Universidade Federal do Rio Grande do Sul
Escola de Engenharia
Departamento de Engenharia Elétrica

Natanael Rissi Bertamoni

Avaliação do esquema de modulação LoRa implementado em GNURadio e sintetizado em SDR

Porto Alegre
2021

Natanael Rissi Bertamoni

Avaliação do esquema de modulação LoRa implementado em GNURadio e sintetizado em SDR

Evaluation of LoRa modulation scheme implemented in GNURadio and synthesized in SDR

Projeto de Diplomação apresentado ao Departamento de Engenharia Elétrica da Escola de Engenharia da Universidade Federal do Rio Grande do Sul como requisito parcial para Graduação em Engenharia Elétrica.

Orientador Prof. Dr. Ivan Müller

Porto Alegre
2021

Natanael Rissi Bertamoni

Avaliação do esquema de modulação LoRa implementado em GNURadio e sintetizado em SDR

Projeto de Diplomação apresentado ao Departamento de Engenharia Elétrica da Escola de Engenharia da Universidade Federal do Rio Grande do Sul como requisito parcial para Graduação em Engenharia Elétrica.

Porto Alegre, 13 de dezembro de 2021

Prof. Dr. Ivan Müller
UFRGS
Orientador

Me. Max Feldman
UFRGS

Prof. Dr. Paulo Francisco Butzen
UFRGS

Porto Alegre
2021

"If you constantly unpack everything for understanding you will never get anything done, and if do not unpack when you need to, you will do the wrong thing."

James B. Keller

RESUMO

Os sistemas de comunicação LPWA, (do inglês, *Low Power Wide Area*), buscam atender a necessidade de transmissão de dados em longas distâncias com baixo consumo energético e com tolerância a altos valores de latência. O esquema de modulação LoRa vem ao encontro à essa necessidade e busca transmitir símbolos através de uma técnica de espalhamento de espectro utilizando modulação M-ária para faixas de frequências não licenciadas. Pelas características intrínsecas dos sistemas de grandes áreas de abrangência, quando não se emprega técnica de prevenção de colisão de pacotes, é comum que hajam colisões. Uma característica da modulação LoRa é a capacidade de autocoexistência por ortogonalidade de portadoras moduladas por diferentes transmissores, sincronizados ou não. Dessa forma, como as colisões podem gerar perda de informação, gasto energético adicional pela retransmissão de dados ou *jamming* não proposital, buscou-se realizar simulações que avaliam se a ortogonalidade gerada por diferentes fatores de espalhamento garantem que não haja interferência inter símbolos. Para isso, utilizando o GNU Radio com blocos disponibilizados pelo Laboratório de Telecomunicações da EPFL, que mapeiam todas operações em bit que a patente do LoRa usada como base menciona, bem como a modulação e demodulação, buscou-se simular e explorar trabalhos comparativos que pudessem fornecer referência para os resultados encontrados. O ajuste do software GNU Radio, o procedimento de coleta de dados de simulação, e o script para geração de gráficos desenvolvidos, serve como base para avaliações futuras deste protocolo de IoT. Resultados preliminares indicaram discrepância da simulação com o esperado na literatura, revelando a necessidade de aprofundar o entendimento dos blocos utilizados na simulação.

Palavras-chave: Rádio Definido por Software, GNURadio, Modulação LoRa, IoT.

ABSTRACT

Low Power Wide Area communication systems seek to meet the need for data transmission over long distances with low energy consumption and tolerance to high latency values. The LoRa modulation scheme meets this need and seeks to transmit symbols through a spread spectrum technique using M-ary modulation for unlicensed frequency bands. Due to the intrinsic characteristics of large coverage systems, when there is no package collision prevention system, it inevitably leads to collisions. A feature of LoRa modulation is the ability of self-coexistence by orthogonality of carriers modulated by different transmitters, synchronized or not. So, collisions can generate information loss, additional energy expenditure by data retransmission or unintentional jamming, we sought to carry out simulations that assess whether the orthogonality generated by different sampling factors ensure that there is no inter-symbol interference. For this, using GNU Radio with blocks made available by the EPFL Telecommunications Laboratory, which map all bit operations that the patent usade as base mentions, as well as modulation and demodulation, we sought to simulate and explore comparative works that could provide reference to the results found. The adjustmente of the GNURadio software, the simulation data collection procedure and the graph generation script developed are the basis for the future of this IOT protocol. Preliminary results indicated a discrepancy between the simulation and what was expected int the literature, revealing the need to deepen the knowledge of the blocks used in the simulation.

Keywords: Software Defined Radio, GNURadio, LoRa Modulation, IoT.

LISTA DE ILUSTRAÇÕES

Figura 1 – Modulacão 17
Figura 2 – Continuidade de Fase 17
Figura 3 – *Downchirp* 18
Figura 4 – *Upchirp* 19
Figura 5 – Símbolo 300 codificado 19
Figura 6 – Classes 21
Figura 7 – Janelas de Transmissão 22
Figura 8 – Pacote 23
Figura 9 – *Payload* 24
Figura 10 – Cabeçalho MHDR 24
Figura 11 – Tipo de *Frame* 24
Figura 12 – *FCtrl Downlink* 25
Figura 13 – *FCtrl Uplink* 25
Figura 14 – Codificação e decodificação em bit 25
Figura 15 – Sequência usada no branqueamento 26
Figura 16 – Sequência de Bits do código de Hamming 27

Figura 17 – Exemplo de aplicação contendo transmissão e recepção 30
Figura 18 – Avaliação Inicial 32
Figura 19 – Amplitude Modulação 33
Figura 20 – SF de interesse = 6 34
Figura 21 – SF de interesse = 9 35
Figura 22 – SF de interesse = 9 36
Figura 23 – Primeiro pacote errado 37
Figura 24 – SF de interesse = 6 38
Figura 25 – SF de interesse = 6 39
Figura 26 – SF de interesse = 6 40
Figura 27 – SF de interesse = 9 41
Figura 28 – SNR 42

LISTA DE ABREVIATURAS E SIGLAS

AM	*Amplitude Modulation*
BLE	*Bluetooth Low Energy*
BPSK	*Binary Phase Shift Keying*
BW	*Bandwidth*
CFO	*Carrier Frequency Offset*
CSMA/CA	*Carrier-Sense Multiple Access with Collision Avoidance*
CSS	*Chirp Spread Spectrum*
DSP	*Digital Signal Processing*
DSSS	*Direct Sequence Spread Spectrum*
DTFT	*Discrete Time Fourier Transform*
EPFL	*École Polytechnique Fédérale de Lausanne*
FFT	*Fast Fourier Transform*
FHDR	*Frame Header*
FHSS	*Frenquency Hopping Spread Spectrum*
FM	*Frequency Modulation*
FSK	*Frequency Shift Keying*
GPL	*General Public License*
IoT	*Internet of Things*
IP	*Internet Protocol*
ISM	*Industrial Scientific and Medical*
LoRa	*Long Range*
LoRaWAN	*Long Range Wide Area Network*
LPWAN	*Low Power Wide Area Network*
MAC	*Media Access Control*
MHDR	*Media Access Control Header*
MIC	*Message Integrity Code*
OOT	*Out of Tree*
PAM	*Phase Amplitude Modulation*
PAN	*Personal Area Network*

QAM	*Quadrature Amplitude Modulation*
RFU	*Reserved for Future Use*
SF	*Spreading Factor*
SIR	*Signal to Interference Ratio*
SNR	*Signal to Noise Ratio*
SoC	*System on Chip*
STO	*Sampling Time Offset*
USRP	*Universal Software Radio Peripheral*
VLSI	*Very Large Scale Integration*
WSN	*Wireless Sensor Network*

SUMÁRIO

1 INTRODUÇÃO 12

2 FUNDAMENTAÇÃO TEÓRICA 15
2.1 Modulação e Demodulação LoRa 16
2.2 Protocolo de Comunicação LoRaWAN 20
2.2.1 Classes do Protocolo LoRaWAN 21
2.2.2 Camada de Enlace 22
2.3 Esquemas de Codificação e Decodificação 25
2.3.1 Branqueamento 25
2.3.2 Código de Hamming 26
2.3.3 Interleaving 27
2.3.4 Codificação Gray 28

3 DESENVOLVIMENTO DO TRABALHO 29

4 CONCLUSÕES 44

REFERÊNCIAS 46

APÊNDICE A – MODULAÇÃO 48

APÊNDICE B – CORREÇÃO CÓDIGO DE INSTALAÇÃO 51

APÊNDICE C – ERRO DE SIMULAÇÃO 52

1 INTRODUÇÃO

O conceito IoT, do inglês *Internet of Things*, trata da troca de informações através da Internet por uma vasta gama de objetos que podem ter sensores embutidos e recebem informações do ambiente no qual estão instalados, transmitindo para um *gateway*. Admitindo a vasta gama de sensores existentes, tais como sensores vestíveis, aplicações automobilísticas, agricultura de precisão e construções civis (BERNIER; DEHMAS; DEPARIS, 2020), a Internet das coisas ou IoT, é um tópico de grande interesse onde a comunicação se torna relevante. Visto que a IoT tem por objetivo dar conectividade aos objetos e ainda não há um protocolo de comunicação dominante, esse trabalho se insere nesse contexto e aproveita para explorar as capacidades físicas do esquema de modulação LoRa .

LoRa é uma especificação de camada física de comunicação sem fio, sendo propriedade da empresa Semtech. Ela é utilizada no protocolo de comunicação LoRaWan e usa técnica de modulação *chirp*, de espalhamento espectral, que tem o objetivo de explorar uma grande variabilidade de aplicações e o gerenciamento de quantidades massivas de objetos, sendo que esses podem estar conectados a um mesmo gateway em grandes distâncias.

Apesar da camada física ser proprietária, o protocolo de comunicação LoRaWAN é aberto e é um protocolo voltado ao baixo consumo de energia e grandes áreas de abrangência de rede, por isso, se enquadra na classe LPWAN, do inglês*Low Power Wide Area Network*. Ele foi desenvolvido para conectar dispositivos alimentados por bateria à Internet utilizando o espectro de radio frequência não licenciado ISM, do inglês*Industrial Scientific and Medical*, (TAPPAREL et al., 2020).

Outros esquemas de modulação, além do LoRa, podem ser usados em IoT e, analisando as características particulares de alguns dos esquemas de modulação existentes, pode-se escolher o que melhor se adequa à utilização da banda ISM em longas distâncias. Nesse sentido, diversos protocolos de comunicação digital sem fio já foram desenvolvidos para lidar com as demandas da aplicação IoT, tais como BLE, do inglês Bluetooth Low Energy, Zigbee e Wi-Fi Halow.

Kalaa et al. (2016) esclarecem que o *Bluetooth*, IEEE 802.15.1, é protocolo de rede sem fio pessoal que utiliza um canal de 1MHz em 2,4 GHz, com modulação FHSS, do inglês *Frequency Hopping Spread Spectrum* e taxa de transmissão de dados máxima de 1Mb/s. Menciona também que as redes convencionais *Bluetooth* podem se conectar com até oito nós, sendo 7 escravos e 1 mestre. Outro tipo de rede *Bluetooth* existente é o BLE, utilizado para transmitir menores taxas de dados, por isso, apresenta consumo energético menor. Da mesma forma que o Bluetooth clássico, o BLE utiliza a banda ISM em 40 canais, dos quais três são nomeados como canais de aviso e são usados para estabelecer conexão, e os 37 canais restantes são usados para troca de informação Kalaa et al. (2016).

Com o avanço dos sistemas sem fio e dos SoC, do inglês *system on chip*, um grande número de dispositivos para redes sem fio baseados em Wi-Fi foram desenvolvidos para aplicações em

sistemas de baixa potência, Noreen, Bounceur e Clavier (2017). O padrão IEEE 802.11 fornece as especificações para a camada MAC, do inglês *Media Acces Control*, e a camada física para a implementação nas frequências de 900 MHz, 2,4 GHz 3,6 GHz, 5 GHz e 60 GHz. A operação em 2,4 GHz é a mais comumente encontrada e pode acomodar até 14 canais distintos.

O ZigBee utiliza o padrão IEEE 802.15.4 que define as especificações da camada física e enlace. O IEEE 802.15.X abrange redes de áreas ditas PAN, *Personal Area Network* . Segundo Committee (2020), pode-se utilizar as topologias estrela e ponto-a-ponto, onde cada dispositivo tem um endereçamento único. Para cada sub-área que compõe um conjunto de dispositivos próximos, em ambas as topologias há um coordenador de rede. Na topologia estrela só é possível que os nós se conectem através do coordenador, mas na topologia ponto-a-ponto os nós são livres para conectarem entre si no seu raio de enlace. No enlace, utiliza-se o protocolo CSMA/CA, do inglês *Carrier-sense Multiple Acces with Collision Avoidance*, nos quais os nós competem uns com os outros para acessar a rede, sendo que em teoria mais de 64 mil nós podem ser conectados. O ZigBee usa a técnica de modulação DSSS, do inglês *Direct Sequence Spread Spectrum*, com taxa de transmissão máxima de 250 Kbps em 2,4 GHz com resistência ao *Jamming*.

Nesse trabalho, foram descritas a modulação e a demodulação LoRa, bem como a parte do protocolo de comunicação LoRaWAN que detalha quais são as partes que compõem um pacote, desde o preâmbulo sincronizador até o verificador de integridade. Objetivando avaliar o desempenho da demodulação e decodificação do pacote quando interferido por sinais ortogonais quanto ao fator de espalhamento, foi realizada a simulação completa do esquema de modulação LoRa utilizando as operações em bit descritas na patente Seller e Sornin (2013), desde a transmissão até a recepção.

Para validar os resultados da simulação de ortogonalidade, buscou-se por trabalhos similares para usar como referência e suporte. Assim, foram utilizados os blocos feitos para GNURadio e disponibilizados pelo laboratório de telecomunicações da EPFL, *Ecole Polytechnique Fédérale de Lausanne*, e artigos com resultados para usar como comparativos.

O software GNURadio é uma plataforma aberta e gratuita para simulação e síntese por SDR, *Software Defined Radio*, que fornece ferramentas para processamento de sinais em blocos. O GNURadio pode ser instalado nos sistemas operacionais Windows e Linux. Foram necessárias mudanças nas instalações a fim de corrigir erros, e também foram usadas versões de softwares específicas para a correta instalação. As versões utilizadas nas instalações do sistema operacional, Pybombs, GNURadio, Python e as dependências necessárias foram devidamente documentadas. Com isso, foram realizadas simulações da transmissão de sinais de interesse com diferentes fatores de espalhamento e potência fixa, além da adição de sinais interferente com potências diversas. Também foram realizadas simulações a fim de comparar com os resultados da SIR, do inglês *Signal to Interference Ratio*, com ruído gaussiano interferente.

