INSTITUTO FEDERAL DE SANTA CATARINA

PEDRO HENRIQUE DA SILVA HAMES

Automatização de testes na camada física de equipamentos IEEE 802.11 utilizando SDR

São José - SC

7 de dezembro de 2018

AUTOMATIZAÇÃO DE TESTES NA CAMADA FÍSICA DE EQUIPAMENTOS IEEE 802.11 UTILIZANDO SDR

Trabalho de conclusão de curso apresentado à Coordenadoria do Curso de Engenharia de Telecomunicações do campus São José do Instituto Federal de Santa Catarina para a obtenção do diploma de Engenheiro de Telecomunicações.

Orientador: Roberto Wanderley da Nobrega

São José - SC

7 de dezembro de 2018

PEDRO HENRIQUE DA SILVA HAMES

AUTOMATIZAÇÃO DE TESTES NA CAMADA FÍSICA DE EQUIPAMENTOS IEEE 802.11 UTILIZANDO SDR

Este trabalho foi julgado adequado para obtenção do título de Engenheiro de Telecomunicações, pelo Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, e aprovado na sua forma final pela comissão avaliadora abaixo indicada.

São José - SC, 7 de dezembro de 2018:

Roberto Wanderley da Nobrega, Dr.
Orientador
Instituto Federal de Santa Catarina

Tiago Semprebom, Dr.
Instituto Federal de Santa Catarina

RESUMO

Nos últimos anos, a necessidade de conexão com a internet dos dispositivos (smartphones, smartTVs, câmeras de vigilância, etc.) cresceu e continuará crescendo. Para atender às expectativas dos usuários por conectividade e facilitar a instalação e a operação dos equipamentos, muitos fabricantes optaram pela tecnologia Wi-Fi para os seus produtos. Com a demanda crescente por conectividade Wi-Fi, os fabricantes de roteadores e *access points*, a fim de responder às necessidades do mercado o mais rápido o possível, estudam formas de reduzir o tempo de desenvolvimento destes equipamentos. Viu-se que boa parte deste tempo é gasto nas rotinas de testes exaustivos realizados nestes equipamentos para garantir boa performance e confiabilidade. Buscando diminuir o tempo investido para a realização da validação, este trabalho propõe um sistema baseado em SDR para automatizar os testes de canal de operação nos roteadores Intelbras Zeus. O sistema tem como objetivo executar a rotina de testes em todos os canais de operação destes equipamentos diminuindo o tempo de investido e aumentando a confiabilidade, além disso ao final, é emitido um relatório com os resultados obtidos.

Palavras-chaves: Rádio definido por software. Automatização de testes. IEEE 802.11. Mascaramento espectral.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE CÓDIGOS

LISTA DE ABREVIATURAS E SIGLAS

ADC *analog to digital converter* 19

Anatel Agência Nacional de Telecomunicações 23

API *application program interface* 15

DHCP *Dynamic Host Configuration Protocol*

Fc Frequência central

FFT *fast Fourier transform* 27

FI frequência intermediária

HTTP *Hypertext Transfer Protocol*

IEEE *Institute of Electrical and Electronics Engineers*

IP *Internet Protocol*

ISM *Industrial Scientific and Medical* 23

JSON *JavaScript Object Notation*

QoS *quality of service*

RF *radio frequency* 5

SDR *software defined radio* 15

SMA *SubMiniature version A*

SNMP *Simple Network Management Protocol*

TCC trabalho de conclusão de curso 15

URI *Uniform Resource Identifier*

USB *Universal Serial Bus*

WLAN *Wireless Local Area Network* 23

ZEUS *Zero Effort Unified System* 5

SUMÁRIO

1 INTRODUÇÃO

A crescente demanda por conectividade Wi-Fi[1] (CISCO SYSTEMS, INC., 2016) estimulou a utilização dos padrões IEEE 802.11 (STEPHENS, 2016) em *access points*, sejam eles para conexões domésticas ou corporativas. Dado o aumento da necessidade por conta do usuário, os fabricantes deste nicho de mercado buscam cada vez mais por novidades e novas tecnologias para atender às expectativas dos consumidores.

Desenvolver equipamentos competitivos e que acompanhem o mercado é um dos principais desafios das equipes de desenvolvimento. Grande parte deste tempo se destina às rotinas criteriosas e desgastantes de testes realizados pelas equipes de confiabilidade a fim de garantir robustez e boa performance. Com dois objetivos básicos, diminuir o tempo de testes e aumentar o seu grau de confiança, é parte da rotina dos desenvolvedores criar procedimentos de testes automatizados para diminuir o tempo de validação, principalmente os daqueles massivos onde é necessário realizar uma mesma rotina repetidas vezes alterando somente um ou alguns parâmetros do equipamento ou ambiente de testes. A configuração de canal de operação e largura de banda de um *access point dual band*, por exemplo, pode chegar a mais de 250 combinações diferentes, o que torna o teste manual inviável e como alternativa fazem-se testes somente com amostras de todo o espaço amostral.

