Introduction to the Software-defined Radio Approach

A. F. B. Selva, A. L. G. Reis, K. G. Lenzi, L. G. P. Meloni, *Member, IEEE* and S. E. Barbin, *Member, IEEE*

***Abstract*— Telecommunications have been in constant evolution during past decades. Among the technological innovations, the use of digital technologies is very relevant. Digital communication systems have proven their efficiency and brought a new element in the chain of signal transmitting and receiving, the digital processor. This device offers to new radio equipments the flexibility of a programmable system. Nowadays, the behavior of a communication system can be modified by simply changing its software. This gave rising to a new radio model called Software Defined Radio (or Software-Defined Radio - SDR). In this new model, one moves to the software the task to set radio behavior, leaving to hardware only the implementation of RF front-end. Thus, the radio is no longer static, defined by their circuits and becomes a dynamic element, which may change their operating characteristics, such as bandwidth, modulation, coding rate, even modified during runtime according to software configuration. This article aims to present the use of GNU Radio software, an open-source solution for SDR specific applications, as a tool for development configurable digital radio.**

***Keywords*— GNU radio, Software-defined radio, SDR, USRP.**

I. Introdução

ATUALMENTE na área de telecomunicações novas tecnologias são pesquisadas para os mais variados meios. O setor vem crescendo exponencialmente tanto em número de usuários, quanto na necessidade de largura de banda dos sistemas de comunicações. A digitalização dos sistemas de comunicações, tais como da televisão e do rádio deverá ser realidade na grande maioria dos países ao redor do mundo ainda nesta década. Novas aplicações vem surgindo, tais como a transmissão de vídeo em 3D, codificação em super alta-definição, áudio espacial com maior número de canais, entre outras, que exigem taxas de bits crescentes. Fornecer soluções capazes de suportar as necessidades de um mercado tão dinâmico e inovador tornou crítica a capacidade de adaptação desses novos equipamentos de rede e de dispositivos de usuários.

O conceito de rádio definido por software (SDR) permite substituir a tradicional implementação em hardware dos dispositivos de comunicação por uma implementação mais flexível, que faz uso de dispositivos programáveis controlados por software, como por exemplo, um computador pessoal ou um processador embarcado.

A SDR não é nova, pode-se mesmo atribuir o seu nascimento ao da área de processamento digital de sinais. Este fato advém da observação de que em conjunto com os algoritmos de transformadas rápidas, tais como as FFTs, e dos filtros digitais, vieram as técnicas digitais de modulação e demodulação por software. Entretanto nos dias de hoje, há um enorme interesse da comunidade científica e empresarial por SDR, isto pelo fato de que esta tecnologia atingiu um nível de desenvolvimento tal que permite a implementação prática de vários algoritmos que até então não podiam ser executados em tempo real, devido às limitações de velocidades dos circuitos integrados. Um cenário análogo a este foi visto nos anos 70 do século passado, quando não se tinham os circuitos especializados de processamento digital de sinais (DSP), quando era necessária a implementação de algoritmos de DSP por meio de vários circuitos especializados, o que tornava a aplicação cara e pouco difundida. Tal como ocorreu com a difusão da tecnologia de DSPs, espera-se que a tecnologia SDR seja o *breakthrough* das aplicações em eletrônica das próximas décadas.

Segundo o *Wireless InnovationForum* [1], define-se SDR como "radio na qual algumas ou todas as funções da camada física são definidas por software". Em outras palavras SDR refere-se à tecnologia onde módulos de software são executados em tempo real em plataformas genéricas de microprocessadores, processadores digitais de sinais ou em circuitos lógicos programáveis na implementação de funções de geração de sinal a ser transmitido (modulação) ou a detecção do sinal de rádio recebido (demodulação). A tecnologia de SDR pode ser empregada em diversas aplicações de rádio-frequência, tais como, Bluetooth, WLAN, GPS, Radar, WiMAX, LTE, etc.Um rádio definido por software nada mais é do que um sistema de comunicação baseado em software, cujas principais características de operação podem ser modificadas em tempo de execução: o sistema pode ser facilmente re-configurado para desempenhar diferentes funções de acordo com a necessidade. Para suportar diferentes padrões de comunicação, tais como WiMAX, LTE, Wi-Fi ou qualquer outro, como também ajustar-se às variações do canal de comunicação, como interferências e outras distorções, basta carregar diferentes tipos de software em memória, sem precisar substituir todo equipamento de rádio, tal como era feito em décadas passadas. Desta forma, as funções de hardware podem ser reduzidas às técnicas de conversão para cima e para baixo do sinal modulado para o canal de comunicação em radiofreqüência. O processador faz

