Design and implementation of a compact Vector Network Analyzer

By
Marcos Martinez

A thesis submitted in partial fulfillment of
the requirements for the degree of

Master of Science
(Electrical Engineering)

at the
University of Wisconsin- Madison
2012

ABSTRACT

Vector Network Analyzers (VNAs) are extensively used in Radio Frequency circuit design to characterize RF components and systems. They are also used for materials characterization, as analyzing the purity of petroleum or for detecting skin tumors. These instruments perform the measurement of the S-parameters, which are used to describe the electrical behavior of linear electrical networks.

In this project we are developing a miniature device that can be connected to a smartphone. Together, our device and the smartphone can provide some basic VNA features. The device will be in charge of the RF signals generation and detection, while the phone will be used to provide a User Interface (making use of its touch screen) and data processing. This new device would be of great importance in applications that require high mobility, as system's validation in difficult locations (for example an antenna located on the top of a tower); it can also be of great use in education, because it would be a cheap instrument that could be used in university laboratories or high schools. In research it could be used in portable medical instruments and, finally, radio amateurs could have a great and affordable tool to use in their home projects.

This project has been carried out within the van der Weide research group at the College of Engineering of the University of Wisconsin – Madison.

CONTENTS

ABSTRACT 1

1. INTRODUCTION 4

2. THEORY 5

3. PREVIOUS WORK ON THE FIELD 14

4. SIMULATIONS 16

5. DESIGN 24

a. HARDWARE 24

b. SOFTWARE 28

6. RESULTS 30

7. CONCLUSIONS 35

8. REFERENCES 37

9. APPENDICES 39

1. INTRODUCTION

Miniaturization has been a constant objective in technology development, with a bigger emphasis in the electronics field. The development of smaller electronic components has fostered the portability of electronic devices, making it possible nowadays to carry a phone, a music/video player, a camera and a gps in our pocket. In parallel to this, miniaturization has been employed in more specific fields as in instrumentation where we can find battery-powered portable Spectrum Analyzers, Oscilloscopes and even Network Analyzers, facilitating the portability of such instruments and having a great impact on field work. Following both tendencies it can be foreseen that the next step will consist on the integration of such portable instruments with smartphones, making use of the user interface of the latest, as well as their processing power. This has already happen in the oscilloscope field, where we can find the miniature, smartphone connectable oscilloscope iMSO-104 from Oscium [1].

Based on this and with the previous work on the field of compact Network Analyzers performed within the van der Weide's research group by A. Wangsanata [2] and Min K. Choi and Min Zhao [3], it was decided to focus the work of this project in the design, prototyping and validation of a smartphone-connectable Compact Vector Network Analyzer.

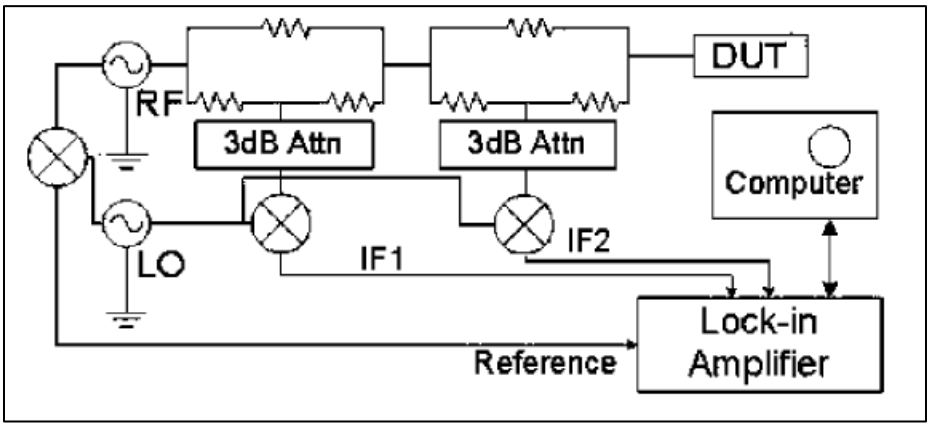


Figure 1: Circuit diagram of reflectometer developed by Min K. Choi and Min Zhao [3]

To complete this objective, most of the high-performance laboratory instruments used previously will be replaced by low-cost chip components, assessing their limitations and their impact on the VNA performance.

To carry out this project in first place the theory underneath VNA's is developed, as well as the different radio frequency components involved. Next a simulation prototype will be described, and the results of the simulations analyzed. Next the design of the prototype will be presented, covering the two main aspects, the hardware design and the software design. Finally the measurements of the prototype will be analyzed and compared to the results of the simulation and the measurements of a commercial VNA. At the end some conclusions will be presented.

2. THEORY

Network Analyzers are widely used in the development of electronic devices, and more concretely in the development of high frequency electronic devices. Current network analyzers' frequencies of operation can range from a few hertzs to tenths of Gigahertzs.

To understand how a network analyzer works we have to refer to high frequency network analysis. This analysis is useful when the electric length of our circuit is comparable to the wavelength of the electric signals we are going to work with, because in this case the currents and voltages are not equal along the circuit's length. In this document we will only focus on S-parameters' theory, but an in depth explanation of transmission lines' analysis and network analysis can be found in chapters 3 and 4 of [4].

For the S-parameter approach we consider every electronic circuit as a multi-port network, where each port can act simultaneously as input and output of signals. As an example we will work with a two port network as in Figure 2.

Figure 2: Example of a 2 port microwave network

Since the S-parameter approach is based on waves instead of voltages or currents we need to define waves coming into the network (a_1 and a_2), and waves coming out of the network (b_1, and b_2). For an n-port network these waves are defined as:

$$a_n = \frac{V_{0n}^{+}}{\sqrt{Z_{0n}}}, \quad b_n = \frac{V_{0n}^{-}}{\sqrt{Z_{0n}}}$$

V_{0n}^{+} and V_{0n}^{-} correspond to the voltage waves coming into and out of port n. Furthermore, the reason for the normalization is that $|a_n|^2$ represents the incident power at port n, and $|b_n|^2$ the reflected power.

Provided these waves, we can define the S parameters as:

$$S_{m,n} = \frac{b_m}{a_n}\Big|_{a_i=0,\ \forall i\neq n}$$

Therefore, we can measure the $S_{m,n}$ parameter of a network by measuring the power coming out of the m port while injecting a signal through the n port when all the ports are impedance matched. Because of this, most of the current VNA's only have two ports, since we can characterize devices with n ports simply connecting the VNA to a different pair of ports each time and connecting matched loads to the remaining ports.

Finally, with all the obtained S-parameters we can construct the S-matrix corresponding to the network as:

$$\begin{bmatrix} b_1 \\ \dots \\ b_n \end{bmatrix} = \begin{bmatrix} S_{11} & \dots & S_{1n} \\ \dots & \dots & \dots \\ S_{n1} & \dots & S_{nn} \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ \dots \\ a_n \end{bmatrix}$$

The first Network Analyzers were scalar, since they could only measure the power of the signals coming into the device and out of it, but the advances in the instrumentation technologies made possible to coherently measure the incoming and outcoming waves, being able to calculate the phase of the S parameters too. These new network analyzers are called Vector Network Analyzers or VNA's. The importance of the phase measurement is of great interest in the communications field, where the bandwidth of signals is growing, and non-linearities in the phase response of the electronic components can result in the distortion of the signal. Also, the possibility of measuring the phase of the S-parameters makes it possible to use them to characterize the electrical properties of different materials.

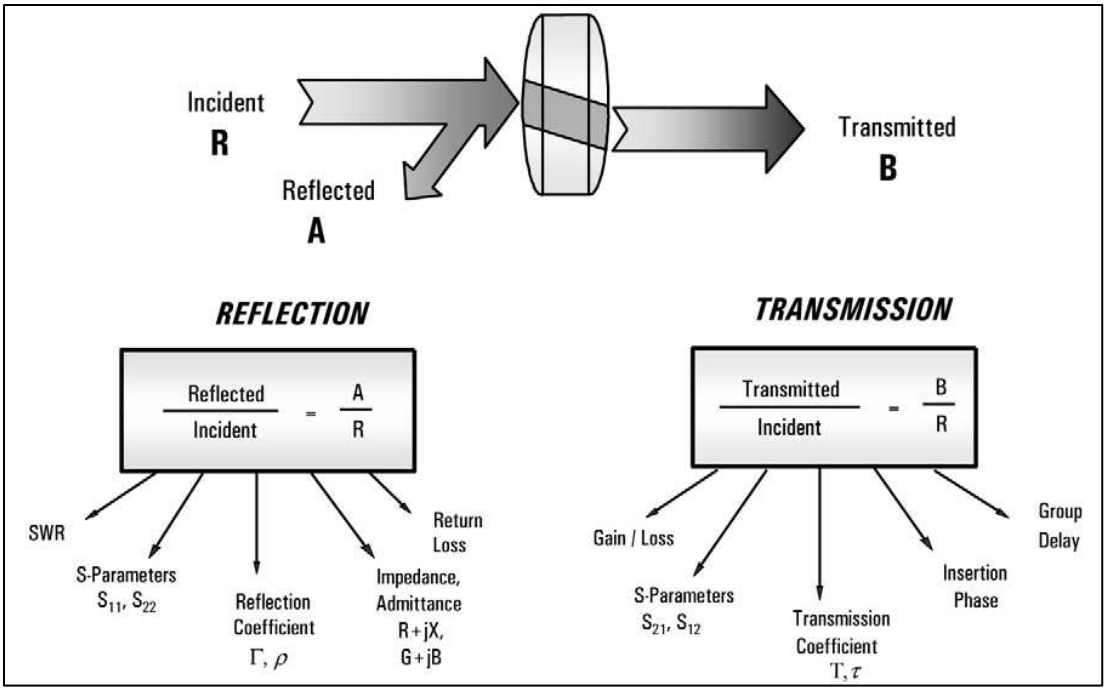


Figure 3: Network Analyzer terminology [5]

One important aspect of the S-parameters is that they vary with frequency, thus a given device can have different S-parameters at different frequencies.

Essentially, what a VNA does is, for a given frequency, generate a sinusoidal signal of that frequency, feed that signal to the port 1 of the Device Under Test (DUT), and measure the signal that is reflected at that port and the signal that is transmitted to the port 2, then calculate the ratios of these signals with respect to the signal injected to the DUT and finally display the results. It will do the same procedure for each frequency that form part of the frequency set the VNA has been configured to sweep.

The main elements of a VNA are depicted in Figure 4. These elements are: synthesizer (variable frequency source), signal separator (DC1 and DC2), signal detector (RX's), processor and display.

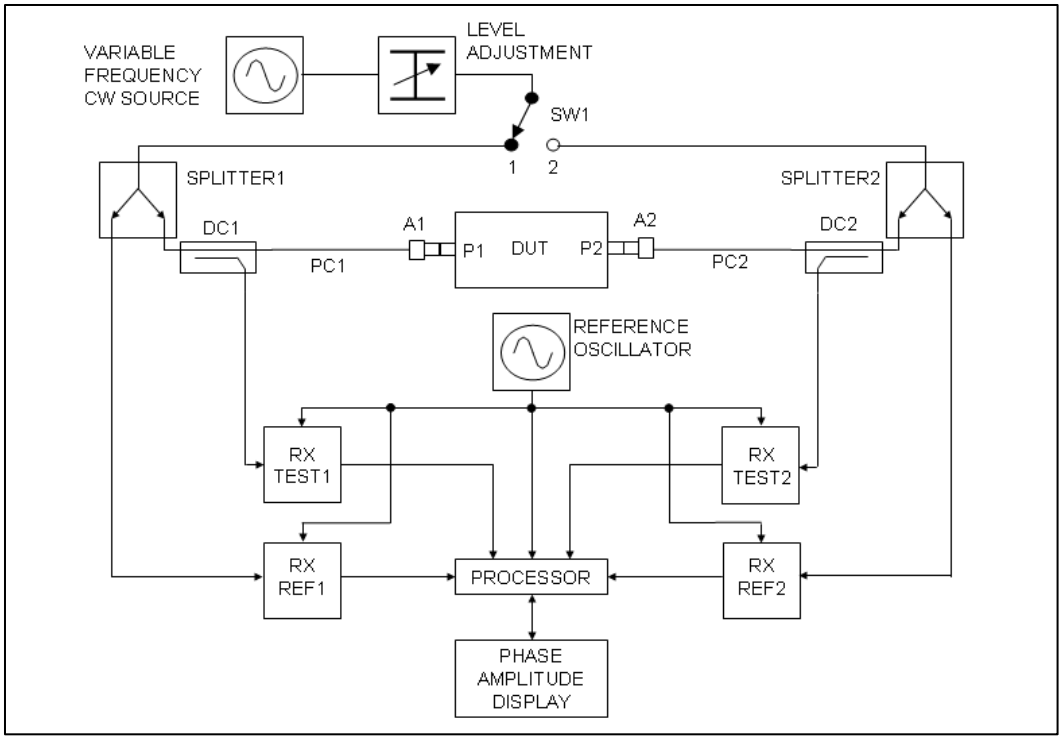


Figure 4: Basic elements of a VNA

Another important aspect of any network analyzer involves calibration. Since any component the signal has to go through from the VNA's synthesizer to the detector has an impact in the signal (in terms of changes in amplitude and delay) we need to make sure that we are able to deembed these effects from the measurement and therefore make sure that the characterization corresponds solely to the DUT. For this reason three known loads and one "thru" are used. The three loads are used to calibrate the input port, and are Match, Open and Short. The Match load provides a perfect match to the VNA's port, and therefore no reflection comes from it. The Open and Short loads both provide total reflection, but with different phase. The Open load's reflection has zero phase, while the Short load's reflection has a phase of 180 degrees or π radians (again all the theory underneath can be consulted in [4]). Having the S_{11} measurements of each load we can deembeed the effects produced by the VNA components and cables on the measurement of the S11 following the procedure described in [6]. In the case of the "thru", this is simply an impedance matched connection between the two ports of the VNA with near zero electrical length and attenuation. This way we get a reference for total transmission and zero phase delay from port 1 to port 2.

Once the concepts of VNA and the S parameters have been introduced, our approach to a new compact VNA will be presented. In our approach, we propose to perform a coherent measurement of the reflected signal through the use of an IQ demodulator used as a detector. By

means of the IQ demodulator we can mix the reflected signal with a sample of the signal generated by the synthesizer, getting the information of phase and magnitude of the reflected signal with only one measurement. The block diagram of the proposed VNA can be seen in Figure 5.

