INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Analisador de Espectro Inteligente para Deteção de Comunicações de *UAVs* utilizando Rádios Definidos por *Software*

Tiago Francisco da Costa Soeira

Mestrado em Engenharia de Telecomunicações e Informática

Orientador:
Doutor Nuno Manuel Branco Souto, Professor Associado
ISCTE-IUL

Co-Orientador:
Doutor Renato Branco Ferreira, Professor Auxiliar Convidado
ISCTE-IUL

Outubro, 2024

TECNOLOGIAS
E ARQUITETURA

Departamento de Ciências e Tecnologias da Informação

Analisador de Espectro Inteligente para Deteção de Comunicações de *UAVs* utilizando Rádios Definidos por *Software*

Tiago Francisco da Costa Soeira

Mestrado em Engenharia de Telecomunicações e Informática

Orientador:
Doutor Nuno Manuel Branco Souto, Professor Associado
ISCTE-IUL

Co-Orientador:
Doutor Renato Branco Ferreira, Professor Auxiliar Convidado
ISCTE-IUL

Outubro, 2024

Agradecimentos

Durante a realização do projeto tive apoio e ajuda de diversas pessoas este agradecimento vai para todas as pessoas que diretamente ou indiretamente apoiaram-me durante o projeto.

Gostaria de reconhecer a minha gratidão à minha família e amigos pelo apoio e compreensão constantes durante todo este percurso.

Gostaria de agradecer aos orientadores deste projeto, o Professor Nuno Souro e ao Professor Renato Branco pela orientação e apoio prestado. Gostaria também de agradecer ao ISCTE – IUL por todo o percurso e experiências durante o mestrado.

Um agradecimento ao Instituto de Telecomunicações pelo apoio prestado durante todo o mestrado e pela disponibilização do material utlizada no projeto.

Um agradecimento aos meus colegas de curso, nomeadamente Roberto Miguel, Ruben Bento e Filipe Gonçalves pelo apoio e ajuda prestado durante o mesmo, mas também pelas boas experiência e amizades que tornaram este percurso inesquecível.

Resumo

Drones já fazem parte do nosso dia a dia, sendo utilizados para vários fins, como vigilância, reconhecimento e uso recreativo. Com o aumento da utilização, é importante estabelecer medidas de controlo, de forma a garantir o bom funcionamento e segurança. Uma dessas medidas é a necessidade de sistemas *anti-drone*, capazes de identificar e possivelmente neutralizar estes equipamentos.

Esta dissertação tem como objetivo criar um sistema *anti-drone* de baixo custo para a deteção de *drones* através do varrimento do espectro de frequência, implementado com recurso a um *Software Defined Radio*. O *Software Defined Radio* utilizado no projeto foi o *HackRF One*, com um *switch Opera Cake*. O sistema foi implementado na plataforma *GNU Radio*, um software que permite a criação de sistemas SDR para o processamento de sinais de radiofrequência RF. No mesmo foi implementado um sistema capaz de efetuar o varrimento das gamas de frequências mais utilizadas por *drones* (2,4 GHz, 5 GHz e 5,8 GHz), determinando a possível existência de *drones* através da análise da potência do espectro do ambiente analisado. Caso exista um *drone* no ambiente, o sistema identifica também a frequência e o canal do sinal utilizado para a sua comunicação.

Foram realizados vários testes em ambientes e condições diferentes de modo a testar a capacidade de deteção do sistema. Os testes foram realizados em três ambientes e a três distâncias diferentes, tendo-se obtido uma taxa de acerto superior a 70% na maioria dos casos.

Palavras-Chave: *Software Defined Radio, Drone*, espectro de frequência, GNU Radio

Abstract

Drone have become a common in our day-to day lives, being used for various purposes, such as surveillance, area recognition and recreational use. With the increased use, it is important to establish control mechanism to ensure the proper functioning and safety. One of these mechanisms is the need for anti-drone systems, capable of identifying and possibly neutralizing this equipment.

This dissertation aims to create a low-cost anti-drone system for detecting drones by scanning the frequency spectrum, implemented using a Software Defined Radio. The Software Defined Radio used in the project was HackRF One, with an Opera Cake switch. The system was implemented on the GNU Radio platform, software that allows the creation of SDR systems for processing RF radio frequency signals. It implemented a system capable of scanning the frequency ranges most commonly used by drones (2.4 GHz, 5 GHz and 5.8 GHz), determining the possible existence of drones by analysing the power of the spectrum in the analysed environment. If there is a drone in the environment, the system also identifies the frequency and signal channel used for communication.

Several tests were carried out in different environments and conditions in order to test the system's detection capacity. The tests were carried out in three different environments and at three different distances, with a hit rate of over 70% in most cases.

KeyWords: Software Defined Radio, Drone, Frequency Spectrum, GNU Radio

Tabela de Conteúdos

Capítulo 1- Introdução 1
1.1. Motivação 2
1.2. Perguntas de Investigação 2
1.3. Objetivos 3
1.4. Metodologia de Desenvolvimento 3
1.5. Estrutura da Tese 3
Capítulo 2- Revisão da Literatura 5
2.1. Legislação portuguesa acerca de Drones/UAV 6
2.2. Características dos *Drones* 7
2.3. Comunicações dos UAVs 7
2.4. Software Defined Radio 11
2.5. Métodos de deteção de *drones* 13
2.6. GNU Radio 14
2.7. Projetos Relacionados 15
Capítulo 3- Arquitetura e Implementação do Sistema 17
3.1. Arquitetura do Sistema 17
3.2. Materiais e Ferramentas 17
3.2.1. HackRF One 17
3.2.2. Antenas 18
3.2.3. Opera Cake 20
3.2.4. Parrot BEBOP 2 21
3.2.5. FreeFlight Pro 22
3.3. Implementação do Sistema 24
3.3.1. Spectrum Detection 28
3.3.2. Funcionamento do Sistema 32
Capítulo 4- Testes e Resultados 35
4.1. Testes Controlados 35
4.1.1. Condições 35
4.1.2. Resultados 36
4.2. Testes Reais 40
4.2.1. Condições 40
4.2.2. Resultados 41
4.3. Métricas de precisão 51
4.3.1. Nenhum sinal RF 51

4.3.2. Sinal a 50 centímetros 52
4.3.3. Sinal a 5 metros 53
4.3.4. Sinal a 16 metros 55
Capítulo 5- Conclusão e Trabalho Futuro 58
5.1. Conclusão 58
5.2. Trabalho Futuro 59

Lista de Figuras

Figura 1-Exemplo de estrutura SDR [28] 12
Figura 2- Comparação entre um recetor analógico e um recetor de um SDR [29] 12
Figura 3- Diagrama de blocos do Sistema 17
Figura 4- SDR HackRF One [39] 18
Figura 5- Sistema com a antena Omnidirecional conectada 19
Figura 6- Sistema com a antena Direcional conectada 20
Figura 7- Opera Cake [42] 21
Figura 8- Parrot Bebop 2 [44] 22
Figura 9- Interface de controlo do drone da aplicação FreeFlight Pro 22
Figura 10- Interface das definições de rede da aplicação FreeFlight Pro 23
Figura 11- Fluxograma do sistema de deteção no Gnu Radio 25
Figura 12- Fluxograma do Sistema 31
Figura 13- Prompt do Sistema 32
Figura 14- Terminal em que nenhum sinal foi encontrado 33
Figura 15- Terminal com um sinal encontrado 34
Figura 16- Ambiente dos testes controlados 36
Figura 17- Hardware do sistema 36
Figura 18- Ambiente urbano dos testes reais 40
Figura 19- Configuração do teste de 50 centímetros 41
Figura 20- Configuração do teste de 5 metros 42
Figura 21- Configuração dos testes com a antena direcional 47
Figura 22- Interface da aplicação FreeFlight Pro 49
Figura 23- Evolução da percentagem de positivos verdadeiros 56
Figura 24- Evolução da percentagem de análises no canal correto 57

Lista de Tabelas

Tabela 1- Resultado da primeira fórmula e filtros 5
Tabela 2- Resultado da segunda fórmula e filtro 5
Tabela 3- Gamas de frequência utilizadas na comunicação de drones [8] 8
Tabela 4- Potência de transmissão para diferentes gamas de frequência [53] 18
Tabela 5- Métricas dos testes a 50 centímetros 52
Tabela 6- Métricas dos testes 5 metros 54
Tabela 7- Métricas dos testes a 16 metros 55

Lista de Acrónimos

Sinal RF: Sinal de Radiofrequência

SDR: Software Defined Radio

GNSS: Global Navigation Satellite System

FHSS: Frequency-Hopping Spread Spectrum

DSP: Digital Signal Processor

FPGA: Field Programmable Gate Arrays

P: Positivos

N: Negativos

PV: Positivos Verdadeiros

PF: Positivos Falsos

NF: Negativos Falsos

NV: Negativos Verdadeiros

PPV: Percentagem de Positivos Verdadeiros

PPF: Percentagem de Positivos Falsos

PNV: Percentagem de Negativos Verdadeiros

PNF: Percentagem de Negativos Falsos

PPVCC: Percentagem de Positivos Verdadeiros no Canal Certo

CAPÍTULO 1

Introdução

UAVs (*Unmanned aerial vehicles*), ou como são mais conhecidos, *drones*, são equipamentos controlados remotamente. Inicialmente os mesmos foram desenvolvidos para utilização militar, como reconhecimento e ataque, mas atualmente existem aparelhos que qualquer pessoa pode adquirir. A maioria dos *drones* são controlados a partir do solo, diretamente por um controlador, porém existem equipamentos mais avançados (normalmente utilizados por exércitos) onde o mesmo consegue navegar em rotas predefinidas. Os *drones* normalmente estão agrupados em 3 grupos. A maioria dos *drones* para uso recreativo enquadram-se no grupo de uso privado, em que qualquer pessoa pode obter um aparelho para uso recreativo, vantajoso por exemplo para a produção de conteúdos multimédia. O segundo grupo é para uso comercial, como por exemplo na agricultura, topografia e segurança pública. O último grupo é para uso militar, como ataques e reconhecimento.

Nos últimos anos tem se notado um aumento na utilização e acesso a estes dispositivos, e com isso surge também o aumento do risco de uso indevido, seja por utilizadores amadores que apenas estão a ser irresponsáveis, como por entidades ou organizações mal-intencionadas. Um exemplo disso é na recente Guerra na Ucrânia [1], onde são utilizados *drones* para ataque e reconhecimento, devido ao seu custo reduzido comparado com outras soluções, e também graças à rapidez e destreza que os mesmos possuem.

Neste projeto, é proposto o desenvolvimento de um sistema *anti-drone* de baixo custo que consegue efetuar uma verificação das gamas de frequência mais utilizadas por estes equipamentos e detetar a frequência e o canal do sinal estabelecido na comunicação entre o controlador e o *drone*. O sistema foi desenvolvido na plataforma Gnu Radio e utiliza o SDR *HackRF One*, onde é utilizado um *switch* chamado *Opera Cake* para conseguir utilizar diferentes antenas dependendo da gama de frequência que se está a analisar. Para testar o desempenho e robustez do sistema, foram realizados vários testes em ambientes e condições diferentes. Os testes foram realizados em três ambientes e a três distâncias diferentes, tendo-se obtido uma taxa de acerto superior a 70% na maioria dos casos, conseguindo na maioria dos casos determinar a frequência, canal e largura de banda do sinal detetado.

Para o projeto foi preciso perceber como funciona a comunicação estabelecida pelos *drones*, e as gamas de frequências mais utilizadas por estes equipamentos. Foi também

necessário perceber as ferramentas e funcionalidades principais dos *SDRs*, tendo em vista o desenvolvimento do sistema.

Este projeto resultou num artigo científico intitulado "*Smart Spectrum Analyser for UAV link detection using Software Defined Radios"*, encontrando-se atualmente em fase de submissão

1.1. Motivação

Nos últimos anos pôde-se observar um aumento da utilização de *drone*, como nas áreas de vigilância, indústria, agricultura, mas também de modo recreativo. Com o aumento da acessibilidade deste tipo de equipamentos, também aumenta o risco do uso indevido ou perverso. Com isso é necessário existirem medidas de controlo para estes equipamentos, para garantir a segurança e privacidade da sociedade, seja de utilizadores menos experientes ou desinformados, como de entidades com atividades mal-intencionadas.

Este projeto tem como objetivo atender à necessidade da existência de sistemas de controlo de *drones*. O mesmo consiste em um sistema capaz de fazer um varrimento do espectro de frequências e determinar a presença de *drones* no ambiente analisado, identificando a frequência e canal utilizado para a comunicação entre a aeronave e o controlador.

1.2. Perguntas de Investigação

No âmbito do tema abordado e o desenvolvimento do projeto, propõe-se as seguintes questões:

- Quais as gamas de frequências mais utilizadas por equipamentos *drone*?
- Que tipos de sinais são utilizados para a comunicação entre o controlador/telefone e o *drone* e quais são as suas propriedades?
- Qual o melhor hardware em termos de relação custo -eficácia para efetuar o varrimento do espectro e deteção da comunicação de um *drone*?
- Qual a técnica mais viável para conseguir detetar a presença de um *drone* no espectro de frequência, quando se recorre a placas SDR?
- Tendo em conta a diversidade de fontes RF e os diferentes tipos de *drones*, qual a melhor técnica para executar uma deteção robusta?

1.3. Objetivos

O objetivo do projeto é desenvolver um sistema de baixo custo que seja capaz de identificar as frequências utilizadas por equipamentos *drones* não autorizados. A solução irá ser implementada através de uma plataforma SDR. Para tal é necessário investigar a melhor forma de detetar a gama de frequência que está a ser utilizada para a comunicação entre o controlador e o *drone* e desenvolver um mecanismo de varrimento nas gamas de frequência mais utilizadas por estes dispositivos, tendo em conta as diferentes fontes de rádio frequência.

1.4. Metodologia de Desenvolvimento

O projeto foi planeado tendo em vista diversas fases de desenvolvimento, sendo que a primeira foi a pesquisa acerca dos equipamentos *drones* e o seu comportamento. Durante a mesma também foi feita uma adaptação sobre a utilização de equipamentos SDR e o *software* GNU Radio, onde foram testadas diferentes funcionalidades. A fase seguinte consistiu na implementação, em que com a informação adquirida na fase anterior, foi desenvolvido um sistema capaz de atender às propostas do projeto. Durante a mesma o sistema teve várias alterações, tanto em funcionalidades como também na estrutura. A fase final foi a de testes, onde foram executados uma série de testes em ambientes e condições diferentes, de modo a analisar a performance do sistema. Durante todas as fases o relatório do projeto também foi desenvolvido, tendo mais ênfase na última.

1.5. Estrutura da Tese

A tese está estruturada em 5 capítulos, em que o primeiro trata-se da introdução. O segundo consiste na revisão de literatura, onde foi descrita a pesquisa feita pertinentes ao projeto. No capítulo 3 é apresentada a metodologia do projeto, contando com a descrição dos equipamentos e softwares utilizados e a explicação do sistema desenvolvido. O capítulo 4 conta com a apresentação dos testes realizados de modo a analisar a performance do mesmo, contando com diferentes condições e ambientes. O último capítulo apresenta a conclusão do projeto e sugestões para trabalhos futuros.

CAPÍTULO 2

Revisão da Literatura

Neste capítulo será apresentado a revisão de literatura acerca da legislação portuguesa das características dos *drones*, de SDRs, de métodos de deteção e da plataforma Gnu Radio. A metodologia utilizada para o desenvolvimento da pesquisa e estado da arte foi o *Systematic Literature Review*. Em suma, a estratégia do SLR baseia-se na resposta às perguntas de investigação em diversas bases de dados. As base de dados utilizadas foram a *IEEE Xplore*, a *ACM* e a *Scopus*, em que foram utilizadas 2 fórmulas: *"(((:"SDR*" OR:"Software Defined Radio") AND ("Drone*" OR :"UAV*")))"* e *"((:"SDR*" OR :"Software Defined Radio") AND ((:"Spectrum Analyser" OR :"Spectrum Sensing")))"*. Também foi utilizada efeito "*snowball*", que consiste na identificação de novos artigos através de referências de artigos já conhecidos. Também foram disponibilizados alguns artigos por parte dos orientadores da dissertação. Para o resultado das fórmulas foram utilizados 4 filtros: 1º toda informação dos artigos, 2º formulas aplicadas aos resumos (*Abstracts*), 3º fórmula aplicada aos títulos das publicações. Foram retirados os artigos repetidos (4º) e selecionados os relevantes para a revisão de literatura (5º). Apenas foram considerados artigos a partir de 2013. Os resultados dos filtros podem ser observados nas tabelas 1 e 2.

Tabela 1- Resultado da primeira fórmula e filtros

	1º	2º	3º	4º	5º	Referências
IEEEXplore	214	168	32	32	2	[2] [3]
ACM	17718	631	57	57	0	
SCOPUS	3998	209	15	6	1	[4]

Tabela 2- Resultado da segunda fórmula e filtro

	1º	2º	3º	4º	5º	Referências
IEEEXplore	208	139	26	26	5	[5] [6] [7] [8] [9]
ACM	285447	29738	2105	2105	0	
SCOPUS	6460	311	18	10	1	[10]

Os restantes artigos, relatório e websites utilizados na revisão de literatura do projeto foram identificados através de uma pesquisa direta ou por efeito *snowball*.

2.1. Legislação portuguesa acerca de Drones/UAV

Em Portugal a utilização de *drones* está sujeita a regulamentações que os seus proprietários têm de seguir para garantir o bom funcionamento dos mesmo e a segurança pública.

A utilização destes aparelhos está dividida em 3 categorias, a aberta, específica e certificada.

A categoria aberta é destinada à maioria dos aficionados pela prática da utilização e controlo de aeronaves não tripuladas em que têm as seguintes regulamentações. [11] A aeronave tem de ter uma massa à decolagem inferior a 25 kg, com uma marcação de conformidade de classe europeia

- Não é obrigatório uma autorização operacional da ANAC (Autoridade Nacional de Aviação Civil)
- Só podem operar na VLOS (*Visual Line-Of-Sight*)
- Não podem operar em zonas proibidas ou de restrição operacional de aeroportos ou heliportos
- Não podem operar a mais de 120 metros do terreno (existem algumas exceções como para planadores)
- Não se pode voar sobre concentração de pessoas

A categoria específica é uma classe intermédia de operação, que tem menos restrições comparado com a categoria anterior. Porém a mesma está sujeita às seguintes regulamentações [12].

- Só podem operar com uma autorização operacional da ANAC
- Podem operar acima de 120 metros
- Podem operar acima da BVLOS (*Beyond* VLOS)
- Pode operar em áreas proibidas, mas com restrições

Por fim, a categoria certificada [13] possui algumas restrições comuns com as naves tripuladas, que são:

- Necessidade de uma certificação aeronáutica
- Pode ser destinado ao transporte de pessoas
- Pode transportar mercadorias perigosas
- A aeronave tem de possuir uma dimensão igual ou superior a 3 metros

Todos os pilotos têm de ser sujeitos a uma formação, que varia dependendo da categoria pretendida. Todos os *drones* tem de estar registados na plataforma informática da ANAC [14] [15].

2.2. Características dos *Drones*

Os *drones* são equipamentos desenhados para operaram remotamente, estabelecendo uma comunicação com um controlador. A parte de *hardware* consiste em componentes responsáveis pela movimentação e operação do equipamento. Os componentes diretamente responsáveis pelo movimento são as hélices e os motores. As hélices geram o impulso para mover o equipamento. Cada hélice está ligada a um motor, que é responsável por girar as mesmas e criar a uma força suficiente para controlar a aeronave. Um *drone* pode ter um número diverso de hélices, sendo o mais comum 4 (quadricóptero).

Um *drone* também possui uma bateria que é usada para alimentar os motores e componentes eletrónicos, sendo as baterias de polímero de lítio as mais utilizadas.

Como componentes eletrónicos, estes equipamentos possuem um controlador de voo, uma placa de distribuição de energia, um controlador eletrónico de velocidade e diversos sensores. O controlador de voo é como se fosse o "cérebro" do equipamento, pois é responsável por processar toda a informação recebida, seja pelos diversos sensores presentes no próprio dispositivo, como pelo sinal proveniente do controlador. É a partir deste componente que as ordens de direção e deslocamento são processadas, passando essa informação para os motores. O controlador eletrónico de velocidade controla a velocidade de cada motor. Por fim, a placa de distribuição de energia, como o nome indica, distribui energia entre os diferentes componentes [16].

A aeronave consegue ser controlada remotamente através de um controlado remoto, que é um componente externo em que o utilizador consegue controlar a direção, ângulo e altitude. O mesmo possui uma antena para receber e transmitir um sinal rádio.

2.3. Comunicações dos UAVs

Como mencionado no subcapítulo anterior, um *drone* é constituído por dois equipamentos, a aeronave e o controlador. Para a operacionalidade da aeronave é necessário estabelecer uma comunicação através sinais de radio frequência entre os dois equipamentos,

para enviar comandos de direção, telemetria e multimédia. Algumas aeronaves também têm a capacidade comunicar com um sistema GNSS para obter informação acerca da orientação e navegação. Este tipo de comunicação é utilizado em *drones* que têm a capacidade de efetuar voo autónomos e a funcionalidade de voltar à "base".

Na tabela 3 pode-se observar as diferentes gamas de frequências utilizadas para estes sinais, bem como os sistemas de transmissão utilizados por várias marcas.

Tabela 3- Gamas de frequência utilizadas na comunicação de drones [8]

Tipo de Comunicação	**Gamas de Frequência**	**Sistemas de Transmissão**
Controlador (Telemetria)	27 MHZ 35 MHz 72 MHz 433 MHz 968 MHz 2,4 GHz 5,8GHz	DSM2 (Spektrum) DSMX (Spektrum) ACCESS (FrSky) FASST (Futaba) S-FHHS (Futaba) T-FHHS (Futaba) OcuSync (DJI) Lightbridge (DJI)
Multimédia	328-334 MHz 433 MHz 2,4 GHz 5,8 GHz	
Navegação	L1: 1,563 – 1,587 GHz L2: 1,215 – 1,2396 GHz L5: 1,164 – 1,189 GHz G1: 1,593 – 1,610 GHz G2: 1,237 – 1,254 GHz G3: 1,189 – 1,214 GHz E1: 1,559 – 1,592 GHz E5a/b: 1,164 – 1,215 GHz E6: 1,260 – 1,300 GHz	

A comunicação entre *drones* e os seus controladores é estabelecida em gamas de frequência entre 900 MHz e 5,8 GHz.