A. F. B. Selva, Universidade Estadual de Campinas (UNICAMP), Campinas, Brasil, andrefselva@gmail.com
A. L. G. Reis, Universidade Estadual de Campinas (UNICAMP), Campinas, Brasil, andre.lgr@gmail.com
K. G. Lenzi, Universidade Estadual de Campinas (UNICAMP), Campinas, Brasil, klenzi@cpqd.com.br
L. G. P. Meloni, Universidade Estadual de Campinas (UNICAMP), Campinas, Brasil, meloni@decom.fee.unicamp.br
S. E. Barbin, Universidade de São Paulo (USP), São Paulo, Brasil, barbin@usp.br

o tratamento digital do esquema de modulação e comunica-se com os conversores ADC e DAC, interligados a um *front-end* de radiofrequência (RF) para o deslocamento para a freqüência de transmissão e vice-versa. A Fig. 1 ilustra este conceito.

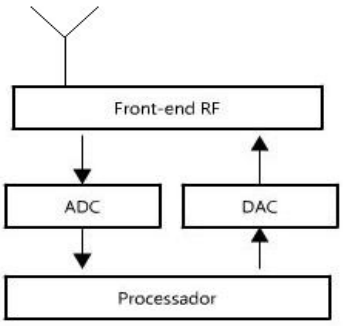

Figura 1. Funcionamento de um SDR.

Um dos primeiros sistemas SDR conhecidos é o SpeakEasy [2], projeto militar do governo norte-americano iniciado nos anos 70. O projeto visava emular mais de uma dezena de rádios militares, operando entre 2 e 2000 MHz, além de permitir a incorporação futura de novos padrões de modulação [3]. O termo *Software-Defined Radio*, por sua vez, foi introduzido em 1991, por Mitola, que publicou um dos primeiros artigos sobre o assunto em 1992 [4]. Também foi o proponente do termo Rádio Cognitivo [5] que aliado aos conceitos de SDR, utiliza elementos cognitivos para determinar o modo de operação do rádio [6].

Rádiosdefinidos por software são excelentes opções em sistemas que precisam suportar múltiplos protocolos de rádio ao mesmo tempo, bastando pequenas alterações em software para atender tal necessidade. Sistemas de operadoras de celular, por exemplo, se enquadram nesta categoria.

II. Por que usar SDR?

Além da flexibilidade, os SDRs possuem diversas outras vantagens sobre as implementações tradicionais de rádios. São soluções mais robustas a variações de temperatura e envelhecimento de seus componentes, visto que transfere o processamento para o domínio digital, deixando de ter seu desempenho atrelado à precisão dos componentes analógicos do rádio. Também possui toda a facilidade de desenvolvimento que um ambiente de software fornece como ferramentas para simulação e correção de erros. Por transferir ao software grande parte do processamento, reduz a complexidade do hardware dos rádios modernos, limitando-os apenas a implementação do *front-end* de RF. Isso também implica em alta integração, dada a eliminação, através do uso de um processador, de muitos dos elementos ativos e passivos do rádio, antes responsáveis pelo processamento e modulação do sinal. Todos esses fatores têm influência no custo final reduzido do produto, um SDR bastante popular, o SoftRock RXTX, custa apenas algumas dezenas de dólares americanos. Um comparativo entre diversos SDRs pode ser encontrado em [7].

Menor *time-to-market* é outra vantagem desse modelo de rádio, já que uma mesma plataforma pode ser rapidamente modificada para operar em diferentes padrões pela simples substituição do software. Questões de manutenção e operação dos equipamentos também são beneficiadas pela arquitetura do sistema. Correções de erros, adaptações ou aprimoramento dos serviços podem ser feitos sem que ele precise ser desligado ou sem que a infra-estrutura deva ser alterada , o que é muito interessante para prestadoras de serviços de radiodifusão.

