2.274	1.005
90.45 kHz	2.158	1.005
164.72 kHz	2.261	1.005
282.31 kHz	1.938	1.005
513.21 kHz	1.799	1.005
894.07 kHz	1.856	1.005

From Table 3, it can be concluded that the results obtained with the implemented system are very linear in the working frequency range. Moreover, the slope of each of the lines remains practically constant. All the numerical results of this experiment are shown in Table 4. Also, it is very interesting to see the graphical linear regressions found in Figure 22 which finally confirms the near perfect linearity of the system.

Table 4. Measurement results of seven resistances

	Frequency (kHz)											
Resistance	**0.952**	**2.856**	**4.760**	**10.47**	**18.09**	**31.42**	**59.98**	**90.45**	**164.72**	**292.31**	**513.21**	**894.07**
10 Ω	12.80	12.79	12.80	12.51	12.25	11.78	11.18	10.79	10.32	10.15	9.71	9.65
25 Ω	25.20	25.23	25.21	25.19	25.24	25.24	25.28	25.26	26.26	25.22	25.20	25.14
51 Ω	50.78	50.8	50.77	50.76	50.79	50.78	50.81	50.83	50.82	50.85	50.85	50.87
100 Ω	103.89	103.88	103.89	103.75	103.80	103.72	103.67	103.56	103.56	103.162	103.80	103.18
200 Ω	200.09	200.09	200.03	200.01	200.02	199.96	199.94	199.93	199.90	199.96	199.00	200.08
510 Ω	508.35	508.29	508.25	508.25	508.24	508.17	508.17	508.17	508.15	508.14	508.07	507.88
1kΩ	997.38	997.42	997.41	997.44	997.58	997.62	997.77	997.78	997.83	997.68	997.62	997.29

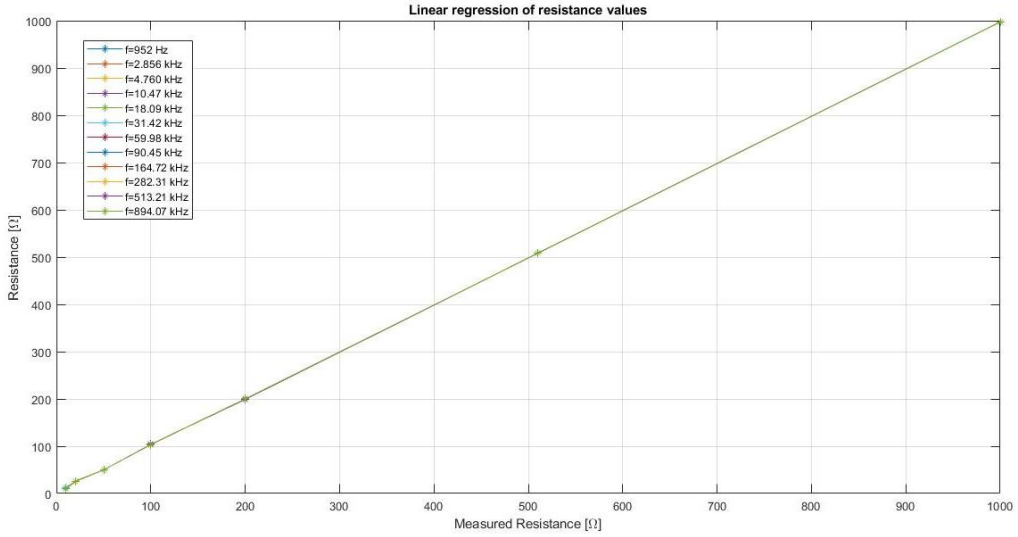


Figure 22. Linear regression for each frequency

3.3. Cell model

The second experiment is related to the model of the cell explained in the Introduction, which is shown in Figure 4.b. A set of two resistances with values 24 Ω and 47 Ω and a 33 nF capacitor has been used in order to characterize the frequency response of the cell model. Considering the circuit in Figure 4.b, it is expected to have the addition of R_1 and R_2 at low frequencies, as the capacitor C can be considered as an open circuit. On the other hand, at higher frequencies we expect to have just the R_1 component. Using the implemented system, the results obtained are the following:

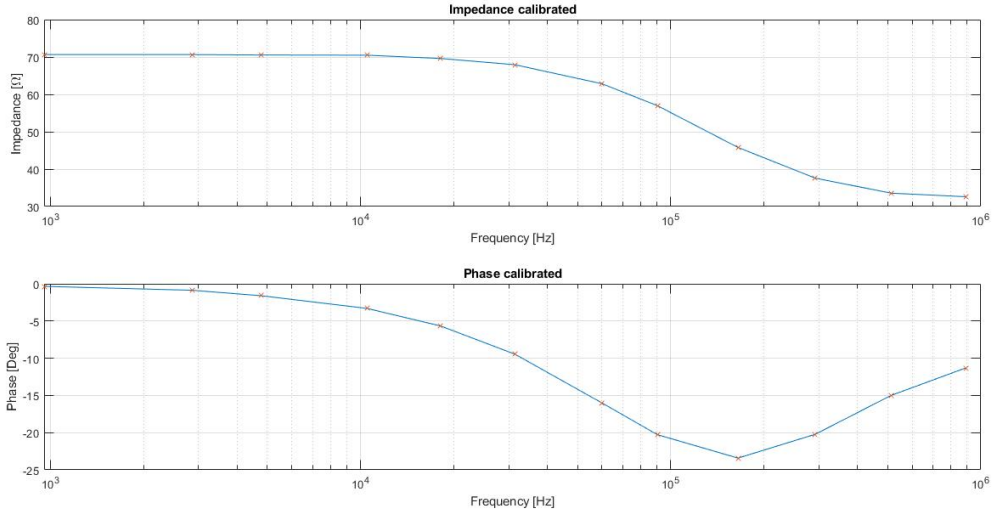


Figure 23. Spectrum of the simple model from Figure 4.b measured with the implemented system. $R_1 = 24\ \Omega$, $R_2 = 47\ \Omega$ and $C = 33\ nF$.

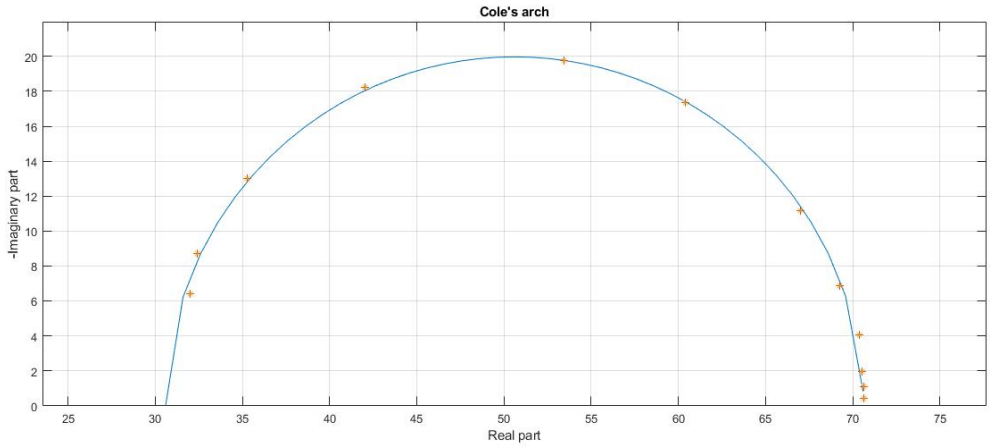


Figure 24. Cole-Cole arch of the simple model from Figure 4.b with estimated values $R_e = 70.61\ \Omega$, $R_\infty = 30.59$, $f_c = 111.18\ kHz$ and $\alpha = 0.00106$