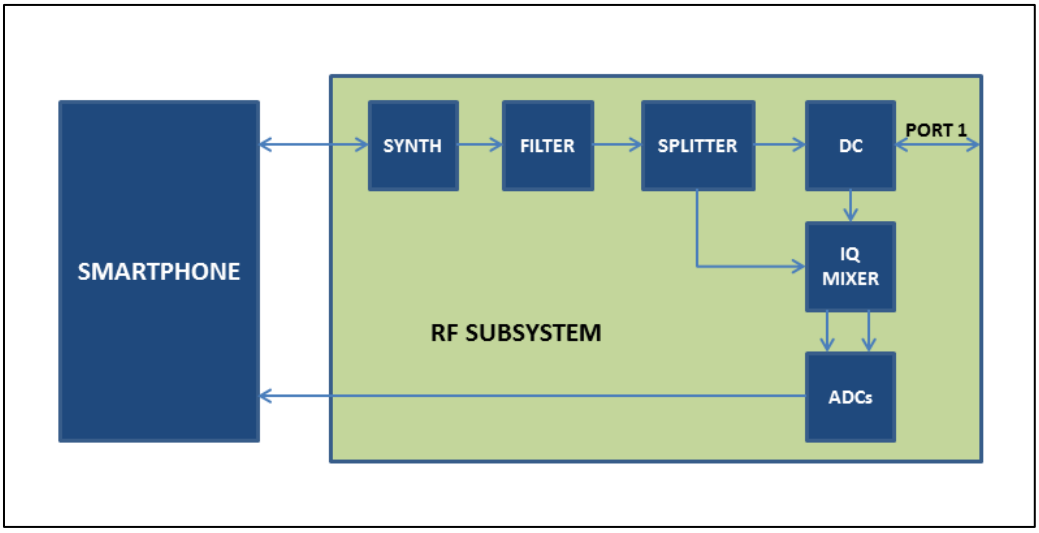


Figure 5: Block diagram of proposed VNA

This approach is inspired in the approach presented by Min K. Choi and Min Zhao in [3], reducing its complexity by using just one mixer and substituting the costly signal generators by a synthesizer on a chip, and the lock-in amplifier by an IQ demodulator.

From this point we will focus on the measurement of the S11 parameter since the approach can be extended to measure the S21 parameter by minimally modifying the architecture. This modification consists on switching the input of the IQ mixer from the Directional coupler to the port 2 of the VNA.

The synthesizer of a VNA is in charge of generating the sinusoidal signals that will be used to characterize a given DUT. For this reason, the synthesizer shall be able to generate single frequency signals covering the working bandwidth of the VNA. In our case the required VNA bandwidth will be 400-4000MHz.

The synthesizer used in our design is depicted in Figure 6 and consists of 3 basic elements. The crystal oscillator provides a stable signal used as the reference to generate the high frequency signals; the Voltage Controlled Oscillator (VCO) generates the RF signals, and the Phase Locked Loop (PLL) compares the signal coming from the frequency reference to a divided version of the signal generated by the VCO, producing an output voltage that is proportional to this difference. This signal coming from the PLL is low pass filtered and used to control the

VCO. Through this mechanism, the output frequency of the VCO can be controlled by changing the division ratio of the PLL (N).

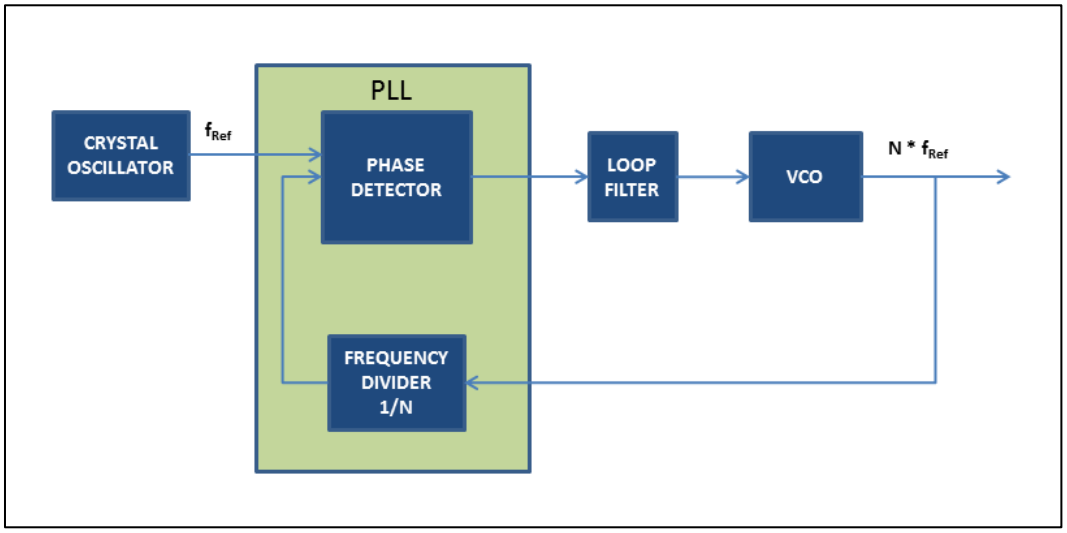


Figure 6: Synthesizer block diagram

The signal generated by the synthesizer is then filtered to improve its purity and divided through a splitter. One half of the signal is directed to a directional coupler that forwards the signal towards the DUT. The directional coupler is in charge of forwarding part of the power reflected by the DUT to the IQ mixer where it is mixed with the signal that comes from the second output of the splitter. The structure of an IQ mixer is shown in Figure 7. This component mixes the signal that is fed through the RF port with two versions of the signal fed through the LO port, one with a 0º phase delay and another one with an offset of 90º. Since in our case both the signal reflected by the DUT and the LO signal have the same frequency the I and Q outputs are just two DC signals that are proportional to the phase difference between the signal reflected by the DUT and the LO signal. We can represent this mixing process in a polar plot where each signal is represented by a vector whose length is proportional to the magnitude of the signal, and its angle corresponds to the signal's phase. Clearly, the mixing process is performing a projection of the signal reflected by the DUT to the x and y axis, represented by the in phase and quadrature versions of the LO. This process is shown in Figure 8. In this case the DC levels of the I and Q outputs correspond to the magnitude of the projections of the signal reflected by the DUT to the in-phase and quadrature LO signals. From these DC values the magnitude and phase of the reflected signal can be calculated with the following relations:

$$\rho = \sqrt{V_I^2 + V_Q^2}$$

$$\alpha = tan^{-1}\left(\frac{V_I}{V_Q}\right)$$

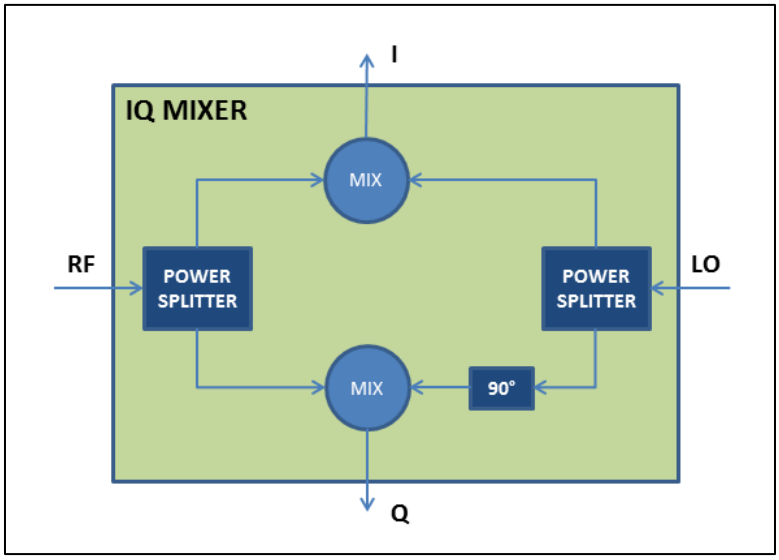


Figure 7: IQ mixer block diagram

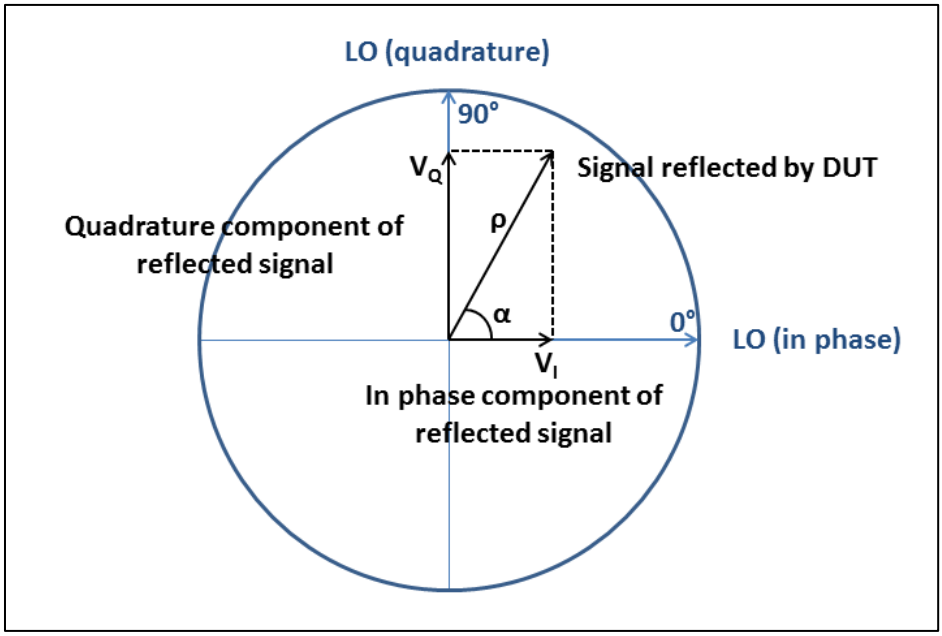


Figure 8: Polar representation of IQ mixing process

Once our approach has been presented, the high level requirements can be set. The requirements for the system are given in Table 1 and could be the requirements for a low performance VNA.

Table 1: System requirements

Requirement	**Value**	**Notes**
Measurement capability	S11, S21	Unidirectional
Size	< 5x5x5 cm	
Weight	< 100 g	
Data Interface	USB	Generic for smartphones/pc
Voltage supply	5 V	Compatible with USB
Power Consumption	< 400 mA	Will impact phone use
Amplitude measurement accuracy	< 0.5 dB	
Phase measurement accuracy	< 7.2 deg	2%
Amplitude minimum detectable signal	< -40 dB	Will drive ADC's dynamic range.
RF interface	SMA x2	Port 1 and Port 2
RF working band	400-4000 Mhz	This impacts in all subsystems requirements
RF output power	> -15 dBm	
RF output power control	Attenuation of up to 9 dB in steps of 3 dB	This requirement is based on Analog Devices PLL ADF4650 specs

Power LED	Yes	The Mini ARV will include a green power led that will lit when the devices is properly powered
Maximum measure time	< 100 ms per frequency	If resolution and frequency span are set to maximum the measurement time could become very large.

PREVIOUS WORK ON THE FIELD

As mentioned in the introduction there have been different efforts within the van der Weide research group to reduce the complexity of VNA's in order to provide a cheap and simple way to perform complex microwave measurements. In [3] a wideband reflectometer (1-12GHz) is presented, using a simple resistive power divider for the signal separation instead of a directional coupler. This change provides a compact solution as well as a wideband response, but with the cost of a reduced dynamic range, since the forward wave is leaked to the detector through the resistive power divider. Another approach was that one of A. Wangsanata [2] in which a planar VNA was designed. In this case the signal separation was realized on a single PCB (using planar directional couplers). The limitation of this approach lays on the band limitation of the planar directional couplers.

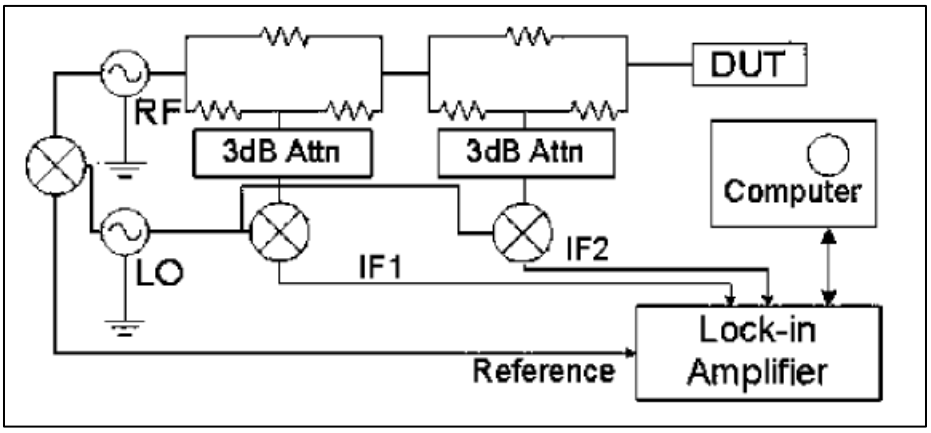


Figure 9: Block diagram of reflectometer proposed in [3]

Other efforts on the field have been done to develop VNAs at higher frequency bands. In [7] a compact one-port VNA was developed to work in the Ka-band (26.5-40GHz). This VNA relies on a phase shifter to be able to measure both the amplitude and the phase of the reflection coefficient. The addition of this shifter simplifies the architecture of the reflectometer but as a drawback, the measurement time is increased for each frequency point since it has to measure for different configurations of the phase shifter.

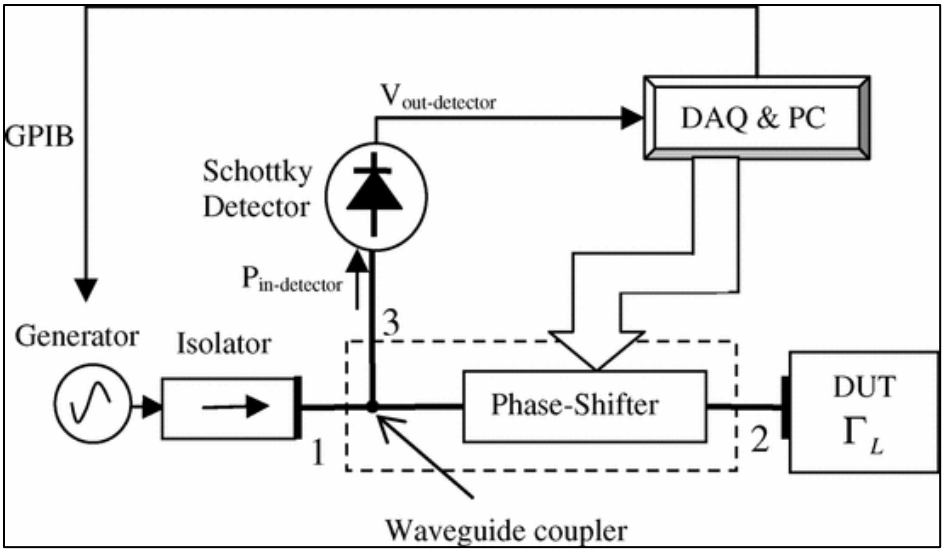


Figure 10: Block diagram of the reflectometer proposed in [7]

The same approach has been used in [8] to build a reflectometer for Complex permittivity measurement of biological material. In this case the reflectometer was integrated on a Silicon MEMS probe. The block diagram of the proposed reflectometer can be seen in Figure 11.

