UNIVERSIDADE FEDERAL DO AMAPÁ
DEPARTAMENTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS – DCET
CURSO DE BACHARELADO EM ENGENHARIA ELÉTRICA

HUGO BRUNO SANTOS ARAÚJO

ANÁLISE DE DESEMPENHO DE CÓDIGOS CORRETORES DE ERROS EM REDES DE SENSORES SEM FIO PARA APLICAÇÕES EM SMART GRID

MACAPÁ-AP
2019

HUGO BRUNO SANTOS ARAÚJO

ANÁLISE DE DESEMPENHO DE CÓDIGOS CORRETORES DE ERROS EM REDES DE SENSORES SEM FIO PARA APLICAÇÕES EM SMART GRID

Trabalho de Conclusão de Curso apresentado à Universidade Federal do Amapá como exigência parcial para obtenção do título de Bacharel em Engenharia Elétrica.

Orientadora: Profa. Dra. Fernanda Regina Smith Neves Corrêa

MACAPÁ-AP
2019

Dados Internacionais de Catalogação na Publicação (CIP)
Biblioteca Central da Universidade Federal do Amapá
Elaborada por Orinete Costa Souza – CRB-2/1709

Araújo, Hugo Bruno Santos.

Análise de desempenho de códigos corretores de erros em redes de sensores sem fio para aplicações em smart grid / Hugo Bruno Santos Araújo ; orientadora, Fernanda Regina Smith Neves Corrêa. – Macapá, 2019.

81 f.

Trabalho de Conclusão de Curso (Graduação) – Fundação Universidade Federal do Amapá, Coordenação do Curso de Bacharelado em Engenharia Elétrica.

1. Redes elétricas inteligentes. 2. Nós sensores. 3. Reed-Solomon (RS). 4. Low Density Parity Check (LDPC). 5. Razão sinal-ruído (SNR). 6. Taxa de erros de bit (BER). I. Corrêa, Fernanda Regina Smith Neves, orientadora. II. Fundação Universidade Federal do Amapá. III. Título.

621.319 A663a
CDD: 22. ed.

UNIVERSIDADE FEDERAL DO AMAPÁ
DEPARTAMENTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
COORDENAÇÃO DO CURSO DE ENGENHARIA ELÉTRICA

ATA DE DEFESA DE TCC

Às 15:20 horas do dia 21 de junho de 2019, nas dependências da Universidade Federal do Amapá, reuniu-se a Banca Examinadora para defesa de TCC 2 intitulado "ANÁLISE DE DESEMPENHO DE CÓDIGOS CORRETORES DE ERROS EM REDES DE SENSORES SEM FIO PARA APLICAÇÕES EM SMART GRID" de autoria do aluno HUGO BRUNO SANTOS ARAÚJO regularmente matriculados no Curso de Engenharia Elétrica desta universidade. A banca Examinadora foi assim constituída: Profª. Dra. Fernanda Regina Smith Neves Corrêa, Presidente da Banca e Orientadora, Prof. Dr. Geraldo Neves de Albuquerque Maranhão e Profª. Ma. Kellen Diane de Carvalho Gomes, como examinadores. Concluída a defesa, foram realizadas as arguições e comentários. Em seguida procedeu-se o julgamento pelos membros da Banca Examinadora, tendo o projeto sido (APROVADO/REPROVADO) APROVADO, com NOTA (0 a 10 pts) 9,7. E, para constar, eu, Profª. Dra. Fernanda Regina Smith Neves Corrêa, presidente da Banca Examinadora, lavrei a presente ata que, após lida e achada conforme, foi assinada por mim e demais membros da Banca Examinadora.

Macapá (AP), 21 de junho de 2019.

Fernanda Smith

Profª. Dra. Fernanda Regina Smith Neves Corrêa
Presidente

Prof. Dr. Geraldo Neves de Albuquerque Maranhão
Membro

Kellen Gomes

Profª. Ma. Kellen Diane de Carvalho Gomes
Membro

AGRADECIMENTOS

Agradeço primeiramente a minha família, em especial a minha mãe Izaete e a minha tia Rutileide, que sempre estiveram ao meu lado e me apoiaram em todas as decisões que tomei.

Agradeço aos meus amigos Gleice, André e Camila, pelos momentos memoráveis e pela cumplicidade durante a graduação.

Agradeço também aos meus colegas da AFAP, que me acompanharam durante o TCC e me incentivaram até o fim.

Agradeço a Coordenação De Aperfeiçoamento De Pessoal De Nível Superior – CAPES pela oportunidade de ter participado do programa Ciência Sem Fronteiras. Não estaria aonde estou hoje se não tivesse sido por essa oportunidade.

A minha orientadora, Profa. Fernanda Smith, por estar sempre presente e disposta a me ajudar. Agradeço a confiança em mim.

Agradeço a todos que direta ou indiretamente contribuíram para a conclusão dessa fase na minha vida acadêmica.

RESUMO

O desequilíbrio entre geração e demanda de energia, falhas de equipamentos e a escassez de técnicas eficientes de monitoramento e controle indicam que é necessário tomar medidas rumo a um sistema com uma melhor integração com sistemas de comunicação e controle. A tecnologia *Smart Grid* é caracterizada por sua comunicação bidirecional, infraestrutura de monitoramento, sensoriamento e controle, e integração com fontes de geração distribuída e renováveis. Associadas a ela estão as Redes de Sensores Sem Fio (RSSF), que se apresentam como uma importante ferramenta na modernização do Sistema de Potência, com seu baixo custo de implantação e características flexíveis. Porém, o canal de comunicação que envolve a tecnologia *Smart Grid* e as RSSF sofre com condições ambientais desfavoráveis, interferências, problemas de conectividades e mudanças de topologia, fazendo com que, técnicas de correção de erros sejam essenciais para assegurar a confiabilidade na transmissão de dados. Neste trabalho foram analisadas as performances de dois códigos corretores de erro, *Reed-Solomon* (RS) e *Low Density Parity Check* (LDPC) – códigos de verificação de paridade de baixa densidade, para uma Rede de Sensores sem Fio com três nós sensores, e diferentes rotas de transmissão de dados. Constatou-se que é mais eficiente enviar pacotes de dados através de vários nós para curtas distâncias do que para longas distâncias. Verificou-se também que o código LDPC apresenta melhor desempenho em termos de taxa de erro de bit comparado com o código RS e um pacote sem codificação. Por fim, são feitos comentários com base nos resultados para algumas aplicações de *Smart Grid*.

Palavras-chave: *Reed-Solomon* (RS), *Low Density Parity Check* (LDPC), nós sensores, taxa de erros de bit, razão sinal-ruído (SNR).

ABSTRACT

The imbalance between generation and energy demand, equipment failures and the lack of efficient monitoring and control techniques indicate that it is time to take steps toward a better integration with communication and control systems. The Smart Grid technology is characterized by its two-way communication, monitoring, sensoring and control infrastructure, and integration with renewable energy and distributed centers of energy generation. Associated to it there is the Wireless Sensor Networks (WSN), a key technology in the modernization process of the existing Power Grid, with their low-cost deployment and flexibility. However, the Smart Grid and WSN communication channel is affected by hostile environmental conditions, interference, connectivity issues and dynamic topology changes, and so it is indispensable the utilization of error correction control to assure reliability in the data transmission. In this work, two error control codes were studied, Reed-Solomon (RS) and Low Density Parity Check (LDPC), applied to a Wireless Sensor Network with three sensor nodes, and different transmission routes. It was attested that is more efficient to send data packets through multiple nodes and short distances than for longer distances. It was also verified that the LDPC code had a better performance considering bit error rates than the Reed-Solomon code and an uncoded packet. Finally, a few comments are made based on the results for some Smart Grid applications.

Keywords: Reed-Solomon (RS), Low Density Parity Check (LDPC), sensor nodes, bit error rate, SNR.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

3G	*Third Generation of Mobile Networks*
4G	*Fourth Generation of Mobile Networks*
ARQ	*Automatic Request for Retransmission*
ASK	*Amplitude shift keying*
AWGN	*Additive White Gaussian Noise*
BAN	*Building Area Networks*
BCH	*Bose-Chaudhuri-Hocquenghem*
BER	*Bit Error Rate*
BPSK	*Binary Phase Shift Keying*
FAN	*Field Area Networks*
FEC	*Forward Error Correction*
FSK	*Frequency shift keying*
G.hn	*Gigabit Home Networking*
GD	Geração Distribuída
GF	*Galois Field*
GW	*Gateway*

HARQ	*Hybrid ARQ*
IAN	*Industrial Area Networks*
IEEE	*Institute of Electric and Electronic Engineers*
HAN	*Home Area Networks*
LDPC	*Low Density Parity Check*
LLR	*Log-Likelihood Ratio*
NAN	*Neighborhood Area Networks*
OQPSK	*Offset Quadrature Phase Shift Keying*
PLC	*Power Line Communications*
PSK	*Phase shift keying*
PSTN	*Public Switch Telephone Network*
QC-LDPC	*Quasi-Cyclic LDPC*
QC-LDPC-BC	*Quasi-Cyclic LDPC Block Codes*
QoS	Qualidade de serviço
QPSK	*Quadrature Phase Shift Keying*
RS	*Reed Solomon*
RSSF	Redes de Sensores sem Fio

SEP	Sistema Elétrico de Potência
SNR	Razão sinal-ruído
WAN	*Wide Area Networks*
WiMax	Worldwide Interoperability for Microwave Access

LISTA DE SÍMBOLOS

k	Número de bits de mensagem de um código de blocos linear
n	Tamanho da palavra-código de um código de blocos linear, número de nós de variável na matriz de verificação de paridade
$n-k$	Bits de paridade ou redundância
$\mathbf{G}$	Matriz geradora de um código
I_k	Matriz identidade $k \times k$
$\mathbf{P}$	Sub-matriz de paridade, matriz de coecientes de paridade
$\mathbf{u}$	Vetor contendo os bits de mensagem
$\mathbf{v}$	Vetor contendo os bits codificados, palavra-código
$\mathbf{H}$	Matriz de verificação de paridade de um código
$\mathbf{r}$	Vetor da palavra recebida
$c(X)$	Representação polinomial da palavra-código
$g(X)$	Polinômio gerador
$m(X)$	Polinômio da mensagem
$b(X)$	Polinômio do resto da divisão
m	Número de bits

p	Número de elementos de um campo finito de Galois
$RS(N, K)$	Forma sistemática dos códigos RS
N	Número de símbolos da palavra-código de um código RS, número de bits codificados de um código LDPC
K	Número de símbolos de mensagem de um código RS, número de bits de informação de um código LDPC
R	Número de símbolos de paridade de um código RS, taxa de um código LDPC
t	Capacidade de correção de um código RS, número de colunas da matriz de verificação de paridade de um código QC-LDPC na forma compacta
β	Elemento primitivo de um Campo de Galois, raiz do polinômio do verificação de paridade
$h(X)$	Polinômio do verificação de paridade
$C(X)$	Representação polinomial da palavra-código de um código RS
$G(X)$	Polinômio gerador de um código RS
$M(X)$	Polinômio da mensagem de um código RS
$r(X)$	Polinômio do resto da divisão de um código RS
c	Número de linhas da matriz de verificação de paridade de um código QC-LDPC na forma compacta

b	Tamanho das sub-matrizes da matriz de verificação de paridade de um código QC-LDPC e fator de expansão de H
$P_{i,j}$	Sub-matriz da matriz de verificação de paridade de um código QC-LDPC
$\mathbf{H_c}$	Matriz de verificação de paridade de um código QC-LDPC na forma compacta
$p_{i,j}$	Sub-matriz da matriz de verificação de paridade de um código QC-LDPC na forma compacta
$s(t)$	Forma de onda senoidal
$r(t)$	Amplitude ou contorno de uma onda senoidal
f_c	Frequência de uma onda senoidal
$\psi(t)$	Fase de uma onda senoidal
$x(t)$	Sinal transmitido
$n(t)$	Ruído do canal
$y(t)$	Sinal recebido
$\sigma_n{}^2$	Variância do ruído
N_0	Densidade espectral de potência do ruído
SNR	Razão sinal-ruído
P_{sinal}	Potência do sinal transmitido

$\mathrm{P_{ruído}}$	Potência do ruído do canal
$\mathrm{E_b}$	Energia por bits transmitidos
$\mathrm{n1, n2, n3}$	Nós sensores
$\mathrm{h1, h2, h3}$	Saltos na RSSF
BER	Taxa de erros de bits
P_r	Potência recebida
$\mathrm{P_t}$	Potência transmitida
A	Constante que depende das características da antena e da interferência e ruído do canal.
γ	Expoente de atenuação
$\mathrm{d_0, d_3}$	Distância de referência para $\mathrm{h_3}$
$\mathrm{P_{r3}}$	Potência recebida para $\mathrm{h_3}$
$\mathrm{P_{r2}}$	Potência recebida para $\mathrm{h_2}$
$\mathrm{P_{r1}}$	Potência recebida para $\mathrm{h_1}$
δ_2	Distância relativa de $\mathrm{h_2}$ para $\mathrm{h_3}$
δ_1	Distância relativa de $\mathrm{h_1}$ para $\mathrm{h_3}$

SNR_2	SNR para h_2
SNR_1	SNR para h_1

SUMÁRIO

1 INTRODUÇÃO

As redes de energia convencionais estão mudando e a demanda de energia cresce cada vez mais a cada ano. Sistemas mais antigos e obsoletos estão sobrecarregados e necessitam de ferramentas de monitoramento e controle mais eficientes (AMIN e WOLLENBERG, 2005; FARHANGI, 2010). Sistemas mais recentes observam cada vez mais a descentralização das fontes geradoras de energia e se mostram mais robustos. Segundo Amin e Wollenberg (2005), Gungor et al. (2010), Bilgin et al. (2016), a *Smart Grid* surgiu como um conjunto de ideias que visam otimizar operação e facilitar a comunicação desde a geração até o consumo nessas redes. A concessionária, por exemplo, pode se comunicar mais facilmente com o consumidor, e este tem uma participação mais ativa na tomada de decisões; ele passa a monitorar o seu próprio consumo com mais facilidade e pode atuar como gerador de energia com um sistema fotovoltaico instalado em sua residência.

Como alternativa aos sistemas de comunicação das *smart grids*, as Redes de Sensores sem Fio (RSSF) (TUNA et al. 2013; GARG, 2014; AKYILDIZ et al. 2002) vêm sendo cada vez mais utilizadas e se apresentam como uma tecnologia promissora devido ao seu baixo custo de implementação e sua composição. Elas atuam monitorando inúmeros parâmetros das redes e coletando dados de interesse, mas por utilizarem o canal *wireless* para transmissão de informação, elas são suscetíveis a interferências e ruídos comuns do meio, e às variações ambientais. Além disso, as limitações físicas dos sensores também influenciam no seu desempenho.

Uma informação transmitida de forma incorreta pode resultar em diagnósticos errados ou diagnósticos falsos. Por exemplo, os dados coletados de um equipamento crítico com sinais de falha em uma subestação podem ser recebidos no centro de controle com métricas positivas devido a interferências no meio de transmissão, o que não reflete o estado real do equipamento. Ou então, leituras de energia erradas de um medidor podem resultar em prejuízos ao consumidor e a subestação. Portanto é imprescindível que se minimize a ocorrência de erros com a utilização de técnicas de correção de erros.

Diante disso, a teoria de codificação de canal e os códigos corretores de erro se mostram uma ferramenta essencial para dar confiabilidade na transmissão da informação em *Smart Grid*. Os códigos corretores de erro possibilitam que a

comunicação entre o transmissor e o receptor ocorra com mais confiabilidade por meio da adição de redundância nas mensagens transmitidas (SHANNON, 1948; MOREIRA e FARRELL, 2006; HAYKIN, 2004) permitindo que a informação seja recuperada sem a necessidade de retransmissão.

Os principais códigos corretores de erros utilizados em RSSF são códigos baseados na retransmissão de pacotes na presença de erros na transmissão, ou códigos com baixa capacidade de correção de erros devido às limitações de processamento e energia dos nós sensores. Mas considerando que a decodificação dos pacotes transmitidos é feita ao final da transmissão e por dispositivos mais potentes, a utilização de códigos corretores de erro mais potentes como os códigos *Reed Solomon* (RS) e *Low Density Parity Check* (LDPC) torna-se viável.