Este trabalho está organizado da seguinte forma: no capítulo 2 é feita fundamentação teórica, no capítulo 3 são mostrados os métodos e materiais utilizados e os resultados obtidos e o capítulo

4 apresenta conclusões e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Para sustentar esse trabalho de diplomação buscou-se referências com embasamento científico que abordassem os temas do trabalho de forma a construir uma linha de desenvolvimento adequada. Rappaport (2009) afirma que os sistemas de comunicação modernos utilizam técnicas de modulação digital e que com os avanços da integração de larga escala, VLSI do inglês *Very Large Scale Integration*, e do DSP, do inglês *Digital Signal Processing*, os sistemas de modulação digitais apresentam diversas características técnicas que o diferencia, tais como maior imunidade ao ruído, maior robustez aos problemas no canal, possibilidade de inserção de códigos de controle de erros de bit bem como técnicas complexas de criptografia.

Rappaport (2009) menciona que para se escolher um sistema de modulação digital, múltiplos fatores devem ser levados em consideração, entre eles, taxa de erro de bit em situações de baixa relação sinal ruído, SNR (*Signal to Noise Ratio*) imunidade aos problemas de multi caminhos, largura de banda utilizada, e dificuldades e o custos de implementação. Ressalta-se também a eficiência de protocolo de camada física, ou seja, a taxa de dados atingida em relação à largura de banda utilizada.

Para que se possa atender uma grande quantidade de usuários, Rappaport (2009) afirma que o custo e a complexidade do transceptor devem ser minimizados, sendo modulações de simples detecção economicamente favoráveis. Por outro lado, Bernier, Dehmas e Deparis (2020) afirmam que, enquanto a crescente popularidade dos produtos baseados em tecnologia LoRa é claramente devida à sua acessibilidade, ou seja, o baixo custo e simplicidade de implantação, o processamento do sinais necessário para recuperar sinais modulados LoRa é, pelo contrário, relativamente complexo. O LoRa emprega técnica de modulação chamada Chirp, de mudança de frequências, em que a informação é codificada por uma mudança de frequências em função dos bits a serem transmitidos. O receptor LoRa deve compensar deslocamentos de frequência e amostragem que podem ocorrer devido a referências não sincronizadas.

Rappaport (2009) também afirma que o desempenho do esquema de modulação é afetado por vários problemas de canal, entre eles, a interferência provocada por múltiplos usuários utilizando o mesmo ambiente, sensibilidade da detecção ao *jitter* causado por canais variáveis no tempo, características importantes a serem levadas em consideração na escolha de um esquema de modulação. Noreen, Bounceur e Clavier (2017) afirmam que o LoRa é um esquema de modulação LPWAN para aplicações IoT que tem permitido o uso de sensores sem fio conectados em larga escala, permitindo a evolução do conceito WSN, do inglês *Wireless Sensor Network*, com custo viável. Como a modulação LoRa usa a banda inteira do canal designado para transmitir o sinal, torna-se resiliente ao ruído, ao efeito doppler e ao desvanecimento.

Em uma rede de sensores sem fio conectados em larga escala, é característico que seja composta de muitos nós espalhados geograficamente distantes, o que causa diversos problemas de propagação do sinal. Noreen, Bounceur e Clavier (2017) afirmam que incluem-se nesses

problemas interferência entre sinais causada por outros transmissores, atenuação devido às grandes distâncias, bloqueio dos sinais causados por grandes obstáculos e recepção de múltiplas cópias do mesmo sinal que são causadas por reflexões.

Devido à escassez do espectro de rádio, não é possível haver comunicação sem que não hajam interferências entre diferentes sinais, consequentemente o receptor receberá sinais indesejados. Podendo estes sinais indesejados ser do mesmo protocolo ou de outros protocolos como por exemplo, os já citados Bluetooth, ZigBee, Wi-Fi e SigFox, Noreen, Bounceur e Clavier (2017).

2.1 MODULAÇÃO E DEMODULAÇÃO LORA

A modulação LoRa é uma das derivações do CSS, do inglês *Chirp Spread Spectrum*, , e utiliza todo o espectro designado para transmitir um símbolo. Utilizar todo o espetro para transmitir um símbolo traz importantes características, tais como alta imunidade ao ruído e a sinais de outras naturezas.

No artigo de Hou et al. (2020) cita-se que a modulação LoRa pode ser decodificada mesmo na presença de co-interferência e inter-interferência de fator de espalhamento. Segundo Elshabrawy e Rober (2018), e também mencionado em Hou et al. (2020) e Georgiou e Raza (2017), um sinal com co-interferência de fator de espalhamento pode ser decodificado se a potência for pelo menos 6dB a mais que qualquer outro sinal recebido. Já a detecção com interferência causada por sinais ortogonais devido a diferentes fatores de espalhamento, ou seja inter-interferentes, pode variar em de 16 até 36 dB, Goursaud e Gorce (2015), Georgiou e Raza (2017).

Cada símbolo da modulação LoRa tem fator de espalhamento SF pertencente ao conjunto {6, 7,8,9,10,11,12} bits e consiste preferencialmente de N = 2^{SF} chips, ou seja base 2 para facilitar a demodulação, Tapparel et al. (2020) eSeller e Sornin (2013). Em banda base, se um símbolo s pertence ao conjunto S sendo S o conjunto dos inteiros $\{0, 1, ...N-1\}$ que varre a a banda inteira β, o símbolo começando na frequência $(\frac{s\beta}{N} - \frac{\beta}{2})$ e sua frequência incrementa por $\frac{\beta}{N}$ em cada chip. Quando a frequencia $\frac{\beta}{2}$ é alcançada, a frequência passa a ser positiva e a frequência continua a aumentar em passos de $\frac{\beta}{N}$ até que o módulo da frequência inicial é alcançada, Tapparel et al. (2020).

Segundo Tapparel et al. (2020), a descrição de um sinal LoRa pode ser feita de duas maneiras. A primeira não garante continuidade de fase entre símbolos, visto que a fase inicial depende somente do símbolo em si e por isso não será descrita, pois segundo Centenaro (2016) e a patente Seller e Sornin (2013), as transições de símbolos são feitas com continuidade de fase entre símbolos. A segunda forma, que é citada em Tapparel et al. (2020) e usada em Ghanaatian et al. (2019), Afisiadis et al. (2020), Elshabrawy et al. (2019), Chiani e Elzanaty (2019), a fase permanece continua entre transições de símbolos consecutivos. Essa continuidade é uma das vantagens citadas na patente Seller e Sornin (2013) conforme pode-se verificar na Figura 2, onde é feita a representação no domínio do tempo.

Então, usando a forma que garante continuidade de fase, a representação em banda base de

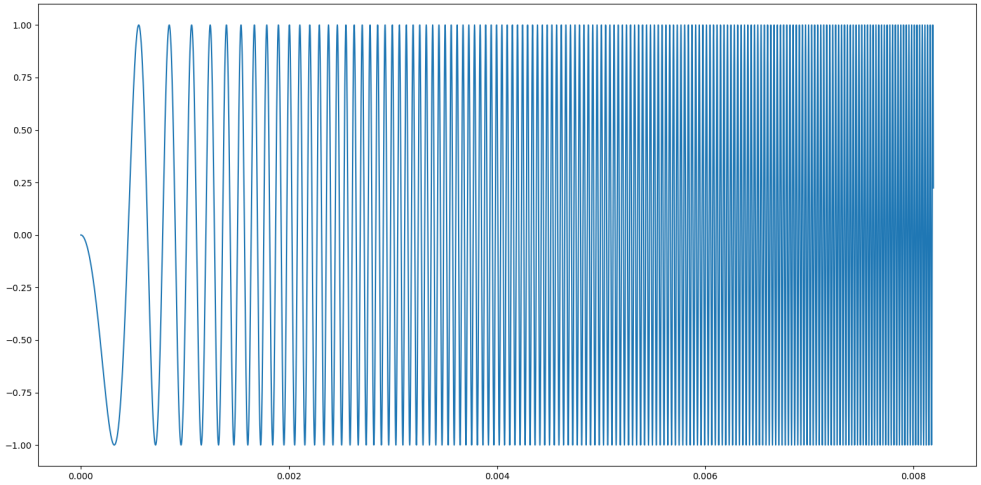
um simbolo LoRa é dada pela Equação 1, onde *j* é a representação dos imaginários.

$$x[n] = e^{j2\pi(\frac{n^2}{2N}+(\frac{s}{N}-\frac{1}{2})n)} \tag{1}$$

Cabe notar que, como explicado em Tapparel et al. (2020), um chirp LoRa modulado no domínio do tempo contínuo ocupa uma banda um pouco maior que β, por isso, usar uma frequência de amostragem igual à β introduz distorção por *aliasing*.

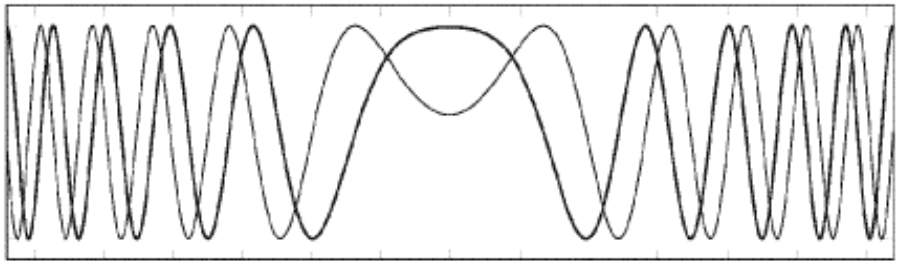
A Figura 1 exemplifica a modulação no domínio do tempo gerada para fins didáticos. Nessa imagem nenhum símbolo é modulado, por isso a frequência é crescente de forma contínua.

Figura 1 – Modulacão



Fonte – O autor

Figura 2 – Continuidade de Fase



Fonte – Seller e Sornin (2013)

Quando um sinal é transmitido, no meio pelo qual esse sinal passa, podem haver outros sinais indesejados como ruídos e sinais de outros transmissores, portanto a soma desses sinais com o sinal LoRa transmitido será chamada de y[n]. Segundo Ghanaatian et al. (2019) a demodulação

do sinal LoRa ocorre pela multiplicação de y[n] pelo sinal complexo conjugado de $x[n]$, que será chamado de $\bar{x}[n]$ e escolhido como sendo o sinal LoRa s igual à 0, também chamado de upchirp, conforme a Figura 4. Após, a transformada discreta de Fourier desse sinal é obtida, DTFT do inglês *Discrete Time Fourier Transform*. A Equação 2 representa todas essas operações onde o símbolo $\odot$ representa multiplicação matricial Hadamard, Tapparel et al. (2020), $\mathbf{y}$ e $\bar{\mathbf{x}}$ são os vetores de y[n] e $\bar{x}$ respectivamente.

$$Y = DFT(\boldsymbol{y} \odot \bar{\boldsymbol{x}}) \tag{2}$$

Finalmente, o elemento de maior energia de Y será o símbolo demodulado de interesse, sendo que na Equação 3, k pertence ao conjunto S e argmax é a função que retorna o elemento k que maximiza seu argumento.

$$s = argmax(\mathrm{Y[k]}^2) \tag{3}$$

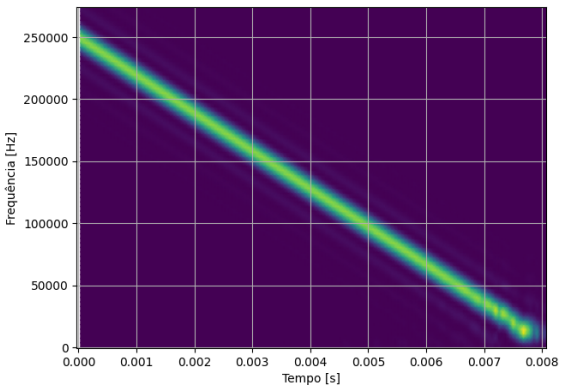
A fim de exemplificar o uso das equações 1, 2 e 3, foi escrito o código em Python3 que gerou as figuras 3, 4 e 5 com as características da Tabela 1 com o código apresentado no Apêndice A. A Figura 3 e a Figura 4 trazem um *downchirp* e um *upchirp* respectivamente, sem nenhum código modulado. Já a Figura 5 apresenta a codificação do símbolo 300 dentre os 1024 possíveis devido ao fator de espalhamento igual a 10.

Tabela 1 – Características modulação LoRa

SF	BW [kHz]	Fator de sobre amostragem
10	125	15

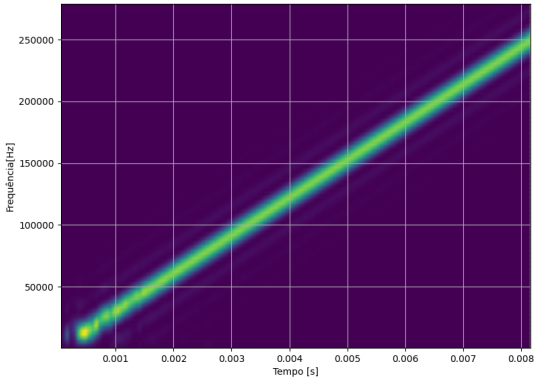
Fonte – O autor

Figura 3 – *Downchirp*



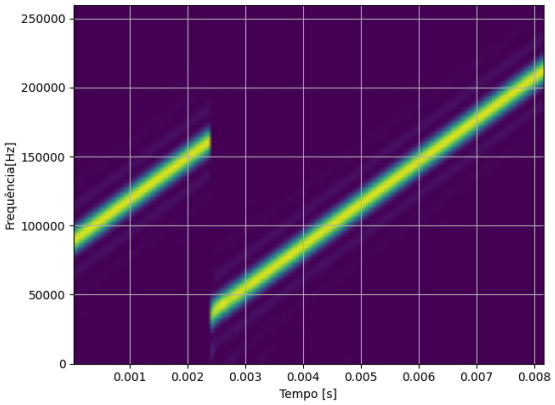
Fonte – O autor

Figura 4 – *Upchirp*



Fonte – O autor

Figura 5 – Símbolo 300 codificado



Fonte – O autor

A comunicação entre nós e *gateways* é distribuida em canais com bandas de 125KHz, 250KHz e 500KHz e diferente taxas de transmissão utilizando faixas ISM. No Brasil as faixas reservadas ao ISM possíveis para o LoRa são de 433 até 435 MHz, 915 até 928 MHz e 902 até 907,5 MHz. Para escolher uma taxa de transmissão, tem-se o compromisso de verificar que o alcance do enlace é comprometido com o aumento da frequência e da taxa de transmissão de dados, Yegin e Seller (2020), e, para se calcular a taxa de transmissão de dados, deve-se levar em consideração o fator de espalhamento, SF, do inglês *Spreading Factor* e a largura de banda BW, do inglês *Bandwidth*, conforme a Equação 4.

$$R(SF, BW) = SF\left(\frac{BW}{2^{SF}}\right) \qquad [bps] \tag{4}$$

A Tabela 2 a seguir exemplifica as taxas nominais que podem ser obtidas com as características associadas. Pode-se perceber que quanto maior o fator de espalhamento, menor a taxa de nominal de transferência de dados para a mesma largura de banda. Como a modulação LoRa

tem a opção de se adicionar correção de erro progressiva com diferentes taxas de código CR, do inglês *Coding Rate*, é necessário corrigir a Equação 4, conforme a Equação 5.

$$R(SF, BW, CR) = SF \left(\frac{4 * BW}{(4 + CR)2^{SF}} \right) \qquad [bps] \tag{5}$$

Tabela 2 – Características modulação LoRa

	SF	BW [kHz]	Taxa nominal[bit/s]	Max Payload [Bytes]
Uplink	10	125	980	11
Uplink	9	125	1760	53
Uplink	8	125	3125	125
Uplink	7	125	5470	242
Uplink	8	500	12500	242
Downlink	12	500	980	53
Downlink	11	500	1760	129
Downlink	10	500	3125	242
Downlink	9	500	5470	242
Downlink	8	500	12500	242
Downlink	7	500	21875	-

Fonte – Adaptado e corrigido de LoRa-Semtech (2021)

Também é mencionado em Yegin e Seller (2020) que a comunicação com diferentes fatores de espalhamento não geram interferênica intersímbolo - chamada de propriedade da quasi-ortogonalidade descrita por Edward et al. (2019). Segundo Georgiou e Raza (2017), enquanto a ortogonalidade cria canais virtuais extras, portanto aumentando a capacidade do *gateway* e da banda, Hou et al. (2020) elucida que transmissões com fatores de espalhamento semelhantes estão suscetíveis a outro tipo de interferência, única às redes LoRa, que é cointerferência e interinterferência de fatores de espalhamento.

