

Universidade Federal do ABC
Graduação em Engenharia de Informação

Henrique Santos Franco

**DESENVOLVIMENTO DE UMA REDE GSM EMPREGANDO A TÉCNICA DE SDR
PARA APLICAÇÕES EM SMART GRIDS**

Santo André – SP

2017

Henrique Santos Franco

**DESENVOLVIMENTO DE UMA REDE GSM EMPREGANDO A TÉCNICA DE SDR
PARA APLICAÇÕES EM SMART GRIDS**

Monografia apresentada ao curso de graduação da Universidade Federal do ABC,
como requisito parcial para obtenção do grau de Engenheiro de Informação

Orientador: Prof. Dr. Ivan Roberto Santana Casella

Santo André – SP

2017

Henrique Santos Franco

DESENVOLVIMENTO DE UMA REDE GSM EMPREGANDO A TÉCNICA DE SDR
PARA APLICAÇÕES EM SMART GRIDS / Henrique Santos Franco – Santo André, SP
2016

Orientador: Prof.Dr. Ivan Roberto Santana Casella

79. p.

Trabalho de Graduação – Universidade Federal do ABC – UFABC, 2016.

1. GSM. 2. SDR. 3. Smart Grids. 4. Controle Sem Fio. 5. OpenBTS. 6. GNU Radio.
I. Ivan Roberto Santana Casella. II. Universidade Federal do ABC. III. Centro de
Engenharia, Modelagem e Ciências Sociais Aplicadas. IV. Desenvolvimento de uma
Rede GSM Empregando a Técnica de SDR para Aplicações em *Smart Grids*.

DEDICATÓRIA

À minha família, pelo apoio e paciência.

À minha noiva, por sua compreensão.

A todos meus professores de graduação da

Universidade Federal do ABC.

Aos meus colegas de curso e estudo.

AGRADECIMENTOS

À Universidade Federal do ABC.

Ao orientador Prof. Dr. Ivan Roberto Santana Casella, pelo pronto auxílio e acompanhamento do projeto. Ao doutorando Fernando Simplicio, pelo suporte na fase final deste projeto.

Aos professores do curso de graduação de Engenharia de Informação.

A todos que direta ou indiretamente contribuíram para a realização desta monografia.

ΕΠΙΓΡΑΦΕ

“Success consists of going from failure to failure without loss of enthusiasm”

Winston Churchill.

RESUMO

Este trabalho de graduação estuda a utilização de redes GSM no envio de sinais de controle para turbinas de geradores eólicos em *Smart Grids*. Inicialmente foram efetuadas as instalações de *Softwares* necessários para utilizar as ferramentas abordadas neste trabalho. Testes de transmissão foram feitos com o GNU Radio e, por fim, foi feita a implementação completa de uma rede GSM por meio do OpenBTS e realizados vários testes através da rede para a transmissão de informações de controle de aerogeradores por meio de SMS. Foi verificado que o envio de mensagens de texto pode ser feito de modo satisfatório, apesar de existirem algumas limitações quanto ao tempo de envio e processamento dos SMS.

Palavras-chave: GSM, SDR, *Smart Grids*, Controle Sem Fio, OpenBTS, GNU Radio.

ABSTRACT

This graduation work studies the use of GSM networks on controlling the delivered power from wind turbines on Smart Grids. Firstly, all necessary softwares were installed in order to use all the tools approached. Transmission tests were performed using GNU Radio and, lastly, a complete implementation of a GSM network with OpenBTS was performed and many tests were carried sending SMS with control information for wind turbines. It was tested that sending SMS messages is feasible, being adequate for controlling voltage levels. However, it was also verified that there are some limitations in the sending time.

Keyword: GSM, SDR, Smart Grids, Wireless Control, OpenBTS, GNU Radio.

LISTA DE FIGURAS

Figura 1: Arquitetura de uma rede GSM [10].....	22
Figura 2: Multiplexação temporal e espectral realizada pela interface de rádio GSM [10].	24
Figura 3: Exemplo de sinal MSK [11].	25
Figura 4: (a) Rádio tradicional em contraste com o (b) SDR [23].....	27
Figura 5: Imagem do RTL-SDR utilizado.....	28
Figura 6: Imagem da USRP E110.....	30
Figura 7: Imagem da BladeRF.	31
Figura 8: Comparação entre fluxos em uma rede tradicional e uma SG (SIEMENS – Smart Grid Division, 2014) [29].	32
Figura 9: Comunicação sem fio para turbinas eólicas [29].....	33
Figura 10: Dispositivos utilizados nos testes de GSM.....	36
Figura 11: Antenas monopolo utilizadas na BladeRF.....	36
Figura 12: <i>SIM Cards</i> utilizados nos testes.	37
Figura 13: Saída do comando ' <i>dmmsg</i> ' após a conexão da USRP E110.	39
Figura 14: Tela de início do sistema operacional Angstrom.....	40
Figura 15: Janela do programa de teste para o RTL-SDR.....	41
Figura 16: Resultado do teste de ' <i>tx_waveforms</i> '......	42
Figura 17: Área de trabalho do <i>AngstromOS</i>	43
Figura 18: Programa de teste do GNR na USRP E110.....	44
Figura 19: Resultado do teste com o GNR.....	45
Figura 20: Diagrama de bloco do GNR utilizado no transmissor.....	45
Figura 21: FFT e diagrama de constelação do transmissor GMSK.....	46
Figura 22: Diagrama de bloco do GNR utilizado no receptor.	47
Figura 23: FFT e diagrama de constelação do receptor GMSK.	48
Figura 24: Autorização do <i>SIM Card</i> e sinal de recepção.	50
Figura 25: Resultado da busca de redes.....	51
Figura 26: Resultado do teste de envio de SMS a partir do <i>Short Code</i> 4000.	52
Figura 27: <i>EasyPIC v7</i> , utilizada para os testes.	53
Figura 28: Imagem aproximada do módulo <i>Telit GL865 QUAD</i> conectada à placa. .	54
Figura 29: Saída no osciloscópio para a mensagem 013588531.....	56
Figura 30: Foto do osciloscópio exibindo a curva dos valores enviados por SMS. ...	56
Figura 31: Onda quadrada utilizando a mensagem: 000008888.....	57

Figura 32: Instante de disparo e tempo decorrido até recebimento para envios espaçados de 8 segundos.....	58
Figura 33: Instante de disparo e tempo decorrido até recebimento para envios espaçados de 10 segundos.....	59
Figura 34: Instante de disparo e tempo decorrido até recebimento para envios espaçados de 15 segundos.....	60
Figura 35: Instante de disparo e tempo decorrido até recebimento para envios espaçados de 20 segundos.....	62

LISTA DE TABELAS

Tabela 1: Níveis de tensão disponíveis no sistema proposto.....	55
Tabela 2: Tempo entre envio e recebimento para envios espaçados de 8 segundos.	59
Tabela 3: Tempo entre envio e recebimento para envios espaçados de 10 segundos.	60
Tabela 4: Tempo entre envio e recebimento para envios espaçados de 15 segundos.	61
Tabela 5: Tempo entre envio e recebimento para envios espaçados de 20 segundos.	62
Tabela 6: Tempo de recebimento considerando o envio a partir de um celular.	63

LISTA DE SIGLAS

ADC	<i>Analog-to-Digital Converter</i>
AM	<i>Amplitude Modulation</i>
AMI	<i>Advanced Metering Infrastructure</i>
AMPS	<i>Advanced Mobile Phone Service</i>
BSC	<i>Base Station Controller</i>
BSS	<i>Base Station Subsystem</i>
BTS	<i>Base Transceiver Station</i>
CSFB	<i>Circuit-Switched Fallback</i>
DAC	<i>Digital-to-Analog Converter</i>
DSL	<i>Digital Subscriber Line</i>
EDGE	<i>Enhanced Data Rates for GSM Evolution</i>
ePDG	<i>Evolved Packet Data Gateway</i>
E-UTRAN	<i>Evolved UMTS Terrestrial Radio Access Network</i>
FDD	<i>Frequency Division Duplex</i>
FDMA	<i>Frequency Division Multiple Access</i>
FFT	<i>Fast Fourier Transform</i>
FM	<i>Frequency Modulation</i>
FPGA	<i>Field Programming Gate Array</i>
GIGE	<i>Geradores de Indução com Rotor Bobinado</i>
GGSN	<i>Gateway GPRS Support Node</i>
GMSC	<i>Gateway MSC</i>
GPRS	<i>General Packet Radio Service</i>
GRC	<i>GNURadio Companion</i>
GSM	<i>Global System for Mobile Communications</i>
HLR	<i>Home Location Register</i>
HSS	<i>Home Subscriber Server</i>
IMEI	<i>International Mobile Equipment Identity</i>
IMS	<i>IP Multimedia Subsystem</i>
IMSI	<i>International Mobile Subscriber Identity</i>
IF	<i>Intermediate Frequency</i>
IP	<i>Internet Protocol</i>
LTE	<i>Long Term Evolution</i>

MIMO	<i>Multiple-input Multiple-output</i>
MME	<i>Mobile Management Entity</i>
MCC	<i>Mobile Country Code</i>
MNC	<i>Mobile Network Code</i>
MS	<i>Mobile Station</i>
MSC	<i>Mobile Switching Center</i>
MTSO	<i>Mobile Telephone Switching Office</i>
OFDM	<i>Orthogonal Frequency Division Multiplexing</i>
PCRF	<i>Policy and Charging Control Function</i>
PGW	<i>Packet Data Network Gateway</i>
PSK	<i>Phase-Shift Keying</i>
PSTN	<i>Public Switched Telephone Network</i>
QAM	<i>Quadrature Amplitude Modulation</i>
QPSK	<i>Quad Phase-Shift Keying</i>
RF	<i>Radio Frequency</i>
RFID	<i>Radio-Frequency Identification</i>
RNC	<i>Radio Network Controller</i>
SC-FDMA	<i>Single-Carrier FDMA</i>
SGW	<i>Serving Gateway</i>
SDR	<i>Software Defined Radio</i>
SIM	<i>Subscriber Identity Module</i>
SIP	<i>Session Initiation Protocol</i>
SMS	<i>Short Message Service</i>
TDD	<i>Time Division Duplex</i>
TDMA	<i>Time Division Multiple Access</i>
TE	<i>Terminal</i>
UMTS	<i>Universal Mobile Telecommunication System</i>
USRP	<i>Universal Software Radio Peripheral</i>
UTRAN	<i>UMTS Terrestrial Radio Access Network</i>
VLR	<i>Visitor Location Register</i>
WCDMA	<i>Wide-Band Code-Division Multiple Access</i>

SUMÁRIO

1. INTRODUÇÃO	15
1.1 Justificativa	18
1.2 Objetivos.....	19
2. SISTEMAS DE TELEFONIA MÓVEL.....	20
2.1 Tecnologia GSM.....	22
2.1.1 Arquitetura.....	22
2.1.2 Interface de Rádio.....	24
3. A TECNOLOGIA DE SDR.....	27
3.1 O RTL-SDR	28
3.2 A USRP E110.....	29
3.3 A BladeRF	30
4. USO DA COMUNICAÇÃO MÓVEL EM <i>SMART GRIDS</i>	32
5. ANÁLISE DO SISTEMA GSM DESENVOLVIDO PARA CONTROLE DE AEROGERADORES	34
5.1 Configuração do Ambiente de Trabalho	34
5.2 Análise da Interface Física do GSM Empregando o GNR	38
5.3 Análise da Interface de Rede do GSM Empregando o OpenBTS	49
5.4 Análise da Aplicação da OpenBTS em <i>Smart Grids</i>	53
6. CONCLUSÕES	65
REFERÊNCIAS BIBLIOGRÁFICAS	66
APÊNDICE A - Instalação do GNR via <i>Pybombs</i>	70
APÊNDICE B - Instalação do OpenBTS	73

1. INTRODUÇÃO

No mundo atual, as redes de telefonia móvel são utilizadas em propósitos além dos originais aos quais foram idealizadas. O padrão GSM (*Global System for Mobile Communications*) foi criado para contemplar de modo satisfatório a função de efetuar chamadas entre usuários. Posteriormente, o GPRS (*General Packet Radio Service*) e o EDGE (*Enhanced Data Rates for GSM Evolution*) foram introduzidos ao GSM, possibilitando o acesso à internet. As gerações de telefonia móvel digital que vieram posteriormente, como o WCDMA (*Wideband Code Division Multiple Access*) e LTE (*Long Term Evolution*) melhoraram ainda mais a transmissão de dados, incorporando mais elementos de redes IP (*Internet Protocol*) aos sistemas de comunicações móveis [1].

Com a transmissão de dados por meio de redes móveis, as possibilidades se tornaram múltiplas e, através de *Smartphones*, tarefas como ir ao banco, responder e-mails e ler notícias se tornaram bem mais simples.

A crescente utilização das redes móveis foi acompanhada de grande desenvolvimento tecnológico. Atualmente, é possível simular e desenvolver sistemas de comunicação móveis dentro de uma sala, por meio de dispositivos específicos para este propósito, como será apresentado neste trabalho.

Um exemplo é o SDR (*Software Defined Radio*), um sistema de rádio comunicação na qual os componentes tipicamente implementados em *Hardware*, como filtros, amplificadores, moduladores, são implementados em *Software*. Não existe uma definição única de quais componentes devem pertencer ao *Software* ou *Hardware* em um sistema SDR, ou seja, um determinado SDR pode conter elementos implementados em *Hardware* que um outro sistema de SDR possui implementados em *Software* [2].

Este sistema é equipado com antenas de transmissão e recepção, além de uma interface com o computador serial ou USB (*Universal Serial Bus*), por exemplo, possibilitando a comunicação com outros dispositivos móveis, como o celular. No entanto, para que qualquer comunicação seja possível, é preciso programar o SDR. Por exemplo, se quisermos que o SDR seja um receptor de rádio FM (*Frequency Modulation*), deve-se programá-lo para receber os sinais da estação de rádio FM desejada, em seguida demodular o sinal e somente então interpretar os dados recebidos. Desse modo, percebe-se que dispositivos SDR possuem grande

versatilidade, pois não foram construídos com um propósito único de ser um transmissor GSM, ou um roteador Wi-Fi (*Wireless Fidelity*). São dispositivos capazes de operar como um ou o outro, não possuindo filtros ou moduladores eletrônicos, em geral. Todos os elementos são programáveis.

Existem diversas abordagens para desenvolvimento de *Softwares* em SDR. Uma possibilidade é o GNR (*GNU Radio*), um ambiente de desenvolvimento que possibilita a representação de sistemas de SDR modularmente, sendo composto por uma biblioteca de blocos de processamento que, quando conectados adequadamente, podem formar sistemas de processamento de sinal complexos [3]. O GNR possibilita desde simulações computacionais até a implementação real de sistemas de SDR com o auxílio de *Hardware*s específicos.

O GNR, apesar de conter uma grande biblioteca, não apresenta nenhum padrão de comunicação específico implementado, como, por exemplo, o 802.11, protocolo do *Wi-Fi*. Para que seja possível utilizar padrões completos de comunicação em um sistema de SDR, é necessária a utilização de outras ferramentas como, por exemplo, o OpenBTS (*Open Base Transceiver Station*) [30].

O OpenBTS é um projeto *Open-Source* de uma aplicação *Unix* que por meio de um USRP (*Universal Software Radio Peripheral*), um sistema de SDR, cria uma interface de rádio GSM para dispositivos móveis. Esta ferramenta utiliza o sistema de SDR para transmitir e receber sinais de GSM e tem como base o GNR. Tal qual uma rede GSM real, é possível estabelecer chamadas entre dois dispositivos conectados, assim como enviar e receber SMS (*Short Message Service*) e acessar a *Internet* por GPRS.

Existem diversas aplicações para o OpenBTS, dentre elas, pode-se citar a possibilidade de uma universidade poder simular uma rede móvel e, assim, utilizá-la para complementar o ensino dos alunos com exemplos práticos, além de fazê-lo com um custo inferior ao de utilizar equipamentos comerciais [4]. Pode-se também efetuar medições de parâmetros na rede móvel, como taxas de erros em determinadas condições, ou até mesmo desenvolver novas tecnologias como o caso do trabalho apresentado por Yuva Kumar [5], na qual o OpenBTS é utilizado para estudar redes cognitivas, que são redes programadas para utilizar o espectro de modo mais eficiente.

Neste projeto, no entanto, explorou-se outra aplicação desta ferramenta. Atualmente, novas demandas têm surgido de redes de distribuição de energia

elétrica, como redução de custo, de consumo de energia ou de emissão de gases poluentes [6]. Para suprir essas necessidades, um novo modelo de rede, chamado de *Smart Grid* (SG), foi proposto. Para que seja possível implementar uma SG são necessárias uma infraestrutura avançada de medições (*Advanced Metering Infrastructure*, AMI), a integração de tecnologias de comunicação para que seja possível coletar dados desta rede de distribuição e tecnologias de controle para os equipamentos de geração e distribuição de energia envolvidos.

A redução de emissão de gases poluentes está diretamente ligada à utilização de fontes renováveis de energia, como por exemplo, energia eólica. A geração de energia eólica é realizada por meio de aerogeradores. A potência elétrica gerada depende do vento e, desse modo, pode variar bastante ao longo do tempo [6].

No entanto, essa variação na geração de potência não é interessante e pode causar problemas. Geração excessiva de energia induz a perdas, ou necessidade de armazenamento, algo que encarece a infraestrutura da rede de distribuição elétrica. Com o intuito de evitar tais perdas e custos, aerogeradores mais modernos possibilitam controlar a forma que aproveitam a utilização da energia proveniente do vento, ou seja, permitem controlar a potência gerada [6].

Nota-se, portanto, a necessidade de efetuar medições assim como controlar os aerogeradores adequadamente. A rede de comunicação necessária para que estas ações sejam tomadas é onde se insere a utilização de sistemas de SDR e o OpenBTS.

A rede de comunicação envolvida em um grande grupo de aerogeradores poderia ser realizada por meio de uma rede cabeada, no entanto isso demandaria altos custos de implementação e manutenção da infraestrutura [8]. Desse modo, a utilização de redes sem fio se torna vantajosa.

Entretanto, redes móveis demandam gastos referentes ao alto custo de equipamentos comerciais. Uma alternativa para redução de custos neste caso é a utilização de sistemas de SDR que, como demonstrado anteriormente, são sistemas de baixo custo. A utilização de tecnologias móveis, como GSM e GPRS, para SG, pode ser justificada pelas vantagens trazidas por estas tecnologias. Redes móveis providenciam uma boa largura de banda para transmissão de dados, os dados são

transmitidos de forma segura e podem operar em grandes áreas, algo essencial para geração de energia eólica [8].

Além disso, a escolha específica do GSM e GPRS se dá pelo fato de serem as tecnologias mais difundidas e, portanto, de mais fácil acesso, sendo possível até mesmo utilizar as redes já implantadas das operadoras de telefonia móvel [8]. Deve-se lembrar também que, mesmo utilizando uma infraestrutura de SDR, os equipamentos para transmissão e recepção próprios para cada tecnologia móvel são necessários. Nesse ponto, uma tecnologia mais difundida como o GSM ainda obteria vantagem de custo na implementação da SG.

1.1 Justificativa

O estudo de sistemas de SDR abre caminho não somente para uma melhor compreensão de redes móveis e sistemas de comunicação via rádio, como também possibilita o desenvolvimento de novos produtos ou tecnologias, como casas inteligentes [9], sistemas de radar ou leitores de RFID (*Radio Frequency Identification*).

Por meio de dispositivos de SDR e também de ferramentas como o GNR, pode-se desenvolver e simular sistemas de comunicação, resultando em testes mais completos do que uma simulação computacional. O fato de um dispositivo de SDR permitir que os elementos antes possíveis somente por *Hardware* (filtros, demoduladores, etc) sejam implementados via *Software* faz com que o trabalho do desenvolvedor seja dinâmico e resultados significativos sejam gerados com maior rapidez. Portanto, o estudo desta tecnologia possibilita o contato com uma ferramenta de desenvolvimento que pode ser utilizada em trabalhos futuros na qual exista a necessidade de simular ou desenvolver sistemas de comunicação.