O desempenho dos códigos pode ser diferente se a rede for composta apenas do transmissor e do receptor ou se a rede tiver dois, três ou muitos nós se sensores. Esse parâmetro influencia diretamente na viabilidade de um código corretor de erro em relação a outro, pois dependendo do que se deseja um código pode ser preferível quando comparado com outro. Os resultados também podem variar de acordo com a prioridade da transmissão de dados, seja o monitoramento em tempo real de parâmetros ou a confiabilidade da informação, com transmissão de dados condensados e diagnósticos.

Nesse sentido, este trabalho propõe uma análise de desempenho de diferentes códigos corretores de erro em redes de sensores sem fio, voltadas para aplicações de *Smart Grid*, levando em consideração as características e desafios da rede de acordo com técnicas de codificação de canal utilizadas na literatura, e visa obter parâmetros de rede ótimos em termos de taxa de erros de bits e tempo de processamento, demonstrar a viabilidade da utilização dos códigos corretores de erro com base nos resultados obtidos, avaliando os resultados voltados para aplicações de Smart Grid.

A metodologia adotada neste trabalho se trata de uma Pesquisa Exploratória, com coleta de dados por meio de pesquisa bibliográfica e a utilização de simulações para análise de dados.

O canal utilizado para a transmissão de informação e aplicação dos códigos corretores de erro foi o *wireless*. O desempenho dos códigos corretores de erro foi avaliado quando foram inseridas no canal interferências do tipo ruído branco

(*Additive White Gaussian Noise* – AWGN) (PROAKIS e SALEHI, 2008) e a atenuação do sinal com a variação da distância entre os nós.

Os códigos cujas performances foram analisadas são os códigos RS e os códigos LDPC, ambos amplamente utilizados hoje, e em diversas aplicações. Mas como mencionado na sessão 6, outros códigos também poderão ser estudados futuramente.

Além disso, serão avaliados parâmetros como taxa de erro de bits e tempo de processamento e também serão comentadas algumas aplicações de *Smart Grid*.

2 CODIFICAÇÃO DE CANAL

2.1 INTRODUÇÃO

Os canais de comunicação estão sujeitos a ruídos e interferências do ambiente, que introduzem erros durante a transmissão, corrompendo a informação transmitida. Tendo em vista a necessidade de transmitir informação sem erros, técnicas de detecção e correção de erros foram desenvolvidas para prevenir erros e recuperar a informação quando preciso.

Em 1948, Shannon (1948) demonstrou que é possível reduzir erros induzidos por um canal com ruídos a qualquer nível desejado sem sacrificar a taxa de transmissão da informação através da codificação da informação de forma apropriada. Assim tiveram início a Teoria da codificação e a Teoria da Informação, se consolidando com a introdução dos códigos de *Hamming* (HAMMING, 1950) e um pouco depois com a invenção dos códigos de *Galoy* (GALOY, 1949). Hoje, os códigos oferecem confiabilidade aos canais de comunicação e computadores, e são usados em diversas aplicações, como por exemplo, nas comunicações móveis, modems, internet, comunicação via satélite, rádio, nos aparelhos de armazenamento de dados (CDs, DVDs) e etc.

O controle de erros pode ser feito de duas formas. Na correção direta de erros (FEC – *Forward Error Correction*), adicionam-se bits de redundância na palavra-código transmitida, que permitem tanto a detecção como a correção de erros durante uma transmissão. Já na solicitação de repetição automática (ARQ - *Automatic Request for Retransmission*), adicionam-se bits de redundância na palavra-código apenas para detecção de erros. Depois de detectado o erro, o receptor solicita a retransmissão da palavra-código que foi corrompida (MOREIRA e FARREL, 2006).

2.2 CÓDIGOS CORRETORES DE ERROS

Os códigos corretores de erros podem ser classificados em dois tipos: códigos de blocos e cíclicos, e códigos convolucionais. A diferença está na utilização ou não de codificadores com memória. Os dois primeiros processam a informação

bloco a bloco de maneira independente e não possuem memória, enquanto os códigos convolucionais, por exemplo, possuem memória.

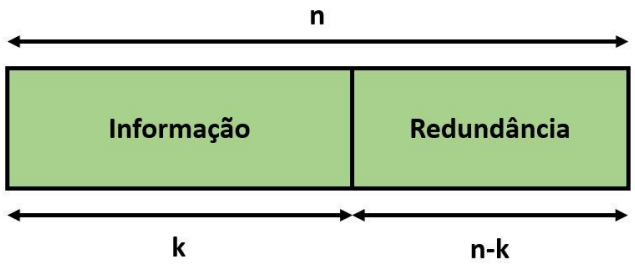
2.3 CÓDIGOS DE BLOCOS LINEARES

Uma mensagem ou informação é geralmente transmitida em dígitos binários ou bits, ou seja, 0 e 1. Chama-se uma sequência de bits de *palavra*. Um código binário, por exemplo, é um conjunto de palavras. E mais especificamente, as palavras-códigos são as palavras pertencentes a esses códigos binários.

Diz-se que um código é linear se duas palavras-códigos quaisquer do código puderem ser somadas em aritmética módulo 2 para produzir uma terceira palavra-código (HAYKIN, 2004). O codificador de um código de blocos linear (n, k) divide a informação em uma sequência de blocos com n bits de tamanho cada, sendo que k bits dos n bits do código são idênticos à sequência de mensagem a ser transmitida (HAYKIN, 2004; LIN E COSTELO, 1983). Existe um total de 2^k mensagens distintas. Os $n - k$ bits do código são bits de paridade ou redundância que são adicionados ao código de acordo com regras de codificação predefinidas.

A adição dessa redundância é que garante que no decodificador a informação possa ser recuperada. Chama-se de *códigos sistemáticos* os códigos em que os bits de informação são transmitidos de forma inalterada, e os bits de paridade aparecem no final ou no começo da palavra codificada, como mostrado na Figura 1.

Figura 1 - Palavra-código em formato Sistemático.



Fonte: Elaborada pelo autor (2018).

Para um código linear sistemático (n, k), a codificação consiste na multiplicação de uma matriz geradora $\mathbf{G}$, composta de duas sub-matrizes, uma

matriz identidade $k \times k$, indicada por I_k e uma sub-matriz de paridade $k \times (n-k)$, indicada por $\mathbf{P}$ (gerada aleatoriamente ou através de alguma análise definida), tal que (MORELOS-ZARAGOZA, 2002),

$$G = [I_k P], \tag{1}$$

por um vetor $\mathbf{u}$ de k bits de informação, produzindo uma palavra-código resultante $\mathbf{v}$ de n bits.

Por exemplo, considerando um código linear $(7,4)$, com a seguinte matriz geradora:

$$G = \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} 1000110 \\ 0100011 \\ 0010111 \\ 0001101 \end{bmatrix}, \tag{2}$$

e a informação a ser codificada como sendo $\mathbf{u} = (1101)$, a palavra-código resultante então seria,

$$v = 1.g_0 + 1.g_1 + 0.g_2 + 1.g_3 = (1000110) + (0100011) + (0001101) = (1101000). \tag{3}$$

A redundância, nesse caso, composta por $n - k$ bits é igual a (000).

O código linear $(7,4)$, com a redundância adicionada no final de todas as possíveis palavras-código, para esse exemplo, é mostrado na Tabela 1.

Tabela 1 - Código de bloco linear com $k = 4$ e $n = 7$.

Informação	Palavra-código
(0000)	(0 0 0 0 0 0 0)
(1000)	(1 0 0 0 1 1 0)
(0100)	(0 1 0 0 0 1 1)
(1100)	(1 1 0 0 1 0 1)
(0010)	(0 0 1 0 1 1 1)
(1010)	(1 0 1 0 0 0 1)
(0110)	(0 1 1 0 1 0 0)
(1110)	(1 1 1 0 0 1 0)
(0001)	(0 0 0 1 1 0 1)
(1001)	(1 0 0 1 0 1 1)
(0101)	(0 1 0 1 1 1 0)
(1101)	(1 1 0 1 0 0 0)
(0011)	(0 0 1 1 0 1 0)
(1011)	(1 0 1 1 1 0 0)
(0111)	(0 1 1 1 0 0 1)
(1111)	(1 1 1 1 1 1 1)

Para cada matriz $\mathbf{G}$ de $k \times n$, existe uma matriz $(n - k) \times k$ denominada de matriz $\mathbf{H}$ ou matriz de verificação de paridade que é ortogonal a matriz $\mathbf{G}$. A matriz $\mathbf{H}$ é utilizada no processo de decodificação dos códigos de blocos lineares, ou seja, é usada para verificar se a palavra-código recebida possui erros ou não após a transmissão. Se considerando uma palavra recebida $\mathbf{r}$ e $r.H^T$ for diferente de zero, então $\mathbf{r}$ não é palavra-código e, portanto, erros ocorreram na transmissão. Ou seja, $\mathbf{r}$ só é uma palavra-código de um código gerado por $\mathbf{G}$, se e somente se (LIN E COSTELO, 1983),

$$r.H^T = 0 \tag{4}$$

A essa verificação dá-se o nome de síndrome.

A síndrome é considerada o primeiro passo da decodificação no receptor de qualquer bloco linear. O restante do processo de decodificação é realizado através de diferentes métodos de decodificação, de acordo com cada código específico.

2.4 CÓDIGOS CÍCLICOS

Os códigos cíclicos compõem uma subclasse dos códigos de blocos lineares. Eles são fáceis de codificar e possuem uma estrutura matemática bem definida.
Um código é dito cíclico se exibir as duas propriedades a seguir (LIN E COSTELO, 1983):

(a) Propriedade da linearidade: a soma de duas palavras-código quaisquer do código também é uma palavra-código.

(b) Propriedade cíclica: qualquer deslocamento cíclico em uma palavra-código também é uma palavra-código.

Os códigos cíclicos são representados na forma polinomial, como se pode ver nos exemplos da Tabela 2. Essa característica permite que sejam algebricamente mais fáceis de analisar e programar.

Tabela 2 - Representação polinomial de palavras-código.

Palavras-código	**Polinômio $\mathbf{c(X)}$**
0000	0
1010	$1 + X^2$
0101	$X + X^3$
1111	$1 + X + X^2 + X^3$

Similar aos códigos de blocos lineares, a codificação é feita através de um polinômio gerador que define o código cíclico e assim como a matriz geradora, definidos por (HAYKIN, 2004; LIN E COSTELO, 1983):

$$g(X) = 1 + g_1 X + \cdots + g_{n-k-1} X^{n-k-1} + X^{n-k}, \tag{5}$$

onde os coeficientes g_i são iguais a 1 e 0.

Considerando um polinômio de mensagem $m(X)$, a codificação é realizada através primeiro da multiplicação do polinômio de mensagem $m(X)$ por X^{n-k}, depois do cálculo do resto $b(X)$ da divisão de $X^{n-k}m(X)$ por $g(X)$, que corresponde aos bits de redundância e por fim, na adição de $b(X)$ a $X^{n-k}m(X)$.

Por exemplo, considere um código cíclico $(7,4)$, com polinômio gerador $g(X) = X^3 + X + 1$ e a mensagem a ser codificada $m(X) = (0101) = X^2 + 1$. Seguindo os

passos da codificação, primeiro multiplica-se $X^3 m(X) = X^5 + X^3$ e depois se calcula o resto $b(X)$, fazendo a divisão polinomial de $X^5 + X^3$ por $g(X)$,

$$c(X) = X^3 m(X) + b(X) = X^5 + X^3 + X^2 = (0101100), \quad (6)$$

Entre os mais importantes códigos cíclicos estão os códigos CRC, que apenas detectam erros, os códigos BCH (*Bose-Chaudhuri-Hocquenghem*) que são códigos binários capazes de corrigir apenas um erro e os códigos Reed-Solomon, que são códigos não binários que operam em múltiplos bits, e que será apresentado na próxima seção.

2.5 CÓDIGOS *REED SOLOMON* (RS)

Dentro os códigos possíveis de serem testados, encontram-se os códigos *Reed Solomon*. Os códigos *Reed Solomon* (RS) (REED E SOLOMON, 1960) surgiram em 1960 por Irving Reed e Gus Solomon. São códigos cíclicos não binários que operam em múltiplos bits sobre um campo finito denominado como Campo de Galois (*Galois Field* - GF) $GF(2^m)$, sendo m o número de bits. Um Campo de Galois é um campo finito de p elementos, designado por $GF(p)$, em que se podem executar operações de adição e multiplicação (LIN E COSTELO, 1983).

Os códigos RS na forma sistemática são especificados como $RS(N, K)$, onde N é definido como o número de símbolos da palavra-código (codificados) e K é definido como o número de símbolos a serem codificados (símbolos de informação ou mensagem). O número de símbolos de redundância é definido com $R = N - K$. É importante observar que a notação foi modificada em comparação à descrição dos códigos de blocos lineares, porque os códigos RS trabalham com múltiplos bits (símbolos) e não mais com bits individuais.

A capacidade de correção dos códigos RS é definida como $t = R/2$, ou seja, como a metade da quantidade de símbolos de redundância adicionados na codificação.

Assim como nos códigos de blocos lineares, as palavras-código são representadas na forma de polinômios, no entanto os coeficientes são definidos no Campo de Galois $GF(2^m)$. Ou seja, cada um dos 2^m elementos do campo, formados

por um conjunto de 0 e 1 (múltiplos bits) são representados por uma potência de β, sendo β um elemento primitivo do campo (SKLAR, 2001).

$$\mathrm{GF}(2^{\mathrm{m}}) = \{0,1,\beta^1,\beta^2,\beta^3,\ldots,\beta^{2\mathrm{m}-2}\} \quad (7)$$

Cada um dos 2^m elementos do Campo podem ser representados por um polinômio de grau $\mathrm{m}-1$ ou menos. Além disso, os elementos do Campo de Galois são gerados através de um chamado *polinômio primitivo*. Um polinômio primitivo $\mathrm{h}(\mathrm{X})$ é um polinômio de grau m que tem o elemento primitivo β como raiz, ou seja, $\mathrm{h}(\beta) = 0$ (MOREIRA E FARREL, 2006).

Como exemplo de construção de um Campo de Galois, considere um Campo de Galois $\mathrm{GF}(2^3)$ com $2^3 = 8$ elementos. Usando o polinômio primitivo $\mathrm{h}(\mathrm{X}) = \mathrm{X}^3 + \mathrm{X}^2 + 1$, pode-se obter a representação polinomial dos elementos fazendo, $\mathrm{X}\mathrm{mod}\mathrm{h}(\mathrm{X})$, resultando na segunda coluna da Tabela 3. A terceira coluna da Tabela 3 representa o polinômio na forma binária gerando a palavra-código. A representação polinomial de palavras-código pode ser encontrada na Tabela 3.

Tabela 3 - Campo de Galois $\mathrm{GF}(2^3)$ gerado pelo polinômio primitivo $\mathrm{h}(\mathrm{x}) = \mathrm{X}^3 + \mathrm{X}^2 + 1$.

Potência em β^1	**Polinômio em $x^i mod h(x)$**	**Palavra-código**
0	0	000
$\beta^0 = 1$	1	001
β^1	X	010
β^2	X^2	100
β^3	$\mathrm{X}^2 + 1$	101
β^4	$\mathrm{X}^2 + \mathrm{X} + 1$	111
β^5	$\mathrm{X} + 1$	011
β^6	$\mathrm{X}^2 + \mathrm{X}$	110

Um exemplo simples da geração de um elemento no $\mathrm{GF}(2^3)$ pode ser visto considerando, por exemplo, o cálculo de β_3.

$$\beta^3 = \mathrm{X}^3 \mathrm{mod}(\mathrm{X}^3 + \mathrm{X}^2 + 1) = \mathrm{X}^2 + 1 \quad \text{(resto da divisão)} \quad (8)$$

A prova do polinômio primitivo, assim como um exemplo de como funciona a operação de soma no Campo de Galois pode ser visto a seguir:

$$h(\beta) = X^3 + X^2 + 1 = \beta^3 + \beta^2 + 1 = X^2 + 1 + X^2 + 1 = 0 \quad (9)$$

Uma lista com diversos Campos de Galois para $3 < m < 10$ pode ser encontrada em (LIN E COSTELO, 1983).

Os códigos RS são ótimas opções para aplicações que exigem estabilidade, pouca latência, no entanto, sem um alto poder de correção.