Visto que o protocolo LoRaWAN na camada MAC é uma variante do ALOHA e esse não tem provisão para evitar colisão de pacotes, portanto, em áreas com muitos dispositivos transmitindo ao mesmo tempo. Mesmo que o protocolo LoRaWAN seja compulsório na aleatoriedade da frequência de envio de pacotes, inevitavelmente haverá pacotes enviados ao mesmo tempo com diferente fatores de espalhamento utilizando o mesmo canal, por isso, poderá haver colisão entre diferentes pacotes enviados por diferentes nós a um mesmo *gateway*.

2.2 PROTOCOLO DE COMUNICAÇÃO LORAWAN

O protocolo de comunicação LoRaWAN, que é otimizado para dispositvos alimentados a bateria, é tipicamente utilizado na topologia estrela onde os gateways fazem a conexão bidirecional entre os nós e o servidor, (YEGIN; SELLER, 2020). Os gateways são conectados aos servidores via protocolo de Internet padrão, IP, e, com o intuito de maximizar o tempo de vida útil da bateria dos dispositivos e aumentar a capacidade da rede, o protocolo de rede LoRaWAN

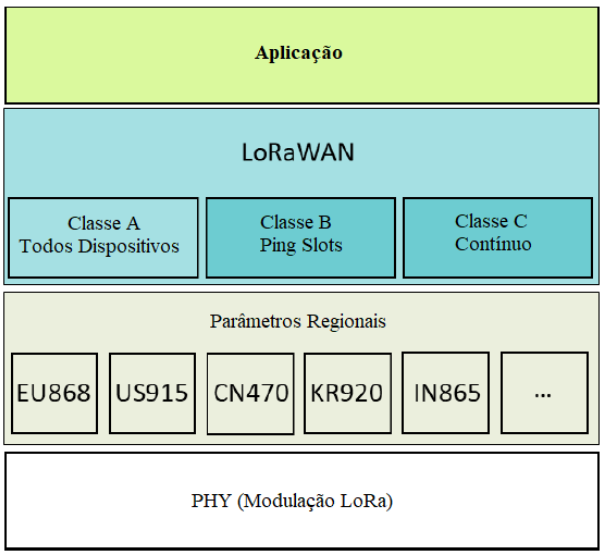
tem a capacidade de gerenciar a taxa de transmissão e a potência de transmisão de cada nó individualmente, Yegin e Seller (2020).

O gerenciamento da frequência das transmissões é feito pelo nó usando usando uma base de tempo acrescida de uma variação aleatória, com protocolo do tipo ALOHA, Yegin e Seller (2020). Com a utilização do protocolo ALOHA, os sinais LoRa podem sofrer com colisões que limitam a capacidade de toda a rede, no entanto, sinais com diferentes fatores de espalhamento são quasi-ortogonais, Edward et al. (2019), por isso, múltiplos sinais do mesmo esquema de modulação podem coexistir. Assim, um nó pode transmitir em qualquer canal, usando qualquer taxa de transmissão disponível pelo protocolo a qualquer momento, mas o nó deve trocar o canal a cada transmissão de modo aleatório e também deve alternar a periodicidade de transmissão da mesma forma. Essas condições garantem maior robustez na demodulação e decodificação e evitam a necessidade de sincronização sistemática dos diversos transmissores.

2.2.1 CLASSES DO PROTOCOLO LORAWAN

A modulação LoRa é um protocolo de camada física, patente da corporação Semtech, e o LoRaWAN é o protocolo da camada MAC especificado pela LoRa Alliance, onde definem-se as classes dos dispositivos usados em diferentes aplicações, apresentando diferentes consumos energéticos em três classes com suas respectivas características de abertura de janelas de transmissão. A classe A tem o menor custo energético, seguido pela classe B e C. Para que se possa comparar como as classes estão alocadas, a Figura 6 identifica as três classes, onde a mesma aplicação, os mesmos parâmetros regionais e a mesma camada física aplicam-se para todas elas.

Figura 6 – Classes



Fonte – Adaptado de Yegin e Seller (2020)

Aqui define-se o uso da expressão *downlink* como sendo a transmissão de dados do *gateway* para o nó e a expressão *uplink* como sendo a transmissão de dados do nó para o *gateway*.

- A classe A permite comunicação bi-direcional, onde cada *uplink* é respondido com dois *downlinks* em uma curta janela de tempo de transmissão. Cada nó tem a independência de programar a periodicidade de transmissão de *uplink*, obedecendo o requisito inicial da aleatoriedade temporal e de canal. Para que a classe A tenha o menor consumo energético entre as classes, ela permite *downlink* somente após uma transmissão de *uplink* e em um curto período de tempo. Assim, a abertura de uma janela de *downlink* vai depender de um *uplink*.

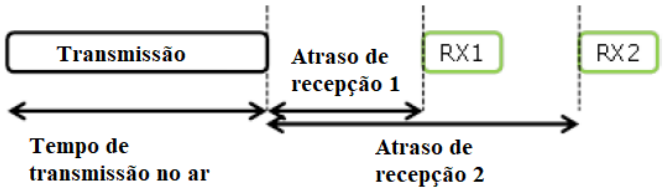
- Além de ter as mesmas janelas de abertura de *downlink* que a classe A, a classe B permite que sejam programadas janelas extras de *downlink*. Para que o nó possa receber os pacotes extras de *downlink*, ele recebe um sincronizador de tempo *beacon* do *gateway*. Isso permite ao servidor da rede saber quando o nó está com a janela de *downlink* aberta.

- A classe C permite que sejam transmitidos dados na direção *downlink* quase continuamente, fechando a janela apenas quando na direção reversa, *uplink*. Essa característica faz com que o consumo de potência média nesses dispositivos seja maior que as classes anteriores.

2.2.2 CAMADA DE ENLACE

Após cada *uplink*, o nó abre uma ou duas janelas de *downlink*, essas janelas de tempo são mostradas na Figura 7 e o instante de abertura das janelas de *downlink* é referenciado a partir do fim da janela de *uplink*. Caso nenhum pacote seja destinado ao nó na primeira janela, RX1, o nó deve abrir uma segunda janela RX2 de *downlink*.

Figura 7 – Janelas de Transmissão



Fonte – Adaptado de Yegin e Seller (2020)

Se um preâmbulo sincronizador é detectado durante a janela de recebimento, o nó deve se manter ativo até que o pacote de *downlink* seja demodulado. Se após demodulado, verificado o endereço e o código de integridade da mensagem, MIC do inglês *Message Integrity Code* , e a mensagem seja destinada ao nó, então o nó não deve abrir uma segunda janela.

A primeira janela de *downlink*, RX1, usa a frequência e taxa de transmissão de dados em função da frequência e taxa usada no *uplink* e, não deve ser aberta depois do tempo de atraso

de recepção 1, conforme Figura 7. A segunda janela de *downlink*, quando necessária, usa taxa e frequência de transmissão fixas e deve ser aberta antes do tempo de atraso de recepção 2, conforme Figura7. A duração das janelas de *downlink*, RX1 e RX2, deve ser pelo menos o tempo suficiente para que o nó detecte o preâmbulo sincronizador de *downlink*.

A primeira parte de um pacote a ser enviado é o preâmbulo sincronizador. Esse pode ser identificado na Figura 8 e consiste de um número variável N de *upchirps*, sendo N pertencente ao conjunto { 6...65535} permitindo detecção efetiva de preâmbulo para grandes variações do SNR, segundo Bernier, Dehmas e Deparis (2020).

No segundo bloco que pode-se identificar na Figura 8, depois do preâmbulo sincronizador, são transmitidos dois identificadores de rede, Yegin e Seller (2020). A sequência de *downchirps*, depois dos identificadores da rede, são dois símbolos utilizados também como sincronizadores de rede. Eles são descritos em Tapparel et al. (2020) e também são usados por Xhonneux e Louveaux (2019) e Bernier, Dehmas e Deparis (2019) para distinguir o STO, do inglês *Sampling Time Offset* e o CFO, do inglês *Carrier Frequency Offset*.

Figura 8 – Pacote

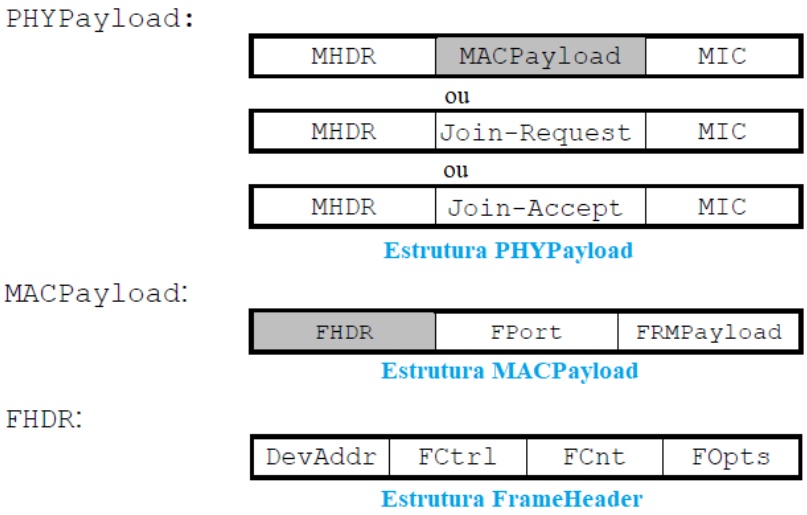
Fonte – adaptado de Tapparel et al. (2020)

Como continuação da explicação da Figura 8, a Figura 9 apresenta mais detalhes da documentação de Yegin e Seller (2020). Todos os pacotes LoRaWAN *uplinks* e *downlinks* portam uma *MACpayload* que é composta de um byte de cabeçalho de MHDR, do inglês *Media Access Control Header*, seguido por uma MAC*payload* de tamanho variável e termina com quatro bytes de código de verificação de integridade da mensagem, MIC, conforme Figura 9.

O cabeçalho MHDR que antecede o *MACPayload* da Figura 9, reapresentado expandido na Figura 10, tem comprimento de um byte é composto pelo tipo de *frame*, Ftype do inglês *Frame Type*, que se diferencia entre oito tipos, conforme Figura 11. *Join-Request* e *Join-accept* são usados para procedimentos de ativação e são melhores descritos em Yegin e Seller (2020). *Unconfirmed data uplink*, *unconfirmed data downlink*, *confirmed data uplink*, *confirmed data downlink* são usados como avisos de recebimento ou não recebimento de *uplink* ou *downlink*. Os dois bits *Major* indicam qual a versão do formato de *Frame* LoRAWAN utilizado e os bits corespondentes ao RFU são reservas para uso futuro, do inglês *Reserved for Future Use*.

O comprimento máximo da *MACPayload* é determinado pela taxa de transferência de dados e pelos parâmetros regionais em que estão localizados, portanto todos os pacotes que não estiverem contidos nos parâmetros são descartados, Yegin e Seller (2020). O MAC*Payload* é composto de um *frame header*, FHDR, um *port field*, FPort, e um frame payload, FRMPayload, sendo os dois últimos opcionais e são agrupados conforme a estrutura MACPayload da Figura 9.

Figura 9 – *Payload*



Fonte – Adaptado de Yegin e Seller (2020)

Figura 10 – Cabeçalho MHDR

Bits	[7:5]	[4:2]	[1:0]
MHDR	FType	RFU	Major

Fonte – Yegin e Seller (2020)

Figura 11 – Tipo de *Frame*

FType	**Description**
000	Join-Request
001	Join-Accept
010	unconfirmed data uplink
011	unconfirmed data downlink
100	confirmed data uplink
101	confirmed data downlink
110	RFU
111	Proprietary

Fonte – Yegin e Seller (2020)

O FHDR é um *frame header* que contém o endereço do nó, DevAddr, do inglês *Device Address*, um byte de controle de *frame*, FCtrl do inglês *Frame Control*, dois *bytes* contadores de *frame*, FCnt do inglês *Frame Counter*, e até quinze *bytes* opcionais Fopts que são usados para transporte de comandos MAC. Como os pacotes são diferenciados entre *downlink* e *uplink*, a estrutura do FCtrl apresentada na Figura 12, usada para *downlink*, e a estrutura na Figura 13, usada para *uplink*, e podem ser melhor comprendidas pela leitura de Yegin e Seller (2020).

Por fim, o MIC da Figura 9 é utilizado para verificar a integridade da mensagem, e utiliza o algoritmo AES-CMAC e está descrito em Song et al. (2006) e em Yegin e Seller (2020).