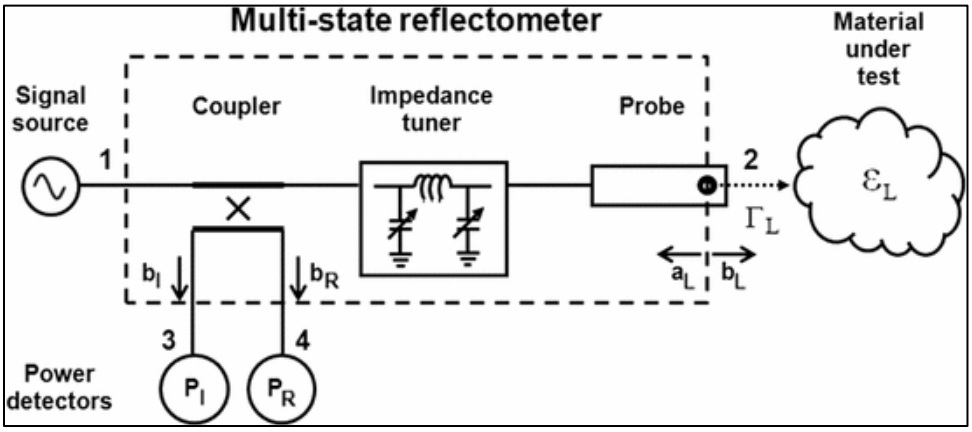


Figure 11: Block diagram of the reflectometer proposed in [8]

3. SIMULATIONS

The simulation task has been performed in two steps. First, an ideal circuit has been simulated to validate the approach. Once the approach has been validated the candidate components shall be chosen and characterized, introducing the characterization parameters of each component into the simulation model. With these more accurate parameters, the impact of each inaccuracy can be assessed, and the design can be refined to cope with these imperfections.

a. Ideal circuit simulation

Once the concept of the compact VNA has been exposed, the first task consists in validating the approach through simulation. In our case we have used the simulation tool ADS from Agilent [9]. This tool is of great use in RF design since it can perform different simulations to characterize an electronic circuit: S-parameters, Harmonic balance, transient simulations, etc. In our case we are going to perform a harmonic balance simulation, not considering the possible impairments in the components (Harmonic distortion, amplitude/phase imbalance, etc.).

The model used in the initial simulations is depicted in Figure 12. The simulation process is performed as follows: First a Harmonic balance simulation is run for each calibration load, saving the voltages corresponding to the I and Q outputs of the IQ mixer. Then, the unknown load is also measured and the voltages of the IQ mixer outputs saved. Finally the 4 sets of measurements are imported into Matlab, where the calibration is performed and the actual S11 parameters of the DUT are calculated. The code used for this purpose can be found in Appendix 1.

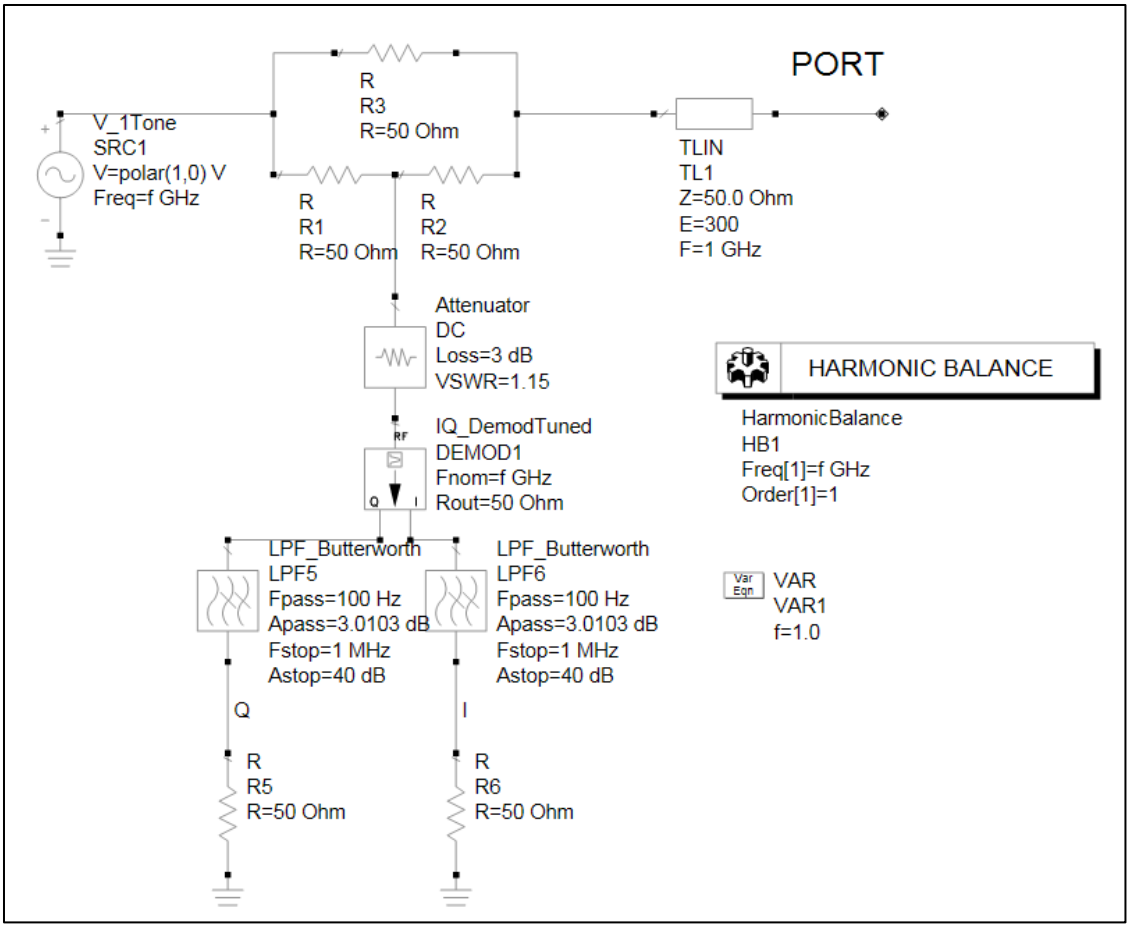


Figure 12: ADS ideal simulation model of proposed VNA

To validate the design a test load has been used to compare the results of the proposed approach with an S-parameter simulation. The load is shown in Figure 13, and it consists on a resonant network composed of inductors and capacitors.

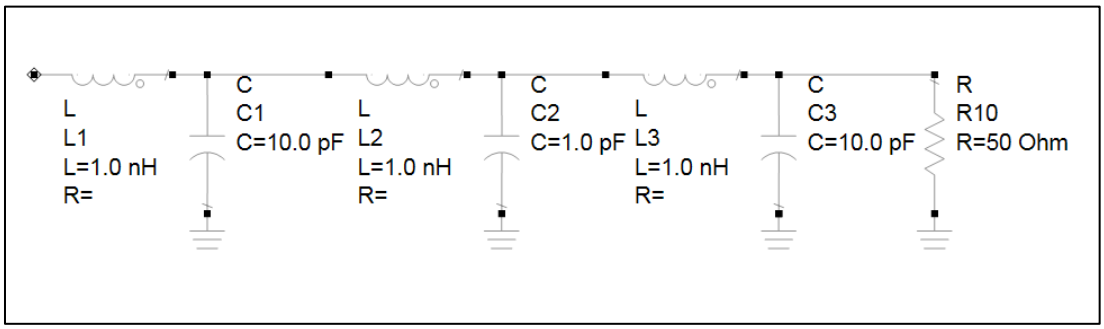


Figure 13: Test load

And its S11 response (magnitude and phase) is given in Figure 14:

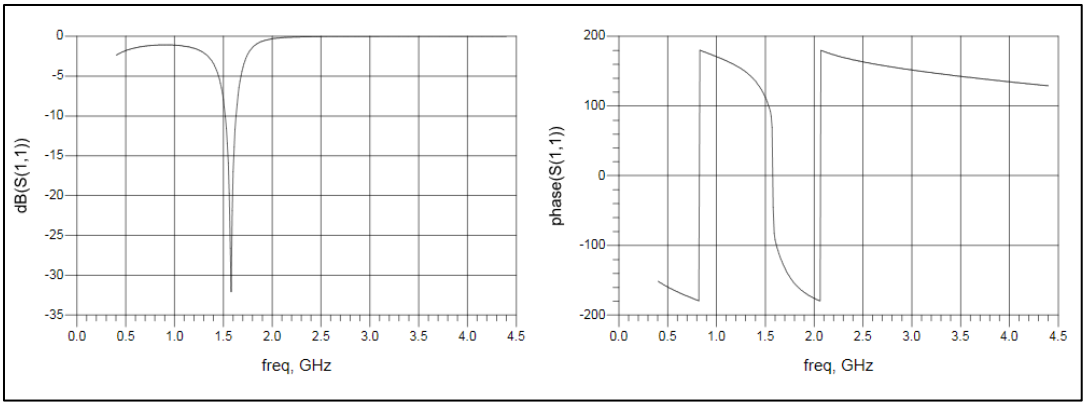


Figure 14: Test load's S11 magnitude and phase

Comparing the results of the actual S11 and the results of the compact VNA model:

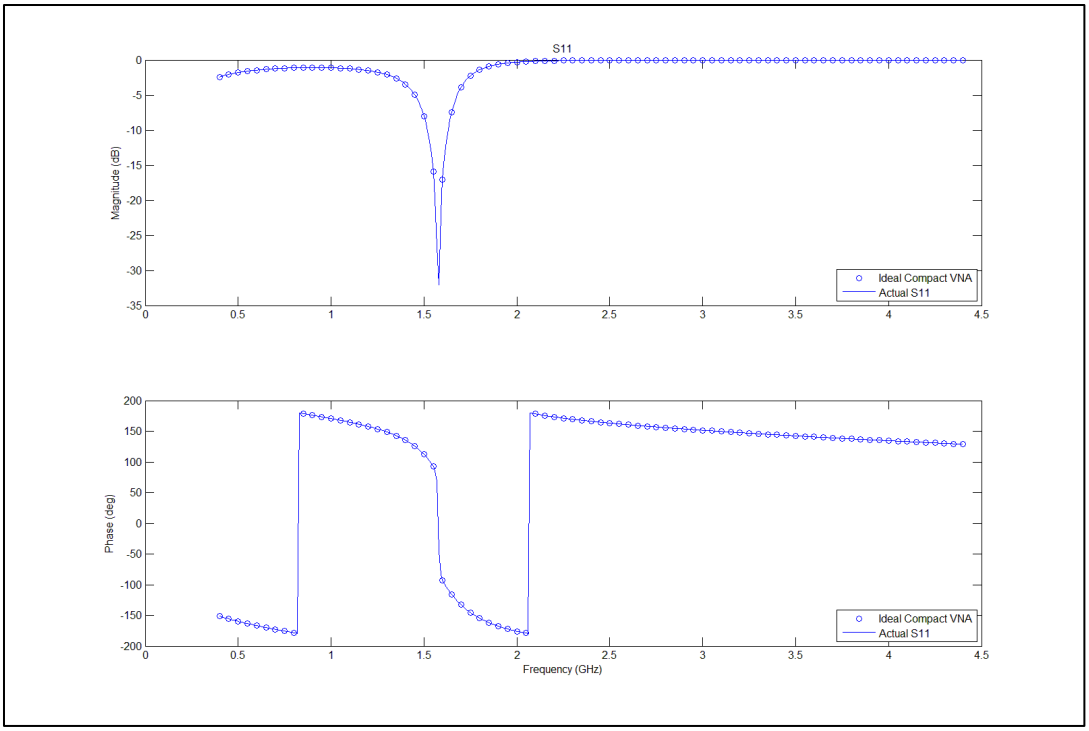


Figure 15: Phase and magnitude of S11. Model vs. Actual S11

Since the results totally match we can conclude that our approach is valid and we can continue with the implementation of our compact VNA.

b. Selection of components

With the approach validated we proceed to select the components that will be used to build the compact VNA. First we will focus on the synthesizer. The requirements for this component are listed in Table 2,

Table 2: Synthesizer requirements

Requirement	Value	Units
Fmin	≤ 400	MHz
Fmax	≥ 4000	MHz
Output power	≥ 0	dBm
Fstep	< 10	MHz
Tuning time	≤ 1	ms

In addition to the electrical requirements we have to select a component that minimizes the required size and power consumption since one of the objectives of the project is to build a compact VNA, therefore the component selected is the Wideband synthesizer ADF4350 from Analog Devices [10]. It can generate signals with frequencies ranging from 137.5 MHz up to 4.4 GHz and with up to 5 dBm of output power. It also integrates both the PLL and the VCO in a single chip, reducing the required pcb space. In the negative side the harmonic content of this synthesizer is rather high, with a 2^{nd} order harmonic at -20 dBc and 3^{rd} order harmonic at -10 dBc. The impact of these harmonics will be assessed in the next simulations and the filtering requirements for the low pass filters set.

The requirements for the IQ mixer are given in Table 3.

Table 3: IQ mixer requirements

Requirement	Value	Units
Fmin	≤ 400	MHz
Fmax	≥ 4400	MHz
Amplitude balance	< 0.2	dB
Phase balance	< 1	degree
Noise figure	< 15 (@ 3 GHz)	dB
P1dB	> 10	dBm
IP3	>25	dBm

For the IQ mixer the chosen component is the Analog Devices Quadrature Demodulator ADL5380 [11]. Its work bandwidth covers from 400 MHz to 6 GHz, with a noise figure < 11.7 dB and a 1 dB compression point of 11.6 dBm.

c. Real circuit simulation

With the components selected we proceed to introduce its parameters into the simulation model so that we can assess the impact of the different impairments on the VNA results. The ADS simulation model of the compact VNA that includes the real parameters of the components is shown in Figure 16. First we will evaluate the impact of the synthesizer and mixer harmonics. The second and third harmonics of the synthesizer and the three first beating products of the IQ mixer will be considered in this case. In Figure 17 the impact of the harmonics on the S11 measurement of the test load can be seen. Since the harmonics have a big impact on the results, degrading the performance of the VNA, they need to be filtered out. Successive simulations are performed reducing the level of the harmonics until the error falls below a given threshold.

