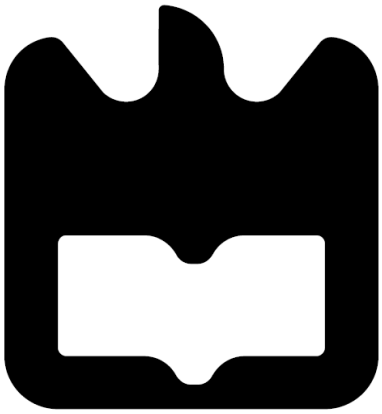
Universidade de Aveiro
2018
Departamento de
Eletrónica, Telecomunicações e Informática

Bruno Ramos Maximino

Bancada laboratorial para investigação e ensino de sistemas de comunicações digitais definidos por *software*.



Universidade de Aveiro
2018

Departamento de
Eletrónica, Telecomunicações e Informática

Bruno Ramos Maximino

Bancada laboratorial para investigação e ensino de sistemas de comunicações digitais definidos por *software*.

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Eletrónica e Telecomunicações, realizada sob a orientação científica do Professor Doutor Paulo Miguel Nepomuceno Pereira Monteiro, Professor Associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e sob a coorientação científica do Doutor Fernando Pedro Pereira Guiomar, Investigador Auxiliar do Instituto de Telecomunicações de Aveiro.

Apoio financeiro do ORCIP no âmbito do projeto
CENTRO-01-0145-FEDER-022141.

Dedico este trabalho aos meus pais e à Catarina, pelo apoio demonstrado ao longo deste percurso.

o júri / the jury

presidente / president

Prof. Doutor Arnaldo Silva Rodrigues de Oliveira
Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor Fernando José da Silva Velez
Professor Auxiliar do Departamento de Engenharia Eletromecânica da Universidade da Beira Interior

Prof. Doutor Paulo Miguel Nepomuceno Pereira Monteiro
Professor Associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

agradecimentos

Em primeiro lugar, gostaria de deixar o meu especial agradecimento aos meus pais. À minha mãe, pelo apoio incondicional demonstrado ao longo deste percurso. Ao meu pai, pelos conselhos que me ajudaram a fazer as escolhas corretas neste percurso. A ambos, o meu muito obrigado por terem sempre acreditado em mim. Sem eles não seria possível ter chegado aqui.

À Catarina, pelo apoio, paciência, conselhos e, principalmente, por ter acreditado sempre em mim.

Aos meus avós, pela preocupação e ensinamentos que sempre me foram dando.

Aos meus tios, principalmente ao meu tio Elmano, pelos conselhos e exemplo que sempre considerei ser.

Aos meus primos, principalmente à Salomé e ao João, pelo apoio e companheirismo.

Aos meus amigos, por me terem proporcionado bons momentos ao longo deste percurso. O meu obrigado por estarem sempre presentes quando precisei deles.

Ao Pedro Mostardinha, pelos bons momentos vividos aquando da realização deste trabalho, pela camaradagem e pelo espírito crítico demonstrado ao longo destes meses.

Ao meu orientador Prof. Paulo Monteiro, por me ter aceite como seu orientando, pela dedicação demonstrada ao longo deste trabalho e pela partilha de conhecimento.

Ao Fernando Guiomar, pelo contributo na parte experimental do trabalho e pela disponibilidade sempre demonstrada ao longo deste tempo.

palavras-chave

Comunicações Digitais, Rádio Definido por *Software*, Formatos de Modulação.

resumo

Esta dissertação tem como objetivo estudar e implementar um novo módulo laboratorial que permita renovar o ensino de Sistemas de Comunicação e de Sistemas de Informação. Devido à sua flexibilidade, a tecnologia que vai ser utilizada é o Rádio Definido por *Software*. Depois de analisados os principais equipamentos que usam esta tecnologia e os *softwares* compatíveis, concluiu-se que, para o propósito desta dissertação, o equipamento mais adequado é o ADALM-PLUTO, da Analog Devices, e *software* é o MATLAB, da Mathworks. Devido ao ADALM-PLUTO ser relativamente recente, antes de ser implementado o sistema de transmissão-receção final, foi necessário realizar alguns testes, tais como medição da potência de saída do ADALM-PLUTO ou otimização de certos parâmetros do sistema. Implementou-se, então, um sistema de transmissão-receção constituído por 2 ADALM-PLUTO, ligados por um cabo. Finalmente, foi avaliado o desempenho do sistema para vários formatos de modulação, desde o QPSK até ao 512QAM, e para diferentes frequências de portadora, desde 900 MHz até 3.7 GHz. O desempenho do sistema foi avaliado através do cálculo da probabilidade de erro, comparando-o com os valores teóricos obtidos. Os resultados obtidos permitiram concluir que o sistema implementado é uma solução muito interessante para o ensino de Sistemas de Comunicação e de Sistemas de Informação.

keywords Digital Communications, Software Defined Radio, Modulation Formats.

abstract This dissertation aims at study and implementation of a new laboratory module in order to renew the teaching of Communication Systems and Information Systems. The flexibility of Software Defined Radio allows its helpful implementation. The search for compatible software and main equipment for the purpose was concluded at ADALM-PLUTO, from Analog Devices, and the software is MATLAB, from MathWorks. Being ADALM-PLUTO a recent equipment, some tests, such as measurement of the output power of ADALM-PLUTO or optimization of certain system parameters, was performed before its implementation in final transmission-reception system. Then, a transmission-reception system consisting of 2 ADALM-PLUTO, connected by a cable, was implemented. At last the system performance was evaluated for various modulation formats, from QPSK to 512QAM, and for different carrier frequencies, from 900 MHz to 3.7 GHz. The system performance was evaluated by calculating the error probability, comparing it with the obtained theoretical values. The results allowed to conclude that the implemented system is an interesting solution for the teaching of Communication Systems and Information Systems.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

ADC	*Analog-to-Digital Converter*
AGC	*Automatic Gain Control*
AWGN	*Additive White Gaussian Noise*
BER	*Bit Error Rate*
CMA	*Constant Modulus Algorithm*
DAC	*Digital-to-Analog Converter*
DETI	Departamento de Eletrónica, Telecomunicações e Informática
DSA	*Dynamic Spectrum Access*
DSP	*Digital Signal Processing*
EVM	*Error Vector Magnitude*
FIR	*Finite Impulse Response*
FPGA	*Field-Programmable Gate Array*
IIO	*Industrial Input Output*
IIR	*Infinite Impulse Response*
ISI	*InterSymbol Interference*
LABVIEW	*Laboratory Virtual Instrument Engineering Workbench*
LMS	*Least Mean Squares*
LNA	*Low Noise Amplifier*
LO	*Local Oscillator*
LPF	*Low-Pass Filter*
LTE	*Long Term Evolution*
MATLAB	*MATrix LABoratory*
MIMO	*Multiple-Input and Multiple-Output*
OFDM	*Orthogonal Frequency Division Multiplexing*
PA	*Power Amplifier*
PAPR	*Peak-to-Average Power Ratio*
PCI	*Peripheral Component Interconnect*
PRBS	*Pseudo-Random Bit Sequence*
PSD	*Power Spectral Density*
QAM	*Quadrature Amplitude Modulation*

QPSK	*Quadrature Phase Shift Keying*
RAM	*Random Access Memory*
RF	*Radio Frequency*
RO	*Roll-Off* (Fator de decaimento)
RRC	*Root-Raised-Cosine*
SDR	*Software Defined Radio*
SMA	*SubMiniature version A*
SNR	*Signal-to-Noise Ratio*
TIA	*TransImpedance Amplifier*
USB	*Universal Serial Bus*
USRP	*Universal Software Radio Peripheral*

1 Introdução

1.1 Motivação e Objetivos

Ao longo das últimas décadas temos assistido a uma constante evolução no ramo das Telecomunicações. Esta evolução levou ao aparecimento de um número infindável de novas tecnologias, tais como WLAN, *Bluetooth* ou LTE. Estas tecnologias têm diferentes exigências como, por exemplo, na largura de banda ou na taxa de bit. Com o objetivo de poupar tempo e recursos financeiros na construção de protótipos de teste para cada uma das tecnologias pretendidas, tornou-se imperativo desenvolver equipamentos que se adaptassem às necessidades de um mercado cada vez mais dinâmico e inovador [1]. É, então, que surge o Rádio Definido por *Software* (*Software Defined Radio*, SDR), que permite substituir a implementação dos dispositivos de comunicação, não reconfiguráveis, baseados em *hardware*, como por exemplo os ASICs (*Application-Specific Integrated Circuit* [2]), por uma implementação reconfigurável por *software*.

O Rádio Definido por *Software* veio revolucionar a área das Comunicações Digitais, tanto ao nível das empresas como, também, ao nível das universidades. Este sistema tornou os equipamentos rádio mais flexíveis e escaláveis [3].

Atualmente, no ensino Universitário, deparamo-nos com a dificuldade de os alunos conciliarem os conceitos teóricos adquiridos com a sua aplicação prática, levando à sua desmotivação. As dificuldades sentidas pelos mesmos na compreensão dos conceitos lecionados nas aulas teóricas podem ser colmatadas com experiências laboratoriais, ajudando à conexão entre noções abstratas com o mundo real [4].

No caso específico do DETI, é utilizado o módulo MCM31, da Elettronica Veneta [5], no ensino da componente prática de Sistemas de Comunicação. Devido ao seu tamanho, torna-se quase impossível aos alunos utilizarem o módulo fora da sala de aula. Foi então que se chegou à conclusão que seria necessário reformular o método de ensino de Sistemas de Comunicação.

Esta dissertação tem o objetivo de estudar e implementar um novo módulo de ensino laboratorial que permita renovar o ensino destas matérias, tornando-o mais eficiente e atualizado. O sistema a implementar é composto por dois ADALM-PLUTO *Active Learning Module* [6] e por um computador, em que a interface entre o *hardware* e o computador será feita através do MATLAB™.

Este projeto foi particularmente motivante na medida em que este tipo de sistemas teve um crescimento exponencial ao longo dos últimos anos e, como instituição de ensino de excelência, cabe-nos a nós fornecer as ferramentas necessárias que permitam que os alunos finalizem os seus estudos com conhecimentos essenciais para a próxima geração de sistemas de comunicação.

1.2 Organização da Dissertação

Esta dissertação está estruturada em seis capítulos, começando com este que contém a motivação e os objetivos que se pretende alcançar.

No Capítulo 2, é apresentada uma breve introdução à tecnologia SDR, em que é explicado o funcionamento geral de um dispositivo SDR, quais os dispositivos existentes e os tipos de *software* que são utilizados.

O Capítulo 3 é dedicado ao dispositivo SDR escolhido, o ADALM-PLUTO. São apresentadas as suas especificações, o seu diagrama de blocos e a interface com o MATLAB.

Como este dispositivo é relativamente recente, carece de alguma informação vital para a sua utilização. Foi, por isso, necessário fazer uma caracterização da sua potência de saída para evitar provocar danos no dispositivo.

No Capítulo 4, é apresentado todo o procedimento, a configuração e os resultados obtidos na medição da potência de saída do ADAM-PLUTO.

Seguidamente, no Capítulo 5, é apresentado o sistema final implementado. Neste capítulo é apresentada a configuração utilizada, a estrutura do código MATLAB utilizado e uma breve explicação dos blocos constituintes do mesmo, os parâmetros iniciais utilizados e a otimização de alguns desses parâmetros. Na parte final do capítulo, são apresentados todos os resultados obtidos e é feita a discussão dos mesmos.

Finalmente, o Capítulo 6 apresenta a conclusão e o trabalho futuro.

São, ainda, apresentados cinco apêndices, que complementam a dissertação.

2 Rádio Definido por *Software*

Este capítulo explica o conceito de Rádio Definido por *Software*, a sua evolução e algumas das suas aplicações nos dias de hoje. É explicado, ainda que simplificado, o modo de funcionamento de um equipamento SDR, tanto na transmissão como na receção. Por fim, são apresentados alguns dos periféricos SDR existentes no mercado e o *software* que utilizam.

2.1 Introdução

De acordo com o *Wireless Innovation Forum* [7], define-se Rádio Definido por *Software* (SDR) como "rádio em que algumas ou todas as funções da camada física são definidas por *software*". Uma definição mais completa é dada pela *International Telecommunications Union* (ITU) [8] e consiste em "um transmissor e/ou recetor de rádio que emprega uma tecnologia que permite que os parâmetros operacionais de RF incluam, mas não se limitem a, faixa de frequência, tipo de modulação ou potência de saída a ser definida ou alterada pelo *software*". Por outras palavras, SDR é um sistema de rádio, reconfigurável por *software*, com a flexibilidade de operar com diferentes tecnologias de acesso de rádio. Pode implementar diferentes esquemas de modulação e formas de onda e, ainda, implementar vários padrões num único dispositivo, através de *hardware* reconfigurável e de *software* eficaz.

O termo Rádio Definido por *Software* foi introduzido por J.Mitola, em 1991 [9], como o conceito do futuro do rádio. O primeiro SDR explorado em larga escala foi no projeto *SPEAKEasy* [10], em 1999, financiado pelo governo norte-americano. Somente depois, no século XXI, e com o desenvolvimento de poderosos *chips* de processamento de sinal é que as atuais plataformas SDR foram desenvolvidas. Atualmente, temos uma grande diversidade de módulos de Rádio Definido por *Software*, como o USRP, da Ettus Research/National Instruments, ou o ADALM-PLUTO, da Analog Devices, utilizados nas mais diversas aplicações de radiofrequência, tais como, GPS, LTE, Radar, WLAN ou *Bluetooth* [1]. A arquitetura SDR permitiu, ainda, a implementação do Radar Definido por *Software* (*Software Defined Radar*) [11] e do DSA (*Dynamic Spectrum Access*) [12].

Uma frase ilustrativa do que um dispositivo SDR representa nos dias de hoje é esta: “De viajantes de negócios a soldados no campo de batalha, a tecnologia SDR tem como objetivo reduzir custos ao fornecer aos utilizadores finais acesso a comunicações sem fio omnipresentes, permitindo que comuniquem com quem precisam, sempre que precisam e da maneira que acharem apropriado” [7].

Para além da sua grande flexibilidade, os SDRs oferecem muitas outras vantagens comparativamente às implementações tradicionais de rádio, exclusivamente por *hardware*. Reduzem a complexidade do *hardware* nos equipamentos de rádio, pois grande parte do processamento de sinal passa a ser implementado por *software*, fazendo com que seja mais simples de produzir e, assim, é reduzido o tempo de desenvolvimento do produto. Isto tudo faz com que o custo final dos equipamentos baixe significativamente [1].

Devido à sua multifuncionalidade, isto é, o facto de um único SDR suportar várias tecnologias de rádio, os SDRs tornaram-se num bem precioso, tanto das empresas como das universidades, na medida em que é possível poupar imenso em recursos financeiros porque podem ser reutilizados em diferentes aplicações [13]. São, também, mais robustos a variações de temperatura, melhorando o seu desempenho [1]. A facilidade de atualização constitui outra vantagem deste tipo de rádios [13].

Apesar de todas as vantagens, estes equipamentos trazem algumas preocupações, tais como interferências, por exemplo, em canais cuja banda de frequência se encontra alocada para serviços de emergência ou ameaças à segurança, através da introdução de *software* malicioso aos terminais do SDR.

A sincronização do relógio dos componentes do *hardware* (ADCs e DACs) com o relógio dos dispositivos do DSP representa outro desafio, na medida em que é importante evitar incompatibilidades de frequências de amostragem [13].

2.2 Funcionamento de um Sistema SDR

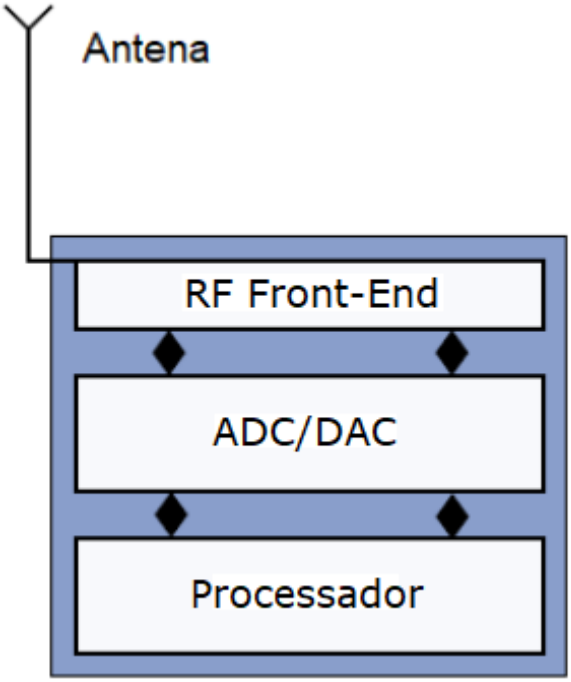


Figura 2.1: Esquema básico da arquitetura de um SDR ideal (Adaptado de [1]*).*

Na Figura 2.1 está representado um esquema simplificado de um Rádio Definido por *Software* ideal, em que se pretende que todos os componentes de um sistema de comunicação tradicional sejam implementados por *software*, deixando o mínimo possível para o lado do *hardware*.

Em SDR, o processador está encarregue do processamento digital de sinal (DSP), programado por *software*. Caso o equipamento esteja a ser utilizado como transmissor, o sinal vindo do processador vai ser convertido para o domínio analógico através de um Conversor Digital-Analógico (DAC). Se estiver a funcionar como recetor, o sinal que vem do *RF Front-End* vai ser novamente convertido para o domínio digital através de um Conversor Analógico-Digital (ADC). O bloco *RF Front-End* é responsável pela transmissão e receção do sinal, pelo acoplamento entre o rádio e a antena e, ainda, pela interface entre banda base e a radiofrequência (RF).

Seguidamente, vai ser explicado com maior detalhe o processo de transmissão e de receção de um sistema SDR, sendo que, dependendo do SDR utilizado, os diagramas de blocos podem ser ligeiramente diferentes.

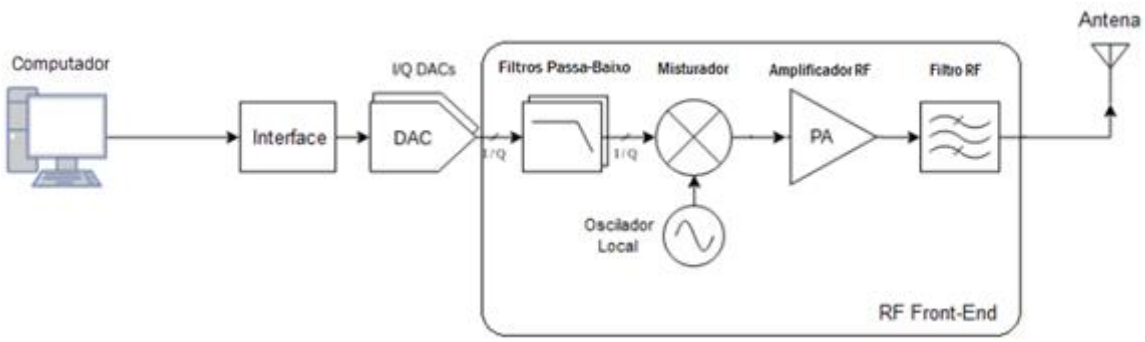


Figura 2.2: Diagrama de Blocos geral de um Transmissor SDR (Adaptado de [14]*).*

O transmissor é responsável pela geração e transmissão dos dados pretendidos. Esses dados são gerados no computador e transmitidos através de uma interface, em geral USB, *Ethernet* ou barramento PCI. O *software* instalado no computador realiza todas as operações de processamento de sinal, tais como modulação e codificação, entre outros. Seguidamente, o sinal digital em banda base é convertido em sinal analógico através de um Conversor Digital-Analógico (DAC). Para finalizar, o sinal entra no bloco *RF Front-End*, onde vai ser feito o deslocamento do espetro do sinal para a frequência de transmissão pretendida (RF). Neste caso, esta etapa é realizada usando o conceito de conversão direta *(Direct-Conversion)*, em que o deslocamento de frequência é feito numa única conversão.

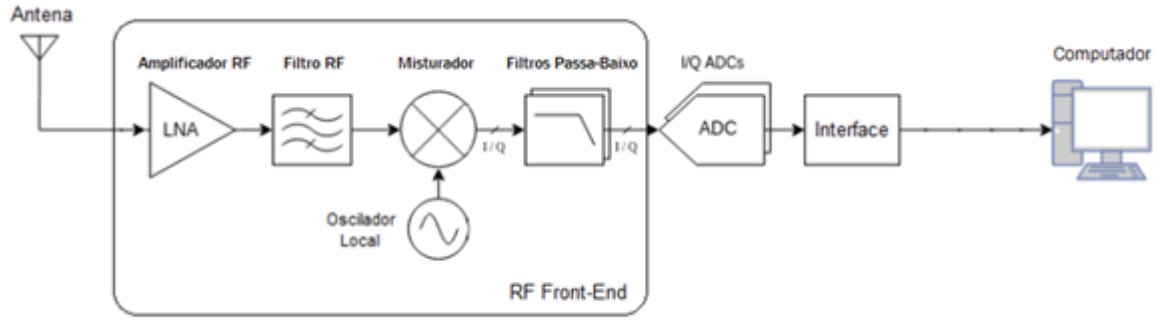


Figura 2.3: Diagrama de Blocos geral de um Recetor SDR (Adaptado de [14]*).*

No recetor, o sinal de radiofrequência (RF) é capturado pela antena e conduzido para o bloco *RF Front-End*. Nesse bloco, o sinal recebido é deslocado em frequência para a banda base. O amplificador de baixo ruído (LNA) deve estar o mais próximo possível da antena e é utilizado para amplificar sinais de baixa amplitude, sem que se aumente significativamente o ruído. Seguidamente, o sinal analógico entra num Conversor Analógico-Digital, onde é convertido num sinal digital. Para finalizar, os dados obtidos são enviados para o computador, onde vão ocorrer as diversas operações de processamento de sinal (desmodulação e descodificação, entre outras), com vista a recuperar o sinal enviado pelo transmissor.

2.3 Dispositivos de Rádio Definido por *Software*

Atualmente existe uma grande variedade de periféricos de Rádio Definido por *Software*, com as mais diversas características. Previamente à execução da componente prática desta dissertação foi necessário fazer um levantamento dos principais dispositivos existentes no mercado e suas características. A Tabela 2.1 mostra alguns desses dispositivos.

Tabela 2.1: Características de alguns periféricos de Rádio Definido por Software [6] [15] [16] .

	USRP B210	**ADALM-PLUTO**	**RTL 2832U (+ R820T2)**
Cobertura RF	70 MHz – 6 GHz	325 MHz – 3.8 GHz	24 MHz - 1766 MHz
Largura de Banda	56 MHz (1x1) / 30.72 MHz (2x2)	20 MHz	3.2 MHz
Número de bits (ADC/DAC)	12 Bits	12 Bits	8 Bits
Taxa de Amostragem (ADC/DAC)	61.44 MS/s	61.44 MS/s	3.2 MS/s
Duplex	*Half* ou *Full*	*Half* ou *Full*	-
Transmissor/Recetor	2 / 2	1 / 1	- / 1
FPGA	Xilinx Spartan 6 XC6SLX150	Xilinx Zynq Z-7010	-
Interface	USB 3.0	USB 2.0	USB 2.0
Software	MATLAB / LabVIEW / GNU Radio	MATLAB / GNU Radio	MATLAB / GNU Radio
Custo	≈ 1300 €	≈ 100 €	≈ 17 €

Depois de analisadas as principais características dos três dispositivos, chegou-se à conclusão de que, para as unidades curriculares Sistemas de Informação e Sistemas de Comunicação, o ADALM-PLUTO (Figura 2.4) seria o mais indicado. Para tal contribuiu o facto de o RTL 2832U (Figura 2.5) só efetuar receção (não tem um transmissor) e de o USRP B210 (Figura 2.6) ser muito mais dispendioso (≈13 vezes superior) relativamente ao ADALM-PLUTO. O seu baixo custo tornou-se importante, permitindo a aquisição de uma elevada quantidade de módulos, sem um investimento avultado, com o intuito de facultar o seu acesso aos alunos, dentro e fora da sala de aula.

Figura 2.4: ADALM-PLUTO, da Analog Devices.

Figura 2.5: RTL 2832U (+R820T2) [16].

Figura 2.6: USRP B210, da Ettus Research/National Instruments.

2.4 *Software* utilizado no SDR

Tanto o USRP B210 como o ADALM-PLUTO são de natureza multiplataforma, podendo ser utilizados pelos Sistemas Operativos OS X™, Windows™ e Linux™.

A tecnologia de Rádio Definido por *Software* tem por base a utilização de *software*, destacando-se o *software* responsável pelo processamento de sinal. Existem diversas opções de *software* para efetuar esse processamento, sendo que as mais utilizadas são o GNU Radio, o MATLAB e o LabVIEW.

O GNU Radio é um *software* gratuito e de código aberto, que permite criar aplicações visuais de processamento de sinal a partir da união de blocos. Pode ser utilizado juntamente com um periférico de *hardware* externo para criar SDR ou, sem qualquer *hardware*, para trabalhar num ambiente de simulação. Os blocos criados em GNU Radio podem ser escritos em Python ou C++. Hoje em dia, é muito utilizado em sistemas sem fio e em sistemas de rádio, tanto na indústria como nas universidades [17].

Outro *software* que pode ser utilizado é o LabVIEW, ambiente de programação gráfico, em linguagem conhecida por "G", criado pela empresa National Instruments™. Os programas não são escritos, mas sim desenhados, e atualmente há uma grande variedade de blocos pré-desenhados. Apesar de ser um ambiente mais gráfico, representa um custo avultado, mesmo em contexto académico [18].

O MATLAB é uma ferramenta de programação para efetuar cálculos com matrizes, detida pela empresa Mathworks. É uma linguagem de alto nível e um ambiente interativo para computação numérica, visualização e programação. Análise de dados, desenvolvimento de algoritmos e criação de modelos e aplicações são as funções mais comuns deste *software*. Para tal, pode-se programar num *script* (código fonte) ou usar um ambiente gráfico, similar ao GNU Radio, chamado Simulink. Devido à existência de licenças deste *software* para ensino, na Universidade de Aveiro, e ao facto de haver uma grande diversidade de *toolboxes* e bibliotecas de funções especializadas em comunicações digitais, este foi o *software* escolhido para a realização desta dissertação [19].

3 ADALM-PLUTO

Como já foi dito anteriormente, toda a parte laboratorial desta dissertação foi realizada com recurso ao ADALM-PLUTO *Active Learning Module*, da Analog Devices Inc. Neste capítulo, é fornecida toda a informação necessária para a utilização do ADALM-PLUTO como um dispositivo de SDR.

3.1 Introdução

O ADALM-PLUTO *Active Learning Module*, da Analog Devices Inc é um dispositivo desenvolvido especialmente para o mundo académico, cujo principal objetivo é familiarizar os alunos com os conceitos de SDR, RF e comunicações sem fios. Este equipamento educativo é bastante portátil (de dimensões reduzidas e leve) em que os alunos podem aplicar os conceitos teóricos assimilados a casos de estudo práticos, sendo uma importante mais-valia no ensino de telecomunicações.

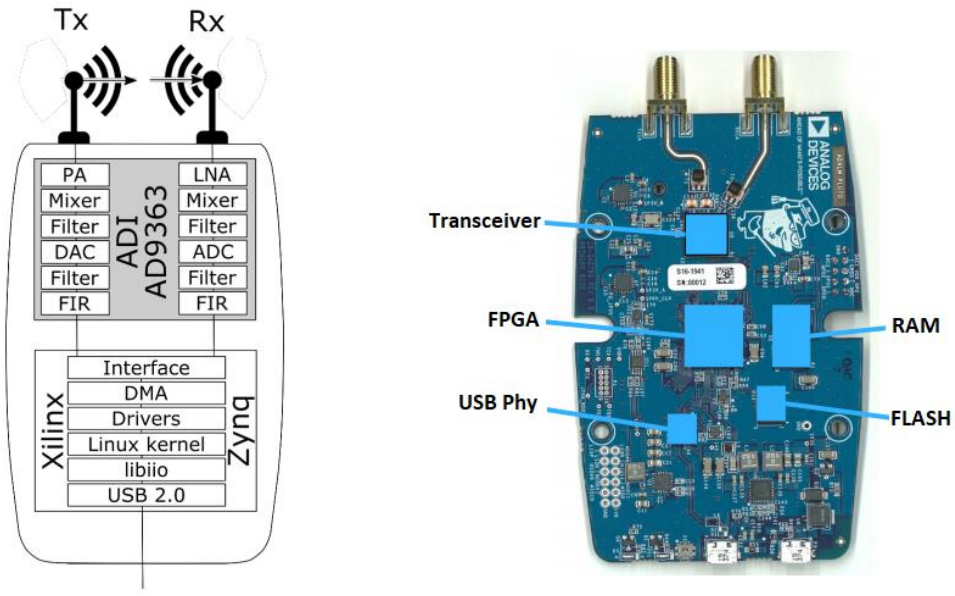


Figura 3.1: Diagrama de blocos do ADALM-PLUTO, esquerda, e imagem do interior do ADALM-PLUTO, direita (Adaptado de [20]*).*

Na imagem da esquerda da Figura 3.1, está representado o diagrama de blocos do ADALM-PLUTO, onde é possível ver que os blocos do *RF Front-End* se encontram todos no *chipset* ADI AD9363 [21]. Estes blocos vão ser abordados com maior detalhe mais à frente. Em baixo do bloco de RF, encontra-se a FPGA (Xilinx Zynq Z-7010), onde é realizada a interface entre o transcetor e o computador. O ADALM-PLUTO utiliza como Sistema Operativo interno o Linux. Com o propósito de transferir informação e controlo, utiliza os *drivers* libiio como interface entre o Linux e o USB 2.0. Libiio foi desenvolvida pela Analog Devices, com o intuito de facilitar o desenvolvimento de interfaces de *software* para dispositivos IIO [22]. Como utiliza estes *drivers*, é suportado pelos diversos Sistemas Operativos OS X™, Windows™ e Linux™. Isto permite que os estudantes tenham uma elevada flexibilidade na escolha do Sistema Operativo [23].