III. Umexemplo de arquitetura SDR: o USRP

O Universal Software Radio Peripheral (USRP), Fig. 2, desenvolvido pela Ettus [8], é uma plataforma para o desenvolvimento de rádios definidos por software. A USRP permite o desenvolvimento de rádios digitais, fornecendo toda a infra-estrutura de processamento digital e de RF. O sistema caracteriza-se por sua alta flexibilidade e seu bom custo benefício. Nas seções a seguir, discutiremos em detalhe cada parte da arquitetura da USRP.

Figura 2. USRP *Universal Software Radio Peripheral*.

A. Antena

A antena de transmissão é o componente que efetivamente realiza a transmissão da informação. No USRP, mais de uma antena pode ser conectada ao dispositivo, de acordo com a necessidade do usuário. Poderíamos, por exemplo, conectar três antenas e transmitir três diferentes estações de rádio FM, cada uma em uma determinada freqüência. Também poderíamos realizar o procedimento inverso, recebendo sinais múltiplos de freqüências distintas. Outra possibilidade é a conexão simultânea de antenas para a transmissão e recepção de sinais, o que poderia ser utilizado, por exemplo, nos modernos sistemas MIMO de comunicações sem fio para a telefonia celular.

B. Placas-filhas

As placas-filhas implementam o *front-end* RF do SDR. Estas placas têm como função realizar a conversão de freqüência do sinal (da banda-base para a freqüência de transmissão ou o contrário). Além disso, também são as placas-filhas que determinam a potência final do sinal a ser emitido, que varia de 50 a 200mW. Algumas placas podem funcionar tanto para recepção quanto para a transmissão de sinais.

O USRP permite a conexão de diversas placas-filhas simultaneamente, criando assim os canais de recepção e transmissão do rádio digital, capazes de enviar sinais nas mais variadas bandas de freqüência. Algumas das placas-filhas disponíveis são a BasicTX e a BasicRX (transmissão e recepção, respectivamente, na faixa de 1 a 250 MHZ), a WBX

(transmissão, de 50 MHz a 2.2 GHz), dentre outras. Uma lista completa das placas está disponível em [8].

C. Placa-mãe

A placa-mãe do USRP N210 inclui conversores ADC e DAC, uma FPGA, uma interface ethernet gigabit para a comunicação com o computador, além de uma série de outros recursos, tais como interfaces seriais de alta velocidade para integração com outras placas, permitindo a elaboração de sistemas MIMO, recursos de relógio, entre outros.

A FPGA pode realizar todo ou parte do processamento digital do sinal, permitindo, caso configurado, transferir o processamento para outros dispositivos computacionais, como o PC, via ethernet. Na versão atual, o USRP utiliza a placa XilinxSpartan, enquanto que em versões anteriores, a Altera Cyclone era utilizada.

De modo geral, a FPGA se encarrega das etapas de banda-base (BB) para freqüência intermediária (IF) feita pelo DUC (*digital up converter*) e de IF para banda-base pelo DDC (*digital down converter*). Tais operações são realizadas por quatro conversores digitais com taxas de decimação e interpolação programáveis.

D. GNU Radio

GNU *Radio* é um conjunto de ferramentas de código aberto que provê um ambiente de desenvolvimento e blocos de processamento para implementar rádios definidos por software. Possui integração imediata com as placas USRPs através de *drivers* especialmente desenvolvidos para suportá-las [9].

As aplicações em GNU Radio são desenvolvidas utilizando a linguagem de programação Python, onde são construídas ligações entre os blocos de processamento. Os blocos, em si, são desenvolvidos em C++ por questões de desempenho. Há ainda uma interface gráfica para a montagem de sistemas, o GNU *Radio Companion* (GRC), muito similar ao conceito de desenvolvimento de simulações no simulink do Matlab [10].