Por fim, além de estudar a base do desenvolvimento em dispositivos SDR, este trabalho visa testar a viabilidade de uma solução SG para controle de potência de turbinas eólicas, relevante para uma geração de energia elétrica mais eficiente, robusta e de menor custo.

1.2 Objetivos

Este trabalho de graduação tem por objetivo principal o estudo de ferramentas de simulação e desenvolvimento de sistemas de comunicação. Para isto, ferramentas como o GNR e sistemas de SDR são utilizados, possibilitando uma abordagem mais completa de um sistema de comunicação.

O objetivo específico deste trabalho consiste em empregar as ferramentas de simulação e desenvolvimento de sistemas de comunicação citadas empregando o conceito de SGs para geração de energia eólica mais eficiente, por meio do controle da potência ativa e reativa de turbinas eólicas.

2. SISTEMAS DE TELEFONIA MÓVEL

Os sistemas de telefonia móvel são geralmente referenciados por suas gerações. Os sistemas em uso atualmente consistem nos sistemas de segunda geração (2G), terceira geração (3G) e quarta geração (4G). Tal qual a sequência abordada usualmente, os sistemas em si consistem em evoluções progressivas e de acordo com a demanda dos usuários. Pouco se fala dos sistemas de primeira geração (1G), pois estes foram totalmente substituídos pela tecnologia de segunda geração (2G). Dos sistemas 2G em diante, são todos digitais e agentes importantes na evolução da *Internet*.

Todos os sistemas de telefonia móvel 1G forneciam serviços de voz baseados em transmissão analógica de dados. Os padrões 1G utilizavam *Frequency Division Multiple Access* (FDMA) e comutação de circuitos em sua arquitetura de rede [9].

Um exemplo de tecnologia 1G é o *Advanced Mobile Phone Service* (AMPS). Este padrão foi desenvolvido nos Estados Unidos pela *Bell Labs* e utilizado comercialmente pela primeira vez em 1983 [13].

Este padrão foi um dos primeiros a utilizar o conceito de célula, utilizado até hoje nas tecnologias de telefonia celular.

O conceito de célula consiste em dividir a área de cobertura em regiões menores, denominadas células, cada uma com uma torre para transmitir e receber chamadas, que permite a conexão de dispositivos àquela célula. Este conceito demonstra que o AMPS pode ser aplicado em larga escala, cobrindo grandes áreas devido à possibilidade de reutilizar frequências após certa distância entre as células. Além disso, a arquitetura de uma rede AMPS consiste basicamente nos transmissores e receptores, estações móveis (MS) e centrais de comutação chamadas de *Mobile Telephone Switching Office* (MTSO).

O 2G se diferencia do 1G, principalmente, por ser totalmente digital. Abaixo, algumas vantagens dos sistemas de telefonia digital são citadas:

- Os dados, por estarem em um formato digital, podem ser facilmente criptografados, garantindo privacidade e segurança aos usuários;
- Sinais analógicos são mais susceptíveis a interferências, levando a uma qualidade de chamadas altamente variável. Em sistemas digitais, é possível aplicar detecção e correção de erros para tornar a comunicação mais robusta;

Um dos padrões de 2G mais difundidos em todo o mundo é o GSM. Ele foi o primeiro a introduzir um *Smart Card*, o SIM (*Subscriber Identity Module*), responsável por armazenar a identidade do usuário na rede, além de permitir a utilização de criptografia. Neste sistema também foi introduzida a troca de mensagens de texto entre dispositivos, o SMS (*Short Message Service*).

Posteriormente, como uma transição entre as tecnologias de segunda e terceira geração, surgiram as tecnologias de 2.5G. Estas, por sua vez, introduziram padrões de *Internet* móvel, inicialmente o GPRS e posteriormente o EDGE [12].

O GPRS apresentou taxas de transmissão de dados teóricas de 170kbps [8], utilizando a modulação original do GSM, o *Gaussian Minimum Shift-Keying* (GMSK). O EDGE (*Enhanced Data Rates for GSM Evolution*) utiliza um esquema de modulação 8PSK (*8 Phase Shift-Keying*), permitindo taxas teóricas de 384 kbps, maiores que as oferecidas pelo GPRS.

Os sistemas de 3G vieram com a proposta de fornecer taxas mais altas de transmissão de dados [10]. Um destes sistemas, o UMTS (*Universal Mobile Telecommunication System*), foi construído como uma evolução do GSM. O GSM, um sistema baseado em comutação de circuitos, uma vez que foi projetado para chamadas, era um sistema limitado para tráfego de *Internet*, baseada em pacotes. O UMTS, resolvendo este problema, inclui um sistema de comutação de circuitos para chamadas e também comutação de pacotes, para tráfego de dados [1].

Os sistemas de 4G, de forma semelhante aos sistemas de 3G, procuram melhorar o sistema de transmissão de dados. O LTE (*Long Term Evolution*) é um sistema 4G que segue como progressão do UMTS [10]. Um grande diferencial do LTE para o GSM e UMTS é que deixa de existir a comutação por circuitos. O LTE é totalmente baseado em comutação de pacotes, incluindo as chamadas telefônicas [1]. No entanto, o recurso de chamadas telefônicas em redes LTE, chamado VoLTE (*Voice over LTE*), não é tão utilizado. Muitas redes LTE dependem do GSM e UMTS para efetuar chamadas de voz, por meio de um *Fallback* no momento em que o usuário deseja efetuar a chamada, o chamado *Circuit-Switched Fallback* (CSFB) [1].

Neste trabalho de graduação foi utilizada a rede GSM na interface aérea de comunicação. Conforme apresentado, é um sistema que possui compatibilidade com as redes mais modernas e apresenta diversos recursos que podem ser utilizados na comunicação com outros dispositivos, como SMS e GPRS. Devido à sua importância

neste trabalho, as próximas seções procuram detalhar o funcionamento deste sistema.

2.1 Tecnologia GSM

As diferentes características da tecnologia GSM são analisadas nesta seção. Serão cobertas, basicamente, a arquitetura, apresentando os componentes de uma rede GSM e a interface de rádio, explicando como é feita a multiplexação, codificação e modulação dos dados.

2.1.1 Arquitetura

A arquitetura de uma rede GSM é composta por um conjunto de entidades, cada uma responsável por um conjunto de tarefas [10]. A Figura 1 apresenta a estrutura de uma rede GSM.

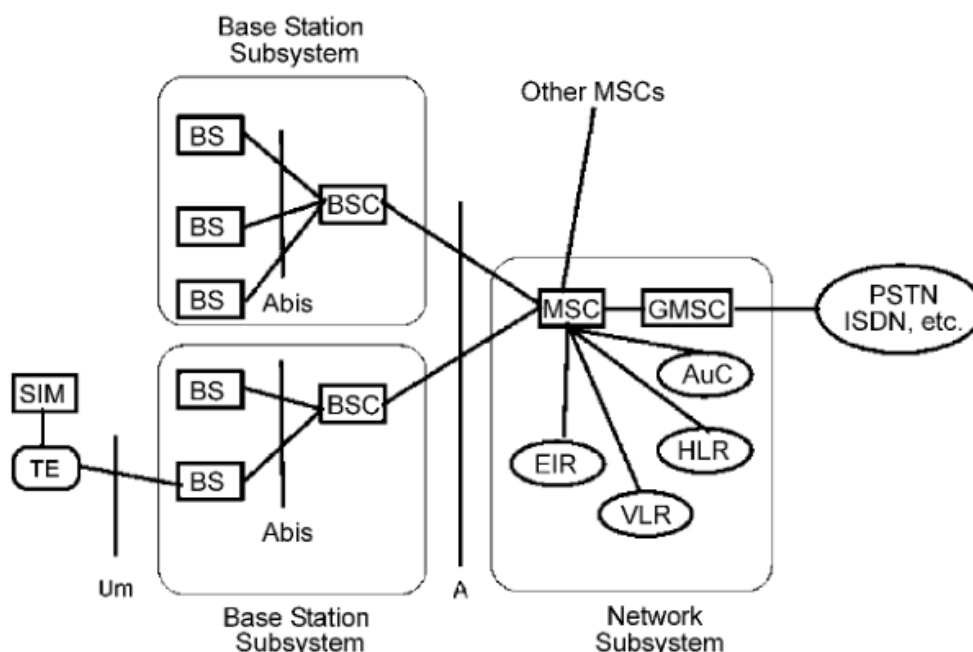


Figura 1: Arquitetura de uma rede GSM [10].

- *Mobile Station*

A *Mobile Station* (MS) é basicamente o dispositivo móvel, ou celular, composto pelo TE (*terminal*) e um *Smart Card* chamado *Subscriber Identity Module* (SIM). O SIM Card garante a mobilidade do usuário por conter a identidade do usuário na rede. O SIM Card armazena o *International Mobile Subscriber Identity*

(IMSI) usado para identificar o usuário na rede, permitindo que mediante a troca do terminal, a identidade do usuário permaneça a mesma. Além disso, o *SIM Card* contém uma chave de criptografia, *Kc (Ciphering Key)*, disponível somente ao próprio *SIM Card* e à operadora de telefonia móvel. Essa chave é necessária para que a comunicação criptografada no GSM seja possível [10, 11].

O terminal também possui uma identificação própria, o *IMEI (International Mobile Equipment Identity)*. Este número pode ser dividido em partes, de modo que seja identificado por fabricante, modelo e número de série, um código para identificar unicamente o dispositivo e um dígito de verificação.

- *Base Station Subsystem*

Esta parte da rede é composta pela BS ou BTS (*Base Station* ou *Base Transceiver Station*) e o BSC (*Base Station Controller*). Estes elementos basicamente representam a interface entre a MS e a rede. A BTS corresponde à interface direta da MS com a rede, pois este é o conjunto de transmissores e receptores que compõem a rede de acesso.

O BSC corresponde ao próximo estágio da rede controlando um grupo de BTSs. Uma de suas funções, por exemplo, é controlar o *Handover*, ou a troca, de um MS entre BTSs [10, 11].

- *Network Subsystem*

Pode-se dizer que este subsistema é responsável pelas tarefas de controle da rede. O MSC (*Mobile Switching Center*) é o componente central desta arquitetura. Tal componente é responsável por fechar os circuitos para chamadas, assim como providenciar as funcionalidades de autenticação, registro, atualização de localização os usuários, entre outras. O MSC também é responsável por, em conjunto com o GMSC (*Gateway MSC*) conectar à rede móvel à rede fixa, como a PSTN (*Public Switched Telephone Network*).

O VLR (*Visitor Location Register*) e HLR (*Home Location Register*) armazenam as informações sobre os usuários. O HLR atua como um banco central, contendo todas as informações administrativas do usuário. O VLR controla uma região, contendo então informações associadas somente aos dispositivos que estão na sua área de cobertura. Essas informações vão além das informações contidas no HLR. O VLR armazena, por exemplo, a localização dos MSs conectados à sua

cobertura de BTSs. O VLR normalmente é associado a um MSC, que por sua vez está associado a um conjunto de BSCs que por sua vez estão associados, cada, a um conjunto de BTSs. Desse modo, é possível notar uma relação hierárquica na arquitetura das redes GSM [10, 11].

2.1.2 Interface de Rádio

Um ponto importante para o desenvolvimento deste trabalho de graduação é conhecer a interface de rádio do GSM devido à intenção de se utilizar um dispositivo SDR para controlar diversas estações (neste caso aerogeradores).

A interface de rádio do GSM é multiplexada de duas formas diferentes: *Time Division Multiple Access* (TDMA) e FDMA. A faixa de frequência destinada ao GSM é dividida em duas subbandas: *Uplink* e *Downlink*. O *Downlink* é responsável pela comunicação no sentido BTS para MS e o *Uplink* no sentido contrário, MS para BTS. Cada subfaixa é dividida em canais de frequência através da técnica FDMA e cada canal de frequência é dividido em 8 *Slots* de tempo empregando a técnica TDMA. Deste modo, a transmissão de voz de cada usuário ocupa um *Slot* de tempo em um canal de frequência. A Figura 2 apresenta uma ideia gráfica das divisões feitas dentro da interface de rádio do GSM [10, 11].

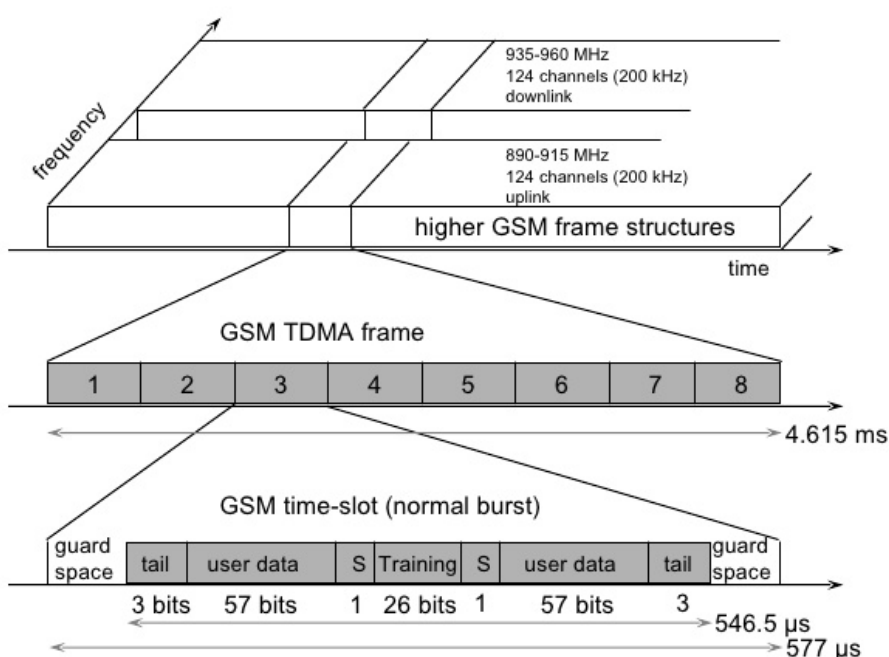


Figura 2: Multiplexação temporal e espectral realizada pela interface de rádio GSM [10].

Para proteger os dados transmitidos no GSM, é utilizada codificação convolucional e *Block Interleaving* [10]. A codificação convolucional adiciona bits a uma sequência, permitindo que durante a decodificação sejam ampliadas as chances de identificar e corrigir erros. O *Block Interleaving* altera a sequência com que os dados são transmitidos. Com o desvanecimento do canal de comunicação, bits muito próximos tendem a apresentar erros. Códigos de correção de erro muitas vezes não detectam e corrigem muitos erros próximos. Alterando a sequência que os dados são enviados, como no *Block Interleaving*, os bits que sofrem erros sob desvanecimento não são próximos, aumentando as chances de correção dos erros [10].

Conforme a Figura 2, cada canal do GSM possui uma largura de banda maior que os 200kHz disponíveis no padrão GSM. O esquema de modulação utilizado é o GMSK, que pode alcançar velocidades de até 270.8kbps utilizando todos os 200kHz de um único canal.

O GMSK é uma modulação em variação de fase, neste caso o *Minimum Shift Keying* (MSK), em que os valores binários 0 e 1 são representados por dois sinais senoidais, sendo um o dobro da frequência do outro, conforme a Figura 3 [11].

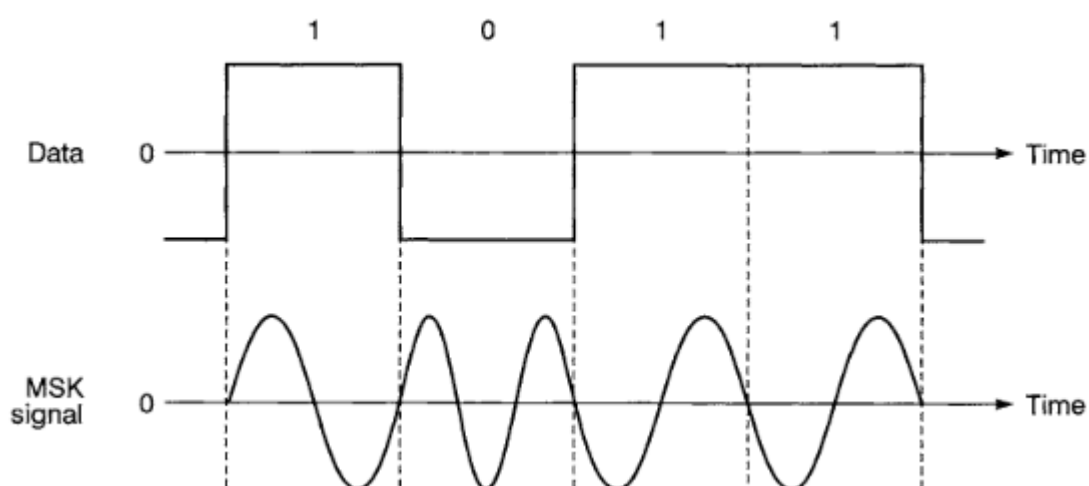


Figura 3: Exemplo de sinal MSK [11].

No entanto, o sinal apresentado ocupa uma largura de banda alta, o que é indesejado. Para resolver este problema filtros podem ser utilizados. O filtro proposto é o Gaussiano, que suaviza as transições entre os bits, resultando em um sinal que

ocupa uma largura de banda inferior. O sinal resultante desta junção de filtro Gaussiano e MSK é o GMSK [11].

3. A TECNOLOGIA DE SDR

Tradicionalmente, um sistema de RF (Radiofrequência) é um conjunto de equipamentos de *Hardware* com funções específicas. Ou seja, é construído para trabalhar em uma determinada faixa de frequência, utilizando esquemas de modulação específicos e, muitas vezes, com protocolos específicos de comunicação. Qualquer mudança requer que o *Hardware* seja modificado [24].

Para resolver as dificuldades apresentadas acima foi introduzido o conceito de SDR, na qual funções antes específicas e realizadas em *Hardware*, como a demodulação, podem ser realizadas em *Software* e se tornam, portanto, reconfiguráveis. A Figura 4 mostra um comparativo entre um sistema de rádio tradicional e um sistema de SDR [23].

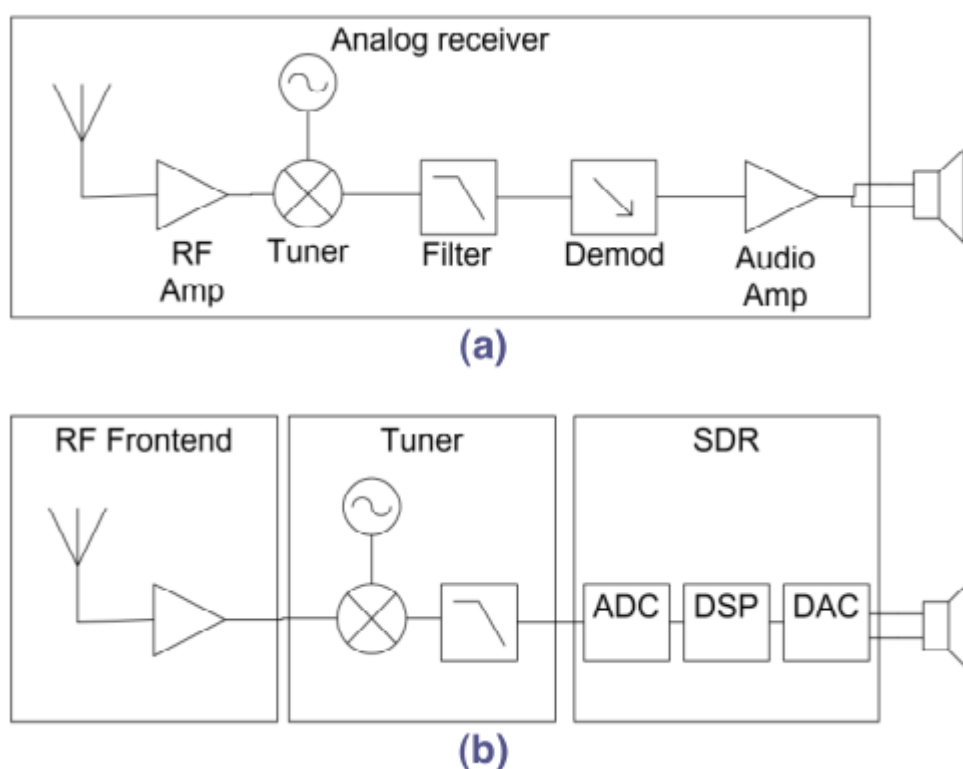


Figura 4: (a) Rádio tradicional em contraste com o (b) SDR [23].

