

**UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS**

Carlos Vinicius Fernandes

**Análise de espectro de rádio frequência utilizando
*Software Define Radio***

São Carlos

2019

Carlos Vinicius Fernandes

**Análise de espectro de rádio frequência utilizando
*Software Define Radio***

Monografia apresentada ao Curso de Engenharia Elétrica com Ênfase em Eletrônica, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Dr. Carlos Dias Maciel

**São Carlos
2019**

AUTORIZO A REPRODUÇÃO TOTAL OU PARCIAL DESTE TRABALHO,
POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS
DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da
EESC/USP com os dados inseridos pelo(a) autor(a).

Fa Fernandes, Carlos Vinicius
 Análise de espectro de rádio frequência
 utilizando Software Define Radio / Carlos Vinicius
 Fernandes; orientador Prof. Dr. Carlos Dias Maciel. São
 Carlos, 2019.

 Monografia (Graduação em Engenharia Elétrica com
 ênfase em Eletrônica) -- Escola de Engenharia de São
 Carlos da Universidade de São Paulo, 2019.

 1. Rádio. 2. Espectro de frequências. 3.
 Monitoramento. 4. Antenas. 5. Software Define Radio. 6.
 Indicadores. 7. Ocupação de canais. 8. Ruído. I.
 Título.

FOLHA DE APROVAÇÃO

Nome: Carlos Vinicius Fernandes

Título: "Análise de espectro de radio frequência utilizando Software Define Radio"

Trabalho de Conclusão de Curso defendido e aprovado
em 9 / 10 / 2019,

com NOTA 9,5 (note, meio), pela Comissão Julgadora:

Prof. Associado Carlos Dias Maciel - Orientador - SEL/EESC/USP

Mestre Talysson Manoel de Oliveira Santos - Doutorando - SEL/EESC/USP

Mestre Jordão Natal de Oliveira Júnior - Doutorando - SEL/EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Rogério Andrade Flauzino

RESUMO

FERNANDES, C. **Análise de espectro de rádio frequência utilizando *Software Define Radio***. 2019. 98p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2019.

Este trabalho aborda o desenvolvimento de um sistema de monitoramento, em banda larga, do espectro de rádio frequência, utilizando um dispositivo de rádio definido por *software* de baixo custo, da família RTL-SDR. São revisados para o leitor conceitos-chaves da teoria de ondas e apresentado o funcionamento do dispositivo de rádio, a fim de esclarecer a interação do dispositivo com o meio e a origem de suas medidas, assim como o processamento pelo qual foram submetidas. Em seguida, são expostos o ferramental de *hardware* e *software* necessários ao trabalho e o desenvolvimento do mesmo, analisando-se a ocupação dos canais de rádio frequência através do cálculo de indicadores propostos ao longo do desenvolvimento do *software* escrito pelo aluno, baseados na distribuição dos dados de potência de cada canal. Por fim, tem-se as considerações finais quanto ao que foi obtido e possíveis horizontes de continuação.

Palavras-chave: Rádio, Espectro de frequências, Monitoramento, Antenas, Software Define Radio, Indicadores, Ocupação de canais, Ruído.

ABSTRACT

FERNANDES, C. . 2019. 98p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2019.

This bachelor thesis addresses the development of a wideband frequency spectrum monitoring system, using a low cost software define radio device of the RTL-SDR family. It is reviewed to the reader key concepts of electromagnetic wave theory together with a description of how the radio device operates, clarifying the interaction of the device and the environment from which it's measurements originates, as well as the processing that the data has been submitted. Following, it is listed the hardware and software setup required for the project and its development, analysing radio channels usage through calculating indicators that are proposed through out the software development done by the author, based on the data distribution of the signal power of each channel. Lastly, final remarks are done about the obtained and future projects possibilities.

Keywords: Radio, Frequency Spectrum, Monitoring, Antennas, Software Define Radio, Indicators, Spectrum Usage, Noise.

LISTA DE FIGURAS

Figura 1 – Representação do fenômeno de difração (Difração 2019)	18
Figura 2 – Modelo simulado de antena dipolo conforme a referência (RAYmaps 2012)	19
Figura 3 – Modelo simulado de antena <i>patch</i> conforme a referência (RAYmaps 2012)	20
Figura 4 – Exemplo de perda de retorno de um dipolo de meia onda - adaptado de (CST 2019)	21
Figura 5 – Relação dos comprimentos dos elementos base de uma antena log periódica (BALANIS 2012)	22
Figura 6 – Exemplo de VSWR de uma antena log periódica - adaptado de (Antenna Theory)	23
Figura 7 – Diagrama de blocos do <i>front-end</i> RF do RTL-SDR	25
Figura 8 – Exemplo de implementação em blocos utilizando o GNU Radio Companion (GNU Radio 2019)	27
Figura 9 – Saída do demodulador FM construído com o GNC	27
Figura 10 – Estrutura do arquivo csv produzido pelo rtl_power	28
Figura 11 – Dispositivo RTL-SDR utilizado	31
Figura 12 – SDR conectado ao laptop reportando status da varredura no terminal	32
Figura 13 – <i>Setup</i> utilizado para as medições; antena à direita	33
Figura 14 – Varredura com Efeito de atenuação nas beiradas da janela amostrada	36
Figura 15 – Varredura com a utilização do recurso <i>crop</i> do rtl_power	36
Figura 16 – Varredura com <i>binsize</i> de 1.367 kHz	37
Figura 17 – Varredura com <i>binsize</i> de 1 MHz	37
Figura 18 – Visualização da variável <i>gap</i>	42
Figura 19 – Fluxograma do <i>software</i> de tratamento dos dados	43
Figura 20 – Potência rms média - 12h00 às 13h00 de 08/Mar/2019	48
Figura 21 – Potência rms média - Todas as leituras	48
Figura 22 – Potência rms média em dB - 12h00 às 13h00 de 08/Mar/2019	49
Figura 23 – Potência rms média em dB- Todas as leituras	49
Figura 24 – Desvio padrão da média rms - 12h00 às 13h00 de 08/Mar/2019	50
Figura 25 – Desvio padrão da média rms - Todas as leituras	50
Figura 26 – Razão entre desvio e média - 12h00 às 13h00 de 08/Mar/2019	51
Figura 27 – Razão entre desvio e média - Todas as leituras	51
Figura 28 – Assimetria - 12h00 às 13h00 de 08/Mar/2019	52
Figura 29 – Assimetria - Todas as leituras	52
Figura 30 – Curtose - 12h00 às 13h00 de 08/Mar/2019	53
Figura 31 – Curtose - Todas as leituras	53
Figura 32 – Poder do teste de normalidade - 12h00 às 13h00 de 08/Mar/2019	54

Figura 33 – Poder do teste de normalidade - Todas as leituras	54
Figura 34 – Probabilidade H0 do teste - 12h00 às 13h00 de 08/Mar/2019	55
Figura 35 – Probabilidade H0 do teste - Todas as leituras	55
Figura 36 – Confronto entre média e desvio padrão - 12h00 às 13h00 de 08/Mar/2019 .	56
Figura 37 – Confronto entre média e desvio padrão - Todas as leituras	56
Figura 38 – Confronto entre média e a razão desvio/média - 12h00 às 13h00 de 08/Mar/2019	57
Figura 39 – Confronto entre média e a razão desvio/média - Todas as leituras	57
Figura 40 – Confronto entre prob. H0 do teste e média - 12h00 às 13h00 de 08/Mar/2019	58
Figura 41 – Confronto entre prob. H0 do teste e média- Todas as leituras	58
Figura 42 – Confronto prob. H0 do teste e a razão desvio/média - 12h00 às 13h00 de 08/Mar/2019	59
Figura 43 – Confronto prob. H0 do teste e a razão desvio/média - Todas as leituras . . .	59

LISTA DE ABREVIATURAS E SIGLAS

EESC	Escola de Engenharia de São Carlos
USP	Universidade de São Paulo
GNU	<i>GNU's Not Unix</i>
UHF	<i>Ultra High Frequency</i>
GSM	<i>Global System for Mobile Communications</i>
ISM	<i>Industrial, Scientific and Medical</i>
ADS-B	<i>Automatic Dependent Surveillance-Broadcast</i>
FFT	<i>Fast Fourier Transform</i> (Transformada Rápida de Fourier)
TDD	<i>Test Driven Development</i> (Desenvolvimento Orientado a Testes)

LISTA DE ABREVIATURAS E SIGLAS

Ω	Ohm
λ	Lambda

SUMÁRIO

1	INTRODUÇÃO	15
1.1	Objetivos	16
1.2	Organização textual	16
2	TEORIA	17
2.1	Rádio Frequência	17
2.2	Antenas	18
2.3	<i>Software Define Radio</i>	23
2.3.1	<i>Hardware Interno</i>	24
2.3.2	Ferramentas de <i>Software Livre</i>	26
2.3.2.1	GNU Radio	26
2.3.2.2	rtl_power.c	28
3	MATERIAIS E MÉTODOS	31
3.1	Materiais utilizados	31
3.2	Metodologia aplicada	34
3.2.1	Faixa de medição escolhida	34
3.2.2	Procedimento de varredura	35
3.2.3	Utilização do Python	38
3.2.4	Desenvolvimento dos indicadores	38
4	RESULTADOS E DISCUSSÕES	41
4.1	<i>Software Desenvolvido</i>	41
4.2	Indicadores	44
5	CONCLUSÕES	61
5.1	Proposta de continuação	61
	REFERÊNCIAS	63
6	ANEXOS	65
6.1	Plano de Atribuição de faixas e frequências no Brasil	65
6.2	Indicadores de todos os dados coletados	68
6.3	Códigos em Python	76

1 INTRODUÇÃO

Hoje vivemos num mundo imerso em ondas eletromagnéticas de diversas naturezas, desde ondas de rádio, até a radiação ionizante do ultra-violeta, uma gama de espectro além do visível aos seres humanos. Justamente por não as ver e não as sentir diretamente, no nosso dia a dia, não contemplamos e até mesmo esquecemos o quão difundido e presente essas ondas são.

Telecomunicações por meio das ondas de rádio frequência, nas áreas de atividade humana é tão sedimentadas quanto a própria energia elétrica que as impulsiona, tendo grande parte de seu desenvolvimento impulsionado durante os períodos de guerra do século XX. Hoje, o uso das ondas de rádio é majoritariamente comercial pelos meios de rádio difusão e pelas redes de telefonia celular, ocupando grande parte do espectro, entretanto isto não significa que não seja mais um ponto importante para as entidades de defesa de um país, pelo contrário, o monitoramento das atividades é crucial à hegemonia e segurança do país.

Através do monitoramento também busca-se garantir o uso devido do recurso, cada vez mais escasso, o espectro de frequências, disponível para uso das atividades, ou seja, os canais de rádio frequência disponíveis. Para tal, é necessário cobrir uma grande faixa de espectro, o que exigiria *hardwares* de rádio dedicados, calibrados para diferentes subfaixas de espectro de banda larga, ou rádios reconfiguráveis que possam cobrir com precisão a atividade das diferentes subfaixas.

Com aumento da capacidade de processamento e surgimento de *hardware* reprogramável via *software* nas últimas décadas, tornou-se possível o desenvolvimento de um rádio em que as funcionalidades de componentes de *hardware* como *mixers*, filtros, amplificadores, moduladores, entre outros, são implementados através de *software*. Tal dispositivo é conhecido por Radio Definido por Software, mais conhecido pelo seu nome em inglês *Software Define Radio* (SDR), sendo um dispositivo muito utilizado para varredura e pesquisa, ideal para a tarefa de monitoramento dos canais de rádio frequência.

Tendo em vista os canais de rádio frequência, a atividade dos mesmos varia com a região e o horário do dia, pois diferentes localidades possuem diferentes serviços e clientes, restritos à área de cobertura do serviço, sendo a alocação destes canais reguladas por um órgão governamental. Os canais então podem ser ocupados por serviços perenes, como no caso de uma estação de rádio, ou de televisão, transmitindo 24 horas por dia, ou por serviços intermitentes em que o canal é utilizado por um período e depois liberado e realocado, como no caso dos serviços de telefonia móvel. Quando um canal não está sendo utilizado, encontra-se ocioso, ou seja, é recurso um recurso livre, mas o uso deste deve ser previsto para atividades determinadas pelo órgão governamental por lei.

Canais não ociosos então podem, ou não, representar atividade conhecida, que deve ser prevista no plano de distribuição de frequências da ANATEL (plano de distribuição completo encontra-se de forma gráfico no anexo 6.1) (ANATEL 2018). O uso indevido dos canais por si só já evidencia atividade ilegal perante o órgão regulador. As demais atividades previstas, porém de origem suspeita, estarão passivas de investigação criminal sobre suas origens e licitude.

Um sistema de monitoramento do espectro necessita então ser capaz de: cobrir um espectro de banda larga e, baseado nos dados processados do sinal recolhido, ser capaz de rotular características encontradas nos diferentes canais em que haja indícios que permitam inferir algo sobre a atividade do canal, ou seja, o sistema deve levantar um indicador, ou um conjunto destes, de ocupação dos canais.

1.1 Objetivos

Neste trabalho, busca-se através da utilização do SDR, realizar um estudo de banda larga de diferentes canais de rádio, em horários pré-determinados e em dias diferentes, visando encontrar indicadores que permitam inferir a atividade dos canais à partir de puramente leituras de potência.

Os indicadores deveram apontar indícios da atividade do canal ser perene, ou intermitente, principalmente baseando-se na variação do uso do canais durante os horários medidos. Também busca-se através dos mesmos indicadores rotular os canais ociosos, cujas leituras são caracterizadas por resíduo radio frequência e outras fontes de ruído.

1.2 Organização textual

O trabalho está organizado de forma a apresentar o desenvolvimento do mesmo sequencialmente. No Capítulo 2 é revisada a teoria necessária ao leitor, tendo em vista os fenômenos os quais ondas de rádio estão submetidas enquanto program, a interface entre a onda carregando a informação e o dispositivo SDR por meio de uma antena, e o *hardware* interno do SDR. Já no Capítulo 3 são listados os *hardware* e *software* envolvidos de forma detalha, e é explicitada a metodologia aplicada aos listados, contextualizando as escolhas e procedimentos necessários ao trabalho. O Capítulo 4 descreve o *software* desenvolvido e os resultados à partir do mesmo, sendo realizada uma discussão dos indicadores produzidos pelo *software*. Por fim, o Capítulo 5 encontram-se as conclusões do trabalho e possíveis horizontes de continuação.

2 TEORIA

Neste capítulo apresenta-se os conceitos necessários ao trabalho para o leitor, concentrando-se na interação com o SDR, através de uma antena e como o *hardware* trata e processa o sinal, expondo sua estrutura interna.

2.1 Rádio Frequência

Rádio frequência, abreviada por RF, designa uma faixa do espectro eletromagnético comumente utilizada para a transferência de informação entre equipamentos eletrônicos via ar, diferentemente de ondas guiadas, encontradas em redes cabeadas de computadores e telefones fixos.

As ondas de rádio são emitidas e recebidas no espaço por meio de antenas, que ditam características da onda como direção de emissão e captação, podendo concentrar-se em uma única direção, ou expandir em várias direções. Toda onda eletromagnética é composta por duas componentes de campos perpendiculares entre si, elétrico e magnético, oscilando em fase. A frequência de oscilação da onda eletromagnética relaciona-se com velocidade de propagação através da equação ([Comprimento de Onda 2019](#)):

$$v = \lambda f \quad (2.1)$$

- Onde:

- ☐ v = velocidade de propagação da onda no meio
- ☐ λ = comprimento de onda
- ☐ f = frequência da onda

Uma vez que a onda está propagando no meio, torna-se susceptível a fenômenos como:

- **Atenuação:** Ao interagir com um material, a energia carregada pela onda é absorvida por este material, acarretando em uma diminuição da amplitude dos campos. Em um meio que não vácuo, conforme propaga, a onda é absorvida pelo material que compõe o meio, portanto quanto maior a distância percorrida no meio, mais energia da onda é absorvida ([Neto 2019](#)). No caso de ondas RF, o meio de propagação principal é o ar, entrando obstáculos como montanhas, vegetação, edifícios entre outros também absorvem a energia, limitando o alcance da mesma.

- **Difração:** Desvio sofrido pela onda ao passar pelas bordas de um obstáculo. Ao contornar um obstáculo, a frente de onda sofre um espalhamento, onde diversas novas frentes de onda são originadas com diferentes direções de propagação (Difração 2019).

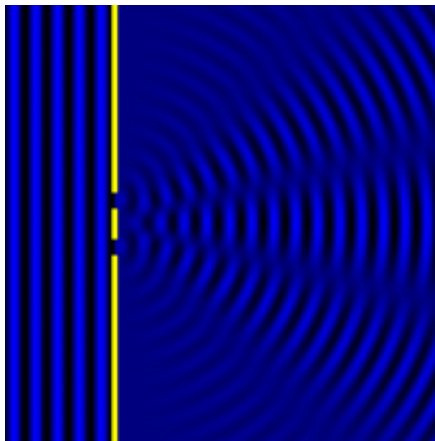


Figura 1 – Representação do fenômeno de difração (Difração 2019)

- **Reflexão e Refração:** Ao incidir sobre uma superfície, ou seja, a interface entre dois meios, a onda tem uma mudança de direção de propagação, podendo retornar ao meio de origem, ou adentrar o outro meio. A parcela da onda que retorna ao meio de origem é a frente de onda **refletida**, e a que adentra é a **refratada**. Quando toda a onda é refletida na interface, tem-se o fenômeno de reflexão total.
- **Interferência:** Superposição de ondas no mesmo ponto, sendo destrutiva quando as fases das ondas são distintas, ou construtiva quando as fases forem idênticas. Em particular, olhando para o espectro de frequência, se diferentes sinais estão sendo transmitidos dentro da mesma faixa de frequência, haverá uma competição de recurso de frequência, mas há formas de evitar ou contornar a interferência nesta situação, porém não será discutido isto neste trabalho.

Este fenômenos em conjunto, são responsáveis pela dispersão das ondas de RF no meio em que vivemos. Toda vez que uma onda de RF encontra um objeto, está sujeita a todos esses efeitos, e a antena do equipamento receptor deve ser dimensionada e posicionada de forma que o sinal carregando informação seja recebido com pelo menos um mínimo de potência para distinguir-se dos sinais interferentes.

2.2 Antenas

No contexto de RF, antenas são elementos capazes de operar como interface entre as ondas propagando no meio e a corrente elétrica que flui nos condutores metálicos, emitindo e recebendo sinais eletromagnéticos, sendo componente essencial de equipamentos transmissão e recepção de sinal (Antenna (radio) 2019). Na transmissão, a antena radia a energia, da corrente suprida

aos seus terminais, na forma de onda eletromagnética. Na recepção, a antena absorve a energia da onda incidente e produz um sinal de corrente elétrica na saída de seus terminais.

Antenas possuem diversos formatos, que determinam a direção em que a onda é transmitida, conhecido por **diagrama de radiação**. Antenas clássicas como dipolo e *patch*, irradiam conforme as figuras:

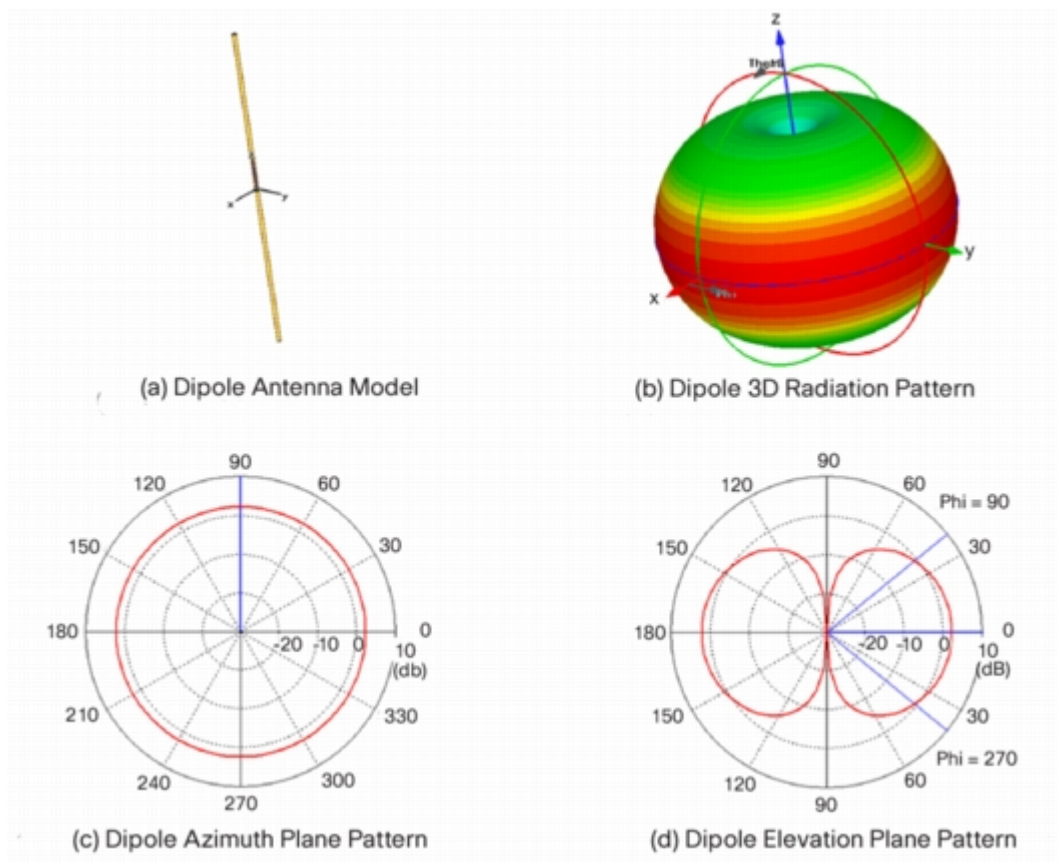


Figura 2 – Modelo simulado de antena dipolo conforme a referência ([RAYmaps 2012](#))

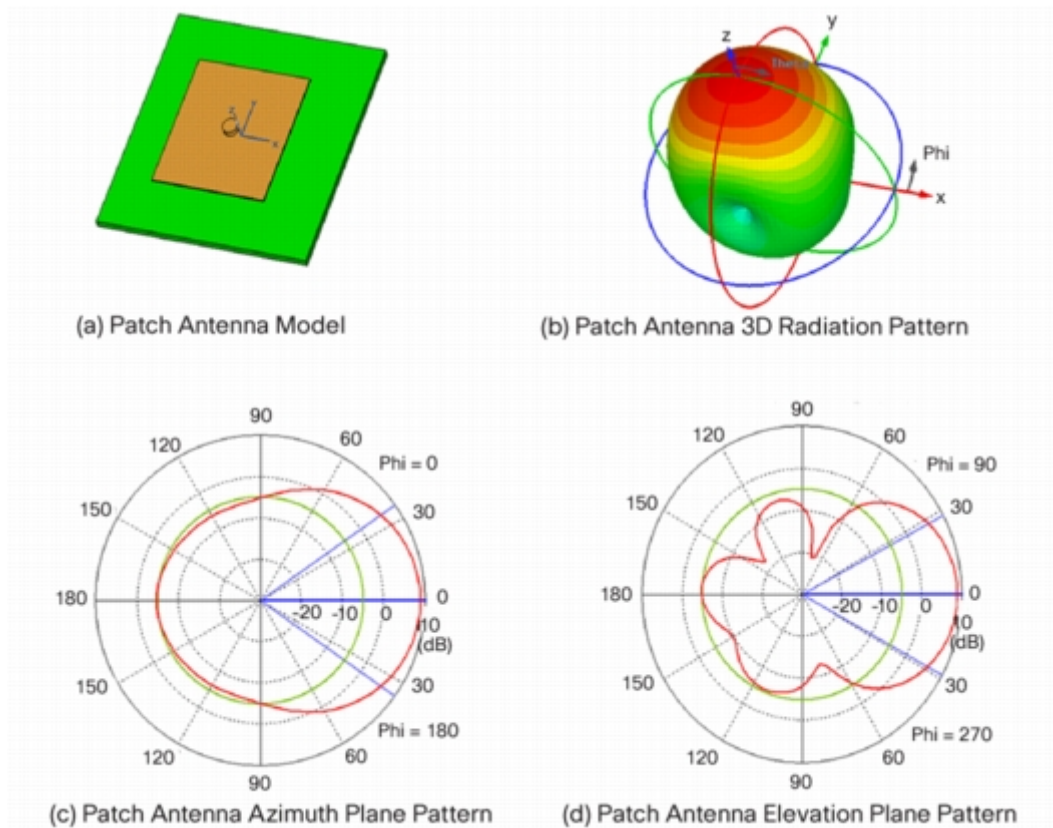


Figura 3 – Modelo simulado de antena *patch* conforme a referência (RAYmaps 2012)

Em particular, antenas do dipolo de meia onda são projetadas para operar com máximo desempenho em $\frac{\lambda}{2}$. Este tipo de antena é comumente usado para fins didáticos e é a base para antenas como Yagi e log periódica, sendo esta última utilizada pelo trabalho. Como mencionado, o máximo desempenho da antena dipolo é em $\frac{\lambda}{2}$, isto é, a antena apresenta máxima transferência de energia em uma frequência $\frac{2c}{\lambda}$. Isto é feito construindo o dipolo de formato cilíndrico com comprimento $\frac{\lambda}{2}$ (Stutzman e Thiele 1997).

Entretanto, um sinal real não contém somente uma componente de frequência, e a figura de mérito que expõe o comportamento da antena no espectro de frequência é designada perda de retorno, ou coeficiente de reflexão (Γ), também conhecido pelo parâmetro S_{11} . A figura 4 aponta o comportamento de um dipolo de meia onda para uma faixa de frequência, expressando que para as demais frequências, reflexão é maior, ou seja, a transferência de energia é menor. Isto ocorre devido ao descasamento de impedâncias entre os meios de transmissão da onda, incluindo o próprio dipolo, pois estes possuem impedância variando com a frequência.

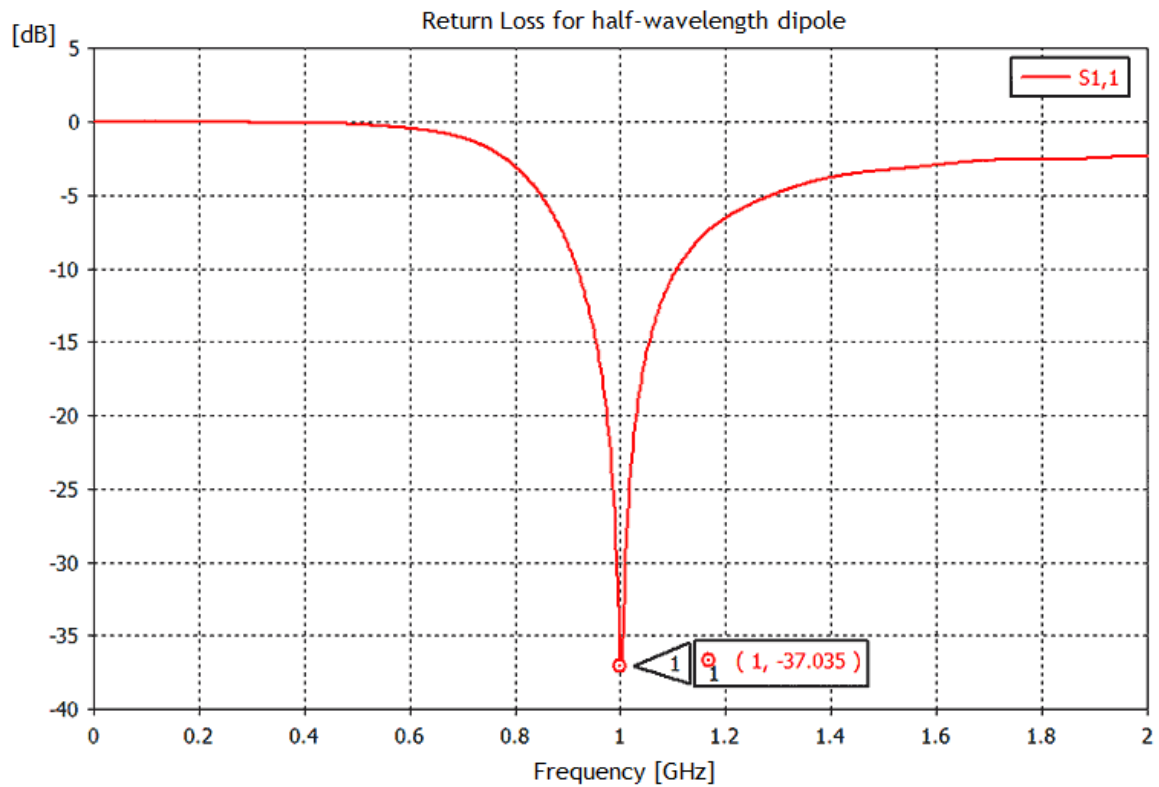


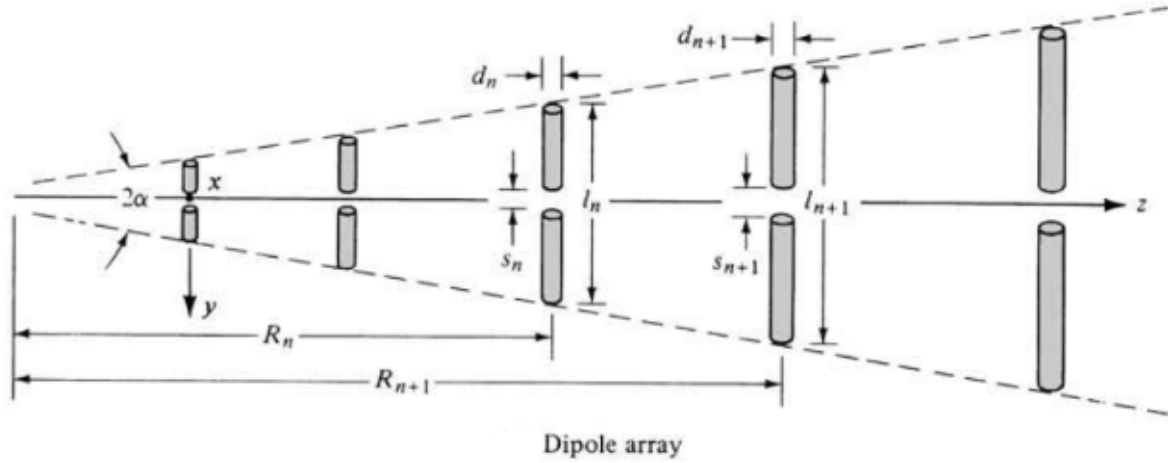
Figura 4 – Exemplo de perda de retorno de um dipolo de meia onda - adaptado de (CST 2019)

Uma forma de medir o casamento de impedância é através da **Relação de Onda Estacionária** (ROE), muitas vezes encontradas na literatura pelo termo em inglês, com relação a tensão, **Voltage Standing Wave Ratio** (VSWR) (Antenna Theory). Esta figura é expressa em função do coeficiente de reflexão tal que:

$$VSWR = \frac{1 + |\Gamma|}{1 - |\Gamma|} \quad (2.2)$$

Quanto menor o VSWR de uma antenna, melhor o casamento entre os meios, sendo VSWR igual a 1 situação ideal de máxima transferência de energia.