Cada bloco do GNU *Radio* possui entradas e saídas que podem ser associadas aos mais variados tipos de fluxos de dados. A partir da conexão entre diferentes blocos básicos, forma-se um diagrama de fluxo de dados, que define as etapas de processamento do sinal de determinado sistema de comunicação. Além das configurações das entradas e das saídas, cada bloco possui um conjunto específico de parâmetros, que definem seu comportamento.

Por exemplo, um bloco que representa um filtro passa-banda apresenta como opções o tipo do filtro, o ganho, a taxa de amostragem, as freqüências de corte, a faixa de transição, dentre outras. Alguns blocos apresentam apenas entradas ou saídas de dados, como é o caso dos geradores de sinais (*sources*) e coletores de informação (*sinks*) de um sistema. O GNU Radio já fornece em torno de uma centena de blocos para os usos mais comuns, como operações matemáticas, conversão de tipo de dados, alguns tipos de modulação, dentre outros. Além destes blocos já disponibilizados pelo software, é possível criar outros de acordo com a necessidade do usuário [9].

Na linguagem Python, como mostrado no exemplo da Fig. 3, duas bibliotecas padrão do GNU Radio, a `gr` e a `audio`, são carregadas. Em seguida, define-se a taxa de amostragem do sistema e a amplitude, criando um grafo de fluxo que representa o sistema. Gera-se duas ondas senoidais, através da função `gr.sig_source_f`, uma de frequência 360 HZ, outra de freqüência 440 Hz. O *audiosink*, que servirá como saída para o sistema, recebe como entrada a soma das senóides. Pode-se então ouvir um som do tipo DTMF na saída da placa de som. Alternativamente ao uso de Python como linguagem para criar as ligações entre os blocos, pode-se utilizar o software GNU *Radio Companion* (GRC). Com o uso do GRC, é possível configurar os blocos e criar suas ligações de forma gráfica simples e intuitiva.

Como exemplo de um código simples para ilustrar o funcionamento do sistema, a Fig. 4 mostra um programa que, através do uso de blocos já disponibilizados pelo GNU Radio, gera duas senóides e as envia para a saída da placa de som do computador [10]. Este código utiliza apenas blocos do GNU Radio já existentes em sua biblioteca.

```
#!/usr/bin/env python
fromgnuradio import gr
fromgnuradio import audio
defbuild_graph ():
sampling_freq = 48000
ampl = 0.1
fg = gr.flow_graph ()
src0 = gr.sig_source_f (ampling_freq,
gr.GR_SIN_WAVE, 350, ampl)
src1 = gr.sig_source_f (sampling_freq,
gr.GR_SIN_WAVE, 440, ampl)
dst = audio.sink (sampling_freq)
fg.connect ((src0, 0), (dst, 0))
fg.connect ((src1, 0), (dst, 1))
returnfg
if __name__ == '__main__':
fg = build_graph ()
fg.start ()
raw_input ('Press Enter to quit: ')
fg.stop ()
```

Figura 3. Exemplo de código do GNU Radio.

Cada bloco recebe alguns parâmetros de configuração específicos para que possa funcionar corretamente. O USRP Source, por exemplo, recebe como parâmetros de configuração o tipo da saída, o endereço do dispositivo, a taxa de amostragem, a freqüência central, o ganho, a banda do sinal, dentre outros. A Fig. 5 exibe a janela de configuração do bloco. Nos exemplos citados até agora, faz-se uso de variáveis para definição de alguns parâmetros do sistema, como a taxa de amostragem, por exemplo.

Com o conjunto de blocos já fornecidos pelo GNU radio, um grande número de aplicações pode ser resolvido sem a necessidade de se criar blocos específicos, o que torna o desenvolvimento mais rápido e simples.

Um ponto que merece ser ressaltado é o fato do GNU Radio, por ser um projeto *open-source*, permite adaptação de seu código, caso necessário. Assim, usuários que demandem algum grau maior de customização podem adaptar o software de acordo com suas necessidades. Além disso, o GNU Radio conta com o suporte de toda a comunidade do GNU/Linux, o que torna fácil o acesso à informação e ajuda em fóruns. O sistema encontra-se em continuo desenvolvimento e

manutenção, devido ao interesse acadêmico e comercial, o que dificulta a possibilidade de descontinuação do projeto.

