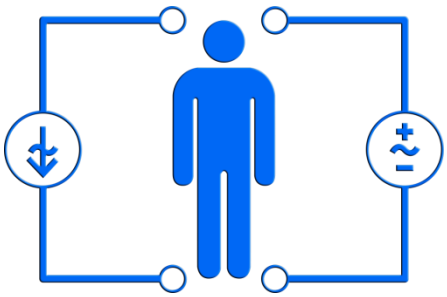
Escola Tècnica Superior
d'Enginyeria de Telecomunicació de Barcelona

MASTER IN TELECOMMUNICATION ENGINEERING

MASTER'S THESIS:

DESIGN AND IMPLEMENTATION OF A LOW-COST FPGA-BASED BIOIMPEDANCE MEASUREMENT SYSTEM



Author: Miguel González Gutiérrez

Director: Ramon Bragós

Year: 2014

Index

Abstract

Currently, many impedance measurement systems have been developed. This project details the design, implementation and characterization of a FPGA-based bioimpedance measurement system, whose goal is obtaining good performance at low costs.

Signal generation and processing circuits were implemented within the FPGA, as well as the NIOS II embedded processor. An ADA conversion board as well as a front-end previously designed and implemented by the group of instrumentation and biomedical engineering were also incorporated. Finally, a communication mechanism between the FPGA and the computer was also designed and implemented.

1. Introduction

Bioimpedance refers to the passive electrical properties of a biological tissue, measured when current flows through it. This impedance varies with frequency and between different tissue types, and varies sensitively with the underlying histology.

Bioimpedancie is not a physiological parameter by itself, but passive electrical properties derived from it (i.e., permittivity, conductivity) exhibit certain states and events of interest for medicine.

Cells may be modelled at low frequencies as a group of electronic components. One of the simplest models employs just three components (figure 1.1). The extracellular space is represented as a resistor (Re), and the intracellular space and the membrane are modelled as a resistor (Ri) and a capacitor (Cm). Both the extracellular space and intracellular space are highly conductive, because they contain salt ions.

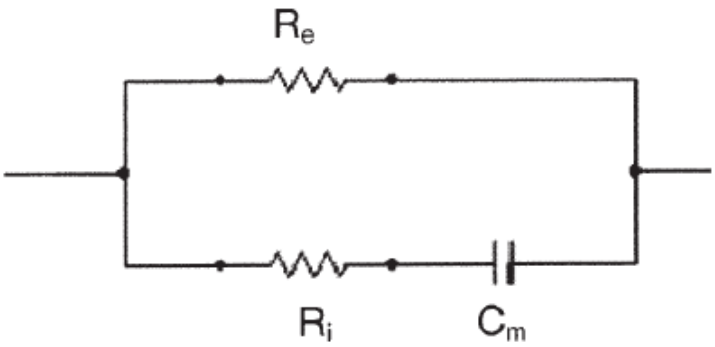


Figure 1.1: the cell modelled as basic electronic circuit

The lipid membrane of cells is an insulator, which prevents current at low frequencies from entering the cells. At lower frequencies, almost all the current flows through the extracellular space only, so the total impedance is largely resistive and is equivalent to that of the extracellular space. As this is usually about 20% or less of the total tissue, the resulting impedance is relatively high. At higher frequencies, the current can cross the capacitance of the cell membrane and enter the intracellular space as well. It then has access to the conductive ions in both the extra- and intra-cellular spaces, so the overall impedance is lower as shown in figure 1.2.

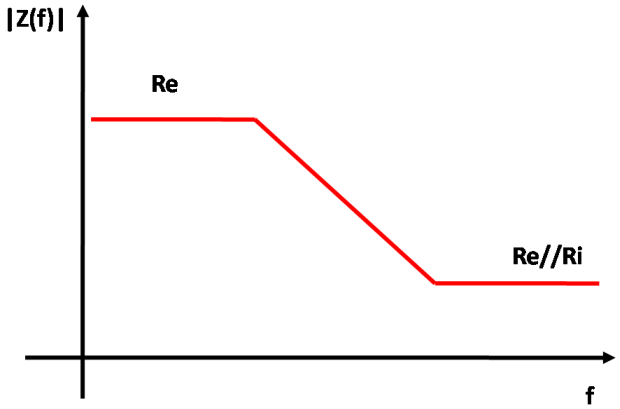


Figure 1.2: ideal frequency response of tissues

The movement of the current in the different compartments of the cellular spaces at different frequencies, and the related resistance and reactance values measured, is usefully displayed as a Cole–Cole plot. This is an extension of the resistance/reactance plot in the complex plane. Instead of the single point for a measurement at one frequency, the values for a range of frequencies are all superimposed. For the electrical circuit model, the arc will be a semicircle (figure 1.3).

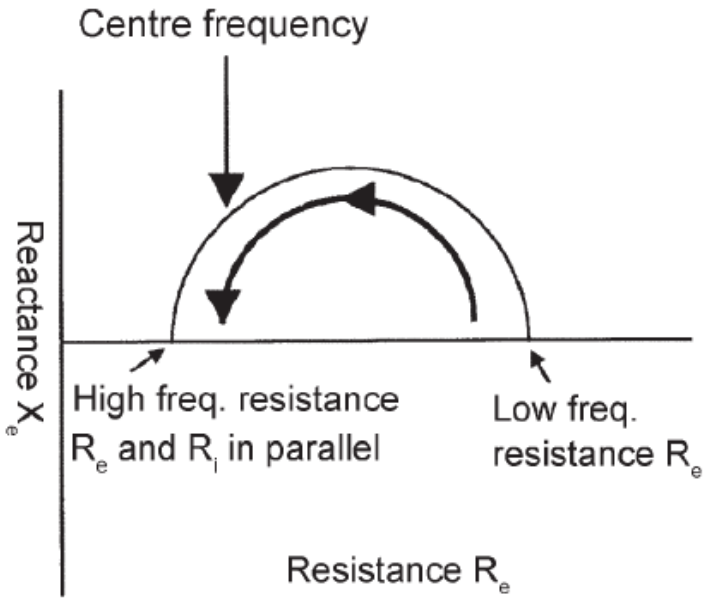


Figure 1.3: Cole-Cole plot

At low frequencies, the measurement is only resistive, and corresponds to the extracellular resistance (no current passes through the intracellular path because it cannot cross the cell membrane capacitance). As the applied frequency increases, the phase angle gradually increases as more current is diverted away from the extracellular resistance, and passes through the capacitance of the intracellular route. At high frequencies, the intracellular capacitance becomes negligible, so current enters the parallel resistances of the intracellular and extracellular compartments. The cell membrane reactance is now nil, so the entire impedance again is just resistive and so

returns to the X axis. Between these, the current passing through the capacitive path reaches a peak. The frequency at which this occurs is known as the centre frequency (Fc), and is a useful measure of the properties of an impedance.

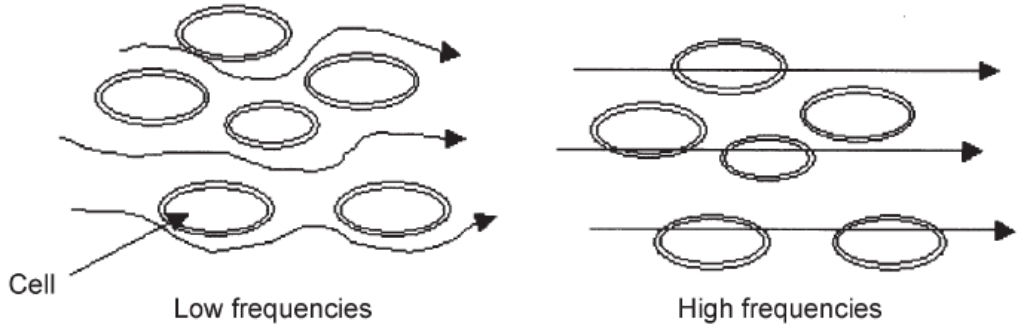


Figure 1.4: the movement of current at both low and high frequencies

In real tissue, the Cole–Cole plot is not exactly semicircular, because the detailed situation is clearly much more complex; the plot is usually approximately semicircular, but the centre of the circle lies below the x-axis. Inspection of the Cole–Cole plot yields the high- and low-frequency resistances, as the intercept with the x-axis, and the centre frequency is the point at which the reactance is greatest. The angle of depression of the centre of the semicircle is another means of characterizing the tissue.

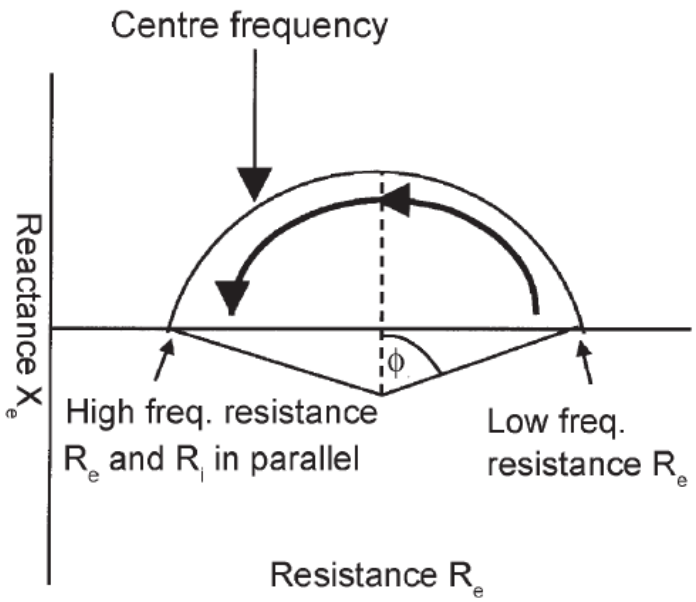


Figure 1.5: Idealized Cole-Cole plot for tissue

Over the frequency ranges used for electrical impedance spectroscopy (EIS), about 100Hz to 100MHz, the resistance and reactance of tissue gradually decreases. This is due to the simple effect of increased frequency passing more easily across capacitance, but also because cellular and biochemical mechanisms begin to operate,

which increases the ease of passage of the electrical current. A remarkable feature of live tissue is an extraordinarily high capacitance, which is up to 1000 times greater than inorganic materials, such as plastics used in capacitors. This is because capacitance is provided by the numerous and closely opposed cell membranes of cells, each of which behaves as a tiny capacitor.

Over this frequency range, there are certain frequency bands where the phase angle increases, because mechanisms come into play which provide more capacitance. They may be seen as regions of an increased decrease of resistance in a plot of resistance against frequency, and are termed ‘dispersions’. At the low end of the frequency spectrum, the outer cell membrane of most cells is able to charge and discharge fully. This region is known as the alpha dispersion and is usually centred at about 100Hz. As the frequency increases, from 10 kHz–10 MHz, the membrane only partially charges and the current charges the small intracellular space structures, which behave largely as capacitances. At these higher frequencies the current can flow through the lipid cell membranes, introducing a capacitive component. This makes the higher frequencies sensitive to intracellular changes due to structural relaxation. This effect is largest around 100 kHz, and is termed the ‘beta dispersion’. At the highest frequencies, dipolar reorientation of proteins and organelles can occur, and affect the impedance measurements of extra- and intracellular environments. This is the gamma dispersion, and is due to the relaxation of water molecules and is centred at 10 GHz. Most changes between normal and pathological tissues occur in the alpha and beta regions of dispersion spectrum[1].

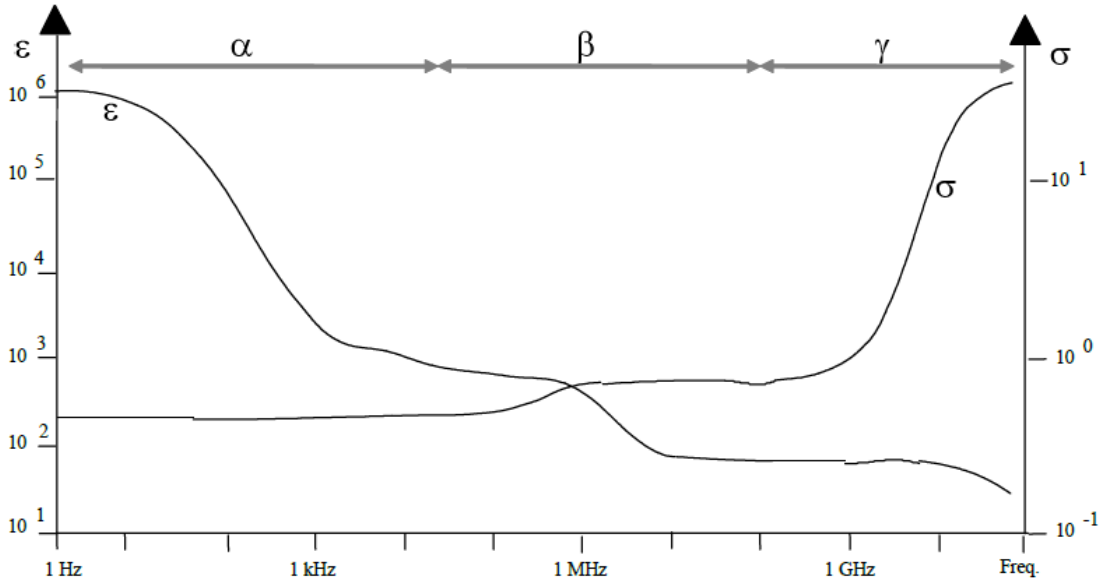


Figure 1.6: evolution of permittivity and conductivity with frequency [2]

Measurements of bioimpedance of biological tissues can be classified into two groups. The first one is the study of impedance variations associated with the circulatory system and the respiratory system which depends on geometry changes. The second

one involves the determination of properties of corporal tissues, such as hydration, lipid percentage and, generally, the state of tissues and cells constituting it.

The instrumentation used for this purpose, besides being relatively cheap, has the advantage of using a nonionizing technique that is not invasive. However, measurement are highly influenced by many factors, such as geometry, movement and blood flow.

Application	Impedance range (magnitude-phase)		Frequency range	Accuracy		# of electrodes
Tomography	1-100 Ω	5-30°	10 kHz-1 MHz	0.1%	0.1°	≥16
Bioelectrical Impedance Analysis (BIA)	0.2-2 kΩ	5-50°	1 kHz-1 MHz	3%	0.5°	4 to 16
Tissue spectroscopy	50-500 Ω	0-40°	0.1 Hz-1 MHz	1%	0.2°	4
Biomass spectroscopy	10-100 Ω	0.1-5°	10 kHz-20 MHz	0.1%	0.02°	4

Table 1.1: specifications for different applications based on bioimpedance measurements [3]

1.1. Project context

When starting this project, two different bioimpedance measurement systems had been developed in final degree projects. One of them, developed by Álvaro Andrade Sánchez & Joan Mendoza Equiza [4], exhibits low costs and good performance but also a lack of flexibility, whereas the other one (developed by Sergi Reig Quiroga & Xavier Fernández Espiga [5]) is a high accuracy system with high costs. Features of both systems are shown in section 1.4.

1.2 Objectives

The objective of this project is the design and implementation of a low-cost FPGA-based multifrequency bioimpedance measurement system, using signal generation and coherent demodulation circuits and incorporating an embedded processor for control tasks, achieving low costs close to the system developed by Álvaro Andrade Sánchez & Joan Mendoza Equiza but providing higher performance and flexibility.

The system allows the user to choose measurement parameters, such as frequencies of measurement, sampling frequencies, number of frequencies to measure simultaneously and the length of the filter (it will be explain later). Although the system is able to perform measurements even over 10 MHz, finally we focus on five different frequencies: 8 kHz, 32 kHz, 48kHz, 64 kHz and 96 kHz, since these are frequencies of special interest in the measurement of tissues (about 10 to 100 kHz) and the respiratory rate and waveform (50 kHz), measuring impedances in the range from 10 Ω to 1 kΩ with an accuracy of 1% in magnitude and a speed of 20 measurements per second at each frequency.

1.3 Methods for measuring bioimpedances

There are many different methods for the measurement of bioimpedances: based on impedance bridging, on resonances, on I-V, on load analysis, etc [6]. The I-V method with 4 wires will be used. With this method, the value of an unknown impedance Z is calculated using the measured values of voltage and current.

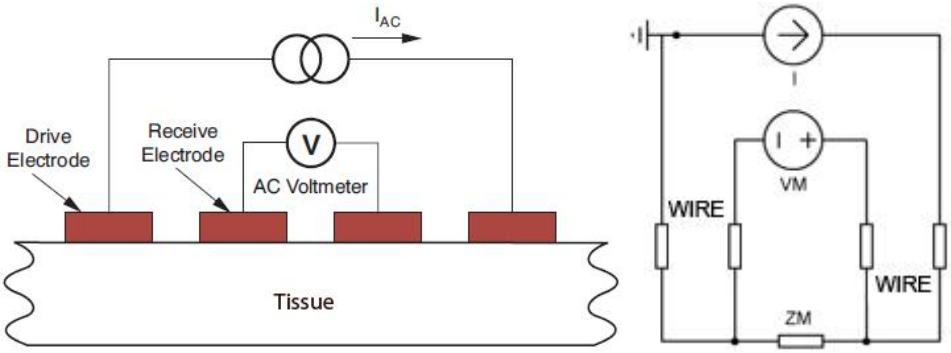


Figure 1.7: scheme of I-V measurement

The Kelvin method or 4-wires method provides two advantages: the impedance of the two conductors that injects the current does not affect the measurement and the current through the other two wires is almost zero due to the high impedance of the voltmeter. This strategy is needed on bioimpedance measurements due to the fact that the electrode-tissue interface impedance is usually higher than the tissue impedance and varies with time and temperature.

1.4. State of the art

As mentioned in section 1.1, this project aim to be an intermediate point between the previously developed systems. Then, a brief description of both systems is presented as well as a commercial chip and another system developed by the group of instrumentation and biomedical engineering. There are several commercial systems that measure bioimpedances but only those developed at the ESI lab will be described.

- Álvaro Andrade Sánchez & Joan Mendoza Equiza's system

This system allow measurements at 1 MHz, with a period between samples of 512 µs and for impedances within the range of 10 to 130 Ω. It was designed to estimate the respiratory rate of car drivers in order to detect drowsiness. It costs approximately 160$ [4].

- Sergi Reig Quiroga & Xavier Fernández Espiga's system

This system offers two operation modes, allowing measurements at one unique frequency or simultaneously at 25 frequencies between 48 kHz and 8 MHz using multi-

sine signals and a FFT module, achieving relative errors below 1%. Its cost is approximately 2500$ [5].

- AD5933

Integrated AD5933. Low cost chip (approx. 20$) that allows the measurement of impedances between 1kΩ and 10 MΩ at lower frequencies than 100 kHz with an accuracy of 5%. Several systems have been developed around this chip, but it is not fast nor accurate enough to measure dynamic systems [7].

- System based on PCI eXtensions for Intrumentation (PXI)

The system includes the embedded dual core controller *PXIe-8130* controlled by software through LabView, the data acquisition board PXIe-5122 with two channels (100 MSPS, 64 MB per channel, 14 bits), the data acquisition board PXIe-5105 with 8 channels (60 MSPS, 128 MB, 12 bits) and the board PXIe-5422 (200 MSPS, 32 MB, 16 bits) as arbitrary wave generator. The system is able to continuously generate an arbitrary excitation reference signal and acquire and store various channels up to 5 MSPS. Resultant data is stored in a binary file for a later processing. Providing high performance, its main drawback is the cost (~40000$) [8].

2.Structure and design of the measurement system

2.1. Global view

The components of the measurement system to be developed are described in this section. The general structure is given in figure 2.1:

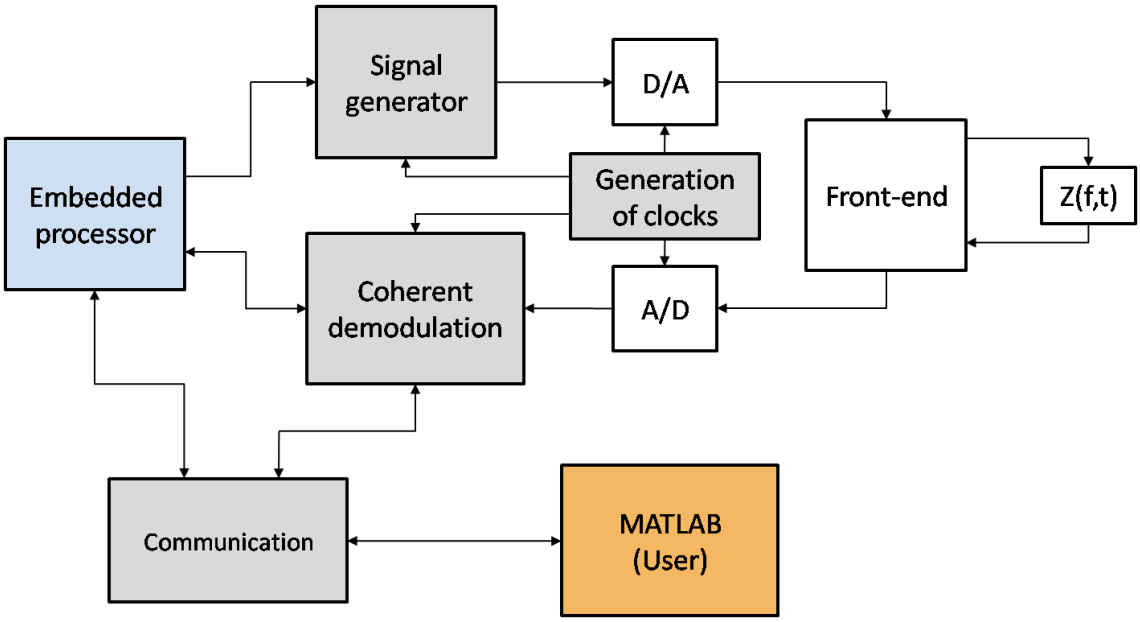


Figure 2.1: general structure of the measurement system

As shown, the system contains an embedded processor which is in charge of controlling the parameters and behavior of the other components. This processor, as well as the grey coloured components, have been implemented within the hardware resources of the FPGA. The *signal generator* block generates a sinusoid signal by a NCO and it is transmitted to the DAC and injected to the analog front-end. After being modified by the impedance under measurement, the sinusoid signal comes back to the system through the ADC and is processed by the *coherent demodulation* block. Once the results are obtained, the *Communication* block manages them for being transmitted to a MATLAB application running on a computer. A MATLAB application has been chosen to implement the user interface, but both signal generation and modulation are performed in the FPGA and only coefficients at every frequency are transmitted through the serial link.

Hardware descriptions of the different components implemented in the FPGA are shown on appendix 9.1.

2.2. Development platform

The project is based on the hardware platforms shown in this section.

2.2.1. DE0-Nano board

The system described on this project is implemented on *Terasic's* DE0-Nano board [9], whose kernel is the Cyclone IV FPGA of *Altera*. This device was chosen because its low cost, being the cheapest development system including an FPGA (60$).

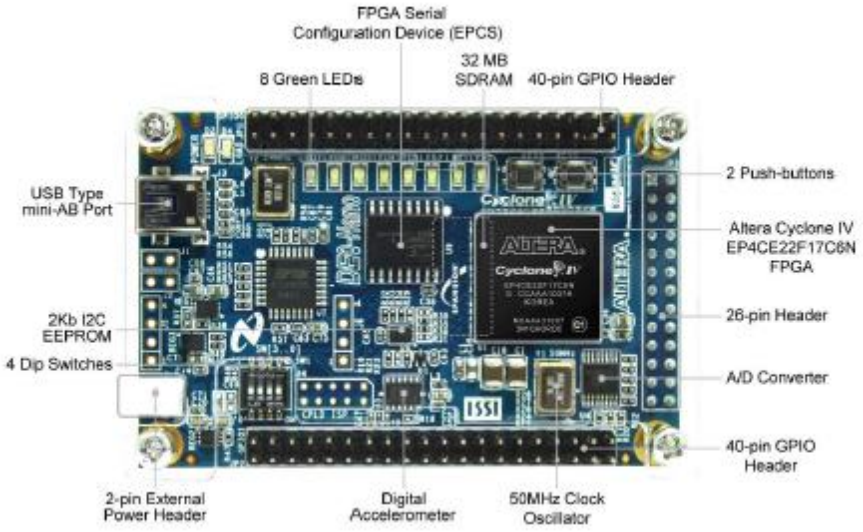


Figure 2.2: front view of DE0-Nano board

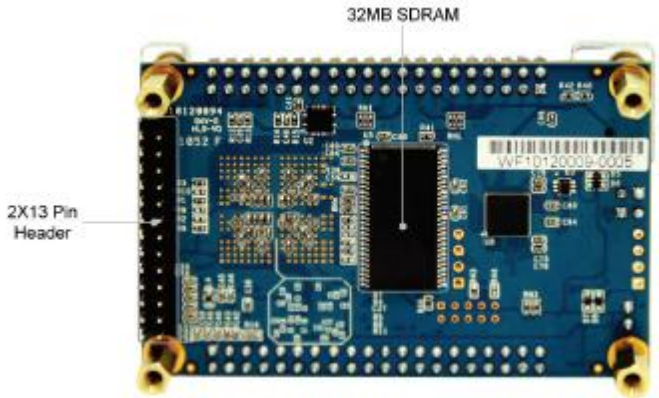


Figure 2.3: back view of DE0-Nano board

The key features of the board are listed below:

- Featured device
  - Altera Cyclone® IV EP4CE22F17C6N FPGA
  - 153 maximum FPGA I/O pins
- Configuration status and set-up elements
  - On-board USB-Blaster circuit for programming
  - Spansion EPCS64
- Expansion header
  - Two 40-pin Headers (GPIOs) provide 72 I/O pins, 5V power pins, two 3.3V power pins and four ground pins
- Memory devices
  - 32MB SDRAM
  - 2Kb I2C EEPROM
- General user input/output
  - 8 green LEDs
  - 2 debounced pushbuttons
  - 4-position DIP switch
- G-Sensor
  - ADI ADXL345, 3-axis accelerometer with high resolution (13-bit)
  - A/D Converter
  - NS ADC128S022, 8-Channel, 12-bit A/D Converter
  - 50 Ksps to 200 Ksps
- Clock system
  - On-board 50MHz clock oscillator
- Power Supply
  - USB Type mini-AB port (5V)
  - DC 5V pin for each GPIO header (2 DC 5V pins)
  - 2-pin external power header (3.6-5.7V)

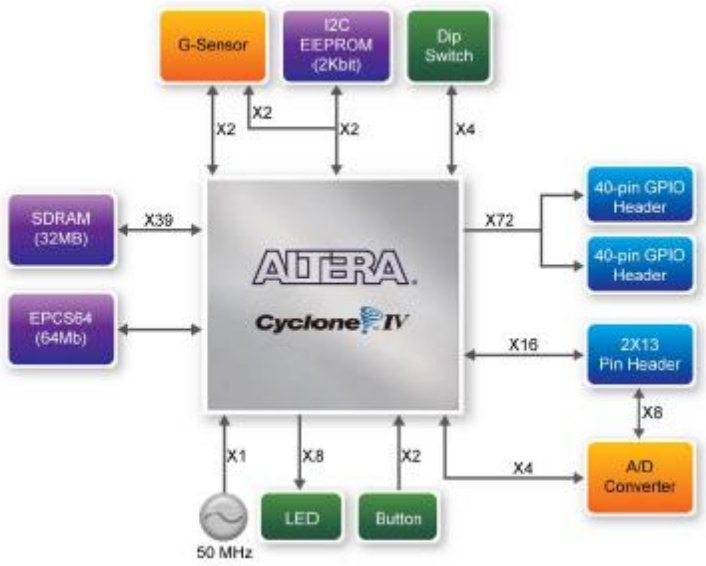


Figure 2.4: block diagram of DE0-Nano board

The DE0-Nano board advantages are its size and weight, as well as its capacity for being reconfigured without having to incorporate a superfluous hardware, distinguishing from other general purpose development boards.

