UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

DEVELOPMENT OF A PLC SIGNAL ANALYZER PROTOTYPE USING THE USRP MODULE, THE GNU RADIO SOFTWARE AND A PLC-USRP INTERFACE PROTOTYPE

Paulo Victor Rodrigues Ferreira

DEZEMBRO

2012

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

DESENVOLVIMENTO DE UM PROTÓTIPO DE UM ANALISADOR ESPECTRAL DE SINAL PLC UTILIZANDO O MÓDULO USRP, O SOFTWARE GNU RADIO E O PROTÓTIPO DA INTERFACE PLC-USRP

Paulo Victor Rodrigues Ferreira

Texto da dissertação apresentada à Universidade Federal de Uberlândia, perante a banca de examinadores abaixo, como parte dos requisitos necessários à obtenção do título de Mestre em Ciências. Aprovada em 21 de Dezembro de 2012

DEVELOPMENT OF A PLC SIGNAL ANALYZER PROTOTYPE USING THE USRP MODULE, THE GNU RADIO SOFTWARE AND A PLC-USRP INTERFACE PROTOTYPE

Paulo Victor Rodrigues Ferreira

Texto da dissertação apresentada à Universidade Federal de Uberlândia como parte dos requisitos para obtenção do título de Mestre em Ciêcias.

Prof. Dr. Antônio Cláudio Paschoarelli Veiga	Prof. Dr. Alexandre Cardoso
Orientador	Coordenador do curso de Pós-Graduação

Agradecimentos

Agradeço a DEUS por me dar saúde e forças para desenvolver esta dissertação e pela oportunidade de ampliar meus conhecimentos. Dedico este trabalho aos meus pais e em especial a minha mãe Jacqueline e minha avó Joana, que me deram todo o apoio necessário. Às minhas irmãs, tios, padrinhos e demais familiares agradeço pela compreensão nos diversos momentos de ausência e pelos conselhos que muito me motivaram e contribuíram para a continuidade do desenvolvimento deste projeto. Sou grato também aos professores do curso do Programa de Pós-Graduação da Faculdade de Engenharia Elétrica, por compartilharem seus conhecimentos e experiências, em especial ao Prof. Dr. Antônio C. P. Veiga, pelo seu profissionalismo e pelas orientações e ensinamentos concedidos ao longo do curso de Mestrado. Também agradeço aos demais colegas pelo apoio e solidariedade durante as dificuldades e principalmente nos momentos de alegrias quando da solução dos problemas. À CAPES, sou grato pelo apoio financeiro, através da concessão da bolsa, porque sem ela o desenvolvimento deste trabalho seria financeiramente inviável.

Obrigado a todos os que direta ou indiretamente contribuíram para o desenvolvimento desta dissertação. Que DEUS abençoe a todos nós, para que possamos dar continuidade ao desenvolvimento do conhecimento científico e contribuir para a solução dos problemas da sociedade.

Resumo

Ferreira, P. V. R. **&** ***Veiga***, A. C. P. DESENVOLVIMENTO DE UM PROTÓTIPO DE UM ANALISADOR ESPECTRAL DE SINAL PLC UTILIZANDO O MÓDULO USRP, O SOFTWARE GNU RADIO E O PROTÓTIPO DA INTERFACE PLC-USRP, FEELT-UFU, Uberlândia, 2012, 127.

O objetivo deste trabalho é desenvolver um protótipo para fazer a aquisição e análise de sinais de redes Powerline Communication (PLC), utilizando o módulo hardware USRP, o software GNU Radio e um protótipo de uma interface PLC-USRP desenvolvida especialmente para permitir a acquisição dos sinais. O desenvolvimento desta dissertação se baseia no fato de não existir ferramenta equivalente no mercado, que permita a aquisição e a análise de sinais PLC. Outro fator motivacional é a utilização de soluções de baixo custo, como o módulo USRP e o software GNU Radio que implementam o conceito de Rádio Definido por Software (RDS). Para implementar a interface física do módulo USRP com a rede PLC e acondicionar os sinais adequadamente, foi desenvolvido um protótipo da interface PLC-USRP. Com a finalidade de certificar o correto funcionamento dessa interface foi desenvolvido um protótipo de um software de baixo custo baseado em RDS para fazer a aquisição da amplitude da resposta em frequência de um circuito eletrônico. Para mostrar os resultados obtidos com o protótipo do analisador espectral de sinal PLC foi realizada a aquisição de seu espectro em tempo real.

Palavras-chave

PLC, USRP, GNU Radio, Resposta em Frequência, Analisador espectral

Abstract

Ferreira, P. V. R. & **_Veiga_**, A. C. P. DEVELOPMENT OF A PLC SIGNAL ANALYZER PROTOTYPE USING THE USRP MODULE, THE GNU RADIO SOFTWARE AND A PLC-USRP INTERFACE PROTOTYPE,
FEELT-UFU, Uberlândia, 2012, 127.

This work aims to develop a prototype that makes the acquisition and analysis of Powerline Communication (PLC) network signals, using the hardware module USRP, the GNU Radio software and a prototype of a PLC-USRP interface specially developed in order to allow the signals acquisition. The development of this project relies on the fact that there is no similar tool on the market, which allows the PLC signals acquisition and analysis. Another issue is the usage of low-cost solutions like the USRP module and the GNU Radio software, that together implement the Software Defined Radio (SDR) concept. For the implementation of the physical USRP interface with the PLC network and to adjust the signals to be suitable to be handled by the USRP module a PLC-USRP interface prototype was designed. Aiming to experimentally validate the correct performance of this interface a low-cost software prototype was designed, also based on the SDR concept to make the acquisition of the frequency response amplitude of an electrical circuit. The effectiveness of the PLC signal spectral analyzer prototype was demonstrated through the acquisition of a PLC spectrum of a real transmitted signal.

Keywords

PLC, USRP, GNU Radio, Frequency Response, Spectral analyzer

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Sometimes, during the development of a new telecommunication's system or even after its release and deployment it's necessary to analyse the signals that are being transmitted or received through a certain channel in order to evaluate the transmitter, the receiver or the channel behaviour at a physical level.

This evaluation enables the certification of the telecommunication's system to make sure that the practical system is compliant with its primary design in terms of performance like ensuring the electromagnetic compatibility (EMC) and in terms of quality of the service delivered. Another usage of this evaluation is to allow that further researches and tests are made seeking the improvement of the system performance.

Therefore, an evaluation system is required to intercept the channel and measure the signals without affecting its characteristics. This system must have an interface that is able to deal with the telecommunication's channel and with the measuring system which will acquire the signals and make all the desired evaluation processing.

1.1 The problem

One telecommunication's technology that is becoming very popular due to a lot of benefits and advantages offered like energy green solutions, home control and automation, internet access connection, remote metering, and etc., also being responsible for supporting most of the smart grid technology concept is the Power Line Communication (PLC) detailed at the Chapter 3.

The PLC physical layer used for transmission and reception of data is not an exclusive channel and there is also another kind of signal that must be treated with caution. With so many solutions, different standards and protocols being developed using the PLC concept is necessary to develop an evaluation tool able "to see" the PLC signal, capture its spectrum and make further processing that is required for a certain application or so, taking into account the electrical hazards and implementing all the protections required.

The most practical analyzer that may be used to evaluate a certain telecommunication's channel is the spectrum analyzer, which allows to see the frequency usage and the power level in a certain frequency range. The problem with this approach is the price of the spectrum analyzer since the most important task that it will perform is to show the spectrum in real time and most of the time it won't make the spectrum available for further processing, like real-time processing the spectrum shown. To perform this, the same system responsible for acquiring the signal and calculating the spectrum must be able to get the same signal and to perform a certain processing on it. Moreover, to develop such a system, it's necessary to have a system that allows flexibility on development and testing of many different processing techniques over the signal acquired.

On the market there isn't a system with such framework, where you just turn it on, connect it to a network and it processes the signal as you want to. The platforms available are expensive spectrum analyzers and some of them are able to perform very specific and limited processing.

Nowadays, the prices of the basic models of such spectrum analyzers starts between US$ 22,000.00 up to US$ 56,000.00 without taxes. So with the need to evaluate a certain telecommunication's channel and acquire its spectrum for making another processing tasks, a cheaper, flexible and computer-based software solution must be developed.

1.2 The solution

In order to build a system that is capable of dealing with a certain channel interface design and at the same time to acquire the signal and process it using a computer-based software, it was found a technology that is essentially an hybrid hardware equipped with a multi-purpose processor. It's called Universal Software Radio Peripheral (USRP), a hardware module that works together with a license-free software called GNU Radio in order to implement the Software Defined Radio (SDR) concept. This SDR concept aims to transform most of the hardware problems and activities into software ones. Some of them are more detailed on Chapter 3.

For the evaluation system to have access to the PLC network an interface must be developed. This interface may connect the GNU Radio software through the USRP hardware module to the PLC network. Some requirements like safety coupling of the analyzer system to the network and voltage signal's levels must be verified and met by the interface design. The development of this interface is detailed on Chapter 5.

After building the interface it's necessary to know its frequency response. This response aims to show the level of interference (gain, attenuation, phase delay, etc.) on the system connected to it.

On the market there are some systems that do this kind of test and they are known as Frequency Response Analysers (FRA). Their price start about US$ 2,400.00 up to US$ 9,495.00 without taxes and shipping, the most basic ones including only the dedicated hardware. The military standard, which is portable, and includes all the required hardware embedded like battery and monitor costs about US$ 14,370.00 also without taxes and shipping.

But the USRP is a low-cost hardware module with some capabilities and a multipurpose processor, then, before developing the PLC-USRP interface, a prototype for amplitude frequency response acquisition was developed aiming to also perform one of the most important function of the FRA equipment. The development of this prototype is detailed in Chapter 4.

Finally, with the PLC-USRP interface and its amplitude frequency response, a small PLC network was build and the prototype of the PLC network analyzer system was connected to it in order to acquire its spectrum. This experiment and the related comments are described at Chapter 6.

1.3 Final considerations of this chapter

This chapter showed the problem to be solved throughout this dissertation, the solution to solve this problem and how this dissertation is structured.

The next chapter will make a general introduction on the Power line Communication's basic principles. The chapter will cover some basic technical information based on the

IEEE 1901-2010 standard like network modes, operational frequency bands and etc.

Chapter 2

Power line Communications

Power line Communications (PLC) is a technology that uses the high, medium and low voltage electrical network to provide telecommunication services. It allows data transmission over electrical wiring, which corresponds to the physical layer in the OSI model, and is therefore, a complementary function to delivery of electrical power.

2.1 The basic principles

In 1838, an English man, Edward Davy, proposed a solution allowing remote measurement to be taken of battery levels of sites far from the telegraph system between London and Liverpool. In 1897, he submitted the first patent for a technique for the remote measurement of electrical network meters communicating over electrical wiring. In 1950, the first PLC systems known as Ripple Control (single-directional with carrier frequency between 100 Hz and 1 kHz) were designed and deployed, over medium- and low-voltage electrical networks, for the remote switching on and off of public lights or for tariff changes [9] .

After 1960, the first European committee for electrotechnical standardization (Cenélec) band PLC system appeared in France, extending from 3 to 148,5 kHz allowing bidirectional communications over the low voltage (LV) network for meter readings and home automation.

By that time, there were more infrastructure of electrical lines than telephone lines, and after the World War II the communications systems were designed for data transmission over high and medium voltage wiring, imitating remote meter readers that already existed by then.

One of the first PLC commercial technologies is known as X10, an older home automation mechanism developed in 1975, which communicates with a throughput of 20 bits per second. The X10 protocol consists of short commands that can switch short distant devices on and off, for example [32].

Nowadays, some devices don't need to communicate over radio, because when a wired network is available it can be used too. A wired infrastructure can be more economical if it is already in place, and in the case of the power lines widespread availability, the network can deliver both power and communication.

There are some places where, also, the wireless communication is not efficient due to long distances, high attenuation/interference, EMC (electromagnetic compatibility) constraints, and most of the time the installation of additional cables, pipes, electronic devices or any kind of environment modification are not allowed, but there is still the need of a reliable data network to be widely deployed. In such a scenario there is a high probability that may exist a power line inside this environment or even very near to it. This is a suitable scenario for PLC to take place and deploy the required network.

With the science evolution and, consequently, the electronic developments the need for high-resolution multimedia and interactive content, high-speed internet access, and many others lead to the development of high-throughput standards with rates up to 200 Mbps and 500 Mbps up to 300 meters by 2012, which are widely deployed within home networks. On the other hand, there are some applications that don't require high throughputs, but need to reach longer distances within a minimum data rate and high security levels through high-voltage grids. Nowadays, for these two different reasons, the two most important alliances, HomePlug Alliance and G3-PLC Alliance, are the main PLC players and drive the development and researches related to this technology.

The HomePlug Alliance [14] is a comprised of industry leaders at each level of the value chain (from technology to services and content) and its members bring the necessary capabilities and financial commitment to the successful development and launch of the technology. Its main objectives are to enable and promote rapid availability, adoption and implementation of cost effective, interoperable and standards-based home power line networks and products for low-voltage networks. Although, each alliance establishes a different kind of standard, all the specifications, tests and constraints throughout this text will be based on the HomePlug AV standard, because of its support for IEEE 1901-2010 [16] standard for high-speed communication devices.

The G3-PLC Alliance [1] was formed to support, promote and implement G3-PLC in smart grid communications applications. Its missions are to support G3-PLC in internationally recognized standards bodies to achieve the rapid adoption of this specification worldwide, develop a framework for equipment testing to facilitate interoperability among its adopters, establish a forum to discuss revisions and enhancements to the specification and identify developers and providers requirements for interoperability and general usability.

The IEEE 1901-2010 standard was the first standard of IEEE related to PLC technology and the HomePlug technology was consolidated as the baseline standard for the power line communications. Between all the existing standards, HomePlug AV is one of the most deployed and used, that is why all the present study aims to be compatible to this standard and will be a starting point and a reference for all the physical layer related issues.

HomePlug Alliance calls the products adhering to IEEE 1901-2010 standard as 'HomePlug AV certified', which are fully interoperable with any other 1901 standard product and older HomePlug standards. HomePlug AV is an extension of the older HomePlug 1.0 standard which had a bit rate on the order of 14 Mbps in comparison with the theoretical 200 Mbps of the HomePlug AV.

Basically, PLC networks correspond to layers 1 (physical) and 2 (data link) of the OSI model, supplying an Ethernet connection service to the layers above. HomePlug focuses on two principal aspects:

- Physical layer:

  Concerns with data transmission over the power line medium. The electrical wiring wasn't designed to transport high frequency data signals, but signals at 110 V or 220 V in frequencies of 50 Hz or 60 Hz. Also, all the electrical equipments that are connected to the electrical network or disconnected create measurable level of perturbations/noise on the transmission channel. And from the EMC point of view, every powered device, in proximity to the wiring, generates electrical perturbations which may be conducted (via wiring) or inducted (via radio signals). To improve data transmission, the physical layer uses optimized techniques for coding, modulation and error correction which results in good connectivity and transmission rates between devices;

On the electrical wiring, where a PLC network is deployed, the resulting signals are the sum of the modulated PLC and the power signal as shown by Figure 2.1 [2]. Figure 2.2 [9] shows the two frequency band allocated to the PLC networks: 3 to 148 kHz for low bit rate PLC and 2 to 30 MHz for high bit rate PLC. HomePlug AV uses the bandwidth from 2 MHz up to 28 MHz.

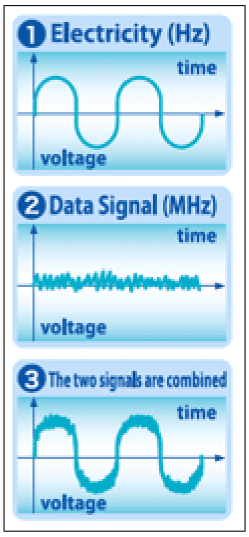


Figure 2.1: Sum of modulated PLC and power signals

- Data link layer:

  Defines the architecture and mechanisms to allow the transmission to take place over the network under the best possible conditions. The network access technique used determine its performance. Event though the PLC is wired, its data link layer has more in common with wireless data link layer. In wired data link layers such as Ethernet, Carrier Sense Medium Access/Collision Detection (CSMA/CD) is often used for collision detection. But wireless and PLC transceivers cannot listen for incoming signals while transmitting their own signals, therefore PLC used CSMA with Collision Avoidance (CA) in HomePlug 1.0. But CSMA/CA algorithm doesn't

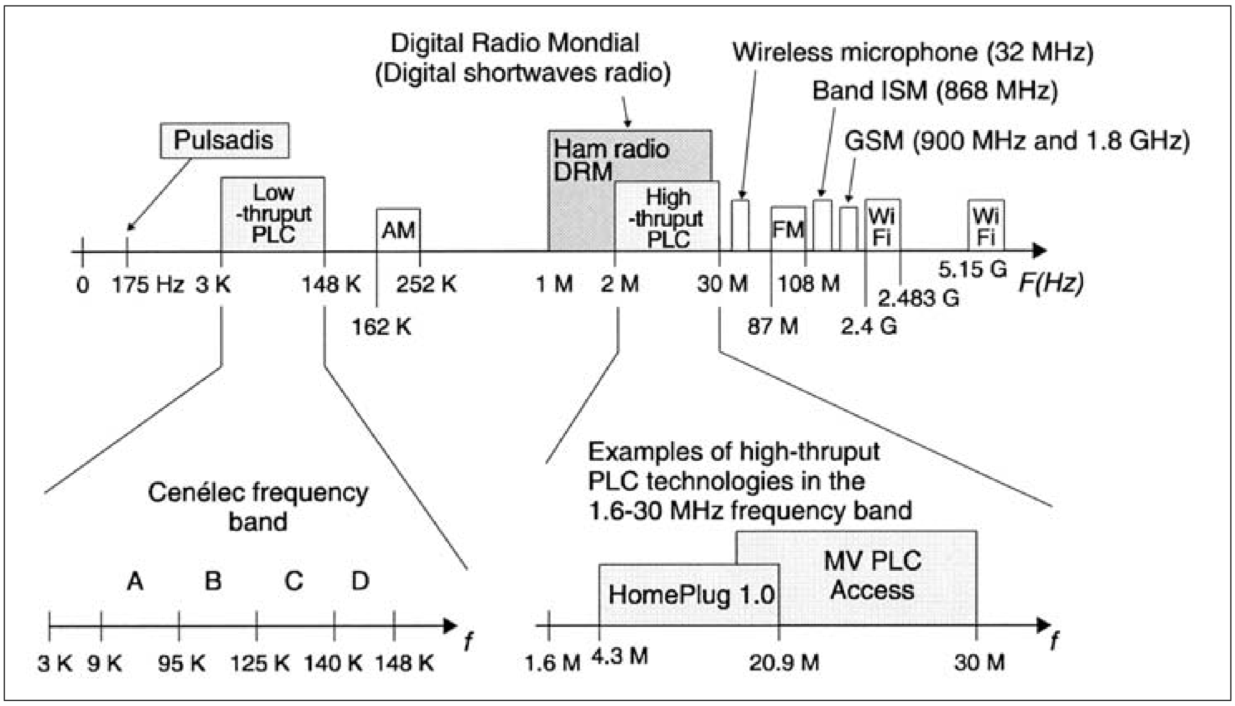


Figure 2.2: PLC networks frequency bands

guarantee a minimum transmission time. This is why HomePlug AV implements Time Division Multiple Access (TDMA) as the medium access system, allocating transmission time slots, which is a required solution when dealing with real-time based applications with a high throughput. It's important to note that one of these time slots are dedicated for the interoperability with PLC devices using CSMA/CA.

This way, the PLC equipment provides a terminal or any device, which needs to be connected to the data network, an Ethernet connection service using a MAC protocol and RJ-45 connectors allowing this device to access services in high layers such as IP, TCP, HTTP, etc.

The topology of power lines networks can be viewed as a medium shared by all equipment carrying multiple PLC signals, transporting data exchanged between terminals of a local network. The concept of shared medium can be divided into two different domains. One is the private electrical network, as shown in Figure 2.3, located behind the meter connecting it to the public electrical network and managed by those in the zone it serves, like an apartment, a house, an office, a factory, etc. The other domain is the public electrical network, as shown in Figure 2.4, which is the distribution network that supplies houses, stores, and factories within a neighbourhood, a town, or a community, and it's is public to the extent that anyone may be a subscriber.

Inside a home building, all branches of the network generally come out of the main circuit breaker panel and meter, and the PLC signals circulate in all branches. The electrical network can be seen as a data bus with PLC devices connected on both the public and private zones [9].

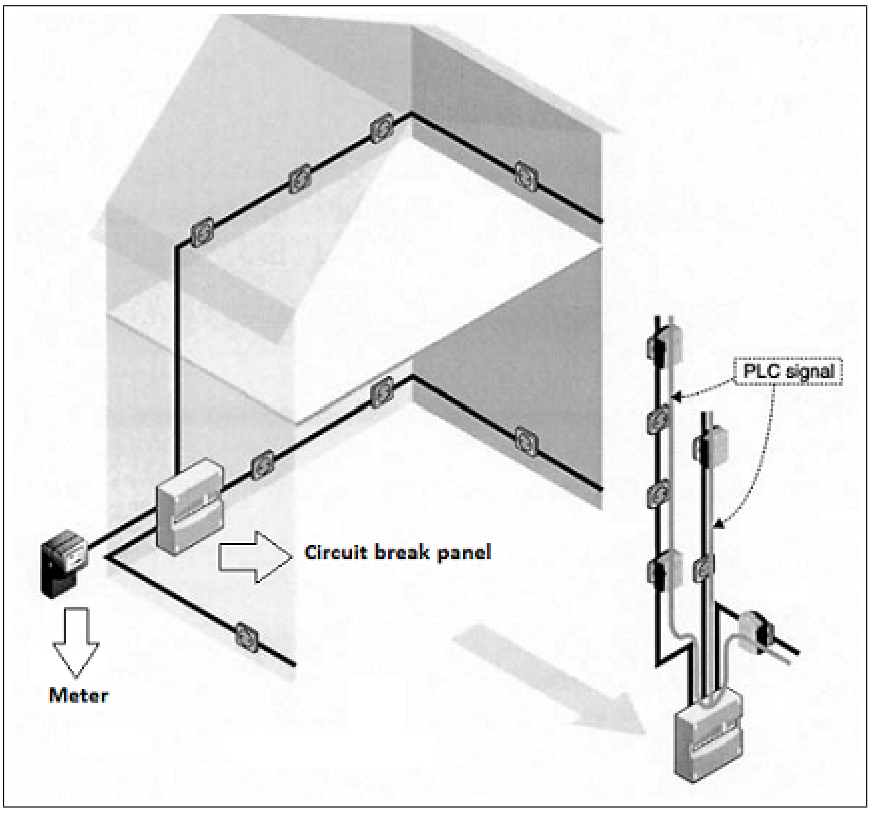


Figure 2.3: Private electrical network

The PLC network acts as an antenna producing electromagnetic radiation in its environment and disturbs other services operating in the same frequency range. Therefore, the regulatory bodies specify very strong limits regarding the electromagnetic emission from the PLC networks, with a consequence that PLC networks have to operate with a limited signal power. This causes a reduction of network distances and data rates and increases sensitivity to disturbances. This is why the PLC transmission power is specified by its Power Spectral Density, instead of the voltage amplitude [15].

Sometimes is necessary to repeat the signal in order to extent its coverage zone. To do that, the PLC repeaters amplify and regenerate the signals by means of one of the two different repeater types. The physical repeater physically amplifies the signal and retransmits it without bandwidth reduction. The logical repeater repeats the signal at a

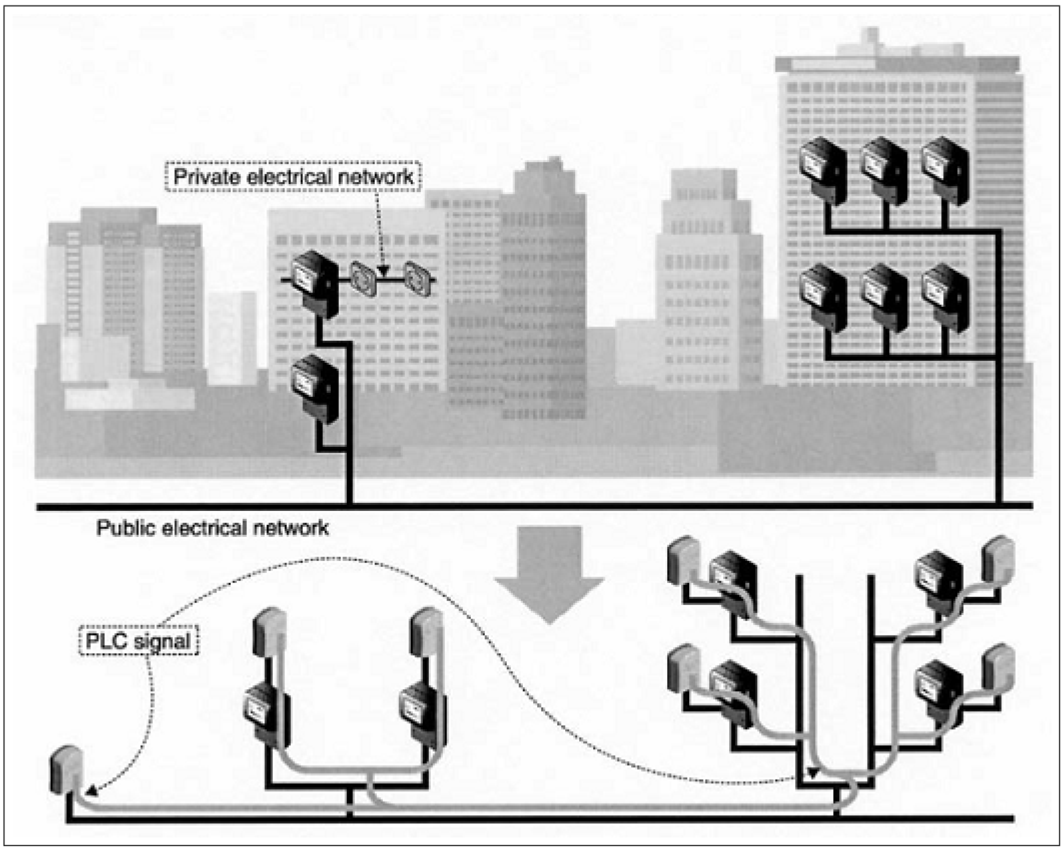


Figure 2.4: Public electrical network

data frame level. It is composed by two PLC repeaters connected by Ethernet interface, one is connected at one segment of the electrical network where the signal is still good and the second is connected at the segment where the signal cannot reach. By doing this, the overall bandwidth is reduced by a factor of two, because two distinct logical PLC networks are created on the same electrical network.

One of the major functionalities of the PLC networks is the network mode, which is used to manage all PLC devices at the same network. There three network modes:

- Master-slave: like a client-server type in an IP network, in which the master device manages the exchanges between the PLC devices of the network

- Peer-to-peer: like a peer-to-peer IP network, where all PLC devices play the same role and have the same hierarchical level and are allowed to change data without being monitored.

- Centralized: a mix of the previous ones where the CCo (central) device manages the medium access link in master-slave mode and the data is exchanged between the devices without going via CCo like in peer-to-peer mode. This is the actually the mode of the HomePlug AV technology.

The disturbances that come from the electrical devices which are also attached to the electrical network may be in the frequency band of the PLC devices. It is interesting to install filters as close as possible to the disturbing devices in order to stop these disturbing frequencies from getting into the PLC network. One must note that using such filters makes the disturbing device unable to access the PLC network. But its manufacturer can foresee this case and implement an internal split placing the filter only on the power supply side, making sure that the PLC signal goes into the internal PLC modem without any filter in its path.

Although the PLC electrical wiring is also a medium shared by the various network devices it's difficult to have access to it and it involves major dangers due to the presence of power signal at 110-220 V/50-60 Hz. The PLC network security is not a big concern because of the difficult to have access to its transmission physical medium as long as in Wi-Fi anyone in the network coverage area has plenty access to the physical medium layer and can even intercept the signals. However, the PLC network is subjected to various types of attacks either to interfere with the PLC operation or to intercept the transmitted information. Also, its signals propagates outside the limits of the private electrical networks in a conducted or radiated way, which implies in the implementation of suitable software security levels.

The main network attack types are eavesdropping, prevention of operation, which leads to the network collapse and network reconfiguration. To avoid any information disclosure, the network traffic must be encrypted in a way that anyone not allowed cannot recover or decipher it. So, to counter-attack these attacks some security techniques are used such as cryptography (to prevent intruders from accessing data), authentication (to identify and authorize data exchange), and integrity control (to know whether the data exchanged was not modified during transmission).

In the HomePlug standard, the PLC logical network is based on an encryption key called Network Encryption Key (NEK) that encrypts the data exchanged between the various PLC devices as shown in Figure 2.5. Each PLC logical network can be configured with a NEK by the following methods:

- Ethernet interface: The configuration frame of the NEK is sent in broadcast mode and all the PLC devices connected by Ethernet interface recover this configuration.

- Electrical interface: The configuration frame of the NEK is sent over the electrical network to the PLC devices. A second key, called Default Encryption Key (DEK) which is specific to each PLC device, printed in a tag given by the manufacturer, must be known. This way, an encrypted NEK is exchanged between two PLC devices, the configuring station and the one that wants to get the NEK and to connect to the network.

- Web interface: Some advanced PLC devices handle key configuration by Web interface.

This assures that the data (encrypted frames) only flows between the electrical interface and the Ethernet interface if the device has the right NEK from the PLC logical network. In addition to that, the PLC frames, known as physical frames which carries the data exchanged over a PLC network, are carried in several frequency bands and the modulation techniques used are different depending on the PLC channel quality, also preventing from attacks intended to accumulate enough frames and to use brute force breaking tools to later try out all possible combinations or decryption algorithms. At each PLC device connected to a network there is a table called Tone Map, that has the information of what modulation technique has been used for each link from itself toward to each other PLC device. This table is continually refreshed in a dynamically time interval, from 10 ms up to several seconds and doesn't contain all the other link's modulation techniques used at the moment.

In HomePlug AV the security is guaranteed by functionalities such as: encryption based on 128-bit AES in cipher block chaining (CPC) mode at data link layer, easy connect button (each PLC device HomePlug AV has one, and it is possible to pair two devices pushing this button on both within an interval of some seconds), rotation of NEK values every hour, use of Direct Access Key (DAK), that is the same as DEK for HomePlug 1.0, a pair of Public-Private Key (PPK) encryption, higher layer authentication protocols like Radius authentication server, among many others.

Many electrical and electronic devices connected to the electrical network produce radio frequency electromagnetic emissions in the surrounding operational environment. These emissions may interfere with the operation of the PLC network and prevent data

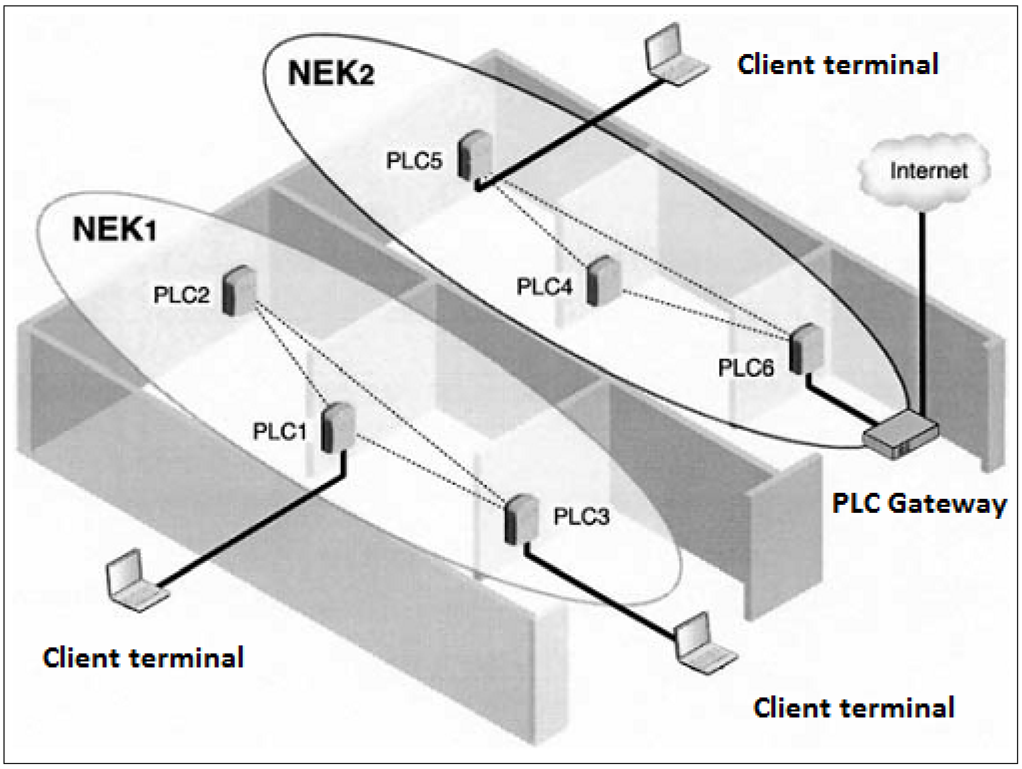


Figure 2.5: PLC logical networks using different NEK

communication if they are in the same PLC frequency range. Reciprocally, PLC devices emit electromagnetic waves that may interfere with the close operating electronic devices. To avoid this, some regulatory bodies such as International Special Committee on Radio Interference (CISPR) establish limits on the transmission power and authorized frequency bands to be used by for PLC devices.

In order to comply with transmission power and frequency band constraints, current PLC technologies, such as HomePlug AV, implement a notching technique which is basically filtering techniques for frequencies already used by other radio communication technologies and consist of listening to the radio channels to readjust the transmitted power at some frequencies.

In the IEEE 1901[16] specifications there are two physical layer modulation standards that can be selected among many other Physical Service Access Point (PHY-SAP) parameters: wavelet OFDM or windowed OFDM, also known as FFT OFDM. Windowed OFDM provides flexible spectrum notching capability where the notches can exceed 30 dB in depth without losing significant useful spectrum outside of the notch. [3]. Also, windowed OFDM has proven to be immune to impulsive noise and resistant to narrowband interference.

In section '13.2.1 - FFT outline description' of IEEE 1901-2010 is defined that the FFT OFDM employs up to 1974 carriers in the range from 1.8 MHz to 50 MHz. Of the carriers below 30 MHz, 917 carriers out of 1155, with an approximately bandwidth of 24,414 kHz each, are used for modulation and the other 238 are turned off or masked aiming not to interfere with licensed services like AM broadcast band, and higher amateur band frequencies. The carrier and spectral mask table from IEEE 1901-2010 standard is shown on Table 2.1.

Table 2.1: Modulation Characteristics

Frequency(MHz)	PSD max. (dBm/Hz)	Carrier status	Service allocation
$F \leq 1.71$	-92	0-70 OFF	AM broadcast
$1.71 < F < 1.8$	-85	71-73 OFF	Between AM & 160 m
$1.8 \leq F \leq 2$	-85	74-85 OFF	160 m amateur
$2 < F < 3.5$	-55	86-139 ON	FFT carriers
$3.5 \leq F \leq 4$	-85	140-167 OFF	80 m amateur
$4 < F < 5.33$	-55	168-214 ON	1901 FFT carriers
$5.33 \leq F \leq 5.407$	-85	215-225 OFF	5 MHz amateur
$5.407 < F < 7$	-55	226-282 ON	1901 FFT carriers
$7 \leq F \leq 7.3$	-85	283-302 OFF	40 amateur
$7.3 < F < 10.10$	-55	303-409 ON	1901 FFT carriers
$10.10 \leq F \leq 10.15$	-85	410-419 OFF	30 m amateur
$10.15 < F < 14$	-55	420-569 ON	1901 FFT carriers
$14 \leq F \leq 14.35$	-85	570-591 OFF	20 amateur
$14.35 < F < 18.068$	-55	592-736 ON	1901 FFT carriers
$18.068 \leq F \leq 18.168$	-85	737-748 OFF	17 m amateur
$18.168 < F < 21$	-55	749-856 ON	1901 FFT carriers
$21 \leq F \leq 21.45$	-85	857-882 OFF	15 m amateur
$21.45 < F < 24.89$	-55	883-1015 ON	1901 FFT carriers
$24.89 \leq F \leq 24.99$	-85	1016-1027 OFF	12 m amateur
$24.99 < F < 28$	-55	1028-1143 ON	1901 FFT carriers
$28 \leq F \leq 29.7$	-85	1144-1220 OFF	10 m amateur
$29.70 < F < 30$	-55	1221-1224 OFF	Unused carriers
$30 \leq F \leq 50$	-85	1225-2048 OFF	Unused carriers

The HomePlug AV frequency band ranges from 2 MHz up to 28 MHz and is divided into 917 frequency sub-bands at the physical layer complying with IEEE 1901-2010 standard. Each band then uses OFDM symbols in order to encode the data in an orthogonal manner in the frequency domain. In each frequency band the data and its OFDM symbols are encoded using a turbo convolutional code. The IEEE 1901-2010 FFT OFDM PLC modem is shown in Figure 2.6 [29]. The PLC modem basic architecture is shown in Figure 2.7 [15].

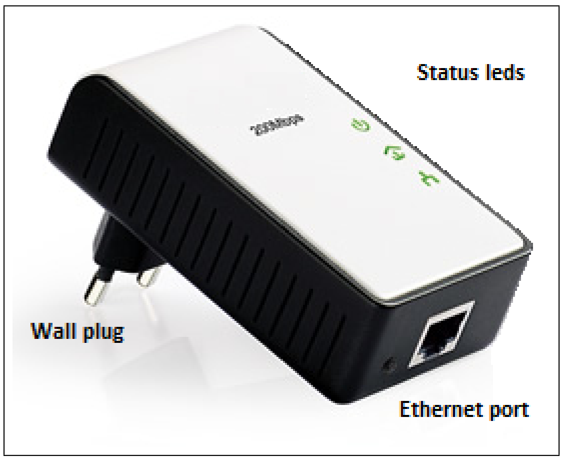


Figure 2.6: IEEE 1901-2010 FFT OFDM PLC modem

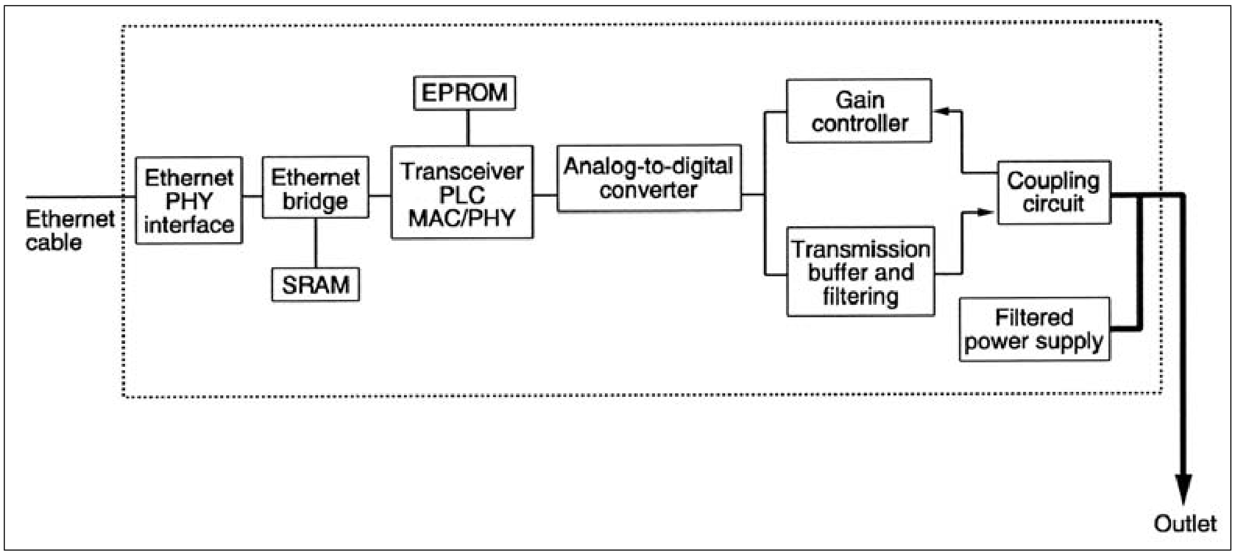


Figure 2.7: PLC modem hardware architecture

The transmission power of PLC devices is measured as a quasi-peak value and not as a mean value, i.e., a uniform distribution of the total transmission power on all the frequency sub-bands or simply, the Power Spectral Density (PSD).

The total transmit power of a transmitter terminated with a standard termination impedance (50 Ohms) should not exceed +20 dBm (100 mW) in the range of 5 kHz-100 MHz in accordance with the recommendations of ITU-T G.9964 [28], IEEE 1901-2010 [16] and IEC CISPR 22 [20]. Figure 2.8 shows the FFT OFDM transmission spectrum mask, showing that the PSD of the PLC devices must be equal to or less than -55 dBm/Hz. The PSD around -85 dBm/Hz or less, is not perceptible for the electrical network and the devices close to the electrical wiring [9].

Another functionality of PLC technology is its throughput optimization based on the

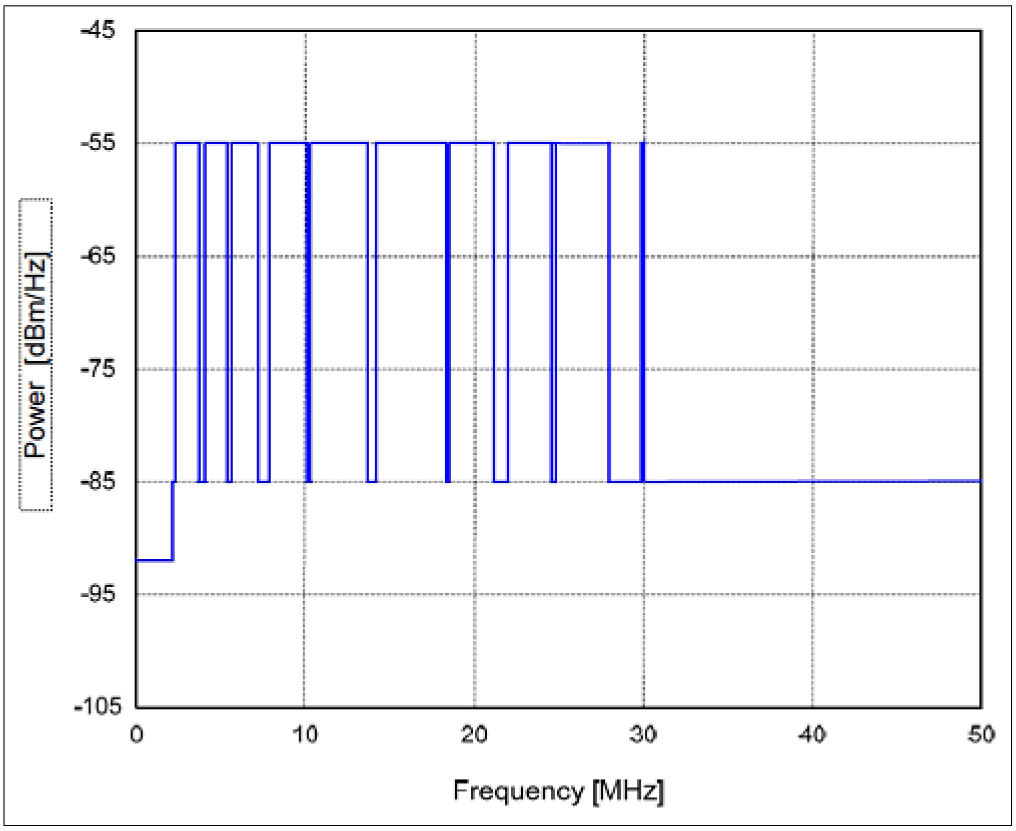


Figure 2.8: FFT OFDM transmission spectrum mask

current channel conditions between network stations. Depending on the link condition and interference and attenuation levels originated by the other electrical devices connected to the network or in proximity, the transmission speed must be permanently readjusted by choosing the carrier modulation mode, which can be different for each carrier, between BPSK, QPSK, 8-QAM, 16-QAM, 64-QAM, 256-QAM or 1024-QAM. The relation of each of these modulation type and its number of bits per carrier are shown in Table 2.2 [9].

Table 2.2: Modulation Characteristics

Modulation Type	Bits per Carrier
BPSK	1
QPSK	2
8-QAM	3
16-QAM	4
64-QAM	6
256-QAM	8
1024-QAM	10

The hardware architecture of a PLC device (modem) is structured around the main component (the HomePlug PLC chip). Around this chip, that implements all the functionalities of the PLC networks, a number of components and electronic circuits are used to optimize the operation of the PLC modem, like the coupling to the electrical network (the physical connection to the electrical outlets), the PLC signal gain control, memories (SRAM - erased when restarting the modem) for storage of information of the PLC network links state network encryption keys, etc. This basic architecture is shown in Figure 2.9 [9]. On the transmitter side, all the encoded data, Frame Control Encoder outputs are lead into a common OFDM modulator that feeds the Analog Front-End. On the receiver side the process is inverted.

