Software Defined Radio on Digital Communications: a New Teaching Tool

André L. G. Reis*, André F. B. Selva*, Karlo G. Lenzi†, Sílvio E. Barbin‡ and Luís G. P. Meloni*
*FEEC – School of Electrical and Computer Engineering
UNICAMP – State University of Campinas
Campinas, SP – Brazil
†DRC – Convergent Networks Department
CPqD – Research and Development Center
Campinas, SP - Brazil
‡EPUSP – Polytechnic School
USP – University of São Paulo
São Paulo, SP - Brazil.
e-mail: andre.lgr@gmail.com, andrefselva@gmail.com, klenzi@cpqd.com.br, barbin@usp.br and meloni@decom.fee.unicamp.br

Abstract—**In this paper we present a Software Defined Radio (SDR) platform, composed by a USRP hardware and GNU Radio, as a new approach for teaching telecommunications at schools of Electrical Engineering. The proposed approach makes the preparation of an experiment faster by using modern communications architectures, which significantly reduce the occurrence of errors during the setup of test beds as well. This gives the students the opportunity to focus their efforts on the learning of communications techniques and algorithms. Furthermore, we intend to make easier the verification and validation processes of implemented models, supported by the use of specialized tools.**

Keywords: GNU Radio; Telecommunications Laboratory; USRP; Digital Signal Processing; Software Defined Radio

I. INTRODUCTION

Nowadays, telecommunications fundamentals are presented through analog platforms i.e. circuits with analog components in undergraduate laboratories. In general, the students are responsible for assembling circuits that implement certain communication techniques, and using several equipments to stimulate and evaluate the performance of those circuits, such as function generators, oscilloscopes, spectrum analyzers, among others. They are asked to report back the results and draw conclusions about the experiment success or difficulties. All these tasks, performed in a short time interval, limit the experiment complexity.

Failures related to assembly mistakes, use of defective components and inappropriate use of equipments may compromise the entire experiment, delay its implementation and, thereby making the learning process more difficult, since students have to spent some time solving these problems, instead of using time effectively to understand the concepts involved in the experiment.

A similar example is the case of Logic Circuits Laboratory [1] classes. A few years ago, this discipline was taught through the use of protoboards and several logic gates ICs. Students, after a short explanation about the experiment, would spent much time cutting and connecting copper wires from one logic gate to another in order to implement their projects. However, due to advances in technology, with the advent of programmable logic devices and emphasis on design specification through hardware description languages, the use of protoboards and ICs became outdated and was gradually removed from courses agenda. Nowadays, Logic Circuits laboratories use development kits with programmable hardware, such as FPGAs (Field-Programmable Gate Array) and CPLDs (Complex Programmable Logic Devices), from manufacturers, such as Altera and Xilinx, along with its development environments (e.g. Quartus II – Altera [2]). Students no longer waste their time creating a lot of wire connections that make the project more susceptible to human failure and also impair modifications and the inspection of certain parts of the circuit. By transferring to synthesis tools the task of routing and connecting those logic elements, it is possible to increase the circuit reliability and the experiment's complexity [3]. This brought more depth and quality to teaching making those devices essential to these courses.

In this paper, we propose to change the current way that telecommunications laboratory classes are taught at universities in general, traditionally done through hardware-only platforms. Our approach is primarily focused on software, supported by a new communication model called Software Defined Radio or SDR, and allows the students to focus their efforts on more relevant aspects of the class like telecommunication systems design, supported by the theory available in the literature, reducing the challenges of assembling and validating the experiment dramatically.

II. THE SOFTWARE DEFINED RADIO

The main idea of an SDR is to transfer tasks performed by hardware to software. System characteristics such as signal modulation scheme, operation frequencies, bandwidth, and

This work is supported by FINEP – Financiadora de Estudos e Projetos - Brazil, contract no. 01.07.0110.00, "Avaliação do Middleware Ginga".

others, are no longer dependent on analog circuits, which in general are pre-defined in a conventional radio equipment. In an SDR they rely on a system that integrates a programmable hardware and software that offers flexibility to modify those characteristics. Thus, this kind of radio can behave in different ways, making it possible to perform changes in the system features by software, simply changing the parameters responsible the definition of its behavior, even in runtime. Furthermore, it is possible to have a completely different communication system just by replacing the software that is executed, keeping the same hardware. For instance, with the same platform one can evolve from an FM Receiver [4] to a complex Digital TV transmitter [5]. This is possible by means of the evolution of programmable devices such as DSPs (Digital Signal Processors) and FPGAs which made this architecture flexible enough to adapt to different applications without the need of modifying the hardware structure itself.