Figura 12 – *FCtrl Downlink*

Bits	7	6	5	4	[3..0]
FCtrl	ADR	RFU	ACK	FPending	FOptsLen

Fonte – Yegin e Seller (2020)

Figura 13 – *FCtrl Uplink*

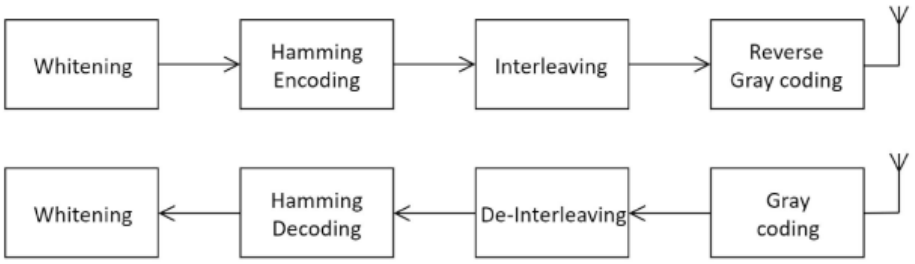
Bits	7	6	5	4	[3..0]
FCtrl	ADR	ADRACKReq	ACK	ClassB	FOptsLen

Fonte – Yegin e Seller (2020)

2.3 ESQUEMAS DE CODIFICAÇÃO E DECODIFICAÇÃO

O processo de codificação e decodificacão da informação em nível de bit é realizado em quatro passos que podem ser visualizado na Figura 14. Descreve-se aqui as operações realizadas por cada bloco, bem como os métodos, enfatizando suas funcionalidades.

Figura 14 – Codificação e decodificação em bit



Fonte – Tapparel (2019)

O pacote anteriormente explicado na subseção 2.2.2 é enviado, mas antes o *header* e o *payload* passam pelos seguintes blocos que fazem operações em nível de bit, para permitir mais opções de ortogonalidade no espaço dos vetores e consequente maior robustez contra interferência de sinais de mesma natureza. Essas operações em bit também tem a propriedade de tornar o sinal mais resistente ao ruído.

2.3.1 BRANQUEAMENTO

Branqueamento ou *whitening* do inglês é uma operação ou exclusivo, XOR, da informação com uma sequência pseudoaleatória que é conhecida em ambos os lados, transmissor e receptor. Essa função tem o objetivo de remover o nível contínuo da informação transmitida, e, embora não garanta que o faça, tem uma probabilidade alta de remover, Tapparel (2019). Para recuperar a informação, basta realizar novamente a operação XOR da informação recebida com a mesma

sequência pseudoaleatória conhecida em ambos os lados. Nos blocos implementados em simulação, a sequência de 255 bytes utilizada é explicitada na Figura 15 a seguir, em codificação hexadecimal. Como se pode notar pela palavra chave *const* da linguagem C++ na Figura 15, foi declarado ao compilador que essa é uma variável que não pode ser modificada, por isso é um parâmetro fixo.

Figura 15 – Sequência usada no branqueamento

```
#ifndef TABLES_H
#define TABLES_H

namespace gr {
    namespace lora_sdr {
        //whitenig sequence
        const uint8_t whitening_seq[] = {
            0xFF, 0xFE, 0xFC, 0xF8, 0xF0, 0xE1, 0xC2, 0x85, 0x0B, 0x17, 0x2F, 0x5E, 0xBC, 0x78, 0xF1, 0xE3,
            0xC6, 0x8D, 0x1A, 0x34, 0x68, 0xD0, 0xA0, 0x40, 0x80, 0x01, 0x02, 0x04, 0x08, 0x11, 0x23, 0x47,
            0x8E, 0x1C, 0x38, 0x71, 0xE2, 0xC4, 0x89, 0x12, 0x25, 0x4B, 0x97, 0x2E, 0x5C, 0xB8, 0x70, 0xE0,
            0xC0, 0x81, 0x03, 0x06, 0x0C, 0x19, 0x32, 0x64, 0xC9, 0x92, 0x24, 0x49, 0x93, 0x26, 0x4D, 0x9B,
            0x37, 0x6E, 0xDC, 0xB9, 0x72, 0xE4, 0xC8, 0x90, 0x20, 0x41, 0x82, 0x05, 0x0A, 0x15, 0x2B, 0x56,
            0xAD, 0x5B, 0xB6, 0x6D, 0xDA, 0xB5, 0x6B, 0xD6, 0xAC, 0x59, 0xB2, 0x65, 0xCB, 0x96, 0x2C, 0x58,
            0xB0, 0x61, 0xC3, 0x87, 0x0F, 0x1F, 0x3E, 0x7D, 0xFB, 0xF6, 0xED, 0xDB, 0xB7, 0x6F, 0xDE, 0xBD,
            0x7A, 0xF5, 0xEB, 0xD7, 0xAE, 0x5D, 0xBA, 0x74, 0xE8, 0xD1, 0xA2, 0x44, 0x88, 0x10, 0x21, 0x43,
            0x86, 0x0D, 0x1B, 0x36, 0x6C, 0xD8, 0xB1, 0x63, 0xC7, 0x8F, 0x1E, 0x3C, 0x79, 0xF3, 0xE7, 0xCE,
            0x9C, 0x39, 0x73, 0xE6, 0xCC, 0x98, 0x31, 0x62, 0xC5, 0x8B, 0x16, 0x2D, 0x5A, 0xB4, 0x69, 0xD2,
            0xA4, 0x48, 0x91, 0x22, 0x45, 0x8A, 0x14, 0x29, 0x52, 0xA5, 0x4A, 0x95, 0x2A, 0x54, 0xA9, 0x53,
            0xA7, 0x4E, 0x9D, 0x3B, 0x77, 0xEE, 0xDD, 0xBB, 0x76, 0xEC, 0xD9, 0xB3, 0x67, 0xCF, 0x9E, 0x3D,
            0x7B, 0xF7, 0xEF, 0xDF, 0xBF, 0x7E, 0xFD, 0xFA, 0xF4, 0xE9, 0xD3, 0xA6, 0x4C, 0x99, 0x33, 0x66,
            0xCD, 0x9A, 0x35, 0x6A, 0xD4, 0xA8, 0x51, 0xA3, 0x46, 0x8C, 0x18, 0x30, 0x60, 0xC1, 0x83, 0x07,
            0x0E, 0x1D, 0x3A, 0x75, 0xEA, 0xD5, 0xAA, 0x55, 0xAB, 0x57, 0xAF, 0x5F, 0xBE, 0x7C, 0xF9, 0xF2,
            0xE5, 0xCA, 0x94, 0x28, 0x50, 0xA1, 0x42, 0x84, 0x09, 0x13, 0x27, 0x4F, 0x9F, 0x3F, 0x7F
        };
    }
}

#endif /* TABLES_H */
```

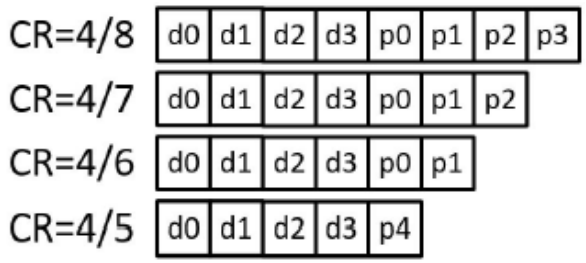
Fonte – Código fonte do arquivo tables.h disponível em Tapparel et al. (2021)

2.3.2 CÓDIGO DE HAMMING

O código de Hamming é utilizado para realizar correção de erros que permite que *bits* sejam recuperados pela inserção de redundâncias em cada *codeword*, nesse caso, a cada quatro bits. Segundo a patente Seller e Sornin (2013) e Tapparel (2019), sabe-se que é possível obter taxas de correção de erro de 4/5, 4/6, 4/7 e 4/8, onde o numerador da fração representa o número de bits da informação e o denominador representa a soma da redundância com o número de bits da informação. Essa operação é realizada precisamente antes da operação de *interleaving*, Tapparel (2019).

Tapparel (2019) explica que o código de Hamming é realizado a partir de operações XOR dos bits da *codeword* como segue na Figura 16, onde d0 representa o *bit* menos significativo e d3 o *bit* mais significativo. Na Figura 16, p0, p1, p2 e p3 são os bits de paridade definidos na Equação 6, Equação 7, Equação 8 e Equação 9, que fazem a operação XOR dos bits da informação, sendo XOR representado por $\oplus$. Já P4 é o bit menos significativo negado das somas de d0, d1, d2 e d3.

Figura 16 – Sequência de Bits do código de Hamming



Fonte – Tapparel (2019)

$$p0 = d0 \oplus d1 \oplus d2 \tag{6}$$

$$p1 = d1 \oplus d2 \oplus d3 \tag{7}$$

$$p2 = d0 \oplus d1 \oplus d3 \tag{8}$$

$$p3 = d0 \oplus d1 \oplus d2 \tag{9}$$

2.3.3 INTERLEAVING

Durante uma transmissão, pode haver corrupção por ruído, desvanecimento ou uma soma de fatores que podem levar à múltiplos bits de um símbolo a serem corrompidos, Tapparel (2019). Quando todos os erros são causados em apenas um símbolo, os erros são altamente correlacionados e os códigos de correção de erros não conseguem corrigir. Também, segundo Seller e Sornin (2013), a probabilidade de acontecer um erro no bit menos significativo é a metade que a probabilidade de ocorrer um erro no segundo bit menos significativo e assim sucessivamente.

Assim, objetiva-se com a função *interleaving* distribuir os erros em diversas *codewords* tal que seja mais efetivo utilizar o código de Hamming para recuperar os símbolos LoRa. Entretanto, essa operação causa aumento na latência visto que é necessário esperar múltiplos símbolos antes de que seja possível recuperar uma *codeword* inteira, Tapparel (2019).

O processo de interleaving se dá na diagonal e pode ser observada a sequência na Tabela 3, onde os bits das mensagens são representados, sendo que o primeiro índice dos elementos de C representa a linha e o segundo a coluna, m e S respectivamente.

Tabela 3 – Tabela *interleaving*

	S0	S1	S2	S3	S4	S5	S6
m =0	C00	C61	C52	C43	C34	C35	C16
m =1	C10	C01	C62	C53	C44	C45	C26
m =2	C20	C11	C02	C63	C54	C55	C36
m =3	C30	C21	C12	C03	C64	C65	C46
m =4	C40	C31	C22	C13	C04	C05	C56
m =5	C50	C41	C32	C23	C14	C15	C66
m =6	C60	C51	C42	C33	C24	C25	C06

Fonte – Seller e Sornin (2013)

2.3.4 CODIFICAÇÃO GRAY

A codificação Gray é um mapeamento entre um símbolo em uma representação numérica para uma sequência binária. A particularidade da sequência obtida é que símbolos adjacentes na representação numérica só diferem de um bit. Embora Tapparel (2019) cite que nas modulações CSS, a qual LoRa está incluido, não seja o ruído branco que cause o erro entre símbolos adjacentes, mas a CFO e SFO.

Para fazer o mapeamento Gray, em Tapparel (2019) é utilizada a Equação 10, onde B é o bit menos significativo, $\oplus$ é a operação XOR e $>>$ a operação de deslocamento de bit para a direita.

$$C = B \oplus (B >> 1) \tag{10}$$

3 DESENVOLVIMENTO DO TRABALHO

Antes de serem inciadas as simulações, foram realizadas preparações de ambiente e correção de softwares a serem apresentados nesse capítulo. Como base para realizar as simulações, foi utilizada a pesquisa da EPFL, *École Polytechnique Fédérale de Lausanne*, que ainda está em andamento no momento de escrita desse trabalho e pode ser encontrada em (TAPPAREL et al., 2021). O projeto tomado como base se chama *LoRa PHY based on GNU Radio*, e a pesquisa tem por objetivo fazer a engenharia reversa de toda a cadeia da camada física LoRa, usar em um protótipo funcional através do rádio definido por software compatível com o GNURadio com todas as funcionalidades que as especificações de modulação, demodulação codificação e decodificação em bit LoRa descrevem, desde o transmissor ao receptor.

Até o momento, já foram desenvolvidos no projeto *LoRa PHY based on GNU Radio* na parte do enlace que corresponde à transmissão, os blocos principais já explicados na seção 2.3, sendo eles *header*, *CRC*, *whitening*, *Hamming encoder*, *interleaver* e *gray mapping*. Já a recepção conta com todos os blocos que fazem a operação inversa dos blocos citados, além desses, blocos que realizam a sincronização do tempo e frequência, ou seja, *Carrier Frequency Offset*, CFO, e *Sampling Time Offset*, STO também foram desenvolvidos.

Os trabalhos que foram gerados a partir dessa pesquisa são Tapparel (2019), Tapparel et al. (2020) e também foram usados para compreensão das operações descritas na patente Seller e Sornin (2013).

Como o projeto da EPFL teve como base o uso do GNURadio, que é um software de desenvolvimento aberto que fornece ferramentas para simulação de sinais e blocos de processamento para implementar em SDR, é possível que sejam desenvolvidos blocos específicos para realizarem as funções desejadas, onde esses blocos adicionais são chamados de OOT, do inglês *Out Of Tree*. Além disso, o GNURadio pode ser usado para realizar experimentos utilizando o hardware SDR USRP, do inglês *Universal Software Radio Peripheral*. Segundo MACHADO (2021), existem diferentes arquiteturas de SDR, sendo que algumas soluções são projetadas apenas para recepção, como o RTL-SDR. Outras soluções tem capacidade de transmissão, recepção e múltiplas entradas e múltiplas saídas (MIMO).

MACHADO (2021) também menciona que o GNURadio é um *framework* que fornece estruturas para processamento de sinais para implementação via software. Ele é arquitetado para o uso de DSP tendo compatibilidade com SDR. Oferece suporte para pesquisa em comunicação sem fio e tem uma construção coletiva com uso da licença GPL, do inglês *General Public License*.