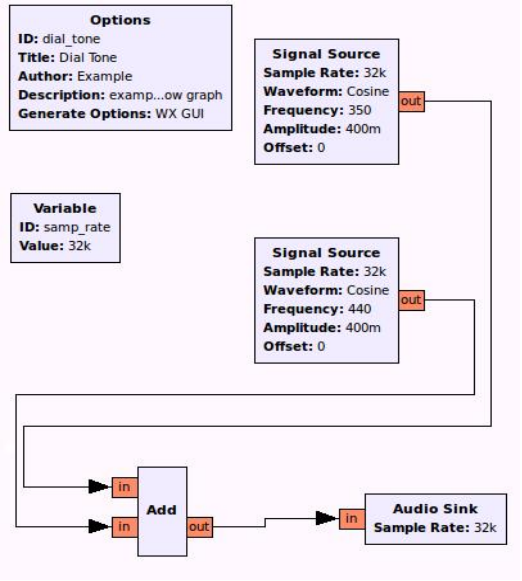

Figura 4. Exemplo de aplicação GRC.

IV. USRP E GNU RADIO

A união entre o USRP e o GNU Radio permite a criação de uma série de sistemas aplicáveis nas mais diversas áreas. Esta proposta de arquitetura mostra-se muito flexível, além de oferecer uma base sólida para o desenvolvimento das aplicações de processamento digital de sinais.

Como um exemplo mais sofisticado, a Fig. 6 mostra a implementação de um receptor FM desenvolvido com o uso do GRC. Neste exemplo, o USRP executa a recepção do sinal de RF, converte sua freqüência de RF para banda-base e envia esse sinal para o PC, onde é coletado pelo GRC para processamento. O sinal, então, é enviado por dois caminhos distintos: um *sink* que imprime na tela a FFT do sinal através do uso do método gráfico wxGUI, e um bloco de recepção de sinais FM, o WBFM *Receive*. A saída deste bloco irá, então, ser multiplicada para uma constante, e enviada para a placa de som através de um *AudioSink*.

Para ilustrar a flexibilidade do sistema, é apresentado outro exemplo na Fig. 7, que implementa um transmissor SSB. O uso em conjunto do GNU Radio e o USRP permite a criação de sistemas de um grande número de esquemas de comunicações em radiofrequência.

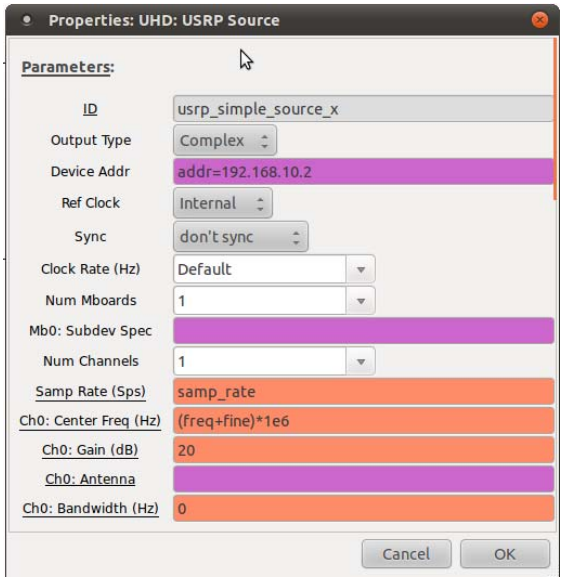

Figura 5. Janela de configuração de um bloco GRC.

Além de todas as vantagens que um SDR pode trazer, como a flexibilidade, a facilidade na manutenção e adaptação, o desenvolvimento de um rádio utilizando USRP e GNU *Radio* é simples, pois torna transparente ao desenvolvedor toda a complexidade de uma arquitetura SDR, deixando-o concentrar-se mais em algoritmos e menos em aspectos de integração e hardware. Tanto o desenvolvimento utilizando o GRC como o desenvolvimento em Python requerem pouco tempo para serem aprendidos, além de tornarem o projeto de rádios tão fácil quanto conectar blocos, reduzindo assim o tempo de desenvolvimento e *time-to-market* de um novo produto.