SDR Platforms are already widely used in scientific field. Its versatility allows several researches in telecommunications to be conducted in very elegant ways as, for example, in cognitive radios [6], OFDM (Orthogonal Frequency Division Multiplexing) modulation [7], GSM base stations [8], etc. Since one can deploy different radios with the same hardware, SDR shows itself as an interesting solution to be used in telecommunications laboratories because it can significantly simplify the execution of experiments and allow a wide variety of systems to be implemented and tested just by modifying an algorithm developed in software. Many devices, like diodes, capacitors, etc., or even equipments can be replaced by a single hardware platform. This does not mean that learning how to use such components and equipments is not important.

From this, two advantages arise: 1) students can dedicate more time to understand the concepts involved in the elaboration and the design of communication systems and 2) they will spent less time setting-up the test bed. Also, this approach reduces the budget necessary to prepare a telecommunications laboratory, since it can be composed just by a PC and an appropriate SDR board for each student group.

There are in the market several solutions for implementing an SDR system. In this paper, we will present the solution proposed in [9], the USRP (Universal Software Radio Peripheral), which is commercially available. Fig. 1, reproduced from [10], shows a scheme of an SDR that uses USRP.

Composed by a motherboard and configured by many daughter boards, the USRP abstract the system RF front-end. The motherboard performs, in general, the base-band to intermediate frequency processing and vice-versa, while daughterboards perform up-conversion/down-conversion of frequency and transmission/reception of signal.

USRP communicates to PC through a Gigabit *Ethernet* interface (as in Ettus N210 [11]) or USB, in earlier models. This allows the personal computer to receive and store data in order to process the digital signals within it in specialized software environments as Matlab or GNU Radio.

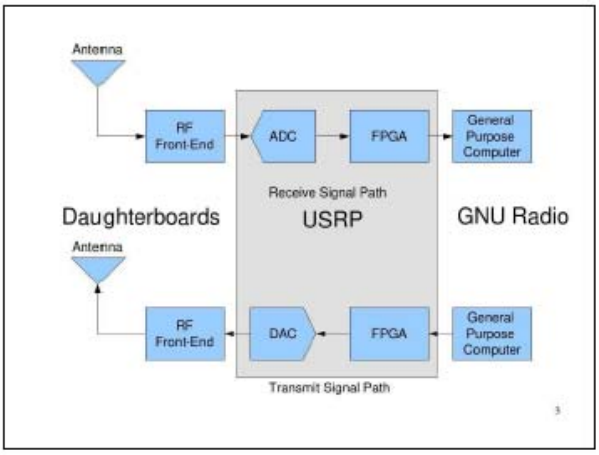

Figure 1. SDR with USRP and GNU Radio [10].

III. GNU RADIO

GNU Radio [12] is an open-source toolkit that provides tools for development and simulation of SDR systems. It is used to design and execute algorithms that define a desired communication system.

There are basically three ways to use GNU Radio. In a high-level perspective, we can use GNU Radio Companion (GRC), which is a graphical tool where we can build an SDR system by connecting signal processing blocks, establishing a processing chain or flow, from signal input to system output. In an intermediate-level, we can use the programming language Python as a way to describe the block connections or in its lowest level, we can modify and even create new processing blocks using C++, chosen due to performance issues, and use these blocks in the higher levels (Python or GRC).

In this paper, we will focus our attention to GNU Radio Companion due to its friendly interface and ease of use. Fig. 2 shows the interface of this environment.

In the right side bar of GRC we find several types of blocks and tools that can be used to build a system. There are signal generators, operators, graphical sinks, filters, among others. There are also blocks responsible to communicate with USRP hardware. The N210 board by ETTUS [11] uses UHD (Universal Hardware Driver) Sink and UHD Source. These drivers allow an application on PC running in GNU Radio to send and receive signals to/from the board. Fig. 3 and Fig. 4 show UHD Sink block and its parameters screen, respectively.

Figure 3. UHD *Sink* block.

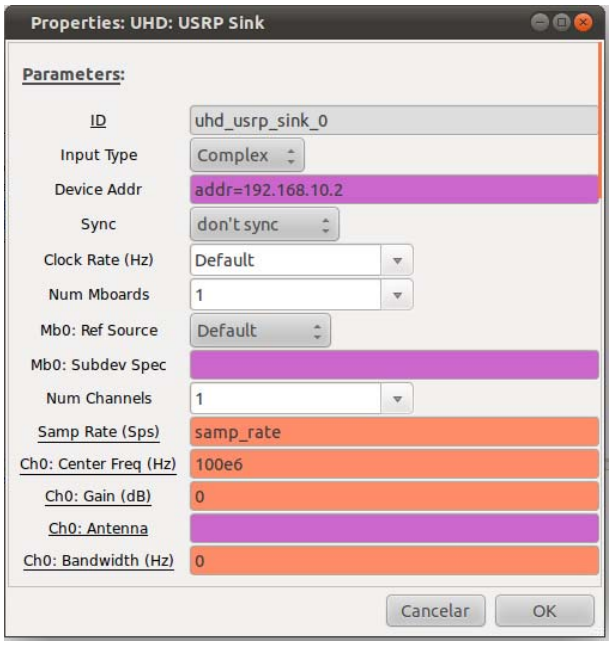

Figure 4. UHD *Sink* parameters.