A faixa de frequência de 2,4 GHz é a mais utilizada para a comunicação de *drones*, pois possui uma grande disponibilidade e é compatível com vários controladores, conseguindo um alcance razoável. Porém a mesma é bastante popular para utilização de outros equipamentos, como equipamentos *Bluetooth* e *WI-FI*, o que pode gerar interferência na gama.

A frequência de 5,8 GHz também está cada vez a ser mais utilizada, principalmente por equipamentos com câmaras incorporadas. A mesma oferece uma menor interferência de sinal proveniente de outros equipamentos wireless, uma melhor transmissão de informação, o que é

bastante importante para o envio de vídeo. A única desvantagem comparada com a faixa anterior é que possui um alcance menor.

A faixa de 900 MHz é utilizada por *drones* que conseguem estabelecer uma comunicação longas distâncias. Os mesmos são mais utilizados para fins industriais e vigilância.

Por fim as faixas entre 1,2GHz e 1,3 GHz também são utilizadas (menor utilização comparado com as anteriores). Esta faixa de frequência consegue manter a operacionalidade a uma longa distância e a grandes altitudes. A mesma é utilizada para fins de pesquisa e uso militar [17].

Também é possível utilizar comunicações WI-FI, LTE-A \4G, 5G e Bluetooth entre o *drone* e o controlador, que permitem uma comunicação mais estável e um maior alcance. [18]

Também existem *drones* com a capacidade de controlo autónomo, que conseguem executar rotas pré-definidas. Para esses casos existem sistemas de rádio-navegação que são necessários, como *Global Positioning System* (GPS) que através de uma rede de satélites, providencia a localização exata do equipamento. O *Inertiual Measurement Units* (IMU), que mede a aceleração, e determina a velocidade do *drone*, também é bastante importante para este tipo de sistemas. Existem *drones*, que através de inteligência artificial, conseguem fazer decisões em tempo real ao analisar informações vindas de sensores, e assim delinear a sua rota, ou ajustar a mesma, como desviar de obstáculos ou decisões mais complexas. Os *drones* autónomos possuem uma série de sensores que os equipamentos convencionais não têm, como sensores de ultrassom e Lidar, para deteção de obstáculos. O poder computacional também é superior, especialmente se o mesmo processar algoritmos de definição de rotas [19].

A maioria dos controladores operam na banda de frequência de 2,4 GHz e utilizam um método de transmissão *Frequency-hopping Spread Spectrum* (FHSS).

O FHSS é um método de transmissão de sinais RF que consiste na troca repentina da frequência da portadora de uma maneira pseudo-aleatória, conhecida tanto pelo transmissor como pelo recetor. Com isto a comunicação tem uma maior robustez, sendo menos suscetível ao ruido do espectro de frequência, o que é útil na gama dos 2,4 GHz. Também é eficaz contra *jamming*, visto que a transmissão não está a ocorrer numa frequência estática, mas sim numa gama onde o sinal está constantemente a "saltar".

A maioria dos *drones* comerciais utilizam sinais *Wi-Fi* para a comunicação entre o controlador e o *drone*. Os mesmos na maioria utilizam um único sinal para transmitir os comandos de movimento do equipamento e para a transmissão de vídeo em tempo real, mas existem *drones* que separam a emissão em 2 sinais.

Grande parte destes equipamentos são *dual-band*, o que significa que podem operar em 2 gamas de frequências (o mais comum são as gamas de 2,4 e 5,8 GHz), pelo que podem "saltar" dentro das 2 gamas para o canal com menor interferência.

Neste projeto foram analisadas duas das maiores marcas de *drones* comerciais, a *DJI* e a *Parrot*.

A *DJI* possui diversos modos de transmissão, que evoluíram ao longo das várias gerações de equipamento. Os primeiros *drones* do fabricante utilizam sinais RF sem um modelo de transmissão em específico. Nos primeiros modelos a maioria dos controladores não possui uma tela, pelo que é necessário conectar um smartphone. Essa ligação é estabelecida através da aplicação DJI GO, que através de um sinal *Wi-Fi*, estabelece uma conexão com o *drone*. No caso de alguns equipamentos como o *DJI Phantom 3 Standart* [20], ao conectar o smartphone, são transmitidos dois sinais. O primeiro sinal é para o controlo do *drone*, que é um sinal RF entre 5,725-5,825 GHz, que é estabelecido entre o *drone* e o controlador. O segundo sinal é um sinal Wi-Fi, em que o controlador age como um *access point* e cria uma rede Wi-Fi, à qual o smartphone se conecta.

Para modelos mais profissionais, a *DJI* criou o *DJI Lightbridge*, um sistema de transmissão desenvolvido para comunicações de longo alcance, tanto para o controlo do equipamento como para vídeo. Este consegue transmitir um sinal vídeo em alta-definição (1080p), operando nas gamas de 2,4 e 5,8 GHz. De acordo com as normas CE (Europa), o equipamento tem um alcance máximo de 3,5 quilómetros. Entre os equipamentos que utilizam o *Lightbridge*, destacam-se as versões Pro e *Advances* do *DJI Phantom* 4 e o *DJI Inspire* 2.

O sistema de transmissão mais recente utilizado pelos *drones* da DJI é o *DJI OcuSync*. Este veio substituir o *DJI LightBridge*, oferecendo uma maior eficiência, alcance e estabilidade. Opera em *dual-band*, nas faixas de frequência de 2,4 GHz e 5,8 GHz. Possui também uma maior resistência a interferências, com um sistema que varre o ambiente e determina as zonas do espectro com menos ruído, efetuando a troca de frequência automaticamente O *DJI OcuSync* consegue transmitir vídeo em 1080p e, a partir da versão 3.0, em 4K. É também possível selecionar manualmente o canal de frequência utilizado para a transmissão do sinal de vídeo, bem como a largura de banda desejada. Em relação ao alcance, os equipamentos com o *DJI OcuSync* podem operar a distâncias até 8 quilómetros em zonas com a norma CE. Entre os equipamentos que utilizam este sistema, destacam-se o DJI Mavic 2 e 3 (todas as suas versões) [21], e o *DJI Phantom* 4 Pro V2.0 [22].

O *DJI OcuSync* utiliza dois sinais distintos: um para os controlos da aeronave e outro para a transmissão de vídeo. Por definição, ambos operam dentro da gama de frequência selecionada (com menor interferência), porém, se o canal utilizado para a transmissão de vídeo for escolhido manualmente, o sinal dos controlos continua a operar na frequência original, operando assim em duas gamas diferentes (apenas é possível selecionar manualmente o canal do sinal de vídeo) [23] [24].

A maioria dos *drones Parrot* utiliza sinais Wi-Fi para a transmissão, com a possibilidade de operar tanto na gama de 2,4 GHz como na de 5,0 GHz. Uma exceção é o *drone Parrot AI*, que à data deste projeto é um dos modelos mais avançados da empresa, utilizando uma conectividade 4G LTE nativa. [25].

A maioria dos equipamentos da *Parrot* pode ser operada por um controlador, designado *SkyController*, ou por um smartphone. Para estabelecer a ligação com o smartphone, é necessário instalar a aplicação *FreeFlight Pro*. Esta aplicação permite controlar a aeronave, selecionar o canal utilizado para a transmissão e visualizar em tempo real as imagens capturadas pelo *drone*. Para a comunicação com o controlador, o *drone* atua como um ponto de acesso Wi-Fi, criando uma rede à qual o smartphone se pode conectar.

2.4. Software Defined Radio

Equipamentos rádio conseguem receber e transmitir frequências rádio, conseguindo estabelecer comunicações com outros equipamentos. No entanto, equipamentos rádio convencionais possuem algumas limitações, pois só é possivel fazer ajustes ou alterações, como a aplicação de amplificadores, moduladores, demoduladores, detetores de RF, filtros, de forma manual [26].

Para ultrapassar esses problemas, o PhD *Joseph Mitola* definiu o conceito *Software Defined Radio* no início dos anos 90, para rádios onde apenas a antena e o ADC e DAC, para o recetor e transmissor respetivamente, seriam componentes físicos, e tudo podendo o resto ser reprogramado. O seu desejo não se concretizou, porém o termo SDR é utilizado para um equipamento radio programável, com alguns componentes hardware imprescindíveis [27]. SDR são equipamentos rádio com a flexibilidade de um sistema programado, o que quer dizer que o seu comportamento pode ser alterado simplesmente com uma modificação de software. Com isso, deixa de ter um comportamento estático, para ter um comportamento dinâmico, onde podem ser alteradas propriedades com o a largura de banda e a modulação.

Os SDR são constituídos por duas partes, o transmissor e o recetor, como é possível observar na figura 1.

Para a programação do software, o SDR usa dispositivos reprogramáveis de alta-velocidade, como o *Digital Signal Processors* (DSP), *Field Programmable Gate Arrays* (FPGA) e *General Purpose Processors* [28].

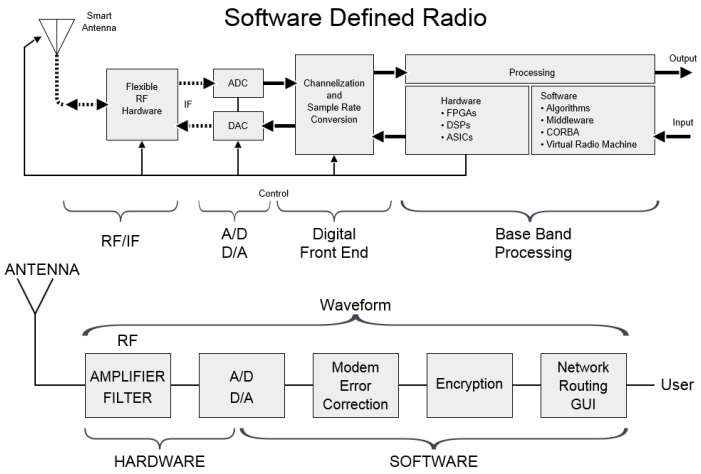

Figura 1-Exemplo de estrutura SDR [28]

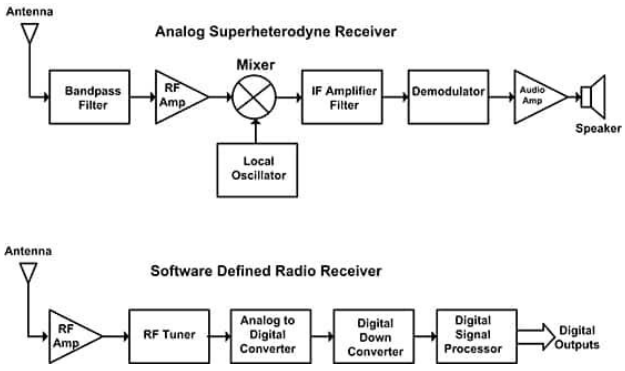

Figura 2- Comparação entre um recetor analógico e um recetor de um SDR [29]

Na figura 2 encontram-se dois diagramas, um recetor rádio-analógico, e um recetor de um SDR, respetivamente.

No recetor analógico, é utilizado um filtro passa-banda para limitar o sinal recebido. De seguida o sinal é amplificado através do Amplificador RF. Ao sinal amplificado é aplicado um deslocamento de acordo com a banda base do sistema (frequência de amostragem). O amplificador IF posteriormente amplifica e filtra o sinal convertido. Por fim o sinal é

desmodulado. Todos estes processos são executados por componentes fisicos, onde caso seja necessário fazer alguma alteração ao sistema, é necessário substituir os componentes.

No recetor presente no SDR, o sinal começa por ser amplificado (processo igual ao recetor analógico). De seguida o sinal RF é convertido em um sinal IF (sinal de frequência intermédia) e é convertido para um sinal digital através do *Analog-Digital Converter* (ADC). A seguir no DDC (*Digital Down Conversion*) o sinal é convertido em amostras de acordo com a banda base(I*/Q samples*). Por fim o sinal é processado no DSP. Também pode ser utilizado um FPGA em vez do DSP, caso seja necessário um processamento de sinal rápido.

2.5. Métodos de deteção de *drones*

Existem diversas técnicas para identificar um *drone*, as quais podem dividir-se em 3 análises: Vídeo (ou visualmente), áudio e sinais RF.

Visualmente é possível identificar um *drone* utilizado uma câmara com uma boa resolução, e desenvolvendo um algoritmo de reconhecimento de imagem, é possível determinar o aparelho e o modelo do *drone* analisado. Contudo este método tem alguns defeitos, por exemplo, caso o sistema necessite de um movimento para começar a sua análise, a presença de aves pode confundir-se com um *drone*. O mesmo acontece para *drones* de asa fixa, muito utilizados com intuitos militares. [29] Para esse tipo de *drones*, a deteção termal pode ser uma boa abordagem. Os *drones* no seu funcionamento emitem calor, que por sua ver emite radiação *infrared*, que pode ser detetada através de sistemas de análise termal. Este tipo de abordagem funciona melhor para *drones* de asa fixa, que costuma ter motores de propulsão que geram emissões de gases quentes. Esta técnica tem algumas desvantagens, como apenas ter uma boa eficácia durante a noite e poder-se confundir aves por *drones*, visto que as mesmas também são uma fonte de calor comparativamente com o ambiente. Também não é eficaz para *drones* comerciais, onde a principal fonte de calor é a bateria, geralmente coberta por plástico. [30]

A deteção de *drones* através de áudio consiste no desenvolvimento de um algoritmo que consiga determinar o modelo que está a operar através do som que o mesmo emite. Os *drones* quando estão a voar, os motores e as hélices emitem um som específico, em que é possível treinar um modelo de dados para associar o mesmo a um modelo. [31]

Existem várias técnicas de deteção de *drones* através do sinal de comunicação, porém a mais popular é através da deteção da energia do mesmo, que foi o método escolhido para este projeto. Esta análise tem a vantagem de poder ser executada em equipamentos com baixa capacidade de processamento (como o *Raspberry PI* em [32]) e de não ser preciso treinar ou ter algum tipo de informação prévia acerca do sinal que se pretende detetar. A mesma consiste na análise da energia de um espectro de frequências e no desenvolvimento de um algoritmo que determina a presença do mesmo. Alguns exemplos de projetos que utilizam esta análise podem ser verificados nas seguintes referências: [5] [9] [32] [33].

2.6. GNU Radio

Como mencionado nos capítulos anteriores, existem aplicações que conseguem implementar sistemas e processar sinais utilizam um SDR, o que permite criar sistemas de rádio personalizados. Em diversos estudos, como em [5] e [8] exploraram a utilização do *software GNU Radio* para efetuar o varrimento de espectro com a deteção de energia, como mecanismo de análise.

O *GNU Radio* é um *software* gratuito e *open-source* que permite a criação de sistemas SDR para o processamento de sinais RF. O mesmo tem compatibilidade com os sistemas informáticos mais utilizado (*Linux, Windows e Mac OS*), porém a sua versão em *Linux* continua a ser a mais completa e fácil de configurar. Este projeto foi desenvolvido no sistema informático *Windows*.

O mesmo conta com o sistema *GNU Radio Companion*, que é um ambiente gráfico onde é possível implementar e desenvolver diferentes sistemas. O mesmo possui blocos com finalidades já definidas como filtros, equalizadores, desmoduladores, descodificadores,etc. Também existe a possibilidade de criar blocos novos para implementar uma ação especifica, ou simplesmente para acrescentar mais flexibilidade a blocos já existentes. Esses blocos denominam-se de *out-of-tree*, e podem ser programados em *Python* ou *C++* [34] [35].

Os blocos podem ser interligados para criar um sistema de processamento de sinais. Os mesmos podem-se agrupar em 3 grupos:

- *Source Block*- Blocos com output em que o seu objetivo é injetar informação no fluxograma. Um exemplo é o *osmocom Source*, que capta a informação recebida numa determinada gama de frequências, e depois "alimenta" o resto do fluxograma [36].

- *Sink Block*- Blocos que tem o objetivo de receber informação. Um exemplo é o *QT GUI Waterfall Sink*, que recebe um sinal e apresenta-o numa espectrograma [37].
- *Input* e *Output Block*- Blocos que recebem um conjunto de dados, transforma-os e devolve ao fluxograma. Um exemplo é o bloco *stream to vector*, que converte um fluxo de itens em um fluxo de vetores [38].

Os ficheiros criados pelo *GNU Radio Companion* tem a extensão .grc, e na sua execução é gerado um script *Python*.

2.7. Projetos Relacionados

Existem diversos projetos que têm como base a análise do espectro de frequências e deteção de sinais utilizados por equipamentos *drone*. Os mesmos ajudaram a compreender as soluções já existentes e a perceber, com diferentes ideias e métodos, qual seria a melhor e a mais eficaz abordagem para o desenvolvimento deste projeto.

Um dos exemplos verificados foi o projeto [32], que consiste numa solução em *GNU Radio* que é capaz de detetar sinais transmitidos por *drones*, através da deteção da energia no espectro verificado, e consequentemente bloquear os mesmos. O sistema divide-se em 3 etapas, em que na primeira é feita a varrimento do espectro sem a operação de um drone, para estabelecer valores base. O segundo é a comparação do sinal observado com os valores detetados nos sinais base, em que um *threshold* é calculado e são verificadas quantas amostras do sinal em questão estão acima do mesmo, bem como a percentagem das mesmas em relação às amostras totais e o valor mínimo e máximo das amostras acima do *threshold*. Esta informação é armazenada automaticamente num bloco de notas, onde posteriormente é analisado para concluir acerca da existência da utilização de um *drone*. A terceira consiste na transmissão de um sinal de ruído na frequência detetada pelo sistema, para interromper a comunicação do *drone*. Uma das limitações deste projeto é que a determinação da frequência utilizada pela comunicação *drone* é feita manualmente, analisando o conteúdo dos ficheiros de texto.

Outro exemplo é o projeto [33], que consiste no desenvolvimento de uma solução capaz de detetar a presença de *drones* e classificá-los. O mesmo divide-se em 2 partes, a primeira é a deteção de energia, em que é verificado se um *drone* está a operar no ambiente e a segunda é análise aos sinais OFDM que são utilizados pelos *drones* na sua comunicação, de modo a construir um modelo de *machine learning* com as características dos mesmos. A existência de

um drone é considerada caso média das amostras da potência do sinal for maior que um *threshold*. Este projeto tem como objetivo principal classificar os *drone* em operação, determinando a marca e modelo do equipamento.

CAPÍTULO 3

Arquitetura e Implementação do Sistema

Este capítulo consiste na descrição do desenvolvimento do sistema, bem como todos os componentes e ferramentas utilizadas.

3.1. Arquitetura do Sistema

O sistema divide-se em 5 partes, sendo a primeira o *HackRF*, que é o SDR utilizado, onde no mesmo já consta a antena utilizada. O segundo é o *Gnu Radio*, que é o software que é utilizado para efetuar ações com o SDR. O próximo é o cálculo da energia do espectro, onde o sinal é transformado. A quarta parte é o *Spectrum Detection*, o script *Python* responsável pela decisão relativa à sua presença. A última parte é o terminal, que é onde o resultado do sistema é apresentado. A figura 3 pode-se observar um diagrama de blocos do sistema.

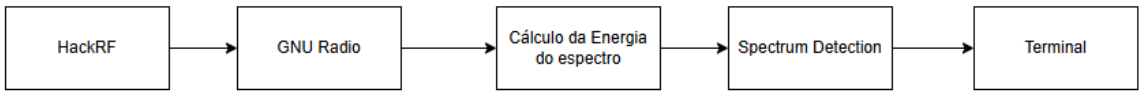

Figura 3- Diagrama de blocos do Sistema

3.2. Materiais e Ferramentas

Neste subcapítulo são referidos os materiais e ferramentas utilizadas ao longo do projeto, como *o SDR HackRF*, o *switch Opera Cake*, as antenas e o *drone* utilizado nos testes.

3.2.1. HackRF One

O *HackRF One* [39] é um equipamento SDR desenvolvido pela *GrandScott Gadgets*, com uma capacidade de operação entre 1 MHz e 6GHz, com uma potência de transmissão de 15 dBm. O ganho e transmissão e receção, tanto como o filtro da banda base podem ser configuradas através de um software. Possui uma porta SMA conectar a antena, uma porta USB 2.0 para a conexão e alimentação através de um computador e entradas 3 saídas *clock*.

A potência de transmissão varia em relação à frequência, observável na tabela 4:

Tabela 4- Potência de transmissão para diferentes gamas de frequência [53]

Intervalo de Frequências	Potência máxima de transmissão
10 MHz - 2150 MHz	5 dBm -15 dBm
2150 MHz - 2750 MHz	13 dBm - 15 dBm
2759 MHz- 4000 MHz	0 dBm - 5dBm
4000 MHZ - 6000 MHz	-10 dBm - 0dBm

Em relação à potência de receção, o fabricante sugere que o máximo seja -5dbm, pelo que se exceder esse valor pode danificar o equipamento. Teoricamente o *HackRF One* conseguiria receber sinais até 10 dbm, desde que não fosse aplicado nenhum amplificador na receção.

Este equipamento opera em *Half-duplex*, o que significa que consegue receber e transmitir um sinal, mas não ao mesmo tempo, o que é um dos fatores para este equipamento ter sido escolhido para este projeto. Outros fatores foram o facto que conseguir operar entre os 1 e 6 GHz e ser uma opção monetariamente mais acessível, com um preço entre os 300-400€.

Este foi o SDR escolhido pois é a melhor opção em relação a qualidade/preço que consegue abranger todo espectro analisado neste projeto. A placa SDR pode ser observada na figura 4.

Figura 4- SDR HackRF One [39]

3.2.2. Antenas

No projeto foram utilizadas duas antenas para a receção e o consequente varrimento do espectro de frequências, sendo as mesmas uma antena omnidirecional, e uma antena direcional.

A antena omnidirecional consegue operar em 3 gamas de frequências, 2,4 GHz, 5 GHz e 6 GHz, porém o rendimento nas duas últimas não é o melhor, pelo que os resultados neste projeto focam-se mais na gama dos 2,4 GHz. Apenas foi feito um teste na gama dos 5 GHz, porém a mesma apresentou limitações na captura, pelo que o transmissor (*drone*) tinha de estar muito perto da antena. A antena possui uma impedância de 50 Ohm e um ganho de 5 dBi. [40]

Na figura 5 pode-se observar a antena omnidirecional utilizada.