2.5.1 Codificação

Na codificação dos Códigos RS na forma sistemática, como já visto anteriormente, os bytes ou símbolos de redundância chamados aqui de $R = n - k$, são acrescentados aos bytes de informação k gerando uma palavra-código RS de tamanho $n = k + R$. Os bytes de redundância ou paridade são o resto da divisão polinomial da mensagem $M(X)$ de tamanho k, por um polinômio gerador $G(X)$. A palavra-código RS na forma polinomial pode ser expressa como (LIN E COSTELO, 1983):

$$C(X) = M(X)X^R \mathrm{mod} G(X) \quad (10)$$

Segue um exemplo para entender a codificação RS usando Campos de Galois.

Considere o código $RS(7,5)$, com elementos do $GF(2^3)$ e a mensagem a ser transmitida com tamanho $k - 5$ como, $(001\ 101\ 111\ 010\ 011)$, e na forma polinomial usando a Tabela 3: $M(X) = X^4 + \beta^3 X^3 + \beta^4 X^2 + \beta X + 5$, com o polinômio gerador definido por: $G(X) = X^2 + \beta^6 X + \beta^3$.

$$X^R M(X) = X^2 M(X) \quad (11)$$

$$X^2[X^4 + \beta^3 X^3 + \beta^4 X^2 + \beta X + \beta^5] \quad (12)$$

$$X^6 + \beta^3 X^5 + \beta^4 X^4 + \beta X^3 + \beta^5 X^2 \quad (13)$$

O polinômio $X^2M(X)$ é então dividido por: $G(X)$, resultando no resto da divisão $r(X) = \beta^6X + 1$. O polinômio $r(X)$ representa a paridade a ser adicionada no final do polinômio mensagem $X^2M(X)$, resultando no polinômio da mensagem codificada RS $C(X)$:

$$C(X) = X^6 + \beta^3X^5 + \beta^4X^4 + \beta X^3 + \beta^5X^2 + \beta^6X + 1 \quad (14)$$

Resultando na palavra código: (001 101 111 010 011 110 001).

2.5.2 Decodificação

A decodificação dos códigos RS é uma operação bem mais complexa que a codificação, passando por várias etapas. Mais precisamente, o processo de decodificação pode ser dividido em 4 etapas[1]:

1) Calcular as síndromes (SKLAR, 2001);

2) Determinar o polinômio localizador de erros (algoritmo de *Berlekamp-Massey*) (BERLEKAMP,1965, MOREIRA E FARREL, 2006, LIN E COSTELO, 1983);

3) Determinar as raízes do polinômio localizador de erros *(Chien Search)* (MOREIRA E FARREL, 2006, LIN E COSTELO, 1983);

4) Calcular a magnitude dos erros (algoritmo de *Forney)* (MOREIRA E FARREL, 2006, LIN E COSTELO, 1983).

De posse da posição e do valor dos erros é possível definir o polinômio de erros que será somado com a palavra recebida, resultando na palavra transmitida.

Caso a quantidade de erros de uma palavra recebida for maior do que a capacidade de correção de erros do código RS, a decodificação não pode ser realizada e nenhum erro da palavra recebida é corrigido, ou seja, a palavra permanece inalterada após passar pela decodificação do RS.

1 Detalhes sobre os cálculos e implementações das etapas citadas podem ser encontrados nas referências citadas ao lado.

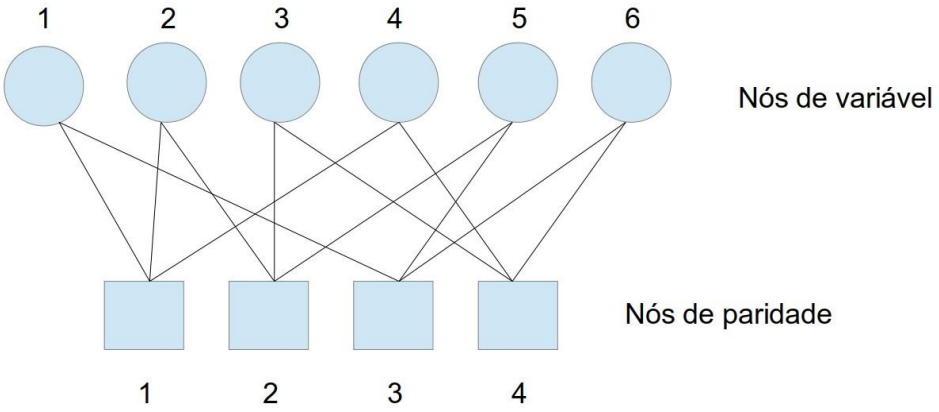
2.6 CÓDIGOS *LOW DENSITY PARITY-CHECK* (LDPC)

Os códigos LDPC (*Low-Density Parity-Check*), ou códigos de verificação de paridade de baixa densidade são códigos com alto poder de correção que surgiram em 1960 com a tese de doutorado de Gallager (GALLAGER, 1963). Devido ao poder de processamento limitado da época, os códigos LDPC permaneceram esquecidos por quase 35 anos. Durante esse período, apenas o trabalho de R. Michael Tanner em 1981 (TANNER, 1981) que generalizou os códigos LDPC e introduziu a representação gráfica do código, mais tarde denominada de Grafo de Tanner, teve importância. Apenas na década de 90, os códigos LPDC foram redescobertos por McKay e Neal (MCKAY, 1999).

São códigos de blocos lineares caracterizados pela sua matriz de verificação de paridade $\mathbf{H}$, que deve ser esparsa, ou seja, possuir uma baixa densidade de 1's. Além dessa condição de esparsidade, um código LDPC por si só não é diferente de nenhum outro código de bloco linear. A grande diferença está no fato dos códigos LDPC utilizarem algoritmos de decodificação iterativa que funcionam bem quando a matriz de verificação de paridade é esparsa.

A matriz de verificação de paridade de um código LDPC pode ser representada na forma de um gráfico bipartido, ou grafo de Tanner (TANNER, 1981). Tanner considerou que qualquer código de bloco linear poderia ser representado por um gráfico bipartido. Um gráfico bipartido é formado por nós de dois tipos, que se conectam apenas aos nós de tipo diferente. Os nós de variável (ou nós de bit) são representados por círculos, e os nós de paridade, por quadrados, como pode ser visto na Figura 2.

Figura 2 - Grafo de Tanner.



Fonte: Elaborada pelo autor (2019)

O grafo possui ramos ligando os nós de variável aos nós de paridade apenas quando o valor do elemento $h_{i,j}$ na matriz de verificação de paridade 1. A matriz de verificação de paridade $\mathbf{H}$, de tamanho $p \times n$, correspondente ao gráfico da Figura 2 é dada por:

$$\mathbf{H} = \begin{bmatrix} 110100 \\ 011010 \\ 100011 \\ 001101 \end{bmatrix} \tag{15}$$

na qual as colunas da matriz $\mathbf{H}$ correspondem aos nós de variável, e as linhas correspondem aos nós de paridade.

As matrizes de verificação de paridade dos códigos aleatórios geralmente não possuem uma estrutura. Essa falta de estrutura dificulta a codificação e a decodificação, deixando-as complexas. Recentemente, códigos com estrutura, como os códigos Cíclicos ou Quasi-Cíclicos, foram criados, simplificando a codificação e a decodificação. Os códigos Quasi-Cíclicos serão os usados neste trabalho e serão descritos mais adiante.

A vantagem dos códigos LDPC para os outros códigos de blocos lineares está justamente na sua decodificação iterativa que permite uma alta confiabilidade, no entanto, introduzindo um excesso de processamento e, portanto, uma alta latência para o sistema. Os códigos LDPC estão em praticamente todas as aplicações de sistemas de comunicação modernas como *WiFi* e *WiMAX* justamente pela alta

capacidade de correção de erros, podendo se aproximar da chamada capacidade de Shannon, ou seja, o valor da quantidade máxima de informação livre de erros que pode ser transferida em um canal de comunicação.

2.6.1 Codificação

Uma das formas de se realizar a codificação de códigos LDPC é através da matriz geradora $\mathbf{G}$, assim como na codificação dos códigos de blocos lineares convencionais. No entanto, esse método de codificação não é viável para os códigos LDPC, pois estes geralmente possuem comprimento de palavra-código longo, e pelo fato de $\mathbf{G}$ não ser esparsa, a complexidade da operação de codificação é da ordem de n^2, na qual n é o tamanho da palavra-código. Outros métodos podem ser usados que não utilizam a matriz $\mathbf{G}$, e sim a matriz de verificação de paridade $\mathbf{H}$, se aproveitando da sua esparsidade, como, por exemplo, o método de codificação com complexidade quase linear proposto por Richardson e Urbanke, no qual mais detalhes podem ser encontrados em (RICHARDSON E URBANKE, 2001).

2.6.2 Decodificação

Uma das grandes vantagens dos códigos LDPC está na sua decodificação iterativa, que possibilita ao código apresentar um excelente desempenho para diferentes tipos de canais. A decodificação iterativa é baseada na troca de mensagens no grafo de Tanner, onde as mensagens passam de trás para a frente entre os nós de variável e de paridade, iterativamente, até que um bom resultado seja alcançado. Os algoritmos chamados de *message-passing* são um dos algoritmos usados na decodificação iterativa dos códigos LDPC, podendo ser classificados como (JOHNSON, 2009):

1) Algoritmo *Bit-flipping* (Decisão abrupta): utiliza mensagens binárias. Uma decisão binária sobre cada bit recebido é passada ao decodificador. O algoritmo para quando uma palavra-código válida é encontrada.

2) Algoritmo *Belief-Propagation* (Decisão Suave): as mensagens são funções de probabilidades que representam o nível de confiança sobre o valor dos bits da palavra-código. Os valores probabilísticos podem ser representados como razões de verossimilhança logarítmica (*log-likelihood ratios* (LLRs)), permitindo que os cálculos

sejam feitos através de soma e produto. Enquanto probabilidades precisam ser multiplicadas, LLRs só precisam ser adicionadas, reduzindo a complexidade.

O algoritmo utilizado nesse trabalho será o algoritmo *belif propagation* que apresenta um melhor desempenho.

2.6.3 Códigos LDPC Quasi-Cíclicos (QC-LDPC)

Os códigos LDPC utilizados neste trabalho são os chamados *quasi-cyclic LDPC block codes* (QC-LDPC-BC) (MYUNG, 2005). Os códigos QC-LDPC-BC são encontrados em praticamente todas as implementações utilizando códigos LDPC, devido a sua codificação eficiente, como por exemplo, no padrão G.hn (ITU G.9960, 2010), padrão que unifica as redes domésticas, e no padrão IEEE802.11n (IEEE P802.11n, 2009), padrão para redes sem fio locais.

Os códigos QC-LDPC pertencem a uma classe de códigos LDPC cuja matriz de verificação de paridade consiste em sub-matrizes rotativas de blocos. Um código QC-LDPC com taxa $R = \mathrm{K}/\mathrm{N}$, em que N é o número de bits codificados e K o número de bits de informação, é definido por uma matriz de verificação de paridade $(\mathrm{N}-\mathrm{K}) \times \mathrm{N}$ consistindo num conjunto $\mathrm{c} \times \mathrm{t}$ de $\mathrm{b} \times \mathrm{b}$ sub-matrizes, onde $\mathrm{b} = \mathrm{N}/\mathrm{t}$:

$$\mathbf{H} = \begin{bmatrix} \mathrm{P}_{1,1} & \mathrm{P}_{1,2} & \cdots & \mathrm{P}_{1,\mathrm{t}} \\ \mathrm{P}_{2,1} & \mathrm{P}_{2,2} & \cdots & \mathrm{P}_{2,\mathrm{t}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{P}_{\mathrm{c},1} & \mathrm{P}_{\mathrm{c},2} & \cdots & \mathrm{P}_{\mathrm{c},\mathrm{t}} \end{bmatrix} \tag{16}$$

Cada $\mathrm{b} \times \mathrm{b}$ sub-matriz $\mathrm{P}_{\mathrm{i,j}}$ ou é uma matriz com apenas zeros, ou é uma matriz identidade rotativa. Uma matriz identidade rotativa é uma matriz identidade que pode ter suas colunas deslocadas ciclicamente para a direita de um valor inteiro.

A matriz de verificação de paridade $\mathbf{H}$ pode ser também expressa na sua forma compacta $\mathbf{H_c}$, sendo descrita como:

$$\mathbf{H_c} = \begin{bmatrix} \mathrm{p}_{1,1} & \mathrm{p}_{1,2} & \cdots & \mathrm{p}_{1,\mathrm{t}} \\ \mathrm{p}_{2,1} & \mathrm{p}_{2,2} & \cdots & \mathrm{p}_{2,\mathrm{t}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{p}_{\mathrm{c},1} & \mathrm{p}_{\mathrm{c},2} & \cdots & \mathrm{p}_{\mathrm{c},\mathrm{t}} \end{bmatrix} \tag{17}$$

Na forma compacta, uma sub-matriz com apenas zeros é especificada com $p_{i,j} = -1$, e uma sub-matriz identidade rotativa é especificada por um inteiro $p_{i,j}$ entre 0 e $b - 1$, o qual define o número de deslocamentos para direita das colunas da matriz identidade. A própria matriz identidade é especificada por $p_{i,j} = 0$.

As matrizes **H** na forma compacta do padrão G.hn foram as utilizadas nesse trabalho, para as taxas de código de 1/2 e 2/3. As matrizes utilizadas para as duas taxas de código, 1/2 e 2/3 são mostradas, respectivamente, nas Figuras 3 e 4.

Figura 3 - Matriz **H** do padrão G.hn, e taxa de código ½ (c=12; t=24; b=80) para N=1920 bits, K=960 bits.

27	-1	-1	-1	55	19	-1	30	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	0	-1	1	-1	70	-1	47	-1	62	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	41	-1	-1	-1	44	-1	-1	59	60	25	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1
16	77	-1	-1	-1	5	-1	48	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1
-1	-1	-1	45	-1	27	-1	46	19	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1
-1	-1	63	-1	-1	-1	55	-1	-1	-1	48	26	10	-1	-1	-1	-1	0	0	-1	-1	-1	-1
-1	-1	-1	42	-1	21	-1	58	-1	41	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1	-1
-1	-1	-1	-1	78	0	-1	7	52	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1
-1	29	9	-1	-1	-1	37	-1	-1	-1	35	21	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1
-1	-1	22	72	-1	-1	47	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1
35	-1	-1	-1	-1	13	-1	35	-1	70	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0
-1	46	28	-1	-1	-1	38	-1	-1	-1	8	-1	10	58	-1	-1	-1	-1	-1	-1	-1	-1	-1	0

Fonte: (ITU G.9960, 2010)

Figura 4 - Matriz **H** do padrão G.hn, taxa de código 2/3 (c=8; t=24; b=270), para N=6480 bits, k=4320 bits.

78	-1	-1	167	237	-1	3	-1	266	-1	-1	102	153	-1	-1	212	-1	0	-1	-1	-1	-1	-1	-1
-1	83	189	-1	-1	68	-1	178	-1	90	205	-1	-1	13	4	-1	-1	0	0	-1	-1	-1	-1	-1
-1	226	147	-1	46	-1	-1	76	-1	116	-1	211	-1	112	-1	118	-1	-1	0	0	-1	-1	-1	-1
92	-1	-1	214	-1	236	241	-1	157	-1	143	-1	214	-1	207	-1	-1	-1	-1	0	0	-1	-1	-1
144	-1	-1	258	264	-1	53	-1	114	-1	172	-1	-1	82	262	-1	62	-1	-1	-1	0	0	-1	-1
-1	153	120	-1	-1	199	-1	126	-1	61	-1	183	15	-1	-1	134	-1	-1	-1	-1	-1	0	0	-1
-1	100	-1	141	-1	36	-1	17	-1	156	-1	124	162	-1	-1	57	0	-1	-1	-1	-1	-1	0	0
196	-1	187	-1	73	-1	80	-1	139	-1	57	-1	-1	236	267	-1	62	256	-1	-1	-1	-1	-1	0

Fonte: (ITU G.9960, 2010)

3 *SMART GRID*

3.1 DEFINIÇÕES

As *Smart Grid* (Redes Inteligentes em português) são infraestruturas modernas de energia que oferecem mais eficiência, segurança e confiabilidade (AMIN e WOLLENBERG, 2005; GUNGOR et al., 2010). Elas surgiram a partir da urgência em resolver problemas do Sistema Elétrico de Potência (SEP) convencional, que vem aumentando sua dimensão progressivamente e se tornando cada vez mais complexo.