The development process consists, basically, in setting-up and connecting blocks in a certain way so that the sequence of connection defines a processing flow. The output of last stage should provide the desired signal.

The possibility to set-up the block parameters gives high flexibility to this kind of development. We can focus our efforts on algorithm, and not on the hardware that will perform it. Thus, we can change, for instance, the input or output signal central frequency through a slider in runtime, modify modulation characteristics, among others. These benefits would not be easily achieved in a regular test-bed for telecommunications laboratory.

Each block deals with a certain data type on its interfaces (e.g. float, complex, etc), which in many cases is configurable through the parameters *Input Type* or *Output Type*. If the block does not offer support to modify the data type, it still can be done by type-casting blocks offered by GRC. In order to distinguish such types, GRC uses a color code on the interface of blocks. Table I shows a partial list of the relation between color and data type. The full list can be found in menu *Help/Types* of GRC.

TABLE I. RELATIONSHIP BETWEEN CONNECTOR COLOR AND DATA TYPE.

Connector Color	***Data Type***
Blue	*Complex*
Orange	*Float*
Yellow	*Short*
Magenta	*Byte*
Green	*Integer*

GRC still has a terminal that shows status messages about the system, in runtime. Hardware communication problems, wrong parameter value, etc., are examples of status messages that can be shown in this terminal.

IV. CURRENT SCENARIO ON TEACHING TELECOMMUNICATIONS FUNDAMENTALS

Nowadays, telecommunications laboratory is taught through the use of analog components in the experiments. In Reference [13] we can find the experiment guides used in discipline Communications Laboratory I, taught in School of Electrical and Computer Engineering (FEEC) at State University of Campinas (UNICAMP). In this discipline, students practice many communications concepts learned in theory such as AM and FM modulation and demodulation, frequency spectrum of signals, etc, using analog circuits and equipments like signal generators, spectrum analyzers and oscilloscopes.

The high cost of equipments traditionally used to execute a telecommunications laboratory might be prohibitive to universities that have low budget. By using software defined radio platforms, it would be possible to reduce the cost of those laboratories so it could be more accessible to such universities. A telecommunications laboratory could be executed just using personal computers and SDR boards. An Ettus N210 board [11] along with daughter boards BasicTX and BasicRX [14], for instance, cost less than two thousand dollars [9]. There are other SDR boards that cost even less [15]. Just the cost of a signal vector generator overcomes this value. All these equipments could be replaced by the tools included in GNU Radio.

By replacing those components and equipments by a PC-based platform, it would also be possible to reduce errors during experiments, increase its complexity, reduce assembly time and give to the student more focus on algorithms and communications techniques. In other words, the same benefits achieved in Logic Circuits laboratories by migrating from protoboards to development kits with FPGAs could be also achieved here by correct use of SDR platforms.

Another point is that model simulation and verification in PC presents several advantages since it is easy to modify parameters of the model or even the model as a whole and test its output before doing any physical implementation. Furthermore, mathematical toolboxes, as Matlab or Octave, can be used to help processing these data.

V. AN EXAMPLE OF EXPERIMENT USING SOFTWARE DEFINED RADIO

In [16] we find a guide to an experiment executed in Communications Laboratory I discipline at UNICAMP. This experiment consists in practical study of amplitude modulation.

In this section we will present an alternative implementation of some topics of this experiment using GNU Radio Companion in order to establish a comparison between the traditional approach and the one proposed here.

An amplitude-modulated wave is described by (1), according to [16].

$$x_c(t) = [1 + mx(t)]A_c\cos(2\pi f_c t) \quad (1)$$

The parameters of (1) are described in Table II.

TABLE II. AM MODULATION PARAMETERS.

Parameter	*Description*
$A_c\cos(2\pi f_c t)$	*Non-modulated carrier*
A_c	*Constant*
m	*Modulation Index*
$x(t)$	*Message*
f_c	*Carrier frequency*

We can describe (1) as a blocks diagram using GRC. Fig. 5 shows this description.

To generate the carrier and the "message", we used two "Signal Source" blocks, configured to meet the experiment requirements. The operations performed with these signals are represented by the blocks "Add" and "Multiply". Parameters of the system are stored using "Variable" blocks. The equivalents to oscilloscope and spectrum analyzer are the "WX GUI Scope Sink" and "WX GUI FFT Sink" blocks, respectively, which produce graphical outputs of the processed signal. It is worth emphasizing that it is necessary to use a "Throttle" block, otherwise, in case of simulation without audio or USRP blocks, the GUI elements will be locked, given that GRC will consume all CPU available [17].