Figura 5- Sistema com a antena Omnidirecional conectada

A antena direcional é constituída por uma placa de circuito impresso, operacional apenas na gama de 2,4 GHz. A mesma tem um ganho de 10,5 dB. [41]

Na antena pelo facto de ser direcional, para alem de ter um ganho superior à antena omnidirecional, também capta menos ruido, conseguindo obter melhores resultados. A única desvantagem é que a mesma tem de estar a apontada ao equipamento que pretendemos detetar.

A figura 6 representa a antena direcional utilizada no projeto.

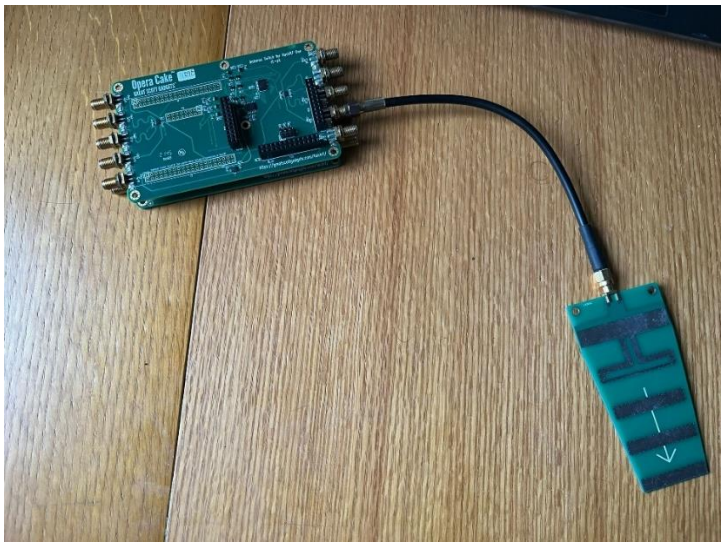

Figura 6- Sistema com a antena Direcional conectada

A antena tem um papel importante no projeto, visto que têm de garantir a captura dos sinais com clareza e precisão. A performance do sistema depende diretamente do desempenho da antena, sendo que o mesmo apresentará melhores resultados quanto maior for a potência do sinal detetado, desde que a mesma não seja muito elevada.

Para este projeto não foi possível utilizar uma antena que conseguisse garantir a captura dos sinais com clareza nas gamas de frequência 5 GHz e 5,8 GHz.

3.2.3. Opera Cake

Como mencionado anteriormente, a comunicação estabelecida entre o controlador e os *drones* pode ser feita em diversas gamas de frequências, sendo as gamas de 2,4GHz e 5,8 GHz as mais utilizadas. Para fazer o varrimento do espectro em diferentes gamas de frequências foi utilizado um equipamento chamado *Opera Cake*.

O *Opera Cake* é um *switch* de antenas que se pode acoplar a um Hack RF, fabricado pela Great Scott. O mesmo é constituído por 2 1x4 *switches*, porém também dá para **conectá-los** entre si, tornando-se num 1x8. O mesmo divide-se em 2 lados, onde existem 4 portas SMA fêmeas. No lado "A" as portas estão organizadas entre A0-A4, e no lado "B" entre B0-B4", sendo a porta A0 e B0 primarias e as restantes secundárias. Através de *software* (comando *hackrf_operacake)* é possível configurar as portas utilizadas. Através desse comando podemos associar as portas secundárias a uma porta primária, sendo que a primária está conectada à porta da antena do *HackRF* através de um cabo SMA. Com isso a *HackRF* irá assumir o que está programado no *Opera Cake*. O *Opera Cake* pode funcionar em 3 modos: Manual, Frequência e Tempo. O modo manual é como o nome indica, a troca manual da porta; No modo frequência

a porta A0 altera a sua conexão automaticamente para a porta programada tendo em conta a frequência onde o sistema está a operar; o modo tempo altera a conexão de A0 tendo em conta um tempo programado.

Este equipamento é uma excelente solução para quando se quer operar em diversas gamas de frequência, sendo possível configurar tudo por software em vez de trocar as antenas manualmente [42]. Na figura 7 pode-se observar este equipamento.

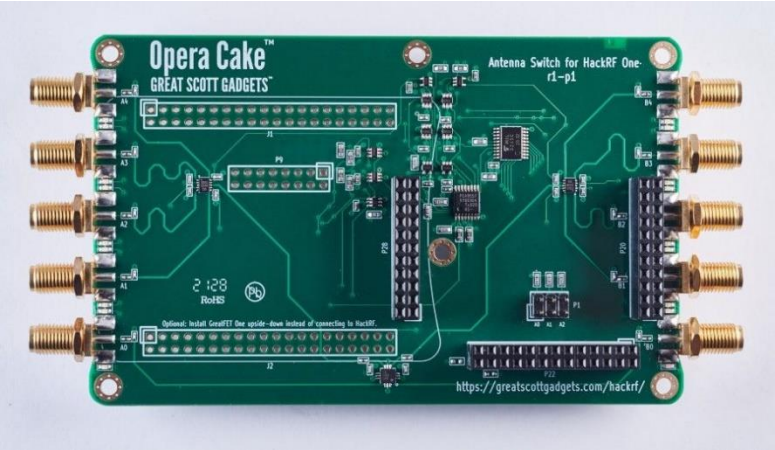

Figura 7- Opera Cake [42]

3.2.4. Parrot BEBOP 2

Para a execução dos testes foi utilizado um *drone BeBop 2* desenvolvido pela empresa *Parrot* [43] e lançado no dia 14 de Dezembro de 2015. O mesmo é um quadricóptero (quatro hélices), com uma autonomia de 25 minutos e tem um alcance de 300 metros quando operado por um *smartphone* e até 2 Km com o controlador *SkyController*. O equipamento possui uma câmara de 14 *Megapixels*, capaz de gravar em *Full HD* (1080p x 1920p) e 8 Gb de memória para armazenar os mesmos. Em relação à transmissão, o mesmo opera através de sinais WI-FI nas gamas de frequência de 2,4 e 5 GHz com uma potência até 21 dBm [44]. A seleção do canal utilizado para o sinal por definição é automática, mas é possível manualmente trocar o canal.

Na gama de 2,4 GHz o mesmo consegue operar nos canais 1-11, e em 5 GHZ nos canais 36-48. O *drone* também conta com um modulo GNSS (*Global Navigation Satellite System*) para geolocalizar e medir a velocidade da aeronave. Para o controlo é utilizada a app *FreeFlight* PRO, em que o *drone* atua como um *Access Point* e cria a sua própria rede Wi-Fi, em que o telefone tem de estar conectado. A partir dai é possível controlar o *drone*, alterar as definições de transmissão e ver os vídeos que o mesmo gravou. Em relação ao *SkyController*, o mesmo

possui um sistema operativo Android com a app *FreeFlight* integrada, conectando-se da mesma maneira que por um *smartphone* [45]. Na figura 8 pode-se observar este equipamento.

Figura 8- Parrot Bebop 2 [44]

3.2.5. FreeFlight Pro

A *FreeFlight Pro* é uma aplicação mobile desenvolvida pela *Parrot* que é utilizada para pilotar os *drones* da gama Bebop e Disco. A aplicação tem uma interface intuitiva, permitindo controlar a aeronave em tempo real e ter acesso a informações acerca do voo. Na mesma é possível ajustar os parâmetros de voo como inclinações, velocidades, altura e distância máximas. Na interface de voo, a mesma apresenta valores de telemetria como a altura, distância, velocidade, capacidade da bateria, etc. Também é possível controlar a aeronave através da aplicação. A figura 9 representa a interface de controlo do drone.

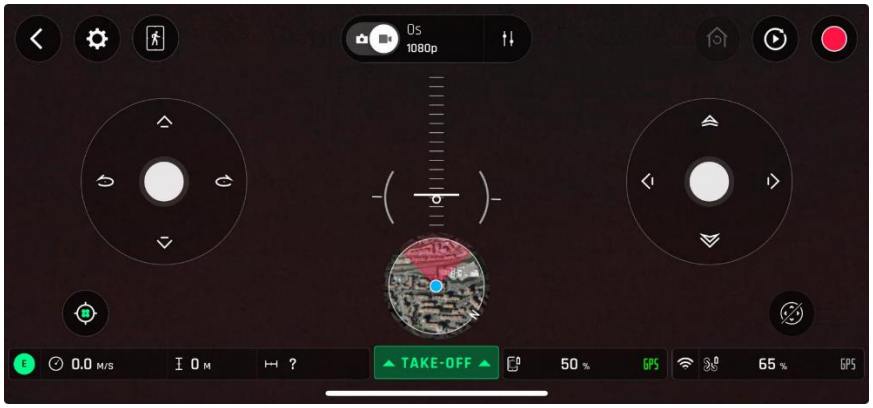

Figura 9- Interface de controlo do drone da aplicação FreeFlight Pro

Através do *FreeFlight Pro* é possível gravar o voo em alta-definição (1080p). Também é possível alterar alguns parâmetros da captura de vídeo, como a resolução, a frequência de imagem(fps)…

Em relação à comunicação com o *drone*, a aplicação possui uma interface onde é possível determinar a localização do *drone* e a gama de frequência em que a comunicação se pode estabelecer. Existem 4 seleções possível, sendo a primeira "Todos", em que a comunicação vai ser estabelecida no melhor canal possível, tendo em conta a potência do sinal e o ruido presente nas gamas de frequência. A segunda e terceira opção é a também a seleção do melhor canal possível, mas apenas na gama de 2,4 e 5GHz, respetivamente. A quarta opção é a seleção manual do canal, onde o utilizado pode escolher em que canal quer estabelecer a conexão. Na interface também está presenta a opção "Exterior", em que caso a mesma seja selecionada apenas é possível utilizar um canal na gama de 2,4 GHz para a comunicação. Isso deve-se ao facto de a mesma possuir uma capacidade de alcance maior comparada com a gama dos 5GHz, o que pode ser relevante para voos com distâncias maiores. A interface das definições de rede pode ser observada na figura 10.

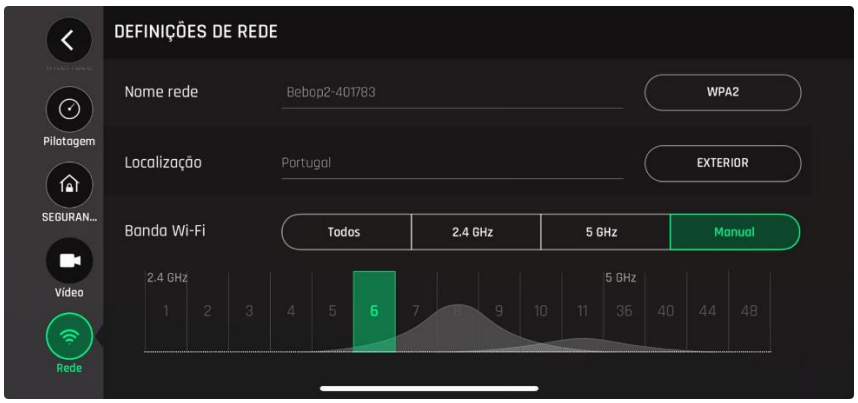

Figura 10- Interface das definições de rede da aplicação FreeFlight Pro

Com a aplicação é possível planear voos, estabelecendo uma rota pré-definida, porém esta funcionalidade tem um custo adicional.

O *FreeFlight Pro* tem compatibilidade com controladores, como o *SkyController* mencionado anteriormente, possibilitando voos mais precisos e com distâncias maiores.

3.3. Implementação do Sistema

O principal objetivo do projeto é conseguir determinar se existe um *drone* a operar na área onde se está a efetuar o teste. Para isso foi desenvolvida uma solução que faz o varrimento do espectro de frequências mais utilizado por estes equipamentos. O método de deteção escolhido foi a deteção de energia, que como especificado anteriormente, baseia-se na determinação da energia de uma gama de frequência e compará-la com um valor limite estabelecido, ou verificar se é um valor irregular em relação à energia verificada na gama sem a presença de um *drone*. A figura 11 representa o fluxograma do sistema de deteção no Gnu Radio:

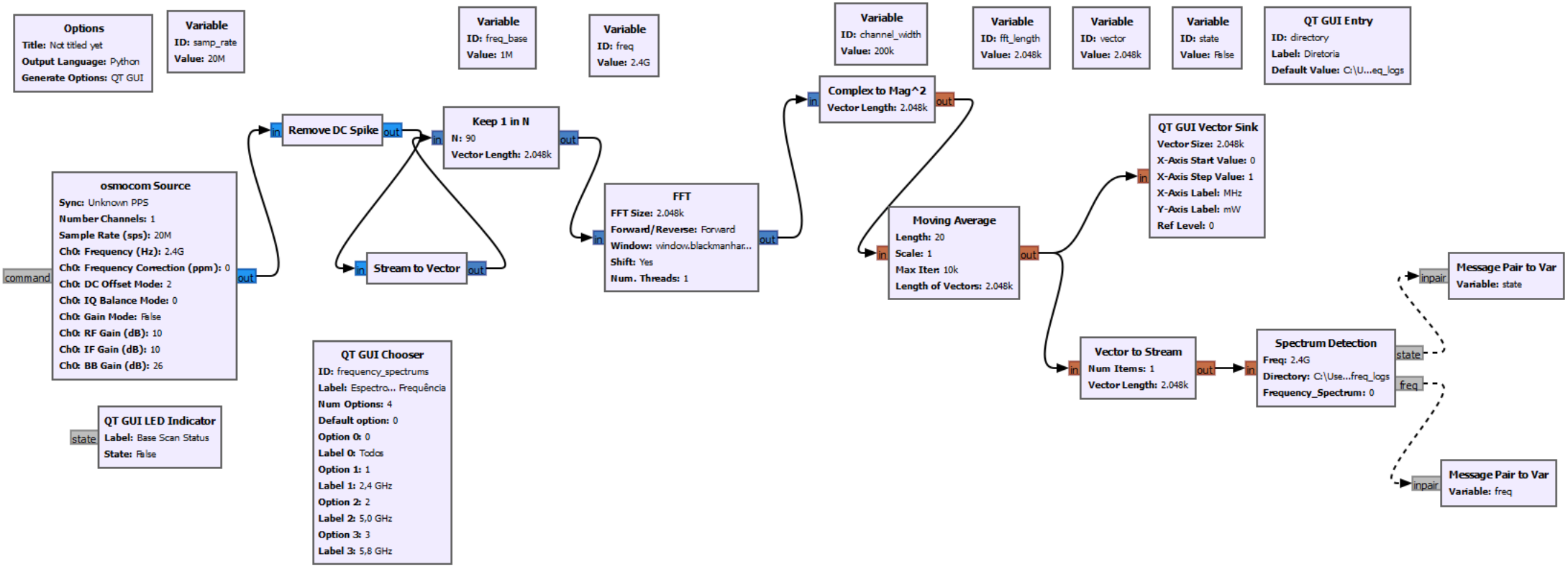

Figura 11- Fluxograma do sistema de deteção no Gnu Radio

Como verificado na figura 11, o sistema começa com o bloco *osmocom Source*, que permite a integração do SDR com o GNU Radio. Através do mesmo consegue-se captar a sequência de amostras proveniente do *HackRF*. O mesmo faz parte do projeto *gr-osmosdr*, que é um software *open-source* para sistema de comunicação móvel. O bloco tem bastantes parâmetros, mas os mais importantes é a taxa de amostragem (que determina o comprimento da gama de frequência observada), a frequência central e os ganhos RF, IF e BB.

O ganho RF tem como objetivo amplificar o sinal recebido da antena antes de qualquer processamento, o que melhora a receção de sinais mais fracos, porém amplifica também o ruido. O ganho IF amplifica o sinal numa frequência intermédia, onde o mesmo é mais estável, o que pode melhorar a qualidade do sinal sem amplificar tanto ruido como o ganho anterior. O ganho BB amplifica o sinal de banda base, onde pode ser utilizado para calibrar o sinal como etapa final. A taxa de amostragem utilizada no sistema é 20 Msps e o ganho RF, IF e BB são 10 dB, 10 dB, 26 dB, respetivamente. Estes valores foram escolhidos depois de uma análise em que a presença de um sinal fosse inequívoca, sem aumentar muito o ruído.

De seguida, a informação captada pelo *osmocom source* passa pelo bloco *Remove DC Spike*, que remove o pico de energia na frequência central IQ DC. O *HackRF* é um sistema de amostragem em quadratura, o que significa que todas as amostras são IQ (*Quadrature Signals*), e por esse motivo aparece um pico de energia na frequência central. Na maioria das aplicações, e inclusive no *GNU Radio*, o mesmo não tem impacto nenhum, e pode-se ignorar, mas por uma questão de apresentação foi adicionado este bloco. [46]. O mesmo utiliza um filtro IIR (*Infinite Impulse Response*), que remove esse pico, não alterando nenhum componente do espectro observado. [47]

O próximo bloco no sistema é o *Stream to Vector*, que converte um fluxo de dados em um vetor com 2048 amostras. O mesmo é necessário para um dos próximos blocos que o FFT, necessita de receber um vetor de amostras. [48]

De seguida encontra-se o bloco *Keep 1 in N*, que como o nome indica, reduz a taxa de amostragem do sistema, mantendo apenas 1 amostra (neste caso um vetor) a cada 90. O sistema sem este bloco, para além de necessitar de um grande poder computacional, provoca "*overruns*" no GNU Radio, o que queria dizer que o mesmo não conseguia processar a informação em tempo-real. De modo a estabilizar o sistema, foi implementado este bloco, o que recebe o conjunto de vetores de amostras, e retorna 1 a cada 90. [49]

O próximo bloco no sistema é o bloco FFT, que calcula a transformada rápida de *Fourier*. O mesmo é utilizado para transformar o sinal do domínio do tempo para o domínio da frequência. O bloco foi configurado para utilizar 2048 amostras em cada execução, e o tipo de janela (*window*) aplicada a cada amostra antes da FFT é a *Blackman-Harris*. (A *window* serve para minimizar os efeitos de ponta e prevenir uma fuga espectral). [50]

De seguida é utilizado o bloco *Complex to Mag2* com o intuito de calcular a potência de cada amostra resultante da FFT. A magnitude de cada amostra é expressa pela seguinte função:

$$(c[n] = \sqrt{r[n]^2 + i[n]^2}) \quad (3.1)$$

A magnitude ao quadrado, que equivale à energia de cada amostra tem a seguinte fórmula:

$$c[n]^2 = r[n]^2 + i[n]^2 \quad (3.2)$$

O bloco retorna um vetor com a potência das amostras recebidas em watts. [51]

O resultado do bloco anterior passa pelo bloco *Moving Average*, que aplica uma média movel (*Moving Average*) ao sinal de entrada. O mesmo conta com os parâmetros comprimento, escala, iterações máximas e comprimentos das amostras. O comprimento define quantas amostras são utilizadas para calcular a média. O valor utilizado no projeto é 20, o que permite uma suavização do sinal sem causar um grande atraso no mesmo e sem amplificar os sinais de ruído. A escala é utilizada para multiplicar com o valor médio, em que é utilizado o valor 1. A iteração máxima corresponde a um limite máximo de amostras que são utilizadas numa execução. O bloco tem como objetivo suavizar o sinal, ao controlar as variações rápidas, e reduzir o ruído. O limite de iterações utilizado é 10000. O comprimento das amostras é o comprimento dos sinais de input, neste caso 2048. Com os parâmetros, o bloco retorna o resultado da seguinte fórmula:

$$Output[i] = scale * \sum input[i - length: i]. \text{ [52]} \quad (3.3)$$

Neste ponto do sistema existe uma bifurcação do fluxo, sendo que o resultado do último bloco vai para o bloco *QT GUI Vector Sink*, que apenas representa graficamente o vetor recebido. Este bloco serve apenas para o utilizador ter uma noção visual da potência verificada em cada gama de frequências. O segundo caminho vai fazer a verificação da potência verificada no espectro, para concluir se existe um *drone* a operar no mesmo.

O vetor é convertido novamente para um fluxo de amostras pelo bloco *Vector to Stream*, e passa pelo bloco *Spectrum Detection*. O *GNU Radio* tem a possibilidade de criação de blocos personalizados, como mencionado em capítulos anteriores. O *Spectrum Detection* é um bloco personalizado, desenvolvido em *Python*, que faz a verificação da potência detetada e conclui em relação à existência de um *drone*.

Para conseguir fazer o varrimento das duas gamas de frequência mais utilizadas por *drones* (2,4, 5 e 5,8 GHz), foi utilizado o *switch Opera Cake*, mencionado em capítulos anteriores. O mesmo opera no modo de frequência, onde entre os 0-3GHz o *HackRF* utiliza a antena da porta A2, e entre 3-6 GHz é utilizada a da porta B1.

Durante o varrimento são efetuados saltos de frequência, onde no bloco *Spectrum Detection* são enviadas mensagens internas com a chave correspondente à frequência, e com o valor atualizado. O bloco *Message Pair to Var* recebe essa mensagem e converte o valor para a variável correspondente no GNU Radio. O mesmo também é feito para a variável "*state*" que tem o intuito de assinalar quando o varrimento da potência base acabou.

3.3.1. Spectrum Detection

O bloco *Spectrum Detection* é o bloco responsável por analisar a presença de um *drone*. O mesmo tem como entrada um fluxo de dados com um comprimento de 2048 amostras. Como parâmetros tem a frequência, a diretoria para o armazenamento temporário de ficheiros e uma variável para a seleção das gamas de frequência que se pretende analisar.

A execução do bloco divide-se em 2 passos, a verificação da potência base do espectro e a verificação da existência de comunicações de *drones*. Durante a execução do script são executados saltos de frequência de modo a percorrer toda a gama de frequências. Por definição o *HackRF* consegue operar em uma largura de banda de 20 MHz, porém durante os testes foi observado que perto dos limites (-10/+10 MHz da frequência central) a análise espectral apresentava algumas falhas, pelo que foi optado por fazer saltos de 10 MHz, com análises com 20 MHz de largura de banda, de modo a garantir a inexistência de pontos "mortos". Desta forma, cada análise de gama de frequência cobre uma pequena parte da análise anterior e da seguinte, permitindo confirmar a continuidade dos resultados ao longo das frequências.