À direita, na Figura 3.1, é apresentada uma imagem do interior do ADALM-PLUTO, que ilustra a localização espacial dos componentes do dispositivo, onde se destaca o transcetor, a FPGA, a memória *flash*, a memória RAM e o *chip* USB.

3.2 Especificações

Nesta secção vão ser apresentadas as principais características do ADALM-PLUTO *Active Learning Module* (Tabela 3.1) e do transcetor AD9363, presente no mesmo (Tabela 3.2). As especificações completas do transcetor devem ser consultadas na sua folha de especificações (*data sheet)* [21].

Tabela 3.1: Especificações do ADALM-PLUTO [6].

	Valor Típico
Fonte de alimentação (USB)	4.5 V – 5.5 V
Interface	USB 2.0
Temperatura de Operação	10°C - 40°C
Transmissor/Recetor	1 / 1
Core	Single ARM Cortex® -A9 @ 667 MHz
Transcetor	RF Agile Transceiver AD9363
FPGA	Xilinx Zynq Z-7010 (28 k células)
Memória *flash*	Micron QSPI Flash MT25QU256ABA 256 MB
Memória RAM	Micron DDR3L MT41K256M16 4 GB
Software	MATLAB (Simulink) / GNU Radio

Tabela 3.2: Principais especificações do transcetor AD9363 [21].

	Valor Típico
Cobertura RF	325 MHz – 3.8 GHz
Transmissor/Recetor	2 / 2
Duplex	*Half* ou *Full*
Largura de Banda (máximo)	20 MHz
Número de bits (ADC/DAC)	12 Bits
Taxa de Amostragem (ADC/DAC)	65.2 kS/s - 61.44 MS/s
Precisão de frequência	± 25 ppm
Proteção RF	Não
Potência de saída máxima (TX)	7 dBm (a 3.5 GHz)
Ruído (TX)	≤ -157 dBm/Hz (*noise floor*)
Potência de entrada máxima (RX)	2.5 dBm
EVM (TX/RX)	-34 dB (2%)
Figura de ruído (RX)	< 3.5 dB
Resolução do oscilador local (LO)	10 Hz

Questões relativamente ao ADALM-PLUTO e ao seu transcetor podem ser colocadas na página de suporte *"EngineerZone"* fornecida pela Analog Devices [24].

3.3 Diagrama de blocos

Nesta secção vai ser apresentado o diagrama de blocos simplificado do ADALM-PLUTO *Active Learning Module* (Figura 3.2) e o diagrama de blocos, mais detalhado, do transcetor AD9363, que representa toda a componente RF do sistema.

A Figura 3.2 é uma ilustração mais geral do que foi abordado nas duas secções anteriores, em que foram especificadas as funções de alguns blocos e os componentes que os constituem.

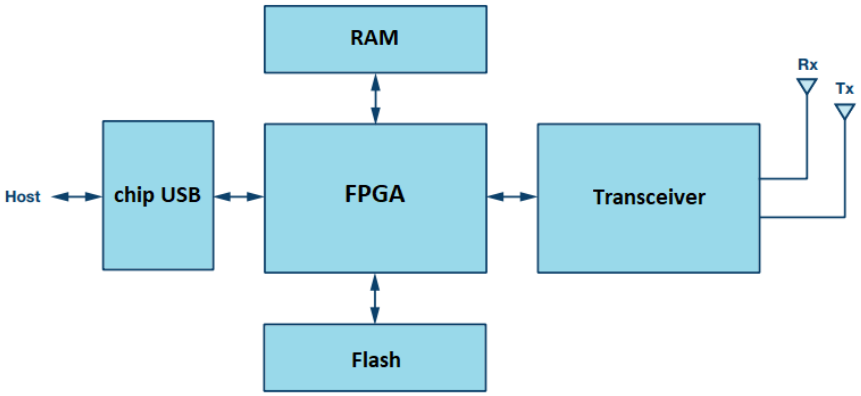


Figura 3.2: Diagrama de blocos simplificado do ADALM-PLUTO (Adaptado de[6]*).*

Seguidamente, na Figura 3.3, é apresentado o diagrama de blocos funcional do transcetor AD9363, da Analog Devices.

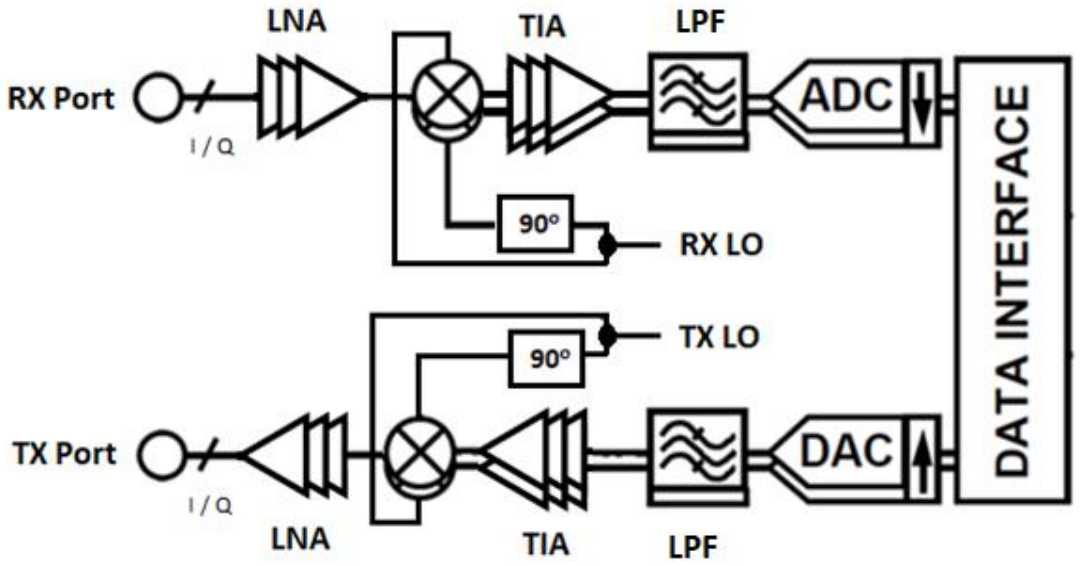


Figura 3.3: Diagrama de blocos funcional do AD9363 (Adaptado de [21]*).*

O diagrama de blocos original tem dois canais independentes, cada um com três entradas e duas saídas, o que permite que este transcetor possa ser utilizado em sistemas MIMO. Por uma questão de visualização, optou-se por representar apenas um canal, com uma entrada e uma saída.

A receção é realizada através de um sistema de conversão direta, em que o sinal entra num amplificador de baixo ruído (LNA) para ser amplificado sem que seja introduzido muito ruído. Este sinal é convertido nos sinais I e Q, em banda base, através do batimento (no misturador) com os sinais RF (em quadratura), produzidos pelo oscilador local. O bloco do amplificador de transimpedância (TIA) e do filtro passa-baixo (LPF) proporcionam um sinal filtrado e com os níveis de amplitude apropriados à entrada das ADCs, onde os sinais são convertidos para o domínio digital. Após a decimação (*Digital Down Sampling)*, que reduz a taxa de amostragem, os sinais são enviados para o processador através de uma interface USB.

Tal como na receção, a transmissão também é feita através de um sistema de conversão direta. Os sinais I e Q enviados pelo processador, em banda base, após realizada a interpolação (Digital Up Sampling), são convertidos para o domínio analógico através das DACs. Estes sinais, após passarem pela filtragem analógica (LPF) e amplificados pelo TIA, modulam as portadoras em fase e em quadratura (produzidas pelo oscilador local (LO)), através do misturador. Os sinais modulados I e Q após serem somados, são amplificados pelo LNA.

3.4 Recursos MATLAB

Nesta secção são apresentados todos os requisitos necessários para a utilização do ADALM-PLUTO com o MATLAB e algumas das características dessa interface, nomeadamente os parâmetros passíveis de serem alterados, a versão de MATLAB, entre outras.

Apesar da versão MATLAB R2017b já suportar a utilização do ADALM-PLUTO, ao longo da realização da dissertação, chegou-se à conclusão de que a versão mais robusta é o MATLAB R2018a. Isto deve-se ao facto de a versão anterior ser a primeira que suportava este dispositivo, sendo que algumas funcionalidades ainda não estavam completamente desenvolvidas.

Para ser possível utilizar o MATLAB como *software* de processamento de sinal do sistema de Rádio Definido por *Software*, composto pelo ADALM-PLUTO, é necessário instalar algumas *toolboxes*. São elas:

- *Communications System Toolbox Support Package for Analog Devices ADALM-Pluto Radio*, fornece todas os *drivers* necessários para a interface entre o *software* e o dispositivo SDR e

permite a utilização do ADALM-PLUTO como um dispositivo SDR, através das funções do MATLAB e dos blocos do Simulink. É a *toolbox* principal [25];

- *Communications System Toolbox*, fornece algoritmos e aplicações para a análise, projeto e simulação de sistemas de comunicação no MATLAB e no Simulink. Oferece ferramentas tais como o cálculo de BER, diagrama de olho ou constelação do sinal;

- *DSP System Toolbox*, fornece algoritmos e aplicações para projetar, simular e analisar sistemas de processamento de sinal no MATLAB e no Simulink. Permite projetar filtros FIR, IIR e adaptativos, entre outros;

- *Signal Processing Toolbox*, oferece funções e aplicativos para analisar, pré-processar e extrair recursos de sinais amostrados uniformemente e não uniformemente. Entre outras funcionalidades, esta *toolbox* permite medições de SNR e de distorção.

No Apêndice A encontra-se o manual de instalação do sistema MATLAB – ADALM-PLUTO, onde são explicados os passos necessários para a correta instalação e utilização deste módulo de ensino.

Para além das *toolboxes*, a MathWorks disponibiliza uma série de exemplos que permitem uma rápida familiarização com as várias funcionalidades do sistema, tais como, por exemplo, um transmissor e um recetor QPSK [26].

Com o objetivo de tornar possível a simulação e o desenvolvimento de aplicações de Rádio Definido por *Software*, a Mathworks desenvolveu um conjunto de funções e objetos, dos quais se destacam [27]:

- *findPlutoRadio*, que nos dá informação (número de série e número de identificação do rádio) sobre todos os ADALM-PLUTO que se encontram conectados ao computador;

- *configurePlutoRadio*, apesar do ADALM-PLUTO utilizar o transcetor AD9363, é possível, com esta função, configurá-lo para operar numa gama de frequências mais ampla e com uma taxa de amostragem maior (AD9364). Esta função permite configurar o *firmware* do rádio ADALM-PLUTO [28];

- *comm.SDRRxPluto*, é um objeto do sistema [29] que permite a receção de dados de um rádio ADALM-PLUTO. Este objeto tem vários parâmetros que precisam de ser configurados

de acordo com as preferências do utilizador. Na Figura 3.4, está representado um exemplo de configuração do objeto em questão.

```
radioRx = comm.SDRRxPluto(...
    'RadioID',              'usb:0', ...
    'CenterFrequency',      0.7e9, ...
    'GainSource',           'Manual', ...
    'Gain',                 25, ...
    'BasebandSampleRate',   2000000, ...
    'OutputDataType',       'double', ...
    'SamplesPerFrame',      2^17, ...
    'ChannelMapping',       1, ...
    'FrequencyCorrection',  0);
```

Figura 3.4: Excerto de código de um exemplo de configuração *do objeto do recetor.*

A Tabela 3.3 apresenta a gama de valores dos vários parâmetros, de acordo com [30].

Tabela 3.3: Representação da gama de valores dos parâmetros do comm.SDRRxPluto.

Parâmetro	Gama de valores (valor por defeito)
Frequência Central	70 MHz – 6 GHz (2.4 GHz)
Fonte de Ganho	"*AGC Slow Attack*" / "*AGC Fast Attack*" / "*Manual*"
Ganho	-4 dB - 71 dB (10 dB)
Mapeamento de canal	1
Taxa de amostragem	65.105 KHz – 61.44 MHz (1 MHz)
Tipo de dados de saída	*double / single / int16* (*int16*)
Amostras por trama	2 – 16 777 216 amostras/trama (20 000)
Correção de frequência do oscilador	-200 ppm – 200 ppm (0)

A gama de valores aceitável para o ganho varia consoante a frequência central. Se forem incompatíveis, é retornado um erro. A fonte desse ganho depende se o sinal contém níveis de potência que mudam lentamente (*AGC Slow Attack*) ou rapidamente (*AGC Fast Attack*), podendo ser alterado manualmente (*Manual*). Relativamente ao número de amostras por trama (pacote de dados), é recomendado que se utilize pelo menos 3660 amostras/trama

para que o desempenho do sistema não seja afetado. Finalmente, por vezes, é necessário realizar uma correção de frequência devido ao desvio (*offset*) do oscilador local. Quando é utilizado o valor por defeito, a calibração de fábrica é acionada.

- *comm.SDRTxPluto*, é um objeto do sistema que permite a receção de dados de um rádio ADALM-PLUTO. Tal como o objeto do recetor, tem vários parâmetros que precisam de ser configurados de acordo com o objetivo do utilizador. Na Figura 3.5, está representado um exemplo de uma configuração do objeto do transmissor.

```
radioTx = comm.SDRTxPluto(...
    'RadioID',              'usb:1', ...
    'CenterFrequency',      0.7e9, ...
    'Gain',                 -10, ...
    'BasebandSampleRate',   2000000, ...
    'ChannelMapping',       1, ...
    'FrequencyCorrection',  -2);
```

Figura 3.5: Excerto de código de um exemplo de configuração *do objeto do transmissor.*

A Tabela 3.4 apresenta a gama de valores dos vários parâmetros, de acordo com [31].

Tabela 3.4: Representação da gama de valores dos parâmetros do comm.SDRTxPluto.

Parâmetro	**Gama de valores (valor por defeito)**
Frequência Central	70 MHz – 6 GHz (2.4 GHz)
Ganho	-89.75 dB - 0 dB (-10 dB)
Mapeamento de canal	1
Taxa de amostragem	65.105 KHz – 61.44 MHz (1 MHz)
Tipo de dados de saída	*double* / *single* / *int16* (*int16*)
Correção de frequência	-200 ppm – 200 ppm (0)

No transmissor toda a gama de ganho pode ser alterada, pois a frequência já não limita a sua excursão. Tal como no objeto do recetor, em alguns casos é necessário fazer uma correção de frequência devido ao desvio do oscilador local. Quando é utilizado o valor por defeito, a calibração de fábrica é acionada. O objeto do transmissor é mais simples, na medida em que tem de se configurar um menor número de parâmetros.

4 Medição da Potência de Saída

Como o ADALM-PLUTO não tem proteção RF, verificou-se que seria necessário efetuar o estudo da variação da potência de saída do rádio para evitar danos no módulo. Neste capítulo, estão presentes o procedimento utilizado e os resultados obtidos na realização deste teste, facultando ao utilizador dados cruciais para a correta utilização deste módulo de ensino. Esta experiência foi realizada no laboratório de Rádio Frequência do Instituto de Telecomunicações de Aveiro.

4.1 Procedimento e configuração utilizada

A configuração utilizada (Figura 4.1) para a realização destas medições era composta pelos seguintes equipamentos:

- Rádio: ADALM-PLUTO;
- Medidor de Potência: HP 437B *Power Meter* [32];
- Sonda: Keysight 8487D *Power Sensor* [33];
- Atenuadores de 20 e 30 dB;
- Computador;
- Chave de torque.

Figura 4.1: Configuração *utilizada para a medição da potência de saída do ADALM-PLUTO.*

Antes de efetuar qualquer medição, foi necessário calibrar o medidor de potência com uma referência, para tornar as medições mais precisas. Para medir a potência de saída, isto é, a potência medida na porta TX do ADALM-PLUTO, foi criado um *script* MATLAB (Apêndice B) em que o sinal a ser transmitido é uma sinusóide. Mediante os parâmetros do objeto do emissor, *comm.SDRTxPluto*, foi feita uma variação do ganho do emissor, *radioTx.Gain*, para obter o gráfico da potência de saída em função do ganho do transmissor. Este teste foi repetido várias vezes, para valores de frequência diferentes.

Na Tabela 4.1 estão os valores assumidos para cada um dos parâmetros do objeto do transmissor do ADALM-PLUTO.

Tabela 4.1: Especificação dos parâmetros do comm.SDRTxPluto, na medição da potência.

Parâmetro	**Valor**
Frequência Central	500 MHz, 1 GHz, 1.85 GHz, 4 GHz, 6 GHz
Ganho	0 dB – 89.7 dB
Mapeamento de canal	1
Taxa de amostragem	200 KHz
Correção de frequência	-2 ppm

4.2 Resultados

Para os casos em que os valores de ganho do transmissor são mais elevados, foi necessário introduzir atenuação no sistema, para que não fosse ultrapassada a potência máxima da sonda (-20 dBm). A atenuação é introduzida entre a porta TX do ADALM-PLUTO e a sonda de medição de potência.

A relação obtida entre a potência de saída e o ganho do transmissor do ADALM-PLUTO, para vários valores de frequência, está representada na Figura 4.2.

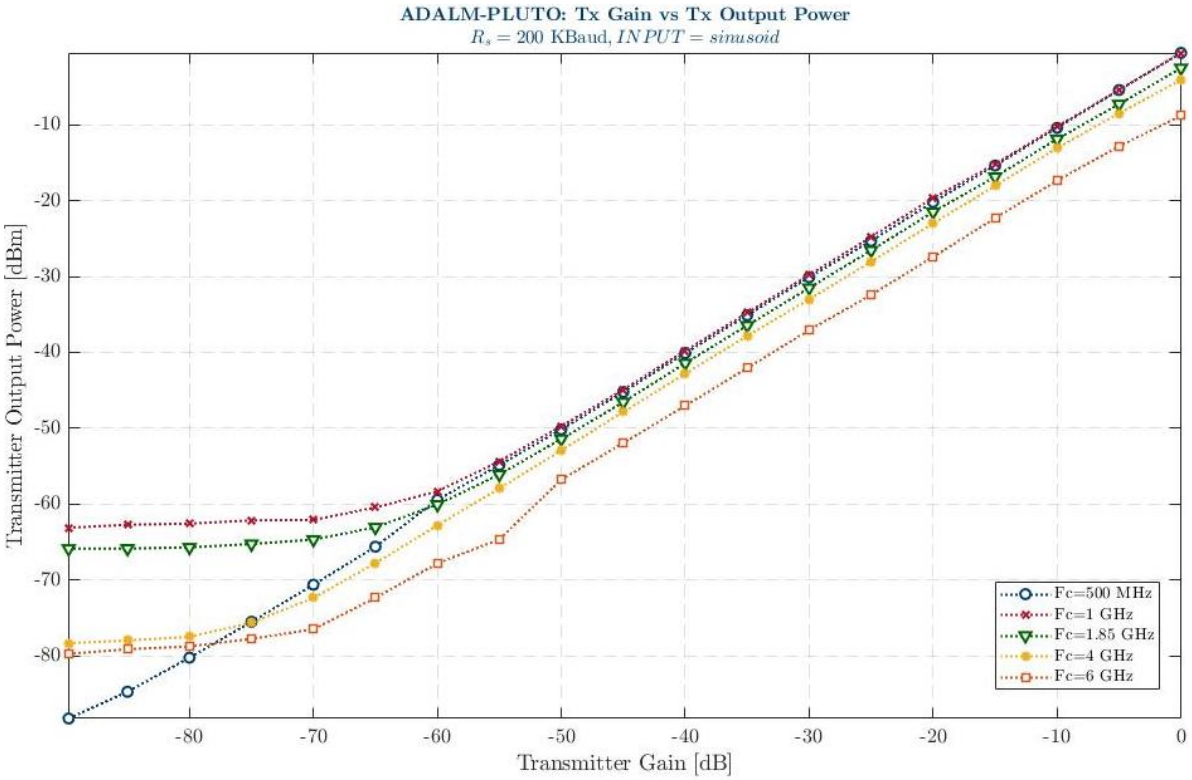


Figura 4.2: Relação entre a potência de saída e o ganho do transmissor, para diferentes valores de frequência central.

Adicionalmente, foi realizado o mesmo teste para o USRP B210. Os resultados obtidos encontram-se no Apêndice C e estão de acordo com os resultados de [34].

4.3 Análise de resultados

Através da figura anterior é possível verificar que o ADALM-PLUTO apresenta uma proporcionalidade direta da potência de saída em função do ganho, para ganhos do transmissor iguais ou superiores a -50 dB. Esta será a zona de funcionamento pretendida e a que vai ser utilizada ao longo da dissertação.

Outra conclusão que se pode retirar, na região de proporcionalidade, é que a potência de saída aumenta quando o ganho aumenta e diminui, ligeiramente, quando a frequência central aumenta.

Caso se pretenda ligar um ADALM-PLUTO a outro, através de um cabo, é necessário ter cuidado com o balanço de potências. A seguinte relação tem de ser respeitada para evitar danos no recetor.

$$P_{TX\ máx} - atenuação \leq P_{RX\ máx} \quad (1)$$

Como a potência máxima do transmissor ($P_{TX\ máx}$) obtida é – 0.5 dBm e a potência máxima que o recetor suporta ($P_{RX\ máx}$) é 2.5 dBm [21], a relação anterior é sempre respeitada. Isto significa que não é obrigatória a utilização de atenuadores entre o emissor e o recetor. Ainda assim, achou-se que seria mais seguro introduzir um atenuador de 20 dB no sistema, de modo a criar uma margem de segurança. Esta margem assume um papel fundamental na medida em que o ADALM-PLUTO não tem qualquer proteção face a potências superiores à especificada.

5 Implementação do Sistema e Resultados

Este capítulo consiste na explicação detalhada do sistema que se pretende implementar no MATLAB e na apresentação e discussão dos resultados da dissertação. O sistema a implementar não é mais do que um sistema de transmissão-receção, em que podem ser utilizados vários formatos de modulação, desde o QPSK até ao 512QAM. Inicialmente, é apresentada a configuração utilizada para implementar esse sistema. Seguidamente, é fornecida a estrutura do código MATLAB utilizado e é dada uma explicação das funções mais importantes que o constituem. Com o objetivo de melhorar o sistema implementado, alguns parâmetros foram otimizados. Essa otimização é outra das secções deste capítulo. Para finalizar, são apresentados os resultados do desempenho do sistema e sua análise. Os dados experimentais obtidos neste capítulo estão disponíveis num repositório de dados online [35].

5.1 Configuração utilizada

O sistema de comunicações testado (Figura 5.1) era composto pelos seguintes equipamentos:

- 2 x ADALM-PLUTO;
- Cabo SMA;
- Atenuador de 20 dB;
- 2 x Cabo USB;
- Computador;
- Chave de torque.

Figura 5.1: Configuração *utilizada para a implementação do sistema de transmissão-receção usando SDR.*

Na Figura 5.1, está representada a configuração utilizada para a implementação do sistema verificando-se que os ADALM-PLUTO estão ligados por um cabo SMA + atenuador de 20 dB. Nesta implementação, é de realçar que são necessárias duas instâncias do MATLAB R2018a, uma para o transmissor e outra para o recetor.

5.2 Estrutura do código MATLAB

Nesta secção é abordado o código MATLAB utilizado no sistema de transmissão-receção e é dada uma explicação das funções que constituem o núcleo do código. Como o objetivo primordial da dissertação era a criação de um novo módulo de ensino laboratorial de apoio às aulas de Sistemas de Informação e de Sistemas de Comunicação, para poupar tempo e recursos, optou-se por reutilizar o código existente para Comunicações Óticas [36]. O código do transmissor, do recetor e do processamento de sinal encontra-se no Apêndice B. Nesta secção, vai ser dada mais ênfase ao código referente ao processamento de sinal (DSP) do sinal a transmitir e ao código do

processamento de sinal do sinal recebido. Do transmissor e do recetor, importa saber que o ruído aditivo gaussiano branco (AWGN) é introduzido antes da transmissão do sinal, para cada valor de SNR pretendido. Dada a impossibilidade de introduzir o ruído no caminho entre o transmissor e o recetor, optou-se por introduzi-lo antes da transmissão do sinal, pois representa uma melhor aproximação a um sistema de comunicação real. Se o ruído fosse introduzido na receção, teríamos um sistema completamente ideal, em que o sinal recebido estava limpo, ou seja, sem qualquer ruído, não sendo possível avaliar a influência que alguns componentes de *hardware* têm no sinal. Esse ruído é gerado e adicionado ao sinal pela função *setSNR.m*, presente em *TX_main_function_PLUTO_noise.m* (Apêndice B).

5.2.1 Diagrama de blocos do transmissor

Na Figura 5.2, é apresentado o diagrama de blocos da função MATLAB que executa o processamento digital do sinal a transmitir (Apêndice B). A seguir ao diagrama é facultada uma breve explicação de cada um dos blocos que o constituem.

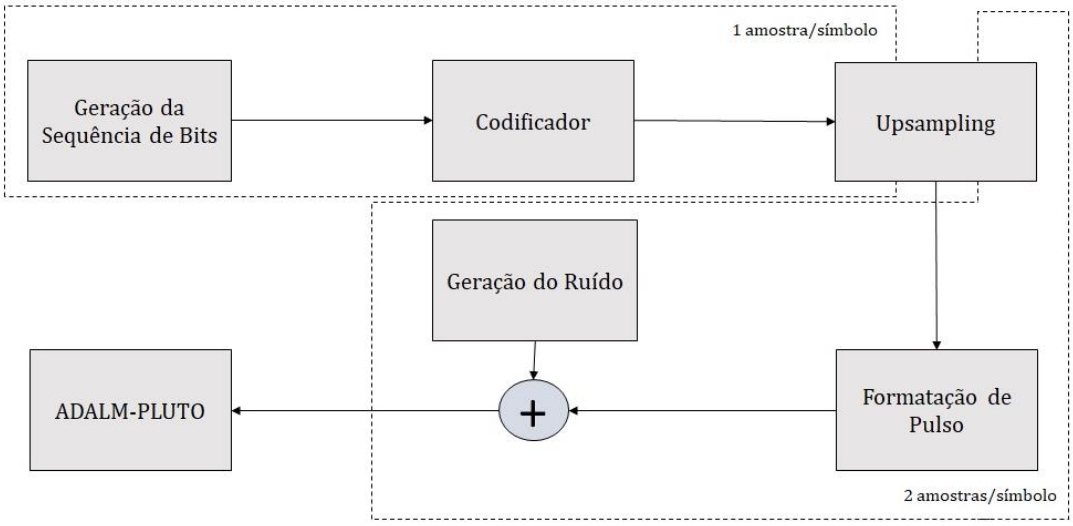


Figura 5.2: Diagrama de blocos da função que executa o DSP no emissor (SC_transmitter.m, que se encontra dentro da TX_main_function_PLUTO_noise.m).

5.2.1.1 Geração da Sequência de Bits

Depois de definidos os parâmetros do sinal e da modulação, tais como taxa de símbolo, fator de decaimento ou formato de modulação, entramos no primeiro subsistema do DSP do transmissor, propriamente dito. Este subsistema consiste no procedimento de geração da sequência pseudoaleatória (PRBS) de bits a transmitir, com um tamanho fixo de 2^{16} bits. É realizada pela função *QAM_PRBSgenerator.m* que se encontra dentro da função *Tx_QAM.m* que, por sua vez, se encontra dentro da *SC_transmitter.m* (Apêndice B).

5.2.1.2 Codificador

Após a geração dos bits, é implementado o codificador QAM. Primeiramente, é feito o mapeamento dos bits em símbolos, seguindo o código de *Gray*. O número de símbolos gerados obedece à Equação 2, em que N_{bits} é o número de bits gerados (2^{16}) e M é o formato de modulação escolhido. O mapeamento é realizado pela função *bit2sym.m*. Depois é realizada a conversão dos símbolos num sinal complexo. Essa conversão é realizada pela função *symbol2signal.m.* Esta função e a *bit2sym.m* encontram-se dentro da função *Tx_QAM.m* que, por sua vez, se encontra dentro da *SC_transmitter.m* (Apêndice B). À saída deste bloco, temos uma constelação em banda base do sinal a transmitir.

$$N_{símbolos} = \frac{N_{bits}}{\log_2 M} \quad (2)$$

5.2.1.3 Upsampling

Depois de implementado o mapeador QAM, é feita uma reamostragem do sinal. Passou-se a ter duas amostras por símbolo em vez de apenas uma amostra por símbolo inicial, ou seja, foi feito um *upsampling* de 2. Esta reamostragem é realizada na função *SC_transmitter.m* (Apêndice B).

5.2.1.4 Formatação de Pulso

Com duas amostras por símbolo, só faltava realizar a formatação do pulso para que o sinal ficasse pronto a ser transmitido. Essa formatação é realizada por um filtro FIR do tipo RRC, com 128 coeficientes complexos. Este filtro é realizado na função *pulseShaper.m* que se encontra dentro da função *SC_transmitter.m* (Apêndice B).