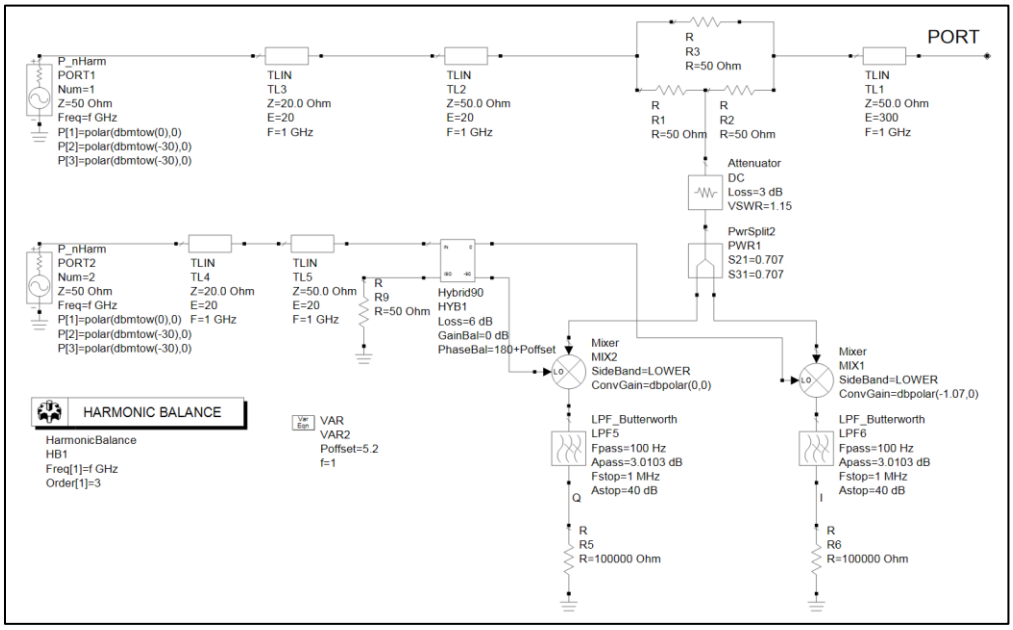


Figure 16: ADS real simulation model of proposed VNA

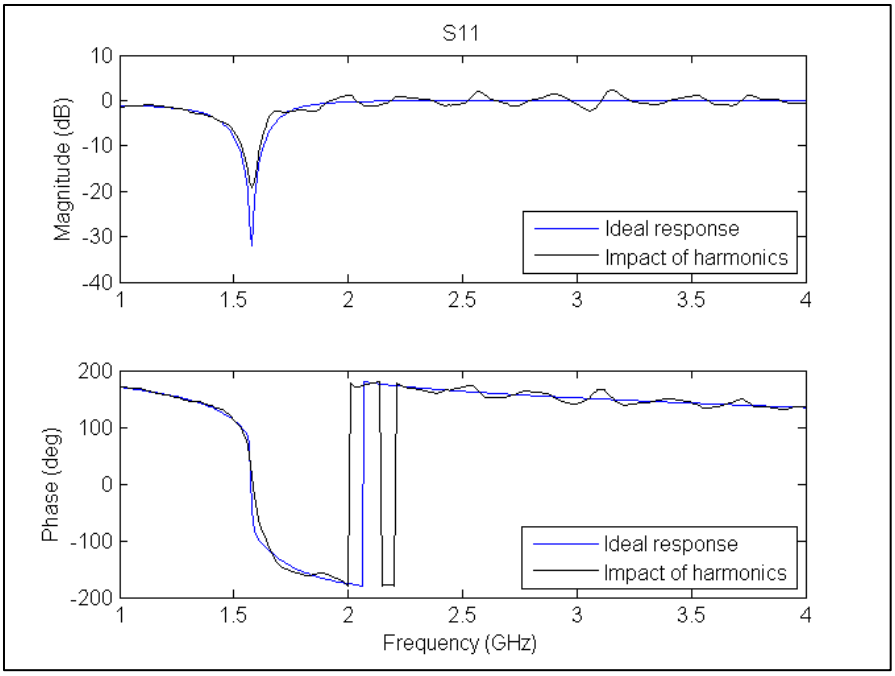


Figure 17: Impact of harmonics on the S11 measurement. Harmonics: 2nd -20 dBc, 3rd -10 dBc

In Figure 18 it can be seen that an attenuation of 10 dB for the 2nd harmonic and 20 dB for the 3rd harmonic are enough to keep the amplitude error below 0.1 dB and the phase error below 1 degree. These values will be the requirements for the filters that follow the synthesizer.

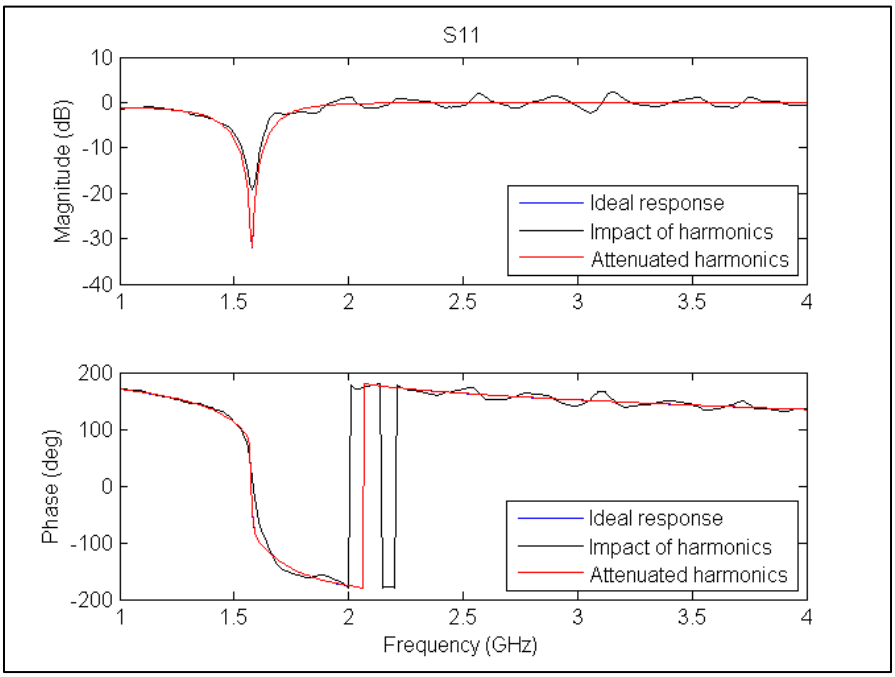


Figure 18: Impact of harmonics filtering. 2nd harmonic attenuated 10 dB. 3rd harmonic attenuated 20 dB.

Since the unwanted signals are the 2nd and 3rd harmonics the filters used will be Low pass filters. Also, since the total bandwidth of the VNA is larger than an octave, multiple filters will be needed to filter out the harmonics. Based on the filtering distribution proposed shown in Figure 19, four different filters will be needed. The requirements for each filter are provided in Table 4.

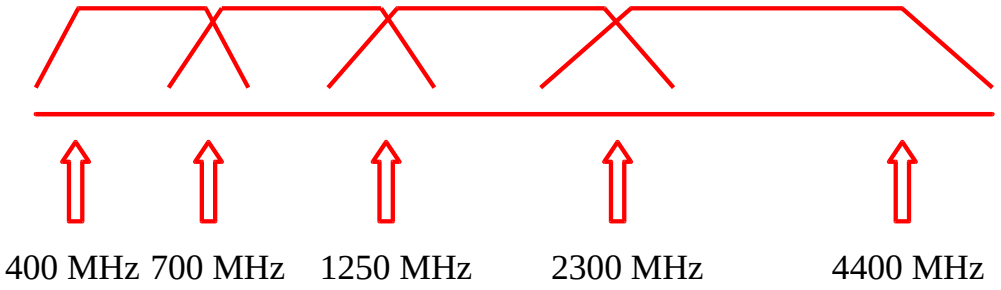


Figure 19: Proposed filtering bank

Table 4: Synthesizer filtering requirements

Filter	Cutoff frequency (-3 dB)	Frequency band (-10 dBc)	Frequency band (-20 dBc)	Insertion Loss	Passband ripple
Filter 1	700 MHz	800-1400 MHz	1200-2100 MHz	< 3 dB	< 3 dB
Filter 2	1250 MHz	1400-2500 MHz	2100-3750 MHz	< 3 dB	< 3 dB
Filter 3	2300 MHz	2500-4600 MHz	3750-6900 MHz	< 3 dB	< 3 dB
Filter 4	4400 MHz	4600-8800 MHz	6900-13200 MHz	< 3 dB	< 3 dB

Other impairments assessed through simulations are the amplitude and phase imbalance of the I and Q branches of the IQ mixer. Due to imperfections in the manufacturing process the insertion loss and phase delay of the two outputs of these mixers have a small offset that can impact the performance of the systems that use them. For the mixer that we are using we can find on its datasheet [11] a maximum amplitude imbalance of 0.07 dB and a maximum phase offset of 0.2 degree. Using these values in the mixer's component of the simulation model we obtain the results shown in Figure 20. Clearly these offsets don't have a noticeable impact in the final results. This is due to the fact that the calibration process compensates for these variations in the IQ mixer's response.

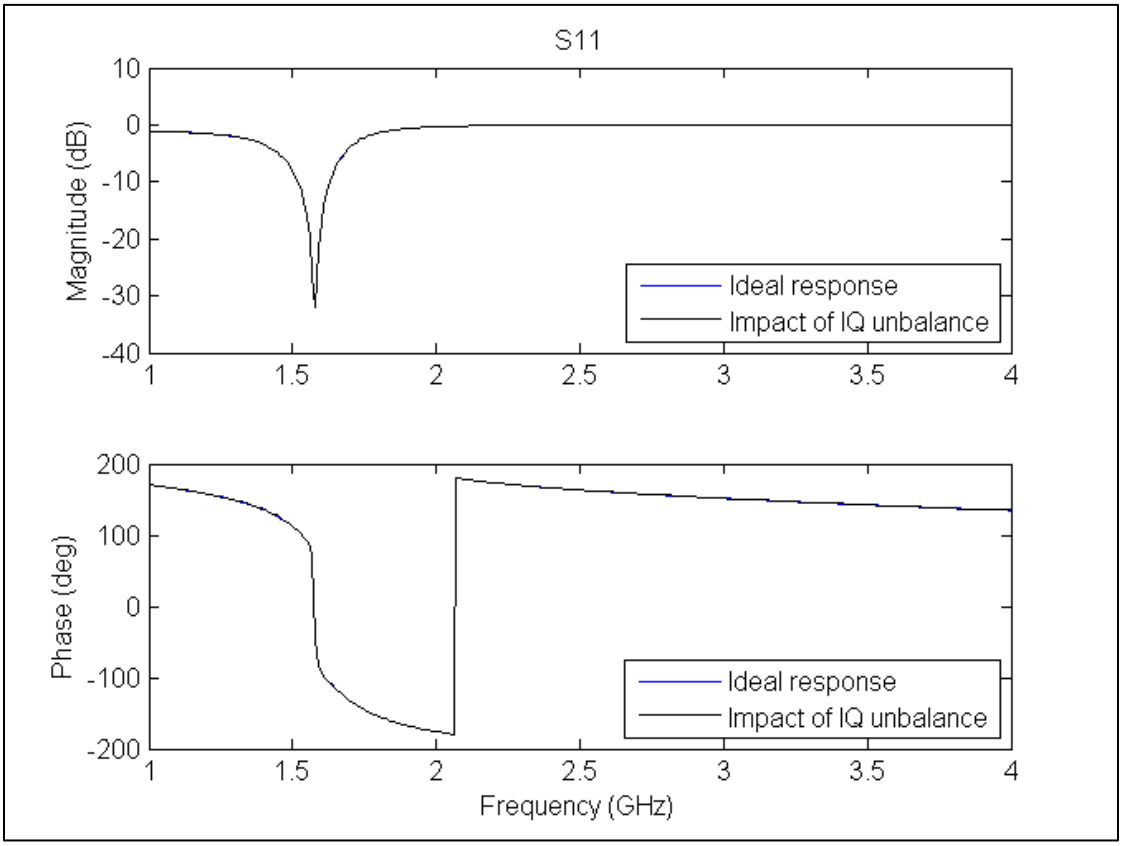


Figure 20: Impact of IQ mixer's amplitude and phase unbalances. Amplitude unbalance: 0.07 dB. Phase unbalance: 0.2 deg.

4. DESIGN

Once all the components to be used are chosen and the results validated through simulation we proceed to the design of both the hardware and the software parts of the compact VNA. At this point a decision has to be taken with regard to the use of a real smartphone for the prototype or simulate this element with a computer. Since the calibration algorithm has already been designed and tested using Matlab we have decided to implement the user interface and signal processing in a computer. Furthermore, we need to interface the synthesizer control and the IQ outputs to the computer. For the synthesizer we will use the 5007 Dual 137.5-4400MHz Frequency Synthesizer Module from Valon Technology [12]. This board includes two Analog Devices ADF4350 wideband synthesizers plus a crystal oscillator and a USB interface. The rest of hardware elements will be part of a custom pcb, while the ADCs will be part of a National Instruments Data Acquisition board PCI-MIO-16-XE-50 [13]. This board is connected to a PCI slot of the computer.

a. HARDWARE

The four different hardware elements that need to be considered are: Synthesizer board, Low pass filters, VNA PCB and Data Acquisition board.

The Synthesizer board will come from Valon Technology. In addition to the two ADF4350 frequency synthesizers it includes a microcontroller that offers a USB interface for the control of the synthesizers, a 10 MHz crystal oscillator and two output power amplifiers. It drains up to 170 mA at 5 V when one synthesizer is used.

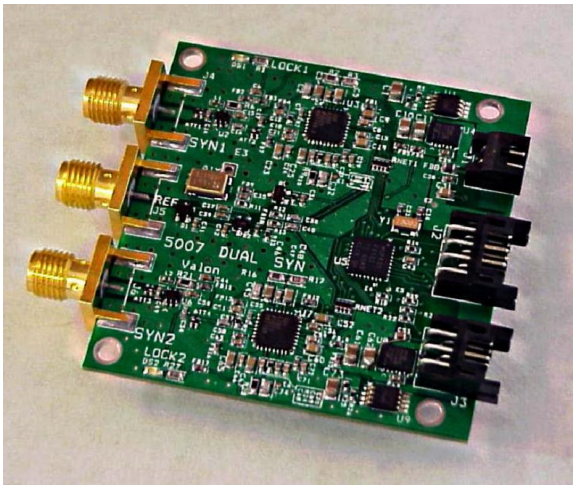


Figure 21: Dual 137.5-4400MHz Frequency Synthesizer Module from Valon Technology, LLC.

For the low pass filters two options have been considered. The first one consists on using ceramic compact low pass filters from Mini-Circuits, like the one in [14]. This filter has a cutoff frequency of 3950 MHz with 20 dB attenuation at 4300 MHz and a footprint of only 3.2x1.6 mm. The frequency response of this filter is shown in Figure 22. These filters are a good option, but their drawback is that they are offered only at a discrete number of cutoff frequencies that can be of use in some of our bands but not for the whole bandwidth. The second option consists on designing and building the filters using discrete inductors and capacitors. To validate this approach the 4th filter in Table 4 has been designed. Using the Filter Design tool from Nuhertz [15] a 5th order Chebyshev low pass filter has been designed and implemented using LC elements from Coilcraft and ATC. A picture of the realized filter and its measured S parameters are shown in Figure 23.

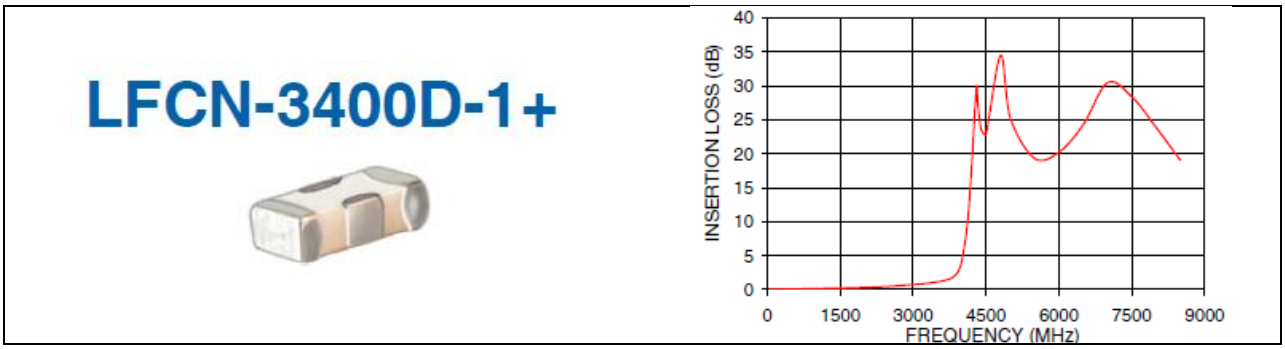


Figure 22: Ceramic low pass filter from Mini-Circuits. Component drawing (left) and frequency response (right).