Baseado na solução apresentada por [32], no início de cada execução do sistema é feita uma análise do espectro sem a presença de *drones*. A mesma serve para ter uma referência de

possíveis sinais que possam estar a atuar na gama de frequência analisada, e assim detetar a presença de ruido e ajustar a análise tendo em conta essa informação. Durante esta etapa, são criados ficheiros de texto na diretoria que é descrita como parâmetro, para cada salto de frequência, com o nome "base_power_(Frequência Base)MHz". O sistema divide a gama de frequência em intervalos de frequência com saltos de 10 MHz, analisando o espectro base durante 10 segundos para cada intervalo. No respetivo ficheiro, é escrito o valor médio da potência detetada em cada amostra. Durante 10 segundos o ficheiro base atinge em média em média um tamanho de 11 MB, com 1850 análises feitas, o que dá 185 análises por segundo.

Pelo facto de gamas de frequência utilizadas por equipamentos *drone* serem bastante utilizadas é normal, em meios urbanos, que haja algum ruido, principalmente na gama de 2,4 GHz, e, portanto, esta solução foi desenvolvida para ter em conta os sinais "habituais" do ambiente. Durante a execução do sistema, pode-se escolher que gama se pretende analisar, existindo as opções 2,4 GHz, 5GHz, 5,8 GHZ e todas.

Ao terminar a primeira parte, inicia-se a próxima etapa, que é análise do espectro para determinar a existência de *drones*.

O sistema começa por executar a função *estimate*, que tem o objetivo de detetar séries de pontos na sequência de amostras que ultrapassam um dado *Threshold*, com interrupções mínimas. Ao ser detetada uma série com esses requisitos, a mesma é capturada até haver uma interrupção de vinte amostras com um valor abaixo do *Threshold*. Apos isso, caso seja detetada outra série, a função considera a que tem o maior comprimento. Caso a série tenha mais de 300 valores, o primeiro valor da série é o limite inferior e o último o limite superior da serie detetada. Caso a média dos valores da série seja maior do que $1{,}5 * \text{Threshold}$, a função é novamente chamada (em recursividade) com o dobro do valor de *Threshold* utilizado. Isto é feito para ajustar a deteção à potência do sinal detetado, conseguindo assim determinar os limites da série com mais detalhe. A função retorna os limites da série detetada (caso não detete nenhuma série, os limites são *None*)

Após a execução da função, é armazenado o valor médio da energia das amostras e caso existam, os limites determinados na função anterior. Esta informação é armazenada num ficheiro auxiliar na diretoria selecionada com o nome "power_(Frequência Base)MHz". Ao fim de 1 segundo é calculada a percentagem de amostras que registaram um possível sinal e é comparada com a mesma percentagem na análise base. Caso haja uma discrepância nos valores, ou a percentagem calculada seja maior que 85% e a percentagem base maior que 60% (zonas com interferência, e, portanto, as diferenças entre as duas percentagens podem nem sempre

apresentar discrepâncias notáveis), é assumido que existe um possível sinal "anormal" naquela gama. Esta análise é feita para todas as divisões de frequência da gama analisada.

No final da análise anterior, é determinada a conclusão acerca da existência de um sinal de comunicação *drone*. Para todas a divisões de frequência em que foi assumida a possível existência de um sinal *drone* (resultado da comparação das percentagens), é executada a função *getLimits*. A mesma acede à informação dos ficheiros auxiliares criados para essas divisões de frequência, e retorna a mediana dos limites inferiores e superiores das séries detetadas (mediana dos resultados da função *estimate*). Com a execução da função para todas as divisões de frequência em que foi assumida a existência de um sinal *drone*, são adquiridos todos os limites das mesmas, que serão utilizados na função *calcLimits*.

A função *calcLimits* tem como objetivo determinar a presença de um sinal que possa ser assumido como um sinal anormal de um *drone*. A mesma executa uma série de verificações para verificar se existe uma série de divisões que descrevem a existência de um *drone*. Essas verificações foram desenvolvidas através da verificação de padrões e comportamento de sinais de comunicação *drone*, tendo em conta possíveis sinais de ruido presentes no espectro. A função retorna, caso detete um sinal *drone*, o limite inferior e superior da largura de banda do mesmo.

Nesse caso, é determinado o canal *Wi-Fi* onde o sinal se encontra, através da função "*find_closest_channel*".

No final, o sistema escreve na consola a conclusão da análise, em que caso seja determinado um sinal aparecerá a seguinte mensagem: Sinal com largura de banda de (Largura de banda do sinal) MHz foi encontrado no canal (número do canal) no range de (gama de frequência analisada) GHz entre as frequências (Limite inferior) e (Limite superior). Caso nenhum sinal seja encontrado, a mensagem é a seguinte (Nenhum sinal encontrado na gama dos (gama de frequência analisada). Caso seja detetado sinais de ruido no espectro analisado, a mensagem tem a seguinte descrição: "porém foi detetado ruido perto das gamas de (gamas onde foi detetado ruido)" Esta mensagem foi adicionada pois no caso de um sinal *drone* estar a operar numa zona com ruido, os resultados podem não ser muito precisos, e, portanto, é necessário ter em conta as condições da gama observada. Na figura 12 pode-se verificar o fluxograma do sistema:

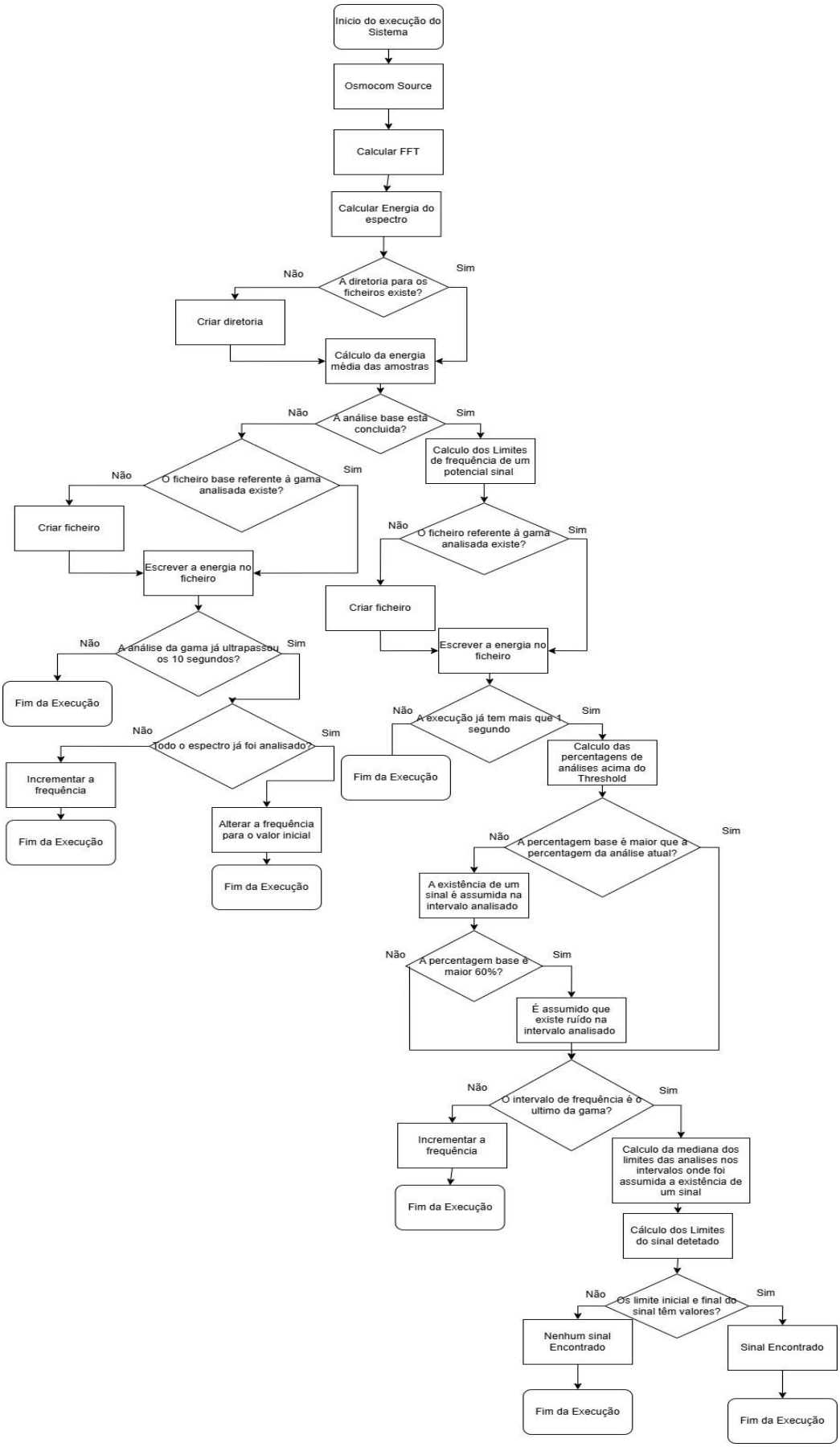

Figura 12- Fluxograma do Sistema

3.3.2. Funcionamento do Sistema

Depois da explicação do sistema, este subcapítulo descreve a utilização do sistema.

Primeiramente, antes de correr o sistema, é necessário configurar o *Opera Cake*, sendo necessário anexá-lo ao *HackRF*, e correr os seguintes comandos:

"hackrf_opercake -a A2 -a B1"

Este comando estabelece a conexão entre a portas do OperaCake A2, que é onde está a antena para os 2,4 GHz, e B1, onde está a antena para as gamas de 5 e 5,8 GHz, à porta primária A1. A porta A1 está ligada por um cabo SMA à porta da antena do HackRF. De seguida é necessário executar o comando:

"hackrf_operacake -m frequency -f A2:0:3000 -f B1:3000:6000"

O mesmo configura o *opekacake* para caso esteja a operar numa frequência ente 0 e 3000 MHz, utilizar a antena da porta A2, e no caso de operar numa frequência entre 3000:6000, utilizar a antena da porta B1.

Com o *Opera Cake* configurado, o sistema pode iniciar-se. Ao iniciar o sistema aparecerá o *prompt* da figura 13:

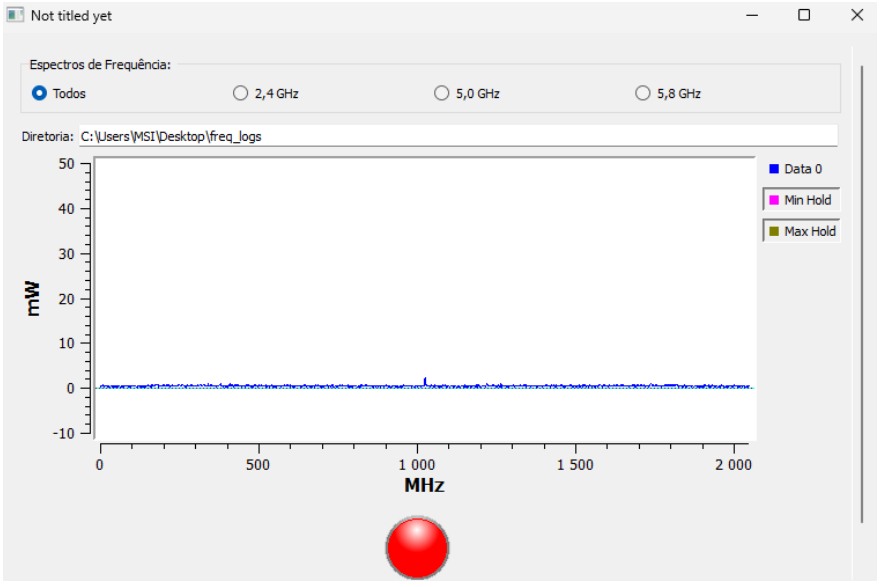

Figura 13- Prompt do Sistema

No mesmo pode-se alterar e observar alguns detalhes acerca da análise. O primeiro campo é a seleção dos espectros de frequência que se pretendem analisar. No campo seguinte, a "Diretoria", é onde os ficheiros auxiliares irão ser armazenados. Caso a diretoria não exista, a mesma será criada. Caso existam ficheiros na mesma, eles serão apagados. O gráfico central representa a energia do espectro verificado, representando as amostras utilizadas na análise. Cada gráfico representa a energia a cada 20 MHz, em que o vetor com o índice 1024 representa a energia da frequência central da análise, como se pode verificar na seguinte equação:

$$(\frac{FFT\ Length}{2} = \frac{2048}{2} = 1024) \qquad (3.4)$$

Por fim, existe um LED que indica se a verificação base já foi concluída. Caso a mesma esteja a decorrer, o LED tem a cor vermelha, caso contrário, o LED irá ter cor verde.

Quando a análise base termina, o resultado do sistema é retornado no terminal, em que caso nenhum sinal seja encontrado aparecerá a informação que se pode observar na figura 14:

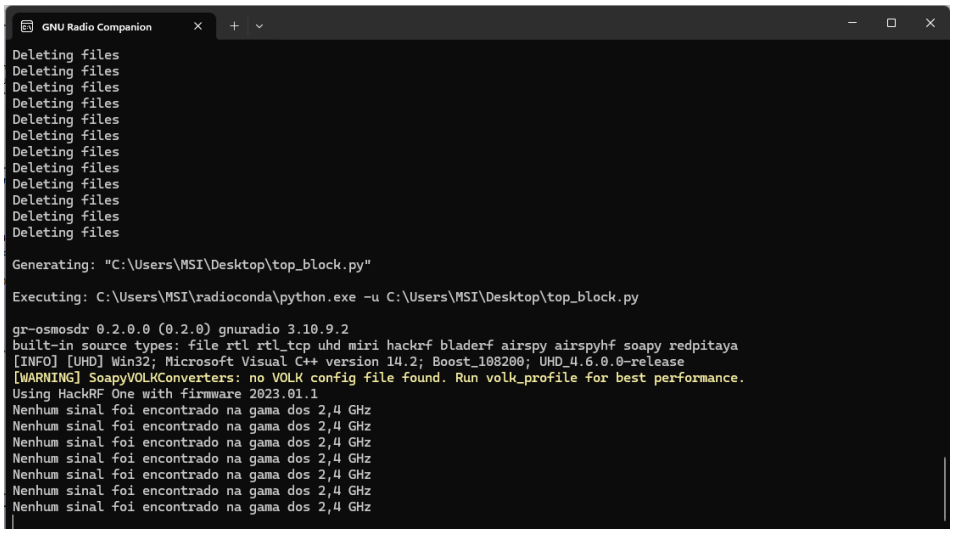

Figura 14- Terminal em que nenhum sinal foi encontrado

Caso detete um sinal, a mensagem retornada é a que se apresenta na figura 15:

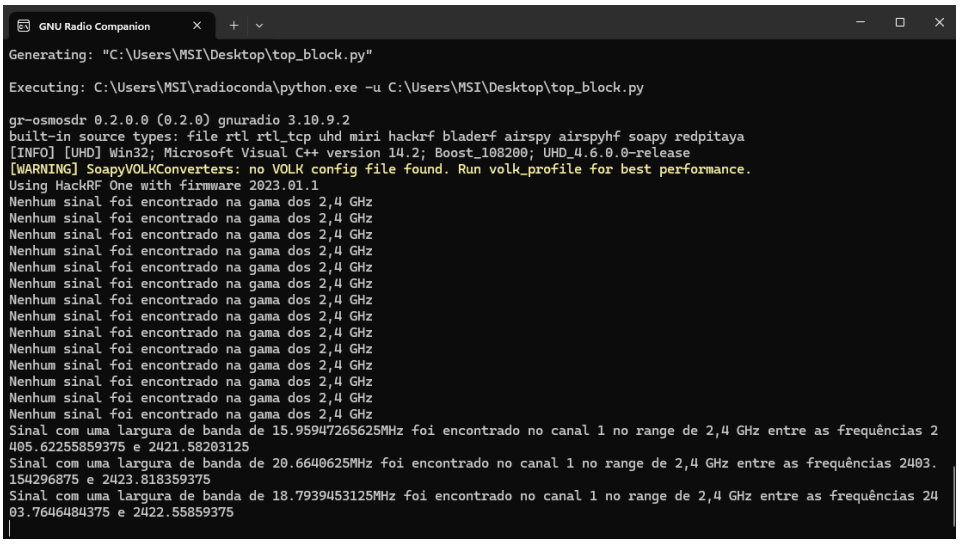

Figura 15- Terminal com um sinal encontrado

CAPÍTULO 4

Testes e Resultados

Este capítulo apresenta os testes e resultados do sistema sob diferentes ambientes e condições. Nos mesmos foi avaliado o desempenho e eficiência do sistema para os cenários distintos, com métricas de performance aplicadas aos resultados. Todos os testes tiveram uma duração média de 5 minutos.

Nos testes apenas foi possível testar a gama de frequências de 2,4 GHz, pois não foi possível adquirir uma antena que conseguisse captar sinal nas restantes gamas com consistência. Por isso o tempo de execução da análise base e do espectro são inferiores comparado com um varrimento completo. É estimado que a análise base da gama de 2,4 GHz tenha uma duração de 1 minuto, enquanto nas 3 gamas demoraria perto de 5 minutos. A análise do espectro após a base, na gama de 2,4 GHz tem uma duração prevista de 10 segundos, enquanto no varrimento das 3 gamas, a mesma seria de 50 segundo (porém o sistema apresentaria resultados à medida que ia analisando cada gama)

4.1. Testes Controlados

O objetivo dos testes controlado é testar o sistema num ambiente com condições estáveis e previsíveis, tentando mitigar ao máximo a influência de fatores externos.

4.1.1. Condições

Os testes foram executados num ambiente remoto, sem a presença de outros sinais RF. Foi utilizada uma placa *HackRF* como transmissor de modo a garantir um sinal sempre estável.

O sinal foi emitido no canal 6 (2427-2447 MHz) com o objetivo de testar a capacidade de identificar e determinar os canais de transmissão através da deteção do sinal RF emitido pelo transmissor. Os testes foram realizados em diferente distância entre o recetor e o transmissor, sendo estas 50 cm, 5 metros e 16 metros. Nos testes foram utilizadas duas antenas, uma omnidirecional e uma direcional. Pode-se observar o ambiente onde foram realizados os testes com pouco interferência na figura 16.

Figura 16- Ambiente dos testes controlados

4.1.2. Resultados

a) Sem presença de um sinal RF com a antena Omnidirecional no ambiente remoto

Neste teste foi analisado a deteção de sinais *drone* caso não exista nenhum sinal a operar. O objetivo do mesmo é analisar o número de "falsos positivos" num ambiente controlado. Na figura 17 pode-se verificar o sistema.

Figura 17- Hardware do sistema

Foram realizadas 32 análises, durante 5 minutos, o que significa que foram feitas 6,4 análises por minuto e que cada análise em média demorou 9,375 segundos. Das 32 análises, todas concluíram que nenhum sinal foi detetado.

b) *HackRF* a 50 centímetros com antena Omnidirecional

Neste teste foi analisado o desempenho e robustez do sistema com a presença de um sinal transmitido por um *HackRF* a 50 cm do mesmo, utilizando a antena omnidirecional. O objetivo do teste é determinar a taxa de acerto em relação à presença de um sinal, e a determinação do canal utilizado.

Foram realizadas 32 análises, durante 5 minutos, o que significa que foram feitas 7 análises por minuto e que cada análise em média demorou 9,37 segundos. Das 32 análises, 6 detetaram um sinal no canal 6, o que corresponde a uma percentagem de 18,75%. Vinte e cinco análises detetaram um sinal num canal adjacente, sendo que vinte e quatro detetaram um sinal no canal 5 e uma no canal 7, o que corresponde a uma percentagem de 78,13%. Uma análise determinou a existência de um sinal no canal 4, o que corresponde a uma percentagem de 3,12%. Todas as análises determinaram a existência de um sinal.

c) *HackRF* a 5 metros com antena Omnidirecional

Neste teste foi analisado o desempenho do sistema na presença de um sinal emitido por uma *HackRF*, que se encontrava a uma distância de 5 metros. A antena utilizada para transmitir o sinal foi a omnidirecional. O objetivo do teste é determinar a taxa de acerto em relação à presença de um sinal, e a determinação do canal utilizado, comparando com os resultados dos testes anteriores.

Foram realizadas 27 análises, durante 5 minutos, o que significa que foram feitas 5,4 análises por minuto e que cada análise em média demorou 11,11 segundos. Das 27 análises, 25 detetaram um sinal no canal 6, o que corresponde a uma percentagem de 92,6%. Uma análise detetou um sinal num canal adjacente, que foi no canal 5, o que corresponde a uma percentagem de 3,7%. Apenas em uma análise se concluiu que nenhum sinal estava a operar, correspondendo a uma percentagem de 3,7%.

d) *HackRF* a 50 centímetros com antena direcional

Neste teste foi analisado o desempenho e robustez do sistema com a presença de um sinal transmitido por um *HackRF* a 50 cm do mesmo, utilizando a antena direcional. O objetivo do teste é determinar a taxa de acerto em relação à presença de um sinal, e a determinação do canal utilizado.

Foram realizadas 32 análises, durante 5 minutos, o que significa que foram feitas 7 análises por minuto e que cada análise em média demorou 9,37 segundos. Das 32 análises, 18 detetaram um sinal no canal 6 (2427-2447MHz), o que corresponde a uma percentagem de 56,25%. Doze análises detetaram um sinal num canal adjacente, sendo que onze detetaram um sinal no canal 5 e uma no canal 7, o que corresponde a uma percentagem de 37,5%. Duas análises determinaram a existência de um sinal no canal 4, o que corresponde a uma percentagem de 6,25%. Todas as análises determinaram a existência de um sinal.

e) *HackRF* a 5 metros com antena direcional

Neste teste foi analisado o desempenho e robustez do sistema com a presença de um sinal transmitido por um *HackRF* a 5 metros do mesmo, utilizando uma antena direcional. O objetivo do teste é determinar a taxa de acerto em relação à presença de um sinal, e a determinação do canal utilizado.

Foram realizadas 33 análises, durante 5 minutos, o que significa que foram feitas 6,6 análises por minuto e que cada análise em média demorou 9,09 segundos. Das 33 análises, 17 detetaram um sinal no canal 6, o que corresponde a uma percentagem de 51,5%. Onze análises detetaram um sinal num canal adjacente, sendo que dez detetaram um sinal no canal 5 e uma no canal 7, o que corresponde a uma percentagem de 33,3%. Em cinco análises foi determinada a existência de um sinal nos canais 4, correspondendo a uma percentagem de 12,12%. Todas as análises determinaram a existência de um sinal.