O SEP existente hoje é um produto da rápida urbanização e do crescimento da população. As suas estruturas são altamente interconectadas, e uma mudança em qualquer local pode causar impactos imediatos em uma área imensa, e seus efeitos se propagam pela rede quase imediatamente (AMIN e WOLLENBERG, 2005; FARHANGI, 2010). O SEP também sofre com o fato de que os sistemas de proteção são limitados e aplicados localmente, e os sistemas de supervisão e controle podem responder com atraso.

Durante anos, imensas estruturas e usinas elétricas vêm sendo construídas para suprir a demanda crescente de energia, mas elas vêm acompanhadas de perdas de energia nas conexões de transmissão e distribuição, principalmente.

Diante disso, surgiu a ideia de *Smart Grid.* Segundo Amin e Wollenberg (2005), Farhangi (2010), *Smart Grid* é uma coleção de tecnologias, conceitos, topologias e abordagens que permitem que as hierarquias de geração, transmissão e distribuição sejam substituídas por um cenário inteiramente integrado, orgânico e inteligente, em que os objetivos e as necessidades de todas as partes são atendidas por meio da transmissão eficiente de dados e serviços.

Portanto, uma *Smart Grid* é uma rede que acomoda uma variedade de opções de geração, e.g. central, distribuída, intermitente e móvel. Ela permite que consumidores interajam e participem ativamente no gerenciamento de energia e redução de custos. Ela também possui características de *self-healing*, detecta falhas e age de acordo para evitar ou minimizar problemas. E como ela não pode simplesmente substituir o SEP convencional, eles devem coexistir, combinando as suas capacidades e funcionalidades.

As *Smart Microgrid* se derivam desse conceito. Visando o crescimento orgânico das *Smart Grid*, as *Smart Microgrid* são redes interconectadas de sistemas de energia que conseguem operar estando ou não ligadas ao SEP (FARHANGI, 2010). As suas principais características são:

1) Descentralização de fontes geradoras, permitindo que demandas locais sejam atendidas e o excesso gerado seja injetado no SEP. Maior integração com fontes de energias renováveis tais como fotovoltaico, solar e biomassa;

2) Melhor gerenciamento de energia, viabilizando o atendimento de diversos tipos de carga, incluindo residencial e industrial.

3) Utilização de medidores inteligentes e sensores capazes de monitorar diversos parâmetros, e.g. tensão, corrente, potência ativa, potência reativa, entre outros, com precisão e fidelidade.

4) Mais confiabilidade e segurança para a rede, com mais resistência contra sinistros e rápida recuperação após a ocorrência de faltas;

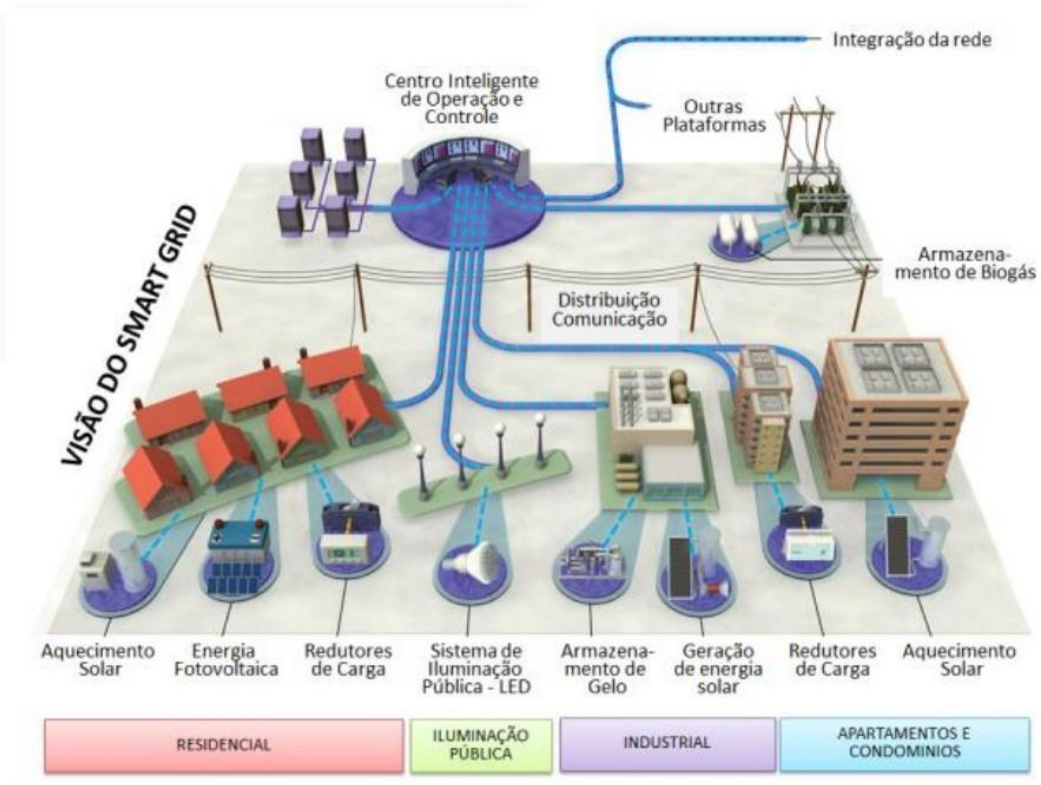
5) Incorpora uma infraestrutura de comunicação que permite que os componentes da rede troquem informações e comandos bidirecionalmente, de forma segura e confiável;

6) Integração com dispositivos inteligentes e aplicações residenciais;

7) Maior participação do consumidor no gerenciamento de consumo e custos de energia.

A Figura 5 apresenta uma visão geral dessa tecnologia e como ela integra diversos sistemas (NMENTORS, 2013).

Figura 5 – Visão geral de uma *smart grid.*



Fonte: (NMENTORS, 2013).

3.2 APLICAÇÕES DE *SMART GRID*

Como citado em 3.1, a *Smart Grid* engloba diversas características e componentes, como, por exemplo, a Medição Inteligente, que envolve a utilização de dispositivos para a leitura do consumo de energia, gás, água, entre outros, dos clientes (DOE, 2016).

As empresas de energia elétrica vêm substituindo os antigos medidores analógicos, que têm suas leituras feitas manualmente de mês a mês, por medidores inteligentes de alta tecnologia. Esses medidores são capazes de coletar automaticamente informações sobre o consumo de energia do consumidor e transmitir para a concessionária de energia (SCARPIN, 2018; SILVEIRA, 2019).

Os medidores inteligentes fornecem medições rápidas e precisas do consumo de energia, eliminando a necessidade de estimar as contas mensais dos clientes ou visitas para leitura de energia.

De acordo com Cao et al. (2008), os sistemas de medição podem ser divididos em sistemas cabeados, como, por exemplo, *Power Line Communications*

(PLC) e *Public Switch Telephone Network* (PSTN), e sistemas sem fio, tais quais o *Bluetooth*, *WiMax*, 3G, 4G, entre outros, de acordo com o meio de comunicação adotado. Os sistemas cabeados possuem várias desvantagens, como, por exemplo, a dificuldade de serem estendidos para longas distâncias, enquanto os sistemas sem fio adotam tecnologias avançadas de comunicação e são mais flexíveis.

A partir deste cenário, a medição inteligente passou a ter papel fundamental e foi considerada a “base” para a criação das redes elétricas inteligentes. Além de requisitos básicos, como leitura, corte e religação remotos, outros requisitos mais complexos também passaram a compor o seu conceito, como, por exemplo, informações de tensão e corrente em tempo real, de falta de energia, gestão e controle de perdas, ajuste da modalidade tarifária e maior interação com o consumidor (SCIAMANA, 2019):

Outra componente integral da *Smart Grid* é a Geração Distribuída (GD), que consiste na geração de energia mais próxima das cargas consumidoras, com independência total ou parcial das concessionárias – geração isolada ou ilhada – e que geralmente utilizam fontes renováveis de energia, tais quais a geração fotovoltaica, eólica, biomassa, entre outros (STAROSTA, 2017).

Por estarem localizados próximos das cargas consumidoras, esses sistemas de GD visam minimizar as perdas com transmissão de energia, e permitem que localidades não atendidas pelo SEP possam ter energia elétrica com sistemas de geração isolados. No caso de sistemas fotovoltaicos e eólicos, por exemplo, as fontes de energia são intermitentes, portanto, precisam de baterias para armazenar energia quando não estão conectados diretamente no SEP. Quando conectados à rede elétrica convencional, esses sistemas precisam atender aos requisitos de operação e segurança das concessionárias (STAROSTA, 2017).

3.2.1 **Informação em Tempo Real x Comunicação de Diagnósticos**

Um importante requisito dos sistemas de medição inteligente é o fornecimento de informações de consumo e gastos em tempo real ou próximo disso, incluindo também dados de falhas e falta de energia para a concessionária. Na geração e transmissão de energia elétrica, sensores podem ser utilizados para monitorar em tempo real dados de voltagem, corrente, temperatura, entre outros, de subestações,

transformadores, linhas de transmissão e unidades geradoras (KAYASTHA et al., 2012).

Dentro da *Smart Grid*, informações online e confiáveis do fornecimento de energia são de extrema importância, desde a geração até o consumo. O monitoramento online permite que falhas em equipamentos e acidentes sejam evitadas (GUNGOR et al., 2010).

Porém, a medição e coleta de informações em tempo real gera uma grande quantidade de dados que precisa ser processada, transmitida, analisada e armazenada posteriormente (DAKI et al., 2017). Pode-se, então, quando não houver urgência no recebimento de informação, ser feita a transmissão de informações condensadas e precisas, na forma de diagnósticos, reduzindo assim a quantidade de dados transmitidos e em intervalos maiores entre as transmissões. Dependendo da aplicação, pode-se ou não se exigir mais confiabilidade na informação transmitida, o que leva a escolha de códigos corretores de erro mais potentes ao invés de transmitir dados sem qualquer codificação.

Falhas de diagnóstico (falsos positivos ou falsos negativos) podem trazer grandes impactos para o SEP. Um diagnóstico errado ou falso positivo, por exemplo, pode levar a inspeções desnecessárias, comprometendo a disponibilidade de componentes da rede por um determinado período e, consequentemente, afetando uma grande quantidade de consumidores. Já um falso negativo pode, por exemplo, resultar em defeitos na rede elétrica por falta de inspeções ou manutenção (DE HAAS, 2011).

3.3 COMUNICAÇÃO EM *SMART GRID*

A infraestrutura de comunicação de uma *Smart Grid* inclui *Home Area Networks* (HAN), *Neighborhood Area Networks* (NAN), e *Wide Area Networks* (WAN), além de uma variedade de tecnologias, desde conexões cabeadas a redes sem fio (DOE, 2010; GRZEIDAK et al., 2011; KIM et al., 2010).

1) HAN: As HAN ficam localizadas próximos ao consumidor e proporcionam acesso a aplicações domésticas que permitem que o usuário final esteja ciente do seu consumo e custos com energia, além de outros dispositivos que executam tarefas diárias. As residências enviam as suas medições de energia pela HAN para dispositivos de medição localizados do lado de fora, e que serão transmitidos para

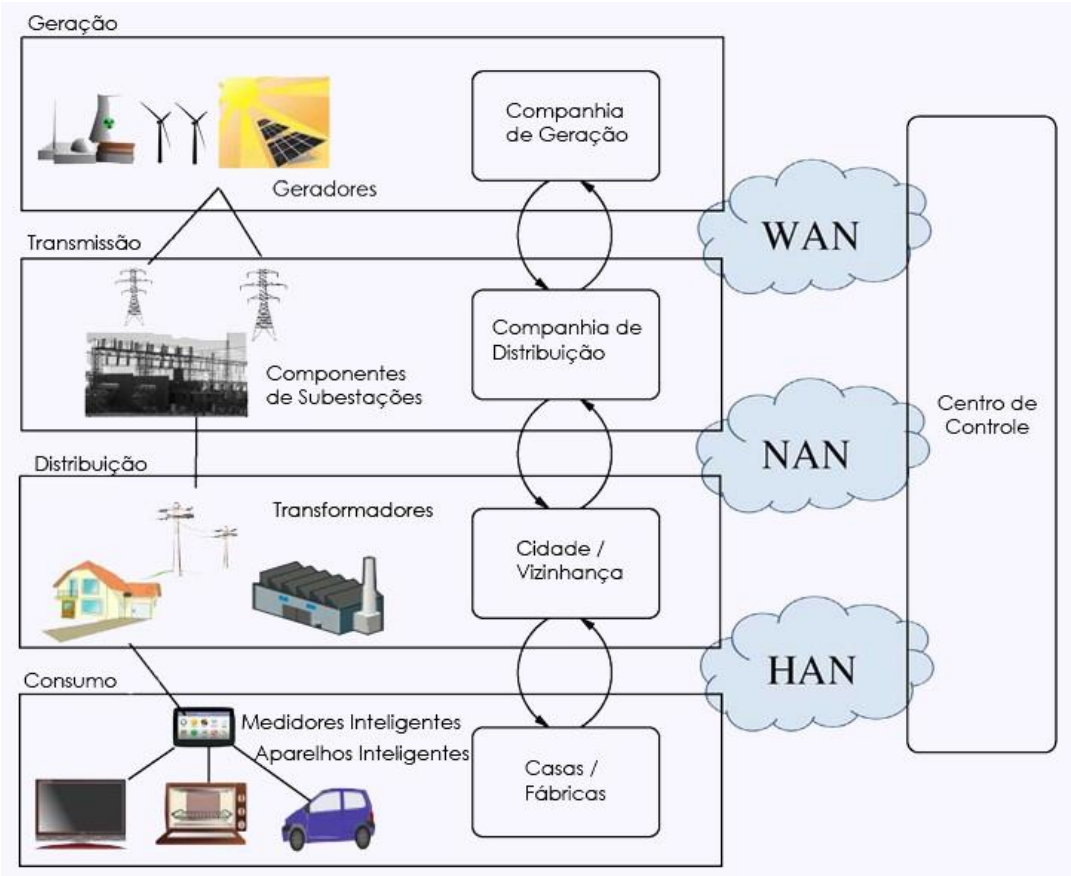
centros de controle posteriormente. São estruturas que não precisam de uma faixa de banda muito alta, até 100 Kbps, e não precisam de baixa de latência (GUNGOR et al., 2013). Algumas das tecnologias que atendem aos requisitos das HAN são o *Wi-Fi* e o *ZigBee* (LI e ZHANG, 2014). *Building Area Networks* (BAN) é o termo usado para se referir a redes parecidas, instaladas em empresas ou edificações, e *Industrial Area Networks* (IAN) para indústrias.

2) NAN: As NAN, ou também *Field Area Networks* (FAN), servem como ponte entre o consumidor e as subestações. Geralmente são descritas como as redes para áreas de distribuição de energia. As tecnologias utilizadas incluem *Ethernet*, fibra óptica, PLC e *WiMAX* (LI e ZHANG, 2014).

3) WAN: São redes de longas distâncias, com altas taxas de transferência, e que promovem a comunicação das concessionárias e subestações. As melhores tecnologias utilizadas são conexões cabeadas, *WiMAX* e redes de telefonia (LI e ZHANG, 2014).

A Figura 6 apresenta um demonstrativo do que foi comentado.

Figura 6 – Comunicação em *Smart Grid.*



Fonte: (DUARTE, 2012).

Muitos trabalhos vêm sendo desenvolvidos buscando principalmente formas inovadoras de redes de comunicação para as *Smart Grid* (YU e ANSARI, 2016). Por exemplo, os sistemas de comunicação para medição inteligente, dispositivos agregadores de dados, frameworks para detecção de atividades irregulares na Smart Grid, e as Redes de Sensores Sem Fio, que serão comentadas no capítulo a seguir.