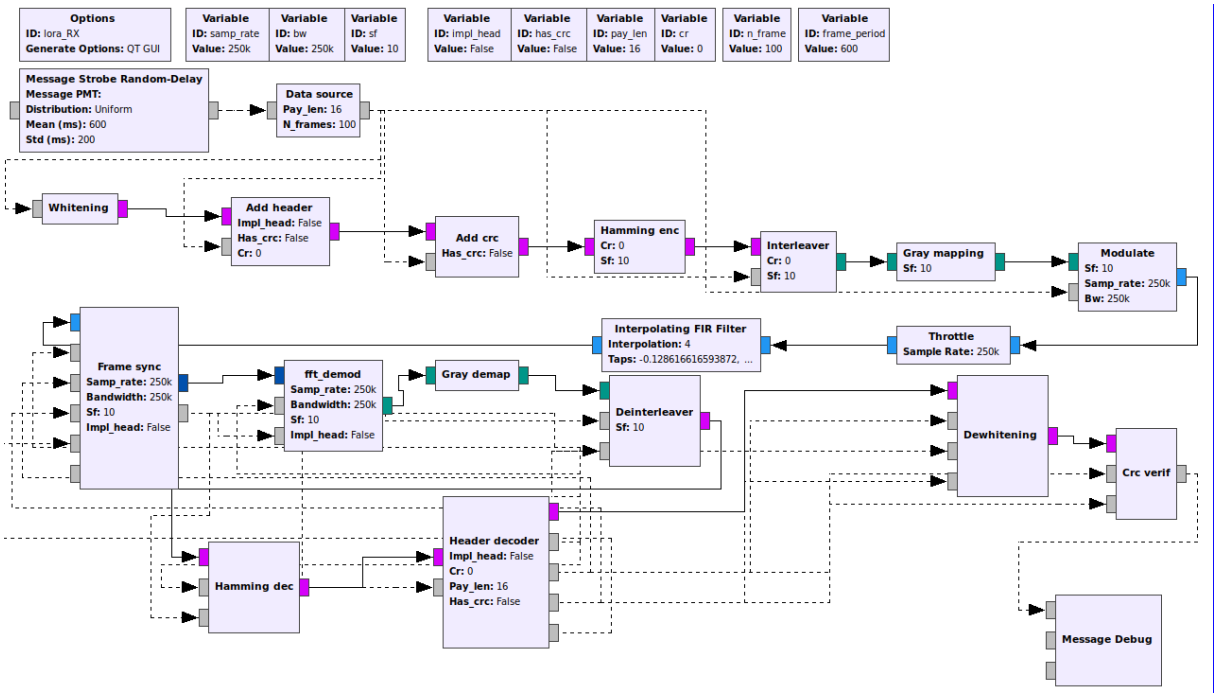
O elemento básico do GNURadio é uma estrutura em bloco que pode ter funcionalidades de processamento de sinais, processamento de fluxo de dados, operações matemáticas complexas como filtros e FFT (*Fast Fourier Transform*). Quando instalado, o GNURadio tem por padrão uma biblioteca com diversos blocos, entre eles, moduladores digitais e analógicos de esquemas

de modulação como AM, FM, BPSK, FSK, PAM, QAM, PAM filtros e blocos para conectar ao hardware USRP.

Além dos blocos da biblioteca padrão, podem ser instalados no GNURadio os módulos OOT, e esses módulos podem ser compostos de um ou mais blocos que podem ser instalados nas versões compatíveis de GNURadio para a qual foram programados. Blocos OOT permitem que funcionalidades adicionais sejam criadas e usadas em conjunto com a biblioteca padrão. Um exemplo de modulação com blocos criados por Tapparel et al. (2021) pode ser vistos na Figura 17. Os blocos são desenvolvidos na linguagem C++ e a interface com o usuário através dos blocos é feita na linguagem Python, ou seja, as informações de entrada do usuário são interpretadas em Python.

As entradas dos blocos em Python possibilitam que operações que não utilizam bibliotecas fora da instalação padrão do Python sejam utilizadas, por exemplo, operações de aritmética simples, a utilização de notação científica ou operações lógicas. Por fim, quando a estrutura de blocos está pronta para ser simulada, a compilação é feita na linguagem Python.

Figura 17 – Exemplo de aplicação contendo transmissão e recepção



Fonte – O autor

Anteriormente as diversas instalações de software do GNURadio, é necessário realizar a instalação de uma das distribuições do Sistema Operacional Linux, e, nesse caso, escolheu-se uma distribuição estável e compatível com as instalações a serem executadas posteriormente. A versão gratuita do Ubuntu 18.04 LTS foi escolhida, e essa distribuição está disponível em CanonicalLtd (2021).

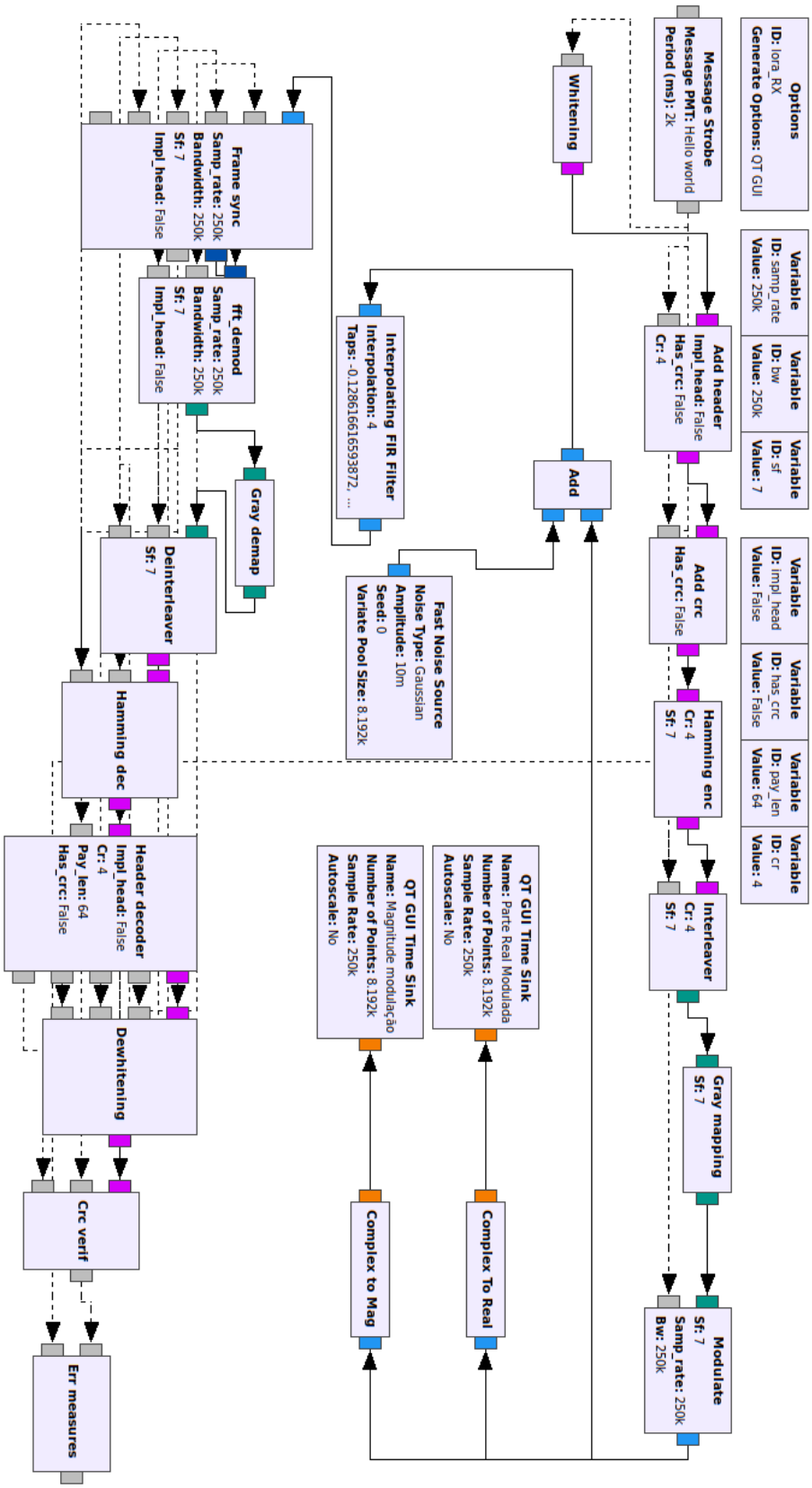
Como se teve por objetivo instalar OOT, foi utilizado Pybombs para realizar a instalação do GNURadio. O Pybombs é uma ferramenta para auxiliar a instalação do GNURadio, módulos

OOT e fontes binárias. Para realizar a instalação dos pacotes referentes ao projeto da EPFL, seguiu-se as instruções de Tapparel et al. (2021), mas alterações de código de instalação foram necessárias para completar a instalação com sucesso. O Apêndice B traz as alterações realizadas no arquivo frame_sync_impl.cc.

Ainda, para que houvesse compatibilidade de versões de software dos pacotes OOT desenvolvidos pela EPFL e o GNURadio, foi instalada a versão retroativa GNURadio 3.7.11, visto que já existe a versão 3.8 disponível. Por isso, também foi necessário realizar a instalação do Python2.7.

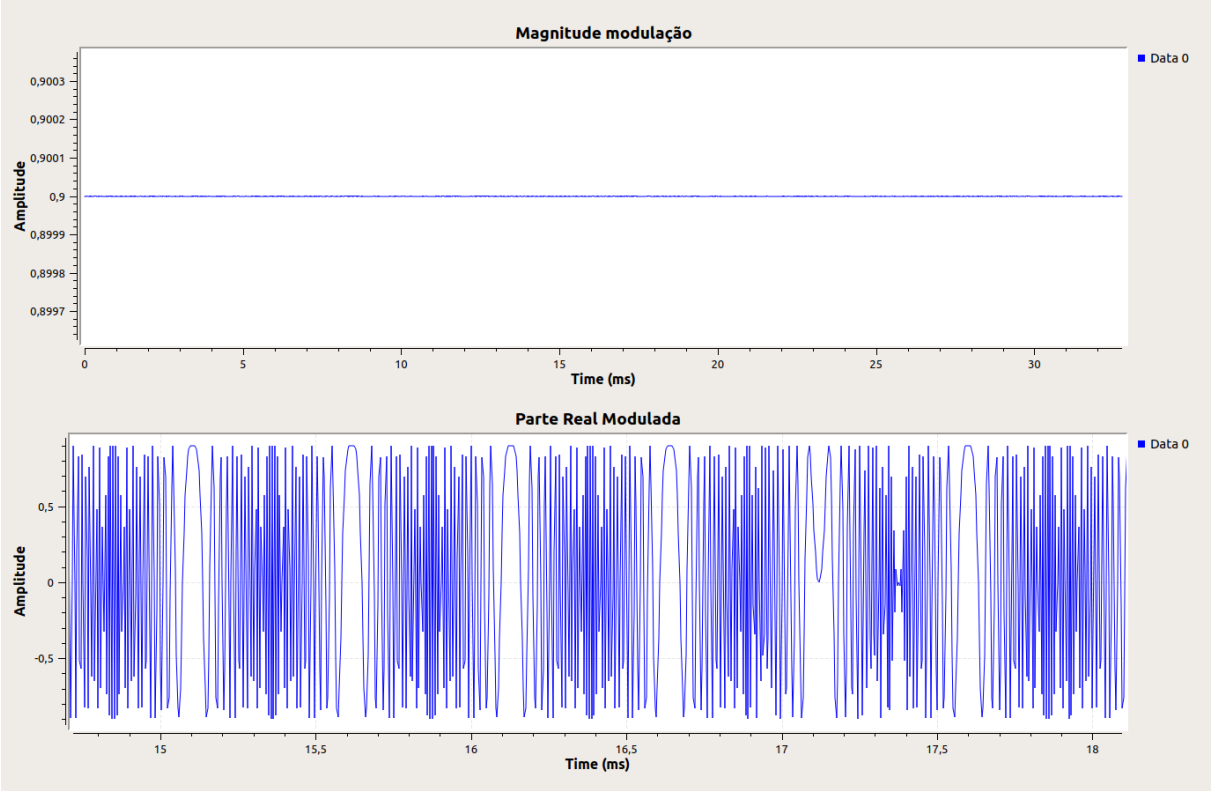
Com o objetivo de calcular a SNR e a SIR, coube um estudo inicial para descobrir e garantir qual é o valor da amplitude de saída do bloco de modulação LoRa. Para isso, foi realizado um pequeno teste demonstrado a seguir. Foi gerada a modulação, separando a parte real do sinal e calculada a magnitude do sinal complexo. O projeto da simulação está demonstrado na Figura 18. Esse teste inicial, com resultados na Figura 19, permitiu descobrir a magnitude da amplitude de saída do modulador, no caso 0,9. Para padronizar a amplitude com valor 1, todas as saídas dos moduladores simulados a seguir foram divididas por 0,9. Na figura Figura 19 também pode-se observar que a continuidade de fase entre símbolos e a formação do chirp no domínio do tempo.

Figura 18 – Avaliação Inicial



Fonte – O Autor

Figura 19 – Amplitude Modulação



Fonte – O Autor

A fim de comparar a metodologia utilizada por Tapparel et al. (2021), buscou-se no artigo Croce et al. (2018) onde foram publicados resultados referentes a símbolos inter-interferentes ortogonais pelo fator de espalhamento. Nesse artigo é analisada a modulação LoRa numericamente utilizando o MATLAB e código próprio para mostrar que colisões entre pacotes modulados com diferentes fatores de espalhamento podem causar perda de pacotes se a potência do sinal ortogonal interferente for grande o bastante.

Cabe enfatizar que no artigo Croce et al. (2018) foi negligenciada a codificação em bit *whitening*, embora na patente Seller e Sornin (2013) seja descrito que após o preâmbulo sincronizador, o cabeçalho e a *payload* passam por essa operação. As outras operações em bit, *interleaving*, codificação Hamming e codificação *gray* foram realizadas.

Na simulação feita por Croce et al. (2018) a amplitude do sinal de referência tem valor igual 1 e o sinal inter-interferente é variante, consequentemente variando a SIR, do inglês *Signal to Interference Ratio*. O resultado da soma do sinal interferente com o sinal de interesse foi demodulado sem a adição de canal com ruído e, para cada simulação, foram gerados pacotes aleatórios interferentes até que ocorressem um total de 100 erros. Não foi citada a largura de banda e os resultados encontram-se na Figura 20 e Figura 21.

Nas simulações realizadas que resultaram na Figura 22 com o objetivo de comparar com a Figura 21, foi utilizada amplitude do sinal de interesse igual a 1 e amplitude do sinal inter-interferente variante. A banda de transmissão utilizada tem largura de 250KHz, e foram

utilizadas as operações em *bit*, *coding rate*, *whitening*, *interleaving* e nenhum cabeçalho foi utilizado. Foram transmitidos pacotes com 32 caracteres aleatórios em intervalo de tempo de 200 ± 50 ms com distribuição linear.

Para gerar cada ponto do gráfico de cada fator de espalhamento foram transmitidos 201 pacotes, mas aproveitados 200 para o estudo. O último pacote de cada simulação foi descartado e foi adotado esse método devido ao atraso de decodificação em relação ao instante de envio do pacote. Como a simulação foi feita de modo a encerrar logo após ao envio do último pacote e a decodificação tem um atraso, foi necessário gerar esse atraso no encerramento da simulação.