Como mencionado, o dipolo é a base para uma antena log periódica. Tal antena dita de *barda larga*, construída a partir de um arranjo, ou *array*, de dipolos, cujos os comprimentos são dimensionados conforme mostrado abaixo:



$$\frac{1}{\tau} = \frac{l_2}{l_1} = \frac{l_{n+1}}{l_n} = \frac{R_2}{R_1} = \frac{R_{n+1}}{R_n} = \frac{d_2}{d_1} = \frac{d_{n+1}}{d_n} = \frac{s_2}{s_1} = \frac{s_{n+1}}{s_n}$$

Figura 5 – Relação dos comprimentos dos elementos base de uma antena log periódica (BALANIS 2012)

De forma simplificada, o comportamento deste tipo de antena é dado pela soma da atuação de cada dipolo presente, sendo assim possível cobrir uma certa faixa do espectro com boa transferência de energia, isto é, com um bom ganho da potência do sinal. A figura 6 expressa o vswr de uma antena log periódica típica, projetada para transmissões UHF.

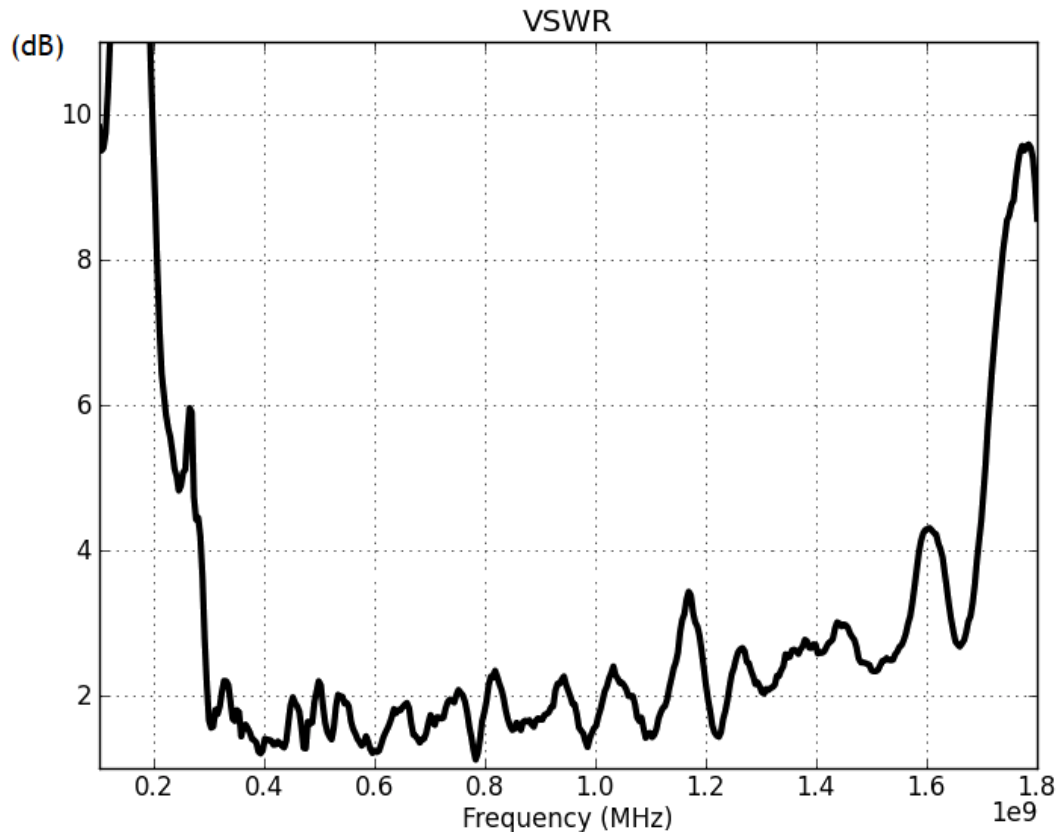


Figura 6 – Exemplo de VSWR de uma antena log periódica - adaptado de ([Antenna Theory](#))

2.3 Software Define Radio

O *Software Define Radio* (SDR), ou rádio definido por *software*, é um dispositivo capaz de processar a banda base de um sinal RF, implementando funcionalidades de componentes de *hardware* como *mixers*, filtros, amplificadores, moduladores, entre outros através de *software*, tendo a capacidade de trabalhar com diversos protocolos de rádio, sendo um dispositivo reprogramável.

Existem hoje no mercado tanto SDR de *hardware* proprietário, como o *Universal Software Radio Peripheral* (USRP), produzido pela Ettus Research (pertencente a National Instruments), quanto SDRs de *hardware* livre como o HackRF One. Entretanto, uma característica comum a grande parcela dos SDR no mercado, incluindo os mencionados, é o uso de *software* livre, característica forte da comunidade envolvida com o uso e estudo do dispositivo. A organização *Open Source Mobile Communications* (Osmocomm) possui projetos voltados ao desenvolvimento de *software* livre destes dispositivos, em particular, para os baseados nos demodulares de baixo custo da Realtek, como o RTL2832U, sendo conhecidos por RTL-SDR ([SDR \(Software Defined Radio\) rtl-sdr 2017](#)). Outro projeto de *software* livre de grande presença é o GNU Radio, em desenvolvimento desde 2001 e escrito nas linguagens C++ e Python, trás um conjunto de

ferramentas para o processamento de sinais baseando-se em blocos, podendo utilizar a interface gráfica GNU Radio Companion (GRC) ([GNU Radio 2019](#)).

Aplicações deste dispositivo incluem radio armador, receptores de radio FM, receptor de TV Digital, *aircraft tracking*, interceptação de transmissões de voz não encriptadas e outras aplicações desenvolvidas pela comunidade ([RTL-SDR.COM 2019](#)).

Um receptor SDR, ou *transceiver* configurado como receptor, é excitado na entrada com o sinal provindo de uma antena. Este sinal é tratado, amostrado e processado conforme detalhado na subseção seguinte, entregando ao *hardware* conectado via USB, o sinal processado em tempo real, cabendo ao *software* não só regular parâmetros do processamento, como também a manipulação final dos dados para uso.

Um *transceiver* configurado como transmissor fará necessariamente o inverso do processo acima, excitando os terminais de uma antena com um sinal digital provindos do processo inverso.

2.3.1 *Hardware* Interno

A subseção seguinte detalha o *hardware* interno e funcionamento de um receptor SDR do tipo RTL-SDR, uma vez que o dispositivo utilizado para o trabalho pertence a tal família. A figura 7 apresenta em um diagrama de blocos do *hardware* em questão, ou seja, o *front-end* RF do dispositivo.

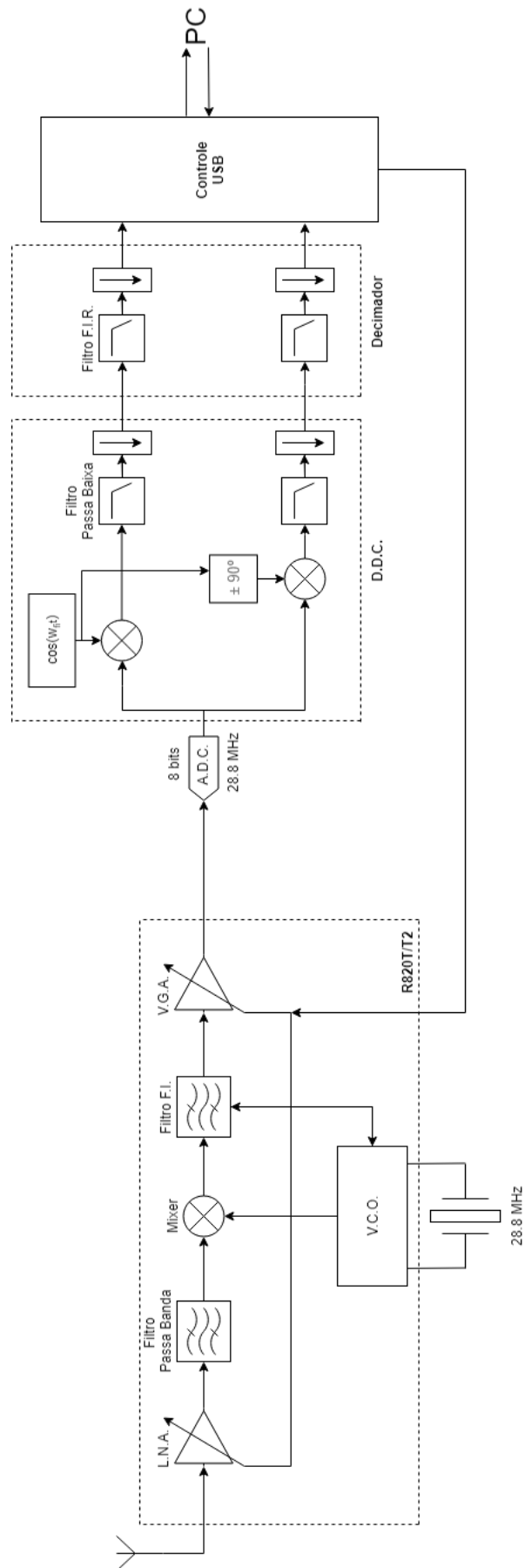


Figura 7 – Diagrama de blocos do *front-end* RF do RTL-SDR

Após a captação pela antena, o primeiro estágio é composto pelo sintonizador do dispositivo. No caso do RTL-SDR, as duas famílias de chips mais comuns são o Elonics E4000 e o Raphael Micro R820T/R820T2, ambos de estrutura equivalente.

O sinal recebido, passa por um amplificador de baixo ruído (LNA) e é filtrado por um filtro passa banda. Em seguida, é realizado o batimento de frequência com um oscilador local (VCO) de forma a converter a frequência do sinal para uma frequência intermediária (FI), filtrando-o e amplificando novamente por meio de um amplificador de ganho variável (VGA). O ganho produzido neste estágio é controlado por sinais do demodulador, que por sua vez é controlado por *software*, de forma otimizar a relação sinal ruído (SNR) do estágio. A faixa de operação do R820T varia de 42 MHz a 1002 MHz de acordo com o fabricante, entretanto os testes da comunidade determinam a faixa de operação 24 MHz a 1766 MHz ([Ferreira 2017](#)).

A próxima etapa trata-se da conversão do sinal recebido, ainda analógico, para digital. O RTL2832U possui um conversor analógico-digital (ADC) de 8 bits com amostragem de 28.8 MHz. O sinal então é modulado em quadratura, filtrado e subamostrado pelo digital downconverter (DDC).

Por fim o sinal modulado em quadratura é decimado utilizando um filtro FIR e aplicando outra subamostragem, trazendo a frequência das amostras para um valor entre uma das duas faixas, 225.001 kHz - 300 kHz Hz, ou 900.001 kHz - 3.2 MHz. Entretanto, 2.56 MHz é o valor testado e acordado entre a comunidade como o maior valor para amostragem em que o dispositivo se comporta de forma estável ([Ferreira 2017](#)).

2.3.2 Ferramentas de *Software* Livre

Nesta subseção, apresenta-se uma visão geral de funcionamento dos dois softwares livres utilizados pelo trabalho para a obtenção dos dados, sendo um o GNU Radio, utilizado para teste do dispositivo mas não atuando sobre os resultados do trabalho, e o outro uma das ferramentas da Osmocom *rtlsdr lib*, o *rtl_power*, escrito em linguagem C.

2.3.2.1 GNU Radio

GNU Radio é um pacote de desenvolvimento para processamento de sinais em blocos para implementação em *software radios*. Pode ser utilizado com *hardware* RF externo para criar SDRs, ou sem *hardware* em um ambiente de simulação, sendo vastamente utilizado para pesquisa e na indústria ([GNU Radio 2019](#)).

Além de implementação escrita, o pacote conta com a interface gráfica GNU Radio Companion, que permite a programação no formato de fluxograma, exemplificado pela figura 8.

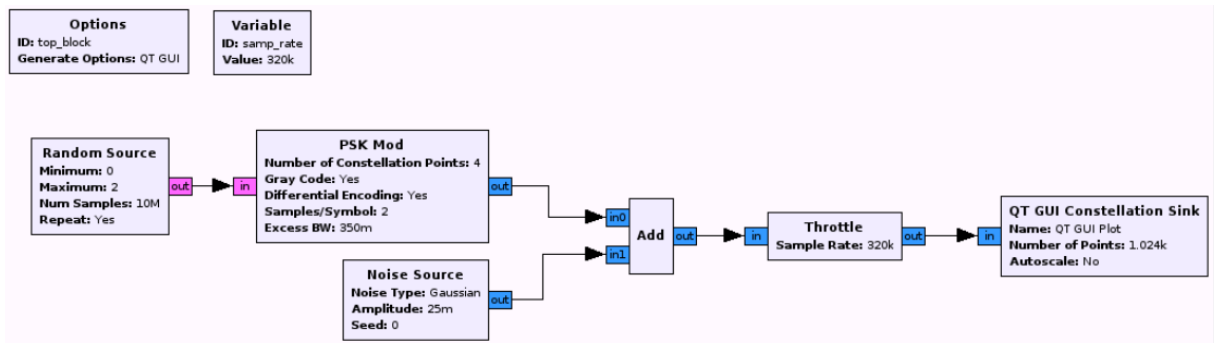


Figura 8 – Exemplo de implementação em blocos utilizando o GNU Radio Companion (GNU Radio 2019)

Uma das grandes características de trabalho do GRC é o tratamento e a visualização em tempo real do sinal. Para o trabalho, isto foi aproveitado na construção de um demodulador FM em tempo real para testes das frequências de interesse, sendo o formato de saída de tal demodulador apresentado na figura 9.

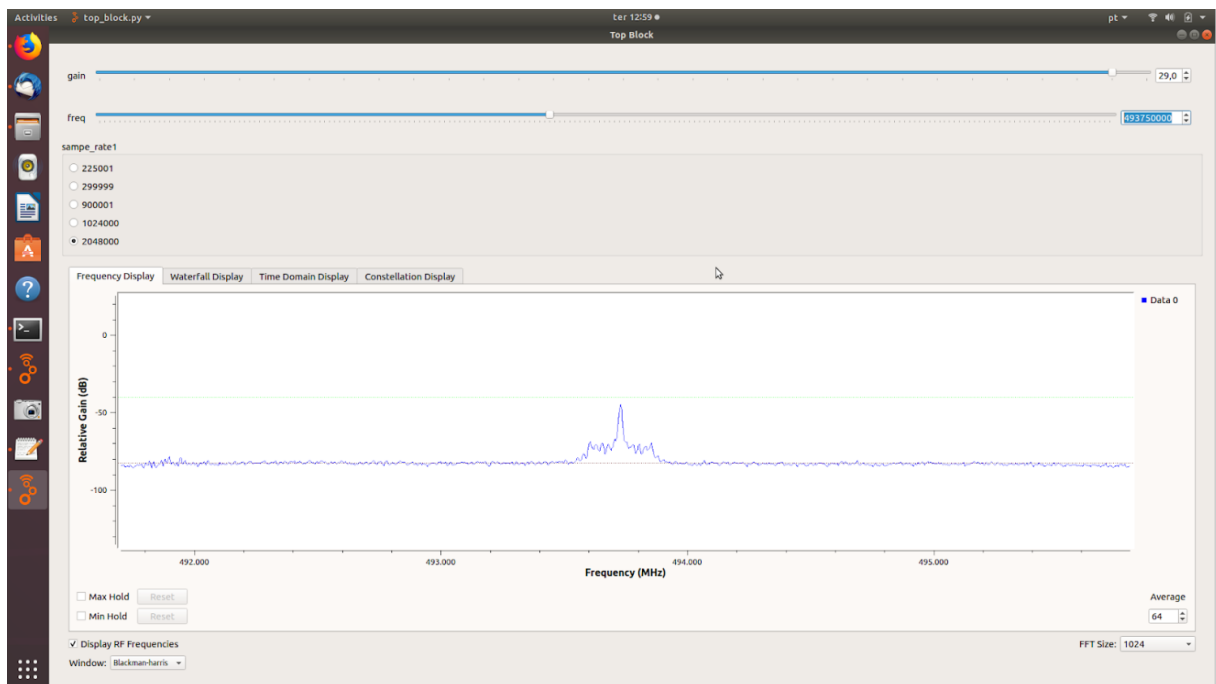


Figura 9 – Saída do demodulador FM construído com o GNC

Entretanto, para o objetivo do trabalho, de varredura em banda larga, a ferramenta não foi utilizada, uma vez que é necessário a realização de saltos em frequência em intervalos regulares já que a banda que se deseja varrer é muito maior que a banda do dispositivo. A ferramenta é inadequada devido a baixa velocidade em que realiza salto em frequência, uma vez que não tem

suporte nativo para tal e isto só foi implementado após alteração direta no código Python gerado pelo GNU Radio e o executando direto no terminal, pois deixa de funcionar com o GRC.

2.3.2.2 rtl_power.c

A biblioteca `rtlsdr lib` da Osmocomm, contém diversas funcionalidades de código livres voltadas a RTL-SDRs. Em particular, a ferramenta `rtl_power`, desenvolvida por Kyle Keen, é voltada para aplicações de varredura em banda larga com parâmetros ajustáveis e explorando a capacidade do dispositivo (Keen 2017), sendo umas da ferramenta base muito utilizada na construção de diversas outras ferramentas do gênero de varredura pela a comunidade radio armadora do RTL-SDR.

A ferramenta `rtl_power` é um programa escrito em C que utiliza de outras dependências da `rtlsdr lib`, realizando e registrando em um arquivo csv a Transformada Rápida de Fourier (FFT) da potência do sinal amostrado, reajustando rapidamente a frequência de operação do dispositivo para cobrir um espectro largo com vários rápidos saltos em frequência.

Ao utilizar a ferramenta, basicamente configura-se as frequências mínimas e máximas da banda que deseja-se varrer, o espaçamento das amostras da FFT, ou *binsize*, e a duração da varredura. A banda que se deseja varrer então é quebrada em várias faixas menores, de banda equivalente a banda de operação do dispositivo, sendo cada uma das faixas menores amostrada em um intervalo, aplicado-se e registrado-se a FFT da faixa. Em seguida, realiza-se o salto em frequência para a próxima faixa menor e o processo se repete. É possível ainda realizar múltiplos ciclos de varredura dentro de um intervalo, definindo-se um tempo de integração no qual a ferramenta realiza uma média das amostras do sinal naquele período.

A figura 10 esquematiza como os valores de potência do sinal são registrados no arquivo csv em dB.

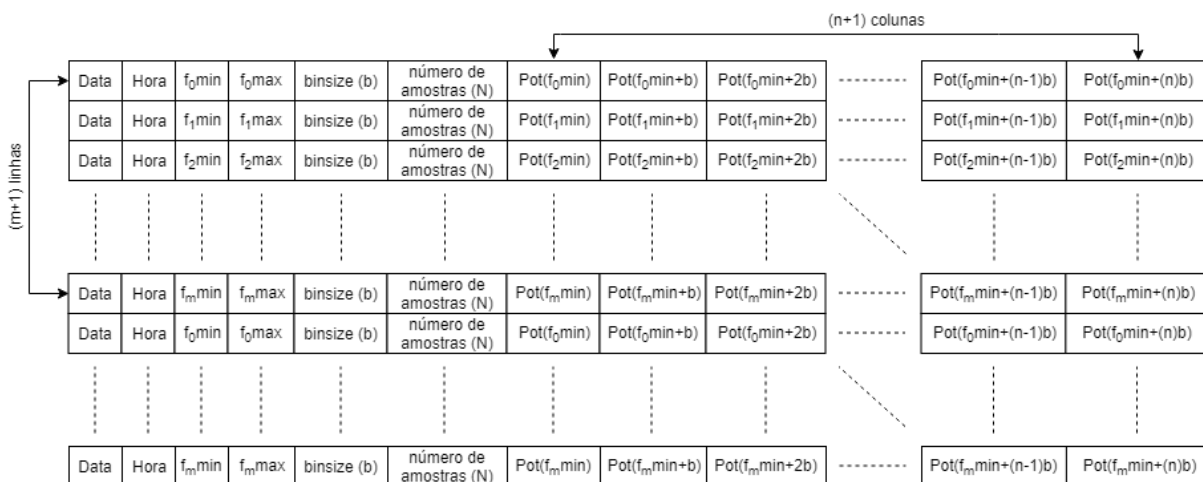


Figura 10 – Estrutura do arquivo csv produzido pelo `rtl_power`

A primeira coluna encontra-se a data da varredura, e a segunda a hora no formato hh:mm:ss. Como cada linha do csv expressa um salto em frequência, em uma varredura de múltiplos ciclos, os campos mudam conforme o horário em a varredura que foi registrada.

A terceira e quarta coluna encontram-se a frequência mínima e máxima de uma das das subfaixas da banda que deseja varrer, tendo seu valor repetido após o número de saltos necessários para uma varredura completa, expressando o próximo ciclo de varredura. Portanto a frequência mínima na primeira linha de cada ciclo, e a frequência máxima na última linha de cada ciclo, equivalem a banda que se deseja varrer.

A quinta coluna é o *binsize*, e este não muda ao longo do arquivo. A sexta é a quantidade de amostras registradas para o cálculo dos valores de potência. Da sétima coluna em diante, tem-se os valores de potência registrados, sendo a sétima a coluna o valor de potência para a frequência mínima daquela faixa, a oitava coluna o valor de potência para a frequência + o *binsize*, a nona coluna para $2 * binsize$ e assim por diante até a última coluna.

3 MATERIAIS E MÉTODOS

Neste capítulo expõe-se tudo o que foi utilizado no projeto quanto a *hardware* e *software*, e a forma de utilização destes para produzir o Capítulo 4, dividindo o capítulo em duas subseções. Em particular, na segunda subseção, são justificadas as diretrizes tomadas para o desenvolvimento do projeto.

3.1 Materiais utilizados

Nesta subseção tem-se listados todas as ferramentas utilizadas durante a realização do trabalho, tanto *hardware* quanto *software*.

Elementos de *hardware*:

- Receptor RTL-SDR V3
- Antena Log periódica de 14 Elementos
- Laptop Lenovo y50-70 (Computador pessoal do aluno)

Elementos de *software* utilizados:

- Sistema operacional uBuntu 18.04 LTS
- Osmocom rtl-sdr lib
- Linguagem Python (v 3.7)
- Pacote de desenvolvimento GNU Radio v3.7



Figura 11 – Dispositivo RTL-SDR utilizado

Dados comerciais do SDR:

- Largura de banda: Até 2,4 MHz (estável)
- Faixa de frequência de entrada: 500 KHz - 1766
- Impedância de entrada típica: 50 Ω
- Conversor Analógico-Digital: Realtek RTL2832U de 8 bits
- Sintonizador: Rafael Micro R820T2

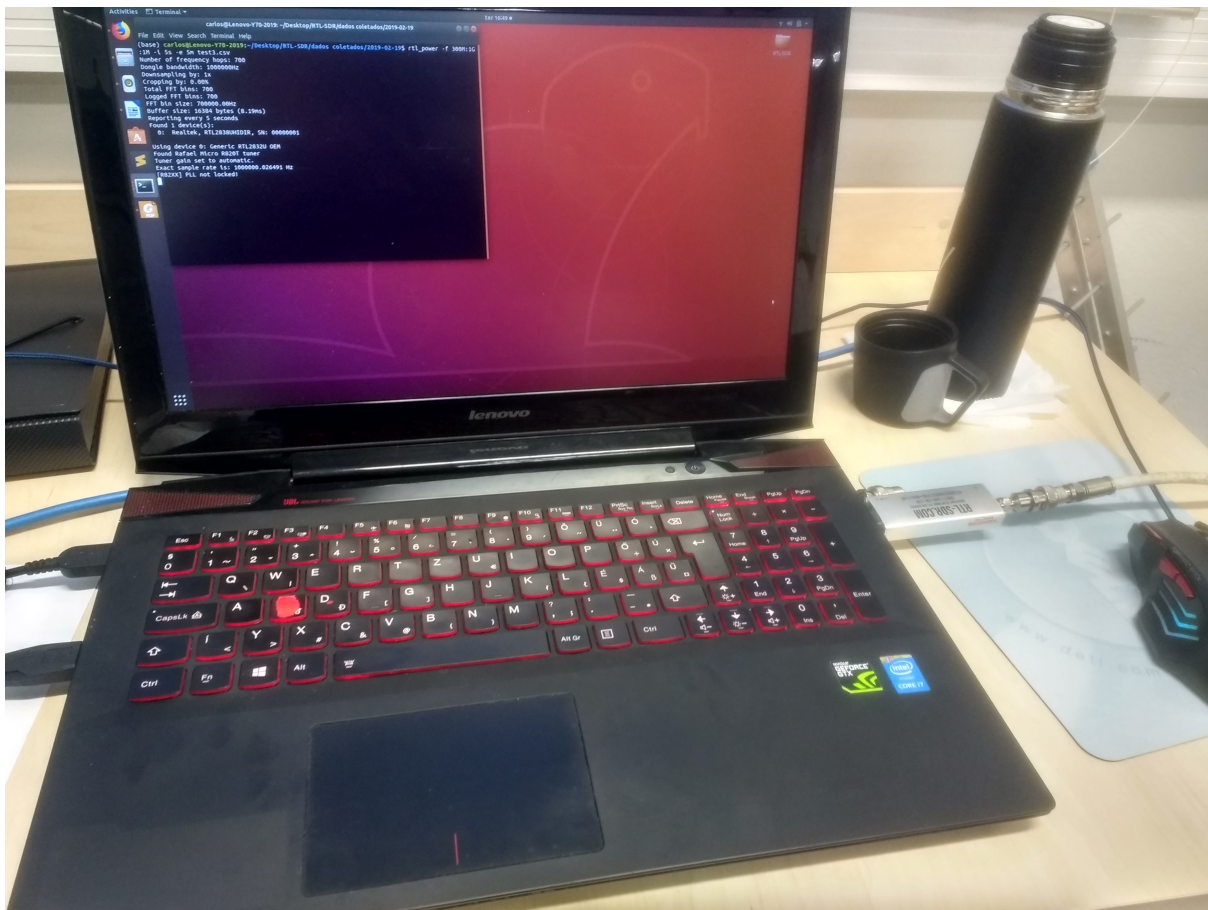


Figura 12 – SDR conectado ao laptop reportando status da varredura no terminal

Dados comerciais da antena ([S&L-LAMBDA 2019](#))

- Frequência de operação: 470 MHz - 806 MHz
- Ganho: 12 dB - 14.15 dB
- Impedância: 75Ω
- V.S.W.R $< 1,3 : 1$
- Polarização: Horizontal
- Fabricação em alumínio



Figura 13 – *Setup* utilizado para as medições; antena à direita

3.2 Metodologia aplicada

3.2.1 Faixa de medição escolhida

Tendo em vista a antena, cada elemento (dipolo) da mesma possui uma resposta em frequência contínua, que ressoa em $\frac{\lambda}{2}$. De forma simplificada, a soma das respostas em frequência dos diferentes elementos constrói a resposta, de característica banda larga, da antena. A faixa de 470 MHz a 806 MHz corresponde a faixa na qual o conjunto de dipolos da antena melhor responde, isto é, a faixa na qual há menor atenuação sobre o sinal RF por ela recebido. Fora desta faixa, a antena ainda capta sinais, entretanto estes são mais atenuados conforme a frequência do sinal recebido dista da faixa de operação da antena.

Como num cenário real, os sinais de diferentes frequência são recebidos com potências diferentes, nada impede que um sinal fora desta faixa seja recebido com tamanha potência que este ainda é registrado pelo conjunto após ter sido atenuado na recepção. Levando isto em conta, a faixa de interesse escolhida para as medições foi de 300 MHz a 1 GHz, aumentando o leque de dados disponível. Dentro desta faixa são encontrados serviços como radiodifusão, radioarmador, radionavegação, radiolocalização, radioastronomia, meteorologia, pesquisa e operação espacial, serviços móveis, entre outros. Alguns dos serviços citados e suas respectivas faixas de frequências encontram-se listados abaixo ([ANATEL 2018](#)):

- 328,6 MHz - 335,4 MHz radionavegação (Sistemas de Aterrissagem por Instrumentos - ILS)
- 400,15 MHz - 401 MHz Pesquisa espacial
- 406,1 MHz - 410 MHz radioastronomia
- 450 MHz - 470 MHz Móvel (4G) ^a
- 470 MHz - 608 MHz Canais UHF 14 a 36 (TV)
- 608 MHz - 614 MHz Radio astronomia (canal UHF 37)
- 608 MHz - 698 MHz Canais UHF 38 a 51 (TV)
- 698 MHz - 960 MHz Móvel (GSM, 3G e 4G)
- 902 MHz - 928 MHz ISM

As faixas de serviços móveis comerciais encontram-se listadas a seguir:

- 451-458 MHz / 461-468 MHz

^a Licitadas juntamente com as frequências de 2,5GHz em 2012, ainda não estão sendo utilizadas pelas operadoras ([Teleco 2019](#))

- 703-748 MHz / 758-803 MHz 4G
- 806-821 MHz / 851-866 MHz GSM
- 824-849 MHz / 869-894 MHz GSM
- 898,5-901 MHz / 943,5-946 MHz GSM e 3G
- 907,5-915MHz / 952,5-960 MHz GSM e 3G

3.2.2 Procedimento de varredura

Para a realização das varreduras, foi utilizado o programa `rtl_power.c`, devida a alta capacidade de rápido salto em frequência, alterando a frequência do oscilador local do R820T2, para que o batimento desta com a do sinal recebido, produza na saída do misturador a frequência intermediária (FI) da arquitetura super heteródina do *front-end* RF do SDR.

Deseja-se varrer o espectro de 300 MHz a 1 GHz buscando não só transmissões perenes, como *broadcast* de rádio e TV, como também transmissões intermitentes. Tais transmissões intermitentes podem ser providas de diversas fontes, desde uma transmissão ADS-B ([ADS-B 2019](#)), indicando localização de uma aeronave, transmissões provindas de *links* de radio astronomia e radionavegação, até mesmo *links* de comunicação móvel, pois para o nosso receptor, estes canais são exemplos canais que não estão ocupados a todo momento. Em particular no caso de comunicações móveis, estas não estão restritas somente aos serviços de telefonia comerciais (GSM, 3G e 4G), englobam também transmissões de diferentes radiocomunicadores (ex: *walk-talk*).