4 REDES DE SENSORES SEM FIO (RSSF)

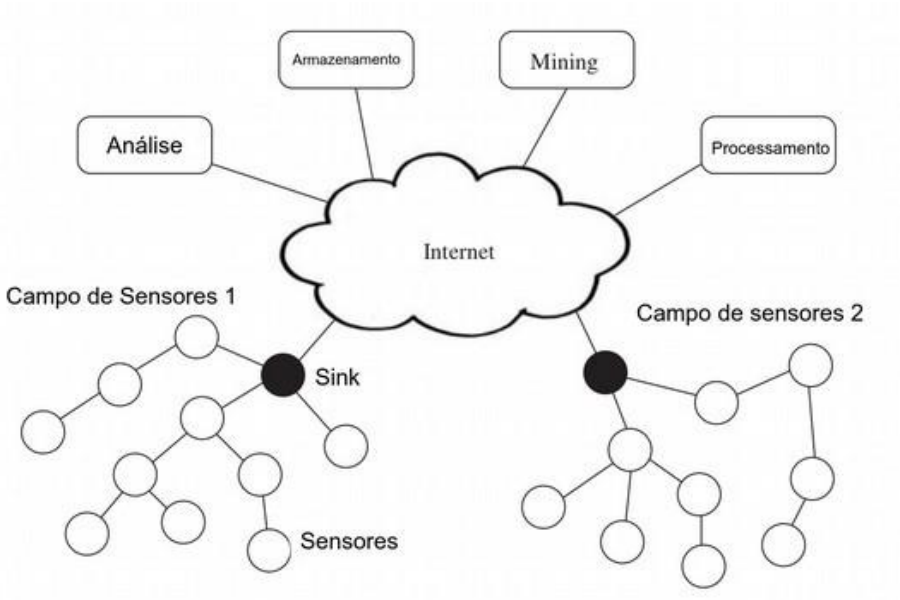
4.1 DEFINIÇÕES

A maior utilização da tecnologia de Sistemas Microeletromecânicos – do inglês *Microeletromechanical Systems* (MEMS) – e os recentes avanços da comunicação sem fio e eletrônica digital possibilitaram que as Redes de Sensores Sem Fio (RSSF) ganhassem atenção em todo o mundo, facilitando o desenvolvimento de nós sensores multifuncionais com baixo custo, baixa potência, pequenos em tamanho e capazes de se comunicar em curtas distâncias (YICK et al., 2008).

Segundo Akyildiz et al. (2002), Loureiro et al. (2003), Akyildiz & Vuran (2010), Prathap et al. (2012), uma RSSF é uma rede composta por diversos nós sensores, operando em conjunto para monitorar uma região e observar um determinado fenômeno. Os dados coletados são transmitidos para um nó sorvedouro (em inglês *sink*) que é responsável para transmitir para um servidor ou estação base para tratamento e análise (ROCHA FILHO, 2014). Um nó sensor sem fio tem, além de uma unidade de sensor, componentes de processamento, comunicação e armazenamento (DARGIE e POELLABAUER, 2010).

A Figura 7 mostra um esquema simplificado de uma RSSF.

Figura 7: Esquema simplificado de uma RSSF.



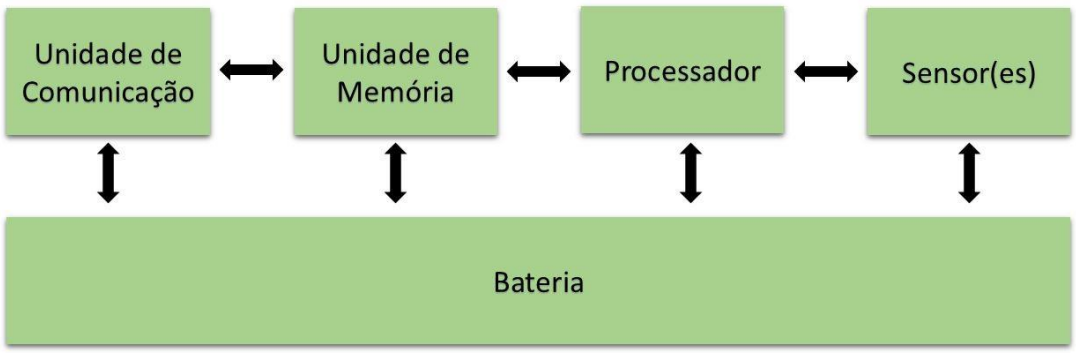
Fonte: (DARGIE e POELLABAUER, 2010), adaptado.

Nós sensores são aparelhos de baixa potência e baixo custo, equipados com um ou mais sensores internamente, um processador, uma unidade de memória, uma fonte de energia, rádio e, não obrigatoriamente, um atuador, que pode ser usado para ajustar e monitorar parâmetros dos próprios sensores, entre outras utilidades. Inúmeros tipos de sensores, como por exemplo de temperatura, magnéticos, biológicos, termais, ópticos, entre outros, também podem ser acoplados aos sensores nodais para medir parâmetros ambientais (GARG, 2014; AKYILDIZ et al. 2002).

Como esses sensores têm capacidade limitada de memória e podem ser utilizados em locais de difícil acesso, eles possuem uma unidade de rádio para comunicação *wireless* com a unidade base de aquisição de dados (GARG, 2014; AKYILDIZ et al. 2002).

A bateria é a principal fonte de energia do dispositivo. Também podem ser instaladas fontes de potência secundárias como painéis solares, por exemplo, se as condições ambientais forem favoráveis. As Figuras 8 e 9 apresentam a composição esquemática dos componentes de um nó sensor, e um exemplo de nó sensor, respectivamente.

Figura 8: Composição em diagrama de blocos de um sensor.



Fonte: Elaborada pelo autor (2018).

Figura 9: Nó sensor MICAz.

Fonte: (MICAZ, 20--).

O nó sensor da Figura 9 é um nó sensor MICAz, fabricado pela MEMSIC (MICAZ, 20--). Ele é baseado no microcontrolador ATmega128L, que é um microcontrolador de baixa potência. Um único dispositivo é capaz de operar sensores, processamento e comunicações de rede simultaneamente. Possui entradas analógicas e digitais, e diversas interfaces para utilização com diversas unidades periféricas. O *Datasheet* dele está anexado no Apêndice A.

Os nós sensores mais comuns custam em torno de € 80 a € 100 (ADVANTICSYS, 2019) e eles podem variar bastante em uma RSSF de acordo com a sua aplicação. Por exemplo, um simples sensor pode monitorar um único fenômeno físico ou ser responsável por uma única tarefa, enquanto outros dispositivos mais complexos sãos capazes de combinar diferentes técnicas de sensoriamento, ou coordenação dos nós de uma rede e agregação de dados para a transmissão para a estação base. Eles também podem variar quanto as capacidades de comunicação, podendo ser por infravermelho, ultrassom ou rádio frequência (HENKEL, 20--).

As principais características dos nós sensores em termos de arquitetura de *software* são (AKYILDIZ e VURAN, 2010):

1) Eficiência energética: Como as RSSF costumam operar em ambientes remotos ou com difícil acesso é necessário minimizar o número de comunicações. Portanto foram desenvolvidos protocolos e configurações de rede para reduzir o consumo de energia.

2) Modularidade: No caso de alguns dispositivos sensores mais sofisticados, é possível modificar o *hardware* de acordo com a finalidade. Logo, o *software* deve ser modificado de acordo.

3) Endereçamento dos sensores: Os sensores podem ser identificados individualmente dependendo da finalidade.

4) Auto-organização: Os nós são capazes de se autoconfigurar, escolhendo a melhor rota de transmissão de acordo com parâmetros do sistema.

5) Tarefas colaborativas/Agregação de dados: Cada nó executa várias operações: coleta, processamento e transmissão de dados, e roteamento de dados dos outros nós também. Caso a região de coleta seja muito hostil, a RSSF é capaz de reduzir o tráfego de informação na rede e sintetizar os dados coletados.

6) Mobilidade: Dependendo do ambiente que a RSSF está coletando dados, os sensores podem ou não se deslocar. Por exemplo, quando se deseja monitorar o consumo de energia elétrica de uma residência os sensores são estáticos. Por outro lado, em aplicações de rastreamento os sensores podem ser móveis (DUARTE et al., 2011; JUANG et al., 2002).

De acordo com Yick et al (2008), existem dois tipos de RSSF: estruturadas e não estruturadas. Uma RSSF não estruturada contém uma grande quantidade de nós sensores, que podem ser distribuídos de forma aleatória na área observada ou muito próximos dela. Após a instalação, a rede opera por conta própria e sem a intervenção externa para que possa executar a coleta e transmissão de dados, e devido a quantidade de nós nesse tipo de rede, é mais difícil fazer manutenção e detecção de falhas nos dispositivos. Em uma RSSF estruturada, os nós são projetados de uma forma pré-planejada, permitindo que menos nós sensores sejam utilizados e visando locais específicos para o seu melhor funcionamento.

Entre as principais características das RFFS, destacam-se:

1) Baixo custo: Os nós sensores são baratos, e não é necessária nenhuma estrutura pré-existente ou cabeamento para suporte. Cada nó sensor é incorporado à RSSF assim que é adicionado à rede.

2) Escalabilidade: RSSF podem ser ajustadas com adição de mais dispositivos para atender a uma determinada área ou aplicação. Quanto mais nós sensores são adicionados, mais robusta fica a rede, permitindo que haja mais interconexões entre os nós, e aumentando a área de abrangência da rede.

3) Adaptabilidade: As redes são capazes de se ajustar de acordo com mudanças ambientais ou com as necessidades do projetista. Consegue responder a mudanças de topologias e falhas nos sensores com facilidade, assim como alterar o modo de operação dos nós.

4.2 APLICAÇÕES

As RSSF podem consistir dos mais diversos tipos de sensores, e.g. magnéticos, termais, acústicos, sísmicos, entre outros, o que permite que uma grande variedade de parâmetros seja observada, como, por exemplo, temperatura, umidade, níveis de pressão e som, movimento e direção. Como resultado, surge um vasto campo de aplicações em que elas podem ser utilizadas, que inclui análise e predição climática, monitoramento ambiental e segurança, por exemplo.

Entre as principais aplicações das RSSF estão, segundo Akyldiz et al. (2002):

1) Aplicações militares: operações militares de comando, controle, inteligência, monitoramento e rastreamento (AKYILDIZ e VURAN, 2010).

2) Aplicações ambientais: monitoramento dos movimentos de pássaros e animais pequenos, irrigação e agricultura de precisão, pesquisa meteorológica e geofísica, detecção de incêndios em florestas e grandes áreas, detecção de inundações e atividade vulcânica, entre outros.

3) Aplicações médicas: controle de medicamentos, monitoramento de pacientes, rastreamento de médicos e pacientes, e monitoramento e avaliação de sinais vitais e casualidades em emergências (PINTO, 2015; AKYILDIZ e VURAN, 2010).

4) Aplicações de *Smart Grid:* As RSSF nas smart grids podem ser divididas em três grupos: consumidor, transmissão e distribuição, e geração (TUNA et al., 2013).

(a) Aplicações voltadas ao consumidor: gerenciamento doméstico de energia e demanda, medição inteligente, gerenciamento de painéis fotovoltaicos, controle e monitoramento de equipamentos;

(b) Aplicações voltadas para a transmissão e distribuição de energia: monitoramento de linhas de transmissão terrestres e subterrâneas, diagnóstico de faltas, detecção de interrupções, monitoramento de temperatura de condutores.

(c) Aplicações voltadas à geração de energia: monitoramento de geradores, monitoramento remoto de parques eólicos e fotovoltaicos, e qualidade de geração.

Entretanto, as condições de propagação comumente encontradas nos sistemas de potência são bastante severas para esses sensores, aumentando os desafios de comunicação pelo link *wireless* e necessitando de atenção especial quando forem instalados nas *smart grids*. Além disso, garantir qualidade de serviço (QoS) em meio a interferências e dificuldades do meio wireless, e os limites de banda e as limitações físicas dos próprios sensores também se mostram como desafios a serem estudados (TUNA et al., 2013).

4.3 COMUNICAÇÃO EM RSSF

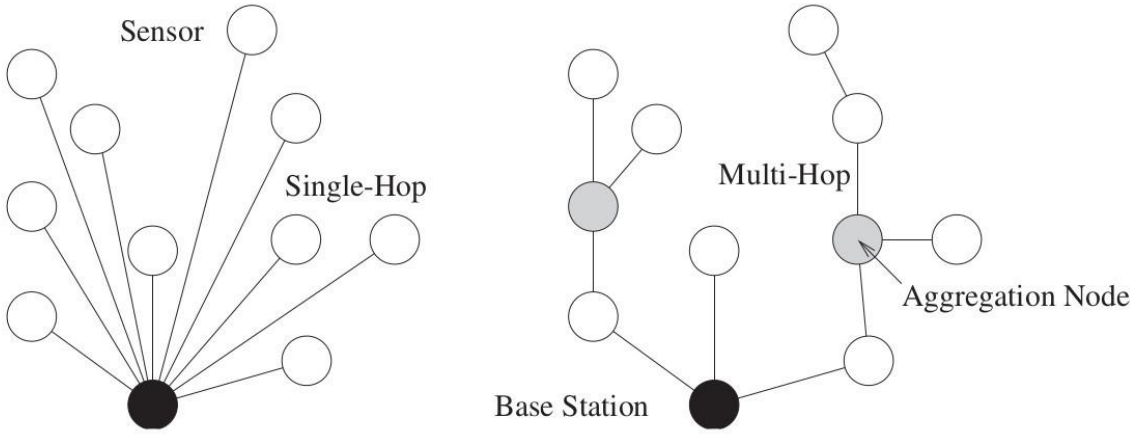
O padrão utilizado para redes sem fio é o IEEE 802.11 (IEEE 802.11, 2016) e seus familiares. Eles utilizam diferentes bandas de frequência, por exemplo, os IEEE 802.11b (IEEE 802.11b, 1999) e IEEE 802.11g (IEEE 802.11g, 2003) utilizam a banda de 2.4-GHz, já o protocolo IEEE 802.11 a utiliza a banda de 5-GHz. O IEEE 802.11 era utilizado no início das RSSF e é utilizado hoje em aplicações quando a demanda de banda é alta, mas devido à alta energia necessária para transmitir cabeçalhos em redes baseadas no IEEE 802.11, esse padrão se torna indesejável para redes de sensores de baixa potência. Isso levou ao desenvolvimento do protocolo IEEE 802.15.4 (GUTIERREZ et al. 2001) que foi criado especificamente para comunicações de curtas distâncias em redes de sensores de baixa potência e hoje tem bastante suporte comercial e da academia. Ele opera nas frequências de banda de 868 MHz, 915 MHz e 2.45-GHz, e as taxas de transferência suportam 20 até 250 kbps (DARGIE e POELLABAUER, 2010).

Entre os principais protocolos utilizados em conjunto com o padrão IEEE 802.15.4 destacam-se o *Zigbee* e o *WirelessHART* (LENNVALL e SVENSSON, 2008). O *Zigbee* é um protocolo específico para comunicações sem fio com baixa potência, baixas taxas de transferência e baixo custo, voltado principalmente para automação doméstica e tarefas de controle e monitoramento, e hoje é o nome

comercial para o padrão IEEE 802.15.4. Apesar de ser amplamente utilizado em aplicações domésticas, o *Zigbee* não é a melhor opção para aplicações industriais, nas quais é imprescindível ter confiabilidade e segurança na transmissão de informação. Nesses quesitos, o *WirelessHART* é uma opção mais adequada pois além de possuir todas as características das RSSF mencionadas anteriormente em 4.1, ele também oferece mais segurança na comunicação.

Quanto a topologia, as RSSF podem se organizar em estrela, em que todos os nós se comunicam com um nó base ou central – chamado de comunicação do tipo *single-hop* – ou em malha ou árvores, como pode ser visto na Figura 10. Nessas, os nós se comunicam entre si ou com nós sensores responsáveis por agregar os dados coletados. Assim, os dados são transmitidos por vários nós até ser recebido no seu destino – chamado de comunicação em *multi-hop* (DARGIE e POELLABAUER, 2010).

Figura 10 – Comunicação Single-hop vs. multi-hop em redes de sensores.



Fonte: (DARGIE e POELLABAUER, 2010).