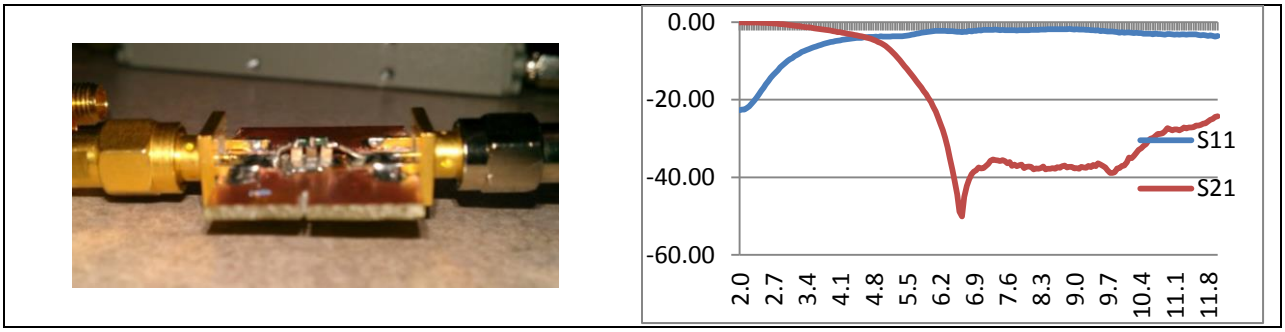


Figure 23: DC-4.3 GHz low pass filter. Picture of filter (left) and measured S parameters (right).

The radio PCB will hold the filters, switches, directional coupler, IQ mixer and RF connectors. This board has been designed using Altium Designer Summer 2009 [16]. The PCB schematic is shown in Figure 24, and the PCB design in Figure 25. The PCB has been designed using FR406 with a dielectric constant of 3.96 and a loss tangent of 0.015. It is a 4 layer PCB

with a total thickness of 62 mils and 1oz copper. The two internal layers are the ground plane and the power plane, while the top layer is used for routing the RF traces and the bottom layer is used for routing the control lines. Since we want a controlled impedance lines for the RF traces, the LineCalc tool from ADS is used to calculate their dimensions. The transmission lines used have a grounded coplanar waveguide structure because it improves the shielding of the circuit against interferences and reduces the unwanted radiation. The grounded coplanar waveguide structure is depicted in **Error! Reference source not found.**, and the calculated dimensions are:

Line width: 15 mil

Spacing: 10 mil

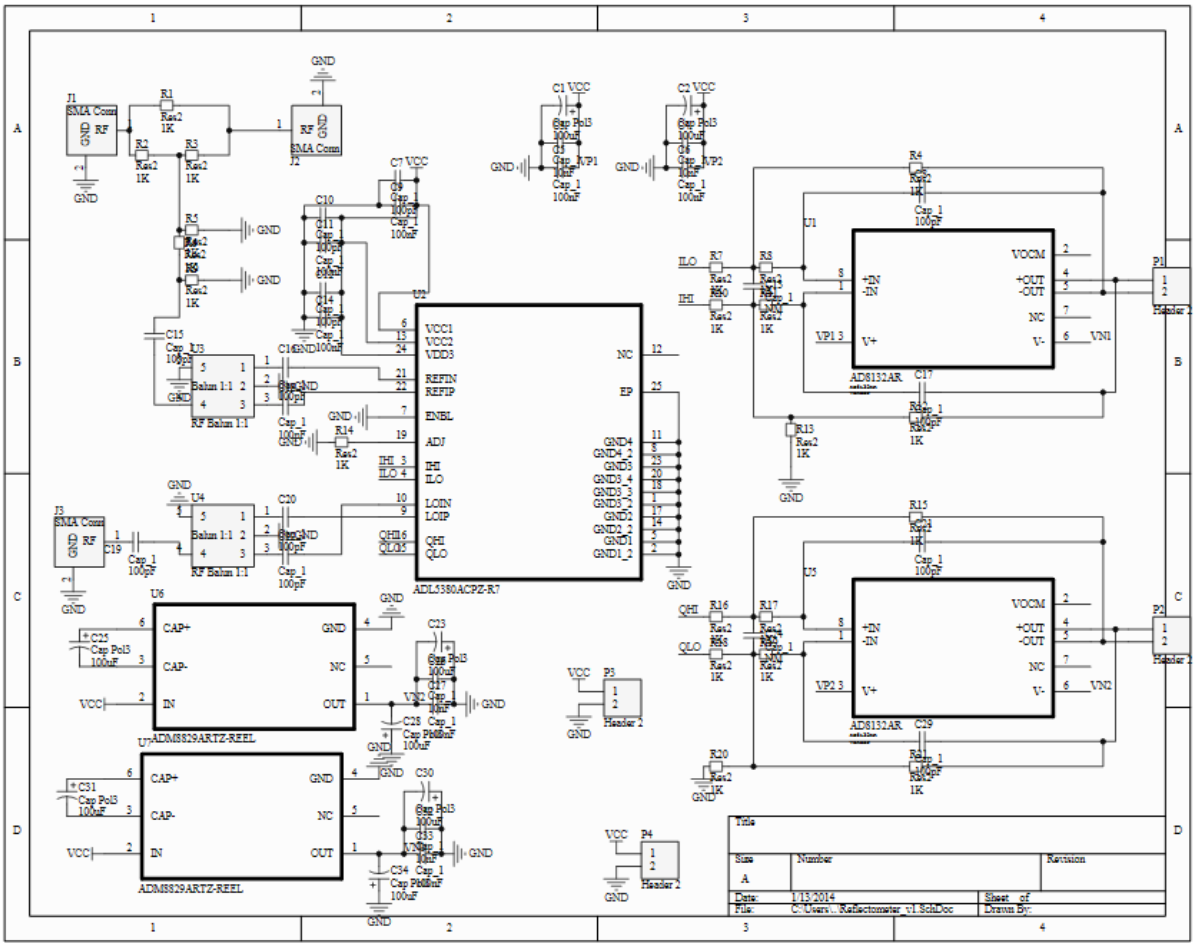


Figure 24: PCB schematic

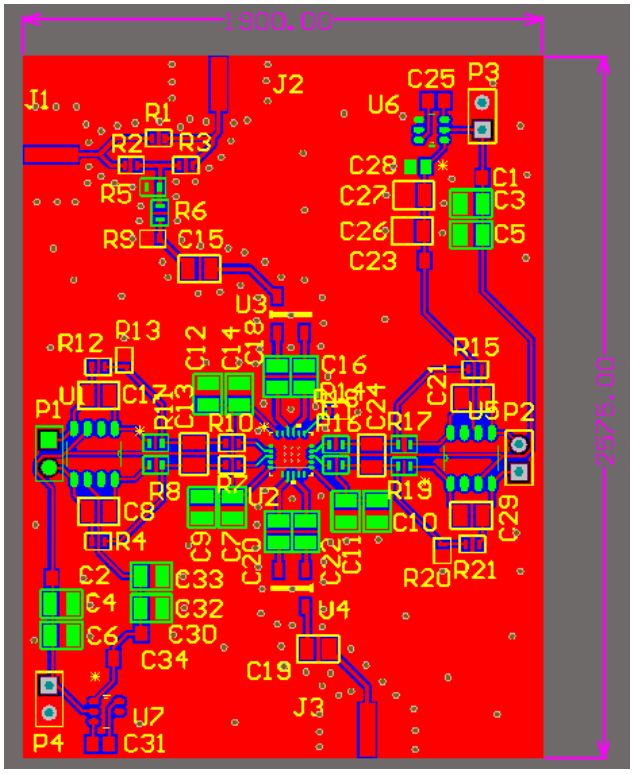


Figure 25: PCB design

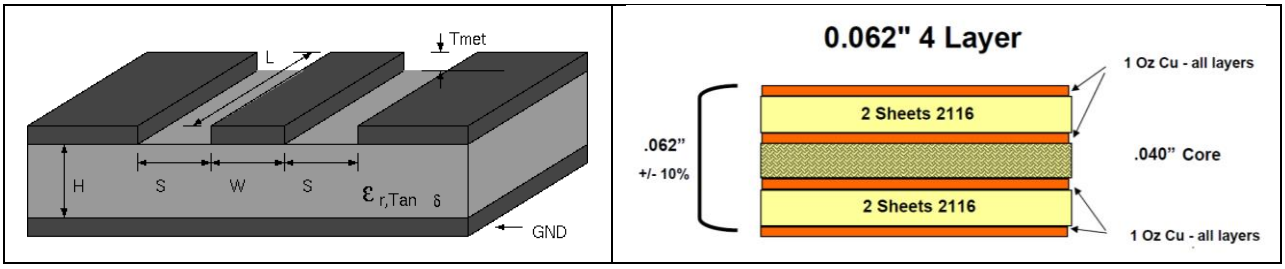


Figure 26: Grounded coplanar waveguide transmission line structure (left) and pcb stack (right).

The ADCs will be part of the National Instruments Data acquisition board PCI-MIO-16-XE-50 [13] that is connected to a PCI slot of the computer and will be in charge of sampling the I and Q outputs of the IQ mixer. Since these outputs are differential we will use the differential inputs of the data acquisition board. This board offers up to 8 fully differential 16 bit 20kSps ADCs. Moreover the DAQ board implements a conditioning stage to adapt the input signal to the full scale range of the ADC.

b. SOFTWARE

On the software side there are four main elements, the graphic user interface (GUI), the control of the synthesizer, the data capturing from the I and Q outputs of the mixer and the signal processing.

For the graphic user interface we have used the Matlab tool GUIDE. It is intended to design interfaces, easing the process of its creation. First, a design document is created, indicating the general structure of the GUI, the different windows and buttons, as well as the actions that should happen when every button is pressed. This document can be found in Appendix 2. The final GUI has a main window on the left, divided in half, with the upper half displaying the S11 magnitude and the lower half displaying the S11 phase. The control buttons are aligned in a column on the right. Some captures of the final GUI are given in Figure 27.

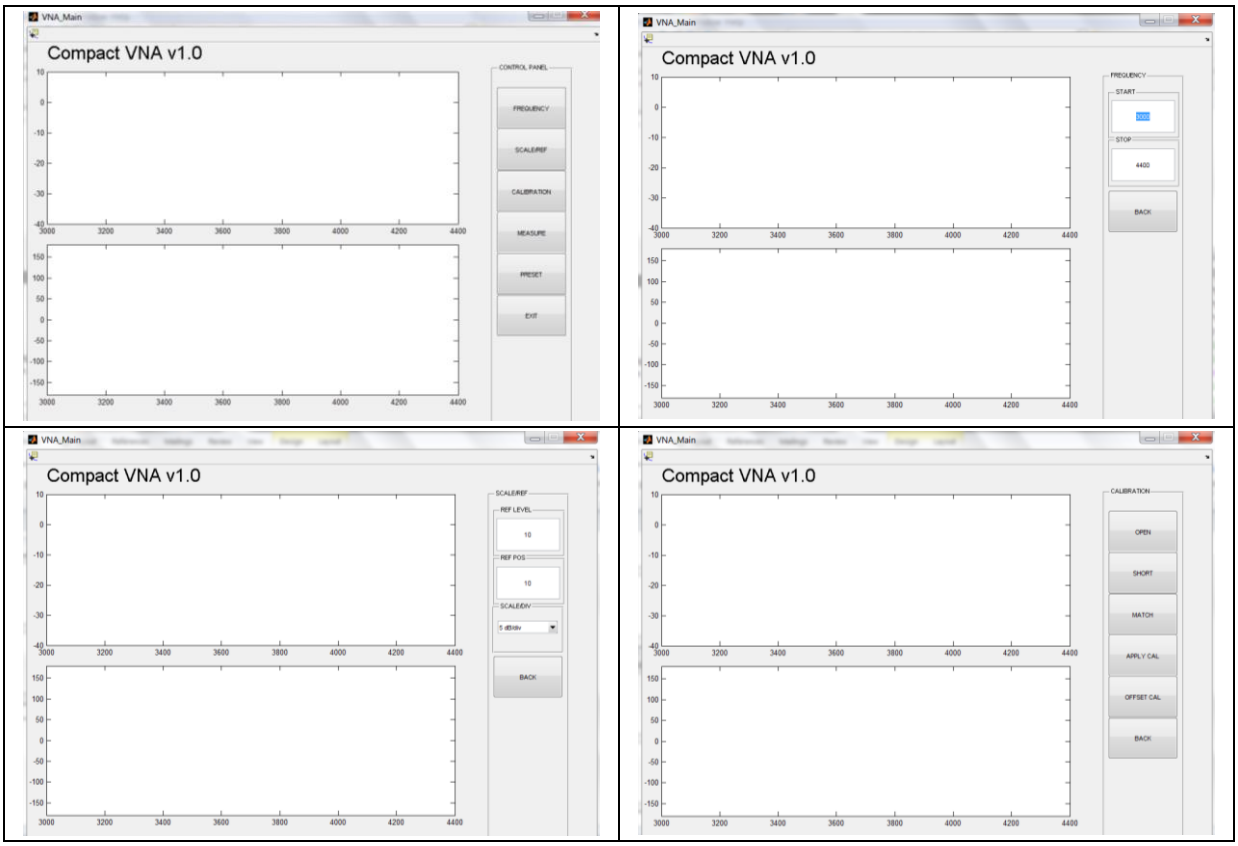


Figure 27: VNA GUI Screenshots

Once the GUI is created it needs to be able to communicate with the synthesizer to retrieve its configuration and be able to change its parameters. For this purpose a set of Matlab functions have been programmed. The functions' names and their description are given in Table 5 and their code is given in Appendix 3.

Table 5: Synthesizer control functions

Function	**Description**	**Parameters**
Getoptions()	Read the current options of the synthesizer.	@param[out] opts Receives the options. @return True on successful completion.
Getfrequency()	Read the current frequency being generated by the synthesizer.	@param[out] frequency Receives the frequency in MHz. @return True on succesful completion.
Get_vco_range()	Read the current range of the VCO.	@param[out] vcor Receives the current VCO extent. @return True on successful completion.
GetEPDF()	Calculate effective phase detector frequency	
Set_options()	Set the options of the synthesizer.	@param[in] opts Structure holding the new options. @return True on successful completion.
Set_vco_range()	Set the range of the VCO. This affects the allowable frequency range of the output.	@param[in] vcor The extent to set. @return True on successful completion.
Set_frequency()	Set the synthesizer to the desired frequency, or best approximation based on channel spacing.	@param[in] frequency The desired output frequency in MHz. (The range is determined by the minimum and maximum VCO frequency. @param[in] chan_spacing The "resolution" of the synthesizer.