5.2.1.5 Geração de Ruído

Para finalizar o DSP do transmissor, procedeu-se à geração do ruído gaussiano branco aditivo (AWGN). Esse ruído é gerado pela função *setSNR.m, que se encontra inserida na função TX_main_function_PLUTO_noise.m* (Apêndice B). Esta função vai gerar o ruído AWGN necessário para obter o valor de SNR pretendido, que foi passado como argumento da função, pelo utilizador. Por sua vez, esse ruído vai ser adicionado ao sinal que vem do bloco de formatação do pulso. Posto isto, faltava apenas enviar o sinal para o transmissor do ADALM-PLUTO, para que a transmissão ficasse completa.

5.2.2 Diagrama de blocos do recetor

Seguidamente, na Figura 5.3, é apresentado o diagrama de blocos da função MATLAB que executa o processamento digital do sinal recebido (Apêndice B). A seguir ao diagrama é facultada uma breve explicação de cada um dos blocos que o constituem. Por uma questão de simplificação, optou-se por realizar o DSP não em tempo real, mas quando o sistema de transmissão-receção terminou.

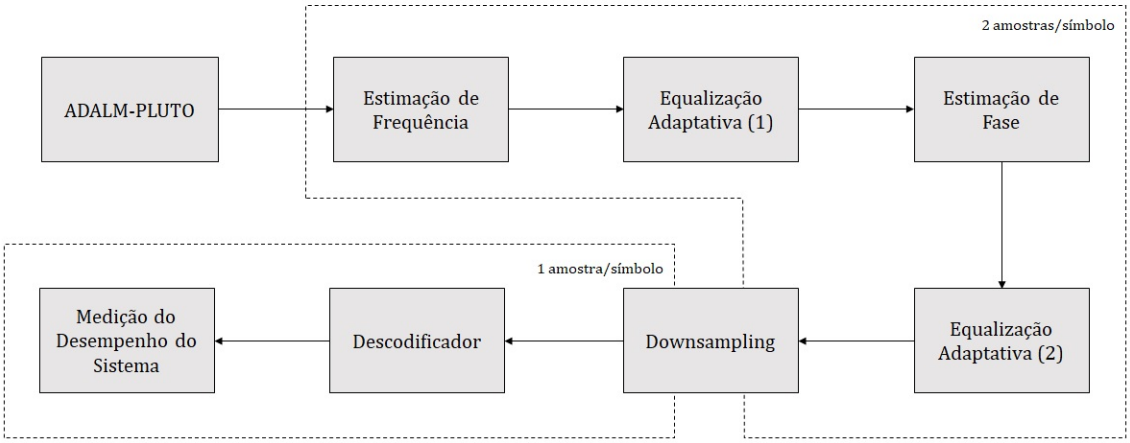


Figura 5.3: Diagrama de blocos da função que executa o DSP (rxDSP_SC_OB2B_LAB.m).

5.2.2.1 Estimação de Frequência

Finalizada a receção do sinal, com duas amostras por símbolo, por parte do recetor do ADALM-PLUTO, fez-se a estimação de frequência através do método espetral [37]. Este método consiste na observação do pico espetral do sinal e permite que o desvio de frequência seja removido. A estimação de frequência é realizada pela função *myFE.m,* que, por sua vez, se encontra dentro da função *SC_freqEstimation.m* (Apêndice B).

5.2.2.2 Equalização Adaptativa (1)

Na maioria dos sistemas de comunicação digital ocorre uma dispersão temporal do sinal transmitido ao longo do canal, o que faz com que dados que foram transmitidos num dado instante interfiram com dados transmitidos noutros. Quando isto acontece, diz-se que há interferência entre símbolos (ISI). Com o objetivo de reduzir a ISI fez-se uma equalização adaptativa [38]. O equalizador utilizado é constituído por filtros FIR com 15 coeficientes e utiliza o algoritmo CMA (1x1). O seu fator de convergência é 10^{-3}. O CMA é um método *data-aided*, ou seja, utiliza os dados como suporte [39]. Esta equalização é realizada pela função *adaptiveEq.m*, que se encontra dentro da função *SC_adaptiveEq.m* (Apêndice B).

5.2.2.3 Estimação de Fase

Após a realização da primeira equalização adaptativa, procedeu-se à estimação de fase que tem como objetivo principal a recuperação de fase do sinal, ou seja, reduzir o erro de fase [40]. O algoritmo utilizado é o *Viterbi & Viterbi* [41], aplicado em blocos de 101 símbolos. A estimação de fase é feita pela função *myCPE.m*, que se encontra dentro da função *SC_phaseEstimation.m* (Apêndice B).

5.2.2.4 Equalização Adaptativa (2)

A seguir à estimação de fase, realizou-se uma segunda equalização adaptativa, cujo objetivo era otimizar a anterior. Ao contrário do primeiro, este equalizador utiliza filtros FIR (*real-valued*) com 51 coeficientes e fator de convergência 10^{-3}. O algoritmo utilizado para a estimação dos coeficientes do filtro é o LMS (2x2) [38]. Neste processo, a equalização das componentes I e Q é feita separadamente. Tal como a primeira equalização, é realizada pela função *adaptiveEq.m*, que se encontra dentro da função *SC_adaptiveEq.m* (Apêndice B).

5.2.2.5 Downsampling

Depois de terminada a equalização do sinal, foi feita uma reamostragem do sinal. Passou-se a ter uma amostra por símbolo em vez das duas amostras por símbolo iniciais, ou seja, foi feito um *downsampling* de 2. Esta reamostragem é realizada pela função *applyResample.m,* que se encontra dentro da função *SC_matchedFilter.m* (Apêndice B).

5.2.2.6 Descodificador

Depois da reamostragem do sinal para uma amostra por símbolo, procedeu-se ao desmapeamento dos símbolos. Esta função sincroniza o sinal transmitido com o sinal recebido e aplica o desmapeamento de símbolos, isto é, pega nos pontos da constelação e transforma-os numa sequência de bits consecutivos. O descodificador é realizado pela função *symDecoder.m*, que se encontra dentro da função *SC_rxDECODER.m* (Apêndice B).

5.2.2.7 Medição do desempenho do Sistema

Para finalizar o DSP, procedeu-se ao cálculo das variáveis que permitem avaliar o desempenho do sistema. Neste caso, efetuou-se o cálculo da BER e da EVM, cujos resultados serão apresentados e discutidos mais à frente, neste capítulo. A medição do desempenho do sistema é realizada pelas funções *BER_eval.m* e *EVM_eval.m*, que se encontram inseridas na função *SC_performance Analyser.m* (Apêndice B).

5.3 Parâmetros Iniciais

Inicialmente, foi necessário assumir determinados valores para os parâmetros do objeto do transmissor (Tabela 5.1), para o objeto do recetor (Tabela 5.2) e, ainda, para os parâmetros do sinal a transmitir (Tabela 5.3). Esta assunção não foi mais do que o ponto de partida da parte prática desta dissertação, sendo que alguns desses parâmetros necessitaram de ser reajustados para a obtenção de melhores resultados, como demostra a Secção 5.5.

Tabela 5.1: Parâmetros iniciais do comm.SDRTxPluto.

Parâmetro	Valor
Frequência Central	1.85 GHz
Ganho	- 10 dB
Mapeamento de canal	1
Taxa de amostragem	200 KHz
Tipo de dados de saída	*double*
Correção de frequência	-2 ppm

Tabela 5.2: Parâmetros iniciais do comm.SDRRxPluto.

Parâmetro	Valor
Frequência Central (f_c)	1.85 GHz
Fonte de Ganho	"*Manual*"
Ganho	10 dB
Mapeamento de canal	1
Taxa de amostragem	200 KHz
Tipo de dados de saída	*double*
Amostras por trama	2^{14}
Correção de frequência	0 ppm

Tabela 5.3: Parâmetros iniciais do sinal a transmitir.

Parâmetro	Valor
Formato de Modulação (M)	16 QAM
Taxa de símbolo (R_s)	50 Kbaud
Número de símbolos	4096
Fator de decaimento da formatação do sinal (β)	1
Bits por símbolo (BpS)	$\log_2(M) = 4$

5.4 Estimação analítica da BER em canais AWGN

Na próxima secção vai ser realizada a otimização de alguns parâmetros do sistema. Essa otimização consiste na comparação dos resultados obtidos, para os diferentes valores de cada parâmetro a otimizar, com uma curva teórica. Nesta secção vão ser apresentadas as expressões que permitem

calcular essa curva teórica, chamada "*AWGN theory*". Todas as expressões seguintes foram adaptadas de [42].

A curva teórica que se pretende obter não é mais do que a representação da dependência da BER com a SNR. A estimação da BER teórica é dada por:

$$BER = \frac{k_2}{\log_2(M)} erfc\left(\sqrt{k_1 SNR}\right) \quad (3)$$

Onde M é o tamanho da constelação do sinal, erfc(.) é a função de erro complementar e k_1 e k_2 são duas constantes dependentes da modulação. Essas constantes são dadas por:

$$k_1 = \begin{cases} \frac{3}{2(M-1)}, & para\ QAM\ quadrado \\ \frac{48}{31M-32}, & para\ QAM\ em\ cruz\ e\ M \geq 32 \\ \frac{1}{3+\sqrt{3}}, & para\ QAM\ em\ cruz\ e\ M = 8 \end{cases} \quad (4)$$

E por:

$$k_2 = \begin{cases} 2\left(1 - 1\sqrt{M}\right), & para\ QAM\ quadrado \\ \frac{G_p N}{2}, & para\ QAM\ em\ cruz \end{cases} \quad (5)$$

Onde G_p é o número médio de bits que diferem entre símbolos adjacentes na constelação e N é o número médio dos vizinhos mais próximos de cada símbolo na constelação dos QAM em cruz. O G_p é dado por:

$$G_p = \begin{cases} 5/4, & para\ M = 8 \\ 7/6, & para\ M = 32 \\ 1 + \frac{1}{\sqrt{2M}} + \frac{1}{3M}, & outros \end{cases} \quad (6)$$

E o N é dado por:

$$N = \begin{cases} 3, & para\ M = 8 \\ 4 - \frac{6}{\sqrt{2M}}, & outros \end{cases} \quad (7)$$

5.5 Otimização dos parâmetros do Sistema

Depois de estabelecido o ponto de partida, foi necessário ajustar alguns parâmetros com o objetivo de otimizar o desempenho do sistema implementado. Parâmetros como o ganho do recetor, o ganho do transmissor, o fator de decaimento da formatação do sinal ou o desvio de frequência, a ser introduzido no transmissor, são alguns dos que necessitaram de ser otimizados, individualmente.

5.5.1 Desvio de Frequência

Através da observação do espetro do sinal (PSD), foi possível verificar que, para o valor por defeito de desvio de frequência dos osciladores locais, 0 ppm, quer para o transmissor, quer para o recetor, o sinal não se encontrava centrado, levando à degradação do desempenho do sistema. Com o intuito de corrigir este desvio entre os osciladores, foi necessário fazer uma correção na frequência, através da introdução de um desvio no objeto do transmissor. Como este equipamento carece de informação acerca destes pormenores de utilização, foi realizado um teste em que se utilizaram os parâmetros especificados na Secção 5.3, modificando somente o parâmetro de correção de frequência do transmissor (*radioTx.FrequencyCorrection*).

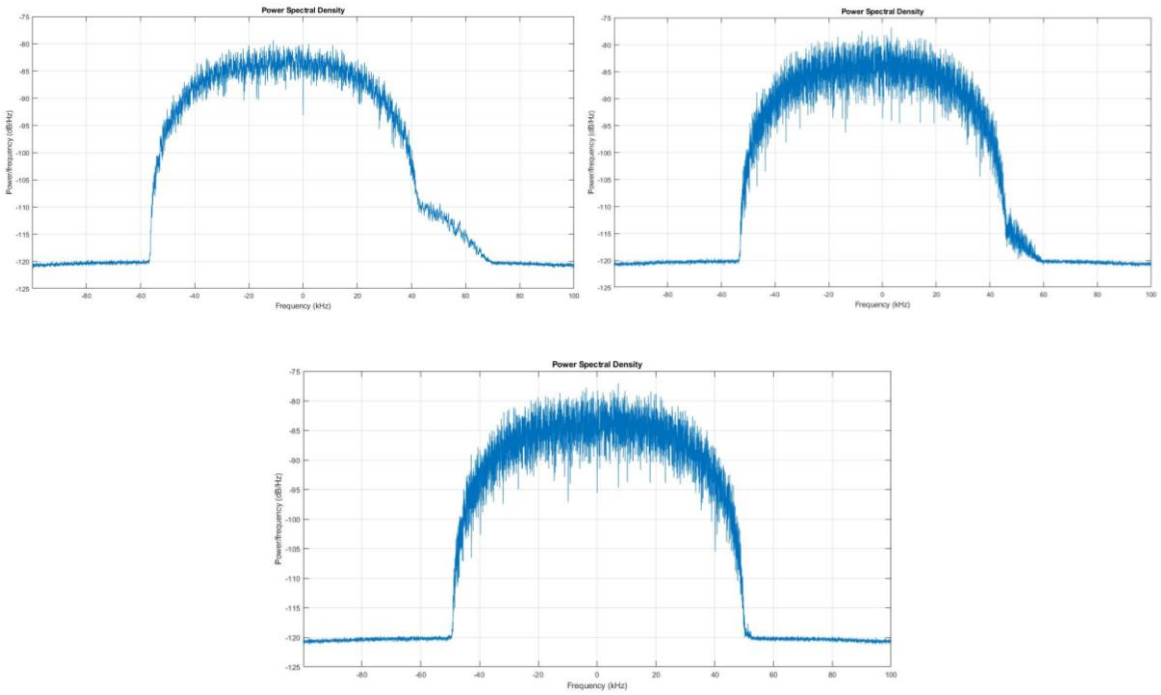


Figura 5.4: Espetro do sinal recebido para diferentes desvios de frequência do transmissor (2 ppm esquerda em cima, 0 ppm direita em cima e -2 ppm em baixo).

Pela Figura 5.4 é possível concluir que o sinal se encontra centrado para um desvio de frequência de -2 ppm. Para os restantes valores testados, 0 e 2 ppm, o sinal encontra-se descentrado e a PSD deixa de ser simétrica. Por estas razões, o valor ótimo é -2 ppm pelo que foi utilizado daqui para a frente.

5.5.2 Número de bits utilizados da ADC

Outro dos parâmetros otimizado foi o número de bits da ADC utilizados efetivamente. O objetivo era utilizar o máximo de bits disponíveis da ADC que, segundo o fabricante, é 12 bits. Para tal, foi criado um *script* MATLAB em que se fixou o valor do ganho do transmissor (G_{TX}) e se variou o valor ganho do recetor (G_{RX}). O número de bits utilizados da ADC foi obtido através do número de valores únicos do sinal recebido (utilizando a instrução *unique()*). O sinal transmitido foi uma sinusóide. Os valores de G_{TX} utilizados obedecem à relação de linearidade obtida no Capítulo 4, que diz que, para o ADALM-PLUTO ter um comportamento linear, é necessário que o G_{TX} seja igual ou superior a -50 dB. Em relação aos valores do G_{RX}, foi feito um varrimento entre -4 e 62 dB. Os restantes parâmetros foram especificados de acordo com a Secção 5.3, com exceção da taxa de símbolo (R_s), que, neste caso, foi de 200 Kbaud.

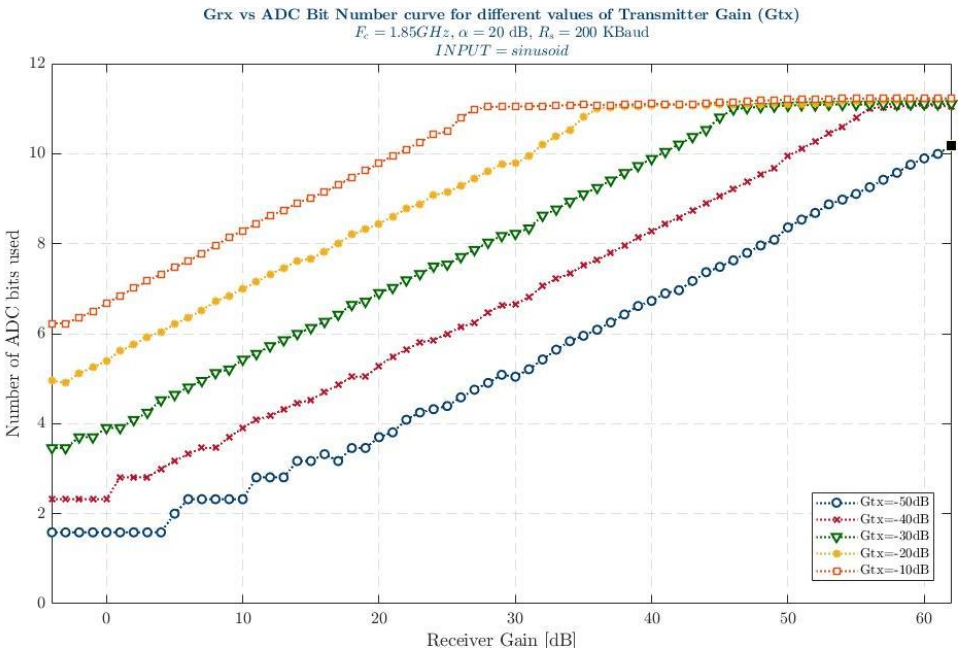


Figura 5.5: Número de bits utilizados da ADC para vários valores de G_{TX} e de G_{RX}.

Da Figura 5.5 pode-se concluir que, quanto maior for o G_{TX} e o G_{RX}, maior será o número de bits utilizados da ADC. Como o valor ótimo de bits da ADC é aquele imediatamente inferior ao seu máximo, para evitar os efeitos não lineares causados pelo *clipping*, pode-se estabelecer que os pares G_{TX}/G_{RX} ótimos são:

- G_{TX} = -10 dB e G_{RX} = 25 dB;
- G_{TX} = -20 dB e G_{RX} = 35 dB;
- G_{TX} = -30 dB e G_{RX} = 45 dB;
- G_{TX} = -40 dB e G_{RX} = 55 dB;

Pela Figura 5.5 foi possível verificar que nunca se consegue utilizar os 12 bits efetivos da ADC. No máximo, consegue-se utilizar 11 bits. A partir dos valores anteriores foi possível chegar à relação ótima entre o G_{TX} e o G_{RX}.

$$G_{TX} + G_{RX} - atenuação = -5\,dB \quad (8)$$

Em jeito de confirmação da relação anterior, foi feito um gráfico idêntico ao anterior, mas em que os ganhos foram substituídos pelas potências, medidas no Capítulo 4.

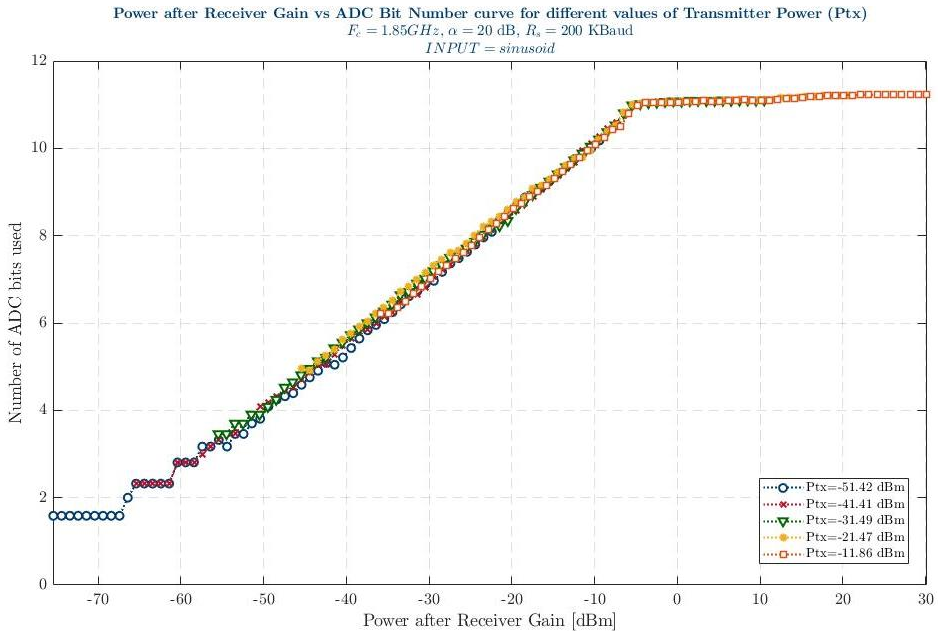


Figura 5.6: Número de bits utilizados da ADC em função da potência (depois do ganho do recetor).

Como era de esperar, as curvas da Figura 5.6 convergiram para uma só curva, confirmando a relação. Se o objetivo do utilizador deste sistema for obter os melhores resultados possíveis, a relação anterior tem de ser respeitada.

5.5.3 Taxa de amostragem

Com o objetivo de tornar o sistema mais realista, achou-se conveniente diminuir o valor do fator de decaimento, que inicialmente era 1. Foram realizadas algumas simulações para valores inferiores a 1 e obtiveram-se resultados diferentes dos esperados. Isto deve-se a vários fatores, como a sincronização do relógio dos ADALM-PLUTOs, a diferença de fase ou o desvio de frequência. Para evitar estes fenómenos, chegou-se à conclusão que seria necessário aumentar a taxa de símbolo (R_s), para que esses efeitos não fossem tão significativos.

Através de algumas simulações, verificou-se que, apesar de se aumentar o valor da R_s, tanto o recetor como o transmissor não têm capacidade para operar a taxas de amostragem superiores a 2.5MHz. Como se pretende utilizar um fator de *upsampling* de 2, o valor máximo de R_s que se pode utilizar é 1 Mbaud.

$$Taxa\ de\ amostragem = Upsampling * Taxa\ de\ símbolo \quad (9)$$

Mesmo com uma taxa de símbolo de 1 Mbaud, o valor efetivo da taxa de amostragem ainda não chega aos 2 MHz, como pretendido. Isto deve-se ao facto da função *step()* não conseguir aumentar a sua velocidade de processamento, ou seja, a velocidade à qual esta função envia ou recebe cada trama não é reconfigurável. Para contornar este problema, aumentou-se o valor do parâmetro amostras por trama inicial de 2^{14} para 2^{17}.

$$Número\ de\ símbolos = \frac{R_s * Amostras\ por\ trama}{Taxa\ de\ amostragem} \quad (10)$$

Nesta subsecção foram otimizados alguns parâmetros importantes do sistema, tais como:

- Taxa de símbolo (R_s) = 1 Mbaud;
- *Upsampling* = 2;
- Taxa de amostragem = 2 MHz (do recetor e do transmissor, equação 9);

- Amostras por trama = 2^{17};
- Número de símbolos (por trama) = 65336 (equação 10).

5.5.4 Ganho do Recetor e do Transmissor

Depois de se ter chegado à relação ótima (Subsecção 5.5.2) entre o ganho do transmissor (G_{TX}) e o ganho do recetor (G_{RX}), foi necessário otimizar o par G_{TX}/G_{RX}. Para tal, foi avaliado o desempenho do sistema para os vários pares de ganhos. O sinal testado tem uma modulação 128QAM, uma frequência central de 1.85 GHz, um fator de decaimento de 0.1 e uma taxa de símbolo de 1 Mbaud.

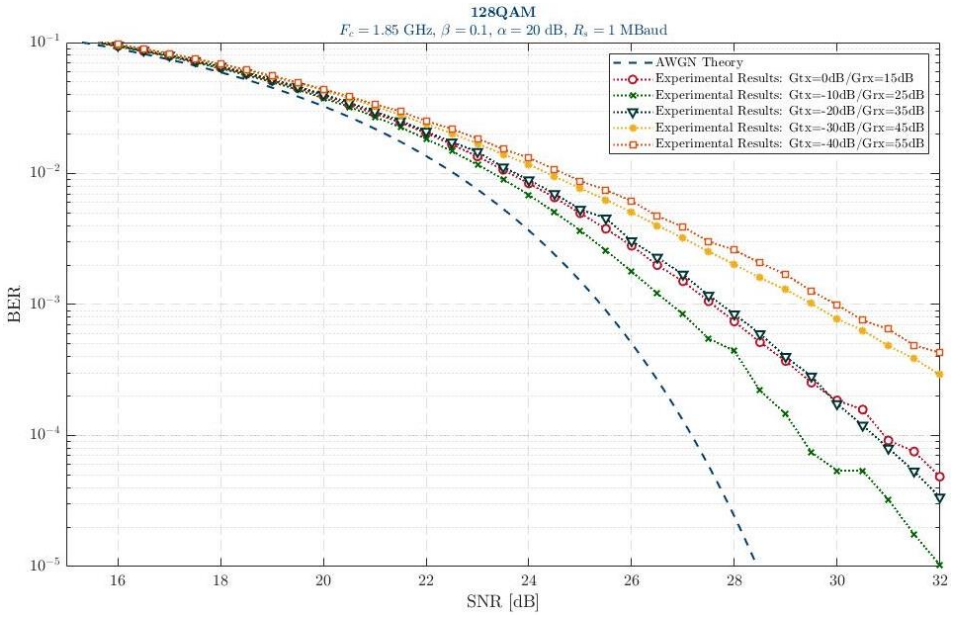


Figura 5.7: Comparação do desempenho do sistema para vários pares G_{TX}/G_{RX}.

Através da Figura 5.7 é possível concluir que o par que apresenta um melhor desempenho, isto é, que a sua curva se aproxima mais da curva teórica, é o par G_{TX} = -10 dB/G_{RX} = 25 dB, sendo, por isso, o par de ganhos utilizado nas restantes simulações.

Como esta otimização foi realizada escolhendo um só formato de modulação, 128QAM, foi necessário verificar se os valores ótimos de G_{TX} e de G_{RX} para esse formato seriam, também, os valores ótimos para os restantes formatos estudados. Para tal, foi realizada uma otimização mais fina do G_{TX} e do G_{RX}, individualmente.

Primeiramente, procedeu-se à otimização do ganho do transmissor. Para isso, fixou-se um valor de BER a testar ($5x10^{-4}$) e um valor de ganho do recetor (25 dB) e fez-se um varrimento do ganho do transmissor, para cada formato de modulação.

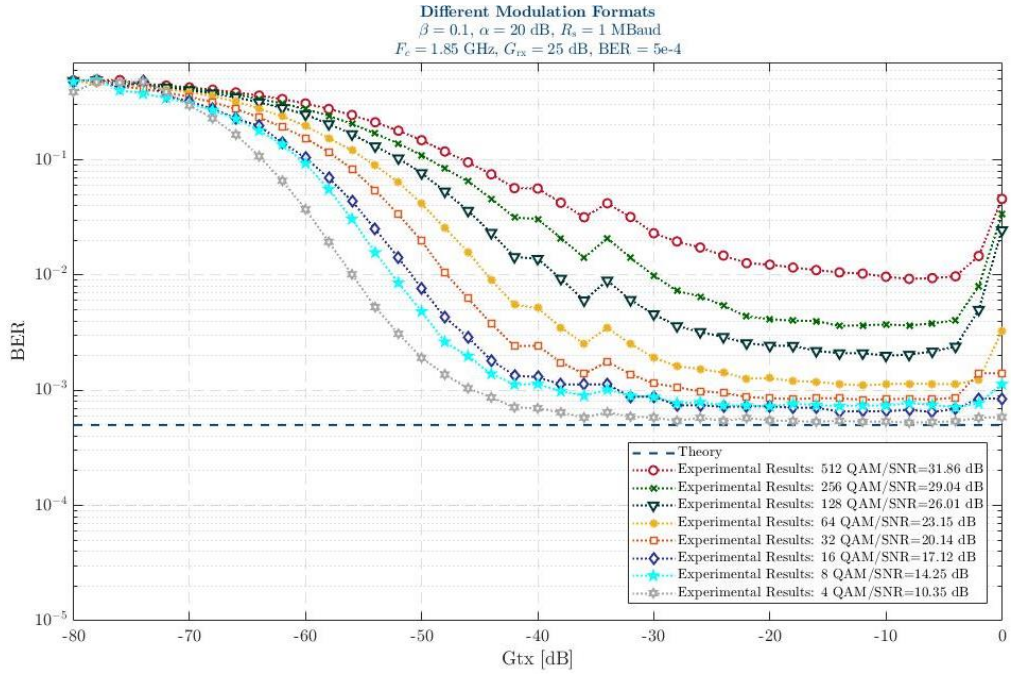


Figura 5.8: Otimização do G_{TX} para vários formatos de modulação.

A partir da Figura 5.8 concluiu-se que o valor escolhido anteriormente para o G_{TX} (-10 dB) podia ser utilizado em todos os formatos de modulação, na medida em que se encontra dentro da gama de G_{TX} em que o desempenho do sistema é máximo.

Depois de otimizado o G_{TX}, efetuou-se uma otimização idêntica do G_{RX}. Para isso, fixou-se um valor de BER a testar ($5x10^{-4}$) e o valor otimizado de ganho do transmissor (-10 dB) e fez-se um varrimento do ganho do recetor, para cada formato de modulação.