The parameters of (1) used in the system presented in Fig. 5 from first section of practical tasks of [16] are presented in Table III. It was used a scale adjustment of 10^3 for convenience.

TABLE III. AM MODULATION PARAMETERS FOR THE EXPERIMENT.

Parameter	*Value*
$A_c\cos(2\pi f_c t)$	$\cos(2\pi 10^3 t)$
A_c	*1*
m	*50%*
f_c	*1 kHz*

For a "message" described by

$$x(t) = \sin(2\pi 20t), \quad (2)$$

one obtains the following outputs from Scope Plot (equivalent to oscilloscope) and from FFT Plot (equivalent to spectrum analyzer), showed by Fig. 6 and Fig. 7, respectively.

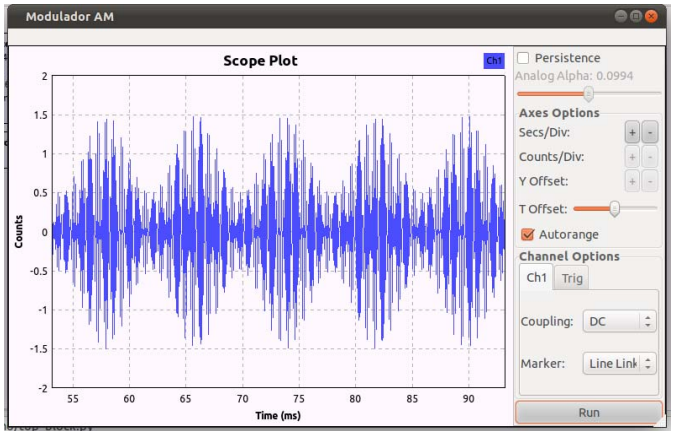

Figure 6. *Scope Plot* – Amplitude-modulated sine wave.

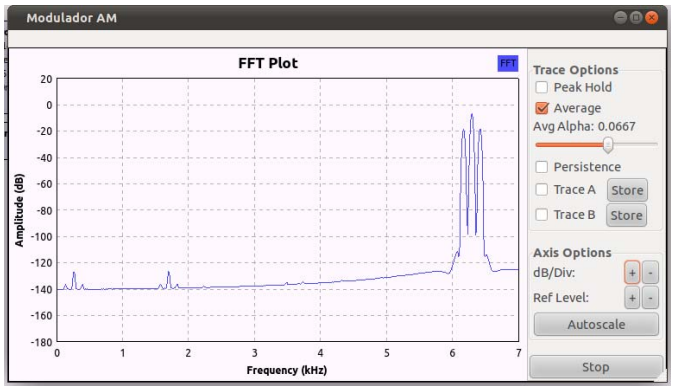

Figure 7. *FFT Plot* – Frequency spectrum of the amplitude-modulated sine wave.

Our goal now is to calculate the modulation index m through the graphics we obtained by simulating the system. There are two ways to obtain such index.

First, we can use the temporal output of the system and obtain m through

$$m = (A_{max} - A_{min})/(A_{max} + A_{min}), \quad (3)$$

where A_{max} and A_{min} are maximum and minimum modulated-wave's amplitude, respectively, according to [16].

Through the system frequency spectrum, analyzing Fig. 8 [16], we can obtain a relation to calculate m, presented by (4).

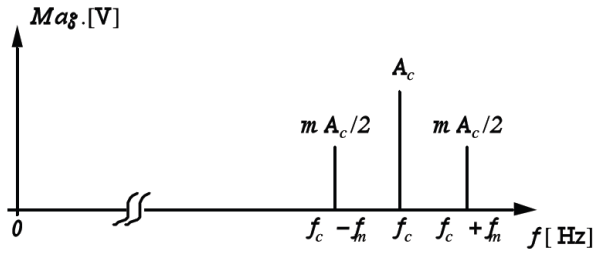

Figure 8. Frequency spectrum of a Tonal Modulation[16].

$$m = 2(\text{amplitude in } f_c\text{-}f_m)/(\text{amplitude in } f_c) \quad (4)$$

From Fig. 6, the modulated wave has a maximum amplitude value of approximately 1.48 and minimum amplitude of 0.49. Then, modulation index value is, using (3)

$$m = (1.48\text{-}0.49)/(1.48+0.49) = 0.50. \quad (5)$$

Now using Fig. 7 referent to frequency spectrum, the amplitude in f_c-f_m is approximately -19 dB, in other words, 0.11. The amplitude in f_c is approximately -7.2 dB, that is, 0.44. Thus, using (4), modulation index is

$$m = 2(0.11)/0.44 = 0.50. \quad (6)$$

As expected by configuration showed in Table III, the modulation index measured by (5) and (6) is approximately 50%.

Changing the message to a triangular wave, as required by this laboratory guide [16], we obtain the following graphics, presented in Fig. 9 and Fig. 10.