Assim, a varredura é realizada entrando com o comando no terminal:

```
_power -f 300M:1G:22k -i 30s -c 0.5 -e 1h 'survey.csv'
```

Onde os comandos inseridos explicitam:

- “-f 300M:1G:22k ”: espectro varrido, de 300 MHz a 1 GHz, com a resolução da FFT imediatamente menor a 22 kHz, multipla da frequência de amostragem (21,875 KHz).
- “-i 30s ”: intervalo de integração das leituras, 30 segundos.
- “-c 0.5 ”: percentual de corte, ou *crop*, de 50%. Na prática, os filtros internos ao SDR produzem atenuação nas beiradas da janela amostrada e este recurso é usado para contornar este efeito (maior detalhamento a seguir).
- “-e 1h ”: duração da varredura, ou *exit time*, de 1 hora.
- “'survey.csv' ”: nome dado ao arquivo de saída.

Como mencionado, internamente o SDR produz certa atenuação nas bordas da janela amostrada, exemplificado na figura 14. Ao utilizarmos o recurso de corte, são mantidas as leituras

mais ao centro de acordo com o percentual escolhido, e o salto do dispositivo é recalculado para cobrir o espectro perdido, dando assim mais saltos. Escolhendo-se 50% garantimos não só que as leituras terão atenuação negligível devido aos filtros internos, como também multiplicamos a quantidade de saltos por um número inteiro (dobrando os saltos). Percentuais de corte como 30% e 40% produzem pequenos gaps de frequência entre as linhas do arquivo csv de saída, necessitando de correção via *software*, escolhendo-se 50% isto não é necessário.

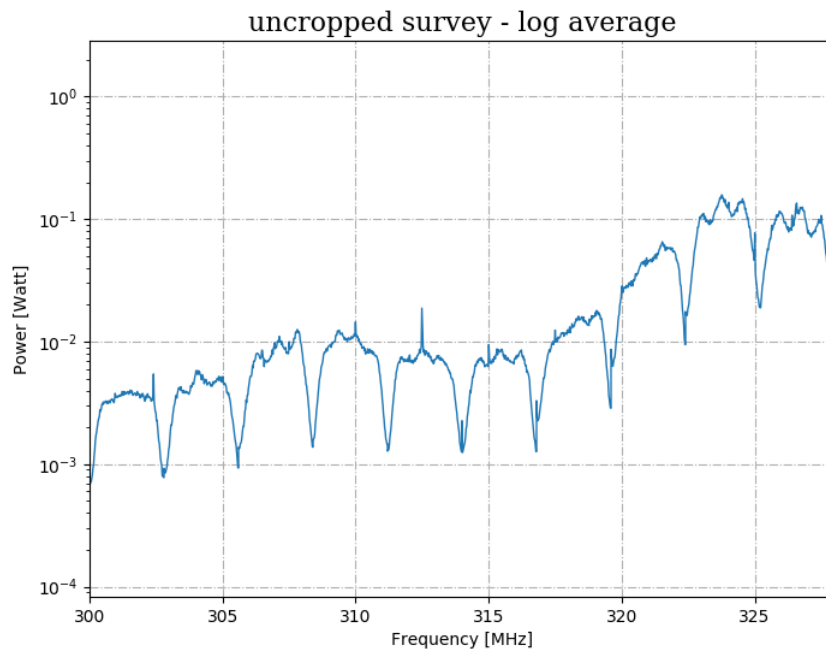


Figura 14 – Varredura com Efeito de atenuação nas beiradas da janela amostrada

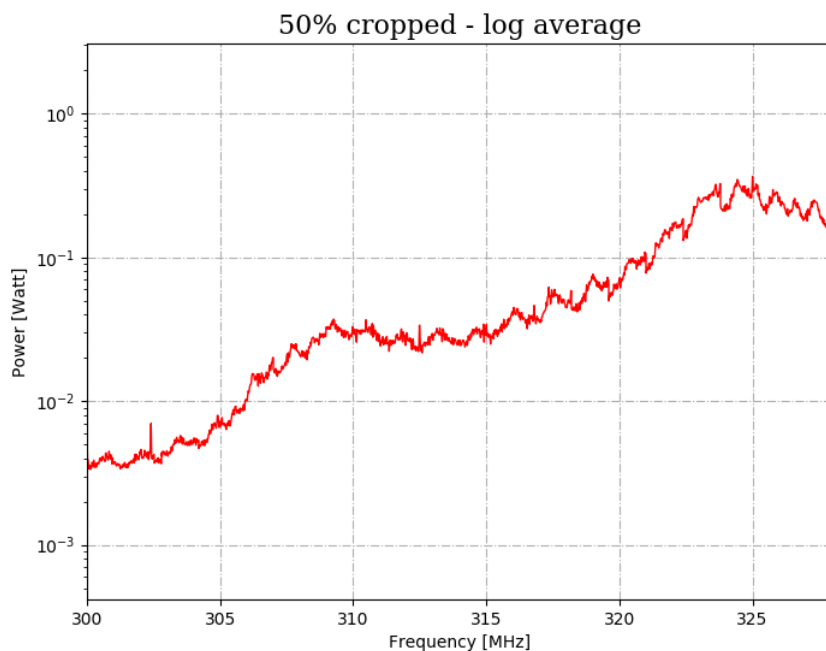


Figura 15 – Varredura com a utilização do recurso *crop* do *rtl_power*

A duração de 1 hora foi escolhida arbitrariamente. A resolução da FFT de aproximadamente

22 kHz foi escolhida arbitrariamente após uma bateria de teste no laboratório. Submúltiplos da frequência de amostragem abaixo não só aumentavam o tamanho do arquivo de saída proporcionalmente, como também acabam comprometendo a visibilidade dos dados. Valores muito acima, resultam em poucos pontos por janela, dando pouca visibilidade aos dados. As figuras 16 e 17 demonstram estes efeitos.

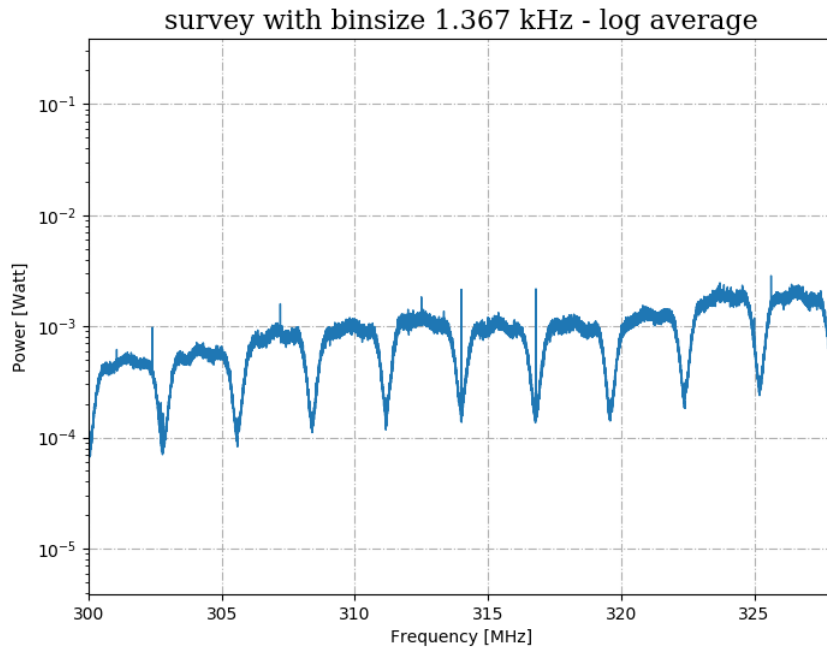


Figura 16 – Varredura com *binsize* de 1.367 kHz

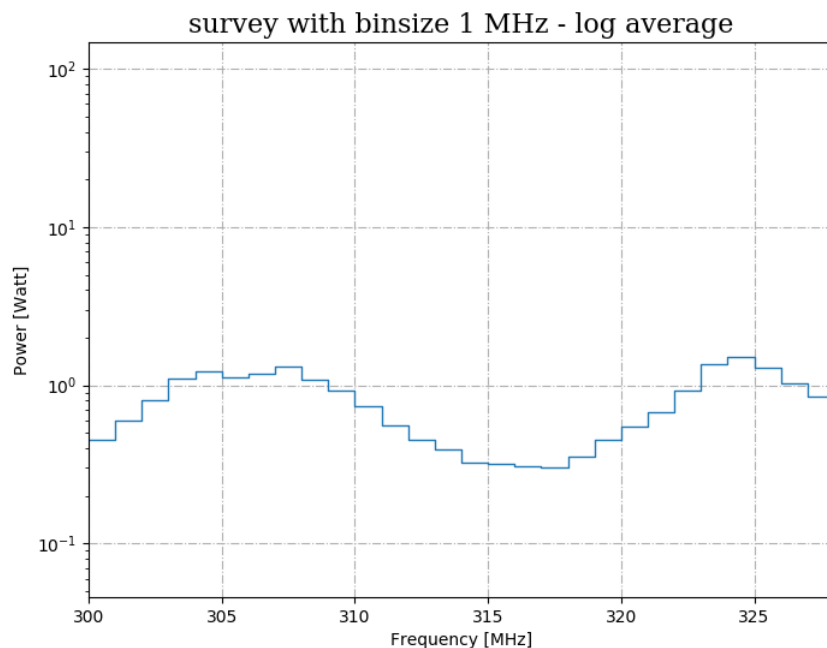


Figura 17 – Varredura com *binsize* de 1 MHz

O intervalo de integração de 30 s foi escolhido por conveniência pois intervalos muito pequenos, além de aumentarem proporcionalmente o tamanho do arquivo de saída, também

correm o risco de serem menores que o tempo total que o dispositivo leva para realizar todos os saltos de uma iteração da varredura, sendo o *default* 10 s (Keen 2017). Outro motivo para a escolha de 30 s foi visando o quanto este tempo poderia influenciar sobre a leitura de canais intermitentes, pois o dado retornado pela ferramenta já é uma média das leituras de potência neste intervalo, se escolhermos um intervalo muito grande pode acabar "diluindo" a diferença de potência do canal ocupado/livre.

Quanto as diferentes varreduras em si, foram realizadas 8 varreduras ao longo de 4 dias seguidos, uma das 12h00 às 13h00, e a outra das 15h00 às 16h00. Estes 4 dias foram dia 8 (Sexta), 9 (Sábado), 10 (Domingo) e 11 (Segunda) de Março de 2019. Supõe-se que a atividade dos canais, principalmente os de telefonia, terão uma boa flutuação de um horário próximo do almoço e outro no meio da tarde, e que a atividade durante a semana seja diferente nesses horários em relação ao fim de semana, o que ajudaria a distinguir os canais. Todas as demais varreduras anteriores a estas datas, serviram o propósito de estudo do dispositivo, definição de *setup*, TTD *software* em Python e embora não estejam diretamente nos resultados, fizeram parte do desenvolvimento do projeto.

3.2.3 Utilização do Python

O arquivo .csv gerado de cada varredura, dispõe os dados da forma mencionada na seção 2.3.2.2. Com a larga duração da varredura, alta da resolução da FFT e o pequeno intervalo de integração, a quantidade de dados tem manuseio limitado em *softwares* de planilhas como o Microsoft Excel e o LibreOffice Calc, chegando até mesmo a exceder o limite de colunas por planilha dos mesmos para varreduras alta resolução (16384 colunas para o Excel (Microsoft 2019) e 1024 para o Calc (LibreOffice 2019)). Isto dificulta tanto manipulação quanto visualização dos dados, sendo isto um dos motivos pela escolha do Python para a realização do projeto.

A linguagem de código aberto, formada por uma grande comunidade, com desenvolvimento voltado à legibilidade do código, possui uma gama de bibliotecas voltadas à análise de dados. Para o desenvolvimento do projeto, a biblioteca **Pandas** do Python, permite carregamento do arquivo .csv em uma estrutura tabular *DataFrame*, facilitando manipulação dos dados. Outras bibliotecas que também são utilizadas no projeto são **NumPy** e **SciPy**, permitindo a implementação de métodos computacionais eficientes e otimizados para operar com grande número de dados, além de submódulos para aplicações em Estatística, sendo grandes facilitadores para o desenvolvimento do projeto. Para a visualização dos resultados, é utilizada a **Matplotlib**.

3.2.4 Desenvolvimento dos indicadores

Durante o processo, o GNU Radio foi utilizado no início para recepção e visualização em tempo real buscando canais ocupados para usá-los de referência ao construir os indicadores, como por exemplo o canal entre 482 MHz e 488 MHz (Tv Clube - canal 16 UHF digital), que é uma transmissão praticamente permanente. Outro exemplo são as faixas de telefonia celular listadas

anteriormente, onde se espera diferentes níveis de atividade ao longo do dia.

Os indicadores são produzidos a partir do arquivo de saída do `rtl_power`, portanto os dados são média da potência registrada nos intervalos de integração. Sendo assim, os indicadores partem de dados de potência do sinal, é necessário ter isto em mente e diferir da situação em que os dados são a amplitude do sinal ao longo do tempo, onde o ruído branco por exemplo, pode ser modelado por ruído AWGN, de média nula e potência igual desvio padrão.

No caso do projeto, os dados de potência serão sempre maior que zero, por mais que sejam pequenos, e mesmo que por hipótese, o ruído tenha a mesma média, isto não se refletirá nos dados principalmente devido ao VSWR da antena, nem levando-se ainda em conta o possível alteração da relação da impedância de entrada do dispositivo com a da antena devido a frequência.

Assim, busca-se por indicadores que permitam distinguir os canais de uso perene, intermitentes e ociosos baseados na distribuição dos dados de potência ao longo da duração de uma varredura, das diversas varreduras. A hipótese em que para as diferentes varreduras, os canais que estão sempre em uso, como os ociosos, terão o mesmo comportamento, permitindo encontrar similaridades nas diferentes varreduras. Vale ressaltar que os dados do arquivo de saída são em dB Watt e portanto precisam ser convertidos em Watt para uma manipulação segura.

4 RESULTADOS E DISCUSSÕES

Neste capítulo encontram-se, tudo o que foi produzido no projeto. Primeiramente, uma vez que os códigos Python encontram-se por completo em anexo, tem-se discutido o algoritmo por trás do *software* produzido em uma sessão, já que o *software* em si é um produto do projeto. Na sessão seguinte, encontram-se os indicadores produzidos pelo *software*, com uma descrição comparando uma varredura isolada e a todas as varreduras concatenadas, discutindo-se as similaridades e diferenças encontradas.

A segunda sessão contém em sequência, dois gráficos para cada indicador, um da varredura das 12h às 13h do dia 08, e um das 8 varreduras combinadas. Os indicadores em ordem são: Potência rms média, desvio padrão da potência rms, razão entre desvio e média, assimetria, curtose, estatística e probabilidade p de rejeição de H_0 no teste normalidade D'Agostino k^2 (D'Agostino's K-squared test 2019). Todos os indicadores cobrem o espectro de 300 MHz à 1 GHz. Seguindo a sequência, são apresentados gráficos multi eixos, seguindo o mesmo modelo de dois gráficos por página, para fácil confronto visual dos indicadores acima.

4.1 Software Desenvolvido

O *software* divide-se em dois submódulos, o principal que trata os dados e o secundário para a visualização dos indicadores. A seguir discute-se somente o principal.

Dado o arquivo csv de entrada, este é convertido em *Dataframe*, e deste extrai-se três características importantes: o *shape* do *Dataframe* (quantidade de linhas e colunas), o *binsize*, que é o espaçamento entre os pontos na janela da FFT e a variável chamada de "*gap*", que nada mais é que a quantidade de linhas até a próxima iteração da varredura para aquela janela de frequências, ou seja, é a quantidade de saltos em frequência de cada iteração da varredura.

Conforme mencionado na sessão 2.3.2.2 do capítulo 2, possui a estrutura da figura 10, com isto a extração destas características torna-se simples. A figura 18 explicita a repetição das frequências inicial e final de um salto em frequência, de um *Dataframe* de "*gap*" conhecido de 500 linhas. Para encontrar o *gap* a partir do próprio *Dataframe*, trata-se de uma simples comparação linha a linha dos valores de frequência, dentro de um *loop while* com um contador.

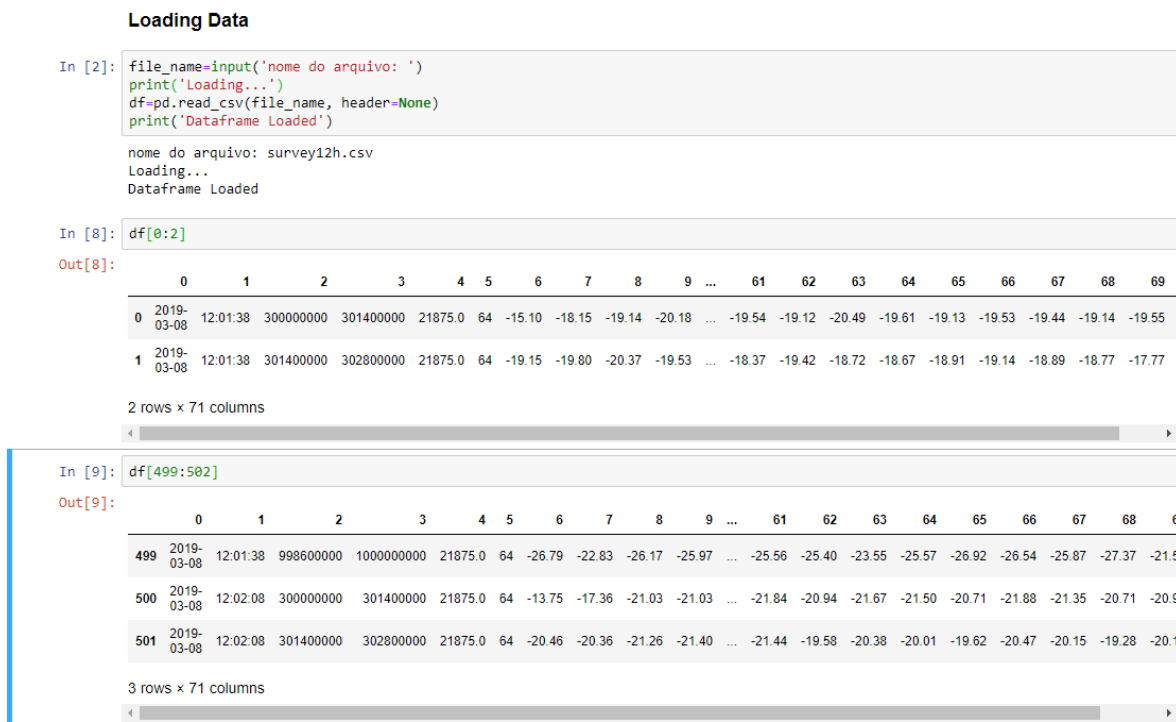


Figura 18 – Visualização da variável *gap*

Feito isto, o software inicia um mapeamento, partindo da linha 0, para cada dado de potência de determinada frequência, ou "*bin*", busca-se todos os dados de potência da varredura para aquela frequência nos diferentes momentos da varredura e os armazena num *array* Numpy. Isto é feito dentro de uma estrutura de repetição aninhada de três *for loop*, para cada coluna, em dada linha, o próximo *bin* da varredura está na mesma coluna *gap* linhas abaixo, ao mapear todos os dados de uma mesma frequência, construímos um *array* com todas as leituras ao longo do período varrido, tendo assim a distribuição dos dados de potência daquela determinada frequência. Avança-se para a próxima coluna e se repete o processo até a última coluna do *Dataframe*, para então repetir o processo para a próxima a linha, realizando todo este processo para as primeiras *gap* linhas. A figura 19 é um fluxograma do algoritmo deste submódulo.

Com cada *array* Numpy, podemos utilizar de recursos já implementados nas bibliotecas NumPy e SciPy, para calcular características da distribuição como média e desvio padrão, ou até mesmo outras funções de autoria própria que utilizem a distribuição como um dos argumento. Para o projeto, os indicadores são produzidos à partir destes *arrays*, buscando-se então métricas relevantes para o contexto das varreduras, listadas e discutidas na sessão seguinte, utilizando da Matplotlib para a visualização.

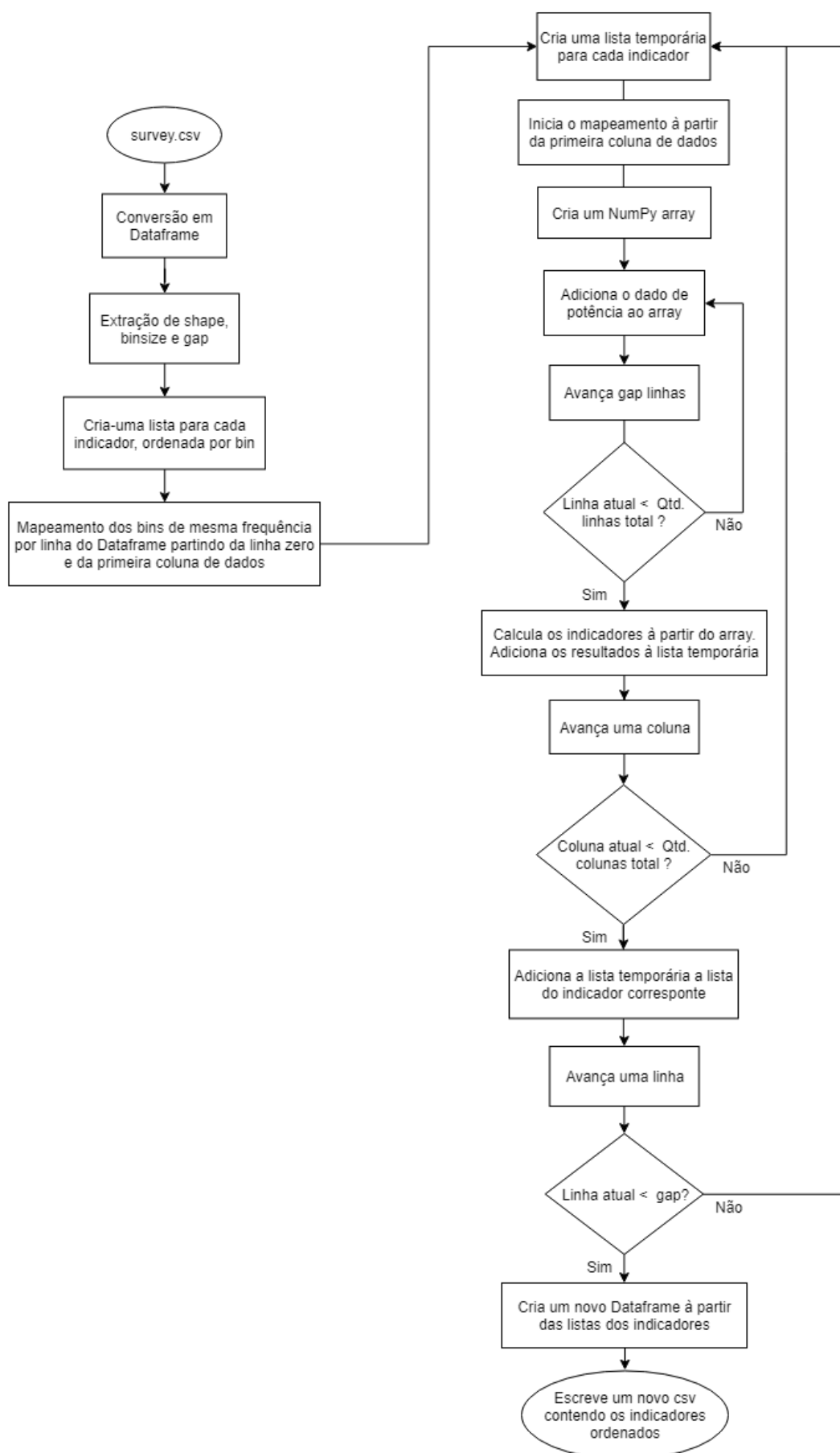


Figura 19 – Fluxograma do *software* de tratamento dos dados

4.2 Indicadores

Como mencionado anteriormente, esta sessão primeiramente possui dois gráficos para cada indicador, um de varredura isolada e outro das varreduras concatenadas.

Para ponto de partida, optou-se pela média, sendo um dos indicadores mais simples e diretos. Nas figuras 20 e 21 podemos observar as várias faixas e picos acima da linha próxima de zero. Dos principais notados, tem-se as faixas de 325 - 375 MHz, 860 - 890 MHz, como também tem-se picos isolados em torno de 470 MHz, 485 MHz, 537 MHz, 674 MHz e 957 MHz. Nota-se também que para a maioria dos picos mais expressivos, sua potência média ao longo da varredura é maior na leitura isolada, indicando maior ocupação do canal no horário do almoço e/ou possivelmente canais parte do tempo ociosos, ou transmitindo com baixa potência em horário de menor atividade, ao analisar as leituras concatenadas, principalmente com relação aos picos em 674 MHz e 957 MHz, e na faixa de 860 - 890 MHz.

Agora, temos de lembrar da real sensibilidade do aparelho e que *links* de baixa potência são comuns, e um gráfico linear não tem boa visibilidade para as leituras de baixa potência. Pensando nisso, as figuras 22 e 23 promovem uma releitura dos dados de potência anteriores em um gráfico semilog. Este tipo de gráfico, evidencia como o VSWR da antena influencia ao tentar identificar o que possivelmente é puramente ruído na leitura (ou seja canais ociosos) pois existe um certo envelope em torno dos mínimos da varredura. No geral, a maior parte da ocupação se concentra acima de -20 dB, entretanto seria “muito simplório” utilizar este valor como divisor (*threshold*). Além da influência do VSWR ser concorrente ao valor de separação para toda a faixa, não é possível ter uma distinção de perene e intermitente.

Pensando justamente em uma distinção de perene e intermitente, analisa-se em sequência o desvio padrão, por ser uma medida utilizada para indicar a uniformidade dos dados. Uma distribuição que apresenta alta variação das amostras com relação a média (tendo assim desvio padrão maior), indica que a potência daquele canal varia bastante ao longo da duração da varredura, o que pode significar que hora aquele canal tem um pico de potência/uso e hora não, como também uma variação próxima de zero indica um nível de potência/uso estável, ou em outras palavras, um candidato a indicador do uso do canal intermitente ou perene. Nas figuras 24 e 25, observa-se alguns grupos de picos espalhados, alguns em comum entre as figuras e outros não. Nota-se que as varreduras concatenadas possuem alguns picos onde não há variações (ou há, mais próximas próximas de zero) na leitura isolada, citados nas faixas 325 - 375 MHz e 825 - 835 MHz), sendo que a primeira faixa apresentou picos de potência nas figuras 20 e 21, e segunda faixa não. É notado também, na leitura isolada, uma variação maior na faixa 860 - 890 MHz, que também possui maior média potência ao longo da hora varrida, sabendo que esta faixa contém serviços de telefonia GSM, é de se esperar uma maior atividade deste serviço durante o horário do almoço.

Ao realizar uma analogia quanto ao gráfico linear de potência média, repete-se o problema

de visibilidade das leituras de baixa potência. Sabe-se que o desvio padrão possui a mesma unidade dos gráficos anteriores (no caso watt) e que sua medida é em relação à própria média. Se utilizarmos de exemplo, das figuras 21 e 25, o desvio de 10 em 470 MHz, com potência de 1.75 e o desvio de 0.75 em 674 MHz com potência 0.75, temos no segundo caso que o desvio é da mesma magnitude que a própria média. Podemos ver que desvio padrão direto não é um bom indicador, mas uma razão entre o desvio e a média é um candidato a indicador mais expressivo, uma vez que contorna o problema do VSWR, dando o desvio em relação à média para cada frequência medida.

Utilizando agora a razão entre desvio e média, nas figuras 26 e 27, no gráfico isolado, vemos que não só surgiram vários novos pequenos picos, dando visibilidade melhor às leituras de baixa potência quanto a ser perene ou não, como também no gráfico combinado há picos de 2x (em 470 MHz) a 5x (faixa acima de 850 MHz) o que foi lido no gráfico de desvio padrão das leituras concatenadas, apontando para um uso intermitente mais alto. Uma vez que, nas leituras concatenadas a média tende a baixar, por o canal ficar mais tempo ocioso, as leituras de maior potência (no momento do uso) possuem uma distância da média ao quadrado grande o suficiente para "jogar" a variância pra cima, o que explica o aumento do desvio padrão registrado. Outro ponto é com relação a transmissões perenes, a Tv Clube por exemplo, possui desvio padrão baixo, apontando para uma média de potência mais estável.

Como citado anteriormente, busca-se por indicadores partindo unicamente da distribuição, e como pode-se notar, nenhuma hipótese foi levantada quanto ao formato da distribuição além de que os canais ruidosos devem possuir indicadores comuns, entretanto com apenas média e desvio padrão, há poucos indícios para realizar uma distinção mais rebuscada entre intermitentes e perenes. Conforme visto, certas frequências, de canais supostamente intermitentes, possuem alta razão desvio/média nas leituras concatenadas devido a uma parcela menor dos dados ter potência muito mais alta, logo a forma da distribuição possivelmente tem uma grande parcela dos dados concentrados abaixo da média e uma parcela menor bem acima da média. Até o momento os indicadores utilizados provêm do primeiro e segundo momento da distribuição (média e desvio padrão), e seguindo esse questionamento que poucas amostras, de valores extremos resultam em alto desvio (ou *outliers*), levanta-se o uso de indicadores de assimetria e curtose da distribuição para realizar este estudo.

Observando os gráficos de assimetria em 28 e 29, nota-se uma grande variação dos valores, desde abaixo de zero a acima de 10 na leitura isolada e acima de 30 nas leituras concatenadas. Pouco são os casos da assimetria ser abaixo de zero, na maioria esmagadora dos casos, as distribuições são acima de zero, ou seja, possuem a cauda para a direita, evidenciando a existência de uma parcela menor das amostras bem acima da média, impactando no primeiro e segundo momentos da distribuição (indício de *outliers*). Quanto a os casos próximos de zero, que também possuem baixa potência média e baixa razão desvio/média, estão agora mais propensos a ser classificados como ruído. Nota-se em inspeção visual dos gráficos que os valores de mais alta

assimetria coincidem com os picos dos gráficos anteriores, entretanto há valores de alta assimetria em frequências não se enquadrando no caso mencionado.