Besides, it provides for designers three different options of power supply, which is really important for mobile designs where the power supply is crucial.

One of the handicaps of this board is the lack of communication ports with external devices if not using the GPIO pin headers. Furthermore, due to the low speed of the A/D and D/A converters, it will be needed to incorporate the THDB-ADA board (section 2.2.2)

FPGA: Cyclone IV

Altera's Cyclone IV family devices provide medium performance at low costs, what is beneficial for cost-sensitive projects. Besides, the Cyclone IV is one of the FPGAs with the lowest power consumption for a reduced size. Providing up to 115.000 logical elements, it consumes up to a 30% less of power than previous versions.

The version included in the DE0-Nano board is the Cyclone IV EP4CE22, whose features are depicted in the table below:

DEVICE	EP4CE6	EP4CE10	EP4CE15	EP4CE22	EP4CE30	EP4CE40	EP4CE55	EP4CE75	EP4CE115
Logic Elements	6.272	10.320	15.408	22.320	28.848	39.600	55.856	75.408	114.480
M9K memory blocks	30	46	56	66	66	126	260	305	432
Embedded memory (Kbits)	270	414	504	594	594	1.134	2.340	2.745	3.888
18-bit x 18-bit multipliers	15	23	56	66	66	116	154	200	266
PLLs	2	2	4	4	4	4	4	4	4
Maximum user I/Os	179	179	343	153	532	532	374	426	528
Maximum differential channels	66	66	137	52	224	224	160	178	230

Figure 2.5: features of Cyclone IV devices

2.2.2 THDB_ADA

This board carries out the D/A conversion of the signal coming from the FPGA for the measurement of the impedance, as well as the A/D conversion of the resulting signal coming from the front-end. The THDB-ADA provides full compatibility with the DE0-Nano board since it is developed by the same manufacturer (*Terasic*).

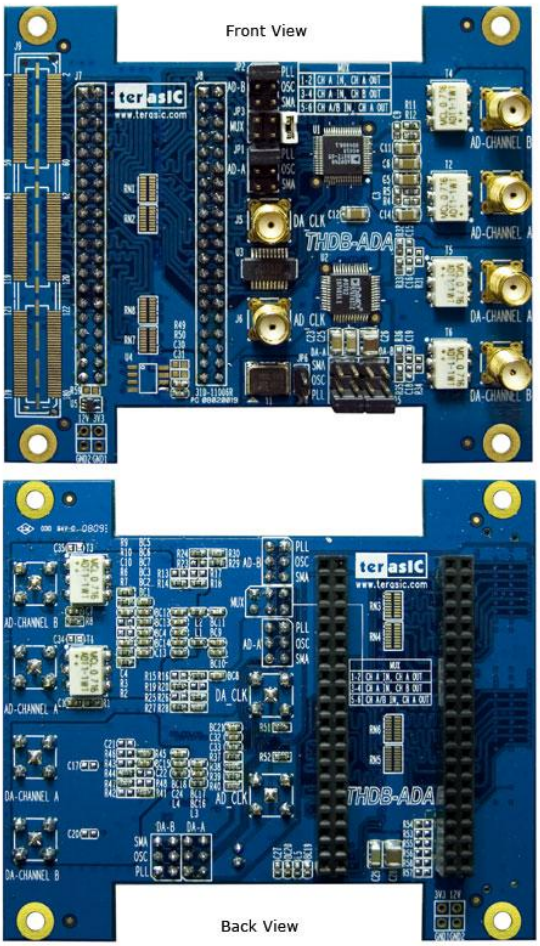


Figure 2.6: ADA GPIO

The main features of the THDB-ADA [10] are listed below:

1. Dual AD channels with 14-bit resolution and data rate up to 65 MSPS
2. Dual DA channels with 14-bit resolution and data rate up to 125 MSPS
3.GPIO interface, which is fully compatible with DE1/DE2/DE2_70/DE2_115/DE3/DE4.
4. Clock sources include oscillator 100MHz, SMA for AD and DA each, and PLL from GPIO interface
5. AD converter analog input range 2V p-p range.
6. DA converter output voltage range 2V p-p range.
7. DA and AD converters do not support DC signaling

ADC and DAC provide data rates up to 65 MSPS and 125 MSPS respectively, whereas maximum frequencies of work are lower than 50 MSPS, so they fulfill this specification. In spite of the fact that each converter provides two channels, only use one of them is used.

The schematics of both converters are shown below:

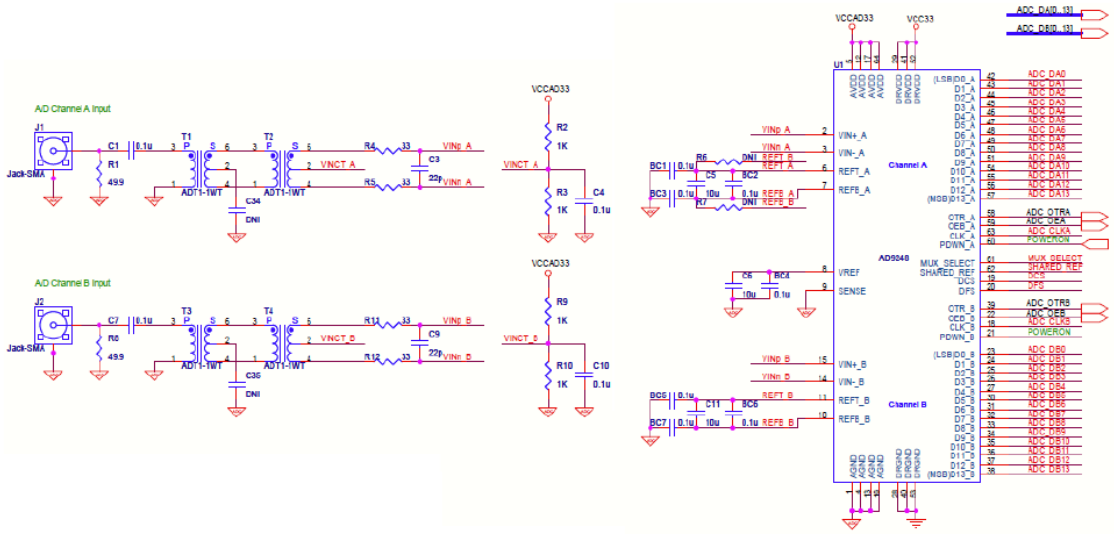


Figure 2.7: Analog/digital converter schematic

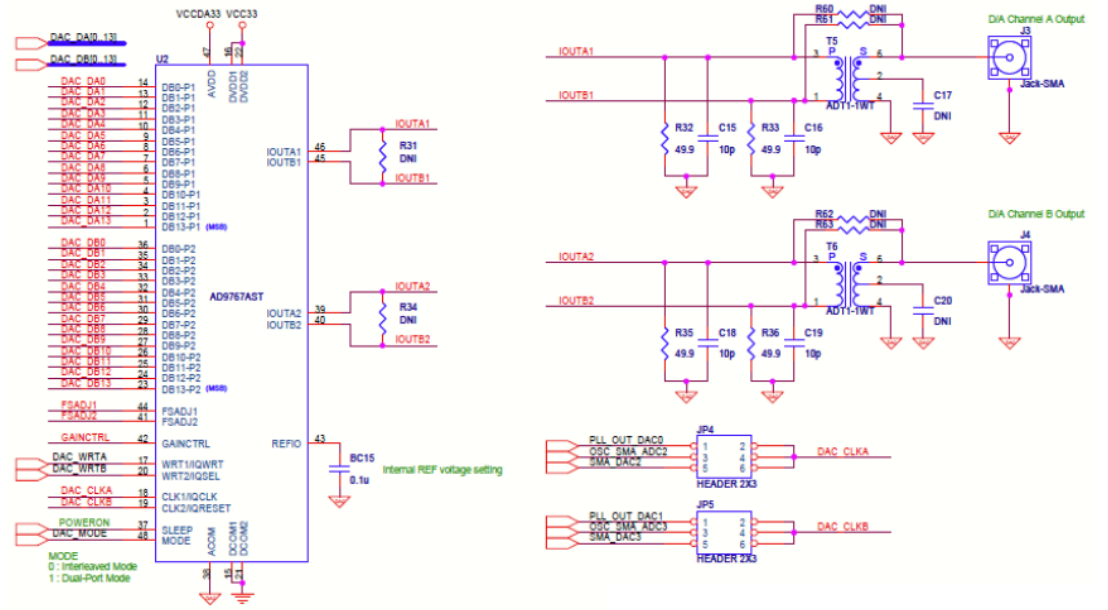


Figure 2.8: Digital/analog converter schematic

As depicted in the schematic, two transformers (ADT1-1WT from *Minicircuits*) are located at the input of the ADC, transforming incoming unipolar signals into differential signals, since the AD9248 works with differential signals. Besides, those transformers are used for AC coupling, eliminating any DC that could remain at the inputs.

However, ADT1-1WT transformers have a high-pass response with a cutoff frequency of 400kHz. As mentioned in section 1.2, the system focuses on the band from 10 to 100 kHz, so that the transformers on one of the channels were replace with the ADT1-6T, from the same manufacturer and encapsulation and featuring a cut-off frequency of 30 kHz which is the lowest cutoff frequency transformer which has a compatible pin-out.

Regarding to the DAC, outputs of both channels are differential currents which are injected to a load, transforming them into differential voltages that are turned into unipolar voltages by means of the previously mentioned transformers. In this case, the role of the transformers is fundamental, since they suppress any vestige of DC. Once again, as it was done with the ADC, the transformer of one channel was replaced with a ADT1-6T.

Low-pass filters are not used (reconstruction filter and anti-aliasing filter) because the Front-end has a limited bandwidth in both the current injection and voltage detection.

Finally, underlining that DE0-Nano/THDB-ADA interfacing is not direct, since physical dimensions do not match between them. For this reason, the matching board shown below was built:

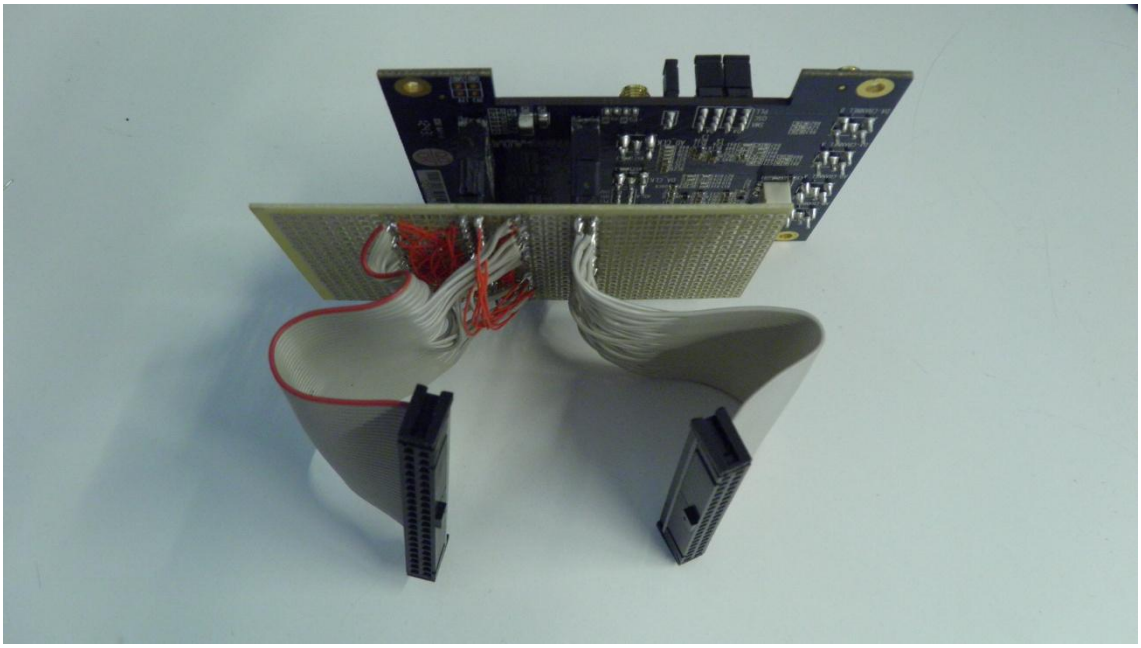


Figure 2.9: DE0-Nano/THDB-ADA matching board

The next table lists pin associations between DE0-Nano board and THDB-ADA:

DE0-Nano board GPIO 0		THDB-ADA GPIO 0 (J7)		
Pin no.	**FPGA pin no.**	**Pin no.**	**Schematic name**	**Description**
1	PIN_A8	1	ADC_OTRA	A/D Out-of-Range Indicator Channel A
2	PIN_D3	19	PLL_OUT_ADC0	PLL Clock input Channel A
3	PIN_B8	3	ADC_OTRB	A/D Out-of-Range Indicator Channel B
4	PIN_C3	4	ADC_DB1	A/D Data Output bit 1 Channel B
5	PIN_A2	5	ADC_DB2	A/D Data Output bit 2 Channel B
6	PIN_A3	6	ADC_DB4	A/D Data Output bit 4 Channel B
7	PIN_B3	7	ADC_DB3	A/D Data Output bit 3 Channel B
8	PIN_B4	8	ADC_DB5	A/D Data Output bit 5 Channel B
9	PIN_A4	9	ADC_DB6	A/D Data Output bit 6 Channel B
10	PIN_B5	10	ADC_DB8	A/D Data Output bit 8 Channel B
11	VCC-SYS	11	-	-
12	GND	12	GND	Ground
13	PIN_A5	13	ADC_DB7	A/D Data Output bit 7 Channel B
14	PIN_D5	14	ADC_DB9	A/D Data Output bit 9 Channel B
15	PIN_B6	15	ADC_DB10	A/D Data Output bit 10 Channel B
16	PIN_A6	16	ADC_DB12	A/D Data Output bit 12 Channel B
17	PIN_B7	17	ADC_DB11	A/D Data Output bit 11 Channel B
18	PIN_D6	18	ADC_DB13	A/D Data Output bit 13 Channel B
19	PIN_A7	2	ADC_DB0	A/D Data Output bit 0 Channel B
20	PIN_C6	21	PLL_OUT_ADC1	PLL Clock input Channel B
21	PIN_C8	20	ADC_DA0	A/D Data Output bit 0 Channel A
22	PIN_E6	22	ADC_DA1	A/D Data Output bit 1 Channel A
23	PIN_E7	23	ADC_DA2	A/D Data Output bit 2 Channel A
24	PIN_D8	24	ADC_DA4	A/D Data Output bit 4 Channel A
25	PIN_E8	25	ADC_DA3	A/D Data Output bit 3 Channel A
26	PIN_F8	26	ADC_DA5	A/D Data Output bit 5 Channel A
27	PIN_F9	27	ADC_DA6	A/D Data Output bit 6 Channel A
28	PIN_E9	28	ADC_DA8	A/D Data Output bit 8 Channel A
29	VCC-SYS	29	VCC3	3.3V Power
30	GND	30	GND	Ground
31	PIN_C9	31	ADC_DA7	A/D Data Output bit 7 Channel A
32	PIN_D9	32	ADC_DA9	A/D Data Output bit 9 Channel A
33	PIN_E11	33	ADC_DA10	A/D Data Output bit 10 Channel A
34	PIN_E10	34	ADC_DA12	A/D Data Output bit 12 Channel A
35	PIN_C11	35	ADC_DA11	A/D Data Output bit 11 Channel A
36	PIN_B11	36	ADC_DA13	A/D Data Output bit 13 Channel A
37	PIN_A12	37	POWER_ON	Power-Down Function for Channel A & B
38	PIN_D11	38	ADC_OEB	A/D Output Enable Pin for Ch. B
39	PIN_D12	39	-	-
40	PIN_B12	40	ADC_OEA	A/D Output Enable Pin for Ch. A

Table 2.1: DE0-Nano board GPIO 0/THDB-ADA GPIO 0 pin association

DE0-Nano board GPIO 1		THDB-ADA GPIO 1 (J8)		
Pin no.	**FPGA pin no.**	**Pin no.**	**Schematic name**	**Description**
1	PIN_T9	1	SMA_DAC4	SMA D/A External Clock Input (J5)
2	PIN_F13	2	DAC_DA13	D/A Data bit 13 Channel A
3	PIN_R9	3	OSC_SMA_ADC4	SMA A/D External Clock Input (J5) or 100MHz Oscillator Clock Input
4	PIN_T15	4	DAC_DA12	D/A Data bit 12 Channel A
5	PIN_T14	5	DAC_DA11	D/A Data bit 11 Channel A
6	PIN_T13	6	DAC_DA9	D/A Data bit 9 Channel A
7	PIN_R13	7	DAC_DA10	D/A Data bit 10 Channel A
8	PIN_T12	8	DAC_DA8	D/A Data bit 8 Channel A
9	PIN_R12	9	DAC_DA7	D/A Data bit 7 Channel A
10	PIN_T11	10	DAC_DA5	D/A Data bit 5 Channel A
11	VCC-SYS	11	-	-
12	GND	12	GND	Ground
13	PIN_T10	13	DAC_DA6	D/A Data bit 6 Channel A
14	PIN_R11	14	DAC_DA4	D/A Data bit 4 Channel A
15	PIN_P11	15	DAC_DA3	D/A Data bit 3 Channel A
16	PIN_R10	16	DAC_DA1	D/A Data bit 1 Channel A
17	PIN_N12	17	DAC_DA2	D/A Data bit 2 Channel A
18	PIN_P9	18	DAC_DA0	D/A Data bit 0 Channel A
19	PIN_N9	25	DAC_DB11	D/A Data bit 11 Channel B
20	PIN_N11	20	DAC_WRTA	Input Write Signal Channel A
21	PIN_L16	22	DAC_DB13	D/A Data bit 13 Channel B
22	PIN_K16	21	PLL_OUT_DAC1	PLL Clock Input Channel B
23	PIN_R16	23	-	-
24	PIN_L15	24	DAC_DB12	D/A Data bit 12 Channel B
25	PIN_P15	19	PLL_OUT_DAC0	PLL Clock Input Channel A
26	PIN_P16	26	DAC_DB9	D/A Data bit 9 Channel B
27	PIN_R14	27	DAC_DB10	D/A Data bit 10 Channel B
28	PIN_N16	28	DAC_DB8	D/A Data bit 8 Channel B
29	VCC-SYS	29	VCC3	3.3V Power
30	GND	30	GND	Ground
31	PIN_N15	31	DAC_DB5	D/A Data bit 5 Channel B
32	PIN_P14	32	DAC_DB7	D/A Data bit 7 Channel B
33	PIN_L14	33	DAC_DB4	D/A Data bit 4 Channel B
34	PIN_N14	34	DAC_DB6	D/A Data bit 6 Channel B
35	PIN_M10	35	DAC_DB1	D/A Data bit 1 Channel B
36	PIN_L13	36	DAC_DB3	D/A Data bit 3 Channel B
37	PIN_J16	37	DAC_DB0	D/A Data bit 0 Channel B
38	PIN_K15	38	DAC_DB2	D/A Data bit 2 Channel B
39	PIN_J13	39	DAC_WRTB	Input Write Signal Channel B
40	PIN_J14	40	DAC_MODE	Mode Select. 1=dual port, 0=interleaved

Table 2.2: DE0-Nano board GPIO 1/THDB-ADA GPIO 1 pin association

2.3. Front-end

The front-end used in this project was developed on a previous project of the group of instrumentation and biomedical engineering of the electronic engineering department at the UPC.
It is based on an unipolar current source that converts the reference voltage signal coming from the FPGA into a current signal to measure the impedance. The front-end contains additional circuits to measure the differential voltages and the current supplied to the impedance, by means of a stage with a differential amplifier and buffers and a transimpedance amplifier, respectively.

The front-end features a reference signal input, an output with the impedance voltage measurement and an output with the measurement of the current injected to the impedance.
A ± 5V voltage and ground is needed for biasing integrated circuits of the board. The block-diagram and a frontal view of the front-end are shown in the pictures below:

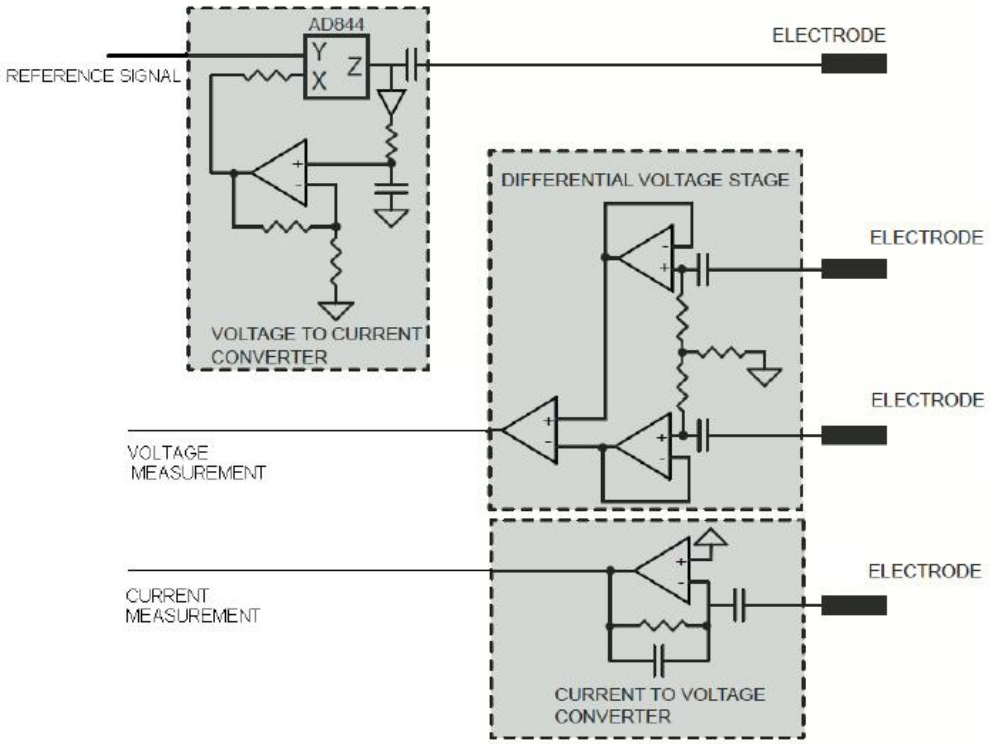


Figure 2.10: block diagram of the analog front-end used for the measurement of bioimpedance

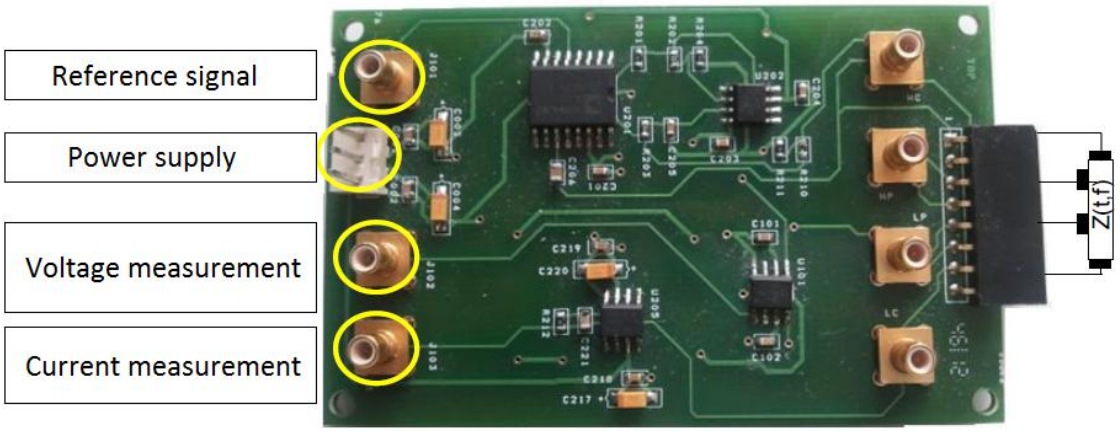


Figure 2.11: front view of the analog front-end used for the measurement of bioimpedance

For the minimal system that is being built, the current measurement output of the front-end will not be used. Given that the current source has an output impedance high enough (2MΩ//4pF), it can be assumed that the current through the load is constant.

2.4. Signal generator

A *Megafunction* provided by *Altera* is used to generate the sinusoidal signal that will be transmitted to the analog front-end. This *Megafunction* is a NCO (*Numerically controlled oscillator*). This module provides at its outputs sinus and cosine signals at a concrete frequency that is specified by the phase increment at one of the inputs of the module. Those signals are generated in two's complement and are used as local oscillators in the coherent demodulation.

With respect to the NCO configuration, it was set a magnitude precision of 14 bits (it fits the resolution of converters), an angular resolution of 16 bits and a phase accumulator precision of 32 bits. Besides, it is multiplier-based as the FPGA provides many of them and it is poor in memory resources.