As modulações comumente utilizadas nas RSSF são as *Binary Phase Shift Keying* (BPSK), *Quadrature Phase Shift Keying* (QPSK) e *Offset Quadrature Phase Shift Keying* (OQPSK) (AKYILDIZ e VURAN, 2010). Em um esquema comum de modulação, uma forma de onda senoidal representada por:

$$s(t) = r(t)\cos\left(2\pi f_c t + \psi(t)\right) \quad (18)$$

que possui três componentes:

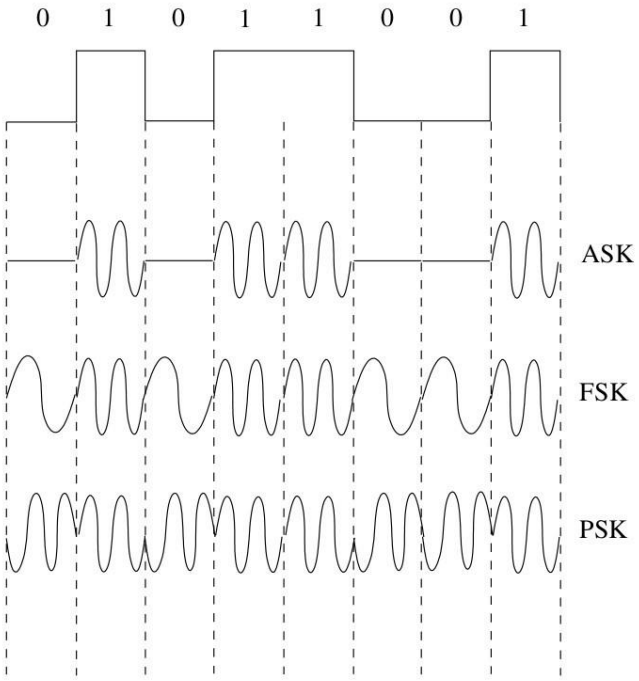
(a) Amplitude $r(t)$: A amplitude ou contorno da onda.

(b) Frequência f_c: A frequência da onda.

(c) Fase $\psi(t)$: A fase da onda.

A informação digital é transmitida a partir da modificação de um desses três componentes de acordo com os bits que serão transmitidos. O que leva aos três principais esquemas de modulação digital, que podem ser vistos na Figura 11, e que servem de base para várias outras:

Figura 11 – Os três esquemas básicos de modulação, *amplitude shift keying* (ASK), *frequency shift keying* (FSK), e *phase shift keying* (PSK).



Fonte: (AKYILDIZ e VURAN, 2010).

1) *Amplitude shift keying* (ASK): Baseia-se na modificação da amplitude do sinal de acordo com a informação a ser transmitida.

2) *Frequency shift keying* (FSK): Nesse esquema, a frequência do sinal é ajustado de acordo com a informação a ser transmitida.

3) *Phase shift keying* (PSK): Por último, a fase do sinal de onde é ajustado de acordo com os bits que serão transmitidos.

Nas modulações BPSK e QPSK, são transmitidos 1 e 2 bits por símbolo, respectivamente. OQPSK é uma variante da QPSK, na qual utiliza mudanças de fase que não ultrapassam 90^{o} por vez, diferente da QPSK, que permite mudanças de fase de até 180^{o} (DAERI et al.; 2016). Como as duas têm performances parecidas, optou-se em utilizar a QPSK nas simulações do capítulo 5.

4.4 CONTROLE DE ERRO EM RSSF

As RSSF possuem baixos requisitos de energia e são caracterizadas pela natureza colaborativa dos sensores. Para evitar que uma informação errônea seja retransmitida múltiplas vezes, aumentando o consumo de energia de rede, técnicas de controle de erro são extremamente importantes. O principal propósito dessas técnicas em RSSF é prover confiabilidade na comunicação via *wireless.*

As principais técnicas de correção de dados utilizadas nas RSSF são (AKYILDIZ e VURAN, 2010):

1) *Automatic Request for Retransmission* (ARQ), que consiste basicamente na retransmissão de pacotes na ocorrências de erros durante a transmissão;

2) *Hybrid* ARQ (HARQ): técnica que combina ARQ e FEC (*Foward Error Correction*) ou correção direta de erros, ou seja, que utilizam códigos corretores de erros para detecção e correção de erros na transmissão.

3) Códigos BCH: códigos binários capazes de corrigir apenas t erros.

O código RS e os códigos LDPC também são utilizados, mas em menor frequência devido às limitações de processamento e energia dos nós sensores.

4.5 DESAFIOS DAS RSSF EM *SMART GRID*

Os principais desafios técnicos para a execução de RSSF em aplicações de *smart grid* são apresentados a seguir (GUNGOR et al., 2010; TUNA et al., 2013; AKYILDIZ et al., 2002, SAHIN et al., 2014).

1) Limitações de recursos dos sensores: Os sensores têm capacidade limitada em função das suas características de *hardware.* Como consequência disso, deve-se evitar o uso irresponsável dos seus recursos já que em muitos casos os sensores operam sem interferências externas ou manutenção.

2) Condições ambientais e topologias dinâmicas: Em ambientes de distribuição e transmissão dos sistemas de potência, a conectividade e a topologia da rede pode variar em função das características do canal wireless. Os sensores também estão sujeitos a condições ambientais adversas, o que pode levar ao mal funcionamento ou a falhas na informação transmitida.

3) Requisitos de Qualidade do Serviço (QoS): Dependendo das aplicações, são necessários diferentes requisitos de QoS em termos de confiabilidade, latência e taxa de transferência. Por exemplo, para diagnósticos de faltas e respostas em tempo real, as RSSF devem ser confiáveis, precisas e rápidas.

4) Capacidade variável do canal e erros de pacote: A performance da comunicação das RSSF sofre com as variações ambientais. Perdas de caminho, interferências no sinal e erros de bits são fatores essenciais na escolha das características do canal a ser utilizado para que a transmissão seja confiável e estável.

5 SIMULAÇÕES E RESULTADOS

5.1 RUÍDO ADITIVO

O modelo de canal utilizado nas simulações a seguir é o modelo de canal de ruído aditivo (RYAN e LIN, 2009; PROAKIS e SALEHI, 2008). Neste modelo, o sinal transmitido $\mathrm{x(t)}$ é corrompido por um processo aleatório de ruído aditivo $\mathrm{n(t)}$, resultando no sinal recebido $\mathrm{y(t)}$:

$$\mathrm{y(t) = x(t) + n(t)} \tag{19}$$

Quando $\mathrm{n(t)}$ é um processo de ruído gaussiano branco chamamos esse modelo de canal, de canal de ruído gaussian branco aditivo (AWGN). O sinal $\mathrm{n(t)}$ é obtido a partir de um processo aleatório gaussiano com média zero e variância $\sigma_{\mathrm{n}}{}^{2} = \mathrm{N_0}$. A quantidade $\mathrm{N_0}$ indica a densidade espectral de potência do ruído unilateral.

A relação entre a potência do sinal do transmitido e a potência do ruído é chamada de relação ou razão sinal-ruído (*Signal-to-noise ratio* – SNR) e é definida por:

$$\mathrm{SNR} = \frac{\mathrm{P_{sinal}}}{\mathrm{P_{ruído}}}, \tag{20}$$

em que $\mathrm{P_{sinal}}$ é a potência do sinal transmitido e $\mathrm{P_{ruído}}$ é a potência do ruído do canal. A equação (5.2) pode também ser expressa por:

$$\mathrm{SNR} = \frac{\mathrm{E_b}}{\mathrm{N_0}}, \tag{21}$$

onde $\mathrm{E_b}$ é a energia por bits transmitidos no sinal e $\mathrm{N_0}$ a densidade espectral de potência do ruído, mencionado anteriormente.

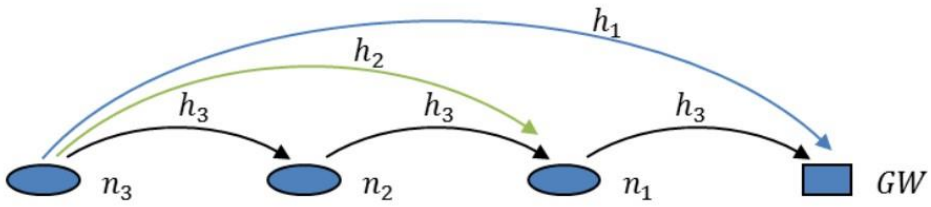
A relação sinal-ruído é o parâmetro mais utilizado para medir a qualidade de um sinal em um canal de comunicação. A SNR expressa em decibéis (dB) quanto o nível de um sinal excede o nível de ruído em um determinado canal ou banda (FREEMAN, 2005; PROAKIS e SALEHI, 2008).

Este é o modelo de canal predominantemente usado na análise e no projeto de sistemas de comunicação e será, portanto, o modelo de canal utilizado para analisar o comportamento dos códigos de correção de erros.

5.2 MODELO DA REDE

Com base em Eriksson (2011), o modelo de rede escolhido é o de uma rede simples com três nós equidistantes $\mathrm{n1}, \mathrm{n2}$ e $\mathrm{n3}$, conectados a um nó coordenador ou um *Gateway* (GW), como pode ser visto na Figura 12.

Figura 12 – Modelo da rede escolhido para análise.



Fonte: (ERIKSSON, 2011).

Na Figura 12, cada nó é capaz de rotear dados de outros nós da mesma rede. A informação é transmitida do nós mais distante $\mathrm{n3}$ até o GW por meio de um único salto (*single-hop*) ou múltiplos saltos (*multi-hop*). Existem três rotas até o GW; através de h_1, de h_2 e h_3, e de três saltos com h_3. Além disso, assume-se que GW é responsável por realizar a decodificação dos pacotes no final de cada transmissão. A tomada de decisão em termos de rotas será com base na taxa de erros de bits (*Bit Error Rate* – BER), que é definida por:

$$\mathrm{BER} = \frac{\text{Total de bits com erros}}{\text{Total de bits transmitidos}}, \tag{22}$$

de acordo com a SNR do canal.

Assume-se também que todos os nós têm conhecimento das possíveis rotas de transmissão assim como dos códigos de correção de erro. O GW possui conhecimento do estado da rede e pode tomar decisões de acordo com as características da rede.

Considerando ambientes industriais ou domésticos, a qualidade do canal pode variar dependendo da posição dos nós, assim como da distância dos nós até o GW. Para determinar como a SNR varia entre os saltos, um modelo de atenuação que se aproxima aos efeitos da propagação de sinal por um meio *wireless* é adotado aqui. Atenuação ou perda do caminho é a redução da potência de um sinal eletromagnético à medida que é transmitido pelo canal *wireless*. Esse efeito pode ocorrer devido a vários fatores, como, por exemplo, as condições ambientais do local da rede ou da distância entre o transmissor e o receptor.

De acordo com Goldsmith (2005), existe um modelo simplificado de atenuação que é geralmente utilizado para análises comparativas entre vários *designs*. Esse modelo consegue capturar a essência da propagação de sinal sem precisar recorrer a modelos mais complicados, e é definido como:

$$P_r = P_t A \left(\frac{d_0}{d}\right)^{\gamma}, \tag{23}$$

onde $\mathrm{P_r}$ e $\mathrm{P_t}$ são as potências recebidas e transmitidas, respectivamente, e A é uma constante que depende das características da antena e da interferência e ruído do canal. O expoente de atenuação γ é determinado com base no ambiente de propagação. Segundo Goldsmith (2005), os valores de A, $\mathrm{d_0}$ e γ podem ser obtidos por meio de aproximação de modelos analíticos ou empíricos. Podem também ser determinados através de medições. A Tabela 4 apresenta alguns valores típicos de γ (GOLDSMITH, 2005).

Tabela 4 – Valores típicos de γ para determinados ambientes.

Ambiente	**Intervalos de γ**
Macrocélulas Urbanas	3.7 – 6.5
Microcélulas Urbanas	2.7 – 3.5
Predial (mesmo andar)	1.6 – 3.5
Predial (múltiplos andares)	2 – 6
Empreendimentos, Lojas	1.8 – 2.2
Fábricas – Indústria	1.6 – 3.3
Doméstico	3

Dada a potência do sinal recebido P_{r3} após a transmissão com um salto curto h_3, as potências dos sinais recebidos nos saltos mais longos h_1 e h_2 podem ser encontradas através de:

$$P_{r2} = P_{r3} A (\delta_2)^{\gamma} \text{ e} \tag{24}$$

$$P_{r1} = P_{r3} A (\delta_1)^{\gamma} \tag{25}$$

onde

$$\delta_2 = \left(\frac{d_3}{d_2}\right) \tag{26}$$

$$\delta_1 = \left(\frac{d_3}{d_1}\right) \tag{27}$$

são as distâncias relativas dos respectivos saltos em função do salto mais curto h_3 para a distância d_3.

Para simplificar, os valores de A, $d_0 = d_3$ e γ são os mesmos para todos os saltos. As equações (24) e (25) podem ser expressas em termos de SNR:

$$SNR_2 = SNR_3 (\delta_2)^{\gamma} \text{ e} \tag{28}$$

$$SNR_1 = SNR_3 (\delta_1)^{\gamma} \tag{29}$$

Sendo assim, os parâmetros de distância d_3 e SNR SNR_3 para o salto mais curto h_3 serão adotados como referência, e, a partir deles, as razões δ_1 e δ_2 e os valores de SNR SNR_1 e SNR_2 para os saltos h_1 e h_2 serão calculados por meio de (26), (27), (28) e (29).

5.3 SIMULAÇÕES

A Tabela 5 resume os parâmetros utilizados nas simulações. Os valores de A, δ_1, δ_2 e γ foram definidos de acordo com Eriksson (2011) e Goldsmith (2005).

Tabela 5 – Parâmetros adotados para as simulações do trabalho.

A	1
γ	3
δ_1	1/3
δ_2	1/2
FEC	RS (255, 239), LDPC (6480, taxa de 2/3), LDPC (1920, taxa de 1/2)
Canal	AWGN
Modulação	QPSK

Os códigos RS e LDPC, e os cenários da RSSF com 1, 2, e 3 saltos foram implementados em linguagem C++ e estão em anexo nos apêndices B e C. A informação transmitida é codificada apenas uma vez, no primeiro nó, e decodificada no GW. Cada código foi simulado para uma quantidade 1000 frames, ou seja, o código foi executado para 1000 palavras-códigos para dar mais confiabilidade nos resultados obtidos, a fim de obter valores médios de BER. Cada simulação gera uma palavra-código aleatória, portanto, para 1000 frames, são geradas 1000 palavras-códigos. A quantidade de frames foi definida em função do tempo de processamento do código LDPC.

Para o RS, o código utilizado foi o (255, 239), ou seja, a palavra-código possui 239 bytes de informação (K), 255 bytes no total (N), e 16 bytes de paridade, ou seja, capacidade de correção de até 8 bytes por palavra-código (frame), totalizando 2040 bits transmitidos com a paridade. Para o LDPC, foram utilizados dois códigos.

Primeiramente, considerou-se a matriz de verificação de paridade H da Figura 4, com tamanho de código N=6480 bits, K = 4320 bits de informação e 2160 bits de paridade, e taxa de código de 2/3, ou seja, 2/3 da palavra-código contém informação. Para um segundo momento, também foi utilizado um código com N=1920 bits, K= 960 bits de informação, 960 bits de paridade, e taxa de código de ½, ou seja, metade da palavra-código possui bits de informação.

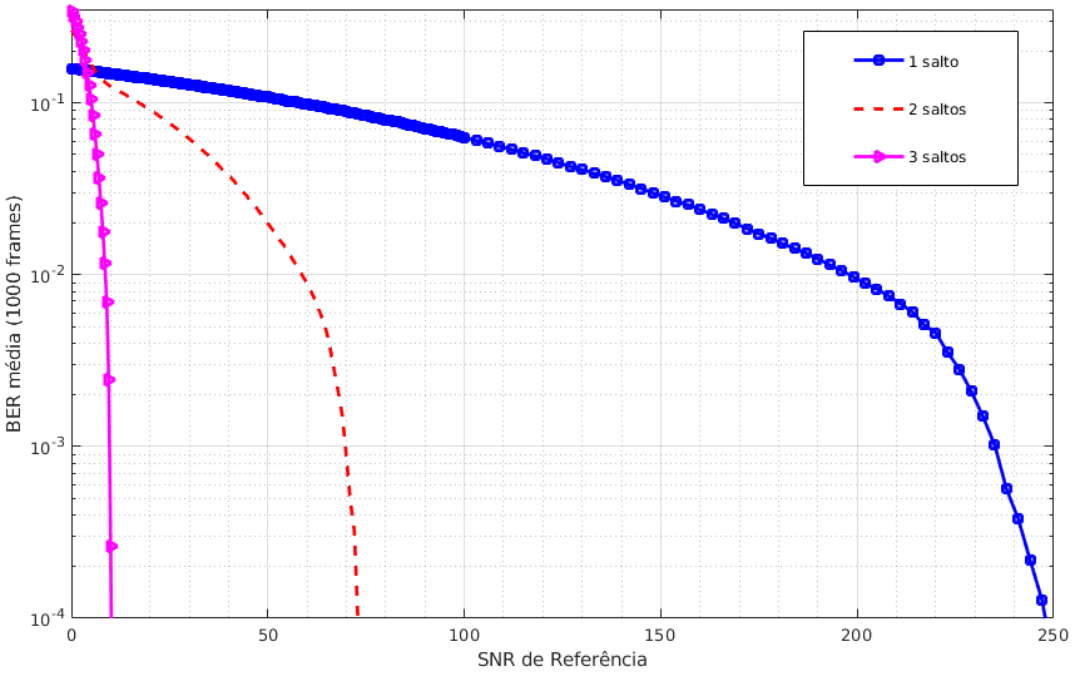
Adotando A = 1, δ_1 = 1/3, δ_2 = 1/2 e γ = 3, os valores de SNR para os trechos h_1 e h_2 serão ajustados de acordo. Por exemplo, quando $\mathrm{SNR}_3 = 5\mathrm{dB}$, $\mathrm{SNR}_2 = 0.625\mathrm{dB}$ e $\mathrm{SNR}_1 = 0.185\mathrm{dB}$. Então, para o cenário de três saltos, a mensagem passa por três canais com a SNR de referência SNR_3 até chegar no GW; para o cenário de dois saltos, a mensagem passa uma vez pelo trecho com SNR = SNR_2 e uma vez

pelo de SNR = SNR_3; e por fim, para o cenário de um único salto, a mensagem passa uma vez pelo trecho com SNR = SNR_1.