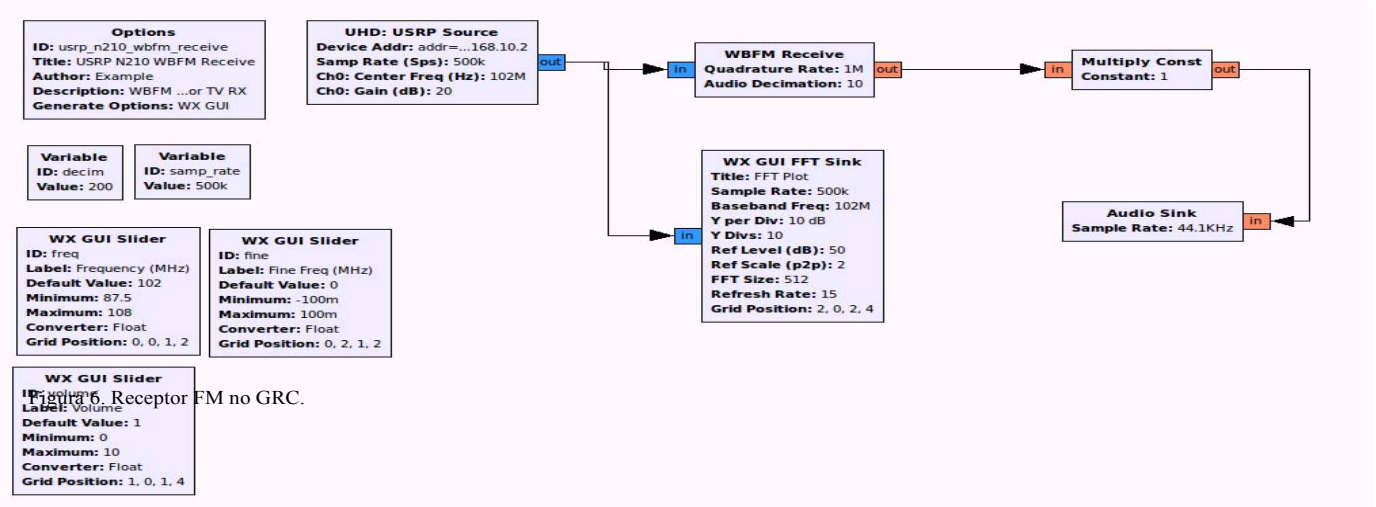

Figura 6. Receptor FM no GRC.