Figure 23 demonstrates that the numbers are consistent with what would be expected theoretically from such configuration. Nevertheless, it is more interesting to do the inverse step, estimate the values of the components from the data. For this, it is very useful the Cole-Cole arch. In the representation from Figure 24, they are plotted the real and (negative) imaginary values of the impedance, for the twelve frequency values present in the multi-sine signal, forming an arch (increasing frequency from right to left). From this arch, a value of R_0 and R_∞ can be determined just by taking the two intercepts with the horizontal axis. Moreover, the value of R_0 will correspond to the right intercept while the R_∞ will correspond to the left intercept. At the maximum point of the arch, the value of the central frequency will be found. Finally, the α parameter corresponds to the angle with respect to the horizontal axis at which the center of the circle would be found. This is all described in equation (4) presented in **Circuital models** section.

In the case of this experiment, the obtained results are $R_e = 70.61\ \Omega$, $R_\infty = 30.59$, $f_c = 111.18\ kHz$ and $\alpha = 0.00106$. Considering the circuit used for the experiment, the value of R_e would correspond to the additive effect of R_1 and R_2 at low frequencies. The value of R_∞ corresponds to R_1 which, at higher frequencies, is the only one remaining. Concluding the

experimental results are well adjusted taking into account possible variations due to the components tolerances

3.4. Body measurements

3.4.1. Muscular tension

As a final experiment it has been decided to make body measurements to finally determine the accuracy and good functionality of the system.

The first experiment is focused on detecting changes on muscular tension. For this purpose, two pairs of electrodes have been putted in the forearm of the subject under test. The subject was asked to maintain the arm relaxed for 5 seconds, then apply tension for 5 more seconds and finally relax again. The results obtained were the following:

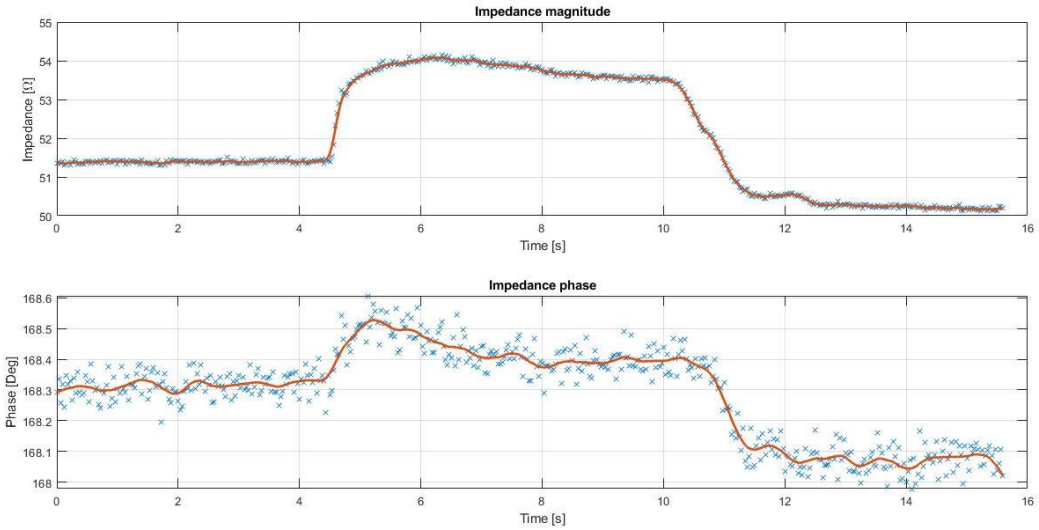


Figure 25. Magnitude and phase of arm contraction

Figure 25 shows the magnitude and phase of the arm contraction of the subject. It is also interesting to comment the small dispersion present in the measurements, as it can be seen the difference between the relaxed and contracted arm is just 3.5 Ω and the system is perfectly capable of detecting that change without need of post-processing. Moreover, in the phase graph the difference is even smaller, in the range of 0.1º-0.2º difference between relaxed and contracted, and it is also detected by the system. Finally, there are two interesting effects that should be commented. The first one is related to the contraction and relaxation moments, as it can be seen in Figure 25, the contraction is more abrupt than the relaxation, this is because the muscle fibers last less time in contracting (around 0.4 seconds) than in relaxing (around 1.6 seconds). The second effect is related to the different impedance levels at the two relaxed states. It can be noticed that the initial relaxed impedance value is higher than the final one. Although there is no concrete explanation to this effect, it could be due to an initial transitory stage after a muscle contraction.

3.4.2. Ventilation and heartbeat detection

Now that the muscle contraction has been validated. A second and more interesting experiment was done. It consisted in attaching two electrodes to one arm and two to the other

thus being able to detect the respiration of the subject. Furthermore, as the system is accurate enough, it should be possible to detect the heart rate from the obtained signal.

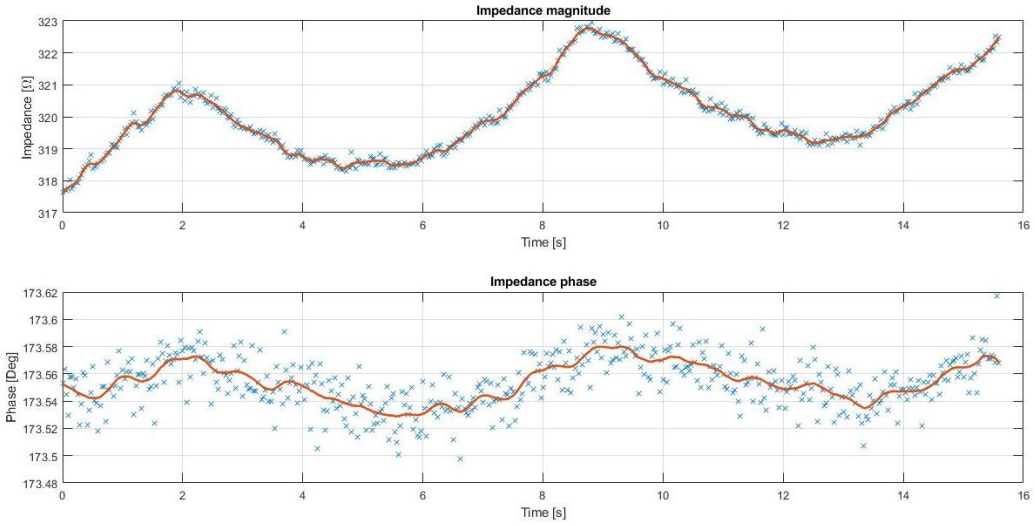


Figure 26. Magnitude and phase of ventilation signal

It can be perfectly seen how, during the 16 seconds of the experiment, the subject did six deep respirations which were perfectly recorded by the system. It must be said that, in the case of the respiration signal, the changes in phase are even more difficult to detect than in the case of muscular tension as they are in the order of 0.02º. Nevertheless, again the implemented impedance analyzer can record those changes.