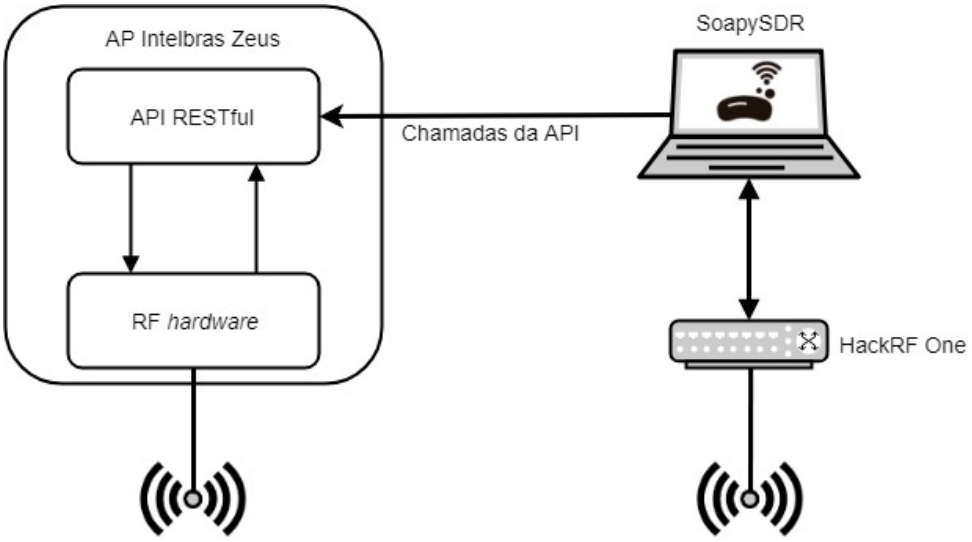
Atualmente no mercado existem equipamentos fabricados pela Agilent Technologies[2] ou Ixia Company[3] capazes de realizar os testes propostos neste trabalho de conclusão de curso (TCC) além de outros ainda mais complexos. Em contrapartida, o investimento inicial necessário é muito elevado, o que faz com que o *payback*, tempo necessário para que o uso do equipamento tenha rendimento monetário suficiente para custear o investimento inicial, seja muito elevado. Por esse motivo já no início dos estudos definiu-se que utilizar *software defined radio* (SDR) seria a melhor opção por dois fatores, o primeiro é a flexibilidade para evoluir e alterar o sistema de acordo com as necessidades de novos testes e o segundo é o valor do investimento, que reduz o *payback* do sistema consideravelmente.

Visando acompanhar as tendências do mercado de redes Wi-Fi, a Intelbras S/A[4] desenvolveu uma plataforma de desenvolvimento unificado para sistemas embarcados chamada ZEUS. Esta plataforma tem como propósito unificar o *software* embarcado de toda a linha de dispositivos de redes e é estruturada em três camadas: interface de usuário, *application program interface* (API) e sistema.

O objetivo deste TCC é, utilizando SDR, desenvolver um sistema capaz de automatizar a aferição da ocupação espectral de *access points* IEEE 802.11g/n/ac de acordo com as máscaras especificadas pela própria IEEE (STEPHENS, 2016), sistema o qual é estruturado de acordo com a Figura 1. Através deste sistema, os roteadores Wi-Fi serão submetidos à uma rotina de testes

[1] Wi-Fi *Alliance*: <https://www.wi-fi.org/>
[2] https://www.agilent.com/
[3] https://www.ixiacom.com/
[4] http://www.intelbras.com.br/

Figura 1 – Diagrama de blocos do sistema.



a fim de validar o seu funcionamento em todas as possíveis combinações de canal de operação e largura de banda, diminuindo o tempo gasto pelo validador na realização do testes e garantindo maior confiança nos resultados. Tem-se como objetivos específicos a integração deste sistema com a plataforma de desenvolvimento Intelbras ZEUS e também a elaboração de um relatório com o resultado dos testes realizados.

Componentes presentes na Figura 1:

- **AP Intelbras ZEUS:** equipamento que será submetido à rotina de testes.
- **SoapySDR:** API de código aberto para desenvolvimento de sistemas SDR responsável pela obtenção das amostras, processamento e análise do sinal obtido através do HackRF One. Além da integração com o HackRF One, este módulo do sistema também será responsável pela integração com o AP Intelbras ZEUS através de chamadas da API.
- **HackRF One:** Hardware responsável pela captura do sinal proveniente do dispositivo que será testado.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo apresentar os principais conteúdos estudados e utilizados durante o desenvolvimento deste TCC.

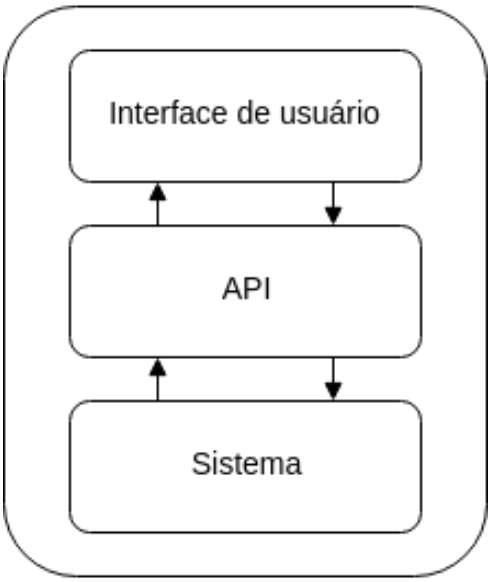
2.1 Intelbras ZEUS

Com a entrada das multinacionais no mercado brasileiro, manter-se na ponta sendo um fabricante nacional vem se tornando um difícil desafio. A Intelbras S/A, que é uma empresa 100% nacional, busca alternativas para se posicionar cada vez melhor no mercado de redes sem fio empresariais, para isso, tornou-se necessário construir uma plataforma de desenvolvimento de *software* para os produtos. Assim nasceu o Intelbras ZEUS (*Zero Effort Unified System*), a atual plataforma utilizada pela equipe de pesquisa e desenvolvimento dos produtos.

2.1.1 Arquitetura

Segundo (FRANTZ; WEIDLE, 2015), o projeto ZEUS é uma plataforma de desenvolvimento de *software* para sistemas embarcados baseada no OpenWRT[1]. Por meio dela, são gerados os *firmwares* específicos para cada equipamento da linha de redes sem fio corporativas da Intelbras S/A. A arquitetura do ZEUS é baseada em três camadas: interface de usuário, API e sistema, conforme descrito na Figura 2.

Figura 2 – Arquitetura do ZEUS.



A seguir, tem-se a descrição dos componentes presentes na Figura 2.

[1] https://openwrt.org/

- **Interface de usuário:** normalmente uma aplicação *web* ou *mobile* capaz de interagir com o usuário, seja para a exibição de dados ou o envio de informações ao dispositivo, como configurações, por exemplo.
- **API:** baseada em RESTful (FIELDING et al., 1999), esta parte da arquitetura é responsável por abstrair os recursos e funcionalidades do sistema em chamadas baseadas nos métodos HTTP: DELETE, GET, POST e PUT.
- **Sistema:** realiza as interações com o *hardware* e provê em mais alto nível para a camada de API os recursos necessários para cada funcionalidade disponibilizada ao usuário.