With the GUI and the functions to configure the synthesizer done we need to create the code to get the data from the IQ mixer through the ADCs. This code has a main loop that, for every frequency in the frequency set, configures the synthesizer and obtains a reading of the I and Q outputs of the mixer. This code is given in Appendix 4.

Finally, once the data for every frequency is captured the software needs to apply the calibration by using the same calibration algorithm used in the simulations and provided in Appendix 1.

5. RESULTS

The lab setup to test the VNA prototype is shown in Figure 28. It consists on a computer that runs the VNA software, a power supply, the synthesizer board, RF board, DAQ interface board and DUT.

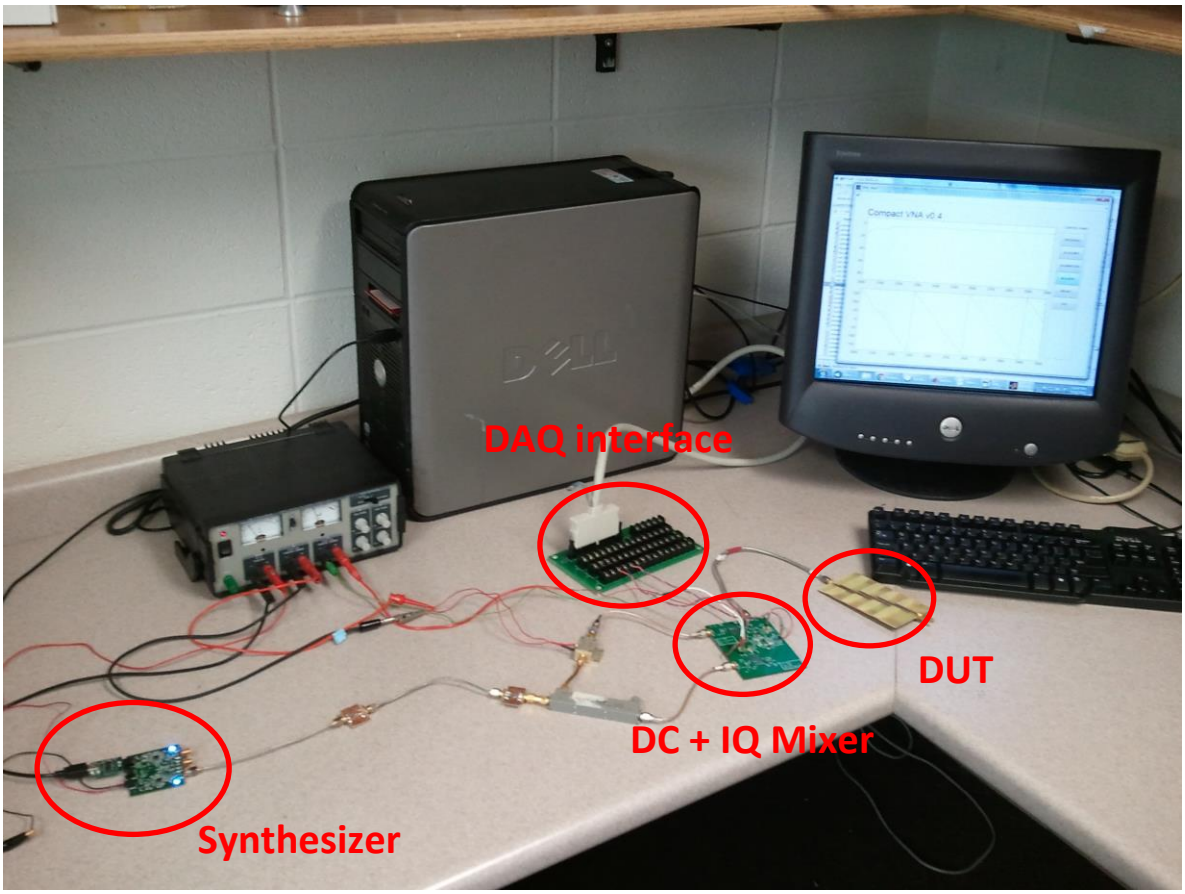


Figure 28: Lab test setup of Compact VNA Prototype

Multiple loads have been used to test the prototype. First a 50 Ohm load has been used to check the noise floor of the Compact VNA. The magnitude of the reflection coefficient of a 50 Ohm load is ideally -∞ dB, therefore any value measured for the S11 will be due to noise. The Magnitude measurement of the S11 parameter of a 50 Ohm load is shown in Figure 29.

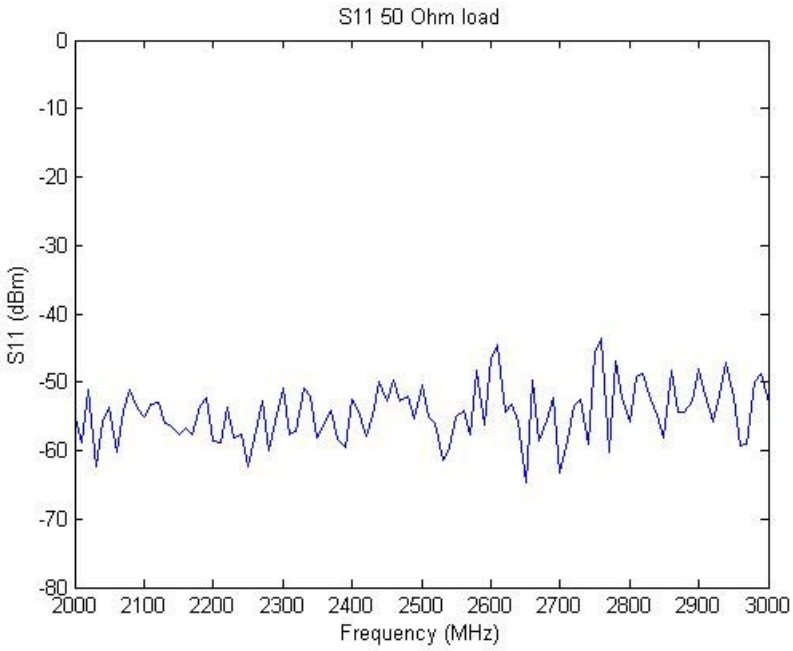


Figure 29: S11 magnitude of a 50 Ohm load

In high end VNA's this value can range from -50 dB (using an electronic calibration kit) to below -80 dB for a mechanical calibration kit. Hence our prototype's performance is more than acceptable for a low cost, compact VNA.

Next we are using three different loads to compare the S11 results from our prototype with a commercial VNA. The commercial VNA used is an N5232A PNA-L Microwave Network Analyzer from Agilent [17]. The S11 phase and magnitude measurements of the first load are given in Figure 30.

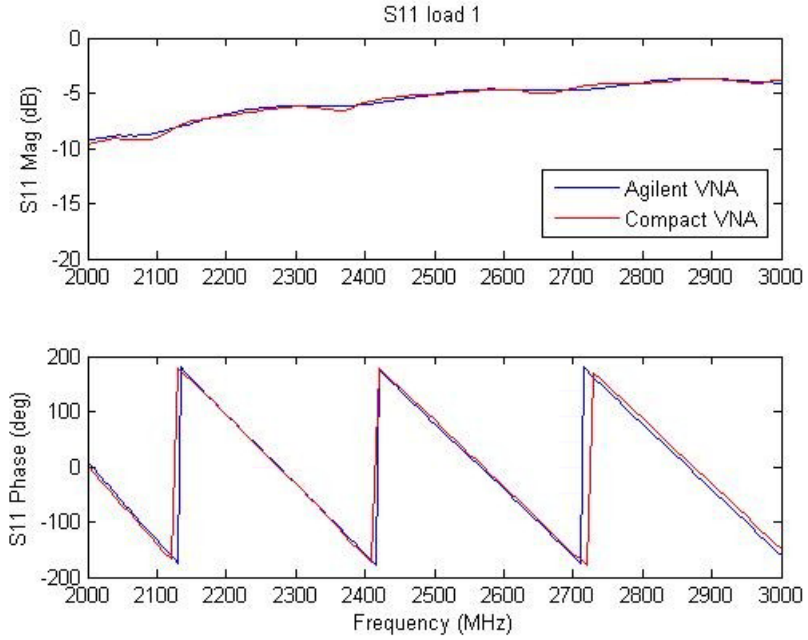


Figure 30: S11 of load 1 measured with the proposed compact VNA and the Agilent VNA

The S11 phase and magnitude measurements of the second and third loads are given in Figure 31 and Figure 32 respectively.

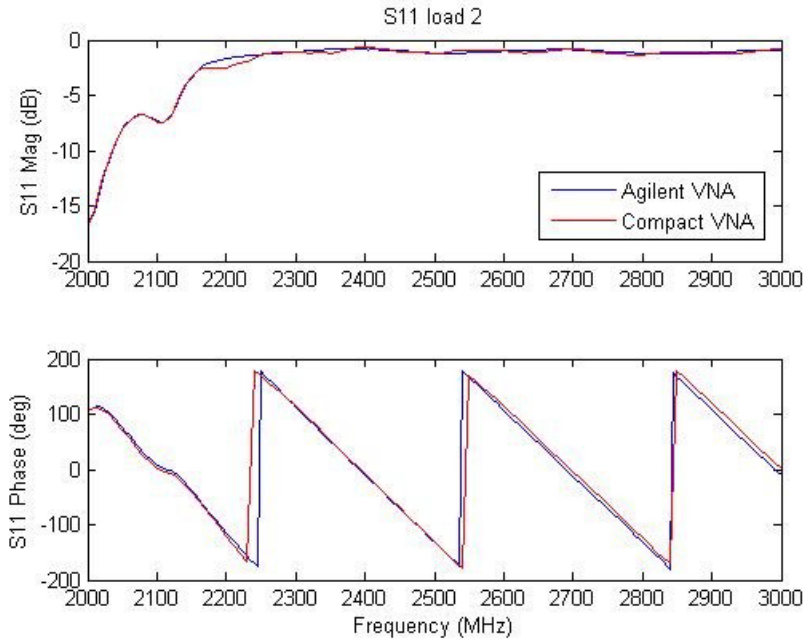


Figure 31: S11 of load 2 measured with the proposed compact VNA and the Agilent VNA

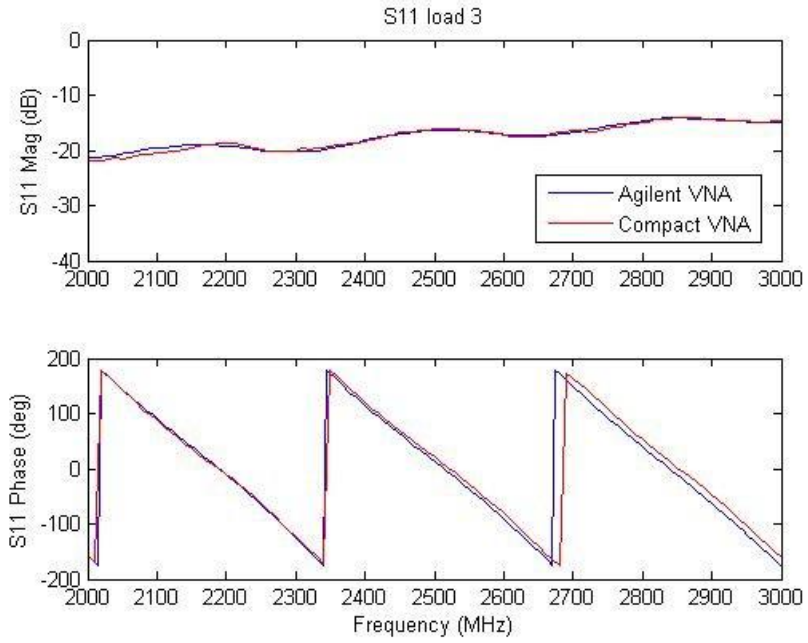


Figure 32: S11 of load 3 measured with the proposed compact VNA and the Agilent VNA

A summary of the magnitude and phase errors is given in Table 6.

Table 6: Summary of magnitude and phase errors

Load	**Magnitude max. error (dB)**	**Magnitude mean error (dB)**	**Phase max. error (deg)**	**Phase mean error (deg)**
Load 1	0.48	0.19	19	10
Load 2	0.85	0.15	13	6
Load 3	0.91	0.31	17	7

Finally a high pass filter with a deep null at 3.77 GHz is also measured. The magnitude and phase measurements are given in **Figure 33** and **Figure 34**.

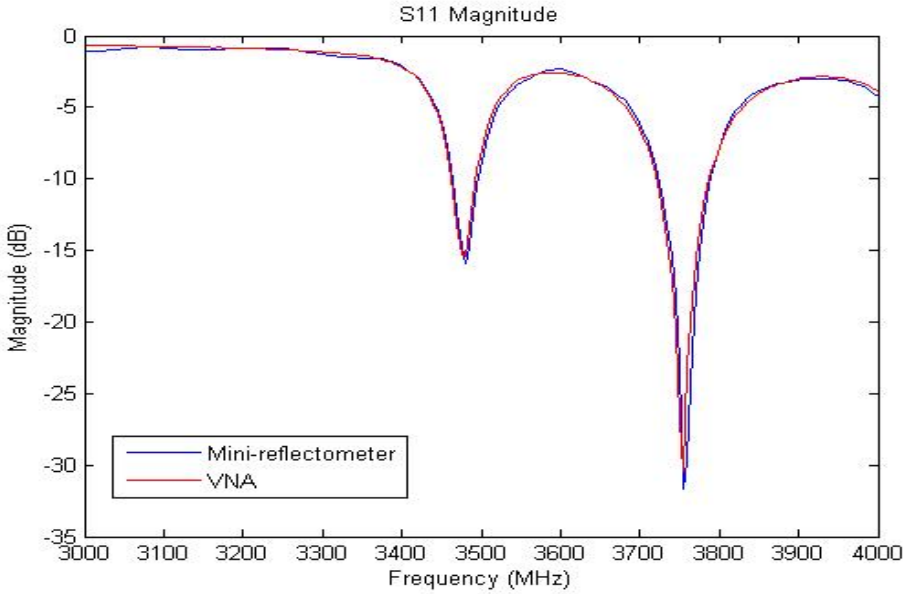


Figure 33: S11 magnitude of high pass filter measured with the proposed compact VNA and the Agilent VNA

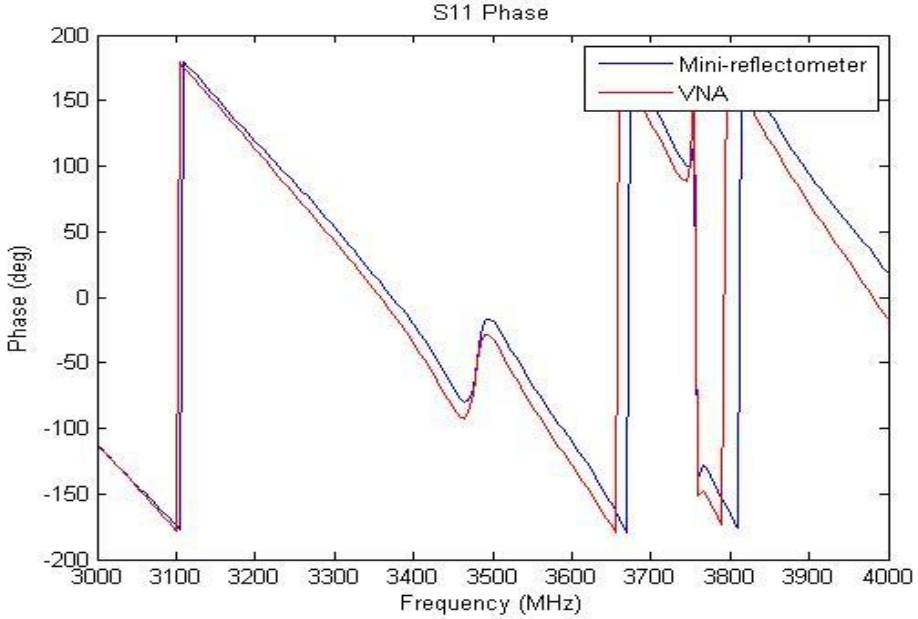


Figure 34: S11 phase of high pass filter measured with the proposed compact VNA and the Agilent VNA

6. CONCLUSIONS

In this work we have proposed, designed and prototyped a compact VNA that provides the capability of measure the phase and magnitude of the reflection coefficient of a load for the frequency range between 137.5 and 4400 MHz. The performance of the compact VNA has been compared to a commercial VNA from Agilent, verifying that the errors are kept below 1 dB for the magnitude and below 20 degrees for the phase (< 6%), showing that the results of the built prototype are in accordance to the simulations' results.