As it has been mentioned, by processing the respiration signal we can obtain the heart one. Filtering the respiratory signal and subtracting it from the original one, it is possible to isolate just the heart rate signal which, in the case of this experiment is the following:

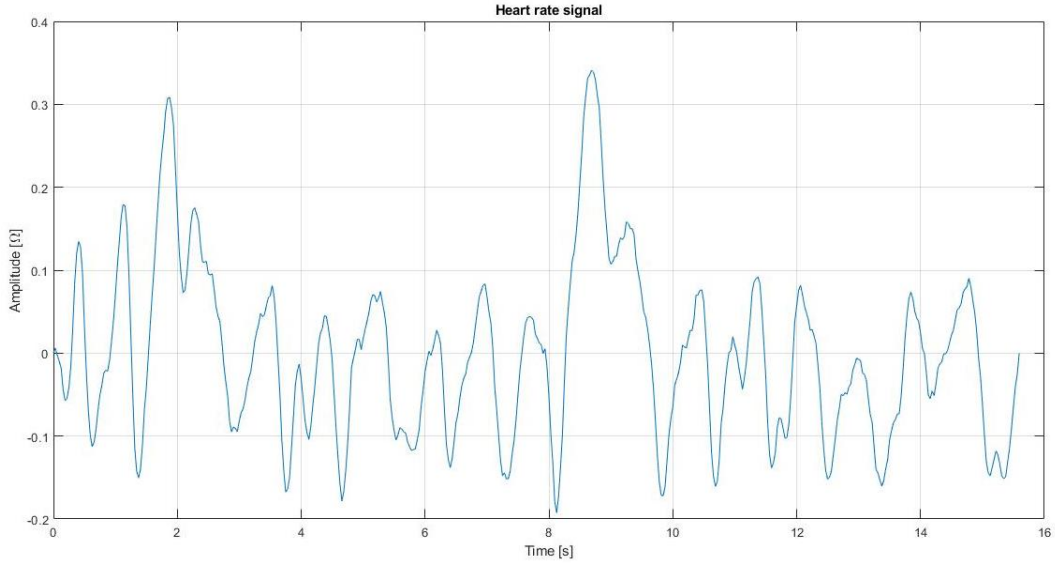


Figure 27. Heart signal from respiratory signal

Considering that the heart beats at a frequency of 1 Hz – 1.5 Hz, by doing the FFT of the signal in Figure 27 and checking the fundamental frequency it is possible to finally determine heart rate of the subject:

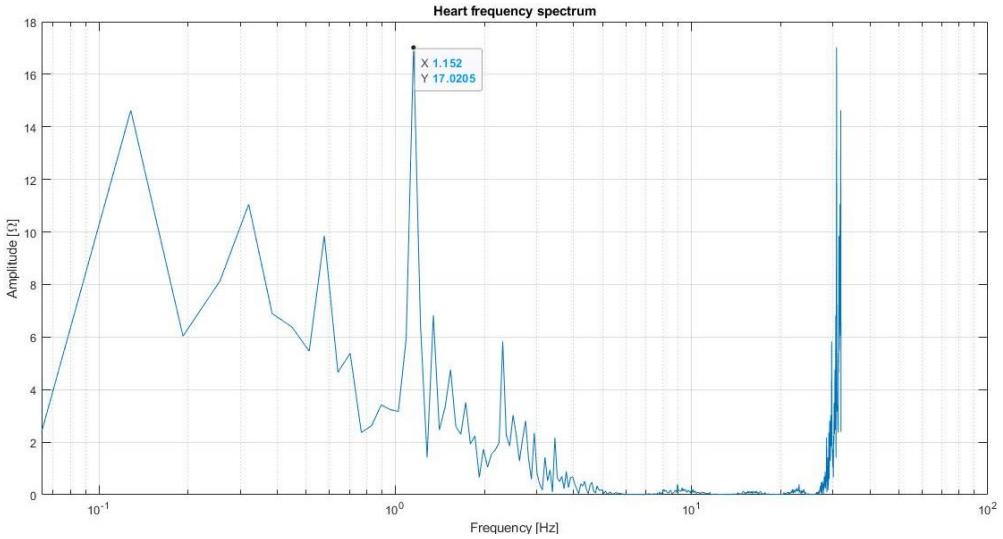


Figure 28. FFT module of heart signal

From Figure 28 we know that the fundamental frequency is at 1.152 Hz. Considering that 1 Hz corresponds to 60 bpm, the subject under test had a cardiac frequency of 69 bpm at the moment of the test, which is a reasonable and normal value for cardiac frequency.

3.4.3. Body fat mass

Finally, the last experiment consists in a leg-to-arm measurement with the objective of estimating the body fat mass. The Cole-Cole arch in Figure 30 has been necessary to calculate some parameters related with the body mass. The obtained values are $R_0 = 409.38\ \Omega$, $R_\infty = 269.44\ \Omega$, $\alpha = 0.273$ and $f_c = 26.52\ kHz$. According to the approximations in [9] and [10], the total body water considering a body weigh W = 85 kg and a height H = 170 cm of the subject, as well as the estimated parameters of effective resistivity of the electrical fluid $\rho_\infty = 104.3\ \Omega \cdot cm$, a dimensional body shape factor of $k_b = 4.3$ and a body density of $D_b = 1.05\ kg/L$:

$$TBW = \frac{1}{100}\left(\frac{\rho_\infty k_b H^2 \sqrt{W}}{R_\infty \sqrt{D_b}}\right)^{\frac{2}{3}} = 57.21\ L \tag{17}$$

With (13) and assuming a free fat mass of $k_h = 0.732\ L/kg$, the free fat mass is:

$$FFM = \frac{TBW}{k_h} = 78.15\ kg \tag{18}$$

Finally, the body fat mass can be obtained subtracting the ft free mass from the weight:

$$FM = W - FFM = 6.86\ kg \tag{19}$$

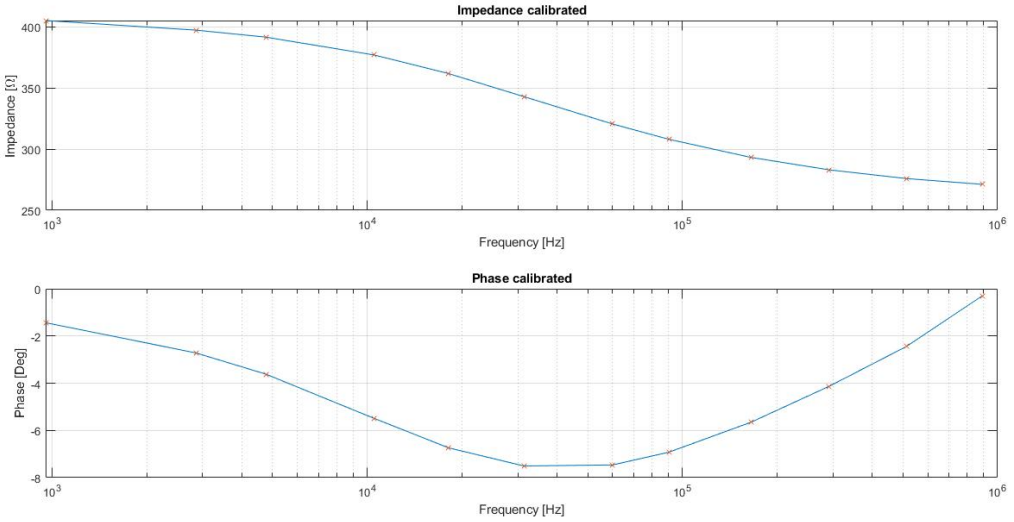


Figure 29. Body fat mass impedance spectrum

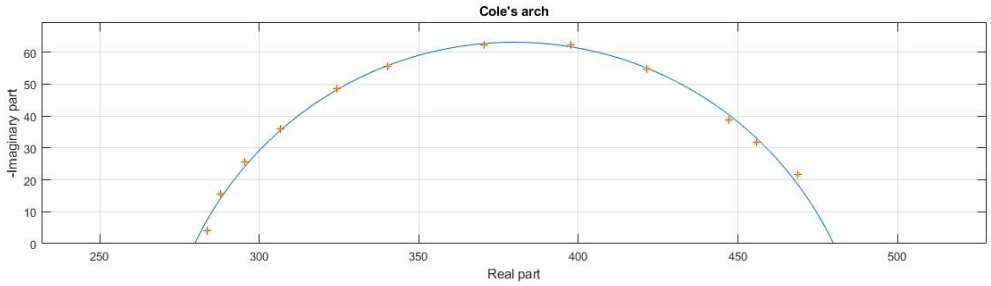


Figure 30. Cole-Cole arch of the body fat mass experiment results. $R_0 = 409.38\,\Omega$, $R_\infty = 269.44\,\Omega$, $\alpha = 0.273$ and $f_c = 26.52\,kHz$.

4. Conclusions and future work

4.1. Conclusions

This work illustrates the implementation of an impedance spectrum analyzer implemented in the RedPitaya development board fully controller by software. Moreover, all the processing and calculation of the impedance coefficients is done inside the processor of the board thus achieving a rate of 30 spectra/second and needing less information transfer through the communication link.

The core of this project is the RedPitaya development board which has already implemented the FPGA hardware blocks in charge of signal generation and acquisition. It has been implemented an algorithm in charge of correctly synchronizing the input and output channels of the board, storing the received raw data from the ports and process it to obtain the final impedance coefficients. Everything without the need of external hardware (excepting the analog front-end).

As this work is developing a biomedical application, the experiments to test and validate the system have been designed in order to give useful parameters in that field as it is the example of the real body fat mass or the cardiac frequency. According to the obtained results, all the desired initial specifications have been successfully accomplished.

4.2. Future work

Now that a functional acquisition and processing system has been developed and validated, there are some improvements regarding future work that can be done to finally obtain a complete closed system or just to improve the actual one. As the designer of this work my future proposals are:

- Improve the FFT calculation by separating the voltage and current calculations in separate threads. This way the algorithm would improve the spectra per second rate achieved. Moreover, from a programming point of view, it makes more sense to have a parallel calculation than the sequential approach implemented on this work as the acquisition process is parallelized. Owing to the time dedication in other aspects it has not been possible to implement.

- Implement a peak detector inside the main algorithm to automatically detect the FFT maximums. As it has been explained in this project, the methodology used to obtain the coefficients is by directly extracting them from the buffer as they have known positions, the multi-sine signal is known. On very interesting improvement would be to automatically detect these peaks so the system could work with any multi-sine signal, even though it was unknown.

- Finally, the most useful improvement that would close the hole project is to implement a web service running inside the RedPitaya processor thus allowing the manipulation and visualization of data in real time from any mobile phone or computer. With this final improvement, the designed system would be completely autonomous, anyone with the knowledge of using a mobile or a pc could perfectly use it.

References

[1] S. Reig Quiroga, Design, implementation and validation of the software of an FPGA-based multifrequency impedance analyser, Barcelona, 2013.

[2] R. Balderas Jiménez, Disseny i implementació d'un sistema d'espectroscòpia d'impedància dinàmic basat en el mòdul Zynq, Barcelona, 2016.

[3] H. P. Schwan, "Analysis of dielectric data: Experience gained with biological material", IEEE Trans. Electrical Insulation. Vol. 20, No. 6, pp. 913-922, 1985.

[4] H. P. Schwan, C. Ed. A. y N. C., “Dielectric properties of cell tissues”. "Interactins between electromagnetic fileds and cells" chapter, PlenumPress, New York, 1985.

[5] K. S. Cole, "Membranes, Ions and Impulses. A chapter of classical biophysics”, University of California Press. Berkeley, 1968.

[6] B.Sanchez, G. Vandesteen, R. Bragós and J. Schoukens. Basics of broadband impedance spectroscopy measurements using periodic excitations., Barcelona: Universitat Politécnica de Catalunya, 2012.

[7] B. Sanchez, X. Fernandez, S. Reig, R. Bragós, An FPGA-based frequency response analyzer for multisine and stepped sine measurements on stationary and time-varying impedance, Measurement Science and Technology, vol. 25, no. 1, p. 015501, 2014..

[8] J. Z. Bao, C. C. Davis and R. E. Schmukler, Impedance spectroscopy of human erythrocytes: System calibration and nonlinear modeling, IEEE Trans. Biomed. Eng., 1993.

[9] B. Sanchez, A. L. P. Aroul, E. Bartolome, K. Soundarapandian and R. Bragós, Propagation of Measurement Errors Through Body Composition Equations for Body Impedance Analysis, Barcelona, June 2014. .

[10] I. Montsech, Design, implementation, and test of a low-cost, low-power, electrical impedance spectroscopy system for biomedical applications, Barcelona: UPC, 2020..

Annex A: Main firmware code implemented

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>
#include <math.h>
#include "rp.h"
#include "kiss_fftr.h"
#define PI 3.14159265

// FUNCTIONS DECLARATION
int acq_fpga_fft_init();
int acq_fpga_fft_clean();
int acq_fpga_fft(double *ch_in, double **ch_out_mod, double
**ch_out_pha);

// GLOBAL VARIABLES DECLARATION
static const int BUFFER_SIZE = 16384;
static const int FFT_LENGTH = 16384;

kiss_fft_cpx *rp_kiss_fft_out = NULL;
kiss_fftr_cfg rp_kiss_fft_cfg  = NULL;

int main(int argc, char **argv){

    if(rp_Init() != RP_OK){
            fprintf(stderr, "Rp api init failed!\n");
    }

    // REQUIRED FILES DECLARATION (SIGNAL, FILTER, V, I)
    FILE *signal_in = fopen("multisine16k_1V_ampTest.csv","r");
    FILE *trian_filt = fopen("triang_filt.csv","r");
    FILE *imp_V = fopen("impedance_V.txt","w");
    FILE *imp_I = fopen("impedance_I.txt","w");

    // BUFFER GENERAL VARIABLES AND SIGNALS MEMORY ALLOCATION
    uint32_t buff_size = 16384;
    uint32_t write_pos;
    float *x = (float *)malloc(buff_size * sizeof(float));
    double *tri_filt = (double *)malloc(buff_size * sizeof(double));

    // KNOWN POSITIONS FOR FFT COEFFICIENTS
    int i,k,j;
    int num_iterations = 100;
    int num_coeffs = 12;
    int max_fft_pos[12] = {1,3,5,11,19,33,63,95,173,307,539,939};