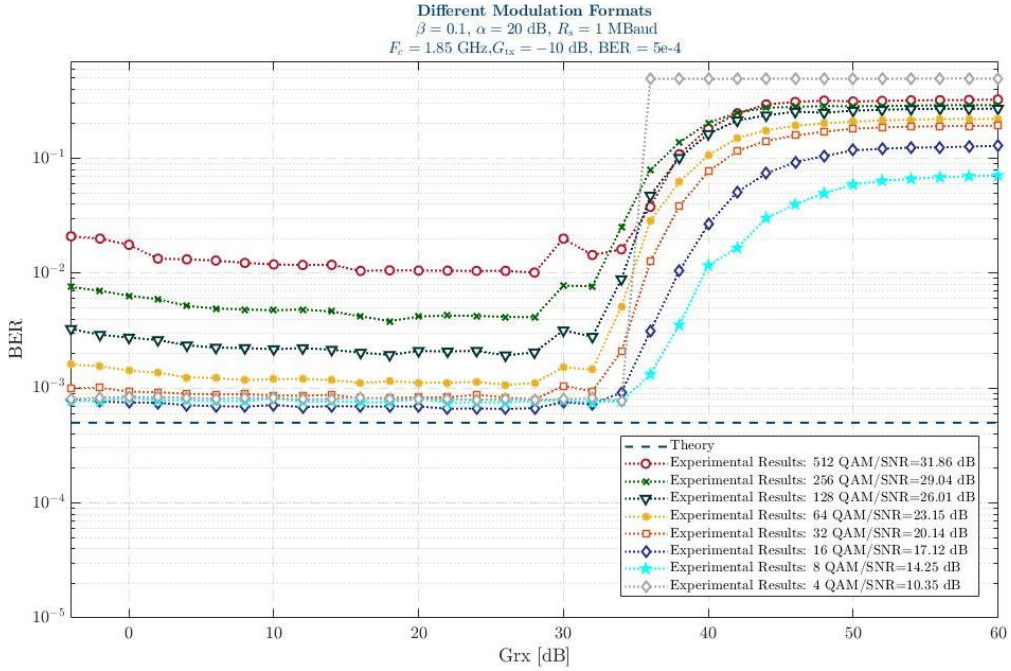


Figura 5.9: Otimização do G_{RX} para vários formatos de modulação.

Através da observação das curvas da Figura 5.9, foi possível verificar que o valor de G_{RX} escolhido anteriormente (25 dB) podia ser utilizado para todos os formatos de modulação, pois encontra-se dentro da gama de G_{RX} em que o desempenho do sistema é máximo.

5.5.5 Fator de decaimento *(roll-off)*

Tal como foi dito anteriormente, na Subsecção 5.5.3, o valor do fator de decaimento precisava de ser reajustado. Para tornar o sistema mais realista e para o otimizar, foi avaliado o seu desempenho para vários valores de fator de decaimento, compreendidos entre 0.1 e 1. O sinal testado tem uma modulação 64QAM, uma frequência central de 1.85 GHz, um G_{TX} de -10 dB, um G_{RX} de 25 dB e uma taxa de símbolo de 1 Mbaud.

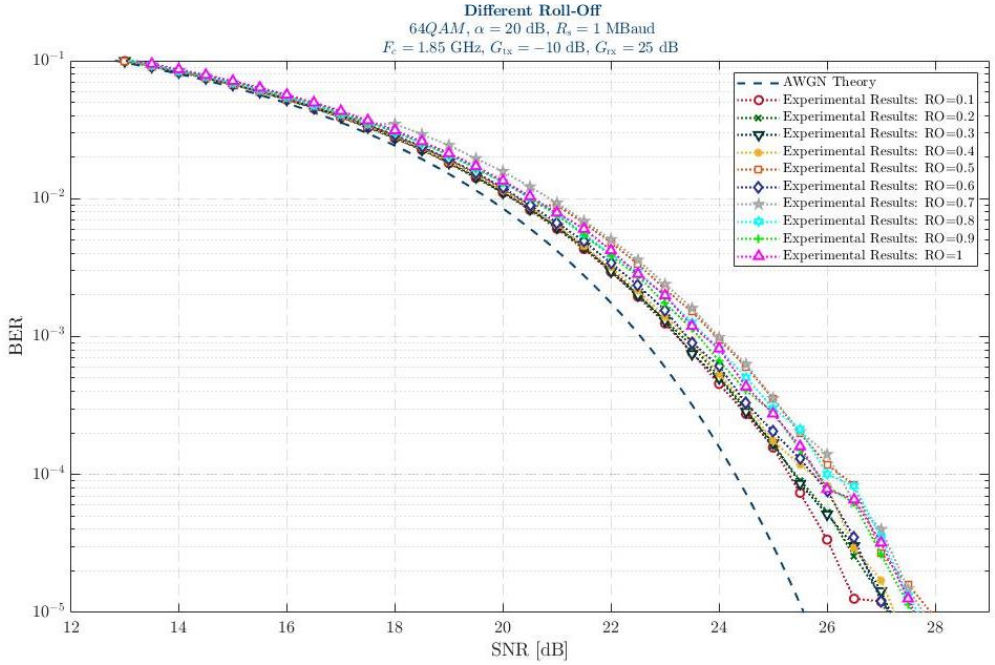


Figura 5.10: Comparação do desempenho do sistema para vários RO.

Devido ao facto das várias curvas se encontrarem muitas vezes sobrepostas e por uma questão de facilidade de consulta, de seguida é apresentada uma tabela que reflete os valores presentes na Figura 5.10.

Tabela 5.4: Comparação do desempenho do sistema para vários RO (valores da Figura 5.10).

Fator de decaimento (RO)	**Penalidade [dB]**		
	BER = 10^{-2}	**BER = 10^{-3}**	**BER = 10^{-4}**
0.1	0.36	0.63	1.01
0.2	0.46	0.77	1.12
0.3	0.46	0.69	1.09
0.4	0.51	0.86	1.43
0.5	0.86	1.40	1.94
0.6	0.58	0.86	1.44
0.7	1.13	1.44	1.91
0.8	0.80	1.22	1.71
0.9	0.78	1.07	1.54
1	0.84	1.18	1.53

Através da Figura 5.10 e da Tabela 5.4, é possível verificar que o valor do fator de decaimento que permite obter um melhor desempenho, isto é, em que se obtém uma menor penalidade em relação à teoria, é o 0.1. Isto deve-se ao facto de que, para valores do fator de decaimento mais baixos, o sinal ocupa uma largura de banda inferior, reduzindo o impacto da filtragem passa-baixo criada pelo transcetor. Daqui para a frente, é este valor utilizado para o RO.

5.6 Resultados

Depois de realizada a otimização dos parâmetros do sistema, foram realizadas as simulações que permitiram obter os resultados desta dissertação. Nesta secção, são apresentados os resultados obtidos para cada formato de modulação testado e os resultados obtidos para frequências diferentes (f_c). Para a obtenção destes resultados, foram utilizados os parâmetros descritos nas Secções 5.3 e 5.5. No Apêndice E, encontram-se mais alguns resultados obtidos durante a realização da dissertação, mas que não têm o grau de importância dos apresentados nesta secção.

5.6.1 Desempenho para os diferentes formatos de modulação

Nesta subsecção são apresentados os gráficos que permitem avaliar o desempenho do sistema para cada formato de modulação testado, desde o QPSK até ao 512QAM. Nesses gráficos são visíveis os valores das penalidades do sistema, isto é, a distância a que curva prática se encontra da curva teórica, para diferentes valores de BER (10^{-2}, 10^{-3} e 10^{-4}). Por uma questão de tempo de processamento, optou-se por terminar a avaliação do desempenho do sistema para um valor mínimo de BER=10^{-5}. Para obter resultados para BERs mais baixas, era necessário um número maior de símbolos comparados, que resultaria num aumento do tempo de simulação do sistema. São, ainda, apresentadas as figuras com as constelações dos sinais recebidos (depois do DSP), sendo que alguns formatos têm a forma de um quadrado (QPSK, 16QAM, 64QAM e 256QAM) e outros a forma de uma cruz (8QAM, 32QAM, 128QAM e 512QAM).

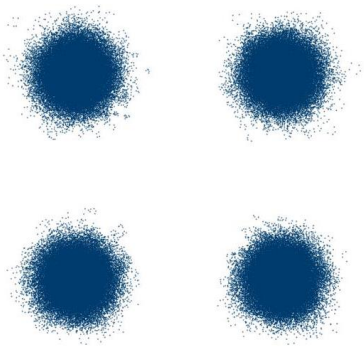


Figura 5.11: Constelação do sinal QPSK, para uma SNR de 16 dB.

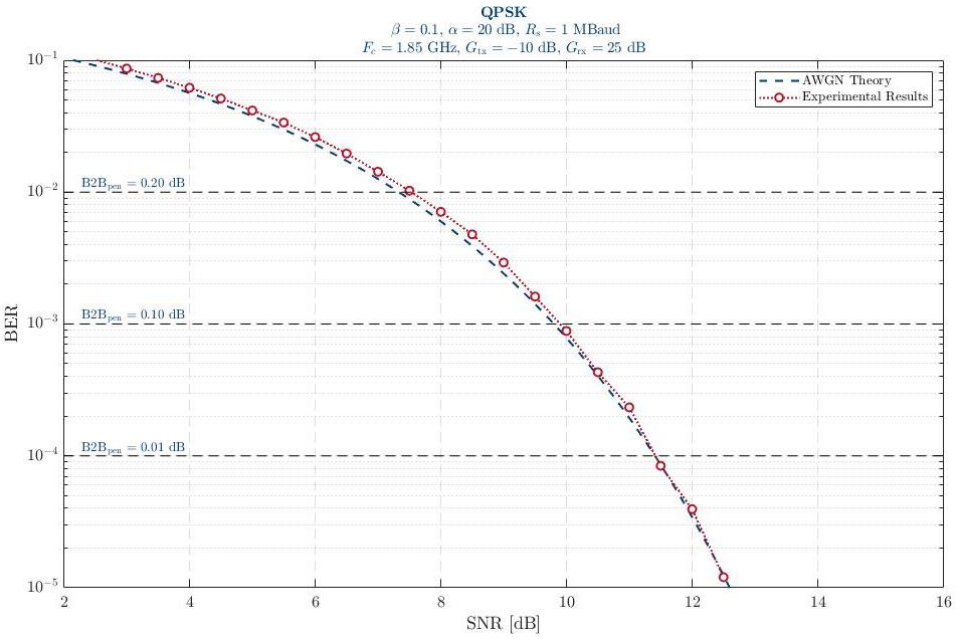


Figura 5.12: Desempenho do sistema com modulação QPSK.

A análise do desempenho do sistema implementado iniciou-se com o formato de modulação mais simples, o QPSK. A constelação deste tipo de modulação, Figura 5.11, é composta por 4 símbolos diferentes, codificados (código de *Gray*) com 2 bits cada. As constelações de todos os formatos de modulação foram obtidas para o máximo de SNR testada, para que fosse mais fácil a visualização dos diferentes símbolos que as constituem.

Relativamente à Figura 5.12, que representa a evolução da BER consoante a SNR do sinal, verificamos que, à medida que se aumenta a SNR, a BER vai diminuindo. Isto significa que quanto maior for a SNR melhor vai ser o desempenho do sistema, pois quanto maior for a SNR maior vai

ser a diferença entre a potência do sinal e a do ruído, ou seja, o ruído vai ter uma menor influência. Da mesma figura, verifica-se que a curva prática se encontra quase sobreposta à curva teórica (*AWGN theory*), devido ao facto dos símbolos do QPSK se encontrarem bastante afastados uns dos outros. Isto acontece porque, para formatos de modulação com baixo PAPR (QPSK, 8QAM e 16QAM), o BER *floor* (BER mínima para o sistema sem ruído, ou seja, o limite mínimo da probabilidade de erro) é muito baixo ou mesmo nulo. Pela Figura 5.12, apesar das diferenças serem muito pequenas, pode-se verificar a tendência de a penalidade diminuir à medida que a BER vai diminuindo. Isto resulta do facto de, para formatos de modulação com baixo M (QPSK, 8QAM e 16QAM), o ruído adicionado ser o fator dominante na degradação do desempenho do DSP. Para BERs mais altas (10^{-2} e 10^{-3}), a SNR é bastante baixa, o que quer dizer que há muito ruído adicionado. Este ruído é nocivo para o DSP, que tem de recuperar o sinal transmitido no meio de uma grande quantidade de ruído, provocando um pouco de penalidade adicional. Já para BERs mais baixas (10^{-4}), o ruído adicionado é bastante baixo e não prejudica tanto o DSP e, ao mesmo tempo, ainda estamos longe do BER *floor* do sistema.

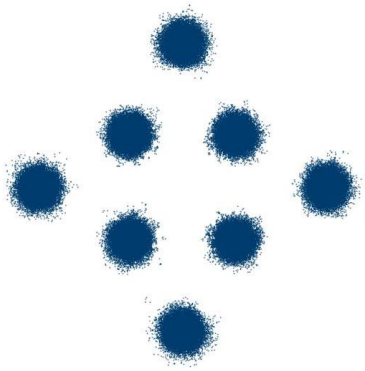


Figura 5.13: Constelação do sinal 8QAM, para uma SNR=20 dB.

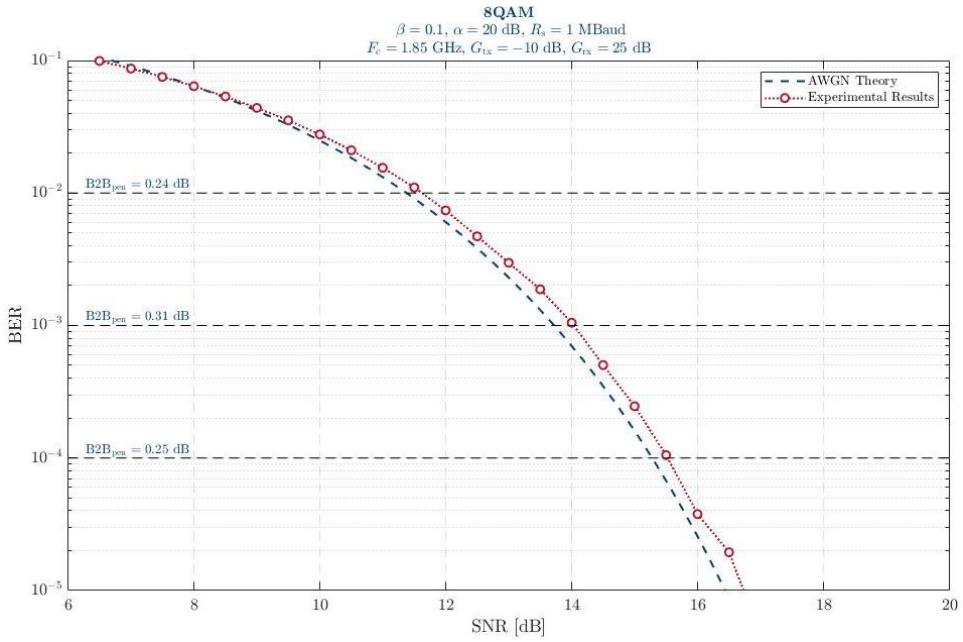


Figura 5.14: Desempenho do sistema com modulação 8QAM.

Completada a análise do desempenho do sistema QPSK, seguiu-se a análise do formato de modulação seguinte, o 8QAM. Na Figura 5.13 está representada a constelação do 8QAM, composta por 8 símbolos diferentes. Ao contrário do QPSK, em que era utilizada uma constelação quadrangular (*square* QAM), neste tipo de modulação foi utilizada uma constelação em cruz (*cross* QAM). Segundo [43], este tipo de constelação é utilizada quando o número de bits por símbolo é ímpar (3 bits por símbolo, neste caso) e pode reduzir, em média, a SNR em 1 dB. Neste tipo de constelações, a codificação dos símbolos é realizada pela aproximação de *Smith* do código de *Gray* e não pelo código de *Gray* exato [43]. A utilização desta aproximação é a causa de a curva prática se encontrar abaixo da curva teórica, para SNRs baixas.

Na Figura 5.14 pode-se verificar que, tal como no QPSK, à medida que a SNR aumenta a BER diminui e, consequentemente, o desempenho do sistema melhora. Relativamente à evolução da penalidade do sistema com a diminuição da BER, espera-se que o 8QAM tenha um comportamento parecido com o do QPSK, pelas razões já referidas. No entanto, diferenças de penalidade inferiores a 0.1 dB são incluídas no erro experimental intrínseco ao sistema. Comparativamente à Figura 5.12, verifica-se que, apesar de residualmente, as penalidades aumentaram. Ainda assim, os resultados obtidos são bastante satisfatórios na medida em que a penalidade relativamente à teórica é bastante reduzida. Isto deve-se ao facto do ruído AWGN ser dominante e à baixa distorção introduzida pelo *hardware*.

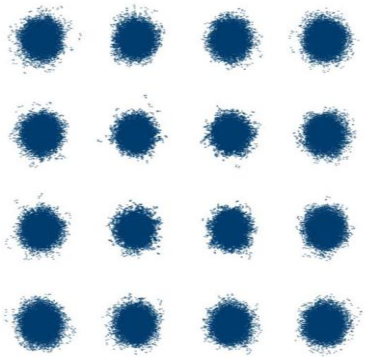


Figura 5.15: Constelação do sinal 16QAM, para uma SNR=23 dB.

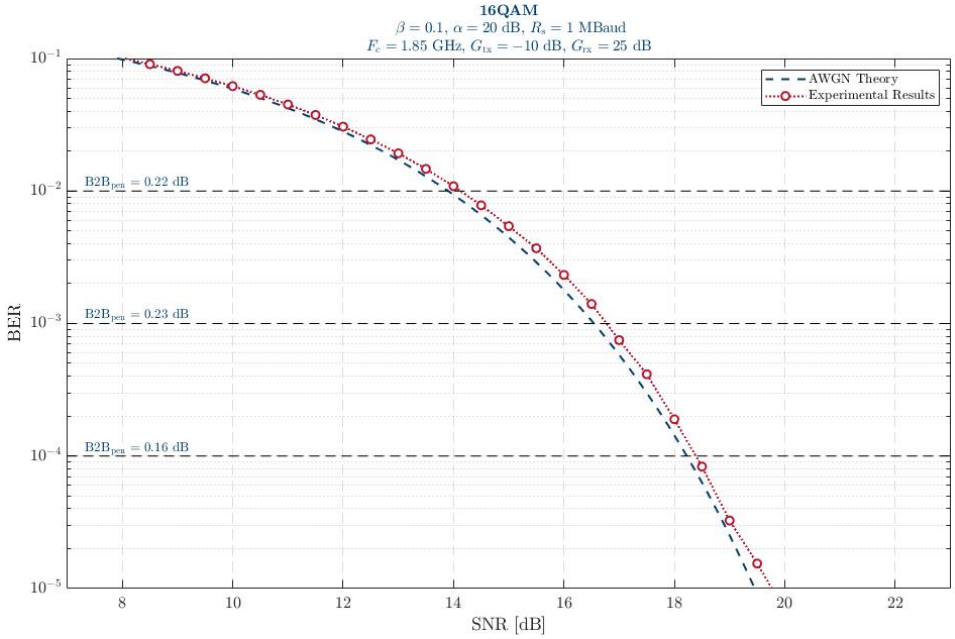


Figura 5.16: Desempenho do sistema com modulação 16QAM.

Continuando a aumentar a complexidade do formato de modulação, chegou-se a um dos formatos com maior relevo no ensino de Sistemas de Comunicação nos dias de hoje, o 16QAM. Na Figura 5.15, está representada a sua constelação composta por 16 símbolos diferentes, codificados (código de *Gray*) com 4 bits cada. Tal como o QPSK, a sua constelação tem o formato de um quadrado.

Tal como nos formatos de modulação estudados anteriormente, através da Figura 5.16, pode-se verificar que quanto maior for a SNR melhor vai ser o desempenho do sistema, ou seja, menor vai ser a BER. Como já foi referido anteriormente, o 16QAM é um formato de modulação com um PAPR

baixo, em que o ruído AWGN é dominante, devido à baixa distorção introduzida pelo *hardware*. Estas características fazem com que o desempenho seja muito próximo do teórico, ou seja, as penalidades são reduzidas. Relativamente à evolução da penalidade do sistema com a diminuição da BER, espera-se que o 16QAM tenha um comportamento idêntico ao do QPSK, pelas razões já referidas. No entanto, devido ao erro experimental intrínseco ao sistema, não é correto avaliar minuciosamente estas diferenças entre os valores de penalidade, pois são inferiores a 0.1 dB. Os resultados obtidos são bastante satisfatórios na medida em que a penalidade, relativamente à teoria, é bastante reduzida (inferior a 3 dB).

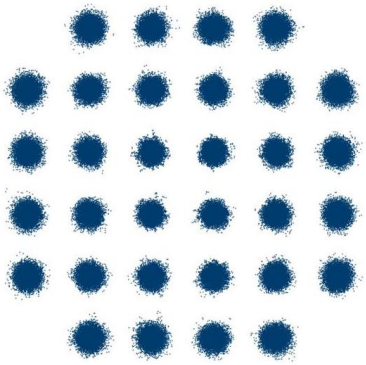


Figura 5.17: Constelação do sinal 32QAM, para uma SNR=26 dB.

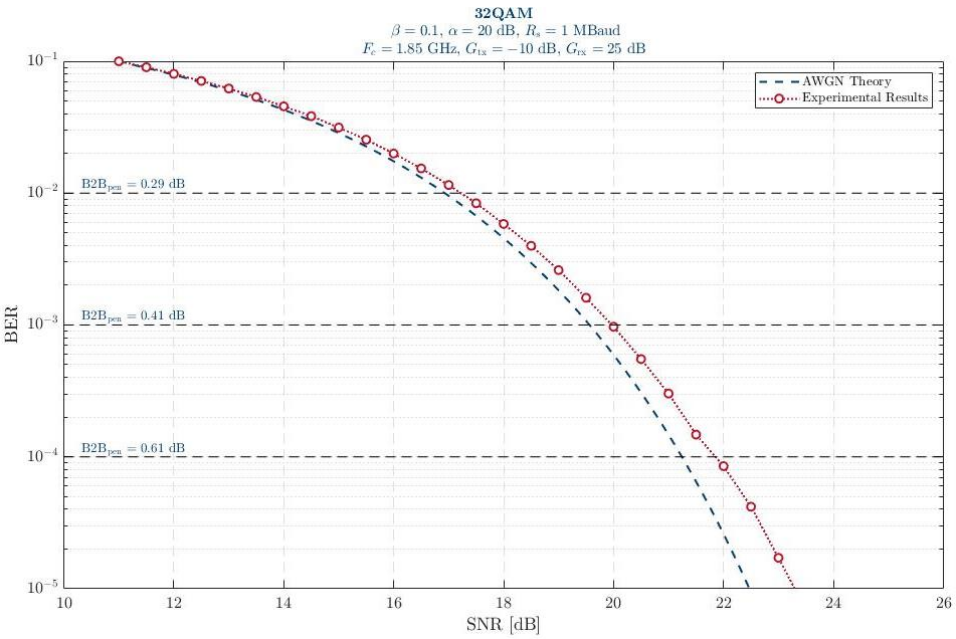


Figura 5.18: Desempenho do sistema com modulação 32QAM.

Após completar a análise do desempenho do sistema 16QAM, seguiu-se o 32QAM. Na Figura 5.17, está representada a sua constelação, em que são visíveis os 32 símbolos diferentes. Tal como no 8QAM e ao contrário do QPSK e do 16QAM, neste tipo de modulação foi utilizada uma constelação em cruz (*cross*). Os símbolos são codificados através da aproximação de *Smith* do código de *Gray* e cada símbolo é codificado por 5 bits [43]. Neste formato de modulação, já é possível verificar que os símbolos exteriores, isto é, os que estão nas extremidades da constelação, têm uma nuvem *Gaussiana* mais larga do que os símbolos que se encontram no centro da constelação. Isto quer dizer que esses símbolos estão a uma maior distância do seu local exato (>EVM). Como se pode verificar, o transcetor tem uma resposta não linear, isto é, gera ruído dependente da potência do sinal, daí que os símbolos exteriores (com maior amplitude) tenham mais ruído adicional.

Tal como em todos os formatos de modulação estudados anteriormente, através da Figura 5.18, pode-se verificar que quanto maior for a SNR melhor vai ser o desempenho do sistema, ou seja, menor vai ser a BER. É possível verificar que a penalidade relativamente à curva teórica vai aumentando à medida que a BER vai diminuindo. Para obter estes valores de BER não é necessário introduzir tanto ruído como nos formatos de modulação anteriores. Um exemplo disso é o caso em que para obter uma BER=10^{-2}, é necessário uma SNR≈14 dB no 16QAM enquanto que no 32QAM necessito de uma SNR≈17 dB. Por isso, contrariamente aos formatos de modulação anteriores, o ruído introduzido deixa de ser o fator dominante na degradação do desempenho do sistema. O fator que influencia mais o desempenho do sistema passa a ser o aparecimento de efeitos não lineares no sistema. Esta distorção não linear é provocada pela necessidade de introduzirmos uma maior potência de pico para obter a mesma BER e pelo facto deste formato de modulação ter um PAPR mais elevado que os anteriores. Como os efeitos não lineares não são contabilizados na expressão teórica, vai haver um afastamento visível da curva prática em relação à teórica. Como consequência, o BER *floor* aumenta com o aumento do formato de modulação. Isto não é mais do que uma limitação do transcetor, que não é capaz de criar o sinal exatamente igual ao teórico. Por estas razões, para BERs mais baixas, temos uma penalidade superior, pois encontramo-nos mais perto do BER *floor* do sistema. Os resultados obtidos são bastante satisfatórios na medida em que a penalidade é bastante reduzida.

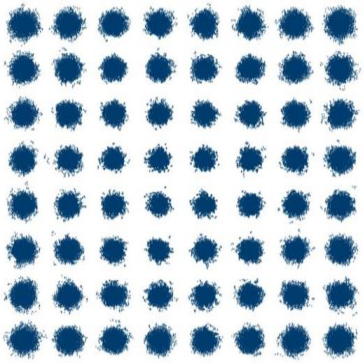


Figura 5.19: Constelação do sinal 64QAM, para uma SNR=29 dB.

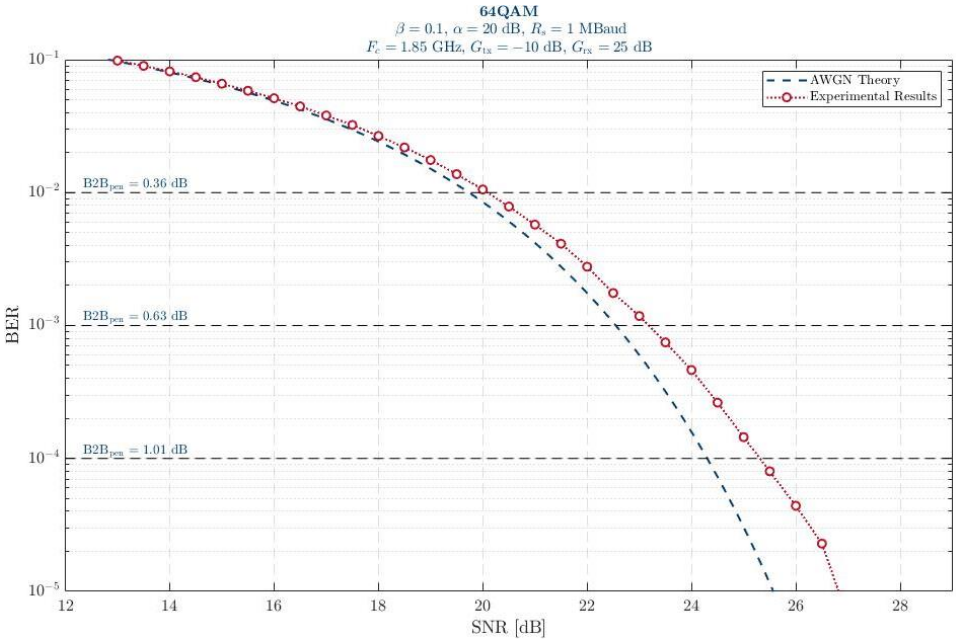


Figura 5.20: Desempenho do sistema com modulação 64QAM.

Depois de completada a análise do desempenho do sistema 32QAM, partiu-se para o próximo formato de modulação na escala de complexidade, o 64QAM. Na Figura 5.19, está representada a sua constelação, composta por 64 símbolos, codificados (código de *Gray*) com 6 bits cada. Tal como o QPSK e o 16QAM, a sua constelação tem o formato de um quadrado. Ainda relativamente à sua constelação, é possível verificar que os símbolos exteriores têm uma nuvem *Gaussiana* mais dispersa do que a dos símbolos centrais, o que quer dizer que esses símbolos estão a uma maior distância do seu local exato, resultando numa maior EVM. Este fenómeno já se tinha verificado no 32QAM, mas aqui é mais percetível, muito pelo facto de ter uma PAPR superior.

Tal como nos formatos de modulação estudados anteriormente, através da Figura 5.20, pode-se verificar que quanto maior for a SNR do sistema melhor vai ser o desempenho do sistema, ou seja, menor vai ser a BER obtida. Pelas razões já referidas, a penalidade relativamente à teoria vai aumentando à medida que vamos para valores de BER mais baixos. Comparativamente com o 32QAM, os valores de penalidade aumentaram. Os resultados obtidos continuam a ser bastante satisfatórios, na medida em que a penalidade, relativamente à curva teórica, é bastante reduzida.

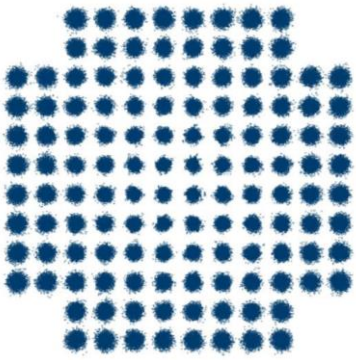


Figura 5.21: Constelação do sinal 128QAM, para uma SNR=32 dB.