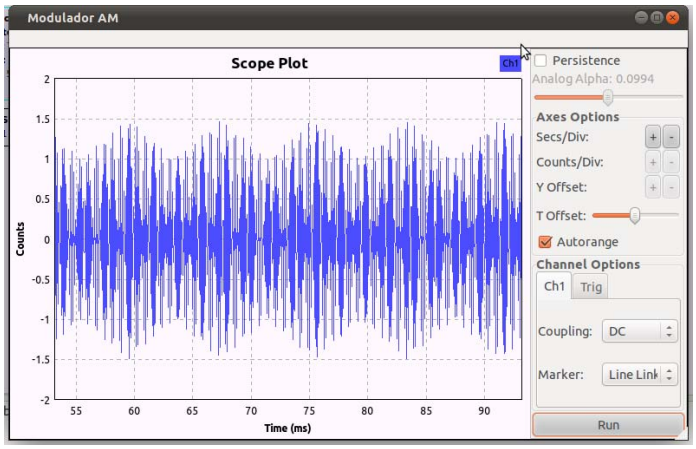

Figure 9. *Scope Plot* – Amplitude-modulated triangular wave.

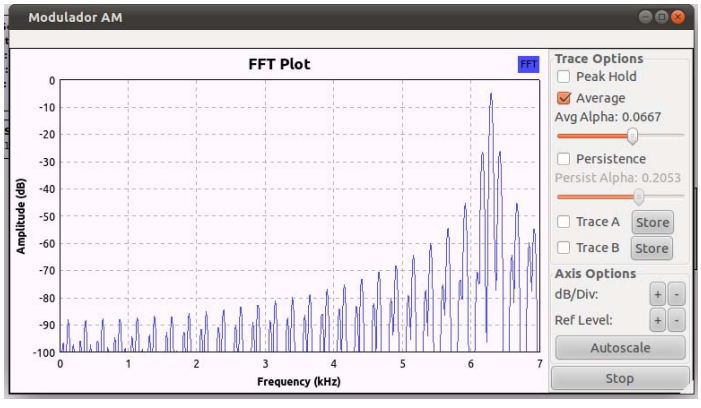

Figure 10. *FFT Plot* – Frequency spectrum of amplitude-modulated triangular wave.

To conclude the equivalence demonstration of this experiment in GRC, using a Gaussian noise with amplitude 0.5 and seed 42 as "message", we obtain the graphics showed in Fig. 11 and Fig. 12.

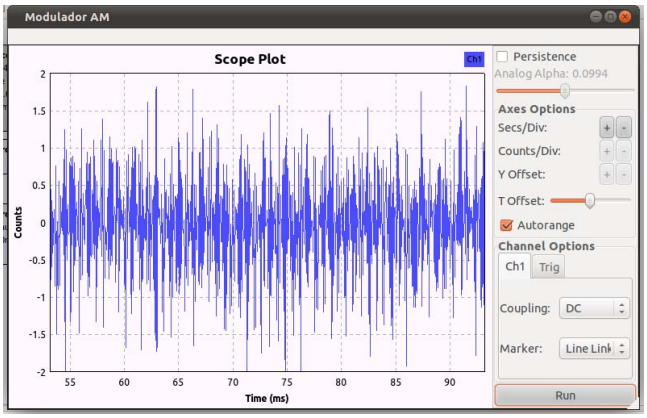

Figure 11. *Scope Plot* – Amplitude-modulated noise.

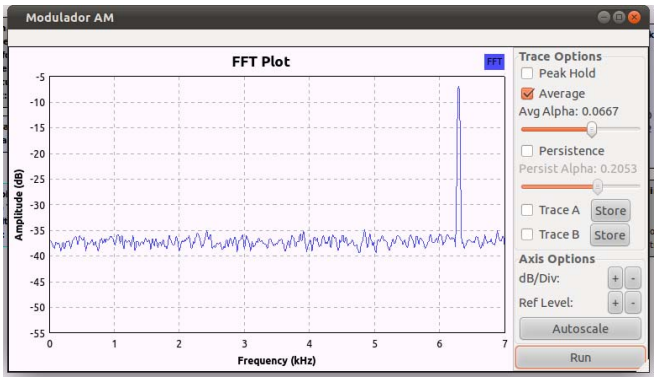

Figure 12. *FFT Plot* – Frequency spectrum of amplitude-modulated noise.

VI. DISCUSSIONS

In the traditional execution of the experiment presented in Section V, we use the function generator, oscilloscope and spectrum analyzer. Also, for the experiment as a whole, we use an analog AM Modulator build in board, showed in Fig. 13 [16].

The functioning of all these devices can be replaced by GNU Radio Companion, as we showed. Even though we have not executed the part of experiment that uses the board presented in Fig. 13, this could also be replaced by the schematic showed in Fig. 5, since both implementations perform AM Modulation of a message, one by means of analog circuits and other by means of mathematical model implemented via software, respectively.