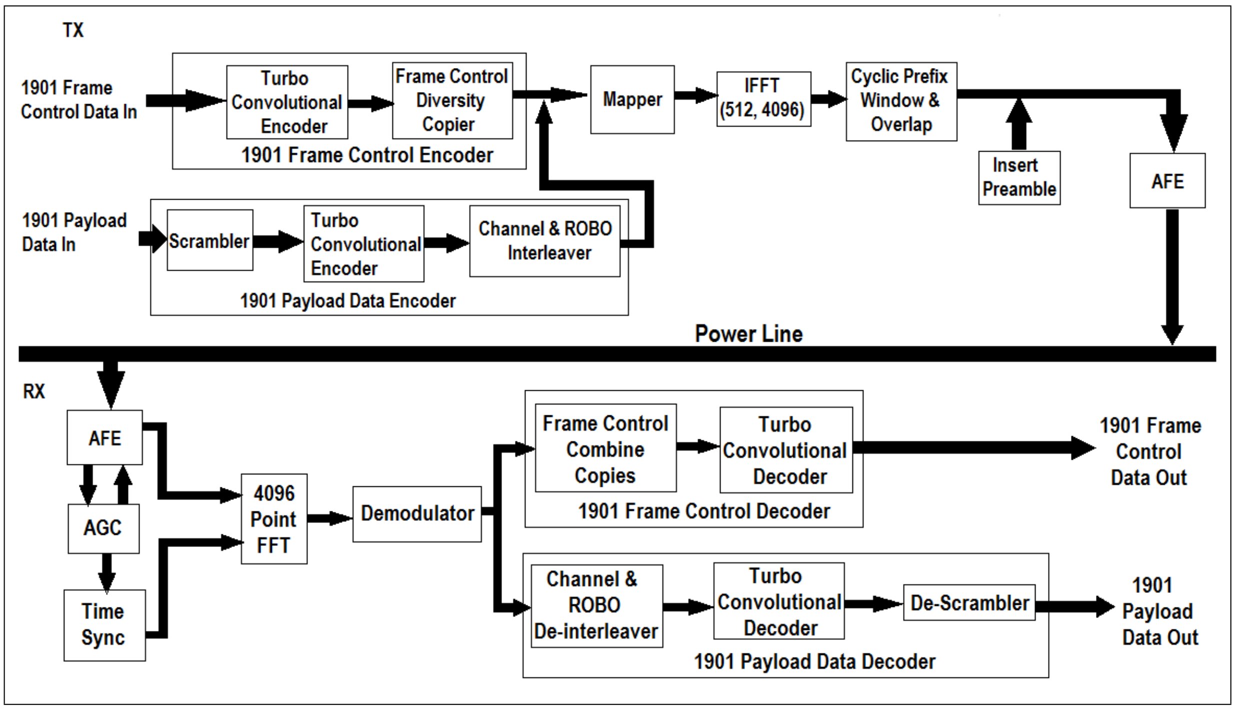

TX
1901 Frame Control Data In
Turbo Convolutional Encoder
Frame Control Diversity Copier
1901 Frame Control Encoder
Mapper
IFFT (512, 4096)
Cyclic Prefix Window & Overlap
Insert Preamble
AFE
1901 Payload Data In
Scrambler
Turbo Convolutional Encoder
Channel & ROBO Interleaver
1901 Payload Data Encoder
Power Line
RX
AFE
AGC
Time Sync
4096 Point FFT
Demodulator
Frame Control Combine Copies
Turbo Convolutional Decoder
1901 Frame Control Decoder
1901 Frame Control Data Out
Channel & ROBO De-interleaver
Turbo Convolutional Decoder
De-Scrambler
1901 Payload Data Decoder
1901 Payload Data Out

Figure 2.9: IEEE 1901 FFT OFDM transceiver

The methods for accessing the medium used by the PLC networks consists of connecting the PLC devices to the electrical network aiming to reach the best performance at the physical level and useful throughput at the upper layer level. There are two different methods or couplings types: capacitive coupling and inductive coupling. PLC devices mostly use capacitive coupling. The inductive coupling is much more efficient than capacitive coupling. It uses the electromagnetic induction method between two electrical wirings or between an electrical wiring and a coil wound around the power line wiring. An inductive coupler reduces the attenuation by 10 to 15 dB for some frequencies in comparison with a capacitive coupler [9]. However, this method requires access to the electrical wirings, unlike capacitive coupling, which is restricted to the connection of a device to an outlet.

The low-voltage networks cover the last few hundreds of meters between the customers and the transformer unit and offer an alternative solution through PLC technology for the realization of the so-called “last mile” in the telecommunications access area. In order to make the PLC signals available at a certain electrical network like just after the MV/LV transformer aiming to deploy an access neighborhood network, PLC signal injectors also known as PLC base/master stations are used as shown in Figure 2.10. These devices usually use inductive coupling.

For a building that has one or many private PLC networks to isolate its indoor (in-home) area from the outdoor area not allowing the PLC signal to be transmitted through the power meter unit is used a gateway device. This gateway which is installed near to the building meter is used to divide the PLC access network from an in-home PLC network and can act also as a local base station or repeater. It also converts the transmitted signals between the frequencies that are used in the access and in-home areas. The standards and its related frequencies used for broadband PLC access networks is out of the scope of this text.

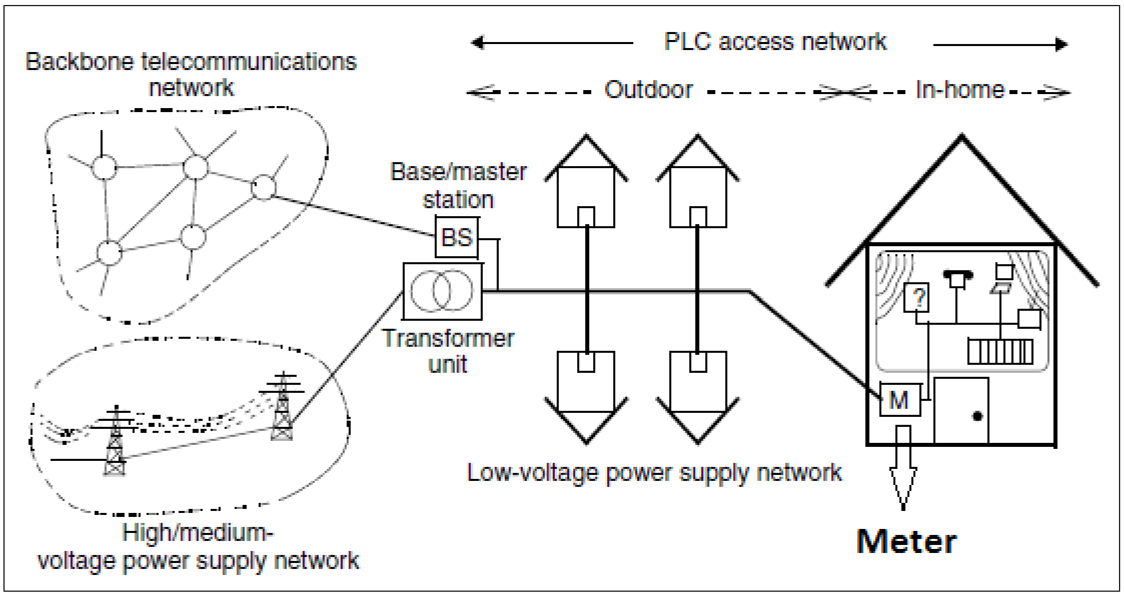

PLC access network
Backbone telecommunications network
Outdoor
In-home
Base/master station
BS
Transformer unit
?
M
Low-voltage power supply network
High/medium-voltage power supply network
Meter

Figure 2.10: PLC access network

Since none of the current communication system offers by itself the ideal capabilities to be solely and widely deployed, hybrid networks appear to be the solution that makes the best use of each of the existent technologies. One of the most successful example of such coexistence and interoperability is seen between PLC and Wi-Fi. First of all, their operating frequencies do not interfere with each other. Together they can interoperate in many different ways:

- PLC acting as a backbone network for Wi-Fi which in turn has a great capacity for distribution of signals through wide areas that handle several mobile devices;

- Wi-Fi bypassing areas with high attenuation for PLC and vice-versa; and

- Wi-Fi and PLC manufactured together in one device aiming a high throughput using the maximum transfer rate current available (depending on the Wi-Fi and PLC channels conditions) considering that both technologies dynamically adapts to reach the best performance. This solution also gives a more reliable link because the traffic can be balanced between the two systems. Finally this hybrid solution can operate in redundancy mode if one of the systems (Wi-Fi or PLC) fails completely.

PLC technology can be associated with almost any existent communication system and the major functionality responsible for allowing this interoperability is the Ethernet interface implemented on the PLC standard.

2.2 Final considerations of this chapter

This chapter gave a basic insight on the PLC technology, its basic principles and some operational technical information based on the IEEE 1901-2010 standard.

The next chapter will make a general introduction on the Software Defined Radio concept including some basic technical information of the hardware and software used to implement this concept which is a big part that composes the solution to the problem proposed on Chapter 1.

Chapter 3

Software Defined Radio

This chapter aims to give a general idea about what is the technological concept of the Software Defined Radio (SDR), its basics hardware and software modules and some examples of applications.

3.1 The SDR concept

The science evolution affects many fields of knowledge and this wouldn't be different with the telecommunications engineering. One of the most recently evolution on this field was the migration of the analog communications systems to a 100% digital environment. As a result of this migration, the digital radio concept has been developed through the Software Defined Radio (SDR) concept which seeks to replace most of all analog components that process the signals for digital ones implemented on software. This is synthesizes the main idea of a reconfigurable radio, which has its communications' systems functions implemented on software running on a multi-purpose processor.

The benefit of performing the required signal processing in software instead of using dedicated integrated circuits in hardware is that since software can be easily replaced in the radio system, the same hardware can be used to create many kinds of radios for many different transmission standards and consequently in a variety of different applications [6].

This reconfigurability idea was firstly defined on a paper published in May, 1995 by Joe Mitola on the IEEE Communication Magazine [23], but since 1990 the US Department of Defence military program SPEAKeasy had already conceived the software radio as a way of improving interoperability [30].

It's interesting to note that since the SDR concept is a new model of telecommunication's systems, it is just one of several stages that compose the evolution process of this technology concept maturity. This concept was firstly thought to ideally be an Adaptive-Intelligent Software Radio (AI-SR), which because of its extremely complexity for the time it was necessary to start developing models more simple like the SDR until the scientists and engineers reach and domain all the required electronic technology to deploy a Cognitive Radio (AI-SR)

The AI-SR will be a device capable of adapting to its operational environment aiming an optimized performance and spectral efficiency (better spectrum management) through transmitting and receiving signals at any allowed frequency, power level and bandwidth using any modulation technique. Its basic functionality is its radio ability to automatically adapt its own operation mode without human interference in order to reach those prior objectives in accordance with the external propagation, spectrum and network conditions, also meeting the QoS required on each type of service being executed at the time.

In order to execute these functions it will be necessary a high level of artificial intelligence, high-performance processors, real-time adaptive algorithms including blind source detection ones, minimum QoS protocols, etc. For example, the radio must adapt to the actual environment in real time, its own waveform to an optimized one dynamically developed based on the actual environment attenuation levels requiring a high level support of a mobile infrastructure network.

The objectives mentioned in this chapter aren't to go deep inside the AI-SR world, but to justify its base technology, the SDR, and to prove that it's reliable while supporting the basic principle of the future telecommunication's systems.

Back to the SDR domain, it's very important to remind that a digital radio is a radio where the information is digitalized somewhere between the antenna and the I/O devices (keypad, microphone, speaker, display). But not every digital radio can be a SDR. A radio can be digital, but if the signal processing after the analog/digital conversion is done by an Application-Specific Integrated Circuit (ASIC) it's not a SDR.

Only a digital radio with a multi-purpose processor that allows flexible reconfiguration of almost all functions used to implement the communication system is a SDR. An example of such processor is the Field Processing Gate Array (FPGA) which accepts many different software programs that implement different modulation techniques, sample rates, error detection and correction algorithms, cryptography levels, transmission rates, spectral efficiency (bits per Hertz), among many others.

The main objective of a SDR system is to digitally sample the signals as close as possible to the antenna, while on the receiver mode and to convert the digital signals into analog as close as possible to the antenna on the transmitter mode. Then process the digitized signals in a multi-purpose processor using a reconfigurable software. In Figure 3.1 there is the SDR concept flow graph [12].

Nowadays, there are some test-beds that allow the prototype of the some functionalities of a SDR system and are composed by two main parts: one hardware module and one software module, Parts 1 and 2 respectively, on Figure 3.1. One detail that must be observed is that the SDR concept was originally conceived for mobile communications, so the software will be running on the mobile stations or on the network's base stations when the system will be deployed. But, while using a test-bed the software will be running on a PC.

The basic functions shown in Figure 3.1 are distributed as following: the software module process in real time the signals and virtually implement any type of modulator, demodulator, filter, oscillator or any mathematical operation required to the signal processing and the hardware module accommodates the receiver and the transmitter antennas, makes the acquisition and conversion between analog and digital signals, etc.

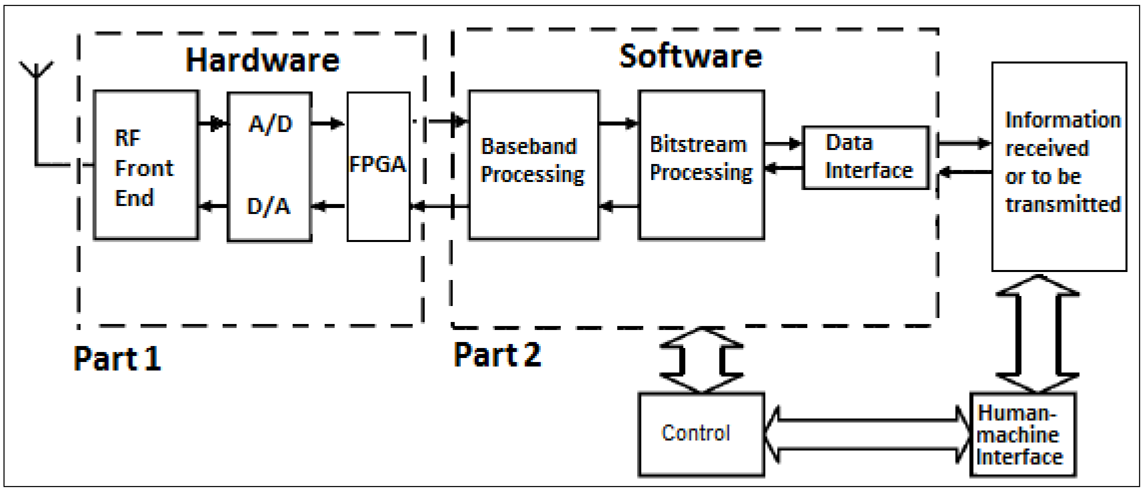


Figure 3.1: SDR concept flow graph

A short description of each block of the flow graph shown on Figure 3.1:

- Part 1:

FPGA: The hardware processor/function manager;

A/D and D/A: Are the converters from analog to digital and vice-versa;

RF Front End: the analog hardware interface;

The frequency shift between IF or higher frequencies and baseband is performed by DDC's and DUC's in the USRP

- Part 2:

  Baseband processing: signal modulation, demodulation, baseband filtering, etc.;

  Bit-stream Processing: manage the information received or to be transmitted digitally re-sampling the message to adequate the sample rate for transmission or reception and all the digital processing over the 'pure' signal including the real-time external control by human-machine interface; and

  Data interface: software external I/O drivers and ports able to receive and transmit informations inside and outside the processing software to/from external devices like speakers, screen, storage files, etc.

- Control:

  Allows the user/network to control the software for transmission or reception of informations desired based on

  Information from mobile network infrastructure;

  Available RF bands;

  Available air protocols;

  User needs;

  Minimum performance requirements;

  etc;

  It's also possible to control the FPGA via software by customizing its image before loading it !

Like in many technologies development processes, the market starts offering some different options of hardware and software for standard development kits. In the SDR area there are many options of both hardware and software, but one in special offers the two most important characteristics desired on a kit: high performance hardware with wide frequency range and bandwidth and a free-license open-source software that best supports the hardware.

This chosen hardware is the USRP and the software is the GNU Radio both described on the next two sections.

3.2 The hardware module Universal Software Radio Peripheral (USRP)

The USRP is a low-cost software defined radio platform that implements the Part 1 (hardware) on Figure 3.1 and it's made by the Ettus Research [27]. It has been the choice of thousands of engineers worldwide for the SDR hardware for algorithm development, exploration and prototyping.

It is originally designed for RF applications from DC to 6 GHz, supports Multiple-Input Multiple Output (MIMO) antenna systems operations, etc. Some applications areas are: white spaces, mobile phones, public safety, spectrum monitoring, radio networking, cognitive radio, satellite navigation and amateur radio.

One USRP module is composed by one motherboard, some daughter-boards, some analog/digital converters (ADC's), some digital/analog converters (DAC's), a central processor unit, a computer interface and a power supply. The Figure 3.2 shows the basic architecture of the USRP first motherboard model (USRP1).

There is a family of models of motherboards having different configurations listed on Table 3.1 and a family of daughter-boards also having different configurations listed on Table 3.2, both found at [27].

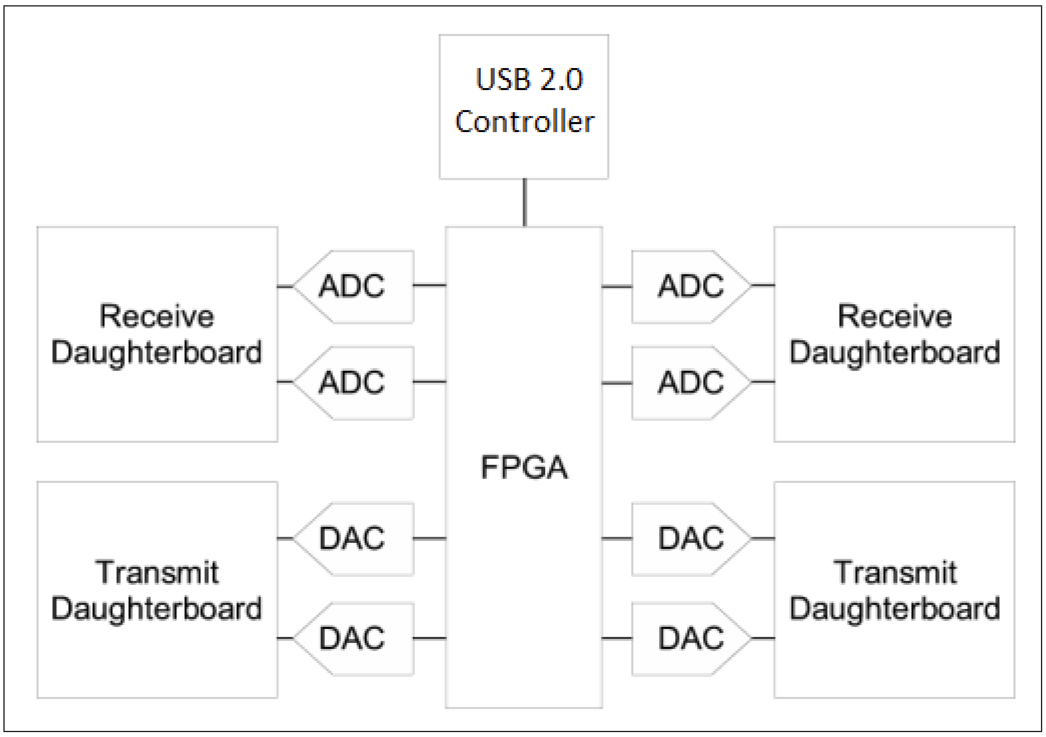


Figure 3.2: USRP1 motherboard architecture

Table 3.1: USRP Motherboards Family

Model	Channel	Interface	BW	DAC	ADC	Cost US
N200*	1 Tx/Rx	GigE	50	16-bit,400 M sps	14-bit,100 M sps	$1,500
N210*	1 Tx/Rx	GigE	50	16-bit,400 M sps	14-bit,100 M sps	$1,700
E100	1 Tx/Rx	Emb.*	4-8	14-bit,128 M sps	12-bit,64 M sps	$1,300
E110	1 Tx/Rx	Emb.*	4-8	14-bit,128 M sps	12-bit,64 M sps	$1,500
USRP1*	2 Tx/Rx	USB 2	16	14-bit,128 M sps	12-bit,64 M sps	$700
B100	1 Tx/Rx	USB 2	16	14-bit,128 M sps	12-bit,64 M sps	$650

Table 3.1 keys:

* = MIMO capable;

BW = Bandwidth in MHz using 16-bit sample; and

Emb* = Embedded CPU model OMAP 3730.

Table 3.2: USRP Daughter-boards Family

Model	Type	Frequency	BW	Power out	Noise (dB)	Cost US
TVRX2	Rx	50-860 MHz	10	n/a	4-10	$200
RFX900	FD	750-1050 MHz	30	200 mW	5-10	$275
RFX1200	FD	1150-1450 MHz	30	200 mW	5-10	$275
RFX1800	FD	1.5-2.1 GHz	30	100 mW	5-10	$275
RFX2400	FD	2.3-2.9 GHz	30	50 mW	5-10	$275
WBX	FD	50 MHz-2.2 GHz	40	100 mW	5-10	$450
SBX	FD	400 MHz-4.4 GHz	40	100 mW	5-10	$475
XCVR2450	HD	2.4-2.5 GHz	33	100 mW	5-10	$400
DBSRX2	Rx	800 MHz-2.35 GHz	1-60	N/A	N/A	$150
LFTX	2 Tx	DC-30 MHz	60*	1 mW	N/A	$75
LFRX	2 Rx	DC-30 MHz	60*	N/A	N/A	$75
Basic Tx	2 Tx	1-250 MHz	100*	1 mW	N/A	$75
Basic Rx	2 Rx	1-250 MHz	100*	N/A	N/A	$75

Table 3.2 keys:

*Only when the two ports are used as a complex pair;

FD = Full-Duplex;

HD = Half-Duplex; and

BW = Bandwidth in MHz;

From Table 3.1 the USRP1 model was chosen for two main reasons:

- 1) low-cost; and
- 2) two transmitter and two receiver sides which allows two modes of operation:

  four communication channels: two single transmitter and two single receiver;

  two communications channels: one complex transmitter and one complex receiver.

The USRP1 motherboard, which is depicted on Figure 3.3 [13] has a FPGA model EP1C12 responsible for the coordination of activities like adjusting the signals' transmission (interpolation) or reception (decimation), data flow control, intermediate frequency shift from/to baseband, high-speed operations like digital up/down conversions and so on.

The main parts of this motherboard are the four high-speed ADC's with a resolution of 12-bits per sample, sampled at a rate of 64 M samples per second (sps) and four high-

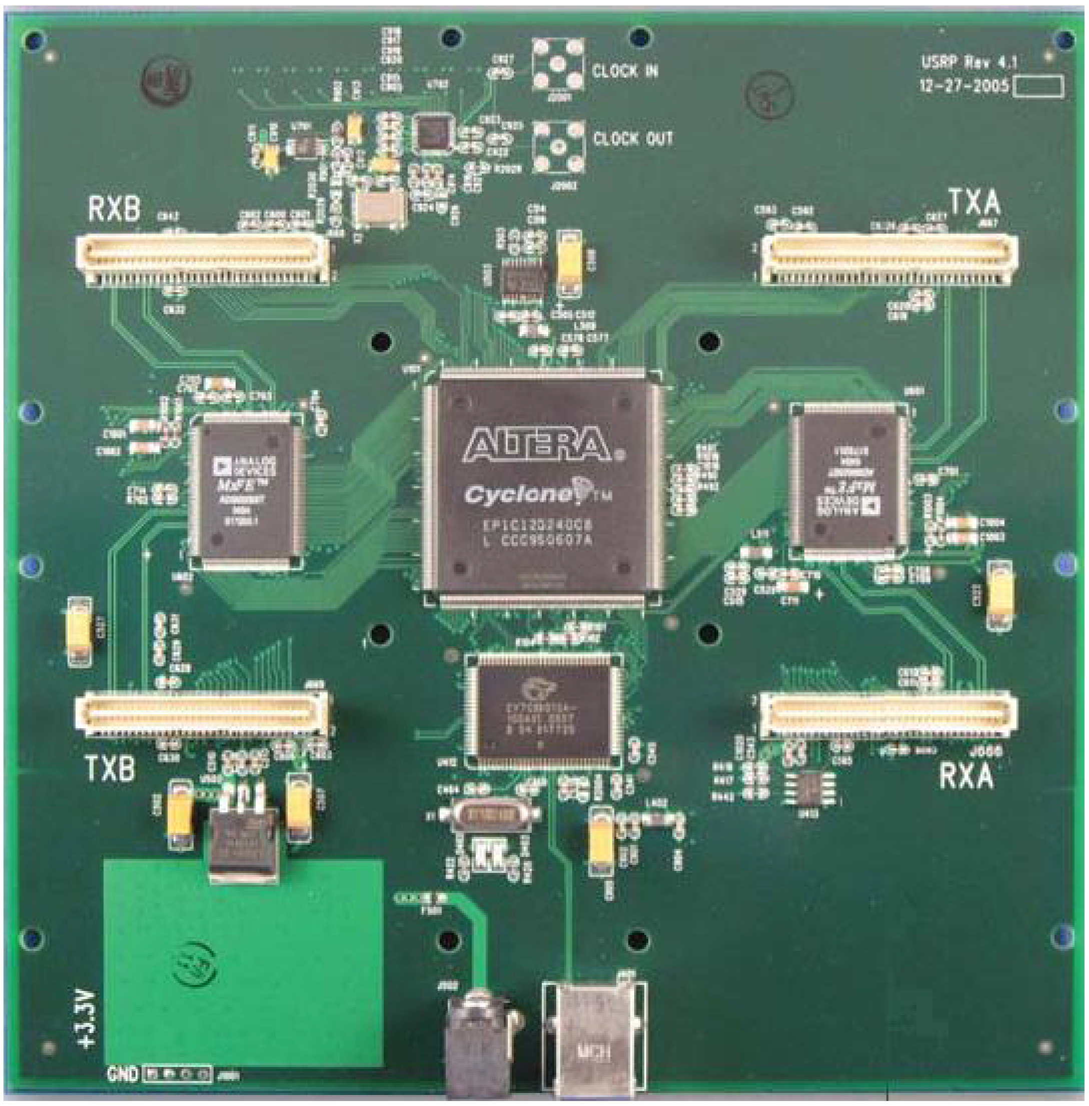


Figure 3.3: USRP1 motherboard

speed DAC's with a resolution of 14-bits per sample sampled at a rate of 128 Msps. The ADC's and DAC's are directly connected to the FPGA which uses an USB 2.0 interface to connect with a computer that has the GNU Radio software running on it [13].

It's interesting to note that the FPGA includes the digital down converters (DDC's) in the Rx path. They're implemented with 4 stages cascaded integrator-comb (CIC) high-performance filters and 31 tap half-band filters cascaded with these CIC's for spectral shaping and out of band signals rejection.

The basic function of the DDC is to down-convert the incoming signal's frequency from Intermediate Frequency band (IF) to baseband., after that it also decimates the signal in order to adapt the ADC sample rate to fit the USB 2.0 data rate limitation. At the Tx path the digital up-converters (DUC's) firstly interpolate the signal to achieve a higher sample rate for the DAC and later convert it from baseband to IF band delivering the resultant signal to the DAC. Part of the DUC's, the CIC interpolators are inside FPGA but all the rest, including frequency up-converters (complex multipliers) are inside the DAC chip [13].

It isn't explicated on Table 3.1 nor on Table3.2, but the USRP1 can operate on Full-duplex mode with its four RF channels (two full-duplex systems) and the only constraint is that the USB 2.0 total data rate must be considered, this is, the sum of the amount of all Tx and Rx paths must be 32 M Bytes per second (Bps) or less.

Another particularity of the USRP1 module is that it uses complex sampling, this implies that all the USRP internal and the transmitted and received signals are in complex format and each sample sent over the USB interface is composed of its real and imaginary part, 16-bits or 8-bits each, resulting in a 4-bytes or 2-bytes size complex sample.

The total received or transmitted rate by the USRP1 through the USB interface using 4 B complex-sample size is:

$$\frac{32\ M\ Bps}{4\ B\ per\ complex\ sample} = 8\ M\ complex\ samples\ per\ second \tag{3.1}$$

These samples are complex with real and imaginary parts, so the Nyquist sample rate of 8 M complex samples per second (8 Msps from real branch and 8 Msps from imaginary branch) allows the acquisition of an effective maximum complex bandwidth of 8 MHz (4 MHz real and 4 MHz imaginary) [13],[25],[27]. On Table 3.1 the BW value of 16 MHz is achieved by the trade-off between bandwidth and precision when the sample size is reduced from 16-bits to 8-bits each, which gives a 2 bytes complex sample. A little bit more about the USRP complex sampling, data rates and bandwidth are detailed on Chapter 6.

In order to allow the communication between any USRP module and the real external world there are the daughter-boards which are responsible for implementing the RF Front End of the SDR system on the Part 1 of the Figure 3.1. Its main function is to implement a connection with an external antenna and to syntonize the on-board local oscillator as nearest as possible to the target frequency for transmission, shifting the baseband spectrum to a higher desired frequency or for reception where the high frequency signal will be down-converted to IF frequency.

Some of the daughter-boards have two connectors on the same board to be used as a complex pair, like in the LFTX, LFRX, Basic Tx and Basic Rx shown respectively on Figure 3.4. These are the models used for all the prototypes tests described on this text. As seen on Table 3.2, each model is responsible for covering an specific frequency range of the spectrum. However, the range covered by each one of them is very wide and different. But there is a common range between them allowing that applications become realizable using more than one daughter-board like the ones which operates on the frequency range from 1 MHz up to 30 MHz considering these four different models.

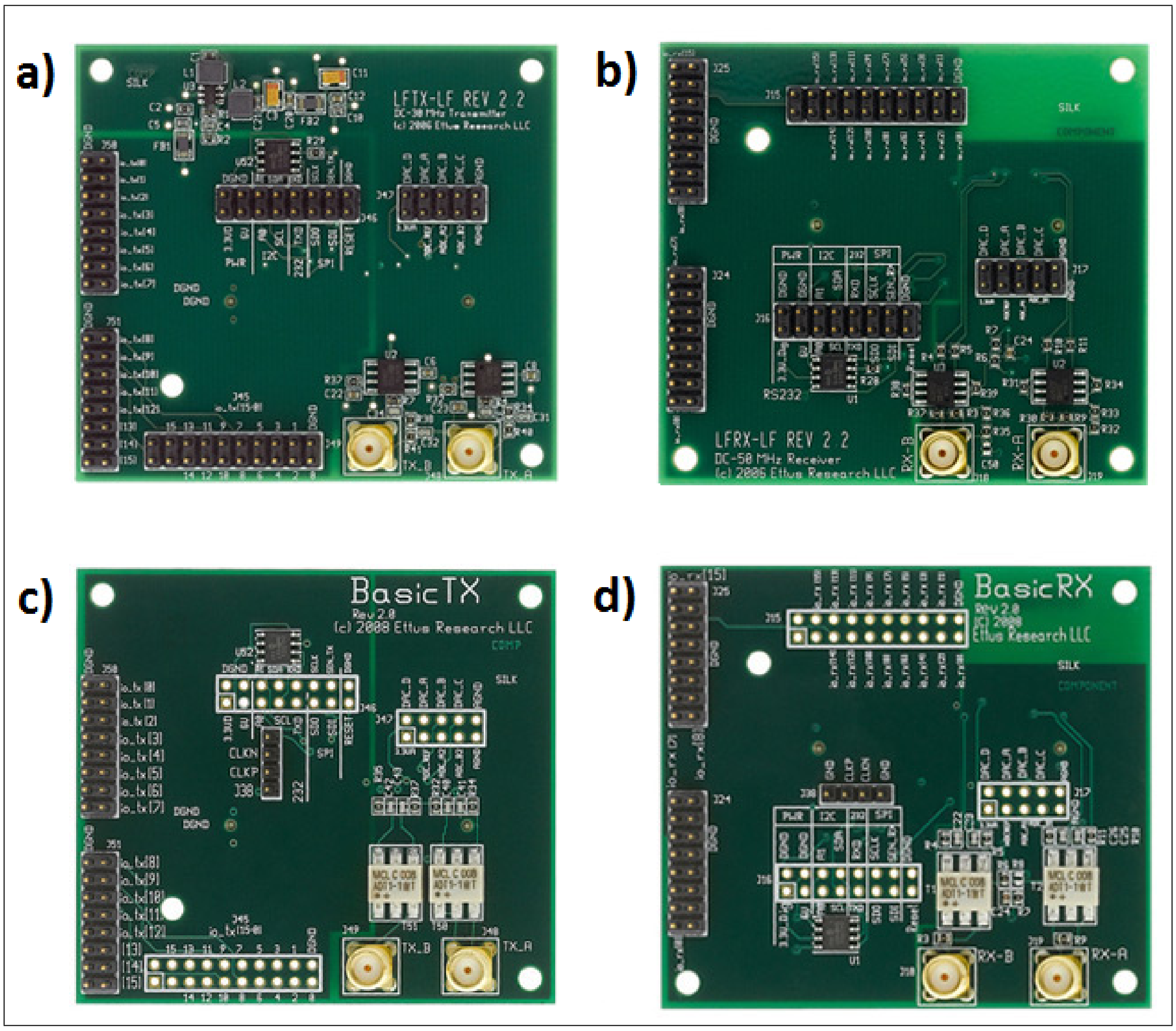


Figure 3.4: USRP daughter-boards

3.3 The software module GNU Radio

The GNU Radio is the software that implements the Part 2 (software) on Figure 3.1. It's an open source license-free development tool kit which runs on Linux Ubuntu and implements the software radios using a readily-available, low-cost external RF hardware, like USRP module's family. It is widely used in academic and commercial environments to support wireless communications research as well as to implement real-world radio systems [6].

It can be downloaded from [7]. The installation guide can be found on [8]. It's licensed under the GNU General Public License (GPL) version 3. All of the code is copyright of the Free Software Foundation.

There is also other paid software solutions like LabVIEW^{TM} running on Windows XP/Vista/7 and Simulink® inside MATLAB® which offers support for Windows, Linux and Mac platforms [27].

The GNU Radio implements a communication system through some baseband processing blocks that were previously programmed to implement the desired functions like modulation, demodulation, filtering, coding, decoding, frequency up or down conversions, signal generators, equalization, synchronization, USRP interfacing and many others. Each of these blocks has some configurable parameters which must be set correctly in order to meet the design requirements and the USRP constraints. This software can handle only digital data, so analog hardware is used to convert this data between analog and digital and to shift the signals to the desired central frequency. That requirement aside, any data type can be passed from one block to another - like bits, bytes, vectors, bursts or more complex data types.

Some basic blocks are standard installed, but if it's necessary to add new or customized blocks its completely possible and feasible.

The concept behind GNU Radio communication systems relies on the flow graph principle in which some information is got from the source(s) block(s), then goes through a processing chain made from one or more processing blocks and finally gets out through the sink(s) block(s). Each processing block is implemented using C++ coding because of the performance-critical signal processing path and joint with a special 'coding glue' implemented through Python scripts which are, in fact, the source code of the communication system. On Figure 3.5 is depicted the GNU Radio development layer stack. An example of Python script is shown on Figure 3.6.

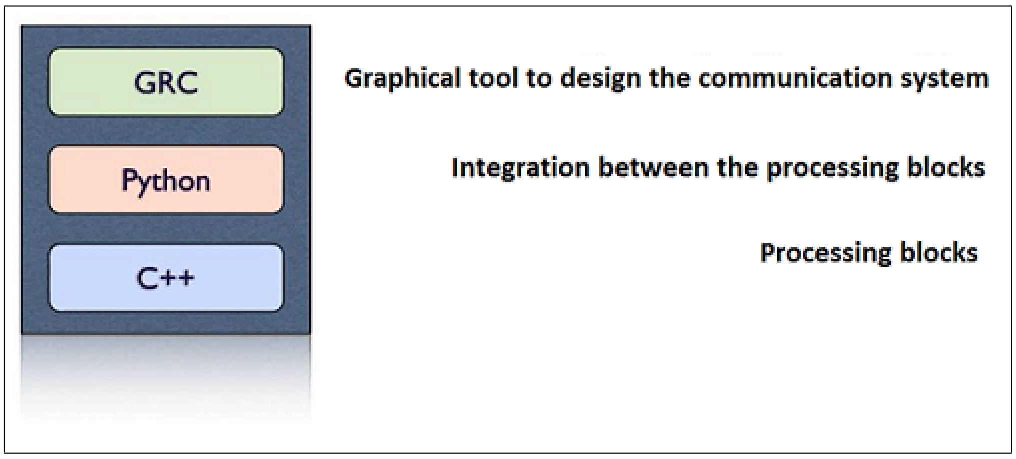

GRC
Python
C++
Graphical tool to design the communication system
Integration between the processing blocks
Processing blocks

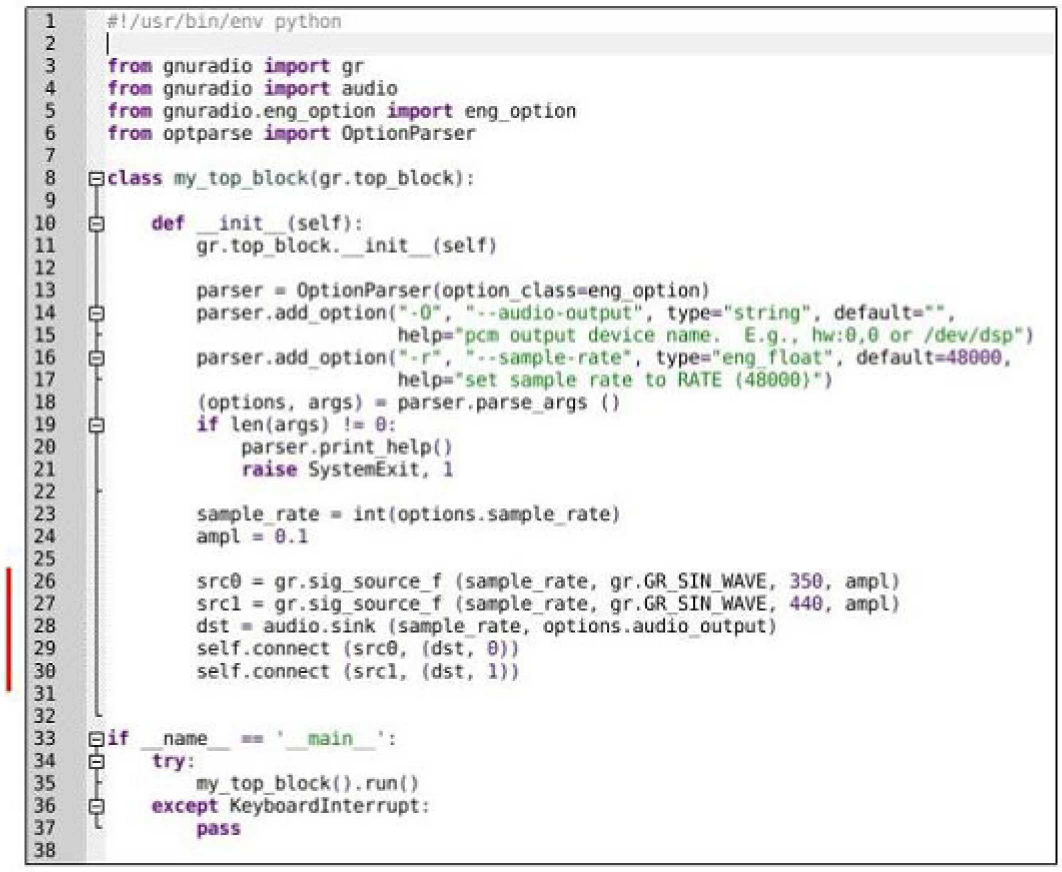
Figure 3.5: GNU Radio development layer stack

To make the building of a communication system simple, faster and intuitive, along with the GNU Radio comes a graphical interface software called GNU Radio Companion (GRC). This tool allows to build the systems in a visual drag-and-drop mode based on the blocks flow graph concept. After choosing all the processing blocks, setting each one of their parameters and linking all the blocks, for the communications system start working it must be ran from GRC. The first event of the GRC is to create a Python script based on the flow graph and deliver it to GNU Radio. From now on, the processing will run normally by GNU Radio since all the parameters had been set correctly.

Also, it's possible to create a complete communication system through command lines directly at the Linux terminal using the same GNU Radio processing blocks, however, it will be more difficult and take too long.

On Figure 3.7 there is a GRC flow graph designed to implement a Wideband FM Transmitter using GNU Radio. It's possible to see some processing blocks like the source, low-pass filter, band-pass filter, frequency modulator, sink, among many others. Also it is important to note the source and sink blocks. The source blocks can feed the signals into the communication system and can be a local file source or a signal generator, also they can be combined with many others. The sink blocks are the output blocks which are responsible for implementing the interface with the hardware module, like the USRP, where the virtual processed signals meet the real world and come to reality.

Through the flow graph is possible to connect some signal analyzers like oscilloscope and spectrum analyzer. This also allows to analyse the signal behaviour in time and frequency after each processing stage of the entire communication system. Another function



```
#!/usr/bin/env python

from gnuradio import gr
from gnuradio import audio
from gnuradio.eng_option import eng_option
from optparse import OptionParser

class my_top_block(gr.top_block):

    def __init__(self):
        gr.top_block.__init__(self)

        parser = OptionParser(option_class=eng_option)
        parser.add_option("-O", "--audio-output", type="string", default="",
                          help="pcm output device name.  E.g., hw:0,0 or /dev/dsp")
        parser.add_option("-r", "--sample-rate", type="eng_float", default=48000,
                          help="set sample rate to RATE (48000)")
        (options, args) = parser.parse_args ()
        if len(args) != 0:
            parser.print_help()
            raise SystemExit, 1

        sample_rate = int(options.sample_rate)
        ampl = 0.1

        src0 = gr.sig_source_f (sample_rate, gr.GR_SIN_WAVE, 350, ampl)
        src1 = gr.sig_source_f (sample_rate, gr.GR_SIN_WAVE, 440, ampl)
        dst = audio.sink (sample_rate, options.audio_output)
        self.connect (src0, (dst, 0))
        self.connect (src1, (dst, 1))


if __name__ == '__main__':
    try:
        my_top_block().run()
    except KeyboardInterrupt:
        pass
```

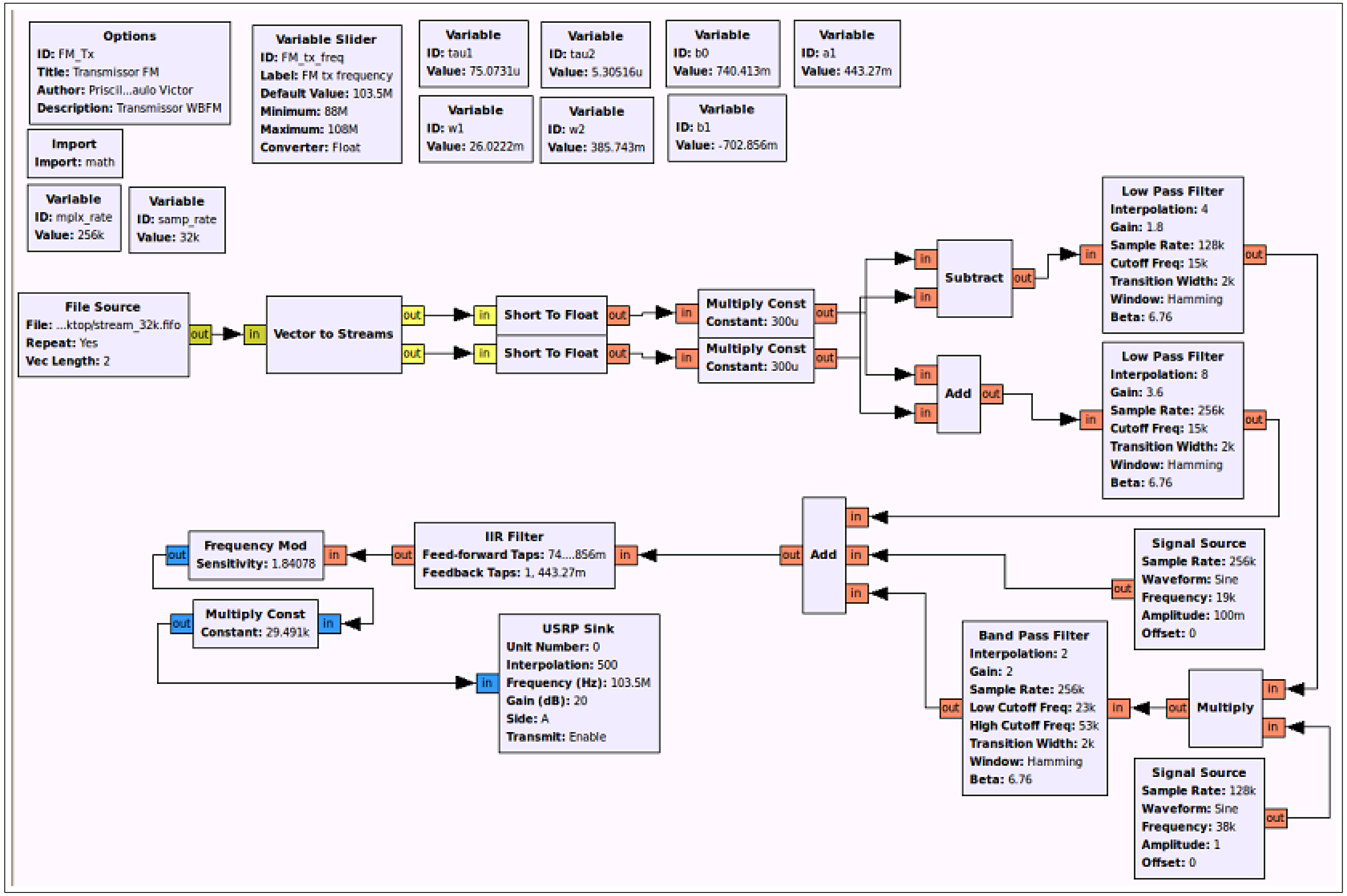
Figure 3.6: Python script

is to record the received signals, also from different points of the system. Some of the configurable parameters can be set during the execution of the system

3.4 Final considerations of this chapter

This chapter gave a basic insight on the SDR concept, the GNU Radio software and the USRP hardware module.

The next chapter will describe the design of the frequency response amplitude acquisition prototype. This prototype allows the acquisition of the frequency response of an electronic circuit using the SDR concept. A customized software based on GNU Radio software was completely designed to perform all the required tasks using the USRP hardware module.