Este teste apresentou uma melhoria na percentagem de análises no canal correto em relação ao teste anterior, dado que o sinal recebido pelo sistema tem uma potência elevada a 50 centímetros, o que faz com que o ruído à sua volta e as réplicas também tenham uma potência mais elevada. Isso faz com que o sistema determine o limite inferior da largura de banda um

pouco inferior ao real, o que afeta na determinação do canal, o que explica as 11 análises que determinaram um sinal no canal 5 no teste anterior.

f) *HackRF* a 15 metros com antena direcional

Neste teste foi utilizada uma antena direcional no recetor e analisado o desempenho do sistema na presença de um sinal transmitido por uma *HackRF* a 15 metros. Com a antena omnidirecional foi verificado que não era possível detetar qualquer sinal a partir dos 10 metros de distância, portanto não foi possível comparar com os resultados à mesma distância transmitindo com uma *HackRF*, porém é possível comparar com o teste com o sinal transmitido pelo *drone*.

Foram realizadas 25 análises, durante 5 minutos, o que significa que foram feitas 5 análises por minuto e que cada análise em média demorou 12 segundos. Das 25 análises, 17 detetaram um sinal no canal 6, o que corresponde a uma percentagem de 68%. Três análises detetaram um sinal num canal adjacente, que foi no canal 7, o que corresponde a uma percentagem de 12%. Em 3 análises foi detetado sina nos canais 8 e 9, que corresponde a 12%. Apenas em duas análise foi concluído que nenhum sinal estava a operar, correspondendo a uma percentagem de 8%.

g) *Drone* a operar nos 5 GHZ com a antena Omnidirecional

Como mencionado anteriormente, para o projeto não foi possível adquirir uma antena que conseguisse captar sinal nas gamas de 5 e 5,8 GHz com consistência, porém este teste foi realizado para demonstrar que apesar de tal, o sistema é capaz de detetar sinais também nessas gamas de frequência. No teste foi utilizada a antena omnidirecional, que consegue captar sinal a 5 GHz caso o transmissor do mesmo esteja muito perto da antena. Este teste não representa o comportamento real de um *drone*, apenas é para demonstrar que no caso de utilização de uma antena que consiga captar a gama de 5 GHz com consistência, o sistema consegue determinar a existência de um sinal.

O sinal foi emitido no canal 44, com 20 MHz de largura de banda. Foram realizadas 31 análises, durante 5 minutos, o que significa que foram feitas 6,2 análises por minuto e que cada análise em média demorou 9,67 segundos. Das 31 análises, 24 detetaram um sinal no canal 6,

o que corresponde a uma percentagem de 77,4%. Sete análises concluíram que nenhum sinal estava a operar, correspondendo a uma percentagem de 22,58%.

4.2. Testes Reais

O objetivo dos testes reais é avaliar o sistema num ambiente com condições reais, onde possam existir outros sinais a operar. O sistema terá de ser capaz de identificar zonas no espectro que tenham interferência e adaptar a sua decisão tendo em conta essa informação.

4.2.1. Condições

Os testes reais foram executados em 3 ambientes diferentes, sendo o primeiro num exterior remoto, com baixa interferência. Nos testes o mesmo vai ser mencionado como ambiente remoto.

O segundo foi executado numa zona exterior urbana, com a presença de sinais de interferência. Nos testes o mesmo é mencionado como ambiente urbano. O terceiro ambiente é interno, onde a energia dos sinais de ruído é maior, com o objetivo de testar a robustez do sistema. Nos testes o mesmo é mencionado como ambiente interno.

Comi transmissor foi utilizado o *drone Parrot Bebop 2*, em que a conexão foi estabelecida no canal 6 (2427-2447MHz). Os testes foram realizados nas seguintes distâncias: 50 centímetros, 5 metros e 10 metros. Todos os testes foram feitos com o *drone* no solo. Nos testes foram utilizadas duas antenas, uma omnidirecional e uma direcional. Na figura 18 é possível observar o segundo ambiente dos testes.

Figura 18- Ambiente urbano dos testes reais

4.2.2. Resultados

a) *Drone Parrot BeBop* 2 a 50 centímetros com a Antena Omnidirecional no Ambiente Remoto

Neste teste foi analisado o desempenho e robustez do sistema com a presença de um sinal *drone* a 50 cm do mesmo. O objetivo do teste é determinar a taxa de acerto em relação à presença de um sinal, e a determinação do canal utilizado. A figura 19 retrata a configuração de este teste.

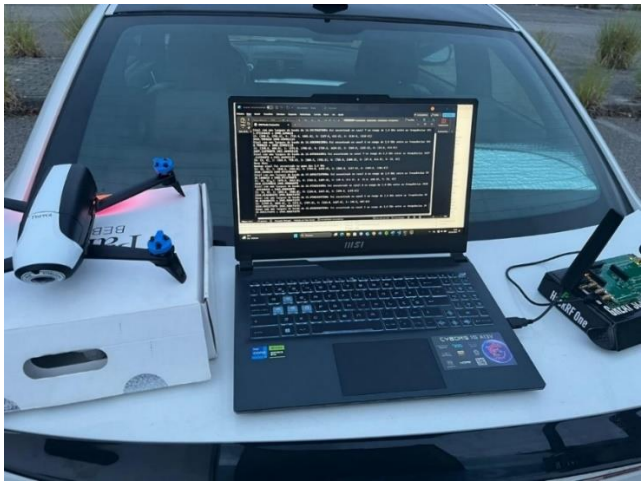

Figura 19- Configuração do teste de 50 centímetros

Foram realizadas 29 análises, durante 5 minutos, o que significa que foram feitas 5,8 análises por minuto e que cada análise em média demorou 10,3 segundos. Das 29 análises, 26 detetaram um sinal no canal 6, o que corresponde a uma percentagem de 89,7%. Duas análises detetaram um sinal num canal adjacente, sendo este o canal 7, o que corresponde a uma percentagem de 6,9%. Apenas uma análise concluiu que nenhum sinal estava a operar, correspondendo a uma percentagem de 3,4%.

b) *Drone Parrot BeBop* 2 a 5 metros com a Antena Omnidirecional no Ambiente Remoto

Neste teste foi analisado o desempenho e robustez do sistema com a presença de um sinal *drone* a 5 metros do mesmo. O objetivo do teste é determinar a taxa de acerto em relação à

presença de um sinal, e a determinação do canal utilizado, a uma distância superior ao teste anterior. Na figura 20 pode-se verificar as condições do teste.

Figura 20- Configuração do teste de 5 metros

Foram realizadas 35 análises, durante 6 minutos, o que significa que foram feitas 5,83 análises por minuto e que cada análise em média demorou 10,28 segundos. Das 35 análises, 24 detetaram um sinal no canal 6, o que corresponde a uma percentagem de 68,57%. Sete análises detetaram um sinal num canal adjacente, sendo que uma detetou um sinal no canal 5 e seis no canal 7, o que corresponde a uma percentagem de 20%. Em duas análises foi determinada a existência de um sinal nos canais 4 e 8, correspondendo a uma percentagem de 5,7 %. Duas análises concluíram que nenhum sinal estava a operar, correspondendo a uma percentagem de 5,7%.

c) *Drone Parrot BeBop* 2 a 16 metros de comprimento e a 5 metros de altura com a Antena Omnidirecional no Ambiente Remoto

Neste teste foi analisado o desempenho e robustez do sistema com a presença de um sinal *drone*, com este em voo, um comprimento de 16 metros e a 5 metros de altura, o que corresponde a uma distância de 16,76 metros. O objetivo do teste é determinar a taxa de acerto em relação à presença de um sinal, e a determinação do canal utilizado, a uma distância superior aos testes anteriores.

Foram realizadas 35 análises, durante 6 minutos, o que significa que foram feitas 5,83 análises por minuto e que cada análise em média demorou 10,28 segundos. Das 35 análises, 12 detetaram um sinal no canal 6, o que corresponde a uma percentagem de 34,28%. Treze análises detetaram um sinal num canal adjacente, sendo que 7 detetaram um sinal no canal 5 e seis no canal 7, o que corresponde a uma percentagem de 37,14%. Em 7 análises foi determinada a existência de um sinal nos canais 4 e 8, correspondendo a uma percentagem de 20 %. Três análises concluíram que nenhum sinal estava a operar, correspondendo a uma percentagem de 8,57%.

Apesar de o sistema apresentar um bom resultado na deteção de um sinal RF, apenas 34% das análises acertaram no canal certo. Isso deve-se ao facto de quanto maior a distância, menor é a potência do sinal recebido pelo sistema, o que pode gerar algumas incongruências nas determinações dos canais. Neste teste, a maioria das análises que detetaram um sinal num canal adjacente apresentaram uma largura de banda inferior a 15 MHZ, o que depois pode afetar a determinação do canal. É de relembrar que a diferença entre o canal correto e os canais adjacentes é de +/- 5MHz.

d) Sem presença de um sinal RF com a antena direcional no Ambiente Remoto

Neste teste foi analisado a deteção de sinais *drone* caso não exista nenhum sinal a operar, utilizando uma antena direcional. O objetivo do mesmo é analisar o número de "falsos positivos" num ambiente controlado.

Foram realizadas 31 análises, durante 5 minutos, o que significa que foram feitas 6,2 análises por minuto e que cada análise em média demorou 9,68 segundos. Das 31 análises, todas concluíram que nenhum sinal foi detetado.

e) *Drone Parrot BeBop* 2 a 50 centímetros com antena direcional no Ambiente Remoto

Neste teste foi analisado o desempenho e robustez do sistema com a presença de um sinal *drone* a 50 centímetros do mesmo, utilizando uma antena direcional. O objetivo do teste é determinar a taxa de acerto em relação à presença de um sinal, e a determinação do canal utilizado.

Foram realizadas 33 análises, durante 5 minutos, o que significa que foram feitas 6,6 análises por minuto e que cada análise em média demorou 9,09 segundos. Das 33 análises, catorze detetaram um sinal no canal 6, o que corresponde a uma percentagem de 42,42%. Dez análises detetaram um sinal num canal adjacente, sendo que oito detetaram um sinal no canal 5 e duas no canal 7, o que corresponde a uma percentagem de 30,3%. Em sete análises foi determinada a existência de um sinal nos canais 4 e 8, correspondendo a uma percentagem de 21,2%. Duas análises concluíram que nenhum sinal estava a operar, correspondendo a uma percentagem de 6,06%.

Comparativamente com o teste com a antena omnidirecional, o teste com a antena direcional obteve piores resultados. Isso deve-se ao facto de a mesma conseguir detetar os sinais com uma potência maior, o que faz com que tanto o ruído à sua volta como as suas réplicas tenham uma potência maior. Posto isso, o sistema está a detetar alguns sinais com uma largura de banda superior à real, o que posteriormente afeta o cálculo do canal que está a ser utilizado. Este contratempo só acontece quando a potência do sinal recebido é muito elevada, pelo que a partir dos 5 metros o sistema acaba por estabilizar.

f) *Drone Parrot BeBop* 2 a 5 m com antena direcional no Ambiente Remoto

Neste teste foi analisado o desempenho e robustez do sistema com a presença de um sinal *drone* a 5 metros do mesmo, utilizando uma antena direcional. O objetivo do teste é determinar a taxa de acerto em relação à presença de um sinal, e a determinação do canal utilizado.

Foram realizadas 31 análises, durante 5 minutos, o que significa que foram feitas 6,2 análises por minuto e que cada análise em média demorou 9,67 segundos. Das 31 análises, 20 detetaram um sinal no canal 6, o que corresponde a uma percentagem de 64,51%. Nove análises detetaram um sinal num canal adjacente, sendo que três detetaram um sinal no canal 5 e seis no canal 7, o que corresponde a uma percentagem de 29,03%. Duas análises concluíram que nenhum sinal estava a operar, correspondendo a uma percentagem de 64%.

g) *Drone Parrot BeBop* 2 a 16 metros de comprimento com antena direcional no Ambiente Remoto

Neste teste foi analisado o desempenho e robustez do sistema com a presença de um sinal *drone* a uma distância de 16 metros, utilizando uma antena direcional. O objetivo do teste é determinar a taxa de acerto em relação à presença de um sinal, e a determinação do canal utilizado, a uma distância superior ao teste anterior.

Foram realizadas 31 análises, durante 5 minutos, o que significa que foram feitas 6,2 análises por minuto e que cada análise em média demorou 9,67 segundos. Das 31 análises, cinco detetaram um sinal no canal 6, o que corresponde a uma percentagem de 16,12%. Vinte e três análises detetaram um sinal num canal adjacente, sendo que duas detetaram um sinal no canal 5 e 21 no canal 7, o que corresponde a uma percentagem de 74,19%. Em 2 análises foi determinada a existência de um sinal em outros canais, correspondendo a uma percentagem de 6,45%. Uma análise concluiu que nenhum sinal estava a operar, correspondendo a uma percentagem de 3,22%.

h) Nenhum sinal RF com Antena omnidirecional no ambiente urbano

Neste teste foi analisado a deteção de sinais *drone* no caso de nenhum sinal proveniente de um *drone* esteja a ser transmitido. O objetivo do mesmo é analisar o número de "falsos positivos" num ambiente controlado.

Foram realizadas 30 análises, durante 5 minutos, o que significa que foram feitas 6 análises por minuto e que cada análise em média demorou 10 segundos. Das 30 análises, todas concluíram que nenhum sinal foi detetado.

É necessário mencionar que nenhum sinal com uma potência dos sinais de interferência considerável foi detetado, pelo que mesmo num meio urbano, os sinais de ruido não foram muito impactantes.

i) *Drone Parrot BeBop* 2 com Antenna omnidirecional a 50 cm no ambiente urbano

Neste teste foi analisado o desempenho e robustez do sistema com a presença de um sinal *drone* a 50 cm do mesmo. O objetivo do teste é determinar a taxa de acerto em relação à presença de um sinal, e a determinação do canal utilizado.

Foram realizadas 29 análises, durante 5 minutos, o que significa que foram feitas 5,8 análises por minuto e que cada análise em média demorou 10,3 segundos. Das 29 análises, 21 detetaram um sinal no canal 6, o que corresponde a uma percentagem de 72,4%. Uma análise detetou um sinal num canal adjacente, sendo este o canal 7, o que corresponde a uma percentagem de 3,5%. Quatro análises determinaram a existência de um sinal no canal 8, o que corresponde a 13,79%. Três análises concluíram que nenhum sinal estava a operar, correspondendo a uma percentagem de 10,3%.

j) *Drone Parrot BeBop* 2 a 5 m com Antena Omnidirecional no ambiente urbano

Neste teste foi analisado o desempenho e robustez do sistema com a presença de um sinal *drone* a 5 metros do mesmo, utilizando a antena omnidirecional. O objetivo do teste é determinar a taxa de acerto em relação à presença de um sinal, e a determinação do canal utilizado, a uma distância superior ao teste anterior.

Foram realizadas 26 análises, durante 5 minutos, o que significa que foram feitas 5,2 análises por minuto e que cada análise em média demorou 11,53 segundos. Das 26 análises, 12 detetaram um sinal no canal 6, o que corresponde a uma percentagem de 46,15%. Seis análises detetaram um sinal num canal adjacente, sendo este o canal 7, o que corresponde a uma percentagem de 23,07%. Oito análise conclui que nenhum sinal estava a operar, correspondendo a uma percentagem de 30,7%.

k) Nenhum sinal RF com Antena direcional no ambiente urbano

Neste teste foi analisado a deteção de sinais *drone* no caso de nenhum sinal proveniente de um *drone* esteja a ser transmitido. O objetivo do mesmo é analisar o número de “falsos positivos” num ambiente urbano.

Foram realizadas 32 análises, durante 5 minutos, o que significa que foram feitas 6,4 análises por minuto e que cada análise em média demorou 9,375 segundos. Das 32 análises, todas concluíram que nenhum sinal foi detetado.

l) *Drone Parrot BeBop* 2 a 5 m com Antena Direcional no ambiente urbano

Neste teste foi analisado o desempenho e robustez do sistema com a presença de um sinal *drone* a 5 metros do mesmo, utilizando a antena direcional. O objetivo do teste é determinar a taxa de acerto em relação à presença de um sinal, e a determinação do canal utilizado, a uma distância superior ao teste anterior. A figura 21 mostra a configuração do sistema.

Figura 21- Configuração dos testes com a antena direcional

Foram realizadas 33 análises, durante 5 minutos, o que significa que foram feitas 6,6 análises por minuto e que cada análise em média demorou 9,09 segundos. Das 33 análises, 21 detetaram um sinal no canal 6, o que corresponde a uma percentagem de 63,64%. Nove análises detetaram um sinal num canal adjacente, sendo que duas análises detetaram um sinal no canal 5 e sete análises no canal 7, o que corresponde a uma percentagem de 27,27%. Uma análise determinou que um sinal estava a operar no canal 8, correspondendo a uma percentagem de 3,03%. Duas análises concluíram que nenhum sinal estava a operar, correspondendo a uma percentagem de 6,06%.

m) *Drone Parrot BeBop* 2 a 10 m com Antena Direcional no ambiente urbano

Neste teste foi analisado o desempenho e robustez do sistema com a presença de um sinal *drone* a 10 metros do mesmo, utilizando a antena direcional. O objetivo do teste é determinar a taxa de acerto em relação à presença de um sinal, e a determinação do canal utilizado, a uma distância superior ao teste anterior. Devido à localização dos testes reais, não foi possível testar com o *drone* em voo, pelo que este é o teste real com maior distância.

Foram realizadas 21 análises, durante 4 minutos, o que significa que foram feitas 5,25 análises por minuto e que cada análise em média demorou 11,43 segundos. Das 21 análises, 12 detetaram um sinal no canal 6, o que corresponde a uma percentagem de 57,14%. Quatro análises detetaram um sinal num canal adjacente, sendo que o mesmo o no canal 7, o que corresponde a uma percentagem de 19,04%. Três análises detetaram um sinal fora dos canais adjacentes, sendo que duas detetaram no canal 8 e uma no canal 10, o que corresponde a uma percentagem de 14,28%. Duas análises concluíram que nenhum sinal estava a operar, correspondendo a uma percentagem de 9,52%.

n) Nenhum sinal com Antena Omnidirecional num ambiente interno

Neste teste foi analisada a deteção de sinais *drone* no caso de nenhum sinal proveniente de um *drone* esteja a ser transmitido num ambiente onde existam sinais de ruido, utilizando uma antena omnidirecional. O objetivo do mesmo é analisar o número de “falsos positivos” num ambiente com sinais de interferência.

Foram realizadas 34 análises, durante 5 minutos, o que significa que foram feitas 6,8 análises por minuto e que cada análise em média demorou 8,82 segundos. Das 34 análises, 23 concluíram que não existia num sinal anormal a operar na gama observada, o que corresponde a uma percentagem de 67,65%. Foi também detetado ruido na zona de frequência perto de 2,44 e 2,45 GHz. Onze análises determinaram a existência de um sinal.

o) *Drone Parrot BeBop* 2 com Antena Omnidirecional a 50 centímetros num ambiente interno

Neste teste foi analisado o desempenho e robustez do sistema com a presença de um sinal *drone* num ambiente interno, onde existe a presença de sinais de ruido, utilizando a antena omnidirecional. O objetivo do teste é determinar a taxa de acerto em relação à presença de um sinal, e a determinação do canal utilizado. Foi utilizado à mesma o canal 6, onde se encontrava alguma interferência. É necessário mencionar que mesmo num ambiente com interferência, o *drone* iria sempre escolher um canal com a menor interferência possível. Neste teste o canal é selecionado manualmente para analisar o comportamento do sistema numa zona com interferência, como se pode observar na figura 22.

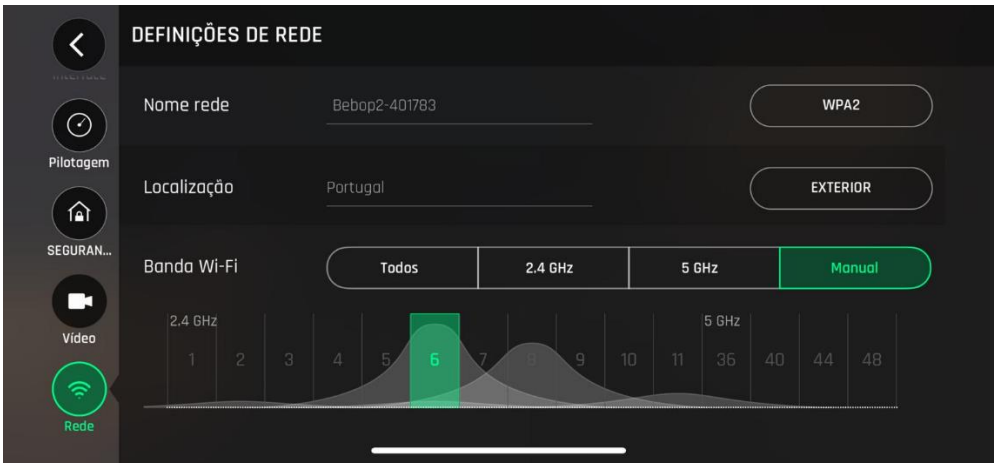

Figura 22- Interface da aplicação FreeFlight Pro

Foram realizadas 32 análises, durante 5 minutos, o que significa que foram feitas 6,4 análises por minuto e que cada análise em média demorou 9,38 segundos. Das 32 análises, 4 detetaram um sinal no canal 6, o que corresponde a uma percentagem de 12,5%. Treze análises detetaram um sinal num canal adjacente, sendo que 11 análises determinaram que o sinal se encontrava no canal 5 e duas no canal 7, o que corresponde a uma percentagem de 37,5%. Doze análises determinaram que o sinal se encontrava no canal 4, porém todas as análises apresentaram uma largura de banda maior comparada com as demais (mais de 25 MHz), pelo que se pode assumir que sinais de ruido misturam-se com o sinal que se pretendia detetar, justificando assim a quantidade das análises, que correspondem a uma percentagem de 37,5 %. Três análises concluíram que nenhum sinal estava a operar, correspondendo a uma percentagem de 9,38%.

p) Nenhum sinal com Antena Direcional num ambiente interno

Neste teste foi analisado a deteção de sinais *drone* no caso não da inexistência de um sinal, sinal emitido por um *drone* num ambiente onde existam sinais de ruido, utilizando uma antena direcional. O objetivo do mesmo é analisar o número de "falsos positivos" num ambiente onde pode existem interferência.