Na Figura 4, um receptor analógico pode ser representado com todos os seus elementos, desde a recepção até o processamento, como parte integrada do *Hardware*. Em (b), representando a estrutura de um sistema de SDR, as antenas, amplificadores e sintonizadores fazem parte do *Hardware* deste tipo de sistema. O

processamento e conversão analógico digital (ADC) e digital para analógico (DAC) são realizadas por *Software*. Como o processamento é realizado por *Software*, a modulação e codificação dos sinais não dependem da estrutura física do dispositivo. Assim, estes aspectos do sistema de comunicação são programáveis [23]. Esse modelo permite que um único dispositivo possa ser utilizado para diferentes aplicações. Pode-se tornar o dispositivo uma estação de rádio FM, um roteador Wi-Fi, ou uma BTS GSM.

Existem no mercado, atualmente, diversas soluções em SDR com diferentes configurações de *Hardware*, como capacidade de processamento e interfaces para periféricos e configuração. Neste trabalho foram utilizados três dispositivos diferentes, que serão apresentados em detalhe: RTL-SDR, USRP E110 e a BladeRF [32, 34, 35].

3.1 O RTL-SDR

O dispositivo mais simples utilizado neste trabalho é o RTL-SDR, um *Dongle* USB para capturar sinais de TV. Foi descoberto que muitos fabricantes deste tipo de dispositivo utilizam o *Chip* RTL2832U para captura de sinais de rádio [19]. Com isso, foi descoberto que os sinais recebidos também poderiam ser interpretados por outros *Softwares*, como o GNR [32]. A maior limitação deste dispositivo é que ele apresenta somente a capacidade de receber sinais, e não de transmitir. A Figura 5 mostra uma imagem do RTL-SDR utilizado.



Figura 5: Imagem do RTL-SDR utilizado.

Por meio do GNR é possível utilizar este dispositivo como um receptor de rádio FM, como um simples analisador de espectro ou um receptor para diferentes modulações, como o GMSK, AM e 8PSK.

3.2 A USRP E110

As USRP são plataformas de SDR desenvolvidas pela *Ettus Research* [35] que possuem um FPGA (*Field programmable Gate Array*) que trata as conversões DAC, ADC, conversão de banda-base para alta frequência e vice-versa. Enquanto isso, o computador conectado à USRP trata as outras operações. Uma descrição detalhada dos elementos presentes em uma USRP é apresentada abaixo [26].

- *Antena*

A antena é o componente que realiza a transmissão e recepção de informação. Dependendo do modelo do USRP, é possível utilizar diversas antenas e até mesmo um sistema MIMO (*Multiple-Input Multiple-Output*).

- *Placas-filhas*

As placas-filhas implementam o sistema de conversão do sinal de IF (*Intermediate Frequency*) para RF e compõem parte do *Front-end*. Além disso, são estas placas que definem a potência de saída do sinal a ser emitido.

- *Placa-mãe*

A placa-mãe de uma USRP inclui os conversores DAC e ADC, o FPGA e conexões, como portas seriais ou USB. Além disso, a placa-mãe conecta o dispositivo a periféricos como uma fonte de *Clock*, ou permitindo a elaboração de sistemas MIMO.

- *FPGA*

O FPGA realiza parte ou todo o processamento digital dos sinais. Assim, o FPGA pode transferir o processamento para outros dispositivos, como o computador conectado à placa.

O modelo específico utilizado neste trabalho de graduação, a USRP E110, além dos elementos citados, possui também um computador embarcado, com uma distribuição Linux instalada, de modo que possa ser utilizada sem o auxílio de um computador externo para controlá-la. A Figura 6 apresenta a E110.



Figura 6: Imagem da USRP E110.

3.3 A BladeRF

A bladeRF é um sistema de SDR de baixo custo fabricado pela Nuand [34], que pode ser operada via USB 3.0, e possui um FPGA *Altera Cyclone 4* para o processamento de sinais. Ela oferece a possibilidade de trabalhar com frequências de 300 MHz a 3.8 GHz, com transmissor e receptor independentes. Além disso, o pós-processamento, entre o FPGA e a interface USB 3.0 é feito por um processador ARM9 de 200 MHz [21]. A Figura 7 apresenta uma imagem da BladeRF.

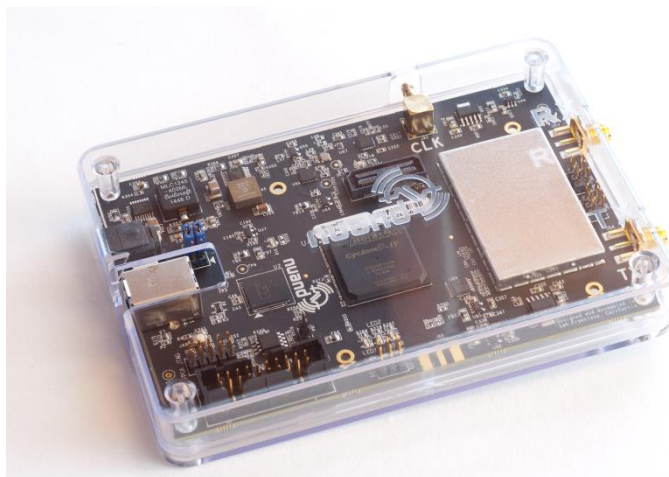


Figura 7: Imagem da BladeRF.

É importante notar que a BladeRF, diferente da USRP E110, não possui um sistema embarcado. Desse modo, ela depende do processamento de um computador para que possa operar.

4. USO DA COMUNICAÇÃO MÓVEL EM SMART GRIDS

As SG são redes de transmissão e distribuição de energia elétrica utilizando tecnologias para melhorar sua confiabilidade, rendimento e controle [24]. Elas vêm resolver problemas das redes de distribuição de energia atuais. Há muita perda no sistema de distribuição atual, as fontes renováveis são de difícil inserção e a arquitetura da rede é linear e simples. Na Figura 8 é apresentada a dinâmica de distribuição de energia de uma rede elétrica atual comparada a uma SG.

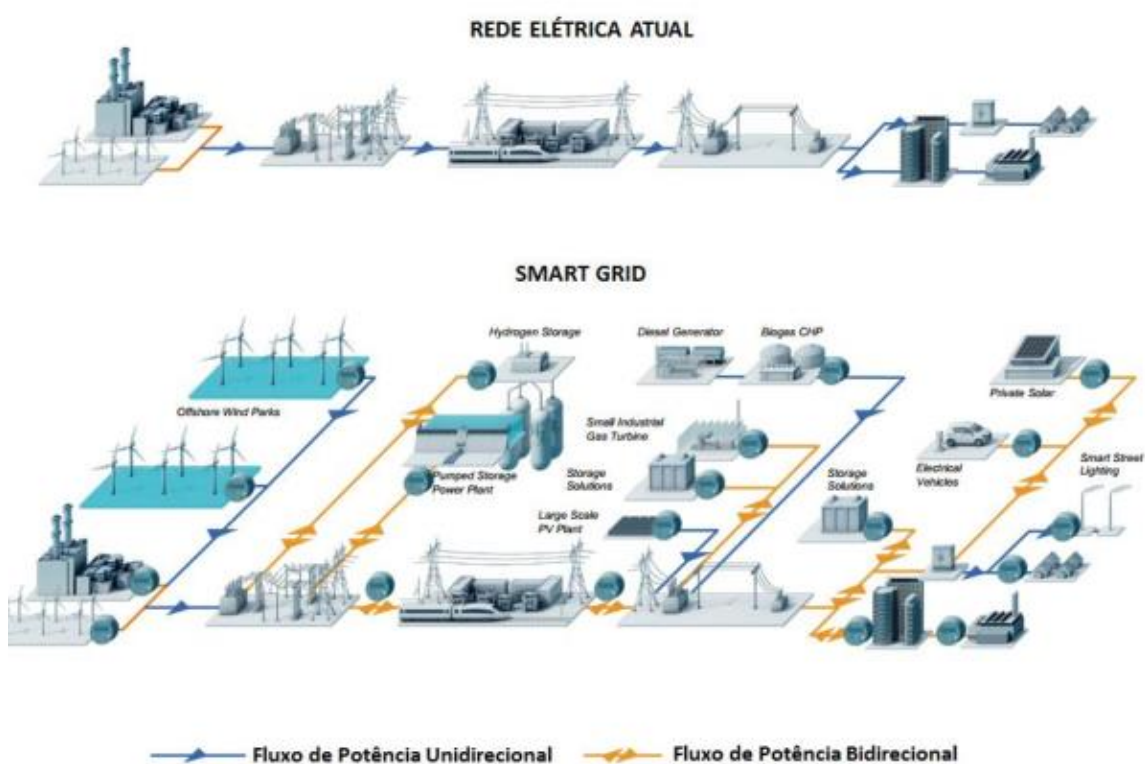


Figura 8: Comparação entre fluxos em uma rede tradicional e uma SG (SIEMENS – Smart Grid Division, 2014) [29].

Para a implementação de redes deste tipo é necessário que a esta seja integrada, que se comunique. No entanto, é necessário também construir tecnologias para que essa interação seja possível. São necessários sensores, medidores modernizados, tecnologias de computação distribuída, informação em tempo real e interfaces de melhoria de apoio à decisão [29].

Neste trabalho de graduação foi estudada uma forma de utilizar redes móveis para o envio de sinais de controle para aerogeradores. Para este controle pode-se utilizar as comunicações sem fio para que seja possível um melhor gerenciamento dos aerogeradores. Devido às grandes áreas ocupadas pelos aerogeradores, as redes sem fio permitem integração sem a necessidade de uma grande rede de cabeamento, o que seria caro e de difícil manutenção. Uma forma básica de um sistema de controle é apresentada na Figura 9.

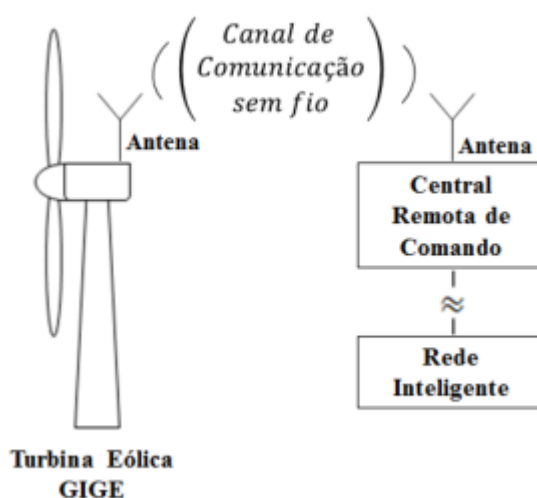


Figura 9: Comunicação sem fio para turbinas eólicas [29].

O objetivo deste trabalho de graduação foi desenvolver um sistema de comunicação sem fio baseado no conceito de SDR utilizando a tecnologia GSM para a transmissão das referências de potência para o sistema de controle elétrico de aerogeradores. Desse modo, é interessante analisar o envio destes sinais utilizando dispositivos SDR com *Softwares* como GNR e OpenBTS.

5. ANÁLISE DO SISTEMA GSM DESENVOLVIDO PARA CONTROLE DE AEROGERADORES

5.1 Configuração do Ambiente de Trabalho

O ambiente de trabalho configurado para a execução de todos os testes executados é apresentado nesta seção. Serão apresentadas as formas de instalação dos *Softwares* utilizados, como o GNR e o OpenBTS, como também os dispositivos utilizados em cada *Software*.

Os softwares de simulação utilizados neste trabalho, *GNR* e *OpenBTS*, requerem a utilização de um sistema operacional *Linux* para funcionarem. No entanto, não é qualquer distribuição ou versão que resultará em um bom funcionamento das ferramentas. O mau funcionamento se deve ao fato das ferramentas apresentarem diversas dependências que muitas vezes não estão presentes em todos os sistemas operacionais. Além disso há também pouco suporte, de modo que este trabalho procurou seguir as condições indicadas pelos desenvolvedores dos softwares com o intuito de obter o melhor desempenho.

Assim, inicialmente, foi escolhido o sistema operacional mais apropriado para as ferramentas a serem utilizadas. Conforme o manual de instalação do *OpenBTS* [30], o sistema operacional mais indicado é o *Ubuntu* na versão 14.04.

Após a instalação do sistema operacional, foi instalado o GNR. Existem diversas formas de instalar o software, entre elas o repositório padrão do *Ubuntu*, o qual requer apenas um comando que automaticamente resolve todas as dependências, porém o problema desta estratégia é que pode ser instalada uma versão mais básica e, muitas vezes, desatualizada, já que o repositório não necessariamente contém a versão mais recente dos *Softwares* disponíveis. Outra forma de instalar, adequada a usuários mais avançados ou que possuem necessidades específicas, como uma versão específica do GNR ou do sistema operacional, é efetuar o *Download* e compilar manualmente o código-fonte, necessitando, portanto, resolver todas as dependências por conta própria. Este método exige também que o usuário instale ferramentas extras, como as bibliotecas para utilização de sistemas de SDR, manualmente.

Neste trabalho, no entanto, foi utilizado um terceiro método. Tal método é mais recente, mais simples que a instalação manual e mais completa, pois inclui as bibliotecas para utilização de sistemas de SDR. O método em questão utiliza uma ferramenta chamada *Pybombs* (*Python Build Overlay Managed Bundle System*) [31]. O *Pybombs* é um administrador de instalações do GNR que possui o objetivo de resolver as dependências e instalar softwares por meio de receitas de instalação, ou seja, para cada projeto a ser instalado por meio do *Pybombs*, existe uma receita de instalação para que esta seja feita sem erros. A receita utilizada neste trabalho inclui o software GNR com as bibliotecas e *Drivers* necessários para utilização de sistemas de SDR. O procedimento para instalação do *Pybombs* pode ser visto em detalhes no Apêndice A.

A instalação do OpenBTS pode ser encontrada em um livro disponibilizado pela *Range Networks*, chamado *Getting Started With OpenBTS* [33]. Os primeiros tópicos do manual consistem em apresentar o hardware necessário para que seja possível utilizar o OpenBTS. Desse modo, são necessários: um dispositivo SDR utilizado para transmissão e recepção do OpenBTS, um computador *Linux* para execução do *Software*, antenas para o dispositivo SDR, *SIM Cards* de teste e telefones GSM.

O dispositivo SDR utilizado para a simulação do OpenBTS é a BladeRF, diferente dos outros testes de camada física, que utilizaram a E110. Nos testes realizados foram utilizados diferentes dispositivos, entre eles, o Asus Zenfone 2 Laser e o Samsung Galaxy Y, apresentados na Figura 10.



Figura 10: Dispositivos utilizados nos testes de GSM.

A BladeRF apresenta antenas individuais para transmissão e recepção. Desse modo, foram utilizadas duas antenas monopolo apresentadas na Figura 11.



Figura 11: Antenas monopolo utilizadas na BladeRF.

Por outro lado, os *SIM Cards* de teste não foram providenciados. Assim, foram utilizados *SIM Cards* comuns, de operadoras de telefonia móvel, como os apresentados na Figura 12, respectivamente, da Vodafone Hungria e da TIM Brasil.



Figura 12: *SIM Cards* utilizados nos testes.

Conforme verificado nos manuais do OpenBTS, utilizar *SIM Cards* de operadoras trazem algumas desvantagens. Para que seja possível utilizar criptografia na troca durante a comunicação é necessário que o OpenBTS conheça a chave Kc do *SIM Card*, conforme apresentado anteriormente. No entanto, o *SIM Card* da operadora possui essa informação embutida, além de não ser possível modificá-la. Desse modo, utilizando *SIM Cards* de operadoras, é possível somente a comunicação sem criptografia entre a BTS e o dispositivo móvel.

Preenchendo os requisitos básicos para instalação do *Software*, deve-se verificar a compatibilidade do dispositivo de SDR, uma vez que o OpenBTS está disponível somente para um número pequeno de sistemas de SDR. Essa limitação ocorre porque diferentes dispositivos possuem diferentes *Drivers* para seus transmissores. A instalação padrão fornecida no manual, supõe que o usuário possua um dos dispositivos compatíveis [30]. A lista de dispositivos é a seguinte: SDR 1, USRP 1, B100, B110, B200, B210, N200 e N210. Como pode ser visto, a BladeRF não está entre os dispositivos listados. Portanto, para que seja possível utilizá-la, é preciso que o usuário forneça a interface de transmissão que existe entre o OpenBTS e o sistema de SDR. Primeiramente, o *Driver* da BladeRF teve de ser instalado e, em seguida, um transmissor de GSM teve de ser compilado para a utilização no OpenBTS e somente então a instalação do OpenBTS pôde ser efetuada. As instruções de instalação para o OpenBTS são apresentadas em detalhes no Apêndice B.

5.2 Análise da Interface Física do GSM Empregando o GNR

Os primeiros testes empregaram o GNR para simular os principais elementos da camada física do GSM. Estas simulações envolveram a modulação, codificação convolucional e codificação de bloco permitindo avaliar a eficiência da interface aérea do GSM na transmissão dos dados. Pode-se a partir destas simulações inferir a influência da modulação e codificação do GSM na transmissão dos dados. Podendo então, ao comparar com a implementação completa, feita pelo OpenBTS, verificar a influência de outros elementos do padrão GSM na qualidade da transmissão. Dentre estes, pode-se citar o *Frequency Hopping*, que evita que um determinado canal de comunicação que possua desvanecimento severo em uma determinada frequência esteja constantemente alocado a um mesmo usuário.

As simulações efetuadas nesta fase do projeto consistiram em transmitir uma sequência longa de bits repetidamente utilizando o GNR na USRP E110 e a recepção foi feita no RTL-SDR, também utilizando o GNR.

O primeiro teste efetuado foi a captação de uma rádio FM utilizando o *Software 'rtl_fm'*, instalado junto com o *Driver* do RTL-SDR. O intuito deste teste foi verificar o funcionamento correto do dispositivo. O comando utilizado para a captação é apresentado abaixo.

```
rtl_fm -M wbfm -f 102.1M | play -r 32k
```

O comando acima recebe alguns parâmetros para sua execução. Estes parâmetros são explicados abaixo.