Voltando-se a análise quanto a curtose, mais sensível a *outliers*, o resultado imediato das figuras 30 e 31, ao comparar-se com os gráficos de assimetria é coincidência das frequência de ocorrência dos maiores picos, reforçando o quanto essas frequência são afetadas por *outliers*, indicando canais de uso intermitente com uso de potência elevado durante as transmissões em um curto período. O segundo resultado é analisando a magnitude das leituras, que para a maior parte do espectro varrido, está muito distante de zero, que embora também possa ser observado nos gráficos de assimetria, há discrepância maior, com picos que ultrapassam 60 (alguns chegando a ultrapassar 100) na caso isolado, e ultrapassando 300 nas leituras concatenadas (alguns chegando a ultrapassar 800). Isso nos leva a questionar que, ou boa parte do canais ruído possuem distribuição bem diferente de uma normal, ou o número de amostras é baixo para ter-se uma boa perspectiva quanto ao formato.

Ao utilizar os indicadores de curtose e assimetria para extrair conclusões sobre o formato da distribuição de determinadas frequências, abre-se a porta para incluir nos indicadores, a análise de normalidade do dados da distribuição desta frequências. Testes de normalidade clássicos em estatística como os de D'Agostino k2, Anderson-Darling e Shapiro-Wilk ([Normality test 2019-Brownlee 2018](#)), podem aprofundar ainda mais os indicadores para a distinção de transmissões perenes e intermitentes. Tais testes visam determinar se um conjunto de dados é bem modelado por uma distribuição normal, computando a probabilidade de a distribuição em questão ser modelada de tal forma.

Busca-se então averiguar se transmissões possuem comportamento com tal teste, opta-se submeter o conjunto de dados ao teste de D'Agostino k2, que realiza transformadas da curtose e assimetria, sendo uma sequência natural ao estudo da forma da distribuição a partir do terceiro e quarto momentos.

Utilizando da **SciPy**, obtem-se a estatística combinada das transformadas de assimetria e curtose (soma quadrática)([D'Agostino's K-squared test 2019](#))([scipy.stats.normaltest 2019](#)) e a probabilidade **p** de que o teste não pertença a uma distribuição normal (ou seja, probabilidade de negação da hipótese nula, tomando $p > 0.05$ como valor de referência para falha da hipótese ([D'Agostino's K-squared test 2019](#)). Com relação a estatística do teste, nota-se uma grande diferença com relação a ordem de grandeza da estatística do teste nas figuras 32 e 33, chegando a ser de 10x a 20x maior nas leituras concatenadas na maioria dos picos, com a ocorrência de um pico peculiar em torno de 650 MHz de 40x. Já com relação a probabilidade de negação da hipótese de normalidade, vide as figuras 34 e 35, há uma grande redução nas faixas de frequências com probabilidade $p > 0.05$ nas leituras concatenadas, de forma que, se interpretada diretamente como probabilidade de distribuição normal, tem-se um resultado contraditório, pois um vasto espectro é deixado de fora, sendo consideradas transmissões intermitentes.

No anexo 6.2, tem-se todas as figuras para cada indicador das 8 leituras lado a lado, em

particular, a figura referente à probabilidade p do teste de normalidade usados e vê se claramente que em nenhum caso isolado assemelha-se ao caso do indicador para as leituras concatenadas. Isto evidencia que existe alguma diferença entre os mesmos canais ruidosos nas diferentes leituras que não é clara nos demais indicadores.

Entretanto, as leituras combinadas são composta de varreduras de horários diferentes, e assume-se que os canais ociosos sejam compostos de resíduo eletromagnético. Uma vez que o uso dos canais difere nos diferentes horários e dias, levanta-se a nova hipótese de que o resíduo total das atividades, ao longo da faixa espectral, também pode variar, e por mais baixa que seja essa variação em escala absoluta, é possível, que em relação à magnitude do próprio ruído, já seja o suficiente para ao concatenar as leituras, falhar no teste de normalidade, sendo só consideradas normais, as distribuições de canais que tenham ocupação devido a transmissões com flutuação normal ao longo do dia e em dias diferentes, ou seja, canais candidatos à transmissões RF perenes, como por exemplo canais de televisão, radio-controle das estações base do sistema de telefonia.