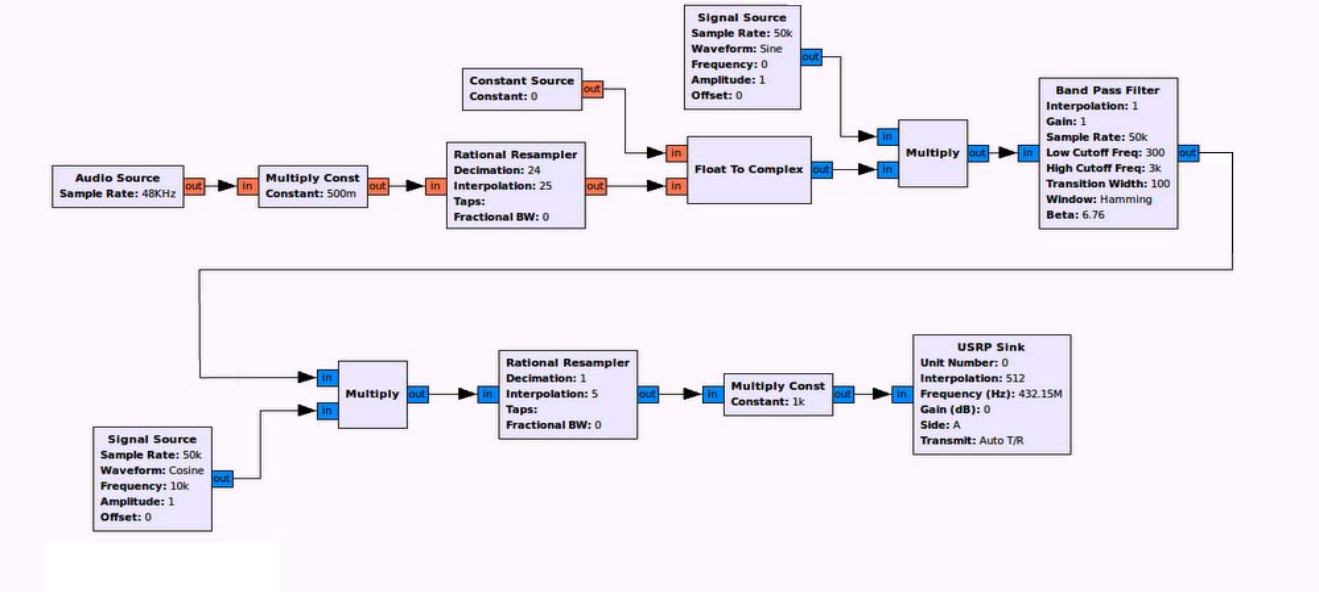

Figura7.Transmissor SSB no GRC.

Com o uso das ferramentas apresentadas, os detalhes de implementação física perdem importância na criação de novas soluções em radiofreqüência, priorizando o desenvolvimento de alto-nível, mais focado no algoritmo do que em sua arquitetura.

Dentre as diversas soluções já desenvolvidas com o uso do USRP e do GNU Radio, pode-se citar aplicações em MIMO [11] e o OpenBTS, aplicação Unix que fornece uma inteface de comunicação GSM [12]. O projeto visa criar uma nova forma de rede para telefones celulares, a um custo de aproximadamente um décimo das tecnologias atuais. O sistema provê, por exemplo, uma boa solução na implementação de redes de celulares em áreas rurais, onde as redes convencionais não possuem alcance.

Há também desenvolvimentos nas mais diversas áreas de comunicações, como na implementação de uma rede Bluetooth [13], na implementação de uma PLC (*Power Line Communications*) entre um motor e um conversor de freqüência [14], ou ainda as diversas aplicações na área de rádio cognitivo [6] e SDR [15].

V. Conclusão

Com o avanço das técnicas de processamento digital de sinais, cada vez mais os rádios tradicionais podem dar lugar aos rádios definidos por software, graças ao desenvolvimento ágil e a flexibilidade que possuem, adequando-se melhor à realidade atual do mercado.

Neste artigo, foi exibido uma das várias soluções em SDR: o uso do USRP em conjunto com o GNU Radio, uma forma rápida de se desenvolver *Software-DefinedRadio*. Amparada por um grande número de bibliotecas de processamento digital de sinal e de comunicações digitais, junto com *drivers* para integração imediata com as placas USRPs, essa combinação torna esse sistema uma ferramenta de projeto extremamente poderosa para o desenvolvimento de solução de rádio. A união das duas ferramentas vem sendo largamente utilizada e diversas soluções já vem sendo implementadas com seu uso.

Tal ferramenta é certamente muito útil, tanto para fins acadêmicos quanto para o desenvolvimento de produtos comerciais de telecomunicações, permitindo um *time-to-market* rápido e grande flexibilidade para suportar, em uma mesma arquitetura, diversas soluções para comunicação sem fio.

Agradecimentos

Este trabalho recebeu apoio da FINEP – Financiadora de Estudos e Projetos - Brasil, contrato no. 01.07.0110.00, “Avaliação de Middleware Ginga”.

Referências

[1] The Wireless Innovation Forum, http://www.wireless innovation.org.
[2] R.J. Lackey and D.W. Upmal, Speakeasy: The Military Software Radio, IEEE Communications Magazine, May 1995.
[3] Minh Nguyen, Software Radio (R)Evolution and Its Application to Aeronautical Mobile Communications, May 19-22, 2003.
[4] J. Mitola, Software radios-survey, critical evaluation and future directions, IEEE National Telesystems Conference, pages 13/15-13/23, May 1992.
[5] J. Mitola, Cognitive Radio: making software radios more personal. IEEE Personal Communications, Aug. 1999.
[6] Scaperoth, D., Cognitive Radio Platform Development for Interoperability Military Communications Conference, 2006.
[7] Scotty Cowling, Hands-On Software Defined Radio, 2008 Dayton Hamvention SDR Forum.
[8] Matt Ettus, Universal software radio peripheral. http://www.ettus.com.
[9] Eric Blossom, How to Write a Signal Processing Block, http://www.gnu.org/software/gnuradio/doc/howto-write-a-block.html.
[10] Eric Blossom, Exploring GNU Radio. http: //www.gnu.org/software/gnuradio/doc/exploring-gnuradio.html.
[11] Anna Kaszuba, MIMO Implementation with Alamouti Coding Using USRP2, Progress In Electromagnetics Research Symposium Proceedings 2011.
[12] The OpenBTSProjethttp://openbts.sourceforge.net.
[13] Dominic Spill, BlueSniff: Eve meets Alice and Bluetooth. WOOT '07 Proceedings of the first USENIX workshop on Offensive Technologies, 2007.