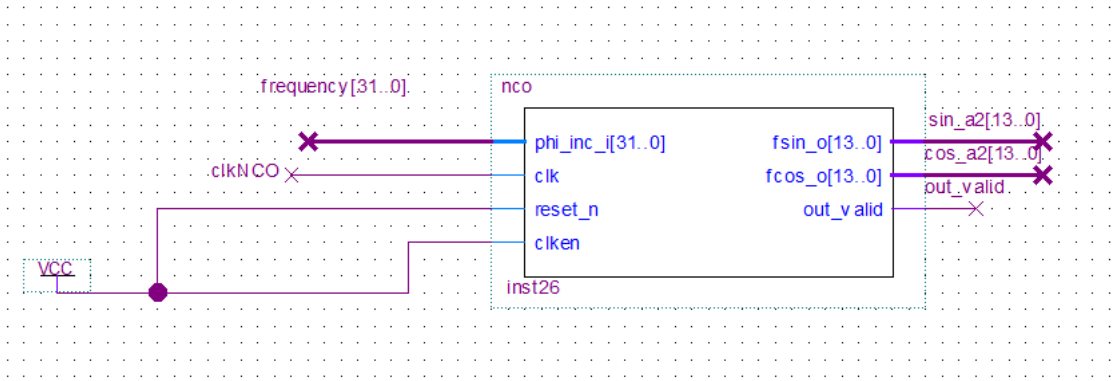


Figure 2.12: NCO used to generate sinusoidal signals

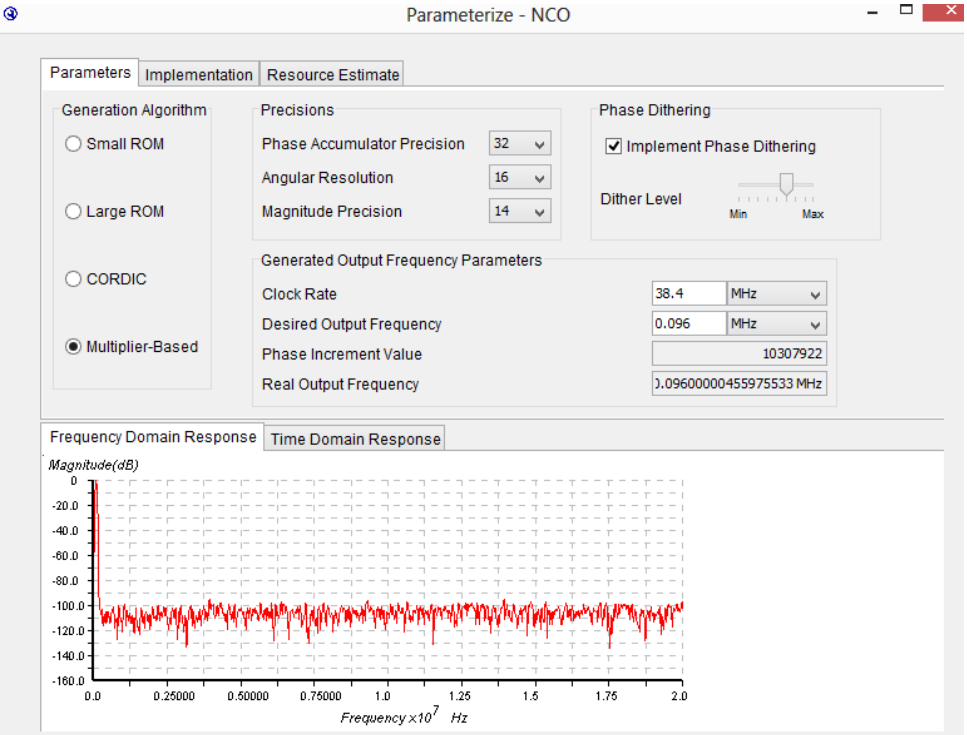


Figure 2.13: NCO configuration

2.4.1. Reference signal

In order to minimize the jitter effect, it is important that the reference signal of the NCO fulfill some features. This is that the periods of the frequencies synthesized by the NCO should have an integer number of cycles of the reference signal. To achieve it and according to the frequencies of measurement (defined in section 1.2), a reference frequency of 38.4 MHz was chosen.

2.4.2. Interfacing with the THDB-ADA

As mentioned before, the outputs of the NCO are in two's complement, whereas the DAC works in pure binary. For this reason the "a2tobin" module was developed. It converts two's complement into pure binary.

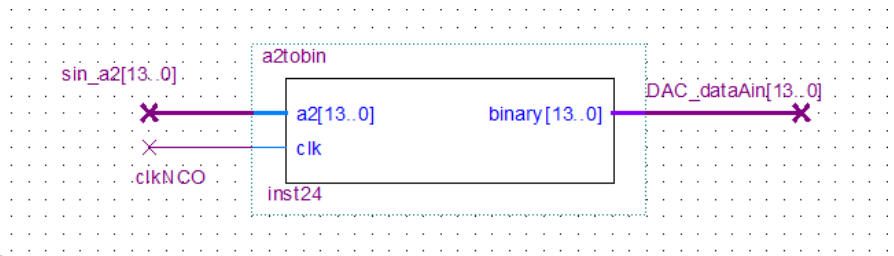


Figure 2.14: module to convert two's complement into pure binary

2.5. Coherent demodulation

In order to extract the information of the signal injected by the front-end, a coherent demodulation is performed, extracting in-phase and quadrature components (I and Q). The value of the impedance under measurement depends on time and frequency.

$$Z\,(t,f) \approx Z(t) \cdot \tilde{Z}(f) \tag{2-1}$$

Since for each measurement the system works at one specific frequency, it is considered that variations in time of the impedance are much slower than 1/f, being f the lowest frequency of measurement, so the impedance will only depend on time.

$$Z(t,f) \cong Z(t) \cdot \tilde{Z}(f) = Z^{'}(t)|_{f=f_0} = Re[Z^{'}(t)] + j\ Im[Z^{'}(t)] \tag{2-2}$$

The injected signal is a sinusoid, therefore the differential voltage in time after being amplified by a gain G will be:

$$V_d(t) = I_0 \cdot Z^{'}(t) \cdot G_A \cdot \sin(2\pi f_0 t) \tag{2-3}$$

$$V_d(t) = I_0 \cdot Re[Z'(t)] \cdot G_A \cdot \sin(2\pi f_0 t) + I_0 \cdot Im[Z'(t)] \cdot G_A \cdot \cos(2\pi f_0 t) \tag{2-4}$$

$$V_d(t) = I_0 \cdot |Z'(t)| \cdot G_A \cdot \sin(2\pi f_0 t + \varphi) \tag{2-5}$$

A digital version of homodyne demodulation has been implemented, where the signal to be demodulated is multiplied by a same-phase and same-frequency signal and by another one shifted 90°. It is done multiplying the differential voltage by a sinus and a cosine of the same frequency and incorporating a low pass filter, as shown in the figure below:

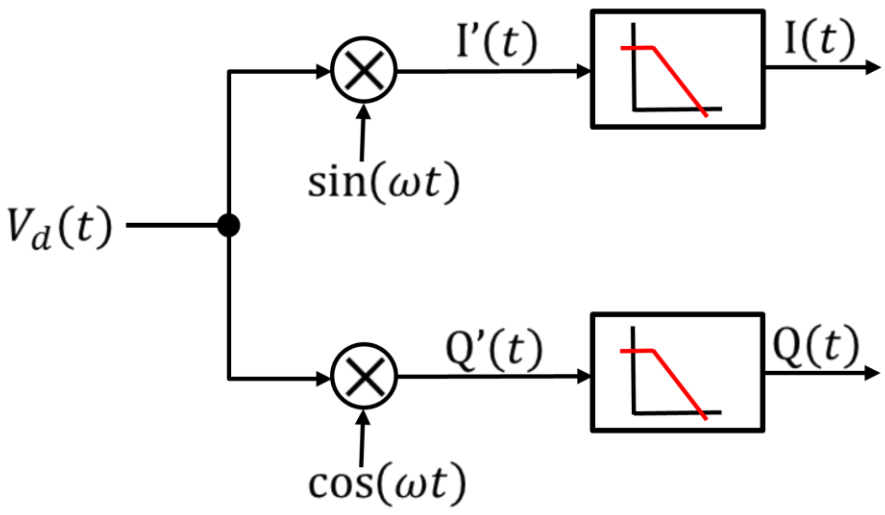


Figure 2.15: block diagram of demodulation

Once the differential voltage is multiplied, I'(t) and Q'(t) are obtained.

$$I'(t) = V_d(t) \cdot \sin(2\pi f_0 t) = I_0 \cdot |Z^{'}(t)| \cdot G_A \cdot \sin(2\pi f_0 t + \varphi) \cdot \sin(2\pi f_0 t) \tag{2-6}$$

$$I'(t) = I_0 \cdot |Z'(t)| \cdot G_A \cdot \frac{\cos(\varphi) - \cos(2 \cdot 2\pi f_0 t + \varphi)}{2} \quad (2\text{-}7)$$

$$Q'(t) = V_d(t) \cdot \cos(2\pi f_0 t) = I_0 \cdot |Z'(t)| \cdot G_A \cdot \sin(2\pi f_0 t + \varphi) \cdot \cos(2\pi f_0 t) \quad (2\text{-}8)$$

$$Q'(t) = I_0 \cdot |Z'(t)| \cdot G_A \cdot \frac{\sin(2 \cdot 2\pi f_0 t + \varphi) + \sin(\varphi)}{2} \quad (2\text{-}9)$$

Since the information is in the component at zero frequency, it is needed to implement a low pass filter. The output after filtering will be:

$$I(t) \cong \frac{I_0 \cdot |Z'(t)| \cdot G_A \cdot \cos(\varphi)}{2} \quad (2\text{-}10)$$

$$Q(t) \cong \frac{I_0 \cdot |Z'(t)| \cdot G_A \cdot \sin(\varphi)}{2} \quad (2\text{-}11)$$

The multiplication is performed using a digital multiplier implemented on the FPGA, whereas the low pass filters are implemented by accumulating samples and dividing them to obtain the average with a moving average filter. When implementing digitally the low pass filter, a response with the shape of a *sinc* function is obtained:

$$y(t) = \frac{1}{T} \int_t^{T+t} x(t)dt = x(t) * h(t) \quad (2\text{-}12)$$

$$h(t) = \prod\left(\frac{t}{T}\right) \quad (2\text{-}13)$$

$$h(f) = \frac{\sin(\pi \cdot f \cdot T)}{\pi \cdot f \cdot T} \quad (2\text{-}14)$$

Frequency response is shown in the next figure, where:

$$T = \frac{N}{f_s} \quad (2\text{-}15)$$

$$N = number\ of\ samples\ accumulated$$
$$f_s = sampling\ frequency$$

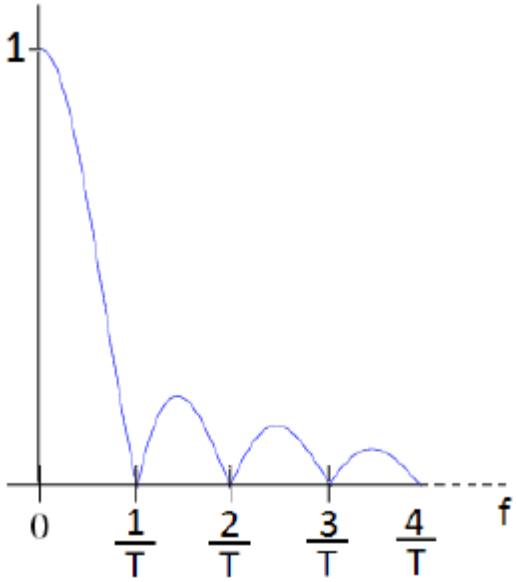


Figure 2.16: frequency response of digital low pass filter

The final structure to be implemented on the FPGA is shown in the figure below:

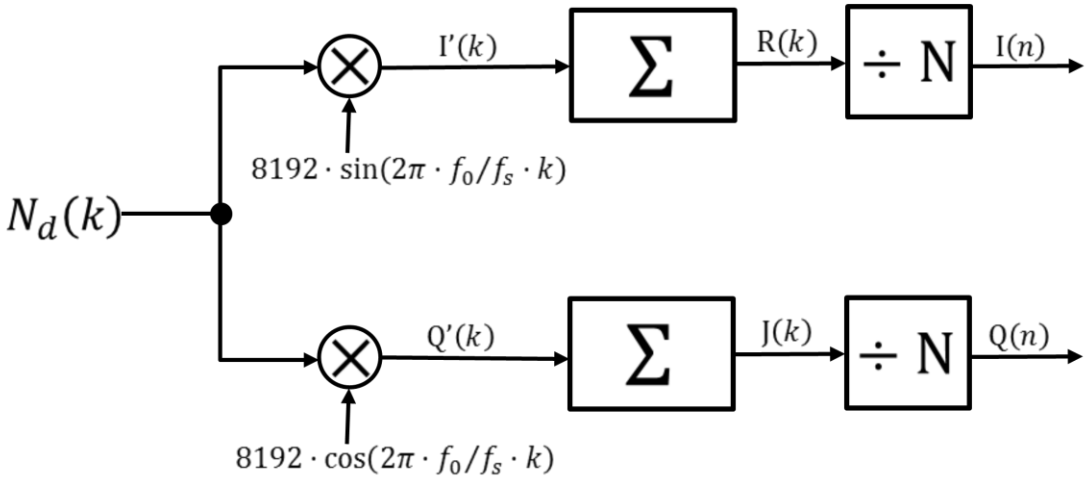


Figure 2.17: block diagram of the demodulation to be implemented on FPGA

Input signal Nd[k] depends on the reference voltage of the ADC and the differential voltage Vd(t):

$$N_d[k] = \left(\frac{V_d(t)}{V_{REF}} \cdot 2^{n-1}\right) = V_d(t) \cdot 8192 \qquad (2\text{-}16)$$

When multiplying this input by the cosine and sinus, I'[k] and Q'[k] are obtained:

$$I'[k] = \left(I_0 \cdot |Z'(k)| \cdot G_A \cdot \sin\left(2\pi \cdot {}^{f_0}/_{f_s} \cdot k + \varphi\right) \cdot 8192\right) \cdot 8192 \cdot \sin\left(2\pi \cdot {}^{f_0}/_{f_s} \cdot k\right) \qquad (2\text{-}17)$$

$$Q'[k] = \left(I_0 \cdot |Z'(k)| \cdot G_A \cdot \sin\left(2\pi \cdot {}^{f_0}/_{f_s} \cdot k + \varphi \right) \cdot 8192 \right) \cdot 8192 \cdot \cos\left(2\pi \cdot {}^{f_0}/_{f_s} \cdot k \right) \quad (2\text{-}18)$$

After accumulating N samples, R[n] and J[n] are obtained:

$$R[n] = \sum_{k=n\cdot N}^{k=(n+1)\cdot N-1} I_0 \cdot |Z'(k)| \cdot G_A \cdot \frac{8192^2}{2} \cdot \left(\cos(\varphi) - cos(2 \cdot 2\pi \cdot {}^{f_0}/_{f_s} \cdot k + \varphi) \right) \quad (2\text{-}19)$$

$$J[n] = \sum_{k=n\cdot N}^{k=(n+1)\cdot N-1} I_0 \cdot |Z'(k)| \cdot G_A \cdot \frac{8192^2}{2} \cdot \left(\sin(\varphi) + sin(2 \cdot 2\pi \cdot {}^{f_0}/_{f_s} \cdot k + \varphi) \right) \quad (2\text{-}20)$$

The demodulation is finished when dividing accumulated samples by the number of samples:

$$I[n] = \sum_{k=n\cdot N}^{k=(n+1)\cdot N-1} I_0 \cdot |Z'(k)| \cdot G_A \cdot \frac{8192^2}{2\cdot N} \cdot \left(\cos(\varphi) - cos(2 \cdot 2\pi \cdot {}^{f_0}/_{f_s} \cdot k + \varphi) \right) \quad (2\text{-}21)$$

$$Q[n] = \sum_{k=n\cdot N}^{k=(n+1)\cdot N-1} I_0 \cdot |Z'(k)| \cdot G_A \cdot \frac{8192^2}{2\cdot N} \cdot \left(\sin(\varphi) + sin(2 \cdot 2\pi \cdot {}^{f_0}/_{f_s} \cdot k + \varphi) \right) \quad (2\text{-}22)$$

2.5.1. Implementation of the coherent demodulation on the FPGA

The previously explained scheme of demodulation was implemented on the FPGA as shown in figure 2.18:

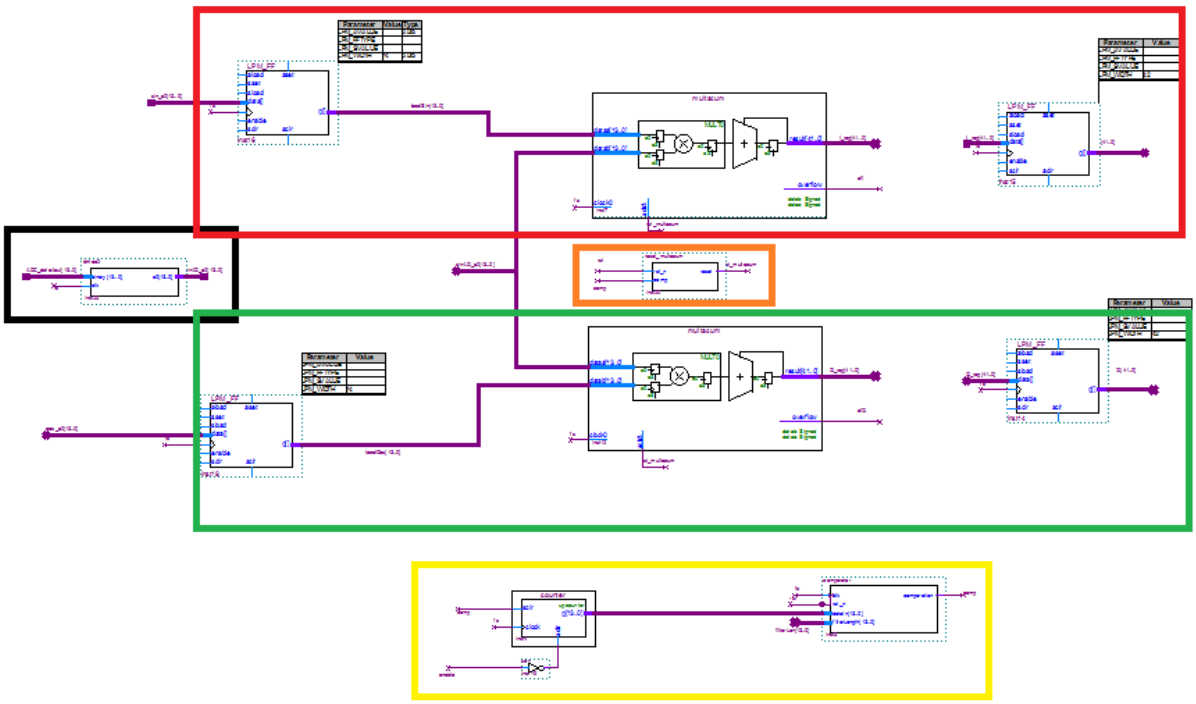


Figure 2.18: coherent demodulation implemented on FPGA

Five main blocks are differentiated in the scheme. The black one contains the "bintoa2" module, whose work is converting the pure binary incoming signal into a two's complement signal. The second one, in red, corresponds to the branch of the demodulator that obtains the I component, whereas the green one is the branch corresponding to the Q component. In both cases, the sampling frequency is determined by *fs*. The procedure is the one explained in section 2.5. The yellow block sets the length of the moving average filter, i.e., each time the "multacum" module reaches the desired number of samples, it generates a pulse (*comp*), resetting these modules and indicating to other blocks that a measurement has been performed and it is available. Finally, the orange block resets "multacum" modules when either the "reset" signal or the "comp" signal are activated.

2.6. Communication

As mentioned in section 2.2.1, DE0-Nano board only offers one communication port, the GPIO's. Therefore, the communication system shown below was designed:

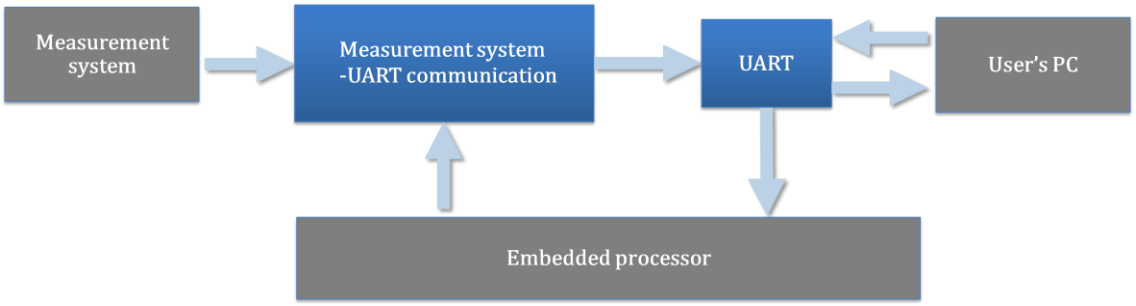


Figure 2.19: scheme of communication

"Measurement system-UART communication" and "UART" modules were implemented within the FPGA resources.

Given the expected data throughput is not high, an asynchronous serial port, able to be connected to an RS232-to-USB converter, was chosen.

2.6.1. UART

A simple UART was designed and implemented, showing the next features:

- 8 data bits
- 1 stop bit
- No parity
- Configurable baud rate (it is configured to 128Kbps)

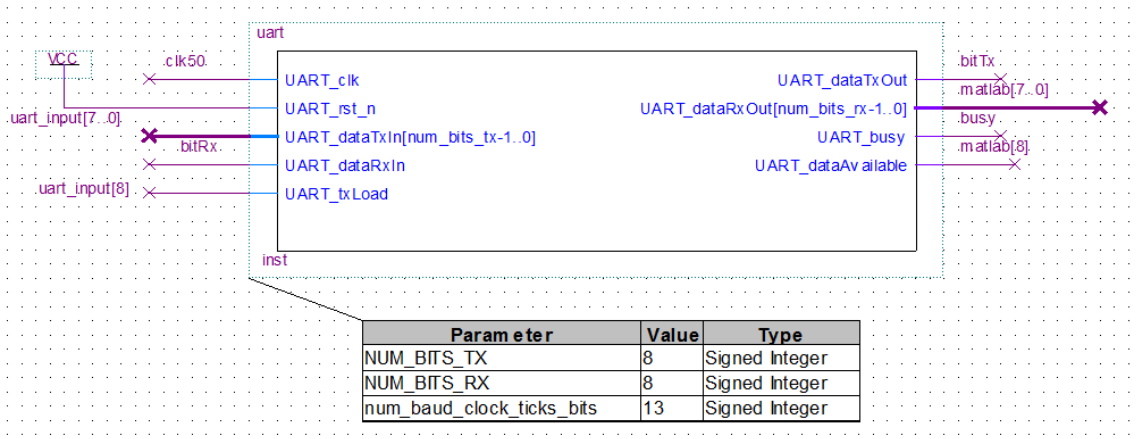


Figure 2.20: UART module on Quartus II

Two different blocks can distinguish inside the UART, one for the transmission and another one for the reception:

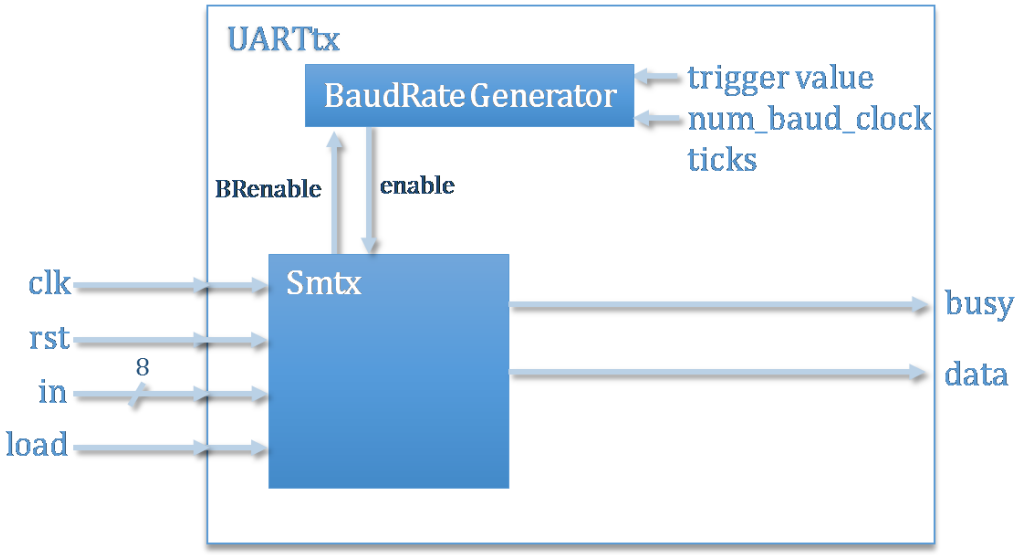


Figure 2.21: scheme of the UART transmission block

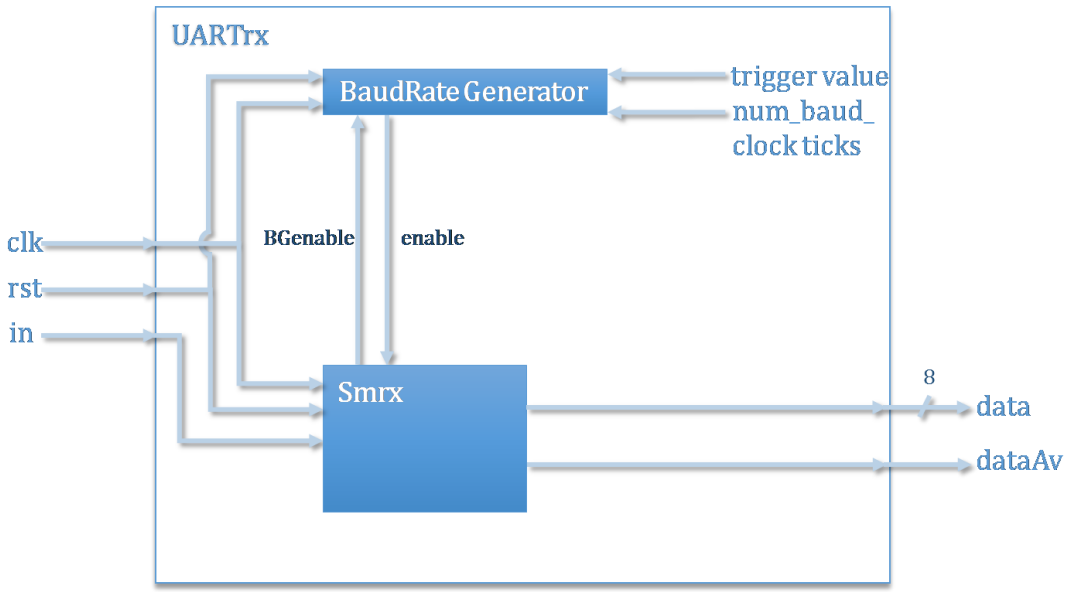


Figure 2.22: scheme of the UART reception block

These blocks include three different modules. *BaudRateGenerator* is a kind of timer whose purpose is the generation of clock ticks of period *Tb*, where *Tb* is the bit slot time and corresponds with the inverse of the data rate. The clock frequency of this module is 50 MHz and the data rate is 128 Kbps, so it basically has to generate a pulse every 50 MHz/128 Kbps cycles, i.e., every 390 clock cycles, having a relative error of

1% (3% relative error is accepted). It also allows the configuration of a trigger level. The other two modules are the state machines for reception and transmission.

State machines of SmRx and SmTx are shown below, as well as pin assignation of the GPIO of DE0-Nano board.