Options
ID: FM_Tx
Title: Transmissor FM
Author: Priscil...aulo Victor
Description: Transmissor WBFM

Import
Import: math

Variable
ID: mplx_rate
Value: 256k

Variable
ID: samp_rate
Value: 32k

Variable Slider
ID: FM_tx_freq
Label: FM tx frequency
Default Value: 103.5M
Minimum: 88M
Maximum: 108M
Converter: Float

Variable
ID: tau1
Value: 75.0731u

Variable
ID: w1
Value: 26.0222m

Variable
ID: tau2
Value: 5.30516u

Variable
ID: w2
Value: 385.743m

Variable
ID: b0
Value: 740.413m

Variable
ID: b1
Value: -702.856m

Variable
ID: a1
Value: 443.27m

File Source
File: ...ktop/stream_32k.fifo
Repeat: Yes
Vec Length: 2

Vector to Streams

Short To Float

Short To Float

Multiply Const
Constant: 300u

Multiply Const
Constant: 300u

Subtract

Add

Low Pass Filter
Interpolation: 4
Gain: 1.8
Sample Rate: 128k
Cutoff Freq: 15k
Transition Width: 2k
Window: Hamming
Beta: 6.76

Low Pass Filter
Interpolation: 8
Gain: 3.6
Sample Rate: 256k
Cutoff Freq: 15k
Transition Width: 2k
Window: Hamming
Beta: 6.76

Signal Source
Sample Rate: 256k
Waveform: Sine
Frequency: 19k
Amplitude: 100m
Offset: 0

Multiply

Signal Source
Sample Rate: 128k
Waveform: Sine
Frequency: 38k
Amplitude: 1
Offset: 0

Band Pass Filter
Interpolation: 2
Gain: 2
Sample Rate: 256k
Low Cutoff Freq: 23k
High Cutoff Freq: 53k
Transition Width: 2k
Window: Hamming
Beta: 6.76

Add

IIR Filter
Feed-forward Taps: 74....856m
Feedback Taps: 1, 443.27m

Frequency Mod
Sensitivity: 1.84078

Multiply Const
Constant: 29.491k

USRP Sink
Unit Number: 0
Interpolation: 500
Frequency (Hz): 103.5M
Gain (dB): 20
Side: A
Transmit: Enable

Figure 3.7: GRC flow graph of a Wideband FM Transmitter

Chapter 4

A low-cost electronic circuit frequency response amplitude acquisition prototype

This chapter describes the development of a prototype for frequency response amplitude acquisition of electronic circuits. The prototype uses the software defined radio concept and is composed by two main parts: the communication system software based on GNU Radio and the USRP hardware module.

4.1 Objective

The main objective of this prototype development is to create a low-cost tool for acquisition of the frequency response amplitude of an external electronic circuit. In this text the external electronic circuit under test is the PLC-USRP interface. It's important to know exactly the attenuation level of such device through the PLC operating frequency range and to make any correction if necessary while using the interface on future PLC signal analyzer systems.

The second objective is to use this prototype to acquire the frequency response amplitude of the transceiver system itself, in order to know the frequency response amplitude of all the different possible configurations allowed with the four USRP daughter-boards used. With the transceivers' frequency response amplitude it's possible to choose the best configuration, which is characterized by the one with the smallest attenuation levels through the entire frequency range to be used on the acquisition of the frequency response amplitude of the interface [12].

The four daughter-boards used are the following: BasicTx and BasicRx both covering the 1-250 MHz range and LFTX and LFRX both covering the DC-30 MHz. Even though they cover a very wide and different frequency range, there is a common range between all of them, allowing that transmission and/or reception applications can be deployed using more than one configuration. This fact justifies that a comparative study between the frequency response amplitude of the four different transceiver system configurations be done.

4.2 The prototype

Basically, the frequency response amplitude acquisition prototype must transmit a frequency varying signal to the device under test, syntonize the current transmission frequency at the receiver and finally, read and save the measured level of the received signal. When acquiring the PLC-USRP interface frequency response amplitude, the interface is connected between the transmitter and the receiver. But as the first tests will be the acquisition of the transceiver response, the transmitter and receiver are connected directly.

Using the SDR concept, most part of the prototype is software-based, developed on GNU Radio, and only the RF interface and sampling are done using the USRP hardware module. The first part of the prototype is the transmitter side and is the most simple. The second part is the receiver side, which is a little more complex. Both were developed on GRC using the GNU Radio processing blocks and some additional Python codes [12].

The prototype was developed based on the USRP model USRP1 and on the software GNU Radio v.3.2 installed on the Ubuntu 10.04 LTS.

4.2.1 The transmitter path

The transmitter major objective is to make the frequency sweep. It is composed by two software processing blocks: the signal generator block, known as Signal Source and the USRP interface block known as USRP Sink. The signal generator block allows the generation of some different kinds of waveforms at a maximum frequency which is theoretically half the maximum sampling rate, but as seen in Chapter 3, section 3.2, it shall be limited by the maximum USRP interface with the computer, which in this case is the USB 2.0. The USRP sink block is used to configure the transmission frequency (for frequency shifting), the sampling rate used on the transmitter path, the connector used by the antenna, the interpolation factor and all the parameters related to the configuration of the USRP transmitter path.

The frequency sweep was made between the common frequency range of the used daughter-boards: from 1 up to 30 MHz. Three initial approaches were tested and are summarized on Figure 4.1. Each of them has some constraints, which imply on some trade-off's.

In the first scenario, a constant level baseband signal was generated to be transmitted through the whole frequency range by the USRP and whose frequency was increased at a discrete amount after a constant time interval up to the desired frequency limit . At the transmitter side of USRP, the frequency shift, done by the DUC's located at the DAC's chips, from baseband up to IF is basically an AM modulation. The problem with this first scenario relies on the fact that the recovered signal at the receiver is a constant level, so it's impossible to make any phase shift measurement. But its amplitude level measurement is the simplest one to be implemented because it is only necessary to read the current sample value. The constant level baseband signal is software generated and the frequency varying carrier is generated on the DUC.

In the second scenario a baseband cosine signal was generated and its frequency was varied from 1 up to 30 MHz using the signal generator block. In this case, the transmission frequency was set to zero in order to avoid any frequency shift from baseband to IF, thus the DUC wasn't used. This technique allows the phase shift measurement, because the received signal may be a cosine. However, using the USRP1 model the implementation of this scenario is completely impracticable due to the high sampling rate required, which is not supported by the USB 2.0. As will be seen later on this chapter, the amplitude level measurement is kind of complicated and implies that the sample rate can be much more increased than just by the Nyquist criteria.

Nowadays, with the most recent USRP model N210, which uses Gigabit Ethernet interface, this second scenario can be deployed and may allow that some tests could be made.

The third scenario, the most practical, is a mix of the two previous scenarios and that's why it is used throughout the following chapters. The baseband signal is a cosine signal with a fixed frequency. This signal modulates the carrier signal, whose frequency varies through the desired range. It allows phase shift measurement and requires a simpler amplitude level measurement than the second scenario. Also requires a low baseband sample rate, which is based on the fixed frequency chosen. In this method the baseband is software synthesized and the IF is hardware synthesized.

One interesting issue raised during the tests, which will be better explained in the final of this chapter, is the phase shift measurement problem.

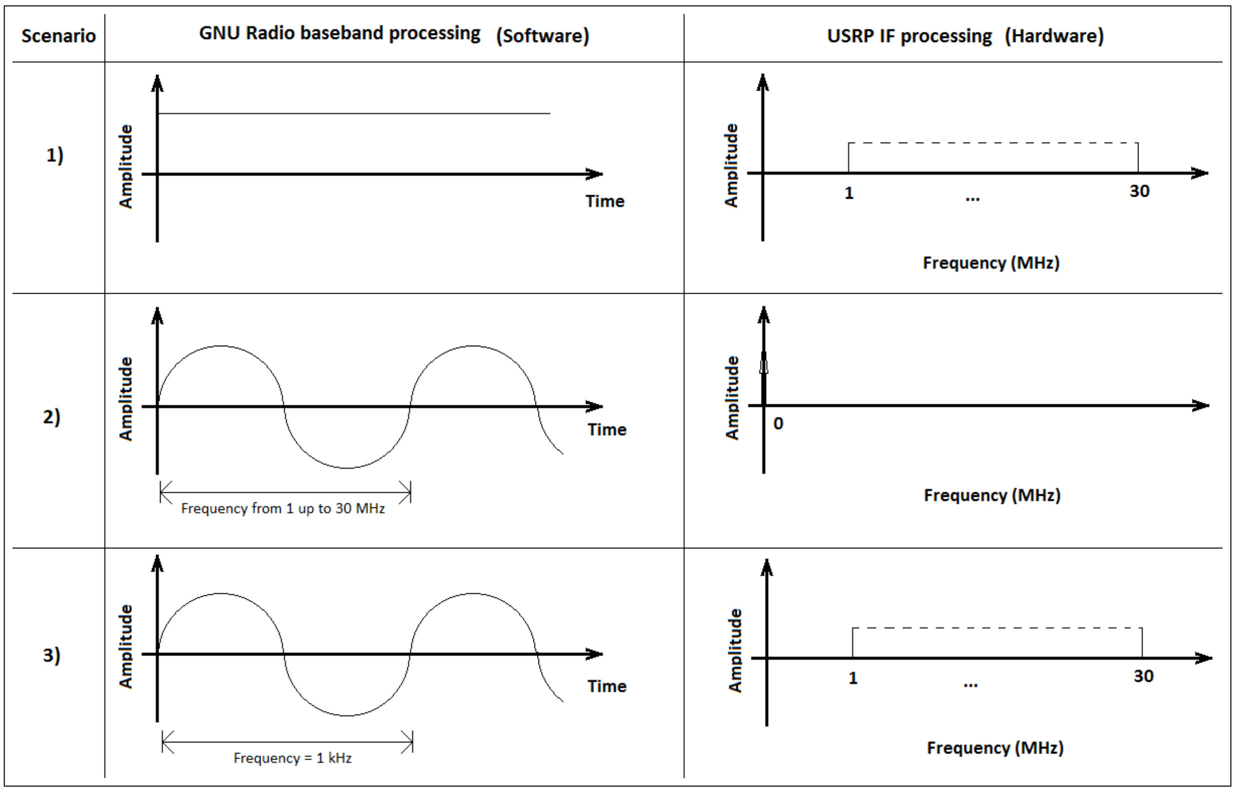


Figure 4.1: Approaches for signal generation at transmitter side

The connection configurations and the values shown on the GRC blocks will be discussed on the Section 4.3 together with the frequency response amplitude results.

The GRC flow graph of the transmitter is shown in Figure 4.2 with the Signal Source and USRP Sink blocks. There is also two additional blocks, Complex to Real and Complex to Imaginary which converts the Signal Source complex output into its two components, the real and the imaginary ones. Each converter is followed by a Probe block which isn't implemented on GRC, only on GNU Radio. To save the measurement levels at each reading, a small routine was developed. This simple routine intended to get the current sample amplitude reading and save it in a new row of a txt file, which is already stored on the same directory of the prototype's source code. Each txt file stores measurement data only from the flow that it is attached to. The Probe code and this routine are written in Python and run just after a call from the main routine asking for a frequency change. The transmitter measurements are read and stored for further processing purposes like comparing the received signals with the transmitted ones. When estimating the phase shift, the transmitter's probe outputs are used right away.

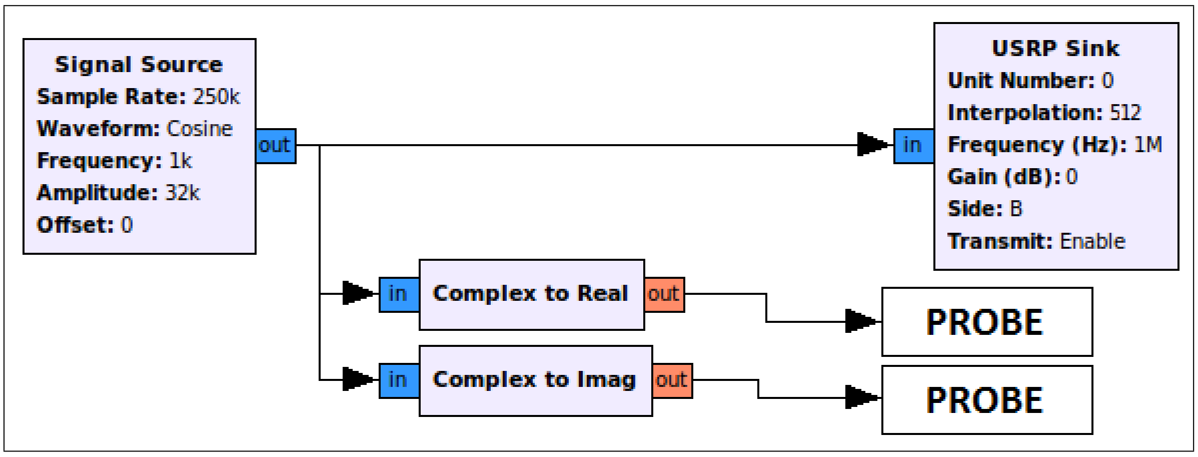


Figure 4.2: Transmitter GRC flow graph

4.2.2 The receiver path

At the receiver side, the USRP Source configures the reception frequency (for frequency shifting), the decimation factor, the connector of the receiving antenna, and all the parameters related to the configuration of the USRP receiver path. After the Source, the signal is already inside the computer. Firstly it passes through a low-pass filter to get rid of high frequency noise. After the filter there are the Complex to Real and to Imaginary converters. Following each converter there is a Probe block, which is used in the same way as the ones in the transmitter path and are also used for phase shift estimation. After the Complex to Real block there is a simple measurement arrangement which was designed based on some conclusions from Subsection 4.2.3. Finally, after the measurement set there is another probe used for the amplitude level measurement. This receiver flow graph is shown in Figure 4.3.

The results of the two different receiver daughter-boards for the tests using the same transmitter daughter-board were subtracted one from another for comparison reason only. Considering that the transmitted and received signals are complex, the phase shift or difference was calculated simply by the difference between the arctan of the received signal and the arctan of the transmitted signal.

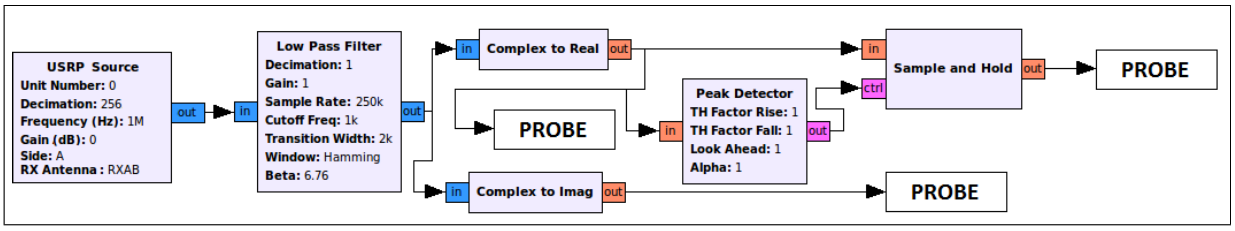


Figure 4.3: Receiver GRC flow graph

4.2.3 Amplitude measurement on GNU Radio baseband signals

Aiming to guarantee that the baseband measurements made using the GNU Radio based prototype can be trustful, a simple test system was developed. It was composed by a cosine signal generator with an amplitude equal to 100 (as the following tests will be performed internally to GNU Radio only, in this subsection consider this value as a reference). Its frequency swept from DC up to 1 MHz. In order to measure the maximum amplitude of the generated signal, the probe block (which is in fact the probe script previously described) was connected directly to the generator's output. So, in the end, all the measurements made should be equal to 100.

Initially, the chosen sample rate was the double of the maximum frequency to be sampled, 2 M samples per second (sps). A total of 100 measurements were made and they were also equally spaced in frequency. Just for graphical reasons, probe block's output was the squared root of the squared signal, just to get only positive values.

After the first test, it was observed that the probe function wasn't synchronized with the signal source so the amplitude was measured at different instants resulting in random values from 0 up to 100. To solve this, a pair of processing blocks composed by a peak detector and a sample and hold (S&H) was added to the input of the probe block seeking to assure that the maximum peak is constant and positive. In this way the probe input will always be a constant signal with the current maximum signal peak and the probe can read this value without being synchronized with the signal generator.

However, after this change there was an improve in the results only for lower frequencies, whereas in higher frequencies the measurements were still varying randomly but within a smaller range. This led to the adding of more pairs in series at the probe input.

The results only started to become constant at 100 level after adding 15 pairs in series for a signal having 1 MHz maximum frequency and a constant sample rate of 2 Msps. The results of this experiment are shown in Figure 4.4, where the red line represents the measurements without the pair, the green line represents the usage of only one pair and the blue line represents the usage of 15 pairs.

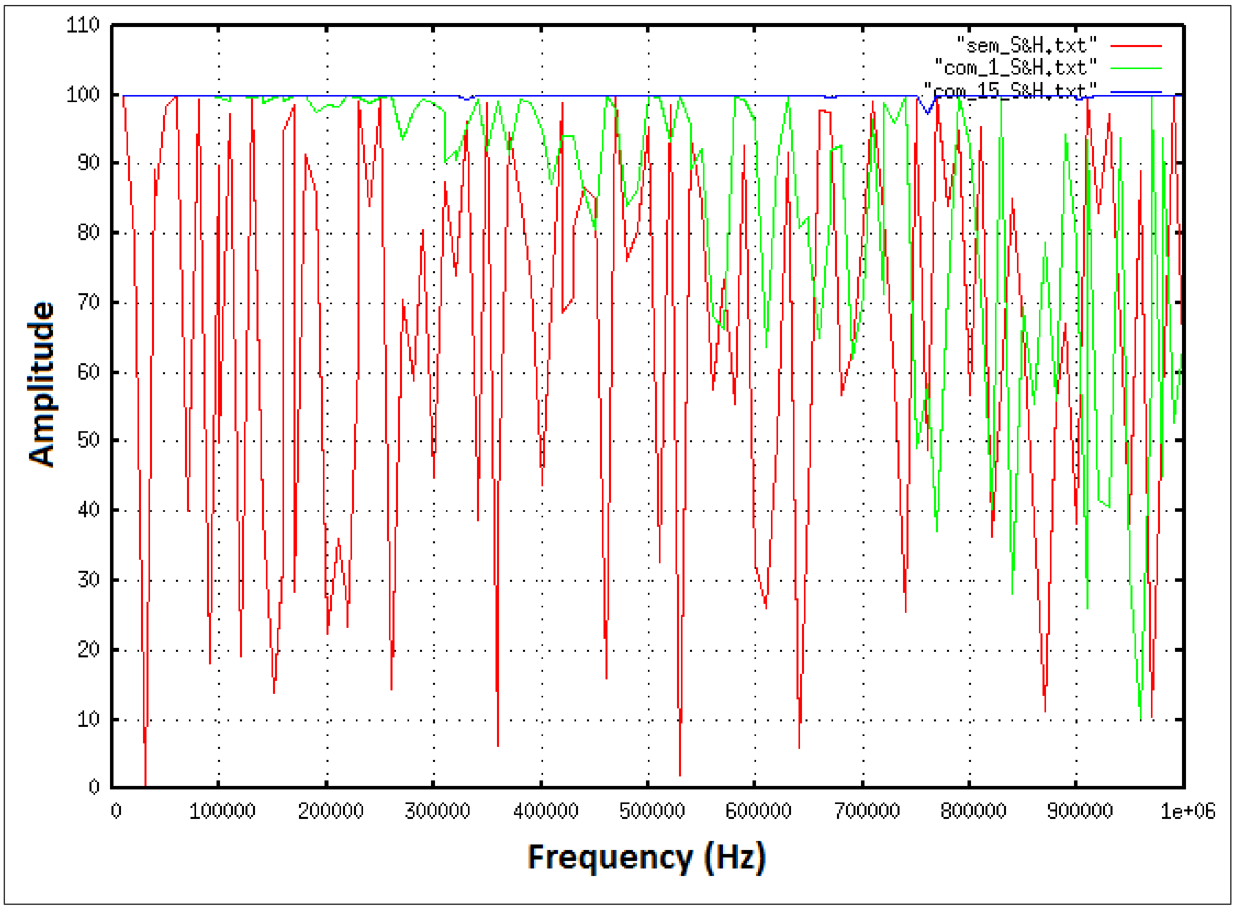


Figure 4.4: Amplitude measurement without the pair, with one pair and with 15 pairs

Afterwards there was an attempt to look for a measurement system with a reduced number of pairs, but with the same or even better performance obtained while using 15 pairs. Altering just one variable, the sample rate to 8 Msps and keeping only one pair of peak detector-S&H with the frequency sweeping from DC up to 1 MHz, this new specification could be achieved.

But as the desired operational frequency range of the amplitude measurement system prototype is from 1 MHz up to 30 MHz the sample rate required would up at least 128 Msps, using only one pair, which is unachievable for both ADC and USB 2.0 interface. Another tests showed that it would be necessary a proportional number of pairs in case of using a sample rate which is the double of the maximum frequency of the signal to be sampled.

Thus, further analysis shall be made with regard to the cost-benefit relationship of the computational performance between the advantages of using a higher number of pairs or a higher sample rate (also limited by the interface data rate between the USRP and the computer).

To show the efficiency of the measurement prototype proposed, a low-pass filter block (already implemented in GNU Radio and in GRC) was added between the signal generator and the 15 pairs. Its cutoff frequency was set to 500 kHz. The results are shown in Figure 4.5. This simple prototype allows the frequency response amplitude acquisition of any GNU Radio software processing block without the need of an external hardware module, like the USRP.

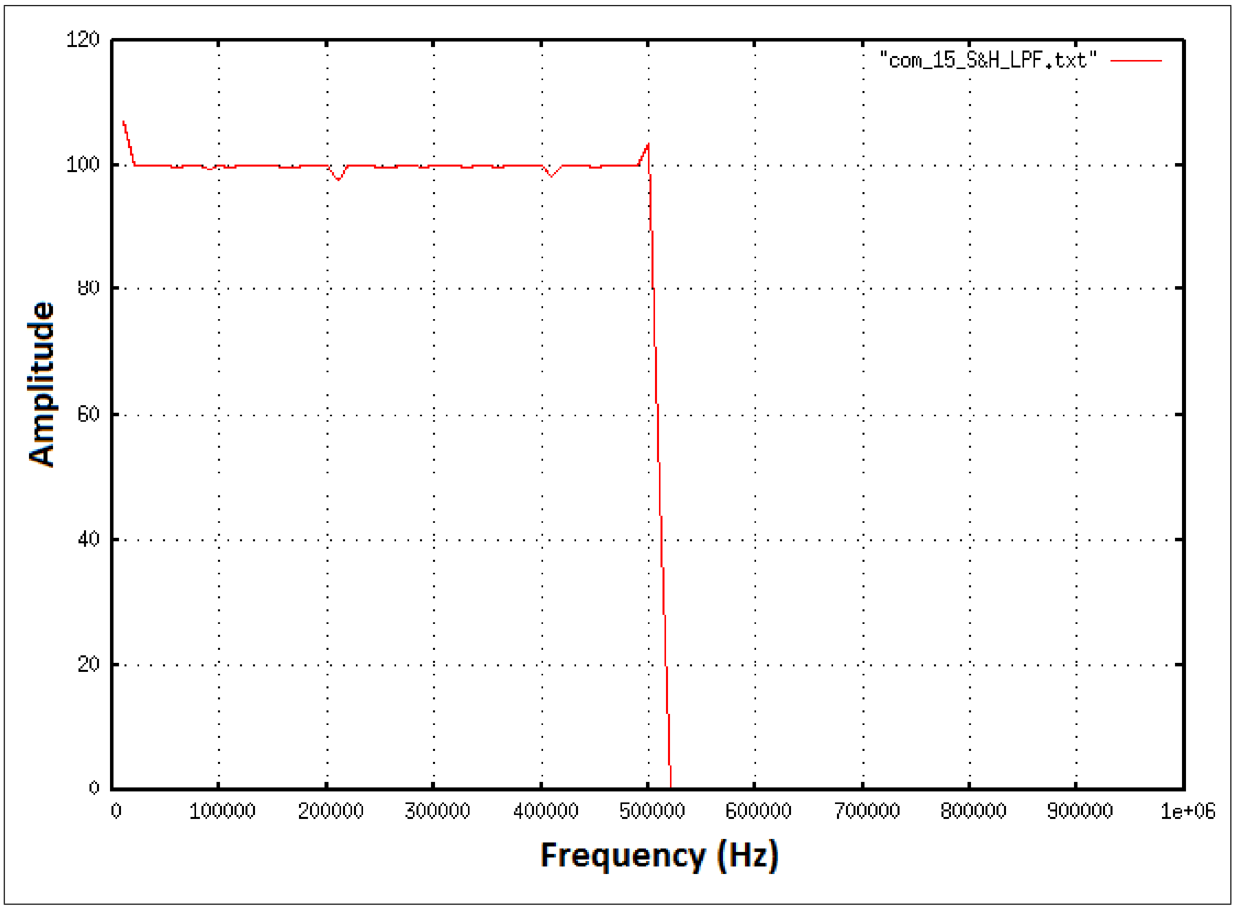


Figure 4.5: Amplitude frequency response of a low-pass filter

4.3 The prototype in action: semi-automatic method

Due to some transmitter and receiver paths constraints, a signal with amplitude and frequency known was used to modulate a frequency varying carrier through the entire frequency range in which the measurements must be made. This modulated signal then goes from the transmitter daughter-board to the receiver daughter-board. At the receiver side, the signal is demodulated and processed by the following receiver blocks. Thus, the prototype measures the amplitude variations on the baseband signal and make some calculus to find out the phase shift. This way it is possible to compare the transmitted amplitude level and phase with the received ones.

This results will show the frequency response of the transceiver system and allow the acquisition of the frequency response of any device connected between the transmitter and receiver daughter-boards. To do this is only necessary to subtract the transceiver frequency response from the frequency response acquired with the device under test connected to it.

All the four daughter-boards used have 2 connectors each, one for real signal and one for imaginary signal. For the acquisition of each transceiver configuration frequency response, the transmitter and receiver boards were connected by both I and Q channels using the standard coax cables included at the USRP kit and certified by Ettus Research labs.

For the four different daughter-boards configurations, the transmitter and receiver prototype configurations remained the same. A cosine signal was transmitted with a fixed frequency of 1 kHz (which will be detailed later on this section) and amplitude equal to 32000 resolution levels of DAC from a total of ± 32767 possible levels which proportionally relates to ± 1 V at the transmitter daughter-board output [6] [13].

The transmitter interpolated the signal by the maximum factor (512), in order to match the original signal sample rate (250 ksps) with the DAC sample rate (128 Msps). This resulting signal modulated a carrier, which frequency was varied from 1 MHz up to 30 MHz between 500 measurement points, 58 kHz apart from each other. The frequency change was made manually and used a virtual slider feature block from GRC.

The prototype code was built in a way that immediately after the main routine received a frequency change request, before executing it, the current probes levels were read and their values stored. Thus, it allows the frequency change interval to control the waiting time necessary for the system to achieve a steady-state after each new frequency call.

In this semi-automatic approach it wasn't possible to keep a constant time interval between each frequency change, resulting in some amplitude readings errors. Also the total test time could not be optimized. But the manual frequency variation was made in such a way that a considerable large time interval, on the order of some seconds, was guaranteed for most of all measurements returning acceptable and comprehensive results.

As soon as the transmitter frequency changed, the receiver tuned to it. Next, it was made the decimation by a factor equals to 256 to decrease the ADC sample rate from 64 Msps to 250 ksps (the original sample rate). After getting into the computer prototype software, the signal passed through a low-pass filter to get rid of the high frequency noise and any modulation image. The cutoff frequency was set to 1 kHz with a Hamming window with transition width of 2 kHz, which resulted in a very sharp filter response [12].

As mentioned before, for amplitude measurements only real signals were used. At the receiver the baseband signal was processed by a peak detector configured to detect the positive peak of a cosine waveform and to trigger the S&H block also during the negative cosine cycle. The S&H block sampled the value of its input signal at the moment of the trigger and held it until the next trigger.

In Subsection 4.2.1, it was shown that the approach using a fixed frequency baseband signal to modulate a carrier was a good one. Also, in Subsection 4.2.3, the two simplest amplitude measurement techniques proposed were to use a lot of peak detectors-S&H pairs in series or only one pair together with a very high sample rate, on the order of four times the minimum Nyquist.

Considering the minimum sample rate used at the transmitter side of the prototype (250 ksps) and the need to use a fixed frequency value for the baseband cosine signal, it was chosen the approach of using only one pair and a fixed frequency value 250 times lower than the sample rate, i. e., 1 kHz. Of course, lower frequency values than 1 kHz (not DC) also should have worked, but the value of 1 kHz was chosen after the sample rate issue, just for simplicity of being neither a high value nor a so low.

It seems that choosing a lower frequency value for the 'message signal' would affect the time interval of the response of the peak detector, in such a way that there will be a delay in the triggering time when comparing to a higher frequency value. Some further research could be done in order to analyse the effects of choosing a lower baseband frequency value if this prototype will be used for any other different applications. In the automatic method in Subsection 4.4 is shown that the time interval between setting a new frequency and reading its demodulated signal has sufficient periods of waveform for the prototype achieve the steady-state and allows a trustful measurement.

The first and second tests were performed using the transmitter daughter-board BasicTx. The first one used the receiver daughter-board BasicRx and the second used the receiver daughter-board LFRX. After these tests, the stored measured samples were subtracted and the difference between them was stored in another file. The results were plotted using the Gnuplot tool which needs that a Python package, called Gnuplot.py, be installed to use the Gnuplot from within Python, allowing to plot data on the fly as they are computed (just after all the measurements finish).

In Figure 4.6 [12] is shown the frequency response amplitudes for the first and second configurations mentioned above and their difference, where the configuration using the LFRX was subtracted from the one using the BasicRx. The red line is the configuration using the BasicRx, the green line is the configuration using the LFRX and the blue line is the difference between them.

The third and fourth tests were performed using the transmitter daughter-board LFTX. The third one used the receiver daughter-board BasicRx and the fourth used the receiver daughter-board LFRX.

In Figure 4.7 [12] is shown the frequency response amplitudes for the third and fourth configurations mentioned above and their difference, where the configuration using the LFRX was subtracted from the one using the BasicRx. The red line is the configuration using the BasicRx, the green line is the configuration using the LFRX and the blue line is the difference between them.

For comparison purposes only, in Figure 4.8 [12] there are the results of all the four previous tests and the two differences. The red line is the BasicTx-BasicRx configuration, the green line is the BasicTx-LFRX configuration, the light-blue line is the LFTX-BasicRx configuration, the pink line is the LFTX-LFRX configuration, the dark-blue line is the difference shown in Figure 4.6 and the brown line is the difference shown in Figure 4.7.

Figure 4.6 shows that the transceiver configuration BasicTx-BasicRx had a better performance with regard to the amplitude variation, i. e., suffered low attenuation through the frequency range from 1 up to 30 MHz, than the BasicTx-LFRX, which suffered high

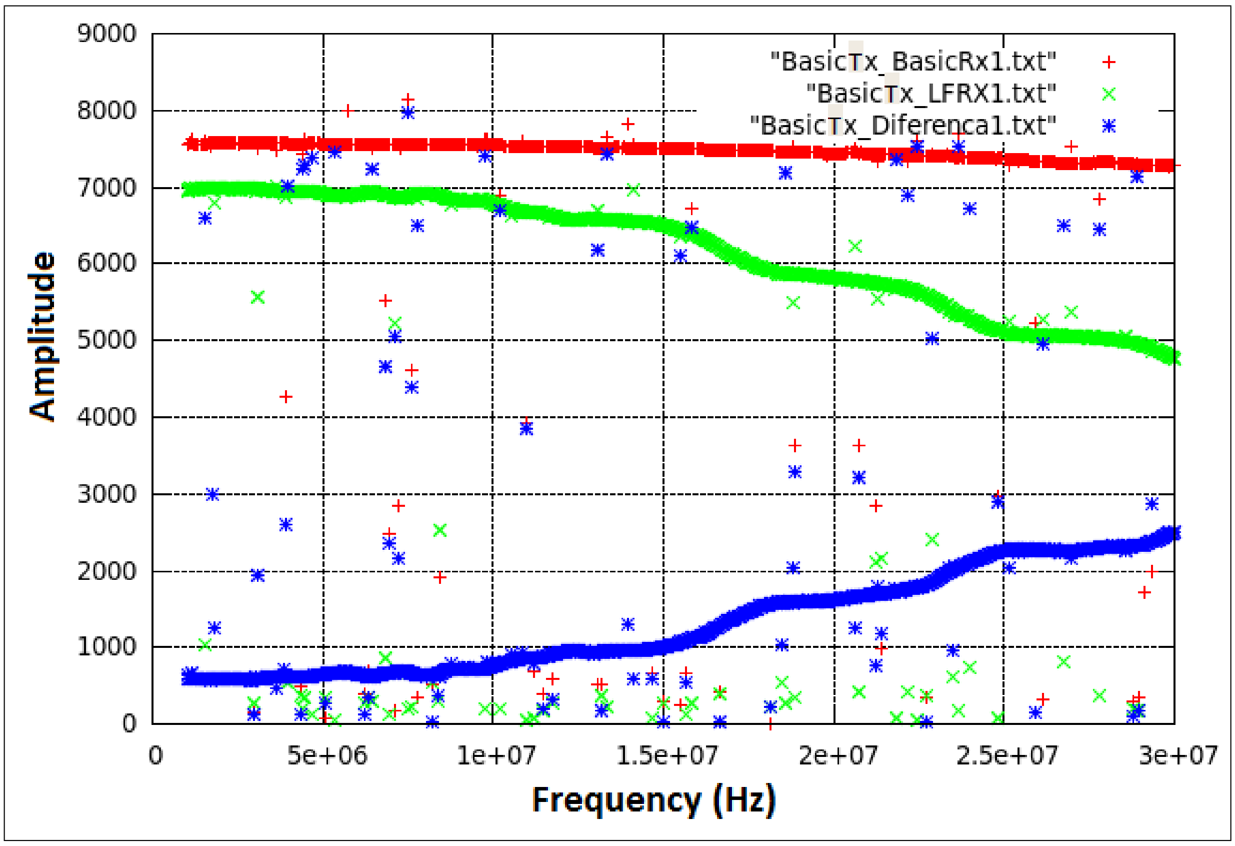


Figure 4.6: Frequency response amplitude of the transceiver system using the transmitter daughter-board BasicTx

attenuation from 10 MHz and above. Figure 4.7 shows that both configurations had very close performance up to 15 MHz. But the LFTX-LFRX suffered more attenuation than the LFTX-BasicRx considering the entire frequency range.

To summarize these comparisons, a general one can be made using the Figure 4.8 to select the worst and best configurations between the four possible ones. The BasicTx-LFRX showed to be the worst configuration with severe attenuation levels and the LFTX-BasicRx is the best configuration with only a very low attenuation starting at 28 MHz.

In addition to all the four previous tests made, another one was set just for curiosity reason after getting trustful tests results from the prototype. This test was made with the BasicTx-BasicRx and the basic prototype configurations, but with the frequency range changed to cover from 1 MHz up to 250 MHz and the frequency resolution to 498 kHz aiming to keep the same 500 measuring points. The frequency response amplitude is shown in Figure 4.9 [12]. It illustrates two shadowed transceiver frequency ranges: the first from 44.824 up to 84.166 MHz and the second from 172.810 up tp 212.152 MHz.

Seeking to find out if the transmitter was working well, a Wideband FM transmitter

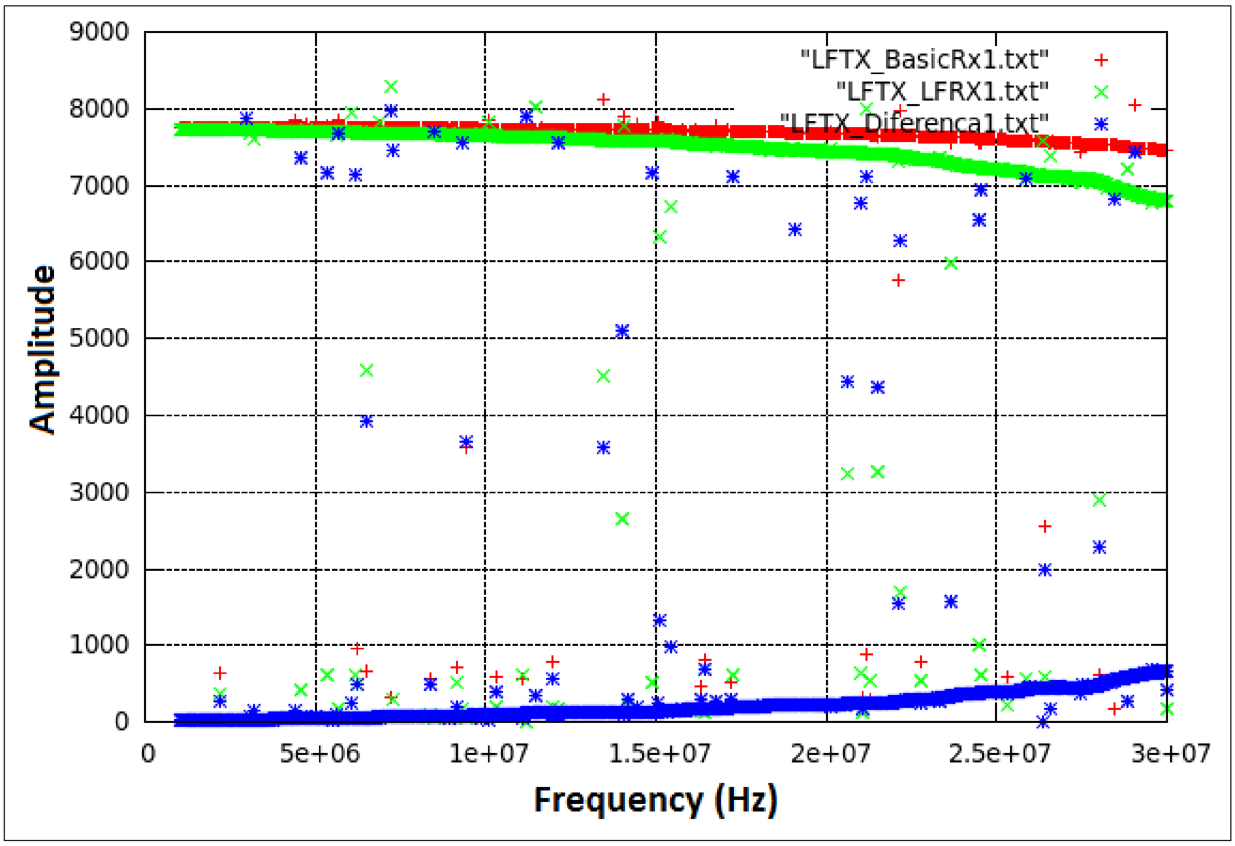


Figure 4.7: Frequency response amplitude of the transceiver system using the transmitter daughter-board LFTX

developed in GRC was turned on and the reception was tested using a receiver already implemented in GRC. At the frequencies outside the shadowed range the transmission and reception of the music occurred normally, but when the transmission frequency was any one inside of any of the two shadowed ranges the receiver also tuned at the same transmission frequency and nothing was heard.

Also, it was used a spectral analyzer model Agilent RF FieldFox Analyzer N9912A to test the transmitter side only. On this test a baseband 1 kHz cosine signal was transmitted with an amplitude equal to 32000 levels of DAC (almost 2 V peak-to-peak) which modulates a carrier varying between 1 up to 250 MHz. The BasicTx I output was connected directly to the analyzer input and as a result any reading of signal level could be made when the carrier was set to any of those two shadowed ranges, validating the previous prototype measurements. This test with the analyzer confirms that the transmitter path can not operate in those frequency ranges.

On the transmitter path there are two possible locations that may be responsible for this operational failure. The first one is the BasicTx transmitter daughter-board and the

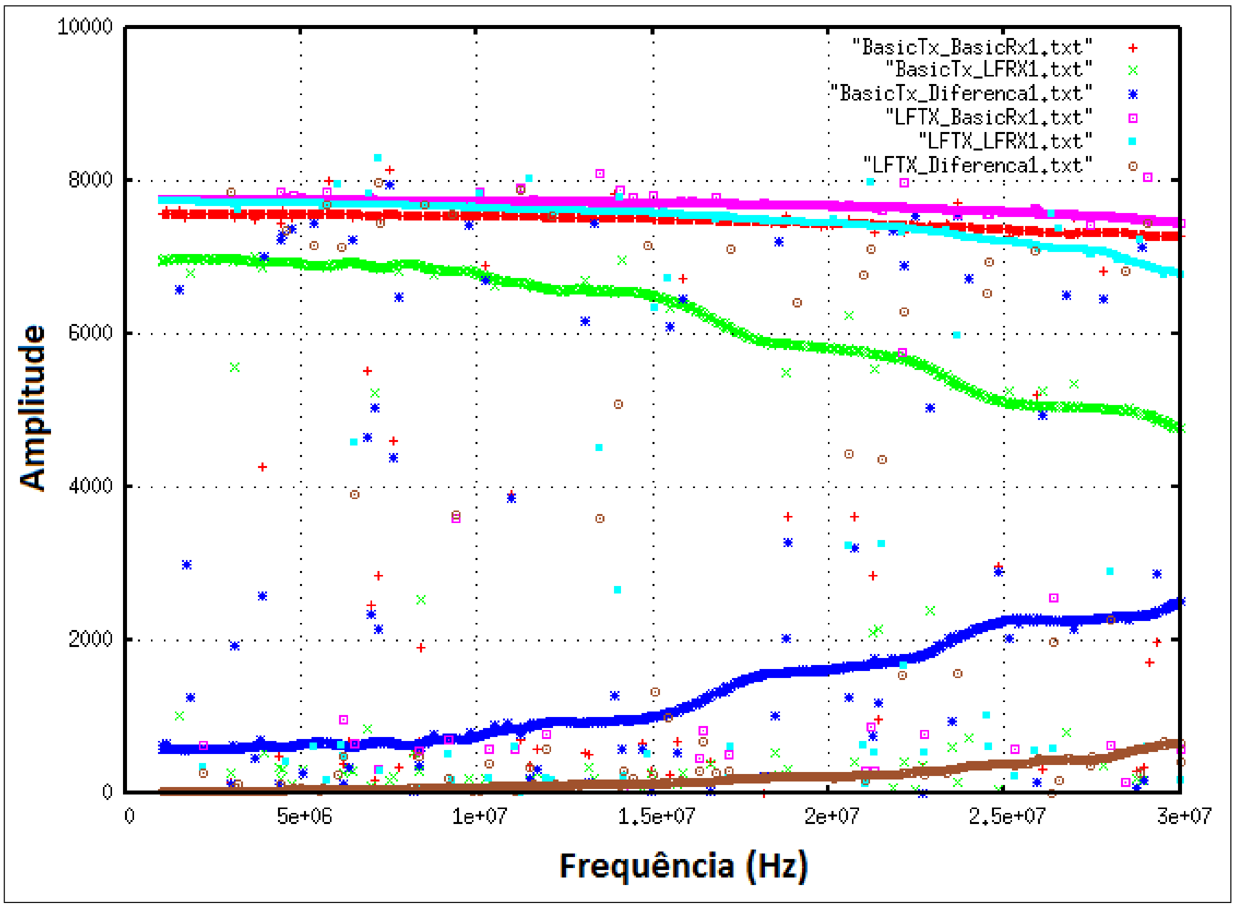


Figure 4.8: Frequency response amplitude of the four transceiver system configurations and their differences

second one is the DUC's installed inside de DAC's. Thus, to find which one of these is the root cause another transmitter daughter-board, which covers these shadowed ranges, must be tested with the spectral analyzer. If the failure happens again the failure may be due to the DUC's, if not, may be something wrong with that daughter-board. It's also important to note that the receiver path must also be tested using a transmitter operating on the shadowed frequency ranges just to assure that there is nothing wrong with it.

To conclude this section, in Figures 4.6, 4.7 and 4.8, it's possible to see some measurement dots far from the high density location of the majority. Maybe this is due to the fact that the system is semi-automatic with the frequency change calls being all manual. In such a situation the computer process can 'freeze' and when it starts again a wrong amplitude is generated as an error.

In order to solve that problem, to avoid the manual frequency change calls and give the prototype a higher control over some variables and a visual mode, an automatic method was developed and is described on the next section.