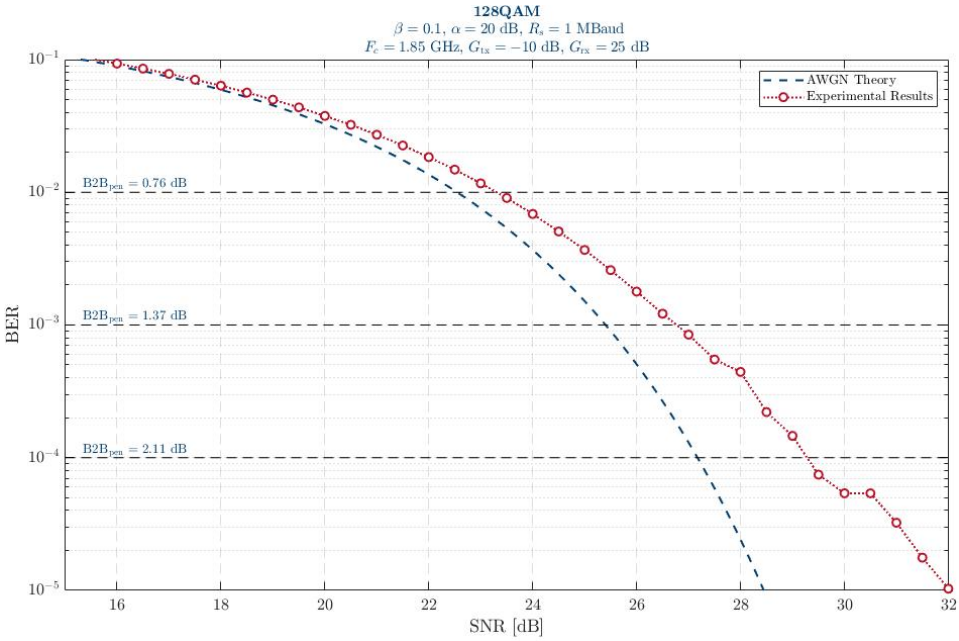


Figura 5.22: Desempenho do sistema com modulação 128QAM.

Prosseguindo no incremento da complexidade do formato de modulação, chegou-se ao 128QAM. Na Figura 5.21, está representada a sua constelação, em que são visíveis os 128 símbolos. Tal como nos formatos em que o número de bits por símbolo é ímpar, vai ser utilizada uma constelação em cruz. Os símbolos são codificados através da aproximação de *Smith* do código de *Gray*, em que cada símbolo é codificado por 7 bits [43]. Tal como no 32QAM e no 64QAM, é possível verificar que os símbolos centrais da constelação apresentam uma nuvem *Gaussiana* mais compacta em relação aos símbolos exteriores, em que é mais dispersa. Isto acontece devido aos efeitos não lineares do sistema. Uma consequência deste fenómeno é um maior EVM dos símbolos dispostos na periferia da constelação relativamente aos símbolos interiores. Comparativamente ao 64QAM, este fenómeno é mais visível, pois o 128QAM tem uma PAPR superior.

Quanto ao desempenho do sistema 128QAM (Figura 5.22), tal como nos restantes formatos, verifica-se que quanto maior for a SNR menor vai ser a BER obtida, ou seja, melhor vai ser o desempenho do sistema. É possível constatar que a penalidade em relação à curva teórica vai aumentando à medida que a BER vai diminuindo e que esses valores obtidos são substancialmente superiores aos do 64QAM, devido ao facto deste formato de modulação precisar de uma maior potência de pico para obter a mesma BER e de ter uma PAPR superior, causando distorção linear que vai degradar o desempenho do sistema. Para finalizar, pode-se afirmar que os resultados obtidos são satisfatórios, na medida em que as penalidades obtidas são aceitáveis (<3 dB).

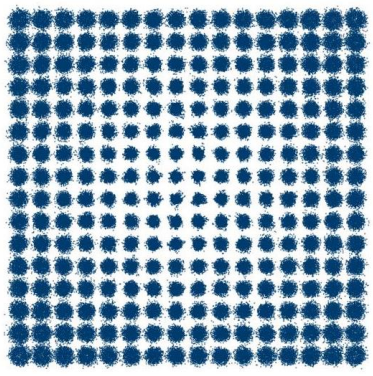


Figura 5.23: Constelação do sinal 256QAM, para uma SNR=35 dB.

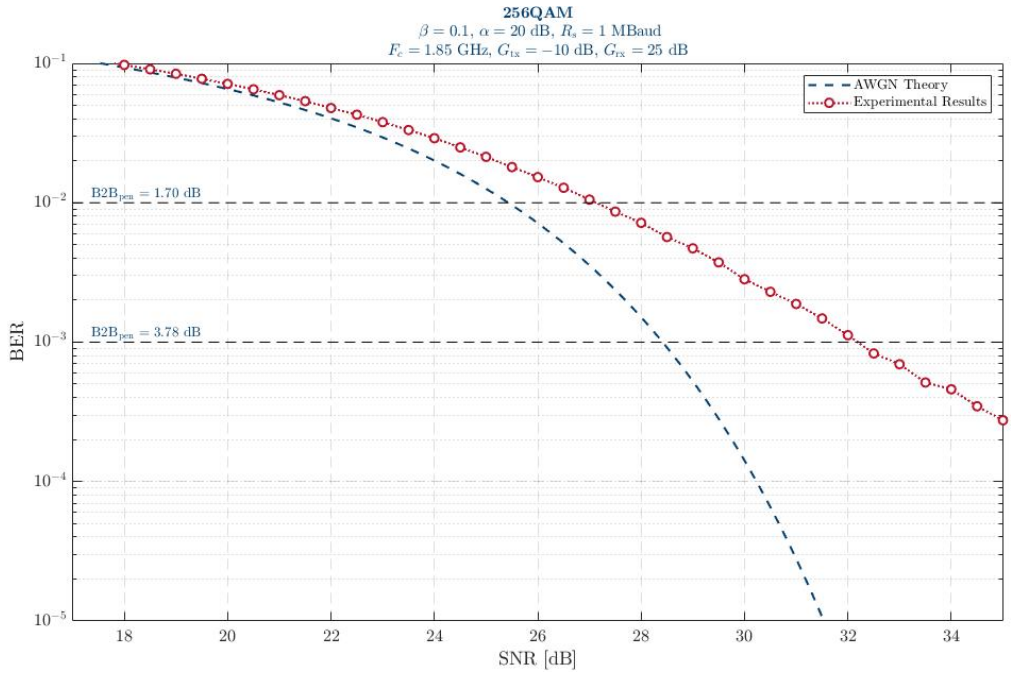


Figura 5.24: Desempenho do sistema com modulação 256QAM.

Terminada a análise do desempenho do sistema 128QAM, iniciou-se a avaliação do 256QAM. Na Figura 5.23, está representada a sua constelação, composta por 256 símbolos, codificados (código de *Gray*) com 8 bits cada. Tal como nos formatos em que o número de bits por símbolo é par, a sua constelação tem o formato de um quadrado. Através da sua constelação, verifica-se que os símbolos exteriores têm uma nuvem *Gaussiana* mais dispersa do que a dos símbolos centrais. Isto quer dizer que os símbolos estão a uma maior distância do seu local exato, resultando numa maior EVM. Como já foi explicado, este fenómeno deve-se à não linearidade do transcetor e já se tinha verificado em outros formatos de modulação, sendo que, aqui, é mais visível, devido a ter uma maior PAPR relativamente aos formatos anteriores.

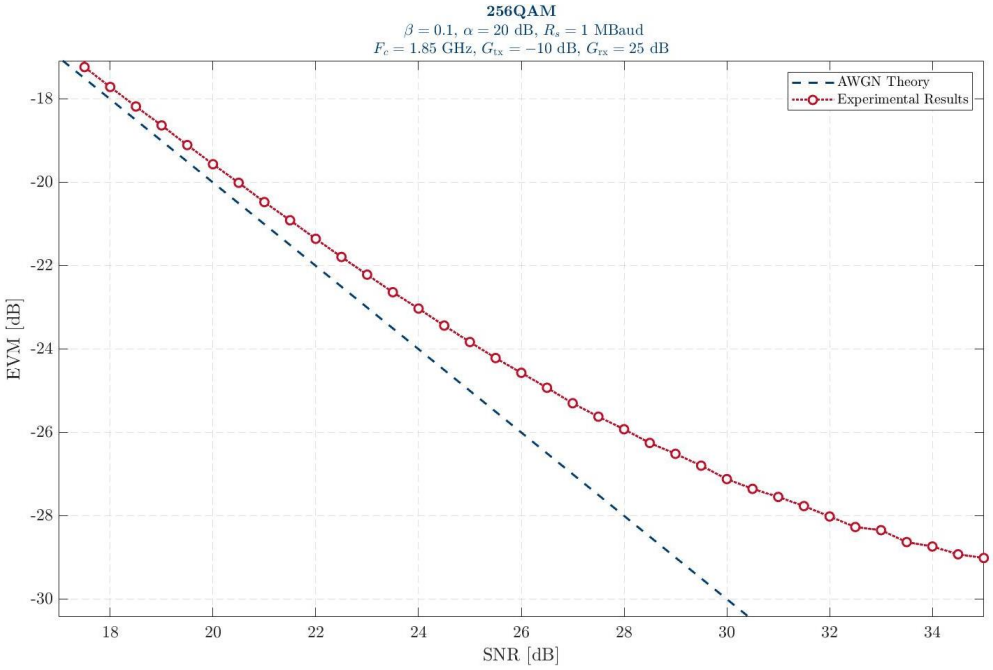


Figura 5.25: Desempenho do sistema (EVM) com modulação 256QAM.

Quanto ao desempenho do sistema (Figura 5.24) tal como nos restantes formatos, verifica-se que quanto maior for a SNR menor vai ser a BER obtida, ou seja, melhor vai ser o desempenho do sistema. Pode-se constatar que a penalidade vai aumentando à medida que a BER vai diminuindo, pelos motivos já mencionados, e que esses valores são superiores aos obtidos para os demais formatos de modulação. Ao contrário dos casos anteriores, os resultados obtidos não são totalmente satisfatórios, na medida em que, na obtenção de BERs inferiores a 10^{-2}, a penalidade face à teoria já não é aceitável (>3 dB). A Figura 5.25 ajuda a entender o porquê de a penalidade ser tão elevada, ao ponto de já não ser visível o valor para uma BER de 10^{-4}, na Figura 5.24. Para além de estarmos perigosamente próximos do BER *floor*, aqui, entram as limitações do próprio *hardware*. Tal como acontece na Figura 5.24, à medida que vamos aumentado a SNR, a penalidade de EVM obtida vai sendo cada vez maior.

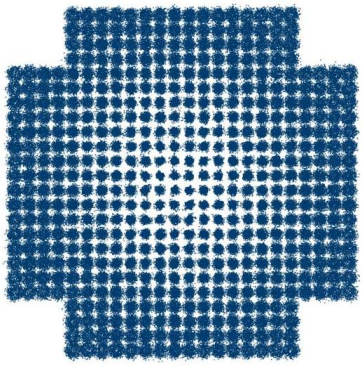


Figura 5.26: Constelação do sinal 512QAM, para uma SNR=38 dB.

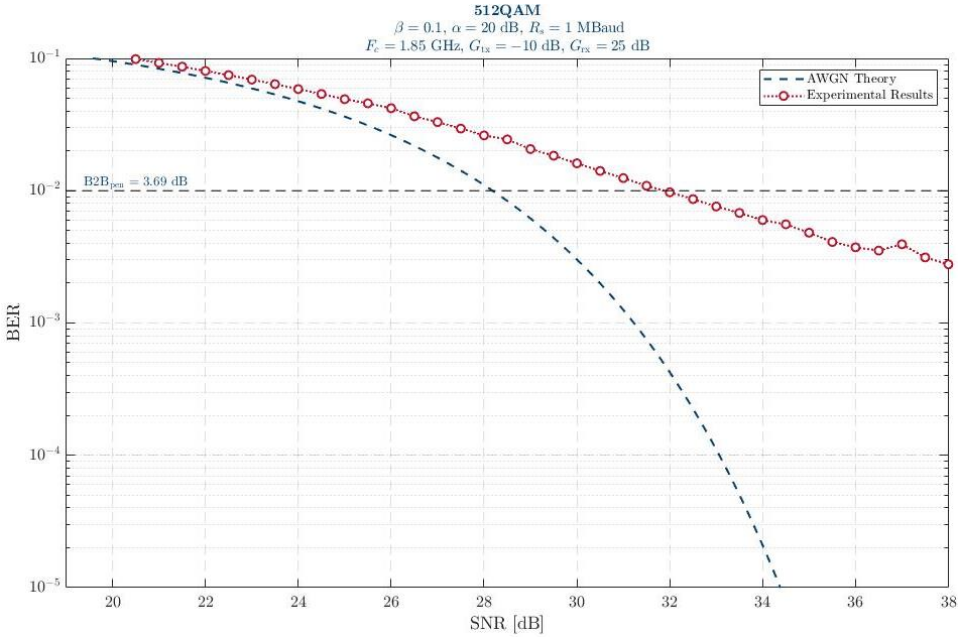


Figura 5.27: Desempenho do sistema com modulação 512QAM.

Para finalizar a avaliação do desempenho do sistema para cada formato de modulação, chegou-se ao formato mais complexo abordado nesta dissertação, o 512QAM. Na Figura 5.26, está representada a sua constelação, em que são visíveis os 512 símbolos. Tal como nos outros formatos em que o número de bits por símbolo é ímpar, é utilizada uma constelação em cruz (*cross*). Os símbolos são codificados através da aproximação de *Smith* do código de *Gray*, em que cada símbolo é codificado por 9 bits [43]. Através da sua constelação, é possível verificar que os símbolos centrais apresentam uma nuvem *Gaussiana* mais compacta em relação aos símbolos exteriores. O facto de a nuvem *Gaussiana* ser mais dispersa nos símbolos exteriores quer dizer que esses símbolos estão

a uma maior distância do seu local exato, ou seja, têm uma maior EVM. Como já foi explicado, este fenómeno deve-se à não linearidade do transcetor e já se tinha verificado em outros formatos de modulação. O 512QAM é o caso em que este fenómeno é mais visível devido ao facto de ser o que tem maior PAPR.

Relativamente ao desempenho do sistema (Figura 5.27), tal como nos restantes formatos, verifica-se que quanto maior for a SNR menor vai ser a BER obtida, ou seja, melhor vai ser o desempenho do sistema. Pelos motivos já mencionados, é possível verificar que a penalidade vai aumentando à medida que a BER vai diminuindo e que esses valores são superiores aos obtidos para os restantes formatos de modulação. Os resultados obtidos não são satisfatórios na medida em que a penalidade face à teoria é sempre superior a 3dB. A razão é a mesma que foi abordada no caso do 256QAM, sendo que, neste caso, é um pouco pior, pois já não é possível ver a penalidade para uma BER de 10^{-3} ($\approx$ 11 dB). O facto de obtermos penalidades tão elevadas de implementação diz-nos que o 256QAM e o 512QAM exigem uma equalização não linear para compensar os efeitos não lineares, introduzidos pelo *hardware*.

Tabela 5.5: Comparação do desempenho do sistema para vários M (valores das Figuras 5.12 a 5.27).

Formato de Modulação (M)	**Penalidade [dB]**		
	BER = 10^{-2}	**BER = 10^{-3}**	**BER = 10^{-4}**
QPSK	0.20	0.10	0.01
8QAM	0.24	0.31	0.25
16QAM	0.22	0.23	0.16
32QAM	0.29	0.41	0.61
64QAM	0.36	0.63	1.01
128QAM	0.76	1.37	2.11
256QAM	1.70	3.78	
512QAM	3.69		

Após a avaliação individualizada do desempenho do sistema para cada formato de modulação, compactaram-se os dados obtidos para a penalidade na Tabela 5.5. Esta tabela tem o objetivo de tornar a leitura e interpretação dos resultados mais simples e rápida. Pela observação dos dados da tabela, chegamos à conclusão de que há duas regiões de operação distintas, a primeira

constituída pelo QPSK, 8QAM e 16QAM e a segunda pelo 32QAM, 64QAM, 128QAM, 256QAM e 512QAM.

Na primeira, o fator que tem maior influência no desempenho do sistema é o ruído AWGN introduzido no sistema, quando a distorção não linear é desprezável. Como já foi explicado anteriormente, quando a SNR é baixa, o ruído adicionado é nocivo para o DSP, pois tem de recuperar o sinal transmitido no meio de uma grande quantidade de ruído. A influência do ruído no DSP é visível na medida em que, para BERs mais baixas (SNRs mais altas), ou seja, quando há menos ruído adicionado, a penalidade é menor. Devido à existência de erro experimental, esta tendência não é tão visível no 8QAM e no 16QAM. Nesta região, o desempenho do sistema testado é muito próximo ao teórico.

Na segunda região, composta pelos formatos de modulação com um maior número de pontos na constelação, o fator que tem maior influência no desempenho do sistema passa a ser a existência de efeitos não lineares (distorção) e a proximidade com o BER *floor* do formato de modulação em questão (explicado com maior detalhe no 32QAM). A necessidade de mais potência de pico para obter a mesma BER e o facto de terem uma PAPR superiores aos formatos da outra região provocam o aparecimento desses efeitos no transmissor e no recetor, que, por não serem contabilizados na expressão teórica, fazem com que a curva prática se afaste da teórica. À medida que vamos aumentando a complexidade do formato de modulação, o BER *floor* vai sendo cada vez maior, deteriorando a resposta do sistema para BERs mais baixas. Esta limitação do transcetor cria uma tendência nesta região, onde à medida que vamos para BERs mais baixas, a penalidade é maior.

Podemos, então, concluir que, para o mesmo valor de BER (por exemplo BER=10^{-2}), a penalidade aumenta com o incremento da complexidade do formato de modulação. Nesta tendência, temos uma exceção, o 8QAM. Era de esperar que o 8QAM tivesse uma penalidade inferior ao 16QAM. Uma explicação possível para a penalidade ser superior à do 16QAM é que o 8QAM, como tem uma constelação em cruz, tem símbolos com 5 níveis de amplitude, em fase e em quadratura (Figura 5.13), enquanto que o 16QAM só tem 4 níveis de amplitude na sua constelação (Figura 5.15). Os pontos da constelação que se encontram no zero do eixo da amplitude são os responsáveis por esta diferença. Como foi mencionado anteriormente, alguns valores de penalidade não foram incluídos na tabela, pois não eram visíveis nos gráficos apresentados.

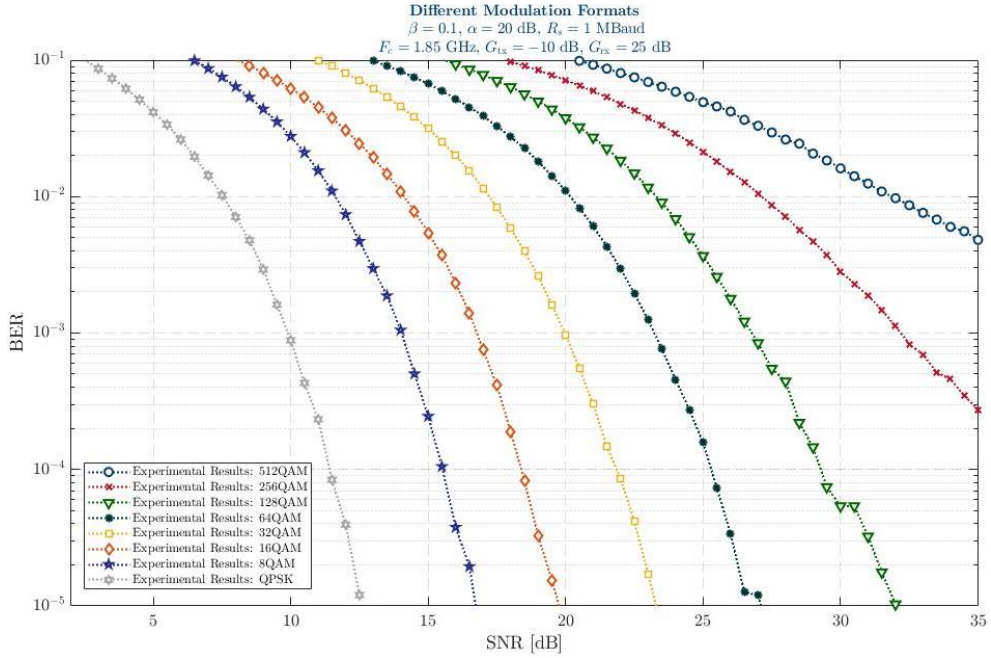


Figura 5.28: Comparação do desempenho do sistema para os vários formatos de modulação.

Após terminar a avaliação do desempenho do sistema para cada um dos formatos de modulação, achou-se interessante compilar num só gráfico (Figura 5.28) todas as curvas obtidas, de modo a que fosse possível comparar o desempenho para cada caso. Na Figura 5.28, é possível observar que há uma tendência comum a todos os formatos de modulação: à medida que se aumenta a SNR, a BER diminui. Isto acontece porque, quanto maior for a SNR, maior vai ser a potência do sinal em relação à do ruído, ou seja, menor vai ser a influência do ruído. Daqui podemos concluir que o sistema tem um melhor desempenho para valores de SNR mais elevados.

Outra conclusão que se pode retirar da Figura 5.28 é que à medida que vamos avançando na complexidade do formato de modulação, isto é, à medida que vamos aumentado o número de símbolos da constelação (M), o desempenho do sistema vai sendo cada vez pior. Um exemplo ilustrativo desta tendência é que, para uma SNR de 12 dB, obtemos valores de BER diferentes para cada formato de modulação: 3.9×10^{-5} para o QPSK, 7.4×10^{-3} para o 8QAM e 3.1×10^{-2} para o 16QAM (valores aproximados). A razão para isto acontecer é que, quanto maior for o tamanho da constelação, mais próximos vão estar os símbolos uns dos outros para a mesma potência média transmitida, ou seja, a distância entre símbolos vai ser menor (visível nas figuras das constelações apresentadas anteriormente). Quanto menor for essa distância, menor vai ser a margem de ruído,

isto é, o sistema vai ser menos tolerante ao ruído. Esta intolerância provoca o aparecimento de um maior número de erros, resultando na degradação do desempenho do sistema.

Para finalizar, podemos concluir que, se não tivermos em conta a eficiência espetral, o formato de modulação testado que apresenta melhor desempenho é o QPSK. Estes resultados estão de acordo com [44].

5.6.2 Desempenho para diferentes frequências de portadora

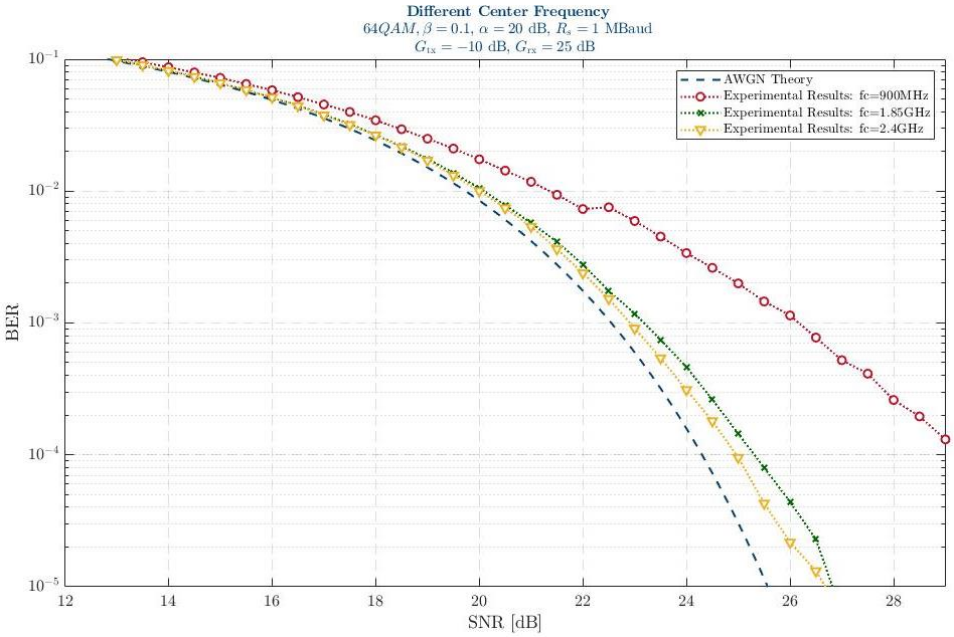


Figura 5.29: Comparação do desempenho do sistema para várias frequências centrais.

Nesta subsecção, é apresentado o gráfico do desempenho do sistema 64QAM consoante a frequência da portadora (ou central) escolhida. Nesse gráfico, é possível verificar a evolução da BER com a SNR, para diferentes valores de frequência. Primeiramente, foram apenas testados três valores de f_c. São eles: 900 MHz, 1.85 GHz e 2.4 GHz. Através da Figura 5.29, pode-se verificar que, quanto maior for a frequência central utilizada, melhor vai ser o desempenho do sistema, isto é, menor vai ser a distância entre a curva prática e a curva teórica. Verificado este comportamento, foi realizada uma caracterização mais pormenorizada dos valores das penalidades (para BER=10^{-2}, 10^{-3} e 10^{-4}) em função da frequência central. Esta caracterização está representada na Figura 5.30.

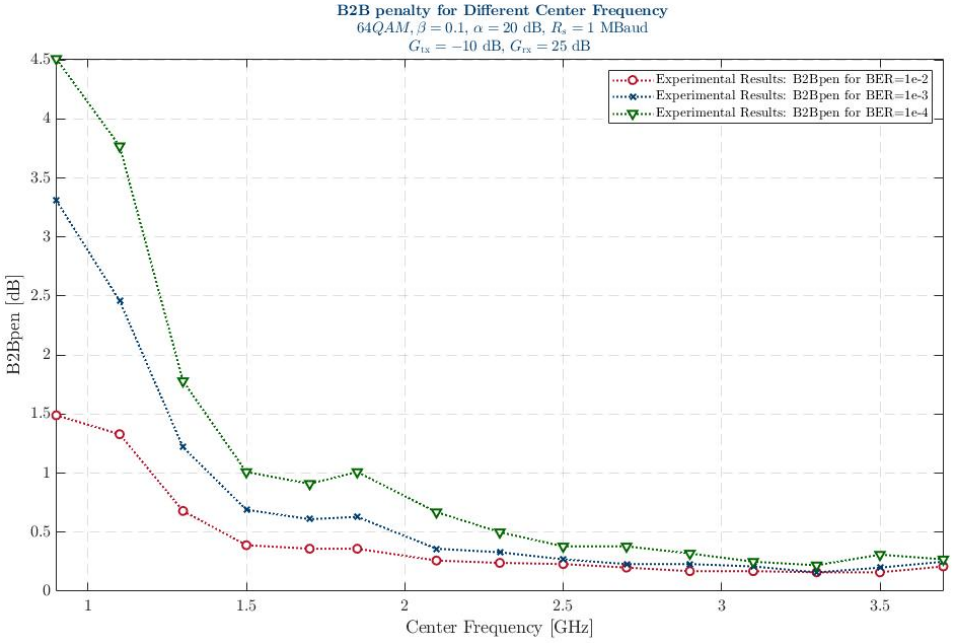


Figura 5.30: Evolução do valor das penalidades consoante a frequência central.

Através da figura anterior, é possível verificar que à medida que vamos aumentando a frequência central menor vai sendo a penalidade (*B2Bpen*). Esta tendência é comum aos três valores de BER testados, na medida em que têm um comportamento muito semelhante. Através destes resultados é possível verificar que o valor de frequência utilizado ao longo da dissertação (f_c=1.85 GHz) não é o valor ótimo no que diz respeito ao desempenho do sistema. Apesar desta constatação, achou-se que não seria necessário refazer os testes até aqui realizados, na medida em que o objetivo da dissertação não é otimizar ao máximo o desempenho deste sistema, mas sim explorar as capacidades do mesmo. Através da Figura 5.30, o valor ótimo de f_c a utilizar é 3.3 GHz, pois é o que apresenta penalidades menores. Pela tendência, seria de esperar que para uma frequência de 3.7 GHz o desempenho fosse ainda melhor, podendo atribuir-se isto ao facto de diferenças inferiores a 0.1 dB, numa medição em que há sempre um erro experimental associado, não serem significativas. A explicação para o facto do sistema se comportar desta maneira (Figura 5.30) prende-se com as limitações a nível do *hardware* que o ADALM-PLUTO apresenta.

6 Conclusão e Trabalhos Futuros

Neste último capítulo, estão presentes as conclusões a que se chegou após a realização da dissertação e, ainda, algumas sugestões de tópicos para trabalhos futuros.

6.1 Conclusão

Nos dias de hoje, em que os equipamentos eletrónicos se tornam obsoletos muito rapidamente, é imperativo a utilização de tecnologias que permitam que esses equipamentos se tornem mais versáteis e adaptáveis. Uma dessas tecnologias é o SDR, que, tal como já foi dito, permite que o mesmo equipamento seja utilizado para várias finalidades.

Como o objetivo da dissertação é renovar o ensino da componente prática de Sistemas de Informação e de Sistemas de Comunicação, no DETI, foi necessário estudar e implementar um novo módulo laboratorial de ensino. Depois de analisados os principais equipamentos que utilizam a tecnologia SDR e os diferentes *softwares* existentes, concluiu-se que o ideal seria utilizar o ADALM-PLUTO com recurso ao MATLAB.

Terminada a escolha do sistema a implementar, foi realizada a otimização de alguns parâmetros e realizado um vasto conjunto de simulações que permitiram avaliar o desempenho do sistema. Da otimização de parâmetros foi possível concluir que há um par de ganhos ótimo e que quanto menor for o fator de decaimento melhor vai ser o desempenho do sistema.

Através da avaliação do desempenho, foi possível retirar algumas conclusões importantes. A primeira foi que quanto maior for a SNR menor vai ser a BER obtida, sendo comum a todos os formatos de modulação testados. Relativamente à penalidade obtida, concluiu-se que é maior quanto menor for a BER obtida, ou seja, o sistema tem um pior desempenho para valores de BER mais baixos. Esta conclusão apenas não é percetível para os formatos de modulação QPSK, 8QAM e 16QAM, em que as diferenças de penalidade não são tão visíveis devido à taxa de erro mínima testada ser muito superior ao BER *floor*.