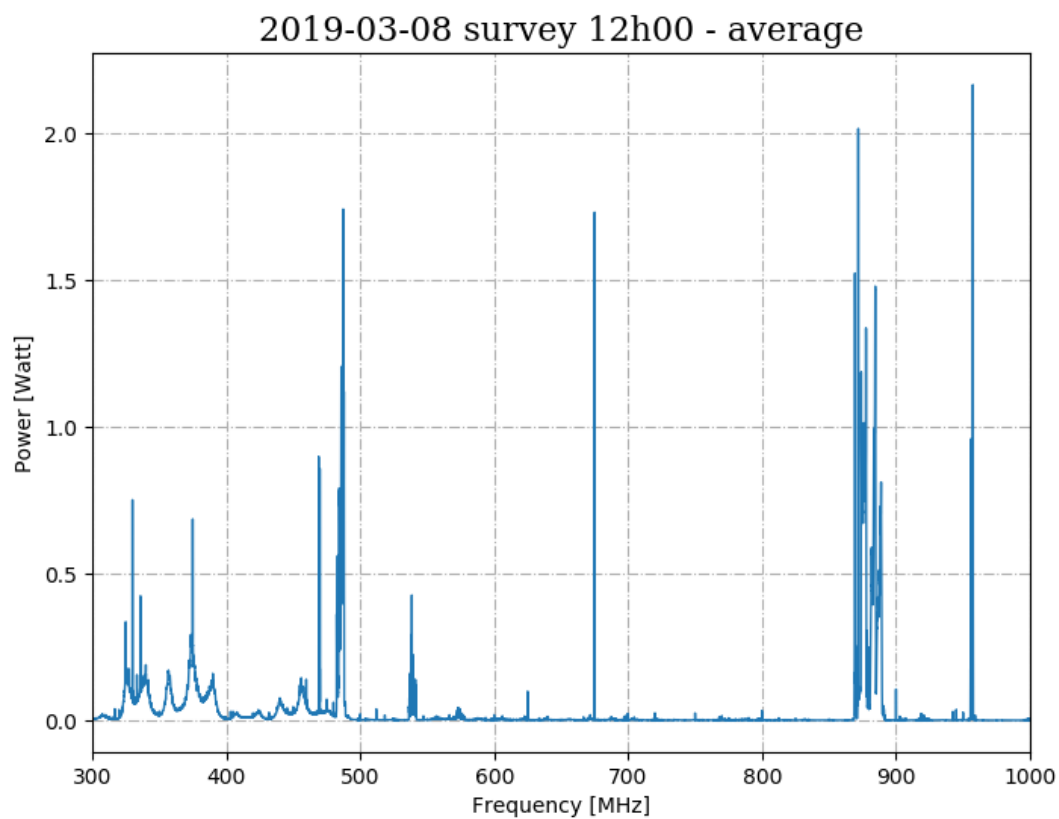


Figura 20 – Potência rms média - 12h00 às 13h00 de 08/Mar/2019

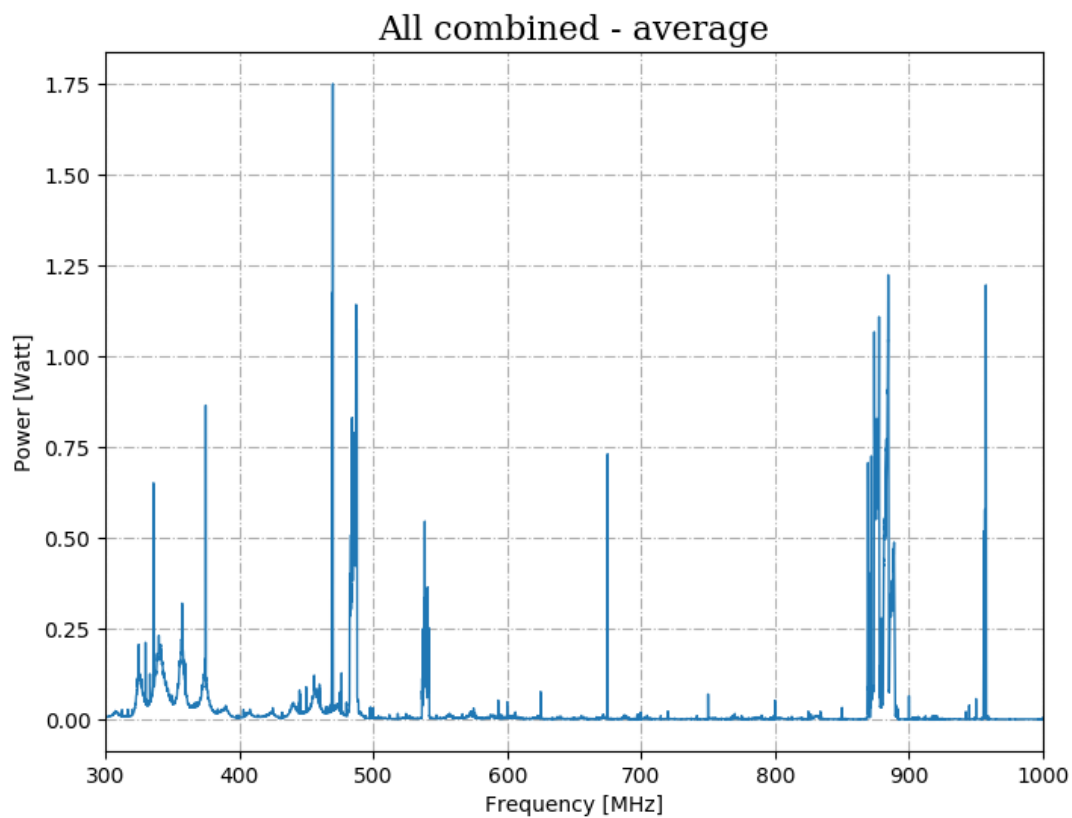


Figura 21 – Potência rms média - Todas as leituras

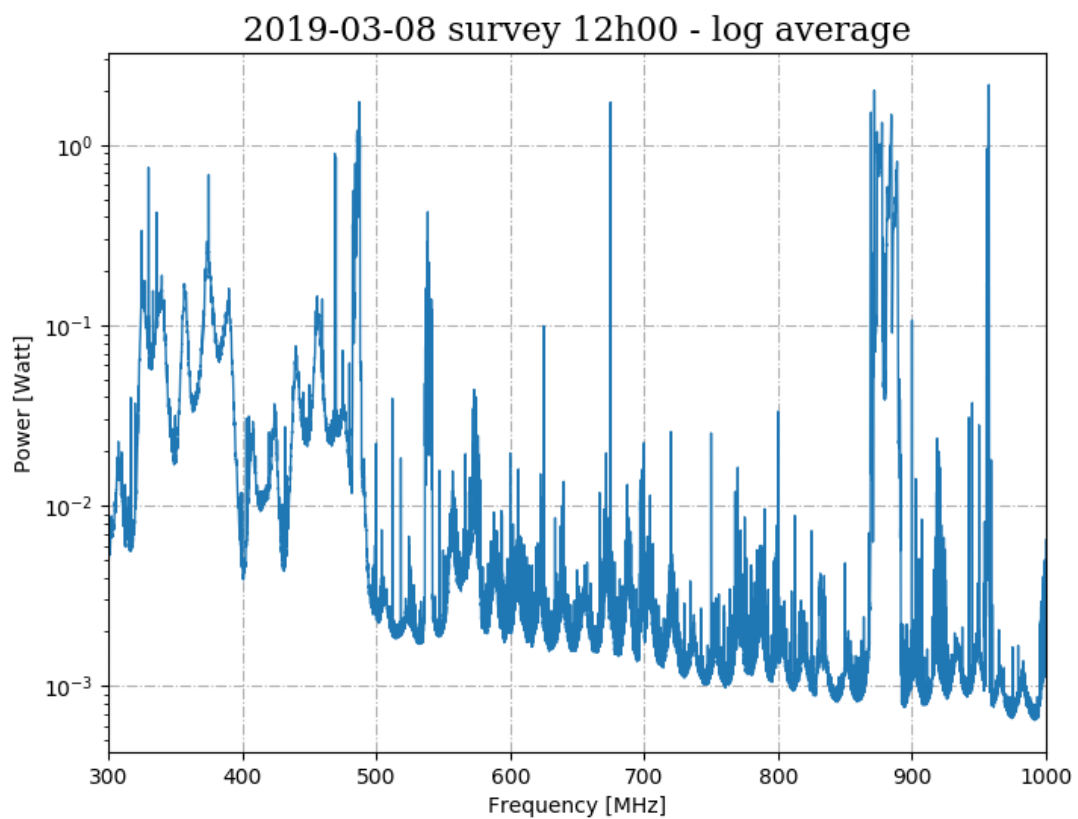


Figura 22 – Potência rms média em dB - 12h00 às 13h00 de 08/Mar/2019

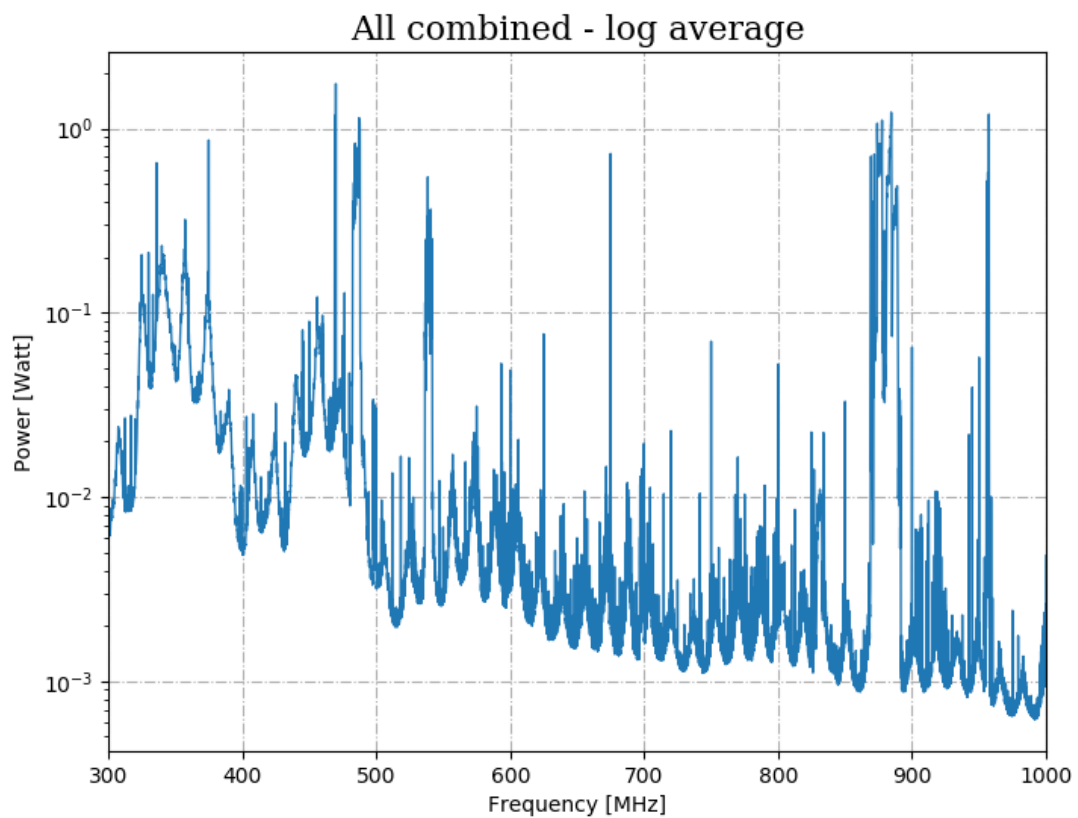


Figura 23 – Potência rms média em dB- Todas as leituras

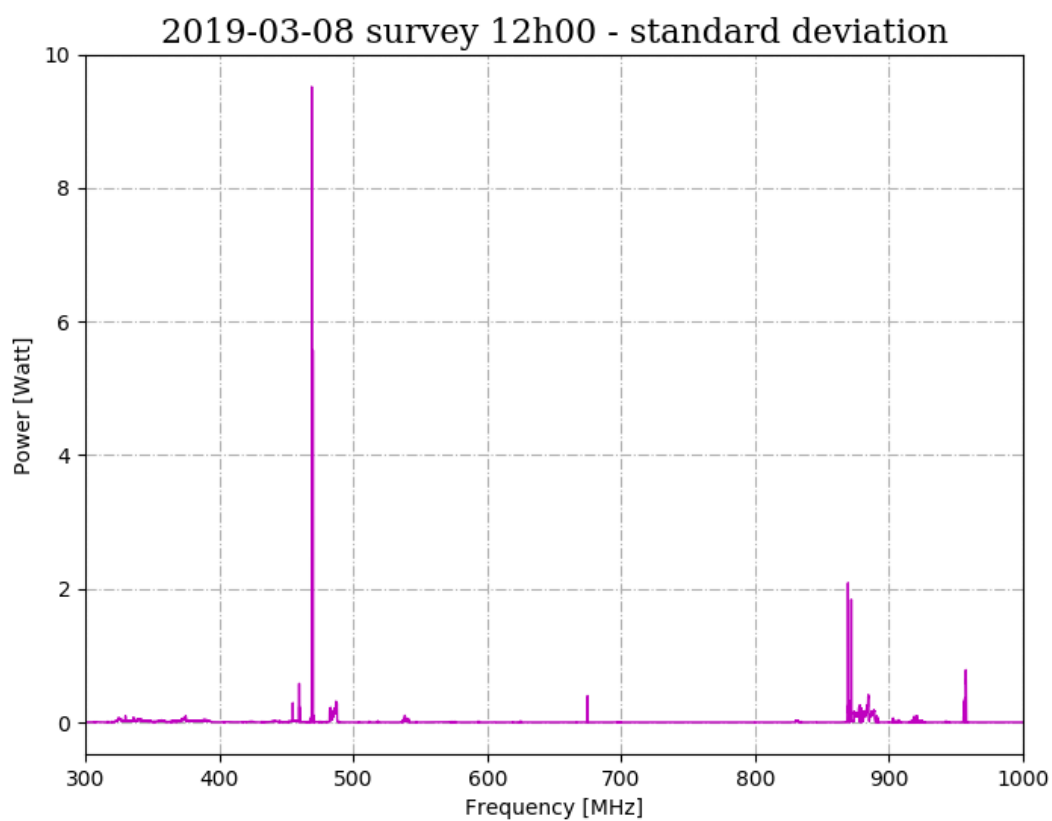


Figura 24 – Desvio padrão da média rms - 12h00 às 13h00 de 08/Mar/2019

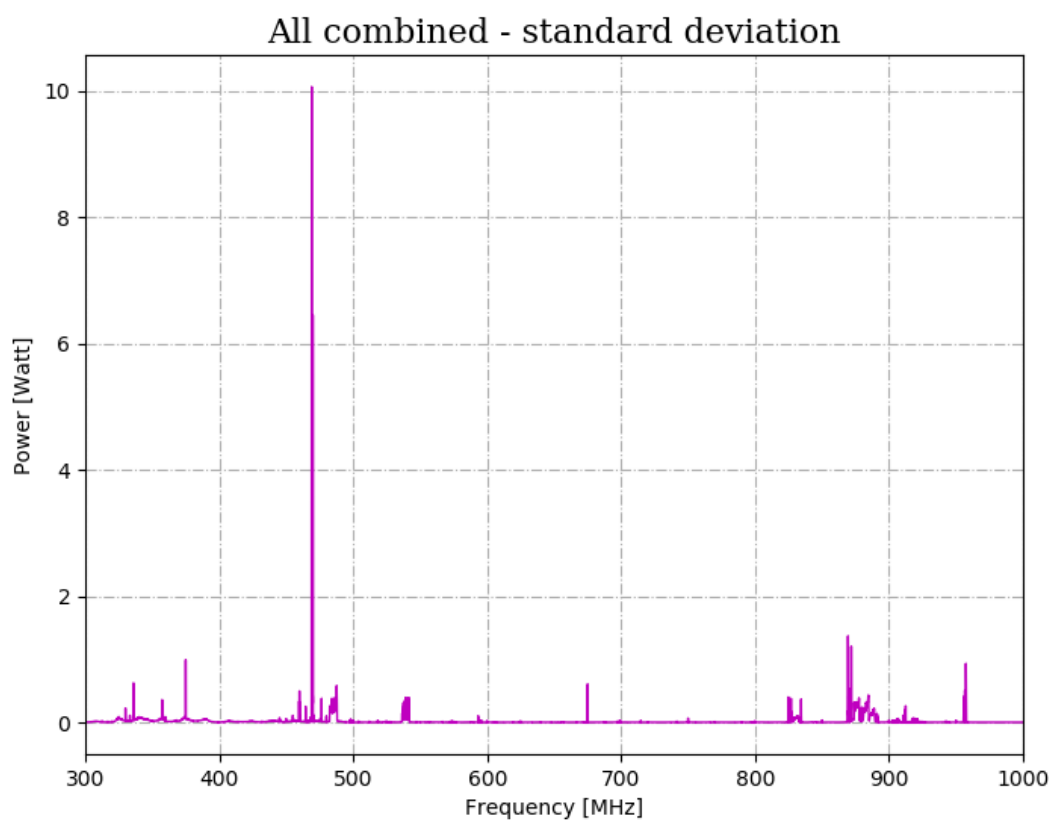


Figura 25 – Desvio padrão da média rms - Todas as leituras

2019-03-08 survey 12h00 - standard deviation to average ratio

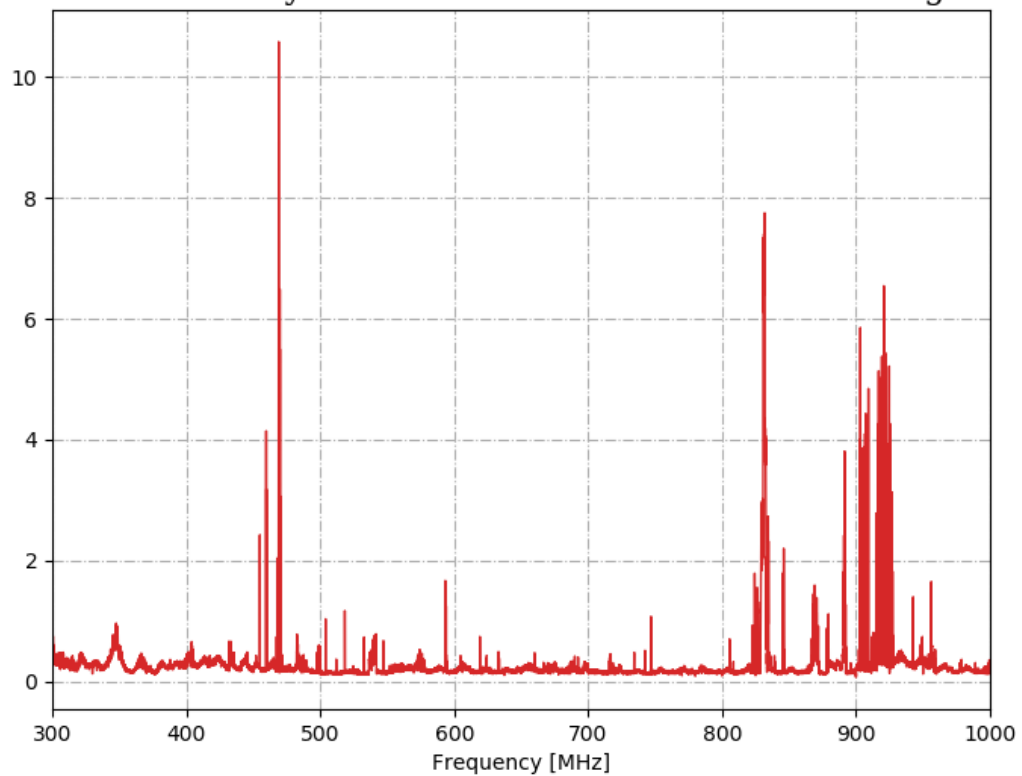


Figura 26 – Razão entre desvio e média - 12h00 às 13h00 de 08/Mar/2019

All combined - standard deviation to average ratio

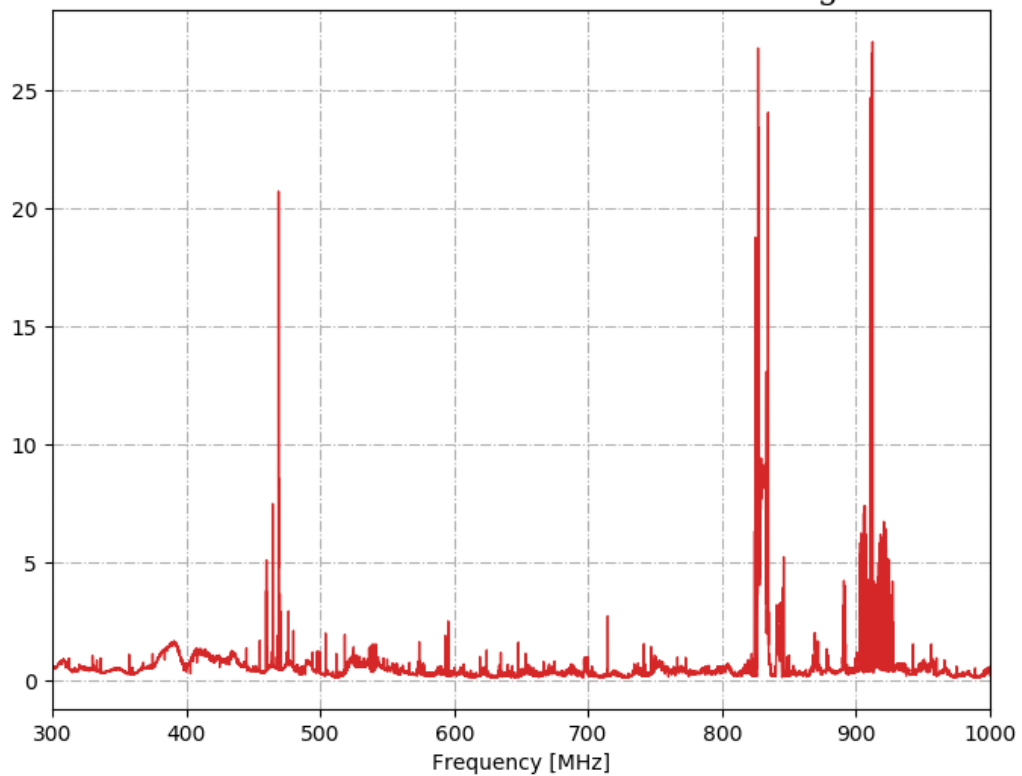


Figura 27 – Razão entre desvio e média - Todas as leituras

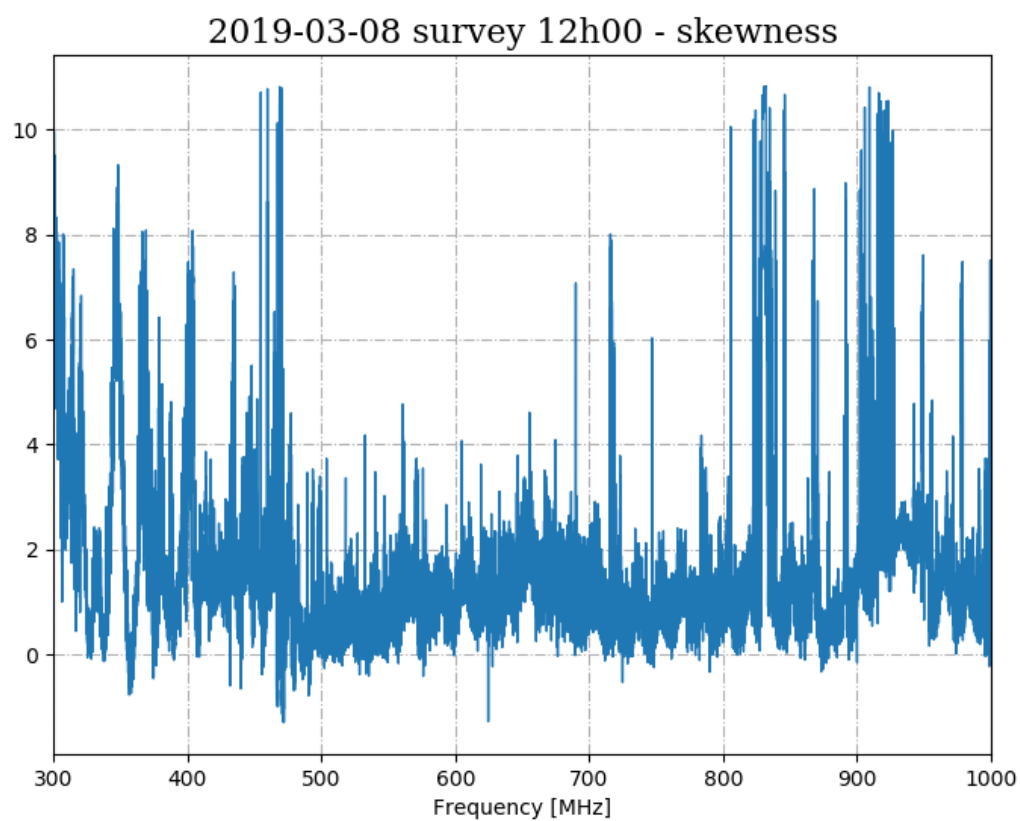


Figura 28 – Assimetria - 12h00 às 13h00 de 08/Mar/2019

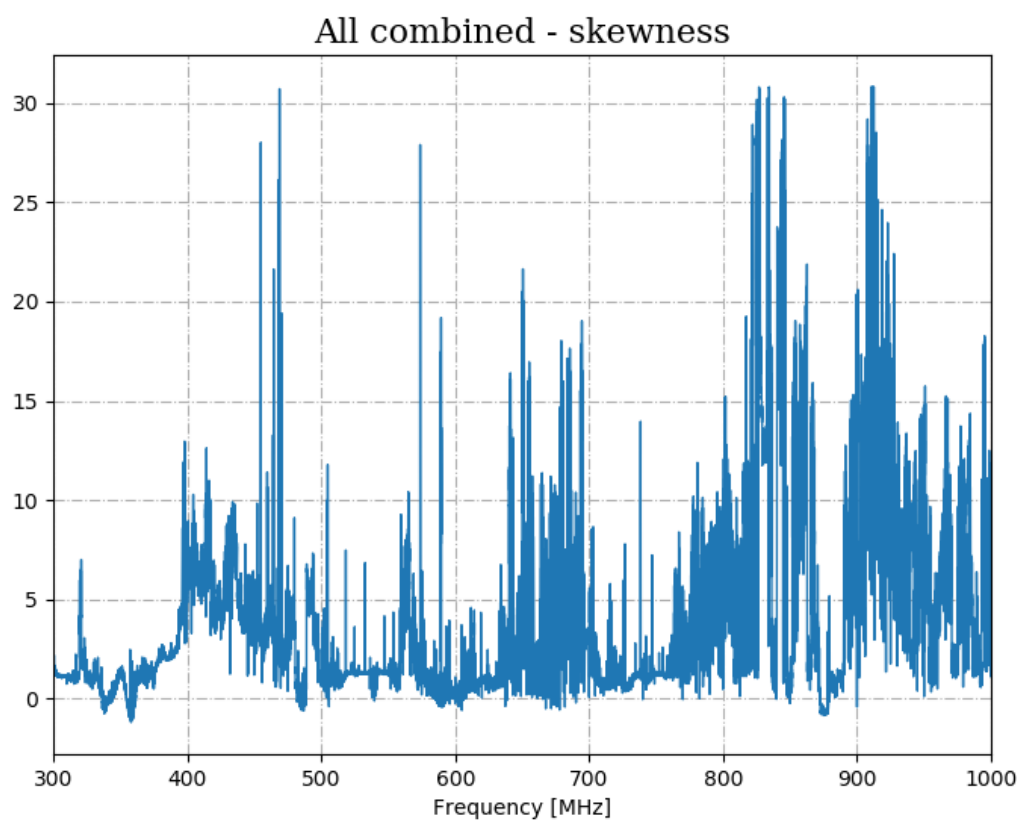


Figura 29 – Assimetria - Todas as leituras

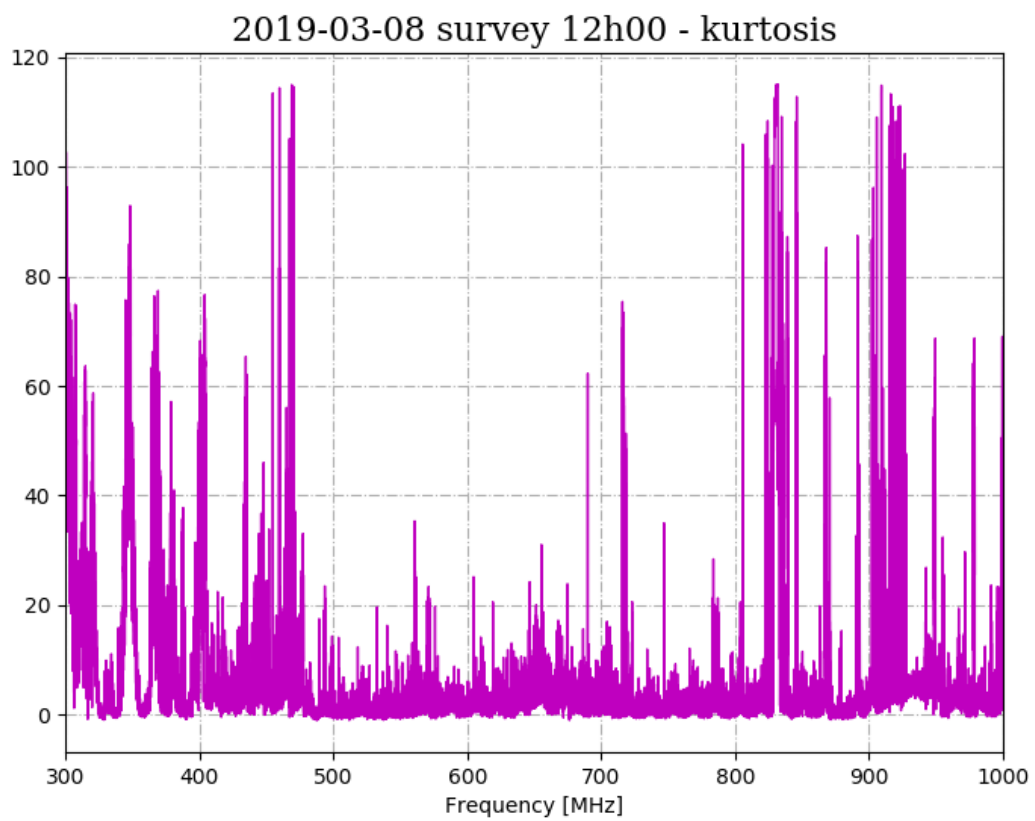


Figura 30 – Curtose - 12h00 às 13h00 de 08/Mar/2019

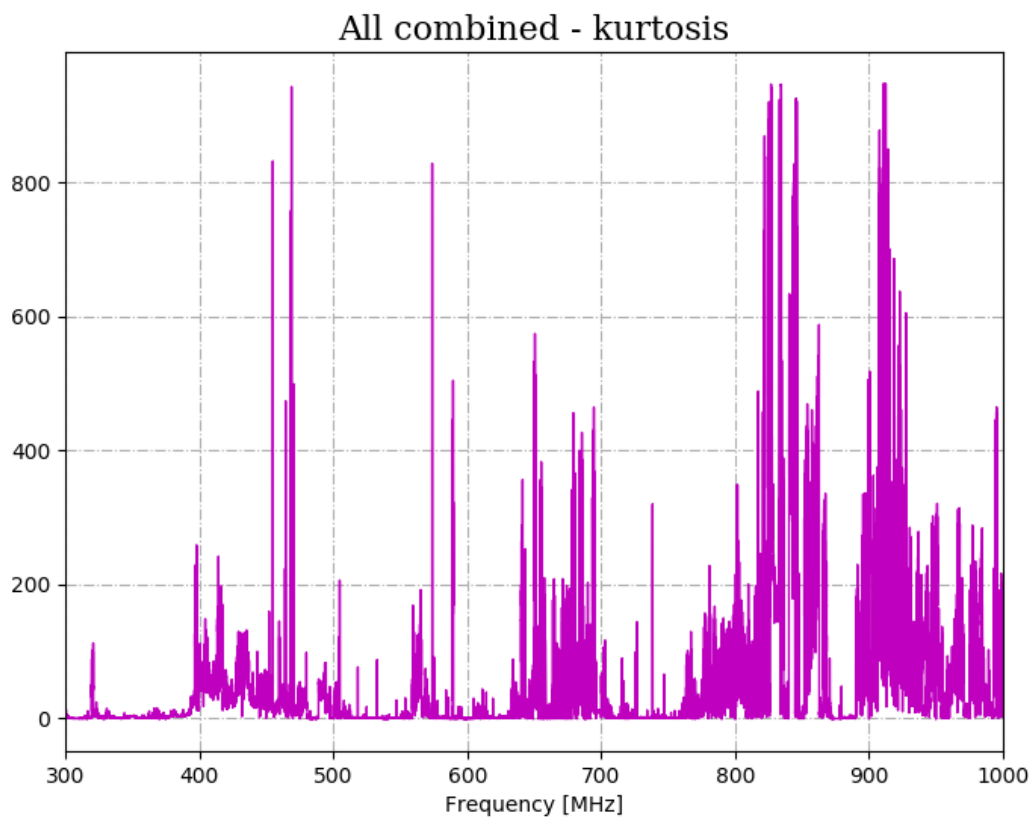


Figura 31 – Curtose - Todas as leituras

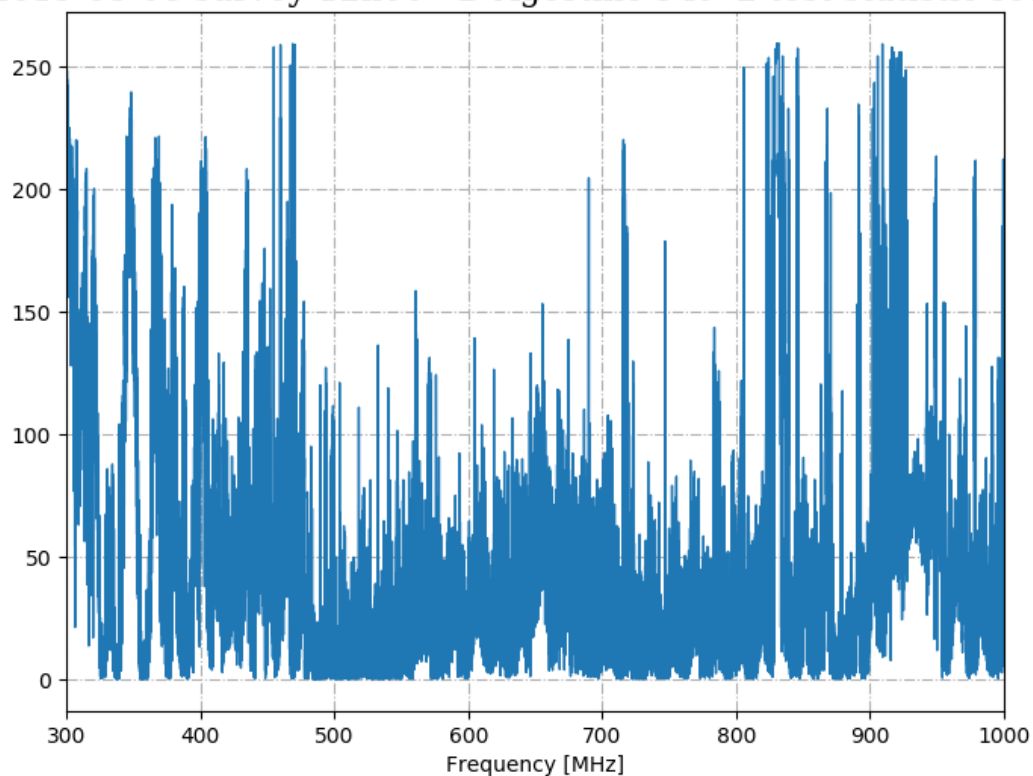
2019-03-08 survey 12h00 - D'Agostino's K^2 test statistic score

Figura 32 – Poder do teste de normalidade - 12h00 às 13h00 de 08/Mar/2019

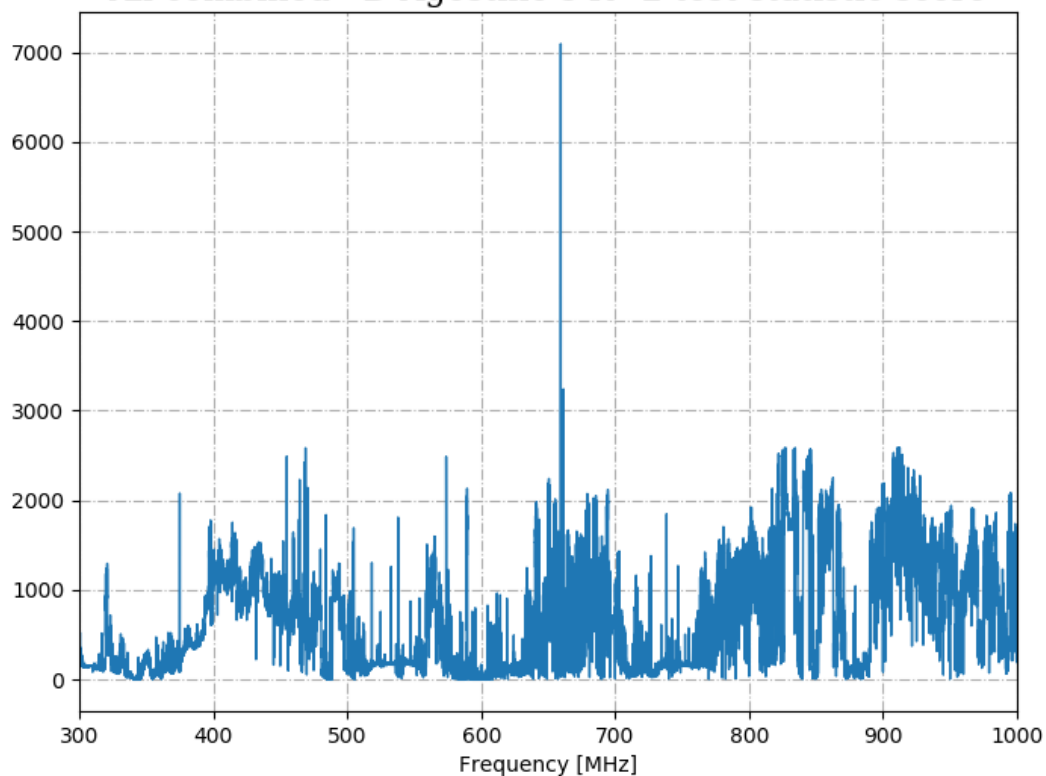
All combined - D'Agostino's K^2 test statistic score

Figura 33 – Poder do teste de normalidade - Todas as leituras

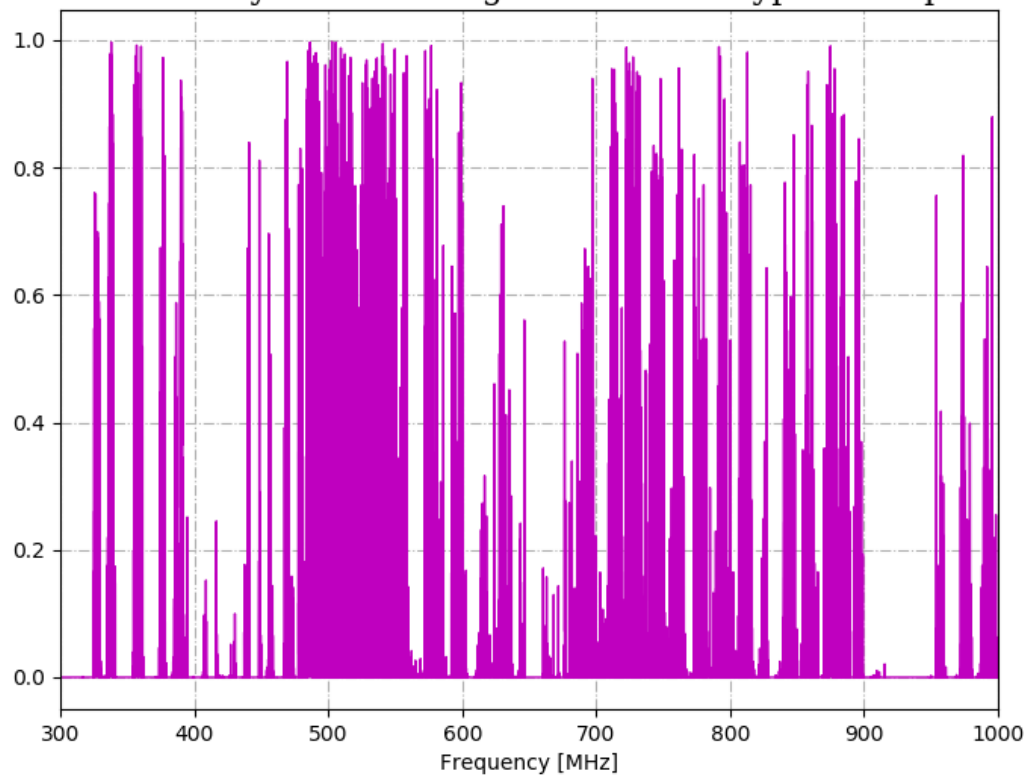
2019-03-08 survey 12h00 - D'Agostino's K^2 hypothesis p-value (p)

Figura 34 – Probabilidade H0 do teste - 12h00 às 13h00 de 08/Mar/2019

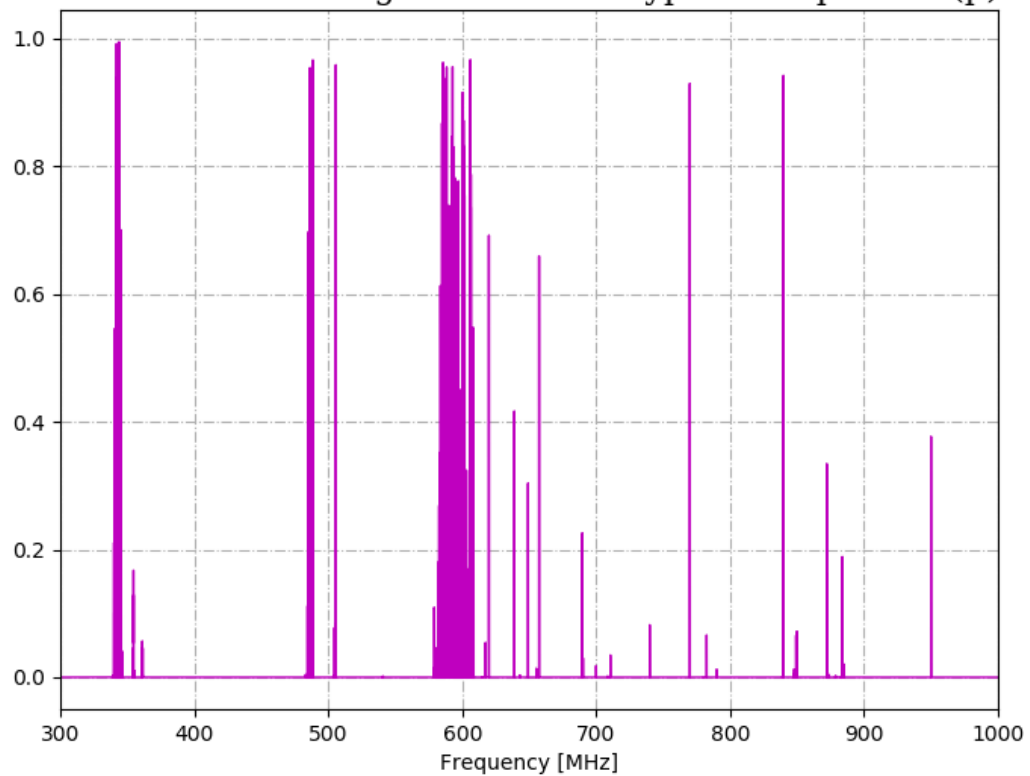
All combined - D'Agostino's K^2 hypothesis p-value (p)

Figura 35 – Probabilidade H0 do teste - Todas as leituras

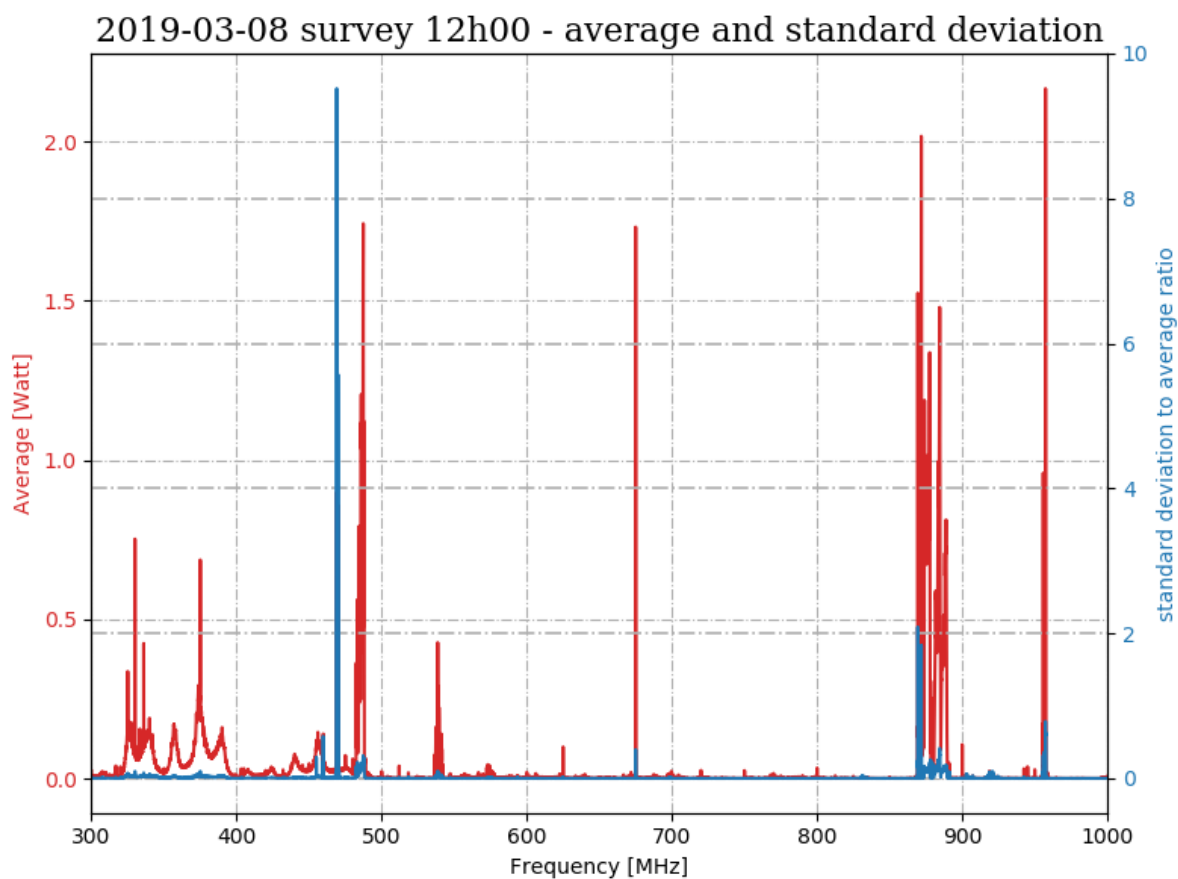


Figura 36 – Confronto entre média e desvio padrão - 12h00 às 13h00 de 08/Mar/2019

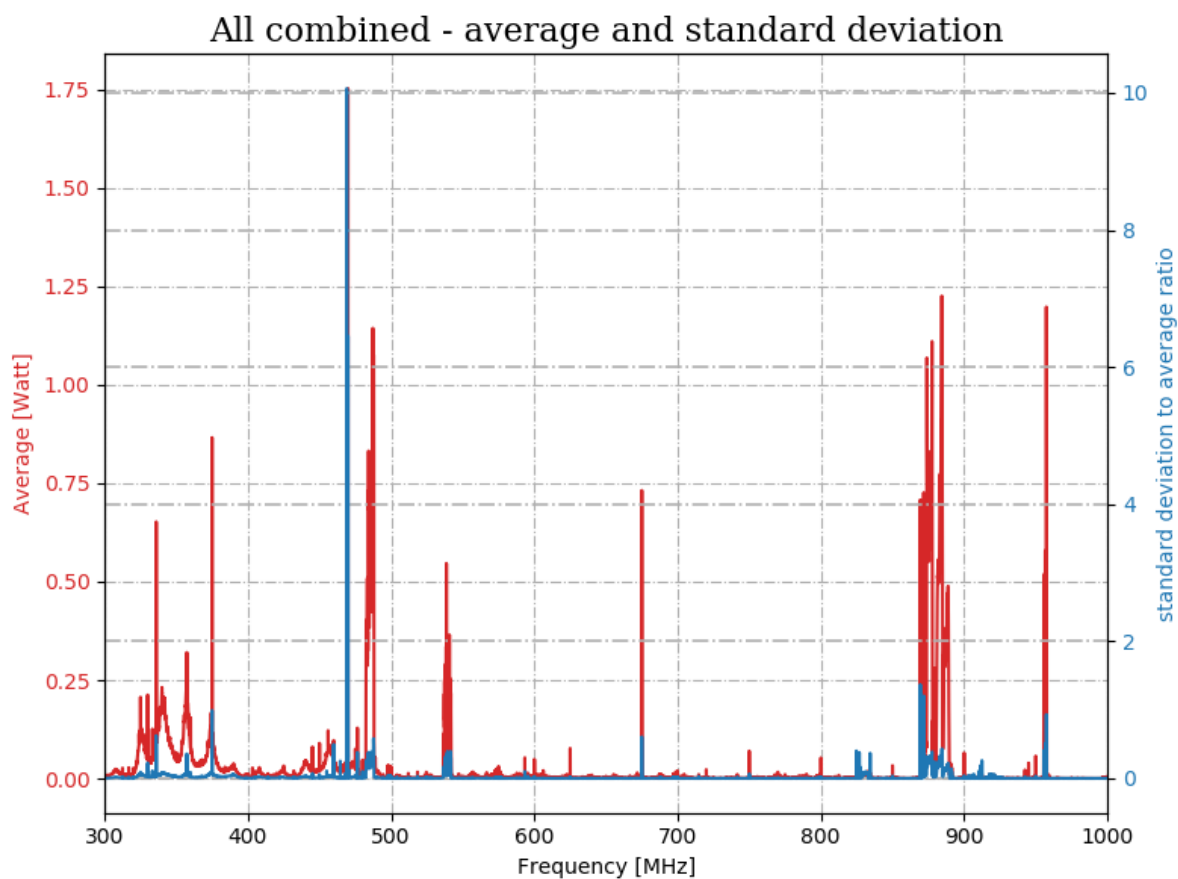


Figura 37 – Confronto entre média e desvio padrão - Todas as leituras

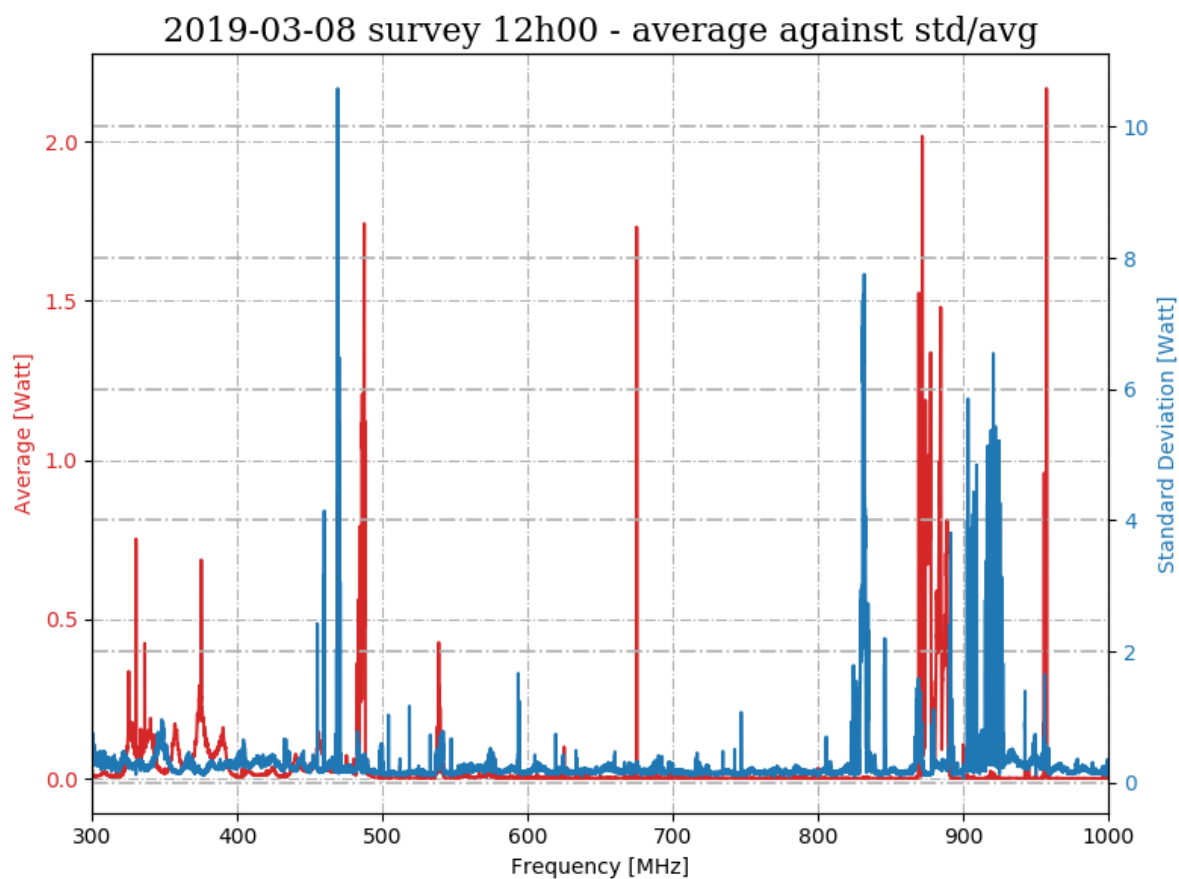


Figura 38 – Confronto entre média e a razão desvio/média - 12h00 às 13h00 de 08/Mar/2019