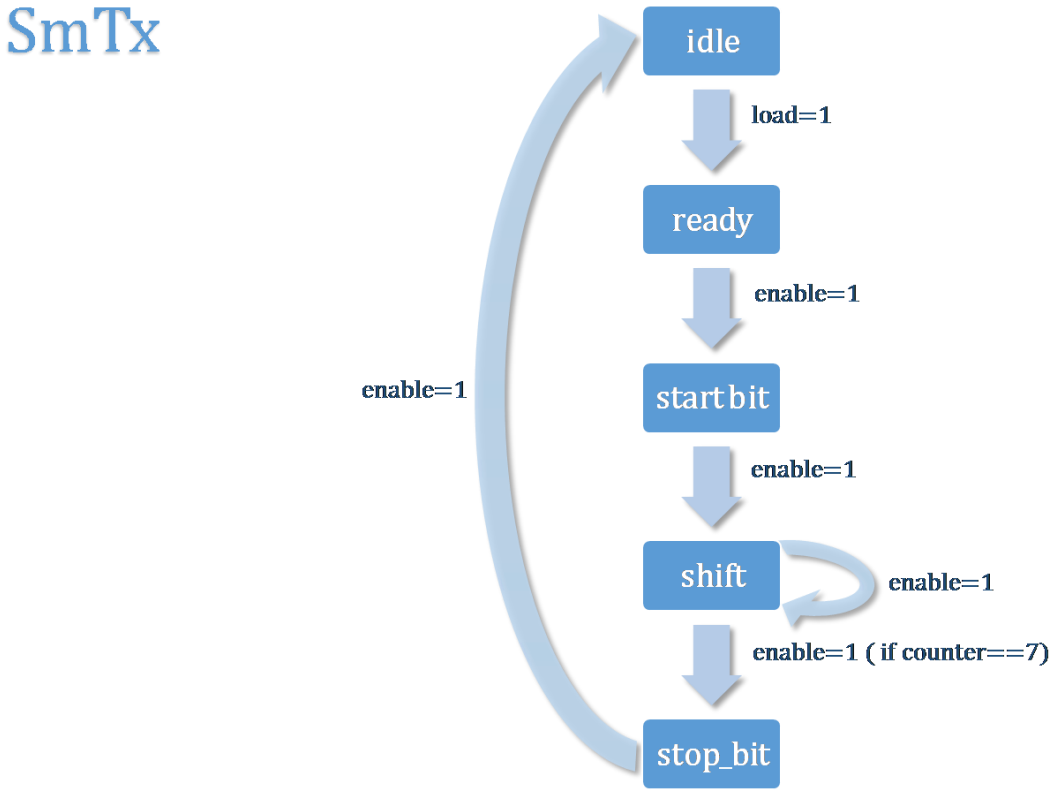


Figure 2.23: state machine for transmission

When load signal is asserted, the SmTx module enables the *BaudRateGenerator*, registers the byte to be transmitted,asserts the output signal *Busy* and updates its state to *Ready*. On *Ready* state, when a tick coming from the *BaudRateGenerator* is detected, it changes to *Start bit* state, where the start bit ('0') is transmitted. When a tick is detected again, the start bit has already been transmitted and the SmTx updates its state to *Shift*. On this state, the LSB of the byte is transmitted until the arrival of a new tick. Then a right shift operation is performed and the new LSB transmitted until the next tick, and so on. Once all bits are transmitted, SmTx changes to *Stop bit* state, where the stop bit ('1') is transmitted. With the arrival of a new tick it updates again to *Idle* state, disabling the *BaudRateGenerator* and negating the output signal *Busy*.

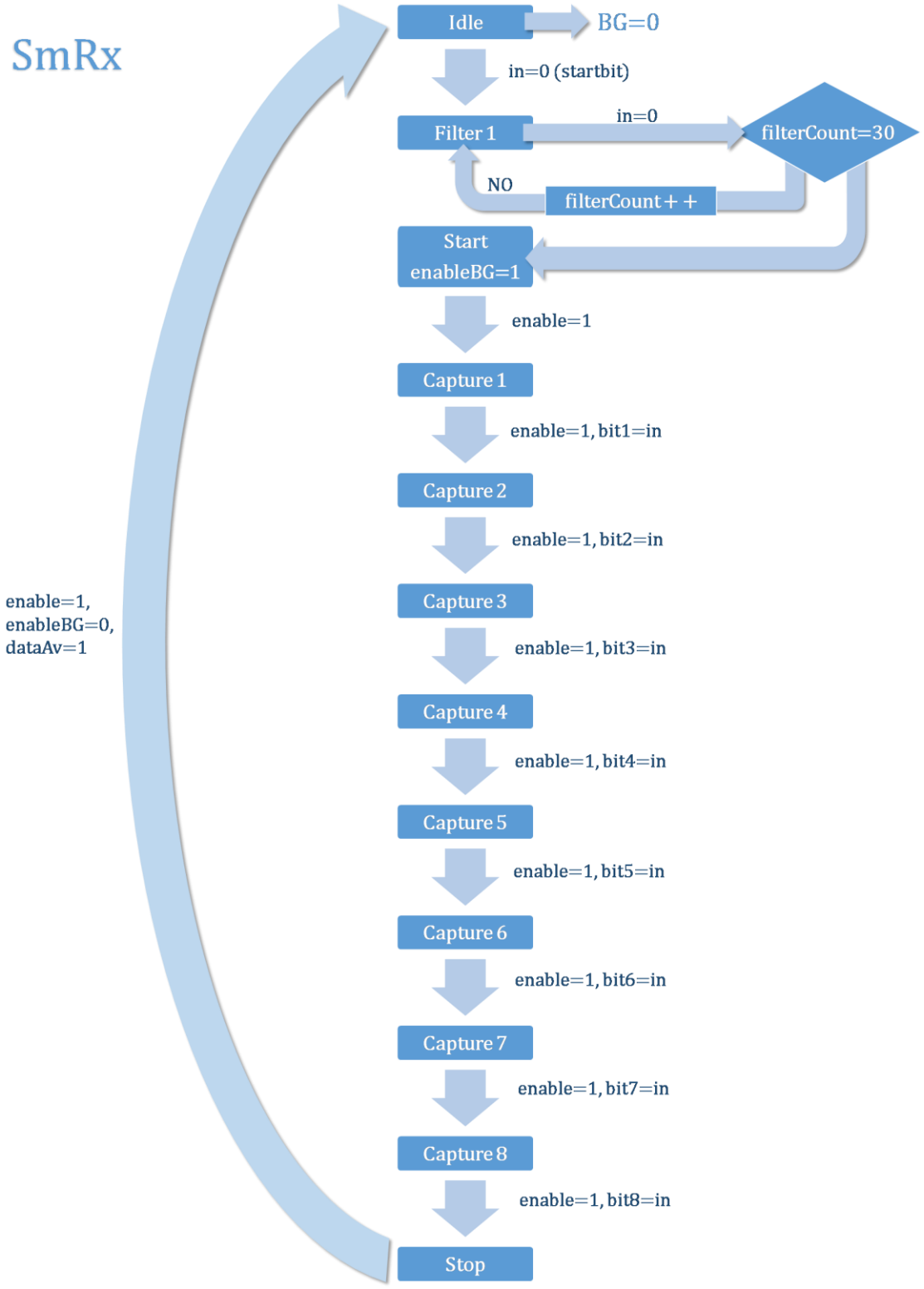


Figure 2.24: state machine for reception

When the transmitter is sending no information bytes, the incoming signal is '1' . So the detection of '0' means that a new byte is about to be received. When it happens, the SmRx updates to state *Filter*. On this state, if 30 consecutive samples of the input bit are '0', the state is updated to *Start*, otherwise it comes back to *Idle*. It is important to remember that this module is sampling at 50 MHz whereas the bit rate is 128 Kbps, so

detecting one '0' with this oversampling does not necessarily means that a complete '0' bit is being received, it could be due to the noise in the transmission. Therefore the *Filter* state is needed. Once on *Start* state, the *BaudRateGenerator* is enabled (with a trigger value of one third of the number of baud clock ticks) and when a tick is received, the state changes to *Capture 1*. On *Capture X* states, input bit is registered in the X position of the byte when a tick is detected and the state is updated to *Capture X+1* until all 8 bits are captured. Then changes to *Stop* state, where '1' is received and comes back to *Idle* state with the arrival of a new tick, disabling the *BaudRateGenerator* and outputting the received byte and *dataAv* =1.

Input bit (Rx)	Pin no. 16, PIN_G16 (2x13 header)
Output bit (Tx)	Pin no. 14, PIN_F16 (2x13 header)

Table 2.3: pin assignation for the UART

2.6.2. Measurement system-UART communication

Three modules were designed for the communication between the measurement system and the UART, as shown below:

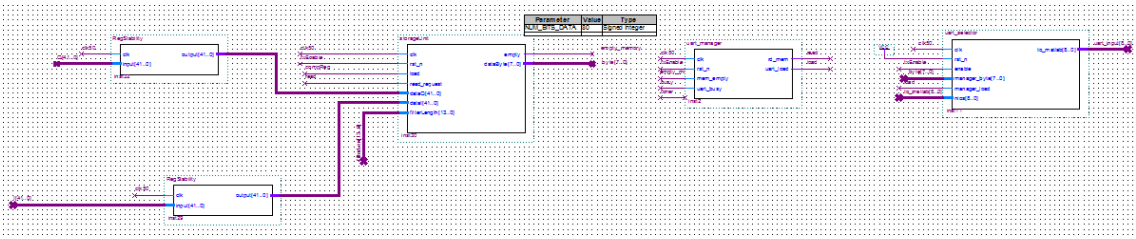


Figure 2.25: modules for the communication between the measurement system and the UART

Let's start explaining the *uart_manager* module. The function of this module is managing the information of measurements stored on *storageUnit* for its transmission to the UART. The way to do it is indicating to *storageUnit* to transmit a byte of information to the UART providing that it is not busy and there are still bytes of information stored on *storageUnit*. The state machine of this module is shown below:

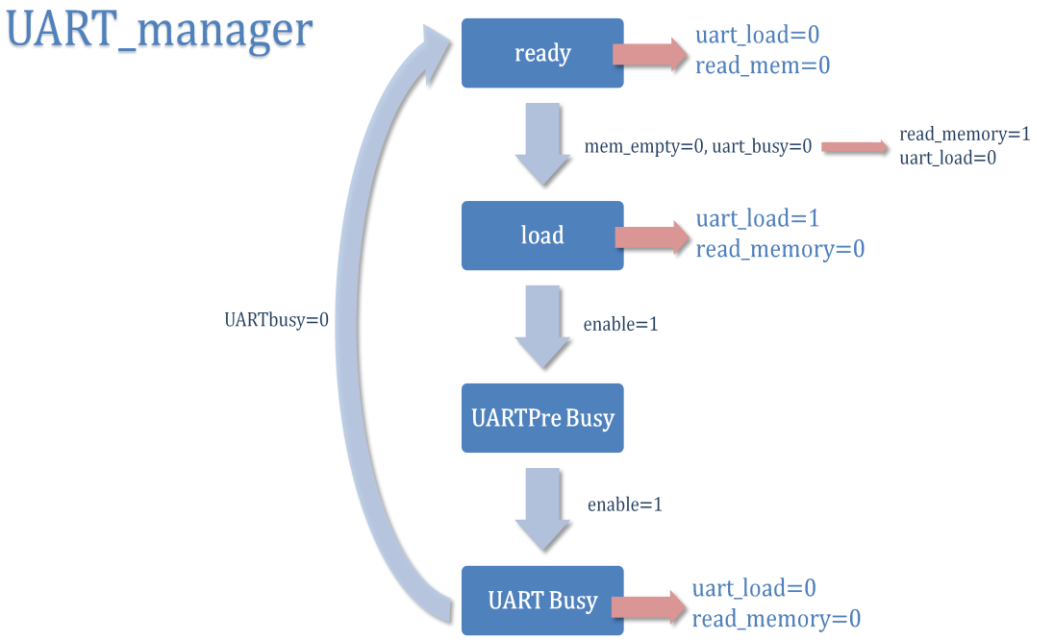


Figure 2.26: UART_manager state machine

StorageUnit module stores the data of measurements obtained each time that comp signal indicates there is new data available. It also performs the average, dividing the value coming from the accumulator by the value contained in length.

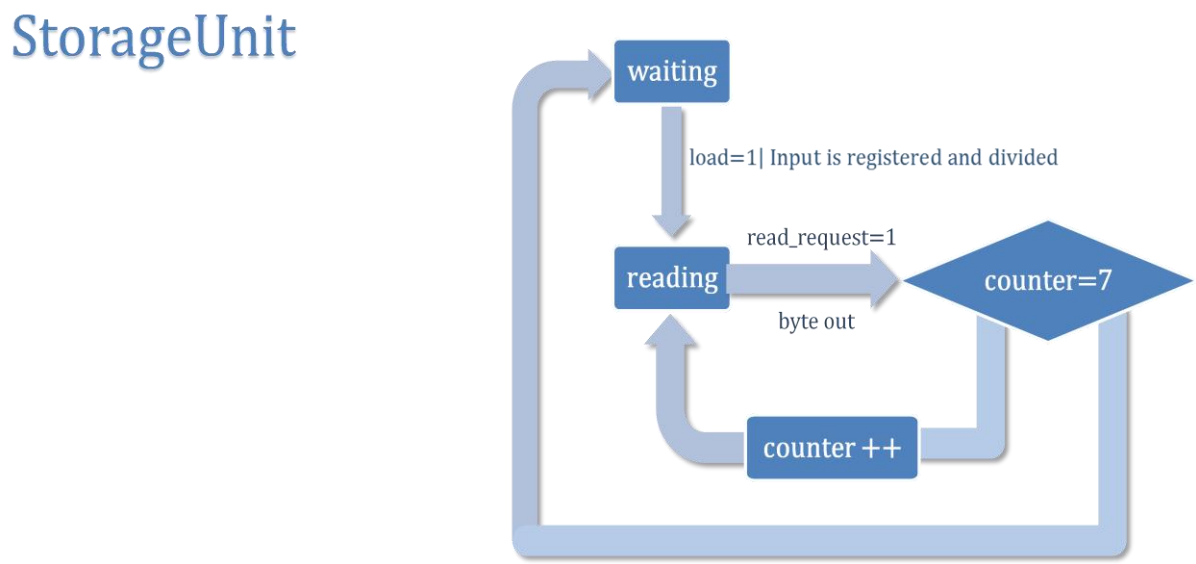


Figure 2.27: storageUnit state machine

This module exhibits a peculiarity. As can be observed, the clock frequency of the modules communicated with the UART is 50 MHz, whereas the measurement system works at the frequency determined by *fs*. Two different circuits working with different clock domains (different frequency or phase) are asynchronous between them. To solve it, *RegStability* modules were incorporated to eliminate metastabilities.

The last module is *UARTselector*. There is only one UART available to send information to user's PC, so that it is needed some kind of mechanism to give the control of the medium either to the measurement system or to the NIOS II. This is the function of this module.

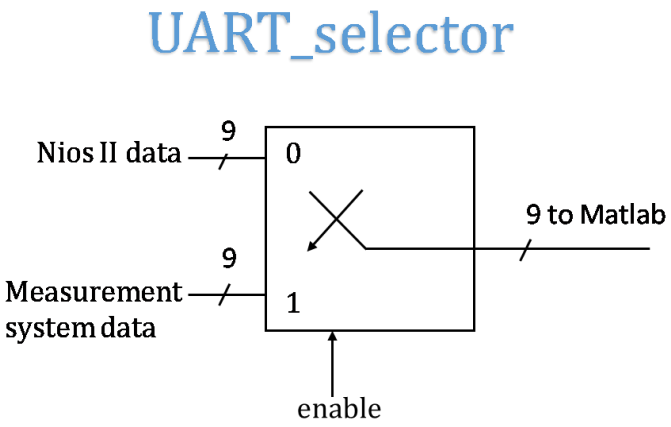


Figure 2.28: UARTselector scheme

2.6.3. UART-PC communication

The last step is the communication between the UART and the user's PC. GPIO pin assignation was provided in section 2.6.1. However, in order to successfully transmit data it is needed to convert TTL voltage levels (3.3 V) into RS232 levels. To carry it out, it was used the Maxim Integrated MAX3227ECAE+ circuit [11], that guarantees a data rate of 1Mbps for a minimum supply voltage of 3 V.

This integrated circuit was mounted following the specifications of the technical data sheet.

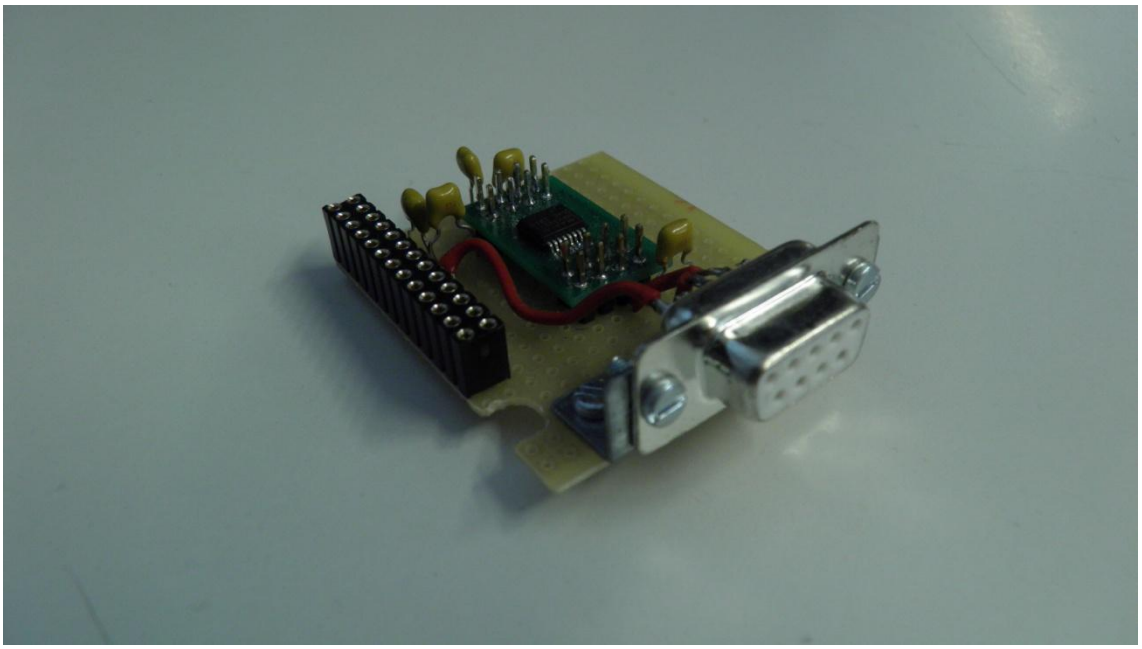


Figure 2.29: TTL-RS232 matching board

Finally, the UC232A USB-to-serial converter from Aten [12] was used to connect the RS232 interface with the USB interface of the PC. This converter features a data transfer rate up to 230 Kbps.

Figure 2.30: UC232A USB-to-serial converter

2.6.4. Limitations on the communication

The RS232 to USB converter features a maximum data transfer rate of 230 Kbps. The standard data transfer rate under this value is 192 Kbps, so that the communication system has 416.6 us to transmit a sample of measurement at one determined frequency. This is the minimum time that should be required to compute the next sample, otherwise the current sample would start to be transmitted before the previous one was transmitted completely. So this parameter establish the first limitation on the relation between the length of the filter and the sampling frequency, that in the end refers to the relation between the length of the filter and frequency of measurement.

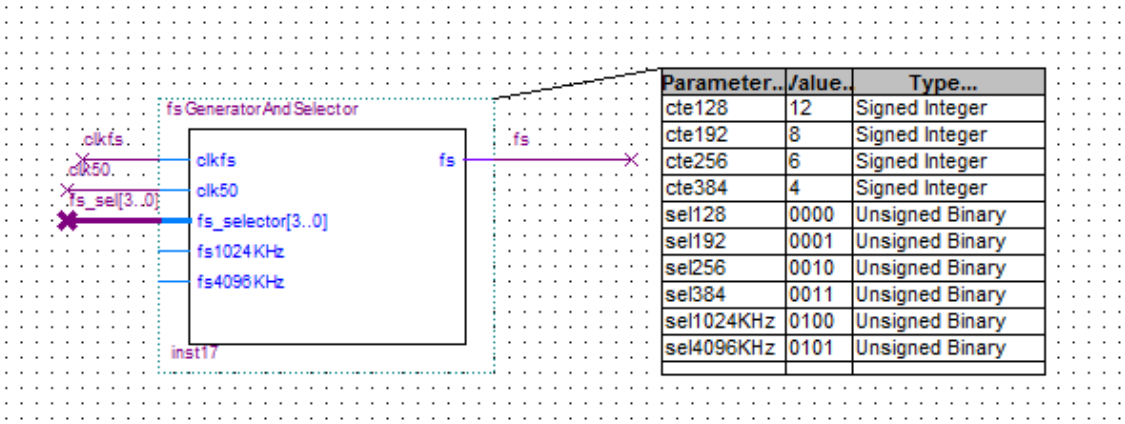
Supposing a filter length of 1024 samples, the maximum sampling frequency would be of 2.457 MHz. It means a measurement frequency of 614.49 kHz with a 4 times sampling.

Therefore, this converter (which is the bottleneck of the communication system) should be replaced with another one of higher performance to increase this limit, so that the new bottleneck would be the MAX3227ECAE+.

2.7. Generation of sampling clocks

Different sampling frequencies are needed for the measurements at each of the frequencies specified in section 1.2. The number of different clock frequencies implemented using PLLs is limited, so they had to be generated using a higher clock frequency (1.536 MHz) applied to several counters. This task is carried out by the

fsGeneratorAndSelector module that also selects the proper sampling frequency given to the measurement block by means of the *fs_sel* signal.



Parameter..	Value..	Type...
cte128	12	Signed Integer
cte192	8	Signed Integer
cte256	6	Signed Integer
cte384	4	Signed Integer
sel128	0000	Unsigned Binary
sel192	0001	Unsigned Binary
sel256	0010	Unsigned Binary
sel384	0011	Unsigned Binary
sel1024KHz	0100	Unsigned Binary
sel4096KHz	0101	Unsigned Binary

Figure 2.31: fsGeneratorAndSelector module

2.8. Embedded processor

Altera's Nios II processor is the most widely used processor in the FPGA industry, since provides a great flexibility and supports RTOS. It is an Altera's 32 bit proprietary architecture, whose modules are interfaced through the Avalon bus. These modules can be from interval timers to RAM controllers.

To implement the Nios II processor, Quartus provides a tool called Qsys, wich allows us to define and customize all modules of the embedded system.

The modules that comprise our system are listed below:

- Nios II processor
- On-Chip Memory
- JTAG UART
- Interval timer
- Parallel I/O

2.8.1. Nios II processor

Designer can choose among three options when selecting the processor, depending on the hardware resources he wants to use. Obviously, the more resources used, the processor provides higher performance. The three options are: economy core, standard core and fast core. The economy core uses only 600 logic elements and two M9Ks. This is the simplest core, whereas standard and fast cores are absolutely deterministic, jitter free real-time performance with unique hardware real-time features. Economy core would be enough for this application, but the fast core was used since the FPGA provides enough resources.

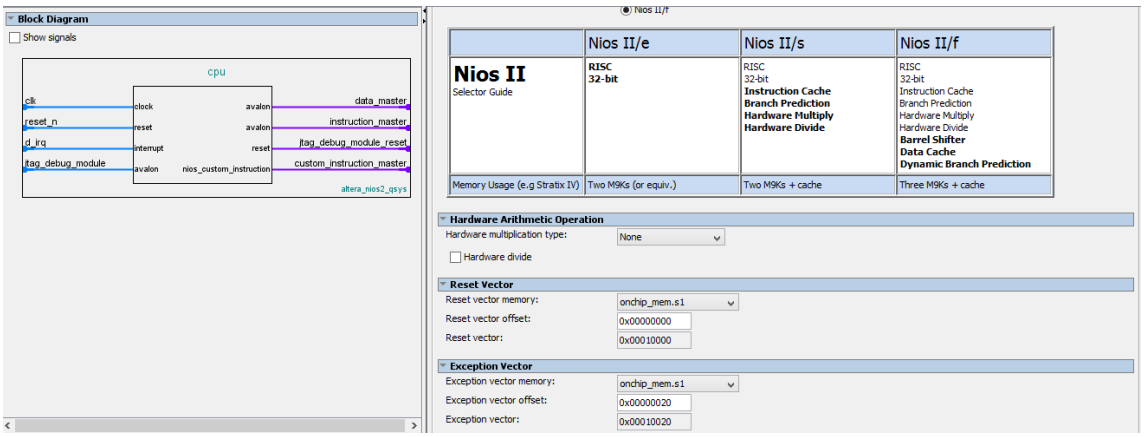


Figure 2.32: Nios II core configuration

This processor has a RISC architecture of 32 bits, using between 1400 and 1800 logic elements. Besides, it works at 50 MHz, which is enough for this application (in fact the frequency of work is tunable, but it was not needed to change it). Both the reset vector and the exception vector are located on the On-Chip memory described below.

2.8.2. On-Chip Memory

On-Chip memory with data width of 32 bits and a total size of 20480 bytes was implemented. This is the RAM used by the processor and where the program is stored.

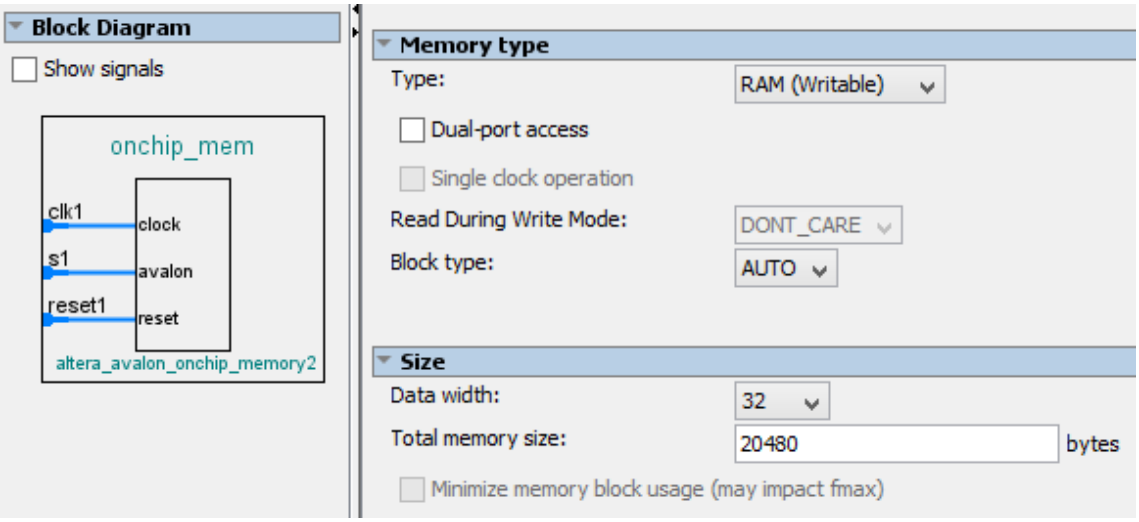


Figure 2.33: On-chip memory configuration

2.8.3.JTAG UART

This is the controller needed to program the processor form the user's PC.

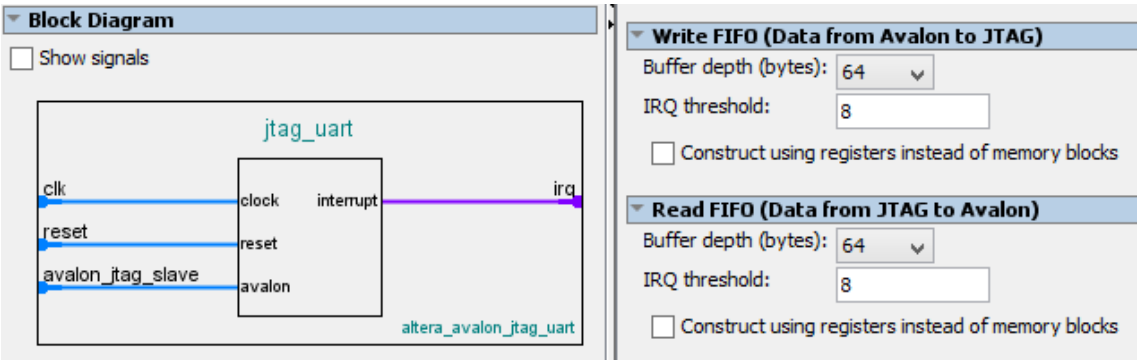


Figure 2.34: JTAG UART configuration

2.8.4. Interval timer

This module produces an interruption every 30 ms. The reason of use of this module is obtaining a periodicity in the reception of measured data. One measurement at all frequencies is obtained in roughly 28 ms but, for obtaining a reference time base, the next measurement is performed 30 ms after the previous one, so that it is sure that a new measurement is obtained at each frequency every 30 ms.