- -M wbfm: indica que a modulação é *Wide Band Frequency Modulation*;
- -f 102.1M: indica que a frequência central para sintonizar será de 102.1 MHz;
- play: comando do Linux para acessar a saída de áudio do computador;
- -r 32k: taxa de amostragem da saída de áudio;

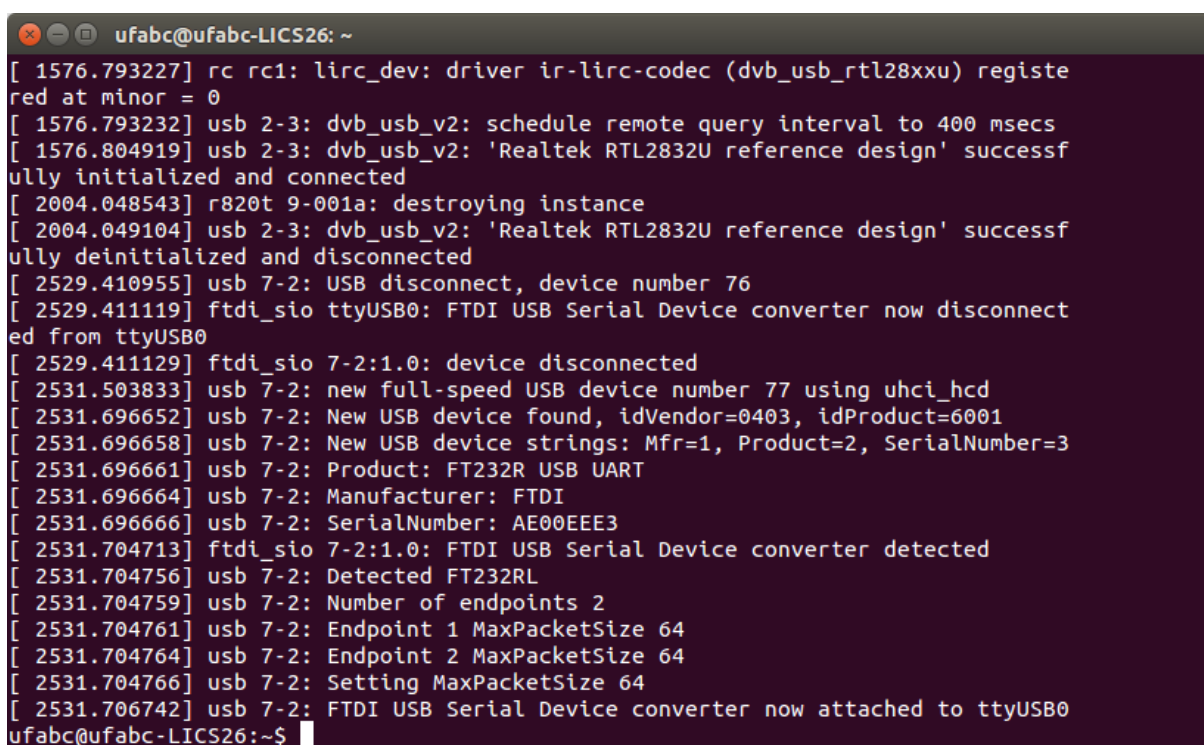
Com o comando acima foi possível ouvir à rádio com perfeição, sem erros. Desse modo, foi observado que o dispositivo está foi instalado corretamente e está funcionando de forma adequada.

O próximo dispositivo testado foi a USRP E110. Devido ao fato deste dispositivo ser embarcado, conforme apresentado, o mesmo possui um sistema

operacional instalado que pode ser utilizado para desenvolvimento e execução, como também pode ser acessado por uma sessão via USB. Os primeiros testes efetuados foram por meio da interface USB.

Ao conectar o dispositivo na porta USB do computador, o mesmo é alocado ao arquivo de conexão do *Ubuntu*. Para que seja possível identificar qual o arquivo localizado, pode-se utilizar um comando chamado *'dmesg'*. Este comando permite verificar as mensagens geradas pelo sistema operacional quando um dispositivo for conectado ou desconectado, acessado e também quando algum erro ocorre.

Ao conectar a USRP E110 no computador, a saída do comando *'dmesg'* apresentada pode ser observada na Figura 13.



```
ufabc@ufabc-LICS26: ~
[ 1576.793227] rc rc1: lirc_dev: driver ir-lirc-codec (dvb_usb_rtl28xxu) registered at minor = 0
[ 1576.793232] usb 2-3: dvb_usb_v2: schedule remote query interval to 400 msecs
[ 1576.804919] usb 2-3: dvb_usb_v2: 'Realtek RTL2832U reference design' successfully initialized and connected
[ 2004.048543] r820t 9-001a: destroying instance
[ 2004.049104] usb 2-3: dvb_usb_v2: 'Realtek RTL2832U reference design' successfully deinitialized and disconnected
[ 2529.410955] usb 7-2: USB disconnect, device number 76
[ 2529.411119] ftdi_sio ttyUSB0: FTDI USB Serial Device converter now disconnected from ttyUSB0
[ 2529.411129] ftdi_sio 7-2:1.0: device disconnected
[ 2531.503833] usb 7-2: new full-speed USB device number 77 using uhci_hcd
[ 2531.696652] usb 7-2: New USB device found, idVendor=0403, idProduct=6001
[ 2531.696658] usb 7-2: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 2531.696661] usb 7-2: Product: FT232R USB UART
[ 2531.696664] usb 7-2: Manufacturer: FTDI
[ 2531.696666] usb 7-2: SerialNumber: AE00EEE3
[ 2531.704713] ftdi_sio 7-2:1.0: FTDI USB Serial Device converter detected
[ 2531.704756] usb 7-2: Detected FT232RL
[ 2531.704759] usb 7-2: Number of endpoints 2
[ 2531.704761] usb 7-2: Endpoint 1 MaxPacketSize 64
[ 2531.704764] usb 7-2: Endpoint 2 MaxPacketSize 64
[ 2531.704766] usb 7-2: Setting MaxPacketSize 64
[ 2531.706742] usb 7-2: FTDI USB Serial Device converter now attached to ttyUSB0
ufabc@ufabc-LICS26:~$
```

Figura 13: Saída do comando *'dmesg'* após a conexão da USRP E110.

Observe que a última mensagem apresentada informa que o dispositivo serial foi conectado ao *ttyUSB0*. Esse é o nome do arquivo de comunicação que deve ser acessado para se conectar com a USRP E110.

Para que esta conexão seja possível, é preciso de um *Software* que permita utilizar a comunicação serial como um terminal. O *Software* utilizado neste trabalho foi o *screen*. O mesmo pode ser obtido por meio do repositório do *Ubuntu* conforme o comando abaixo.

```
sudo apt-get install screen
```

Com o *Software* instalado e a placa conectada, o acesso serial pode ser feito utilizando o comando abaixo.

```
sudo screen /dev/ttyUSB0
```

O resultado é apresentado na Figura 14.

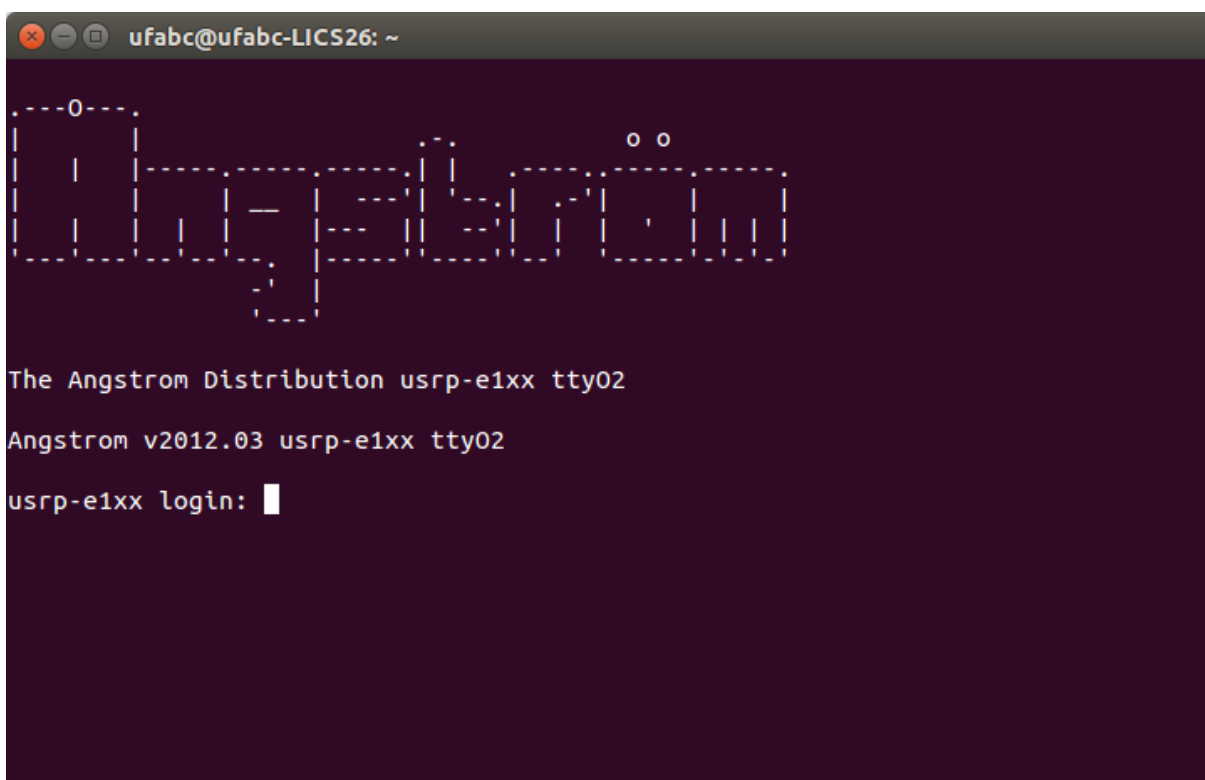


Figura 14: Tela de início do sistema operacional Angstrom.

Conforme a Figura 14, é notável que uma sessão de terminal foi inicializada no sistema operacional da USRP E110.

O teste efetuado para a USRP E110 demandou uma atuação em conjunto com o RTL-SDR. Foi utilizada uma função da USRP chamada '*tx_waveforms*'. Essa função permite transmitir formas de onda, como senos e cossenos, na frequência desejada, desde que esteja dentro dos limites possíveis do sistema de SDR. Além

disso, foi desenvolvido um programa para o GNR que desenha uma FFT do espectro a partir do sinal recebido por meio de um RTL-SDR.

Abaixo, o comando do 'tx_waveforms' utilizado.

```
./tx_waveforms --wave-freq 100e3 --wave-type SINE --freq 800e6 --
ampl 0.5 --ant TX/RX
```

O comando acima recebe como parâmetro a frequência da onda, que neste caso foi selecionada como sendo 100kHz, a forma de onda, um seno, a frequência de transmissão, de 800MHz, e a amplitude de 0.5.

O programa do GNR utilizado para detectar a transmissão deste sinal é apresentado na Figura 15.

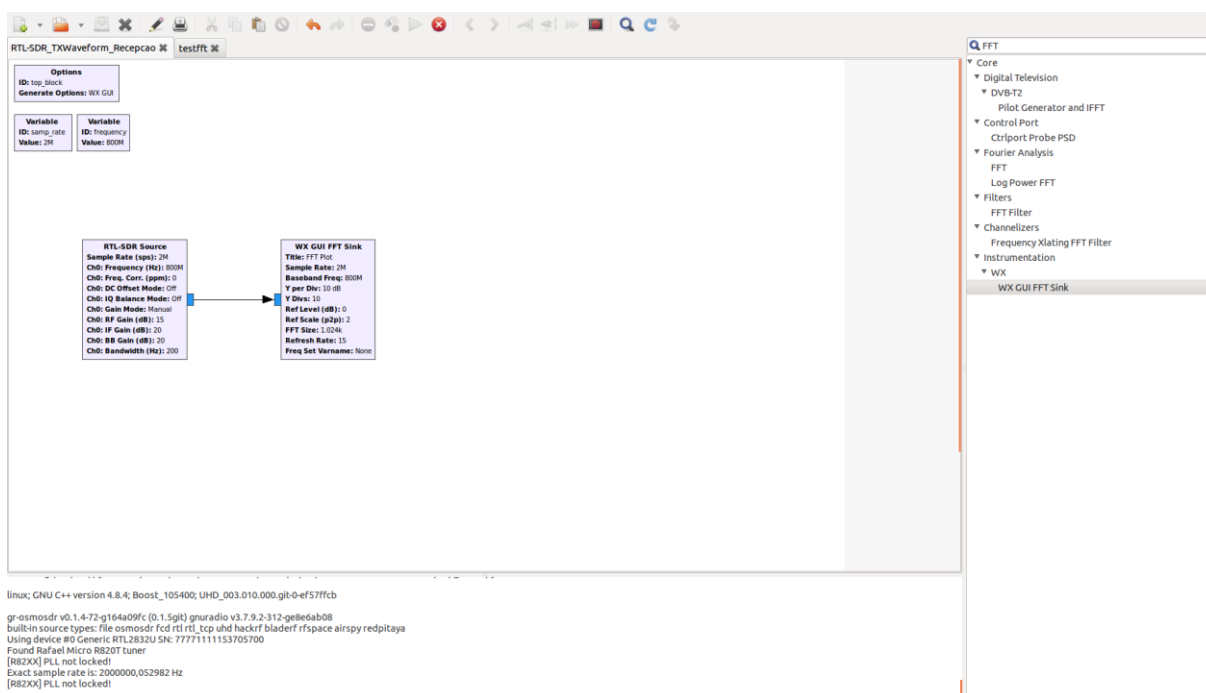


Figura 15: Janela do programa de teste para o RTL-SDR.

O programa acima utiliza apenas dois blocos de função do GNR: RTL-SDR *Source*, responsável por receber amostras do RTL-SDR e o WX GUI FFT *Sink*, responsável por desenha em tempo real a FFT do sinal recebido. O resultado para este teste é apresentado na Figura 16.

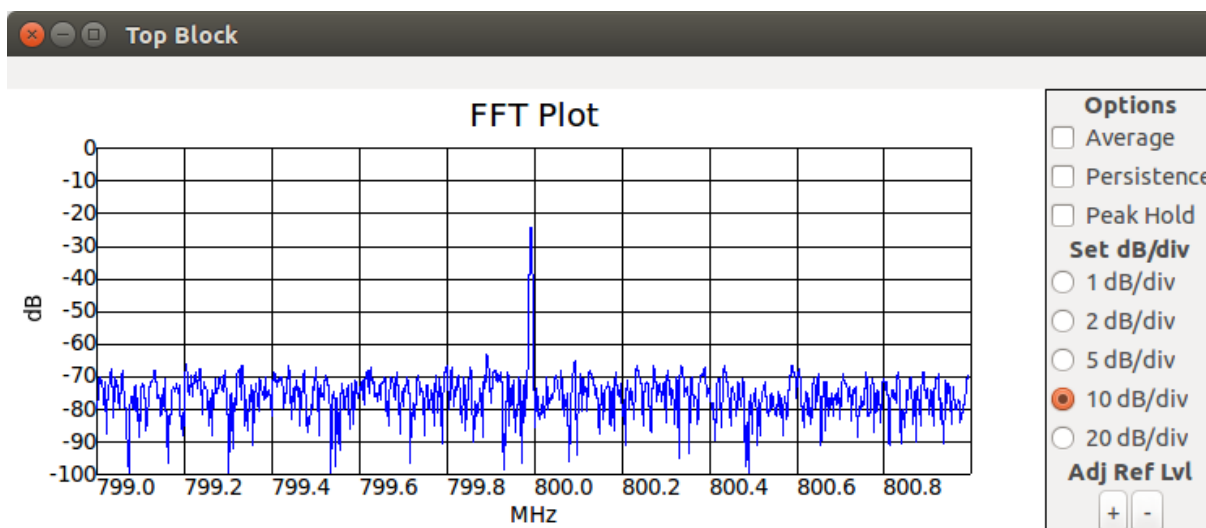


Figura 16: Resultado do teste de 'tx_waveforms'.

Observe pela FFT acima, que um pico é apresentado em 800MHz. Este pico demonstra que o sinal transmitido pelo USRP foi recebido pelo RTL-SDR.

O último teste efetuado consistiu em utilizar a interface gráfica da USRP. O funcionamento da interface gráfica é essencial para o desenvolvimento de programas no GNR, uma vez que o *Software* depende da sua interface gráfica por blocos para funcionar devidamente.

Deste modo, foram conectados um monitor via HDMI, um teclado e um *Mouse* via *Hub* USB em sua porta de periféricos.

Ao inicializar e efetuar o *Log In* no sistema operacional, a área de trabalho do dispositivo foi apresentada. Abaixo, a Figura 17 mostra um exemplo da área de trabalho do *AngstromOS*.

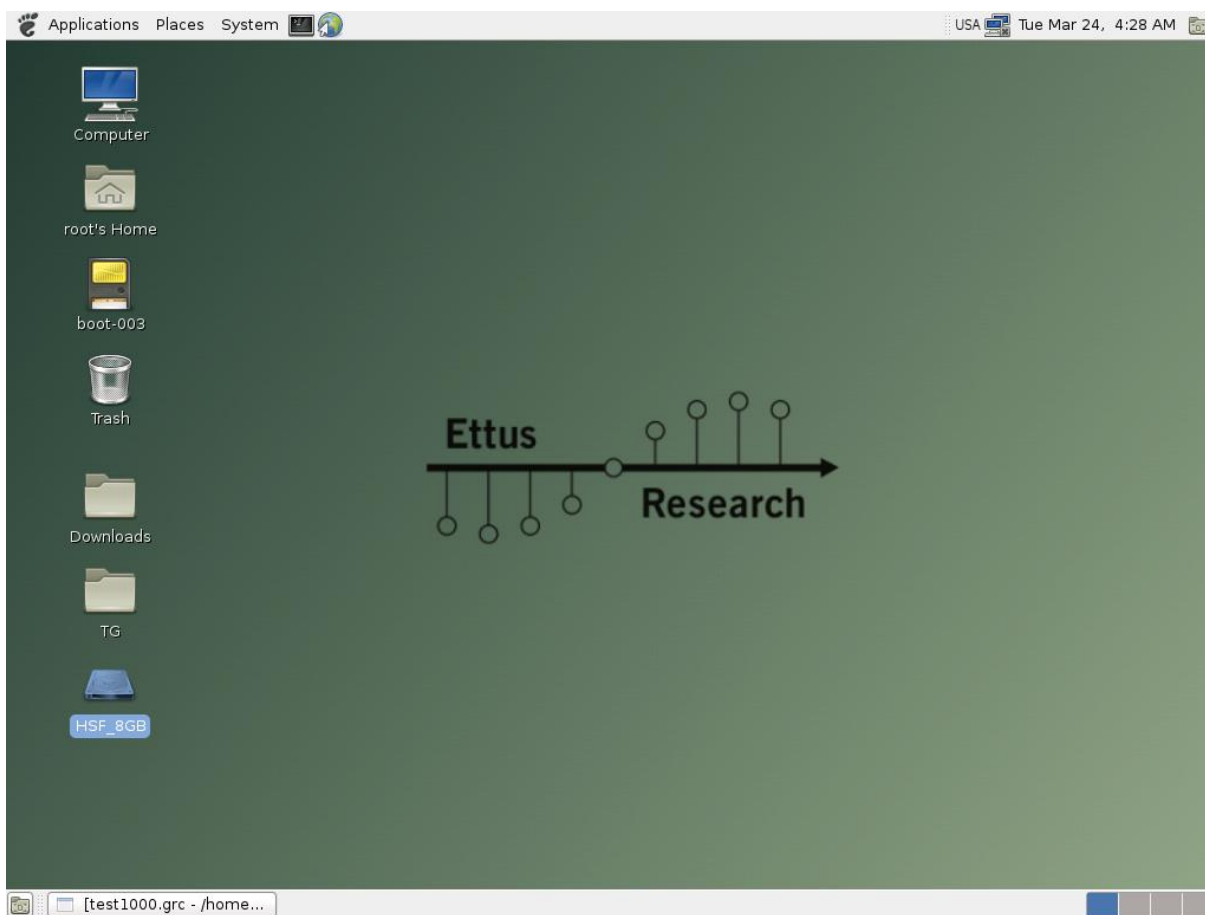


Figura 17: Área de trabalho do *AngstromOS*.

Com o acesso à interface gráfica o GNR da USRP foi testado reproduzindo a função desempenhada pelo comando `'tx_waveforms'`. O programa desenvolvido é apresentado abaixo na Figura 18.