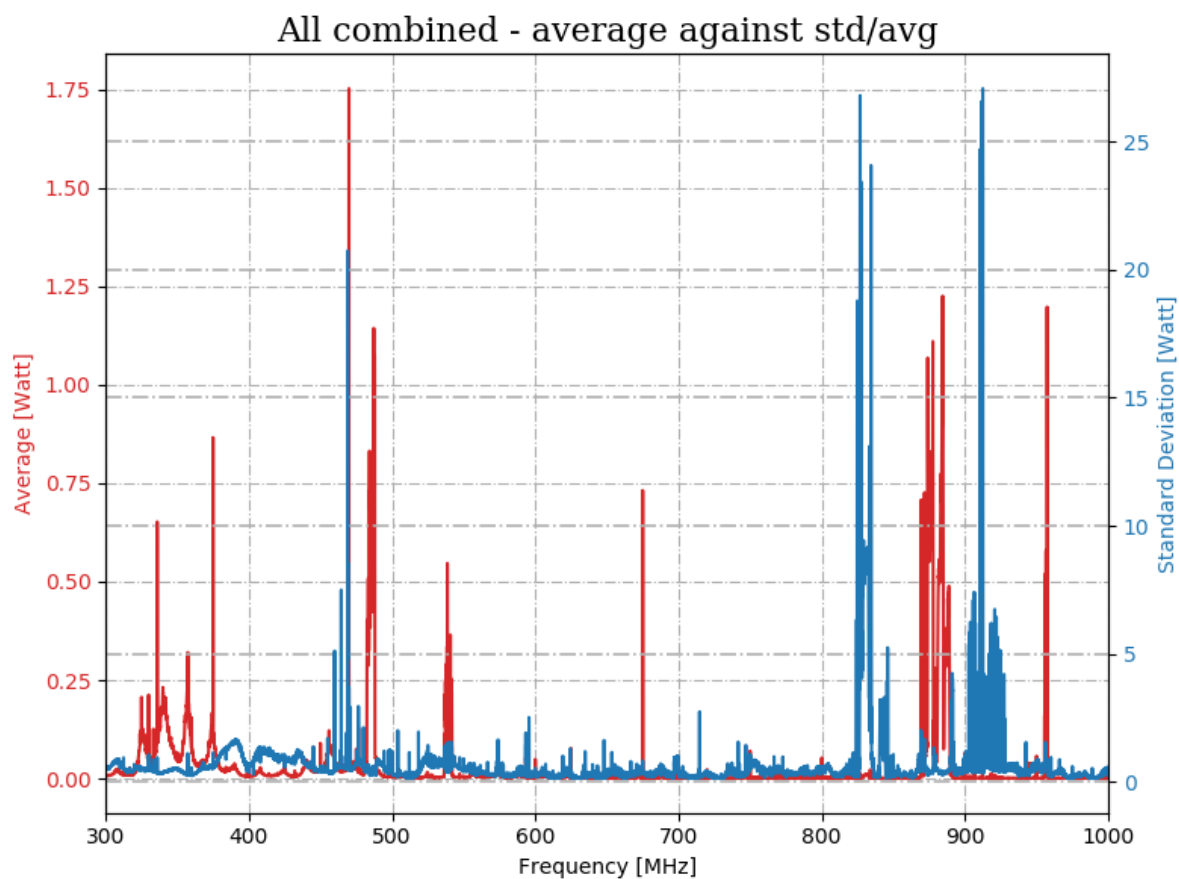


Figura 39 – Confronto entre média e a razão desvio/média - Todas as leituras

2019-03-08 survey 12h00 - Normality test against average power

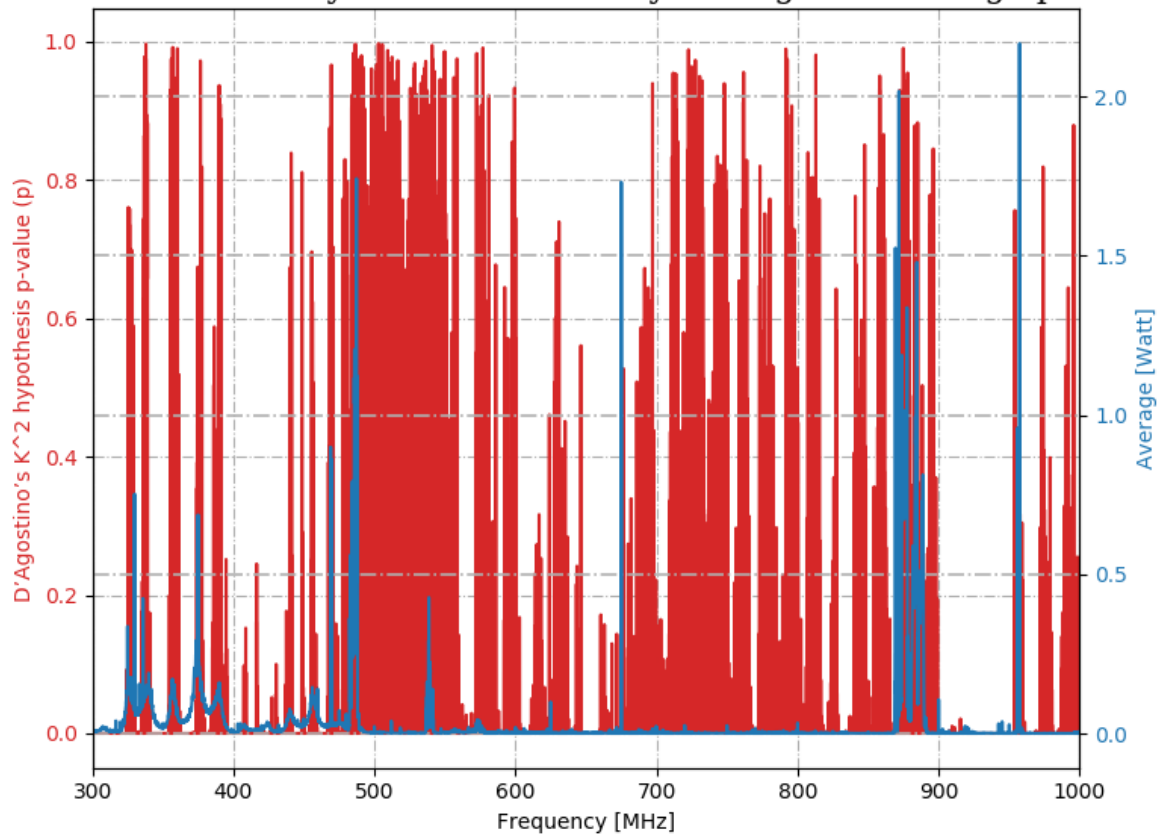


Figura 40 – Confronto entre prob. H0 do teste e média - 12h00 às 13h00 de 08/Mar/2019

All combined - Normality test against average power

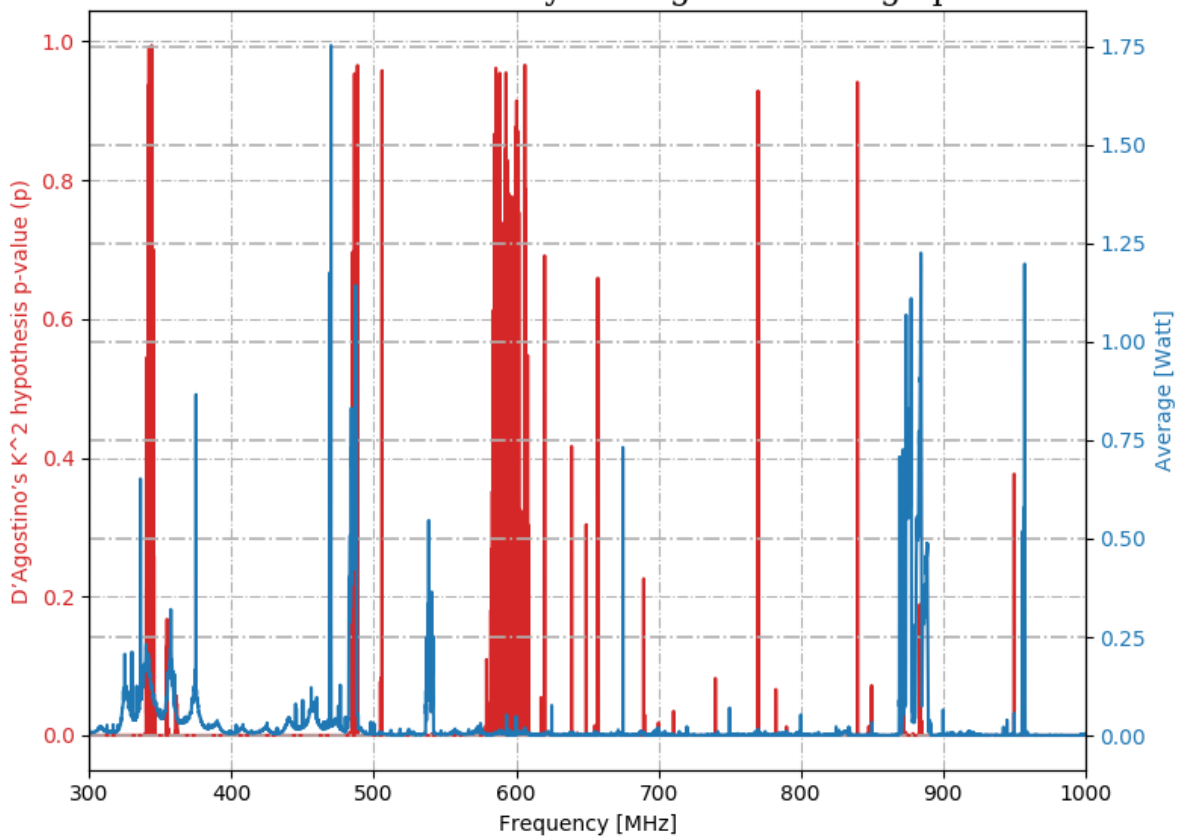


Figura 41 – Confronto entre prob. H0 do teste e média- Todas as leituras

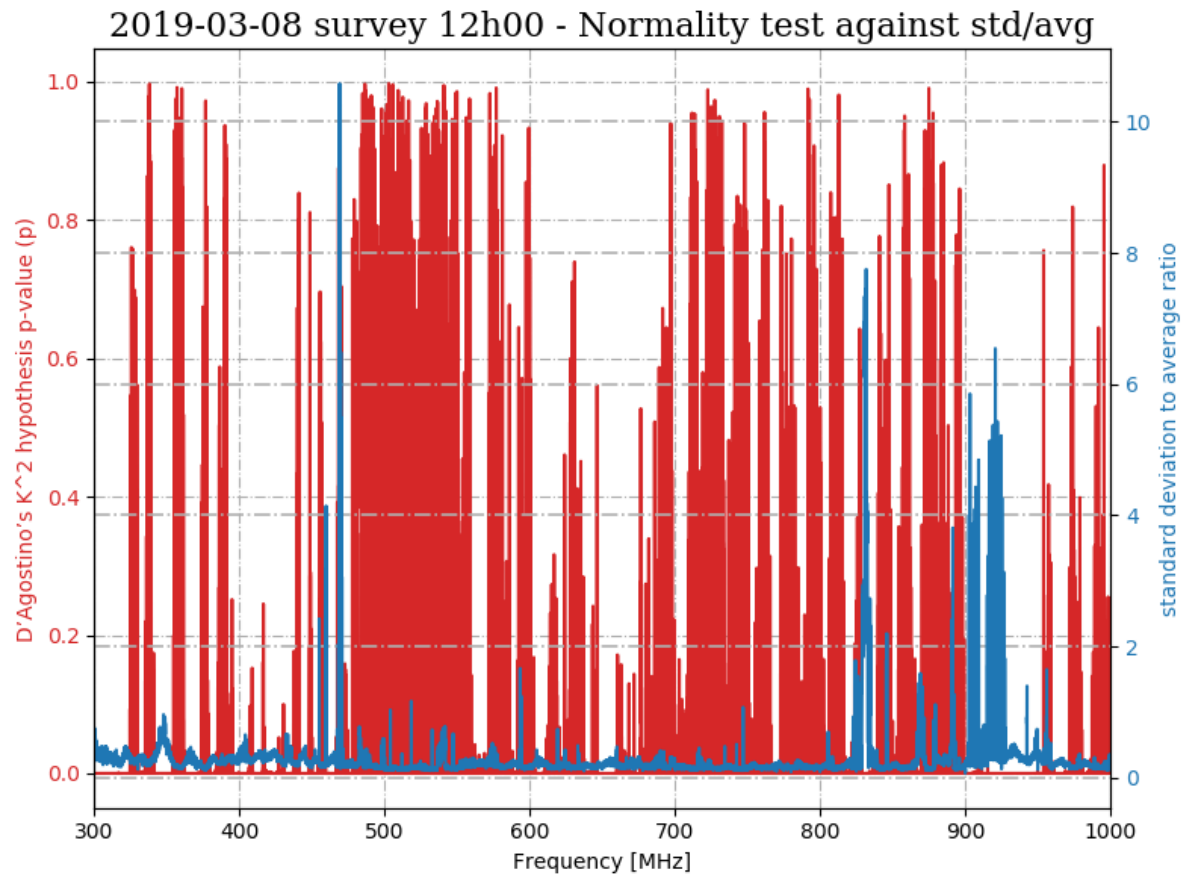


Figura 42 – Confronto prob. H0 do teste e a razão desvio/média - 12h00 às 13h00 de 08/Mar/2019

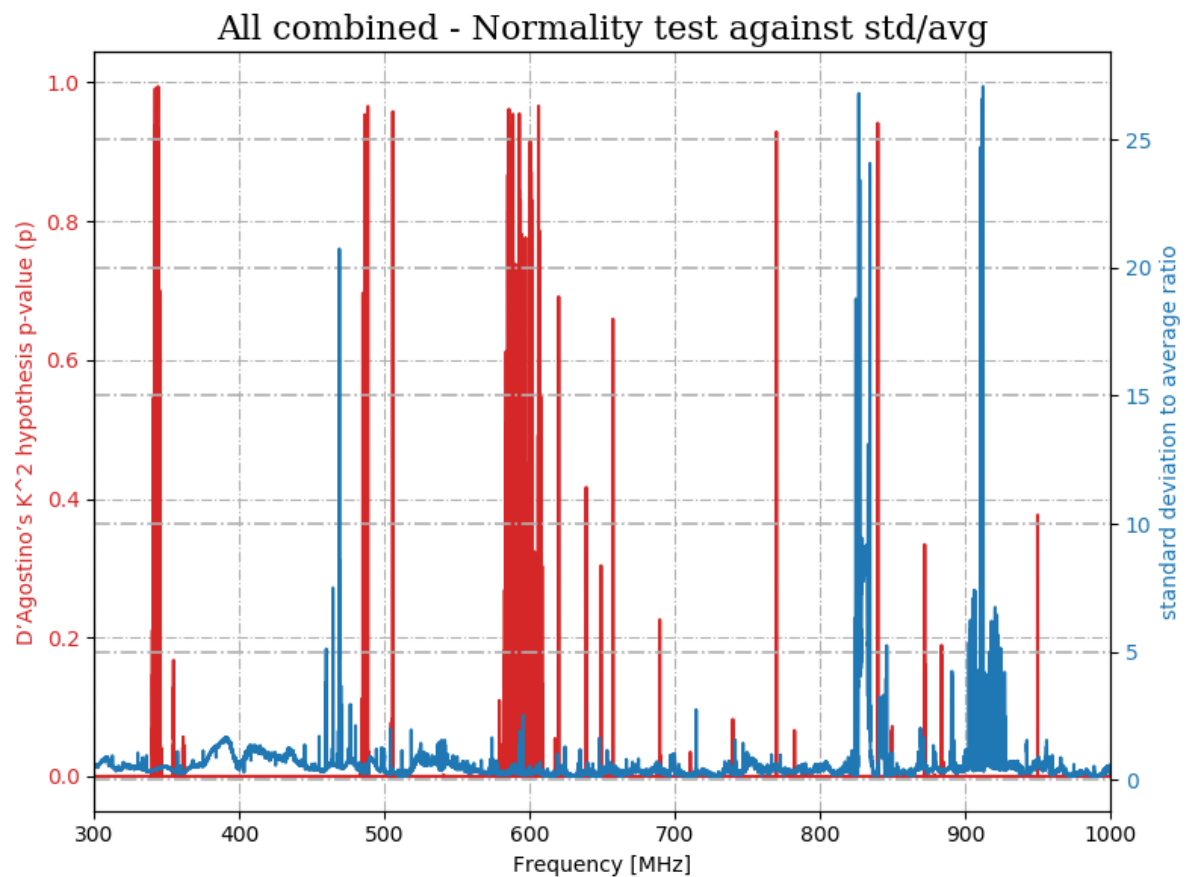


Figura 43 – Confronto prob. H0 do teste e a razão desvio/média - Todas as leituras

5 CONCLUSÕES

O estudo de distribuição de frequências deste trabalho, é uma alternativa simples para reconhecimento e coleta de informações de sinais de rádio frequência, com o estudo focado em análise de espectro de banda larga. Considerando que o SDR também pode ser utilizado por uma *rapsberry pi* com sistema operacional Linux, rodando os mesmos programas, é possível criar uma alternativa de ainda mais baixo custo para este tipo de estudo, sendo um ponto bastante atrativo para continuar o desenvolvimento do *software*. Os indicadores levantados, no que diz respeito ao *software* desenvolvido em relação a cálculos, utilizam somente ferramenta estatísticas de código aberto, não dependem de outros *hardwares* para tratamento de sinal ou fontes de dados externos, sendo possível extrair informações do espectro através da análise da distribuição dos dados.

A hipótese de que as diferentes leituras possuem similaridades de distribuição de espectro é congruente com os indicadores mais simples como o média e desvio padrão, entretanto os indicadores de forma da distribuição, e principalmente o teste de normalidade, não evidenciam conformidade com a hipótese principal, necessitando de uma análise individual para cada caso, pois estes últimos resultados não afirmam que os canais ruidosos das diferentes leituras terão comportamento similar. Uma possível justificativa sendo as diferenças entre as leituras de baixas potências, que por si só abre a possibilidade de realização de outro trabalho. No caso, o maior desafio do trabalho é distinguir ruído de transmissões de baixa potência, que pode ser contornado levantando-se experimentalmente a curva de V.S.W.R. da antena, entretanto isto representa um custo adicional ao *setup* do trabalho.

Por fim, mesmo com uma aparelhagem simples, e de baixo custo para a realização de atividades correlacionadas à inteligência de sinal, os resultados obtidos são expressivos com relação transmissões de maior potência, que são facilmente mapeadas e distinguidas entre intermitentes e perenes, e intrigantes com relação às transmissões de baixa potência, sendo um estimulante para a continuação do trabalho.

5.1 Proposta de continuação

Como próximo passo, será construtivo levantar uma base de dados maior e explorar as técnicas como as de aprendizado de máquinas e sua implementação, utilizando as relações dos indicadores baseados puramente em dados de potências. A proposta é o desenvolvimento de um classificador que permita a distinção entre canais sempre ociosos, com transmissões perenes e transmissões intermitentes, rotulando assim todo o espectro de frequência. Um software deste nível será muito útil para não só mapear como os canais se comportam, como também para encontrar anomalias e uso indevido do recurso finito que é o espectro de frequência.

REFERÊNCIAS

ADS-B. *Automatic Dependent Surveillance – Broadcast*. [S.l.], 2019. Disponível em: https://en.wikipedia.org/wiki/Automatic_dependent_surveillance_%E2%80%93_broadcast. Acesso em: 01 Outubro 2018.

ANATEL. *PLANO DE ATRIBUIÇÃO, DESTINAÇÃO E DISTRIBUIÇÃO DE FREQUÊNCIAS NO BRASIL*. [S.l.], 2018. Disponível em: <http://www.anatel.gov.br/setorregulado/component/content/article/181-radiofrequencia/atribuicao-destinacao-e-distribuicao-de-faixas/226-plano-de-atribuicao-distribuicao-e-destinacao-de-radiofrequencias>. Acesso em: 01 Fevereiro 2019.

ANTENNA (radio). In: WIKIPÉDIA: a enciclopédia livre. Wikimedia, 2019. Disponível em: [https://en.wikipedia.org/wiki/Antenna_\(radio\)](https://en.wikipedia.org/wiki/Antenna_(radio)). Acesso em: 01 Julho 2019.

ANTENNA Theory. [S.l.]. Disponível em: <http://www.antenna-theory.com/>, Urlaccessdate={01Julho2019}, Author={PeterJosephBevelacqua}, Year={ }.>

BALANIS, C. A. *Antenna theory: analysis and design*. [S.l.: s.n.], 2012.

BROWNLEE, J. *A Gentle Introduction to Normality Tests in Python*. [S.l.], 2018. Disponível em: <https://machinelearningmastery.com/a-gentle-introduction-to-normality-tests-in-python/>. Acesso em: 01 Maio 2019.

COMPRIMENTO de Onda. In: WIKIPÉDIA: a enciclopédia livre. Wikimedia, 2019. Disponível em: https://pt.wikipedia.org/wiki/Comprimento_de_onda. Acesso em: 01 Julho 2019.

CST. *Wire Dipole Antenna | CST example*. [S.l.], 2019. Disponível em: <http://cst2cm.acceptance.brimit.com/-/media/cst/academia/examples/wire-dipole-antenna/wire-antenna-s-parameter.ashx>. Acesso em: 01 Julho 2019.

D'AGOSTINO'S K-squared test. In: WIKIPÉDIA: a enciclopédia livre. Wikimedia, 2019. Disponível em: https://en.wikipedia.org/wiki/D%27Agostino%27s_K-squared_test. Acesso em: 01 Maio 2019.

DIFRAÇÃO. In: WIKIPÉDIA: a enciclopédia livre. Wikimedia, 2019. Disponível em: <https://pt.wikipedia.org/wiki/Difra%C3%A7%C3%A3o>. Acesso em: 01 Julho 2019.

FERREIRA, J. *Intro to RTL-SDR, Part I - Principles and Hardware*. [S.l.], 2017. Disponível em: <http://ajoo.blog/intro-to-rtl-sdr-part-i-principles-and-hardware.html>. Acesso em: 01 Janeiro 2019.

GNU Radio. [S.l.], 2019. Disponível em: <https://wiki.gnuradio.org>. Acesso em: 01 Setembro 2018.

KEEN, K. *Rtl Power Basic scripting*. [S.l.], 2017. Disponível em: <http://kmkeen.com/rtl-power/>. Acesso em: 01 Novembro 2019.

LIBREOFFICE. *Frequently asked questions - Calc*. [S.l.], 2019. Disponível em: <https://wiki.documentfoundation.org/Faq/Calc/022>. Acesso em: 03 Março 2019.

MICROSOFT. *Especificações e limites do Microsoft Excel*. [S.l.], 2019. Disponível em: <<https://support.office.com/pt-br/article/especifica%C3%A7%C3%B5es-e-limites-do-microsoft-excel-1672b34d-7043-467e-8e27-269d656771c3>>. Acesso em: 03 Março 2019.

NETO, J. da S. *Atenuação das radiações*. [S.l.], 2019. Disponível em: <<http://lief.if.ufrgs.br/~jader/atenuacao.pdf>>. Acesso em: 01 Julho 2019.

NORMALITY test. In: WIKIPÉDIA: a enciclopédia livre. Wikimedia, 2019. Disponível em: <https://en.wikipedia.org/wiki/Normality_test>. Acesso em: 01 Maio 2019.

RAYMAPS. *Some Common Antenna Radiation Patterns*. [S.l.], 2012. Disponível em: <<http://www.raymaps.com/index.php/some-common-antenna-radiation-patterns/>>. Acesso em: 01 Julho 2019.

RTL-SDR.COM. [S.l.], 2019. Disponível em: <<https://www.rtl-sdr.com/>>. Acesso em: 01 Setembro 2018.

SCIPY.STATS.NORMALTEST. [S.l.], 2019. Disponível em: <<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.normaltest.html>>. Acesso em: 01 Maio 2019.

SDR (Software Defined Radio) rtl-sdr. [S.l.], 2017. Disponível em: <<https://osmocom.org/projects/rtl-sdr/wiki>>. Acesso em: 01 Setembro 2018.

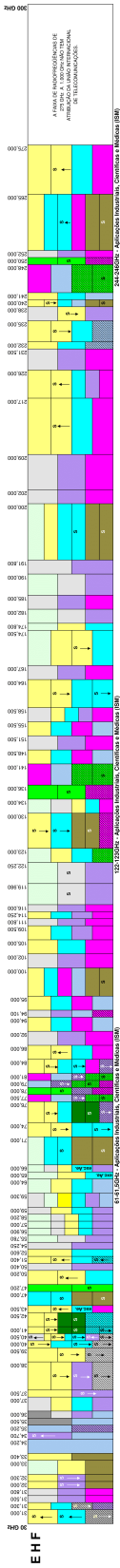
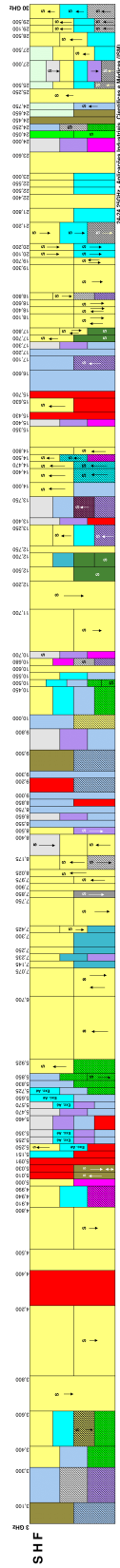
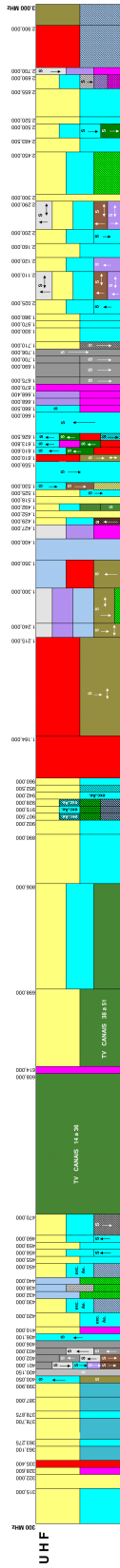
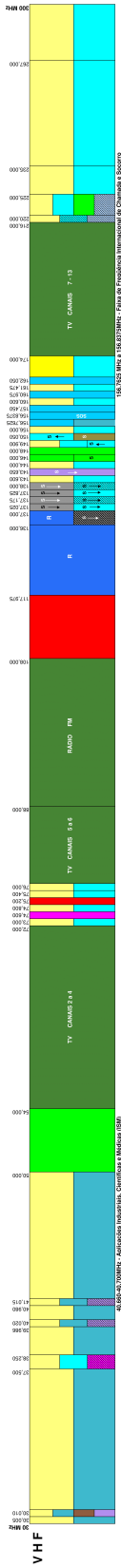
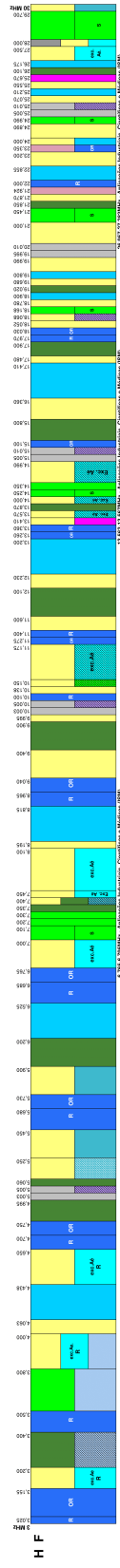
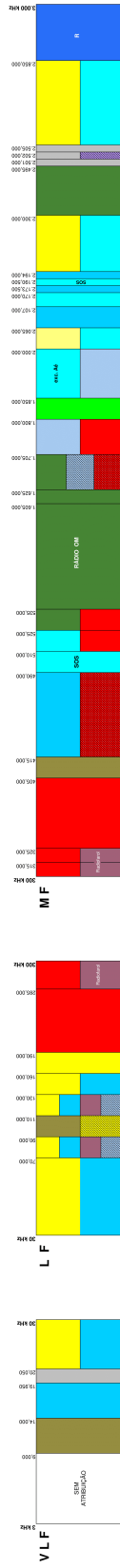
S&L-LAMBDA. *S&L Antenas*. [S.l.], 2019. Disponível em: <<http://www.slantenas.com.br/index.php/antenas-fixas/antena-fixa-direcional/item/74-antena-direcional-log-periodica-uhf-de-14-elementos>>. Acesso em: 01 Outubro 2018.

STUTZMAN, W. L.; THIELE, G. A. *Antena Theory and Design*. [S.l.: s.n.], 1997.

TELECO. *Frequências de Celular*. [S.l.], 2019. Disponível em: <<http://www.teleco.com.br/areasc.asp>>. Acesso em: 01 Fevereiro 2019.