Com esta arquitetura baseada em camadas e as interfaces entre elas bem definidas, é possível se necessário atualizar somente uma delas, isso acontece por exemplo quando há uma troca de plataforma de *hardware*, em que somente a camada de sistema é desenvolvida novamente.

2.1.2 API pública

Já no início da construção da API do ZEUS, optou-se por tornar a sua documentação aberta à comunidade (INTELBRAS S/A, 2018), assim projetos como esse TCC poderiam ser desenvolvidos não somente pela equipe Intelbras, mas sim por quaisquer outros desenvolvedores. As chamadas da API do ZEUS são divididas em três grupos, sistema, interfaces e serviços, em que cada um deles fica responsável por gerenciar parte dos recursos do equipamento.

- **Sistema:** este grupo de chamadas possibilita ao usuário reiniciar o seu equipamento, gerenciar as contas de acesso e realizar atualizações de *firmware*, por exemplo.
- **Interfaces:** este grupo permite que o usuário configure os atributos das interfaces de rede dos seus equipamentos, como taxa e potência de transmissão, endereços IPs, etc.
- **Serviços:** com as chamadas deste grupo, o usuário pode ativar e configurar os serviços de rede do seu dispositivo, como DHCP, SNMP, QoS, *firewall*, entre outros.

Baseadas nos métodos HTTP (FIELDING; RESCHKE, 2014) RESTful, os dados enviados e recebidos necessariamente serão descritos em formato JSON (BRAY, 2017) e para ter acesso às chamadas, é preciso antes de tudo realizar a autenticação e obter um *token*, o qual terá validade somente por alguns minutos se não utilizado. Para obter um *token*, é necessário realizar uma chamada com o método POST para a URI `<IP>/system/login` enviando os dados formatado de acordo com o Código 2.1.

Código 2.1 – Dados enviados via método POST descritos em JSON.

```
{
    "data" :
    {
        "username" : "usuário",
        "password" : "senha"
    }
}
```

Como resultado, tem-se os dados descritos no Código 2.2.

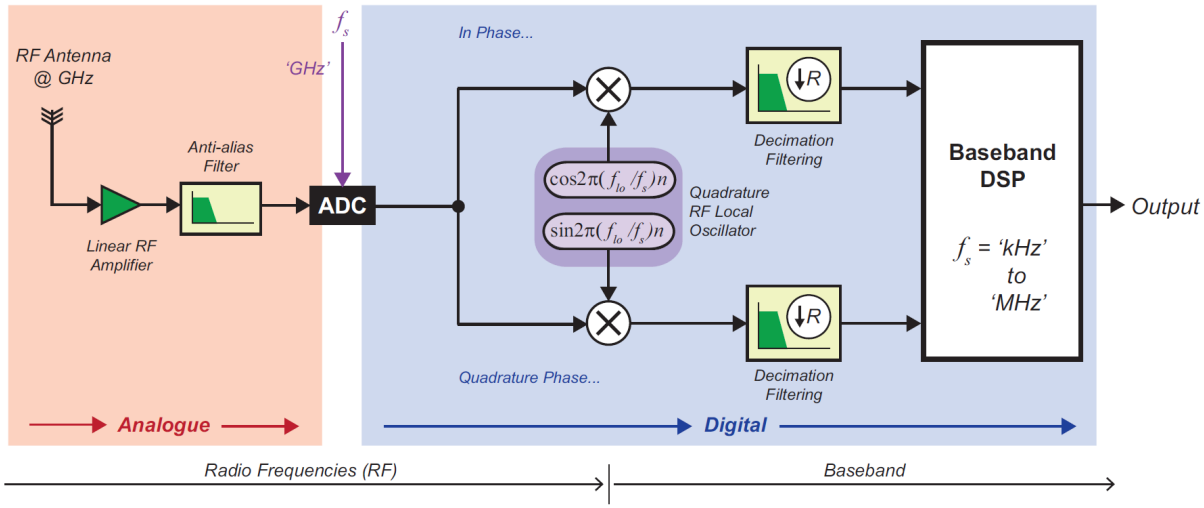
Código 2.2 – Token obtido através do método "/system/login".

```
{
    "data" :
    {
        "Token" : "e6223ae4a109705437fe69263dfb7ad3"
    }
}
```

2.2 Rádio definido por software

Rádio definido por software, ou SDR, acrônimo para o termo em inglês *software defined radio*, é uma técnica utilizada para transferir para o *software* a maior parte do processamento de sinais que em sistemas convencionais seria feita em *hardware*, como mostra a Figura 3, em que o sinal é amostrado ainda na frequência de RF. A principal vantagem destes sistemas é a alta flexibilidade, escalabilidade e portabilidade, todas estas características são provenientes do processamento de sinais realizado no domínio de *software*. Em contrapartida, a necessidade de uma frequência de amostragem extremamente elevada, na casa dos GHz, inviabiliza a aplicação prática desta topologia.

Figura 3 – Sistema de rádio com conversor em frequência de RF.

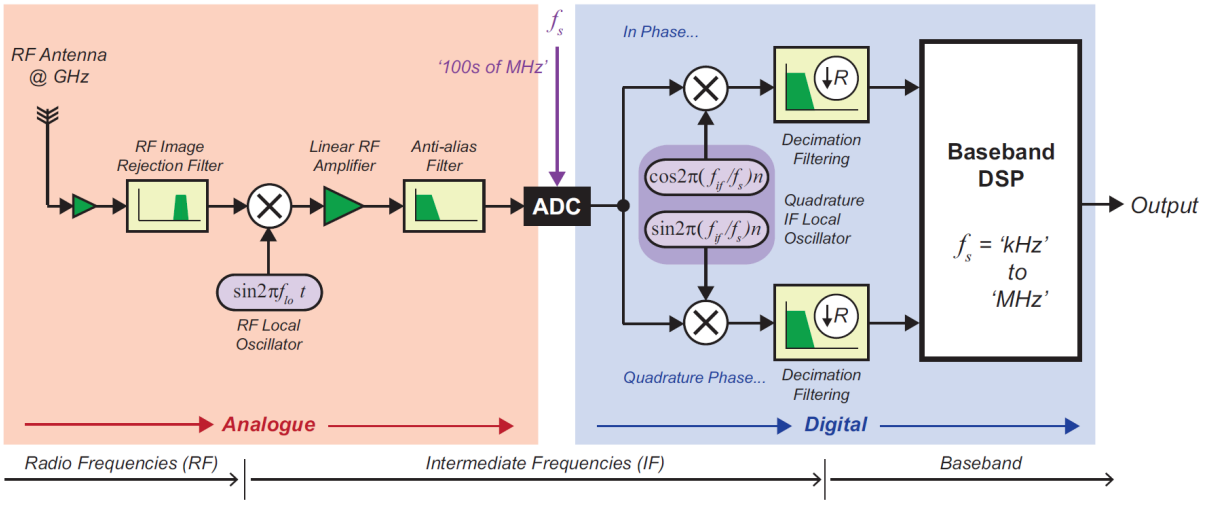


Fonte: Stewart et al. (2015)