Depois de realizada a avaliação do desempenho para todos os formatos de modulação indicados, foi possível concluir que quanto maior for o M maior vai ser a penalidade do sistema, isto é, quanto maior for a complexidade do formato de modulação pior vai ser o desempenho do sistema.

Através da análise dos resultados obtidos para cada formato de modulação, é possível, ainda, concluir que esses mesmos resultados são bastantes satisfatórios, à exceção do 256QAM e o do 512QAM, onde as penalidades obtidas já são consideradas elevadas. Posteriormente, foi realizado um teste que avalia a dependência do desempenho do sistema com a frequência da portadora escolhida (f_c), concluindo-se que quanto maior for essa frequência, dentro da gama de frequências de operação admissível, melhor vai ser o desempenho do sistema.

Após a realização desta dissertação, conclui-se que a tecnologia em estudo é uma mais-valia no ensino de Sistemas de Comunicação e de Sistemas de Informação, na medida em que abre a possibilidade de fazer demonstrações, em contexto de sala de aula, com relativa facilidade, motivando o aluno a explorar esta área e ajudando-o na compreensão de alguns conceitos, que, se fossem abordados apenas teoricamente, teriam mais dificuldade.

O módulo de ensino laboratorial escolhido, ADALM-PLUTO com recurso ao MATLAB, revelou-se uma ótima escolha devido à simplicidade de utilização e ao facto da utilização de ferramentas de alto nível (como o MATLAB) possibilitar um ganho de produtividade, na medida em que permite que o utilizador se abstraia de certos detalhes de implementação. Esta abstração faz com que os alunos se foquem quase exclusivamente nos aspetos funcionais do transmissor e do recetor.

Para finalizar, esta dissertação fornece um conjunto de ficheiros de código MATLAB, que, com pequenas adaptações, pode ser utilizado com outros dispositivos SDR.

6.2 Trabalhos Futuros

Concluída esta dissertação e cumprido o objetivo principal da mesma de explorar a capacidade de utilização do ADALM-PLUTO no ensino, são sugeridos vários tópicos que considero serem interessantes como trabalho futuro.

Para tornar a aprendizagem mais intuitiva e simples, o código MATLAB pode ser agrupado e esquematizado em blocos do Simulink, onde os alunos apenas precisam de especificar alguns parâmetros, sem que tenham necessidade de aceder ao núcleo do código.

Outro tópico de estudo interessante seria a implementação de códigos de correção de erros, que permitiria que o sistema funcionasse para BERs mais baixas do que aquelas testadas nesta dissertação. Estes códigos iriam permitir uma melhoria no desempenho do sistema implementado.

A exploração de algumas das limitações do sistema, tal como o facto de apresentar um pior desempenho a frequências mais baixas e a circunstância de não se conseguir aumentar a taxa de amostragem, é outro dos tópicos que seria interessante aprofundar no futuro.

O estudo e compensação dos efeitos não-lineares presentes no sistema seria outro tópico interessante, na medida em que permitiria obter um conhecimento mais aprofundado do mesmo.

Seria interessante fazer um conjunto de testes num cenário mais realista, isto é, em que o cabo utilizado na ligação transmissor-recetor seria substituído pelas antenas emissora e recetora.

Num nível de complexidade superior, seria interessante incluir um sistema com múltiplas portadoras (OFDM).

Referências

[1] A. F. B. Selva, A. L. G. Reis, K. G. Lenzi, L. G. P. Meloni, and S. E. Barbin, “Introduction to the software-defined radio approach,” *IEEE Lat. Am. Trans.*, vol. 10, no. 1, pp. 1156–1161, 2012.

[2] X. Cai, M. Zhou, and X. Huang, “Model-Based Design for Software Defined Radio on an FPGA,” *IEEE Access*, vol. 5, pp. 8276–8283, 2017.

[3] G. Suciu, M. Vochin, C. Diaconu, V. Suciu, and C. Butca, “Convergence of software defined radio: WiFi, ibeacon and epaper,” in *Networking in Education and Research: RoEduNet International Conference 15th Edition, RoEduNet 2016 - Proceedings*, 2016.

[4] E. Haque, T. F. A. Nayna, and F. Ahmed, “An activating framework using software defined radio to enhance classroom teaching for wireless communication courses in developing countries,” in *Proceedings of 2016 SAI Computing Conference, SAI 2016*, 2016, pp. 823–827.

[5] Elettronica Veneta, “MCM31 / EV (DIGITAL MODULATION).” [Online]. Disponível em: http://www.elettronicaveneta.com/education/index.php?option=com_docman&task=doc_details&gid=2360&Itemid=728. [Acedido em: 08-Jun-2018].

[6] Analog Devices Inc, “ADALM-PLUTO SDR Active Learning Module.” [Online]. Disponível em: http://www.analog.com/media/en/news-marketing-collateral/product-highlight/ADALM-PLUTO-Product-Highlight.pdf. [Acedido em: 11-Jun-2018].

[7] T. W. I. Forum, “What is Software Defined Radio?” [Online]. Disponível em: https://www.wirelessinnovation.org/Introduction_to_SDR. [Acedido em: 08-Jun-2018].

[8] ITU-R, “Report ITU-R Sm.2152: Definitions of Software Defined Radio (SDR) and Cognitive Radio System (CRS),” Genebra, 2009.

[9] J. Mitola, “Software radios: Survey, critical evaluation and future directions,” in *Software Radio Technologies: Selected Readings*, 2001, pp. 3–11.

[10] P. G. Cook and W. Bonser, “Architectural overview of the SPEAKeasy system,” *IEEE J. Sel. Areas Commun.*, vol. 17, no. 4, pp. 326–336, 2001.

[11] Y. K. Kwag, J. S. Jung, I. S. Woo, and M. S. Park, “Multi-band multi-mode SDR radar platform,” in *Proceedings of the 2015 IEEE 5th Asia-Pacific Conference on Synthetic Aperture Radar, APSAR 2015*, 2015, pp. 46–49.

[12] K. Lin, “QoE-Driven Spectrum Assignment for 5G Wireless Networks Using SDR,” *IEEE Wirel. Commun.*, vol. 22, no. December, pp. 48–55, 2015.

[13] D. Sinha, A. K. Verma, and S. Kumar, “Software defined radio: Operation, challenges and possible solutions,” in *Proceedings of the 10th International Conference on Intelligent Systems and Control, ISCO 2016*, 2016.

[14] LuaRadio, “New to SDR?” [Online]. Disponível em: http://luaradio.io/new-to-sdr.html. [Acedido em: 11-Jun-2018].

[15] National Instruments, “USRP B200/B210 Bus Series.” [Online]. Disponível em: https://www.ettus.com/content/files/b200-b210_spec_sheet.pdf. [Acedido em: 11-Jun-2018].

[16] “About RTL-SDR.” [Online]. Disponível em: https://www.rtl-sdr.com/about-rtl-sdr/. [Acedido em: 11-Jun-2018].

[17] The GNU Radio Foundation Inc, “About GNU Radio.” [Online]. Disponível em: https://www.gnuradio.org/about/. [Acedido em: 11-Jun-2018].

[18] National Instruments, “O que é o LabVIEW?” [Online]. Disponível em: http://www.ni.com/newsletter/51141/pt/. [Acedido em: 12-Jun-2018].

[19] MathWorks, “MATLAB.” [Online]. Disponível em: https://www.mathworks.com/products/matlab.html. [Acedido em: 12-Jun-2018].

[20] R. Getz and M. Hennerich, “Stupid Pluto Tricks with the ADALM-PLUTO,” *Fosdem*, 2018. [Online]. Disponível em: https://fosdem.org/2018/schedule/event/plutosdr/attachments/slides/2503/export/events/attachments/plutosdr/slides/2503/pluto_stupid_tricks.pdf. [Acedido em: 12-Jun-2018].

[21] Analog Devices Inc, “Data Sheet RF Agile Transceiver AD9363.” [Online]. Disponível em: http://www.analog.com/media/en/technical-documentation/data-sheets/AD9363.pdf. [Acedido em: 13-Jun-2018].

[22] Analog Devices Inc, “What is libiio?” [Online]. Disponível em: https://wiki.analog.com/resources/tools-software/linux-software/libiio. [Acedido em: 13-Jun-2018].

[23] Analog Devices Inc, “ADALM-PLUTO Overview.” [Online]. Disponível em: https://wiki.analog.com/university/tools/pluto. [Acedido em: 13-Jun-2018].

[24] Analog Devices Inc, “EngineerZone - Analog Devices Support Community.” [Online]. Disponível em: https://ez.analog.com/. [Acedido em: 13-Jun-2018].

[25] MathWorks, “ADALM-PLUTO Radio Support from Communications System Toolbox.” [Online]. Disponível em: https://www.mathworks.com/hardware-support/adalm-pluto-radio.html. [Acedido em: 20-Jun-2018].

[26] MathWorks, “Communications System Toolbox Support Package for Analog Devices ADALM-Pluto Radio Examples.” [Online]. Disponível em: https://www.mathworks.com/help/supportpkg/plutoradio/examples.html. [Acedido em: 20-Jun-2018].

[27] MathWorks, “Communications System Toolbox Support Package for Analog Devices ADALM-Pluto Radio Functions.” [Online]. Disponível em: https://www.mathworks.com/help/supportpkg/plutoradio/functionlist.html. [Acedido em: 21-Jun-2018].

[28] MathWorks, “ConfigurePlutoRadio.” [Online]. Disponível em: https://www.mathworks.com/help/supportpkg/plutoradio/ref/configureplutoradio.html. [Acedido em: 21-Jun-2018].

[29] MathWorks, “What Are System Objects?” [Online]. Disponível em: https://www.mathworks.com/help/matlab/matlab_prog/what-are-system-objects.html. [Acedido em: 21-Jun-2018].

[30] MathWorks, “Comm.SDRRxPluto System object.” [Online]. Disponível em: https://www.mathworks.com/help/supportpkg/plutoradio/ref/comm.sdrrxpluto-system-object.html. [Acedido em: 21-Jun-2018].

[31] MathWorks, “comm.SDRTxPluto System object.” [Online]. Disponível em: https://www.mathworks.com/help/supportpkg/plutoradio/ref/comm.sdrtxpluto-system-object.html. [Acedido em: 22-Jun-2018].

[32] Keysight Technologies, “437B High Performance Single Channel Average Power Meter.” [Online]. Disponível em: https://www.keysight.com/en/pd-1000001373%3Aepsg%3Apro-pn-437B/high-performance-single-channel-average-power-meter?cc=US&lc=eng. [Acedido em: 26-Jun-2018].

[33] Keysight Technologies, “8487D Diode Power Sensor.” [Online]. Disponível em: https://www.keysight.com/en/pd-1000001943%3Aepsg%3Apro-pn-8487D/diode-power-sensor?cc=US&lc=eng. [Acedido em: 26-Jun-2018].

[34] R. Zitouni, L. George, and B. Copernic, “Output Power Analysis of a Software Defined Radio Device,” in *2016 IEEE Radio and Antenna Days of the Indian Ocean (RADIO)*, 2016, pp. 6–7.

[35] F. P. Guiomar, B. Maximino, P. Loureiro, and P. P. Monteiro, “ADALM-PLUTO SDR Experimental Data: Single-Carrier QAM at 1 MBaud,” 04-Nov-2018. [Online]. Disponível em: https://doi.org/10.5281/zenodo.1477072. [Acedido em: 06-Nov-2018].

[36] F. Guiomar, “OptDSP: Digital signal processing MATLAB library for the simulation of coherent optical communication systems (v1.1),” 2017. [Online]. Disponível em: https://www.zenodo.org/record/883156#.W0coEtJKjct. [Acedido em: 12-Jun-2018].

[37] Q. Observations, D. C. Rife, S. Member, and R. R. Boorstyn, “Single-Tone Parameter Estimation from,” *IEEE Inform. Theory*, vol. I, no. 5, pp. 591–598, 1974.

[38] M. S. Faruk and K. Kikuchi, “Compensation for in-phase/quadrature imbalance in coherent-receiver front end for optical quadrature amplitude modulation,” *IEEE Photonics J.*, vol. 5, no. 2, 2013.

[39] G. H. Godard, “Self-recovering equalization and carrier tracking in two dimensional data communication systems,” *IEEE Trans. Commun.*, vol. 28, no. 11, pp. 1867–1875, 1980.

[40] S. Hoffmann *et al.*, “Frequency Estimation and Compensation for Coherent QPSK Transmission with DFB Lasers,” *Coherent Opt. Technol. Appl.*, vol. 20, no. 2, pp. 1569–1571, 2008.

[41] A. J. Viterbi and A. M. Viterbi, “Nonlinear estimation of PSK-modulated carrier phase with applications to burst digital transmission,” *IEEE Inform. Theory*, vol. IT-29, no. 4, pp. 543–551, 1983.

[42] F. P. Guiomar, L. Bertignono, A. Nespola, and A. Carena, “Frequency-Domain Hybrid Modulation Formats for High Bit-Rate Flexibility and Nonlinear Robustness,” *J. Light. Technol.*, vol. 36, no. 20, pp. 1–1, 2018.

[43] P. K. Vitthaladevuni, M. S. Alouini, and J. C. Kieffer, “Exact BER computation for cross QAM constellations,” *IEEE Trans. Wirel. Commun.*, vol. 4, no. 6, pp. 3039–3050, 2005.

[44] R. E. Ziemer and W. H. Tranter, *Principles of Communication Systems, Modulation, and Noise*, 7th ed. Wiley, 2014.

[45] Ettus Research, “USRP Hardware Driver and USRP Manual.” [Online]. Disponível em: http://files.ettus.com/manual/page_usrp_b200.html. [Acedido em: 20-Jun-2016].

Apêndice A – Manual de instalação do Sistema MATLAB-PLUTO

Este documento visa explicar, de uma forma simples, os passos que precedem à utilização do ADALM-PLUTO com o *software* MATLAB.

Requisitos do sistema:

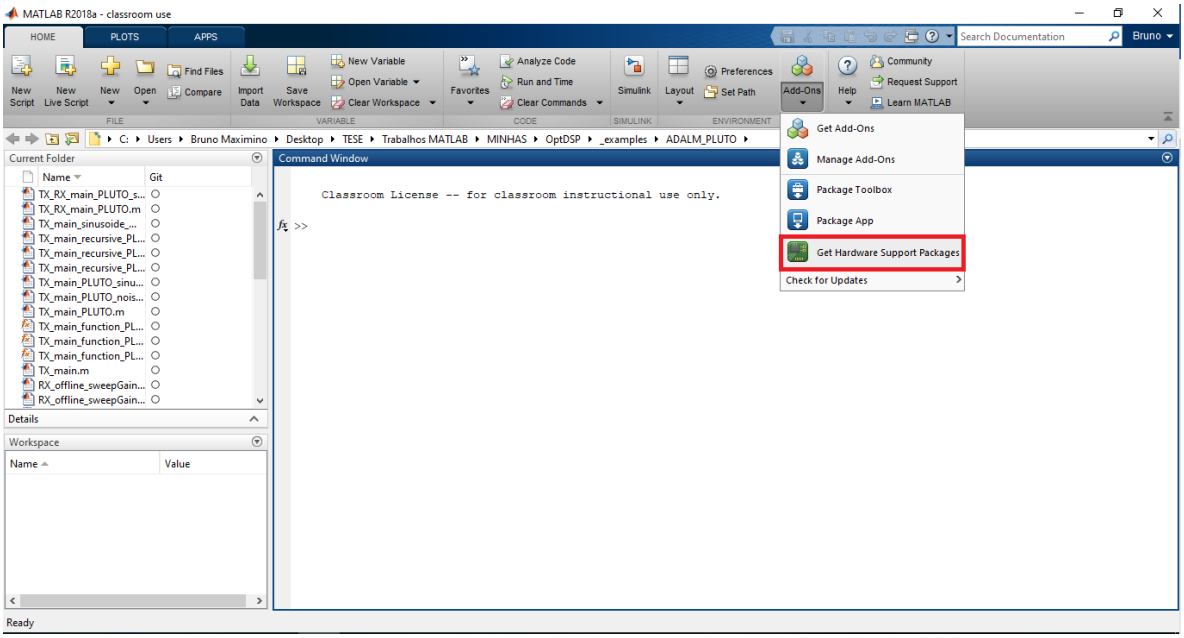
- MATLAB R2018a ou posterior;
- ADALM-PLUTO "*SDR Active Learning Module*";
- Chave de torque (opcional);
- Atenuadores, 20 dB ou 30 dB (opcional);
- 2 antenas;
- Cabo;

Seguidamente, vão ser facultadas todas as instruções necessárias para a realização das tarefas pretendidas com o ADALM-PLUTO, através do MATLAB.

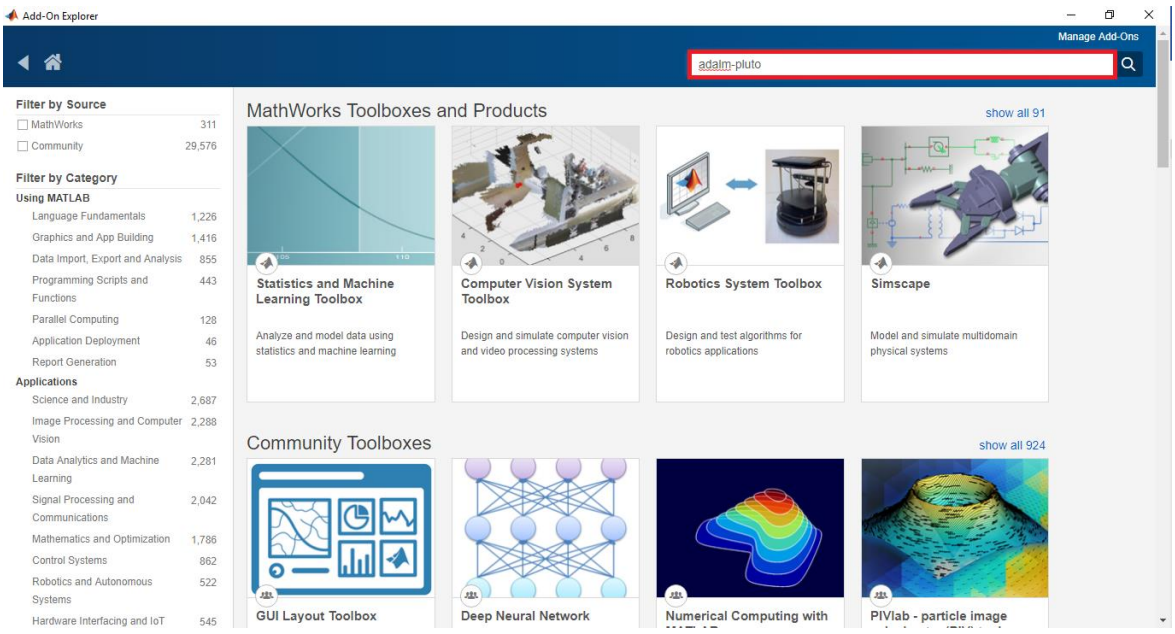
Passo 1: Instalação do MATLAB R2018a.

Passo 2: Instalação das *Toolboxes* do MATLAB necessárias.

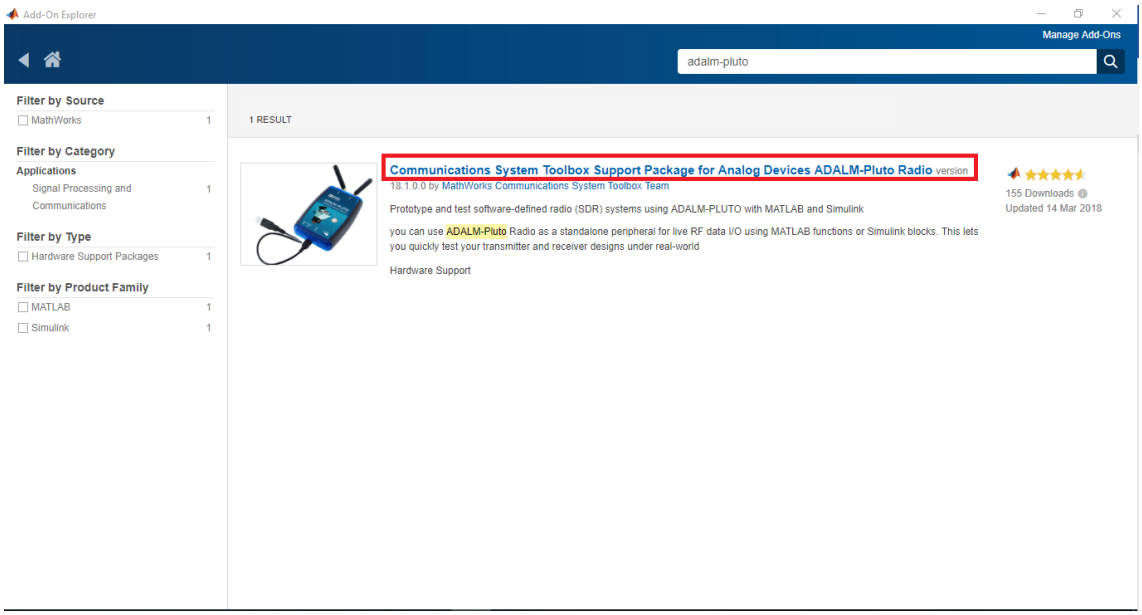
Com o MATLAB aberto, na janela "*HOME*", na secção *"Environment",* clique em *"Add-Ons".* Dentro dessa janela, clique em *"Get Hardware Support Packages":*



Seguidamente, na janela *"search for add-ons",* pesquise por ADALM-PLUTO:

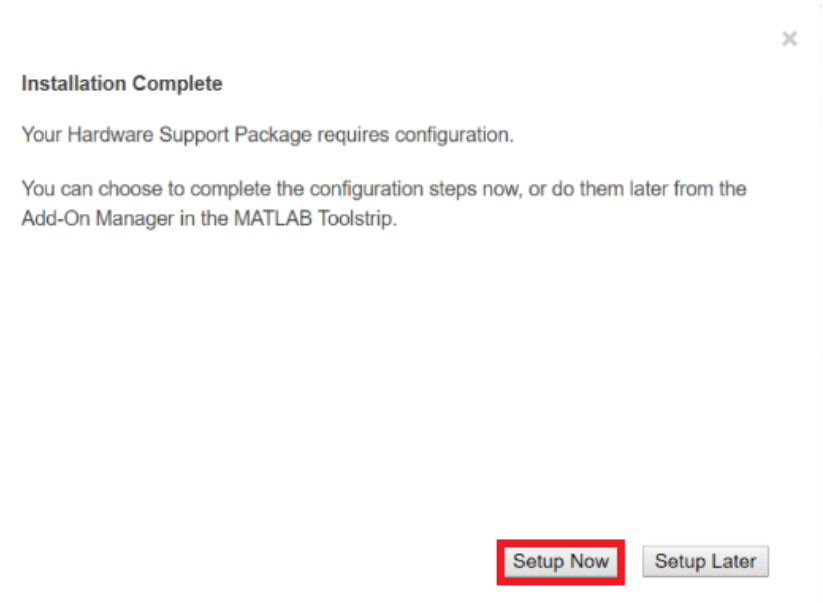


Selecione a opção *"Communications System Toolbox Support Package for Analog Devices ADALM-Pluto Radio"* e clique em *"install":*

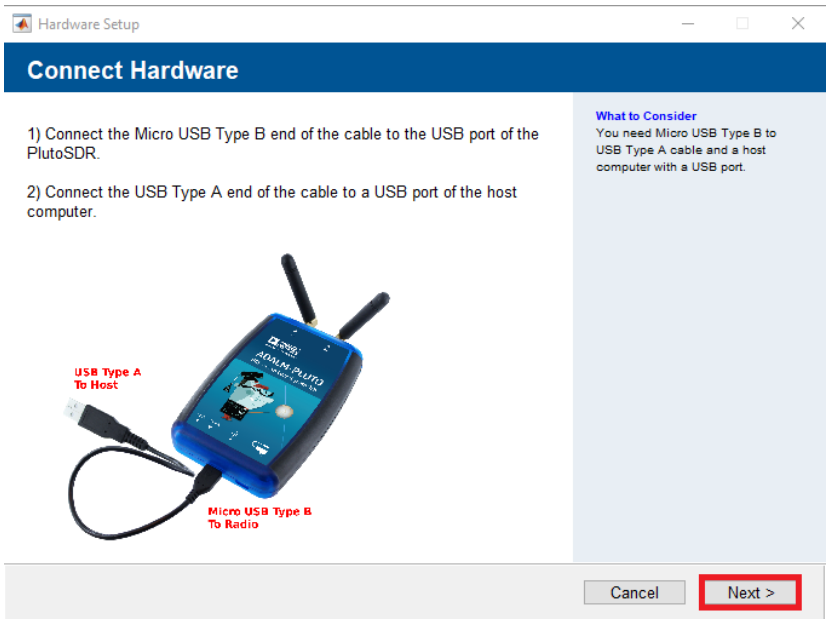


Na figura seguinte, clique em *"next*" e espere até a instalação estar completa.

Quando a instalação acabar, escolha a opção *"setup now".*

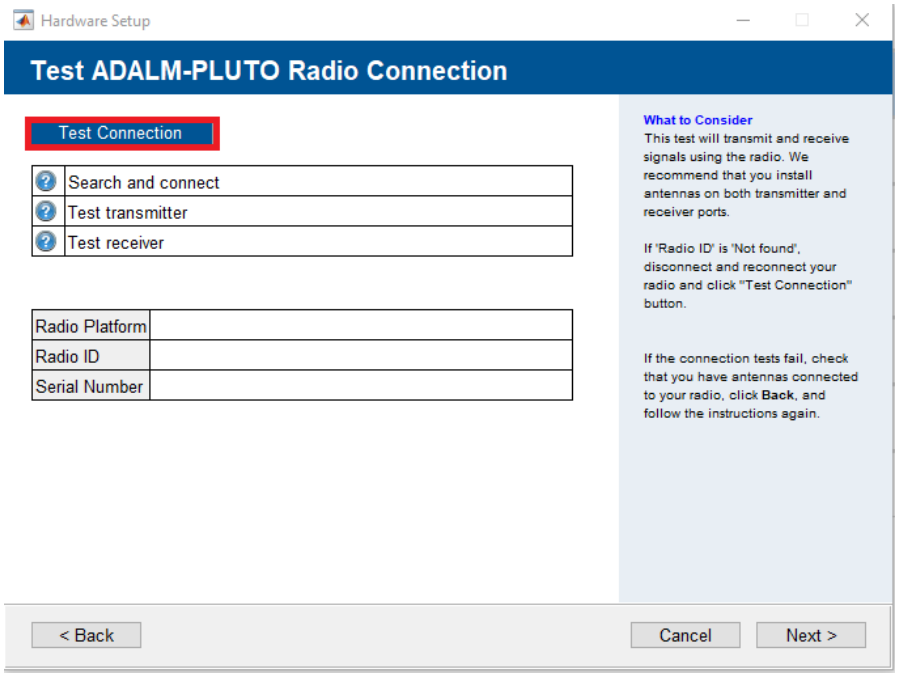


Agora, vai aparecer a seguinte página:

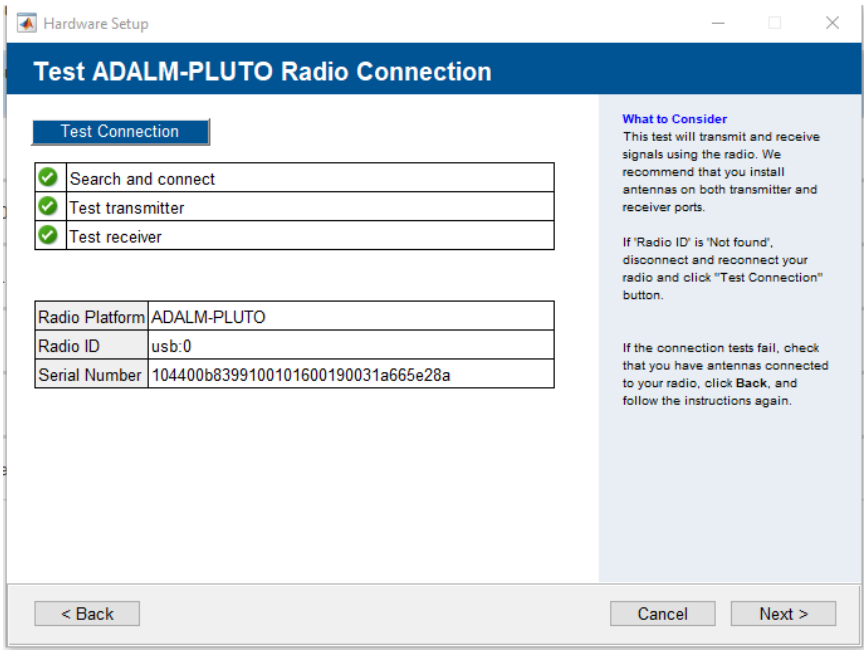


Conecte o cabo USB do ADALM-PLUTO ao seu PC e clique em *"Next"* (figura anterior). Agora, conecte uma antena na porta TX e outra na porta RX. Alternativamente, conecte uma das extremidades do cabo à porta TX e outra à porta RX.

Clique em "*Test Connection*" para verificar se existe conexão.