The proposed compact VNA could be used for device and materials' characterization in home projects, education or new portable medical devices.

Future lines of work include the integration of the designed hardware to a smartphone, the investigation of integrating the whole system on a single chip, or develop different applications for the compact VNA.

7. REFERENCES

[1] [Online]. Available: www.oscium.com.

[2] A. Wangsanata, "A planar vector network analyzer," Univ. of Wisconsin, Madison, 2002.

[3] M. K. Choi, M. Zhao, S. C. Hagness and D. W. van der Weide, "Compact Mixer-Based 1–12 GHz Reflectometer," *IEEE Microwave and wireless components letters,* vol. 15, no. 11, 2005.

[4] D. M. Pozar, Microwave Engineering, John Wiley & Sons, Inc., 2012.

[5] Agilent Technologies, "Network Analyzer Basics," 31 Auguts 2004. [Online]. Available: http://cp.literature.agilent.com/litweb/pdf/5965-7917E.pdf. [Accessed 16 November 2012].

[6] B. A. Brown, "Solving the One-Port Calibration Equations," 2004. [Online]. Available: http://na.tm.agilent.com/vnahelp/tip20.html. [Accessed 15 August 2013].

[7] M. Fallahpour, M. Baumgartner, A. Kothari, M. Ghasr, D. Pommerenke and R. Zoughi, "Compact Ka-Band One-Port Vector Reflectometer Using a Wideband Electronically Controlled Phase Shifter," *Instrumentation and Measurement, IEEE Transactions on,* vol. 61, no. 10, pp. 2807,2816, Oct. 2012.

[8] K. Kim, N. Kim, S.-H. Hwang, Y.-K. Kim and Y. Kwon, "A Miniaturized Broadband Multi-State Reflectometer Integrated on a Silicon MEMS Probe for Complex Permittivity Measurement of Biological Material," *Microwave Theory and Techniques, IEEE Transactions on,* vol. 61, no. 5, pp. 2205,2214, May 2013.

[9] Agilent Technologies, "Advanced Design System (ADS)," [Online]. Available: http://www.home.agilent.com/en/pc-1297113/advanced-design-system-ads?&cc=ES&lc=eng. [Accessed 05 11 2012].

[10] Analog Devices, "ADF4350 Datasheet," [Online]. Available: http://www.analog.com/static/imported-files/data_sheets/ADF4350.pdf. [Accessed 15 04 2012].

[11] Analog Devices, "ADL5380 Datasheet," [Online]. Available: http://www.analog.com/static/imported-files/data_sheets/ADL5380.pdf. [Accessed 06 03 2012].

[12] Valon Technology, LLC, "Dual 137.5-4400MHz Frequency Synthesizer Module," [Online]. Available: http://valontechnology.com/5007%20synthesizer.html. [Accessed 04 08 2012].

[13] National Instruments, Inc., "NI PCI-MIO-16XE-50," [Online]. Available: http://sine.ni.com/nips/cds/view/p/lang/es/nid/1069. [Accessed 13 10 2012].

[14] Mini-Instruments, "LFCN-3400D-1+," [Online]. Available: http://217.34.103.131/pdfs/LFCN-3400D-1+.pdf. [Accessed 14 10 2012].

[15] Nuhertz Technologies, LLC, "Filter Design," [Online]. Available: http://www.nuhertz.com/. [Accessed 20 10 2012].

[16] Altium Limited, "Altium Designer," [Online]. Available: http://www.altium.com/en/products/altium-designer. [Accessed 15 10 2012].

[17] Agilent, "N5232A PNA-L Microwave Network Analyzer," [Online]. Available: http://www.home.agilent.com/en/pd-2139627-pn-N5232A/pna-l-microwave-network-analyzer?nid=-536902643.1017780&cc=ES&lc=eng. [Accessed 18 11 2012].

8. APPENDICES

a. Appendix 1

Matlab code for calibration

```
%This code loads the calibration results generated on ADS by the
%reflectometer model (stored in the files calMatch.csv, calOpen.csv and
%calShort.csv), calculates the ABC calibration parameters and applies them
%to the unknown load measurements (stored in measure.csv).
%
% The factors analyzed include: generator mismatch, LO mismatch and IQ
% demodulator impairments (level and phase)
%
% See also: http://www.vnahelp.com/tip20.html
%
% Created and tested on Matlab R2008a

% $Author: Marcos Martinez $ $Date: 2012/06/01 $ $Revision: 5.0 $
% Copyright: van der Weide Lab 2012

clear all
close all


%Step 1: Load calibration data
path=strcat('D:\Dropbox\ADS_Projects\Mini_Instruments_prj\Measures\ideal
circuit\match_i.csv');
fileID = fopen(path);
C = textscan(fileID, '%f32 %f32 %f32','delimiter', ',','Headerlines', 16);
fclose(fileID);
calMatch=[C{1}, C{2}, C{3}];

path=strcat('D:\Dropbox\ADS_Projects\Mini_Instruments_prj\Measures\ideal
circuit\open_i.csv');
fileID = fopen(path);
C = textscan(fileID, '%f32 %f32 %f32','delimiter', ',','Headerlines', 16);
fclose(fileID);
calOpen=[C{1}, C{2}, C{3}];

path=strcat('D:\Dropbox\ADS_Projects\Mini_Instruments_prj\Measures\ideal
circuit\short_i.csv');
fileID = fopen(path);
C = textscan(fileID, '%f32 %f32 %f32','delimiter', ',','Headerlines', 16);
fclose(fileID);
calShort=[C{1}, C{2}, C{3}];

%Step 2: calculate calibration parameters
[ ABC ] = refcal( calMatch, calOpen, calShort );

%Step 3: Load measured data
path=strcat('D:\Dropbox\ADS_Projects\Mini_Instruments_prj\Measures\ideal
circuit\load_i.csv');
fileID = fopen(path);
```

```
C = textscan(fileID, '%f32 %f32 %f32','delimiter', ',','Headerlines', 16);
fclose(fileID);
measure=[C{1}, C{2}, C{3}];
f=measure(:,1);

%Step 4: Calculate actual reflection coefficient based on the measured data
[ gactual ] = refmeas( measure, ABC );

%Step 5: Plot magnitude and phase of the calculated reflection coeff:
subplot(2,1,1)
plot(f,20*log10(abs(gactual)),'o');
title('S11')
ylabel('Magnitude (dB)')
hold
subplot(2,1,2)
plot(f,angle(gactual).*180/pi,'o')
ylabel('Phase (deg)')
xlabel('Frequency (GHz)')
hold

% S11 parameter of the load as computed with the S-param simulation in ADS

path=strcat('D:\Dropbox\ADS_Projects\Mini_Instruments_prj\Measures\ideal
circuit\S11.csv');
fileID = fopen(path);
C = textscan(fileID, '%f32 %s %f32 %f32','delimiter', ',/','Headerlines',
12);
fclose(fileID);
mag=C{3};
phase=C{4};
f=C{1};
flab=C{2};
x = strmatch('GHz', flab);
f1=f(1:x(1)-1)./1000;
f2=f(x(1):length(f));
f=[f1;f2];
subplot(2,1,1)
plot(f,20*log10(mag));
legend('Ideal Compact VNA', 'Actual S11','Location', 'SouthEast')
subplot(2,1,2)
plot(f,phase);
legend('Ideal Compact VNA', 'Actual S11','Location', 'SouthEast')
```

/----------/

```
function [ ABC ] = refcal( calMatch, calOpen, calShort )
%REFCAL calculates the calibration parameters a, b and c from the measured
%and actual reflection coefficients of three known loads.
%
% [ ABC ] = refcal( calMatch, calOpen, calShort )
%
% calMatch: matrix with the I and Q measurements for each frequency with a
% matched load.
% calOpen: matrix with the I and Q measurements for each frequency with an
% open load
```

```
% calShort: matrix with the I and Q measurements for each frequency with a
% short load
%
% The structure of these inputs has to follow these guides: first column is
% the frequency, second column is the I measure and third column is the Q
% measure. All input matrices must have the same size.
%
% ABC: matrix with the calibration parameters a, b and c corresponding to
% each frequency. First column corresponds to a values, second to b and
% third to c.
%
% See also: http://www.vnahelp.com/tip20.html
%
% Created and tested on Matlab R2008a

% $Author: Marcos Martinez $ $Date: 2013/06/01 $ $Revision: 0.1 $
% Copyright: van der Weide Lab 2012


gMatch=0;
gOpen=1;
gShort=-1;
ABC=zeros(length(calMatch),3);
for i=1:length(calMatch)
    gm1=calMatch(i,2)+j*calMatch(i,3);
    gm2=calOpen(i,2)+j*calOpen(i,3);
    gm3=calShort(i,2)+j*calShort(i,3);
    A=[gMatch, 1, -gMatch*gm1;gOpen, 1, -gOpen*gm2;gShort, 1, -gShort*gm3];
    B=[gm1;gm2;gm3];
    X=linsolve(A,B);
    ABC(i,:)=X.';
end


end
```

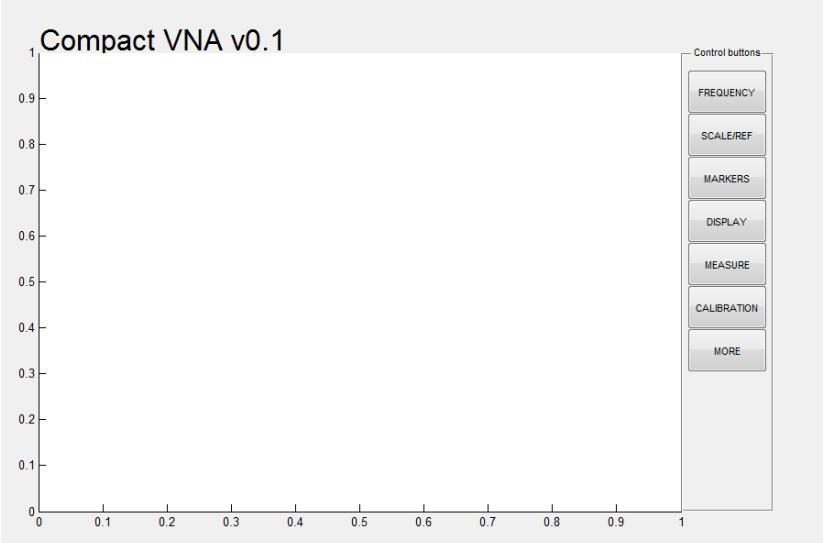
/----------/

```
function [ gactual ] = refmeas( meas, ABC )
%REFMEAS obtains the actual reflection coefficient from the measured
%reflection coefficients (in meas) and the calibration parameters (in ABC).
%The structure of these inputs has to follow these guides: first column is
%the frequency, second column is the I measure and third column is the Q
%measure.
%
% [ gactual ] = refmeas( meas, ABC )
%
% meas: matrix with the I and Q measurements for each frequency of the
% unknown load.
% ABC: matrix with the calibration parameters a, b and c corresponding to
% each frequency. First column corresponds to a values, second to b and
% third to c. This matrix is generated by refcal.m
%

%
```

```
% See also: http://www.vnahelp.com/tip20.html
%
% Created and tested on Matlab R2008a

% $Author: Marcos Martinez $ $Date: 2013/06/01 $ $Revision: 0.1 $
% Copyright: van der Weide Lab 2012

gactual=zeros(length(meas),1);
for i=1:length(meas)
    gm=meas(i,2)+j*meas(i,3);
    a=ABC(i,1);
    b=ABC(i,2);
    c=ABC(i,3);
    ga=(gm-b)/(a-gm*c);
    gactual(i)=ga;
end


end
```

b. Appendix 2
GUI Design document

General GUI structure:



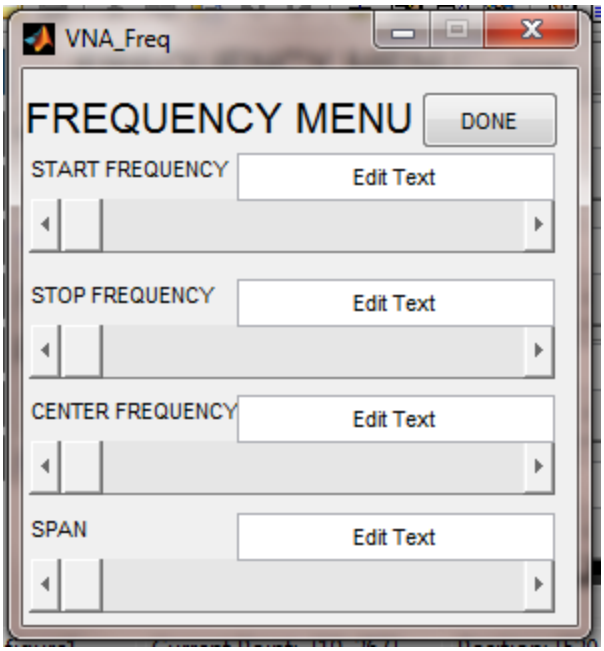
Inspired in commercial VNAs. Main window on the left with control buttons on the right.

When pressing a button change the layout of the right buttons (and keep a back button at the end). Things needed: 1) change panel after pressing a button. 2) Update plot after updating a parameter in the right panel. 3) Open a window with a file search option to open/save parameters. 4) Display an Error/Ok window to tell the user that the action worked or not. 5) Plot format as in a network analyzer.