Com o objetivo de diminuir a frequência necessária na etapa de amostragem e viabilizar a utilização dos sistemas de SDR, o diagrama da Figura 4 é exemplo de um módulo que realiza a conversão para o domínio digital em frequência intermediária. Esta característica nos permite reduzir a frequência de amostragem para cerca de centenas de MHz.

Sistemas de rádio com amostragem em banda base, assim como o representado pelo diagrama da Figura 5, ainda no domínio analógico fazem parte do processamento do sinal, como filtragem, amplificação e deslocamento em frequência, isso tanto na frequência de RF quanto nas frequências intermediárias (FI), desta forma a conversão para o domínio digital é feita pelos *analog*

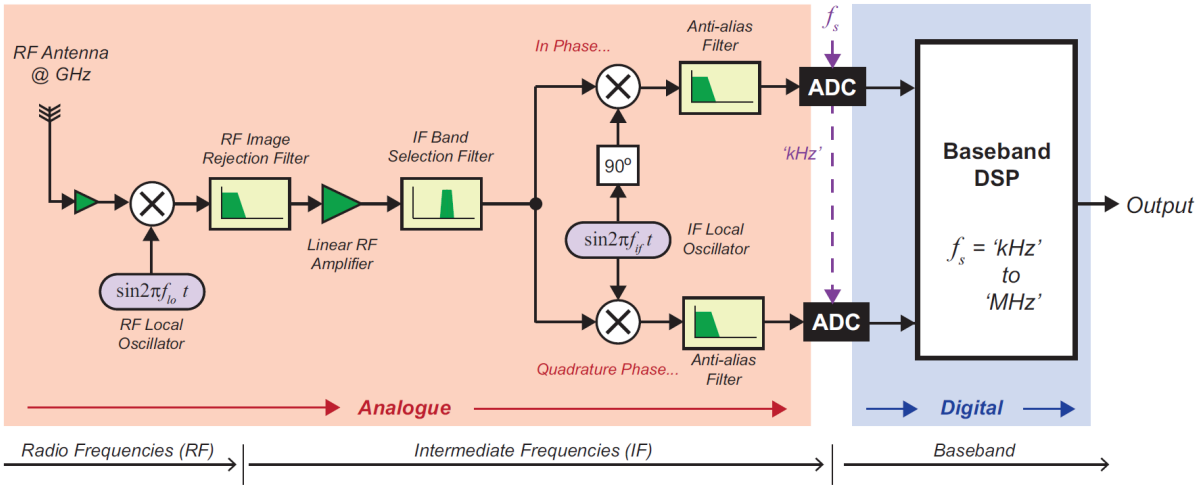
Figura 4 – Sistema de rádio com conversor em frequência de RF.



Fonte: Stewart et al. (2015)

to digital converter (ADC) somente com o sinal em banda base, o que possibilita a utilização de frequências de amostragem mais baixas, e como consequência, a carga de processamento em *software* diminui consideravelmente. O HackRF One, descrito na subseção 2.2.1 é um exemplo de módulo SDR que realiza a conversão para o domínio digital em banda base, embora ele tenha em seu circuito uma frequência intermediária entre 2,3 GHz e 2,7 GHz, ainda assim, antes da amostragem o sinal é deslocado para banda base.

Figura 5 – Sistema de rádio com conversor em banda base.



Fonte: Stewart et al. (2015)

2.2.1 HackRF One

O HackRF One (OSSMANN, 2017) é uma plataforma de *hardware* de projeto aberto amplamente utilizada como periférico nos sistemas de rádio definido por software. Segundo Ossmann, o criador da plataforma, seu principal objetivo com o HackRF One, módulo da Figura 6, é possibilitar através dela o teste, desenvolvimento e aplicação em novas tecnologias de comunicações via rádio. Objetivo o qual foi buscado através de dois aspectos, o primeiro deles consiste em um preço acessível quando comparado com as outras opções semelhantes, e o segundo a sua flexibilidade proveniente das suas características e especificações.

Figura 6 – HackRF One.

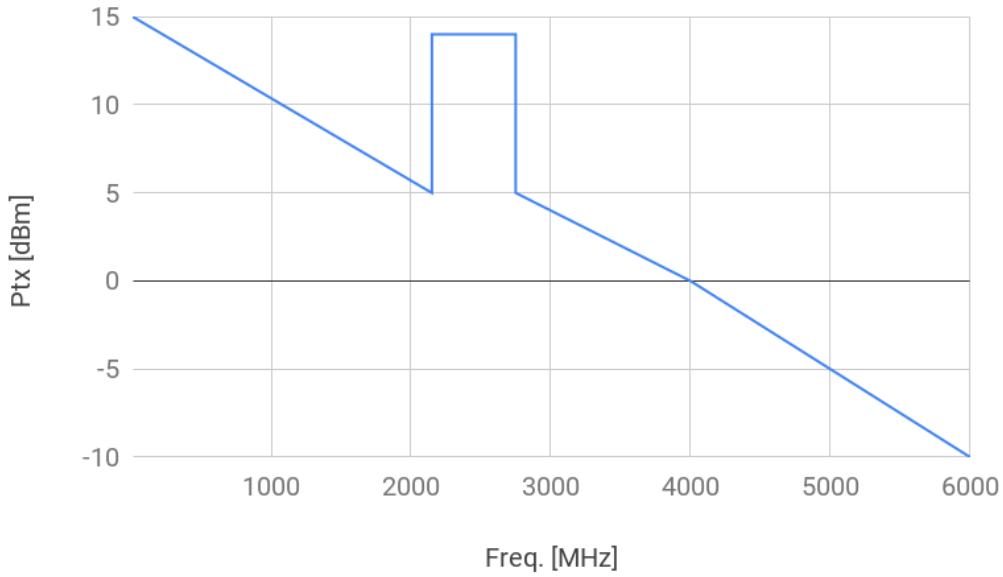
Fonte: <https://github.com/fd0/hackrf-one-pictures>