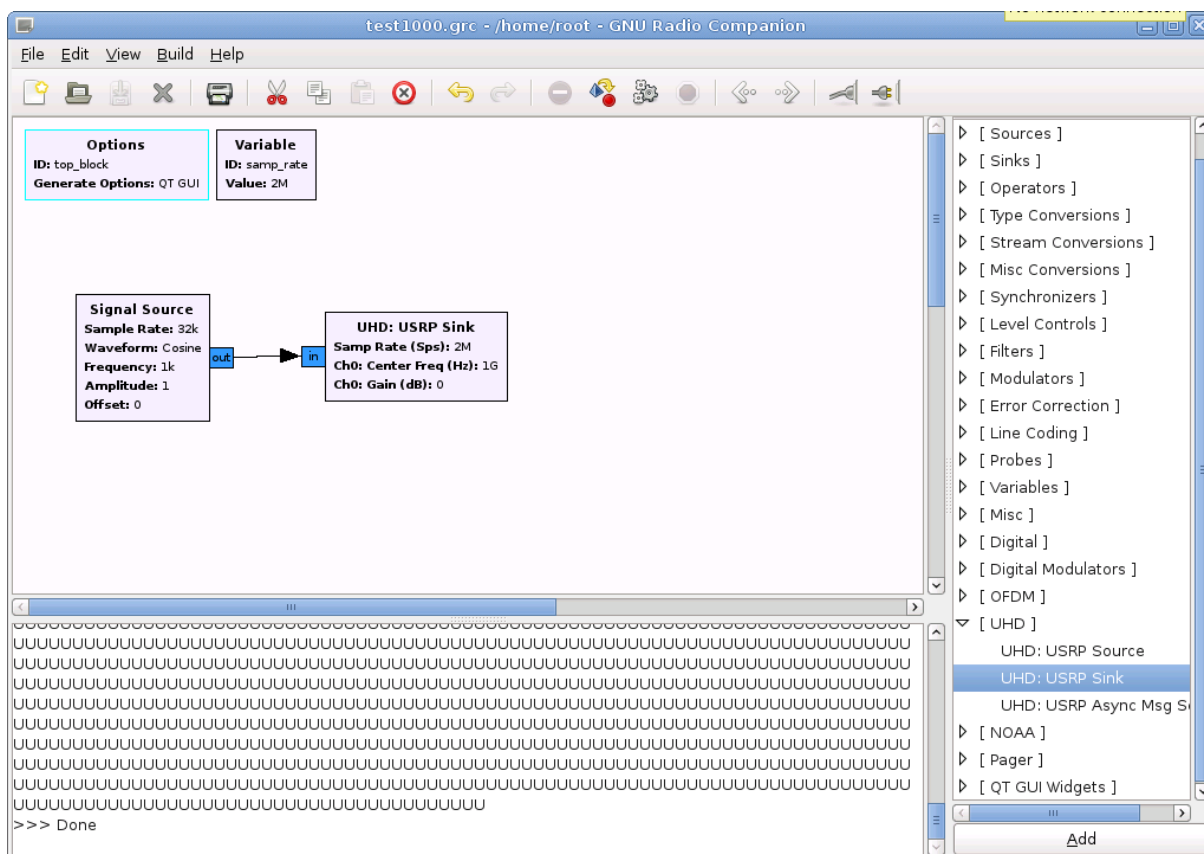


Figura 18: Programa de teste do GNR na USRP E110.

O programa acima consiste apenas em dois blocos: *Signal Source*, responsável por criar uma forma de onda, neste caso o cosseno, e *UHD: USRP Sink*, o qual é responsável por enviar as amostras geradas para a USRP. Vale observar que para o GNR é comum utilizar a palavra *Source* em blocos de origem de sinais ou dados, e *Sink* em saídas. Como observado, tanto o bloco FFT como a saída do USRP são blocos de saída e recebem o nome de *Sinks* no GNR.

Neste teste, o RTL-SDR também utilizou o programa apresentado para a recepção do sinal do comando *'tx_waveforms'* e o seu resultado é apresentado na Figura 19.

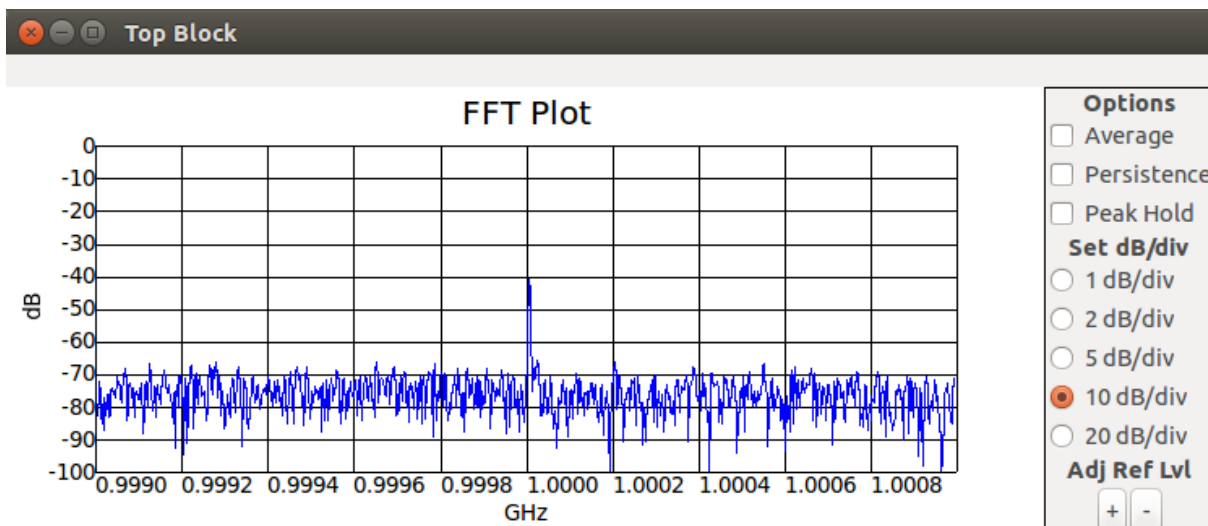


Figura 19: Resultado do teste com o GNR.

Como pode ser observado acima, em ambos os dispositivos o GNR funcionou normalmente, permitindo a transmissão e a recepção de uma forma de onda senoidal. Com os dispositivos funcionando apropriadamente nos *Softwares* a serem utilizados, pôde-se dar prosseguimento ao trabalho.

Utilizando o GNR é possível considerar diversos elementos da camada física do GSM. Três dispositivos de SDR foram utilizados para estes testes: a USRP E110, bladeRF, ambos atuando como transmissores, e o RTL-SDR como receptor.

O diagrama de bloco do transmissor da Figura 20 mostra os elementos utilizados para este teste.

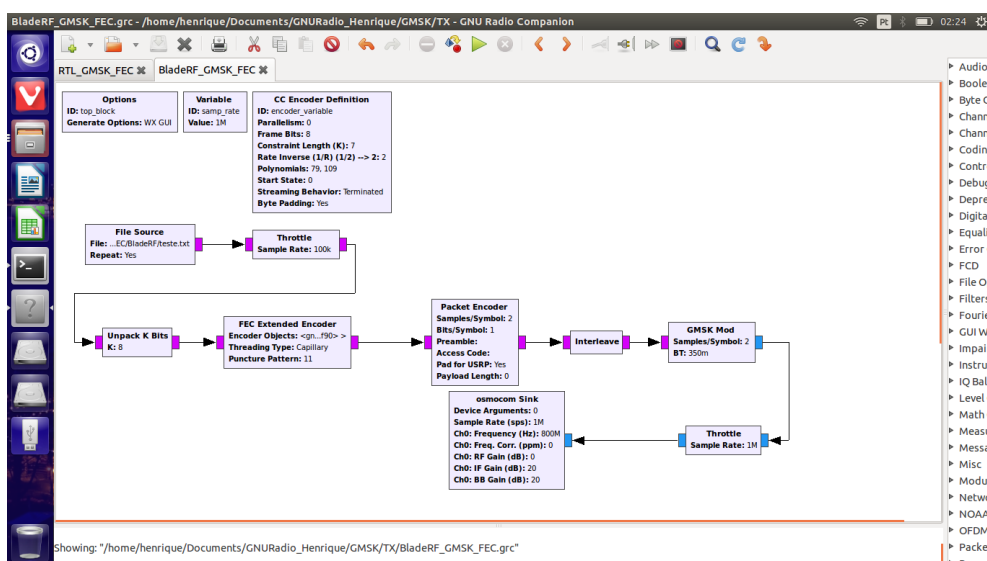


Figura 20: Diagrama de bloco do GNR utilizado no transmissor.

A partir do diagrama acima o sinal transmitido tem sua FFT (*Fast Fourier Transform*) e diagrama de constelação apresentados na Figura 21.

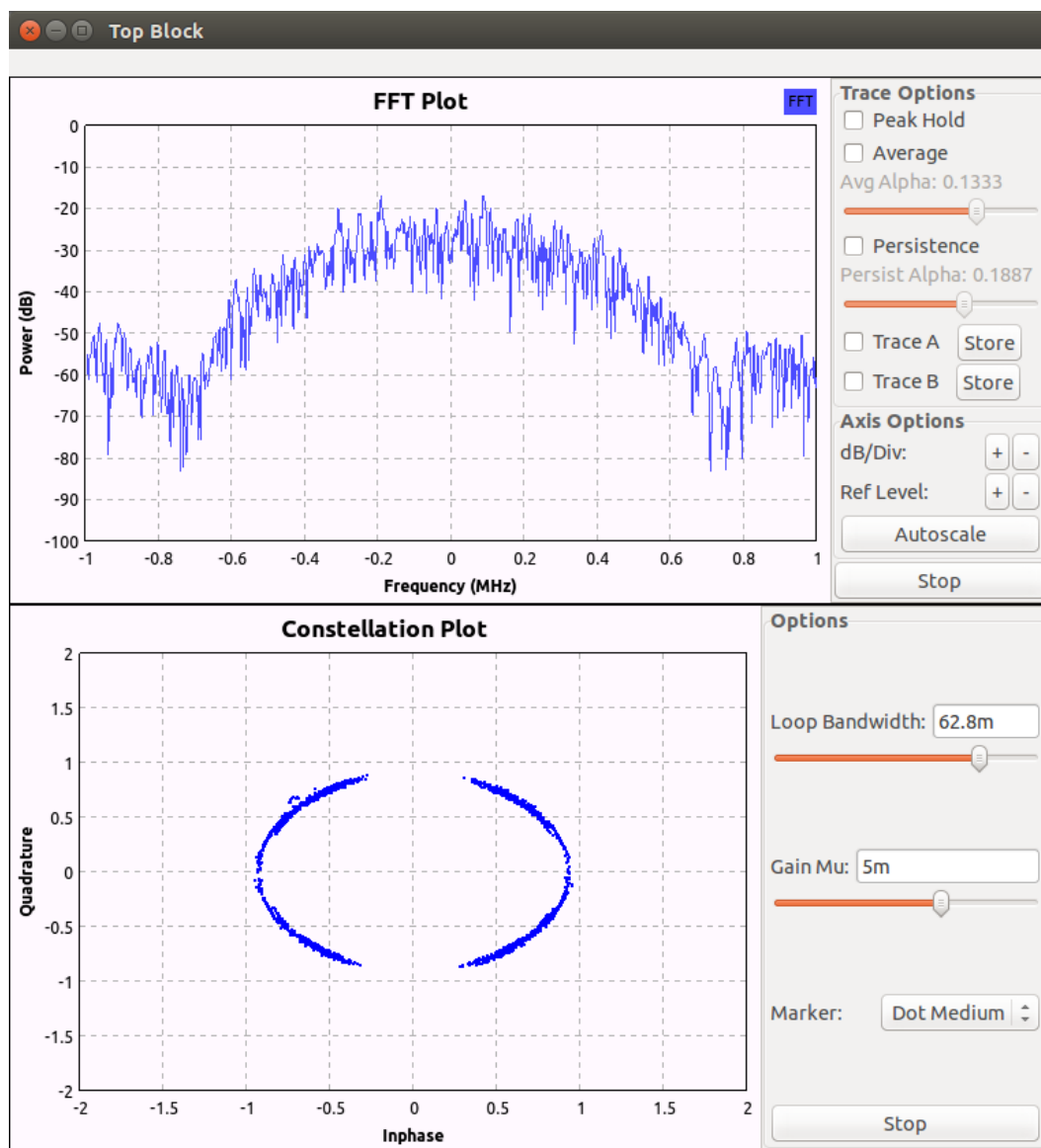


Figura 21: FFT e diagrama de constelação do transmissor GMSK.

Tanto a BladeRF quanto a USRP E110 foram utilizadas neste teste pois a USRP E110 apresenta uma versão mais antiga com codificadores obsoletos. A USRP E110 possui uma versão mais antiga, e não possui muitas opções de codificadores. Por esse motivo foram realizados testes com o codificador convolucional disponível na E110 e também o disponível na BladeRF. Além disso, a E110 é um dispositivo embarcado e essa é uma característica interessante se

considerarmos que ela não necessita de um computador dedicado ao seu funcionamento.

A partir do diagrama de blocos acima, pode-se observar os elementos utilizados para representar a camada física do GSM. Primeiramente, um *File Source*, na qual é inserido o arquivo com os dados a serem transmitidos. O *Throttle*, em seguida, controla a taxa de entrada de dados para se adequar à taxa de amostragem do circuito, ou seja, controla para que a leitura do arquivo seja feita a uma taxa inferior à taxa de amostragem. Para a leitura do arquivo, blocos de 8 em 8 bits são lidos, o que é garantido pelo bloco *Unpack k bits*, cujo valor de bits atribuído é 8. O próximo bloco, o *FEC Extended Encoder*, se encarrega da codificação convolucional aplicada aos dados a serem enviados. Em seguida os pacotes são processados em 1 bit por símbolo, se adequando ao GMSK, utilizando o bloco *Packet Encoder*. O próximo bloco, como o próprio nome informa, é responsável pelo *Interleave* dos bits a serem enviados. Enfim, o módulo GMSK é adicionado ao diagrama e o envio para o sistema de SDR é feito pelo bloco *Osmocom Sink*.

O diagrama para o recebimento dos dados é semelhante ao de transmissão. A diferença entre eles é que os blocos utilizados são os duais dos blocos de transmissão. O diagrama de blocos para recepção é apresentado na Figura 22.

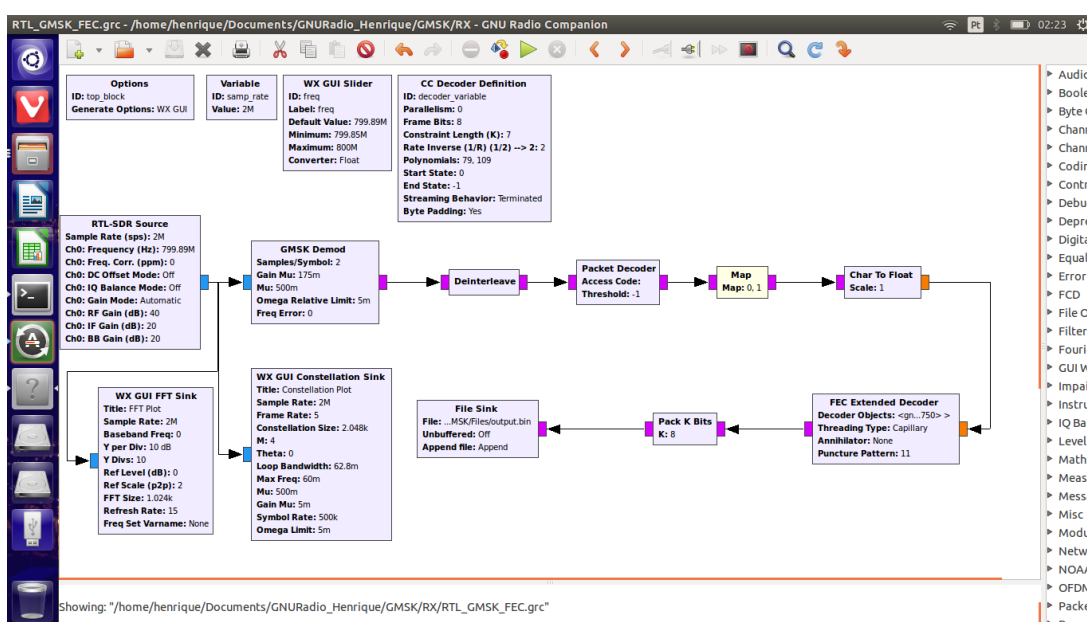


Figura 22: Diagrama de bloco do GNR utilizado no receptor.

A partir do diagrama acima o sinal recebido tem sua FFT e diagrama de constelação apresentados na Figura 23.

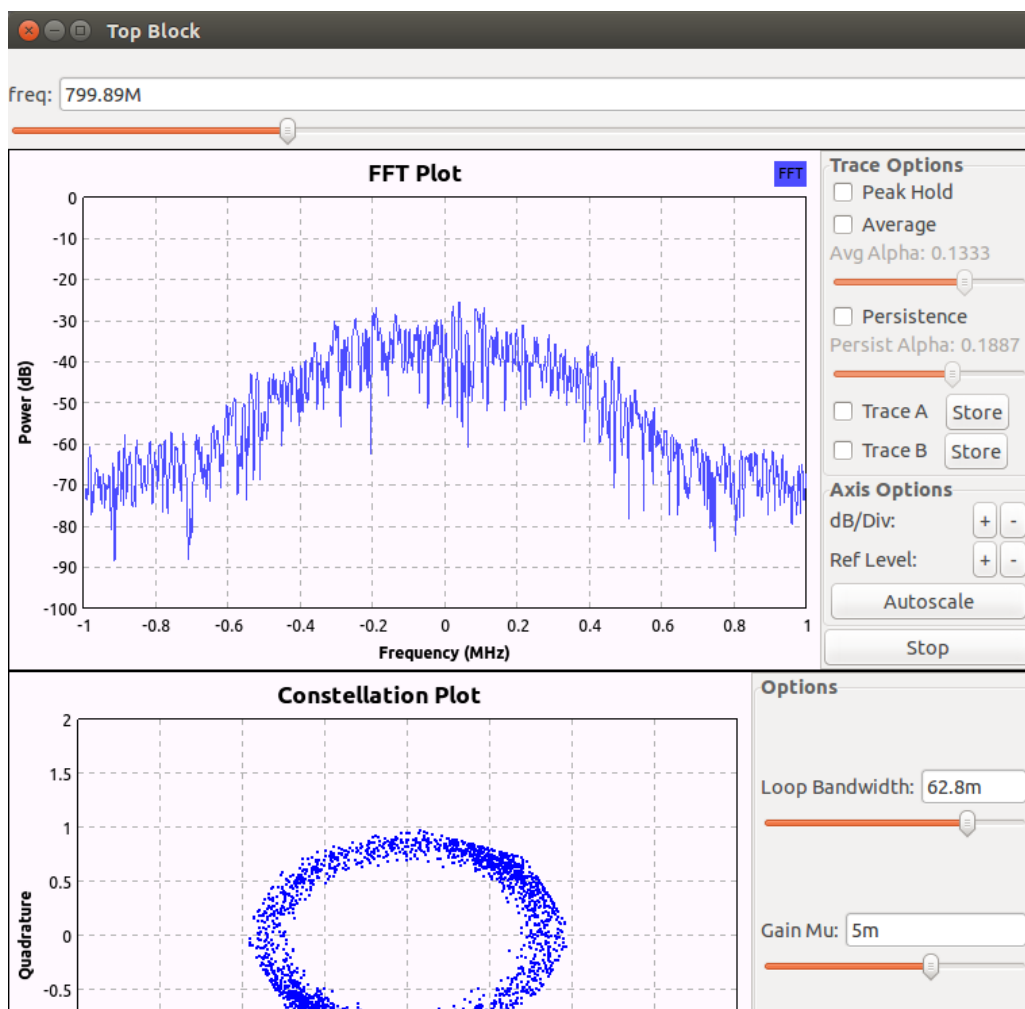


Figura 23: FFT e diagrama de constelação do receptor GMSK.

Nestes testes foi verificada a influência da interface de rádio no envio de informações corretas. Para isso o envio contínuo de uma mensagem longa, de 40 Bytes, foi feito durante 10 minutos. Nos arquivos de saída procurou-se por erros, ou seja, caracteres errados no receptor. Durante estes não foram observados erros, no entanto, ao longo do envio, viu-se a ausência de caracteres da mensagem enviada. Esse erro pode ser observado no exemplo abaixo, utilizando uma mensagem de texto de mais fácil leitura.