6 ANEXOS

6.1 Plano de Atribuição de faixas e frequências no Brasil

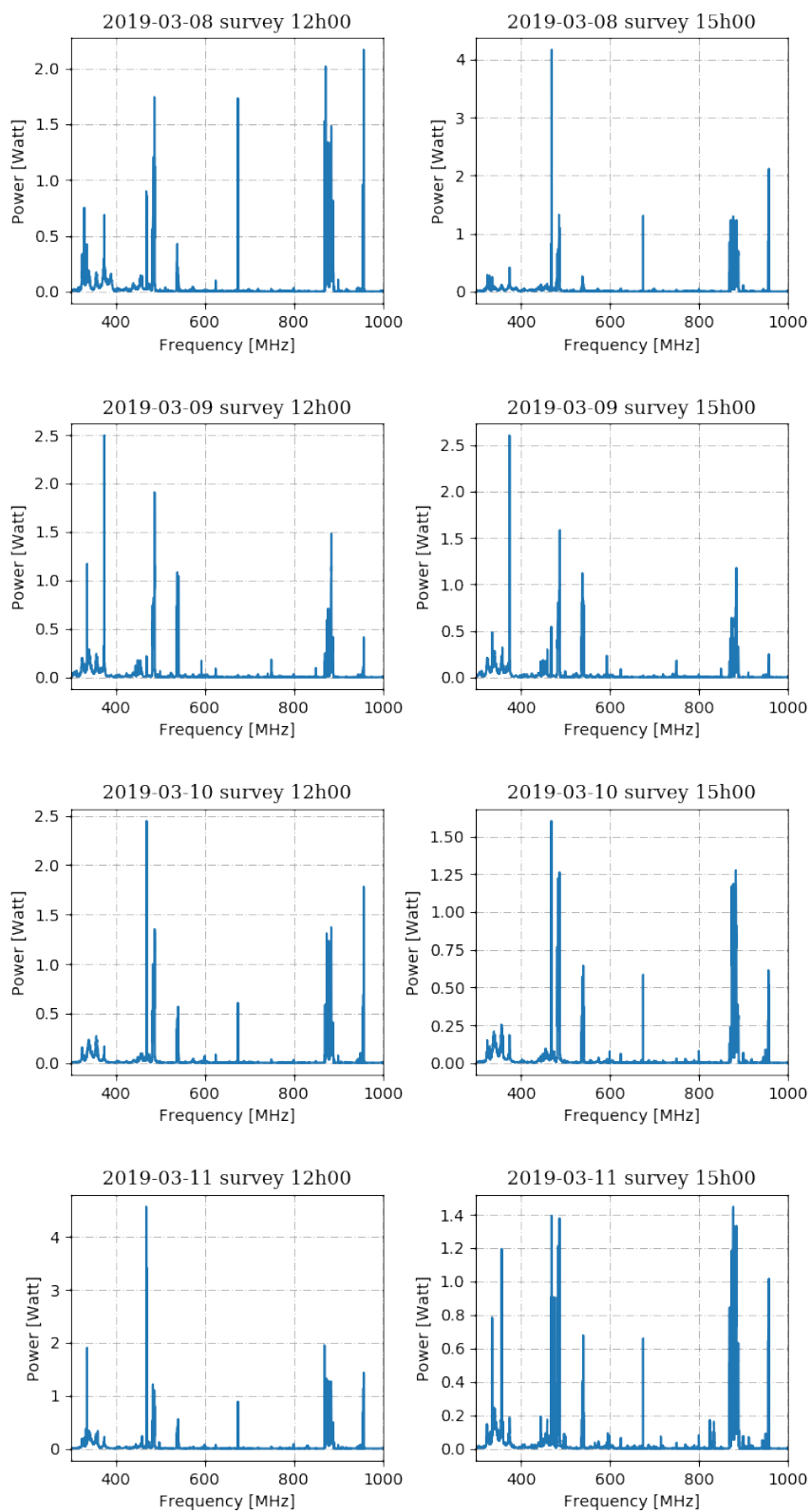
[illegible]

SOS SOCORRO E CHAMADA

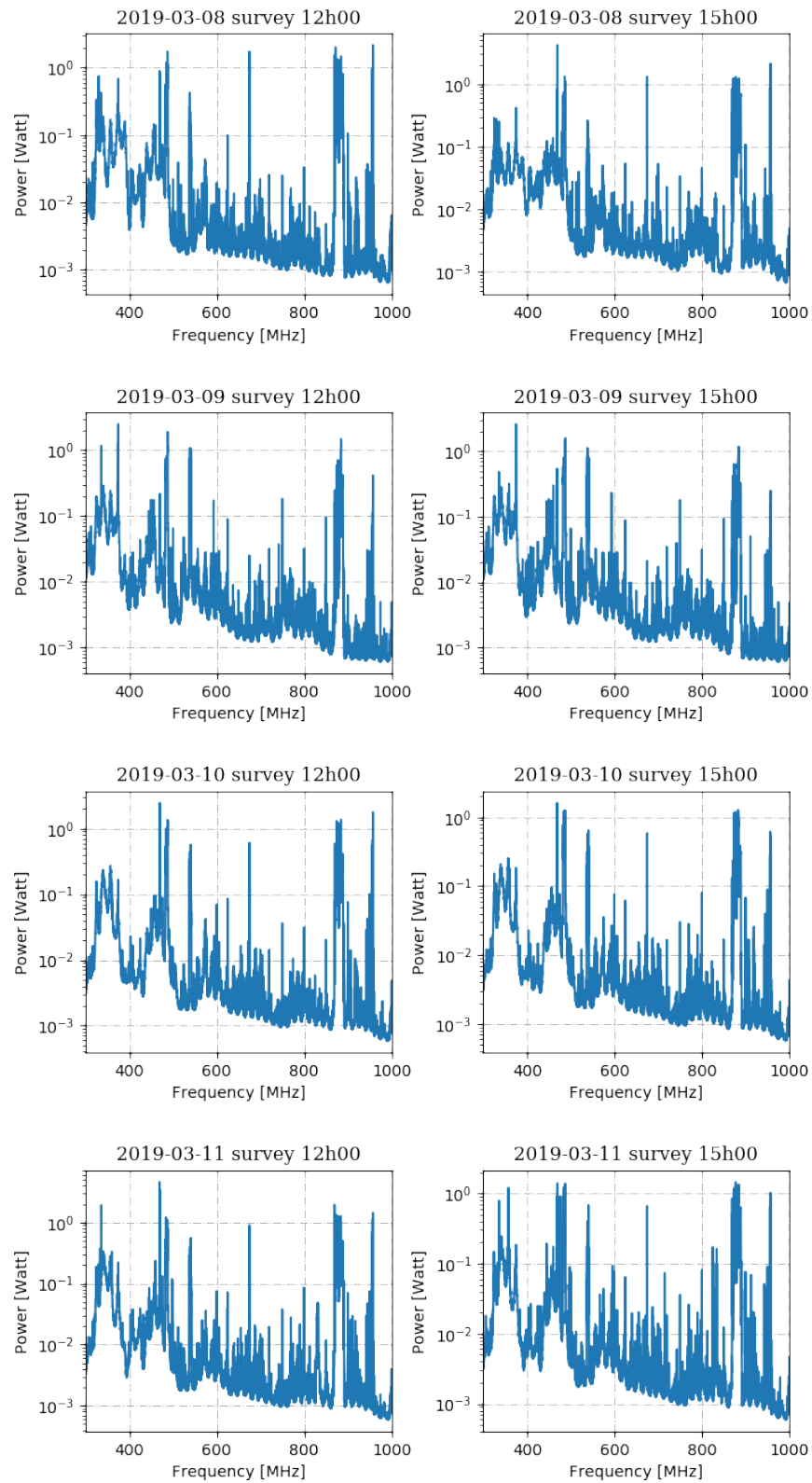
ATUALIZADO EM JANEIRO DE 2018

6.2 Indicadores de todos os dados coletados

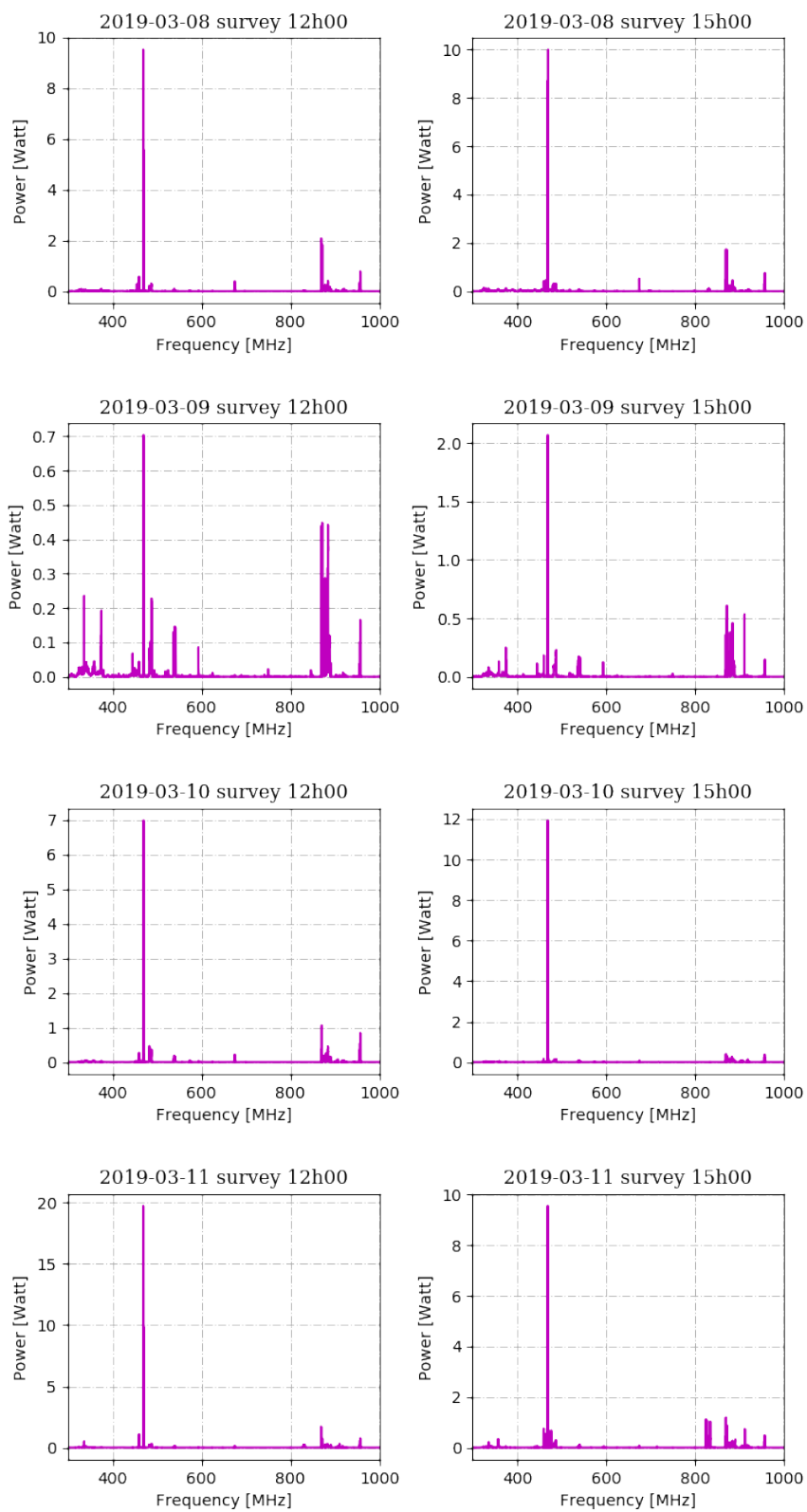
Potência rms média



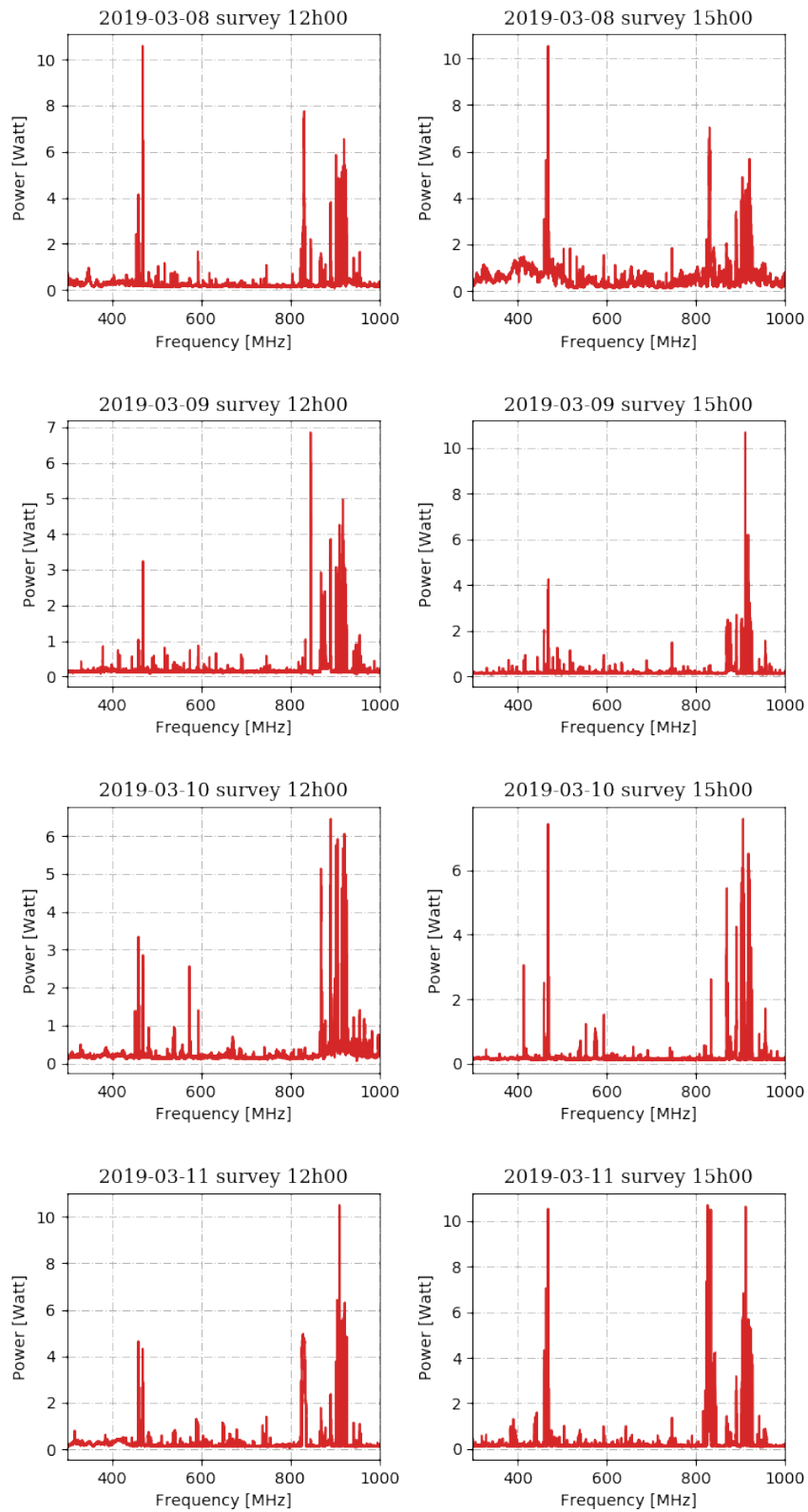
Log potência rms média



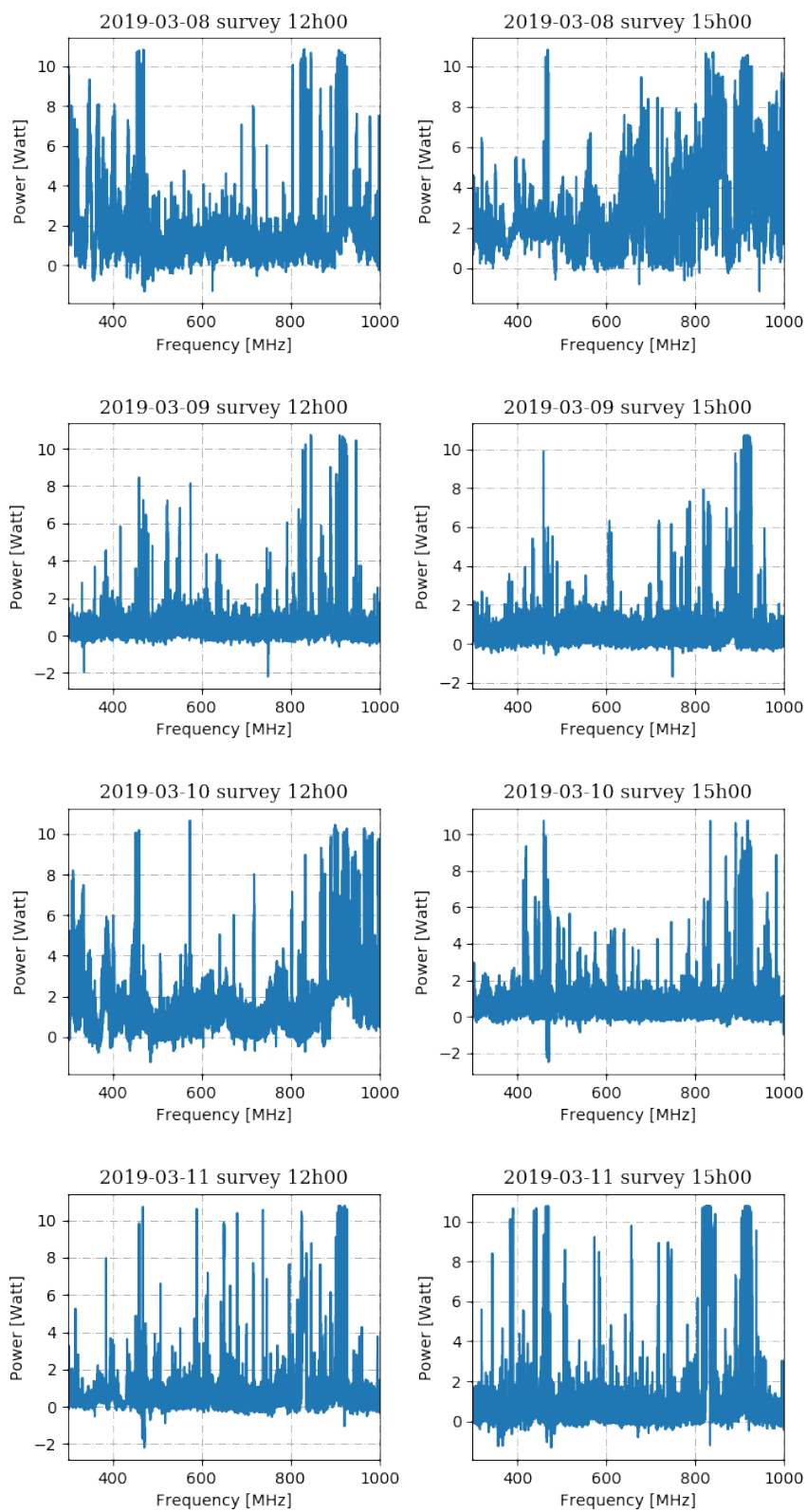
Desvio padrão da potência rms



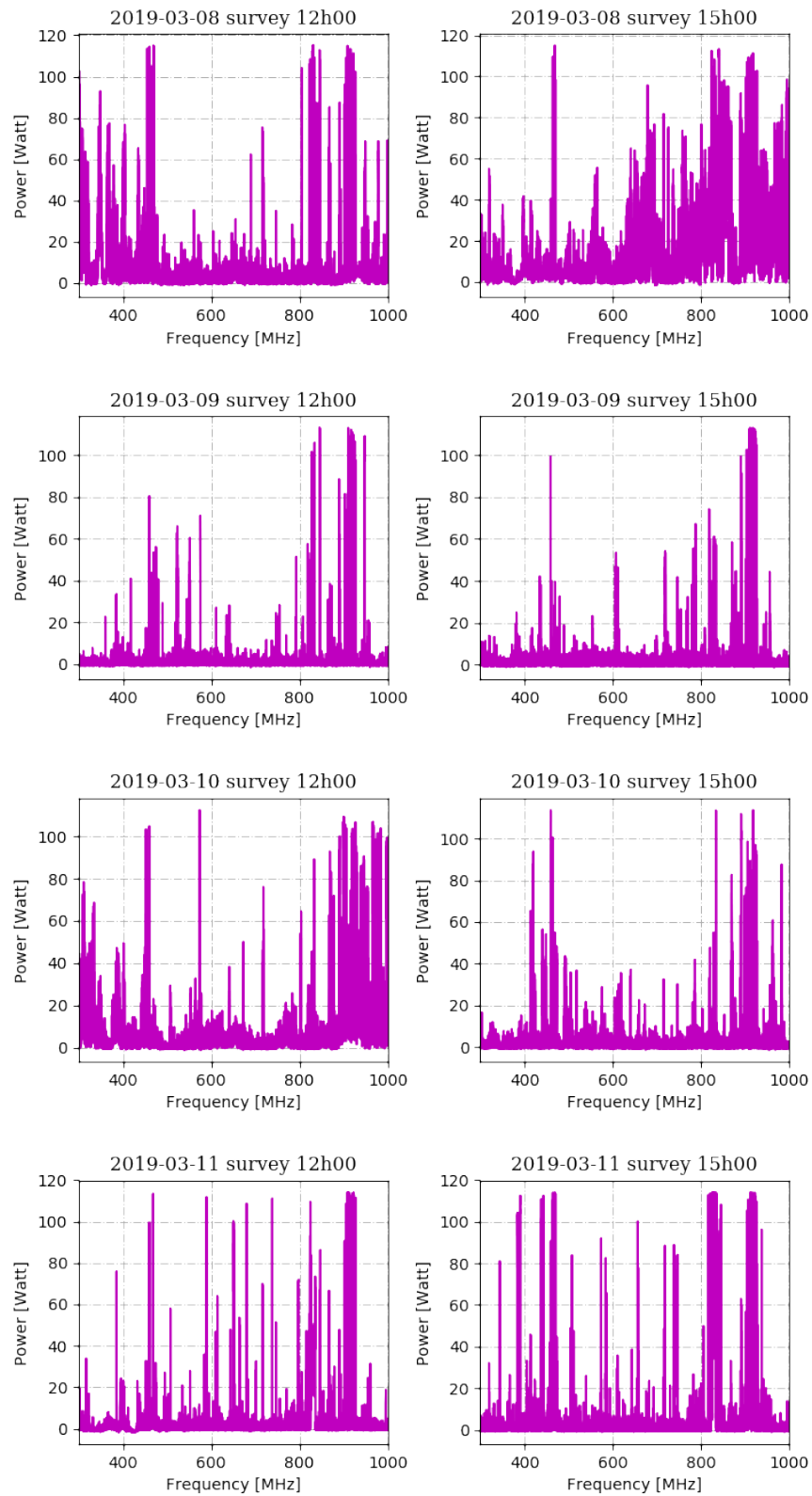
Razão entre desvio e média



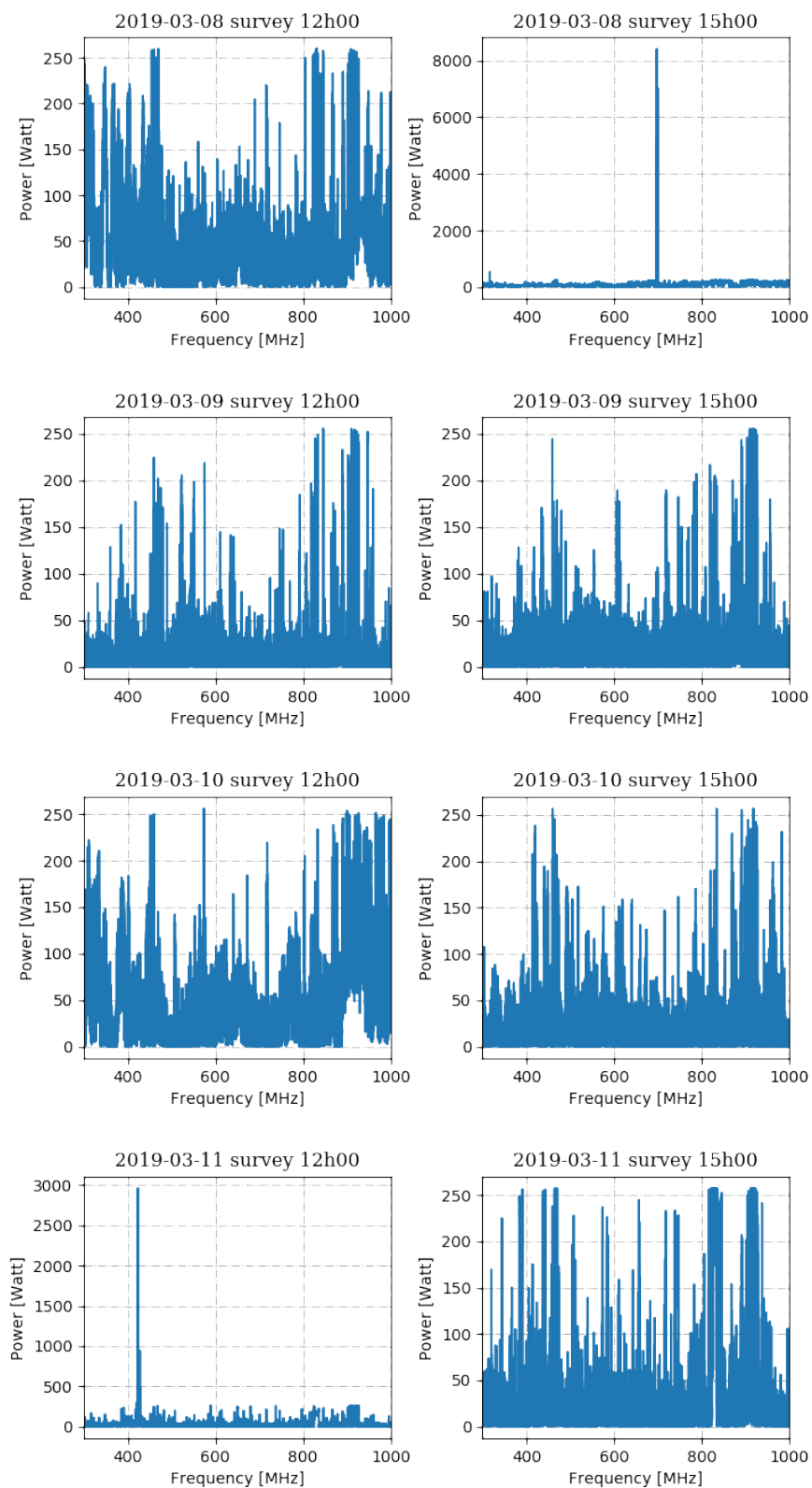
Assimetria



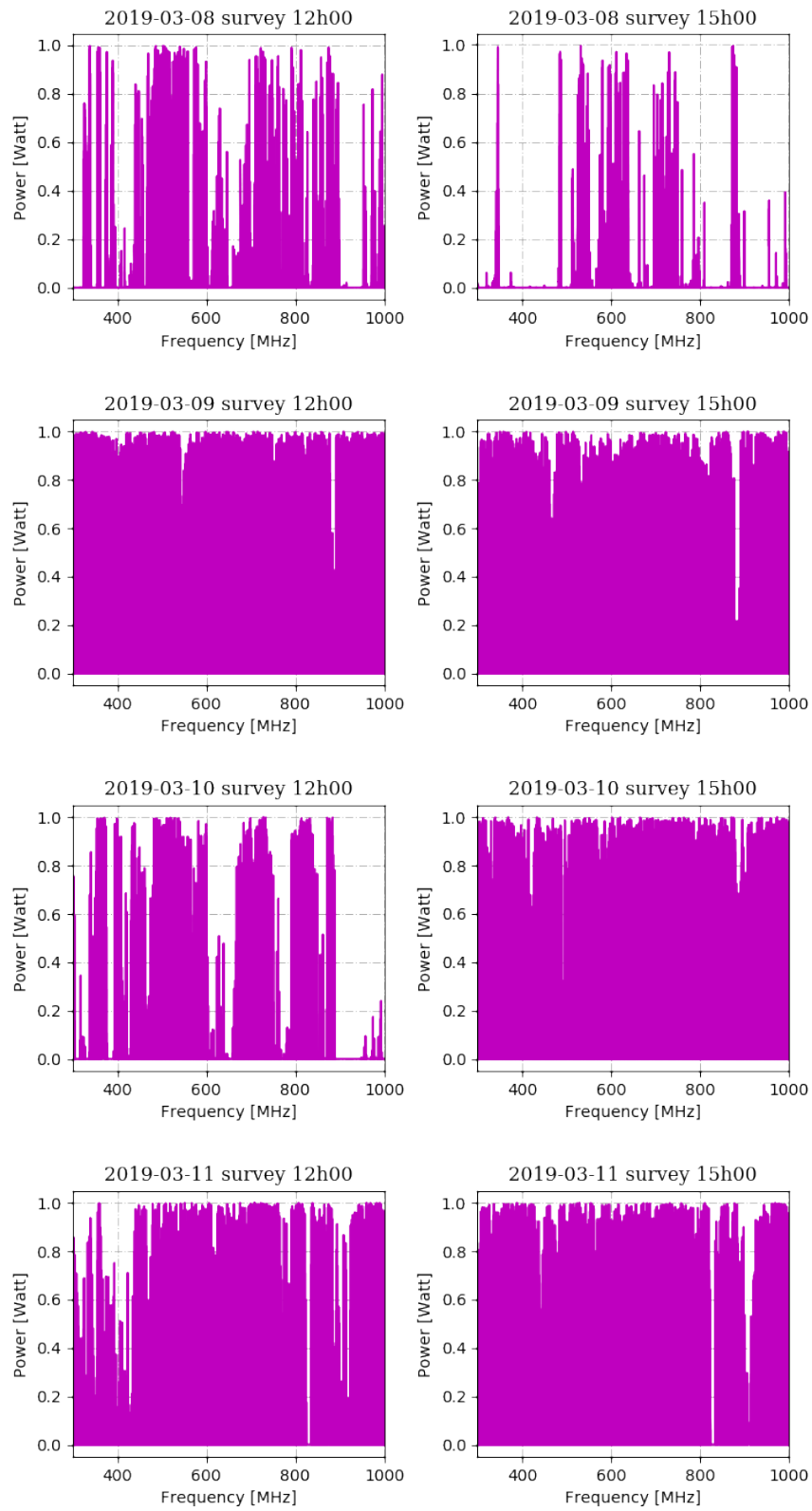
Curtose



Estatística do teste de normalidade



Probabilidade p de rejeição de h_0 no teste normalidade D'Agostino



6.3 Códigos em Python

df_convert

August 6, 2019

```
In [ ]: #Lê um arquivo csv gerado por rtl_power, convertendo o em data frame,  
        #Em seguida localizada as leituras de mesma frequência ao longo da varredura  
        #e as organiza array. São então excetudas funções para levantamento dos  
        #indicadores, que são posteriormente gravados em outro csv
```

0.0.1 Imports

```
In [ ]: import pandas as pd  
        import numpy as np  
        import time  
        from time import sleep  
        from IPython.display import display, clear_output  
        import import_ipynb  
        from power_unit_converter import db2w as db2w  
        from power_unit_converter import w2db as w2db  
        from scipy.stats import kurtosis  
        from scipy.stats import skew  
        from scipy.stats import shapiro  
        from scipy.stats import normaltest  
  
        clear_output()
```

0.0.2 Loading Data

```
In [ ]: file_name=input('nome do arquivo: ')  
        print('Loading...')  
        df=pd.read_csv(file_name, header=None)  
        print('Dataframe Loaded')
```

0.0.3 Data extract

```
In [ ]: #gap é quantidade de linhas até qu f0 se repita novamente, ou seja, é a quantidade  
        #de saltos em frequência até a próxima varredura  
  
        #column 2 row 0 => f0  
        #columns 4 all rows => bin size  
  
        hora=file_name[6:8]
```

```

i=1
while df[2][i]!=df[2][0]:
    i+=1
gap=i
binsize=df[4][0]
print(f'Horário da medição: {hora}h, FFT bin size: {binsize} [Hz]
      e "gap" entre leituras: {gap} linhas')

```

```
In [ ]: rows, columns = df.shape
```

```
In [ ]: #Serão construídos dois novos dataframes a partir do dataframe original. Um terá
#stats (em watts) de cada frequência lida em suas colunas (dfstats), o outro terá
#em cada linha todas as leituras (já convertidas em watts) referentes a uma
#frequência para o estudo da distribuição(freq_reads)
```

```

print('Extraindo dados')
start=time.time()

```

```

max_list=[]
min_list=[]
avg_list=[]
std_list=[]
krt_list=[]
skw_list=[]
shpW_list=[]
shpP_list=[]
k2_list=[]
p_list=[]

```

```
freq_reads=[]
```

```
for k in range (0,gap): #k é a linha de referência das operações a seguir
```

```

    row_max_list=[]
    row_min_list=[]
    row_avg_list=[]
    row_std_list=[]
    row_krt_list=[]
    row_skw_list=[]
    row_shpW_list=[]
    row_shpP_list=[]
    row_k2_list=[]
    row_p_list=[]

```

```

for j in range (6,columns):#coluna 6 é onde se iniciam os dados de potência
    print(f'k:{k} de {gap}; j:{j} de {columns-6}',end='') #display
    clear_output(wait=True)

```

```

freq_reads_row=[]
#referente frequência na posição zero da lista
freq_reads_row.append(df[2][0] + k*1.4e6 + binsize*(j-6))

idem_freq_list=[]
for i in range(k,rows,gap):
    #pulando 'gap', tem-se amostras de mesma frequência de diferentes
    #instantes de varredura; #aqui são adicionados a lista os valores
    #convertidos em watt das amostras de mesma frequência
    idem_freq_list.append(db2w(df[j][i]))
    freq_reads_row.append(db2w(df[j][i]))

#converte para um array numpy
np_idem_freq_list=np.array(idem_freq_list).astype(np.float)

max_pot=np.amax(np_idem_freq_list)
min_pot=np.amin(np_idem_freq_list)
mean=np.mean(np_idem_freq_list)
std=np.std(np_idem_freq_list)
krt=kurtosis(np_idem_freq_list)
skw=skew(np_idem_freq_list)
shpW, shpP = shapiro(np_idem_freq_list)
k2, p = normaltest(np_idem_freq_list)

#As listas a seguir são listas em que cada elemento corresponde aos
#momentos das amostras de armazenadas em idem_freq_list,ou seja,
#cada elemento é calculado a partir de {k, k+1*gap, k+2*gap+....}
#para uma determinada coluna j
row_max_list.append(max_pot)
row_min_list.append(min_pot)
row_avg_list.append(mean)
row_std_list.append(std)
row_krt_list.append(krt)
row_skw_list.append(skw)
row_shpW_list.append(shpW)
row_shpP_list.append(shpP)
row_k2_list.append(k2)
row_p_list.append(p)

#Preenchimento da lista que irá compor (linha a linha) o dataframe
#para estudo da distribuição
freq_reads.append(freq_reads_row)

#No arquivo csv da varredura, a diferença entre as frequência
#do último valor de frequência de última coluna em uma determinada linha
#e o primeiro valor de frequência na próxima linha PODE não ser o