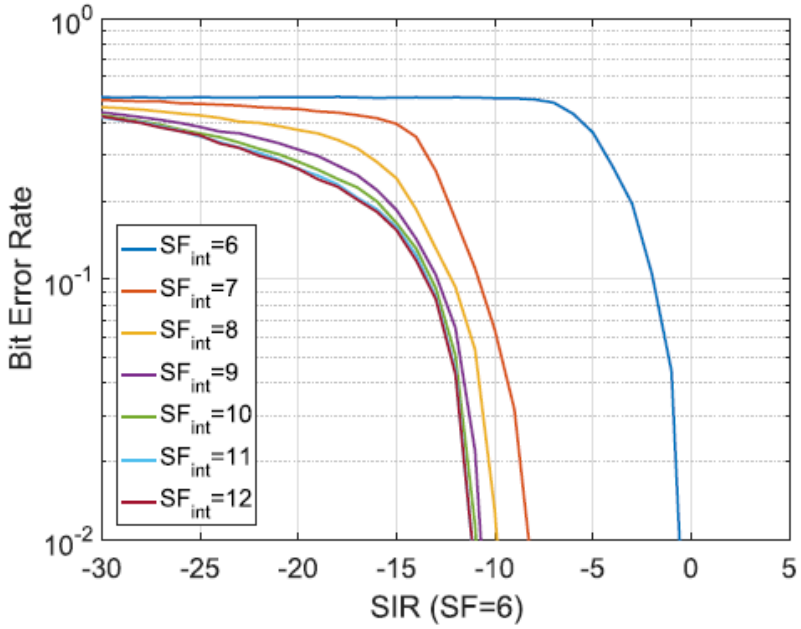
Ainda, cada ponto do gráfico foi contabilizado na forma que apenas os pacotes que chegassem completamente inteiros fossem contabilizados, ou seja, foi considerado um pacote perdido caso houvesse 1 bit ou mais de erro ou se o pacote não fosse decodificado corretamente ou se não chegasse, desta forma foram contabilizados como pacotes certos aqueles que chegassem e fossem totalmente decodificados.

Para gerar o sinal aleatório interferente, foram utilizadas as mesmas configurações do sinal de interesse, exceto o fator de espalhamento e a semente do gerador de caracteres.

Com o objetivo de gerar resultados para comparar diretamente com a Figura 20, foi simulado com as mesmas condições da Figura 22, ou seja, sem usar *header*, mas nenhum pacote foi decodificado. Então, para entender melhor o problema, inseriu-se o *header* e obteve-se os resultados apresentados na Figura 24.

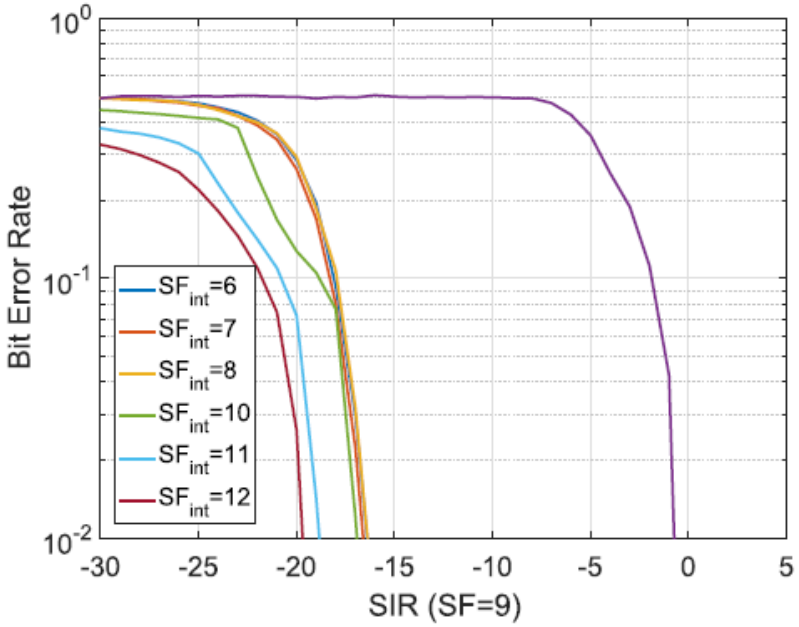
A Figura 26 e a Figura 27 representam os blocos montados para as simulações resultantes das figuras Figura 22 e Figura 24 e as devidas ligações.

Figura 20 – SF de interesse = 6



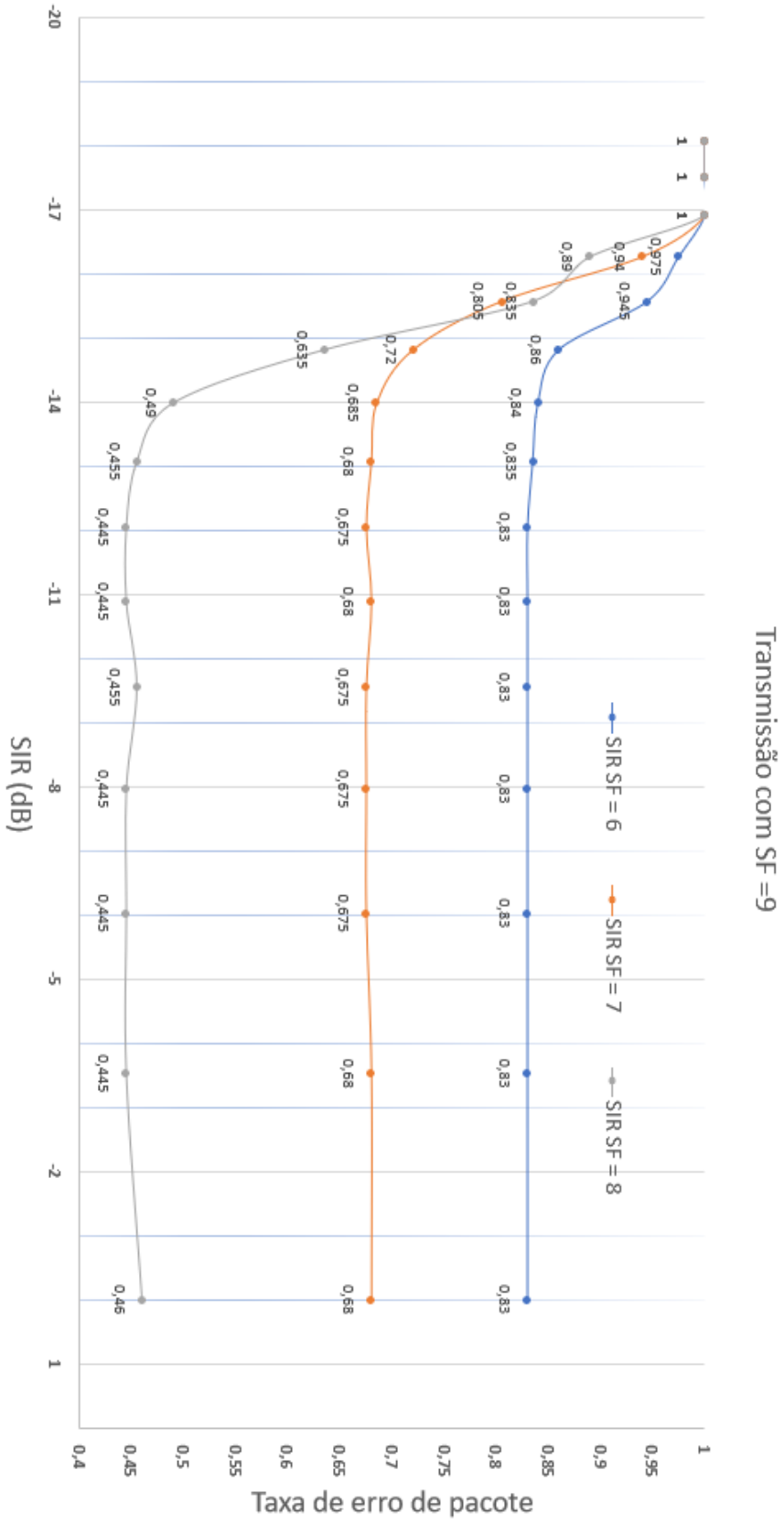
Fonte – Croce et al. (2018)

Figura 21 – SF de interesse = 9



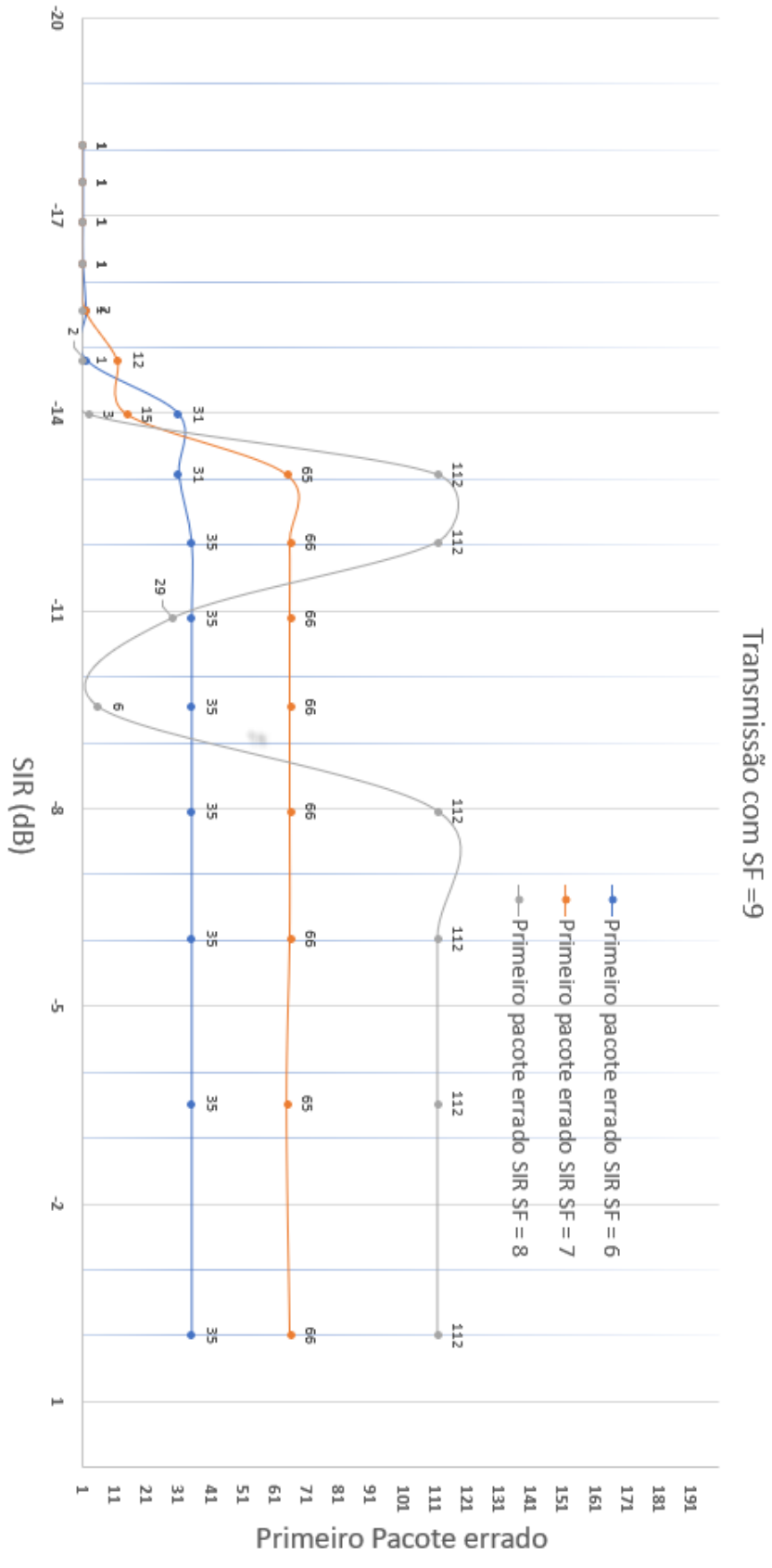
Fonte – Croce et al. (2018)