- ESTA E UMA MENSAGEM DE TESTE. ESTA E UMA MENSAGEM DE TESTE. ESTA E UMA MENSAGEM DE TESTE. ETA E UMA MENSAGEM DE TESTE

Observe na sequência acima que na última vez que a mensagem é enviada, duas letras estão faltando na palavra "ESTA". Não se pode considerar a ausência destes caracteres como um erro no envio, pois um erro seria um ou mais bits trocados, acarretando em um caractere incorreto ou inválido. Falhas como estas ocorrem pela ausência de um processamento dedicado. O sistema operacional utilizado não é dedicado à tarefa de envio ou recebimento de dados no GNR, ou seja, interrupções do sistema e outros processos estão dividindo recursos computacionais, como processamento. Se o processador pausou o processamento do GNR tempo o suficiente para perder caracteres inteiros, estes simplesmente deixaram de aparecer na recepção. Como a correção de erros é tratada por *Byte*, ou seja, por caractere, o GNR também não detecta este tipo de erro. Uma forma de resolver este problema seria aplicar uma correção de erros em blocos maiores de bits, mesmo que o envio ainda seja feito por *Byte*. Desse modo, falhas que ocupam um curto tempo, como esta, poderiam ser detectadas e corrigidas.

Além disso, concluiu-se que a USRP E110 possui capacidade computacional suficiente para reproduzir a camada física do GSM, pois a mesma foi suficiente para efetuar os testes descritos nesta seção.

5.3 Análise da Interface de Rede do GSM Empregando o OpenBTS

A simulação completa do GSM foi realizada utilizando o *Software OpenBTS* e o recurso de SMS, fornecido pelo *SMQueue*.

Os primeiros testes realizados foram de enviar simples SMSs de texto entre os dispositivos registrados na rede. Para que eles pudessem ser usados, é preciso registrá-los na rede GSM local. A interface para registro de SIM Cards presente no OpenBTS é simples, e consiste em um *Script Python* que recebe como parâmetros os dados necessários para o registro. Um exemplo de comando para registrar um telefone celular é apresentado no manual do OpenBTS e pode ser visualizado no bloco abaixo.

```
./nmcli.py sipauthserve subscribers create "ZENFONE TIM"  
IMSI724031462957406 11952475323
```

O comando acima registra o *SIM Card* a partir do seu IMSI e também do MSISDN. Um texto, no caso "ZENFONE TIM", pode ser associado para identificar o dispositivo cadastrado.

Pode-se observar na Figura 24 a autorização do *SIM Card* cadastrado e também o sinal de recepção utilizando o GNR com o RTL-SDR.

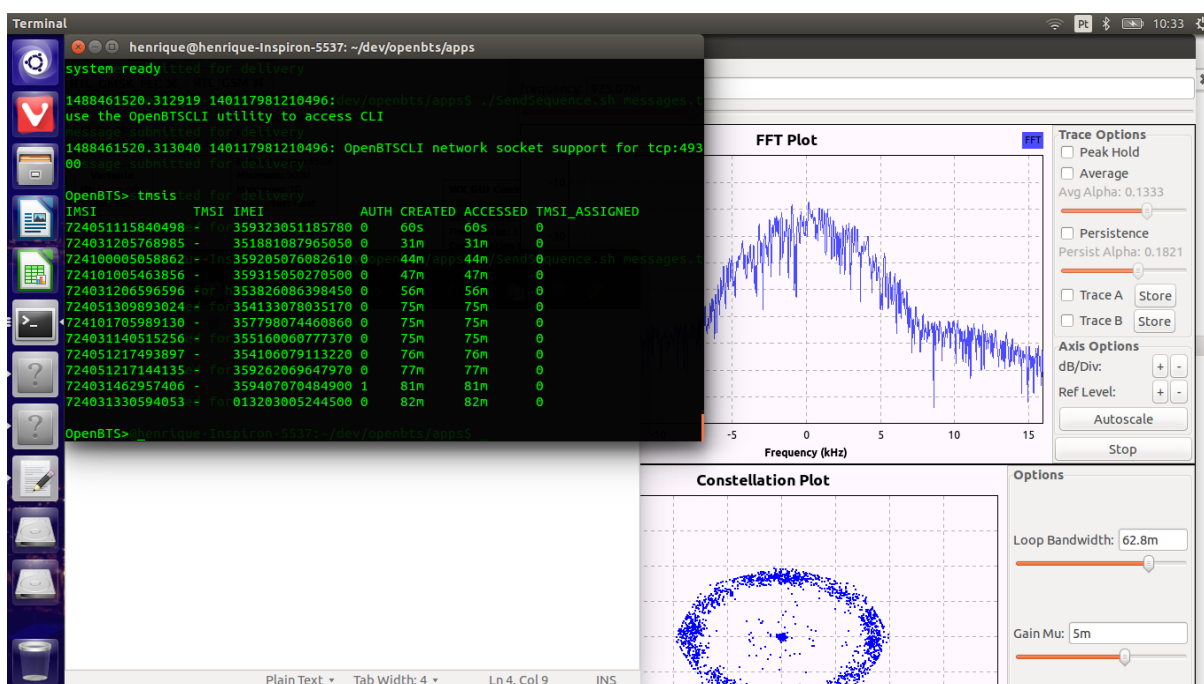


Figura 24: Autorização do *SIM Card* e sinal de recepção.

Observe no terminal aberto na imagem acima que há várias colunas de dados sendo exibidas. A primeira coluna exibe o IMSI do *SIM Card*, o IMEI também é apresentado e, o mais importante, a coluna AUTH, de *Authorization*, que informa se o celular foi autorizado ou não. Se o valor é '1' então sim, foi autorizado. Observe também que entre todos os celulares que tentaram acessar a rede somente um conseguiu. É possível verificar que este único dispositivo possui IMSI correspondente ao IMSI cadastrado. Esse resultado demonstra que o registro do celular no *SIPAuthserve* surtiu efeito no OpenBTS e o dispositivo cadastrado foi autorizado.

Caso o MSISDN seja registrado com o código nacional precedendo o código local, o *Script* automaticamente irá criar uma segunda referência contendo somente o telefone local, sem o código nacional, o que permite que o telefone seja localizado por ambos os números. No entanto, se um *SIM Card* tem seu MSISDN registrado

sem o código nacional, uma segunda entrada contendo o código nacional não será criada, pois é impossível para o *Software* não trata o código nacional presente no IMSI para recriar o MSISDN completo.

Após inicializar o OpenBTS, percebeu-se que não era possível estabelecer uma conexão com a rede criada, mesmo procurando manualmente a mesma. Foi verificado que este problema ocorria devido a interferências com redes existentes após testá-la dentro de uma câmara anecoica. Para que os testes pudessem ser realizados fora da câmara foi preciso selecionar um canal fora da faixa de frequência utilizada pelas operadoras de telefonia móvel. Foi selecionado o canal 975 do GSM 900, reservado para testes. Com este canal, foi possível conectar os dispositivos móveis à rede sem a necessidade de fazê-lo dentro da câmara anecoica. A Figura 25 apresenta o resultado da busca por redes após a mudança de frequência.

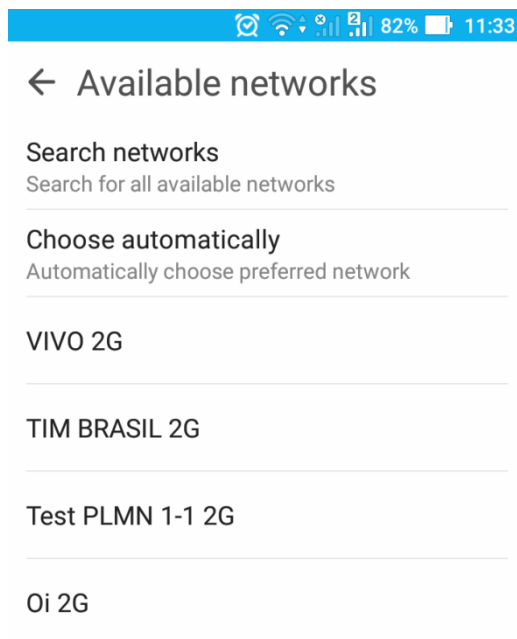


Figura 25: Resultado da busca de redes.

Com a seleção da rede Test PLMN 1-1 2G em ambos os dispositivos móveis foi feito um envio de SMS a partir do terminal de comando do OpenBTS. Este terminal permite que a própria rede envie mensagens para qualquer dispositivo na rede e que também se identifique como o usuário desejado. É possível criar um *Short Code* ou utilizar o telefone de outro dispositivo móvel como telefone de origem para que este seja o telefone caso o recipiente da mensagem deseje retornar o SMS recebido. O comando abaixo, utilizado a interface de linha de comando do OpenBTS permitiu o envio de uma mensagem de teste.

```
sudo ./OpenBTSCLI -c sendsms 724031462957406 4000 "TESTE PARA O TG"
```

O comando *Sendsms* exige, como primeiro parâmetro, o IMSI de destino, em seguida, o telefone de origem que será apresentado ao recipiente e, por fim, a mensagem de texto. A Figura 26 apresenta o resultado do comando de teste acima.

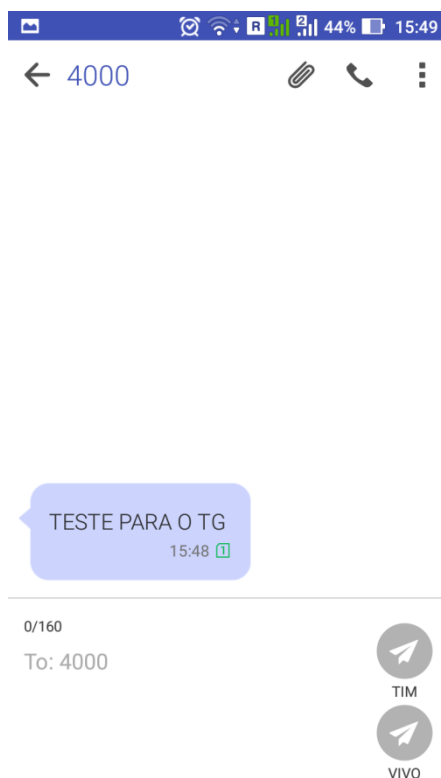


Figura 26: Resultado do teste de envio de SMS a partir do *Short Code* 4000.

A Figura 26 permite verificar também que o dispositivo está conectado no modo *Roaming*, conforme a barra no topo da imagem, à esquerda das barras de intensidade de sinal. A rede criada utiliza um código MNC (*Mobile Network Code*) e MCC (*Mobile Country Code*) diferentes do código da operadora registrada como local no *SIM Card*. Desse modo, a conexão só pode ser efetuada com o *Roaming* ativo.

5.4 Análise da Aplicação da OpenBTS em *Smart Grids*

Para a aplicação proposta do OpenBTS em SGs, deve ser possível transmitir sinais de controle de potência ativa e reativa para aerogeradores por meio de SMSs, conforme testado na seção anterior.

O sistema proposto é formado por uma USRP localizada no centro de operações, a partir da qual são enviados SMSs com sinais de controle pela rede GSM até a placa *EasyPIC v7*, instalada no aerogerador a ser controlado [38]. A *EasyPIC v7* é um dispositivo de desenvolvimento em PIC que acompanha um *Chip* para programar e *Debugging*, além da presença de diversos periféricos. Um periférico necessário para os testes, mas que não faz parte, originalmente, é a *Teli G865 QUAD* [39]. Este periférico é um módulo GSM que possui, além de outras funções, a capacidade de receber e enviar mensagens SMS. Deste modo, o módulo GSM é utilizado em conjunto com a *EasyPIC* para receber comandos via SMS e convertê-los para uma saída em tensão elétrica analógica em uma das saídas de conversão digital-analógica (DAC) da placa, que deve ser conectada ao sistema de controle do aerogerador. A Figura 27 mostra a *EasyPIC v7*, enquanto que a Figura 28 mostra a *Teli G865 QUAD*.



Figura 27: *EasyPIC v7*, utilizada para os testes.



Figura 28: Imagem aproximada do módulo *Telit GL865 QUAD* conectada à placa.

A *EasyPIC* foi programada para receber o seguinte formato de mensagem: primeiro o caractere ":" indicando o início do comando, um segundo dígito para indicar se o comando seria para acender um LED (1) ou indicar sinais de tensão (0) na saída da placa, e oito dígitos de 0 a 8 indicando o nível de tensão de saída. Cada um dos valores de tensão será aplicado por 100ms e, ao término de uma sequência, ela será repetida até que uma nova sequência seja fornecida. A Tabela 1 indica os valores de tensão para cada um dos nove níveis disponíveis.

Tabela 1: Níveis de tensão disponíveis no sistema proposto.

Nível	Tensão (V)
0	0
1	0,3
2	0,6
3	0,9
4	1,2
5	1,5
6	1,8
7	2,1
8	2,4

Com esta configuração foi testada se a placa realmente conseguiria interpretar os SMSs e, então, aplicar os valores de tensão a uma de suas portas de saída. Além disso, verificou-se os valores de tensão entre o valor mais baixo e mais alto possíveis nesta configuração. Por fim foi verificada a utilização deste dispositivo em conjunto com o OpenBTS no envio sequencial de SMS, estimando a frequência de envio para que fosse possível receber e interpretar as mensagens sem falhas.

Nos testes a seguir, conforme padrão apresentado, os SMSs de controle devem conter oito dígitos informando qual valor de amplitude deve ser representado ao longo de 100ms. O programa dentro da placa repete este sinal continuamente até ser recebida outra forma de onda para substituir a anterior.

O primeiro teste realizado foi o de enviar a seguinte sequência: 13588531. Essa sequência deve formar degraus crescentes até atingir o valor de amplitude máxima e em seguida decrescer utilizando os mesmos valores. A Figura 29 apresenta os valores de saída visualizados no osciloscópio.

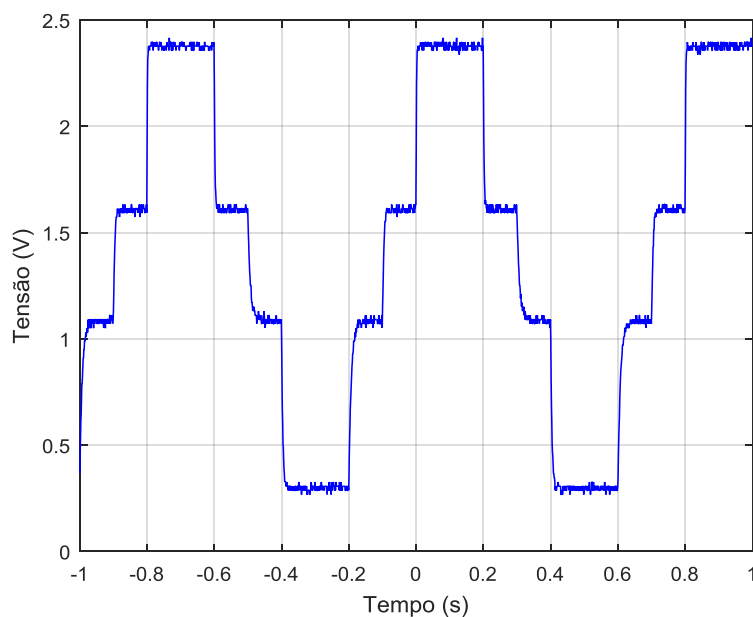


Figura 29: Saída no osciloscópio para a mensagem 013588531.

Também é possível visualizar, na Figura 30, uma foto do osciloscópio com a curva acima.

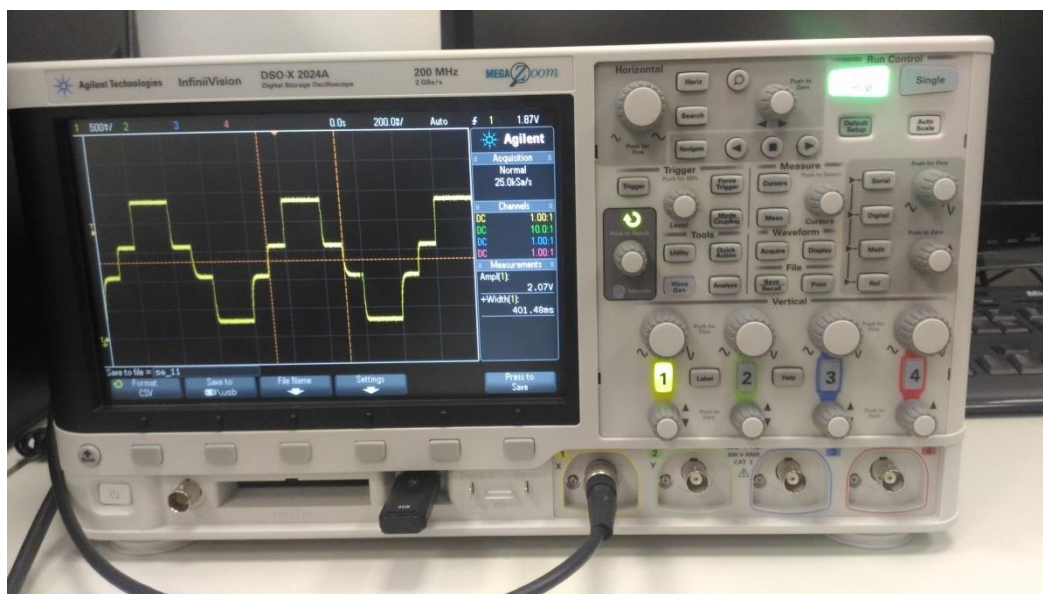


Figura 30: Foto do osciloscópio exibindo a curva dos valores enviados por SMS.

Os testes seguintes consistiram em testar os limites mínimo e máximo para saída da *EasyPIC v7*. Para isso foi enviada uma onda quadrada utilizando a mensagem: 000008888. O resultado é apresentado na Figura 31.

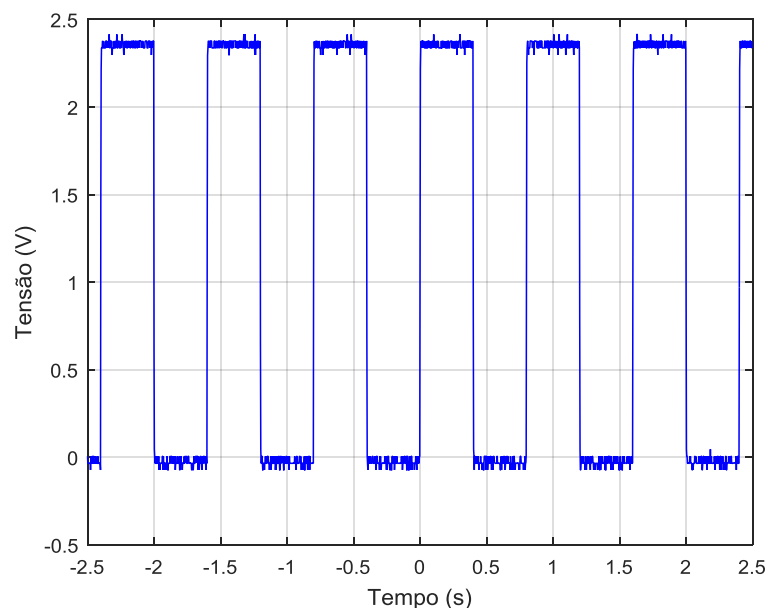


Figura 31: Onda quadrada utilizando a mensagem: 000008888.

Conforme a imagem acima, o limite inferior está em 0V e o superior cerca de 2.4V.

Após os testes de SMS foi verificado que é possível gerar os sinais de controle para aerogeradores através do envio de SMS. No entanto, o envio de mensagens de texto está limitado pela frequência que as mensagens podem ser recebidas e processadas. Para entender melhor essa limitação, foram realizados testes de frequência de envio.

Os testes de frequência de envio foram realizados para duas situações distintas: um *Script* automatizado envia mensagens consecutivas e um celular enviando mensagens individuais e aguardando uma resposta para que seja efetuado um novo envio.

O envio de mensagens consecutivas automaticamente foi feito por meio do *Script SendSequence.sh*, abaixo.

```
while read p; do
    sudo ./OpenBTCLI -c sendsms 216703111418434 +5511952475323 "$p"
    sleep 8
done<$1
```

O *Script* acima abre um arquivo passado por argumento durante a chamada, utiliza um laço *While* para ler linha a linha o arquivo do argumento e executa o

comando de envio de SMS apresentado anteriormente. Após cada envio, o comando *Sleep* é executado para adicionar uma espera antes do envio do próximo SMS.