A seguir serão apresentadas as principais características do HackRF One:

- *Half-duplex transceiver*, o que possibilita a transmissão e recepção em toda a faixa de operação.
- Frequência de operação: 1 MHz até 6 GHz.
- Taxa de amostragem de: 2 MSps até 20 MSps.
- Resolução das amostras: 8 bits.
- Interface de comunicação: USB com conector Micro-B.
- Alimentação: 5 VDC através da própria interface USB.
- Controle da potência de transmissão via *software*.
- Conector da antena SMA fêmea de 50 Ω.
- Saídas e entradas de *clock* para sincronismos com outros sistemas.
- Portátil pelo fato de possuir um *housing* plástico.

Além da flexibilidade com relação à frequência de operação e largura de banda, o HackRF One tem potência de transmissão significativa quando comparado com os demais módulos presentes no mercado, o que possibilita não somente a prototipação de sistemas de rádio frequência mas também a utilização em aplicações práticas sem a necessidade de estágios de amplificação externos. Pelo fato de operar em uma larga faixa de frequência, a potência de saída depende da frequência que será utilizada, a documentação do HackRF One informa os valores de potência de saída obtidos em medições práticas. Na Figura 7 são exibidos os valores aproximados de potência de transmissão em função da frequência de operação.

Figura 7 – Potência de tx [dBm] em função da frequência [MHz].



Adaptado de: <https://github.com/mossmann/hackrf/wiki/HackRF-One#transmit-power>

2.2.2 SoapySDR

SoapySDR (POTHOSWARE, 2018a) é uma API escrita em C/C++ de código aberto para integração de alto nível com *hardwares* para SDR. Através da API do SoapySDR é possível iniciar, configurar, receber e transmitir usando um *hardware* SDR. Em sua concepção, o SoapySDR foi desenvolvido com suporte aos *hardwares* OsmoSDR[2] e UHD[3], entretanto sua arquitetura modular permite que outros fabricantes desenvolvam seus próprios módulos e tornem seus dispositivos compatíveis com a API SoapySDR.

Além das APIs em C/C++ (POTHOSWARE, 2018b) o SoapySDR também possui um módulo para Python3, o que nada mais é do que o mapeamento (*binding*) das chamadas de C++ para Python3 possibilitando a sua utilização também nesta linguagem, ampliando a abrangência da aplicação desta API em diversos sistemas, como no Código 2.3.

[2] http://sdr.osmocom.org/trac/
[3] https://kb.ettus.com/UHD/

Código 2.3 – *Hello world* SoapySDR com o HackRF One.

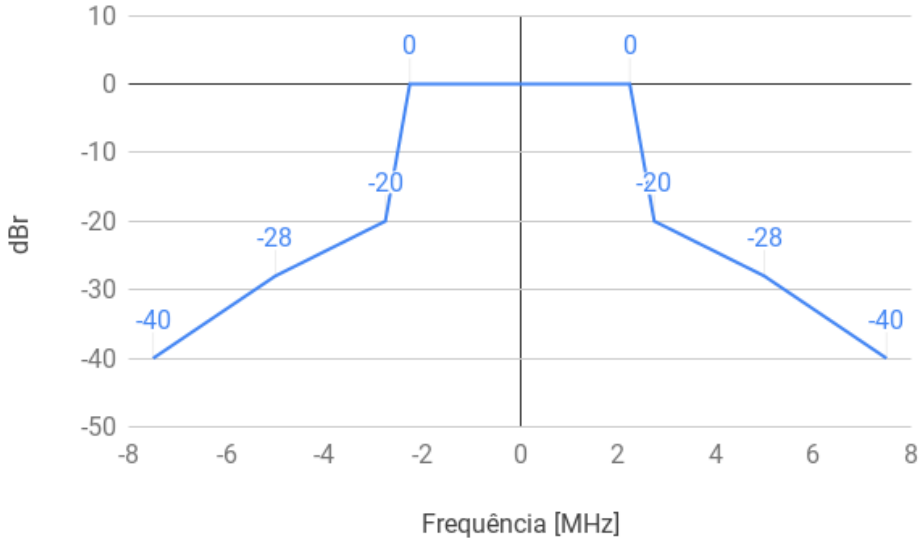
```
14 import SoapySDR
15 from SoapySDR import * #Importa as constantes SOAPY_SDR_*
16 import numpy as np
17
18 if __name__ == "__main__":
19   #Instanciando o HackRF One
20   hackrf = SoapySDR.Device(dict(driver="hackrf"))
21
22   #Definição dos parâmetros iniciais
23   FS = 8e6 #Frequêcia de amostragem [Hz]
24   BW = 200e3 #Largura de banda [Hz]
25   time = 5 #Tempo de captura em segundos
26   Nsamp = int(FS*time) #Número total de amostras
27   Fc = 100.9e6 #Frequência central [Hz]
28
29   #Configurando o HackRF
30   hackrf.setSampleRate(SOAPY_SDR_RX, 0, FS)
31   hackrf.setBandwidth(SOAPY_SDR_RX, 1, BW)
32   hackrf.setFrequency(SOAPY_SDR_RX, 1, Fc)
33
34   # Inicia o stream de captura ainda desativado
35   rxStream = hackrf.setupStream(SOAPY_SDR_RX, SOAPY_SDR_CF32, [0])
36
37   #Ativa o stream de captura
38   hackrf.activateStream(rxStream)
39
40   #Buffer em que serão armazenadas as amostras
41   buff = np.zeros(Nsamp, np.complex64)
42
43   #Loop de captura das amostras
44   while Nsamp > 0:
45     sr = hackrf.readStream(rxStream, [buff], buff.size, timeoutUs=int(1e6))
46     assert sr.ret > 0 # Para o caso de erros na obtenção das amostras
47     Nsamp -= sr.ret
48   #Ao término do loop tem-se em buff as amostras.
49
50   #Desativa o stream de captura
51   hackrf.deactivateStream(rxStream)
52   #Encerra o stream de captura
53   hackrf.closeStream(rxStream)
```