    // STORE TRIANGLE FILTER AND ARBITRARY SIGNAL
    for(i=0;i<buff_size;i++){
        fscanf(signal_in,"%f\n",&x[i]);
        fscanf(trian_filt,"%lf\n",&tri_filt[i]);
    }

    // RESET FPGA GENERATION AND ACQUISITION MODULES
    rp_GenReset();
    rp_AcqReset();

    // CONFIGURE GENERATION FPGA MODULE
    rp_GenWaveform(RP_CH_1, RP_WAVEFORM_ARBITRARY);
```

```
    rp_GenArbWaveform(RP_CH_1, x, buff_size);

    rp_GenFreq(RP_CH_1, 954);
    rp_GenAmp(RP_CH_1, 1.0);

    // CONFIGURE ACQUISITION FPGA MODULE
    rp_AcqSetDecimation(RP_DEC_8);
    rp_AcqSetTriggerLevel(RP_T_CH_2, 0.9);
    rp_AcqSetTriggerDelay(BUFFER_SIZE/2);

    // START GENERATING SIGNAL
    rp_GenOutEnable(RP_CH_1);

    // START CLOCK TO KNOW EXECUTION TIME OF ALGORITHM
    clock_t ex_time;
    ex_time = clock();

    // START LOOP, num_iterations DEFINES THE DESIRED NUMBER OF
SPECTRA
    for(k=0; k<num_iterations;k++){

        // RESERVE THE REQUIRED MEMORY FOR RECEPTION AND FFT BUFFERS
        float *buff_ch1 = (float *)malloc(buff_size * sizeof(float));
        double *buff_worker_ch1 = (double *)malloc(buff_size *
sizeof(double));
        double *buff_fft_ch1_mod = (double *)malloc(sizeof(double) *
buff_size);
        double *buff_fft_ch1_pha = (double *)malloc(sizeof(double) *
buff_size);
        float *buff_ch2 = (float *)malloc(buff_size * sizeof(float));
        double *buff_worker_ch2 = (double *)malloc(buff_size *
sizeof(double));
        double *buff_fft_ch2_mod = (double *)malloc(sizeof(double) *
buff_size);
        double *buff_fft_ch2_pha = (double *)malloc(sizeof(double) *
buff_size);

        // START DATA ACQUISITION MODULE
        rp_AcqStart();

        // SET TRIGGER TO POSITIVE EDGE AND WAIT FOR STATE TO BE
TRIGGERED, ONCE TRIGGERED SIGNALS CAN BE READ
        rp_AcqSetTriggerSrc(RP_TRIG_SRC_CHB_PE);
        rp_acq_trig_state_t state = RP_TRIG_STATE_TRIGGERED;

        while(1){
            rp_AcqGetTriggerState(&state);
            if(state == RP_TRIG_STATE_TRIGGERED){
            break;
            }
        }

        // START PARALLEL READ OF CHANNEL 1 (VOLTAGE MEASUREMENTS) AND
CHANNEL 2 (CURRENT MEASUREMENTS) BUFFERS
        rp_AcqGetWritePointer(&write_pos);
        rp_AcqGetDataV2(write_pos,&buff_size,buff_ch1,buff_ch2);

        //rp_AcqGetOldestDataV(RP_CH_1, &buff_size, buff_ch1);
        //rp_AcqGetOldestDataV(RP_CH_2, &buff_size, buff_ch2);

        // APPLY TRIANGULAR FILTER (NOT NECESSARY)
```

```
        /*for(i=0;i<buff_size;i++){
            buff_worker_ch1[i] = buff_worker_ch1[i]*tri_filt[i];
            buff_worker_ch2[i] = buff_worker_ch2[i]*tri_filt[i];
        }*/

        // INIT FFT MODULES BY ALLOCATING THE REQUIRED MEMORY SPACE
        if(acq_fpga_fft_init() < 0){
            acq_fpga_fft_clean();
            free(buff_ch1);
            free(buff_ch2);
            rp_Release();
            return -1;
        }

        // CALCULATION OF FOURIER TRANSFORM FOR VOLTAGE AND CURRENT
BUFFERS
        acq_fpga_fft(&buff_worker_ch1[0], (double
**)&buff_fft_ch1_mod, (double **)&buff_fft_ch1_pha);
        acq_fpga_fft(&buff_worker_ch2[0], (double
**)&buff_fft_ch2_mod, (double **)&buff_fft_ch2_pha);

        // FROM V AND I BUFFERS, STORE IN EACH OUTPUT FILE JUST THE
KNOWN COEFFICIENTS (12 FOR EACH ITERATION)
        for(j=0;j<num_coeffs;j++){
            fprintf(imp_V,"%lf,%lf,%d\n",
buff_fft_ch1_mod[max_fft_pos[j]],buff_fft_ch1_pha[max_fft_pos[j]],max_
fft_pos[j]);
            fprintf(imp_I,"%lf,%lf,%d\n",
buff_fft_ch2_mod[max_fft_pos[j]],buff_fft_ch2_pha[max_fft_pos[j]],max_
fft_pos[j]);
        }



        // RELEASE ALL USED RESOURCES
        acq_fpga_fft_clean();
        free(buff_ch1);
            buff_ch1 = NULL;
        free(buff_worker_ch1);
            buff_worker_ch1 = NULL;
        free(buff_fft_ch1_mod);
            buff_fft_ch1_mod = NULL;
        free(buff_fft_ch1_pha);
            buff_fft_ch1_pha = NULL;
        free(buff_ch2);
            buff_ch2 = NULL;
        free(buff_worker_ch2);
            buff_worker_ch2 = NULL;
        free(buff_fft_ch2_mod);
            buff_fft_ch2_mod = NULL;
        free(buff_fft_ch2_pha);
            buff_fft_ch2_pha = NULL;

    }
    // SHOW THE TOTAL EXECUTION TIME ON SCREEN
    ex_time = clock() - ex_time;
    double execution = ((double)ex_time)/CLOCKS_PER_SEC;
    printf("Total execution time: %f seconds\n", execution);

    // FREE TRIANGULAR FILTER AND ARBITRARY SIGNAL BUFFERS
    free(x);
```

```
        x = NULL;
    free(tri_filt);
        tri_filt = NULL;
    rp_GenOutDisable(RP_CH_1);
    rp_Release();

    return 0;

}

int acq_fpga_fft_init()
{
    if(rp_kiss_fft_out || rp_kiss_fft_cfg){
        acq_fpga_fft_clean();
    }

    rp_kiss_fft_out =
        (kiss_fft_cpx *)malloc(BUFFER_SIZE * sizeof(kiss_fft_cpx));

    rp_kiss_fft_cfg = kiss_fftr_alloc(BUFFER_SIZE, 0, NULL, NULL);

    return 0;
}

int acq_fpga_fft_clean()
{
    kiss_fft_cleanup();
    if(rp_kiss_fft_out){
        free(rp_kiss_fft_out);
        rp_kiss_fft_out = NULL;
    }
    if(rp_kiss_fft_cfg){
        free(rp_kiss_fft_cfg);
        rp_kiss_fft_cfg = NULL;
    }

    return 0;
}

int acq_fpga_fft(double *ch_in, double **ch_out_mod, double
**ch_out_pha)
{
    double *ch_o = *ch_out_mod;
    double *ch_o_ph = *ch_out_pha;
    int i;

    if(!ch_in || !*ch_out_mod || !*ch_out_pha){
        return -1;
    }

    if(!rp_kiss_fft_out || !rp_kiss_fft_cfg){
         fprintf(stderr, "acq_fpga_fft not initialized");
         return -1;
        }

    kiss_fftr(rp_kiss_fft_cfg, (kiss_fft_scalar *)ch_in,
rp_kiss_fft_out);