With respect to experiment execution time, the initial effort on using GRC is to describe the mathematical expression in terms of signal processing blocks. Then, changing parameters is trivial. Once the model is done, it is possible to verify its behavior in any point by using the graphical analysis tools available at GRC, what makes the process much more efficient

when compared to other verification methods available in hardware, as multimeters and oscilloscopes test probes. It is also possible to load or store data into files for stimuli or for future processing in specialized mathematical tools.

It is also possible to state that GRC implementation is more flexible than circuit based case. In this case, the carrier frequency is fixed in 1 MHz in the circuit because of the LC (Inductor-Capacitor) circuit used, as we can see in Fig. 13. By other hand, it is possible to change this frequency in GRC just by modifying a single variable, which is described in Table II.

By using the USRP hardware (e.g. N210 [11]) we can send the modulated signal through the air with UHD Sink block, previously showed by Fig. 3. We can also use the PC sound card to generate a message (voice signal), received from microphone by the block Audio Source available at GRC. In short, there are several ways to improve the experiment through the use of the features offered by SDR platforms.

Another relevant aspect is that the accuracy of the results obtained in this experiment is higher than the traditional model, because the latter relies on circuits designed with low precision and quality components, while the former is a consolidated commercial solution. Still, the instruments used to measure the variables have limited accuracy and are based on instant measures, while the solution SDR with GNU Radio can establish its measures based on statistics of the received signal, built by a collection of signal samples.

Finally, due to its flexibility in modifying parameter of the system, it is possible to reduce the incidence of experiment report plagiarism. To do so, we just need to define some characteristic of the system based on some sort of combination of SRN (Student Registration Number) digits of the group members. In other words, each report will have unique data and graphics, as long as we choose an appropriate combination expression of the digits.

VII. Conclusion

In this paper we presented a software defined radio platform, composed by a USRP [9] and GNU Radio [12], with the goal of replacing current devices used in conventional telecommunications laboratories. It was shown that we can reduce the time spent in the experiments by replacing the traditional way they are executed, with analog components as diodes and capacitors, to a PC-based platform. This allows the students to dedicate themselves on what really matters, namely the communications techniques and algorithms.

This proposal makes the teaching process more dynamic and efficient as well as it is capable of reducing the costs necessary to prepare a laboratory of communications class at a university. These advantages, along with the need to follow technological advances of communication systems, make this proposal a feasible and interesting solution to improve telecommunications teaching.

References

[1] W. S. Ting, “EA773: Laboratório de Circuitos Lógicos”. [Online]. Available: http://www.dca.fee.unicamp.br/courses/EA773/1s2011/index.html

[2] Altera, “Quartus II Web Edition Software”. [Online]. Available: http://www.altera.com/products/software/quartus-ii/web-edition/qts-we-index.html

[3] W. S. Ting, “EA773 Laboratório de Circuitos Lógicos - Experiência 5: calculadora com memória”. [Online]. Available: http://www.dca.fee.unicamp.br/courses/EA773/1s2011/roteiros/rot_5.pdf

[4] M. Fähnle, “Software-Defined Radio with GNU Radio and USRP/2 hardware frontend: setup and FM/GSM applications”, Hochschule Ulm University of Applied Sciences, 2010.

[5] V. Pellegrini, G. Bacci and M. Luise, "Soft-DVB, a fully software, GNURadio based ETSI DVB-T modulator", in Proc. WSR'08, Karlsruhe, Germany, March 2008.

[6] Z. Yan, Z. Ma, H. Cao, G. Li, and W. Wang, “Spectrum sensing, access and coexistence testbed for cognitive radio using USRP”, 4th IEEE International Conference on Circuits and Systems for Communications, 2008. ICCSC 2008, 26-28 May 2008.

[7] A. Marwanto, M. A. Sarijari, N. Fisal, S. K. S. Yusof and R. A. Rashid, "Experimental study of OFDM implementation utilizing GNU Radio and USRP - SDR," Communications (MICC), 2009 IEEE 9th Malaysia International Conference, 15-17 Dec. 2009.

[8] E. Natalizio, V. Loscrí and, G. Aloi, N. Paolí and N. Barbaro, “The practical experience of implementing a GSM BTS through open software/hardware,” in Applied Sciences in Biomedical and Communication Technologies (ISABEL), 2010 3rd International Symposium, Nov. 2010.

[9] M. Ettus, “Universal Software Radio Peripheral”. [Online]. Available: http://www.ettus.com.

[10] L. K. Patton, “A GNU Radio based software-defined radar”, Wright State University, 2007, p. 10.