2.3 Camada física – IEEE 802.11n e IEEE 802.11ac

Amplamente utilizados em *access points* Wi-Fi, os padrões IEEE 802.11n (2009) e 802.11ac (2013) se consolidaram nas aplicações de *Wireless Local Area Network* (WLAN), parte dessa aceitação se deu pelo fato de operarem em frequências *Industrial Scientific and Medical* (ISM), que no Brasil não necessitam de licença da Agência Nacional de Telecomunicações (Anatel) para utilização.

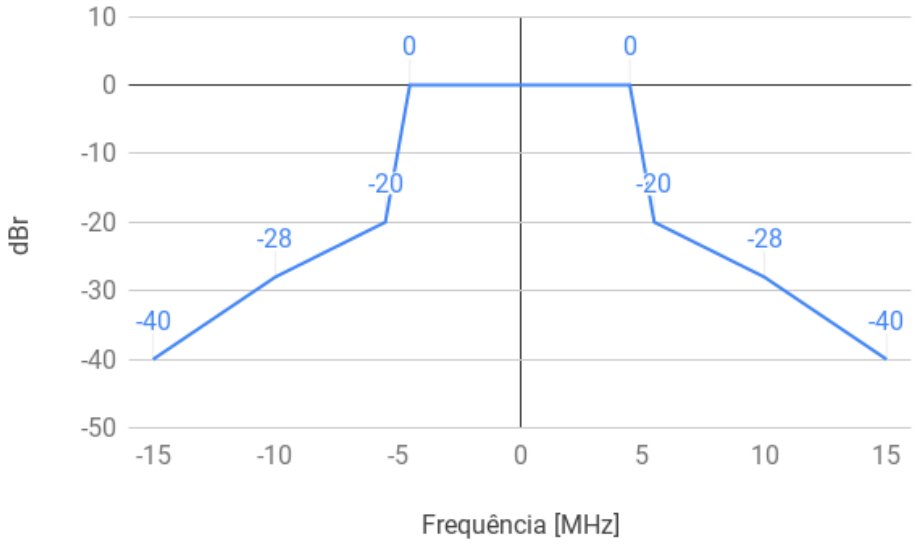
Optou-se no padrão IEEE 802.11n operar nas frequências de 2,4 GHz (2,4 GHz até 2,5 GHz) e 5 GHz (5,15 GHz até 5,85 GHz) com largura de banda de 5 MHz, 10 MHz, 20 MHz ou 40 MHz ou já no IEEE 802.11ac o uso fica restrito a frequência de 5 GHz (5,15 GHz até 5,85 GHz) com larguras de banda de 20 MHz, 40 MHz ou 80 MHz. Neste TCC os testes realizados estão limitados pela largura de banda do HackRF One, por isso, os equipamentos serão submetidos às rotinas de testes somente nas larguras de banda de 5 MHz, 10 MHz e 20 MHz. Os critérios para aprovação serão baseados na ocupação espectral de acordo com as máscaras das figuras 8, 9 e 10 especificadas por (STEPHENS, 2016).

Figura 8 – Máscara de ocupação espectral para operação em 5 MHz.



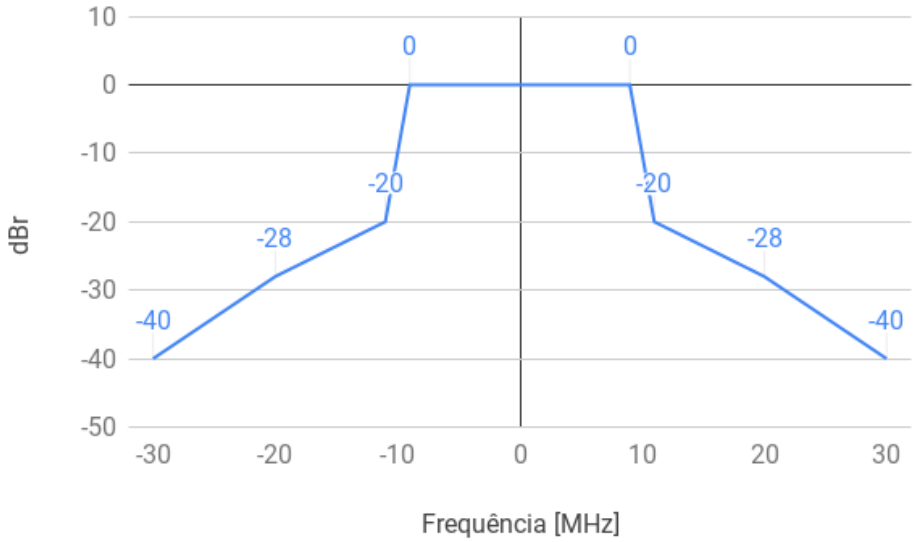
Adaptado de: Stephens (2016).

Figura 9 – Máscara de ocupação espectral para operação em 10 MHz.



Adaptado de: Stephens (2016).

Figura 10 – Máscara de ocupação espectral para operação em 20 MHz.



Adaptado de: Stephens (2016).

Além de escolher a largura de banda, é possível também escolher o canal de operação do dispositivo com base nas Tabelas 1 e 2.

Tabela 1 – Canais disponíveis em 2,4 GHz.

Número do canal	1	2	3	4	5	6	7	8	9	10	11	12	13
Fc [MHz]	2412	2417	2422	2427	2432	2437	2442	2447	2452	2457	2462	2467	2472

Fonte: adaptado de Gast (2013)

Tabela 2 – Canais disponíveis em 5 GHz.