[14] A. Pinomaa, Utilization of Software-Defined Radio in Power Line Communication between Motor and Frequency Converter. IEEE International Symposium on Power Line Communications and Its Applications (ISPLC), 2010
[15] The FlexRadio FLEX-5000A™ Software Defined Radio,http://www.flex-radio.com.

André F. B. Selva is an undergraduate student at the School of Electrical and Computer Engineering – UNICAMP, in Brazil, coursing Computer Engineering. He has been working at the Real Time Digital Signal Processing Laboratoy, developing projects using Software Defined Radios and GNU Radio toolkit. His research interests are digital circuits design, digital signal processing, and communications.

André L. G. Reis is an undergraduate Computer Engineering student at School of Electrical and Computer Engineering - UNICAMP, in Brazil. He has been working at Real Time Digital Signal Processing Laboratory, developing projects using Software Defined Radio and GNU Radio toolkit. His research interests are telecommunications systems and digital signal processing.

Karlo G. Lenzi received a B.Sc. degree in computer engineering from Pontifícia Universidade Católica do Paraná (PUC-PR) in 2003 and a Msc. Degree in electronics from Instituto Tecnológico de Aeronautica (ITA) in 2006. He is currently working towards his PhD. degree in Telecommunications at Universidade Estadual de Campinas (UNICAMP). He is also a researcher at Centro de Pesquisa e Desenvolvimento – CPqD – Wireless Communication Department in Campinas, Brazil. His research interests are wireless communication, digital circuits design and digital signal processing.

Prof. Luís Geraldo Meloni, Ph.D, is Professor at the School of Electrical and Computer Engineering – UNICAMP, in Brazil. Graduated at the Université de Nancy I (1985), France; Master of Science (1982) under-graduation (1980) in Electrical Engineering, both at UNICAMP. He has a large academic and industrial experience in Telecommunications. He has been coordinating the works for the SBTVD Forum - Brazilian System for Digital Television - concerning technologies for Return Channel. He has been working in the Digital Signal Processing since 1982, and participated on several projects using digital signal processors and FPGA in academia and private companies. He was lecturer at University of Brasilia (1990-1993) and now at the State University of Campinas. At UNICAMP, he has been working at the Real Time Digital Signal Processing Laboratory in several fields: new technologies for wireless communications, software-defined radio, DTV middleware and return channel, speech and audio coding, voice over IP, speech recognition, development of software tools for education and learning via Internet. He has many publications in international journals and symposia and also has been instructor in continuing education programs for telecommunications professionals.

Silvio Ernesto Barbin, (M'1974, SM'2001) was Born in Campinas, SP - Brasil at May 04th, in 1952. He under-gradueted on Electric Engineering at EPUSP – Politechnical Scholl of São Paulo University in 1974, getting his master degree and PhD degree at the same institution. He was a visitor researcher at Califórnia University, Los Angeles, CA and professor at New México University, in Albuquerque, NM. He had worked at AEG –Telefunken for 4 years, in Backnang and Ulm, Germany, and for 9 years in São Paulo, SP Brasil. He was technical director of Microline Multiplexadores de RF. Nowadays, he is a professor of the Engineering of Communications and Control Department of the Politechnical School of São Paulo University and General R&D and Inovation Projects Coordinator and Substitute Director of CTI – Centro de Tecnologia da Informação Renato Archer of the Brazilian Science and Technology Ministry.