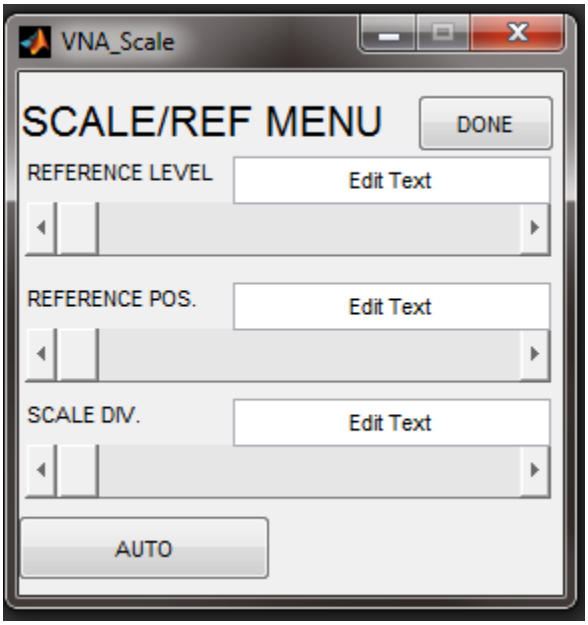
First focus on S11. Later we can think of a way to measure S22 and update the GUI to support this.

Actions:

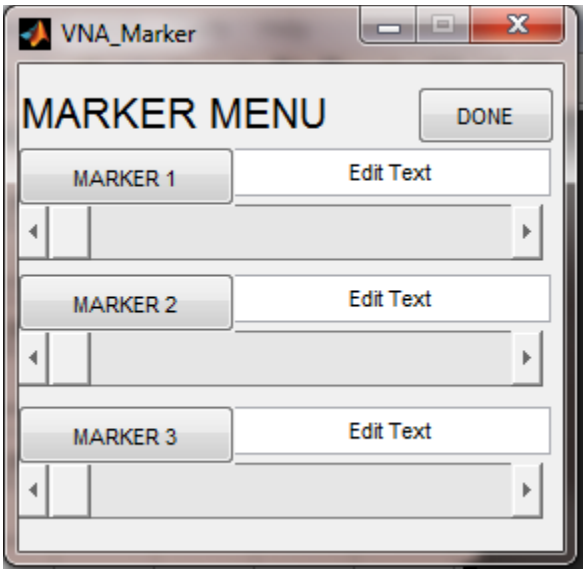
- At the GUI initialization
    - Verify connection to the Synthesizer and NI DAQ (explore if the Synth connection can be found or if the user will have to set it manually).
    - Retrieve information from the last time the app was used (in cfg.txt). (If this file doesn't exist the app will create one and set the main parameters to their default value):
        - Start freq (default: 138MHz)
        - Stop freq (default: 4.4GHz)
        - Ref level (default: 10dBm)
        - Ref position (default: top of graph)
        - Scale/div (default: 10dB/div)
        - ~~Measure (S11, S21, both) (default: S11)~~
        - ~~Display (default: data, single shot)~~
        - *Calibration (the calibration currently used will be saved in Currentcal.cal always, and that is the calibration that will be used after initialization).
        - Markers (default: no markers)
    - Update variables and display with retrieved info.
- When FREQUENCY button is pressed:
    - Launch new window (smaller) with:
        - Fstart: a box with the actual frequency in numbers (editable) and a slide bar that can be changed with the mouse (it will go in fixed steps of 1MHz or something like that)
        - Fstop: similar to fstart
        - Center freq: similar to Fstart
        - Span: similar to Fstart
        - Values displayed come from cfg.txt (at this point will be probably already in their variables).
        - When pressed DONE, update the variables and the cfg.txt too.



- When SCALE/REF button is pressed:
  - Launch new window (smaller) with:
    - Ref level
    - Ref position
    - Scale/Div
    - Auto button
    - When pressed DONE, update the variables and the cfg.txt too.



- MARKERS
    - Launch a panel where we can select how many markers we want and their frequency.
    - Activate/deactivate markers
    - Set freq
    - When pressed DONE, update the variables and the cfg.txt too.



- MEASURE
    - Launch a panel where we can select: S11, S21, both (three buttons).

- DISPLAY
    - Data -> Memory (button)
    - Display data/memory/both
    - Continuous display/discrete

- CALIBRATION
    - Launch a window where we can select:

- 1 port
  - Save calibration
  - Load calibration
  - Perform calibration
    - Open
    - Short
    - Match

- 2 ports
  - Save calibration
  - Load calibration
  - Perform calibration
    - Open
    - Short
    - Match

- EXIT
    - When pressed, the configuration is saved in cfg.txt

NOTES: Regarding calibration… should we keep a general calibration that is applied when changing the frequency options in order to have measurements that are closer to the most accurate measurements?. Without calibration, the results can be very weird, so probably we want to store a “generalcal.cal” file that is applied whenever we don’t have a calibration file.

TASKS:

1) Generate the default data and the last used data files. (First time they will be equal). After launch the last used data will be loaded in the variables.
2) Also generate the default cal data and last used cal data files. (same as with the config data).
3) Make sure that, when exiting the application, the current values are saved in the “last used” file.
4) Add a button: PRESET (load default data in the variables.)
5) Make sure all variables in the menu are updated at launch and when changed.

Functions to control synthesizer:

Getreference

Setreference

Getoptions

Setoptions

Getrflevel

Setrflevel

Getoptions

Setoptions

getedpf

c. Appendix 3

Synthesizer control functions

```
function [result, options]=getoptions()
%[result, options]=getoptions()
%
% get all data from synthesizer A (in options)
% Checked and working fine


s=serial('COM3','BaudRate', 9600);
fopen(s);
fwrite(s, cast(hex2dec('80'), 'char')); % writes binary chars, no terminator
resp = fread(s,24, 'uint8');
cksum=fread(s,1,'uint8');
% idn = fscanf(s);
fclose(s);
%resp is the data, cksum is the checksum byte

sn=verify_checksum(resp,24,cksum);
if (sn==0) % bad checksum
    result=0;
else  %good checksum. proceed with options' extraction
    result=1;


    reg2=uint32(0);

    %unpackint: get 4 bytes and concatenate them in a 4 byte variable

reg2=(bitshift(uint32(resp(9)),24)+bitshift(uint32(resp(10)),16)+bitshift(uin
t32(resp(11)),8)+uint32(resp(12)));

    % C code:
    % reg2=((uint32_t(bytes[0]) << 24) + (uint32_t(bytes[1]) << 16) +
    %            (uint32_t(bytes[2]) <<  8) + (uint32_t(bytes[3])));

    options.low_spur=bitand(bitand((bitshift(reg2,-30)) , 1),
bitand(bitshift(reg2,-29) , 1));
    options.double_ref=bitand(bitshift(reg2, -25) , 1);
    options.half_ref=bitand(bitshift(reg2, -24) , 1);
    options.r=bitand(bitshift(reg2, -14) , uint32(hex2dec('03ff')));

end
```

/----------/

```
function [result, regs,freq]=getfreq()
%[result, regs, freq]=getfreq()
%
% get frequency programmed in synthesizer A (in freq)
%


s=serial('COM3','BaudRate', 9600);
fopen(s);
fwrite(s, cast(hex2dec('80'), 'char')); % writes binary chars, no terminator
resp = fread(s,24, 'uint8');
cksum=fread(s,1,'uint8');
% idn = fscanf(s);
fclose(s);
%resp is the data, cksum is the checksum byte

sn=verify_checksum(resp,24,cksum);
if (sn==0) % bad checksum
    result=0;
else  %good checksum. proceed with options' extraction
    result=1;
    EPDF=getEPDF();
    regs=unpack_freq_regs(resp);
    % unpack frequency registers from bytes and put it in regs
    freq=float('single');

freq=(single(regs.ncount)+single(regs.frac)/single(regs.mod))*single(EPDF)/si
ngle(regs.dbf);

end
```

/----------/

```
function [result,vcor]=get_vco_range()
% [result,vcor]=get_vco_range()
%
%

s=serial('COM3','BaudRate', 9600);
fopen(s);
fwrite(s, cast(hex2dec('83'), 'char')); % writes binary chars, no terminator
resp = fread(s,4, 'uint8');
cksum=fread(s,1,'uint8');
% idn = fscanf(s);
fclose(s);
%resp is the data, cksum is the checksum byte

sn=verify_checksum(resp,4,cksum);
if (sn==0) % bad checksum
    result=0;
else  %good checksum. proceed with options' extraction
    result=1;
    vcor.min=unpack_short(resp,1);
```

```
    vcor.max=unpack_short(resp,3);
end
end
```

/----------/

```
function reference=getEPDF()

reference=float('single');
reference=10; % In MHz
[result, options]=getoptions();
if options.double_ref==1
    reference=2*reference;
end
if options.half_ref==1
    reference=reference/2;
end
if options.r>1
    reference=reference/options.r;
end

end
```

/----------/

```
function set_options(options)
% set_options(options)
%
% set synthesizer A options


s=serial('COM3','BaudRate', 9600);
fopen(s);
fwrite(s, cast(hex2dec('80'), 'char')); % writes binary chars, no terminator
resp = fread(s,24, 'uint8');
cksum=fread(s,1,'uint8');
% idn = fscanf(s);
fclose(s)
%resp is the data, cksum is the checksum byte

sn=verify_checksum(resp,24,cksum);
if (sn==0) % bad checksum
    result=0;
else  %good checksum. proceed with options' extraction and configuration
    result=1;

    reg2=uint32(0);
```

```
    %unpackint: get 4 bytes and concatenate them in a 4 byte variable

reg2=(bitshift(uint32(resp(9)),24)+bitshift(uint32(resp(10)),16)+bitshift(uin
t32(resp(11)),8)+uint32(resp(12)));

    % C code:
    % reg2=((uint32_t(bytes[0]) << 24) + (uint32_t(bytes[1]) << 16) +
    %            (uint32_t(bytes[2]) <<  8) + (uint32_t(bytes[3])));

    %reg2=bitand(reg2, uint32(hex2dec('9c003fff')));
    reg2=bitand(reg2, uint32(hex2dec('9c001fff')));

a=bitor(bitshift(bitand(options.low_spur,1),29),bitshift(bitand(options.doubl
e_ref,1),25));

b=bitor(bitshift(bitand(options.half_ref,1),24),bitshift(bitand(options.r,uin
t32(hex2dec('03ff'))),14));
    c=bitor(a,b);
    d=bitor(c,bitshift(bitand(options.low_spur,1),30));
    reg2=bitor(reg2,d);

    %packint
    resp(9)=bitand(bitshift(reg2, -24), uint32(hex2dec('ff')));
    resp(10)=bitand(bitshift(reg2, -16), uint32(hex2dec('ff')));
    resp(11)=bitand(bitshift(reg2, -8), uint32(hex2dec('ff')));
    resp(12)=bitand(reg2, uint32(hex2dec('ff')));

    %first 8 bits to send correspond to the code to write options 0x00
    bytes=[uint8(0); resp];
    %calculate the checksum of the first 25 bytes
    newcksum=gen_checksum(bytes,25);

    bytesout=[bytes; newcksum];

    s=serial('COM3','BaudRate', 9600);
    fopen(s);
    fwrite(s, cast(bytesout, 'char')); % writes binary chars, no terminator
    resp2 = fread(s,1,'uint8');
    %resp2(1) is the ACK

    fclose(s);
end
```

/----------/

```
function result=set_frequency(freq,chansp)
% result=set_frequency(freq, chansp)
%
%

dbf=int32(1);
[result,vcor]=get_vco_range();
while (freq*dbf <= vcor.min) && (dbf<=16)
```

```
    dbf=dbf*2;
end

if (dbf>16)
    dbf=16;
end


vco=double(freq*dbf);
EPDF=getEPDF();
regs.ncount=uint32(vco/EPDF);
%regs.frac=uint32((vco-regs.ncount*EPDF)/chansp+0.5);
regs.frac=0; % This way we are working in integer-N mode
regs.mod=uint32(EPDF/chansp+0.5);
regs.dbf=dbf;

%reduce frac/mod to simplest fraction

if (regs.frac ~= 0) && (regs.mod ~= 0)
    while (bitand(regs.frac,1)==0) && (bitand(regs.mod,1)==0)
        regs.frac=regs.frac/2;
        regs.mod=regs.mod/2;
    end
else
    regs.frac=0;
    regs.mod=1;
end

%Write values to hardware

bytes(1)=uint8(hex2dec('80'));
s=serial('COM3','BaudRate', 9600);
fopen(s);
fwrite(s, cast(bytes, 'char')); % writes binary chars, no terminator
resp = fread(s,24,'uint8');
cksum=fread(s,1,'uint8');
fclose(s);

sn=verify_checksum(resp,24,cksum);
if (sn==0) % bad checksum
    result=0;
else  %good checksum. proceed with options' extraction
    result=1;
    bytesout=(pack_freq_registers(regs,resp));
    bytesout2=[uint8(0);bytesout];
    %bytesout=[uint8(0);resp]
    bytesout2(26)=gen_checksum(bytesout2,25);
    s=serial('COM3','BaudRate', 9600);
    fopen(s);
    fwrite(s, cast(bytesout2, 'char')); % writes binary chars, no terminator
    result2 = fread(s,1,'uint8');
    fclose(s);
end

if result2~=6
```

```
    result=0;
end
end
```

/----------/

```
function [result]=set_vco_range(vcor)
% [result]=set_vco_range(vcor)
%
%
bytes(1)=uint8(hex2dec('03'));
bytes=pack_short(vcor.min,bytes, 2);
bytes=pack_short(vcor.max,bytes, 4);
bytes(6)=gen_checksum(bytes,5);

s=serial('COM3','BaudRate', 9600);
fopen(s);
fwrite(s, cast(bytes, 'char')); % writes binary chars, no terminator
result = fread(s,1,'uint8');
fclose(s);
end
```

d. Appendix 4

Measure code

```
function data=measure(fstart,fstop, fstep)
% [data,n]=measure(fstart,fstop)
%
% Performs a measurement of the mini VNA
%
% Output
% data: measured data, I and Q in a matrix. n=fstart-fstop+1 frequencies
%
% Input:
% fstart: start frequency in MHz
% fstop: stop frequency in MHz

waitwindow('Title', 'Measuring')

j=1;
[result, options]=getoptions();
options.r=10;  %Comparation freq=1MHz
setoptions(options);

% Precalculated offsets are in the variable off
% load('offset.mat');
```

```
% Serial open for the Synth ctrl.
s=serial('COM3','BaudRate', 9600);
fopen(s);

% Create channel for the DAQ
ai = analoginput('nidaq','Dev1');
addchannel(ai,0:1);
% out = daqhwinfo(ai)
% out.InputRanges
ai.Channel.InputRange = [-1 1];
fs=2000;  %2000 samples per second
set(ai,'SampleRate', fs);

data=zeros(2,(fstart-fstop)/fstep+1); %freq step: 10MHz

samples=100;

for i=fstart:fstep:fstop
    %program synth to freq i

    set_frequency_short2(i,s);
    pause(0.01)
%     if mod((j-1),10)==0
%         start(ai)
%     end
    start(ai)
    data2=getdata(ai,samples);
    stop(ai)
%     figure
%     plot(data)
%     pause(1)
    %DATA=fft(data);
    %out(1,j)=abs(DATA(1,1));
    %out(2,j)=abs(DATA(1,2));
    data(1,j)=sum(data2(:,1))/samples; % off is substracted to compensate the
offset
    data(2,j)=sum(data2(:,2))/samples;
    j=j+1;
    i

end
fclose(s);
delete(waitwindow)
    %measure I and Q from the NI board
    %save measurements in the matrix data
```