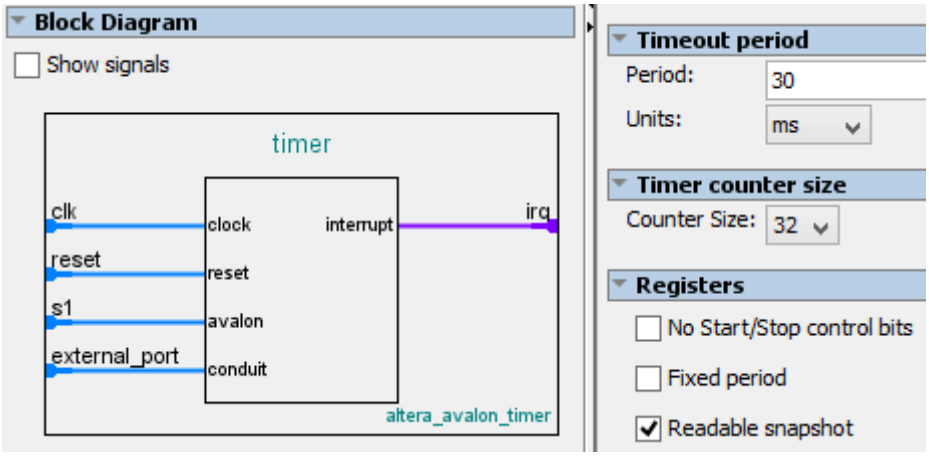


Figure 2.35: interval timer configuration

2.8.5. Parallel I/O

These input/output ports are used by the user through the embedded software to control and transmit information to other blocks. The designer can configure the length (1 to 32 bits) and the direction of each PIO core (output, input, bidirectional). Output PIO cores only drive output and input PIO cores are configured with synchronously capture on rising edge and interrupt generation by edge, i.e., bit *n* is set to one whenever an edge is detected on input port *n* and if bit *n* on *interruptmask* register is enabled, an interruption is generated.

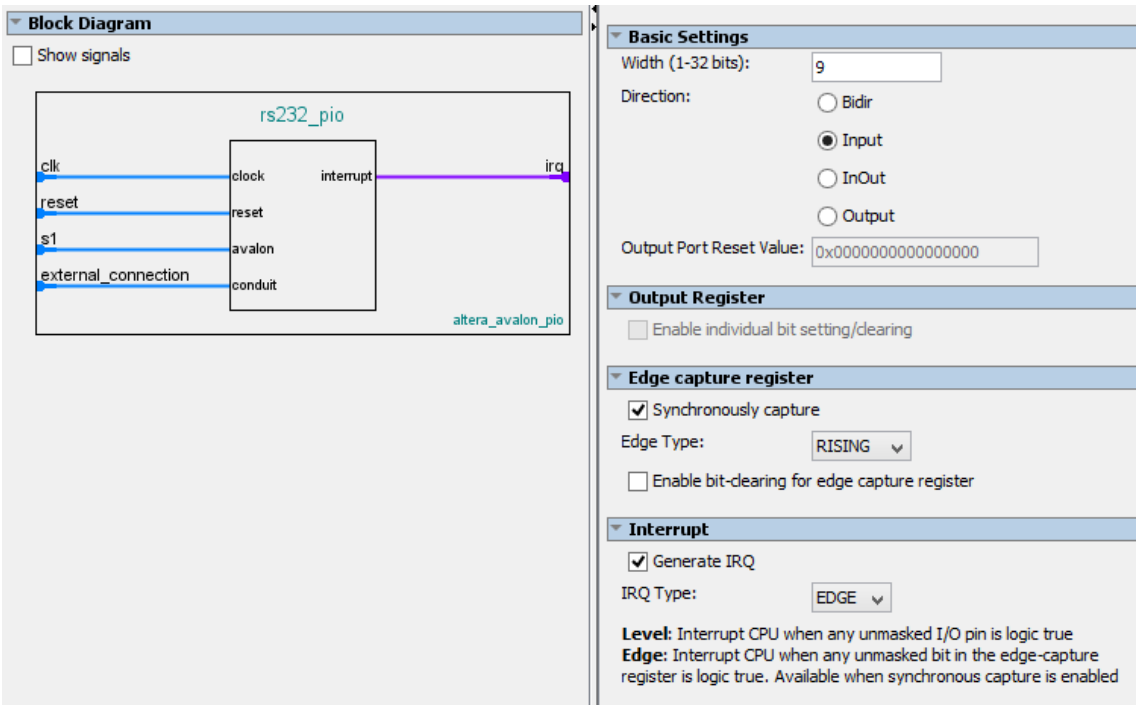


Figure 2.36: configuration of one of the input PIO cores

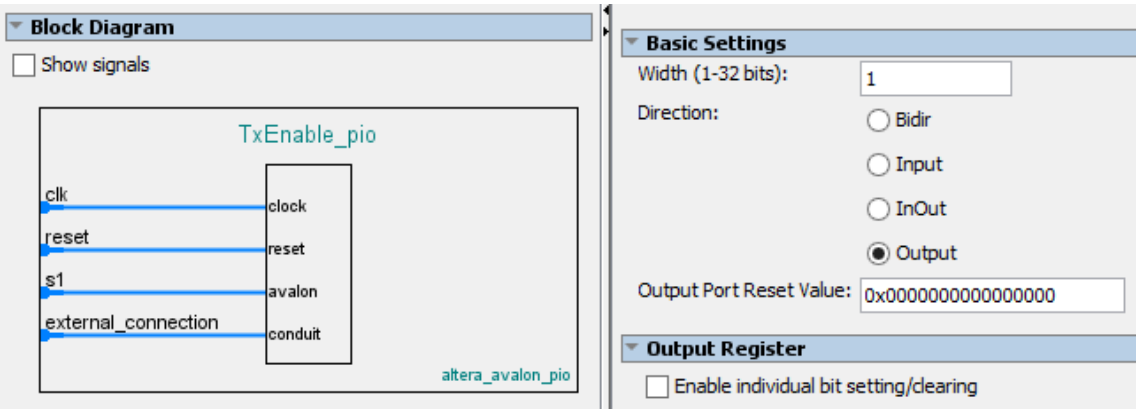


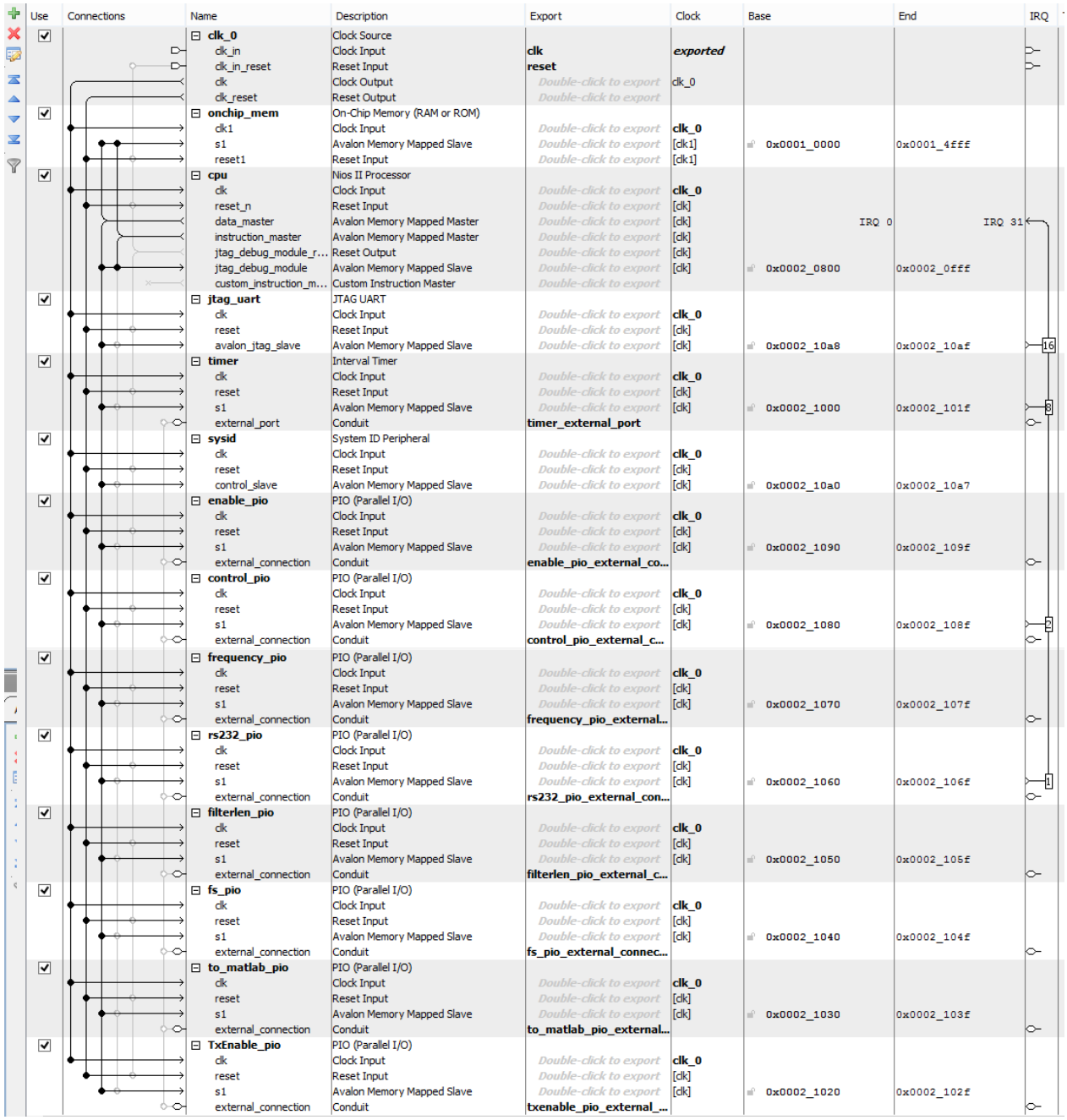
Figure 2.37: configuration of one of the output PIO cores

Next table lists used PIOs and their functions:

PIO name	# bits	direction	Function	Connected to
enable_pio	1	Output	Enables the measurement block to star or to stop measuring	*Counter, resetController*
control_pio	1	Input	Generates an interruption when a sample of measurement is obtained	*RegComp*
frequency_pio	32	Output	Gives the phase increment to the NCO	*NCO*
rs232_pio	9	Input	Incoming byte from UART	*UART*
filterlen_pio	14	Output	Sets the length of the filter	*Comparator, storageUnit*
fs_pio	4	Output	Sets the sampling frequency	*fsGeneratorAndSelector*
to_matlab_pio	9	Output	Byte to be transmitted to MATLAB	*Uart_selector*
txenable_pio	1	Output	Assign UART resources to measurement block	*StorageUnit, uart_manager*

Table 2.38: PIO cores

2.8.6. Global scheme of the embedded system



Use	Connections	Name	Description	Export	Clock	Base	End	IRQ
☑		**clk_0**	Clock Source					
		clk_in	Clock Input	**clk**	***exported***			
		clk_in_reset	Reset Input	**reset**				
		clk	Clock Output	*Double-click to export*	clk_0			
		clk_reset	Reset Output	*Double-click to export*				
☑		**onchip_mem**	On-Chip Memory (RAM or ROM)					
		clk1	Clock Input	*Double-click to export*	**clk_0**			
		s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk1]	0x0001_0000	0x0001_4fff	
		reset1	Reset Input	*Double-click to export*	[clk1]			
☑		**cpu**	Nios II Processor					
		clk	Clock Input	*Double-click to export*	**clk_0**			
		reset_n	Reset Input	*Double-click to export*	[clk]			
		data_master	Avalon Memory Mapped Master	*Double-click to export*	[clk]	IRQ 0	IRQ 31	
		instruction_master	Avalon Memory Mapped Master	*Double-click to export*	[clk]			
		jtag_debug_module_r...	Reset Output	*Double-click to export*	[clk]			
		jtag_debug_module	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0002_0800	0x0002_0fff	
		custom_instruction_m...	Custom Instruction Master	*Double-click to export*				
☑		**jtag_uart**	JTAG UART					
		clk	Clock Input	*Double-click to export*	**clk_0**			
		reset	Reset Input	*Double-click to export*	[clk]			
		avalon_jtag_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0002_10a8	0x0002_10af	16
☑		**timer**	Interval Timer					
		clk	Clock Input	*Double-click to export*	**clk_0**			
		reset	Reset Input	*Double-click to export*	[clk]			
		s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0002_1000	0x0002_101f	8
		external_port	Conduit	**timer_external_port**				
☑		**sysid**	System ID Peripheral					
		clk	Clock Input	*Double-click to export*	**clk_0**			
		reset	Reset Input	*Double-click to export*	[clk]			
		control_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0002_10a0	0x0002_10a7	
☑		**enable_pio**	PIO (Parallel I/O)					
		clk	Clock Input	*Double-click to export*	**clk_0**			
		reset	Reset Input	*Double-click to export*	[clk]			
		s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0002_1090	0x0002_109f	
		external_connection	Conduit	**enable_pio_external_co...**				
☑		**control_pio**	PIO (Parallel I/O)					
		clk	Clock Input	*Double-click to export*	**clk_0**			
		reset	Reset Input	*Double-click to export*	[clk]			
		s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0002_1080	0x0002_108f	2
		external_connection	Conduit	**control_pio_external_c...**				
☑		**frequency_pio**	PIO (Parallel I/O)					
		clk	Clock Input	*Double-click to export*	**clk_0**			
		reset	Reset Input	*Double-click to export*	[clk]			
		s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0002_1070	0x0002_107f	
		external_connection	Conduit	**frequency_pio_external...**				
☑		**rs232_pio**	PIO (Parallel I/O)					
		clk	Clock Input	*Double-click to export*	**clk_0**			
		reset	Reset Input	*Double-click to export*	[clk]			
		s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0002_1060	0x0002_106f	1
		external_connection	Conduit	**rs232_pio_external_con...**				
☑		**filterlen_pio**	PIO (Parallel I/O)					
		clk	Clock Input	*Double-click to export*	**clk_0**			
		reset	Reset Input	*Double-click to export*	[clk]			
		s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0002_1050	0x0002_105f	
		external_connection	Conduit	**filterlen_pio_external_c...**				
☑		**fs_pio**	PIO (Parallel I/O)					
		clk	Clock Input	*Double-click to export*	**clk_0**			
		reset	Reset Input	*Double-click to export*	[clk]			
		s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0002_1040	0x0002_104f	
		external_connection	Conduit	**fs_pio_external_connec...**				
☑		**to_matlab_pio**	PIO (Parallel I/O)					
		clk	Clock Input	*Double-click to export*	**clk_0**			
		reset	Reset Input	*Double-click to export*	[clk]			
		s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0002_1030	0x0002_103f	
		external_connection	Conduit	**to_matlab_pio_external...**				
☑		**TxEnable_pio**	PIO (Parallel I/O)					
		clk	Clock Input	*Double-click to export*	**clk_0**			
		reset	Reset Input	*Double-click to export*	[clk]			
		s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0002_1020	0x0002_102f	
		external_connection	Conduit	**txenable_pio_external_...**				

Figure 2.39: global scheme of embedded system, including interconnections, memory address location and IRQ priority.

3. Embedded software

Software used on Nios II embedded processor is explained in this section. One of the cool things of embedded processors is the flexibility they provide to our systems just replacing the embedded software. Although the system delivered in this project performs measurements at determined frequencies and with static parameters, the user only has to change the software code to obtain different measurements. In fact, it was designed another software which allows us to set the parameters for every measurement (it will be explained in this section).

Nios II processor is programmed with Eclipse-based Nios II software build tools for Eclipse. First step is specifying the *.sopcinfo* to Eclipse. This file contains information about memory assignment of control and data registers of peripherals, clock frequencies, etc, and it is automatically generated by *Qsys* (the tool used for the design of the embedded system). Once it is done, Eclipse generates C files with high level instructions and libraries that give the programmer an easy control of the processor and its peripherals.

The software designed on this project will be referred to as *MultifrequencyFRA* and it is based on *hello_world_small* template, that occupies the smallest memory footprint possible for a hello world application.

Next scheme shows the communication protocol between Nios II embedded software and user's PC:

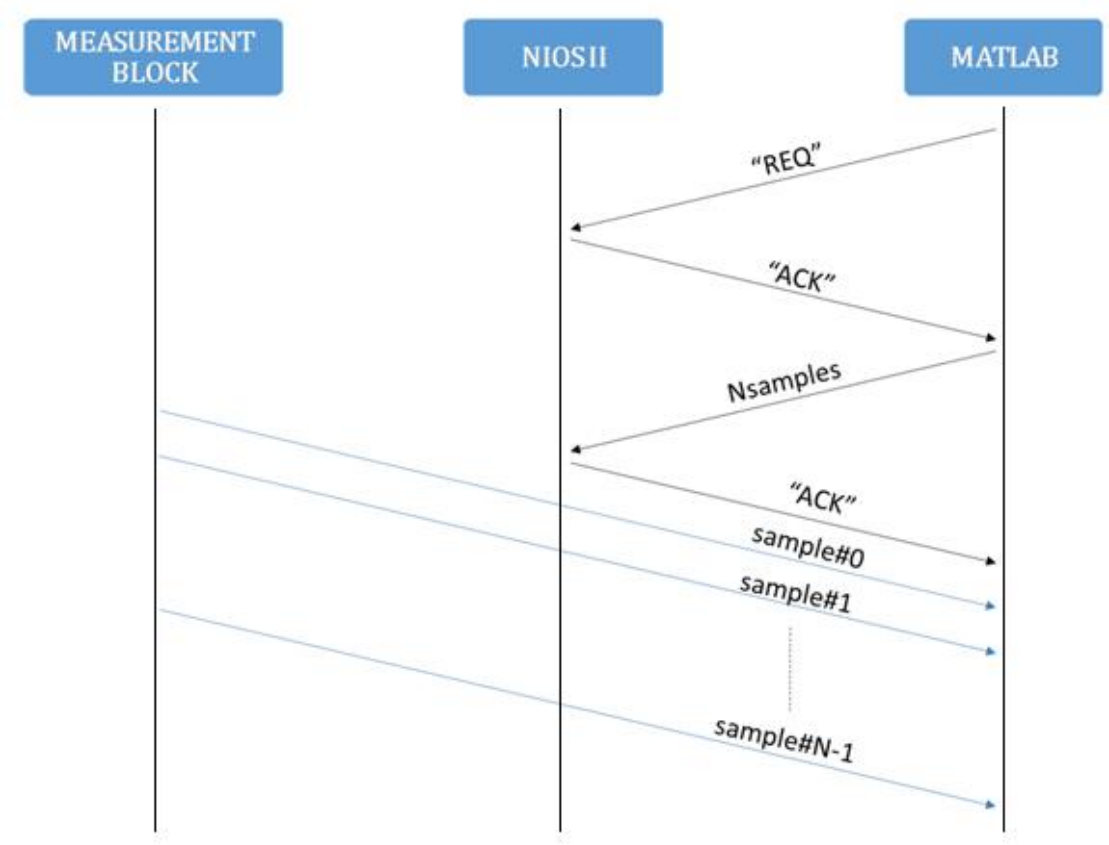


Figure 3.1: communication between MATLAB, on user's PC, and the measurement system

Software running on Nios II processor acts like a server. When the user wants a measurement to be done, he sends a request command. Then it is acknowledged by NIOS II and the user specify the number of samples of the measurement. Once this parameter is received by NIOS II, it notifies the reception sending an ACK and the measurement begins. If during this process there were problems, NIOS II would send a NACK to user's PC.

Request command "REQ" sent from MATLAB corresponds with decimal value 1, whereas "ACK" and "NACK" decimal values are 1 and 0 respectively.

The communication protocol could be as difficult as desired, but this easy one is enough.

Flow diagram of embedded software (appendix 9.3.1) is shown in figure 3.2 in the next page:

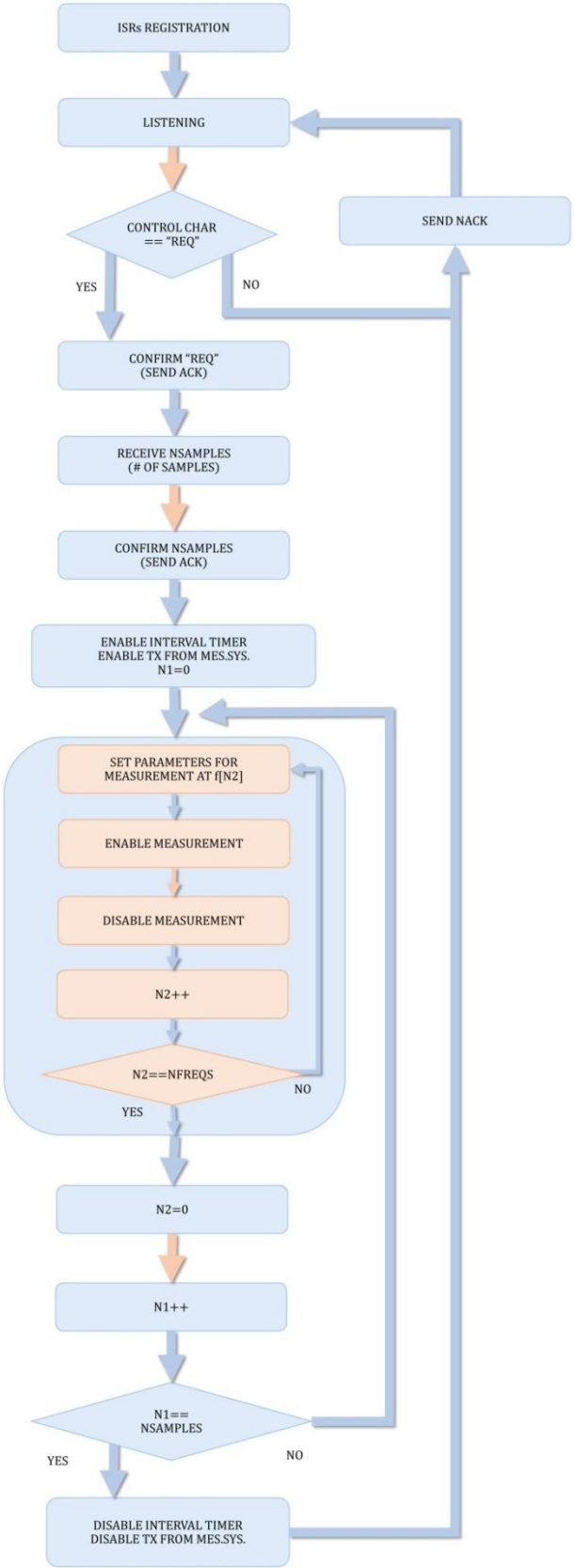


Figure 3.2: diagram flow of embedded software *MultifrequencyFRA*

When embedded software starts, the first thing to do is registering *interruption service routines,* i.e., assign to each interruption the corresponding routine to be executed when it is detected on the interrupt vector. If there are no problems on this process, the message "*ISRs Registered successfully!!!\n*" is printed on Nios Console. Then, a loop begins. First step consists in listening for a request and it is done through the own designed function *getInt2()* which is shown in appendix 9.3.1. This function receives incoming bytes from RS232 PIO and returns the integer corresponding to four consecutive bytes. It is only needed one byte to send the "REQ" message from MATLAB, however we always work with integers (32 bits) since this function also was designed for the *MonofrequencyFRA* software that will be explained later. Once the request command is received, it will be acknowledged sending an "ACK". In case the parameter received is different from "REQ", a "NACK" will be send and NIOS II will come back to the listening state. After the acknowledgement, the system waits for the reception of the number of samples to measure (using again *getnt2()*) and after receiving this parameter it is confirmed sending an "ACK".

Measurements really start from this point. N samples will be measured at each one of the five frequencies mentioned in section 1.2. Firstly, *tx_enable* is asserted. This indicates to the *uart_selector* that the input from the measurement system should be taken. *Interval timer* is also activated. Then the iterative process of measurement begins. Parameters for the first frequency of measurement are set and *enable* is asserted, indicating to the measurement system to start to measure and keeping waiting NIOS II for an assertion in *compReg* PIO. Once the measurement system gets the result and notifies it to NIOS II through *compReg*, *enable* is deactivated and a new iteration is performed for the next frequency of measurement. After having obtained the measurements at the five different frequencies, NIOS II waits for an interruption of the interval timer which means that 30 ms have passed and a new measurement at all the five frequencies should be done, coming back to the previously explained process. It goes on until N measurements are obtained, when NIOS II disables the interval timer, deactivates *tx_enable* and returns to the listening state, waiting for a request.

While developing the software, a large number of functions in the hardware abstraction layer (HAL) application program interface (API) were used. Details on this topic are not given in this document since they are explained in the HAL API reference.

The final size of the program is less than 10 KB, which are download into On-chip memory

To start the program the user just has to run it on Eclipse.

As mentioned before, it was designed another software that allows the user to specify the parameters of the measurement. Flow diagram of this program is shown below:

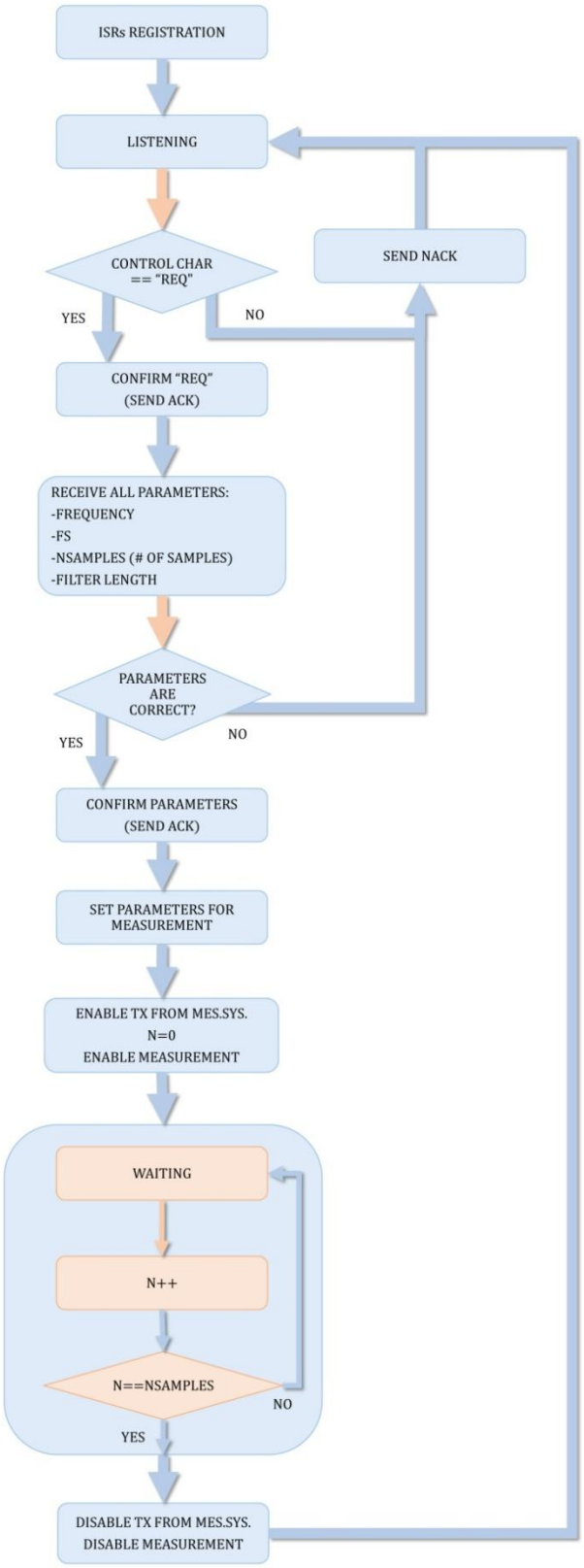


Figure 3.3: flow diagram of embedded software *MonofrequencyFRA*

This is not the software used in the measurements so we will not explain more about it. Besides, the code is given in appendix 9.3.2.