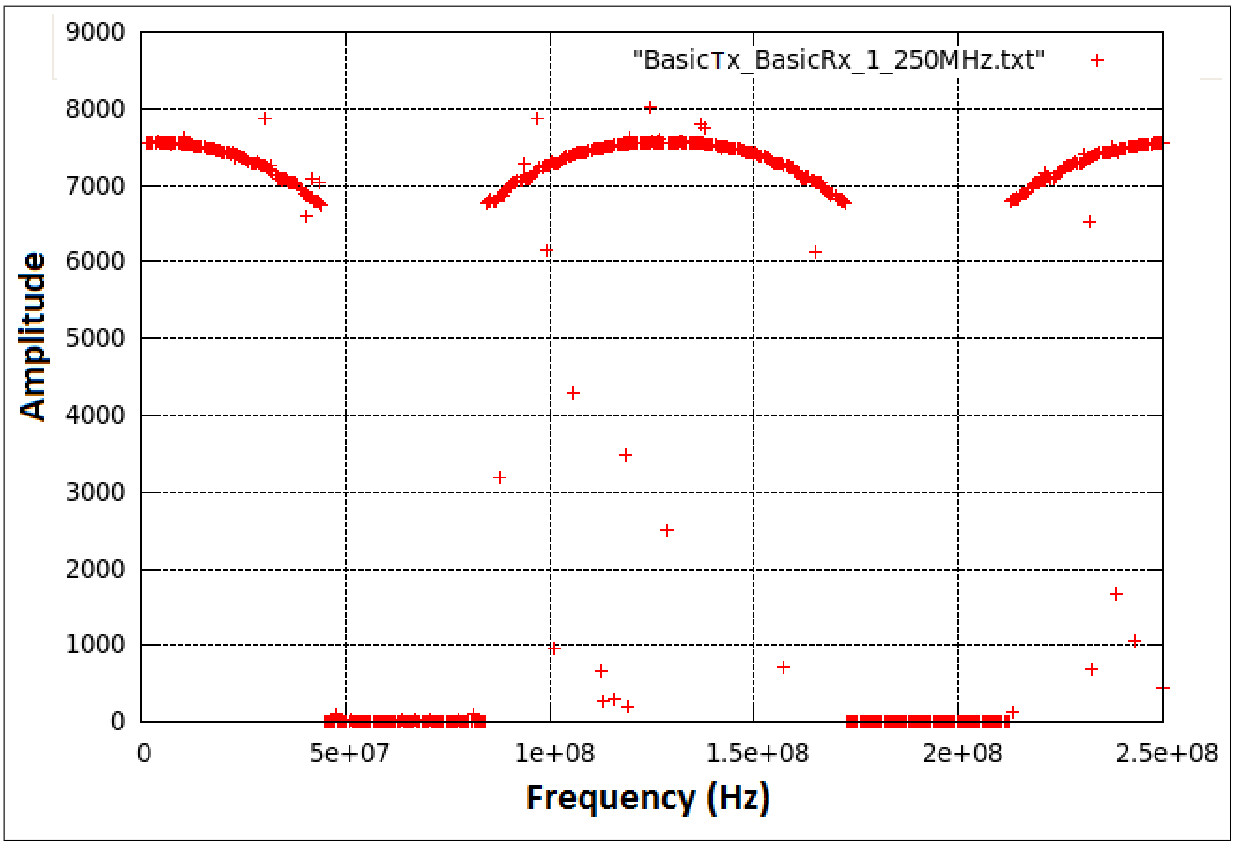


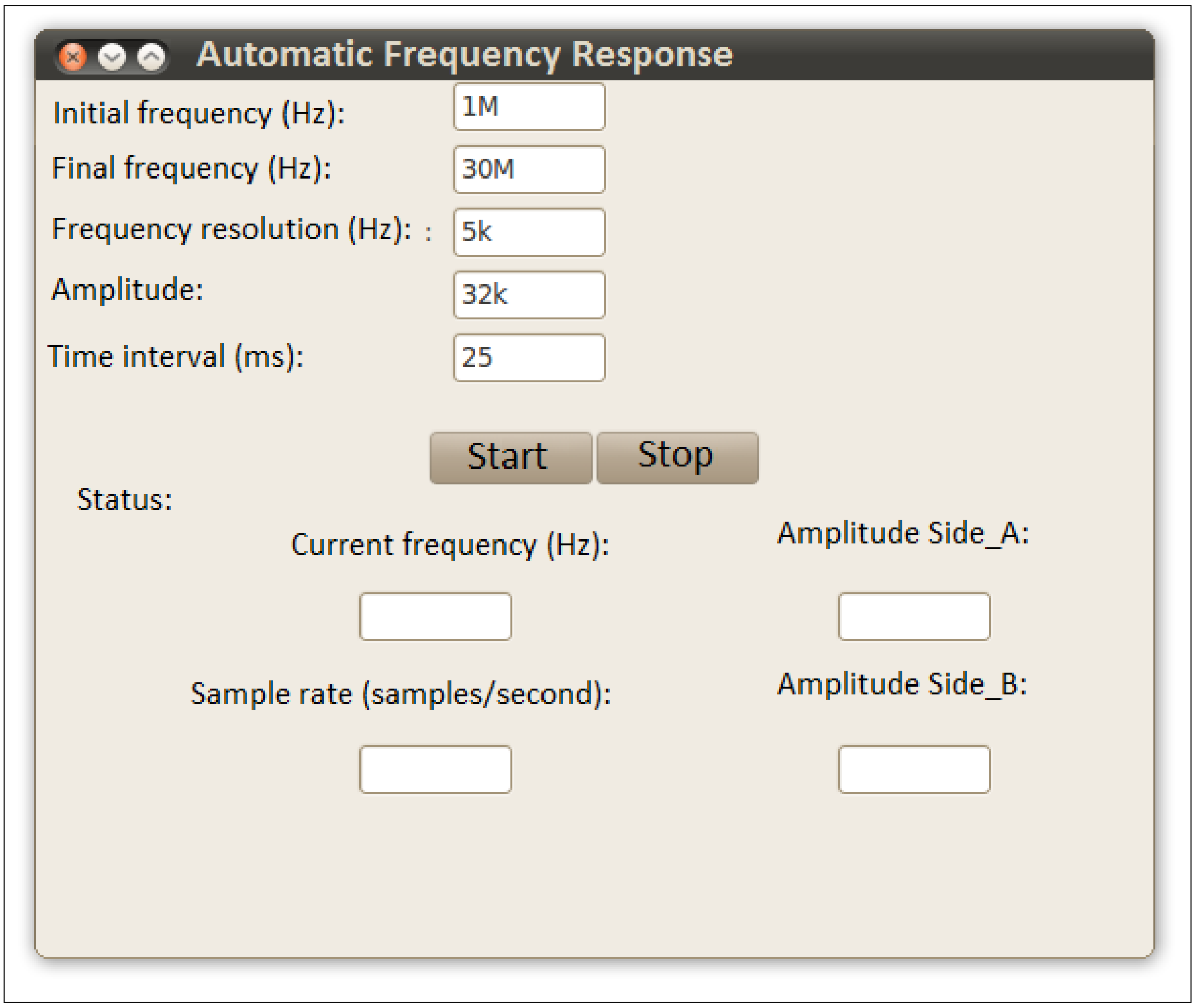
Figure 4.9: Frequency response amplitude of the transceiver system BasicTx-BasicRx from 1 up to 250 MHz

4.4 The prototype in action: automatic method

The idea of making the prototype to operate in an automatic mode is suitable for a user friendly approach, allowing to control some parameters through a graphical interface and to keep both, transmitter and receiver frequency variation and level probing, at a desired constant time interval. In the end the prototype returns a plot of all the measured levels through the frequency range covered on a graph. This complete version of the software is called 'Automatic Frequency Response' or AFR.

Through this interface the user can change some parameters like frequency resolution (referred to the density of equally spaced measurement points), time interval between measurements, frequency range and the transmitted signal amplitude. During the test execution, the user can follow the current test status informations, like the frequency under test, the just read amplitude, etc.

The automatic prototype was build using Python coding with the help of GRC used to build the same basic communication system used in the previous semi-automatic prototype as shown in Figures 4.2 and 4.3. A graphical interface for this automatic prototype was developed and is shown in Figure 4.10.


Automatic Frequency Response

Initial frequency (Hz): 1M

Final frequency (Hz): 30M

Frequency resolution (Hz): : 5k

Amplitude: 32k

Time interval (ms): 25

Status:

Current frequency (Hz):

Amplitude Side_A:

Sample rate (samples/second):

Amplitude Side_B:

Figure 4.10: AFR software graphical interface

This automatic version of the prototype allowed the search for a minimum time interval between the signal generation and transmission and its measurement at the receiver. This minimum time can optimize the test run when a higher frequency resolution test (smaller distance between the measuring points and consequently more measuring points) is required, like decreasing the time required to perform each test.

For this reason some tests were ran, basically with the four possible configurations between BasicTx, LFTX, BasicRx and LFRX daughter-boards. It was chosen some time intervals like 1 ms, 10 ms, 15 ms, 25 ms and 50 ms to perform the same tests. The configurations remained the same: frequency sweep from 1 up to 30 MHz with a cosine signal with amplitude equal to 32000 levels of DAC scale. All the next tests were made using a frequency resolution of 5 kHz, which means that the measuring points are equally spaced 5 kHz apart from each other resulting in 5800 measuring points, whereas on all the previous tests made with the semi-automatic method, the resolution used was 58 kHz with a total of only 500 measuring points. The change in the frequency resolution doesn't have any influence on the measuring system and only improves the plot details.

The results were compared and for 1 ms and 10 ms they are pretty much the same as illustrated by Figures 4.11 and 4.12. In Figure 4.11 the red dots are the transceiver configuration BasicTx-BasicRx, the green ones are from BasicTx-LFRX and the blue ones are the difference between them. In Figure 4.12 the red dots are the transceiver configuration LFTX-BasicRx, the green ones are from LFTX-LFRX and the blue ones are from the difference between them. On both figures the dots' patterns are very far from the ones of the semi-automatic method shown in Figures 4.6 and 4.7.

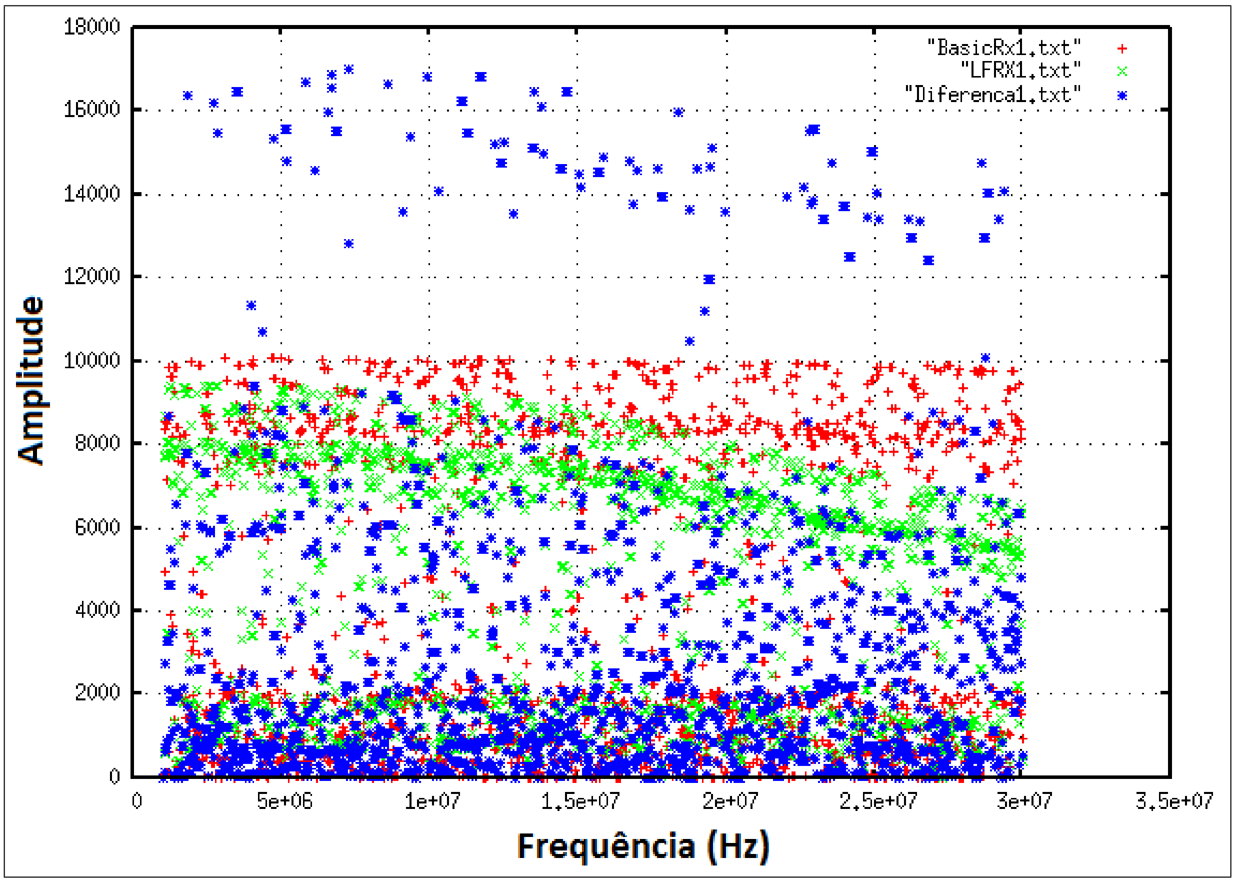


Figure 4.11: Amplitude frequency response with 1 ms time interval between samples for transceivers using BasicTx

After plotting all the results on graphs, the ones with 15 ms interval start to be close to those acquired with the semi-automatic method. But when it is changed to 25 ms and beyond it's possible to get good graphs with a more dense distribution pattern of all the measuring points. Also, as deeply discussed in the end of the previous Section, the

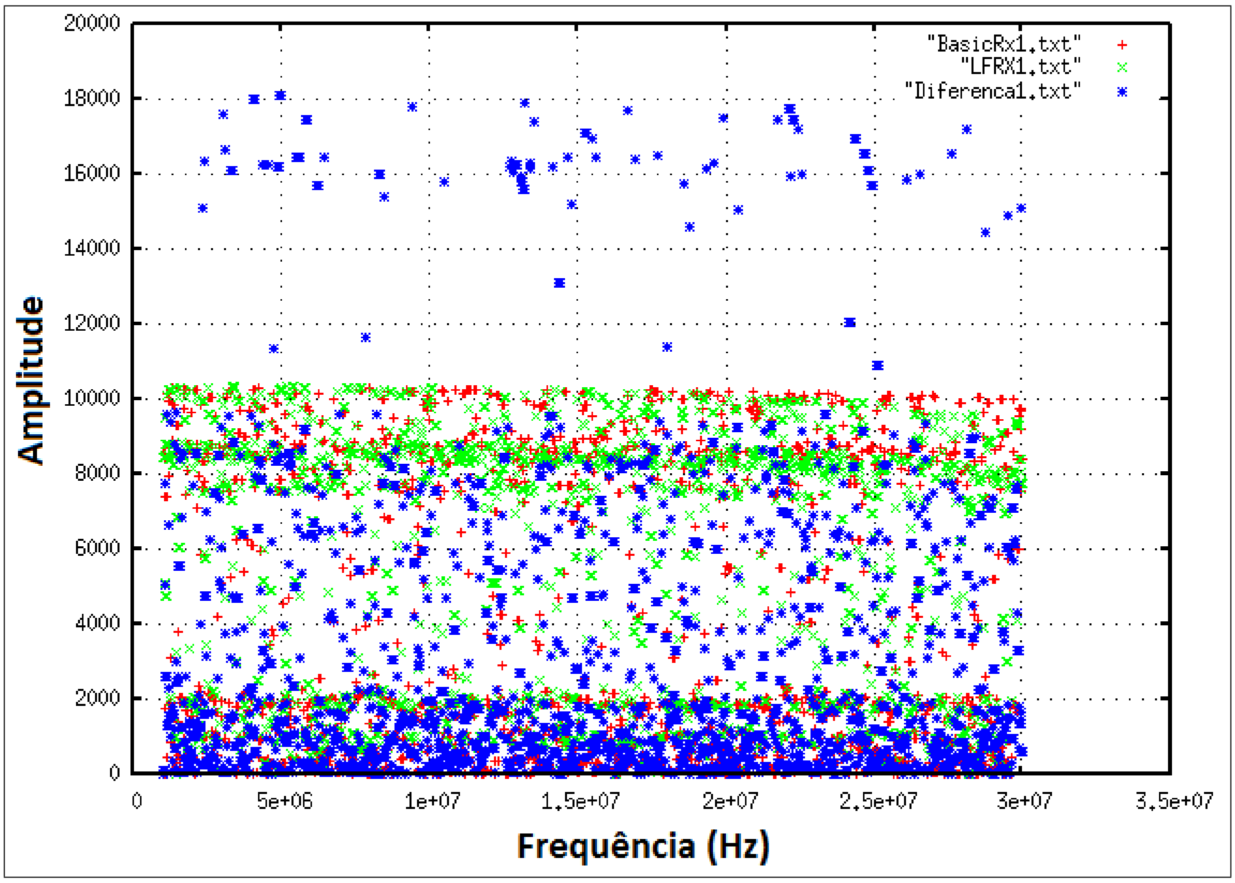


Figure 4.12: Amplitude frequency response with 1 ms time interval between samples for transceivers using LFTX

automatic method main characteristic of keeping a constant time interval between the measurements solved a big problem of the semi-automatic method: the random dots.

The Figure 4.13 shows that there was a huge reduction of random measuring points found in the semi-automatic graph as the one of Figure 4.8, which resulted in a cleaner graph. The keys for the transceiver configurations in Figure 4.13 are as follows: red dots represent BasicTx-BasicRx, green dots represent BasicTx-LFRX, blue dots are the difference between BasicTx-BasicRx and BasicTx-LFRX, pink dots represent LFTX-BasicRx, light-blue represent LFTX-LFRX and brown dots are the difference between LFTX-BasicRx and LFTX-LFRX. For each one of these four tests configurations, considering the frequency range covered (1-30 MHz), the frequency resolution (5 kHz) and the time interval between samples (25 ms) the total time wasted was 145 seconds.

Keeping a constant time interval between measurements it's no big deal but also looking for the best minimum time interval can improve the graphical results very much as long as to optimize the test running time. In addition to that the ideal and constant interval time between measurements seemed to result into less random values when different time intervals were tested and compared with the semi-automatic results in which the time-interval wasn't constant and varied randomly for each new measurement. Also, the higher measurement points resulted in a more precise and accurate response for a very wide bandwidth.

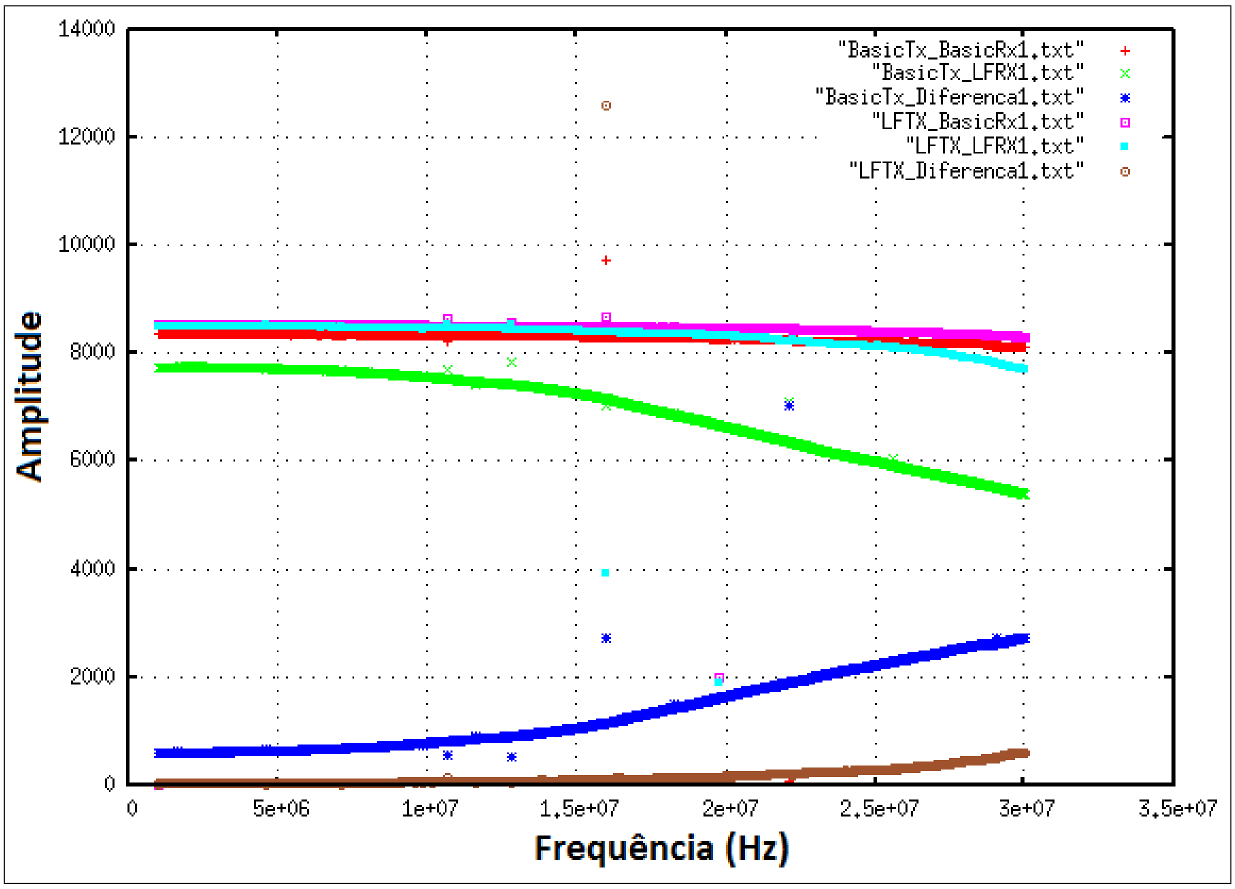


Figure 4.13: Amplitude frequency response with 25 ms time interval between samples of the four transceiver configurations

The last test performed in the end of the previous Section with regard to the full frequency range covered by the transceiver configuration BasicTx-BasicRx from 1 up to 250 MHz was performed again using the automatic prototype. The frequency resolution used was 250 kHz with a time interval equal to 25 ms and a signal amplitude equal do 32000 levels of DAC. As expected the frequency response amplitude resulted in the same two shadowed frequency ranges as illustrated by Figure 4.14: the first one from 44.824 MHz up to 84.166 MHz and the second one from 172.810 MHz up to 212.152 MHz.

Comparing this result with the one in Figure 4.9, it's clear that even with the double measurement points, 996 approximately, the automatic method resulted in less random points outside the major group. This justifies once more that the frequency resolution

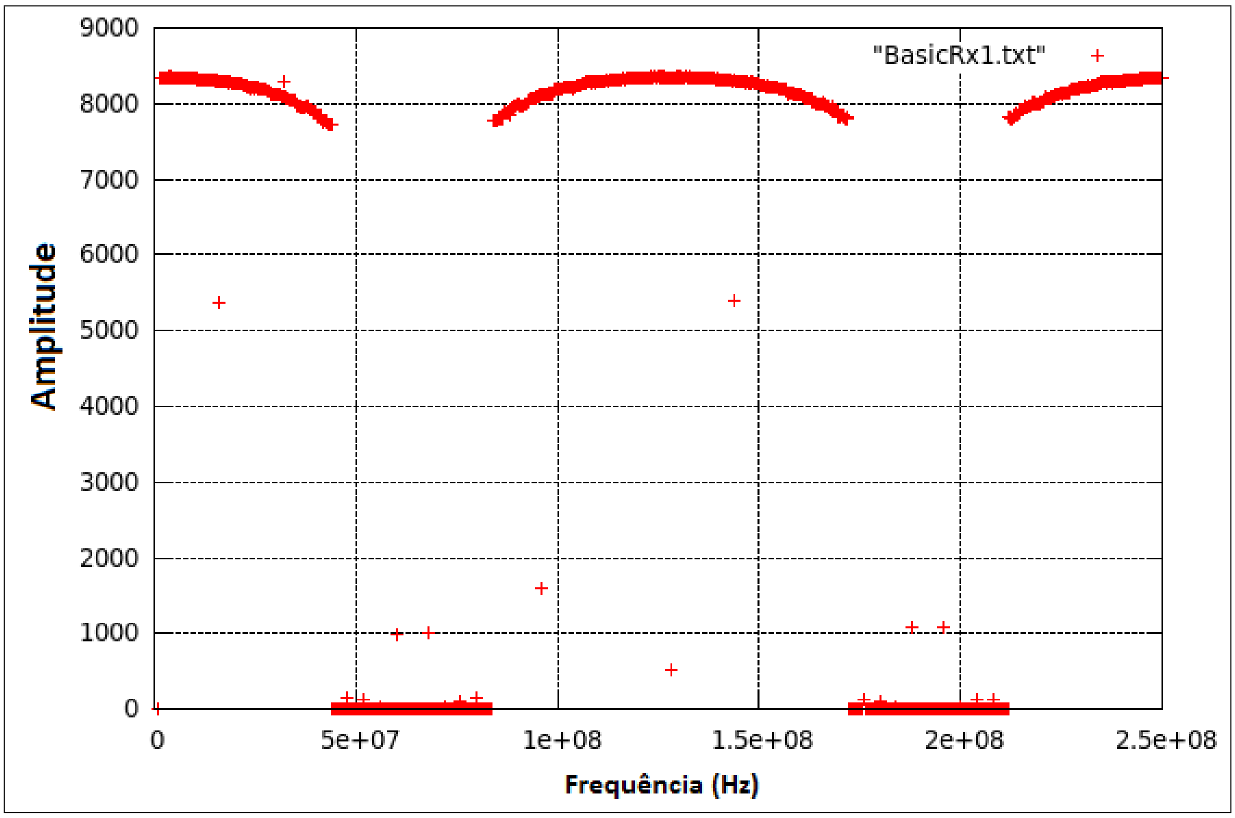


Figure 4.14: Frequency response amplitude of the transceiver system BasicTx-BasicRx from 1 up to 250 MHz using the automatic method

doesn't affect the results and the automatic method together with the ideal time interval can improve the prototype frequency response amplitude graphical results.

4.5 The prototype in action: frequency response phase acquisition using the automatic method

One of the major characteristics of the USRP module is it's capability of handling complex signals. When developing a frequency response acquisition prototype for both acquisition of amplitude and phase the use of complex signals becomes very suitable and facilitates the calculus for determining the phase shift between the transmitted and received signal. Because all the transceiver sides, transmitter and receiver, are in the same equipment and the processed signals are digitized, the shift can be assessed by taking the current phase of the received signal by the arctan using the I and Q signals, calculate the current phase of the transmitted signal the same way and to subtract one from the other.

But the results seemed to be incoherent. Figure 4.15 shows the plotted measured points in a graph for a transceiver configuration BasicTx-LFRX using the basic automatic prototype configuration: baseband cosine signal with frequency equal to 1 kHz and amplitude equal to 32000 DAC levels. It's notable that the phase varies randomly between the frequency range from 1 MHz up to 30 MHz.

For this reason, the waveform was analyzed using the GRC embedded oscilloscope. The received waveform was observed and it was verified the amplitude attenuation throughout the frequency range tested as shown in all the previous figures. However, when the transmitted signal was also on the screen the phase difference between the two signal varied randomly during the carrier frequency sweep, which justifies the results shown in the graph.

While seeking for the root cause of this problem, the same communication system prototype was used, but with a little modifications aiming to find out if the calculation method was properly integrated to the correct probes, etc. The transmitter and receiver USRP interface blocks, source and sink were replaced by a simple cosine signal generator keeping the probes where they were connected to. This very simple test should calculate the phase difference between a constant phase signal transmission and reception. So the signal generator was connected directly to the receive path. As expected the result of the phase difference was null. When it's compared with the results using the USRP, it demonstrates that the calculation method and probing are working well and that the problem may rely on the external USRP hardware module.

To solve this issue further research is needed and if the use of a newer USRP version is possible maybe the problem related to this was already solved.

4.6 The prototype in action: the ultimate approach

It is worth mentioning that if someone desires to acquire only the frequency response amplitude the two receiver sides can be used at the same time with one transmitter side. This hardware configuration can be implemented transmitting the I signal to one receiver side and the Q signal to the other one, leaving one transmitter side empty to implement any independent transmitter system. At the GRC the system building is very simple and uses only one transmitter path and one receiver path. The transmitter is the same used through this entire chapter, and only the receiver needed some additions as

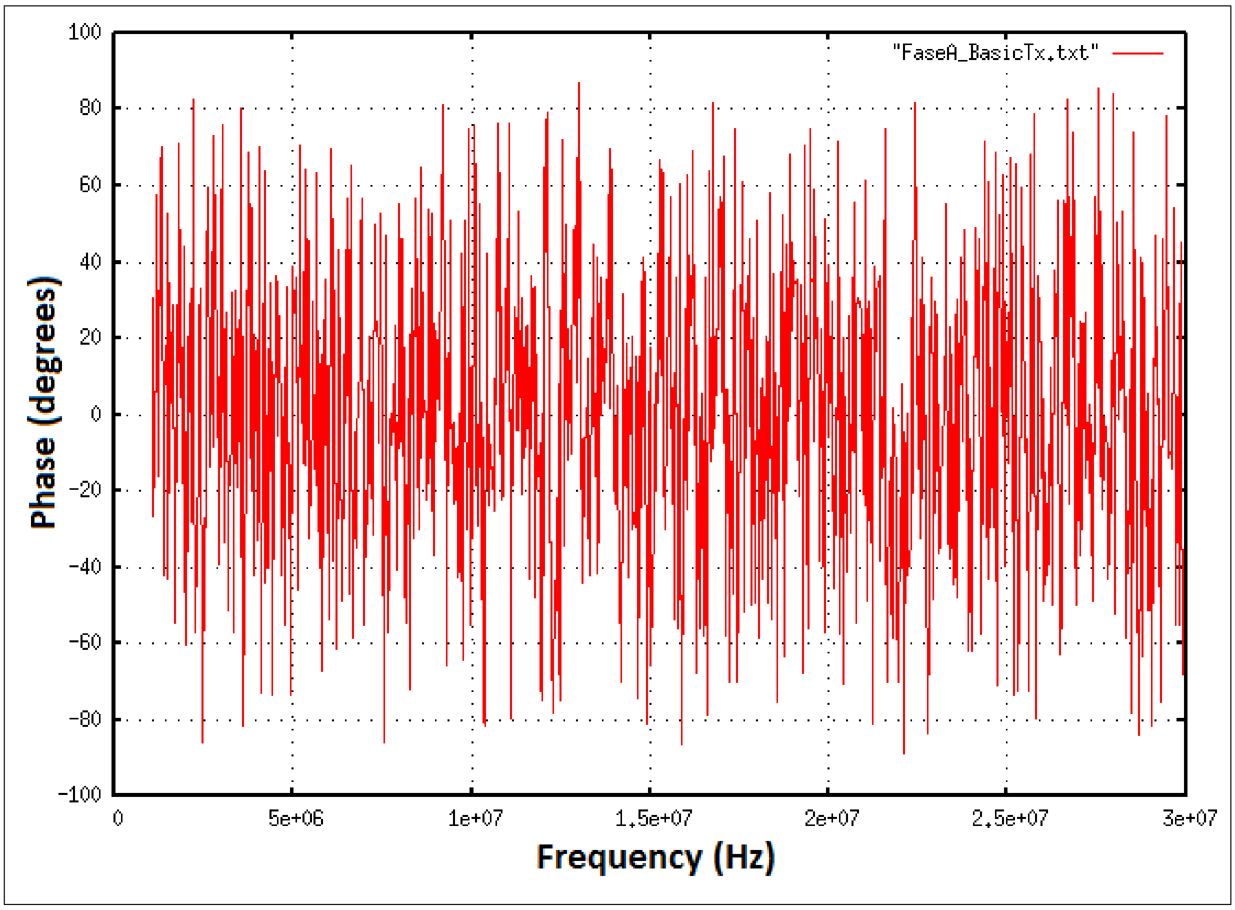


Figure 4.15: Frequency response phase of the transceiver system BasicTx-LFRX

shown in Figure 4.16. The USRP source block can be replaced by the USRP Dual source block, which allows the configuration of two receiver chains with the same sample rate and decimation factor but in different sides and antenna connectors. But the constraint in using such approach is that the three joint chains (one transmitter and two receivers) must be lower or equal to the total USB interface data rate allowed.

The connections are as follows: the transmitter I antenna connector (TXA) is connected to the receiver I antenna connector (RXA) and the transmitter Q antenna connector is connected to the receiver Q antenna connector (RXB).In this way the user can improve the total test time if only the amplitude is of interest.

Moreover, if the user wants to perform two tests at the same time using complex signals, which will be the ideal case when the frequency response phase issue will be solved, the four communication chains must be used: two transmitters and two receivers, since the joint data rate respect the total USB interface maximum rate. In GRC this would result in two independent transmitter paths and two independent receiver paths. But if the two transmitters uses the same final interpolation factor and sample rate only

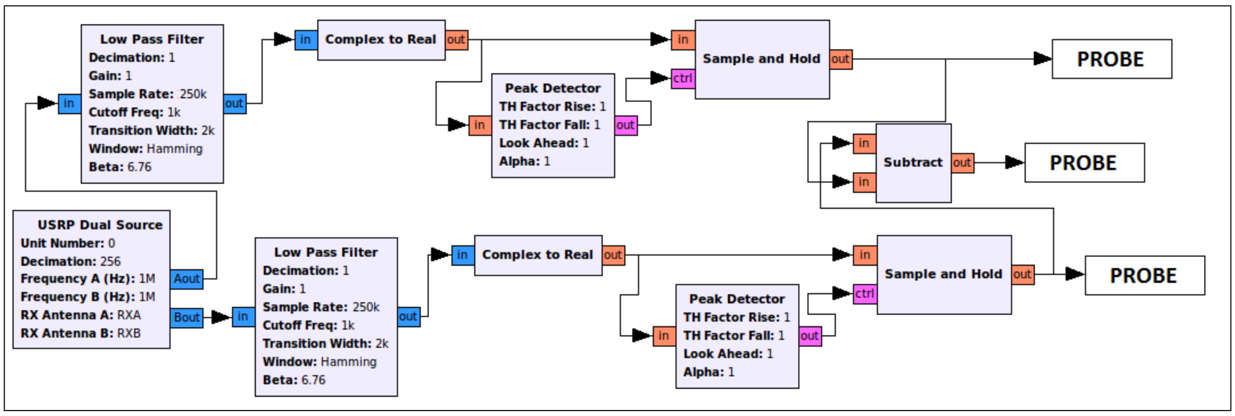


Figure 4.16: Ultimate Frequency response receiver prototype

one USRP sink block must be used, the USRP Dual Sink. The same is valid for the two receivers, using the USRP Dual Source.

This ultimate approach will assure the running of two complete tests at the same time, resulting in the frequency response amplitude and phase.

4.7 Final considerations of this chapter

This chapter showed the design of the frequency response amplitude acquisition prototype. It also showed the frequency response amplitude for the four possible different configurations between the BasicRx, BasicTx, LFRX and LFTX daughter boards. These results allow to identify the attenuation level of a certain configuration. In addition it was showed that the frequency response phase acquisition needs further research given the results achieved with the approach proposed.

The next chapter will describe the design of the PLC-USRP interface prototype. It will be given the ideal interface requirements and the final practical design. In the end the interface prototype is tested using the frequency response amplitude acquisition prototype.

Chapter 5

The PLC-USRP interface prototype development

This chapter describes the PLC and USRP electrical specifications and constraints based on IEEE 1901-2010 and HomePlug AV specifications. The text also cover the development of the PLC-USRP interface design in details, from computer simulations up to building a practical prototype ready for deployment.

5.1 The need of a PLC-USRP interface

The main objective of the PLC-USRP interface is to connect the USRP module to a PLC network. As already known, the PLC signals are conveyed within the electrical network, where there are the power signal of 220/110 V at 60/50 Hz. If someone tries to connect the USRP module directly to the electrical wiring, both power and PLC signals may get into the module and make a huge damage to the whole equipment, which is quite sensitive. The input daughter-boards used, delivery the incoming signals directly to the USRP ADC's which expect a limited maximum level signal. If the input signal overcomes a certain limit by a very high level it may cause serious damages to the USRP, including the PC connected to it via USB interface.

For this reason the interface properly avoids the entrance of the power signal into the module. This can be achieved through filtering the unwanted signals: power signals and noise signals considered out of PLC desired band from 2 MHz up to 28 MHz. With this interface a PLC signal analyzer can be connected to the power line network to make the acquisition of the signals for real-time processing. The information's security issue related to the signal analysis doesn't represent an eavesdropping problem since the HomePlug specification establishes encrypted data transfers.

In addition to the filtering activities, some electrical protection levels may be employed in order to avoid electrical hazards made by long term transients, electrostatic discharges, electronic components failures, etc. Also network coupling and galvanic isolation must be provided. All these protections are more detailed on Subsection 5.2.

At the PLC-USRP interface output, there will be only the PLC signal and a very low noise level. This signal gets into the USRP module, which makes the acquisition down converting it to baseband. Inside the GNU Radio oscilloscope, spectrum analyzer or any other processing block chain, the PLC signal will be ready to be further processed.

5.1.1 The PLC and USRP requirements

In 2 it was said that there are two different methods, or 'couplings', for accessing the transmission medium: inductive coupling and capacitive coupling. An easy way to connect the interface to the electrical network outlets is to use the capacitive coupling. In this way the major high-pass filter is a pair of coupling capacitors intended to get rid of the electrical network power signals at 50/60 Hz.

Regarding the electrical connection between the two different technologies, PLC and USRP, the PLC signal level must be compliant with the USRP input signal level limit. First, the maximum tension level injected by a PLC modem in the electrical network must be estimated. In accordance with Figure 2.8 in Chapter 2 the total power contributed by the spectral components within the whole PLC spectrum ranging from 2 MHz up to 28 MHz can be calculated from the transmitted power spectral density using Equation (5.1) [22].

$$\Delta P_g = \frac{1}{\pi}\int_{\omega_1}^{\omega_2} S_g(\omega)\mathrm{d}\omega \tag{5.1}$$

where,

$S_g(\omega)$ power (dBm) per unit bandwidth (hertz); and

ω_1 and ω_2 lower and upper frequency limits respectively (radians per second).

The PSD figure on Chapter 2 shows some notches which represent some frequency bands not used by PLC communication system, where the sub-carriers were turned off. This must be take into account while determining the interval of the integral in Equation (5.1). Considering this, it's possible to estimate the total bandwidth used by the bandwidth of each sub-carrier and their total number. There are 917 sub-carriers with a bandwidth of approximately 24.414 kHz each, resulting in a total of 22.380 MHz of bandwidth used by a PLC HomePlug AV compliant with IEEE 1901-2010 standard. This leads to $\omega_2 = 2 * \pi * 22.380$, and $\omega_1 = 0$.

The next step is to convert the upper and lower power density from dBm/Hz to W/Hz. Using the conversion Equation (5.2), -55 dBm/Hz = 3.16 nW/Hz and -85 dBm/Hz = 3.16 pW/Hz

$$Power(dBm) = 10log\frac{Power(mW)}{1(mW)} \tag{5.2}$$

The total power is the area under the PSD curve. Since there are two PSD limits, lower and upper, they must me subtracted inside the integral, as follows:

$$\begin{aligned} \Delta P_g &= \frac{1}{\pi}\int_0^{2*\pi*22.380\text{M}} (3.16n - 3.16p)\mathrm{d}\omega \\ &= 0.1413\text{W} \end{aligned} \tag{5.3}$$

Equation 5.3 shows that the maximum power transmitted by a PLC HomePlug AV modem using 917 sub-carriers is approximately 141.3 mW. This is a very low power compared to the power to be transmitted by a PLC signal injector very close to a transformer station in an access network, which is on the order of 25 W up tp 250 W. This is due to the impedance of the network where the PLC transmitter is located. Near the transformer the impedance is very low, 0.2 Ω up to 2 Ω so that more transmission power is required. In other words, the high access impedance seen by a PLC transmitter device from an outlet in a typical house building results in the need of a low power transmission, which is a desirable situation for a low-cost indoor PLC modem [11]

The impedance of the electrical network randomly varies with the characteristic impedance of the cables, the topology of the considered part of network, the random connection and disconnection of appliances to it resulting in a variation within the time, the temperature and also with the frequency measured.

Another important factor to be considered is the local where the impedance is being measured, because if it's close to the transformer station, the impedance will be very small compared to the value measured inside a building several meters far away due to the many parallels connections of a large number of outgoing trunks. In order to get an approximate value, statistical analysis of some measurements has shown that throughout the whole spectrum of interest (from 1 kHz up to 50 MHz) the mean value of the impedance relies between 100 Ω and 150 Ω for a typical home building.

Due to the impedance variation, cable losses and mismatched coupling of devices into and out of the electrical network, the resulting attenuations on the transmitted signals are common in PLC communication's networks [15], [11].

The IEEE 1901 standard [16] consider that the load seen by the transmitter installed inside a common home building at a low-voltage electrical network should be considered as 50 Ω. Consequently, using Equations (5.3), the maximum power to be transmitted is equal to 0.1413 W. The IEEE 1901 standard does not explicitly provide the maximum voltage level to be transmitted, so it can be roughly estimated by Equation (5.4)

$$P = \frac{V^2}{R} \tag{5.4}$$

Considering the network load as 50 Ω and using Equations (5.1) and (5.4), the maximum theoretical voltage level during the transmission of an OFDM symbol using all the 917 sub-carriers would be 2.65 volts or 5.3 volts peak to peak. Nowadays, inside a typical house building there are several appliances and many electronic devices attached in parallel to the electrical network which might contribute to decrease the resultant network impedance to a lower value than the initial 100 Ω or 150 Ω making the impedance value to get close to the 50 Ω load considered by the IEEE 1901 standard. This lower impedance jointly with the losses inserted by the cables may decrease the transmitted voltage level measured at a certain receiver or a measurement point along the power line at a certain distance.

In order to know the transmitted voltage level, a practical measurement experiment was mounted. It consisted basically of transmitting a PLC signal between a PLC transmitter and a PLC receiver through the laboratory's electrical network. The measurements were made by connecting the oscilloscope to the electrical network using some filters designed for the PLC-USRP interface (described on the following sections). The lab electrical network was composed at the moment by four basic computers with only one turned on, eight 1.2 m fluorescent lamps turned on and an air conditioner of 9,000 BTU's also turned on. In addition to all these devices there were more than ten outlets without any device plugged. So in comparison with a typical house, the house may have much more appliances and devices attached and turned on at the same time than the lab, which will result in a lower impedance seen by the PLC transmitter and a lower voltage signal. The PLC transmitter and receiver used were two PLC modems model AV200 by TP-Link compliant with HomePlug AV standard (data transfer rate up to 200 Mbps).

One of the modems acted as the PLC injector, feeding the OFDM modulated Ethernet signal inside the electrical network. The other modem connected the computer to the electrical network. By the computer browser it was requested the streaming of a HD video (high-definition video with the quality set to 720p). The video chosen lasted enough for making the measurements. This kind of streaming requires one of the best performances from the modems but stays limited by the Internet link current conditions, which during the measurements its data rate was close to 3 Mbps resulting in some delays on the video player but assuring a continuous data traffic between the modems. The measurement point was at a distance of 0.5 m far from the transmitter modem and a few centimetres from the receiver modem. The signals at the scope seem to have the same peak about $\pm$ 0.8 V or 1.6 V peak to peak. But as expected, when the measure was made at the outlet closest to the transmitter modem (up to a few centimetres) the readings reached higher values as $\pm$ 2.5 V or 5 V peak to peak. This shows how far the measures must be made to keep a safe operation condition for the USRP in accordance to its input voltage limitation. Also, from this measurements it can be inferred that as the distance increases so does the attenuation, decreasing the transmitted signal's voltage level.

The USRP module has a maximum input voltage of 2 V peak to peak when using BasicRx or LFRX daughter-boards. This value is mainly limited by the AD9862 chip by Analog Devices, which is a Mixed-Signal Front-End Processor for Broadband Communications integrating both DAC and ADC functions of the USRP architecture shown in Figure 3.2 [4], [13]. Considering the maximum transmitted voltage levels of a PLC transmitter (the theoretically calculated one and measured one in the previous experiment), the attenuations that a PLC signal may suffer at the electrical network cables and the USRP input voltage limit, there is no need to adequate the PLC amplitude signal to be received through the USRP daughter-boards BasicRx and LFRX by a resistance divisor or any other kind of attenuator, since the measurement distance from the transmitter is not less than 0.5 meters. In other words, the PLC-USRP interface can attach the USRP module directly to the electrical network at a distance greater than 0.5 meters, far from any PLC modem because every HomePlug AV PLC modem has a transmitter side. Therefore the maximum amplitude of the signal received when getting into the USRP module may be on the order of ±0.8 V peak (1.6 V peak to peak), or even less depending on the attenuation levels during the transmission and the distance between the transmitter and the measuring point.

If someone wants to handle this PLC-USRP interface at shorter distances than 0.5 meters an logically controlled attenuator (automatic attenuation control) should be implemented at the output of the interface with the objective of guarantee that the maximum signal amplitude getting into the USRP will be no greater than 2 V peak to peak. Throughout this text all the following measurements were made at distances greater than 0.5 m so there was no need to design and implement such kind of attenuator.

5.2 The ideal PLC-USRP interface design

Seeking to provide the PLC signals to the USRP module an interface had to be designed to filter out of band signals like noise and high voltage signal at 50/60 Hz. Some electrical protections also had to be added to the project to avoid any hazardous electrical discharge. In this section an ideal interface is proposed. In the next section a practical design and implementation approach is provided due to difficulties in finding some electronic components and constraints related to implementation costs. In the end of each section there are some computer simulations showing the expected behaviour of the circuit.

The ideal interface must implement three basic functions: 'DC blocking' for filtering out 50/60 Hz power signals, electrical protection against high voltage transients and electrostatic discharges and pass-band filtering for out of band (2-28 MHz) noise filtering. The proposal of the pass-band filtering stage of the interface was based on the analysis of the interface integrated circuit by Qualcomm ATHEROS model INT1400 [26] that implements the Analog Front End of the PLC modems used thorough this text.

5.2.1 DC blocking

A receiver attached to the electrical network must have a loose coupling according to [11]. This aims to avoid interference signals from getting into the device attached to the network. To achieve this, a high reactive impedance is required, so the device may handle a low current signal. The IEC 60384-14 [18] Part 14 defines standards of capacitors for electromagnetic interference suppression and connection to the supply mains separated into seven classes of capacitors to be used within line filters. In the case of PLC technology the aim is to filter out the 50/60 Hz signal. The capacitors used to develop this task are known as safety capacitors, suppression capacitors or simply line filter capacitors. This kind of capacitors need to operate under very harsh conditions, handling 110, 220, 230 volts or even more on a steady and ongoing basis.

There are two basic types of line filter capacitors. The first one is the across the line capacitor which is the capacitor that is connected from line to line or line to neutral carrying the incoming AC current. The class X capacitors are across the line capacitors and in case of failure the potential for electrical shock is not present, and would only cause a fuse or circuit breaker to open. The second type is the line to ground capacitor which is the capacitor that is connected from line to ground. The class Y capacitors are line to ground capacitors and in case of failure the potential for electrical shock is present because if the capacitor is bypassed, the chassis of the device involved could have the same line potential which would represent an electrical shock hazard. Each of these classes are subdivided according to the operational peak voltage that can be handled.

The two most popular types are the X2 and the Y2 class capacitors. The X2 is rated to work under an impulse peak of 2500 volts while Y2 works under an impulse peak of 5000 volts. As the Y2 capacitors are more robust, they can deal with higher voltages and are more expensive and larger than the X2. Y2 capacitors are designed to open in case of failure, so one could use a Y2 capacitor to replace a X2 capacitor but the inverse can not be applied. Both X2 and Y2 are indicated to home building usage within appliances connected to low voltage power lines.