[11] M. Ettus, “USRP N200 Series”. [Online]. Available: http://www.ettus.com/downloads/ettus_ds_usrp_n200series_v3.pdf

[12] GNU Radio. [Online]. Available: http://gnuradio.org

[13] G. Fraidenraich, “EE882 - Laboratório de Comunicações I”. [Online]. Available: http://www.decom.fee.unicamp.br/~gf/Decom/EE882.html

[14] M. Ettus, “TX and RX Daughterboards”. [Online]. Available: http://www.ettus.com/downloads/ettus_daughterboards.pdf

[15] FlexRadio Systems, “FLEX-1500 Transceiver”. [Online]. Available: http://cart.flexradio.com/FLEX-1500_p_10.html

[16] M. D. Yacoub, “Unicamp - EE882 - Laboratório de Comunicação I - Experiência 3: modulação em amplitude”. [Online]. Available: http://www.decom.fee.unicamp.br/~gf/Decom/EE882_files/EE882-exp3.pdf

[17] GNU Radio, “GNU Radio Companion - usage tips”. [Online]. Available: http://gnuradio.org/redmine/projects/gnuradio/wiki/GNURadioCompanion#Usage-Tips

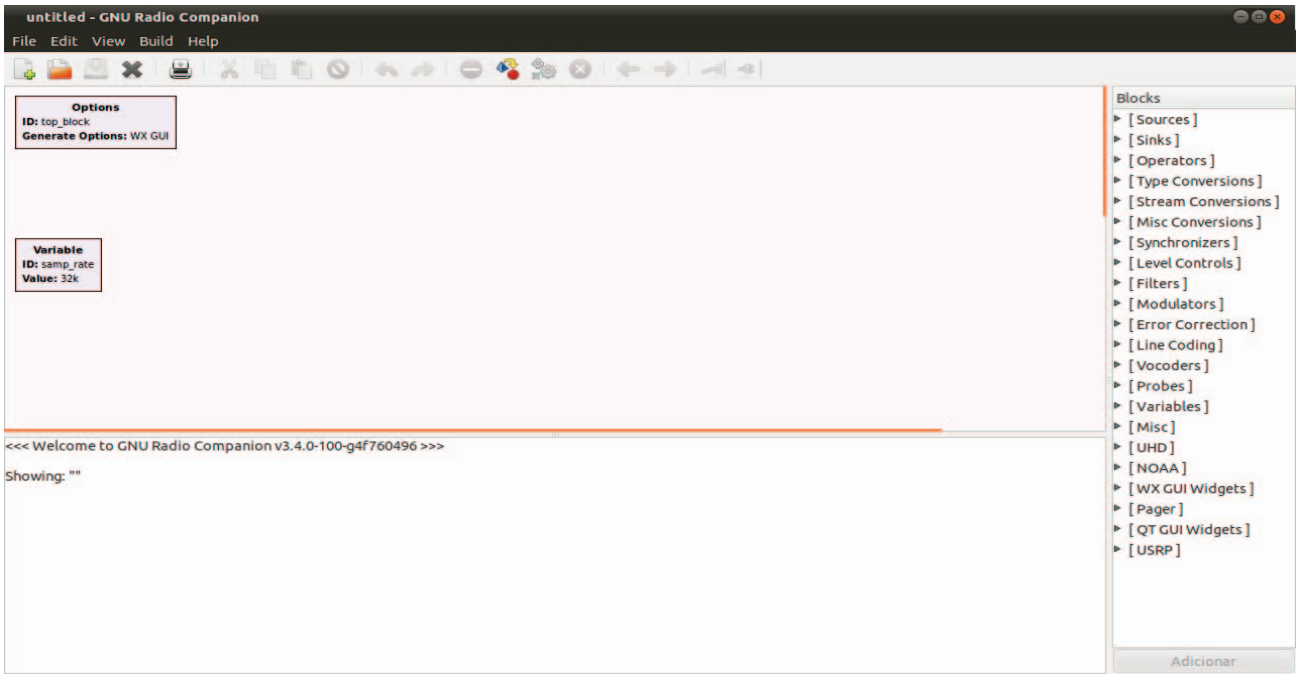

Figure 2. GNU Radio Companion interface.