5.4 RESULTADOS

A Figura 13 mostra os valores médios de BER obtidos com o código RS (255,239) utilizando 1000 frames para três cenários: o primeiro cenário com um único salto h_1, o segundo com um salto h_2 e um salto h_3, e o terceiro com três saltos h_3, e com valores de SNR em cada trecho ajustados em função da SNR de referência SNR_3.

Figura 13 – Performance do código RS (255, 239) para os três cenários estudados.



Fonte: Elaborada pelo autor (2019).

A SNR de Referência plotada na Figura 13 não é o valor real na simulação, sendo usada apenas para facilitar a comparação entre os três cenários. Os valores de SNR de Referência plotados são muito altos, mas em função da atenuação do sinal e da distância entre os nós, os valores de SNR para cada trecho são menores. Como comentado anteriormente em 5.3, para uma SNR de Referência $\mathrm{SNR}_3 = 5\mathrm{dB}$, no primeiro cenário a palavra-código será transmitida por um canal com $\mathrm{SNR}_1 =$

$0.185\,\mathrm{dB}$, no segundo cenário a palavra-código passará por um canal com $\mathrm{SNR}_2 = 0.625\,\mathrm{dB}$ e outro canal com $\mathrm{SNR}_3 = 5\mathrm{dB}$, e no terceiro cenário a palavra-código passará por três canais distintos com $\mathrm{SNR}_3 = 5\mathrm{dB}$.

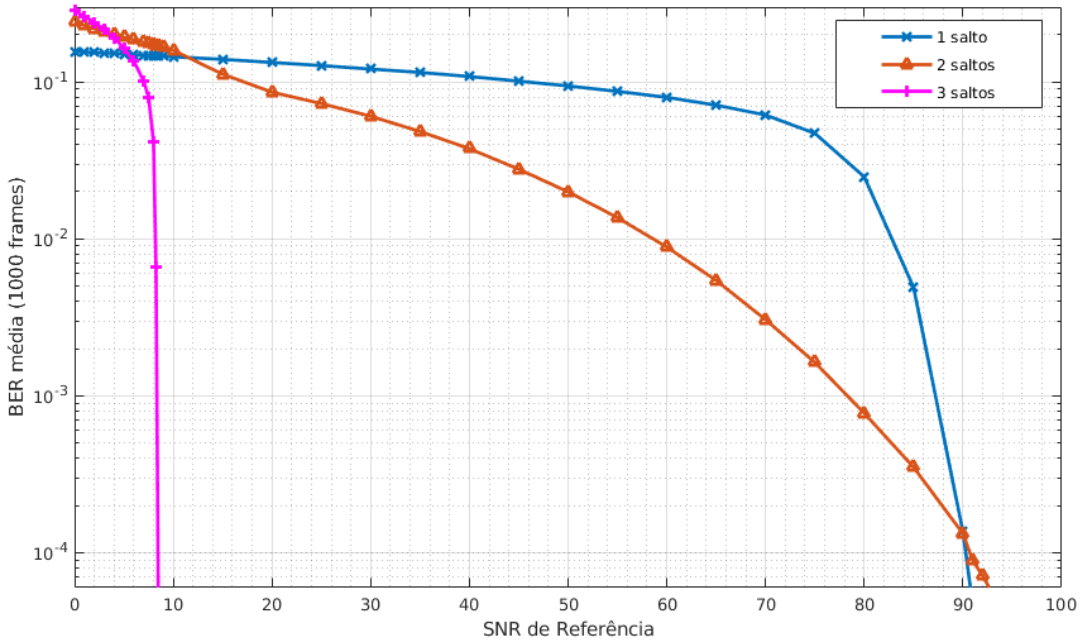
Podemos observar na Figura 13 que o cenário de 3 saltos para de devolver erros em torno de 10.5 dB, enquanto para os outros dois cenários os valor da SNR de referência precisa ser muito alto para o código corrigir todos os erros na mensagem. Isso se dá devido a atenuação do sinal e a variação da distância entre os nós. É importante observar que, a performance de um código corretor de erros é melhor, quanto mais próxima a curva de BER se localizar do eixo y, ou seja, do limite de Shannon, referente a capacidade do canal. No caso da Figura 13, observa-se que a curva de BER, considerando 3 saltos, apresenta melhor performance que as curvas de 2 saltos e de 1 salto, respectivamente. Sendo a de 1 salto o pior caso.

Mais especificamente, podemos observar que na Figura 13, para a SNR_3= 11 dB, a taxa média de BER para 1000 frames é igual a zero para o cenário com três saltos, enquanto para o cenário com dois saltos é igual 0.120504, e para um único salto é 0.14747, ou seja, para o mesmo valor de SNR, observamos que a BER é menor para o cenário de 3 saltos do que para os cenários de 2 e 1 saltos.

Através da análise dos resultados de BER para os 3 cenários, observamos a vantagem na utilização de uma RSSF com mais saltos, ou seja, com mais sensores.

A Figura 14 mostra os valores médios de BER obtidos com o código LDPC com N=6480 bits, para 1000 frames também para os três cenários, em função da SNR de referência SNR_3.

Figura 14 – Performance do código LDPC com N=6480 bits, para os três cenários estudados.



Fonte: Elaborada pelo autor (2019).

Na Figura 14 nota-se uma performance similar da obtida com o código RS (255,239), na Figura 13. O cenário de três saltos para de devolver erros em torno de 9 dB, enquanto para os outros dois cenários o valor da SNR de referência precisa ser muito alto para o código corrigir todos os erros na mensagem. Além disso, observa-se também na Figura 13, que a curva de BER para o cenário com 3 saltos apresenta a melhor performance, do que os cenários com 2 e 1 saltos. Sendo o cenário de 1 salto o pior caso. É importante salientar que na Figura 14, próximo da SNR_3 = 90dB, há uma pequena mudança no comportamento das curvas de BER, fazendo com que a performance para 1 salto passe a ser melhor que para 2 saltos nesses valores de SNR. Isso pode ter ocorrido em função do expoente de atenuação ou até pelo próprio comportamento do código LDPC. De qualquer forma, ainda é mais vantajoso transmitir a mensagem em três saltos para curtas distâncias do que pelos outros dois cenários.

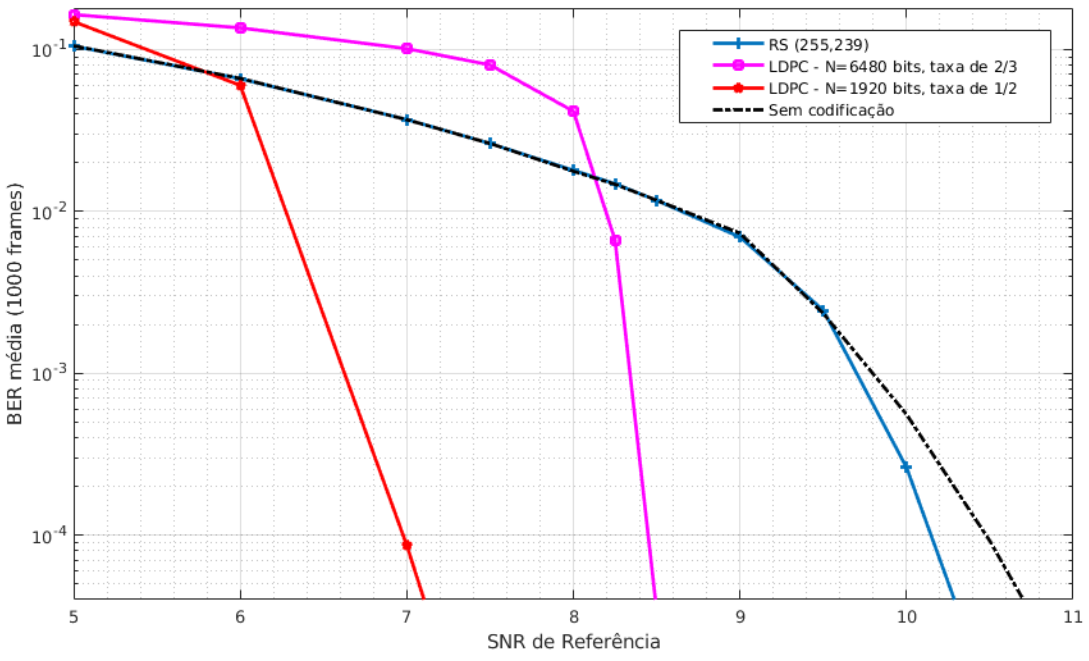
Fazendo as mesmas ponderações para os resultados na Figura 14, para a SNR_3= 9 dB, a taxa média de BER para 1000 frames é igual a zero para o cenário com três saltos, enquanto para o cenário com dois saltos é igual 0,16597, e para um único salto é 0,146262, ou seja, para o mesmo valor de SNR, observamos que a

BER é menor para o cenário de 3 saltos do que para os cenários de 2 e 1 salto, assim como no caso do código RS na Figura 16.

A Figura 15 mostra a performance do código RS (255, 239), de dois códigos LDPC, um para N=6480 bits com taxa de 2/3 e um com N=1920 bits e taxa de ½, e também sem codificação, para o cenário de três saltos, dado que este foi considerado o cenário com a melhor performance dentre os 3 cenário analisados.

Podemos observar na Figura 15, que o código LDPC apresenta melhor performance que o código RS, considerando o cenário com 3 saltos. Além disso, observa-se que, o LDPC com taxa 1/2 apresentou melhor performance que o LDPC de taxa 2/3. Optou-se em testar mais uma opção de código LDPC com taxa 1/2 para mostrarmos que ainda é possível um ganho de performance tanto do LDPC quanto do RS, através da alteração de parâmetros, como taxas de códigos e quantidade de bits de paridade, dependendo do nível de confiabilidade necessário para a transmissão. Além disso, observamos também na Figura 15, que a simples utilização de um código corretor de erros já apresenta vantagens para o sistema quando comparado a uma transmissão sem codificação.

Figura 15 – Performance do código RS (255,239), do código LDPC com N=6480 bits e taxa de 2/3, do código LDPC com N=1920 bits e taxa de ½, e sem codificação, para o cenário de 3 saltos.



Fonte: Elaborada pelo autor (2019).

O tempo de processamento para a transmissão de 1000 frames utilizando o código RS é de aproximadamente 1 segundo, enquanto para o LDPC é de aproximadamente 500 segundos. Isso se dá devido ao fato da decodificação iterativa do LDPC ter um custo computacional muito maior que a decodificação dos códigos RS, trazendo uma certa latência inerente ao sistema.

Podemos concluir, ao analisar os resultados que, a utilização dos códigos RS é preferível para aplicações em que se precisa transmitir ou receber dados em tempo real ou com intervalos bastante curtos, como para medição inteligente ou para monitoramento de unidades de geração distribuída, por exemplo. Os medidores permitem que a informação de gastos de energia e custos sejam informadas em tempo real para o consumidor, e para a geração distribuída, os sensores podem transmitir dados coletados de painéis fotovoltaicos ou de turbinas eólicas, por exemplo, em intervalos pequenos. Dentro do ambiente doméstico, a RSSF conecta vários dispositivos a um roteador central ou coordenador, conectando também ao medidor automático. Os códigos RS vão prover uma informação mais precisa e com tempo de processamento ótimo.

Para os códigos LDPC, aplicações viáveis são as que é imprescindível que haja fidelidade na informação transmitida, mesmo que comprometa o tempo de resposta em função do processamento. Para a transmissão de diagnósticos, os dados coletados pelos nós sensores podem ser processados e analisados em loco pelo GW e apenas o diagnóstico é transmitido para o receptor ou central de controle. Assim, os diagnósticos podem ser transmitidos com intervalos maiores entre eles, não sendo prejudicados pelo tempo de processamento dos códigos LDPC. Para medições de linhas de distribuições e transformadores, a CEA costuma utilizar medidores acoplados às linhas de distribuição que fazem medições de hora em hora durante um mês para que então esses dados sejam analisados pelos setores responsáveis. Uma RSSF com sensores de baixo custo monitorando parâmetros da rede, equipada com um código LDPC e enviando pacotes processados em forma de diagnóstico já apresenta uma melhora considerável no tempo de resposta das concessionárias, por exemplo.

É importante mencionar que a utilização de qualquer código corretor de erro é melhor do que enviar uma mensagem sem codificação. E que as RSSF podem ajustar o modo de operação de acordo com as características do canal, alterando o código utilizado ou os parâmetros de cada código, adicionando mais ou menos bits

de paridade se houver necessidade. e assim aumentando ou diminuindo a capacidade de correção do código.

Não foram analisados os parâmetros de energia e latência para este trabalho. O cenário com três saltos apresentou melhor performance para ambos os códigos, e segundo a literatura, para o cálculo de energia considera-se a energia de codificação da mensagem e a energia gasta com transmissão e recepção da mensagem. Desconsiderou-se a energia gasta com codificação porque geralmente o processamento é feito ao final da transmissão nos centros de controle ou por dispositivos mais potentes da RSSF. Como o mesmo cenário apresentou melhor performance para ambos os códigos, não pareceu viável analisar o gasto de energia da rede. Quanto a latência, a informação mais presente na literatura é associada à quantidade de pacotes retransmitidos, e como este trabalho não utiliza ARQ ou nenhum código híbrido, não foi possível avaliar o parâmetro em tempo hábil.

6 CONCLUSÃO

As *Smart Grid* são vistas como a chave para a modernização do Sistema de Potência convencional, que tem apresentado problemas críticos em função da sua infraestrutura antiga e complexa. Elas requerem sistemas de comunicação eficientes e que facilitem a comunicação entre consumidores e o SEP. Uma subestação fornece energia e informações para os consumidores, e esses em contrapartida podem também responder com informações de consumo para as concessionárias. Além disso, a instalação de técnicas de comunicação cabeadas requerem investimentos altos e gastos com manutenção, além de serem pouco flexíveis.

Assim, as redes de sensores sem fio têm se apresentado como uma importante tecnologia para coleta de informações e monitoramento de sistemas, devido seu baixo custo de implementação e características adaptáveis. Elas podem ser utilizadas associadas a diversas aplicações de *Smart Grid*, mas tem limitações físicas de energia e processamento. Elas precisam atender específicos requisitos de QoS, e dependendo do sistema, precisam transmitir os dados coletados com exatidão e rapidez. Para minimizar gastos de energia com retransmissão de pacotes, utilizam-se técnicas de correção de erros.

Os códigos corretores de erro já são extensivamente utilizados nos sistemas de comunicação visando a transmissão de dados com confiabilidade e fidelidade. Os códigos RS são empregados hoje em diversas aplicações, desde comunicação via satélite e armazenamento de dados. Diversos padrões de sistema de comunicação têm optado por usar códigos LDPC para melhorar o desempenho dos sistemas. Os códigos LDPC possuem a vantagem da alta correção de erros, se aproximando do limite de Shannon e da sua decodificação iterativa de fácil implementação. A utilização de códigos corretores permite que sistemas operem com SNR menores do que sem qualquer técnica de codificação.