Although Qualcomm ATHEROS INT1400 chips on the PLC modems used are followed by X2 capacitors, for safety reasons the right choice for the PLC-USRP interface is a Y2 class capacitor. As previously seen, Y2 is more expensive than a X2. However, a X2 capacitor followed a fuse could increase the safety level but not at Y2 class safety level, which has the feature of opening in case of failure. If the interface is intended to be used inside a low-voltage building the blocking capacitors should rate higher than 220 volts in order to cover both 110 and 220 volts power lines.

The first function to be accomplished by the PLC-USRP interface, is to get rid of power signal at 50/60 Hz allowing all the PLC signals from 2 MHz and above to get into the system. This leads to a cutoff frequency choice limitation that must be between 60 Hz and 2 MHz. Letting the lowest cutoff frequency be 100 Hz, which would assure that the signal at 60 Hz is less than 70.7% of its maximum. Also, taking the higher cutoff frequency as 1 MHz would assure that from 2 MHz and beyond the signals are almost at their maximum value. This two cutoff frequencies establish the maximum and minimum capacitor's capacitance which must be a X2 capacitor acting as a low-pass filter dealing with 60 Hz power signals at 110/220 volts.

To determine the capacitances, the load considered is the USRP input impedance and Equation (5.5) are used. Using the maximum frequency, 1 MHz, results in the minimum capacitance value of $C_{min} = 3.18$ nF, and using the minimum frequency, 100 Hz, results in the maximum capacitance value of $C_{max} = 31.83\ \mu$ F.

$$C = \frac{1}{2\pi R f_{cutoff}} \tag{5.5}$$

where,

C capacitor's capacitance;

R load; and

f_{cutoff} cutoff frequency.

To implement a loose coupling a low capacitance capacitor may be used from a range from 1 nF up to 10 nF [11]. This closely matches with the calculated capacitance range above. Some capacitance values were simulated in a very simple RC circuit using OrCAD Capture by Cadence version 16.3. It was used an AC source with fixed amplitude of 110 volts (the same voltage on the lab power lines) and a sweep from 10 Hz up to 10 MHz was performed to show the low-pass effect and the amplitude at 60 Hz for each capacitance value. In Figure 5.1 there is the simple simulated circuit with the 3 nF capacitor, the lowest assessed value. Figures 5.2 and 5.3 show the frequency response amplitude and the latter one is zoomed showing that the 110 volts at 60 Hz was highly attenuated by this circuit, with an amplitude of only 6 mV. Figures 5.4 and 5.5 show the results when the 33 μF was used. As the cutoff frequency for this maximum capacitance value is very low, only with the zoom is possible to see the low-pass filtering where the amplitude at 60 Hz is 16 volts. Going further on simulating capacitance values aiming a low amplitude at 60 Hz, the maximum value of 10 nF cited by [11] (whose cutoff frequency is 318.3 kHz) which is also among the assessed range was used and the results are shown in Figures 5.6 and 5.7, where the 60 Hz amplitude is 20 mV

Therefore, these simulations have shown that capacitance values between 3 nF and up to 10 nF give good attenuation on 60 Hz signal, resulting in only a few tens of millivolts. In order to assure a high coupling symmetry degree (one important parameter for Electromagnetic Compatibility [EMC] aimed to allow that a pure differential mode signal propagates over a distance as long as possible [21]), the dc blocking capacitors must be symmetric, so there must be one blocking capacitor in each line, regardless if the mains source supplies 220 V or 110 V [11].

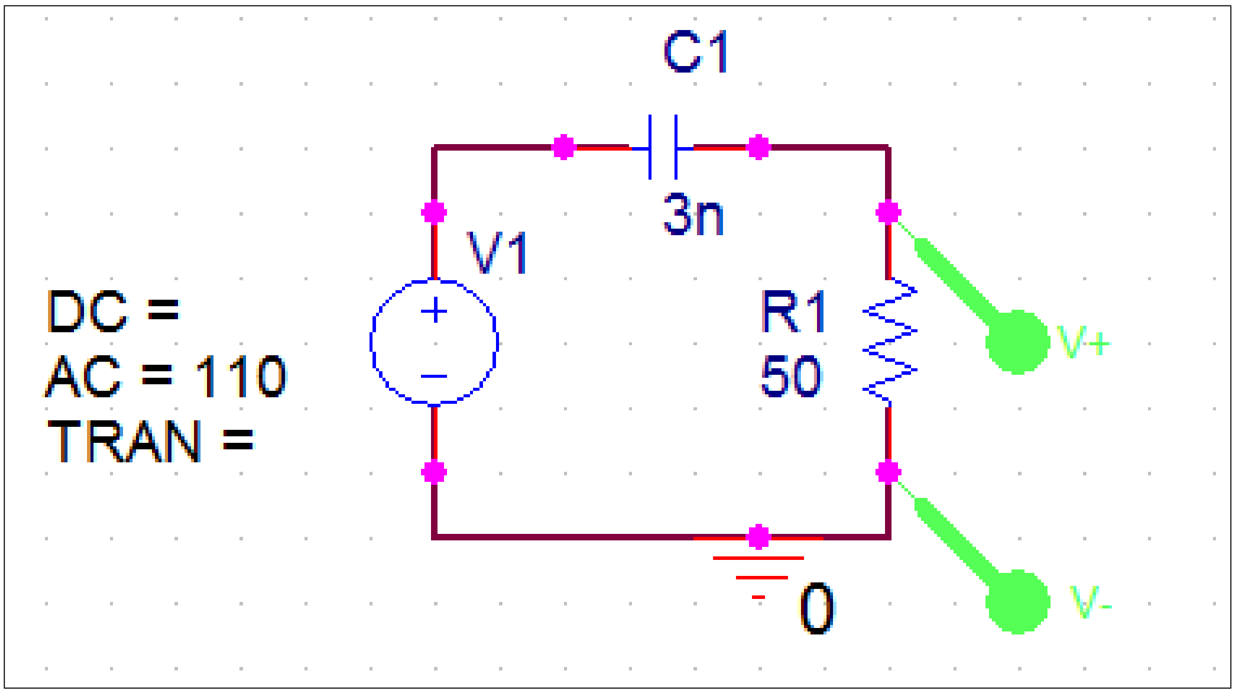


Figure 5.1: Simple low-pass filter

5.2.2 Electrical Protection

As far as electrical protection against high voltage transients and electrostatic discharges are concerned, some additional devices must be installed on the PLC-USRP in-

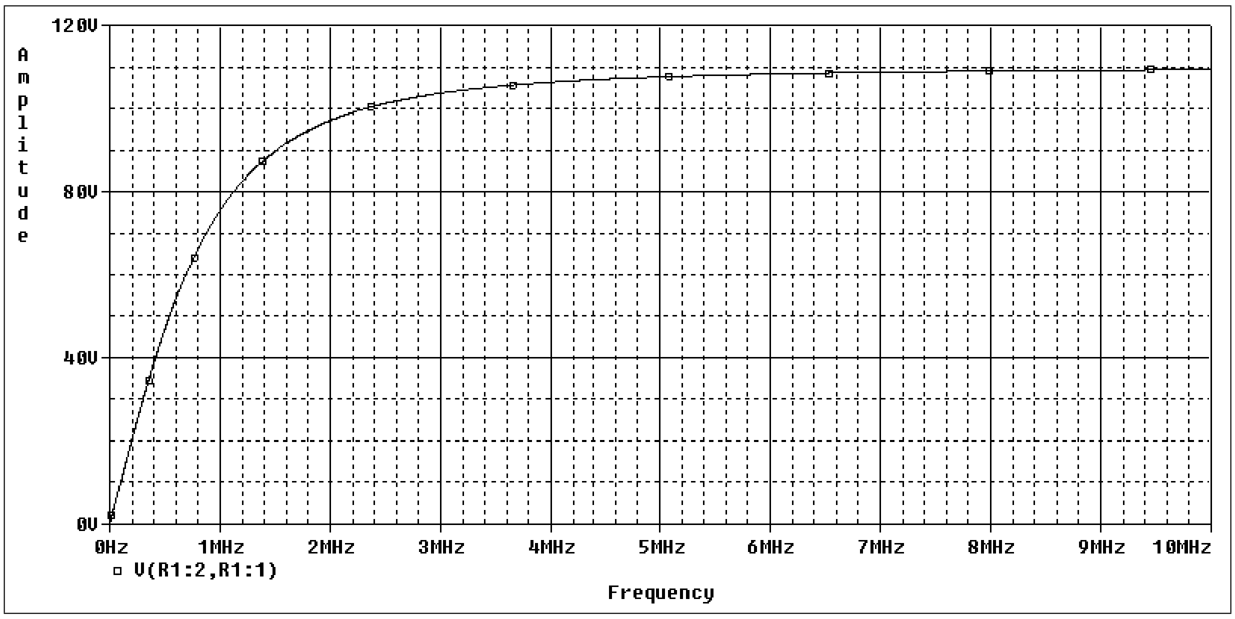


Figure 5.2: Low-pass filter response with C=3 nF

terface. Beyond having to deal with the normal power line voltage, the interface must deal with random power surges and spikes originated from several sources like lighting storms, manoeuvrings at power plants, switching of inductive loads, including failures on the power lines generation, transmission and distribution systems that could unpredictably generate signals with amplitudes of several kilovolts during very short times, able to cause some damage to some electronics attached to the network, including the USRP when using the PLC-USRP interface.

Given the very small sizes of electronic components, the sensitivity to electrical stresses has increased in such a way that some conductive paths inside microchips are unable to handle high voltage transients caused high current discharges due to the low operating voltages, that are susceptible to very low voltage disturbances with the potential of causing overall system interruption and serious damages to the hardware

As seen in the previous subsection, if one decides to use the X2 capacitor (due to cost and size reasons), which in case of failure could open or short. Although its openness won't make any damage to the following circuitry because of the connection between the interface and the power line will be simply lost, if it shorts there shall be some serious damages because the filtering effect will not exist, so the 60 Hz power signal may get into the USRP and burn some IC's. To overcome this potential risk, a fuse is placed in series just before the capacitor within one of the lines to open in case a high current flow tries to enter the system when the capacitor shorts. A standard fuse rated for 250 V and 1 A or

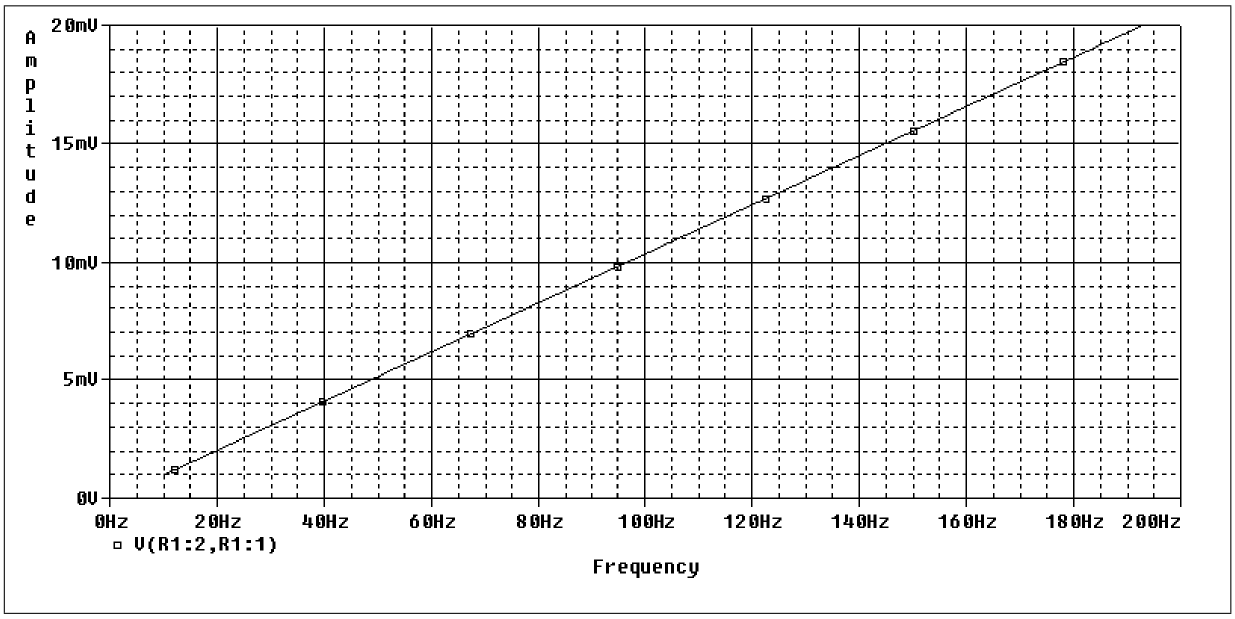


Figure 5.3: Low-pass filter response with C=3 nF zoomed

2.5 A should be enough, considering the low current existing during a normal operational state. Thus, a fuse rated to 1 A or 2.5 A would be enough to detect a failure due to a short circuit and to open the circuit limiting the incoming high current.

To protect against high voltage transients coming from the power lines a Voltage Dependent Resistor (VDR) need to be placed in parallel between the fuse and the X2 capacitors. This device is a non-linear resistor whose resistance changes as a function of the applied voltage, so it can be used for absorption of high voltage long-term transients, limiting high voltage surges like the ones resulted from lightning. The voltage limit level is the VDR's parameter known as Clamping Voltage. On the market the most popular VDR is the zinc-oxide non-linear resistor (ZNR) also called Metal Oxide Varistor (MOV). The VDR is specified mainly by the Maximum Allowable Voltage (MAV) parameter, which is the maximum sinusoidal or DC voltage that can be applied continuously. The usage of varistors in electronic equipment is regulated by the IEC 61051-1 [19].

To operate across the 120 V or 220 V AC lines, the VDR should have a MAV higher than 220 V and conduct a current as close as possible to that of the specified fuse for a specified time during a surge occurrence. The Panasonic's VDR reference guide [24] for AC voltages up to 1200 V in line to ground applications and 480 V in line to line application, lists an example of a possible model to be chosen for the interface. The ERZV05D431 varistor has a MAV of 275 V, Clamping Voltage of 745 V and conducts a current of 5 A during the application of the standard impulse current of 8/20 μ seconds

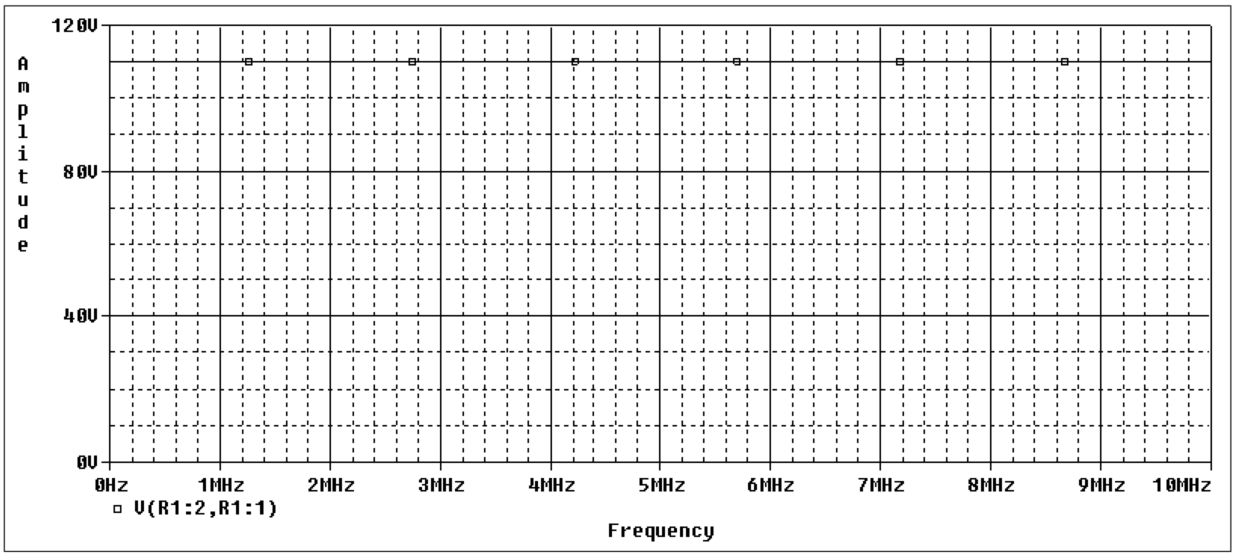


Figure 5.4: Low-pass filter response with C=33 uF

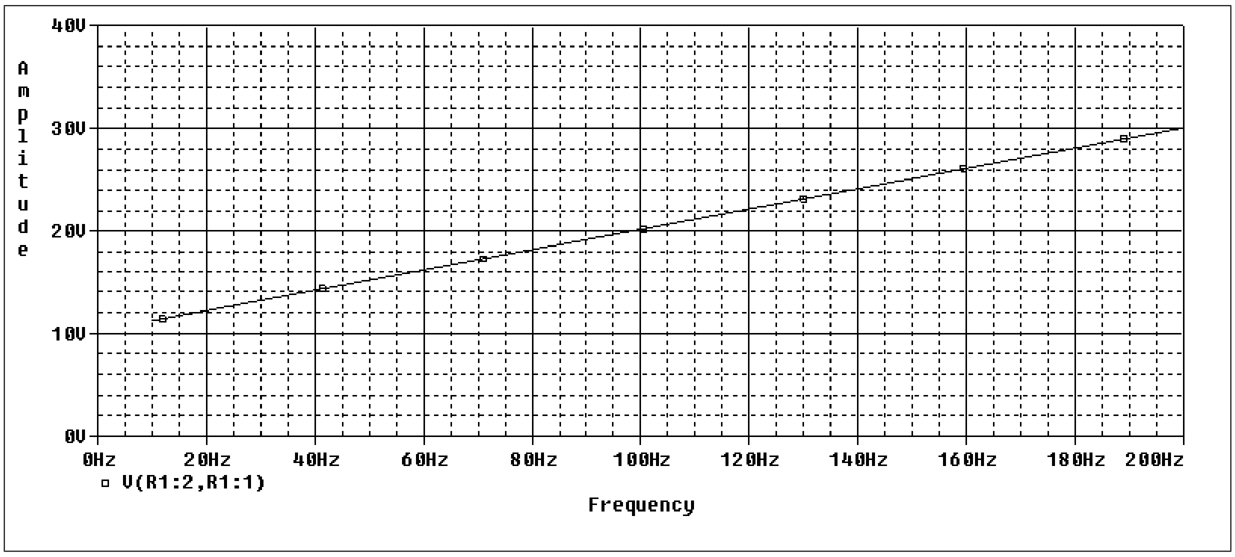


Figure 5.5: Low-pass filter response with C=33 uF zoomed

illustrated by Figure 5.8. A varistor with an equal or even better configuration than that could be implemented on the PLC-USRP interface. One example of a more robust VDR is the one implemented on the PLC modems used on the tests described throughout this text, the VDR model ERZV10D471, which has a MAV of 300 V, clamping voltage of 775 V and conducts a current of 25 A during 8/20 μ seconds.

Just after the fuse, the dc blocking capacitors and the VDR, an electrical or galvanic isolation must be implemented for safety reasons, physically isolating the power line circuit from the USRP circuit. Without this isolation, any wrong or accidental connection or contact in the electronic side at any energized point would cause electric shock by current flowing through a person's body. For this purpose an optocoupler or an isola-

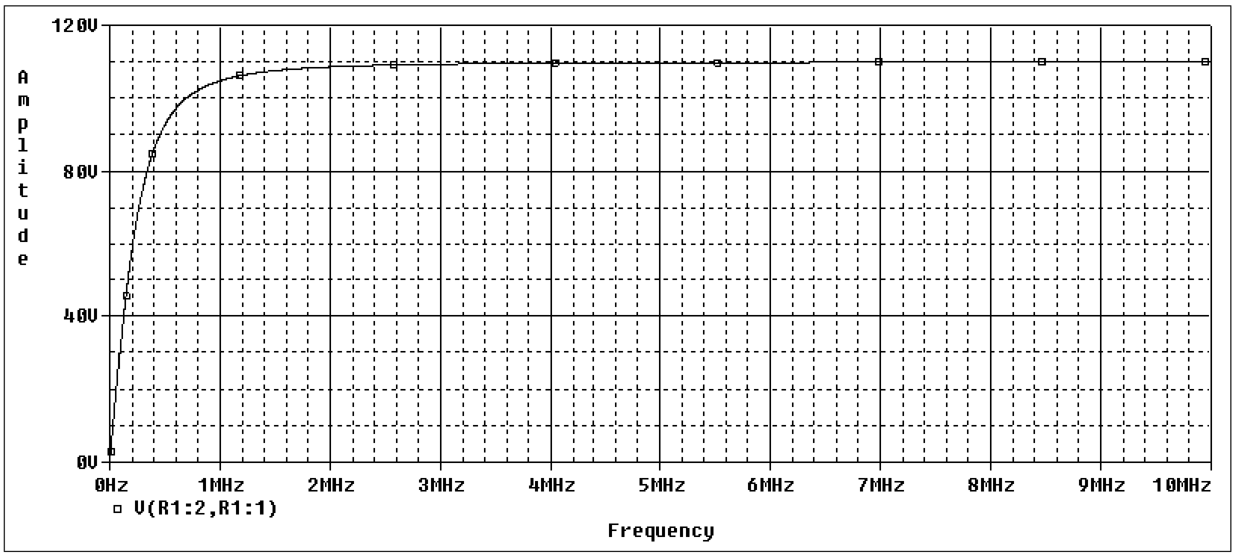


Figure 5.6: Low-pass filter response with C=10 nF

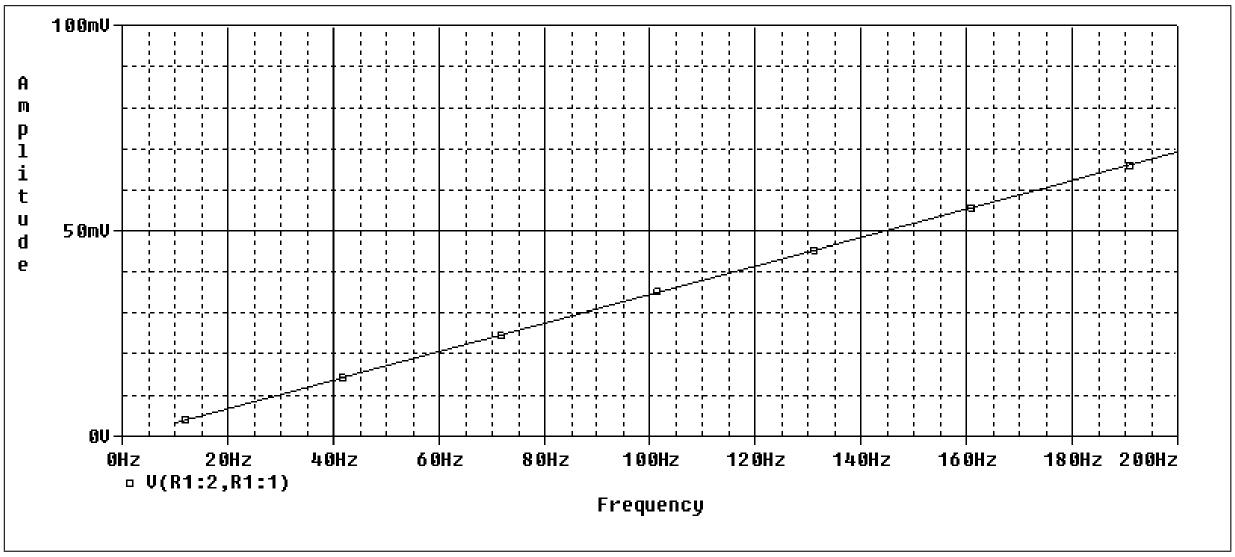


Figure 5.7: Low-pass filter response with C=10 nF zoomed

tion transformer with the turns ratio equal to 1:1 can be used. The constraint related to the usage of an optocoupler relies on the need of a DC source, while the transformer as a passive element does not. But the transformer needs to have a constant frequency response amplitude throughout the wide PLC bandwidth from 2 MHz up to 28 MHz. On the market this kind of transformer is known as power line signal coupler, with three windings, one on the primary for the mains side, and two on the secondary, one for the transmitter path and the other for the receiver path. The PLC-USRP interface only has a receiver side, consequently, a transformer with a single primary and single secondary with a 1:1 ratio rated to 10 V or more would be enough. The PLC modems used on the experiments use the model 0557-7700-41 by Bel Fuse Inc. [5].

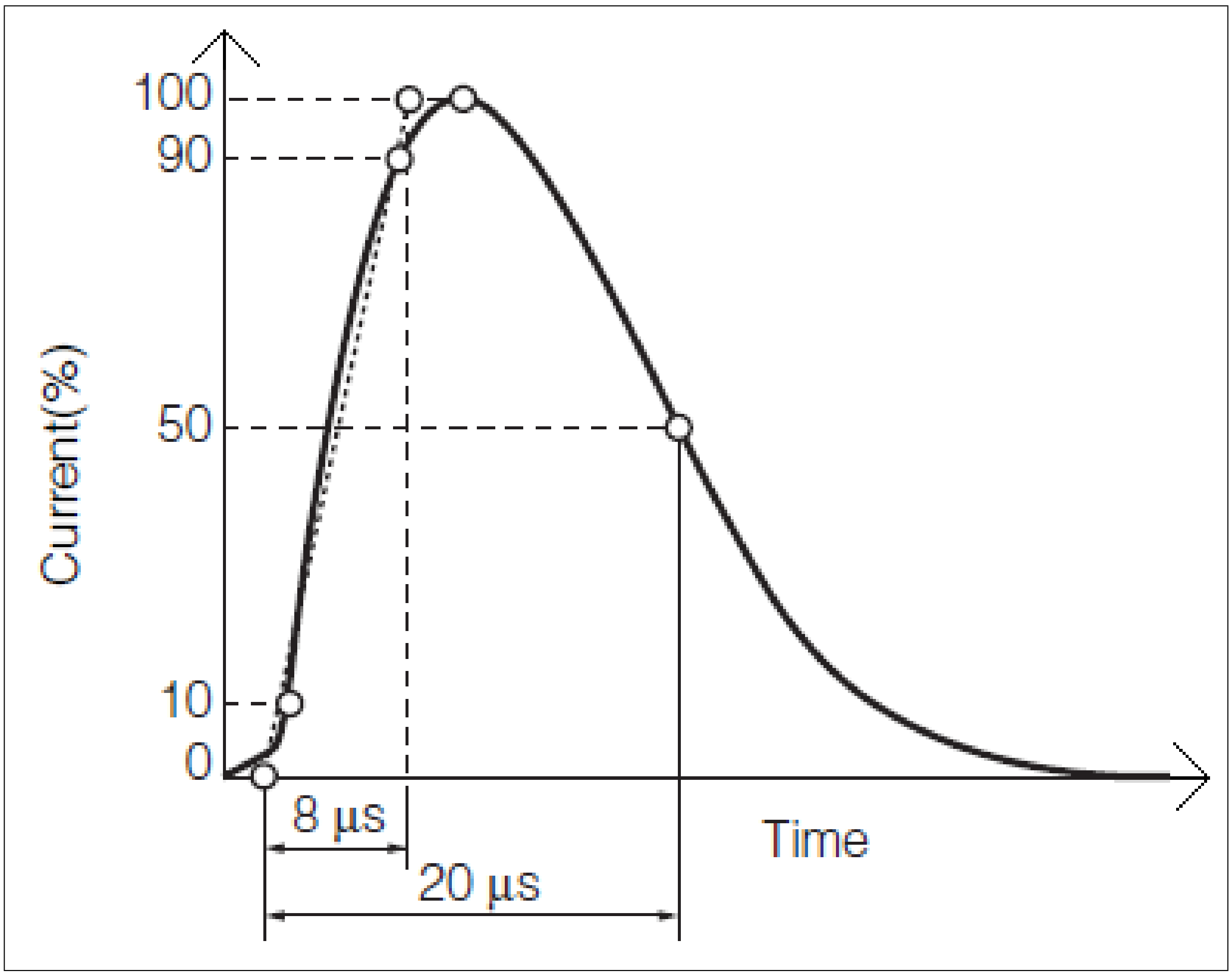


Figure 5.8: Standard impulse current waveform

One last protection device that might be present on the interface is the Transient Voltage Suppressor (TVS) diode array for Electrostatic Discharge (ESD) protection. This device offers superior electrical characteristics that complements the protection coverage timing of the VDR in such a way that it act more quickly and has a lower clamping voltage than the VDR. The TVS diodes absorb transients to steer the current and then, the avalanching or zener diode clamps the voltage level preventing a very high voltage from getting into the device for a short time. One feature of TVS that has becoming very important is its capacitance, due to the increase of data rate and bandwidth applications. The TVS capacitance must be as low as possible to avoid input signal distortion at very high frequencies. Thus the TVS's capacitance may be on the order of 0.4 pF up to 30 pF (ultra-low capacitance).

To choose a TVS diode array, the main specifications that must be verified are: if the stand-off voltage of the TVS is higher than the maximum circuit operational voltage, ensuring that the TVS does not clip the voltage during its normal operational level and if the TVS comply with the IEC 61000-4-2 standard [17]. Concerning this tension level, as the approximately assessed maximum transmitted voltage level assessed is 2.65 V, the interface TVS must have a stand-off voltage a little higher than 2.65 V. There are some models starting at 3.3 V, but choosing one with 5 V would be safer, like the one implemented on the PLC modems used on the experiment. One example of a TVS diode configuration with a stand-off voltage of 5 V is the GBLC05 [31] with a maximum clamping voltage of 9.8 V when the standard impulse current of 8/20 μ seconds with a peak of 1 A is applied.

5.2.3 Pass-band filter

The last section of the PLC-USRP interface is the pass-band filter designed to get rid of all the remaining noise below 2 MHz that may still exist after the dc blocking and the noise above 28 MHz. The pass-band filter is a combination of a high-pass filter in series with a low-pass filter. For robustness reasons the high-pass filter is implemented again, but with a higher cutoff frequency, which attenuates signals at frequencies higher than 100 Hz using low voltage rated capacitors. Therefore, a good approach for implementing the high-pass filter is to couple the interface to the mains, filtering out the power signal at 60 Hz, while protecting the interface using the fuse, VDR, transformer and TVS array and only after that to eliminate the remaining noise using low-voltage components. The same reasons are valid for not implementing the low-pass filter just after the dc blocking capacitors. On the PLC modems there is an equal approach for placing the pass-band filter only after the TVS array.

The main objective of the pass-band filter design is to be as sharp as possible over the two cutoff frequencies and to provide a flat response over the entire desired frequency range. These specifications require a filter with a high attenuation rate and a stable passband and stopband responses. Because the art of filter design relies mainly on approximations of the many solutions that satisfies the specifications for a certain filter, there are some books that simplified the process of designing a filter making some approximations and establishing some standard normalized tables that can be found in Analog and Digital Filter Design [33] by Steve Winder and Analog Electronic Filters [10] by Hercules G. Dimopoulos.

There are some well known approximations like Butterworth, Chebyschev, Inverse

Chebyschev, Pascal, Inverse Pascal, Elliptic and many others. Each one has a different frequency response gain, which is illustrated by Figure 5.9 [10], where Ω is the angular frequency, Ω_C is the cutoff frequency, Ω_S is the stopband frequency, H_0 and H_C are the maximum and minimum allowed gain in the passband respectively, and H_S is the maximum allowed gain in the stopband. The first step is to choose of that best fits the requirements regarding the shape of the response. Among these approximations it can be seen that the Butterworth has a flat passband and stopband responses, and it's attenuation rises almost immediately outside the passband. For these reasons it's a possible solution.

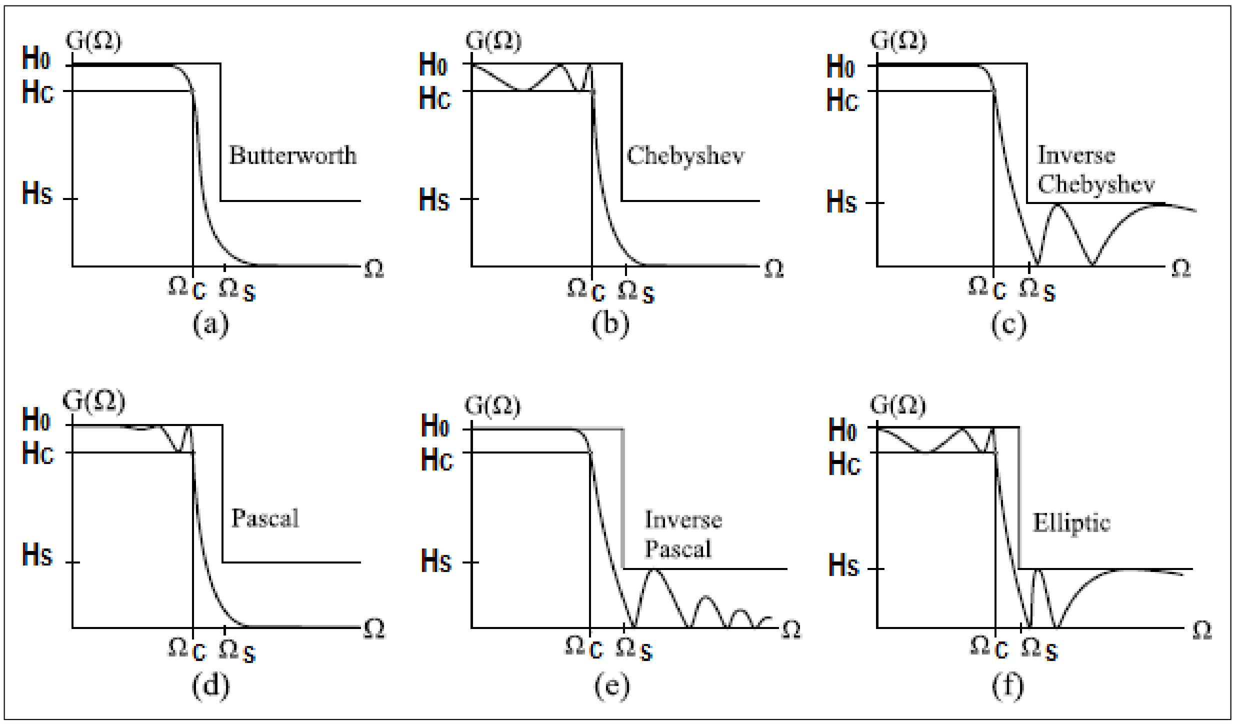


Figure 5.9: Gain plot of some filter approximations

Next, the order of the filter must be chosen based on the desired attenuation rate. The rate at which the Butterworth attenuation rises is n * 6 dB/octave. The pass-band filter frequency response amplitude attenuation needs to be as sharp as possible in such a way that immediately after the low cutoff frequency the signal does not suffer any attenuation and immediately after the high cutoff frequency the signals suffer a high attenuation. Analysing the plot of the attenuation rates at Figure 5.10 [33] for some different poles one can see that with five poles a high attenuation level of 30 dB just in the double or half of the cutoff frequency (depending if the filter is a low-pass or high pass) is achieved with ease of implementation. The following simulations show that this five pole filter uses the lowest pole number capable of giving a sharp response.

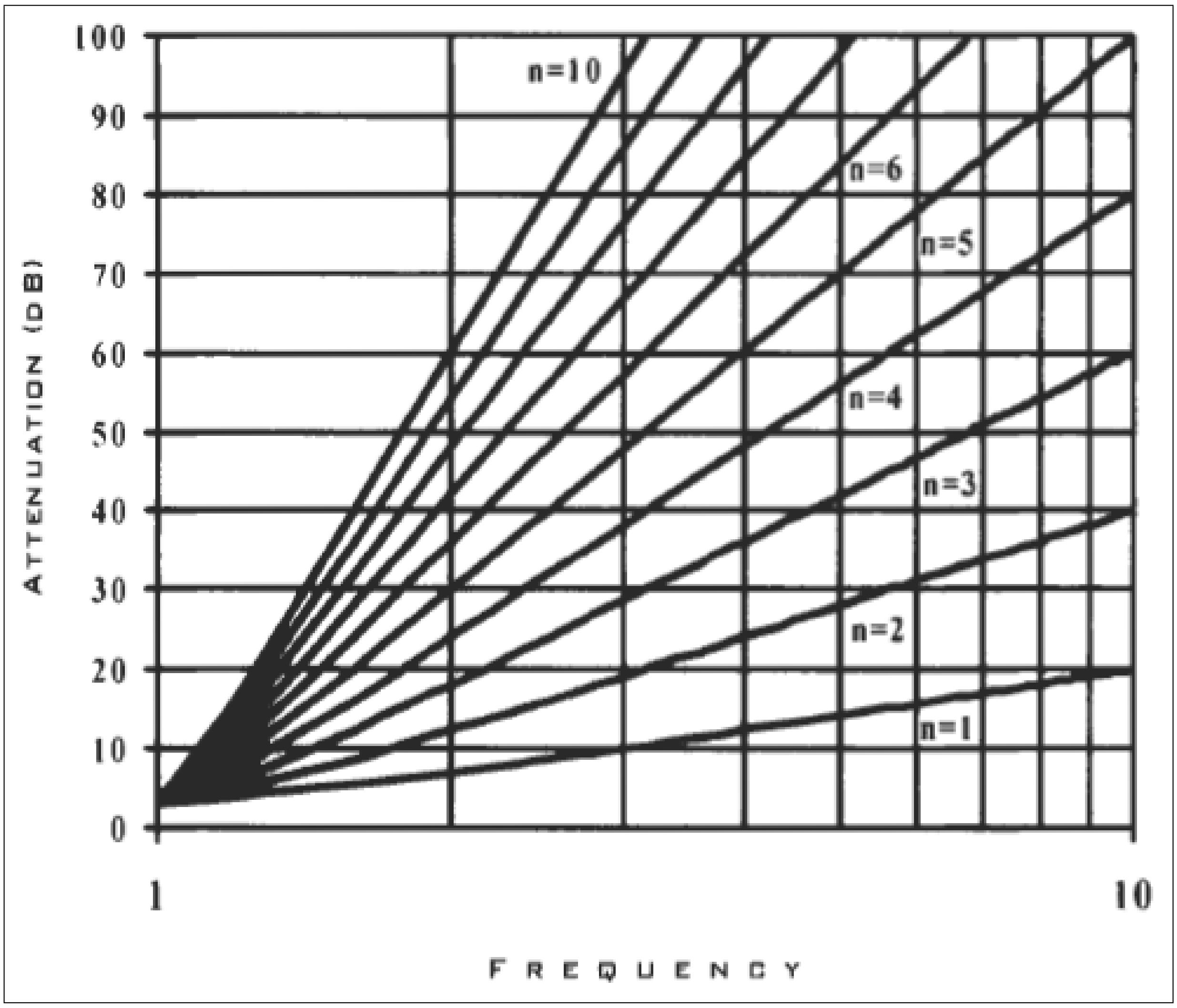


Figure 5.10: Attenuation of the Butterworth filter versus the cutoff frequency multiples [33]

When designing a filter, another specification that must be met is if the filter will be active or passive. Active filters are mostly indicated for applications with narrow up to medium bandwidth, like from several kilohertz up to a few megahertz [33]. Considering that the PLC HomePlug AV bandwidth is very wide, 26 MHz, and the interface has no power supply, a passive filter made of lossless electronic components is suitable.

Previously on this chapter, it was said that the source impedance (power line impedance) inside a home would be no greater than ten times the load impedance (USRP input impedance) throughout the time, given the random switching of many appliances. Considering this, the passband filter will have five poles and will be designed based on the 5th row of Table 5.1 [33], that has values for $R_S = R_L$, when the source and load impedances are not different from each other more than ten times. This table is normalized for $R_S = R_L = 1\Omega$ and for cutoff frequency equal to 1 rad/s, but as the load impedance is 50Ω and the desired cutoff frequencies are different, the elements values must be denormalized.

The last step for designing the passband filter is to establish the initial low and high frequencies values: 1.8 MHz for the high-pass filter and 32 MHz for the low-pass filter, since the HomePlug AV operational frequency ranges from 2 MHz up to 28 MHz.

Concerning the passband filter composition as the combination of a high-pass filter connected in series with a low-pass filter, only to make the understanding easier, the low-pass filter was designed fist than the high-pass filter.

Table 5.1: Normalized Butterworth Element Values, $R_S = R_L = 1\Omega$ [33]

Order	C1	L2	C3	L4	C5	L6	C7	L8	C9	L10
1	2.000	-	-	-	-	-	-	-	-	-
2	1.414	1.414	-	-	-	-	-	-	-	-
3	1.000	2.000	1.000	-	-	-	-	-	-	-
4	0.765	1.847	1.847	0.765	-	-	-	-	-	-
5	0.618	1.618	2.000	1.618	0.618	-	-	-	-	-
6	0.517	1.414	1.931	1.931	1.414	0.517	-	-	-	-
7	0.445	1.246	1.801	2.000	1.801	1.246	0.445	-	-	-
8	0.390	1.111	1.662	1.961	1.961	1.662	1.111	0.390	-	-
9	0.347	1.000	1.532	1.879	2.000	1.879	1.532	1.000	0.347	-
10	0.312	0.907	1.414	1.782	1.975	1.975	1.782	1.414	0.907	0.312
Order	L1'	C2'	L3'	C4'	L5'	C6'	L7'	C8'	L9'	C10'

The low-pass passive Butterworth filter circuit is a network ladder with alternating components like series inductors and shunt capacitors [33]. The order of the filter establishes the number of components. This kind of filter can start with any component but needs to alternate them. Regarding the fact that inductors are a little more difficult to find in stores, it's desired that the filter design has a low number of them; this is why the filter was designed starting with a capacitor. Therefore, in Table 5.1 the order of the elements is: C1, L2, C3, L4 and C5 as shown in Figure 5.11.

The denormalization for a higher impedance requires each component value to be scaled to have a higher impedance. Thus, as the inductor's impedance is proportional to its inductance, the inductance is proportionally increased. On the other hand, as the capacitor's impedance is inversely proportional to its capacitance, the capacitance is decreased proportionally. The proportion of increase and increase of impedance and capacitance is given according to the amount of the load resistance, in other words, this factor will be exactly the load value due to the normalization Table 5.1 for 1Ω. However, the cutoff frequency denormalization requires that the impedance values remain unaltered. To meet this requirement, both inductance and capacitance are reduced proportionally to the cutoff frequency. All the impedance and frequency denormalization steps can be put together into simple mathematical expressions for both inductor and capacitor, and are shown by Equations 5.6 and 5.7 respectively [33].

$$L = \frac{RL^*}{2\pi f_{cutoff}} \tag{5.6}$$

where,

L completely denormalized inductance;

L^* normalized inductance;

R load impedance; and

f_{cutoff} cutoff frequency.

$$C = \frac{C^*}{R2\pi f_{cutoff}} \tag{5.7}$$

where,

C completely denormalized capacitance;

C^* normalized capacitance;

R load impedance; and

f_{cutoff} cutoff frequency.

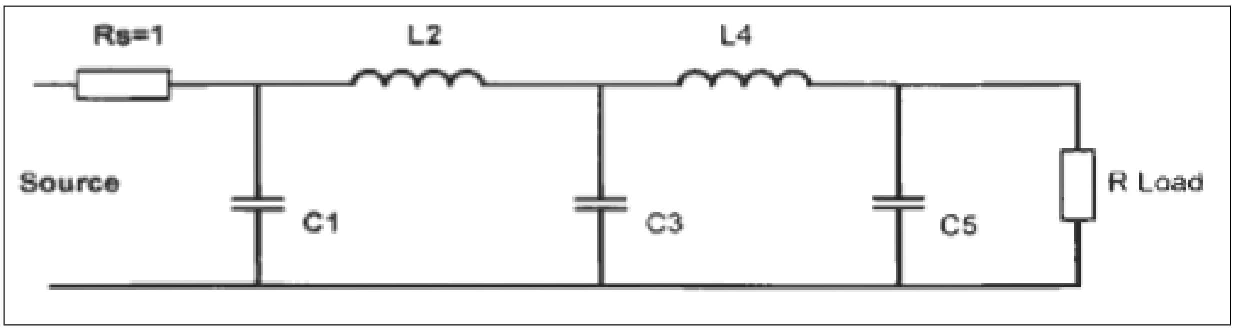


Figure 5.11: Low-pass Butterworth network ladder starting with shunt capacitor

After denormalizing all the values of the 5th row of Table 5.1 with R=50 Ω and f_{cutoff}=32 MHz the values are the following: C1=61.47 pF; L2=0.4 μH; C3=198.94 pF; L4=0.4 μH; C5=61.47 pF.

The high-pass passive Butterworth filter is also a network ladder of inductors and capacitors with the same architecture of the low-pass filter, but each element of the low-pass filter is converted into an element of the high-pass filter. This is why the low-pass was designed first. In the high-pass filter, each capacitor from the low-pass filter is replaced by an inductor with the reciprocal normalized capacitor's value and each inductor in the low-pass filter is replaced by a capacitor with the reciprocal normalized inductor's value. After all these conversions, the first element will be a shunt inductor, the second a series capacitor, and so on, as illustrated by Figure 5.12.

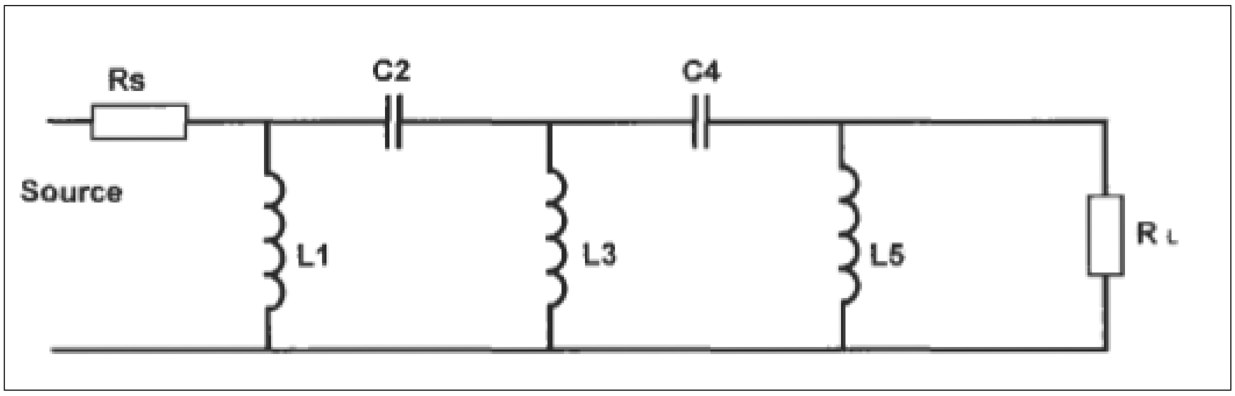


Figure 5.12: High-pass Butterworth network ladder converted from low-pass filter

The values of the high-pass filter elements are calculated as shown in Equation 5.8 for capacitors and inductors.

$$L1 = \frac{1}{C1} = \frac{1}{0.61803} = 1.61804 \tag{5.8}$$

Thus the normalized high-pass elements are: L1=1.61804; C2=0.61803; L3=0.5; C4=0.61803; L5=1.61804.