4. Data acquisition using MATLAB

MATLAB is a powerful mathematical software tool that provides mechanisms not only to manipulate, process and represent data but also to acquire it.

The main goal of the project is to build a digital core for a EIS acquisition system. Then, the main generator and demodulator labours are performed by the FPGA. MATLAB is only used to collect measurements and perform an statistical and graphical analysis.

In figure 3.1 of section 3, it was depicted the communication protocol between MATLAB and the measurement system, including the NIOS II processor. For the communication to work properly, the configuration of the serial port used by MATLAB should be the same than the one implemented in the UART (section 2.6.1). This configuration as well as the rest of the code is shown in appendix 9.4.

Flow diagram of MATLAB code is shown in the figure 4.1:

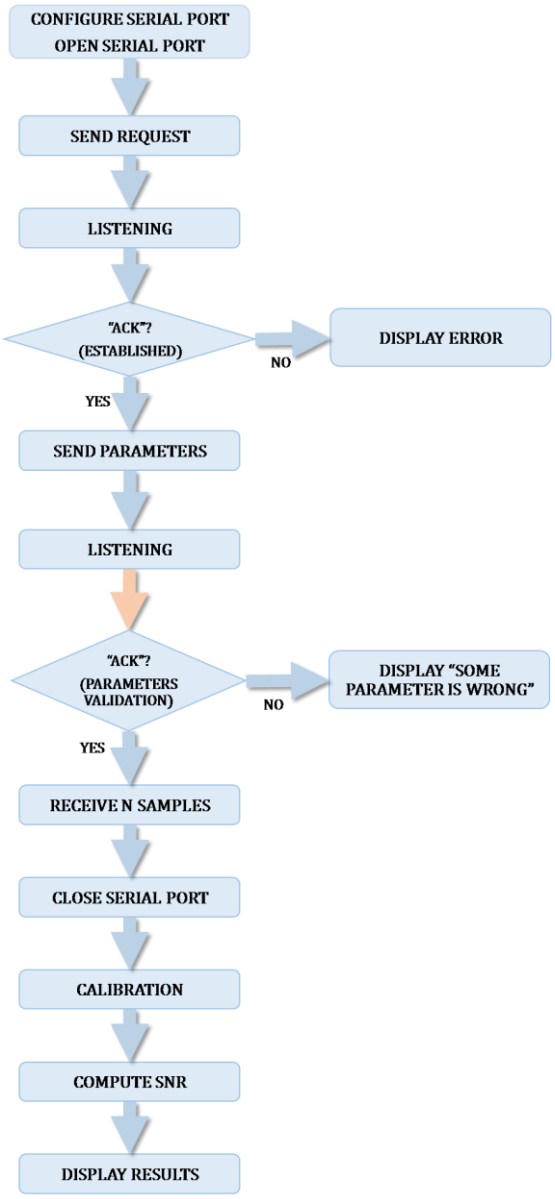


Figure 4.1: flow diagram of MATLAB software (appendix 9.4.1)

After configuring and open the serial port to be used for the communication, a request is sent to the measurement system. Once it is acknowledged, the number of samples to be measured is transmitted to the system. This number can be specified directly at the beginning of the script or the user can specified the amount of time he wants to measure (e.g., 60 seconds) and the number of samples is calculated from it. After this parameter is acknowledged, MATLAB receives the N samples from the measurements system and closes the serial port. Now the data is ready to be processed. The results are calibrated using weights stored in text files that were previously computed using the function *calibration* (appendix 9.4.2). This function obtains calibration weights for the magnitude and phase from a reference impedance and save them into text files.

Finally, calibrated results are plotted on different figures for each frequency. SNR computation is only performed when measuring reference impedances.

More software was developed on MATLAB during this project. It is given in appendix 9.4.

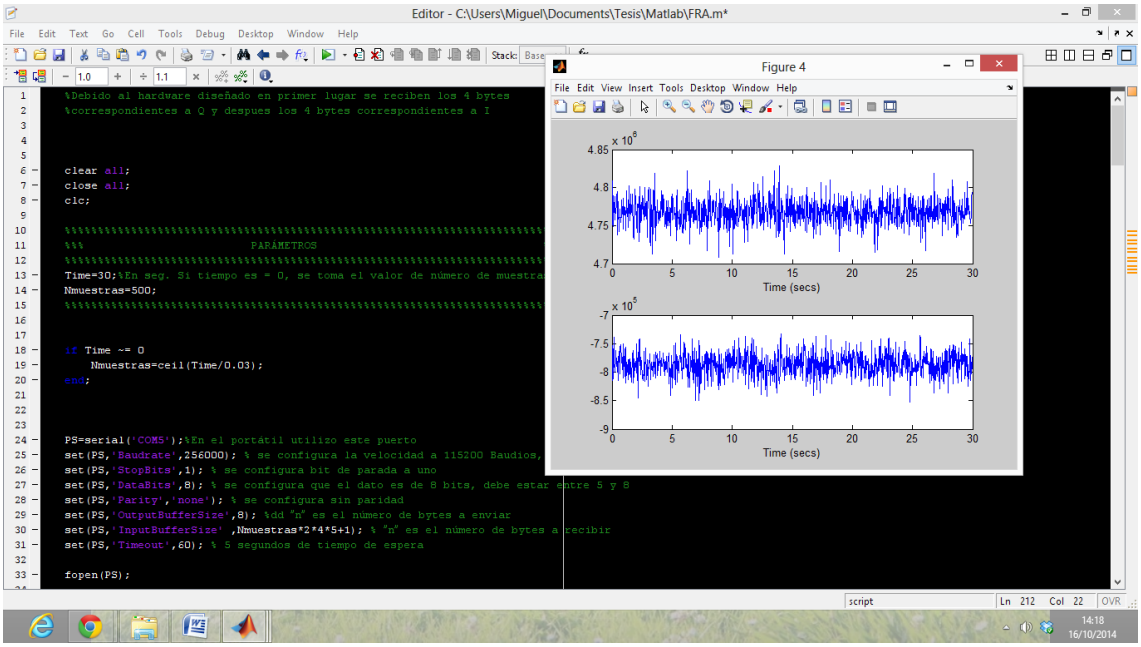


Figure 4.2: screenshot while running the application on MATLAB

5. Measurements and results

5.1. Characterization measurements

Before measuring on the human body, many measurements were performed to characterize the system.

At first, resistors between 10 Ω and 1KΩ were measured during 15 seconds, i.e., 500 samples were measured and averaged for each resistor. Their real values were measured using the multimeter Fluk 185 True RMS available in the lab.

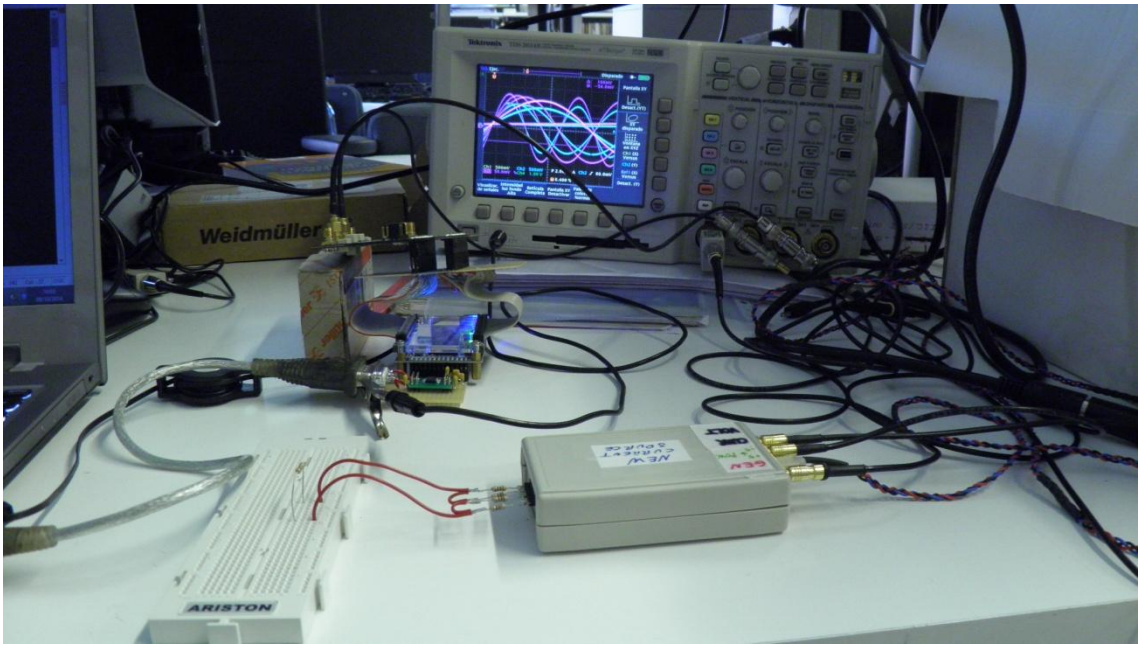


Figure 5.1: the system performing a measurement

The D/A output and D/A input signals were monitored with a Tektronix TDS 3024B Oscilloscope to detect possible signal saturation (e.g., when an electrode is not making a good contact).

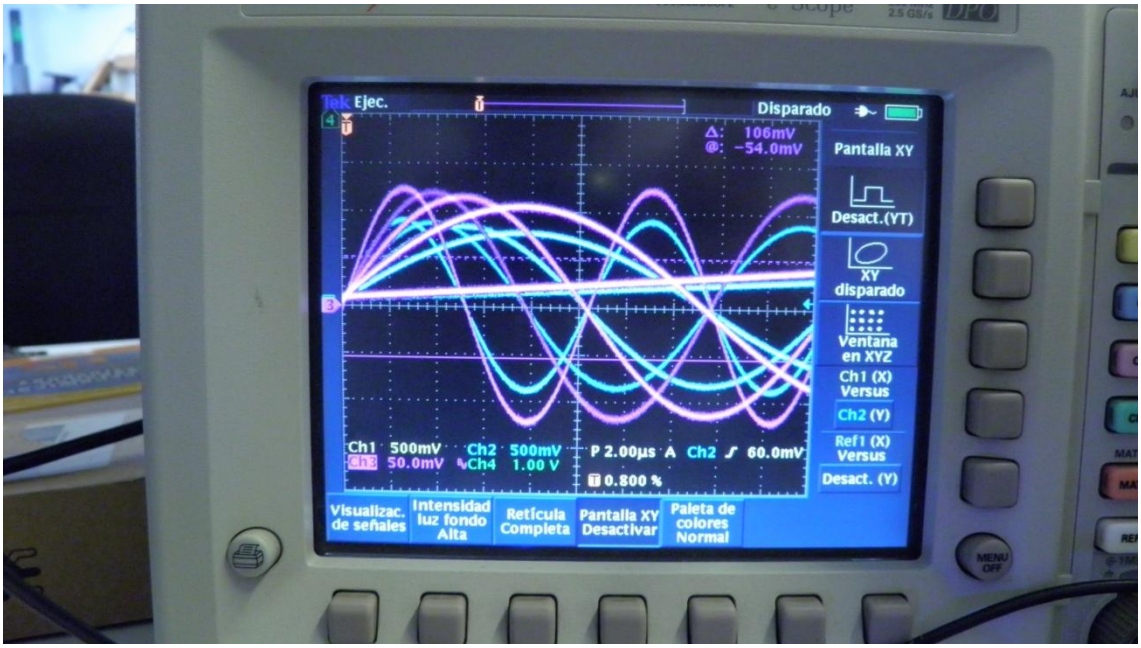


Figure 5.2: D/A output and A/D input signals on oscilloscope

The measurement results are shown in the next graphics:

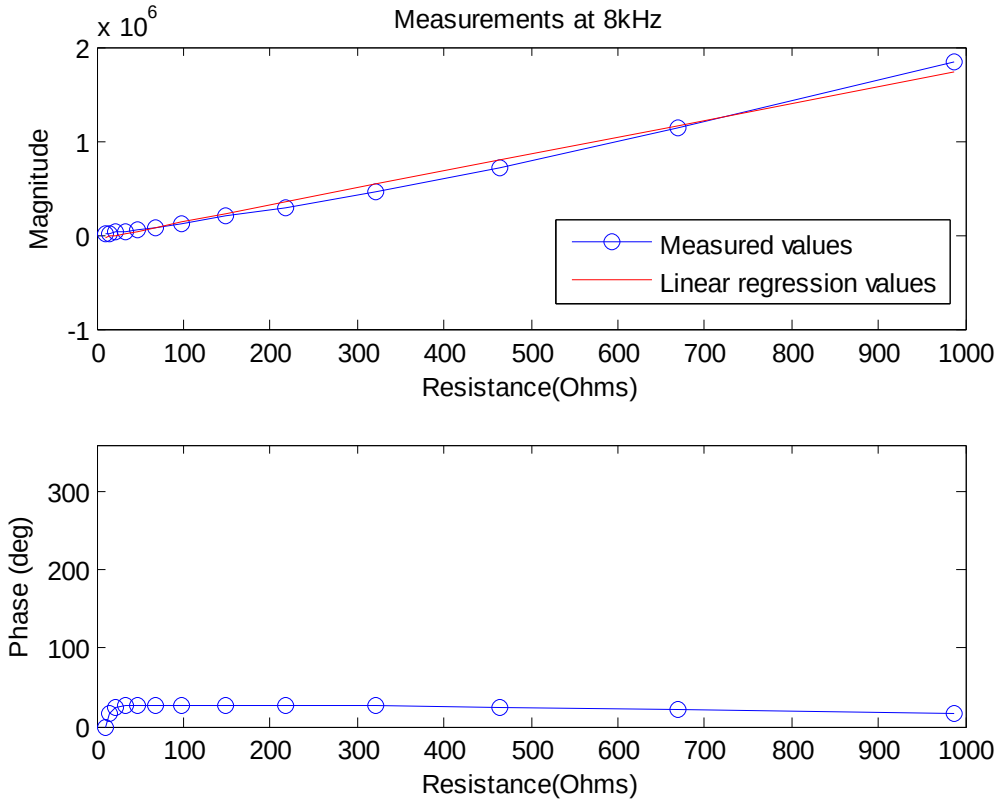


Figure 5.3: measurements at 8 kHz. R^2=0.9903

The result at 8 kHz is almost the expected. However, it can be noticed that the magnitude response is not as linear as it is supposed to be and the phase response is not constant when measuring small resistors. The non-linearity on the magnitude response is due to the transformers on the THDBA-ADA, whose cut-off frequency is 30 kHz. This effect was observed at all frequencies when performing the same measurements before replacing the transformers (cut-off frequency at 400 kHz). The inaccuracy with the phase is due to the front-end. As will be shown later, when measuring small resistors at 8 kHz and 96 kHz the font-end suffers the effect of capacitive coupling.

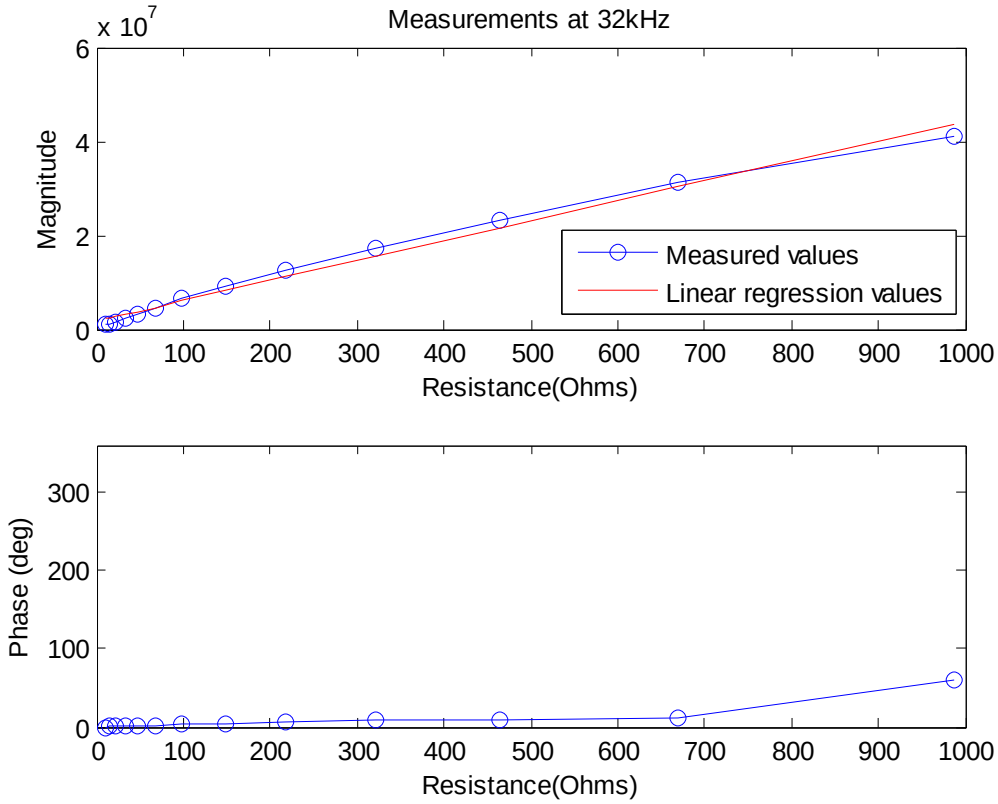


Figure 5.4: measurements at 32 kHz. R^2=0.9887

At 32 kHz the same problem with the magnitude response was found due to the effect of the transformers on the THDB-ADA board. When measuring resistors larger than ~820Ω, the phase suddenly changes due to internal oscillations of the front-end. It can be observed on the oscilloscope that the output signal of the front-end exhibits perturbations in spite of the fact that the signal applied by the DAC is correct.

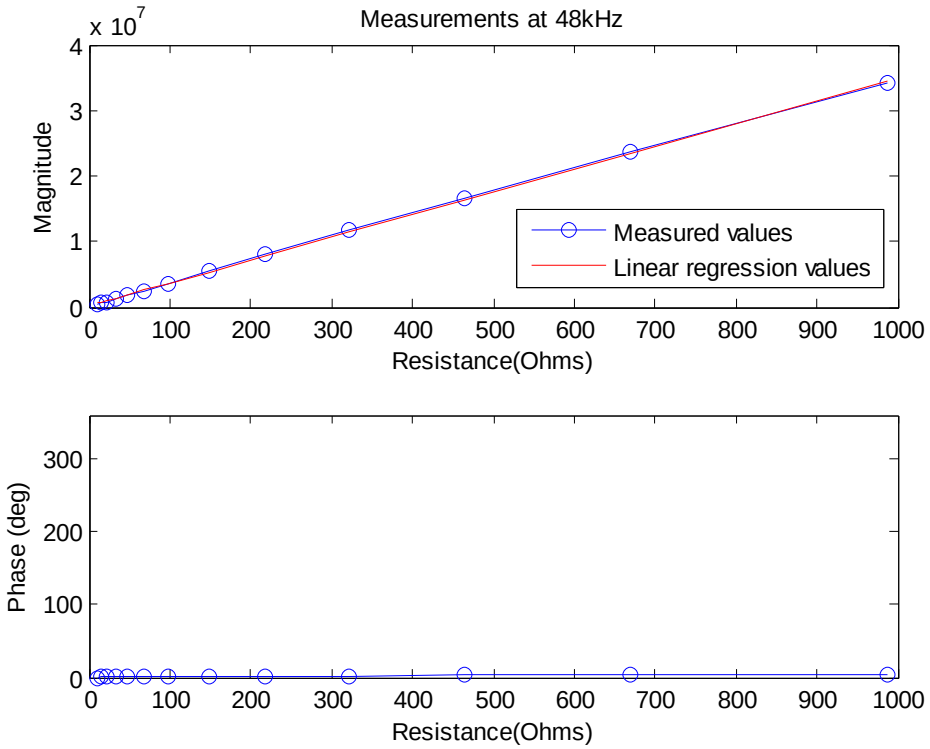


Figure 5.5: measurements at 48 kHz. R^2=0.9996

As shown, the response at 48 kHz is the desired.

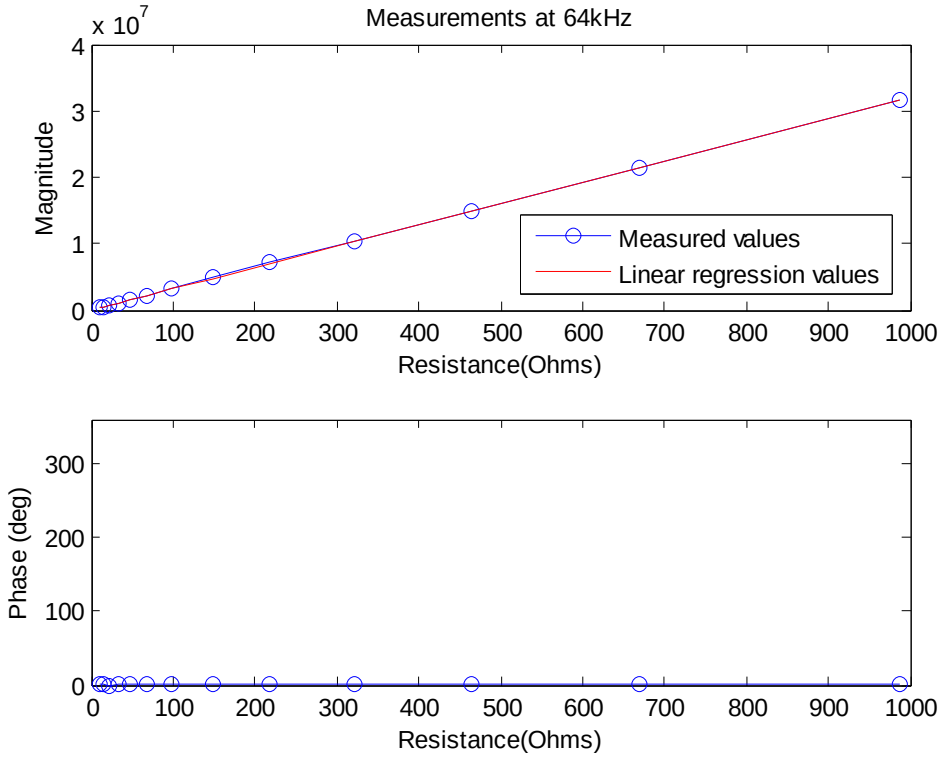


Figure 5.6: measurements at 64 kHz. R^2=0.99999

Again, the response at 64 kHz is the desired.

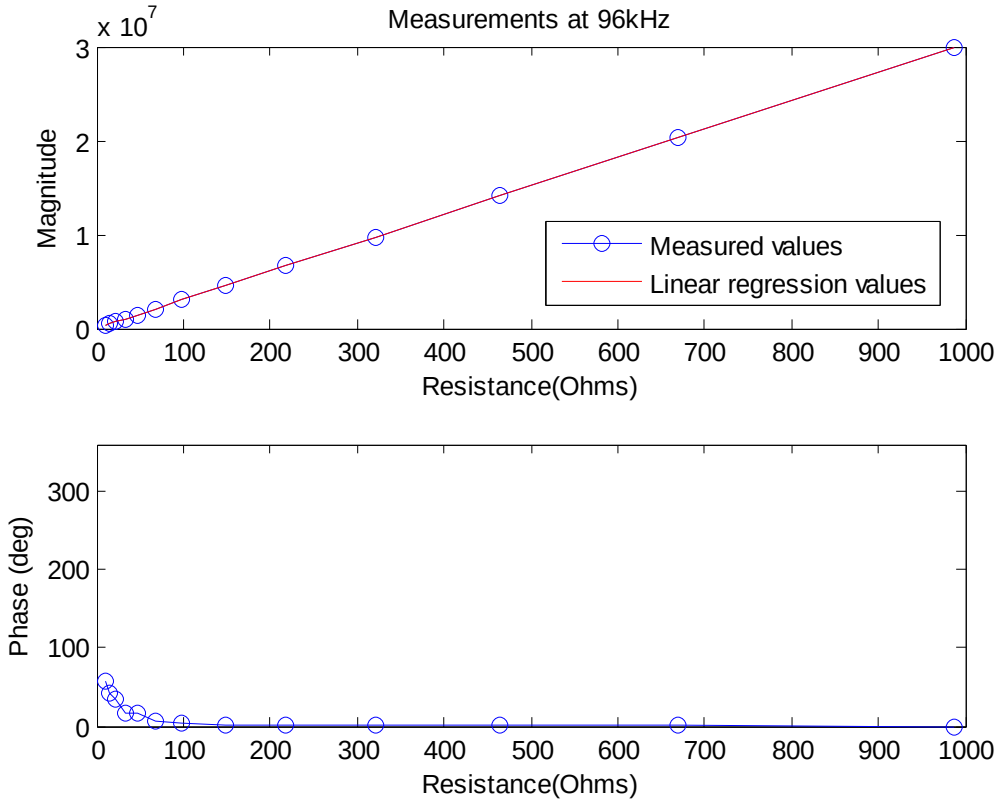


Figure 5.7: measurements at 96 kHz. R^2=0.99999

As mentioned before, measurements of small resistors at 96 kHz are affected by the response of the front-end, due to capacitive coupling.

Solutions to this problems are given in section 6.

After the previous measurements, a resistor of 150Ω was measured during 30 seconds (1000 samples) to check the dispersion of the system. These measurements were taken at 48 kHz, since this frequency was not affected by the front-end and is widely used in bioimpedance measurement. The histogram of the measurement is plotted in figure 5.8, as well as the Gaussian function fitted to it. The system provides a confidence interval of 0.93Ω at the 95,45% (2σ) confidence level (0.627% of the measured value).