Número do canal	36	40	44	48	52	56	60	64	100	104	108	112
Fc [MHz]	5180	5200	5220	5240	5260	5280	5300	5320	5500	5520	5540	5560

Número do canal	116	120	124	128	132	136	140	144	149	153	157	161
Fc [MHz]	5580	5600	5620	5640	5660	5680	5700	5720	5745	5765	5785	5805

Fonte: adaptado de Gast (2013)

3 PROPOSTA

Dados os objetivos explanados no Capítulo 1, este TCC propõe um sistema capaz de automatizar testes de mascaramento espectral em *access points* Intelbras ZEUS utilizando o HackRF One. É fato que o HackRF One possui limitações relacionadas com as suas especificações, como largura de banda, e precisão proveniente da quantidade de bits dos ADCs. Por este motiv o, para aplicações práticas recomenda-se a utilização de módulos SDR mais robustos, com maior largura de banda e mais precisão em seus ADCs, entretanto como o objetivo deste TCC é validar a utilização de SDR para a automatização de testes, pôde-se utilizar o HackRF One limitando a abrangência dos testes realizados em virtude da limitação de largura de banda.

3.1 Funcionamento do sistema

O fluxograma da Figura 11 descreve as etapas de operação do sistema, desde início do ciclo de testes até a elaboração do relatório ao final do ciclo. No início do ciclo de testes, é realizada a autenticação da API para que seja obtido o *token* que será utilizado nas demais chamadas de API para as alterações de configuração. Com o *token* já disponível os testes são realizados em sequência até que todos tenham sido feitos. Por fim, elabora-se um relatório e finaliza-se o ciclo de testes.

Já o fluxograma da Figura 12 mostra como o teste de fato é realizado. Para aferir o mascaramento espectral injeta-se tráfego forçando a transmissão do *access point*, capturam-se as amostras para que se possa calcular a *fast Fourier transform* (FFT) a fim de analisar o sinal no domínio da frequência, armazena-se os valores máximos durante alguns segundos e os compara com os requisitos de aprovação baseados nas máscaras citadas na seção 2.3.

Figura 11 – Fluxograma do sistema.

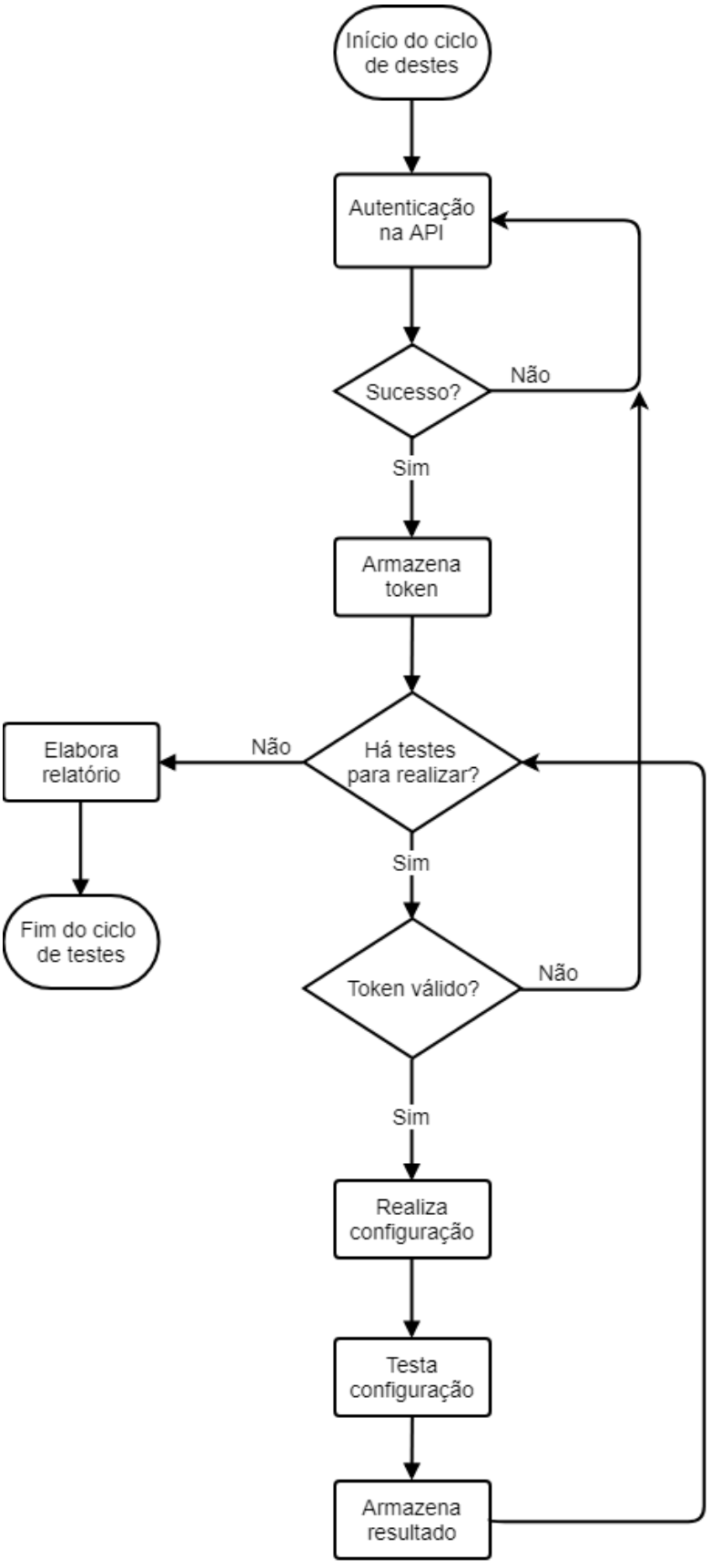
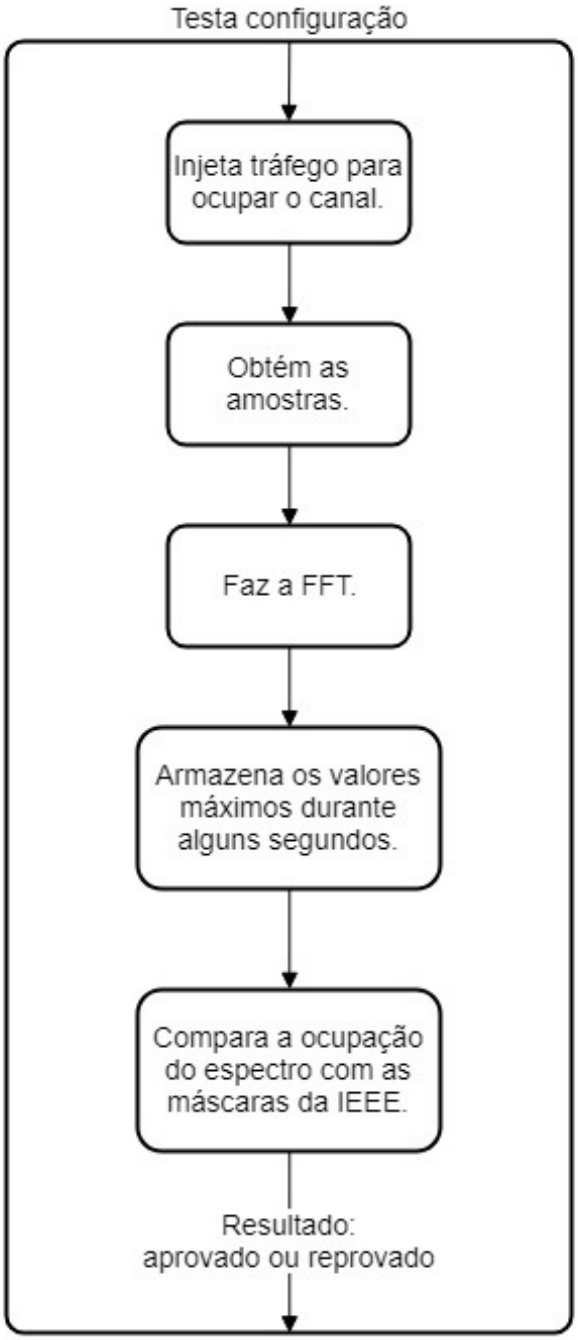