Figura 22 – SF de interesse = 9



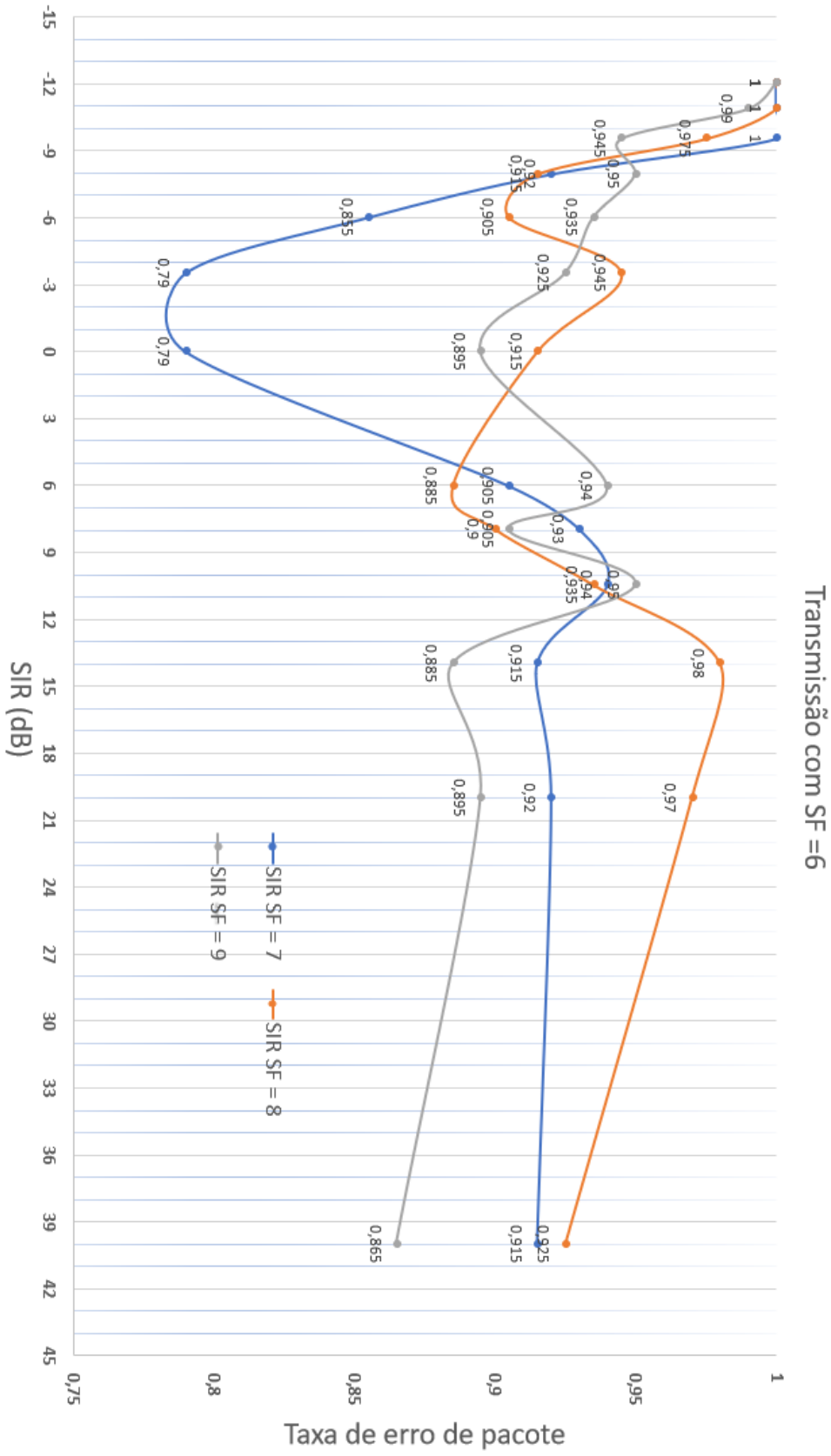
Fonte – o autor

Figura 23 – Primeiro pacote errado



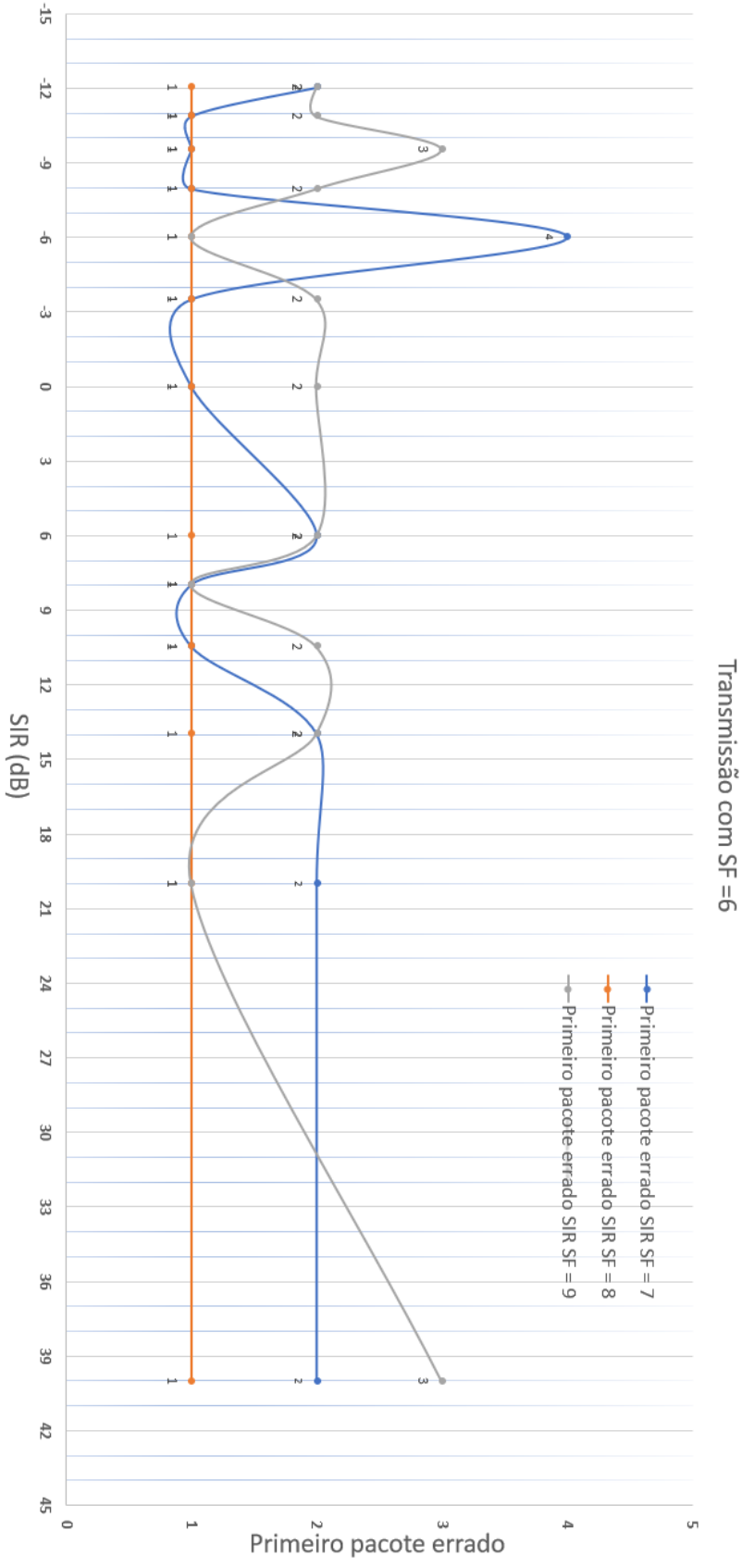
Fonte – o autor

Figura 24 – SF de interesse = 6



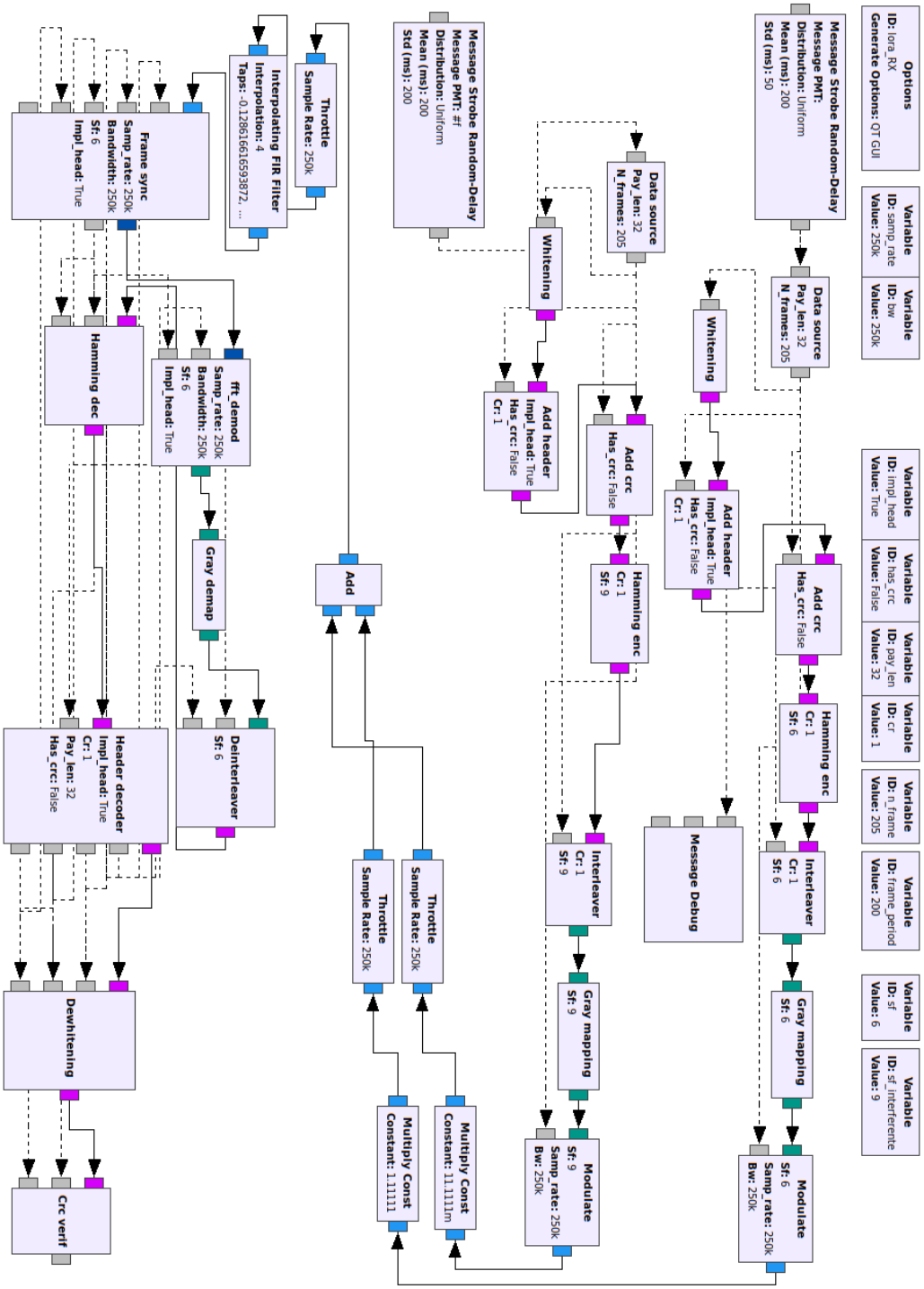
Fonte – o autor

Figura 25 – SF de interesse = 6



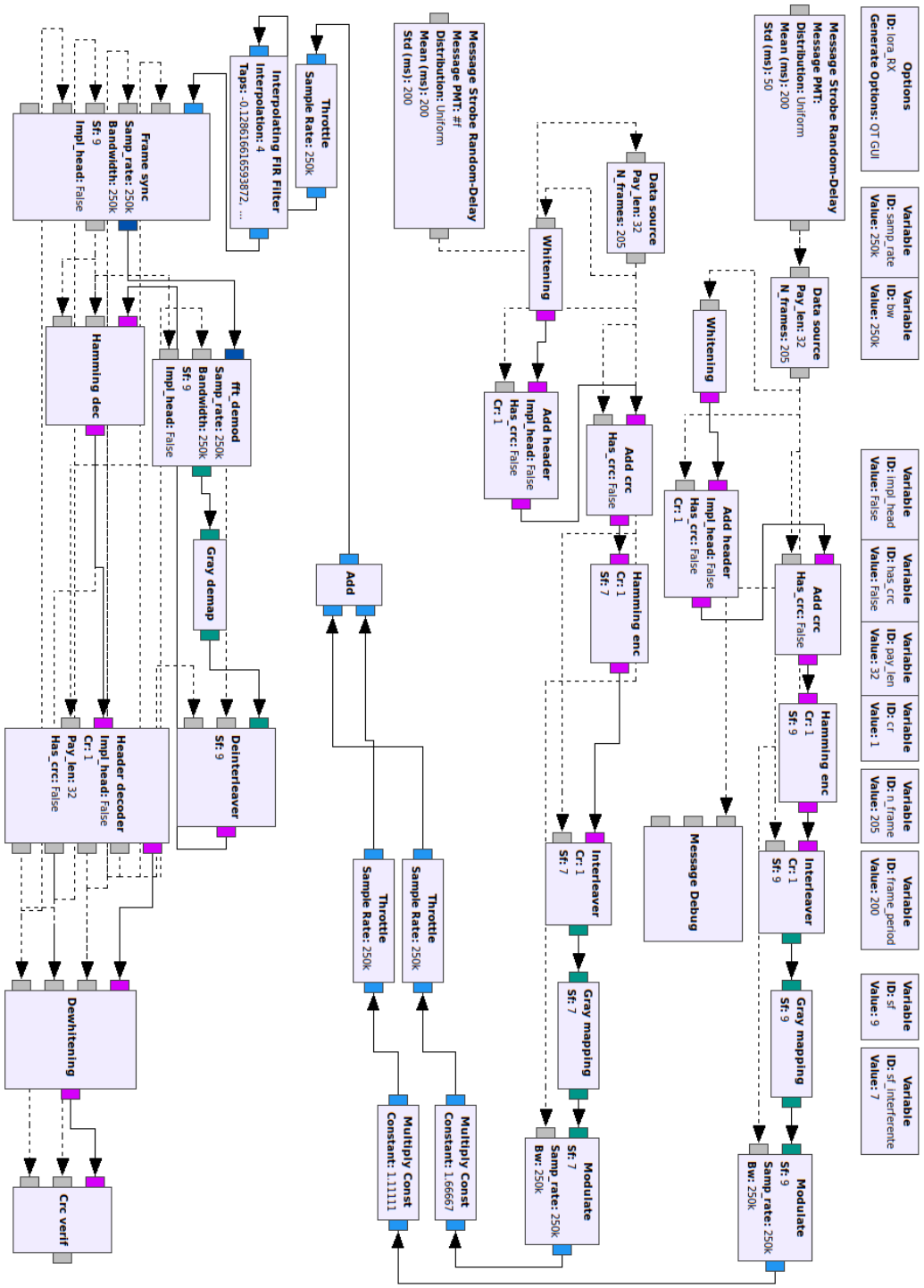
Fonte – o autor

Figura 26 – SF de interesse = 6



Fonte – o autor

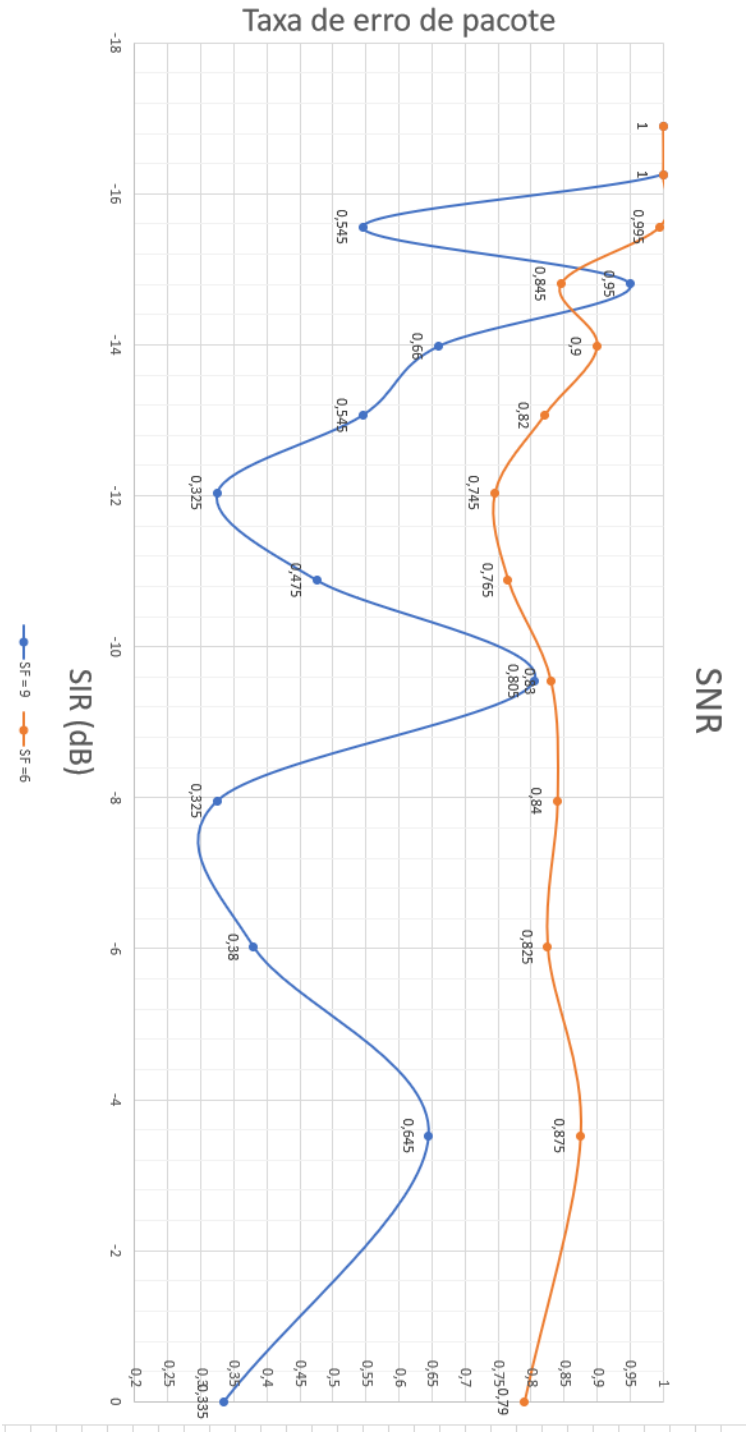
Figura 27 – SF de interesse = 9



Fonte – o autor

A fim de comparar o desempenho de um sinal ortogonal interferindo o sinal LoRa de interesse, simulou-se a transmissão com adição de ruido de distribuição gaussiana, os resultados encontram-se na Figura 28. As mesmas condições aplicadas ao sinal de interesse com fator de espalhamento 9 foram mantidas.