Foram realizadas 31 análises, durante 5 minutos, o que significa que foram feitas 6,2 análises por minuto e que cada análise em média demorou 9,677 segundos. Das 31 análises, todas concluíram que nenhum sinal foi detetado.

q) *Drone Parrot BeBop* 2 com Antena Direcional a 50 centímetros num ambiente interno

Neste teste foi analisado o desempenho e robustez do sistema com a presença de um sinal *drone* num ambiente interno, onde existe a presença de sinais de ruido, utilizando a antena direcional. O objetivo do teste é determinar a taxa de acerto em relação à presença de um sinal, e a determinação do canal utilizado. Foi utilizado à mesma o canal 6, onde se encontrava alguma interferência.

Foram realizadas 29 análises, durante 5 minutos, o que significa que foram feitas 5,8 análises por minuto e que cada análise em média demorou 10,34 segundos. Das 29 análises, 5 detetaram um sinal no canal 6, o que corresponde a uma percentagem de 17,24%. Dezanove análises detetaram um sinal num canal adjacente, sendo que 4 análises determinaram que o sinal se encontrava no canal 5 e quinze no canal 7, o que corresponde a uma percentagem de 65,5%. Cinco análises determinaram que o sinal se encontrava em outros canais, sendo que 1 determinou no canal 4, duas no canal 8 e duas no canal 9 o que corresponde a 17,24 %. Nenhuma análise concluiu que nenhum sinal foi encontrado.

4.3. Métricas de precisão

Neste subcapítulo serão apresentados comparações e conclusões acerca da robustez e precisão do sistema com os resultados adquiridos nos testes descritos no subcapítulo anterior, em conjunto com algumas métricas de precisão.

As métricas de precisão utilizadas neste projeto serão a percentagem de positivos verdadeiro (ou sensibilidade), positivos falsos, negativos verdadeiros, negativos falsos e precisão.

A percentagem de positivos verdadeiros será representada por PPV, que tem a seguinte formula:

$$PPV = \frac{positivos\ verdadeiros}{Positivos}. \quad (4.1)$$

A percentagem de positivos falsos será representada por PPF, com a seguinte fórmula:

$$PPF = \frac{positivos\ falsos}{negativos} \quad (4.2)$$

A percentagem de negativos verdadeiros será representada por PNV, e tem a seguinte fórmula:

$$PNV = \frac{negativos\ verdadeiros}{negativos} \quad (4.3)$$

A percentagem de negativos falsos será representada por PNF, e tem a seguinte fórmula:

$$PNF = \frac{negativos\ falsos}{positivos} \quad (4.4)$$

A precisão tem a seguinte fórmula:

$$Precisão = \frac{positivos\ verdadeiros}{Positivos\ verdadeiros+positivos\ falsos} \quad (4.5)$$

A percentagem de positivos verdadeiros no canal correto é representada por PPVCC. Nas métricas, a deteção de um sinal no canal certo ou adjacente é assumida como positivo, e a "não presença" um negativo.

4.3.1. Nenhum sinal RF

Foram executados 6 testes em ambientes distintos onde nenhuma comunicação *drone* estava a ser transmitida. Entre estes, 2 testes foram executados num ambiente remoto, com condições controladas, com a utilização de antenas distintas. Os restantes foram executados em

condições reais, 2 num ambiente urbano e 2 num ambiente interno, onde existiam outros sinais de interferência.

À exceção do teste realizado numa zona interior com a antena omnidirecional, todas as análises dos testes concluíram que nenhum sinal anormal tinha sido encontrado. Nestes ambientes precisão será igual a 100%, pois não existiram falsos positivos.

No ambiente interno, a percentagem de positivos falsos é de 32,35% e a percentagem de negativos verdadeiros é de 67,65 %

4.3.2. Sinal a 50 centímetros

Foram executados 6 testes onde um sinal RF foi transmitido a 50 centímetros do sistema, sendo que 4 foram em um ambiente controlado, um num ambiente real e um num ambiente interno.

As métricas dos testes estão representadas na tabela 5:

Tabela 5- Métricas dos testes a 50 centímetros

	P	N	PV	PF	NV	NF	PPV	PPF	PNV	PNF	Precisão	PPVCC
Drone Parrot BeBop 2 Antena Omnidirecional Ambiente Remoto	29	32	28	0	32	1	96,55%	0%	100%	3,45%	100,00%	89,70%
Drone Parrot BeBop 2 Antena Direcional Ambiente Remoto	33	31	24	0	31	9	72,73%	0%	100%	27,27%	100,00%	42,42%
HackRf Antena Omnidirecional Controlado	32	32	31	0	32	1	96,88%	0%	100%	3,13%	100,00%	18,75%
HackRf Antena Direcional Controlado	32	31	30	0	31	2	93,75%	0%	100%	6,25%	100,00%	56,25%
Drone Parrot BeBop 2 Antena Omnidirecional Ambiente Urbano	29	30	22	0	30	7	75,86%	0%	100%	24,14%	100,00%	72,40%
Drone Parrot BeBop 2 Antena Omnidirecional Ambiente Interno	32	34	17	11	23	15	53,13%	32%	68%	46,88%	60,71%	12,50%
Drone Parrot BeBop 2 Antena Direcional Ambiente Interno	29	31	24	0	31	5	82,76%	0%	100%	17,24%	100,00%	17,24%

Observando as métricas, pode-se concluir que o sistema a uma distância de 50 cm consegue determinar com eficácia a gama de frequências onde um *drone* está a operar, conseguindo uma percentagem de positivos verdadeiros acima de 72% em todos os cenários. Em relação aos testes controlados, o único resultado mais inesperado é o do teste com *drone* como transmissor

e utilizando a antena direcional no ambiente remoto. O mesmo acontece, pois, o sistema tem um mecanismo para regular a decisão em relação à energia do sinal, porém quando a mesma é muito alta, o sistema pode considerar uma das réplicas ou ruído adjacente ao sinal como o sinal de facto. Isso faz com que o resultado do sistema seja um sinal com uma largura de banda superior, e com os limites um pouco desajustados. Neste teste, como o sistema está muito perto do transmissor, algumas análises concluíram que o sinal está no canal 4 (7), com limites entre 2416-2446 MHz. O canal 6 opera entre 2427-2447 MHz, pelo que se pode verificar que o limite inferior está abaixo do esperado, e assim retornando um valor anormal. Neste teste apenas 6,06% das análises não detetaram um sinal.

Em relação aos testes reais, os resultados apresentaram uma pior eficácia comparado com o ambiente controlado, o que é expectável.

Em relação à determinação do canal correto, verifica-se que sistema consegue fazer a escolha certa, à exceção dos testes com o *drone*, utilizando uma antena direcional e *HackRF* utilizando uma antena Omnidirecional, pela razão anteriormente explicada da deteção de uma largura de banda superior à do sinal.

Nos testes reais, apenas o teste interno apresentou resultado inferior ao esperado, porém o mesmo foi realizado numa gama de frequência com ruido, o que fez com que os resultados fossem ambíguos.

Em relação aos testes no ambiente interno, os mesmos apresentaram percentagens menores de positivos verdadeiros e positivos verdadeiros no canal certo, como já era de esperar pelo facto de o ruído nesse ambiente ser superior comparando com restantes. Também se pode observar que a antena direcional obteve melhores resultados do que o teste com a antena omnidirecional, pelo facto de não terem sido detetados falsos positivos e a percentagem de positivos verdadeiro foi superior.

4.3.3. Sinal a 5 metros

Foram executados 6 testes onde um sinal RF foi transmitido a 5 metros do sistema, sendo que 2 foram com condições controladas, 2 num ambiente remoto e 2 num ambiente urbano.

As métricas dos testes estão representadas na tabela 6:

Tabela 6- Métricas dos testes 5 metros

	P	N	PV	PF	NV	NF	PPV	PPF	PNV	PNF	Precisão	PPVCC
Drone Parrot BeBop 2 Antena Omnidirecional Ambiente Remoto	35	32	31	0	32	4	88,57%	0%	100%	11,43%	100,00%	68,57%
Drone Parrot BeBop 2 Antena Direcional Ambiente Remoto	31	31	29	0	31	2	93,55%	0%	100%	6,45%	100,00%	64,51%
HackRF Antena Omnidirecional Controlado	27	32	26	0	32	1	96,30%	0%	100%	3,70%	100,00%	92,60%
HackRF Antena Direcional Controlado	33	31	28	0	31	4	84,85%	0%	100%	12,12%	100,00%	51,50%
Drone Parrot BeBop 2 Antena Omnidirecional Ambiente Urbano	26	30	18	0	30	8	69,23%	0%	100%	30,77%	100,00%	46,15%
Drone Parrot BeBop 2 Antena Direcional Ambiente Urbano	33	30	30	0	30	3	90,91%	0%	100%	9,09%	100,00%	63,64%

Observando as métricas, pode-se concluir que a 5 metros, o sistema é capaz de identificar a presença de um sinal com eficácia, conseguido em todos os testes uma percentagem de positivos verdadeiros superior a 84% exceto no teste com o drone com a antena omnidirecional no ambiente urbano que foi 69%. Comparativamente com os testes anteriores, podemos observar que o sistema obteve um melhor resultado no teste com o *drone* utilizando uma antena direcional num ambiente remoto. Como a potência do sinal não é muito elevada, as réplicas conseguem ser ignoradas, determinando com mais precisão a presença de um sinal. O teste controlado com o *HackRF* utilizando uma antena direcional também apresenta resultados melhores em relação à determinação do canal correto, pela mesma razão.

À exceção do teste com o *drone* e antena direcional no ambiente remoto, todos os testes apresentaram percentagens de positivos verdadeiros inferiores ao teste anterior, o que é de esperar. Porém continuam a ter uma boa eficácia.

Em relação à percentagem de positivos verdadeiros no canal correto, apenas o teste com o *drone* com uma antena direcional no ambiente remoto e o teste controlado utilizando a *HackRF* com antena omnidirecional apresentaram resultados melhores em relação aos testes anteriores. No entanto, à exceção do teste com o *drone* e uma antena omnidirecional no ambiente urbano, todos os testes acertaram no canal mais do que 50%.

Pode-se também verificar que a antena direcional possui uma melhor percentagem de positivos verdadeiros em todos os testes que utilizam o *drone*. Apenas nos testes que utilizam o *HackRF* como transmissor a antena omnidirecional obteve uma melhor performance. Isso deve ao facto de a potência do sinal emitido pelo *HackRF* ser elevada, maior que a do sinal do *drone*, pelo que o sistema tem mais dificuldade determinar ao certo o intervalo de frequência que o sinal está a utilizar, retornando normalmente intervalos maiores, consequentemente não determinando o canal correto. É importante mencionar que no teste que utiliza a antena direcional todas as análises detetaram um sinal, porém algumas fora dos canais em que se considera um resultado positivo.

Com isto, pode-se verificar que a 5 metros o sistema apresenta uma performance inferior ao teste anterior, porém continua a ter uma boa percentagem de acerto e consegue determinar o canal correto com alguma precisão.

4.3.4. Sinal a 16 metros

Foram executados 4 testes onde um sinal RF foi transmitido a 16 metros do sistema, sendo um dos testes com condições controladas, 2 testes num ambiente remoto e um num ambiente urbano.

Neste conjunto de testes não foi efetuado o teste com o *drone* e uma antena omnidirecional no ambiente remoto porque foi verificado a antena não tem a capacidade detetar um sinal a essa distância. No ambiente urbano o teste foi efetuado apenas a 10 metros pois foi a distância mais próxima possível de testar. Neste teste não foi possível fazer o mesmo com o *drone* em modo voo pelos factos de o ambiente não ser um espaço muito aberto e eu não ter experiência no controlo do mesmo.

As métricas dos testes estão representadas na tabela 7:

Tabela 7- Métricas dos testes a 16 metros

	P	N	PV	PF	NV	NF	PPV	PPF	PNV	PNF	Precisão	PPVCC
Drone Parrot BeBop 2 Antena Omnidirecional Ambiente Remoto	35	32	25	0	32	10	71,43%	0%	100%	28,57%	1	34,28%
Drone Parrot BeBop 2 Antena Direcional Ambiente Remoto	31	31	28	0	31	3	90,32%	0%	100%	9,68%	1	16,12%
HackRF Antena Direcional Controlado	25	31	20	0	31	5	80,00%	0%	100%	20,00%	1	68,00%
Drone Parrot BeBop 2 Antena Direcional Ambiente Urbano	21	30	16	0	30	5	76,19%	0%	100%	23,81%	1	57,14%

O teste de 16 metros é o que melhor consegue simular a atuação de um *drone* num cenário real, e de acordo com as métricas pode ser observado que o sistema consegue determinar a presença de um sinal *drone* com uma boa eficácia, sendo que a sensibilidade (percentagem de positivos verdadeiros) ultrapassa os 70% em todos os tetses.

Com estes testes também pode-se observar que, com o aumento da distância, a diferença de performance dos mesmos com diferentes antenas vai sendo mais significativa, pelo que a antena direcional não só consegue apresentar uma taxa de positivos verdadeiros maior, mas também consegue detetar o canal utilizado com mais precisão.

Também se pode observar que a performance do sistema é melhor nos testes com o *drone* do que nos testes que utilizam a *HackRF* como transmissor, o que poderá ser justificado pela capacidade de transmissão do *HackRF* e da antena utilizada (omnidirecional).

Como de esperado, no ambiente urbano o sistema teve uma pior taxa de positivos verdadeiros comparado com o ambiente remoto, porém continua a ser uma taxa bastante aceitável de modo a concluir que existe um *drone* a operar, e em que canal/frequências.

A figura 23 representa o gráfico da evolução da percentagem de positivos verdadeiros ao longo da distância para os diversos tipos de teste:

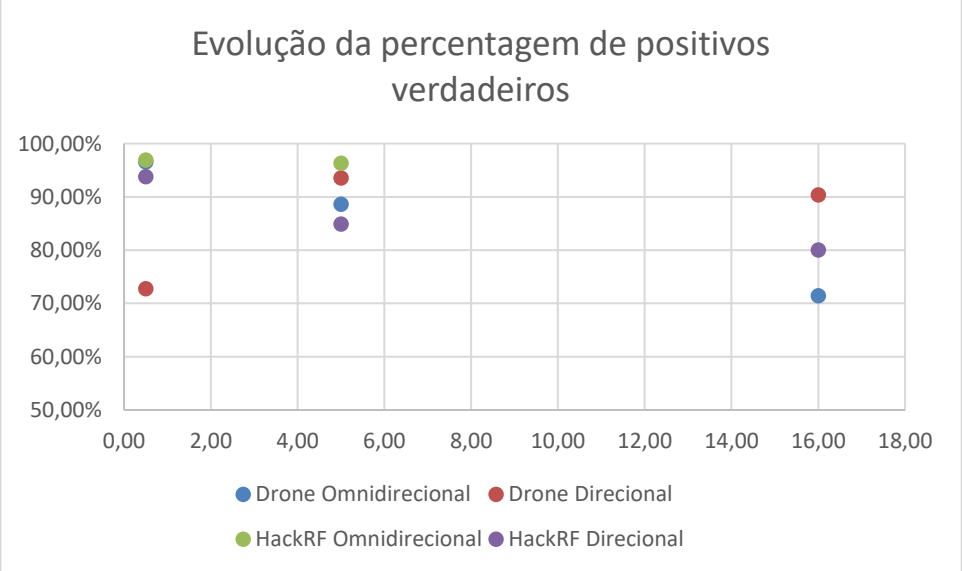

Figura 23- Evolução da percentagem de positivos verdadeiros

No gráfico, o único valor anormal que se verificou no teste foi com o *drone* e uma antena direcional a 50 cm no ambiente remoto, pelos motivos já descritos anteriormente. Com o resto da informação conseguimos perceber que ao longo da distância, a percentagem de positivos verdadeiros vai diminuído, mas não de uma maneira muito abrupta. O teste que apresenta uma

baixa da percentagem mais acentuada à medida que a distância aumenta é o com o *drone* e uma antena omnidirecional no ambiente remoto. Por esse motivo e pelo facto de no teste de 16 metros com o *HackRF* como transmissor e o sistema com a antena omnidirecional não conseguir detetar o sinal, podemos concluir que pelo facto de o ganho da antena ser baixo, o rendimento do sistema ao utilizar a mesma desce bastante à medida que a distância aumenta.

A figura 24 representa o gráfico da evolução das análises que detetaram o sinal no canal correto no ambiente remoto:

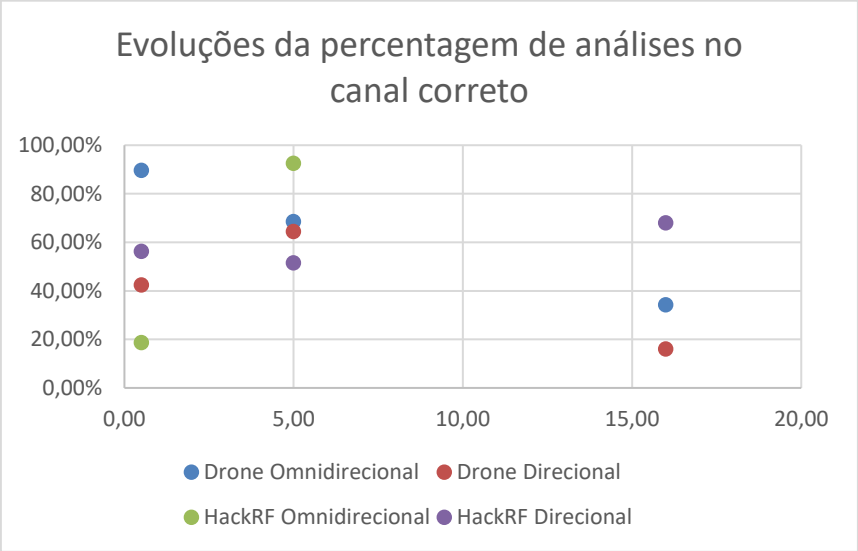

Figura 24- Evolução da percentagem de análises no canal correto

Na figura 24 pode se verificar que, apesar de mais inconstante, a percentagem das análises que determinaram o sinal no canal correto diminui ao longo que a distância aumenta, à exceção dos testes que utilizam o *HackRF*, pois o sinal transmitido possui uma elevada potência a curtas distâncias, o que pode proporcionar resultados um pouco desfasados em relação à determinação do sinal correto. Conforme a distância aumenta, a potência do sinal recebido pelo sistema vai diminuindo, conseguindo estabilizar a determinação do canal.

CAPÍTULO 5

Conclusão e Trabalho Futuro

5.1. Conclusão

O sistema desenvolvido neste projeto foi planeado de modo a conseguir fazer identificar a gama de frequências de um *drone* não identificado, utilizado uma plataforma SDR.

Recorrendo a um *SDR HackRF*, foi desenvolvido um sistema na plataforma GNU Radio que através de um varrimento espectral e da determinação da energia do espectro, consegue determinar se um existe um *drone* a operar no ambiente da análise, e no caso de existir, em que gama de frequências e canal.

O projeto contou com 5 perguntas de investigação, em que todas obtiveram resposta ao longo da dissertação.

A primeira era qual as gamas mais utilizadas pelos *drones*, pelo que foi verificado que são as de 2,4 GHz e 5,8 GHz. A segundo era que tipos de sinais são utilizados na comunicação entre o *drone* e o controlador, pelo que foi verificado que são sinal RF para a telemetria e sinais GNSS para o GPS. A próxima pergunta era acerca de qual o melhor equipamento em termos de custo-eficácia para efetuar o varrimento do espectro e fazer a deteção da comunicação entre um *drone*, pelo que foi concluído que o HackRF One é a melhor opção, pois consegue cobrir todas as gamas de frequência utilizadas pelos *drones* e não tem um custo muito elevado comparado com outras alternativas. A quarta pergunta era sobre a técnica mais viável para detetar um *drone*, pelo que a técnica da deteção e verificação da energia do espectro foi a escolhida. A última pergunta questiona qual a melhor técnica para executar uma deteção robusta, tendo em conta as várias fontes de sinais RF. Das respostas todas esta foi a mais ambígua, pelo que neste projeto não foram feitos testes muito robustos (com diferentes tipos de *drones*, com vários *drones* a operar em simultâneo, *etc*) porém a melhor técnica para atender a essas adversidades foi a deteção base do ambiente, onde é possível verificar que tipos de sinais RF existem no mesmo, e em que frequência estão a operar, de modo a influenciar o resultado do sistema.

Os objetivos propostos para esta dissertação de mestrado foram atingidos, na medida em que através dos testes realizados para o drone Parrot Bebop2, o sistema consegue detetar com bastante eficácia a presença de um *drone*, a gama de frequência e canal utlizado pelo sinal

de comunicação do mesmo. O sistema tem a vantagem de ser um portátil, o que permite a sua utilização em qualquer lado, desde que tenha um computador.

O sistema, porém, tem algumas limitações, visto que quando a energia do especto é muito elevada ou o sinal detetado tem uma grande potência, o sistema pode apresentar resultados um pouco desfasados e também é necessário ter antenas que consigam cobrir todas as gamas de frequência mais utilizadas por estes *drones*.

5.2. Trabalho Futuro

Em relação a trabalhos futuros, o desenvolvimento e melhoramento das métricas do sistema pode focar-se em vários pontos, sendo os principais os seguintes:

- Testes para as 3 principais gamas de frequências
- Testes mais robustos
- Alteração automáticas dos ganhos tendo em conta o ruido da gama analisada
- Desenvolvimento de um modelo de *machine learning*

O mais imediato é testar o sistema para as gamas de frequência que não foram abrangidas nos testes dos projetos, apesar de em princípio o sistema funciona para as 3 principais gamas de frequências os equipamentos *drone*.

Também poderão ser feitos testes mais robustos e com mais condicionantes, por exemplo com *drones* de marcas diferentes, *drones* controlados por um controlador remoto, testes com distâncias maiores, vários *drones* a operar ao mesmo tempo etc.

Poderia ser feito um algoritmo onde os ganhos aplicados na receção do sinal variassem tendo em conta o ambiente analisado, o que poderia melhorar a performance do sistema, principalmente nas gamas de 5 e 5,8 GHz.

Para melhorar a tomada de decisão e a informação que o sistema consegue apurar, poderá ser desenvolvido um modelo de *machine learning* com as características do sinal de cada equipamento analisado, de modo que resposta do sistema para além retornar o intervalo de frequência e canal utilizado, retorne também a marca e modelo do equipamento em questão.

Referências Bibliográficas

[1] A. Guimarães, "Guerra na Ucrânia: Porque é que os drones têm vindo a ganhar importância?," 16 12 2022. [Online]. Available: https://cnnportugal.iol.pt/guerra/ucrania/guerra-na-ucrania-porque-e-que-os-drones-tem-vindo-a-ganhar-importancia/20221216/639cded00cf2254fb288c025. [Acedido em 15 07 2024].