    // FFT limited to Fs/2 as there is no need to save the
periodification
    for(i = 0; i < FFT_LENGTH/2; i++){
```

```
        ch_o[i] = rp_kiss_fft_out[i].r;
        ch_o_ph[i] = rp_kiss_fft_out[i].i;
    }

    return 0;
}
```

Annex B: MatLab representation code

```
clear all
close all
clc
% GENERAL PARAMETERS
Fs = 15.6e6;            % Sampling frequency
T = 1/Fs;               % Sampling period
L = 16384;              % Length of signal
t = (0:L-1)*T;
f = Fs*(0:(L-1))/L;

%% IMPORT SIGNAL AND COEFFICIENTS

coeffs_mod =
csvread('C:\Users\marce\Desktop\UNI\MEE\Q3\IR\TFM_Mem_Results\impedanc
e_mod.txt');
coeffs_pha =
csvread('C:\Users\marce\Desktop\UNI\MEE\Q3\IR\TFM_Mem_Results\impedanc
e_pha.txt');

%% COEFFICIENTS PROCESSING FOR MOD AND PHASE
impedanceX = coeffs_mod(:,3);
module_vector = complex(coeffs_mod(:,1),coeffs_mod(:,2))*1000;
phase_vector = complex(coeffs_pha(:,1),coeffs_pha(:,2));
complex_impedance = (module_vector./phase_vector);
complex_impedance_phase = angle(complex_impedance);

coeffs_pos = [1,3,5,11,19,33,63,95,173,307,539,939]; %Sample positions
of the coefficients

final_coeffs_mod_NC=[];final_coeffs_mod_C=[];
final_coeffs_angle_NC=[];final_coeffs_angle_C=[];

% Calibrate the raw coeicients
zm_51 = [-44.6990494582778 - 0.126608447030290i,-44.6480407350372 +
0.154648964976589i,-44.6055440426653 + 0.319154405132030i,-
44.3245545349840 + 0.603771015154968i,-43.8840667036474 +
0.689576754750671i,-43.4109511134249 + 0.477811324380020i,-
43.0581619624641 - 0.210134149324751i,-42.9017888875397 -
0.752856114990535i,-42.9135068304520 - 2.10007597044607i,-
43.1164354795468 - 2.01112796262936i,-42.7224169807245 -
5.10424827345497i,-42.2538460949995 - 9.40129517714810i];
zm_100 = [-97.7899254065843 + 0.164484765279426i,-97.7101529733256 +
0.325461414572930i,-97.6849058352528 + 0.424582682150166i,-
97.2873397830392 + 0.781036241931290i,-96.7600153400667 +
0.824843915932537i,-96.1348413344693 + 0.379657465219665i,-
95.6076976276256 - 0.687484615630656i,-95.3370585474023 -
1.53141713293398i,-95.2111194734964 - 3.74290792468531i,-
95.6701899136574 - 4.86621429842634i,-95.4466875376101 -
10.0316671419686i,-95.5918206375715 - 17.6279876621993i];
zm_200 = [-194.273739211977 - 0.0438906520337707i,-194.175422360071 +
0.259195216828560i,-194.085521730665 + 0.487906639654271i,-
193.682916806152 + 0.817851154114718i,-192.912333809970 +
0.794260776537110i,-192.081184393955 + 0.187041909008138i,-
191.446395827006 - 1.59044194804481i,-191.229077138486 -
3.11635084429169i,-191.152248470393 - 6.63253324124656i,-
191.554592248566 - 10.1676090235527i,-191.214811549116 -
19.6567850669284i,-191.160931798441 - 34.6162873206757i];
```

```
zm_510 = [-498.757044124281 - 0.0772638247127680i,-498.688648734786 +
0.239958852134571i,-498.633702503224 + 0.623086295325133i,-
498.105199472323 + 0.850158344550908i,-496.771776538016 +
0.551507728217158i,-495.459889341309 - 0.806735721936529i,-
494.508111682810 - 4.66172938610902i,-494.258849833638 -
8.19381077612987i,-494.166599135567 - 16.1131365124827i,-
494.541631693798 - 27.1310214392162i,-494.289543325634 -
49.6983273894444i,-494.530497016288 - 86.9376040633451i];
zm_1000 = [-974.091268885932 - 0.177613185417709i,-973.998433740153 +
0.180874073750818i,-973.802792660464 + 0.764845628678706i,-
973.121418031372 + 0.900882458079994i,-971.005154208130 +
0.315817271001525i,-968.845489474155 - 2.23106845239757i,-
967.337910343682 - 9.25014710685837i,-967.193994928793 -
15.7763221198167i,-967.127331303163 - 30.0205909140823i,-
967.622453956882 - 53.2148208991124i,-967.517976282203 -
95.4940683514293i,-968.541992302988 - 166.438773993765i];
A = [];
C = [];
mean_NC =[];
manual_mod_calib =
[1.03166873581575,1.03220005538645,1.03274255709646,1.03494511891756,1
.03894916474910,1.04325522264493,1.04678839958772,1.04791839100987,1.0
4775872712895,1.04459891051270,1.04257167088971,1.03175026433227];
manual_pha_calib = [-0.762364880755598,-1.30071991262699,-
3.02041563297107,-5.58187385868746,-10.2566081120738,-
20.3197572569601,-31.0999024417382,-57.1230356283771,-
100.258773551602,-180.010555650085,-220.030784037487,0];

for i = 1:size(coeffs_pos,2)
    temp_impedanceX = (impedanceX==coeffs_pos(i));

    % NON CALIBRATED COEFFICIENTS

final_coeffs_mod_NC(i)=mean(abs(nonzeros(diag(temp_impedanceX*complex_
impedance.'))));

final_coeffs_angle_NC(i)=180/pi*mean(nonzeros(diag(temp_impedanceX*com
plex_impedance_phase.')));

    % MANUAL CALIBRATED COEFFICIENTS

final_coeffs_mod_C_man(i)=mean(abs(nonzeros(diag(temp_impedanceX*compl
ex_impedance.'))))*manual_mod_calib(i);

final_coeffs_angle_C_man(i)=(180/pi)*mean(nonzeros(diag(temp_impedance
X*complex_impedance_phase.')));