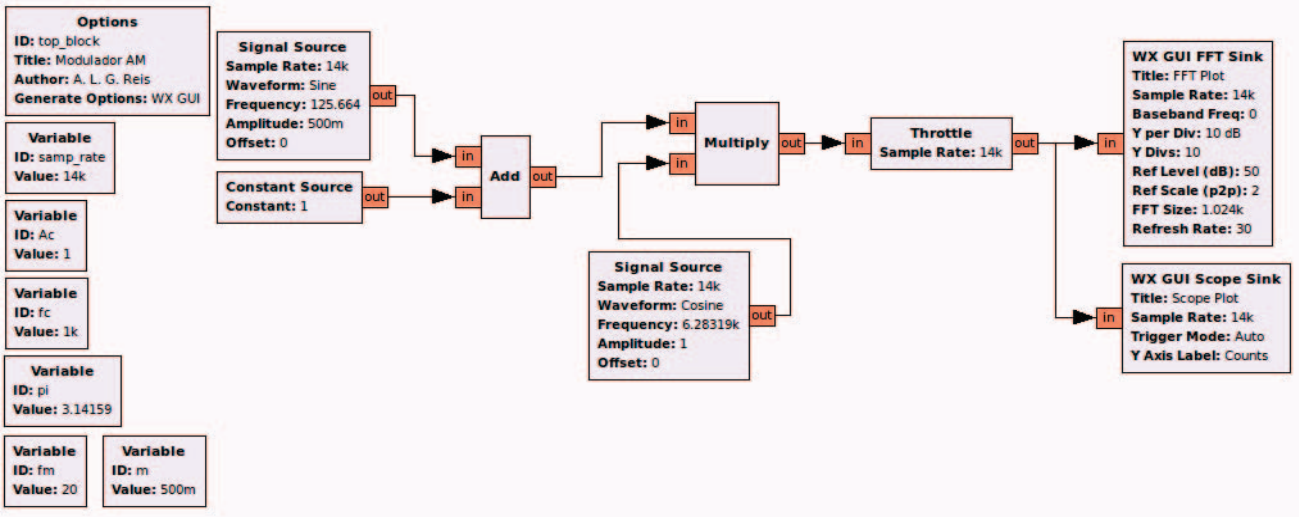

Figure 5. Blocks diagram of AM Modulator.

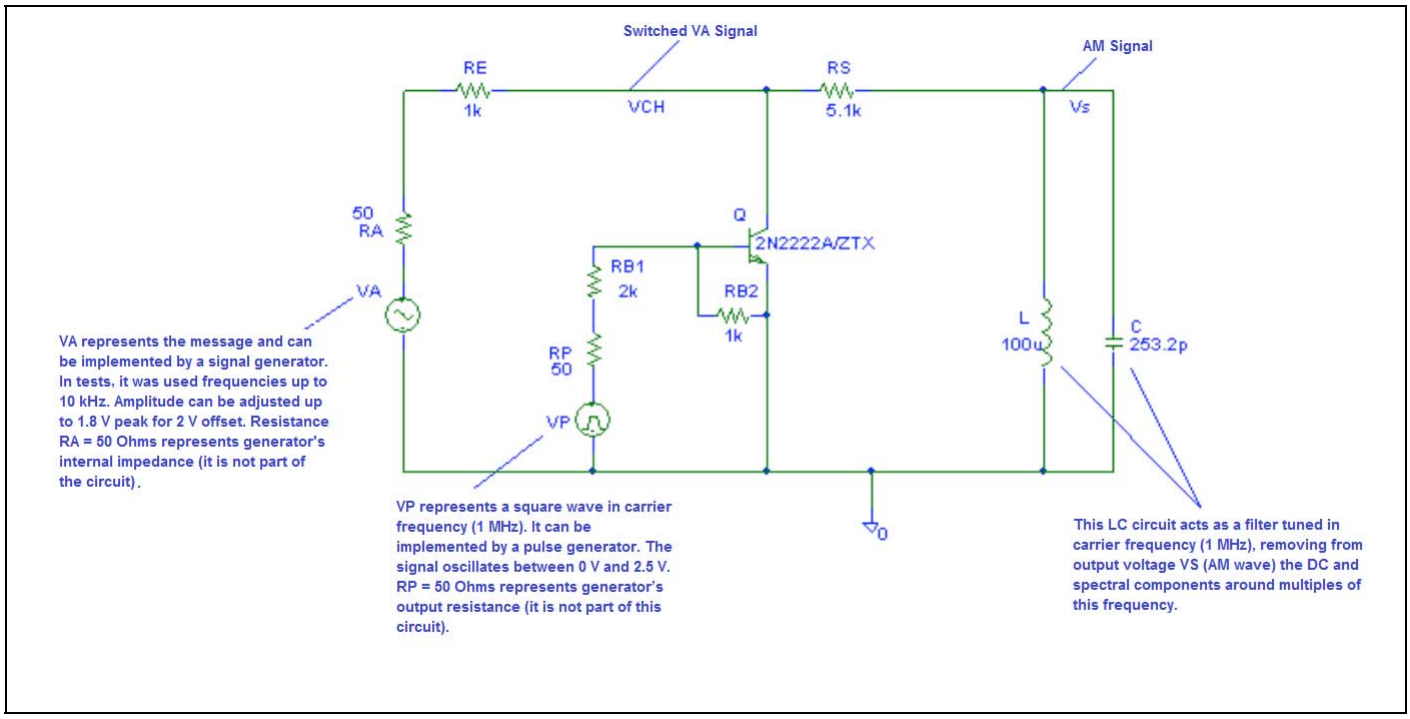

Figure 13. AM Modulator board [16].