On this last design, there are three inductors, so this is not a minimum inductor design and there is the need to have as few as possible inductors inside the filter. Each component on this high-pass filter can be replaced by its 'opposite', i.e., a shunt inductor can be replaced by a series capacitor of the same value, and a series capacitor is replaced by a shunt inductor. The resultant circuit has less inductors than the original one, however it still has an equal response.

Consequently, the previous elements turned into: C1=1.61804; L2=0.61803; C3=0.5; L4=0.61803; C5=1.61804. Figure 5.13 illustrates the network ladder of the minimum inductor high-pass Butterworth filter.

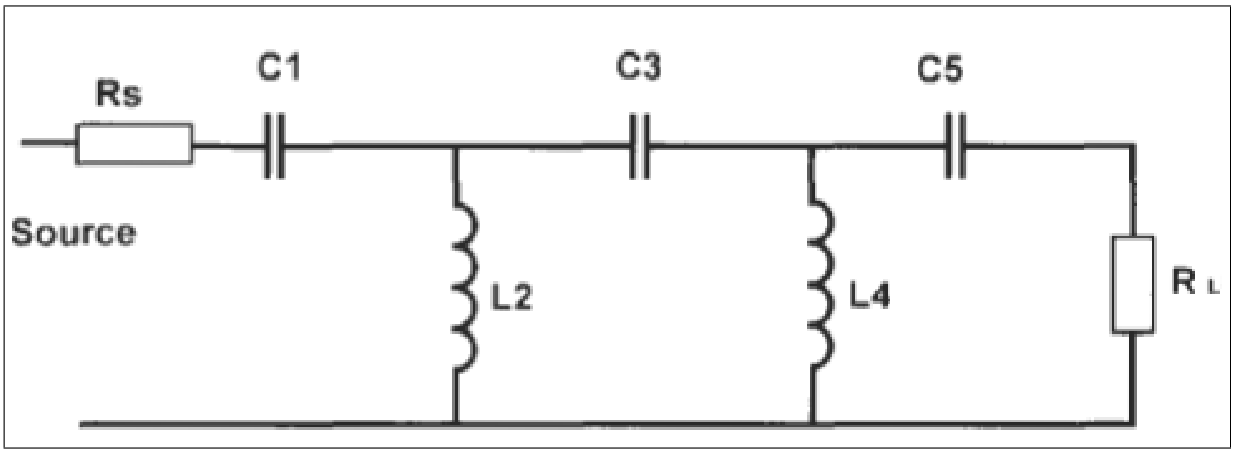


Figure 5.13: Minimum inductor high-pass Butterworth network ladder

Only after this last conversion, the high-pass filter elements can be denormalized in the same way as in the low-pass filter, using Equations 5.6 and 5.7, but with a cutoff frequency of 1.8 MHz. The denormalized values are the following: C1= 2.86 nF; L2= 2.73 μH; C3= 0.884 nF; L4= 2.73 μH; C5=2.86 nF.

After having both low-pass and high-pass element's values, the filter was simulated using OrCAD software to show the theoretical frequency response of the complete passband filter. Figure 5.14 shows the designed filter electric circuit schematic. Just for simulation reasons it was used a voltage source with 3 V peak RMS with an source impedance of 50 Ω and an equal load impedance. It was made a frequency sweep from 1 Hz up to 100 MHz. Figures 5.15 and 5.16 shows the filter frequency response amplitude in log and linear scales respectively.

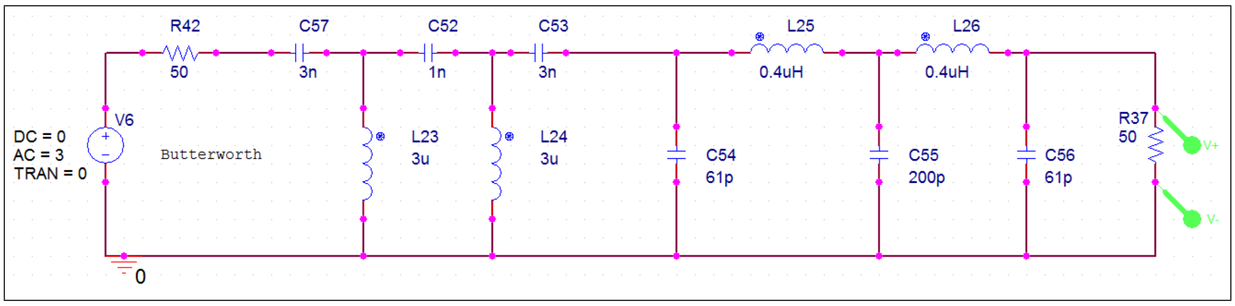


Figure 5.14: Passband Butterworth filter schematic

The linear scale frequency response shows with more detail that at 2 MHz the re-

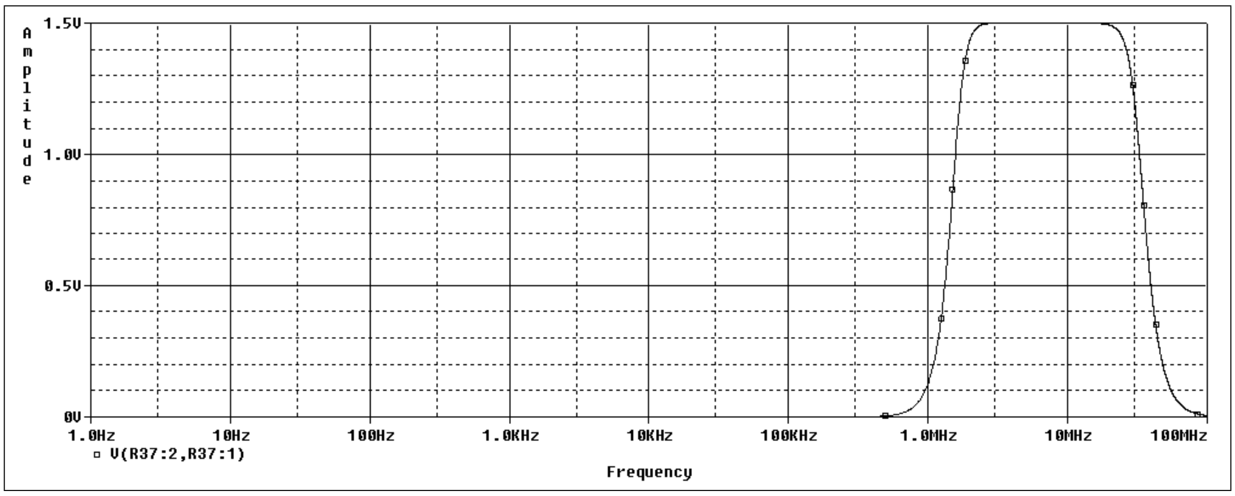


Figure 5.15: Passband Butterworth filter response - log scale

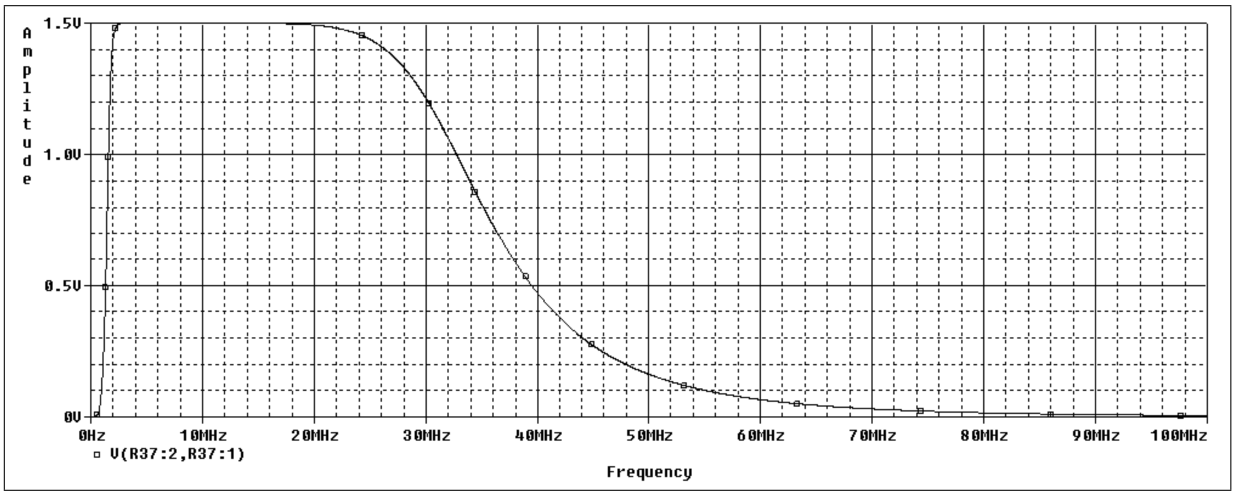


Figure 5.16: Passband Butterworth filter response - linear scale

sponse reaches 100 % of the maximum voltage allowed, 1.5 V, and just after 20 MHz the attenuation starts, reaching 70.7 % of its total value at 32 MHz as desired. However, if a flat response during the entire PLC bandwidth is desired, the low-pass filter should be designed for a higher cutoff frequency, for example, 42 MHz. This result in the new low-pass filter elements values: C1=46.83 pF; L2=0.3 μH; C3=151.57 pF; L4=0.3 μH; C5=46.83 pF.

The new frequency response in linear and log scales are shown in Figures 5.17 and 5.18 respectively. There was a good improvement on the 20 MHz up to 30 MHz with the drawback of a little increase in the higher frequencies responses amplitudes. One possible solution to get rid of this high frequencies is to increase the number of poles, with a resultant increase in component number and lower inductance inductors, which represent a huge percentage of the total cost of the filter. If the cost isn't a relevant issue, one could increase the low-pass filter pole number and achieve a sharper response. In the prototype described on the next section it was used only the 5 pole filter design.

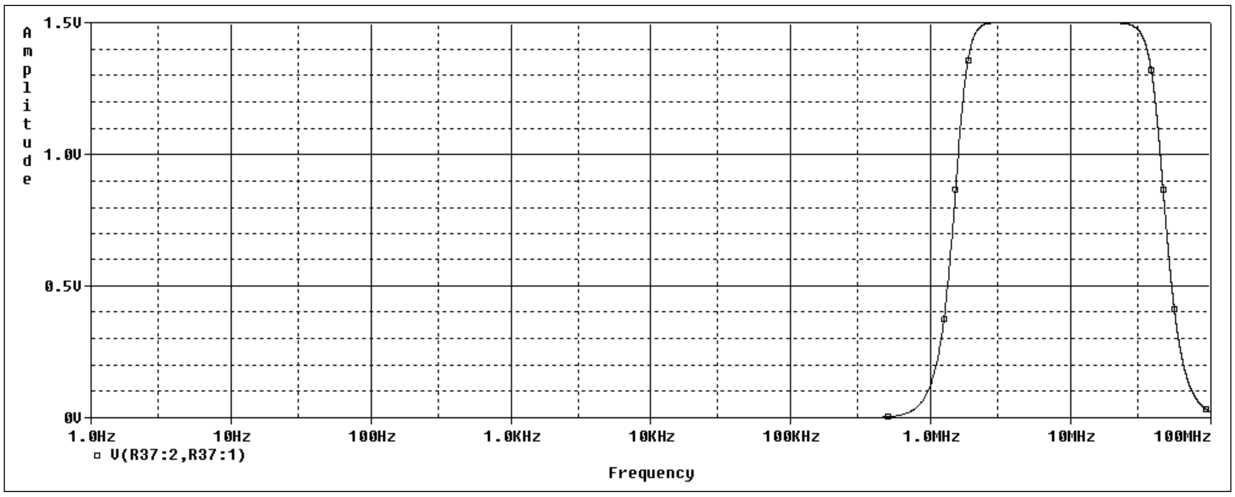


Figure 5.17: New passband Butterworth filter response - log scale

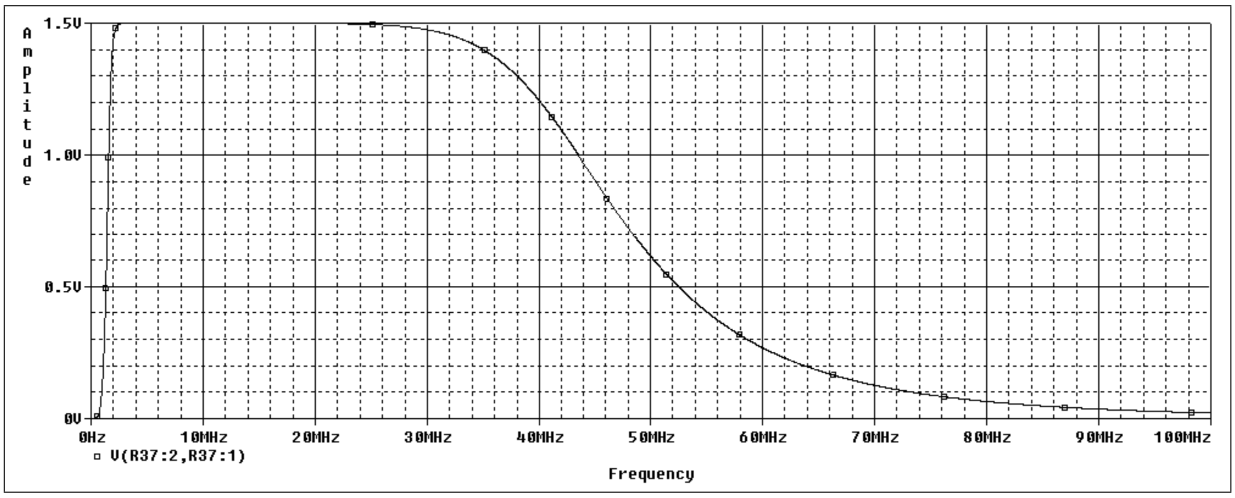


Figure 5.18: New passband Butterworth filter response - linear scale

It's worth noting that if the PLC-USRP interface is intended to be used inside a low-voltage building at 220 V or 110 V, the filter should be symmetric, i.e., all the three series capacitors and all the two series inductors should also be implemented on the other branch of the filter, which would also increase the symmetrical degree for EMC [21]. In addition to that, the symmetrical design may have a higher cost, and even higher if more poles are implemented on the low-pass filter for a sharpen frequency response.

For the prototype developed for this text experiments, only the blocking symmetry was implemented for safety reasons. The filter symmetry was not implemented aiming to keep the design as simple as possible allowing an overall performance analysis of the non-symmetrical interface first, making sure that the proposed interface prototype would work as desired with the minimum requirements.

Finally, the final schematic of the ideal proposed PLC-USRP interface circuit schematic is illustrated by Figure 5.19.

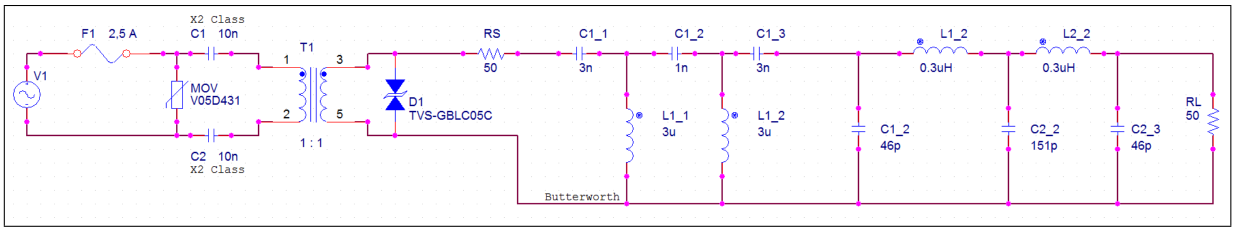


Figure 5.19: Ideal PLC-USRP interface schematic

5.3 The practical PLC-USRP interface design

Based on each component specification detailed on the previous section, the prototype version of the ideal PLC-USRP interface was designed and build. When prototyping, the first action to be taken is to look for the availability of the specified electronic components on the market. When the time and budget is a constraint, as it was the case, only local stores could be contacted. But if the project was a huge one, well financially supported and relied on vital components that could only be found abroad, worldwide stores should be considered jointly with shipping time management.

The local stores could supply only the basic components which were enough for the prototyping and testing of the PLC-USRP interface, as described below. Basic components like fuse, capacitors and inductors could be bought. But specific electronic devices as TVS arrays and 1:1 ratio transformers couldn't. The VDR was found in an printer PCB supply source.

Another constraint regarding prototyping is to find the component with the closest designed value. For example, there is no 46 pF capacitor and an association of 3 15 pF capacitors was used in place. Associations were made also with inductors to achieve as close as possible the designed values.

Table 5.2 lists all components used on the prototype and their total price. In the end were spent approximately a total of 15.67 dollars, without the VDR, the transformer, the TVS array and all the other symmetric components. The 10 nF capacitors aren't a X2 class nor a Y2 class. They're common polyester capacitors, used just for the prototype tests.

Table 5.2: PLC-USRP interface electronic components

Item	Quantity	Price U$
AC cord 6 feet	1	2.5
Fuse	1	$1 * 0.08$
Fuse bed	1	$1 * 0.25$
Capacitor 10 nF / 630 V	2	$2 * 0.45$
VDR 10D391	1	-
Capacitor 1 nF / 50 V	7	$7 * 0.1$
Capacitor 15 pF / 50 V	6	$6 * 0.05$
Capacitor 68 pF / 100 V	2	$2 * 0.07$
Inductor 2.2 μH	16	$16 * 0.6$
Phenolite cooper board 4x6 inches	1	$1 * 1.6$
Total price	-	15.67

After the acquisition of the components, the PCB layout was designed using the free version of ExpressPCB software. The top cooper layer and the silkscreen layer are illustrated by Figure 5.20.

In the end, the circuit was printed over the phenolite board using the transfer paper method through gloss paper laser printed. The chemical used for etching the board by means of a subtractive process for removing the uncovered copper was ferric chloride.

The prototype picture of the PLC-USRP interface after all the assembly is illustrated by Figures 5.21 and 5.22. The black AC cord on the left side connects the interface into the wall plug. On the right side, the green cables connect the interface to the USRP antenna input through a SMA connector.

Finally, to be sure that this circuit is works well, its frequency response amplitude can be acquired using the tool prototyped for this purpose, which was described on Chapter 4. The device under test was the PLC-USRP interface and it was used the best daughter-board configuration found on Chapter 4 (the one that showed a very low attenuation that its effect could be take for granted), LFTX-BasicRx. It's possible to see how close the practical prototype frequency response amplitude is to the OrCAD simulation, taking into account that the components used were found in the very limited local market.

The AFR system configurations were kept the same used on Chapter 4 tests, with the transmitted amplitude of 32000 DAC levels, the total frequency range possible from 1 MHz up to 30 MHz, the time interval after the signal generation as 25 ms and a frequency

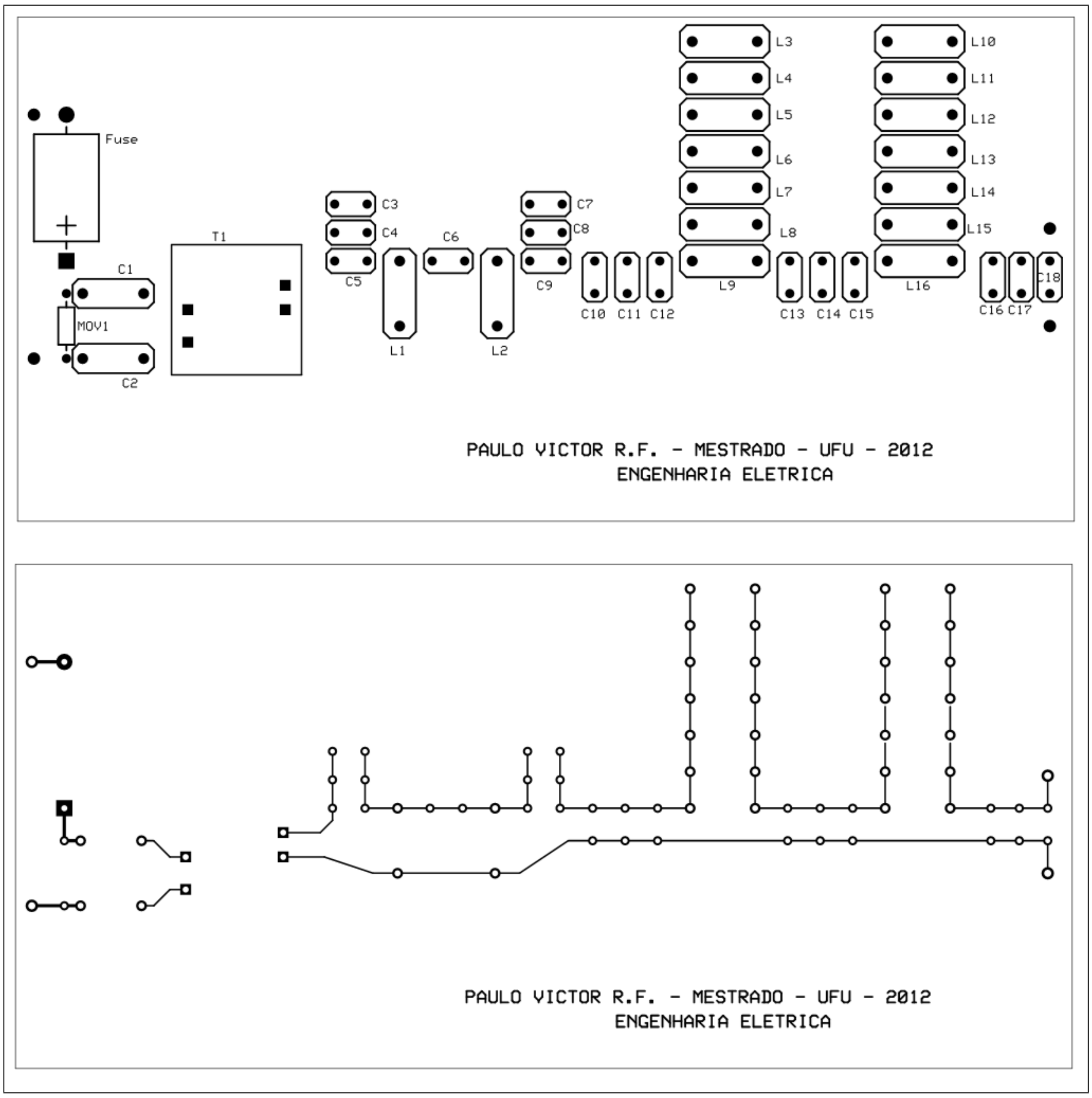


Figure 5.20: PLC-USRP interface PCB layout

resolution of 5 kHz (distance between measurement points). The the software was run and a measurement amplitude file was created.

To evaluate the performance of an intermediary electric circuit (device under test), its response must be obtained without the interferences of the measuring system. So, instead of cancelling the measuring system effects (transceiver frequency response amplitude) in the interface response, the measuring system is took as the reference for the interface response in a dB scale. Each measured amplitude value must be divided by the amplitude value found on the file of the response created when the daughter-boards were connected directly, without an intermediary device, as the ones acquired on Chapter 4.

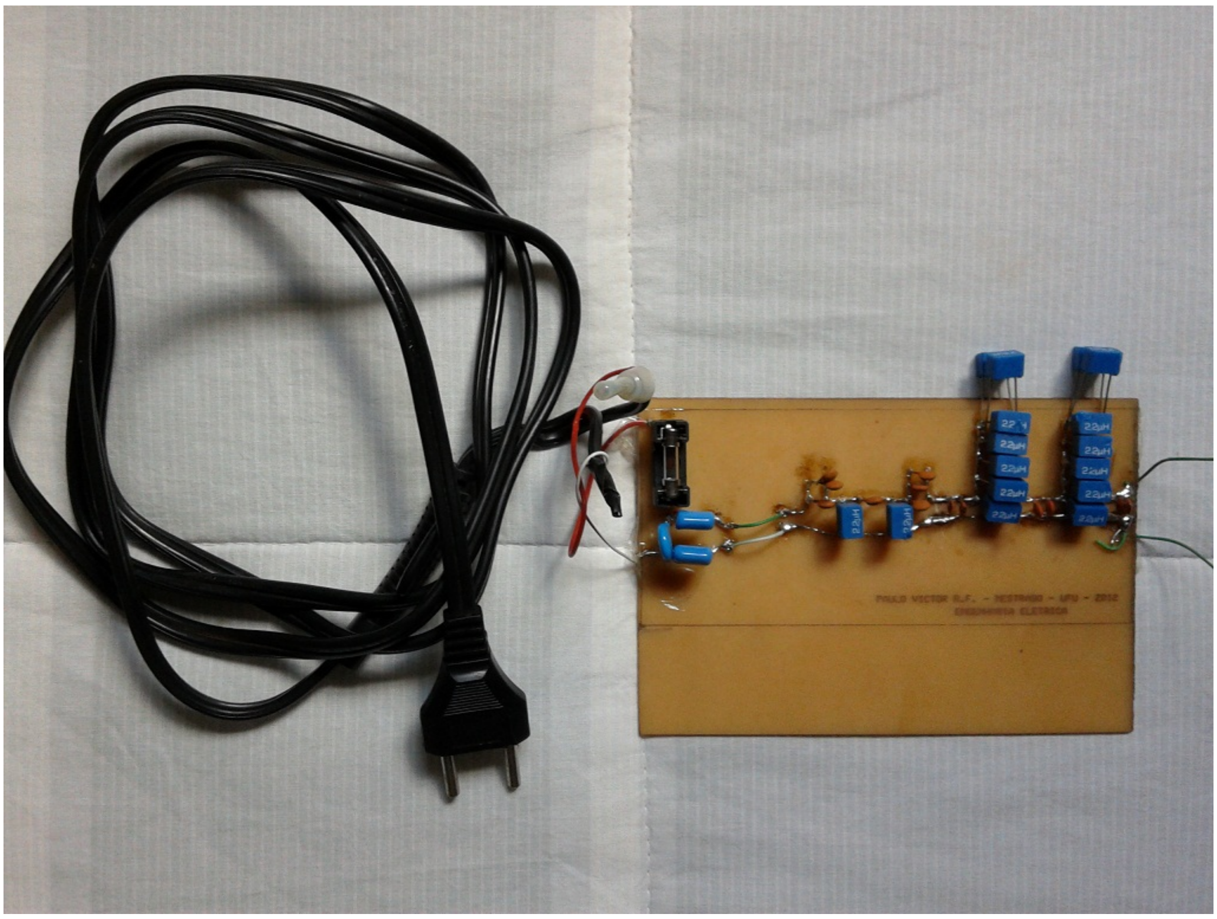


Figure 5.21: PLC-USRP interface prototype with ac cord

The daughter-board configuration used has a frequency response amplitude that can be considered constant through the entire PLC frequency range. This lead to an easier gain assessment coding because only one file had to be handled. The result of this calculus is shown in Figure 5.23. The mouse's arrow was left on the image to show the coordinates values in the left bottom, which shows that the highest PLC attenuation level occurs at 21.489 MHz with an attenuation level of 0.49 dB. This is a consequence of the low-pass filter poles and was expected due to the alternative component values that were used instead of the theoretical ones initially designed. This effect could be previously observed during the OrCAD simulation, when the component values were changed, seeking to minimize the impact over the frequency response when analysing the different values of the market available components.

To summarize, the last picture shows that the PLC-USRP interface prototype, built with available market components, has a frequency response that does not interfere in the signal analysis. Such important feature enables the PLC signal acquisition by the USRP,

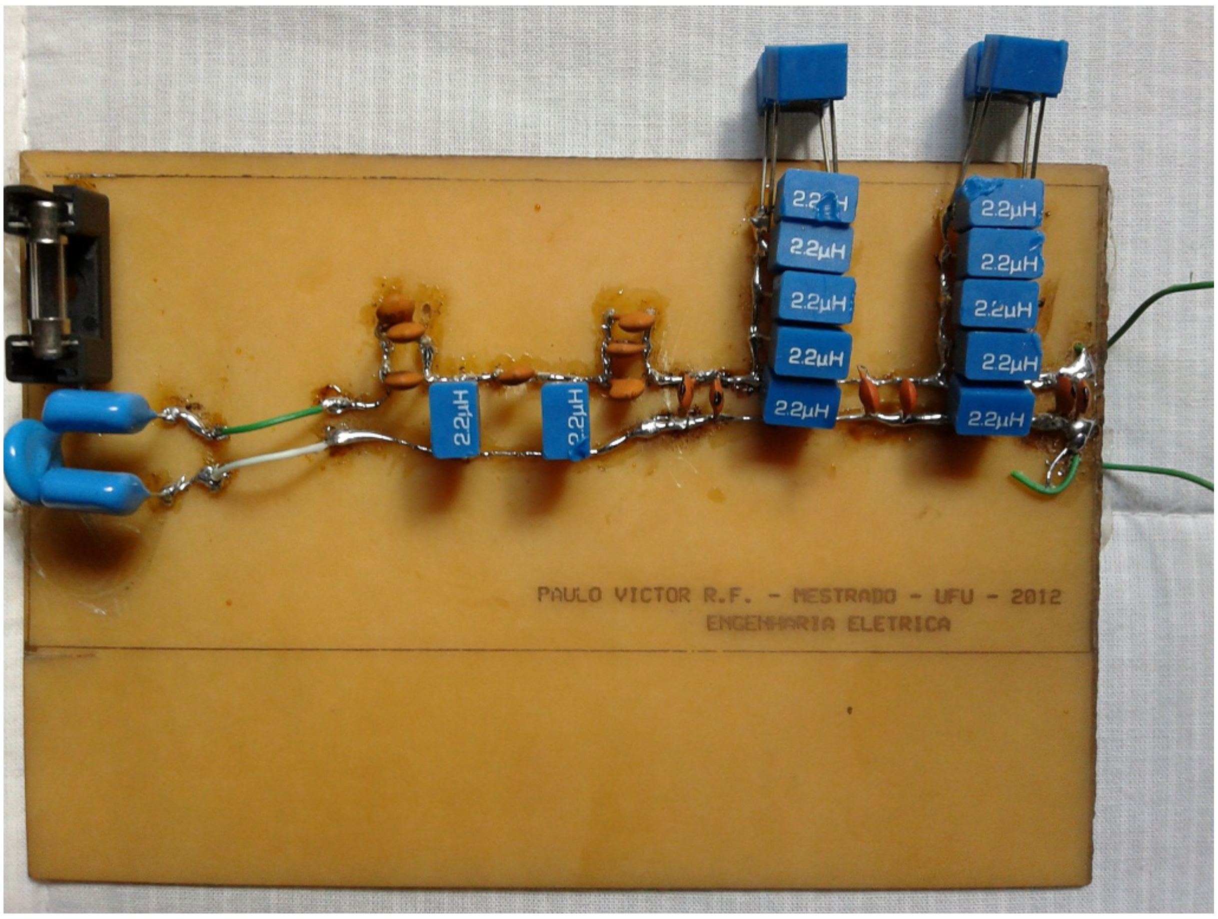


Figure 5.22: Zoomed PLC-USRP interface prototype

which delivers it to the GNU Radio, where all the further desired processing may take place.

5.4 Final considerations of this chapter

This chapter described the PLC and USRP electrical requirements in order to make sure that the PLC output power levels and the allowed USRP input power levels match. In addition, an ideal PLC-USRP interface design was proposed. Due to market unavailability of some components a practical interface design was built based on the ideal one. In the end its gain frequency response was measured using the AFR software prototype.

On the next chapter the PLC signal analyser prototype, which is the solution to the problem described on the first chapter, will be described and tested with an operating PLC network.

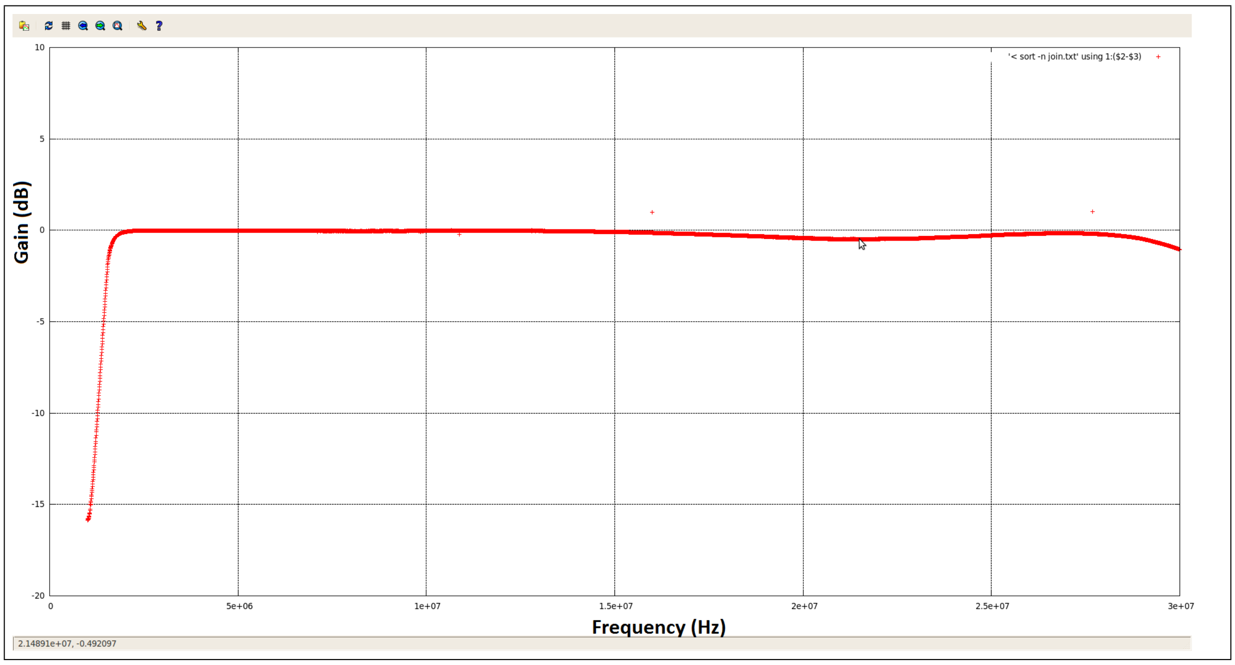


Figure 5.23: PLC-USRP interface prototype gain response using the AFR software prototype

Chapter 6

A low-cost PLC in-building signal analyser prototype

In this chapter the PLC-USRP interface, the USRP module and a GNU Radio receiver are put together to make the acquisition of the PLC signal and process it in the digital domain.

6.1 The basic system

The basic set-up consists of a computer with a Linux operational system (Ubuntu) where is installed the GNU Radio software which is in charge of processing the acquired signal through the USRP module. The USRP is connected to the computer by a USB cable. As previously seen in subsection 4.3, the best daughter-board configuration, which showed the smallest attenuation through the entire PLC frequency range, is the LFTX-BasicRx. However, only the receiver branch will be used, so only the BasicRx daughter-board is used with the USRP module to implement the PLC signal analyser.

The PLC-USRP interface output is connected to the antenna input of the BasicRx daughter-board, and the AC cord (the interface input) is attached to the power line, where there are PLC signals being transmitted and received between the two PLC modems, the same used in Chapter 5.

Figure 6.1 illustrates in a very simple way the diagram block and the connections between the main components that compose the PLC signal analyser prototype.

Before analysing the signals, a PLC network must be deployed. The easiest way to do

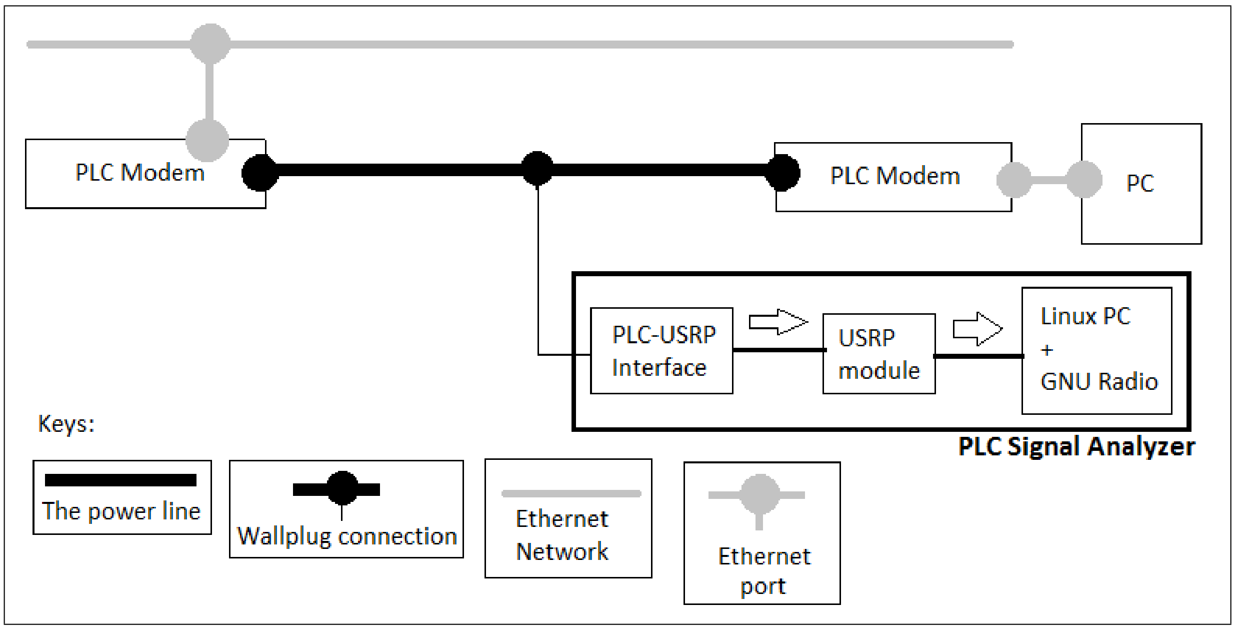


Figure 6.1: PLC signal analyser diagram block

so is to use one of the modems as a signal injector, i. e., a local network gateway, and the other as the client, connected to a host computer that requests data from the network or provides data to the network.

The basic aim of the PLC signal analyser system is to provide the PLC signal to be processed by a software. And to show this principle using the arrangement described above, the acquired PLC signal will have its Fast Fourier Transform calculated by the GNU Radio software, resulting in the spectrum of the current PLC signal being transmitted by the power line and showed live on the computer screen.

The spectrum acquisition is only one of many possible signal processing applications that can be developed using the PLC signal analyser prototype, because its main feature relies on the fact that the live digital PLC signal is provided to a processing software tool. Once inside a processing tool as powerful as GNU Radio is, a lot of processing can be done based on the PLC signal being transmitted or received within the power line network. From transmitted or received power measurement up to and beyond developing cognitive algorithms, making decisions over the current PLC network state for system management, making improvements according to the varying traffic demand, and so on. It is also worth saying that all the 'intelligence' developed using this technique for acquisition and processing the PLC signal at low-voltage in-building power line network can be applied at any other PLC network, for example embedded in a car or aircraft or even in a high-voltage network with the right conditioning of the signal by an interface designed for that purpose, which may deal with high-voltage filtering in this case.

As previously seen in Section 3.2, the maximum bandwidth that can be acquired is 8 MHz, with this USB 2.0 interface implemented on the USRP1 when 16-bits samples are used. Thus, the PLC bandwidth is 26 MHz, ranging from 2 MHz up to 28 MHz. Considering this, to make the acquisition of the complete PLC spectrum the process needs to be repeated 4 times, giving 4 windows that together cover the entire desired bandwidth. Inside the USRP, the data samples are always 16-bits, but by Python it is possible to change the sample size over the USB down to 8-bits, doubling the data rate to 16 M complex sps [13], which results in a bandwidth of 16 MHz, at the cost of decreasing the precision levels.

It is worth saying that the PLC signal is transmitted over only two wires, i.e., it will use only one antenna connector of the USRP module daughter-board, BasicRx's RXA connector, using just one ADC signal path because it is not a quadrature signal. Like the USRP always uses complex sampling, the other ADC signal path, for this application, is unused. Following the ADC's outputs, and before entering into the FPGA DDC, there is a Front-End routing logic, which fills the unused ADC path with null samples after being set by the user, as illustrated by Figure 6.2 [27]. When the 'imaginary' input is filled with zeros, the spectrum becomes complex.

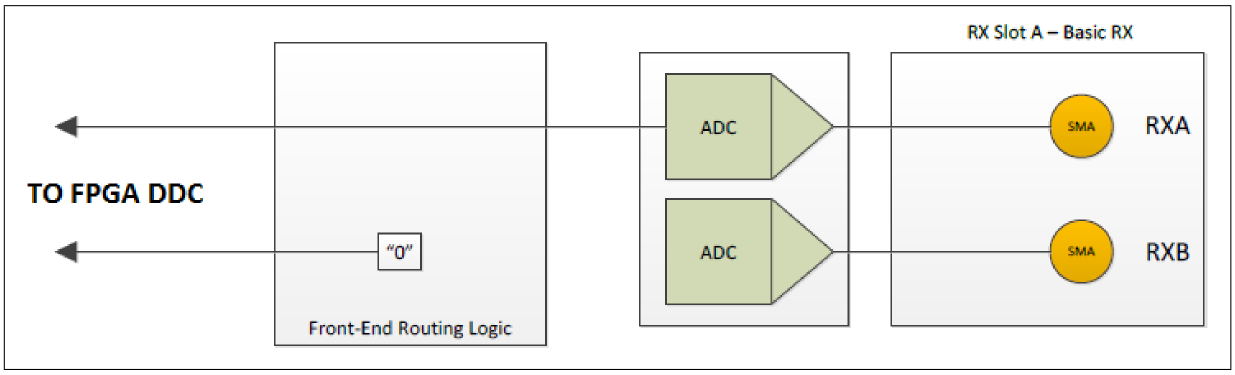


Figure 6.2: USRP Front-End routing

At the DDC input the signal is down-converted from IF to baseband by a complex multiplier due to the USRP complex sampling technique used. One of the complex multiplier inputs will be the PLC sampled signal and the other, the null-samples sourced by the Front-End logic, both sampled at the ADC rate of 64 Msps, as illustrated by Figure 6.3 [13]. After the down conversion the original IF real signal becomes a complex baseband signal and the decimation takes place. At the DDC output there are the resultant I and Q signals which are handled by the GNU Radio software. After the complex multiplier the total bandwidth remains unaltered. The complex multiplication is illustrated in Figure 6.4, and for the PLC signal acquisition experiment, the Input 2 (RXB) path has only null-samples, so the two parts on the result after the multiplication that have a B term are zero. It is worth noting that there is no magic, both I and Q paths are sampled at the same time (when using 16-bits samples, each of these paths has a bandwidth of 4 MHz, so the total bandwidth is 8 MHz) and the complex multiplier allows the system to deal with quadrature signals and IF signals (signals with only real part) as well. Thus, unless otherwise stated, inside the GNU Radio, the input data is always handled as being complex, but can be split into its real and imaginary parts by some processing blocks.

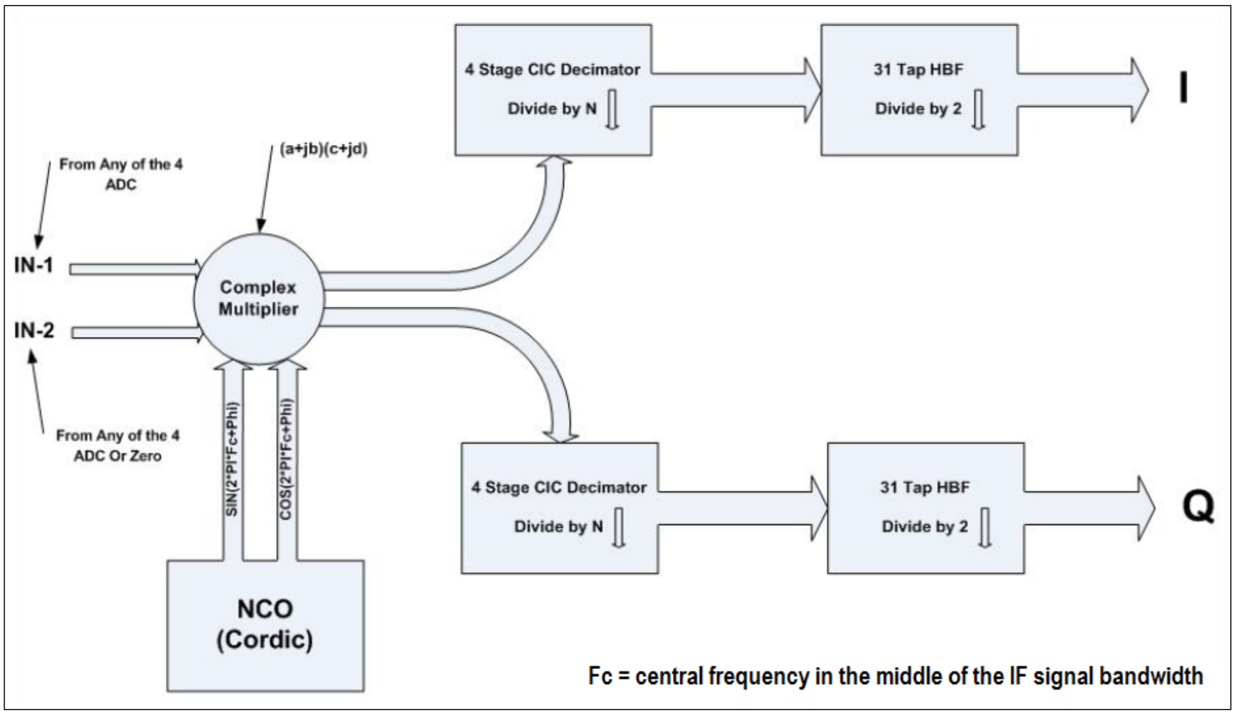


Figure 6.3: USRP FPGA DDC

After making all the connections shown in Figure 6.1, the GNU Radio gets back to the spotlight and needs to be configured. The GNU Radio configuration relies basically on developing a software based digital receiver, using the GRC, as simple as possible to compute the FFT in order to show the live PLC signal spectrum inside the power line.