Figura 12 – Fluxograma da aferição espectral.



3.2 Cronograma

A seguir na Tabela 3, serão elencadas as etapas de desenvolvimento e a data prevista para a execução de cada uma delas.

Tabela 3 – Cronograma das atividades previstas

Tarefa	Mês							
	Dez	Jan	Fev	Mar	Abr	Mai	Jun	Jul
A1	√							
A2		√	√	√				
A3				√	√			
A4					√	√		
A5						√	√	
A6							√	√

- **A1:** Ajustes na documentação final do pré projeto;
- **A2:** Desenvolvimento do módulo de obtenção das amostras e processamento de sinais;
- **A2:** Modelagem e implantação do banco de dados;
- **A3:** Desenvolvimento do módulo de interação com os *access points*;
- **A4:** Integração dos módulos desenvolvidos;
- **A5:** Testes com o sistema em ambiente controlado;
- **A6:** Documentação dos resultados obtidos e escrita do TCC.

REFERÊNCIAS

BRAY, T. *The JavaScript Object Notation (JSON) Data Interchange Format*. RFC Editor, 2017. RFC 8259. (Request for Comments, 8259). Disponível em: <https://rfc-editor.org/rfc/rfc8259.txt>. Citado na página 18.

CISCO SYSTEMS, INC. *VNI Forecast Highlights Tool*. 2016. Disponível em: <https://www.cisco.com/c/m/en_us/solutions/service-provider/vni-forecast-highlights.html>. Acesso em: 17.10.2018. Citado na página 15.

FIELDING, R. T. et al. *Hypertext Transfer Protocol – HTTP/1.1*. [S.l.], 1999. <http://www.rfc-editor.org/rfc/rfc2616.txt>. Disponível em: <http://www.rfc-editor.org/rfc/rfc2616.txt>. Citado na página 18.

FIELDING, R. T.; RESCHKE, J. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. RFC Editor, 2014. RFC 7231. (Request for Comments, 7231). Disponível em: <https://rfc-editor.org/rfc/rfc7231.txt>. Citado na página 18.

FRANTZ, L.; WEIDLE, G. *Plataforma Zeus - Requisitos e escopo inicial*. 2015. Confidencial. Disponível em: <http://redmine.intelbras.com.br/attachments/download/3883/2015.08.25%20Escopo.pdf>. Acesso em: 23.10.2018. Citado na página 17.

GAST, M. S. *802.11ac: A Survival Guide: Wi-Fi at Gigabit and Beyond*. O'Reilly Media, 2013. ISBN 1449357717. Disponível em: <https://www.amazon.com/802-11ac-Survival-Guide-Gigabit-Beyond-ebook/dp/B00FM0OC66?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=B00FM0OC66>. Citado na página 25.

INTELBRAS S/A. *ZEUS V3*. 2018. Disponível em: <http://izeus.citeb.com.br/apiary/v3/en>. Acesso em: 24.10.2018. Citado na página 18.

OSSMANN, M. *HackRF One*. 2017. Disponível em: <https://github.com/mossmann/hackrf/wiki/HackRF-One>. Acesso em: 07.11.2018. Citado na página 21.

POTHOSWARE. *SoapySDR*. 2018. Disponível em: <https://github.com/pothosware/SoapySDR/wiki>. Acesso em: 18.11.2018. Citado na página 22.

POTHOSWARE. *SoapySDR - Vendor and platform neutral SDR interface library*. 2018. Disponível em: <https://pothosware.github.io/SoapySDR/doxygen/latest/classSoapySDR_1_1Device.html>. Acesso em: 18.11.2018. Citado na página 22.

STEPHENS, A. P. *IEEE Std 802.11ai-2016 (Amendment to IEEE Std 802.11-2016) : IEEE Standard for Information technology–Telecommunications and information exchange between systems - Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Acc*. S.l: IEEE, 2016. ISBN 978-1-5044-3631-1. Citado 3 vezes nas páginas 15, 24 e 25.

STEWART, R. W. et al. *Software Defined Radio using MATLAB & Simulink and the RTL-SDR*. Strathclyde Academic Media, 2015. ISBN 0992978726. Disponível em: <https://www.amazon.com/Software-Defined-MATLAB-Simulink-RTL-SDR/dp/0992978726?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0992978726>. Citado 2 vezes nas páginas 19 e 20.