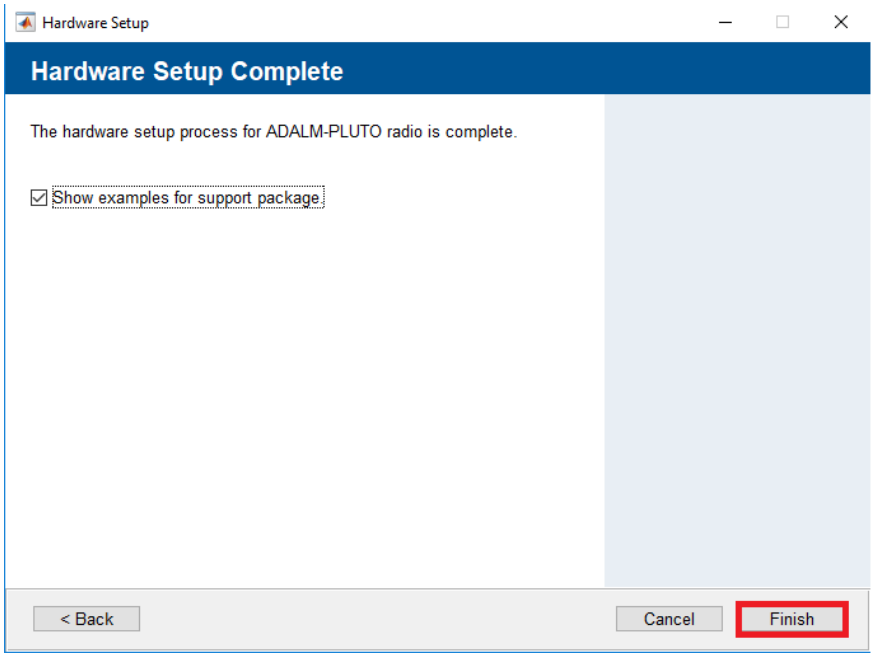


Depois de realizado o teste de conexão, terá de obter a seguinte figura, sendo que o *"Serial Number"* varia consoante o ADALM-PLUTO utilizado:

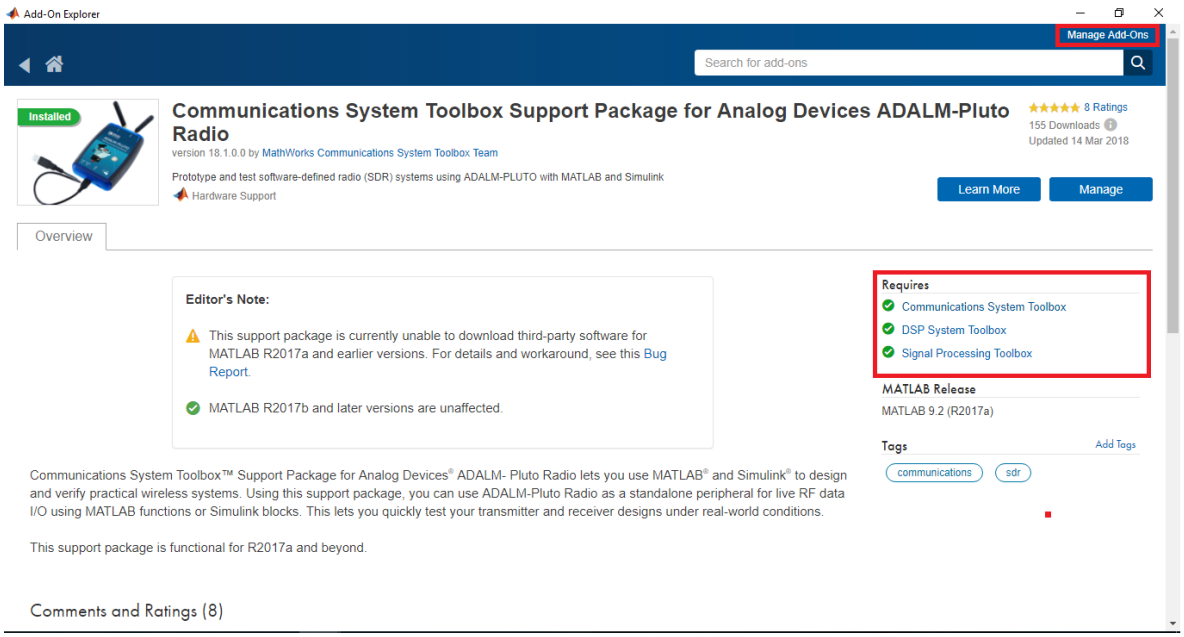


Se algum dos testes falhou (cruz vermelha, na figura anterior), repita este passo até aparecem três setas verdes.

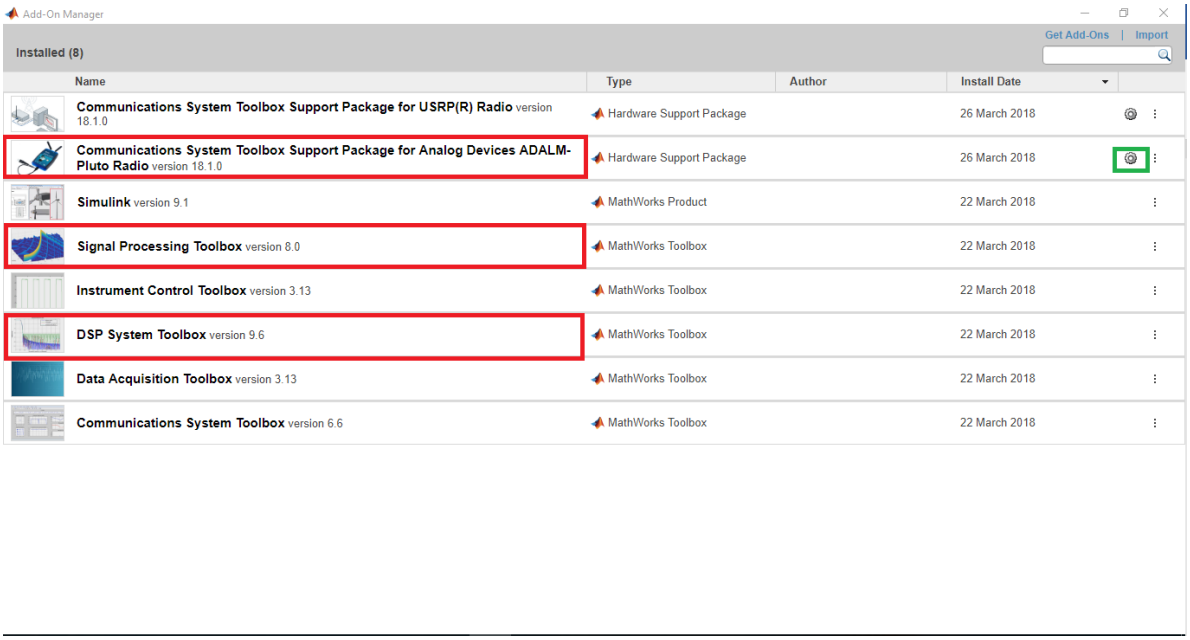
Depois disto, clique em *"finish"* para finalizar a configuração.



Neste momento a *toolbox* do ADALM-PLUTO já se encontra instalada. Agora proceda à instalação das restantes *toolboxes* necessárias (especificadas no campo *"Requires")*



Para verificar se as *toolboxes* estão instaladas, clique em "*Manage Add-Ons*". Caso apareça a janela seguinte, pode avançar na instalação. Caso não apareçam as *toolboxes* instaladas, proceda, novamente, à sua instalação.



Neste momento, está apto para utilizar o ADALM-PLUTO com recurso ao MATLAB R2018a.

Erros mais comuns e resolução:

https://www.mathworks.com/help/supportpkg/plutoradio/ug/common-problems-and-fixes.html

Mais informação sobre o ADALM-PLUTO (*Hardware*):

http://www.analog.com/en/design-center/evaluation-*hardware*-and-*software*/evaluation-boards-kits/adalm-pluto.html#eb-overview

Folha de especificações *(data sheet)* do ADALM-PLUTO:

http://www.analog.com/media/en/news-marketing-collateral/product-highlight/ADALM-PLUTO-Product-Highlight.pdf (simplificado)

https://wiki.analog.com/university/tools/pluto/users/customizing

Mais informação sobre as funções utilizadas no MATLAB:

Recetor:https://www.mathworks.com/help/supportpkg/plutoradio/ref/comm.sdrrxpluto-system-object.html

Emissor: https://www.mathworks.com/help/supportpkg/plutoradio/ref/comm.sdrtxpluto-system-object.html

Conexão: https://www.mathworks.com/help/supportpkg/plutoradio/ref/findplutoradio.html

Apêndice B – Código MATLAB

Medição da potência de saída do ADALM-PLUTO

```
% TITLE:   ADALM-PLUTO Transmitter for Ptx measure
%
% Authors: Fernando Guiomar <guiomar@av.it.pt> and Bruno Maximino
% <brunomaximino@av.it.pt>
% Last Update: 06/03/2018

clear;
clear global;
close all;
clc;

%% Load Libraries
addpath('.\functions\');
addpath(genpath('..\..\'));

%% Initialization
global PROG;
initProg();
PROG.showMessagesLevel = 2;
instrreset

%% Input Parameters
sampleRate = 200e3;
dt = 1/sampleRate;
nSamples = 2^14;
f = 10;

%% Prepare Tx Signal
t = (0:nSamples-1)*dt;
% sinusoid generation
Stx = sin(2*pi*t*f);

%% Connect to ADALM-PLUTO
connectedRadios = findPlutoRadio;

if ~isempty(connectedRadios)
% check if there is any radio connected to the PC
  platform = connectedRadios(2).RadioID;
  address = connectedRadios(2).SerialNum;
else
    error('Could not connect to any radio...')
end

%% SDRTxPluto Transmit data to an ADALM-PLUTO radio
% TX = sdrtx('Pluto') creates an SDR transmitter System object, TX, that
% transmits data to an ADALM-PLUTO radio.  Although the ADALM-PLUTO
```

```
% transmitter System object sends data to an ADALM-PLUTO radio, the
object
% acts as a sink in MATLAB.

% FrequencyCorrection - Frequency correction (ppm)
% Specify the frequency correction value in ppm as a double precision
scalar.
% The valid range is [-200, 200] ppm. The default value is 0, which
% corresponds to the factory-calibrated setting of the radio. This
% property value specifies the parts-per-million change to the baseband
% sample rate and the center frequency.

% send data to ADALM-PLUTO device and specify some parameters:
radioTx = comm.SDRTxPluto(...
    'RadioID',              platform, ...
    'CenterFrequency',      1.85e9, ...
    'Gain',                 -30, ...    % [-89.75;0] dB
    'BasebandSampleRate',   2e5, ...    % SampleRate
    'ChannelMapping',       1, ...      % This property is read-only
    'FrequencyCorrection',  -2);        % [-200;200] ppm

%% Transmit Signal
nBlocks = 1e6;
n = 0;
while n < nBlocks
    tic
    step(radioTx, Stx.');
    elapsedTime = toc;
    dataRate = numel(Stx)/elapsedTime
    n = n + 1;
end

%% Exit PROG
exitProg();
```

Receção de dados através do ADALM-PLUTO

RX_main_recursive_PLUTO_noise.m

```
%% TITLE:   ADALM-PLUTO Receiver recursive script with AWGN
%
% Authors: Fernando Guiomar <guiomar@av.it.pt> and Bruno Maximino
% <brunomaximino@av.it.pt>
% Last Update: 23/04/2018

clear;
clear global;
close all;
clc;

%% Load Libraries:
addpath('.\functions\');
addpath(genpath('..\..'));
```

```
%% Input parameters:
M = 64;                                 % Modulation format
Gtx = -10;                              % Transmitter Gain
Grx = 25;                               % Receiver Gain

%% Connect to ADALM-PLUTO:
connectedRadios = findPlutoRadio;

if ~isempty(connectedRadios)            % check if there is any radio
connected to the PC
    address = connectedRadios(1).RadioID;
    serial = connectedRadios(1).SerialNum;
else
    error('Could not connect to any ADALM-PLUTO...')
end

% first run RX -> connectedRadios(1)

%% Receive Signal:
n = 1;
SNR = floor(ber2snr(1e-1,M)):0.5:ceil(ber2snr(1e-9,M)); % SNR target
values tested

while n <= length(SNR)

    z = fopen('file.txt');

    if z > 0                            % if file.txt exists
        fprintf('\n++++++++++++++++++++++++++++++++++++++++++++++++');
        fprintf('\nIteration number %d of %d -- SNR=%1.1f dB\n',...
        n,length(SNR),SNR(n));

        fclose('all');
        RX_main_function_PLUTO_noise(SNR(n),address,serial,Grx,Gtx);
        delete('file.txt');             % for change SNR value in
TX_main_recursive
        n = n + 1;

    else
        fprintf('\n++++++++++++++++++++++++++++++++++++++++++++++++');
        fprintf('\n\nWaiting for Transmission\n')
        fprintf('\n++++++++++++++++++++++++++++++++++++++++++++++++');
    end
end

%% Exit PROG
exitProg();
```

RX_main_function_PLUTO_noise.m

```
function [] = RX_main_function_PLUTO_noise(SNR,address,serial,Gain_RX,
Gain_TX)

%% TITLE:   ADALM-PLUTO Receiver recursive function script with AWGN
%
% Authors: Fernando Guiomar <guiomar@av.it.pt> and Bruno Maximino
% <brunomaximino@av.it.pt>
% Last Update: 23/04/2018

close all;
clc;

%% Load Libraries:
addpath('.\functions\');
addpath(genpath('..\..'));

%% Initialization:
global PROG;
initProg();
PROG.showMessagesLevel = 2;
instrreset
DIR.dataDir = 'C:\Users\Bruno Maximino\Desktop\TESE\Trabalhos MATLAB\
MINHAS\OptDSP\_examples\ADALM_PLUTO\data\AWGN\FINAL\';
suffix = 'attenuation=20dB';

%% Input Parameters:
M = 64;                                 % QAM constellation size
rollOff = 0.1;
symRate = 1e6;                          % total gross symbol-rate
OH.FEC = 0.2;                           % overhead for FEC (e.g. 0.2=20%)
OH.total = 0.28;                        % total signal overhead including
FEC and other overheads such as those needed for Ethernet/IP, DSP pilots,
etc
UpSampling = 2;
SamplesPerFrame = 2^17;

Fs_DAC = UpSampling*symRate;
Fs_ADC = UpSampling*symRate;

nSyms = symRate/Fs_ADC*SamplesPerFrame; % total number of simulated
symbols
bitRate_net = symRate*log2(M)/(1+OH.total);
nSamplesTotal = 1e6;

%% Get TX Parameters:
[TX,RX] = presetTXparams_ADALM_PLUTO(M,symRate,OH,nSyms,rollOff,Fs_DAC,
Fs_ADC);

%% SDRRxPluto Receive data from an ADALM-PLUTO radio:
% RX = sdrrx('Pluto') creates an SDR receiver System object, RX, that
% receives data from an ADALM-PLUTO radio.
% FrequencyCorrection - Frequency correction (ppm)
% Specify the frequency correction value in ppm as a double precision
scalar.
% The valid range is [-200, 200] ppm. The default value is 0, which
```

```
% corresponds to the factory-calibrated setting of the radio. This
% property value specifies the parts-per-million change to the baseband
% sample rate and the center frequency.

radioRx = comm.SDRRxPluto(...           % send data to ADALM-PLUTO device
and specify some parameters
    'RadioID',              address, ...
    'CenterFrequency',      0.7e9, ...
    'GainSource',           'Manual', ...
    'Gain',                 Gain_RX, ...% [-4;71] dB, default value =
10dB
    'BasebandSampleRate',   Fs_ADC, ...
    'OutputDataType',       'double', ...
    'SamplesPerFrame',      SamplesPerFrame, ...
    'ChannelMapping',       1, ...      % This property is read-only. The
channel mapping is always set to 1.
    'FrequencyCorrection',  0);

%% MSC Transmitter:
[S.txSC,S.tx,TX] = SC_transmitter(TX,0,3);

%% Receive Signal:
nBlocks = ceil(nSamplesTotal/SamplesPerFrame);% number of blocks received
n = 1;
len = 0;

while n <= nBlocks
    tic
    while len <= 0
        [Srx(:,n), len] = step(radioRx);
    end
    elapsedTime = toc;
    dataRate = numel(Srx(:,n))/elapsedTime
    len = 0;
    n = n + 1
end

%% View Spectrum of Received Signal
%scatterplot(Srx(1:2:end).')
%figure,
pwelch(reshape(Srx,1,numel(Srx)),1e4,[],[],Fs_DAC,'centered','psd');

%% Save Rx Signal:
[units,unitFactor] = symRate_setUnits(symRate);
fileName = ['Srx_AWGN_' num2str(M) 'QAM_' ...
    num2str(symRate*unitFactor,'%1.0f'),...
    units '_RO=',num2str(rollOff) ...
    '_txGain=' num2str(Gain_TX) 'dB' ...
    '_rxGain=' num2str(Gain_RX) 'dB_' suffix ...
    '_SNR=' num2str(SNR) 'dB']; %_fc=2.3GHz _fc=700MHz_FO=-2

fileName(fileName == '.') = ',';
filePath = [DIR.dataDir fileName];
save(filePath,'Srx','TX','RX')

end
```

Transmissão de dados através do ADALM-PLUTO

TX_main_recursive_PLUTO_noise.m

```
%% TITLE:   ADALM-PLUTO Transmitter recursive script with AWGN
%
% Authors: Fernando Guiomar <guiomar@av.it.pt> and Bruno Maximino
% <brunomaximino@av.it.pt>
% Last Update: 23/04/2018

clear;
clear global;
close all;
clear all;
clc;

%% Load Libraries:
addpath('.\functions\');                % Add directory to search path
addpath(genpath('..\..\'));

%% Initialization:
DIR.dataDir = 'C:\Users\Bruno Maximino\Desktop\TESE\Trabalhos MATLAB\
MINHAS\OptDSP\_examples\ADALM_PLUTO\data\AWGN\FINAL\';

%% Input parameters:
M = 64;                                 % Modulation format
Gtx = -10;                              % Transmitter Gain
Grx = 25;                               % Receiver Gain
rollOff = 0.1;
suffix = 'attenuation=20dB';
symRate = 1e6;

%% Connect to ADALM-PLUTO:
connectedRadios = findPlutoRadio;

if ~isempty(connectedRadios)            % check if there is any radio
connecetd to the PC
    address = connectedRadios(2).RadioID;
    serial = connectedRadios(2).SerialNum;
else
    error('Could not connect to any radio...')
end

% first run RX -> connectedRadios(2)

%% Run the script for every SNR values:
SNR = floor(ber2snr(1e-1,M)):0.5:ceil(ber2snr(1e-9,M));
n = 1;

while n <= length(SNR)

    target_SNR = SNR(n);
    TX_main_function_PLUTO_noise(target_SNR,address,serial,Gtx,rollOff);

    fprintf('\n++++++++++++++++++++++++++++++++++++++++++++++++++');
    fprintf('\n\nTransmit for SNR = %4.1f dB\n',target_SNR);
```

```
    fprintf('\n++++++++++++++++++++++++++++++++++++++++++++++++++');
    n = n + 1;
end

%% Exit PROG:
exitProg();
```

TX_main_function_PLUTO_noise.m

```
function [] = TX_main_function_PLUTO_noise(snr,address,serial,Gain_TX,
rollOff)

%% TITLE:   ADALM-PLUTO Transmitter recursive function with AWGN
%
% Authors: Fernando Guiomar <guiomar@av.it.pt> and Bruno Maximino
% <brunomaximino@av.it.pt>
% Last Update: 23/04/2018

close all;
clc;

%% Load Libraries:
addpath('.\functions\');                % Add directory to search path
addpath(genpath('..\..\'));

%% Initialization:
global PROG;
initProg();                             % Program initialization
PROG.showMessagesLevel = 2;
instrreset
DIR.dataDir = 'C:\Users\Bruno Maximino\Desktop\TESE\Trabalhos MATLAB\
MINHAS\OptDSP\_examples\ADALM_PLUTO\data\AWGN\FINAL\';

%% Input Parameters:
M = 64;                                 % QAM constellation size
symRate = 1e6;                          % total gross symbol-rate
OH.FEC = 0.2;                           % overhead for FEC (e.g. 0.2=20%)
OH.total = 0.28;                        % total signal overhead including
FEC and other overheads such as those needed for Ethernet/IP, DSP pilots,
etc
UpSampling = 2;
SamplesPerFrame = 2^17;

Fs_DAC = UpSampling*symRate;
Fs_ADC = UpSampling*symRate;
nSyms = symRate/Fs_ADC*SamplesPerFrame; % total number of simulated
symbols
bitRate_net = symRate*log2(M)/(1+OH.total); % net bit-rate

% SNR Parameters:
SNR.SNRout_dB = snr;                    % SNR target value tested
```

```
SNR.noiseSeed = 101;                    % fixed noise seed to guarantee
the same noise is generated between different runs of this m-file, thus
ensuring the same output results.
suffix = 'attenuation=20dB';

%% Get TX Parameters:
[TX,RX] = presetTXparams_ADALM_PLUTO(M,symRate,OH,nSyms,rollOff,Fs_DAC,
Fs_ADC);

%% MSC Transmitter:
[S.txSC,S.tx,TX] = SC_transmitter(TX,0,3);
S.txSC = setSNR(S.txSC,SNR,TX.PARAM,TX.SIG); % Signal with AWGN

%% Prepare Tx Signal:
Stx = S.txSC.'./max(abs(S.txSC)) * 1;    % Signal normalization factor

%% View Tx Signal:
%scatterplot(Stx(1:2:end).')
%figure,
pwelch(double(Stx.'),1e4,[],[],TX.DAC.RESAMP.sampRate,'centered','psd');

%% Configure ADALM-PLUTO:
% configurePlutoRadio(CHIPSET) configures the ADALM-PLUTO radio to
% operate in the specified CHIPSET mode. The radio must be connected to
% computer running this command. CHIPSET must be one of 'AD9364' or
% 'AD9363'.
% If CHIPSET is set to 'AD9363', which is factory default value, LO
% tuning range is between 325 MHz and 3.8 GHz with a maximum bandwidth of
% 20 MHz. If CHIPSET is set to 'AD9364', LO tuning range is between 70
MHz
% and 6 GHz with a maximum bandwidth of 56 MHz.

%configurePlutoRadio('AD9363');
%configurePlutoRadio('AD9364');

%% SDRTxPluto Transmit data to an ADALM-PLUTO radio:
% TX = sdrtx('Pluto') creates an SDR transmitter System object, TX, that
% transmits data to an ADALM-PLUTO radio.  Although the ADALM-PLUTO
% transmitter System object sends data to an ADALM-PLUTO radio, the
object
% acts as a sink in MATLAB.

% FrequencyCorrection - Frequency correction (ppm)
% Specify the frequency correction value in ppm as a double precision
scalar.
% The valid range is [-200, 200] ppm. The default value is 0, which
% corresponds to the factory-calibrated setting of the radio. This
% property value specifies the parts-per-million change to the baseband
% sample rate and the center frequency.

radioTx = comm.SDRTxPluto(...            % send data to ADALM-PLUTO
device and specify some parameters
    'RadioID',              'usb:1', ...
    'CenterFrequency',      0.7e9, ...
    'Gain',                 -10, ...     % [-89.75;0] dB
    'BasebandSampleRate',   2000000, ...
```

```
    'ChannelMapping',       1, ...       % This property is read-only.
The channel mapping is always set to 1
    'FrequencyCorrection',  -2);         % [-200;200] ppm

%% Transmit Signal:
n = 0;
FID = fopen('file.txt','wt');
fclose(FID);

while FID > 0            % if 'file.txt' not exist, fopen returns -1
    tic
    step(radioTx, Stx);
    elapsedTime = toc;
    dataRate = numel(Stx)/elapsedTime
    n = n + 1
    FID = fopen('file.txt');
    fclose('all');

end

% Save TX Signal:
[units,unitFactor] = symRate_setUnits(symRate);
fileName = ['Stx_AWGN_' num2str(M) 'QAM_' ...
    num2str(symRate*unitFactor,'%1.0f') units ...
    '_RO=' num2str(rollOff) ...
    '_txGain=' num2str(Gain_TX) 'dB_' suffix];

fileName(fileName == '.') = ',';
filePath = [DIR.dataDir fileName];
save(filePath,'S','radioTx','TX')

end
```