Concerning this, the GNU Radio receiver must have a source block to interface with the USRP 'real-world' module, which will decimate the ADC sample rate down to the USB acceptable data rate. It will be followed by a FFT Sink which calculates the FFT

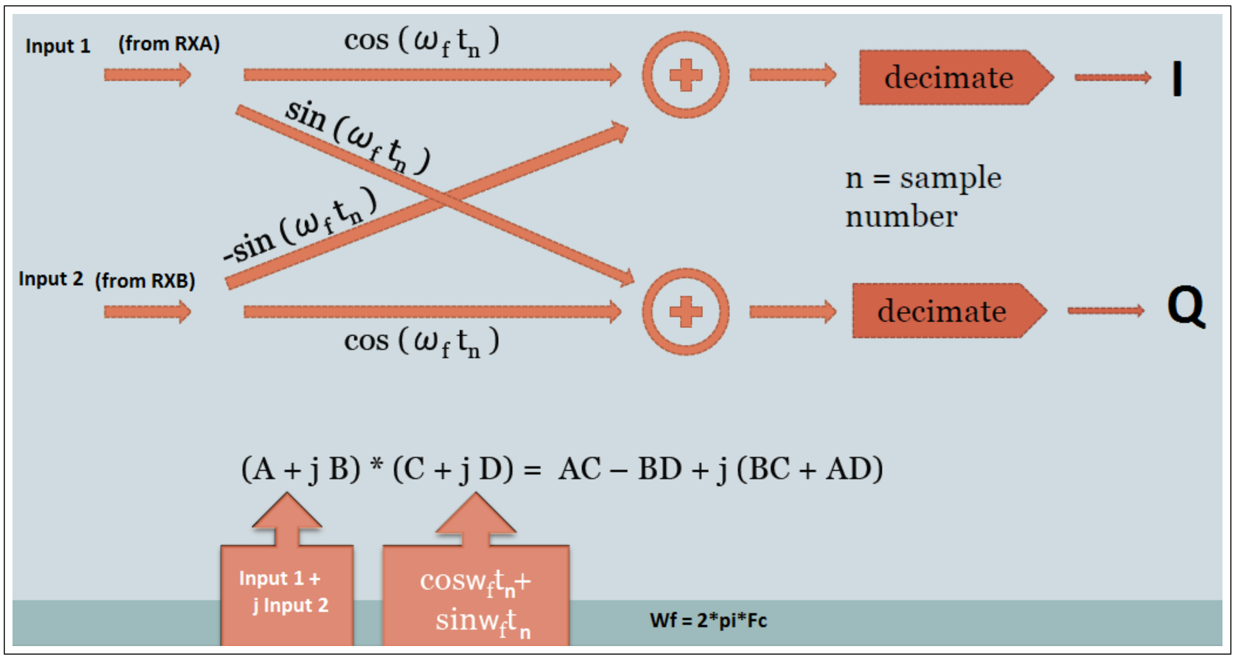


Figure 6.4: Complex multiplication

of its input signal and shows it through a FFT plot window in a continuous basis.

As previously widely discussed, the complex sample rate that gives the maximum amplitude precision regarding the USB constraints needs to be 8 M complex sps. This is the GNU Radio USRP source sample Rate and if requested, this sample rate comes from a decimation factor of 8. The Source Frequency (Hz) is the current frequency tuned by the USRP. Using a complex signal, the spectrum (FFT Plot) is centred at this frequency and is set at the same time through the same virtual variable that sets the FFT Sink baseband frequency. For processing purposes only, before every graphical sink (Spectrum scope or oscilloscope) a Throttle block is required to try to make the computer to keep up with the processed samples delivered by the USRP via the USB cable. If it can not read all of them an 'O' error appears indicating overrun, i.e. the computer is dropping some samples. A better processor would solve this.

In the USRP Source, the antenna is set to just one, A or B, given that the PLC-USRP interface output will connect just to one connector, and by doing so the Front-End logic is set to fill its imaginary output with null-samples.

The final GNU Radio receiver that makes the FFT plot showing the power line PLC signal spectrum is illustrated by Figure 6.5. When the final tests were performed it was used the just released GNU Radio version 3.6.2 running on the new Ubuntu version 12.04 Long-term support (LTS). In this version many old bugs had been fixed, it is more stable working with the USRP1 and a very clean installation was performed, just running the Code 6.1 in the Terminal. Clean installations on Linux are desired because they give a certain guarantee level that the all the functions are installed and that interconnected functions may work well. GNU Radio can not be installed over another previous installation.

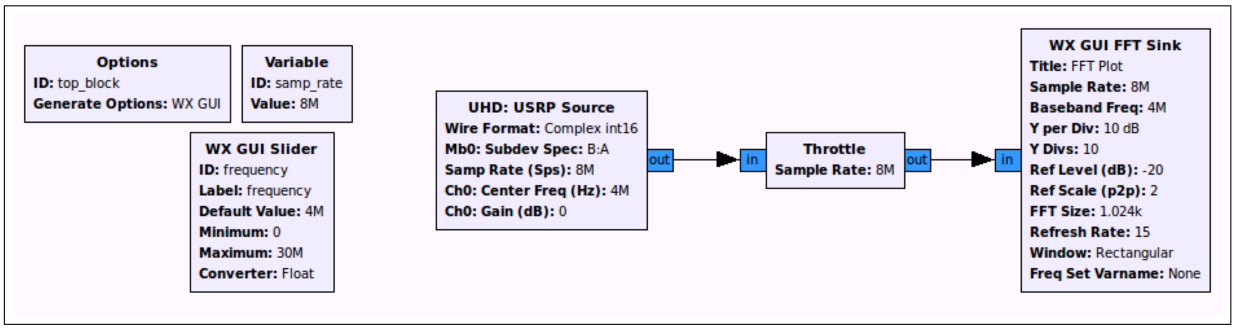
```
$ wget http://www.sbrac.org/files/build-gnuradio && chmod a+x ./build-gnuradio && ./build-gnuradio
```

Code 6.1 - GNU Radio installer script

The Code 6.1 will install all the pre-requisites packages that are required for compiling various parts of GNU Radio on Ubuntu. This method installs all the brand new version of GNU Radio and its requisites, including the new hardware driver, called USRP Hardware Driver (UHD). Because of this new driver, the USRP Source and Sink processing blocks configurations inside GNU Radio changed. On the older versions the Decimation and Interpolation values were configured to match the internal sample rate considering the ADC and DAC sample rate. On the new version the user must set the internal sample rate directly making the usage much easier. So, there is no need to remember the ADC and DAC sample rate all the time, divide it by the desired internal sample rate and find out the decimation or interpolation factor.

The receiver sample rate was set to 8 M complex sps and resulted in a 8 MHz bandwidth spectrum. The sample format over the USB (Wire format) was set to Complex 16-bits, with 4 bytes per sample. The receiver tuned frequency was controlled by varying a virtual slider from 0 Hz up to 30 MHz, which set a variable used by both USRP Source and FFT Sink blocks. The UHD Center Frequency is the frequency tuned by the USRP and receives the value from the virtual slider. The FFT Sink also used the 8 M complex samples and had its baseband frequency tuned by the virtual slider.

The PLC network was deployed inside a home building fed by a low-voltage power line of 220 V at 60 Hz. During the tests, the devices attached to the power line network were just the PLC modems, the PC, the USRP, the LAN router, an ADSL modem, the kitchen refrigerator, all turned on. In stand by there was a television with its set up box. Connected, but turned off, there were some light bulbs (10 at least, 25 W). Also there were several unmatched wall plugs (outlets). One of the modems injected the PLC signal into the power lines and the other acted as the receiver, requesting a video streaming.


Options
ID: top_block
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 8M

WX GUI Slider
ID: frequency
Label: frequency
Default Value: 4M
Minimum: 0
Maximum: 30M
Converter: Float

UHD: USRP Source
Wire Format: Complex int16
Mb0: Subdev Spec: B:A
Samp Rate (Sps): 8M
Ch0: Center Freq (Hz): 4M
Ch0: Gain (dB): 0

Throttle
Sample Rate: 8M

WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 8M
Baseband Freq: 4M
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): -20
Ref Scale (p2p): 2
FFT Size: 1.024k
Refresh Rate: 15
Window: Rectangular
Freq Set Varname: None

Figure 6.5: GNU Radio receiver

The PLC modems were at least 7 meters far from each other. The PLC-USRP interface was plugged 0.5 meters, near to the receiver PLC modem.

6.2 The results

With all the connections made and the receiver set up, the receiver was ran and the frequency tuned to all the values of 4 MHz, 12 MHz, 20 MHz and 28 MHz in order to get the entire PLC spectrum during the video streaming. All these frequency tunings are due to the fact of the ‘limited’ 8 MHz USRP bandwidth facing the 26 MHz PLC bandwidth.

The acquired spectrum windows are shown in Figures 6.6, 6.7, 6.8 and 6.9 respectively. The spectrum changes very quickly and jumps from the noise level up to the transmission power levels very quickly depending on the transmitted data rate. In order to save this maximum measurements, the FFT Peak Hold function was turned on during the entire acquisition period, so in the FFT figures, the blue signal is the current FFT and the green signal is the just measured maximum peak that took place during the last transmission through the power lines and were captured at the location where the PLC-USRP interface was plugged. All the spectrum windows below have a 0 dB scaled to the maximum USRP input voltage level allowed by the ADC, 2 volts peak to peak or 0.707 volts RMS. This reference can be changed at the FFT Sink block in the Ref. Scale (p2p) value.

It is possible to see that the PLC spectrum starts just after 2 MHz and stops just before 28 MHz as shown by Figures 6.6 and 6.9, respectively.

On almost all four FFT windows the FFT values next to the left and right sides showed different attenuation levels, depending on the FFT window type used, which can be set at the FFT Sink block. The analysis of the effects of the FFT windows are out of the scope of the experiments of this text, but the effect of different windows types was observed through some tests.

On all PLC spectrum figures acquired, during the transmissions the noise level was also increased, from -115 db to -70 dB in average. In Figure 6.6, on the left side, below 500 kHz it is possible to see that the peak amplitude reached -60 dB during the transmission of a PLC signal, although the current FFT (blue signal) shows a value below -110 dB, which is 10 dB over the noise level during a transmission but 15 dB far from the power at the frequencies used for transmission. This fact could be due to the many reflected signals by the appliances and wall plugs unmatched (without any device attached to it).

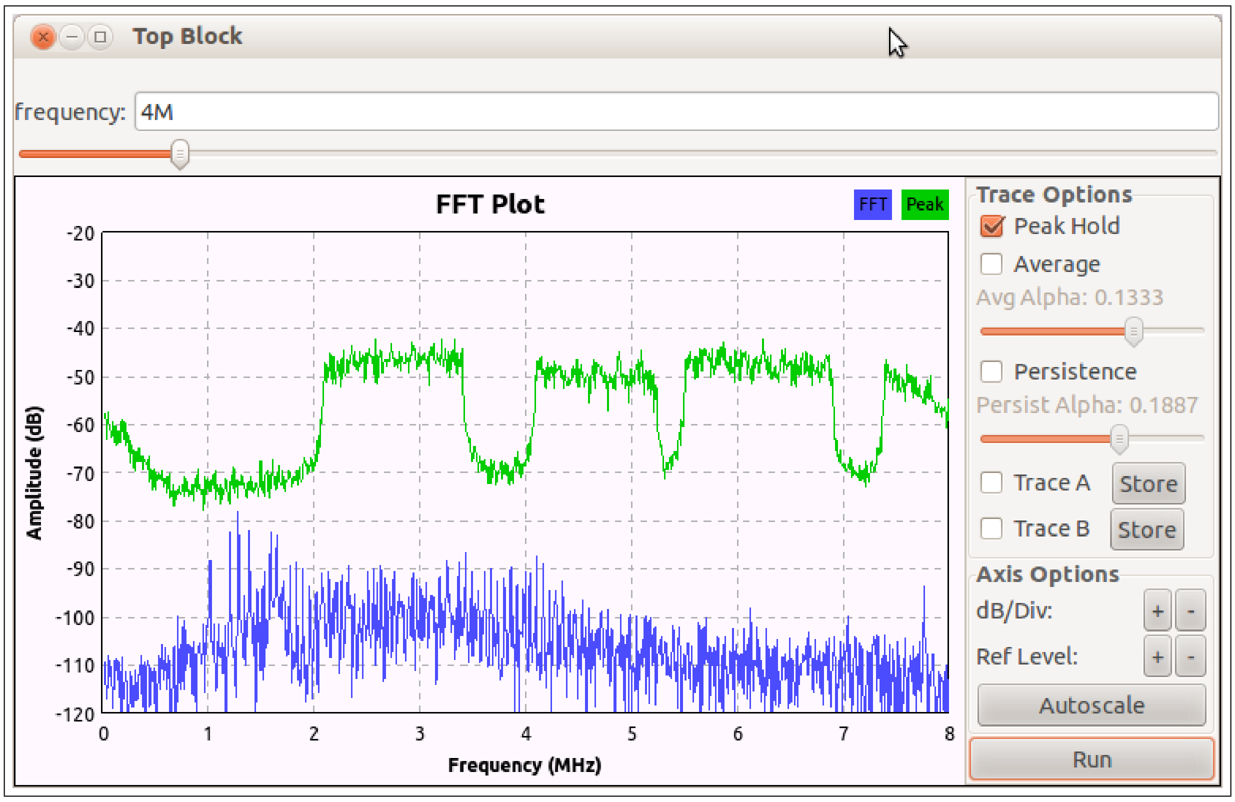


Figure 6.6: PLC FFT Plot from 0 up to 8 MHz

The computer system performance monitor was recorded during the spectrum acquisition and is shown in Figure 6.10. The average download data rate was 100 kbps. The computer used has an Intel Core2 Duo processor with two 2 GHz cores and 4 GB of RAM memory.

During the spectrum acquisition it was observed that depending on the current channel conditions, based on the previous transmissions (based on the exchange of control parameters between the two modems) and depending on the transmitted OFDM symbol, the transmitter modem changes modulation, the number of carriers and the transmission power. Therefore, there are moments when there are a high density of carriers like the

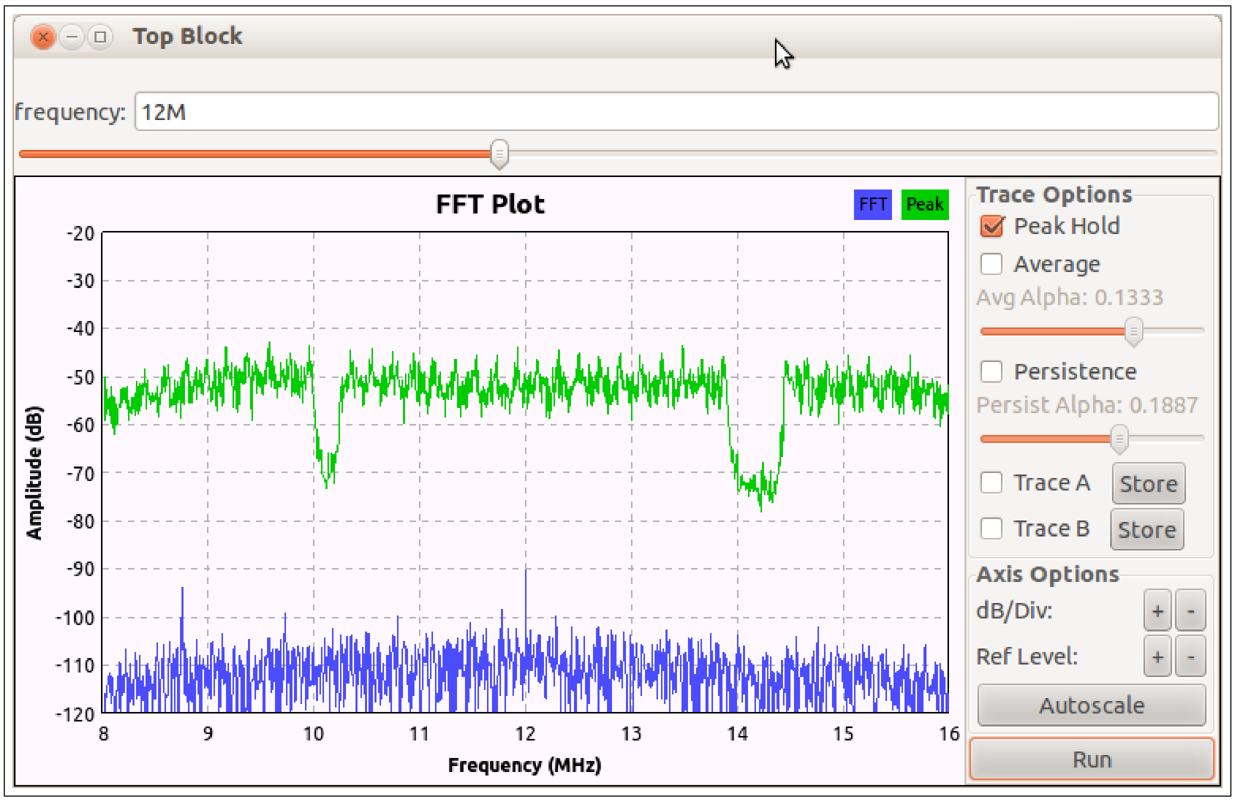


Figure 6.7: PLC FFT Plot from 8 up to 16 MHz

ones shown on the last four FFT plots. But also there are moments where there are a low number of carriers and is possible to see each of them, like the ones illustrated by Figure 6.11 tuned at 4 MHz, and Figure 6.12, tuned at 28 MHz.

Figure 6.13 illustrates PLC spectrum tuned at 28 MHz when the transmitter used many carriers but the transmitted power was lower than that observed on the previous FFT plots, tuned at the same frequency, including the one with a lot of carriers too.

The FFT plot also has another feature: the Average. It measures the average spectrum dB levels, so there is no quick changes and the transmitted carriers can be easily seen going up and down, or, on and off. The Peak Hold can also be used, resulting in the average peak. The 28 MHz centred average spectrum is illustrated by Figure 6.14.

Just to explore the USRP spectrum acquisition limits, the maximum USB bandwidth was tried out, setting the USRP Source Wire Type to Complex 8-bits, resulting in a 2 bytes complex sample, increasing the sample rate to 16 M complex sps and the spectrum bandwidth to 16 MHz. The GRC receiver is shown by Figure 6.15. During the FFT plot observations, the system dropped a high number of samples and the error warning of system overrun did not stopped. Also, as the precision level measurement was directly affected, it was not possible to continuously see the plot at the window. Only when the

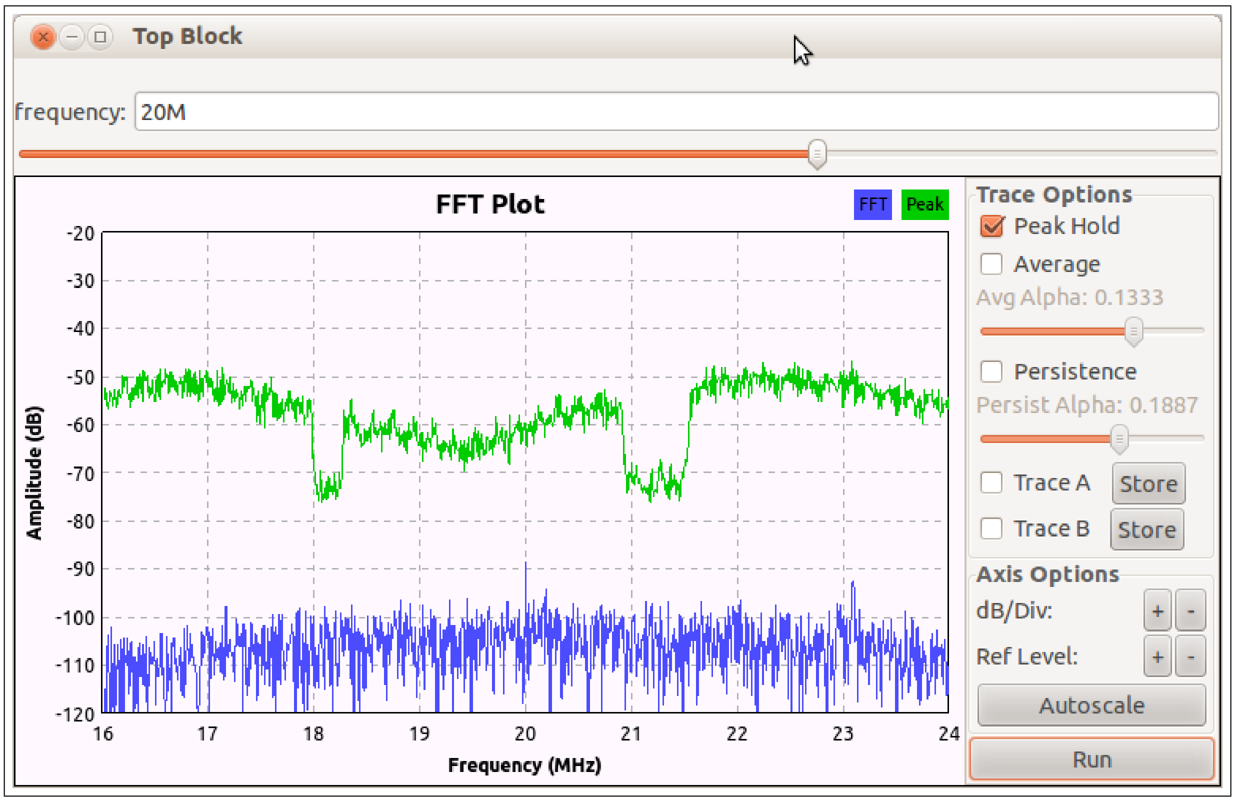


Figure 6.8: PLC FFT Plot from 16 up to 24 MHz

signal reached its maximum value (during a transmission), it blinked on the screen. All these observations could only be reached after decreasing the FFT size down to 256 points.

Another constraint was faced regarding the FPGA image which is loaded into the USRP every time a GNU Radio is run. When using a sample rate higher than 8 M complex sps (any decimation factor below 8), the input half band filters must be disabled because of how the USRP hardware was built. To overcome this, a special FPGA image must be loaded by inserting the Code 6.2 into the Device address value.

It was concluded that comparing the two performances, using 8 MHz and 16 MHz bandwidths, the latter had an worse overall performance.

```
fpga=usrp1_fpga_4rx.rbf
```

Code 6.2 - FPGA with disabled half-band filters

The two PLC spectrum acquired with a 16 MHz bandwidth are shown by Figures 6.16,

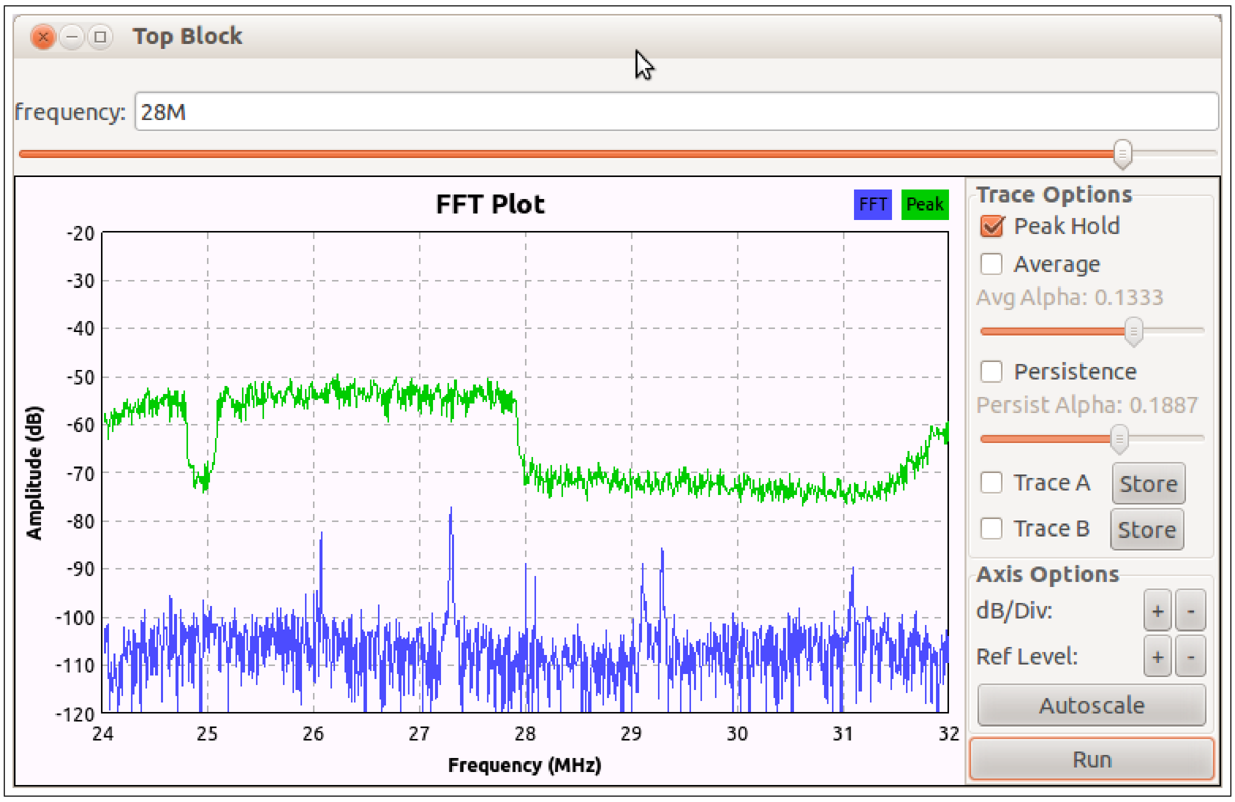


Figure 6.9: PLC FFT Plot from 24 up to 32 MHz

tuned at 8 MHz, and 6.17, tuned at 16 MHz respectively.

Given the two different sample sizes tested, it can be concluded that, although the small size sample was used to reach a wider bandwidth, the practical result, the FFT plot, is not showed in a continuous manner and this could cause failure in a certain application, due to the lack of signal. To avoid such strange signal behaviour, it is highly recommended that the 16-bits complex samples are used, because its FFT plot remains at the screen all the time and it is possible to detect most of the variations in the PLC signal spectrum, like the presence of a certain carrier, the transmitted power in a certain frequency or group of frequency, and so on.

Although neither of these previous approaches are not practical, they are able to show that the technique af acquiring the PLC signal using the USRP module, the GNU Radio and the interface designed is plausible, the whole signal analyser system works well achieving the objective and the bottleneck relies on the interface between the USRP and the Linux PC. Recently, this data rate problem had been tackled through an upgrade on the data interface, going up to 1 Gbps at the most recent USRP model N210, which increases the bandwidth up to 125 M Bps.

The Gigabit Ethernet does not allow this maximum transfer data rate to be reached

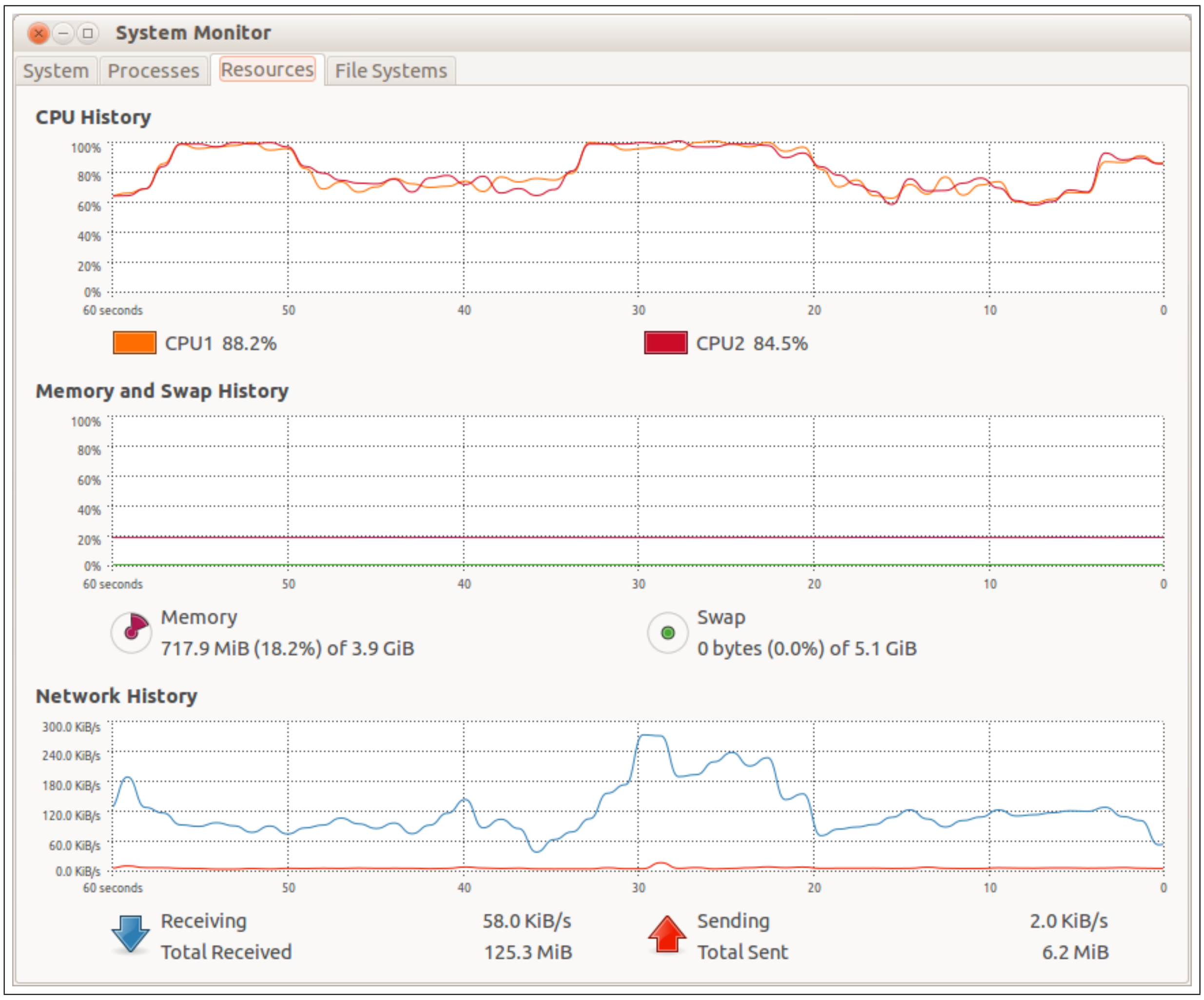


Figure 6.10: Computer performance during spectrum acquisition

due to packet headers, control mechanisms like error detection, coding and acknowledgements at the OSI model network layer. Theoretically, the N210 125 M Bps would result in 30 M complex sps, able to sample a spectrum of 30 MHz which is a little more than the PLC requirement: a sample rate of 26 M complex sps and a data rate of 104 M Bps, considering using only 16-bits complex samples. Therefore the total data rate will be a little greater because of the packet headers and control frames. Some tests need to be performed to measure effective data throughput of the model N210, to be aware of how much of this 1 Gbps can be reached and also, to make all the efforts required to assure that the required data rate is guaranteed everywhere else throughout the USRP module and the computer hardware, like PCI express bus, hard disk access data rate, Ethernet cable and routers, if is the case.

N210 has a huge hardware advantage over the old USRP1 and depending of the application requirements a big PLC spectrum bandwidth can be identified and to fit well

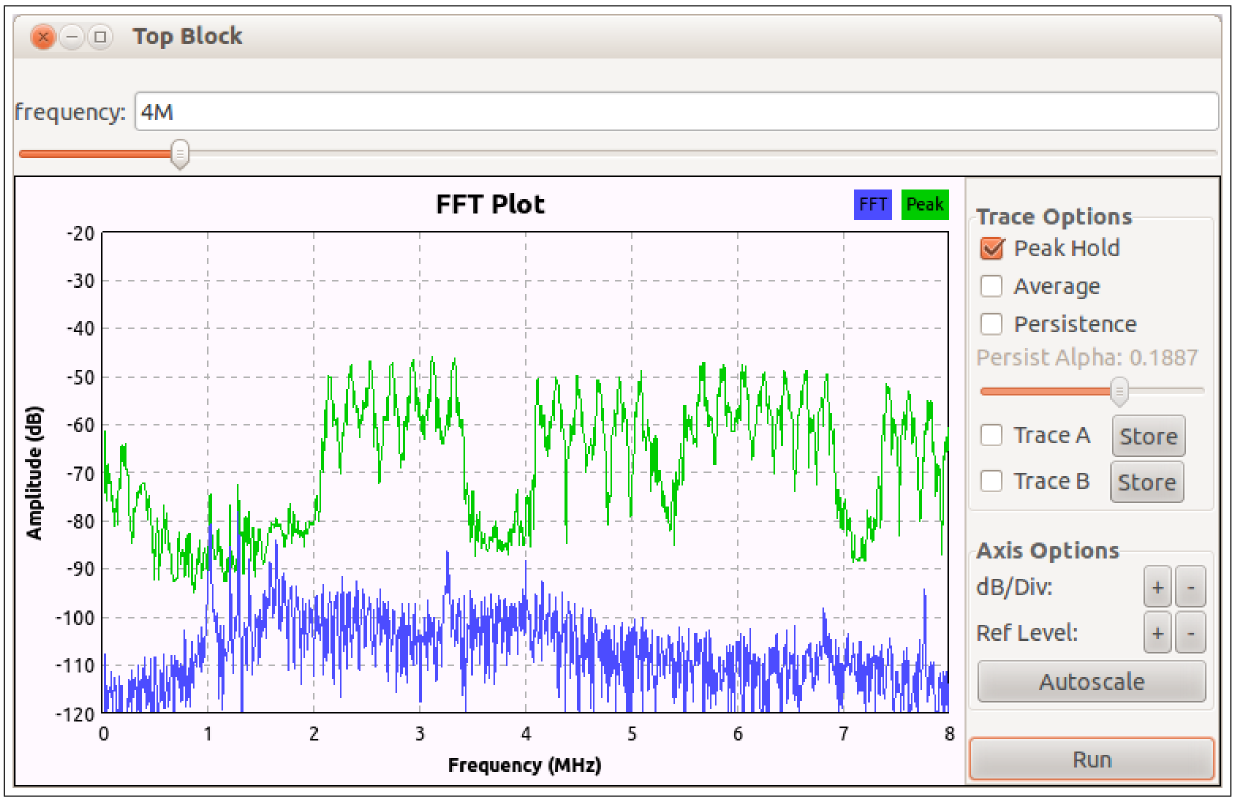


Figure 6.11: PLC spectrum with low carrier density at 4 MHz

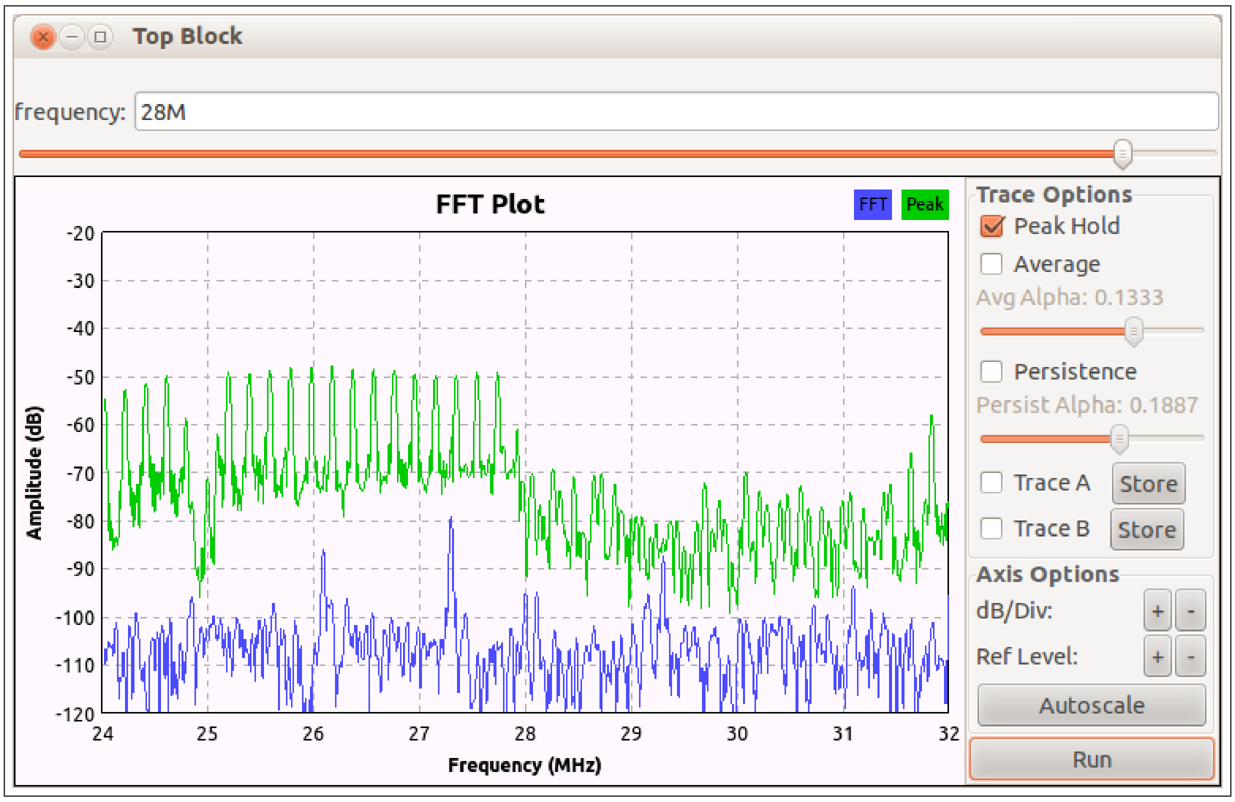
into the bandwidth that the N210 can convey to GNU Radio.

Finally, considering the development of semiconductor technologies, leading to faster, smaller and 'greener' (low power consumption added to higher performance) IC's, chances are that in just a few years there will be USRP models equipped with 4 Gbps or even 10 Gbps Ethernet ports, a lot more of what is really needed today. However, in the near future it will be necessary to solve most of the problems addressed today and the others that may take place, improving not only the PLC signal analyser but communications systems that will demand higher bandwidths.

6.3 Final considerations of this chapter

This chapter described the PLC signal analyser prototype and its main three parts: the PLC-USRP interface, the USRP module and the GNU Radio, which processes the acquired signal. To test this solution, a real PLC network was set up and a live multimedia content was streamed just to generate data traffic. To simplify the processing capability, the real-time FFT of the PLC signals was calculated and showed at the computer screen.

Due to some hardware data rate constraints, the FFT calculation had to be windowed.


Top Block

frequency: 28M

Trace Options
[✓ Peak Hold]
Average
Avg Alpha: 0.1333
Persistence
Persist Alpha: 0.1887
Trace A
Trace B

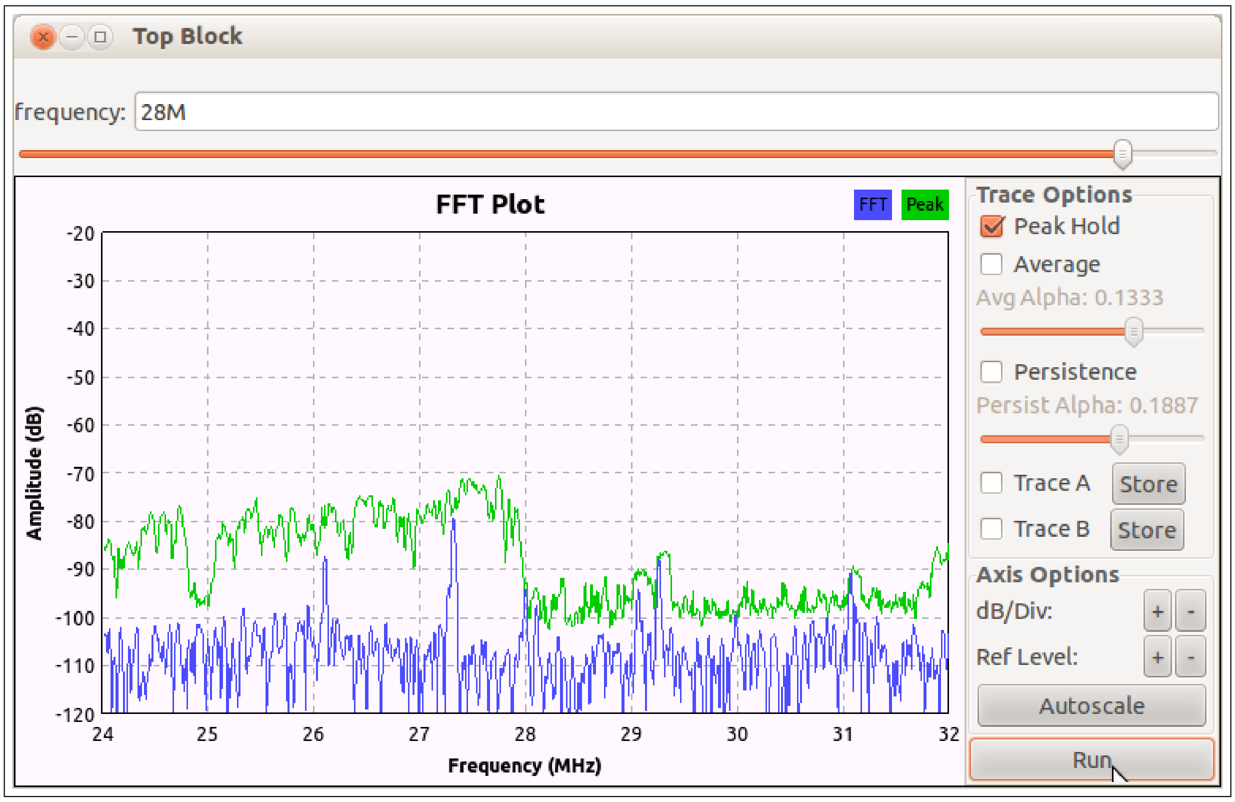
Figure 6.12: PLC spectrum with low carrier density at 28 MHz

The bandwidth limitation was overcame by using a SDR module which has a high data rate throughput like the 1 Gbps from N210 model or equivalent.

The FFT windows could show the desired live PLC FFT and in addition to that some additional information could be extracted from the same window just analysing the spectrum behaviour, for example: the noise floor, the current maximum transmitted power at each sub-carrier, etc.

On future researches based on this signal analyser prototype, key characteristics will be able to be extracted from the current PLC signals being transmitted. Such characteristics may trigger specific functions on a cognitive system making decisions based on sub-carrier detection and power level measurements. These functions could be used for network management and analysis of network quality and security, for example, not only on in-building networks but on access networks too.