[2] D. Santos, P. Sebastião e N. Souto, "Low-cost SDR based FMCW radar for UAV," em *2019 22nd International Symposium on Wireless Personal Multimedia Communications (WPMC)*, Lisbon, Portugal, 2019.

[3] H. Fu, S. Abeywickrama, L. Zhang e C. Yuen, "Low-Complexity Portable Passive Drone," *IEEE Communications Magazine,* vol. 56, nº 4, pp. 112-118, 01 04 2018.

[4] S. Deshmukh e V. Sharma, "An SDR-based anti-drone system with Detection, Tracking, Jamming, and Spoofing Capabilities," em *2022 IEEE Microwaves, Antennas, and Propagation Conference, MAPCON 2022*, Bangalore, India, 2022.

[5] D. Ball, N. Naik e P. Jenkins, "Lightweight and Cost-Effective Sepctrum Analyser Based on Software Defined Radio ans Raspberry Pi," em *2017 European Modelling Symposium (EMS)*, Manchester, UK, 2017.

[6] A. Martian, F. L. Chiper, O. M. K. Al-Dulaimi, M. J. A. Al Sammarraie, C. Vladeanu e I. Marghescu, "Comparative Analysis of Software Defined Radio Platforms for Spectrum Sensing Applications," em *2020 13th International Conference on Communications (COMM)*, Bucharest, Romania, 2020.

[7] M. Nayak, D. Parida, U. Bhanja, D. Dash e K. D. Sa, "A Real Time Implementation of Spectrum," em *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, Kerala, India, 2017.

[8] S. Majumder, "Energy Detection Spectrum Sensing on RTL-SDR based IoT Platform," em *2018 Conference on Information and Communication Technology (CICT)*, Jabalpur, India, 2018.

[9] J. Manco, I. Dayoub, A. Nafkha, M. Alibakhshikenari e H. B. Thameur, "Spectrum Sensing Using Software Defined Radio for Cognitive Radio Networks: A Survey," *IEEE Access,* vol. 10, pp. 131887-131908, 15 12 2022.

[10] M. Perotoni e K. dos Santos, "SDR-based spectrum analyzer based in open-source GNU radio," *Journal of Microwaves, Optoelectronics and Electromagnetic Applications,* vol. 20, nº 3, pp. 542-55, 09 2021.

[11] ANAC, “Categoria Aberta (OPEN),” 04 07 2024. [Online]. Available: https://www.anac.pt/vPT/Generico/drones/categoria_aberta/Paginas/CategoriaAberta.aspx. [Acedido em 05 08 2024].

[12] ANAC, “Categoria Específica (SPEC),” 15 04 2024. [Online]. Available: https://www.anac.pt/vPT/Generico/drones/categoria_especifica/Paginas/CategoriaEspecifica.aspx. [Acedido em 05 08 2024].

[13] ANAC, “Categoria Certificada,” 18 12 2024. [Online]. Available: https://www.anac.pt/vPT/Generico/drones/categoria_certificada/Paginas/CategoriaCertificada.aspx. [Acedido em 05 08 2024].

[14] ANAC, “Formação, exames e certificados de competências dos pilotos remotos na categoria de operações aberta,” 04 07 2024. [Online]. Available: https://www.anac.pt/vPT/Generico/drones/categoria_aberta/formacao_exames_certificados/Paginas/Formacao_ExameseCertificados.aspx. [Acedido em 05 08 2024].

[15] R. Portuguesa, “Drones passam a ter registo obrigatório,” 05 07 2018. [Online]. Available: https://www.portugal.gov.pt/pt/gc21/comunicacao/noticia?i=drones-passam-a-ter-registo-obrigatorio. [Acedido em 13 06 2024].

[16] M. Smith, “DRONE ANATOMY 101,” [Online]. Available: https://static.bhphotovideo.com/FrameWork/Product_Resources/monthlyPDF/winter2018/aerial.pdf. [Acedido em 05 05 2024].

[17] J. Romero, “Drone Guider,” 25 07 2023. [Online]. Available: https://droneguider.com/what-frequency-do-drones-use/. [Acedido em 10 09 2024].

[18] A. Engenharia, “O que é : Tipos de protocolos de comunicação para drones,” 16 08 2023. [Online]. Available: https://aeroengenharia.com/glossario/o-que-e-tipos-de-protocolos-de-comunicacao-para-drones/. [Acedido em 22 02 2024].

[19] J.-. U. A. System, “JOUAV- Unmanned Aircraft System,” 3 10 2023. [Online]. Available: https://www.jouav.com/blog/autonomous-drones.html. [Acedido em 12 02 2024].

[20] L. SZ DJI Technology Co., “Support for Phantom 3 Standard,” 08 2015. [Online]. Available: https://www.dji.com/pt/support/product/phantom-3-standard. [Acedido em 01 09 2024].

[21] L. SZ DJI Technology Co., “DJI,” [Online]. Available: https://www.dji.com/pt/products/camera-drones#mavic-series. [Acedido em 01 09 2024].

[22] L. SZ DJI Technology Co., “Phantom 4 Pro V2.0,” 05 2018. [Online]. Available: https://www.dji.com/pt/phantom-4-pro-v2?site=brandsite&from=landing_page. [Acedido em 01 09 2024].

[23] F. D. Supply, “DJI Ocusync 3 Enterprise - Overview,” 06 05 2024. [Online]. Available: https://www.youtube.com/watch?v=jxDqY8blQKE&t=187s&ab_channel=FloridaDroneSupply. [Acedido em 01 09 2024].

[24] W. Willoughby, “DJI Transmission Systems – Wi-Fi, OcuSync & Lightbridge,” heliguy, 01 03 2022. [Online]. Available: https://www.heliguy.com/blogs/posts/dji-transmission-systems-wi-fi-ocusync-lightbridge?srsltid=AfmBOooai03kt4m2kJnQtJkF74_OlFtaow9i4ILvDhIHhlThWK3ULNfd. [Acedido em 01 09 2024].

[25] Parrot, “Parrot ANAFI Ai Connectivity,” Parrot, 30 06 2021. [Online]. Available: https://www.parrot.com/us/drones/anafi-ai/technical-documentation/connectivity. [Acedido em 2024 09 01].

[26] R. W. World, “SDR Architecture | Software Defined Radio Architecture Benefits,” World, RF Wireless, [Online]. Available: https://www.rfwireless-world.com/Articles/SDR-Software-Defined-Radio-basics.html. [Acedido em 20 02 2024].

[27] J. R. M. Fernández, “Software Defined Radio: Basic Principles and Applications,” *Revista Facultad de Ingeniería,* vol. 24, nº 38, pp. 79-96, 1 2015.

[28] V. Solutions, “What is Software Defined Radio (SDR)?,” VIAVI Solutions, [Online]. Available: https://www.viavisolutions.com/en-us/what-software-defined-radio-sdr. [Acedido em 01 04 2024].

[29] H. Liu, Z. Wei, Y. Chen, J. Pan, L. Lin e Y. Ren, “Drone Detection based on An Audio-assisted Camera Array,” em *2017 IEEE Third International Conference on Multimedia Big Data*, Laguna Hills, CA, USA, 2017.

[30] P. Andraši, T. Radišić, M. Muštra e J. Ivošević, “Night-time Detection of UAVs using Thermal Infrared Camera,” em *6th International Conference on Air Transport (INAIR 2017)*, Prague, Czech Republic, 2017.

[31] S. Jeon, . J.-W. Shin, Y.-J. Lee, W.-H. Kim, Y. Kwon e H.-Y. Yang, “Empirical Study of Drone Sound Detection in real-life environment with deep neural networks,” em *2017 25th European Signal Processing Conference (EUSIPCO)*, Kos, Greece, 2017.

[32] E. M. Moreno, “DRONE DETECTION AND INHIBITION,” TelecomBCN, Barcelona, Spain, 2020.

[33] D. Lopez, “DRONE DETECTION AND CLASSIFICATION USING GNU RADIO,” California State , Pomona, California, United States of America, 2023.

[34] G. R. Software, “Embedded Python Block,” GNU RAdio Software, 1 07 2024. [Online]. Available: https://wiki.gnuradio.org/index.php/Embedded_Python_Block. [Acedido em 01 09 2024].

[35] G. Radio, “Guided Tutorial GNU Radio in C++,” Gnu Radio, 22 05 2022. [Online]. Available: https://wiki.gnuradio.org/index.php/Guided_Tutorial_GNU_Radio_in_C%2B%2B. [Acedido em 01 09 2024].

[36] D. Stolnikov , “osmocom Gnu Radio Blocks,” [Online]. Available: https://osmocom.org/projects/gr-osmosdr/wiki/GrOsmoSDR. [Acedido em 05 05 2024].

[37] G. R. software, “QT GUI Waterfall Sink,” GNU Radio software, 08 07 2024. [Online]. Available: https://wiki.gnuradio.org/index.php/QT_GUI_Waterfall_Sink. [Acedido em 01 09 2024].

[38] G. R. Software, “Stream to Vector,” Gnu Radio Software, 10 12 2020. [Online]. Available: https://wiki.gnuradio.org/index.php/Stream_to_Vector. [Acedido em 05 05 2024].

[39] G. S. GADGETS, “HackRF One,” [Online]. Available: https://greatscottgadgets.com/hackrf/one/. [Acedido em 01 09 2024].

[40] Bingfu, “Bingfu WiFi Antenna 6E Tri-Band (2.4GHz, 5GHz and 6GHz) with 5dBi MIMO RP-SMA Male WiFi Antenna for WiFi Router Fast and Wireless Connections in Smart Homes and Offices,” Amazon, 05 05 2024. [Online]. Available: https://www.amazon.es/dp/B0C2BWZ89C?ref_=ast_sto_dp. [Acedido em 05 05 2024].

[41] ALLOYSEED, “Antena direcional da transmissão da imagem, wifi 2.4g, yagi, 2.35-2.55ghz, DB 10,” AliExpress, 05 05 2024. [Online]. Available: https://pt.aliexpress.com/item/1005007613743268.html?dp=ihasaeqi.github.io&aff_fcid=b51865091f8b4788b44819c080df9d3b-1729531063050-07652-_DBEsyp3&tt=CPS_NORMAL&aff_fsk=_DBEsyp3&aff_platform=portals-tool&sk=_DBEsyp3&aff_trace_key=b51865091f8b4788b44819c080. [Acedido em 05 05 2024].

[42] G. S. Gadgets., “Opera Cake,” Great Scott Gadgets., [Online]. Available: https://hackrf.readthedocs.io/en/latest/opera_cake.html. [Acedido em 01 09 2024].

[43] P. SA, “Parrot,” Parrot SA, [Online]. Available: https://www.parrot.com/en. [Acedido em 18 09 2024].

[44] P. SA, “Parrot Bebop 2,” Parrot SA, 16 09 2016. [Online]. Available: https://web.archive.org/web/20160926202256/https://www.parrot.com/us/drones/parrot-bebop-2#technicals. [Acedido em 18 09 2024].

[45] P. SA, “Parrot Bebop 2,,” 11 2015. [Online]. Available: https://www.parrot.com/assets/s3fs-public/media-public/EN_Pressrelease2015/parrotbebop2theall-in-onedrone.pdf. [Acedido em 18 09 2024].

[46] G. S. Gadgets, “FAQ,” Great Scott Gadgets, [Online]. Available: https://hackrf.readthedocs.io/en/latest/faq.html. [Acedido em 01 09 2024].

[47] G. Radio, “Remove DC Spike,” GNU Radio, 14 04 2021. [Online]. Available: https://wiki.gnuradio.org/index.php/Remove_DC_Spike. [Acedido em 01 09 2024].

[48] G. R. Software, “Stream to Vector,” GNU Radio Software, 10 12 2020. [Online]. Available: https://wiki.gnuradio.org/index.php/Stream_to_Vector. [Acedido em 06 06 2024].

[49] G. R. Software, “Keep 1 in N,” GnU Radio Software, 06 08 2023. [Online]. Available: https://wiki.gnuradio.org/index.php/Keep_1_in_N. [Acedido em 06 06 2024].

[50] G. R. Software, “FFT,” GNU Radio Software, 26 05 2022. [Online]. Available: https://wiki.gnuradio.org/index.php/FFT. [Acedido em 06 06 2024].

[51] G. R. Software, “Complex to Mag Squared,” GNU Radio Software, 13 04 2024. [Online]. Available: https://wiki.gnuradio.org/index.php/Complex_to_Mag_Squared. [Acedido em 06 06 2024].

[52] G. R. Software, “Moving Average,” GNU Radio Software, 18 12 2021. [Online]. Available: https://wiki.gnuradio.org/index.php/Moving_Average. [Acedido em 06 06 2024].

Anexo A- Primeira página do artigo científico

Smart Spectrum Analyser for UAV link detection using Software Defined Radios

Tiago Soeira
Iscte – Instituto Universitário de Lisboa
Lisbon, Portugal
tiago_francisco_soeira@iscte-iul.pt

Nuno Souto
Iscte - Instituto Universitário de Lisboa
Lisbon, Portugal
nuno.souto@iscte-iul.pt

Renato Ferreira
Iscte - Instituto Universitário de Lisboa
Lisbon, Portugal
renato_ferreira@iscte-iul.pt

***Abstract*- Drone have become a common in our day-to day lives, being used for various purposes, such as surveillance, area recognition and recreational use. With the increased use, it is important to establish control mechanism to ensure the proper functioning and safety. One of these mechanisms is the need for anti-drone systems, capable of identifying and possibly neutralizing this equipment. However, it has become clear that drones can pose a threat to public safety of used carelessly. This paper introduces a solution capable of detecting the frequency channels which are being used for communication between the remote controller and UAV. The solution consists in the use of a Software Defined Radio (SDR) platform named HackRF. The solution is developed in an SDK called GNU Radio.**

***Keywords*- Software Defined Radio (SDR), Unmanned aerial vehicle (UAV), GNU Radio, drone, Spectrum Analyser**

I. Introduction

UAVs or as they are more known, drones, are equipment that can be controlled remotely. Initially they were developed for military purposes, such as reconnaissance and attacks, but nowadays these devices are available to everyone. Most of the drones are controlled from the ground using a controller, however there are equipment (normally used for industrial or military purposes) that can navigate predefined routes.

There are several ways to carry out detection and consequently its neutralization, being the use of electronic systems the most effective, such as jammers. Drones typically require a radio navigation (GPS) signals and RF signals between the controller and the aircraft to operate. Given their flexibility, analysing the spectrum using a SDR is a potentially effective approach to detect these signals.

The aim of this work is to develop a low-cost system capable of identifying the frequencies used by unauthorised drones. The solution was implemented using development toolkit named GNU Radio, that provides signal processing to implements a SDR systems. To do this, it was necessary to study the best way to detect the frequencies ranges used for drone communications and develop a scanning mechanism to detect signals in them, considering the possible presence of other radio frequencies signals.

II. UAV Communications

A. Drones

Drones are aircrafts designed to operate remotely, establishing a connection with a controller. These devices are divided into two parts, the drone and the controller.

Inside the drone, there are hardware components responsible for operation of the aircraft. The components directly responsible for the movement are the propellers and motors. The propellers generate the thrust to move the drone, and each one is connected to a motor. A drone can have a diverse number of propellers, the most common being 4 (quadcopter). UAV also have a battery, to power the motors, and multiple electronic components, such as a flight controller, a power distribution board and an electronic speed controller. These components are responsible for processing the information received (ex: movement actions from the controller, sensors, etc), passing it to the motors. [1]

The controller is an external component that can be used to control the direction, speed, angle and altitude of the aircraft, using an RF signal to communicate with the drone.

B. Drone communications

As mentioned in the previous subchapter, a drone consists in two devices: the aircraft and the controller. For the drone being able to operate, it is necessary to establish a communication via radio frequency signals (RF signals) between the two devices to send steering, telemetry and multimedia commands. Some aircrafts can also communicate with a GNSS System to obtain orientation and navigation information. This type of communication is used in drones that can fly autonomously or a return to base functionality. The following table shows the different frequency ranges used for these signals, as well as the transmission systems used by various brands.

Table 1- Frequency Bands used in drone communication [2]

Communication Types	Frequency Band	Transmission system
Controller (Telemetry)	27 MHZ 35 MHz 72 MHz 433 MHz 968 MHz 2.4 GHz 5.8GHz	DSM2 (Spektrum) DSMX (Spektrum) ACCESS (FrSky) FASST (Futaba) S-FHHS (Futaba) T-FHHS (Futaba) OcuSync (DJI) Lightbridge (DJI)
Multimedia	328-334 MHz 433 MHz 2.4 GHz 5.8 GHz	
Navigation	L1: 1.563 – 1.587 GHz L2: 1.215 – 1.2396 GHz L5: 1.164 – 1.189 GHz G1: 1.593 – 1.610 GHz G2: 1.237 – 1.254 GHz G3: 1.189 – 1.214 GHz E1: 1.559 – 1.592 GHz E5a/b: 1.164 – 1.215 GHz E6: 1.260 – 1.300 GHz	

The two most used band for drones' communications are the 2.4 GHz and the 5.8 GHz.

The 2.4 GHz frequency band is the most used for drone communication, as it is widely available and compatible with various controllers and being able to provide a reasonable range. However, this band is used by a lot of other equipment communications, like Bluetooth and Wi-Fi, which can result in some interference.

The 5.8 GHz frequency band is also being more adopted, especially by equipment with built-in cameras (which nowadays is most of the cases). It offers less signal interference from other wireless equipment and better quality of transmission, which is very important for sending