Verificou-se a necessidade de aguardar algum tempo antes de novo envio após alguns testes. O envio quase simultâneo de mensagens sobrecarrega o *SMQueue*, de modo que nem todas as mensagens são de fato entregues. Essa falha, associada à implementação do OpenBTS, limitou a frequência dos envios.

A partir desta análise do tempo de envio, procurou-se efetuar diversos testes, com diferentes esperas, para alcançar um valor seguro de envio automatizado. Esse valor serviria de referência para um controle automatizado real, de modo que somente pudesse ser enviada uma nova mensagem se o tempo de segurança fosse respeitado.

O primeiro teste foi efetuado com uma espera de 8 segundos e 10 mensagens enviadas. A Figura 32 apresenta o instante em que foi disparada a mensagem e o tempo decorrido até o seu recebimento para envios espaçados de 8 segundos.

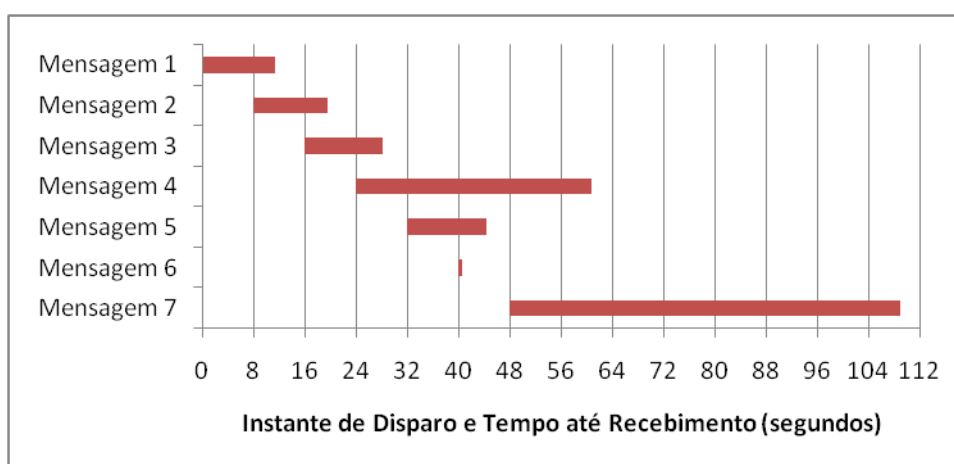


Figura 32: Instante de disparo e tempo decorrido até recebimento para envios espaçados de 8 segundos.

Os valores cronometrados são apresentados na Tabela 2.

Tabela 2: Tempo entre envio e recebimento para envios espaçados de 8 segundos.

índice	tempo decorrido (s)
1	11.31
2	11.43
3	12.17
4	36.68
5	12.39
6	0.59
7	61.01

Com este teste, o tempo médio entre recebimentos ficou em 20,80 segundos, com um desvio padrão da média de 16,03 segundos. A Tabela 2 mostra também que o tempo entre os recebimentos, até a terceira mensagem, é aparentemente constante. No entanto, a partir deste ponto a quarta mensagem demorou cerca de 3 vezes o tempo das anteriores. Além disso, entre a quinta e a sexta mensagem, não houve nem mesmo um segundo entre os envios. E, após um minuto, chegou a sétima mensagem. O restante das mensagens não chegou ou não foi processado pela *EasyPIC*.

O próximo teste utilizou um intervalo de 10 segundos entre as transmissões das mensagens e os resultados podem ser observados na Figura 33 e na Tabela 3.

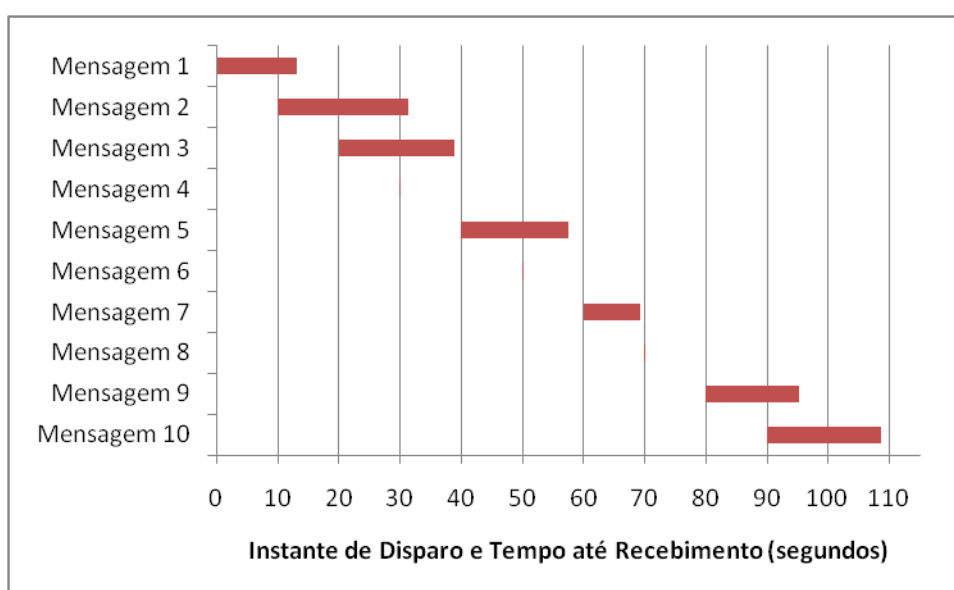


Figura 33: Instante de disparo e tempo decorrido até recebimento para envios espaçados de 10 segundos.

Tabela 3: Tempo entre envio e recebimento para envios espaçados de 10 segundos.

índice	tempo decorrido (s)
1	13.16
2	21.35
3	18.85
4	0.18
5	17.59
6	0.18
7	9.23
8	0.20
9	15.22
10	18.65

Com este teste, o tempo médio entre recebimentos ficou em 11,46 segundos, com um desvio padrão da média de 7,21 segundos. O tempo médio foi muito menor. Isso ocorreu porque muitas mensagens chegaram quase simultaneamente, resultando em registros como o da mensagem de número 4, que chegou cerca de 0,18 segundos após a mensagem 3. No entanto, diferente do teste anterior, todas as mensagens foram recebidas.

No teste seguinte este tempo entre transmissões foi ampliado para 15 segundos. Os resultados podem ser observados na Figura 34 e na Tabela 4.

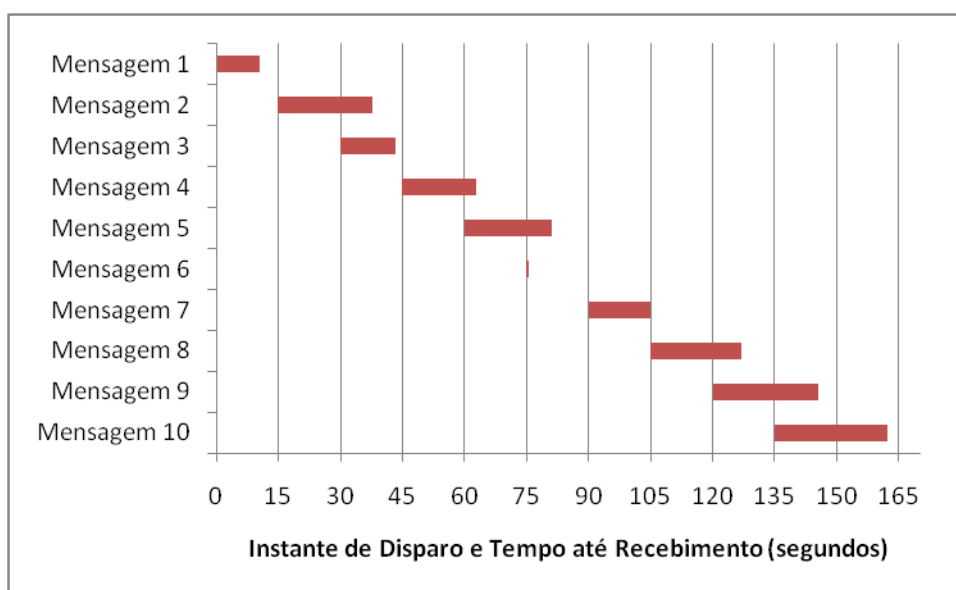


Figura 34: Instante de disparo e tempo decorrido até recebimento para envios espaçados de 15 segundos.

Tabela 4: Tempo entre envio e recebimento para envios espaçados de 15 segundos.

índice	tempo decorrido (s)
1	10.50
2	22.84
3	13.20
4	17.75
5	21.15
6	0.40
7	14.91
8	22.07
9	25.58
10	27.08

Com este teste, o tempo médio entre recebimentos ficou em 17,55 segundos, com um desvio padrão da média de 6,24 segundos. Os resultados para este teste foram semelhantes aos do teste de 10 segundos. Algumas mensagens chegam juntas, mas todas chegaram. Observou-se ainda, conforme esperado, que o tempo de espera fosse maior para as mensagens depois da primeira, pois deve-se levar em conta o tempo decorrido para que o envio fosse acionado pelo *Script*. A vantagem observada entre este teste e o anterior é que apenas uma mensagem chegou quase simultaneamente à anterior.

O último teste efetuado aumentou o intervalo entre transmissões para 20 segundos. O tempo é alto, porém, conforme Figura 35 e Tabela 5, pode-se verificar que os envios alcançaram um tempo de espera regular e sem recebimentos simultâneos.

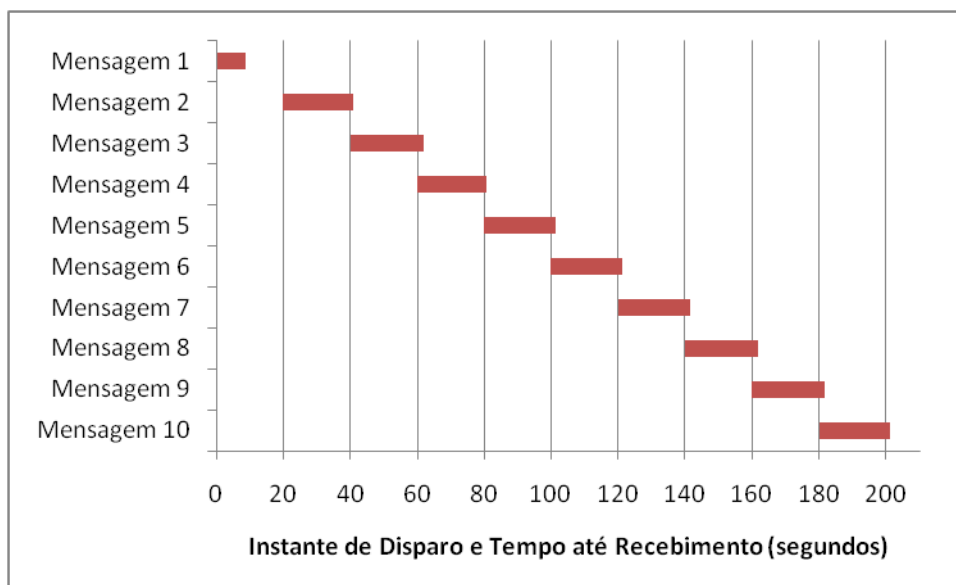


Figura 35: Instante de disparo e tempo decorrido até recebimento para envios espaçados de 20 segundos.

Tabela 5: Tempo entre envio e recebimento para envios espaçados de 20 segundos.

índice	tempo decorrido (s)
1	8.73
2	20.93
3	22.04
4	20.83
5	21.49
6	21.16
7	21.53
8	21.62
9	21.58
10	21.26

Com este teste, o tempo de envio médio ficou em 20,17 segundos, com um desvio padrão da média de 2,27 segundos. O tempo de espera de 20 segundos, apesar de influenciar diretamente no tempo decorrido para o recebimento, mostrou-se estável, com baixo desvio padrão, e sem apresentar recebimentos quase simultâneos.

Por fim, foram realizados testes de envio individuais utilizando um celular. Estes testes têm por objetivo verificar se há um acréscimo significativo no tempo de recebimento se for utilizado um dispositivo conectado à rede para enviar mensagens no lugar do OpenBTS. Espera-se que o tempo para o recebimento, neste caso, seja

maior, uma vez que a mensagem é criada em um celular, enviada à BTS, processada e enviada ao destinatário. Este processo envolve o envio de mensagens para a BTS, o que não ocorre se o SMS for gerado internamente pelo OpenBTS.

Foram feitos dez envios individuais e o tempo de recebimento foi medido. Os resultados são apresentados na Tabela 6.

Tabela 6: Tempo de recebimento considerando o envio a partir de um celular.

índice	Tempo até recebimento (s)
1	19.7
2	18.17
3	18.07
4	17.5
5	15.8
6	17.85
7	12.36
8	12.8
9	21.86
10	12.21

Neste caso, verificou-se uma média de 16,63 segundos e desvio padrão da média de 2.67 segundos. Para comparar este resultado com o obtido gerando mensagens internamente pelo OpenBTS, deve-se considerar somente a primeira mensagem enviada. A primeira mensagem é a única que garante que o SMQueue não esteja sobrecarregado, tornando os resultados comparáveis aos testes com o celular, em que as mensagens foram enviadas separadamente.

A conclusão destes testes é que o envio de SMS pode levar segundos para ser recebido e processado. Outras limitações encontradas, como o envio consecutivo de mensagens, têm relação com a implementação do OpenBTS e o computador utilizado, que sofre sobrecarga mesmo sob uma carga baixa de envios. Esse problema poderia ser reduzido com a utilização de um computador dedicado à execução do OpenBTS, ou por meio de melhorias na forma que o *SMQueue* trata os envios.

Outra forma de melhorar o desempenho, mas que deveria ser balanceada com o tempo de envio e recepção, seria aumentar o tempo coberto por um único SMS. Em vez de cobrir 100ms, por exemplo, poderia cobrir um intervalo de tempo maior. Desse modo, os intervalos de espera entre envio de SMS seriam menos

relevantes. No entanto, esse aumento pode acarretar um aumento do número de caracteres enviados, caso seja necessário aumentar a resolução temporal do sinal enviado, ou seja, representações que sejam inferiores a intervalos de 100ms. Com o aumento do tamanho do SMS, o tempo de envio também seria alterado.

Outra alternativa, seria a utilização da internet para um fluxo mais contínuo de dados. No caso do GSM poderia ser testada a possibilidade de utilizar o GPRS ou EDGE nesta aplicação.

6. CONCLUSÕES

Neste trabalho foi abordado o controle de potência gerada por turbinas eólicas, um problema que faz parte da integração de elementos da rede elétrica em *Smart Grids*.

O modelo para controle estudado consistiu em utilizar uma rede móvel GSM para realizar a interface entre os aerogeradores e outros pontos da rede de energia elétrica, como uma central de controle.

Dessa forma, foram abordados sistemas de SDR para a implementação de uma rede GSM. Diversos testes considerando as camadas físicas e também o envio de SMS para controle foram efetuados, concluindo de modo satisfatório que o envio pode ser feito utilizando a arquitetura de uma rede GSM. Além disso, os testes se estenderam a de fato controlar sinais elétricos. Apesar das limitações de tempo de reenvio de mensagem, em uma situação real, poderia ser investigada a possibilidade de utilizar mensagens mais extensas que cobririam um intervalo de tempo maior, permitindo que uma espera como a observada fosse menos relevante. Além disso, o GPRS, que traria um fluxo contínuo e de maior taxa de transmissão, poderia ser testado e utilizado.

Por fim, este trabalho cobriu o objetivo proposto, na qual deveria ser testada a utilização de redes GSM para controle de aerogeradores.

REFERÊNCIAS BIBLIOGRÁFICAS

- 1 SAUTER, Martin. From GSM to LTE-Advanced: an Introduction to Mobile Networks and Mobile Broadband. 2. ed. New Jersey: John Wiley & Sons, 2014. 456 p.
- 2 TUTTLEBEE, Walter H. W. (Ed.). Software Defined Radio: Enabling Technologies. New Jersey: John Wiley & Sons, 2002. 440 p.
- 3 GNURADIO. About GNU Radio. 2016. Disponível em: <<http://gnuradio.org/about/>>. Acesso em: 5 maio 2016.
- 4 PULGARIN, Jorge E.; PARRA, Esteban; QUINTERO, Alexander Galvis. GSM System Implementation for Academic Purposes. 2013 IEEE Colombian Conference on Communications and Computing (colcom), [s.l.], maio 2013. Institute of Electrical & Electronics Engineers (IEEE).
- 5 S., Yuva Kumar et al. Cognitive GSM OpenBTS. 2014 IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems, [s.l.], p.529-530, out. 2014. Institute of Electrical & Electronics Engineers (IEEE).
- 6 TSAI, Chun-wei et al. A Brief Introduction to Classification for Smart Grid. 2013 IEEE International Conference on Systems, Man, and Cybernetics, [s.l.], p.2905-2909, out. 2013. Institute of Electrical & Electronics Engineers (IEEE).
- 7 KONG, Fanxin et al. Blowing hard is not all we want: Quantity vs quality of wind power in the smart grid. Ieee Infocom 2014 - Ieee Conference On Computer Communications, [s.l.], p.2813-2821, abr. 2014.
- 8 GUNGOR, Vehbi C. et al. Smart Grid Technologies: Communication Technologies and Standards. IEEE Transactions On Industrial Informatics, [s.l.], v. 7, n. 4, p.529-539, nov. 2011.
- 9 VITAS, I.; SIMUNIC, D.; KNEZEVIC, P. Evaluation of Software Defined Radio Systems for Smart Home Environments. 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (mipro), [s.l.], p.562-565, maio 2015
- 10 PAPANIMITRIOU, Georgios I.; POMPORTSIS, Andreas S.; NICOPOLITIDIS, P.. Wireless Networks. New Jersey: John Wiley & Sons, 2002. 424 p.
- 11 POOLE, Ian. Chapter 6 – GSM. In: POOLE, Ian. Cellular Communications Explained: From Basics to 3G. Newnes, 2005. p. 79-102.

- 12 MSHVIDOBADZE, Tinatin. Evolution mobile wireless communication and LTE networks. 2012 6th International Conference on Application of Information and Communication Technologies (aict), [s.l.], p.1-7, out. 2012.
- 13 YOUNG, W. R.. Advanced Mobile Phone Service: Introduction, Background, and Objectives. Bell System Technical Journal, [s.l.], v. 58, n. 1, p.1-14, jan. 1979.
- 14 BLECHER, F.h.. Advanced mobile phone service. IEEE Trans. Veh. Technol., [s.l.], v. 29, n. 2, p.238-244, maio 1980.
- 15 WEY, Jyhi-kong; CHANG, Han-tsung; SUN, Lir-fan. Clone terminator: an authentication service for advanced mobile phone system. 1995 IEEE 45th Vehicular Technology Conference. Countdown to The Wireless Twenty-first Century, [s.l.], v. 1, n. 1, p.175-179, 25 jul. 1995.
- 16 DAHLMAN, E. et al. The 3G Long-Term Evolution - Radio Interface Concepts and Performance Evaluation. 2006 IEEE 63rd Vehicular Technology Conference, [s.l.], v. 1, n. 1, p.137-141, 7 maio 2006.
- 17 APPLIED ELECTRONICS (AE), 2011 INTERNATIONAL CONFERENCE ON, 1., 2011, Pilsen. Applied Electronics (AE), 2011 International Conference on. Pilsen: IEEE, 2011. 4 p.
- 18 CHITRAPU, Prabhakar; AGHILI, Behrouz. Evolution of GSM into the Next Generation Wireless World. 2007 IEEE Long Island Systems, Applications and Technology Conference, [s.l.], p.1-10, maio 2007.
- 19 SHARANAGOUDA, Patil N.; P.V., Hunagund; R.M., Vani. An efficient implementation of Software Defined radio (SDR) using RTL-SDR & GNU radio for spectrum sensing. Indian Journal Of Scientific Research, Kalaburagi, p.289-292, 01 nov. 2015.
- 20 NUAND. BladeRF. 2017. Disponível em: <<https://www.nuand.com/>>. Acesso em: 26 mar. 2017.
- 21 A BOETTCHER, M; BUTT, B M; KLINKNER, S. Low-cost approach for a software-defined radio based ground station receiver for CCSDS standard compliant S-band satellite communications. Iop Conference Series: Materials Science and Engineering, [s.l.], v. 152, p.1-10, out. 2016.