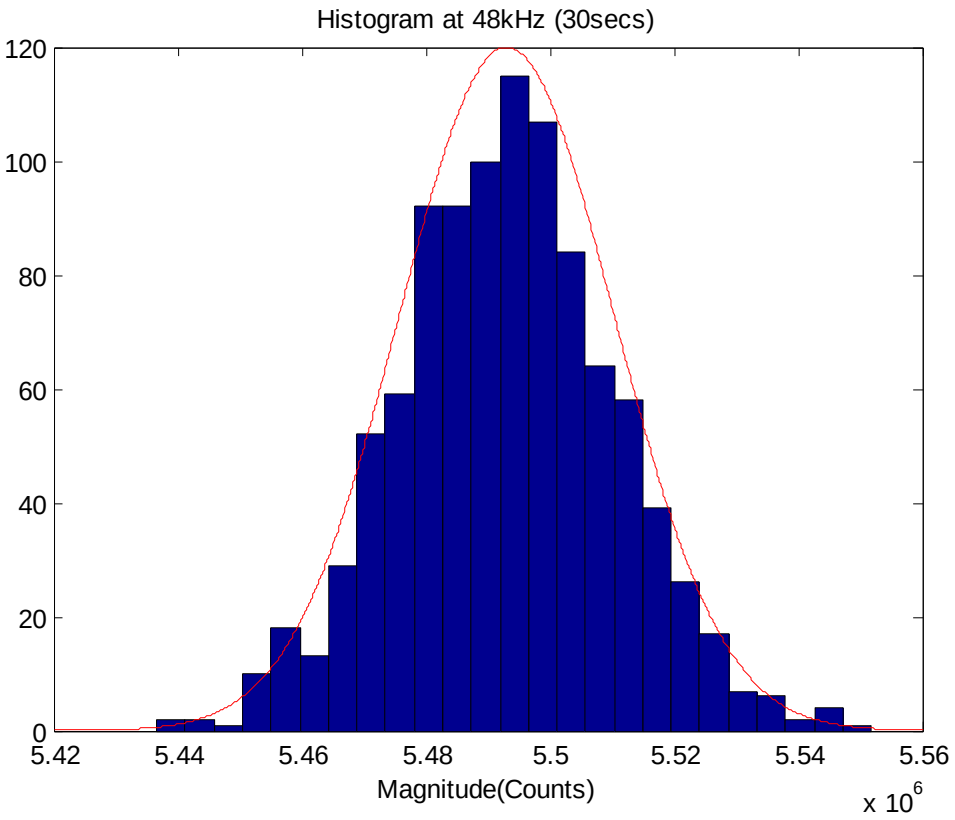


Figure 5.8: histogram of the measurement of 150 Ω resistor at 48 kHz during 30 seconds

Another way of checking out the performance of the system is detecting small changes. In this case, the 150Ω resistor was measured again during approximately 15 seconds and then a resistor of 150KΩ was connected in parallel, reducing the effective resistance by ~0.1%. Mean values were obtained averaging the samples on extremes, avoiding the transition.

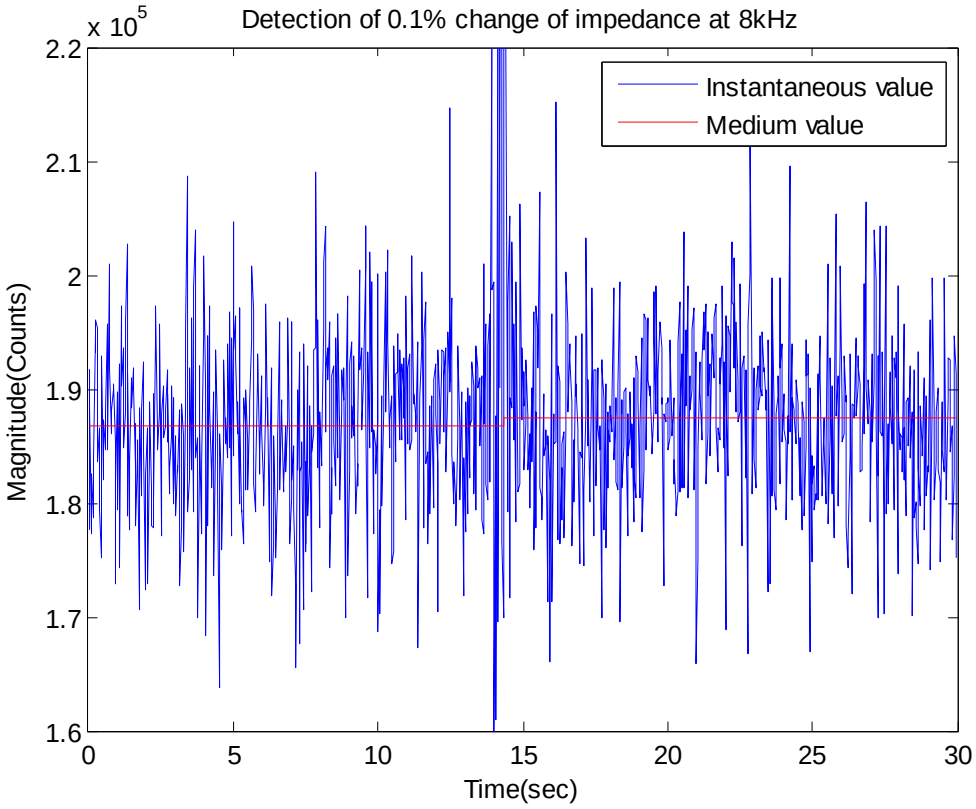


Figure 5.9: detection of 0.1% change of impedance at 8 kHz

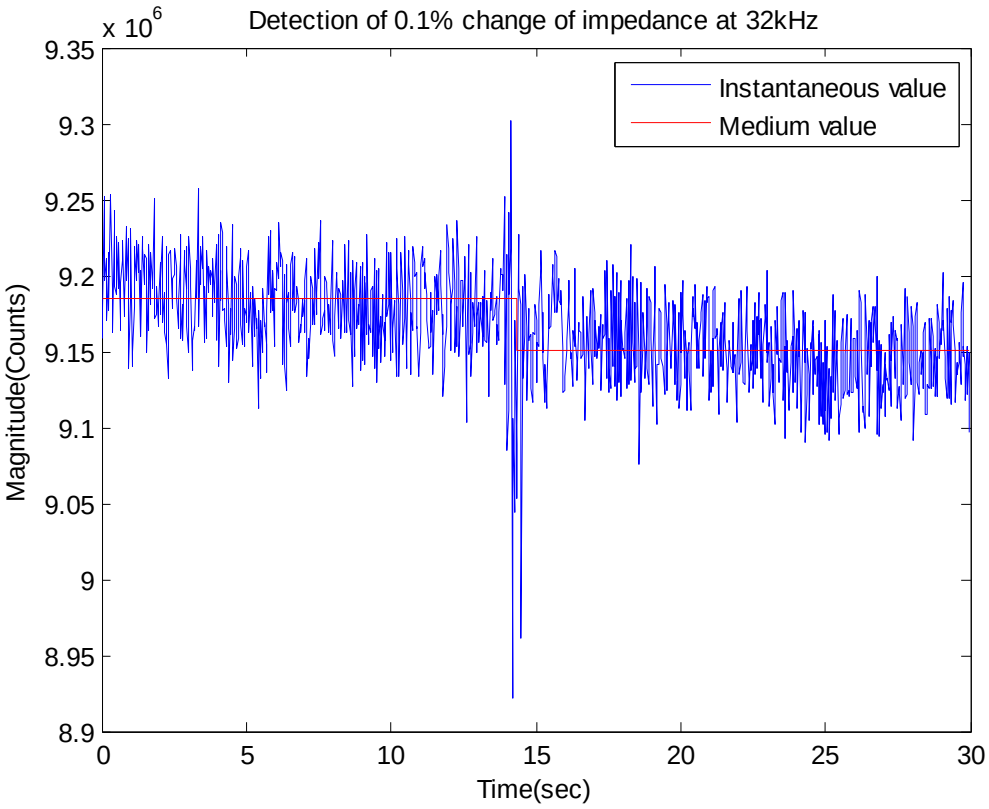


Figure 5.10: detection of 0.1% change of impedance at 32 kHz 21

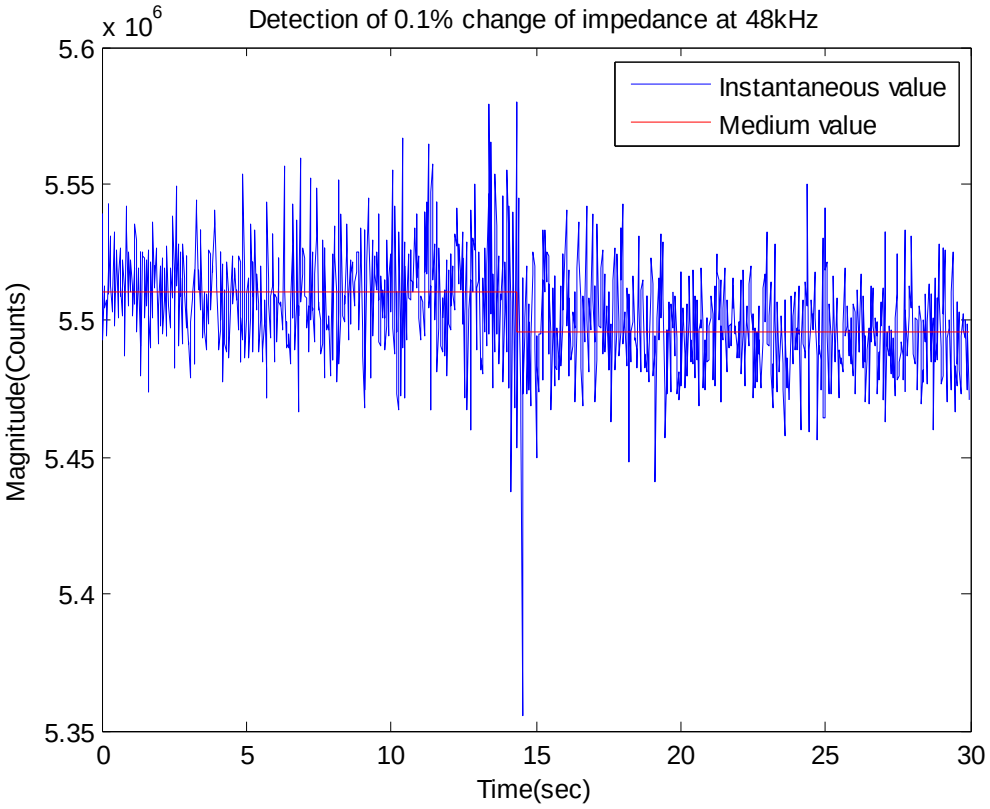


Figure 5.11: detection of 0.1% change of impedance at 48 kHz

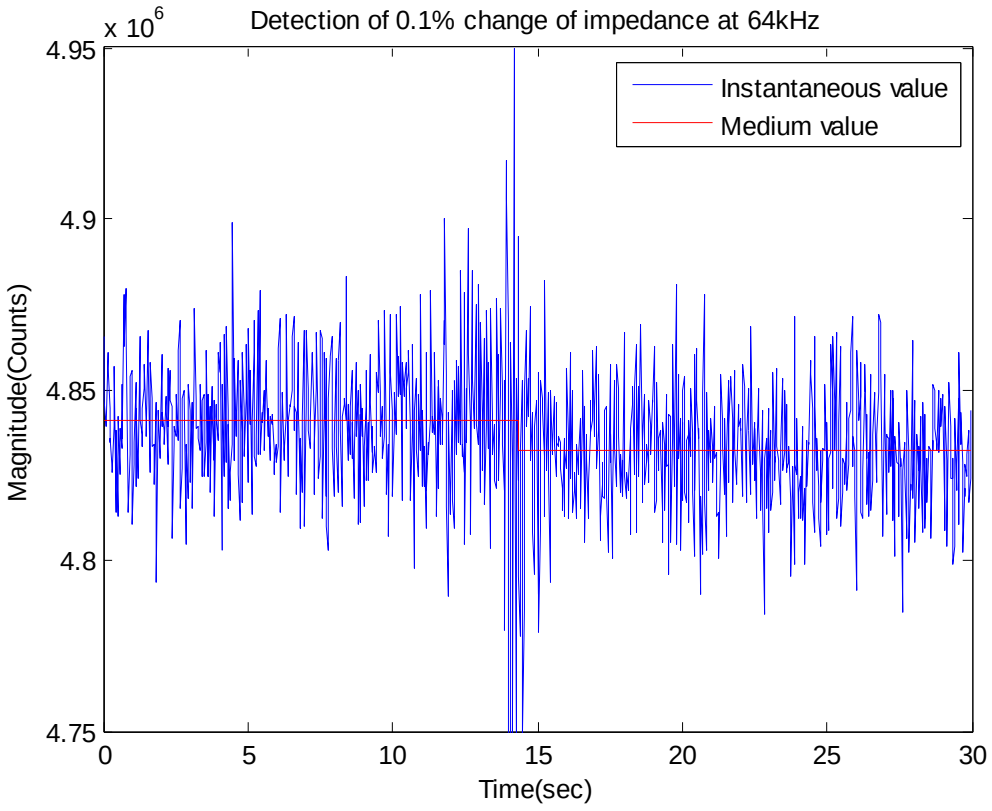


Figure 5.12: detection of 0.1% change of impedance at 64 kHz

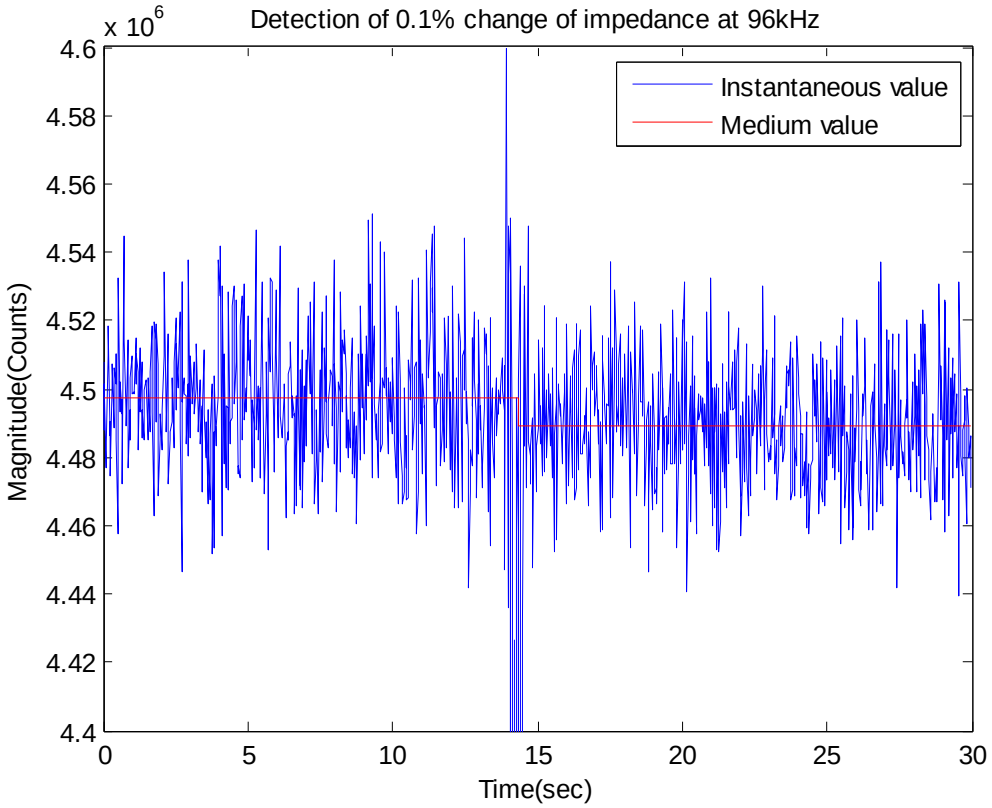


Figure 5.13: detection of 0.1% change of impedance at 96 kHz

As shown, the change of the impedance magnitude was detected at all frequencies but the lowest one.

Finally, the last measurement before measuring on the human body was the measurement of a RC network that consists on one resistor (50 Ω) in series with the parallel of one resistor (150 Ω) and one capacitor (56 nF), simulating a tissue.

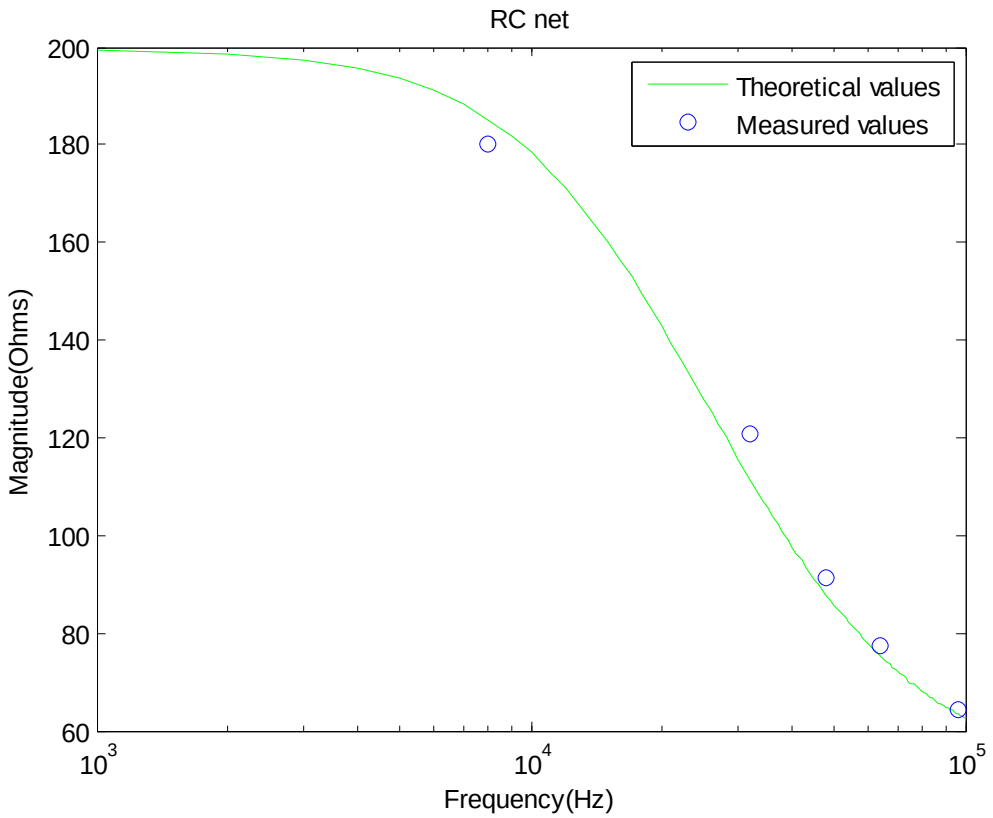


Figure 5.14: magnitude response of the RC network

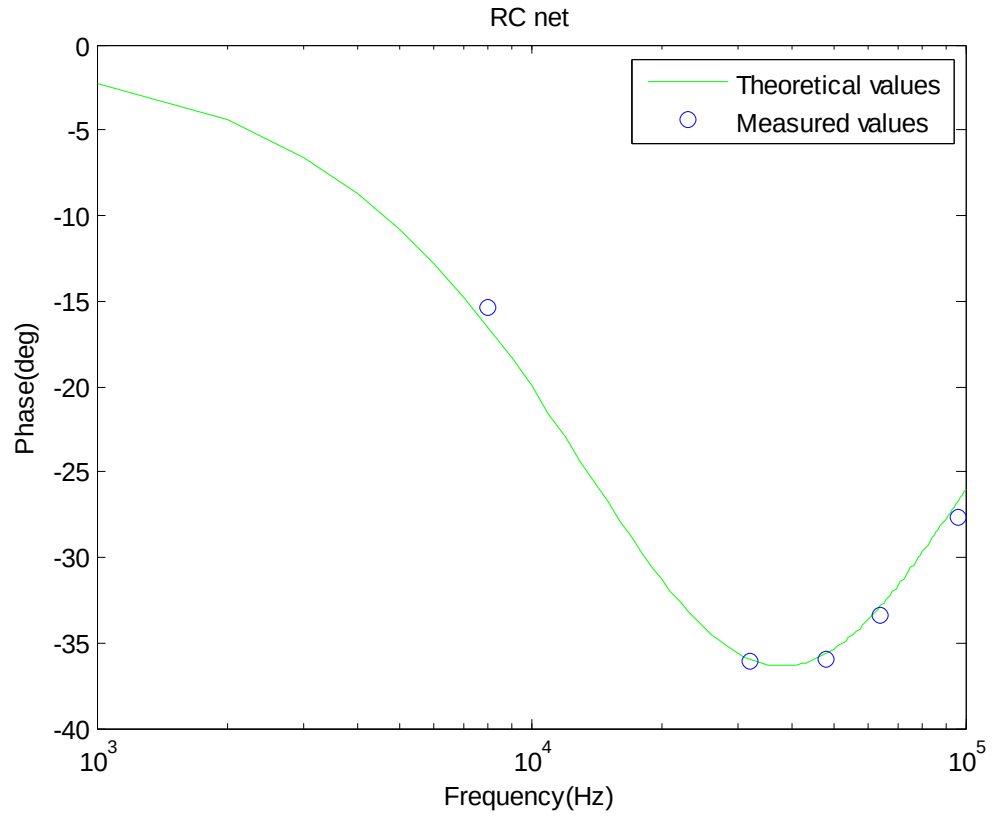


Figure 5.15: phase response of the RC network

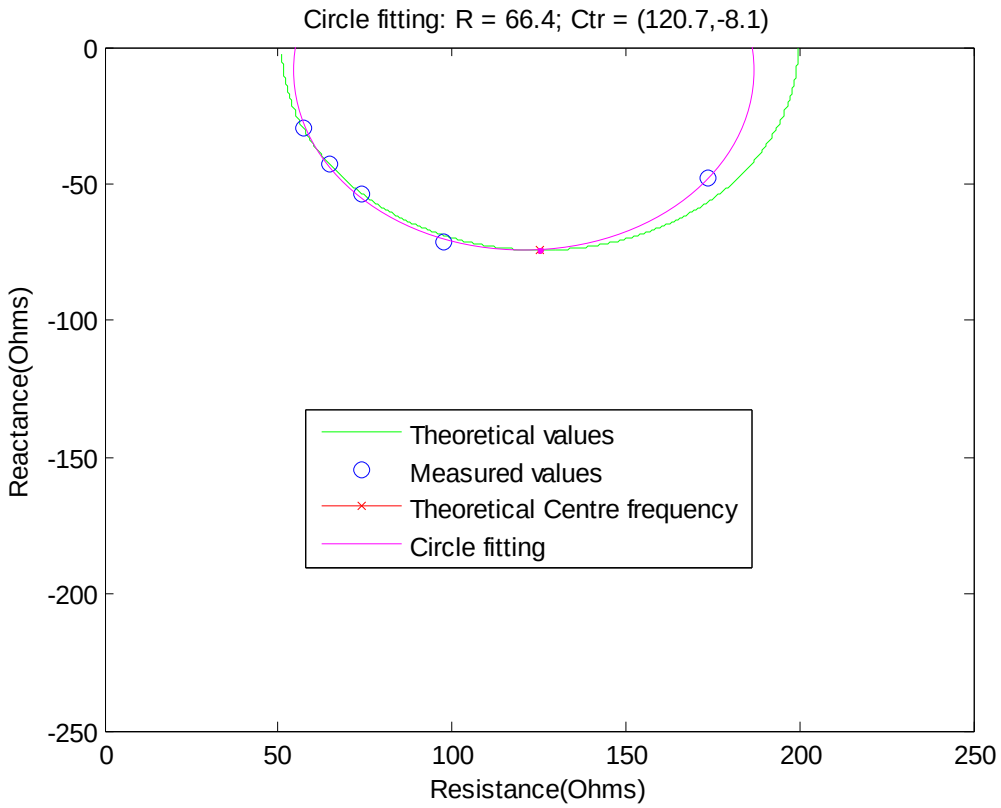


Figure 5.16: Cole-Cole plot

As shown, the measurements corresponds pretty well to the theoretical values (they were calculated with the real value of the resistors -measured using a multimeter-).

5.2. Measurements on the human body

Finally some measurements were performed on a subject. First, the electrodes were placed on the forearm of the subject and measurements were taken during 30 seconds. Then the measurements were repeated, but this time the subject contracted the muscle. As expected, the magnitude of the impedance was increased in the second measurement.