    % VENTILATION REPRESENTATION FOR COEFFICIENT 7
    if(i==7)
        resp =
abs(nonzeros(diag(temp_impedanceX*complex_impedance.')));
        resp_im =
180/pi*nonzeros(diag(temp_impedanceX*complex_impedance_phase.'));

plot(abs(nonzeros(diag(temp_impedanceX*complex_impedance.'))),'x');
    end

    % DISPERSION REPRESENTATION
    hold on;
    yyaxis left;
```

```
%
plot(i,abs(nonzeros(diag(temp_impedanceX*complex_impedance.'))),'x');

plot(i,abs(180/pi*nonzeros(diag(temp_impedanceX*complex_impedance_phas
e.'))),'x');
    yyaxis right;
    stdrd_dev(i) =
std(abs(180/pi*nonzeros(diag(temp_impedanceX*complex_impedance_phase.'
))));
%     stdrd_dev(i) =
std(abs(nonzeros(diag(temp_impedanceX*complex_impedance.'))));
    errorbar(i,0,stdrd_dev(i),'x');

    % FOR CALIBRATION PURPOSES

mean_NC(i)=mean(nonzeros(diag(temp_impedanceX*complex_impedance.')));

    % CALIBRATION USING COEFFICIENTS
    A(i:i,:) = getacal(51,510,1000,zm_51(i),zm_510(i),zm_1000(i));
    C(i:i,:) =
useacal(nonzeros(diag(temp_impedanceX*complex_impedance.')),A(i:i,:));

    final_complex_coeffs(i) = mean(C(i:i,:));
    final_coeffs_mod_C(i) = mean(abs(C(i:i,:)));
    final_coeffs_angle_C(i) = 180/pi*mean(angle(C(i:i,:)));
end
title("Phase impedance spectrum ");xlabel("Coefficient number");grid
on;
yyaxis left; ylabel("Angle [Deg]");
yyaxis right; ylabel("Standard Deviation [Deg]");
hold off;

%% COLE COLE
[ro,ri,alf,fc,ecm2]=colez(real(final_complex_coeffs),imag(final_comple
x_coeffs),f(coeffs_pos+1))
plot(real(final_complex_coeffs),-imag(final_complex_coeffs),'x');grid
on;hold off;
%% PLOTTING OF RESULTS

% NON-CALIBRATED COEFFICIENTS REPRESENTATION
figure;tiledlayout(2,1);
nexttile;semilogx(f(coeffs_pos+1),final_coeffs_mod_NC);hold
on;semilogx(f(coeffs_pos+1),final_coeffs_mod_NC,'x');hold off;grid
on;title("Impedance not calibrated");xlabel("Frequency
[Hz]");ylabel("Impedance [\Omega]");
nexttile;semilogx(f(coeffs_pos+1),final_coeffs_angle_NC);hold
on;semilogx(f(coeffs_pos+1),final_coeffs_angle_NC,'x');hold off;grid
on;title("Phase not calibrated [Deg]");xlabel("Frequency
[Hz]");ylabel("Phase [Deg]");
figure;hist(abs(module_vector),15);xlabel("Impedance
[\Omega]");ylabel("Count");title("Non calibrated impedance
histogram");

% CALIBRATED COEFFICIENTS REPRESENTATION
figure;tiledlayout(2,1);
nexttile;semilogx(f(coeffs_pos+1),final_coeffs_mod_C);hold
on;semilogx(f(coeffs_pos+1),final_coeffs_mod_C,'x');hold off;grid
on;title("Impedance calibrated");xlabel("Frequency
[Hz]");ylabel("Impedance [\Omega]");
nexttile;semilogx(f(coeffs_pos+1),final_coeffs_angle_C);hold
on;semilogx(f(coeffs_pos+1),final_coeffs_angle_C,'x');hold off;grid
```

```
on;title("Phase calibrated");xlabel("Frequency [Hz]");ylabel("Phase
[Deg]");
figure;hist(abs(C),15);xlabel("Impedance
[\Omega]");ylabel("Count");title("Calibrated impedance histogram");