Processamento de sinal

rxDSP_offline_sweepSNR.m

```
%% TITLE:   ADALM-PLUTO DSP for Sweep SNR
%
% Authors: Fernando Guiomar <guiomar@av.it.pt> and Bruno Maximino
% <brunomaximino@av.it.pt>
% Last Update: 04/05/2018

clear;
clear global;
close all;
clc;

%% Load Libraries:
addpath('.\functions\');
addpath(genpath('..\..'));
addpath(genpath('..\..\..\export_fig'));
```

```
%% Initialization:
global PROG;
initProg();
PROG.showMessagesLevel = -1;

% Set data directory:
DIR.dataDir = 'C:\Users\Bruno Maximino\Desktop\TESE\Trabalhos MATLAB\
MINHAS\OptDSP\_examples\ADALM_PLUTO\data\AWGN\FINAL\';
suffix = 'attenuation=20dB';

% Define colors:
red = [0.73 0.07 0.169];
blue = [0.0 0.24 0.431];

%% Input Parameters:
SNR_dB = 29:-0.5:12;
M = 64;                                     % QAM constellation size
symRate = 1e6;                              % total gross symbol-rate
rollOff = 0.1;
useDiffQuadEnc = false;

if(M==4 || M==8 || M==16 || M==32 || M==64 || M==128)
    targetBER = [1e-4 1e-3 1e-2];
elseif(M==256)
        targetBER = [1e-3 1e-2];
else
        targetBER = [1e-2];
end

Gtx = -10;
Grx = 25;
att = 20;
FO = -2;

DSP_option = 'fully data-aided';
% DSP_option = 'DA->DD';
% DSP_option = 'fully blind';

%% Load Tx Signal:
[units,unitFactor] = symRate_setUnits(symRate);
TxFileName = ['Stx_AWGN_' num2str(M) 'QAM_'
num2str(symRate*unitFactor,'%1.0f')...
    units '_RO=' num2str(rollOff) '_txGain=' num2str(Gtx) 'dB_' suffix
'_fc=700MHz_FO=' num2str(FO)];

TxFileName(TxFileName == '.') = ',';
load([DIR.dataDir TxFileName]);
Stx_syms = S.tx;

if useDiffQuadEnc
    TX.SIG.encoding = 'diff-quad';
    TX.QAM = QAM_config(TX.SIG);
end

%% Define DSP Parameters:
DSP = presetDSP_ADALM_PLUTO(TX.SIG,DSP_option,'.mex'); % .mex -> faster
```

```
%% Test All SNRs:
nIter = numel(SNR_dB);
[BER,SER,EVM,MI] = deal(NaN(1,nIter));

for n = 1:nIter
    % Load Rx Signal:
    RxFileName = ['Srx_AWGN_' num2str(M) 'QAM_' ...
        num2str(symRate*unitFactor,'%1.0f'),...
        units '_RO=',num2str(rollOff) '_txGain=' ...
        num2str(Gtx) 'dB_rxGain=' num2str(Grx) 'dB_' suffix, ...
        '_SNR=' num2str(SNR_dB(n)) 'dB_fc=700MHz_FO=' num2str(FO)];
    RxFileName(RxFileName == '.') = ',';
    RxFilePath = [DIR.dataDir RxFileName '.mat'];

    try
        TMP = load(RxFilePath);
        Srx = reshape(TMP.Srx,1,numel(TMP.Srx));

        ti1 = tic;
        fprintf('\n+++++++++++++++++++++++++++++++++++++++++++++++++');
        fprintf('\nIteration number %d of %d -- SNR=%1.1f dB',...
        n,nIter,SNR_dB(n));

        % Rx-DSP:
        [RES,Srx_afterDSP,~,~,DSP_out] = rxDSP_SC_OB2B_LAB(Srx,Stx_syms,
TX.PARAM,TX.SIG,TX.QAM,DSP);

        % Get Results:
        BER(n) = RES.BER.BERx;
        SER(n) = RES.SER.SERx;
        EVM(n) = RES.EVM.EVMx;
        MI(n) = RES.MI.MIx;
        fprintf('\n\tSNR: %1.3f dB',SNR_dB(n));
        fprintf('\n\tBER: %1.3e',BER(n));

        % Save Results:
        SaveRxFilePath = [DIR.dataDir RxFileName '.mat'];

        if isfield(RES.EVM,'EVMx_t')
            RES.EVM = rmfield(RES.EVM,'EVMx_t');
        end
        if useDiffQuadEnc
            RES_dqe = RES;
            save([SaveRxFilePath '_diffQuadEnc'],'RES_dqe','-append');
        else
            save(SaveRxFilePath,'DSP','RES','Srx_afterDSP','-append');
        end

        % Elapsed Time:
        fprintf('\nElapsed Time (Iteration 1): %1.4f [s]\n',toc(ti1));
        fprintf('+++++++++++++++++++++++++++++++++++++++++++++++++\n\n');
    end
end

%% Determine Theoretical BER Curves and Shannon Capacity:
SNRt_dB = 0:0.1:40;
```

```
for n = 1:numel(SNRt_dB)
    BERt(n) = snr2ber(SNRt_dB(n),TX.QAM);
    SERt(n) = snr2ser(SNRt_dB(n),TX.QAM);
    EVMt(n) = 10^(-SNRt_dB(n)/20)*100;
end

%% Estimate B2B Penalty:
for n = 1:numel(targetBER)
    reqSNR(n) = getRequiredSNR_fromBER(BER,SNR_dB,targetBER(n),1e-1);
    B2Bpen(n) = reqSNR(n)-ber2snr(targetBER(n),M);
end

%% Save Results (total array):
[units,unitFactor] = symRate_setUnits(symRate);
savedFile = [num2str(M) 'QAM_Gtx=' num2str(Gtx) 'dB_' ...
    'RO=' num2str(rollOff) '_Grx=' num2str(Grx) 'dB_symRate='
num2str(symRate*unitFactor,'%1.0f'), ...
    units '_' suffix '_fc=700MHz_FO=' num2str(FO)];
savedFile(savedFile == '.') = ',';

if useDiffQuadEnc
    ResFileName = ['results_dqe_' savedFile];
else
    ResFileName = ['results_' savedFile];
end

save([DIR.dataDir ResFileName '.mat'],'BER','SER','EVM',...
    'MI','SNR_dB','reqSNR','B2Bpen');

%% Plot BER vs SNR:
hFig = figure;
yMin = 1e-5;
yMax = 1e-1;
xMin = floor(ber2snr(yMax,M));
xMax = max(SNR_dB);
dx = xMax - xMin;

hPlot(1) = semilogy(SNRt_dB,BERt);
hold on;
hPlot(2) = semilogy(SNR_dB,BER);

% Plot Formatting:
set(hPlot(1),'color',blue,'marker','none','LineStyle','--');
set(hPlot(2),'color',red,'marker','o','LineStyle',':');
set(hPlot,'MarkerSize',6,'LineWidth',1.5,'MarkerFaceColor','w');

% Show Target BERs:
for n = 1:numel(targetBER)
    hLine(n) = line([0 100],[targetBER(n) targetBER(n)]);
    hText(n) = text(xMin+0.02*dx,targetBER(n),['B2B$_\mathrm{pen}=',...
        num2str(B2Bpen(n),'%1.2f') '$ dB']);
end
set(hLine,'linestyle','--','linewidth',0.5,'color','k');
set(hText,'HorizontalAlignment','left','VerticalAlignment','bottom',...
    'color',blue,'Interpreter','latex','fontsize',10);

% Axis Formatting:
```

```
xlabel('SNR [dB]','Interpreter','latex','FontSize',11);
ylabel('BER','Interpreter','latex','FontSize',11);
xAxis = get(gca,'xaxis');
set(xAxis,'TickLabelInterpreter','latex','FontSize',12);
yAxis = get(gca,'yaxis');
set(yAxis,'TickLabelInterpreter','latex','FontSize',12);
set(gca,'PlotBoxAspectRatio',[1 0.6 1],'Box','on');
set(gca,'yTick',10.^(-5:-1));
axis([xMin xMax yMin*0.999 yMax]);

% Title:
hTitle = title({['\textbf{' num2str(M) 'QAM}'],...
    [' $\beta=',num2str(rollOff),...
    '$, $\alpha=',num2str(att),'$ dB, $R_s=',...
    num2str(symRate*unitFactor,'%1.0f') '$ ',units],...
    ['$G_\mathrm{tx}=',num2str(Gtx),...
    '$ dB, $G_\mathrm{rx}=' num2str(Grx),'$ dB']});
set(hTitle,'Interpreter','latex','color',blue,'fontSize',11);

% Legend:
hLeg = legend('AWGN Theory','Experimental Results');
set(hLeg,'Interpreter','latex','fontSize',10,'Location','NorthEast');

% Gridlines:
grid on;
set(gca,'GridLineStyle','--');

% Save as .fig:
if useDiffQuadEnc
    figName = [savedFile '_BER_vs_SNR_dqe'];
else
    figName = [savedFile '_BER_vs_SNR'];
end

figFolder = [DIR.dataDir 'figures\'];
if ~exist(figFolder,'dir')
    mkdir(figFolder);
end
savefig(hFig,[figFolder figName]);

% Save as .pdf:
try
    export_fig([figFolder figName],'-pdf','-transparent');
end

%% Plot EVM vs SNR:
hFig = figure;
yMin = 20*log10(floor(min(EVM))/100);
yMax = 20*log10(ceil(10^(-ber2snr(1e-1,M)/20)*100)/100);
xMin = min(SNR_dB);
xMax = max(SNR_dB);
dx = xMax - xMin;

hPlot(1) = plot(SNRt_dB,20*log10(EVMt/100));
hold on;
hPlot(2) = plot(SNR_dB,20*log10(EVM/100));
```

```
% Plot Formatting:
set(hPlot(1),'color',blue,'marker','none','LineStyle','--');
set(hPlot(2),'color',red,'marker','o','LineStyle',':');
set(hPlot,'MarkerSize',6,'LineWidth',1.5,'MarkerFaceColor','w');

% Axis Formatting:
xlabel('SNR [dB]','Interpreter','latex','FontSize',11);
ylabel('EVM [dB]','Interpreter','latex','FontSize',11);
xAxis = get(gca,'xaxis');
set(xAxis,'TickLabelInterpreter','latex','FontSize',12);
yAxis = get(gca,'yaxis');
set(yAxis,'TickLabelInterpreter','latex','FontSize',12);
set(gca,'PlotBoxAspectRatio',[1 0.6 1],'Box','on');
axis([xMin xMax yMin*0.999 yMax]);

% Title:
hTitle = title({['\textbf{' num2str(M) 'QAM}'],...
    [' $\beta=',num2str(rollOff),...
    '$, $\alpha=',num2str(att),'$ dB, $R_s=',...
    num2str(symRate*unitFactor,'%1.0f') '$ ',units],...
    ['$G_\mathrm{tx}=',num2str(Gtx),...
    '$ dB, $G_\mathrm{rx}=' num2str(Grx),'$ dB']});
set(hTitle,'Interpreter','latex','color',blue,'fontSize',11);

% Legend:
hLeg = legend('AWGN Theory','Experimental Results');
set(hLeg,'Interpreter','latex','fontSize',10,'Location','NorthEast');

% Gridlines:
grid on;
set(gca,'GridLineStyle','--');

% Save as .fig:
if useDiffQuadEnc
    figName = [savedFile '_EVM_vs_SNR_dqe'];
else
    figName = [savedFile '_EVM_vs_SNR'];
end
figFolder = [DIR.dataDir 'figures\'];
if ~exist(figFolder,'dir')
    mkdir(figFolder);
end
savefig(hFig,[figFolder figName]);

% Save as .pdf:
try
    export_fig([figFolder figName],'-pdf','-transparent');
end

%% Exit PROG:
exitProg();
```

rxDSP_SC_OB2B_LAB.m

```
function [RES,Srx,SIG,PARAM,DSP] = rxDSP_SC_OB2B_LAB(Srx,Stx,PARAM,
SIG,QAM,DSP)

% Last Update: 19/02/2018

%% Entrance Message:
entranceMsg('DSP for receiver-side single-carrier processing',0)
tStart = tic;

%% Pre-DSP: Signal Preparation:
[Srx,PARAM] = preDSP_signalPreparation(Srx,PARAM,DSP);

%% Rx Front-End Correction:
[Srx,PARAM,DSP.DC] = RX_frontEndCorrection(Srx,PARAM,DSP);

%% Resampling and Filtering:
[Srx,PARAM,SIG] = SC_matchedFilter(Srx,PARAM,SIG,DSP);

%% 1st Frequency-Mismatch Compensation:
[Srx,DSP.FE1] = SC_freqEstimation(Srx,Stx,PARAM,SIG,DSP,1);

%% 1st Adaptive Equalization:
[Srx,PARAM,DSP.MIMO1] = SC_adaptiveEq(Srx,Stx,PARAM,SIG,QAM,DSP,1);

%% 2nd Frequency-Mismatch Compensation:
[Srx,DSP.FE2] = SC_freqEstimation(Srx,Stx,PARAM,SIG,DSP,2);

%% Carrier-Phase Estimation (1st stage):
[Srx,DSP.CPE1] = SC_phaseEstimation(Srx,Stx,SIG,QAM,DSP,1);

%% 2nd Adaptive Equalization:
[Srx,PARAM,DSP.MIMO2] = SC_adaptiveEq(Srx,Stx,PARAM,SIG,QAM,DSP,2);

%% Downsampling:
[Srx,PARAM,SIG] = SC_downSample(Srx,PARAM,SIG,1);

%% Carrier-Phase Estimation (2nd stage):
[Srx,DSP.CPE2] = SC_phaseEstimation(Srx,Stx,SIG,QAM,DSP,2);

%% Post-DSP Signal Preparation:
[Srx,PARAM] = SC_postDSP(Srx,PARAM,DSP);

%% Rx Decoder:
[DSP.DECODER,Stx] = SC_rxDECODER(Srx,Stx,QAM,DSP);

%% Performance Analyzer:
RES = SC_performanceAnalyzer(Srx,Stx,QAM,DSP);

%% Exit Messages:
myMessages(['\nDSP for Single-Carrier finished - Elapsed Time: ',...
    num2str(toc(tStart),'%2.4f\n'),' [s]'],0);
```

SC_transmitter.m

```
function [Stx_SC,Stx,TX,txSyms,txBits] = SC_transmitter(TX,Ptx_dBm,nRep)

% Last Update: 17/07/2017

%% Input Parameters:
SIG = TX.SIG;
QAM = TX.QAM;
PRBS_Params = TX.PRBS_Params;
PS = TX.PS;

if nargin < 3
    nRep = 1;
end
if ~mod(nRep,2)
    error('Number of periodic signal repetitions must be an even
number.');
end

%% Generate Transmitted Symbols, Bits and Parameters:
nSyms = SIG.nSyms;
if QAM.lambda
    [Stx,txSyms,txBits,TX.QAM] = Tx_ProbShaping(QAM,PRBS_Params,nSyms);
else
    [Stx,txSyms,txBits,TX.PRBS_Params] = Tx_QAM(QAM,PRBS_Params,nSyms);
end

%% Set Sampling-Rate of Single-Carrier Signal:
nSpS = 8;
Fs_SC = SIG.symRate * nSpS;
if isfield(TX,'DAC') && isfield(TX.DAC,'RESAMP')
    if TX.DAC.RESAMP.sampRate > Fs_SC
        nSpS = ceil(TX.DAC.RESAMP.sampRate/SIG.symRate);
        Fs_SC = nSpS * SIG.symRate;
    end
end

%% Replicate Transmitted Signal and Update Parameters:
nSamples = nSyms * nSpS * nRep;
PARAM = setSimulationParams(Fs_SC,nSamples);

if isfield(TX,'DAC') && isfield(TX.DAC,'RESAMP')
    nSpS_trunc = TX.DAC.RESAMP.sampRate / SIG.symRate;
else
    nSpS_trunc = nSpS;
end
ni = floor(nRep/2)*nSyms * nSpS_trunc + 1;
nf = ceil(nRep/2)*nSyms * nSpS_trunc;


%% Create SC Signal
% Pulse Shaping:
Stx_SC = pulseShaper(repmat(Stx,1,nRep),nSpS,PS);

% Normalize SC Power to 1:
```

```
% Stx_SC = Stx_SC./sqrt(mean(abs(Stx_SC).^2,2));                % for
R2016b
Stx_SC = Stx_SC./repmat(sqrt(mean(abs(Stx_SC).^2,2)),1,size(Stx_SC,2));
% only required for versions previous to R2016b

%% Digital Part of Transmitter
if isfield(TX,'DAC')
    [Stx_SC,TX.DAC,PARAM] = Tx_DAC(Stx_SC,TX.DAC,PARAM);
end

%% Truncate the Sequence
Stx_SC = Stx_SC(:,ni:nf);

%% Update Simulation and Signal Parameters:
TX.PARAM = setSimulationParams(PARAM.sampRate,length(Stx_SC));
SIG_out = setSignalParams(SIG,TX.PARAM);
TX.SIG = SIG_out;

end
```

Apêndice C – Resultados USRP-B210

Medição da potência de saída do USRP-B210

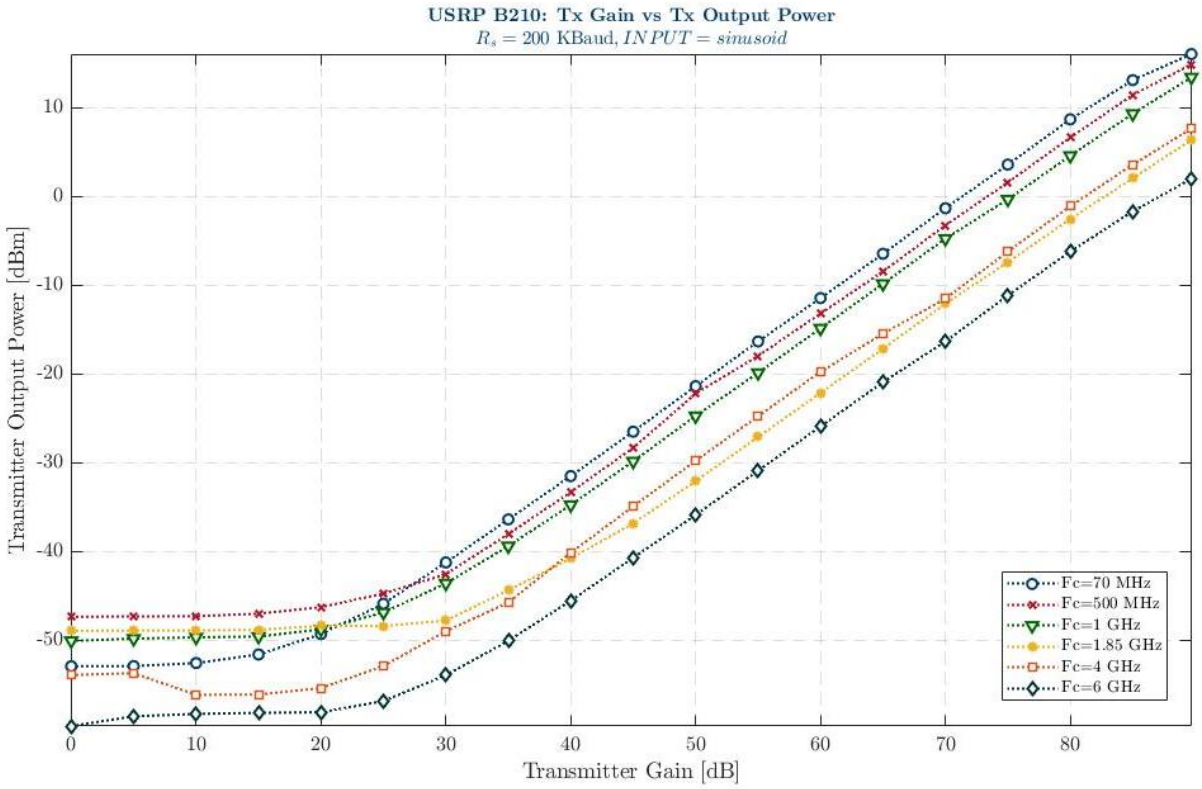


Figura C.1: Relação entre a potência de saída e o ganho do transmissor, para diferentes valores de frequência central.

Sabendo que a potência máxima que o recetor do USRP B210 ($P_{RX\ máx}$) suporta é -15 dBm [45] e que a potência máxima do transmissor ($P_{TX\ máx}$) é 16.03 dBm, para respeitar a Equação 1 da Secção 4.3, é necessário introduzir uma atenuação superior a 31.03 dB.

Tal como no ADALM-PLUTO, a potência de saída aumenta com o aumento do ganho do transmissor e diminui com o aumento da frequência central.

Apêndice D – Manual de Utilização

Este documento visa explicar, de uma forma simplificada, os passos necessários para a correta utilização do sistema implementado. O sistema, em MATLAB, é constituído por três blocos: o transmissor, o recetor e o bloco de processamento de sinal. O objetivo deste apêndice não é a explicação detalhada de cada ficheiro e função do sistema (Capítulo 5), mas sim uma breve explicação de como o utilizar corretamente. Os ficheiros MATLAB aqui abordados encontram-se no Apêndice B.

O transmissor é constituído por dois ficheiros MATLAB: *TX_main_recursive_PLUTO_noise.m* e *TX_main_function_PLUTO_noise.m.*

O recetor é constituído pelos ficheiros *RX_main_recursive_PLUTO_noise.m* e *RX_main_function _PLUTO_noise.m*.

Finalmente, o processamento de sinal é realizado pelo ficheiro *rxDSP_offline_sweepSNR.m*.

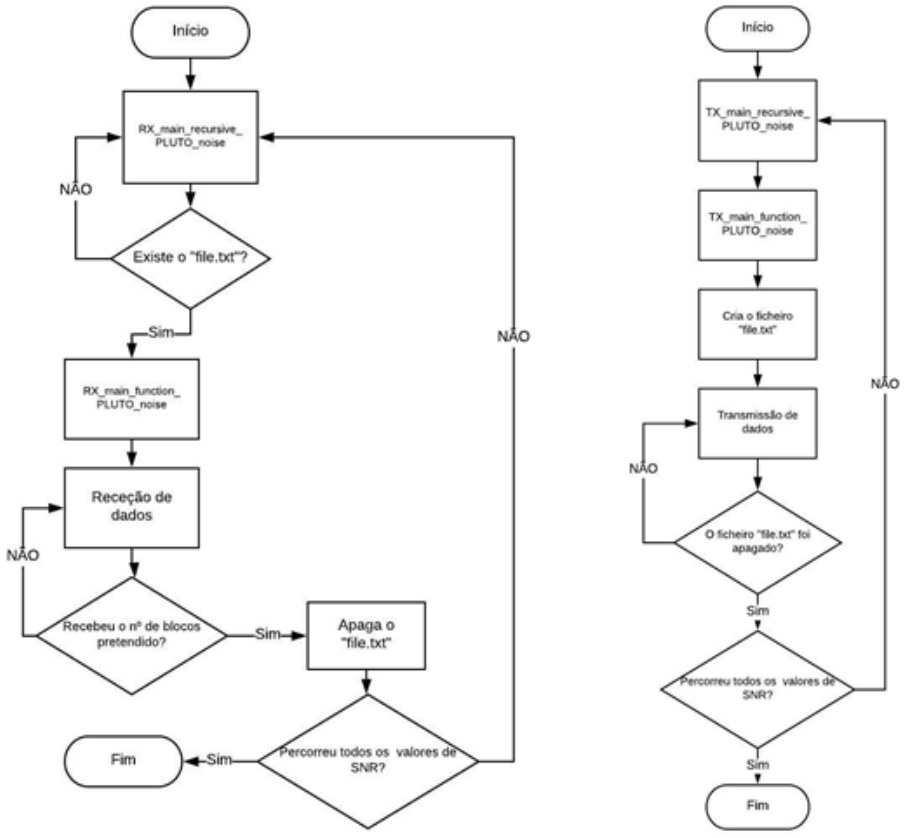


Figura D.1: Fluxograma do código MATLAB do recetor (esquerda) e do transmissor (direita).

Devido à necessidade de utilização de duas instâncias do MATLAB, uma para o transmissor e outra para o recetor, foi fundamental arranjar uma estratégia que permitisse que as duas instâncias comunicassem entre si. A estratégia utilizada foi a criação de um ficheiro de texto (*file.txt*), que funcionasse como uma *flag*, permitindo que o recetor só fosse acionado quando o transmissor estivesse a funcionar.

Na Figura D.1 estão representados os diagramas de fluxo do código do transmissor e do recetor. Em primeiro lugar, coloca-se a correr o ficheiro principal do recetor (*RX_main_recursive_PLUTO_noise.m*). Enquanto o ficheiro de texto (file.txt) não existir, imprime uma mensagem a dizer "Waiting for Transmission". Seguidamente, coloca-se o ficheiro principal do transmissor (*TX_main_recursive_PLUTO_noise.m*) a correr. Esse ficheiro vai chamar a função *TX_main_function_PLUTO_noise.m* que, por sua vez, vai criar o ficheiro de texto. Depois de criado o ficheiro, a transmissão de dados é iniciada. Do lado do recetor, quando é detetada a existência do ficheiro de texto, passamos para a função *RX_main_function_PLUTO_noise.m*, que vai ser responsável pela receção dos dados. Quando é recebido o número de blocos pretendidos, o recetor apaga o ficheiro *file.txt*, dando por terminada a transmissão e receção de dados para um valor de SNR desejado. Este ciclo vai ser repetido para todos os valores de SNR, introduzidos como parâmetro.

Após o término do sistema transmissão-receção, é necessário realizar o processamento de sinal dos dados obtidos (*rxDSP_offline_sweepSNR.m*) para avaliar o desempenho do sistema utilizado.

Depois de seguir o manual de instalação do Sistema MATLAB-PLUTO (Apêndice A), é necessário seguir alguns passos para uma correta utilização do mesmo.

Passo 1: Conectar o cabo SMA + atenuador entre a porta TX de um ADALM-PLUTO e a porta RX de outro.

Passo 2: Ligar os dois ADALM-PLUTO ao computador e abrir uma instância do MATLAB para o transmissor e outra para o recetor.

Passo 3: **Antes de começar a utilizar os ficheiros MATLAB disponibilizados, é necessário ter em atenção alguns pormenores.**

No ficheiro *RX_main_recursive_PLUTO_noise.m* os parâmetros que deverão ser alterados são:

- Formato de modulação: "M";
- Ganho do transmissor e do recetor: "Gtx" e "Grx";
- Quando é executada a instrução "*connectedRadios = findPlutoRadio*" vai ser devolvida uma estrutura (*serial number* + *RadioID*) com os dois rádios que se encontram conectados. É imperativo que, quando se especificar o parâmetro *RadioID*, na instrução "*radioRx = comm.SDRRxPLUTO()"* do ficheiro *RX_main_function_PLUTO_noise.m*, se tenha a certeza que esse *serial number* ou o *RadioID* corresponde ao ADALM-PLUTO que se encontra montado como recetor. A mesma verificação é necessária para os ficheiros do transmissor;
- Valor de SNR que se pretende testar: "SNR".

No ficheiro *RX_main_function_PLUTO_noise.m* os parâmetros que deverão ser alterados são:

- Diretório de gravação dos dados obtidos: "DIR.dataDir";
- Formato de modulação: "M";
- Fator de decaimento (*roll-off*): "rollOff";
- Taxa de símbolo: "symRate";
- Fator de interpolação: "UpSampling";
- Número de amostras por trama: "SamplesPerFrame";
- Número total de amostras: "nSamplesTotal";
- Frequência pretendida: "Center Frequency".

No ficheiro *TX_main_recursive_PLUTO_noise.m* os parâmetros que deverão ser alterados são:

- Diretório de gravação dos dados obtidos: “DIR.dataDir”;
- Formato de modulação: “M”;
- Ganho do transmissor e do recetor: “Gtx” e “Grx”;
- Fator de decaimento (*roll-off*): “rollOff”;
- Tal como nos ficheiros do recetor, é necessário a verificação do *serial number* ou *RadioID* que se especifica no “*comm.SDRTxPLUTO()”.*
- Valor de SNR que se pretende testar: “SNR”.

No ficheiro *TX_main_function_PLUTO_noise.m* os parâmetros que deverão ser alterados são:

- Diretório de gravação dos dados obtidos: “DIR.dataDir”;
- Formato de modulação: “M”;
- Taxa de símbolo: “symRate”;
- Fator de interpolação: “UpSampling”;
- Número de amostras por trama: “SamplesPerFrame”;
- Frequência pretendida: “Center Frequency”.

Finalmente, no ficheiro *rxDSP_offline_sweepSNR.m* os parâmetros que deverão ser alterados são:

- Diretório de gravação dos dados obtidos: “DIR.dataDir”;
- Valor de SNR que se pretende testar: “SNR_dB”.
- Formato de modulação: “M”;

- Taxa de símbolo: “symRate”;
- Fator de decaimento (*roll-off*): “rollOff”;
- Ganho do transmissor e do recetor: “Gtx” e “Grx”;
- Atenuação utilizada: “att”;
- Desvio de frequência (*offset*): “FO”.

Todas as alterações deverão respeitar a gama de valores de cada parâmetro, presente na Secção 3.4.

Passo 4: Colocar o ficheiro *RX_main_recursive_PLUTO_noise.m* a correr.

Passo 5: Colocar o ficheiro *TX_main_recursive_PLUTO_noise.m* a correr.

Passo 6: Depois de acabar a transmissão e receção de dados, falta colocar o ficheiro *rxDSP_offline_sweepSNR.m* a correr, para avaliar o desempenho do sistema.

Apêndice E – Outros resultados obtidos

Desempenho do sistema para os diferentes modos de funcionamento do ADALM-PLUTO

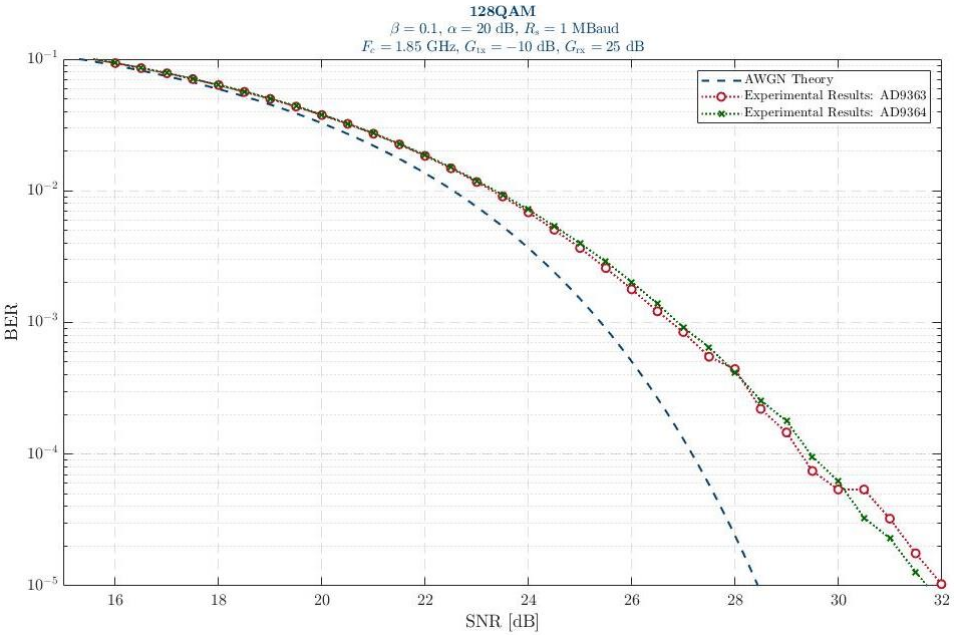


Figura E.1: Comparação do desempenho do sistema para os dois modos de funcionamento.

Após a otimização dos parâmetros do sistema, foi realizado este teste com o objetivo de verificar se o modo de funcionamento do ADALM-PLUTO tem influência no desempenho do sistema. O formato de modulação utilizado foi o 128QAM. Tal como foi dito na Secção 3.4, este rádio pode operar em dois modos: o AD9363 e o AD9364. Da figura anterior pode-se concluir que as diferenças no desempenho do sistema para os dois modos de funcionamento são insignificantes. Portanto, como o transcetor presente no ADALM-PLUTO é o AD9363, é esse modo que vai ser utilizado ao longo da dissertação.

Desempenho do sistema para os diferentes métodos de introdução de ruído

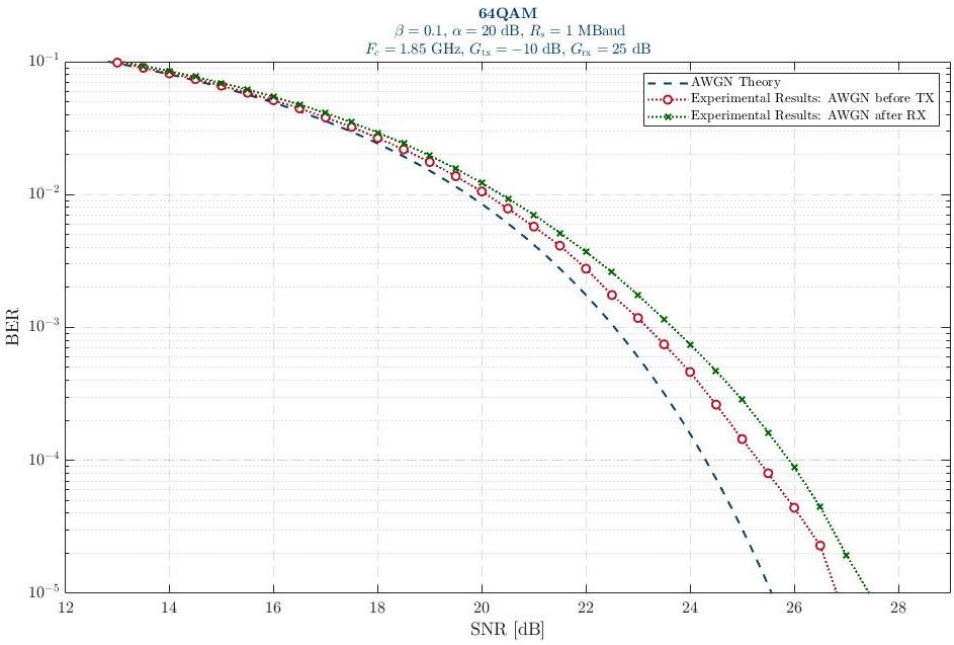


Figura E.2: Comparação do desempenho do sistema para os dois métodos de introdução de ruído.

Depois de realizada a otimização dos parâmetros do sistema, foi realizado um último teste com o objetivo de validar o método de introdução de ruído utilizado até então. Na impossibilidade de introduzir o ruído no caminho entre o transmissor e o recetor, foi introduzido antes da transmissão do sinal (Secção 5.2). Este teste, realizado recorrendo ao 64QAM, tem o objetivo de comparar o desempenho do sistema quando o ruído é introduzido antes da transmissão e quando é introduzido depois da receção. Na figura anterior é possível verificar que o sistema apresenta um melhor desempenho quando o ruído é introduzido antes da transmissão do sinal. Este é o método que foi utilizado ao longo da dissertação.