The next chapter will make a general conclusion about the prototypes developed to solve the problems initially proposed. In the end it will suggest the future work to be done to continue the research and developments related to this work.


frequency: 28M

Peak Hold [✓]

Average [ ]

Avg Alpha: 0.1333

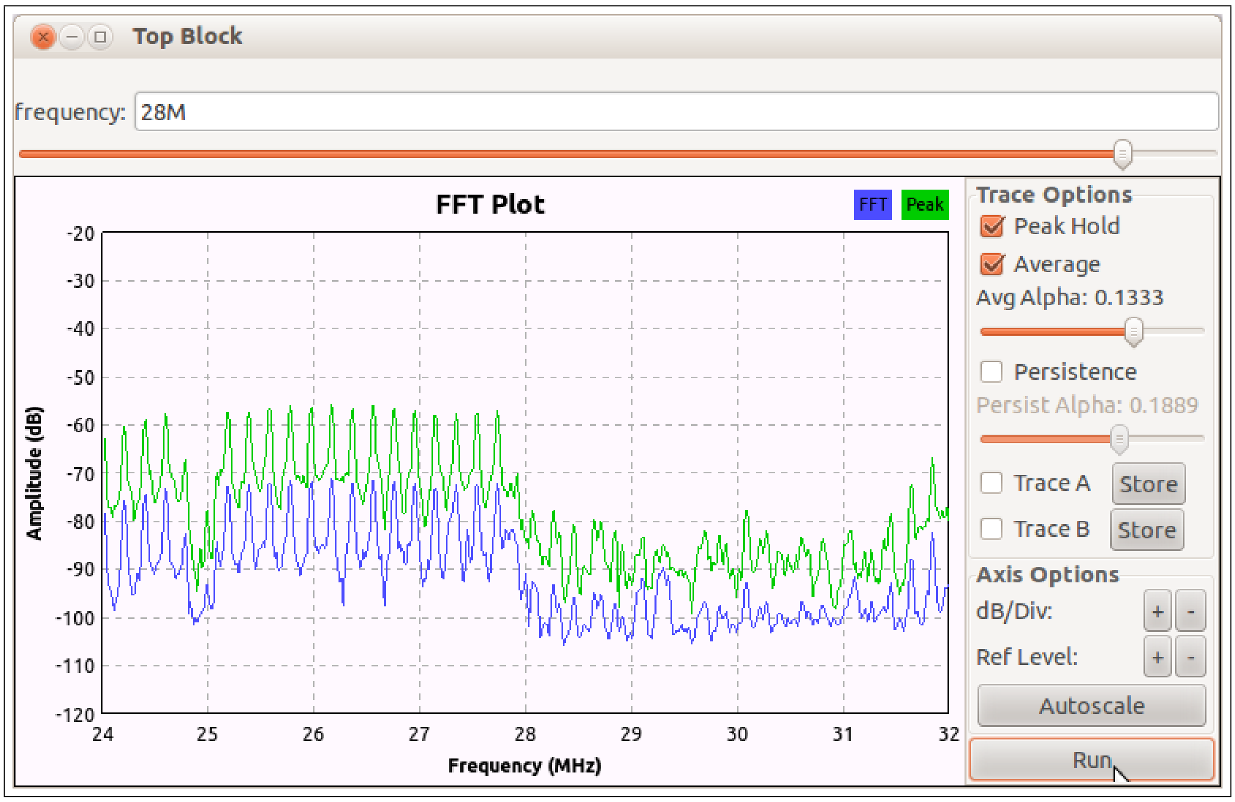
Persistence [ ]

Persist Alpha: 0.1887

Trace A [ ]

Trace B [ ]

Figure 6.13: PLC spectrum with high carrier density and low power at 28 MHz


frequency: 28M

Peak Hold [✓]

Average [✓]

Avg Alpha: 0.1333

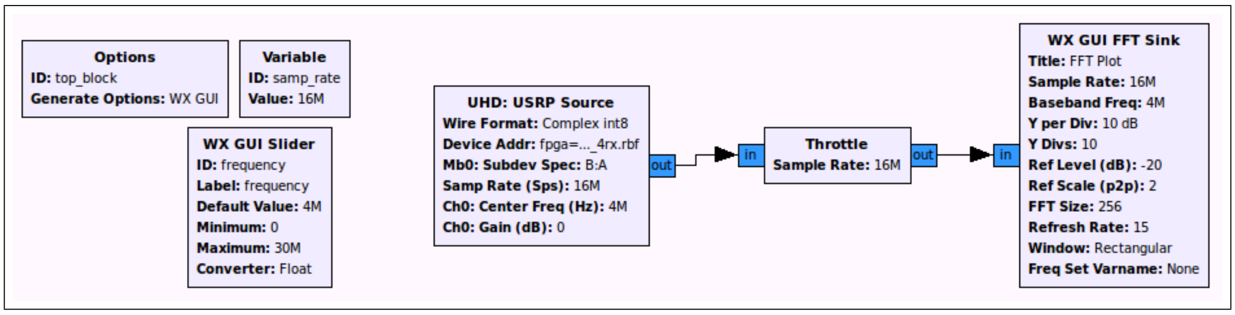
Persistence [ ]

Persist Alpha: 0.1889

Trace A [ ]

Trace B [ ]

Figure 6.14: Average PLC spectrum with low carrier density at 28 MHz



Options
ID: top_block
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 16M

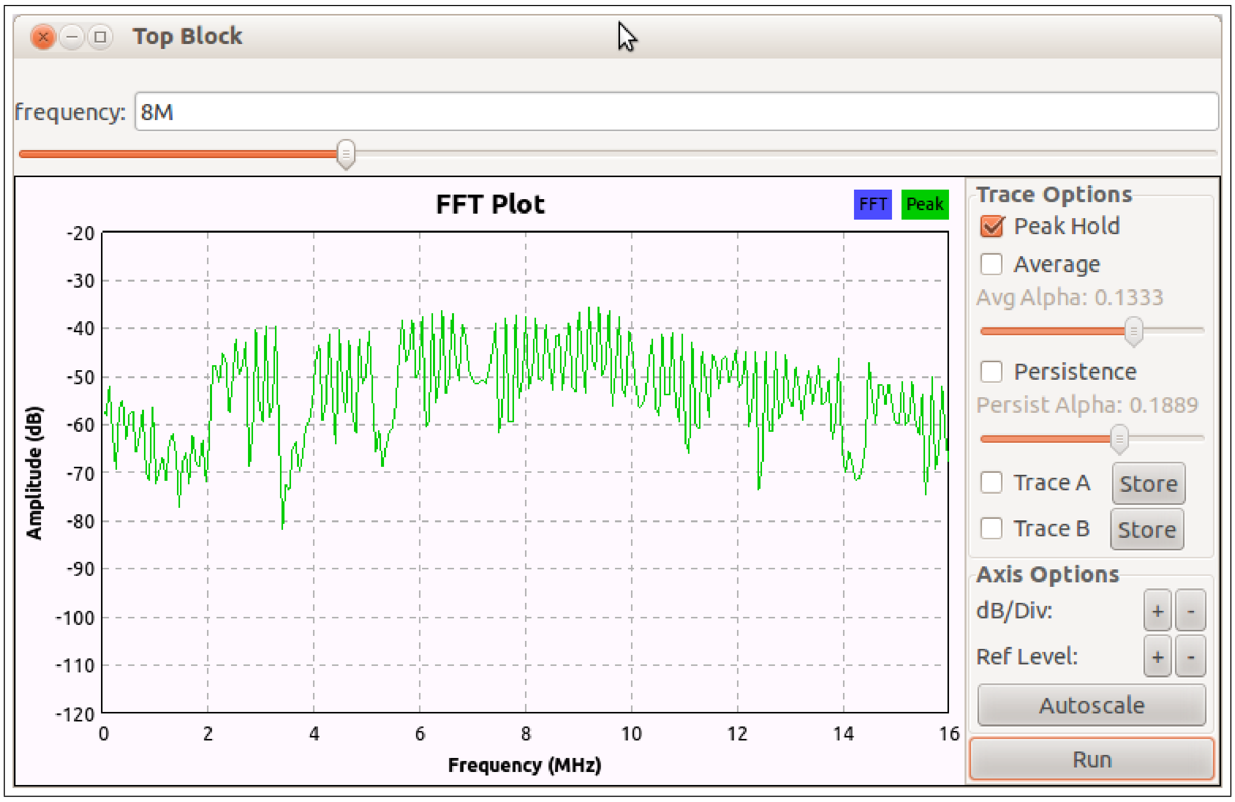
WX GUI Slider
ID: frequency
Label: frequency
Default Value: 4M
Minimum: 0
Maximum: 30M
Converter: Float

UHD: USRP Source
Wire Format: Complex int8
Device Addr: fpga=..._4rx.rbf
Mb0: Subdev Spec: B:A
Samp Rate (Sps): 16M
Ch0: Center Freq (Hz): 4M
Ch0: Gain (dB): 0

Throttle
Sample Rate: 16M

WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 16M
Baseband Freq: 4M
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): -20
Ref Scale (p2p): 2
FFT Size: 256
Refresh Rate: 15
Window: Rectangular
Freq Set Varname: None

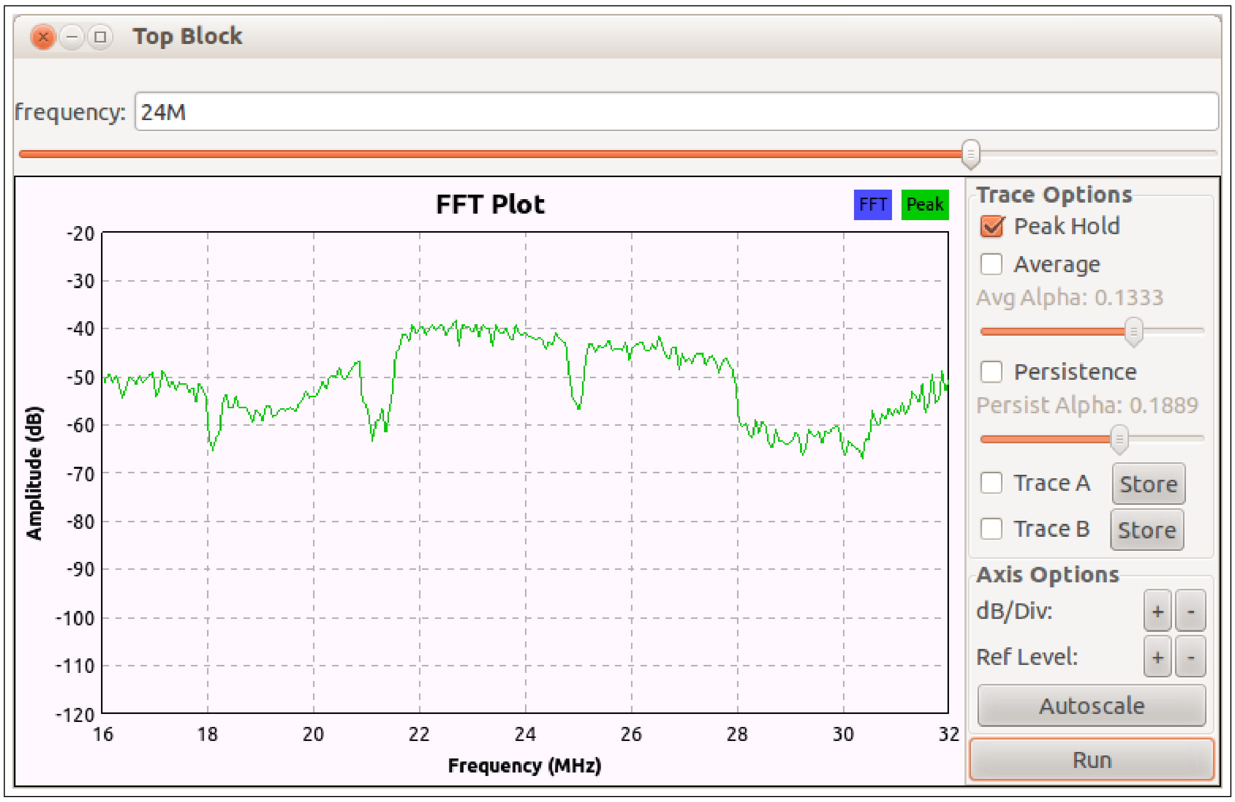
Figure 6.15: GNU Radio receiver for 16 MHz spectrum bandwidth



frequency: 8M

Peak Hold [✓]
Average
Avg Alpha: 0.1333
Persistence
Persist Alpha: 0.1889
Trace A
Trace B

Figure 6.16: PLC spectrum with 16 MHz spectrum bandwidth at 8 MHz



frequency: 24M

[✓ Peak Hold]

Avg Alpha: 0.1333

Persist Alpha: 0.1889

Figure 6.17: PLC spectrum with 16 MHz spectrum bandwidth at 16 MHz

Chapter 7

Conclusions and contributions of this work and Future work

After the successful acquisition of the PLC signal spectrum, it can be concluded that the main objective of this text was achieved.

In the end, two prototypes could be developed: the Automatic Frequency Response software that gets the frequency response amplitude of an electronic device and the PLC signal spectrum analyzer that shows the spectrum of the PLC signal being transmitted through the power lines. Both tools use the same low-cost USRP hardware module and the GNU Radio license-free software.

The AFR prototype development helped to overcome an unexpected problem, the acquisition of a very expensive equipment to acquire the frequency response of an electronic device. This prototype development lead to a low cost solution that was able to meet the current need and to perform the desired task. But only the amplitude acquisition could be performed in an acceptable level, the phase did not, yet. The phase response acquisition will require more effort and time dedicated to it, given the strange behaviour observed during the tests that were made.

This research also contributed for the knowledge of working with the GNU Radio and USRP module for the development of software based communication systems. In addition to that, the USRP module limitations could be known, like its bandwidth of 8 MHz, limited by the USB data rate, limiting the acquisition of the PLC signal wideband spectrum and the shadowed frequency response amplitude of the transceiver system using the BasicTx and BasicRx daughter-boards at some frequency intervals through their theoretical frequency range from 1 MHz up to 250 MHz.

7.1 Future work

Concerning the FRA software, the phase problem need to be tackled in order for the prototype be able to deliver the complete frequency response, magnitude and phase. And the motivation to keep up with working on it is the frequency response amplitude that was already achieved.

Moreover, with the frequency response phase this prototype will be able to be used not only in acquiring electronic device frequency response but also in many different field applications, like finding power transformer faults, determining the acoustical signature of a room or any other environment, and so on.

Another big deal, it to use these two tools for teaching purposes in the academic field, allowing quick frequency response acquisitions, spectrum analysis, development and optimization of SDR communication systems, etc.

Finally, the PLC spectrum analyser prototype needs to be tested with the ultimate USRP module on the market, in order to allow the acquisition of a wideband spectrum as the HomePlug AV one and the others that may come.

In parallel with the tests using a more robust hardware module, a research to develop an algorithm for PLC signal processing might take place. This algorithm may use some kind of 'cognitive intelligence' over the PLC spectrum acquired by the PLC spectrum analyser prototype, that provides a digital signal ready to be processed by the GNU Radio. Such PLC signal processing might allow the network quality analysis, security monitoring and enable that more complex research regarding the PLC signal transmission and reception can be performed in an easy and quick way leading to an improvement of the technology and the development of more advanced solutions based on power line communications.

Bibliography

[1] G3-PLC Alliance, May 2012. http://www.g3-plc.com/.

[2] High Definition Power Line Communication Alliance, June 2012. http://www.hd-plc.org/modules/about/plc.html.

[3] Alessandro, S., Tonello, A. M. Performance report of optimized PHY algorithms - ICT-213311, April 2011. http://www.ict-omega.eu.

[4] Analog Devices. AD9862 - Mixed-Signal Front-End Processor for Broadband Communications, September 2002. http://www.analog.com.

[5] Bel Fuse Inc. Powerline Signal Couplers, August 2009. http://belfuse.com.

[6] Blossom, E., Corgan, J., Ettus, M., Rondeau, T. GNU Radio, December 2009. http://gnuradio.org.

[7] Blossom, E., Corgan, J., Ettus, M., Rondeau, T. Download GNU Radio, May 2012. http://gnuradio.org/releases/gnuradio.

[8] Blossom, E., Corgan, J., Ettus, M., Rondeau, T. Install GNU Radio, May 2012. http://gnuradio.org/redmine/projects/gnuradio/wiki/InstallingGR.

[9] Carcelle, X. *Power Line Communications in practice.* Artech House, 2006.

[10] Dimopoulos, H.,G. *Analog Electronic Filters - Theory, Design and Synthesis.* Springer, 1st edition, 2012.

[11] Dostert, K. *Powerline Communications.* Prentice Hall, 2001.

[12] Ferreira, P. V. R., Diniz, P. C. A., Veiga, A. C. P., Carneiro, M. B. C. Frequency Response Acquisition of a Digital Radio Transceiver Using USRP Module and GNU Radio Software. In *Computational Intelligence, Modelling and Simulation (CIMSiM), 2012 Fourth International Conference on.* http://ieeexplore.ieee.org.

[13] Hamza, F. A. The USRP under 1.5x Magnifying Lens, June 2008. http://gnuradio.org/redmine/attachments/download/129.

[14] HomePlug Alliance. HomePlug Powerline Alliance, May 2012. https://www.homeplug.org/home.

[15] Hrasnica, H., Haidine, A., Lehnert, R. *Broadband Powerline Communications Networks Network Design.* John Wiley and Sons, 2004.

[16] IEEE. IEEE Draft Standard for Broadband over Power Line Networks: Medium Access Control and Physical Layer Specifications. *IEEE P1901/D4.01, July 2010*, pages 1–1589, December 2010.

[17] International Electrotechnical Commission. INTERNATIONAL STANDARD IEC 61000-4-2:Electromagnetic compatibility (EMC)-Electrostatic discharge immunity test. http://webstore.iec.ch.

[18] International Electrotechnical Commission. INTERNATIONAL STANDARD IEC 60384-14: Fixed capacitors for use in electronic equipment, July 2005. http://webstore.iec.ch.

[19] International Electrotechnical Commission. INTERNATIONAL STANDARD IEC 61051-1: Varistors for use in electronic equipment, April 2007. http://webstore.iec.ch.

[20] International Special Committee on Radio Interference. IEC S+ CISPR 22: Information technology equipment, radio disturbance characteristics, limits and methods of measurement, September 2009. http://webstore.iec.ch/.

[21] Klaus, D. New PLC Approaches for High Speed Indoor Digital Networks. *International Symposium on Power Line Communications and Its Applications (ISPLC)*, pages 253–258, 2001.

[22] Lathi, B. P. *Modern Digital and Analog Communication Systems.* Oxford University Press, 3rd edition, 1998.

[23] Mitola, J. The Software Radio Architecture. *IEEE Communications Magazine*, pages 26–38, May 1995.

[24] Panasonic. ZNR Transient/Surge Absorbers, February 2006. http://industrial.panasonic.com.

[25] Proakins, J. C., Salehi, M. *Digital Communications.* McGraw Hill, 5th edition, 2008.

[26] Qualcomm Atheros. INT1400 AFE/Line Driver IC and INT6400 MAC/PHY Transceiver IC, October 2010. http://www.qca.qualcomm.com.

[27] Ettus Research. Ettus research (webpage), June 2012. www.ettus.com.

[28] Telecommunication Standardization Sector of International Telecommunication Union. ITU-T G.9964: Unified high-speed wireline-based home networking transceivers: Power spectral density specification, December 2011. http://www.itu.int.

[29] TP-Link. Tp-link (webpage), June 2012. http://www.tp-link.com.

[30] Tuttlebee, W. *Software Defined Radio - Origins, Drivers and International Perspectives*, volume Vol.I. John Wiley & Sons, Ltd, 2001.

[31] United Micro Device Inc. Single line TVS diode for ESD protection, July 2010. http://www.umdcorp.com.

[32] Vasseur, J. P., Dunkels, A. *Interconnecting Smart Objects With IP: The Next Internet.* Morgan Kaufmann, 2010.

[33] Winder, S. *Analog and Digital Filter Design.* Newnes, 2nd edition, 2002.

Appendix A

Automatic Frequency Response Prototype Software

This Appendix presents the Python source code used for prototyping the AFR software used on the automatic method for the frequency response amplitude acquisition of the four different transceiver configurations using the BasicRx, BasicTx, LFRX and LFTX daughter-boards, the USRP1 module and the GNU Radio software shown in Chapter 4. This prototype implements the complete system including the graphical interface, and is optimized for the acquisition of the responses of the two USRP receiver sides at the same time, saving both the amplitudes and the phases. Moreover, if the user wants to customize the processing block itself, the changes shall be made into each processing block file separately using C++.

In the beginning of each main function there is some comments about the specific action of that part of the code. The code must be run from the Terminal. In the end of the execution the plotting occurs automatically.

A.1 AFR Source Code

The Automatic Frequency Response Prototype Python Source Code

```
#! /usr/bin/env python
#To make programs executable for Linux 1st line of program and change permissions to include execute

#All the libraries need for program execution
import wx
from gnuradio import gr, eng_notation
from gnuradio.gr.pubsub import pubsub
from gnuradio.eng_option import eng_option
from optparse import OptionParser
from gnuradio.gr import firdes
import sys
import math
import numpy as np
import Gnuplot, Gnuplot.PlotItems, Gnuplot.funcutils
from grc_gnuradio import usrp as grc_usrp

# GUI-unaware GNU Radio flowgraph.  This may be used either with command line applications or GUI applications.

#Variables for graphical interface events like pushing buttons, updating variables values
ID_ABOUT = wx.NewId()
ID_EXIT = wx.NewId()
ID_STARTFREQ = wx.NewId()
ID_STEPFREQ = wx.NewId()
ID_STOPFREQ = wx.NewId()
ID_STEPTIME = wx.NewId()
ID_AMPLITUDE = wx.NewId()
ID_START = wx.NewId()
ID_STOP = wx.NewId()

#Defines all the graphical interface parameters, valriables and structure
```

```
class MyFrame(wx.Frame):
    def __init__(self, *args, **kwds):

        # begin wxGlade: Start the graphical interface structure building
        kwds["style"] = wx.DEFAULT_FRAME_STYLE
        wx.Frame.__init__(self, *args, **kwds)

        # Menu Bar
        self.frame_1_menubar = wx.MenuBar()
        self.SetMenuBar(self.frame_1_menubar)
        wxglade_tmp_menu = wx.Menu()
        self.About = wx.MenuItem(wxglade_tmp_menu, ID_ABOUT, "About", "About", wx.ITEM_NORMAL)
        wxglade_tmp_menu.AppendItem(self.About)
        self.Exit = wx.MenuItem(wxglade_tmp_menu, ID_EXIT, "Exit", "Exit", wx.ITEM_NORMAL)
        wxglade_tmp_menu.AppendItem(self.Exit)
        # Menu Bar end

        #Graphical interface textboxes titles, button names and related events
        self.label_5 = wx.StaticText(self, -1, "  Initial Frequency (Hz): ")
        self.text_ctrl_4 = wx.TextCtrl(self, ID_STARTFREQ, "", style=wx.TE_PROCESS_ENTER|wx.TE_PROCESS_TAB)
        self.label_8 = wx.StaticText(self, -1, "  Frequency Resolution (Hz): ")
        self.text_ctrl_7 = wx.TextCtrl(self, ID_STEPFREQ, "", style=wx.TE_PROCESS_ENTER|wx.TE_PROCESS_TAB)
        self.label_6 = wx.StaticText(self, -1, "  Final Frequency (Hz): ")
        self.text_ctrl_5 = wx.TextCtrl(self, ID_STOPFREQ, "", style=wx.TE_PROCESS_ENTER|wx.TE_PROCESS_TAB)
        self.label_9 = wx.StaticText(self, -1, "  Time Interval (ms): ")
        self.text_ctrl_8 = wx.TextCtrl(self, ID_STEPTIME, "", style=wx.TE_PROCESS_ENTER|wx.TE_PROCESS_TAB)
        self.label_7 = wx.StaticText(self, -1, "  Amplitude: ")
        self.text_ctrl_6 = wx.TextCtrl(self, ID_AMPLITUDE, "", style=wx.TE_PROCESS_ENTER|wx.TE_PROCESS_TAB)
        self.panel_1 = wx.Panel(self, -1)
        self.panel_2 = wx.Panel(self, -1)
        self.button_1 = wx.Button(self, ID_START, "Start")
        self.button_2 = wx.Button(self, ID_STOP, "Stop")
        self.label_1 = wx.StaticText(self, -1, "     Status:    ")
        self.label_2 = wx.StaticText(self, -1, "\nCurrent Frequency (Hz):\n")
        self.text_ctrl_1 = wx.TextCtrl(self, -1, "", style=wx.TE_READONLY)
        self.label_3 = wx.StaticText(self, -1, "\nAmplitude Side_A:\n")
        self.text_ctrl_2 = wx.TextCtrl(self, -1, "", style=wx.TE_READONLY)

        self.label_10 = wx.StaticText(self, -1, "\nAmplitude Side_B:\n")
        self.text_ctrl_10 = wx.TextCtrl(self, -1, "", style=wx.TE_READONLY)

        self.label_12 = wx.StaticText(self, -1, "\nSample Rate (sample/second):\n")
        self.text_ctrl_12 = wx.TextCtrl(self, -1, "", style=wx.TE_READONLY)

        self.__set_properties()
        self.__do_layout()
        # end wxGlade

        wx.EVT_MENU(self, ID_EXIT, self.MenuExit)
        wx.EVT_MENU(self, ID_ABOUT, self.MenuAbout)
        wx.EVT_TEXT_ENTER(self, ID_STARTFREQ, self.SetStartFreq)
        wx.EVT_TEXT_ENTER(self, ID_STOPFREQ, self.SetStopFreq)
        wx.EVT_TEXT_ENTER(self, ID_STEPFREQ, self.SetStepFreq)
        wx.EVT_TEXT_ENTER(self, ID_STEPTIME, self.SetStepTime)
        wx.EVT_TEXT_ENTER(self, ID_AMPLITUDE, self.SetAmplitude)
        wx.EVT_BUTTON(self, ID_START, self.Start)
        wx.EVT_BUTTON(self, ID_STOP, self.Stop)

        #If the user wants to run the program into the Terminal, the following parameters set the variables.
        #When using the graphical interface the standard values are already set.
        parser = OptionParser (option_class=eng_option)

        parser.add_option ("-f", "--freq", type="eng_float", default=1e6,
                        help="set waveform frequency")

        parser.add_option ("-a", "--amplitude", type="int", default=32000,
                        help="set waveform amplitude", metavar="AMPL")

        parser.add_option ("-o", "--offset", type="eng_float", default=0,
                        help="set waveform offset")

        #DUC frequency used only to correct the frequency difference between the IF target value and
        #the initially frequency value set (the first attempt)
        parser.add_option ("-c", "--duc-freq", type="eng_float", default=0,
                        help="set Tx DUC frequency to 'freq' value", metavar="FREQ")

        #The default channel number inside the USRP is initially set to 1
        parser.add_option ("-n", "--nchannels", type="int", default=1,
                        help="set number of output channels to NCHANNELS")

        #The ADC to DDC routing, this set to the standard route (Input 1 (RXA) to I0 and Input 2 (RXB) to Q0)
        parser.add_option ("-m", "--mux", type="intx", default=0x98,
                        help="set output mux register")

        (options, args) = parser.parse_args ()

        #Set by heriting Terminal values to graphical variables values
        self.run = False
        self.rpt_open = False
        self.mux = options.mux
        self.nchannels = options.nchannels
        self.duc_freq = 0
        self.offset = options.offset
        self.freq = options.freq
        self.text_ctrl_4.SetValue(eng_notation.num_to_str(self.freq))
        self.amplitude = options.amplitude
        self.text_ctrl_6.SetValue(eng_notation.num_to_str(self.amplitude))

        #Variables initially set, but can be changed by the graphical interface
        self.stop_freq = 30e6
        self.text_ctrl_5.SetValue(eng_notation.num_to_str(self.stop_freq))
        self.step_freq = 5e3
        self.text_ctrl_7.SetValue(eng_notation.num_to_str(self.step_freq))
        self.step_time = 25
        self.text_ctrl_8.SetValue(eng_notation.num_to_str(self.step_time))
        self.timer = UpdateTimer(self,self.step_time)
        self.samp_rate = 250e3

    def __set_properties(self):
        # begin wxGlade: title properties/name
        self.SetTitle("Automatic Frequency Response")
        # end wxGlade

    def __do_layout(self):
        # begin wxGlade: graphical interface layout structure
        sizer_1 = wx.BoxSizer(wx.VERTICAL)
        sizer_2 = wx.BoxSizer(wx.VERTICAL)
        sizer_3 = wx.BoxSizer(wx.HORIZONTAL)
        sizer_5 = wx.BoxSizer(wx.VERTICAL)
        sizer_4 = wx.BoxSizer(wx.VERTICAL)
        sizer_6 = wx.BoxSizer(wx.HORIZONTAL)
        sizer_7 = wx.BoxSizer(wx.VERTICAL)
        sizer_8 = wx.BoxSizer(wx.HORIZONTAL)
        grid_sizer_1 = wx.FlexGridSizer(0, 2, 5, 5)
        #Initial Frequency
        grid_sizer_1.Add(self.label_5, 0, wx.ALIGN_CENTER_VERTICAL|wx.FIXED_MINSIZE, 0)
        grid_sizer_1.Add(self.text_ctrl_4, 0, wx.ALIGN_CENTER_VERTICAL|wx.FIXED_MINSIZE, 0)
        #Final Frequency
        grid_sizer_1.Add(self.label_6, 0, wx.ALIGN_CENTER_VERTICAL|wx.FIXED_MINSIZE, 0)
        grid_sizer_1.Add(self.text_ctrl_5, 0, wx.ALIGN_CENTER_VERTICAL|wx.FIXED_MINSIZE, 0)
        #Frequency Resolution
        grid_sizer_1.Add(self.label_8, 0, wx.ALIGN_CENTER_VERTICAL|wx.FIXED_MINSIZE, 0)
        grid_sizer_1.Add(self.text_ctrl_7, 0, wx.ALIGN_CENTER_VERTICAL|wx.FIXED_MINSIZE, 0)
        #Amplitude
        grid_sizer_1.Add(self.label_7, 0, wx.ALIGN_CENTER_VERTICAL|wx.FIXED_MINSIZE, 0)
        grid_sizer_1.Add(self.text_ctrl_6, 0, wx.ALIGN_CENTER_VERTICAL|wx.FIXED_MINSIZE, 0)
        #Time Interval
        grid_sizer_1.Add(self.label_9, 0, wx.ALIGN_CENTER_VERTICAL|wx.FIXED_MINSIZE, 0)
        grid_sizer_1.Add(self.text_ctrl_8, 0, wx.ALIGN_CENTER_VERTICAL|wx.FIXED_MINSIZE, 0)


        grid_sizer_1.Add(self.panel_1, 1, wx.EXPAND, 0)
        grid_sizer_1.Add(self.panel_2, 1, wx.EXPAND, 0)
        sizer_7.Add(grid_sizer_1, 1, wx.EXPAND, 0)
        sizer_7.Add(sizer_8, 0.5, wx.EXPAND, 0.5)
        sizer_2.Add(sizer_7, 0.5, wx.EXPAND, 0.5)
        sizer_6.Add(self.button_1, 1, wx.ALIGN_CENTER_HORIZONTAL|wx.FIXED_MINSIZE, 0)
        sizer_6.Add(self.button_2, 1, wx.ALIGN_CENTER_HORIZONTAL|wx.FIXED_MINSIZE, 0)
        sizer_2.Add(sizer_6, 0, wx.ALIGN_CENTER_HORIZONTAL, 0)
        sizer_3.Add(self.label_1, 0, wx.FIXED_MINSIZE, 0)
        sizer_4.Add(self.label_2, 0, wx.ALIGN_CENTER_HORIZONTAL|wx.FIXED_MINSIZE, 0)
        sizer_4.Add(self.text_ctrl_1, 0, wx.ALIGN_CENTER_HORIZONTAL|wx.FIXED_MINSIZE, 0)
        sizer_4.Add(self.label_12, 0, wx.ALIGN_CENTER_HORIZONTAL|wx.FIXED_MINSIZE, 0)
        sizer_4.Add(self.text_ctrl_12, 0, wx.ALIGN_CENTER_HORIZONTAL|wx.FIXED_MINSIZE, 0)
        sizer_3.Add(sizer_4, 1, wx.EXPAND, 0)
        sizer_5.Add(self.label_3, 0, wx.ALIGN_CENTER_HORIZONTAL|wx.FIXED_MINSIZE, 0)
        sizer_5.Add(self.text_ctrl_2, 0, wx.ALIGN_CENTER_HORIZONTAL|wx.FIXED_MINSIZE, 0)
        sizer_5.Add(self.label_10, 0, wx.ALIGN_CENTER_HORIZONTAL|wx.FIXED_MINSIZE, 0)
        sizer_5.Add(self.text_ctrl_10, 0, wx.ALIGN_CENTER_HORIZONTAL|wx.FIXED_MINSIZE, 0)

        sizer_3.Add(sizer_5, 1, wx.EXPAND, 0)
        sizer_2.Add(sizer_3, 1, wx.EXPAND, 0)
        sizer_1.Add(sizer_2, 1, wx.EXPAND, 0)
        self.SetAutoLayout(True)
        self.SetSizer(sizer_1)
        sizer_1.Fit(self)
        sizer_1.SetSizeHints(self)
        self.Layout()
        # end wxGlade


    #Graphical Interface MENUS
    def MenuExit(self, event):
        if self.run:
           self.rx_fg.stop()
        if self.rpt_open:
           self.rpt.close()
           self.Close()

    def MenuAbout(self, event):
        dlg = wx.MessageDialog(self, "Automatic Sweep\n"
                                     "for USRP1 and GNU Radio\n",
                              "About", wx.OK | wx.ICON_INFORMATION)
        dlg.ShowModal()
        dlg.Destroy()


        #When the there are no Terminal values when the program is called, inherit values from
        #the text boxes from the graphical interface:
    def SetStartFreq(self,event):
        self.freq = eng_notation.str_to_num(self.text_ctrl_4.GetValue())

    def SetStopFreq(self,event):
        self.stop_freq = eng_notation.str_to_num(self.text_ctrl_5.GetValue())

    def SetStepFreq(self,event):
        self.step_freq = eng_notation.str_to_num(self.text_ctrl_7.GetValue())

    def SetStepTime(self,event):
        self.step_time = eng_notation.str_to_num(self.text_ctrl_8.GetValue())
        self.timer = UpdateTimer(self,self.step_time)

    def SetAmplitude(self,event):
        self.amplitude = eng_notation.str_to_num(self.text_ctrl_6.GetValue())


    def Start(self,event):
        if self.run == False:

        #Building the GNU Radio communication system

        #---->> RECEIVER SIDE <<----#

          #Make the GNU Radio Receiver Flow Graph
          self.rx_fg = gr.top_block()

          #Sets the receiver sample rate
          samp_rate = self.samp_rate

          #In order to make 2 tests at the same time using the sweep of only one transmitter the
          #USRP Dual Source is configured at the receiver side. One receiver daughter-board is in
          #Side A and the other is in Side B. On the graphical interface there is no indication that
          #this is done, but the plotting of this information is done. However, if the used wants to
          #use only one receiver side at time, it shall be use the USRP Source.

          #USRP Source / Receiver input interface of GNU Radio with USRP
          self.rx = grc_usrp.dual_source_c(
                which=0,
                rx_ant_a="RXA", rx_ant_b="RXA",
                rx_source_a="A", rx_source_b="B",
          )
          self.rx.set_decim_rate(256)
          self.rx.set_frequency_a(self.freq+self.offset, verbose=True)
          self.rx.set_frequency_b(self.freq+self.offset, verbose=True)
          self.rx.set_gain_a(20)
          self.rx.set_gain_b(20)

          #Low-pass filters with curoff frequencies and window widths set
          self.low_pass_filter_0 = gr.fir_filter_ccf(1, firdes.low_pass(
                      1, samp_rate, 5e3, 2e3, firdes.WIN_HAMMING, 6.76))
          self.low_pass_filter_0_0 = gr.fir_filter_ccf(1, firdes.low_pass(
                      1, samp_rate, 5e3, 2e3, firdes.WIN_HAMMING, 6.76))

          #Signal splitters: from complex into real and imaginary parts
          self.gr_complex_to_real_0 = gr.complex_to_real(1)
          self.gr_complex_to_real_0_1 = gr.complex_to_real(1)

          self.gr_complex_to_imag_0 = gr.complex_to_imag(1)
          self.gr_complex_to_imag_0_1 = gr.complex_to_imag(1)

          #Pairs peak detectors and sample&hold blocks
          self.gr_peak_detector_xb_0 = gr.peak_detector_fb(1, 1, 1, 1)
          self.gr_peak_detector_xb_0_0 = gr.peak_detector_fb(1, 1, 1, 1)
          self.gr_sample_and_hold_xx_0 = gr.sample_and_hold_ff()
          self.gr_sample_and_hold_xx_0_0 = gr.sample_and_hold_ff()

          #Like the two daughter-boards are being received at the same time it is possible to get the
          #difference between the two received signals too.
          self.gr_sub_xx_0 = gr.sub_ff(1)

          # Signal Prob_0 = measures the received signal by BasicRx. At each test it was used one different
          #transmitter board. Calculates the Magnitude squared. Later its square root will be computed.
          self.gr_probe_avg_mag_sqrd_x_0 = gr.probe_avg_mag_sqrd_f(0, 1)

          # Signal Prob_1 = measures the received signal by LFRX. At each test it was used one different
          #transmitter board. Calculates the Magnitude squared. Later its square root will be computed.
          self.gr_probe_avg_mag_sqrd_x_1 = gr.probe_avg_mag_sqrd_f(0, 1)

          # Signal Prob_2 = measures the difference between the two received signals by both BasicRx and LFRX.
          #Calculates the Magnitude squared. Later its square root will be computed.
          self.gr_probe_avg_mag_sqrd_x_2 = gr.probe_avg_mag_sqrd_f(0, 1)

          #The following phase code measurement was left just for future reasons in case anyone else wants to help
```

```
      #solving the problems found related in estimating the phase difference between RXA and RXB signals

      ############################################
      #Phase estimation
      #Source Rx Channel A probes
      self.gr_probe_sourcea_real = gr.probe_signal_f()
      self.gr_probe_sourcea_img = gr.probe_signal_f()
      #Source Rx Channel B probes
      self.gr_probe_sourceb_real = gr.probe_signal_f()
      self.gr_probe_sourceb_img = gr.probe_signal_f()
      ###############################################

      #Receiver processing blocks connections

      # Source Rx Side B
      self.rx_fg.connect((self.rx, 1), (self.low_pass_filter_0_0, 0))

      self.rx_fg.connect((self.low_pass_filter_0_0, 0), (self.gr_complex_to_real_0_1, 0))
      self.rx_fg.connect((self.low_pass_filter_0_0, 0), (self.gr_complex_to_imag_0_1, 0))
      self.rx_fg.connect((self.gr_complex_to_real_0_1, 0), (self.gr_probe_sourceb_real))
      self.rx_fg.connect((self.gr_complex_to_imag_0_1, 0), (self.gr_probe_sourceb_img))
      self.rx_fg.connect((self.gr_peak_detector_xb_0_0, 0), (self.gr_sample_and_hold_xx_0_0, 1))
      self.rx_fg.connect((self.gr_complex_to_real_0_1, 0), (self.gr_peak_detector_xb_0_0, 0))
      self.rx_fg.connect((self.gr_complex_to_real_0_1, 0), (self.gr_sample_and_hold_xx_0_0, 0))

      # Source Rx Side A
      self.rx_fg.connect((self.rx, 0), (self.low_pass_filter_0, 0))

      self.rx_fg.connect((self.low_pass_filter_0, 0), (self.gr_complex_to_real_0, 0))
      self.rx_fg.connect((self.low_pass_filter_0, 0), (self.gr_complex_to_imag_0, 0))
      self.rx_fg.connect((self.gr_complex_to_real_0, 0), (self.gr_probe_sourcea_real))
      self.rx_fg.connect((self.gr_complex_to_imag_0, 0), (self.gr_probe_sourcea_img))


      self.rx_fg.connect((self.gr_complex_to_real_0, 0), (self.gr_sample_and_hold_xx_0, 0))
      self.rx_fg.connect((self.gr_complex_to_real_0, 0), (self.gr_peak_detector_xb_0, 0))
      self.rx_fg.connect((self.gr_peak_detector_xb_0, 0), (self.gr_sample_and_hold_xx_0, 1))

      self.rx_fg.connect((self.gr_sample_and_hold_xx_0, 0), (self.gr_sub_xx_0, 1))
      self.rx_fg.connect((self.gr_sample_and_hold_xx_0_0, 0), (self.gr_sub_xx_0, 0))

      self.rx_fg.connect((self.gr_sub_xx_0, 0), (self.gr_probe_avg_mag_sqrd_x_2))
      self.rx_fg.connect((self.gr_sample_and_hold_xx_0_0, 0), (self.gr_probe_avg_mag_sqrd_x_0))
      self.rx_fg.connect((self.gr_sample_and_hold_xx_0, 0), (self.gr_probe_avg_mag_sqrd_x_1))


      #Starts the receiver connections and receiver flow graph
      self.rx_fg.start()

      #---->> TRANSMITTER SIDE <<----#

      #In case the UHD driver is used:
      #UHD device address args
      #args = 'serial=4925dd04'

      #Make the GNU Radio Transmitter Flow Graph
      self.sg_fg = gr.top_block()

      #Sets the transmitter sample rate
      samp_rate = self.samp_rate

      #USRP Sink / Transmitter output interface of GNU Radio with USRP
      self._src = gr.sig_source_c(samp_rate, gr.GR_COS_WAVE, 1000, 32000, 0)
      self.sg = grc_usrp.simple_sink_c(which=0, side="B")
      self.sg.set_interp_rate(512)
      self.sg.set_frequency(self.freq, verbose=True)
      self.sg.set_gain(20)
      self.sg.set_enable(True)

      #Sets the maximum sweep frequency
      stop_freq = self.stop_freq

    #Signal splitters: from complex into real and imaginary parts
    self.gr_complex_to_real_0_2 = gr.complex_to_real(1)
    self.gr_complex_to_imag_0_2 = gr.complex_to_imag(1)

    #Phase probes
    #Measuring the splitted parts
    self.gr_probe_sink_real = gr.probe_signal_f()
    self.gr_probe_sink_img = gr.probe_signal_f()


    #Transmitter processing blocks connections
    self.sg_fg.connect((self._src, 0), (self.sg, 0))
    self.sg_fg.connect((self._src, 0), (self.gr_complex_to_real_0_2, 0))
    self.sg_fg.connect((self._src, 0), (self.gr_complex_to_imag_0_2, 0))

    self.sg_fg.connect((self.gr_complex_to_real_0_2, 0),(self.gr_probe_sink_real))
    self.sg_fg.connect((self.gr_complex_to_imag_0_2, 0),(self.gr_probe_sink_img))

    #Starts the transmitter connections and transmitter flow graph
```

```
            self.sg_fg.start()
            self.run = True

        #Stop actions when sweep finishes
        def Stop(self,event):
            if self.run == True:
              self.rx_fg.stop()
              self.sg_fg.stop()
              del (self.sg,self.rx,self.rx_fg,self.sg_fg)
              self.run = False

        #When sweep stops, in Terminal still waits for the user to press any key
        def wait(self, str=None, prompt='Press return to exit...\n'):
            if str is not None:
              print str
            raw_input(prompt)

        #Actions taken at each new frequency set
        def OnUpdate(self):
            if self.run:

              #Computing the probes square root to give the final signal Magnitude
              self.a = math.sqrt(self.gr_probe_avg_mag_sqrd_x_0.level())
              self.b = math.sqrt(self.gr_probe_avg_mag_sqrd_x_1.level())
              self.c = math.sqrt(self.gr_probe_avg_mag_sqrd_x_2.level())

              #Creates the BasicRx1.txt file with the condition to apend information, writting the current frequency
              #and the  measured magnitude
              with open("BasicRx1.txt","a") as rpt:
                            rpt.write(str(self.freq)+" "+str(self.a)+"\n")
              #Creates the LFRX1.txt file with the condition to apend information, writting the current frequency and
              #the measured magnitude
              with open("LFRX1.txt","a")    as rpt:
                            rpt.write(str(self.freq)+" "+str(self.b)+"\n")
              #Creates the Diferenca1.txt file with the condition to apend information, writting the current frequency
              #and the measured magnitude
              with open("Diferenca1.txt","a")       as rpt:
                            rpt.write(str(self.freq)+" "+str(self.c)+"\n")

              #Calculates division between imaginary and real parts of the complex signal measured
              self.div_fase_sink = self.gr_probe_sink_img.level()/self.gr_probe_sink_real.level()
              self.div_fase_sourcea = self.gr_probe_sourcea_img.level()/self.gr_probe_sourcea_real.level()
              self.div_fase_sourceb = self.gr_probe_sourceb_img.level()/self.gr_probe_sourceb_real.level()

              #Computes the arc tangent and transforms from radians to degrees
              self.arctan_sink = math.degrees(math.atan(self.div_fase_sink))
              self.arctan_sourcea = math.degrees(math.atan(self.div_fase_sourcea))
              self.arctan_sourceb = math.degrees(math.atan(self.div_fase_sourceb))

              # Calculates the angle difference and makes the values always positive just to see if it is possible to
              #used this technique. Must be reviewed for improvements.
              self.pha=(math.sqrt(self.arctan_sink*self.arctan_sink))-(math.sqrt(self.arctan_sourcea*self.arctan_sourcea))
              self.diff_sourcea = self.pha
              self.phb=(math.sqrt(self.arctan_sink*self.arctan_sink))-(math.sqrt(self.arctan_sourceb*self.arctan_sourceb))
              self.diff_sourceb = self.phb

              #Creates the FaseA.txt file with the condition to apend information, writting the current frequency and
              #the measured phase difference
              with open("FaseA.txt","a")    as rpt:
                            rpt.write(str(self.freq)+" "+str(self.diff_sourcea)+"\n")
              #Creates the FaseB.txt file with the condition to apend information, writting the current frequency and
              #the measured phase difference
              with open("FaseB.txt","a")    as rpt:
                            rpt.write(str(self.freq)+" "+str(self.diff_sourceb)+"\n")

              #Updated the graphical interface textboxes with the current values (just measured)
              self.text_ctrl_2.SetValue(eng_notation.num_to_str(self.b))
              self.text_ctrl_10.SetValue(eng_notation.num_to_str(self.a))
              self.text_ctrl_1.SetValue(eng_notation.num_to_str(self.freq))
              self.text_ctrl_12.SetValue(eng_notation.num_to_str(self.samp_rate))

              #Increases the frequency values, adding the increment
              self.freq += self.step_freq
              self.sg.set_frequency(self.freq, verbose=True)
              rx_freq = self.freq + self.offset
              self.rx.set_frequency_a(rx_freq, verbose=True)
              self.rx.set_frequency_b(rx_freq, verbose=True)

              #Condition when the sweep finishes. Calls Gnuplot to plot the files with the measurement results
              if self.freq > self.stop_freq + self.step_freq:
                self.Stop(0)
                #Calls gnuplot to plot the results from the files
                #Sets many Gnuplot variables
                g = Gnuplot.Gnuplot()
                g('set multiplot')
                g('set xlabel "Frequency (Hz)"')
                g('set ylabel "Amplitude"')
                #Calls all the files that will be plotted
                plot1 = Gnuplot.File('BasicRx1.txt', title="BasicRx")
                plot2 = Gnuplot.File('LFRX1.txt', title="LFRX")
                plot3 = Gnuplot.File('Diferenca1.txt', title="Diferenca")
                #Puts all the plots on the same window
                g.plot(plot1, plot2,plot3)
                g.title('Frequency Response Amplitude')
                #Calls the wait instance and waits the user to press any key before killing the whole program
                self.wait()

                self.duc_freq = self.freq

    # end of class MyFrame

    #Updates the internal timer and sets the target time value as the time interval between two different
    #frequencies which is configured via graphical interface. dur = 1000 is the fraction of 1 second, so
    #in the beginning every time value is is miliseconds scale.
    #Also, starts the entire flow graph (transmitter and receiver)
    class UpdateTimer(wx.Timer):
        def __init__(self, target, dur=1000):
            wx.Timer.__init__(self)
            self.target = target
            self.Start(dur)

        #Calls the update routine which uptades the frequency values, makes the probes and wrinttings to the files
        def Notify(self):
            #Called every time the timer is over
            if self.target:
                self.target.OnUpdate()

    #Creates and starts the graphical window
    class MyApp(wx.App):
      def OnInit(self):
        frame = MyFrame(None, -1, "Automatic Frequency Response")
        frame.Show(True)
        self.SetTopWindow(frame)
        return True

    app = MyApp(0)
    app.MainLoop()
    #############################################--END-OF-THE-PROTOTYPE--########################################
```