% VENTILATION REPRESENTATION
resp_low_filt = filtfilt(ones(5,1),5,resp);
resp_low_filt_im = filtfilt(ones(10,1),10,resp_im);
figure;tiledlayout(2,1);
nexttile;plot(t1,resp,'x');hold
on;plot(t1,resp_low_filt,'LineWidth',2);hold off;grid
on;title("Impedance magnitude");xlabel("Time [s]");ylabel("Impedance
[\Omega]");
nexttile;plot(t1,resp_im,'x');hold
on;plot(t1,resp_low_filt_im,'LineWidth',2);hold off;grid
on;title("Impedance phase");xlabel("Time [s]");ylabel("Phase [Deg]");
```

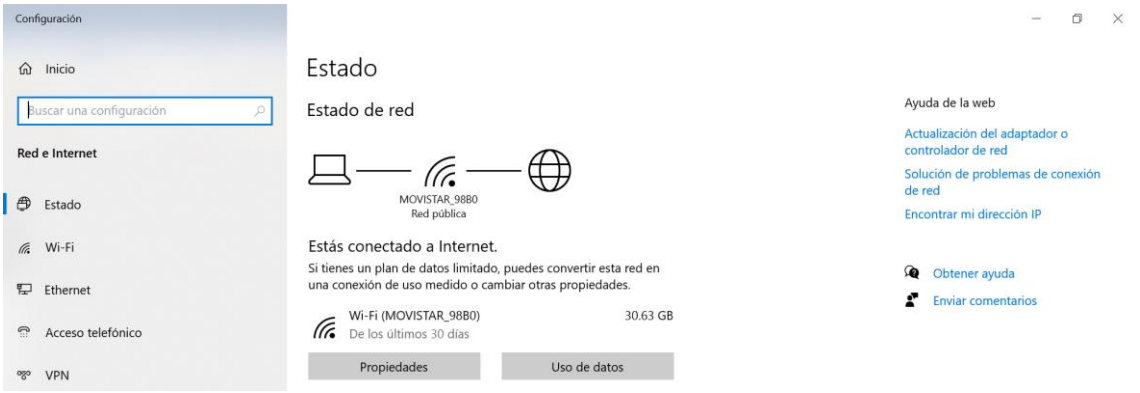
Annex C: RedPitaya guide

This guide is based on the autonomous learning acquired along the project; the process explained is the one that has been used, nevertheless there may be many ways to achieve the same.

Annex C.1: First time connecting RedPitaya to computer

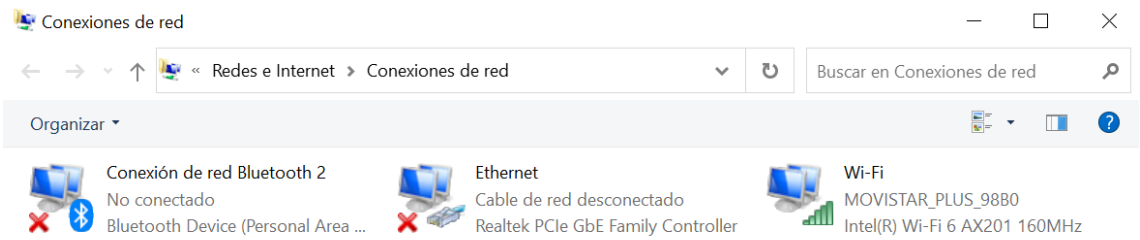
If it is the first time plugging the RedPitaya to the computer using Ethernet link, follow these steps:

1. Go to windows bar and type “Estado de Red”, you should arrive to an interface similar to this one:

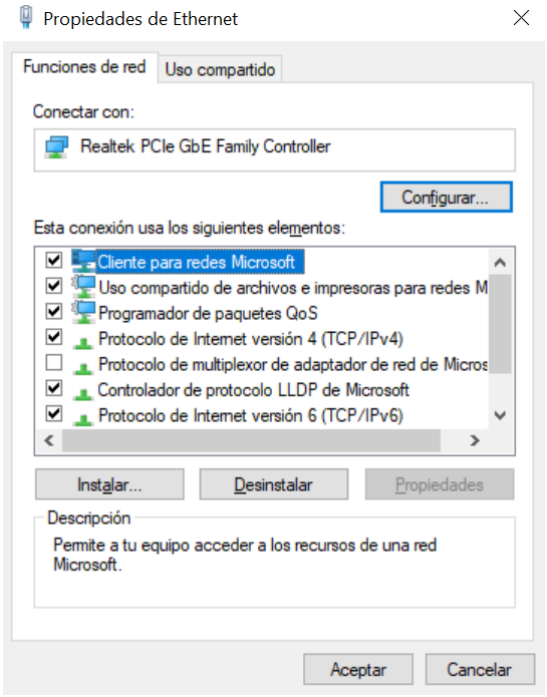


2. Go to “Ethernet” option, the following site will be open:

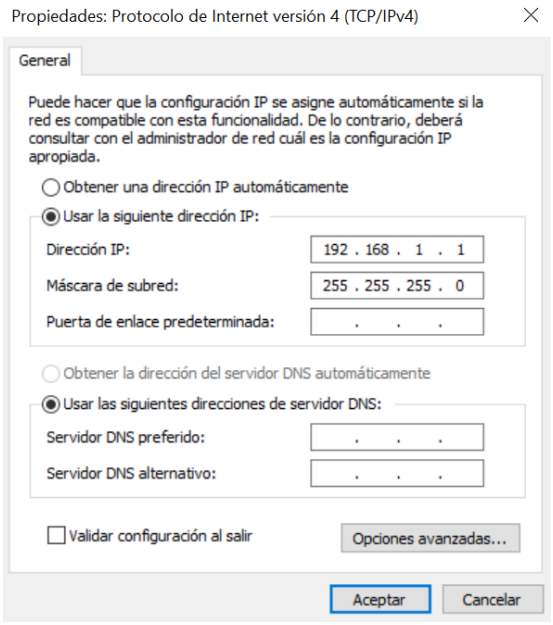
3. Click on “Cambiar opciones del adaptador”, you Will go to a new window:



4. Go to the “Ethernet” option, the new window should look like this:

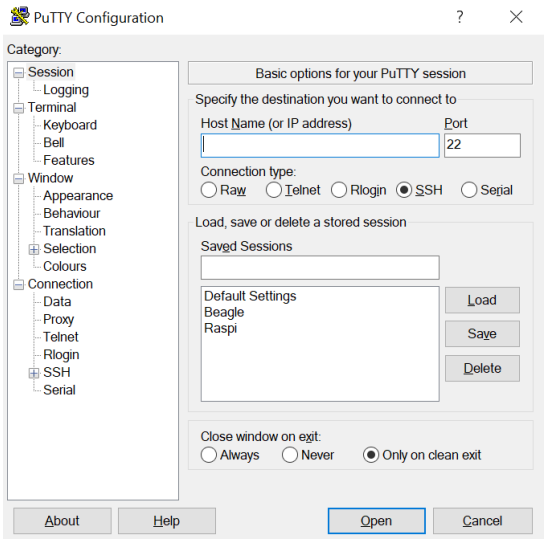


5. Click on “Protocolo de Internet versión 4 (TCP/IPv4)” and select “Propiedades” to open the following window:

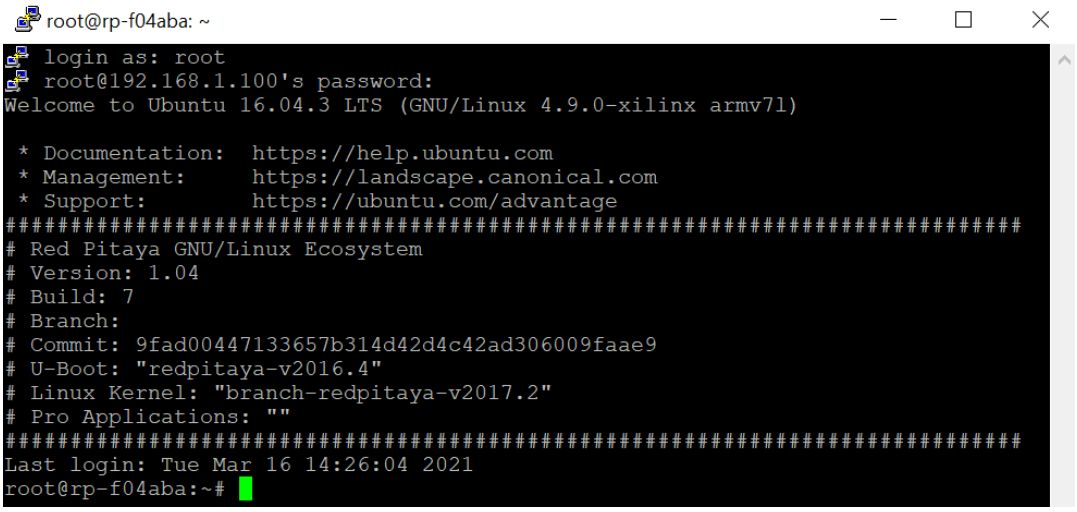


6. As you can see in the previous image, click on “Usar la siguiente dirección IP” and input the desired IP direction. As a recommendation set the 192.168.1.1 as seen in the image. The security mask section will be automatically filled, if not, insert the also the one in the image “255.255.255.0”
7. Now you have set an IP direction that will be assigned when the Ethernet cable is plugged. The next step is to insert the Ethernet cable connecting the RedPitaya with the computer.

8. A software to stablish SSH connection with the RedPitaya must be installed in the computer. For the case of this project the “Putty” software has been installed. It is a free and open software available for Windows (access the following link to install it: https://www.putty.org/).
9. Open the Putty application, the following window should open:



10. If the reader has followed all the steps until here, leave the “Port” section to 22 and insert the Host Name: “192.168.1.100”. Then click “Open”.
11. You will be asked for user and password, both are “*root*”. You should see the following:



12. Congratulations, you are now connected to the RedPitaya board.

Annex C.2: Executing the implemented code

1. Starting from the end of the previous Annex section, the reader may be now inside the root directory. As commented in the work, the operating system of the RedPitaya is linux-based so navigating through the system is equally done as there. To execute the algorithm, go to “Examples” folder by typing:

```
cd RedPitaya/Examples/C/
```

2. The next step is to compile the code. Type the following command:

```
make acquire_multisine_dualfft
```

3. Finally execute the code with:

```
./acquire_multisine_dualfft
```

4. The data will be stored in the files named "*impedance_mod.txt"* and "*impedance_phase.txt"* containing the voltage and current measurements respectively. To transfer the files to the computer, go to the command window of windows (by typing "*cmd"* in the windows bar) and write:

```
scp 192.168.1.100:~/RedPitaya/Examples/C/impedance_mod.txt
      -> C:\Users\Marcel Palet\Desktop\MEE\TFM
```

   This command will copy the file from the first path (the RedPitaya one) to the specified path in the second line. In the second line, the reader must specify the desired path inside his/her computer, this is an example.