Figura 28 – SNR



Fonte – O autor

Adicionalmente, foi tentado simular com fator de espalhamento 11 e 12, mas obteve-se o erro de simulação Apêndice C que não foi possível solucionar. Como essa é uma notificação do bloco de sincronização, cabe verificar se nas outras simulações as baixas taxas de acerto não são causadas pelo mesmo problema.

De acordo com os resultados obtidos, pode-se perceber que quanto menor a SIR ou quanto

menor a SNR, maior a taxa de erro de pacotes, o que é esperado. No caso dos resultados da Figura 22 esperava-se encontrar uma correlação direta de proximidade de fator de espalhamento do sinal de interesse com o sinal interferente com aumento do erro, mas obteve-se justamente o inverso. Ainda na mesma figura, a maior taxa de erro de pacote acontece quando o sinal com fator de espalhamento igual a 6 interfere o sinal de interesse com fator de espalhamento 9, quando comparado com os sinais interferentes com fator de espalhamento 7 e 8 com a mesma SIR.

4 CONCLUSÕES

Nesse trabalho foi avaliado o LoRa, um dos protocolos de comunicações para IoT mais amplamente empregados. Foram realizadas simulações para avaliar a colisão entre pacotes com diferentes ajustes de espalhamento espectral do protocolo, além da avaliação de robustez frente ao ruído de canal. Os resultados obtidos não foram consistentes com a literatura. Comparativamente, foi realizada uma simulação adicionando um canal com ruído a fim de comparar com os resultados obtidos na colisão de pacotes, mas apresentaram anomalias semelhantes com as anteriores.

Levando em consideração apenas os resultados da Figura 22 e os resultados da Figura 24 a tendência dos gráficos mostram que, ao contrário do que foi mostrado em (CROCE et al., 2018) nas Figura 21 e Figura 20, os resultados levam a inferir que a proximidade dos fatores de espalhamento pode não estar correlacionada com a taxa de erro de pacotes, visto que as maiores taxas de erros são obtidas quando o fator de espalhamento que não é o mais próximo está interferindo.

Avaliando os resultados das Figura 23 e Figura 25 que indicam o momento em que o primeiro pacote errado ou perdido foi percebido, pode-se notar que para o caso de um sinal de interesse com fator de espalhamento igual à 9 e um sinal interferente com fator de espalhamento igual à 8 o primeiro pacote errado aconteceu, em média, 66 pacotes depois que quando um sinal com fator de espalhamento igual a 7 estava interferindo e 77 pacotes depois que o quando o sinal com fator de espalhamento igual a 6 estava interferindo. Essa avaliação mostra que, para o mesmo sinal de interesse e variando apenas o sinal interferente, há variação significativa no inicio da perda dos pacotes.

Os altos índices de erro nos resultados alcançados em Figura 22 e Figura 24 levam a inferir que há erros de sincronização possivelmente por dois motivos. O primeiro diz respeito ao acerto da demodulação e decodificação em parte dos pacotes enviados que, caso não estivessem implementados de forma correta, não poderiam teriam sido decodificados. O segundo diz respeito a taxa de acerto dos pacotes enviados após acontecer o primeiro erro, que convergem a zero, ou seja, uma vez que aconteceu o primeiro erro ao demodular ou decodificar, o receptor dificilmente reconhece pacotes corretos novamente.

Numa próxima abordagem sugere-se a investigação para se saber qual a relação da sincronização dos pacotes com as taxas de erro, visto que dos 201 pacotes enviados, sendo apenas 200 de interesse, em parte das simulações quando os erros começaram a acontecer significativamente, menos pacotes foram reconhecidos novamente.

Apesar dos resultados inconsistentes, toda a estrutura de simulação, filtro de dados do terminal Linux e análise dos dados poderá ser reutilizada em trabalhos futuros, tanto no processo de correção da simulação quanto na avaliação de outras simulações. Esses algoritmos são essenciais no processo de validação de pacotes corretos visto que automatizam a verificação e

trazem agilidade à avaliação como um todo.

REFERÊNCIAS

AFISIADIS, O. et al. On the error rate of the lora modulation with interference. **IEEE Trans. Wireless Commun**, 2020.

BERNIER, C.; DEHMAS, F.; DEPARIS, N. Low complexity lora frame synchronization for ultra-low power software-defined radios. **https://hal-cea.archives-ouvertes.fr/cea-02280910**, 2019.

BERNIER, C.; DEHMAS, F.; DEPARIS, N. Low complexity lora frame synchronization for ultra-low power software-defined radios. **IEEE Transactions on Communications, Institute of Electrical and Electronics Engineers**, p. 14, 2020.

CANONICALLTD. **Ubunutu Bionic Beaver**. https://releases.ubuntu.com/18.04/: [s.n.], 2021.

CENTENARO, M. e. a. Long-range communications in unlicensed bands: the rising stars. p. 8, 2016.

CHIANI, M.; ELZANATY, A. On the lora modulation for iot: Waveform properties and spectral analysis. **IEEE Internet of Things Journal**, p. 6, 2019.

COMMITTEE, I. C. S. L. S. **IEEE Standard for Low-Rate Wireless Networks**: Ieee std 802.15.4. [S.l.], 2020. 799 p.

CROCE, D. et al. Impact of lora imperfect orthogonality: analysis of link-level performance. **IEEE COMMUNICATIONS LETTERS, VOL. 22, NO. 4**, p. 4, 2018.

EDWARD, P. et al. On the coexistence of lora and interleaved chirp spreading lora based modulations, wimob. **International Conference on Wireless and Mobile Computing, Networking and Communications**, p. 6, 2019.

ELSHABRAWY, T. et al. On the different mathematical realizations for the digital synthesis of lora-based modulation. **European Wireless Conf.**, p. 6, 2019.

ELSHABRAWY, T.; ROBER, J. Analysis of ber and coverage performance of lora modulation under same spreading factor interference. **Proc.IEEE PIMRC**, p. 6, 2018.

GEORGIOU, O.; RAZA, U. Low power wide area network analysis: Can lora scale? **IEEE WIRELESS COMMUNICATIONS LETTERS**, p. 4, 2017.

GHANAATIAN, R. et al. Lora digital receiver analysis and implementation. **IEEE Int. Conf. on Acoustics**, 2019.

GOURSAUD, C.; GORCE, J. Dedicated networks for iot: Phy/mac state of the art and challenges. **AI Endorsed Trans. Internet Things**, p. 11, 2015.

HOU, Y. et al. A novel mac protocol exploiting concurrent transmissions for massive lora connectivity. **JOURNAL OF COMMUNICATIONS AND NETWORKS, VOL. 22, NO. 2, APRIL 2020**, p. 10, 2020.

KALAA, M. O. A. et al. Evaluating bluetooth low energy in realistic wireless environments. **IEEE Wireless Communications and Networking Conference**, p. 6, 2016.

LORA-SEMTECH. https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan: [s.n.], 2021.

MACHADO, E. R. A container-based architecture to provide services from sdr devices. p. 100, 2021.

NOREEN, U.; BOUNCEUR, A.; CLAVIER, L. A study of lora low power and wide area network technology. **3rd International Conference on Advanced Technologies for Signal and Image Processing - 2017,Fez, Morroco**, p. 6, 2017.

RAPPAPORT, T. S. **Comunicações sem fio : Princípios e práticas**. 2. ed. São Paulo: Pearson Prentice Hall, 2009.

SELLER, O. B. A.; SORNIN, N. **Low power long range transmitter**. Sainte Soulle, France: EUROPEAN PATENT APPLICATION, 2013.

SONG, J. et al. The aes-cmac algorithm. **RFC Editor**, 2006.

TAPPAREL, J. Complete reverse engineering of lora phy. **Telecommunications Circuits Laboratory EPFL, Lausanne**, p. 25, 2019.

TAPPAREL, J. et al. An open-source lora physical layer prototype on gnu radio. p. 5, 2020.

TAPPAREL, J. et al. **LoRa PHY based on GNURadio**. [S.l.], 2021. Disponível em: <https://www.epfl.ch/labs/tcl/resources-and-sw/lora-phy/>. Acesso em: 2 de junho de 2021.

XHONNEUX, D. B. M.; LOUVEAUX, J. A low-complexity synchronization scheme for lora end nodes. **https://arxiv.org/abs/1912.11344**, 2019.

YEGIN, A.; SELLER, O. **LoRaWan@ L2 1.0.4 Specification**. Fremont, CA, 2020. 90 p.

APÊNDICE A – MODULAÇÃO

Modulação

```
import math
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from scipy import signal


#import the pyplot and wavfile modules


class Chirp:

    def __init__(self,SF,BW,CF, OS = 1):
        self.SF = SF  #spreading factor
        self.BW = BW  #Band Width [Rad]
        self.CF = CF  #Central Frequency
        self.OS = OS  #oversampling factor
        # frequencia de amostragem
        self.SAMPLING_FREQ = 2*self.OS*self.BW


        #Calculated with the above definitions

        #Chirp time frame
        self.CT = (2**self.SF)/self.BW

        #Chip time frame
        self.Tn = 1 / self.BW

        #First derivative of frequency
        self.FREQ_DERIVATIVE = self.BW / self.CT


    def demodulation(self,mod_chirp):
```

```
        down_chirp = np.zeros(int(self.SAMPLING_FREQ * self.CT))
        demodulated_chirp = np.zeros(int(self.SAMPLING_FREQ * self.CT))
        time = np.arange(0, self.CT, 1/self.SAMPLING_FREQ)

        for it in range(len(time)):
            frequency = self.FREQ_DERIVATIVE * time[it]
            down_chirp[it] = math.sin ( 2*math.pi*time[it] * frequency)

        down_chirp[:] = down_chirp[::-1]



        f, t, Sxx = signal.spectrogram(down_chirp, self.SAMPLING_FREQ, nfft=
            1024)
        plt.pcolormesh(t, f, Sxx, shading='gouraud')
        plt.ylabel('Frequency [Hz]')
        plt.xlabel('Time [sec]')
        plt.grid()
        #plt.ylim((0,self.BW))
        plt.show()




        for it in range(len(mod_chirp)):
            demodulated_chirp[it] = down_chirp[it] * mod_chirp[it]

        sig = np.fft.ifft(demodulated_chirp)
        modulo_sinal = np.abs(sig) **2

        plt.plot(time[:int(len(time)/2)],modulo_sinal[0:int(len(modulo_sinal)/2)],
            'b-')
        plt.ylabel('Energia')
        plt.xlabel('tempo[s]')
        plt.show()


    def modulation(self):

        chirp = np.zeros(int(self.SAMPLING_FREQ * self.CT))
        time = np.arange(0, self.CT, 1/self.SAMPLING_FREQ)
```

```
        n = 0

        while (n<1000):
            for it in range(len(time)):
                if time[it]<(self.Tn *n):
                    frequency = self.BW + self.FREQ_DERIVATIVE *
                        (time[it]-n*self.Tn)
                else:
                    frequency = 0 - self.FREQ_DERIVATIVE * (time[it]-n*self.Tn)

                chirp[it] = math.sin (2*math.pi* time[it] * frequency)

            f, t, Sxx = signal.spectrogram(chirp, self.SAMPLING_FREQ, nfft=
                1024,noverlap=20)
            plt.pcolormesh(t, f, Sxx, shading='gouraud')
            plt.ylabel('Frequency [Hz]')
            plt.xlabel('Time [sec]')
            plt.grid()
            plt.show()

            #plt.specgram(chirp,Fs=self.SAMPLING_FREQ)
            #plt.xlabel('Tempo')
            #plt.ylabel('Frequencia')
            #plt.ylim(0,125000)
            #plt.show()

            n= n + 300
            self.demodulation(chirp)

            break

def main():
    first = Chirp(SF = 10, BW = 125000, CF =0, OS= 15)
    Chirp.modulation(first)


if __name__ == "__main__":
    main()
```

APÊNDICE B – CORREÇÃO CÓDIGO DE INSTALAÇÃO

Correção código instalação

Original:

```
#wrong network identifier
Linha 392: else if (abs(bin_idx-net_id_1)>1){
#wrong network identifier
Linha 406: if (abs(bin_idx-net_id_2)>1){
```

Corrigido:

```
#wrong network identifier
Linha 392: else if ((bin_idx-net_id_1)>1){
#wrong network identifier
Linha 406: if ((bin_idx-net_id_2)>1){
```

APÊNDICE C – ERRO DE SIMULAÇÃO

Erro de simulação

```
sched: <block frame\_sync (10)> is requesting more input data
than we can provide.
ninput\_items\_required = 8200
max\_possible\_items\_available = 8191
If this is a filter, consider reducing the number of taps
```