```

*#bin_size o rtl_power realiza um ajuste de frequência para cada linha
#e à partir desse valor realiza os saltos de tamanho bin size para as
#próximas amostras nas colunas adjacentes. Como desejamos criar um csv
#de tamanho reduzido com apenas duas colunas, onde uma é a frequência,
#se este passo não for realizado, podemos acabar ignorando o offset
#e acumulando certo erro.*

```
max_list.append(row_max_list)
min_list.append(row_min_list)
avg_list.append(row_avg_list)
std_list.append(row_std_list)
krt_list.append(row_krt_list)
skw_list.append(row_skw_list)
shpW_list.append(row_shpW_list)
shpP_list.append(row_shpP_list)
k2_list.append(row_k2_list)
p_list.append(row_p_list)
```

```
end=(time.time()-start)
print(f'Momentos calculados, tempo de execução: {end:.3f} s')
```

```
In [ ]: print('gerandos novos dataframes...')
```

```
avg_data=[]
std_data=[]
max_data=[]
min_data=[]
krt_data=[]
skw_data=[]
shpW_data=[]
shpP_data=[]
k2_data=[]
p_data=[]
freq=[]

for i in range(0,len(avg_list)):
    for j in range (0,len(avg_list[i])):
        avg_data.append(avg_list[i][j])
        std_data.append(std_list[i][j])
        max_data.append(max_list[i][j])
        min_data.append(min_list[i][j])
        krt_data.append(krt_list[i][j])
        skw_data.append(skw_list[i][j])
        shpW_data.append(shpW_list[i][j])
        shpP_data.append(shpP_list[i][j])
        k2_data.append(k2_list[i][j])
        p_data.append(p_list[i][j])
```

```

freq.append(binsize*j +df[2][i])

dfstats=pd.DataFrame({
    'freq':freq,
    'avg':avg_data,
    'std':std_data,
    'krt':krt_data,
    'skw':skw_data,
    'shpW':shpW_data,
    'shpP':shpP_data,
    'k2':k2_data,
    'p':p_data,
    'min':min_data,
    'max':max_data
})
dfstats.set_index('freq',inplace=True)

dfdistr=pd.DataFrame(data=freq_reads)

print('gravando novos CSV...')
dfstats.to_csv(hora+'h_stats_watt.csv')
dfdistr.to_csv(hora+'h_dist_watt.csv')

print('concluído')

```

power_unit_converter

August 6, 2019

```
In [ ]: #db_watt power in dB W  
        #watt power in W  
  
import numpy as np  
  
def db2w(db_watt) :  
    watt=np.power(10, db_watt/10 )  
    return watt  
  
def w2db(watt):  
    db_watt=10*np.log10(watt)  
    return db_watt
```

ploting_stats

August 6, 2019

0.0.1 Imports

```
In [ ]: %matplotlib notebook
import os
import pandas as pd
import import_ipynb
import generate_plot

from IPython.display import display, clear_output
clear_output()
```

0.0.2 Data load

```
In [ ]: file_name=input('nome do arquivo: ') # 12h_stats_watt.csv; 15h_stats_watt.csv
print('Loading...')
df=pd.read_csv(file_name)
print('Dataframe Loaded')

In [ ]: dia = os.path.basename(os.path.dirname(os.path.realpath(file_name)))
hora=file_name[0:3]
binsize=df['freq'][1]-df['freq'][0]
n=len(df['freq']) #n indicará a quantidade de pontos nos gráficos
print(f'Dia da medição: {dia}, horário: {hora}00, FFT binzise: {binsize}
      e número de pontos: {n}')
```

0.0.3 Plotting

```
In [ ]: common_title = dia + ' survey ' + hora + '00'

#eixo y

#potência média
y_avg=[]
for i in range (0,n):
    y_avg.append((df['avg'][i]))

#desvio padrão
y_std=[]
```

```

for i in range (0,n):
    y_std.append((df['std'][i]))

#ratio std/avg
ratio=[]
inv_ratio=[]
for i in range (0,n):
    ratio.append(y_std[i]/y_avg[i])
    inv_ratio.append(1/ratio[i])

#assimetria
y_skw=[]
for i in range(0,n):
    y_skw.append((df['skw'][i]))

#curtose
y_krt=[]
for i in range(0,n):
    y_krt.append((df['krt'][i]))

#teste de normalidade Shapiro-Wilk
y_shpW=[]
for i in range(0,n):
    y_shpW.append((df['shpW'][i]))
y_shpP=[]
for i in range(0,n):
    y_shpP.append((df['shpP'][i]))

#teste de normalidade DAgostinos K^2 Test
y_k2=[]
for i in range(0,n):
    y_k2.append((df['k2'][i]))
y_p=[]
for i in range(0,n):
    y_p.append((df['p'][i]))

#min e max
min=[]
max=[]
for i in range (0,n):
    min.append(df['min'][i])
    max.append(df['max'][i])

#potência média +- k* desvio padrão
k=3
y_leftlim=[]
y_rightlim=[]
for i in range(0,n):

```

```

        y_leftlim.append(y_avg[i]-k*y_std[i])
        y_rightlim.append(y_avg[i]+k*y_std[i])

#eixo x - frequência em MHz
x=[]
for i in range (0,n):
    x.append(df['freq'][i]/1e6)

In [ ]: ### limites de frequencia plotados
        xInf = 300
        xSup = 1000

In [ ]: ### plots únicos

        title=common_title + ' - average'
        ylabel = 'Power [Watt]'
        generate_plot.singleplot(x, y_avg, 'p', xInf, xSup, ylabel, title)

        title=common_title + ' - log average'
        generate_plot.singlelogplot(x, y_avg, 'p', xInf, xSup, ylabel, title)

        title=common_title + ' - standard deviation'
        generate_plot.singleplot(x, y_std, 'm', xInf, xSup, ylabel, title)

        title=common_title + ' - log standard deviation'
        generate_plot.singlelogplot(x, y_std, 'm', xInf, xSup, ylabel, title)

        title=common_title + ' - standard deviation to average ratio'
        ylabel = ''
        generate_plot.singleplot(x, ratio, 'tab:red', xInf, xSup, ylabel, title)

        title=common_title + ' - average to standard deviaton ratio'
        generate_plot.singleplot(x, inv_ratio, 'tab:blue', xInf, xSup, ylabel, title)

        title=common_title + ' - skewness'
        generate_plot.singleplot(x, y_skw, 'p', xInf, xSup, ylabel, title)

        title=common_title + ' - kurtosis'
        generate_plot.singleplot(x, y_krt, 'm', xInf, xSup, ylabel, title)

        title=common_title + ' - Shapiro-Wilk test statistic (W)'
        generate_plot.singleplot(x, y_shpW, 'p', xInf, xSup, ylabel, title)

        title=common_title + ' - Shapiro-Wilk hypothesis p-value (p)'
        generate_plot.singleplot(x, y_shpP, 'm', xInf, xSup, ylabel, title)

        title=common_title + ' - DAgostinos K^2 test statistic score'
        generate_plot.singleplot(x, y_k2, 'p', xInf, xSup, ylabel, title)

```

```

title=common_title + ' - DAgostinos K^2 hypothesis p-value (p)'
generate_plot.singleplot(x, y_p, 'm', xInf, xSup, ylabel, title)

title=common_title + ' - minimum power'
ylabel = 'Power [Watt]'
generate_plot.twoseriesplot(x, min, 'r', y_leftlim, 'p',
                           xInf, xSup, ylabel, title)

title=common_title + ' - maximum power'
generate_plot.twoseriesplot(x, min, 'r', y_rightlim, 'p',
                           xInf, xSup, ylabel, title)

In [ ]: from power_unit_converter import db2w as db2w
        from power_unit_converter import w2db as w2db

In [ ]: ### plots duplos

color1 = 'tab:red'
ylabel1 = 'Average [Watt]'
color2 = 'tab:blue'
ylabel2 = 'standard deviation to average ratio'
title = common_title + ' - average and standard deviation'
generate_plot.dualplot(x, y_avg, ylabel1, color1, y_std, ylabel2,
                      color2, xInf, xSup, title)

color1 = 'tab:red'
ylabel1 = 'Average [Watt]'
color2 = 'tab:blue'
ylabel2 = 'Standard Deviation [Watt]'
title = common_title + ' - average against std/avg'
generate_plot.dualplot(x, y_avg, ylabel1, color1, ratio, ylabel2,
                      color2, xInf, xSup, title)

color1 = 'tab:red'
ylabel1 = 'DAgostinos K^2 hypothesis p-value (p)'
color2 = 'tab:blue'
ylabel2 = 'standard deviation to average ratio'
title = common_title + ' - Normality test against std/avg'
generate_plot.dualplot(x, y_p, ylabel1, color1, ratio, ylabel2,
                      color2, xInf, xSup, title)

color1 = 'tab:red'
ylabel1 = 'DAgostinos K^2 hypothesis p-value (p)'
color2 = 'tab:blue'
ylabel2 = 'Average [Watt]'
title = common_title + ' - Normality test against average power'
generate_plot.dualplot(x, y_p, ylabel1, color1, y_avg, ylabel2,
                      color2, xInf, xSup, title)

```

generate_plot

August 6, 2019

```
In [2]: import matplotlib.pyplot as plt
```

```
### singleplot
#renderiza um gráfico simples para uma série de dados
def singleplot(x, y, color, xInf, xSup, ylabel, title):
    title_font = {'family': 'serif',
                  'color': 'black',
                  'weight': 'normal',
                  'size': 16,
                  }

    fig=plt.figure(figsize=(8,6))
    plt.plot(x, y, color, marker='',linestyle="--", linewidth=1)
    plt.xlim(xInf,xSup)
    plt.grid(True, linestyle='-.')
    plt.title(title, fontdict=title_font)
    plt.xlabel('Frequency [MHz]')
    plt.ylabel(ylabel)

    return plt.show()

### singlelogplot
#renderiza um gráfico logaritmico para uma série de dados
def singlelogplot(x, y, color, xInf, xSup, ylabel, title):
    title_font = {'family': 'serif',
                  'color': 'black',
                  'weight': 'normal',
                  'size': 16,
                  }

    fig=plt.figure(figsize=(8,6))
    plt.semilogy(x, y, color, marker='',linestyle="--", linewidth=1)
    plt.xlim(xInf,xSup)
    plt.grid(True, linestyle='-.')
    plt.title(title, fontdict=title_font)
    plt.xlabel('Frequency [MHz]')
    plt.ylabel(ylabel)
```

```

    return plt.show()

### twoseriesplot
#renderiza um gráfico contendo duas séries de dados de mesmo contexto
def twoseriesplot(x, y1, color1, y2, color2, xInf, xSup, ylabel, title):
    title_font = {'family': 'serif',
                  'color': 'black',
                  'weight': 'normal',
                  'size': 16,
                  }

    fig=plt.figure(figsize=(8,6))
    plt.plot(x, y1, color1, marker='',linestyle="--", linewidth=0.8)
    plt.plot(x, y2, color2, marker='',linestyle="--",linewidth=1.2)
    plt.xlim(xInf,xSup)
    plt.grid(True, linestyle='-.')
    plt.title(title, fontdict=title_font)
    plt.xlabel('Frequency [MHz]')
    plt.ylabel(ylabel)

    return plt.show()

### dualplot
#renderiza um gráfico contendo duas séries de dados de mesmo distintos
def dualplot(x, y1, ylabel1, color1, y2, ylabel2, color2, xInf, xSup, title):
    title_font = {'family': 'serif',
                  'color': 'black',
                  'weight': 'normal',
                  'size': 16,
                  }

    fig, ax1 = plt.subplots(figsize=(8,6))

    plt.title(title, fontdict=title_font)
    ax1.set_xlabel('Frequency [MHz]')
    plt.xlim(xInf,xSup)

    ax1.set_ylabel(ylabel1, color=color1)
    ax1.plot(x, y1, color=color1)
    ax1.grid(True, linestyle='-.', linewidth=0.8)
    ax1.tick_params(axis='y', labelcolor=color1)

    ax2 = ax1.twinx()

    ax2.set_ylabel(ylabel2, color=color2)
    ax2.plot(x, y2, color=color2)
    ax2.grid(True, linestyle='-.', linewidth=1.2)

```

```

ax2.tick_params(axis='y', labelcolor=color2)

fig.tight_layout()

return plt.show()

### dualplot
#renderiza um gráfico logaritmico contendo duas séries de dados de mesmo distintos
def duallogplot(x, y1, ylabel1, color1, y2, ylabel2, color2, xInf, xSup, title):
    title_font = {'family': 'serif',
                  'color': 'black',
                  'weight': 'normal',
                  'size': 16,
                  }

    fig, ax1 = plt.subplots(figsize=(8,6))

    plt.title(title, fontdict=title_font)
    ax1.set_xlabel('Frequency [MHz]')
    plt.xlim(xInf,xSup)

    ax1.set_ylabel(ylabel1, color=color1)
    ax1.semilogy(x, y1, color=color1)
    ax1.grid(True, linestyle='-.', linewidth=0.8)
    ax1.tick_params(axis='y', labelcolor=color1)

    ax2 = ax1.twinx()

    ax2.set_ylabel(ylabel2, color=color2)
    ax2.semilogy(x, y2, color=color2)
    ax2.grid(True, linestyle='-.', linewidth=1.2)
    ax2.tick_params(axis='y', labelcolor=color2)

    fig.tight_layout()

    return plt.show()

```

all_stats_plotting

August 6, 2019

```
In [ ]: %matplotlib notebook
import os
import pandas as pd
import import_ipynb
import matplotlib.pyplot as plt
import generate_plot

from IPython.display import display, clear_output
clear_output()

In [ ]: files_path=['2019-03-08/12h_stats_watt.csv', '2019-03-08/15h_stats_watt.csv',
                    '2019-03-09/12h_stats_watt.csv', '2019-03-09/15h_stats_watt.csv',
                    '2019-03-10/12h_stats_watt.csv', '2019-03-10/15h_stats_watt.csv',
                    '2019-03-11/12h_stats_watt.csv', '2019-03-11/15h_stats_watt.csv']

print('Loading', end="")
frames=[]
for index, file in enumerate(files_path):
    print('.',end="")
    frames.append(pd.read_csv('../'+file))
print('\nDataframe Loaded')

In [ ]: ns=[]
binsizes=[]
for df in frames:
    i=1
    binsizes.append(df['freq'][1]-df['freq'][0])
    ns.append(len(df['freq']))

print(f'FFT bin sizes: {binsizes} [Hz] e "n" das leituras: {ns} pontos')

binsize=binsizes[0]
n=ns[0]

In [ ]: frames[5]

In [ ]: y=[]
x=[]
for df in frames:
```

```

#
yinner=[]
for i in range(0,n):
    yinner.append((df['avg'][i]))
y.append(yinner)
#eixo x - frequência em MHz
xinner=[]
for i in range (0,n):
    xinner.append(df['freq'][i]/1e6)
x.append(xinner)

ylabel='Power [Watt]'
xlabel='Frequency [MHz]'
color='tab:blue'

fig, (axs1, axs2, axs3, axs4) =
    plt.subplots(4,2, figsize=(8,16), gridspec_kw={'hspace': 0.45, 'wspace': 0.3})
for i in range (0,2):
    axs1[i].plot(x[i],y[i],color=color,)
    axs1[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
    axs1[i].grid(True, linestyle='-.')
    axs2[i].plot(x[i+2],y[i+2], color=color)
    axs2[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
    axs2[i].grid(True, linestyle='-.')
    axs3[i].plot(x[i+4],y[i+4], color=color)
    axs3[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
    axs3[i].grid(True, linestyle='-.')
    axs4[i].plot(x[i+6],y[i+6], color=color)
    axs4[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
    axs4[i].grid(True, linestyle='-.')

title_font = {'family': 'serif',
               'color': 'black',
               'weight': 'normal',
               'size': 12,
               }

axs1[0].set_title('2019-03-08 survey 12h00', fontdict=title_font)
axs1[1].set_title('2019-03-08 survey 15h00', fontdict=title_font)
axs2[0].set_title('2019-03-09 survey 12h00', fontdict=title_font)
axs2[1].set_title('2019-03-09 survey 15h00', fontdict=title_font)
axs3[0].set_title('2019-03-10 survey 12h00', fontdict=title_font)
axs3[1].set_title('2019-03-10 survey 15h00', fontdict=title_font)
axs4[0].set_title('2019-03-11 survey 12h00', fontdict=title_font)
axs4[1].set_title('2019-03-11 survey 15h00', fontdict=title_font)

```

```

In [ ]: y=[]
        x=[]

```

```

for df in frames:
    #
    yinner=[]
    for i in range(0,n):
        yinner.append((df['avg'][i]))
    y.append(yinner)
    #eixo x - frequência em MHz
    xinner=[]
    for i in range (0,n):
        xinner.append(df['freq'][i]/1e6)
    x.append(xinner)

ylabel='Power [Watt]'
xlabel='Frequency [MHz]'
color='tab:blue'

fig, (axs1, axs2, axs3, axs4) =
    plt.subplots(4,2, figsize=(8,16), gridspec_kw={'hspace': 0.45, 'wspace': 0.3})
for i in range (0,2):
    axs1[i].semilogy(x[i],y[i],color=color,)
    axs1[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
    axs1[i].grid(True, linestyle='-.')
    axs2[i].semilogy(x[i+2],y[i+2], color=color)
    axs2[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
    axs2[i].grid(True, linestyle='-.')
    axs3[i].semilogy(x[i+4],y[i+4], color=color)
    axs3[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
    axs3[i].grid(True, linestyle='-.')
    axs4[i].semilogy(x[i+6],y[i+6], color=color)
    axs4[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
    axs4[i].grid(True, linestyle='-.')

title_font = {'family': 'serif',
               'color': 'black',
               'weight': 'normal',
               'size': 12,
               }

axs1[0].set_title('2019-03-08 survey 12h00', fontdict=title_font)
axs1[1].set_title('2019-03-08 survey 15h00', fontdict=title_font)
axs2[0].set_title('2019-03-09 survey 12h00', fontdict=title_font)
axs2[1].set_title('2019-03-09 survey 15h00', fontdict=title_font)
axs3[0].set_title('2019-03-10 survey 12h00', fontdict=title_font)
axs3[1].set_title('2019-03-10 survey 15h00', fontdict=title_font)
axs4[0].set_title('2019-03-11 survey 12h00', fontdict=title_font)
axs4[1].set_title('2019-03-11 survey 15h00', fontdict=title_font)

```

In []: y=[]

```

x=[]
for df in frames:
    #
    yinner=[]
    for i in range(0,n):
        yinner.append((df['std'][i]))
    y.append(yinner)
    #eixo x - frequência em MHz
    xinner=[]
    for i in range (0,n):
        xinner.append(df['freq'][i]/1e6)
    x.append(xinner)

ylabel='Power [Watt]'
xlabel='Frequency [MHz]'
color='m'

fig, (axs1, axs2, axs3, axs4) =
    plt.subplots(4,2, figsize=(8,16), gridspec_kw={'hspace': 0.45, 'wspace': 0.3})
for i in range (0,2):
    axs1[i].plot(x[i],y[i],color=color,)
    axs1[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
    axs1[i].grid(True, linestyle='-.')
    axs2[i].plot(x[i+2],y[i+2], color=color)
    axs2[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
    axs2[i].grid(True, linestyle='-.')
    axs3[i].plot(x[i+4],y[i+4], color=color)
    axs3[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
    axs3[i].grid(True, linestyle='-.')
    axs4[i].plot(x[i+6],y[i+6], color=color)
    axs4[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
    axs4[i].grid(True, linestyle='-.')

title_font = {'family': 'serif',
               'color': 'black',
               'weight': 'normal',
               'size': 12,
               }

axs1[0].set_title('2019-03-08 survey 12h00', fontdict=title_font)
axs1[1].set_title('2019-03-08 survey 15h00', fontdict=title_font)
axs2[0].set_title('2019-03-09 survey 12h00', fontdict=title_font)
axs2[1].set_title('2019-03-09 survey 15h00', fontdict=title_font)
axs3[0].set_title('2019-03-10 survey 12h00', fontdict=title_font)
axs3[1].set_title('2019-03-10 survey 15h00', fontdict=title_font)
axs4[0].set_title('2019-03-11 survey 12h00', fontdict=title_font)
axs4[1].set_title('2019-03-11 survey 15h00', fontdict=title_font)

```

```

In [ ]: y=[]
        x=[]
        for df in frames:
            #
            yinner=[]
            for i in range(0,n):
                yinner.append((df['std'][i])/(df['avg'][i]))
            y.append(yinner)
            #eixo x - frequência em MHz
            xinner=[]
            for i in range (0,n):
                xinner.append(df['freq'][i]/1e6)
            x.append(xinner)

        ylabel='Power [Watt]'
        xlabel='Frequency [MHz]'
        color='tab:red'

        fig, (axs1, axs2, axs3, axs4) =
            plt.subplots(4,2, figsize=(8,16), gridspec_kw={'hspace': 0.45, 'wspace': 0.3})
        for i in range (0,2):
            axs1[i].plot(x[i],y[i],color=color,)
            axs1[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
            axs1[i].grid(True, linestyle='-.')
            axs2[i].plot(x[i+2],y[i+2], color=color)
            axs2[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
            axs2[i].grid(True, linestyle='-.')
            axs3[i].plot(x[i+4],y[i+4], color=color)
            axs3[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
            axs3[i].grid(True, linestyle='-.')
            axs4[i].plot(x[i+6],y[i+6], color=color)
            axs4[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
            axs4[i].grid(True, linestyle='-.')

        title_font = {'family': 'serif',
                       'color': 'black',
                       'weight': 'normal',
                       'size': 12,
                       }

        axs1[0].set_title('2019-03-08 survey 12h00', fontdict=title_font)
        axs1[1].set_title('2019-03-08 survey 15h00', fontdict=title_font)
        axs2[0].set_title('2019-03-09 survey 12h00', fontdict=title_font)
        axs2[1].set_title('2019-03-09 survey 15h00', fontdict=title_font)
        axs3[0].set_title('2019-03-10 survey 12h00', fontdict=title_font)
        axs3[1].set_title('2019-03-10 survey 15h00', fontdict=title_font)
        axs4[0].set_title('2019-03-11 survey 12h00', fontdict=title_font)
        axs4[1].set_title('2019-03-11 survey 15h00', fontdict=title_font)

```

```

In [ ]: y=[]
        x=[]
        for df in frames:
            #
            yinner=[]
            for i in range(0,n):
                yinner.append((df['skw'][i]))
            y.append(yinner)
            #eixo x - frequência em MHz
            xinner=[]
            for i in range (0,n):
                xinner.append(df['freq'][i]/1e6)
            x.append(xinner)

        ylabel='Power [Watt]'
        xlabel='Frequency [MHz]'
        color='tab:blue'

        fig, (axs1, axs2, axs3, axs4) =
            plt.subplots(4,2, figsize=(8,16), gridspec_kw={'hspace': 0.45, 'wspace': 0.3})
        for i in range (0,2):
            axs1[i].plot(x[i],y[i],color=color,)
            axs1[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
            axs1[i].grid(True, linestyle='-.')
            axs2[i].plot(x[i+2],y[i+2], color=color)
            axs2[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
            axs2[i].grid(True, linestyle='-.')
            axs3[i].plot(x[i+4],y[i+4], color=color)
            axs3[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
            axs3[i].grid(True, linestyle='-.')
            axs4[i].plot(x[i+6],y[i+6], color=color)
            axs4[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
            axs4[i].grid(True, linestyle='-.')

        title_font = {'family': 'serif',
                       'color': 'black',
                       'weight': 'normal',
                       'size': 12,
                       }

        axs1[0].set_title('2019-03-08 survey 12h00', fontdict=title_font)
        axs1[1].set_title('2019-03-08 survey 15h00', fontdict=title_font)
        axs2[0].set_title('2019-03-09 survey 12h00', fontdict=title_font)
        axs2[1].set_title('2019-03-09 survey 15h00', fontdict=title_font)
        axs3[0].set_title('2019-03-10 survey 12h00', fontdict=title_font)
        axs3[1].set_title('2019-03-10 survey 15h00', fontdict=title_font)
        axs4[0].set_title('2019-03-11 survey 12h00', fontdict=title_font)
        axs4[1].set_title('2019-03-11 survey 15h00', fontdict=title_font)

```

```

In [ ]: y=[]
        x=[]
        for df in frames:
            #
            yinner=[]
            for i in range(0,n):
                yinner.append((df['krt'][i]))
            y.append(yinner)
            #eixo x - frequência em MHz
            xinner=[]
            for i in range (0,n):
                xinner.append(df['freq'][i]/1e6)
            x.append(xinner)

        ylabel='Power [Watt]'
        xlabel='Frequency [MHz]'
        color='m'

        fig, (axs1, axs2, axs3, axs4) =
            plt.subplots(4,2, figsize=(8,16), gridspec_kw={'hspace': 0.45, 'wspace': 0.3})
        for i in range (0,2):
            axs1[i].plot(x[i],y[i],color=color,)
            axs1[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
            axs1[i].grid(True, linestyle='-.')
            axs2[i].plot(x[i+2],y[i+2], color=color)
            axs2[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
            axs2[i].grid(True, linestyle='-.')
            axs3[i].plot(x[i+4],y[i+4], color=color)
            axs3[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
            axs3[i].grid(True, linestyle='-.')
            axs4[i].plot(x[i+6],y[i+6], color=color)
            axs4[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
            axs4[i].grid(True, linestyle='-.')

        title_font = {'family': 'serif',
                       'color': 'black',
                       'weight': 'normal',
                       'size': 12,
                       }

        axs1[0].set_title('2019-03-08 survey 12h00', fontdict=title_font)
        axs1[1].set_title('2019-03-08 survey 15h00', fontdict=title_font)
        axs2[0].set_title('2019-03-09 survey 12h00', fontdict=title_font)
        axs2[1].set_title('2019-03-09 survey 15h00', fontdict=title_font)
        axs3[0].set_title('2019-03-10 survey 12h00', fontdict=title_font)
        axs3[1].set_title('2019-03-10 survey 15h00', fontdict=title_font)
        axs4[0].set_title('2019-03-11 survey 12h00', fontdict=title_font)
        axs4[1].set_title('2019-03-11 survey 15h00', fontdict=title_font)

```

```

In [ ]: y=[]
        x=[]
        for df in frames:
            #
            yinner=[]
            for i in range(0,n):
                yinner.append((df['k2'][i]))
            y.append(yinner)
            #eixo x - frequência em MHz
            xinner=[]
            for i in range (0,n):
                xinner.append(df['freq'][i]/1e6)
            x.append(xinner)

        ylabel='Power [Watt]'
        xlabel='Frequency [MHz]'
        color='tab:blue'

        fig, (axs1, axs2, axs3, axs4) =
            plt.subplots(4,2, figsize=(8,16), gridspec_kw={'hspace': 0.45, 'wspace': 0.3})
        for i in range (0,2):
            axs1[i].plot(x[i],y[i],color=color,)
            axs1[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
            axs1[i].grid(True, linestyle='-.')
            axs2[i].plot(x[i+2],y[i+2], color=color)
            axs2[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
            axs2[i].grid(True, linestyle='-.')
            axs3[i].plot(x[i+4],y[i+4], color=color)
            axs3[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
            axs3[i].grid(True, linestyle='-.')
            axs4[i].plot(x[i+6],y[i+6], color=color)
            axs4[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
            axs4[i].grid(True, linestyle='-.')

        title_font = {'family': 'serif',
                       'color': 'black',
                       'weight': 'normal',
                       'size': 12,
                       }

        axs1[0].set_title('2019-03-08 survey 12h00', fontdict=title_font)
        axs1[1].set_title('2019-03-08 survey 15h00', fontdict=title_font)
        axs2[0].set_title('2019-03-09 survey 12h00', fontdict=title_font)
        axs2[1].set_title('2019-03-09 survey 15h00', fontdict=title_font)
        axs3[0].set_title('2019-03-10 survey 12h00', fontdict=title_font)
        axs3[1].set_title('2019-03-10 survey 15h00', fontdict=title_font)
        axs4[0].set_title('2019-03-11 survey 12h00', fontdict=title_font)
        axs4[1].set_title('2019-03-11 survey 15h00', fontdict=title_font)

```

```

In [ ]: y=[]
        x=[]
        for df in frames:
            #
            yinner=[]
            for i in range(0,n):
                yinner.append((df['p'][i]))
            y.append(yinner)
            #eixo x - frequência em MHz
            xinner=[]
            for i in range (0,n):
                xinner.append(df['freq'][i]/1e6)
            x.append(xinner)

        ylabel='Power [Watt]'
        xlabel='Frequency [MHz]'
        color='m'

        fig, (axs1, axs2, axs3, axs4) =
            plt.subplots(4,2, figsize=(8,16), gridspec_kw={'hspace': 0.45, 'wspace': 0.3})
        for i in range (0,2):
            axs1[i].plot(x[i],y[i],color=color,)
            axs1[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
            axs1[i].grid(True, linestyle='-.')
            axs2[i].plot(x[i+2],y[i+2], color=color)
            axs2[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
            axs2[i].grid(True, linestyle='-.')
            axs3[i].plot(x[i+4],y[i+4], color=color)
            axs3[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
            axs3[i].grid(True, linestyle='-.')
            axs4[i].plot(x[i+6],y[i+6], color=color)
            axs4[i].set(xlabel=xlabel, ylabel=ylabel, xlim=(300,1000))
            axs4[i].grid(True, linestyle='-.')

        title_font = {'family': 'serif',
                       'color': 'black',
                       'weight': 'normal',
                       'size': 12,
                       }

        axs1[0].set_title('2019-03-08 survey 12h00', fontdict=title_font)
        axs1[1].set_title('2019-03-08 survey 15h00', fontdict=title_font)
        axs2[0].set_title('2019-03-09 survey 12h00', fontdict=title_font)
        axs2[1].set_title('2019-03-09 survey 15h00', fontdict=title_font)
        axs3[0].set_title('2019-03-10 survey 12h00', fontdict=title_font)
        axs3[1].set_title('2019-03-10 survey 15h00', fontdict=title_font)
        axs4[0].set_title('2019-03-11 survey 12h00', fontdict=title_font)
        axs4[1].set_title('2019-03-11 survey 15h00', fontdict=title_font)

```