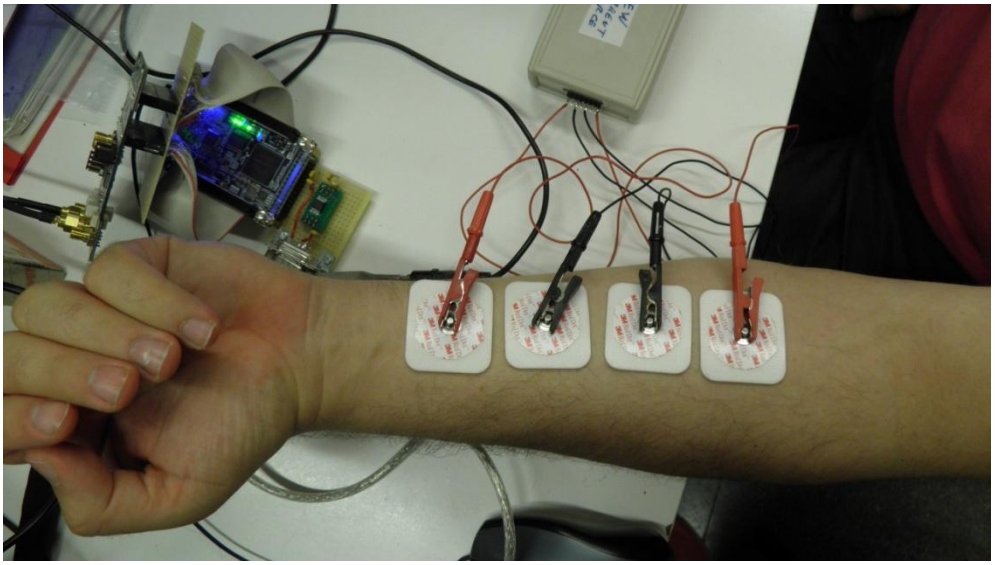


Figure 5.17: electrodes placed on the forearm to measure the bioimpedance of the muscle

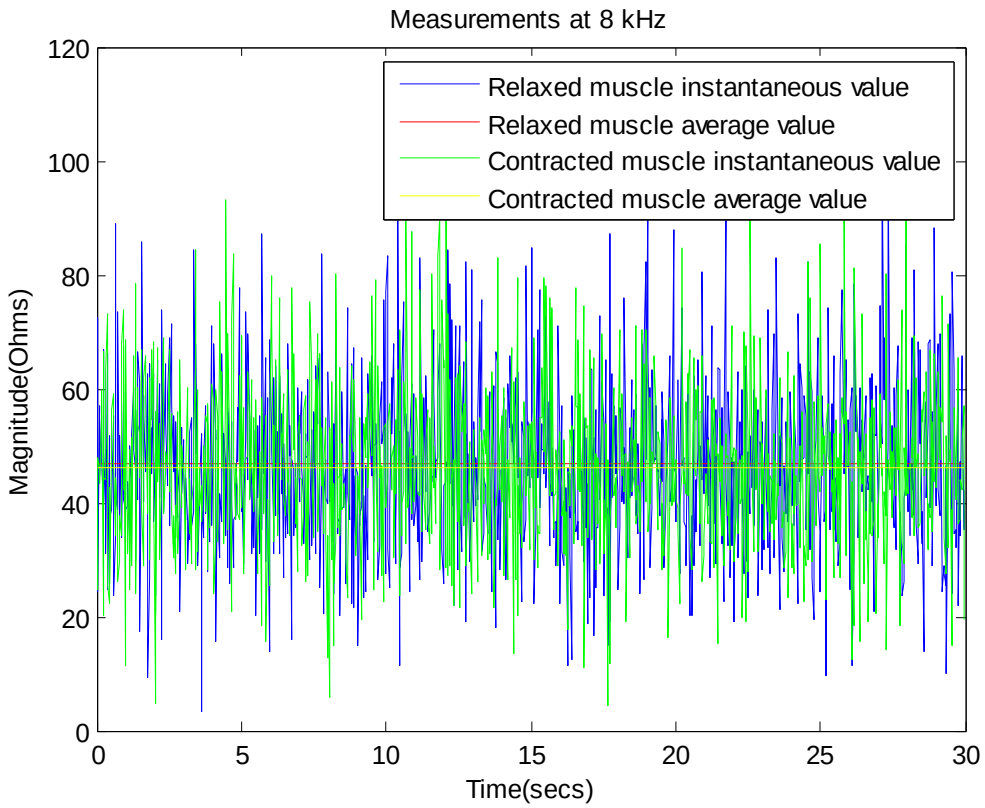


Figure 5.18: measurements of the muscle at 8 kHz

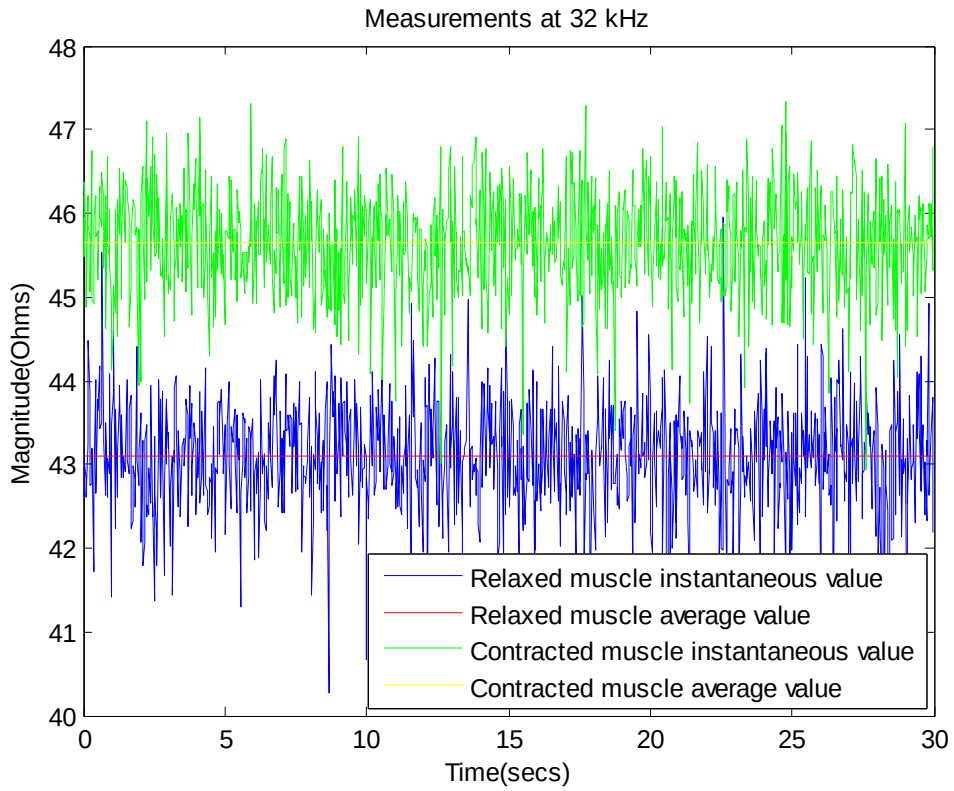


Figure 5.19: measurements of the muscle at 32 kHz

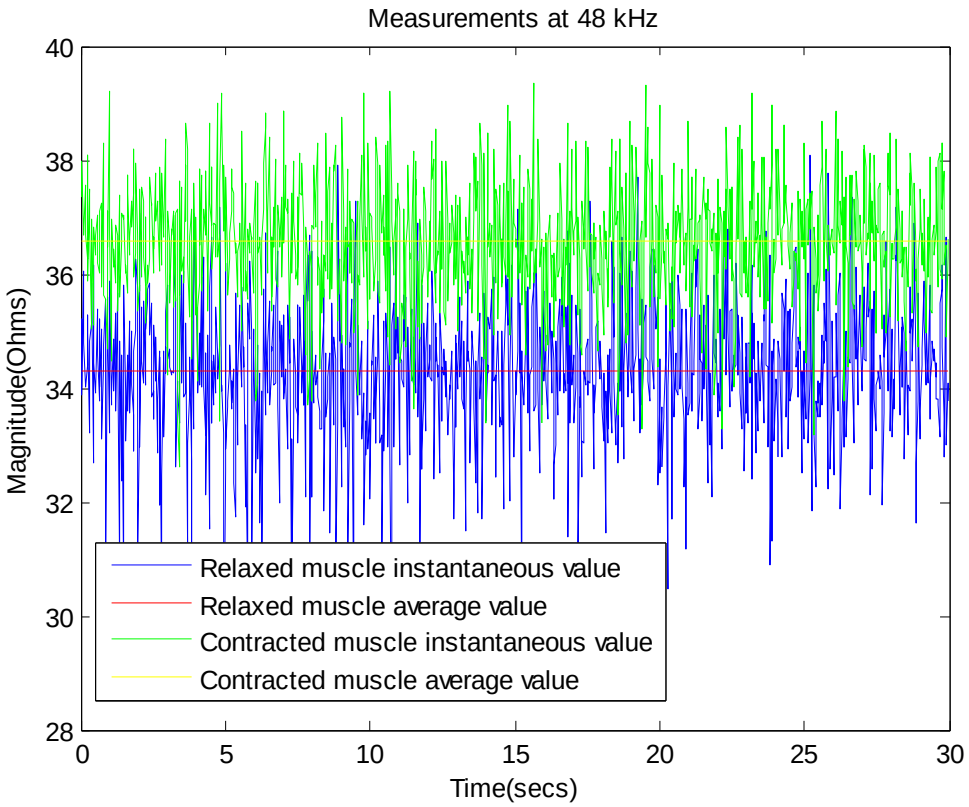


Figure 5.20: measurements of the muscle at 48 kHz

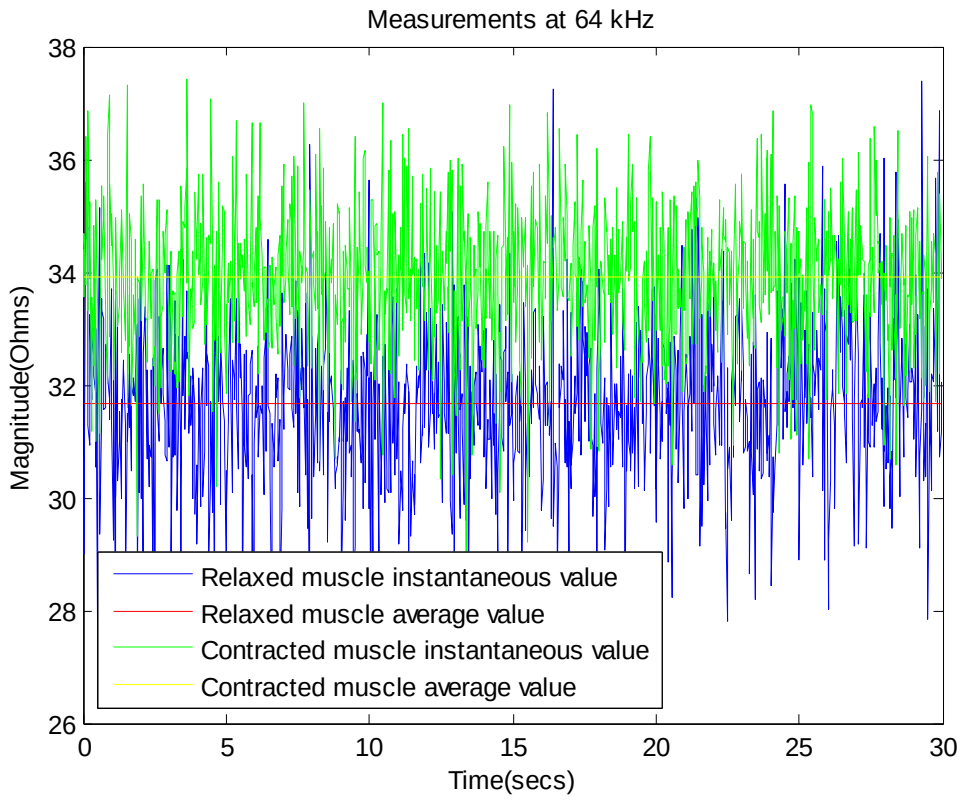


Figure 5.21: measurements of the muscle at 64 kHz

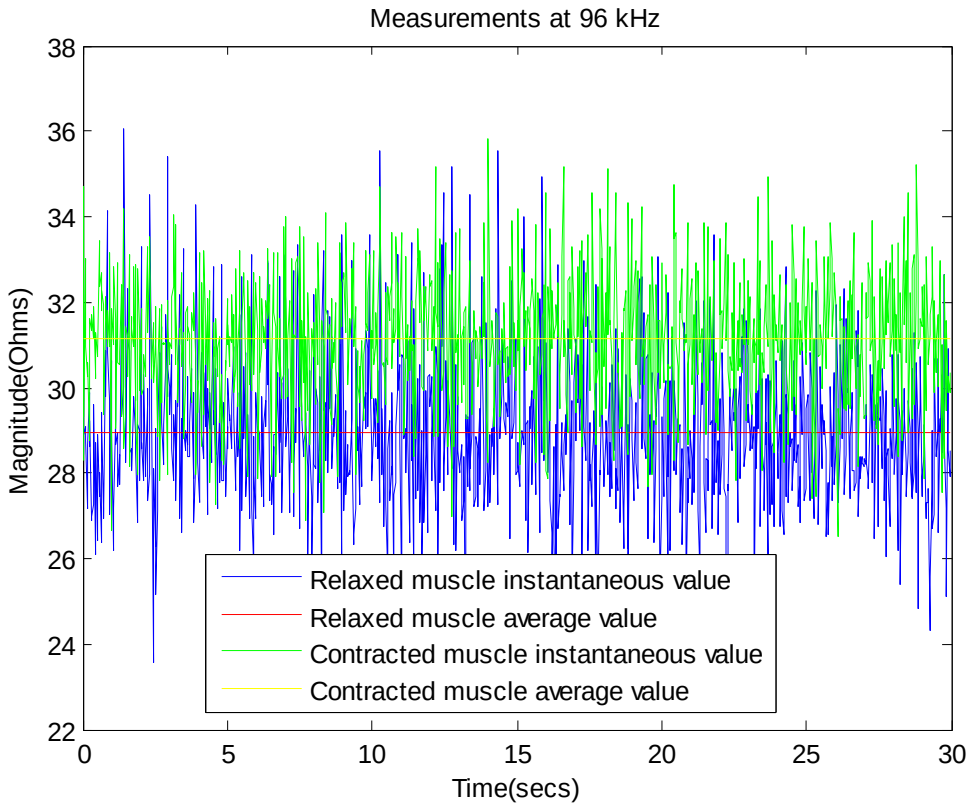


Figure 5.22: measurements of the muscle at 96 kHz

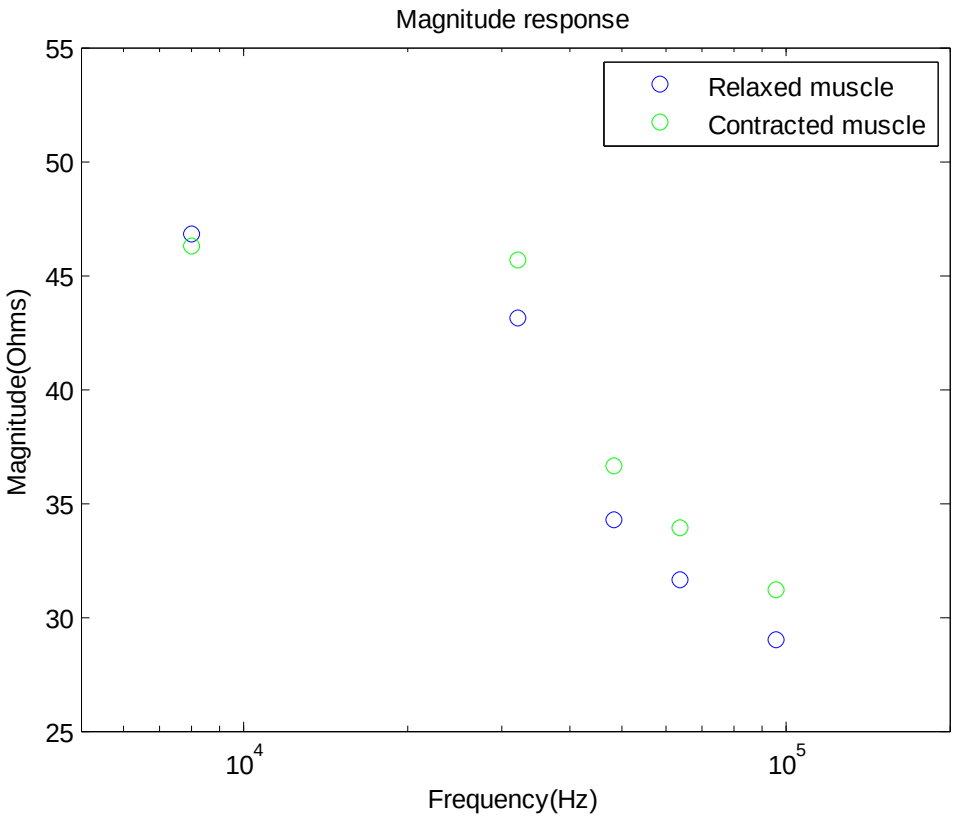


Figure 5.23: magnitude response of the muscle

As shown, measurements correspond with expected values, showing an increment in the impedance magnitude when the subject contracts the muscle. At 8 kHz this change cannot be observed due to the effect of the transformers on the conversion board.

After the previous measurements, electrodes were placed on both forearms to measure the respiratory rate and waveform. Measurements at 48 kHz are shown.

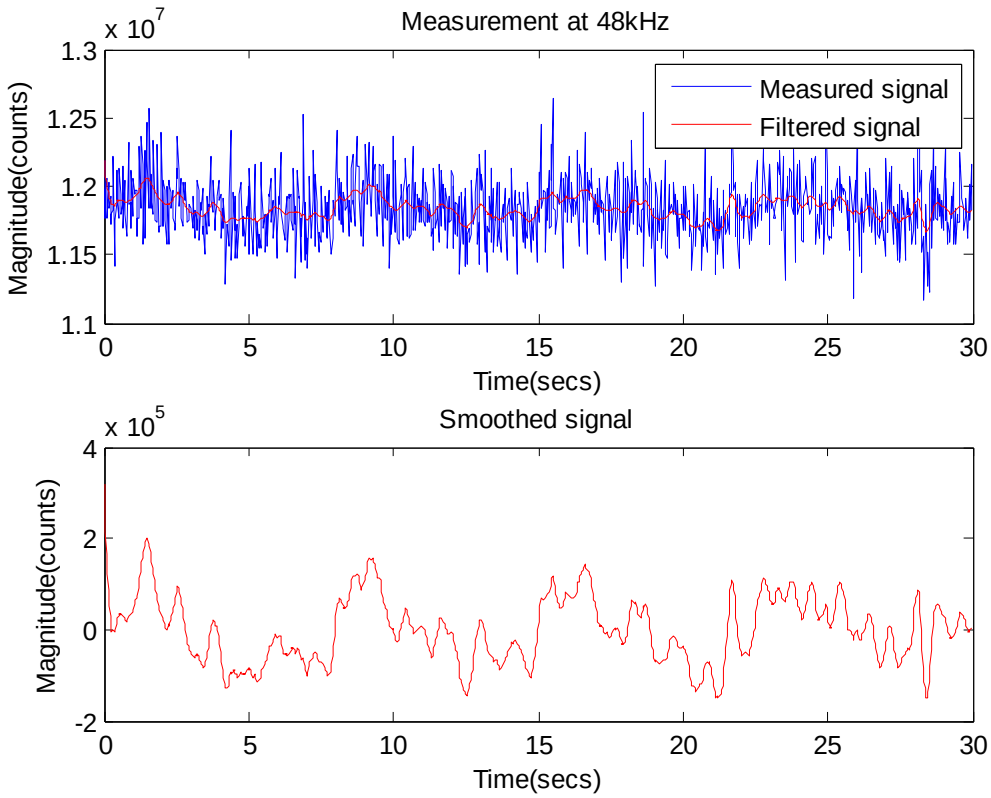


Figure 5.24: measurements of respiratory rhythm with electrodes placed on the forearms

Both respiratory and heart rates can be extracted from these measurements, where a moving average filter with n=10 has been used to extract the signals. The amplitude of the relative variation of respiratory signal is enhanced when measuring on the trunk.

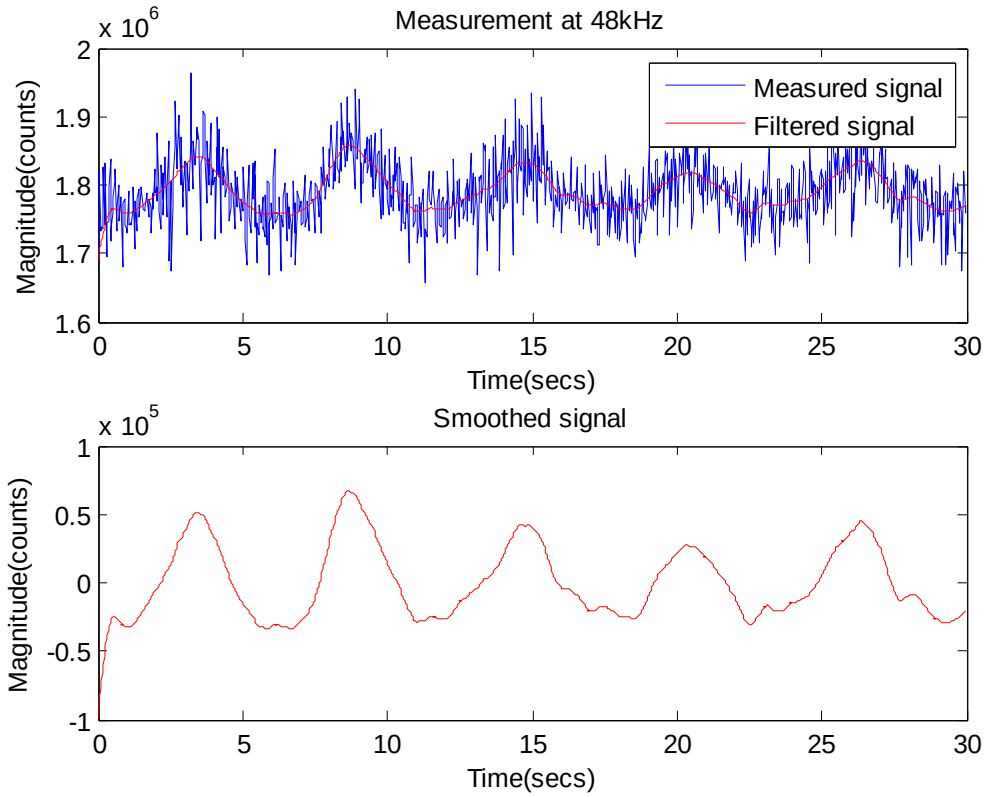


Figure 5.25: measurements of respiratory rhythm with electrodes located on the right side of the trunk. n=20

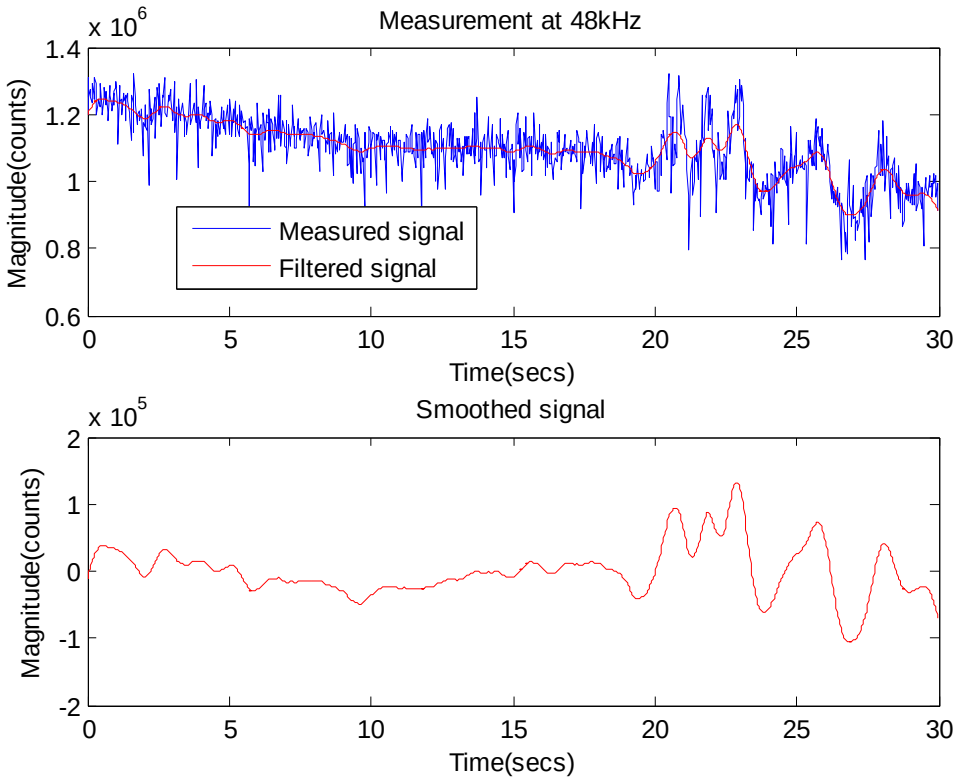


Figure 5.26: measurements of respiratory rhythm with electrodes located on the right side of the trunk. Apnea followed by breathing. n=20

6. Conclusions and future development

6.1. Conclusions

A low-cost FPGA-based bioimpedance measurement system has been designed, implemented and characterized in this project. Signal generation and coherent demodulation circuits were implemented on the FPGA, as well as the NIOS II embedded processor. An AnologDigitalAnalog conversion board (that needed a matching board) and the front-end previously designed and implemented by the group of instrumentation and biomedical engineering were also incorporated. A communication mechanism between the FPGA and the computer was also designed and implemented.

As observed on the characterization stage, the system works as expected but results are degraded due to some spurious effects of the front-end and the transformers on the THDB-ADA.

The system implemented on the FPGA allows the measurement of bioimpedances from very low frequencies up to 12.5 MHz, but in order to obtain accurate results at low frequencies it would be needed to replace transformers on THDB-ADA with active circuits to eliminate the high pass filter response as well as minimize spurious effects of the front-end by common supply and interference isolation.

Objectives have been achieved, designing a system with a cost of approximately 300$ which is able to detect impedance variations below 1% within the band from 10 kHz to 100 kHz, providing 33 spectrums per second. Besides, the spectrum to be measured and the parameters of the measurement are completely configurable just modifying the embedded software, so the user can select other spectral parameters (for instance at higher frequencies) or even measuring at only one frequency.

Regarding to the temporal resolution, it is not too high (33 samples per second at each frequency) since we are measuring low frequencies and therefore measuring times are clearly higher. Performing measurements at only one frequency of 1 MHz with a filter length of 2048 samples, a temporal resolution of 1953 samples per second is achieved. It is interesting for some applications, e.g., to obtain the heart rate from measurements of the respiratory rate.

6.2.Future development

- Replace the transformers on the THDB-ADA with active circuits that remove the high pass filter response, improving the linearity and the accuracy of the system.
- Minimize spurious effects of the front-end, trying to remove them or replacing it with another.
- Minimize costs developing a custom-made ADA conversion board.

- Improve the communication between the UART and the PC, specially the RS232-USB converter that limits the communication.
- It would be interesting to achieve a wireless communication in order to obtain a completely portable system.

7. Acknowledgements

En primer lugar quisiera agradecer a Ramon Bragós la oportunidad de realizar este proyecto y aprender de él, así como toda la ayuda ofrecida. También me gustaría agradecer al Departamento de Bioingeniería Electrónica de la UPC el permitirme realizarlo.

En segundo lugar me gustaría agradecérselo a toda mi familia, cuyo apoyo ha sido constante a lo largo de toda mi formación, y a mis antiguos compañeros de clase (Dani, Edgar, Jorge, Pablo, Fran, Patri, Chema..) con quienes compartí innumerables horas de estudio y buenos momentos.

Por último destacar mis agradecimientos a mi pareja Lonalyn por todo el apoyo dado en la realización del proyecto y a mi hermano gemelo Eduardo, quien ha sido compañero de estudios durante toda mi vida y cuya compañía y ayuda he echado de menos este año.

Muchas gracias a todos.

8. References

[1] Holder, D. Appendix A. Brief introduction to bioimpedance. Electrical Impedance Tomography: Methods, History and Applications. 2010

[2] G.P. Drago, M. Marchesi, S. Ridella, “Thefrequencydependence of ananalytical model of anelectricallystimulatedbiologicalstructures”, Bioelectromagnetics,vol. 5, pp. 47-62, 1984.

[3] Bragós R, “Caracterització de teixits i sistemes biològics mitjançant l'espectroscòpia d'impedància elèctrica”, Tesis doctoral, UPC, 1997.

[4] Álvaro Andrade Sánchez, Joan Mendoza Equiza, "Bioimpedance hardware system for physiological parameters estimation in automotive", final degree project, 2013.

[5] Sergi Reig Quiroga, Xavier Fernández Espiga, "Design, implementation and characterisation of the hardware of an fpga-based multifrequency impedance analyser", final degree project, 2013.

[6] Agilent Technologies Inc.“Agilent Impedance Measurement Handbook: A guide to measurement technology and techniques”, 4th Edition, 2009.

[7] Analog devices. Last checked: 16/10/2014 . Available on: http://www.analog.com/en/rfif-components/direct-digital-synthesis-dds/ad5933/products/product.html

[8] B Sanchez, J Schoukens, R Bragosand G Vandersteen, "Novelestimation of theelectricalbioimpedanceusingthe local polynomialmethod. Application to in-vivo real-timemyocardiumtissueimpedancecharacterizationduringthecardiaccycle",IEEE Transactions on BiomedicalEngineering, vol.58, no.12, 3376-3385,2011.

[9] DE0-Nano User Manual. Last checked: 16/10/2014 . Available on: http://www.altera.com/literature/ug/DE0_Nano_User_Manual_v1.9.pdf

[10] THDB-ADA User Guide. Last checked: 16/10/2014. Available on: http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=73&No=278&PartNo=3

[11] Farnell, electronic components. Last checked: 16/10/2014. Available on: http://es.farnell.com/maxim-integrated-products/max3227ecae/ic-transceptor-rs-232-16ssop/dp/1593349

[12] Aten. Last checked: 16/10/2014. Available on: http://www.aten.com/Mobility-&-USB/USB-Converters/USB-to-Serial-Converter~UC232A.html#.VD_dbfl_tvn

[13] Introduction to the Quartus II software. Last checked: 16/10/2014. Available on: www.altera.com

[14] NCO MegaCore Function User Guide. Last checked: 16/10/2014. Available on: www.altera.com

[15] Nios II Software Developer's Handbook. Last checked: 16/10/2014. Available on: www.altera.com

[16] Embedded Peripherals IP User Guide. Last checked: 16/10/2014. Available on: www.altera.com

[17] HAL API Reference. Last checked: 16/10/2014. Available on: www.altera.com