Anexo B- Código do bloco Spectrum Detection

```
import numpy as np
import statistics
from gnuradio import gr
import pmt
import scipy
import time
import os
from datetime import datetime
import scipy.special as scs

class MyPythonBlock(gr.sync_block):
    def __init__(self, freq = 2400000000, directory = None, frequency_spectrum = 0):
        gr.sync_block.__init__(
            self,
            name='Spectrum Detection',
            in_sig=[(np.float32, 2048)],
            out_sig=None
        )
        self.freq = freq
        self.buffer = 0
        self.lnbin = []
        self.EC = 0
        self.find = False
        self.base_complete = False
        self.message_port_register_out(pmt.intern("freq"))
        self.message_port_register_out(pmt.intern("state"))
        self.handle_clear_base()
        self.directory = directory
        self.frequency_spectrum = frequency_spectrum
        self.channels = {
            1:(2402,2422),
```

```
    2:(2407,2427),
    3:(2412,2432),
    4:(2417,2437),
    5:(2422,2442),
    6:(2427,2447),
    7:(2432,2452),
    8:(2437,2457),
    9:(2442,2462),
    10:(2447,2467),
    11:(2452,2472),
    12:(2457,2477),
    13:(2462,2482),
    32:(5150,5170),
    36:(5170,5190),
    40:(5190,5210),
    44:(5210,5230),
    144:(5710,5730),
    149:(5735,5755),
    153:(5755,5775),
    157:(5775,5795),
    161:(5795,5815),
    165:(5815,5835),
    169:(5835,5855),
}

init = 0
if (frequency_spectrum == 0):
    init = 10
elif( frequency_spectrum == 1):
    init =10
elif( frequency_spectrum == 2):
    init =11
elif( frequency_spectrum == 3):
    init =14
```

```
        self.detect = [False]*init
        self.noise = [0]*init
        self.detectenergy = [0]*init
        self.i= 0

        self.start_time = time.time()



    def handle_clear_base(self):
        directory = r'C:\Users\MSI\Desktop\freq_logs'
        if not os.path.exists(directory):
            return
        for filename in os.listdir(directory):
            print("Deleting files")
            file_path = os.path.join(directory, filename)
            if os.path.isfile(file_path):
                os.remove(file_path)

    def handle_clear_power(self):
        directory = r'C:\Users\MSI\Desktop\freq_logs'
        if not os.path.exists(directory):
            return
        for filename in os.listdir(directory):
            if not filename.startswith("base"):
                file_path = os.path.join(directory, filename)
                if os.path.isfile(file_path):
                    os.remove(file_path)



    def estimate(self,spectrum, Threshold):
        spectrum = np.append(spectrum, np.zeros(1))
        length = len(spectrum)
        initialindex = None
        finalindex = None
```

```
percentage = None
bins = []
bins1 = []
index = 0
length_bins = 0
buffer = 0
binsdata = []
binsdata1 = []

while index < length:
    if spectrum[index] > Threshold:
        buffer = 0
        bins1.append(index)
        binsdata1.append(spectrum[index])
    else:
        if (len(bins1)!=0):
            bins1.append(index)
            binsdata1.append(spectrum[index])
        buffer += 1
    if buffer > 20:
        if len(bins) == 0:
            bins = bins1
            binsdata = binsdata1
            binsdata1= []
            bins1 = []
        else:
            if len(bins1) > len(bins):
                bins = bins1
                binsdata = binsdata1
                binsdata1= []
                bins1 = []
```

```
            index += 1
        if len(bins1) >len(bins):
            bins = bins1
            binsdata = binsdata1
        length_bins = len(bins)
        if length_bins > 300:
            initialindex = bins [0]
            finalindex = bins[-1]
            percentage = length_bins / length
            if statistics.median(binsdata) > Threshold *1.5:
                return self.estimate(spectrum, Threshold*2)
            #lnbins.append(len(bins))
            bins = []
        return initialindex, finalindex, percentage



    def spectrum_energy(self,spectrum):
        spectrum = np.append(spectrum, np.zeros(1))
        length = len(spectrum)
        energy  = 0
        index = 0
        while index < length:
            energy += spectrum[index]
            index += 1

        energy = energy/ length
        return energy

    def send_message(self,freq):
        try:
            msg = pmt.cons(pmt.intern("freq"), pmt.from_double(freq))
            self.message_port_pub(pmt.intern("freq"), msg)
        except Exception as e:
            print(f"Error sending message: {e}")
```

```
    def sleep(self, seconds):
        time.sleep(seconds)

    def count_lines(self,file_path):
        with open(file_path, 'r') as file:
            lines = file.readlines()
            return len(lines)

    def write_file (self, filename, energy,  initialindex = None, finalindex = None, percentage = None):
        if os.path.exists(filename):
            with open(filename, 'a') as file:
                if(initialindex != None and finalindex != None):
                                file.write("%.2f,  %.2f,  %.2f,  %.2f\n"  %  (energy,  initialindex, finalindex,percentage))
                else:
                    file.write("%.2f\n" % (energy))
        else:
            with open(filename, 'w') as file:
                if(initialindex != None and finalindex != None):
                    file.write("%.2f, %.2f, %.2f\n" % (energy, initialindex, finalindex))
                else:
                    file.write("%.2f\n" % (energy))

    def mean_energy (self, filename):
        mean_value = 0
        if os.path.exists(filename):
            with open(filename, 'r') as file:
                numbers = []
                for line in file:
                    stripped_line = line.strip()
                    if stripped_line:
                        number = float(stripped_line.split(',')[0])
                        numbers.append(number)
```

```
        if numbers:
            # Calculate the mean
            mean_value = sum([num for num in numbers if num > 2]) / len(numbers)
            #print (mean_value)

    return mean_value

def percentage_signal (self, filename):
    if os.path.exists(filename):
        with open(filename, 'r') as file:
            numbers = []
            for line in file:
                stripped_line = line.strip()
                if stripped_line:
                    number = float(stripped_line.split(',')[0])
                    numbers.append(number)
        if numbers:
            # Calculate the mean
            percentage =  len([num for num in numbers if num > 4]) /len(numbers)
            #print (mean_value)

    return  percentage

def channel_detector (self, detect, detectenergy):
    channel = 0
    Lband =0
    min_freq = 0
    max_freq = 0



    for i in range(len(detect) - 2):
        nchannels = 0
        j = i
        while(j< len (detect) and detect[j]):
```

```
            nchannels += 1
            j += 1
        if nchannels == 2:
            Lband = 10
            break
        elif nchannels == 3:
            Lband = 20
            value_1= detectenergy[i]
            value_2=detectenergy[i+1]
            value_3 = detectenergy[i+2]

            if value_2 > value_1 and value_1 > value_3:
                channel = ((i+1)*2)
            else:
                channel = ((i+1)*2) -1
            break
        elif nchannels == 4:
            Lband = 20
            value_1= detectenergy[i+1]
            value_2=detectenergy[i+2]

            if value_2 > value_1:
                channel = ((i+1)*2)
            else:
                channel = ((i+1)*2) -1
            break
        elif nchannels >5:
            Lband = 40
            break
    if (Lband != 0):
        if i == 0 :
            min_freq = 2400 +(i*10)
        else:
            min_freq = 2400 + (i-1) *10
```

```
            max_freq = 2400+ (i+channel) *10

        return (channel, min_freq, max_freq, Lband)


    def getLimits (self, filename):
        LimitIni = 0
        LimitFin = 0

        if os.path.exists(filename):
            with open(filename, 'r') as file:
                seconds_values = []
                thirds_values = []
                for line in file:
                    stripped_line = line.strip()
                    if stripped_line:
                        parts = stripped_line.split(',')
                        if len(parts) > 2:
                            second_value = float(parts[1])
                            third_value = float(parts[2])
                            seconds_values.append(second_value)
                            thirds_values.append(third_value)

                if seconds_values:
                    LimitIni = statistics.median(seconds_values)

                if thirds_values:
                    LimitFin = statistics.median(thirds_values)
        return  LimitIni, LimitFin

    def calcLimits (self,freq_mean_dict, freq_range ):
        LimitIni = None
        LimitFin = None
        valueaux = 0
```

```
        valueaux1 = 0
        indexaux = next(iter(freq_mean_dict))
        for index, value in freq_mean_dict.items():
            if LimitIni is None and value[1]>0:
                LimitIni = (20/2048) *  value[0] + (freq_range + (index*10)) - 10
            else:
                if index == indexaux+1:
                    if  (value[0]<valueaux and (valueaux >1650 or (valueaux >1024 and LimitFin is
None))):
                        aux1= (20/2048) *  value[0] + (freq_range + (index*10) ) - 10
                        if (aux1-LimitIni<5 and aux1-LimitIni >0):
                            LimitIni = (LimitIni+ aux1)/2
                        LimitFin = (20/2048) *  value[1] + (freq_range + (index*10) ) - 10
                    elif value[0]<1024 and LimitFin is not None:
                        aux2 =(20/2048) *  value[1] + (freq_range + (index*10) ) - 10
                        if (aux2-LimitFin<5 and aux1-LimitFin >0):
                            LimitFin = (LimitFin+ aux2)/2
                        break
                    elif ((value[0] == 0 or value[1] == 0)or LimitIni is not None) and LimitFin is None:
                        LimitIni=None
                        LimitFin=None
                    else:
                        LimitFin = (20/2048) *  valueaux + (freq_range + (indexaux*10) ) - 10
                        break
                elif (value[0] == 0 or value[1] == 0) and LimitFin is None:
                        LimitIni=None
                        LimitFin=None
                elif (LimitFin is not None and LimitIni is not None) and LimitFin-LimitIni <12:
                        LimitIni=None
                        LimitFin=None
                else:
                    if (LimitFin is None and LimitIni is not None):
                        LimitIni = (20/2048) *  value[0] + (freq_range + (index*10)) - 10
                    elif LimitFin is not None:
```

```
                    break
            indexaux = index
            valueaux= value[1]
            valueaux1 = value[0]
        return LimitIni, LimitFin

    def find_closest_channel(self, freqmin, freqmax):
        closest_channel = None
        closest_diff = float('inf')

        for channel, (lower_freq, upper_freq) in self.channels.items():
            diff_min = abs(freqmin - lower_freq)
            diff_max = abs(freqmax - upper_freq)

            total_diff = diff_min + diff_max



            if total_diff < closest_diff:
                closest_diff = total_diff
                closest_channel = channel

        return closest_channel




    def work(self, input_items, output_items):



        if self.directory is None:
            print("Por favor insira uma diretoria")
        else:

            file_base = "base_power_%.0fMHz" % (self.freq / 1e6)
```

```
file_scan = "power_%.0fMHz" % (self.freq / 1e6 )
directory = self.directory.replace('\\', '\\\\')
if not os.path.exists(directory):
    os.makedirs(directory)

filenamebase = os.path.join(directory, f"{file_base}.txt")
filenamepower = os.path.join(directory, f"{file_scan}.txt")

if os.path.isfile(filenamebase) and self.base_complete == True:
    self.base_complete = True
else:
    self.base_complete = False
    msg = pmt.cons(pmt.intern("state"), pmt.from_bool(self.base_complete))
    self.message_port_pub(pmt.intern("state"), msg)
#self.buffer+=1
in0 = input_items[0]
in1 = input_items[1]

Avg = np.mean(in0)

NoisePower = in1 ** 2
NoiseAvg = statistics.mean(NoisePower)

var = np.var(NoisePower)
stdev = np.sqrt(var)

Qinv = np.sqrt(2) * scs.erfinv(1 - 2 * 0.6)

try:
    spectrum = np.array(in0)

    if (len(spectrum)> 1):
        spectrum = spectrum[:1]
```

```
energy = self.spectrum_energy(spectrum)



if self.base_complete:
    initialindex, finalindex, percentage = self.estimate(spectrum, 2)
    self.write_file(filenamepower, energy,initialindex ,finalindex, percentage)
    current_time = time.time()

    if current_time - self.start_time >= 1:

        self.detectenergy[self.i] = self.mean_energy(filenamepower)

        percentage_base = self.percentage_signal(filenamebase)
        percentage_power = self.percentage_signal(filenamepower)

        if (
            (percentage_base > 0.6 and percentage_power > 0.85) or
            #percentage_base <= 0.4 and (
                percentage_power > (
                    0.01 if percentage_base == 0 else
                    percentage_base * 4 if percentage_base < 0.01 else
                    percentage_base * 1.85 if percentage_base < 0.1 else
                    percentage_base * 1.2
                )
            #)
        ):
            self.find = True

            self.detect[self.i] = True

            if percentage_base> 0.6:
                self.noise [self.i] = self.freq
```

```
                self.start_time = time.time()

                if self.frequency_spectrum == 0:
                    if self.freq < 2490000000:
                        self.freq += 10000000
                        self.i += 1
                    elif self.freq == 2490000000:

                        if (not(self.find)):
                            print ("Nenhum sinal foi encontrado na gama dos 2,4 GHz")
                        else:
                            freq_mean_dict = {}
                            for i in range(len(self.detect)):
                                if(self.detect[i]):
                                    freqfile = 2400 +i*10
                                            Limitini, Limitfin = self.getLimits(os.path.join(directory,
f"power_{freqfile}MHz.txt"))
                                    freq_mean_dict[i] = (Limitini, Limitfin)
                            LimitIniFinal, LimitFinFinal = self.calcLimits(freq_mean_dict, 2400)
                            channel = None

                            if LimitIniFinal is not None and LimitFinFinal is not None:
                                channel = self.find_closest_channel(LimitIniFinal,LimitFinFinal)
                                if LimitFinFinal-LimitIniFinal > 12 and channel is not None:
                                    result = "Sinal com uma largura de banda de " + str(LimitFinFinal-
LimitIniFinal)+ "MHz foi encontrado no canal " + str(channel) + " no range de 2,4 GHz entre
as frequências " + str(LimitIniFinal) + " e "+ str(LimitFinFinal)
                                else:
                                    result ="Nenhum sinal foi encontrado na gama dos 2,4 GHz"
                            else:
                                result = "Nenhum sinal foi encontrado na gama dos 2,4 GHz"

                            if self.noise:
                                result = result + ", porém foi detetado ruído perto das gamas de "
```

```
                    for i, noise_value in enumerate(self.noise):
                        if noise_value != 0:
                            noise_value =noise_value/1000000
                            if i == len(self.noise) - 1:
                                result = result + " e " + str(int(noise_value))
                            else:
                                result = result  + str(int(noise_value))+ ", "

                print(result)

                self.detect = [False]*10
                self.detectenergy = [0]*10
                self.find = False
                self.handle_clear_power()
                self.i = 0
                self.freq = 5170000000
            elif self.freq < 5270000000:
                self.freq += 10000000
                self.i += 1
            elif self.freq == 5270000000:

                if (not(self.find)):
                    print ("Nenhum sinal foi encontrado na gama dos 5 GHz")
                else:
                    freq_mean_dict = {}
                    for i in range(len(self.detect)):
                        if(self.detect[i]):
                            freqfile = 5170 +i*10
                                Limitini, Limitfin = self.getLimits(os.path.join(directory,
f"power_{freqfile}MHz.txt"))
                            freq_mean_dict[i] = (Limitini, Limitfin)
                    LimitIniFinal, LimitFinFinal = self.calcLimits(freq_mean_dict, 5170)
                    channel = None
```

```
                    if LimitIniFinal is not None and LimitFinFinal is not None:
                        channel = self.find_closest_channel(LimitIniFinal,LimitFinFinal)
                        if LimitFinFinal-LimitIniFinal > 12 and channel is not None:
                            result = "Sinal com uma largura de banda de " + str(LimitFinFinal-
LimitIniFinal)+ "MHz foi encontrado no canal " + str(channel) + " no range de 5 GHz entre as
frequências " + str(LimitIniFinal) + " e "+ str(LimitFinFinal)
                        else:
                            result ="Nenhum sinal foi encontrado na gama dos 5 GHz"
                    else:
                        result = "Nenhum sinal foi encontrado na gama dos 5 GHz"

                    if self.noise:
                        result = result + ", porém foi detetado ruído perto das gamas de "
                        for i, noise_value in enumerate(self.noise):
                            if noise_value != 0:
                                noise_value =noise_value/1000000
                                if i == len(self.noise) - 1:
                                    result = result + " e " + str(int(noise_value))
                                else:
                                    result = result + ", " + str(int(noise_value))

                print(result)

                self.detect = [False]*10
                self.detectenergy = [0]*10
                self.find = False
                self.handle_clear_power()
                self.i = 0
                self.freq = 5720000000
            elif self.freq < 5850000000:
                self.freq += 10000000
                self.i += 1
            else:
                if (not(self.find)):
```

```
                print ("Nenhum sinal foi encontrado na gama dos 5,8 Ghz")
            else:
                freq_mean_dict = {}
                for i in range(len(self.detect)):
                    if(self.detect[i]):
                        freqfile = 5720 +i*10
                                Limitini, Limitfin = self.getLimits(os.path.join(directory,
f"power_{freqfile}MHz.txt"))
                        freq_mean_dict[i] = (Limitini, Limitfin)
                print (freq_mean_dict)
                LimitIniFinal, LimitFinFinal = self.calcLimits(freq_mean_dict, 5720)
                print (LimitIniFinal, LimitFinFinal)
                channel = None

                if LimitIniFinal is not None and LimitFinFinal is not None:
                    channel = self.find_closest_channel(LimitIniFinal,LimitFinFinal)
                    if LimitFinFinal-LimitIniFinal > 12 and channel is not None:
                        result = "Sinal com uma largura de banda de " + str(LimitFinFinal-
LimitIniFinal)+ "MHz foi encontrado no canal " + str(channel) + " no range de 5,8 GHz entre
as frequências " + str(LimitIniFinal) + " e "+ str(LimitFinFinal)
                    else:
                        result ="Nenhum sinal foi encontrado na gama dos 5,8 GHz"
                else:
                    result = "Nenhum sinal foi encontrado na gama dos 5,8 GHz"

                if any(noise !=0 for noise in self.noise):
                    result = result + ", porém foi detetado ruído perto das gamas de "
                    for i, noise_value in enumerate(self.noise):
                        if noise_value != 0:
                            noise_value =noise_value/1000000
                            if i == len(self.noise) - 1:
                                result = result + " e " + str(int(noise_value))
                            else:
                                result = result  + str(int(noise_value))+ ", "
```

```
                print(result)

                self.detect = [False]*10
                self.detectenergy = [0]*10
                self.find = False
                self.handle_clear_power()
                self.i = 0
                self.freq = 2400000000

            if self.frequency_spectrum == 1:

                if  self.freq > 2500000000 :
                    self.freq = 2400000000
                    self.i  = 0



                if self.freq < 2490000000:
                    self.freq += 10000000
                    self.i += 1
                else:
                    if (not(self.find)):
                        result = "Nenhum sinal foi encontrado na gama dos 2,4 GHz"
                    else:
                        freq_mean_dict = {}
                        for i in range(len(self.detect)):
                            if(self.detect[i]):
                                freqfile = 2400 +i*10
                                        Limitini, Limitfin = self.getLimits(os.path.join(directory,
f"power_{freqfile}MHz.txt"))
                                freq_mean_dict[i] = (Limitini, Limitfin)
                        LimitIniFinal, LimitFinFinal = self.calcLimits(freq_mean_dict, 2400)
                        channel = None
```

```
                    if LimitIniFinal is not None and LimitFinFinal is not None:
                        channel = self.find_closest_channel(LimitIniFinal,LimitFinFinal)
                        if LimitFinFinal-LimitIniFinal > 12 and channel is not None:
                            result = "Sinal com uma largura de banda de " + str(LimitFinFinal-
LimitIniFinal)+ "MHz foi encontrado no canal " + str(channel) + " no range de 2,4 GHz entre
as frequências " + str(LimitIniFinal) + " e "+ str(LimitFinFinal)
                        else:
                            result ="Nenhum sinal foi encontrado na gama dos 2,4 GHz"
                    else:
                        result = "Nenhum sinal foi encontrado na gama dos 2,4 GHz"

                    if any(noise !=0 for noise in self.noise):
                        result = result + ", porém foi detetado ruído perto das gamas de "
                        for i, noise_value in enumerate(self.noise):
                            if noise_value != 0:
                                noise_value =noise_value/1000000
                                if i == len(self.noise) - 1:
                                    result = result + " e " + str(int(noise_value))
                                else:
                                    result = result  + str(int(noise_value))+ ", "

                print(result)

                self.detect = [False]*10
                self.detectenergy = [0]*10
                self.find = False
                self.handle_clear_power()
                self.i = 0
                self.freq = 2400000000

            if self.frequency_spectrum == 2:
                if self.freq < 5170000000 or self.freq > 5270000000 :
                    self.freq = 5170000000
                    self.i = 0
```

```
                if self.freq < 5270000000:
                    self.freq += 10000000
                    self.i += 1
                else:
                    if (not(self.find)):
                        result = "Nenhum sinal foi encontrado na gama dos 5 GHz"
                    else:
                        freq_mean_dict = {}
                        for i in range(len(self.detect)):
                            if(self.detect[i]):
                                freqfile = 5170 +i*10
                                        Limitini, Limitfin = self.getLimits(os.path.join(directory,
f"power_{freqfile}MHz.txt"))
                                freq_mean_dict[i] = (Limitini, Limitfin)
                        print (freq_mean_dict)
                        LimitIniFinal, LimitFinFinal = self.calcLimits(freq_mean_dict, 5170)
                        print (LimitIniFinal, LimitFinFinal)
                        channel = None

                        if LimitIniFinal is not None and LimitFinFinal is not None:
                            channel = self.find_closest_channel(LimitIniFinal,LimitFinFinal)
                            if LimitFinFinal-LimitIniFinal > 12 and channel is not None:
                                result = "Sinal com uma largura de banda de " + str(LimitFinFinal-
LimitIniFinal)+ "MHz foi encontrado no canal " + str(channel) + " no range de 5 GHz entre as
frequências " + str(LimitIniFinal) + " e "+ str(LimitFinFinal)
                            else:
                                result ="Nenhum sinal foi encontrado na gama dos 5 GHz"
                        else:
                            result = "Nenhum sinal foi encontrado na gama dos 5 GHz"

                        if any(noise !=0 for noise in self.noise):
                            result = result + ", porém foi detetado ruído perto das gamas de "
                            for i, noise_value in enumerate(self.noise):
                                if noise_value != 0:
```

```
                    noise_value =noise_value/1000000
                    if i == len(self.noise) - 1:
                        result = result + " e " + str(int(noise_value))
                    else:
                        result = result  + str(int(noise_value))+ ", "

            print(result)

            self.detect = [False]*11
            self.detectenergy = [0]*11
            self.find = False
            self.handle_clear_power()
            self.i = 0
            self.freq = 5170000000

    if self.frequency_spectrum == 3:

        if self.freq < 5720000000:
            self.freq = 5720000000
            self.i = 0

        if self.freq < 5850000000:
            self.freq += 10000000
            self.i += 1
        else:
            if (not(self.find)):
                result = "Nenhum sinal foi encontrado na gama dos 5,8 GHz"
            else:
                print (self.detect)
                freq_mean_dict = {}
                for i in range(len(self.detect)):
                    if(self.detect[i]):
                        freqfile = 5720 +i*10
```

```
                                Limitini, Limitfin = self.getLimits(os.path.join(directory,
f"power_{freqfile}MHz.txt"))
                                freq_mean_dict[i] = (Limitini, Limitfin)
                        print (freq_mean_dict)
                        LimitIniFinal, LimitFinFinal = self.calcLimits(freq_mean_dict, 5720)
                        print (LimitIniFinal, LimitFinFinal)
                        channel = None

                        if LimitIniFinal is not None and LimitFinFinal is not None:
                            channel = self.find_closest_channel(LimitIniFinal,LimitFinFinal)
                            if LimitFinFinal-LimitIniFinal > 12 and channel is not None:
                                result = "Sinal com uma largura de banda de " + str(LimitFinFinal-
LimitIniFinal)+ "MHz foi encontrado no canal " + str(channel) + " no range de 5,8 GHz entre
as frequências " + str(LimitIniFinal) + " e "+ str(LimitFinFinal)
                            else:
                                result ="Nenhum sinal foi encontrado na gama dos 5,8 GHz"
                        else:
                            result = "Nenhum sinal foi encontrado na gama dos 5,8 GHz"

                        if self.noise:
                            result = result + ", porém foi detetado ruído perto das gamas de "
                            for i, noise_value in enumerate(self.noise):
                                if noise_value != 0:
                                    noise_value =noise_value/1000000
                                    if i == len(self.noise) - 1:
                                        result = result + " e " + str(int(noise_value))
                                    else:
                                        result = result + ", " + str(int(noise_value))

                    print(result)

                    self.detect = [False]*14
                    self.detectenergy = [0]*14
                    self.find = False
```

```
                self.handle_clear_power()
                self.i = 0
                self.freq = 5720000000

            self.send_message(self.freq)



    else:
        self.write_file(filenamebase, energy)
        current_time = time.time()
        if current_time - self.start_time >= 10:
            self.start_time = time.time()

            if self.frequency_spectrum == 0:
                if self.freq < 2490000000:
                    self.freq += 10000000
                elif self.freq == 2490000000:
                    self.freq = 5170000000
                elif self.freq < 5270000000:
                    self.freq += 10000000
                elif self.freq == 5270000000:
                    self.freq = 5720000000
                elif self.freq < 5850000000:
                    self.freq += 10000000
                else:
                    self.freq = 2400000000

                    self.base_complete = True
                    try:
                        msg = pmt.cons(pmt.intern("state"), pmt.from_bool(True))
                        self.message_port_pub(pmt.intern("state"), msg)
                    except Exception as e:
                        print(f"Error sending message: {e}")
```

```
if self.frequency_spectrum == 1:
    if self.freq < 2490000000:
        self.freq += 10000000
    else:
        self.freq = 2400000000

        self.base_complete = True
        try:
            msg = pmt.cons(pmt.intern("state"), pmt.from_bool(True))
            self.message_port_pub(pmt.intern("state"), msg)
        except Exception as e:
            print(f"Error sending message: {e}")
if self.frequency_spectrum == 2:

    if self.freq < 5170000000 or self.freq > 5270000000 :
        self.freq = 5170000000
        self.i = 0
        self.detect = [False]*11
        self.noise = [0]*11
        self.detectenergy = [0]*11

    if self.freq < 5270000000:
        self.freq += 10000000
    else:
        self.freq = 5170000000

        self.base_complete = True
        try:
            msg = pmt.cons(pmt.intern("state"), pmt.from_bool(True))
            self.message_port_pub(pmt.intern("state"), msg)
        except Exception as e:
            print(f"Error sending message: {e}")

if self.frequency_spectrum == 3:
```

```
                if self.freq < 5850000000:
                    self.freq += 10000000
                    self.detect = [False]*14
                    self.noise = [0]*14
                    self.detectenergy = [0]*14
                else:
                    self.freq = 5720000000

                    self.base_complete = True
                    try:
                        msg = pmt.cons(pmt.intern("state"), pmt.from_bool(True))
                        self.message_port_pub(pmt.intern("state"), msg)
                    except Exception as e:
                        print(f"Error sending message: {e}")



            self.send_message(self.freq)


        spectrum = np.array(input_items[0])

    except Exception as e:
        print("Exception:", e)



    return len(input_items)
```