22ZOELLNER, J. et al. A generic C++ toolkit for the development of real-time-capable software defined radio applications. 2014 IEEE International Conference On Consumer Electronics (icce), [s.l.], p.472-475, jan. 2014.

23TUCKER, D. Casey; TAGLIARINI, Gene A.. Prototyping with GNU radio and the USRP - where to begin. IEEE Southeastcon 2009, [s.l.], p.50-54, mar. 2009.

24CARDOSO, Jaqueline Gomes. COMUNICAÇÃO SEM FIO APLICADA AO CONTROLE DAS POTÊNCIAS DE AEROGERADORES DE INDUÇÃO CONECTADOS À REDE ELÉTRICA VISANDO APLICAÇÕES EM SMART GRIDS. 2016. 122 f. Dissertação (Mestrado) - Curso de Pós-graduação em Engenharia Elétrica, Centro de Engenharia, Modelagem e Ciências Aplicadas, Universidade Federal do Abc, Santo André, 2016.

25BROEK, Fabian van Den. Catching and Understanding GSM-Signals. 2010. 113 f. Tese (Doutorado) - Curso de Computer Science, Radboud University Nijmegen, Nimegue, 2010.

26REIS, Andre Luiz Garcia et al. Introduction to the Software-defined Radio Approach. Ieee Latin America Transactions, [s.l.], v. 10, n. 1, p.1156-1161, jan. 2012.

27VEA, Karl David. NUTS: Ground station with GNU Radio and USRP. 2015. 41 f. Tese (Doutorado) - Curso de Master Of Science In Electronics, Department Of Electronics And Telecommunications, Norwegian University Of Science And Technology, Trondheim, 2015.

28 OPENBTS. OpenBTS About Page. Disponível em: <<http://openbts.org/about/>>. Acesso em: 5 maio 2016.

29 Hashmi, M.; Hanninen, S.; Maki, K. Survey of smart grid concepts, architectures, and technological demonstrations worldwide, Innovative Smart Grid Technologies (ISGT Latin America), 2011 IEEE PES Conference on , vol., no., pp.1,7, 19-21. Medellin, 2011.

30 OPENBTS APPLICATION SUITE: User Manual. Range Networks, 2014.

31 Rondeau, T. PYBOMBS. Disponível em: <<http://gnuradio.org/redmine/projects/pybombs/wiki/Wiki/2>>. Acesso em: 20 ago. 2016.

- 32 ABOUT RTL-SDR. Disponível em: <<http://www.rtl-sdr.com/about-rtl-sdr/>>. Acesso em: 12 ago. 2016.
- 33 IEDEMA, Michael. Getting Started with OpenBTS. California: O'reilly, 2015. 122 p.
- 34 NUAND. Nuand. Getting Started: Linux: Easy installation for Ubuntu: The bladeRF PPA. 2016. Disponível em: <<https://github.com/Nuand/bladeRF/wiki/Getting-Started:-Linux>>. Acesso em: 08 mar. 2017.
- 35 ETTUS. USRP Hardware Driver and USRP Manual. Disponível em: <https://files.ettus.com/manual/page_install.html>. Acesso em: 08 mar. 2017.
- 36 GitHub. About. Disponível em: <<https://github.com/about>>. Acesso em: 08 mar. 2017.
- 37 Yate. Why Yate. Disponível em: <<http://yate.ro/products.php?page=yate>>. Acesso em: 08 mar. 2017.
- 38 EasyPIC v7. Disponível em: < <https://shop.mikroe.com/development-boards/full-featured/easy-boards/easypic>>. Acesso em: 19 mar. 2017.
- 39 GL865-DUAL/QUAD V3 Product Description. Disponível em: <http://www.telit.com/fileadmin/user_upload/products/Downloads/2G/CL865/Telit_GL865-DUAL_QUAD_V3_Product_Description_r6.pdf>. Acesso em: 29 mar. 2017.

APÊNDICE A - Instalação do GNR via *Pybombs*

Para instalar o *Pybombs*, é preciso que o ambiente *Python* esteja instalado, assim como o seu instalador, o *Pip*. O *Pip* permite instalar bibliotecas à linguagem *Python* por meio de linha de comando. Esta ferramenta é utilizada para resolver uma única dependência que pode não estar presente na instalação padrão do *Ubuntu*, que foi identificada pelo aluno. Abaixo, o comando utilizado para resolver esta dependência.

```
pip install mako
```

O próximo passo consiste na instalação em si do *Pybombs*. O primeiro comando a ser utilizado, portanto, é o de copiar, ou clonar, os arquivos do projeto provenientes da página do desenvolvedor. Para isso existe um comando que permite copiar um projeto inteiro a partir de sua página no repositório *GitHub*. O comando utilizado para isso se encontra abaixo.

```
git clone https://github.com/gnuradio/pybombs
```

O ideal para esta instalação é que seja feita dentro do diretório *home* do usuário. Isso evita que problemas de permissão ocorram e prejudiquem a instalação ou até mesmo o funcionamento das ferramentas. Caso o usuário desconheça este diretório, o mesmo pode ser obtido ao utilizar os comandos abaixo no terminal.

```
cd ~  
pwd
```

O primeiro comando irá direcionar o terminal do usuário para o diretório *home*, o símbolo '~' indica isso ao comando *cd*. O comando seguinte é o *pwd* (*Print Working Directory*) imprime na tela o diretório atual. Ou seja, ao término desses dois comandos o usuário irá saber o seu diretório *home* e pode efetuar a instalação do *Pybombs* dentro do mesmo.

Para prosseguir com a instalação, entre no diretório do *Pybombs* e utilize os comandos abaixo.

```
python setup.py install
cd ..
```

O primeiro comando executa um *ScriptPython* para a instalação do *Pybombs*. O mesmo pode falhar devido a algum erro de permissão associado aos arquivos do *Python*. Caso isto ocorra, utilize o comando *'sudo'* no início da linha e forneça a senha solicitada. O comando seguinte, *'cd ..'*, é responsável por subir um nível nos diretórios, saindo do diretório de instalação do *Pybombs*.

Como citado anteriormente, o *Pybombs* possui receitas de instalação para as diferentes ferramentas. Neste caso, será feita a instalação do GNR completa, incluindo todas os drivers para utilização de sistemas de SDR. A instalação das receitas foi efetuada conforme os comandos abaixo.

```
pybombs recipes add gr-recipes git+https://github.com/gnuradio/gr-
recipes.git
pybombs recipes add gr-etcetera git+https://github.com/gnuradio/gr-
etcetera.git
```

Após os comandos acima é preciso criar um ambiente virtual para a instalação das ferramentas cujas receitas foram obtidas.

Um ambiente virtual significa que a instalação não precisa ser única no sistema operacional. Ela será acessível somente quando o usuário entrar no ambiente virtual. Essa metodologia permite que diferentes versões de um mesmo *Software* estejam instaladas em um mesmo computador, sem que ocorram conflitos.

Desse modo, é preciso criar um diretório para este ambiente virtual e somente após sua criação efetuar a instalação do GNR. Uma sugestão de criação do diretório para o ambiente virtual seria um novo diretório dentro do *home* do usuário, como por exemplo, *'/home/user_name/gnuprefix'*, sendo *user_name* o nome do usuário dentro do sistema operacional. Tendo criado o diretório citado, o mesmo deve ser utilizado no comando abaixo, na qual a instalação do GNR é feita dentro do ambiente virtual recém-criado.

```
pybombs prefix init /home/user_name/gnuprefix/ -a myprefix -R gnuradio-
default
```

Este comando, por se tratar da instalação em si, pode levar minutos para que seja finalizado. Após seu término, o usuário pode ter acesso às ferramentas instaladas entrando no diretório no qual a instalação foi efetuada e executando o comando abaixo.

```
./setup_env.sh
```

Após este comando, é possível executar qualquer uma das ferramentas instaladas. Este *Script* executado exporta os programas instalados para a variável *PATH* do *Ubuntu*, tornando os comandos acessíveis via terminal, porém somente para a sessão em que o *Script* foi executado, fazendo com que o acesso aos *Softwares* instalados seja apenas temporário, pois o fechamento da sessão finaliza também o acesso aos *Softwares* instalados. Essa abordagem para utilização das ferramentas é que torna a instalação um ambiente virtual.

Com todas as ferramentas instaladas, o trabalho foi continuado. As primeiras tarefas realizadas, como será descrito a seguir, consistiram em testes das ferramentas e do *Hardware* utilizados para o desenvolvimento deste trabalho de graduação.

APÊNDICE B - Instalação do OpenBTS

Instalação dos drivers da BladeRF

A instalação da BladeRF no Ubuntu pode ser feita por meio de um pacote de arquivo pessoal (PPA – *Personal Package Archive*) [34]. Para isto, basta ativar o PPA dentro da distribuição do usuário utilizando os comandos abaixo.

```
sudo add-apt-repository ppa:bladerf/bladerf
sudo apt-get update
```

O primeiro comando ativa o PPA da BladeRF e o segundo atualiza a lista de aplicações e versões do gerenciador do Ubuntu. Em seguida foi instalada a BladeRF e algumas bibliotecas necessárias para a utilização com GNR, entre outras aplicações. Os comandos utilizados seguem abaixo.

```
sudo apt-get install bladerf
sudo apt-get install libbladerf-dev
```

Além da instalação dos *Drivers*, é preciso instalar também a imagem do FPGA compatível com o dispositivo. Para que isso fosse possível, o fabricante disponibiliza as imagens e também uma interface para sua instalação. Com o arquivo da imagem no computador, o comando abaixo permite que ela seja instalada dentro da BladeRF.

```
bladeRF-cli -l bladerf-fpga-hostedx115
```

Após o término deste procedimento é possível utilizar a BladeRF.

Instalação do OpenBTS

Devido a incompatibilidade da BladeRF com o openBTS, sua instalação será feita manualmente, ou seja, sem o auxílio dos *Scripts* criados para facilitar a instalação deste software em distribuições conhecidas, como o Ubuntu.

Para melhorar a descrição, a sequência de procedimentos para a instalação é apresentada abaixo:

1. Resolução de dependências dos softwares a serem instalados;
2. *Download* dos *Softwares* a partir de seus respectivos repositórios;
3. Compilação do transmissor a ser utilizado com o OpenBTS;
4. Efetuar o *Link* entre o transmissor e o OpenBTS, substituindo o transmissor do OpenBTS;
5. Compilar os componentes extras necessários (o OpenBTS acompanha componentes separados para gerir usuários e a fila de SMS, por exemplo);
6. Compilar o OpenBTS;

A lista de dependências que devem ser conferidas e instaladas no sistema estão todas contidas dentro do comando abaixo.

```
sudo apt-get install software-properties-common python-software-properties
autoconf automake libtool debhelper sqlite3 libsqlite3-dev libusb-1.0-0
libusb-1.0-0-dev libortp-dev libortp8 libosip2-dev libreadline-dev
libncurses5 libncurses5-dev pkg-config cdb libsqlite0-dev unixodbc
unixodbc-dev libssl-dev libsctp0 libsctp0-dev libsqliteodbc libzmq3-dev
libzmq3 python-zmq uhd
```

Dependendo do sistema, algumas dessas dependências podem não ser instaladas corretamente. Por este motivo, apesar do comando poder ser executado de uma vez, é interessante executá-lo em pequenos blocos para poder observar se todos os elementos foram instalados corretamente.

Alguns destes elementos podem ser de difícil instalação independente do sistema. Dentre elas, vale citar o libortp8 e libosip2-dev. Essas bibliotecas são nativas do repositório padrão do Ubuntu 12.04. No entanto, não estão presentes em versões mais recentes como o Ubuntu 14.04. Para que este problema seja resolvido, basta adicionar o repositório do Ubuntu 12.04 à versão sendo utilizada e em seguida instalando as bibliotecas, conforme os comandos abaixo.

```
deb http://us.archive.ubuntu.com/ubuntu precise main universe
sudo apt-get update
sudo apt-get install libortp8 libosip2-4
```

Outro problema que pode ser encontrado ao resolver as dependências faz referências às bibliotecas zmq. Muitas vezes as versões presentes no repositório

padrão não representam a versão mais recente. Para que seja possível ter acesso a esta versão, é preciso importar o repositório oficial da biblioteca e somente depois instalá-la. Os comandos abaixo possibilitam estas ações.

```
sudo add-apt-repository ppa:chris-lea/zeromq;  
sudo apt-get update;  
sudo apt-get install libzmq3-dbg libzmq3-dev
```

A última dependência que geralmente causa problemas durante a sua instalação é a UHD (*USRP Hardware Driver*). O ideal para este caso é seguir o manual de instalação da ferramenta diretamente do site da *Ettus* [35], que possui instruções específicas para cada distribuição Linux. No caso do Ubuntu, é possível utilizar tanto o repositório padrão quanto o repositório da *Ettus*, contendo a versão mais recente. Para o segundo caso, os comandos abaixo instalam o UHD e suas dependências.

```
sudo add-apt-repository ppa:ettusresearch/uhd  
sudo apt-get update  
sudo apt-get install libuhd-dev libuhd003 uhd-host
```

Finalizada a instalação das dependências, os softwares que serão utilizados em conjunto ao OpenBTS para prover seu total funcionamento podem ser obtidos a partir de um repositório de projetos que utilizam um controle de versionamento Git, chamado *GitHub* [36]. Criado um diretório onde todos estes elementos serão instalados, este deve ser acessado via terminal e os comandos abaixo executados, para copiar os repositórios.

```
git clone https://github.com/RangeNetworks/openbts.git;  
git clone https://github.com/RangeNetworks/smqueue.git;  
git clone https://github.com/RangeNetworks/subscriberRegistry.git
```

Os comandos acima copiam para um diretório local o OpenBTS, o smqueue, responsável por administrar os SMS's e o *Subscriber Registry*, responsável por controlar os usuários da rede.

Além disso, um módulo para compartilhar e interagir com as informações de usuários na rede deve ser copiado dentro de cada um desses três componentes.

Estes módulos são chamados *CommonLibs* e *NodeManager*. O script *Bash* abaixo permite efetuar a cópia dentro de cada um dos diretórios automaticamente.

```
for D in *; do(
    echo$D;
    echo"=====";
    cd$D;
    git clone https://github.com/RangeNetworks/CommonLibs.git;
    git clone https://github.com/RangeNetworks/NodeManager.git);
done
```

Os recursos acima ainda necessitam de algumas bibliotecas para que sejam compilados. Essas duas bibliotecas, diferente das outras dependências, também são projetos armazenados no *GitHub*. Deve-se, portanto, copiá-las e compilá-las. Para copiá-las, basta utilizar os comandos abaixo.

```
git clone https://github.com/RangeNetworks/libcoredumper.git;
git clone https://github.com/RangeNetworks/liba53.git
```

Para compilar tais bibliotecas, o *Script Bash* abaixo possibilita fazer isto automaticamente.

```
cd libcoredumper;
./build.sh &&\
sudo dpkg -i *.deb;
cd ../liba53;
make &&\
sudo make install;
cd ..
```

Com todos os softwares copiados localmente, antes de compilar seu código-fonte, conforme apresentado, é preciso de um transmissor que seja compatível com a BladeRF. O transmissor compatível encontrado está dentro de um *Software* chamado YateBTS [37].

Yate é chamado pelos desenvolvedores de um motor telefônico (*Telephone Engine*), utilizado para implementar e interagir diversos *Softwares* e tecnologias, como GSM e LTE. Um destes *Softwares*, o YateBTS, implementa a rede de acesso GSM/GPRS baseado no núcleo do Yate. Sendo uma implementação da rede de acesso, o YateBTS apresenta um transmissor e todos os protocolos entre a interface

do *Mobile Terminal* e a BTS. Dentro desta implementação, foi preciso apenas o transmissor para a BladeRF.

Para utilizar o YateBTS, foi preciso acessar suas versões por meio do SVN (*Subversion*), e não o *GitHub*. Primeiramente, o SVN foi instalado utilizando o comando abaixo.

```
sudo apt-get install subversion
```

O *Download* do YateBTS é simples e também se resume a um único comando.

```
svn checkout http://voip.null.ro/svn/yatebts/trunk yatebts
```

Com o intuito de aproveitar somente os elementos necessários do YateBTS, alguns ajustes tiveram de ser feitos. O primeiro deles é remover o carregamento do FPGA sempre que o YateBTS for inicializado. Como o FPGA padrão, já instalado, não é removido do dispositivo de SDR, gravá-lo sempre que for utilizado não é necessário.

Para impedir que estes comandos sejam incluídos na compilação do YateBTS, o comando abaixo cerca o carregamento do FPGA por uma definição de pré-processamento.

```
vim ./yatebts/mbts/TransceiverRAD1/bladeRFDevice.cpp "+112s/^/#ifdef  
NEVER" +133s/^/#endif +wq
```

Dentro do YateBTS existe um *Script* que irá gerar o arquivo de configuração para compilar o *Software*. Para executá-lo foram utilizados os comandos abaixo.

```
cd yatebts  
./autogen.sh
```

O arquivo de configuração gerado por este *Script* é utilizado para compilar todos os elementos do YateBTS. Como existe interessante apenas nos elementos que tangem ao transmissor, demais processos foram comentados utilizando o comando abaixo.

```
vim configure +4263,4291s/^/#/ +wq
```

Após essa edição, o *Script* de configuração foi executado utilizando o comando abaixo.

```
./configure
```

Dentro da árvore de elementos do YateBTS, apenas dois foram necessários para que o transmissor funcionasse: *Peering* e *TransceiverRAD1*. Primeiramente, os comandos abaixo compilar o código-fonte em *Peering*.

```
cd mbts/Peering/
make
```

Em seguida, o *TransceiverRAD1* foi compilado conforme os comandos abaixo.

```
cd ../TransceiverRAD1
vim Makefile "+26s/$/ -lpthread/" +wq
make
cd ../../..
```

Antes de compilar utilizando o comando *Make*, o *Makefile* é editado para adicionar o *Link* a uma biblioteca na compilação dos arquivos de código-fonte. Essa adição foi verificada após alguns testes, pois sua ausência retornava erros de comandos não encontrados dentro de alguns dos arquivos escritos em C++. A necessidade de adicionar tal biblioteca é mais recente que a escrita do arquivo. O compilador presente no Ubuntu 14.04 não inclui por padrão alguns dos *Headers* necessários. Por este motivo esta referência *-lpthread* foi adicionada.

O transmissor foi compilado e teve de ser ligado ao OpenBTS, pois ainda não pertence à sua estrutura. Para isso, primeiramente, o transmissor deve ser copiado para dentro do OpenBTS utilizando o comando abaixo.

```
cp ../yatebts/mbts/TransceiverRAD1/transceiver-bladerf openbts/apps/
```

Mesmo dentro do mesmo diretório, quando o OpenBTS procura por um transmissor, irá consultar um arquivo chamado *transceiver*. O transmissor da BladeRF possui outro nome: *transceiver-bladerf*. Para que o OpenBTS acesse o arquivo correto, foi preciso utilizar um *Link* simbólico entre os arquivos. O *Link* simbólico permite que um determinado arquivo faça referência a outro arquivo quando for acessado. O comando abaixo fez com que o ao ser chamado o *transceiver*, o arquivo *transceiver-bladerf* fosse acessado.

```
ln -sf transceiver-bladerf transceiver
```

Após realizar o link simbólico acima é preciso compilar o código-fonte do OpenBTS, finalizando a instalação. A compilação deve ser feita no diretório raiz do OpenBTS utilizando os comandos abaixo.

```
cd /home/openbts/obts/openbts
./autogen.sh
./configure --with-uhd
make
```

Para executar o OpenBTS inicializa-se o smqueue, o sipauthserve e somente depois o OpenBTS. A partir do diretório atual, execute os três comandos abaixo para inicializar o OpenBTS.

```
sudo ../../smqueue/smqueue/smqueue &
sudo ../../subscriberRegistry/apps/sipauthserve &
sudo ./OpenBTS
```