Este trabalho demonstrou que, para uma RSSF com três cenários, é vantajoso transmitir uma mensagem de nó para nó a curtas distâncias até o dispositivo receptor do que para longas distâncias. Demonstrou também que os códigos estudados, RS e LDPC, apresentam bom desempenho na correção de erros pelo canal *wireless.*

Atestou-se também que o código LDPC apresentou um desempenho melhor em termos de correção de bits comparados com o código RS. Para aplicações de

Smart Grid em que é mais importante a resposta de dados em tempo real, como sistemas de medição e monitoramento de unidades de geração distribuída, é interessante a utilização do código RS, enquanto para aplicações com uma exigência da exatidão da informação transmitida, por exemplo, monitoramento de equipamentos críticos do SEP e comunicação de diagnósticos, é preferível a utilização do código LDPC.

Para trabalhos futuros, pode-se testar com uma RSSF com uma quantidade maior de nós sensores, já que o modelo analisado neste trabalho foi um modelo simplificado. Outros códigos corretores de erro também podem ser testados, comparando com os códigos RS e LDPC já analisados. Além disso, os códigos RS e LDPC implementados estão prontos para receberem outros parâmetros de códigos e da rede.

BIBLIOGRAFIA

ADVANTICSYS, **802.15.4 Mote Modules.** Disponível em: https://www.advanticsys.com/shop/mtmcm5000msp-p-14.html. Acesso em: 15 Jun, 2019.

AKYILDIZ, I. F.; SU W.; SANKARASUBRAMANIAM, Y.; CAYIRCI, E. **Wireless sensor networks: a survey.** (Artigo). Computer Networks, vol. 38, pp. 393–422, 2002.

AKYILDIZ, I. F.; VURAN, M. C. **Wireless Sensor Networks**. John Wiley & Sons Ltd, 2010.

AMIN, S.M.; WOLLENBERG, B. F. **Toward a Smart Grid: power delivery for the 21st century.** (Artigo) **I**EEE Power and Energy Magazine, vol. 3, no. 5, pp. 34–41, 2005.

BERLEKAMP, E. R. **On decoding binary bose-chaudhuri-hocquenghem codes**, IEEE Trans. Inf. Theory, vol. IT-11, pp. 577-580, Out. 1965.

BILGIN, B. E.; BAKTIR, S.; GUNGOR, V. C.**Collecting smart meter data via public transportation buses.** (Artigo) IET Intell. Transp. Syst., vol. 10, no. 8, pp. 515–523, 2016.

CAO, L.; JINGWEN, T.; LIU, Y. **Remote Wireless Automatic Meter Reading System Based on Wireless Mesh Networks and Embedded Technology.** (Artigo). Fifth IEEE International Symposium on Embedded Computing, 2008.

DAERI, A. M.; ELFITURI, M.; ZEREK, A. R. **Quadrature Phase Shift Keying and Offset Quadrature Phase Shift Keying BER Performance Comparison.** (Artigo). 3rd International Conference on Automation, Control, Engineering and Computer Science, ACECS, 2016.

DAKI, H.; HANNANI, A. E.; AQQAL, A.; HAIDINE, A.; DAHBI, A. **Big Data Management in smart grid**: concepts, requirements and implementation. Journal of Big Data, 2017.

DARGIE, W.; POELLABAUER, C. **Fundamentals of Wireless Sensor Networks.** Theory and Practice. 1. ed, John Wiley & Sons Ltd, 2010.

DE HAAS, M. M. P, **Online diagnostics in Smart Grids:** An approach to the future power grid. Delft. Dissertação (Mestrado em Ciências) – Delft University of Technology, 2011.

DOE, **Advanced Metering Infrastructute and Customer Systems.** Tech. Rep., U.S. Department of Energy, Washington, DC, USA, 2016. Disponível em: https://www.energy.gov/sites/prod/files/2016/12/f34/AMI%20Summary%20Report_09-26-16.pdf. Acesso em: 10 Jun, 2019.

DOE, **Communications requirements of smart grid technologies.** Tech. Rep., U.S. Department of Energy, Washington, DC, USA, 2010. Disponível em: https://www.energy.gov/sites/prod/files/gcprod/documents/Smart_Grid_Communications_Requirements_Report_10-05-2010.pdf. Acesso em: 11 Jun. 2019.

DUARTE, L. F. C., ZAMBIACO, J. D.; AIROLDI, D., FERREIRA, E. C.; DIAS, J. A. S. **Characterization and breakdown of the electricity bill using custom smart meters:** a tool for energy-efficiency programs. (Artigo). International journal of circuits, system and signal processing, 2011.

DUARTE, O. C. M. B. **Arquitetura, Redes de Computadores I.** Universidade Federal do Rio de Janeiro – Escola Politécnica, 2012. Disponível em: https://www.gta.ufrj.br/grad/12_1/seg_smartgrid/arquitetura.html. Acesso em: 05 Maio. 2019.

ERIKSSON, O. **Error Control in Wireless Sensor Networks:** A Process Control Perspective. Uppsala University, 2011.

FARHANGI, H. **The path of the smart grid.** (Artigo). IEEE Power and Energy Magazine, vol. 8, pp. 18-28, Jan-Fev. 2010.

FREEMAN, R. L., **Fundamental of Telecommunications.** John Wiley & Sons Ltd, 2005.

GALLAGER, R. G. **Low-Density Parity-Check Codes,** Ph.D. thesis, Cambridge, MA: MIT Press, 1963.

GALOY, M. J. E. **Notes on digital coding,** Proceedings of the I. R. E. (I. E. E. E.), vol. 37, 1949.

GARG, A. **Review of Wireless Local Area Network (WLAN) Standards and Wireless Sensor Networks.** (Artigo). International Journal of Enhanced Research in Science Technology & Engineering, vol. 3, no 5, pp 115-121, 2014.

GRZEIDAK, E.; CORMANE, J.; FERREIRA FILHO, A. L.; ASSIS, F. **Qualidade da energia elétrica no contexto de smart grid.** IX Conferência Brasileira de Qualidade da Energia Elétrica (CBQEE), 2011.

GUNGOR, V. C.; BIN, L.; HANCKE, G. P. **Opportunities and challenges of wireless sensor networks in smart grid.** (Artigo). IEEE Transactions on Industrial Electronics, vol. 57, no. 10, pp. 3557–3564, 2010.

GUNGOR, V. C.; KOCAK, T.; ERGUT, S.; BUCCELLA C.; CECATI C, et al. **A survey on smart grid potential applications and communication requirements.** IEEE Trans. Ind. Inform, vol 9, no. 1, pp. 28-42, Fev. 2013.

GUTIERREZ, J. A, et al. **IEEE 802.15.4:** a developing standard for low-power low-cost wireless personal area networks. IEEE Network, vol 15, i. 5, Set-Out. 2001

HAMMING, R. W. **Error detecting and error correcting codes**, The Bell System Technical Journal, vol. 26, 1950.

HAYKIN, S. **Sistemas de Comunicação: Analógicos e Digitais,** 4th ed. Bookman, 2004.

HENKEL, J. **Software Architecture Design of Wireless Sensor Networks,** Universidade de Wisconsin. 20--.

IEEE 802.11 - **IEEE Standard for Information technology** - Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 2016.

IEEE 802.11b - **IEEE Standard for Information Technology** - Telecommunications and information exchange between systems - Local and Metropolitan networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher Speed Physical Layer (PHY) Extension in the 2.4 GHz band, 1999.

IEEE 802.11g - **IEEE Standard for Information technology** - Local and metropolitan area networks-- Specific requirements-- Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Furthere Higher Data Rate Extension in the 2.4 GHz Band, 2003.

IEEE P802.11nTM/D1.02. **Draft Amendment to STANDARD Information Technology Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Enhancements for Higher Throughput**, Documento IEEE 802.11, Out. 2009.

ITU G.9960. **Unified high-speed wire-line based home networking transceivers – System architecture and physical layer specification,** Geneva, ITU-T Std., Jun. 2010.

JOHNSON, S. J. **Iterative Error Correction Turbo, Low-Density Parity-Check and Repeat-Accumulate Codes**, Cambridge, 2009.

JUANG, P.; OKI, H., WANG, Y.; MARTONOSI, M.; PEH, L. S., RUBENSTEIN, D. **Energy-efficient computing for wildlife tracking:** design tradeoffs and early experiences with zebranet. (Artigo). 2002.

KAYASTHA, N.; NIYATO, D.; HOSSAIN, E.; HAN, Z. **Smart grid sensor data collection, communication,and networking:** a tutorial. (Artigo). Wireless Communications and Mobile Computing, 2012.

KIM, Y. J.; THOTTAN, M.; KOLESNIKOV, V.; LEE, W., **A secure decentralized data-centric information infrastructure for smart grid.** IEEE Communications Magazine, vol. 48, no. 11, pp. 58–65, 2010.

LENNVALL, T.; SVENSSON, S. **A Comparison of WirelessHART and ZigBee for Industrial Applications.** 2008 IEEE International Workshop on Factory Communication Systems. Out. 2008.

LI, W.; ZHANG, X. **Simulation of the smart grid communications:** Challenges, techniques, and future trends. Computers and Electrical Engineering, vol 40, pp. 270-288. 2014

LIN, S.; COSTELO. Jr., D. **Error Control Coding: Fundamentals and Applications.** Prentice-Hall, 1983.

LOUREIRO, A. A.; NOGUEIRA, J. M. S.; RUIZ, L. B.; DE FREITAS MINI, R. A.; NAKAMURA, E. F.; FIGUEIREDO, C. M. S. **Redes de sensores sem fio.** Simpósio Brasileiro de Redes de Computadores, SBRC, 21 o :179–116. 2003.

MACKAY, D. J. C. **Good error-correcting codes based on very sparse matrices,** IEEE Transactions on Information Theory, vol. 45, pp. 399-431, Mar. 1999.

MICAZ. **Datasheet:** MICAz, Tiny Wireless Measurement System. Electronic Publication, 20--.

MOREIRA, J. C.; FARRELL, P. G. **Essentials of Error-Control Coding.** John Wiley e Sons, Ltd., 2006.

MORELOS-ZARAGOZA, R. H. **The Art of Error correcting Coding.** John Wiley e Sons, Ltd., 2002.

MYUNG, S.; YANG, K.; KIM, J. **Quasi-cyclic LDPC codes for fast encoding,** IEEE Trans. Inf. Theory, vol. 51, pp. 2894-2900, Ago. 2005.

NMENTORS. Projetos de Smart Grid; **nMentors**, 2013. Disponível em: http://www.nmentors.com.br/portfolio/portfolio_projetos_smart_grid.htm. Acesso em: 16 Jun, 2018.

PINTO, P. **Afinal o que é uma rede de sensores sem fios?** 2015**.** Disponível em: https://pplware.sapo.pt/tutoriais/networking/afinal-o-que-e-uma-rede-de-sensores-sem-fios-parte-i. Acesso em: junho. 2018.

PRATHAP, U.; SHENOY, P. D.; VENUGOPAL, K.; PATNAIK, L. **Wireless sensor networks applications and routing protocols: Survey and research challenges.** (Artigo). In Cloud and Services Computing (ISCOS), International Symposium on, pp. 49–56. IEEE. 2012.

PROAKIS, J. G, SALEHI, M. **Digital communications.** 5. ed, McGraw-Hill. 2008.

REED, I; SOLOMON, G. **Polynomial codes over certain infite fields,***"*SIAM Journal of Applied Math, vol. 8, pp. 300-304, 1960.

RICHARDSON, T. J., URBANKE, R. **Eficient encoding of low-density parity-check codes.** (Artigo). IEEE Trans. Inf. Theory, vol. 47, no. 2, pp. 638-656, Fev, 2001.

ROCHA FILHO, G. P. **Um sistema de alerta para o mointoramento remoto do consumo de energia usando redes de sensores sem fio.** São Carlos. Dissertação (Mestrado em Ciências de Computação e Matemática) – Universidade de São Paulo, 2014.

RYAN, W. E.; LIN, S. **Channel Codes:** Classical and Modern, Cambridge University Press, 2009.

SAHIN, D., GUNGOR, V. C., KOCAK, T., TUNA, G. **Quality-of-service differentiation in single-path and multi-path routing for wireless sensor network-based smart grid applications.** Ad Hoc Networks, vol 22, pp. 43–60, 2014.

SCARPIN, B. Medidores de Energia Inteligentes – Vantagens e Desafios; **Cubienergia**, 2018. Disponível em: https://www.cubienergia.com/medidores-de-energia-inteligentes. Acesso em: 10 Jun, 2019.

SCIAMANA, M. Medição Inteligente; **Portal O Setor Elétrico**, 2010. Disponível em: https://www.osetoreletrico.com.br/medicao-inteligente. Acesso em: 08 Jun, 2019.

SHANNON, C. E. **A mathematical theory of communication**, The Bell System Technical Journal, vol. 28, 1948.

SILVEIRA, C. B. O que é um Medidor de Energia Elétrica Inteligente; **Citisystems**, sem data. Disponível em: https://www.cubienergia.com/medidores-de-energia-inteligentes. Acesso em: 08 Jun. 2019.

SKLAR, B. **Digital communications: Fundamentals and applications,** 2nd ed. Prentice Hall, 2001.

STAROSTA, J. Os desafios técnicos da geração distribuída; **Portal O Setor Elétrico**, 2017. Disponível em: https://www.osetoreletrico.com.br/os-desafios-tecnicos-da-geracao-distribuida/. Acesso em: 05 Jul, 2019.

TANNER, R. M. **A recursive approach to low complexity codes,** IEEE Transactions on Information Theory, vol. IT-27, no. 5, pp. 533-547, Set. 1981.

TUNA, G.; GUNGOR, V. C.; GULEZ, K. **Wireless Sensor Networks for Smart Grid Applications: A Case Study on Link Reliability and Node Lifetime Evaluations in Power Distribution Systems.** (Artigo). International Journal of Distributed Sensor Networks, vol. 9, no. 2, 2013

YICK, J., MUKHERJEE, D., GHOSAL, D. **Wireless sensor network survey.** (Artigo). Computer Networks, vol 52, Issue, pp. 2292-2330. Ago, 2008.

YU, M.; ANSARI, N. **Smart grid communications: Modeling and validation**. Journal of Network and Computer Applications, vol 59, pp. 247-249, Jan. 2016.

Apêndice A – Datasheet do nó sensor MICAz

MEMSIC

MICAz

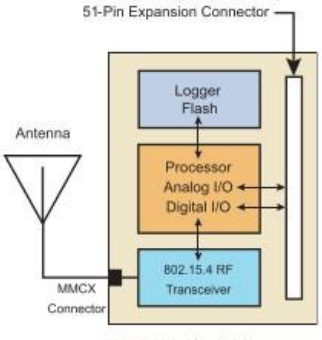
WIRELESS MEASUREMENT SYSTEM

- 2.4 GHz IEEE 802.15.4, Tiny Wireless Measurement System
- Designed Specifically for Deeply Embedded Sensor Networks
- 250 kbps, High Data Rate Radio
- Wireless Communications with Every Node as Router Capability
- Expansion Connector for Light, Temperature, RH, Barometric Pressure, Acceleration/Seismic, Acoustic, Magnetic and other MEMSIC Sensor Boards

Applications

- Indoor Building Monitoring and Security
- Acoustic, Video, Vibration and Other High Speed Sensor Data
- Large Scale Sensor Networks (1000+ Points)

ZigBee™Alliance



MPR2400 Block Diagram

MICAz

The MICAz is a 2.4 GHz Mote module used for enabling low-power, wireless sensor networks.

Product features include:

- IEEE 802.15.4 compliant RF transceiver
- 2.4 to 2.48 GHz, a globally compatible ISM band
- Direct sequence spread spectrum radio which is resistant to RF interference and provides inherent data security
- 250 kbps data rate
- Supported by MoteWorks™ wireless sensor network platform for reliable, ad-hoc mesh networking
- Plug and play with MEMSIC's sensor boards, data acquisition boards, gateways, and software

MoteWorks™ enables the development of custom sensor applications and is specifically optimized for low-power, battery-operated networks. MoteWorks is based on the open-source TinyOS operating system and provides reliable, ad-hoc mesh networking, over-the-air-programming capabilities, cross development tools, server middleware for enterprise network integration and client user interface for analysis and a configuration.

Processor & Radio Platform (MPR2400CA)

The MPR2400 is based on the Atmel ATmega128L. The ATmega128L is a low-power microcontroller which runs MoteWorks from its internal flash memory. A single processor board (MPR2400) can be configured to run your sensor application/ processing and the network/radio communications stack simultaneously. The 51-pin expansion connector supports Analog Inputs, Digital I/O, I2C, SPI and UART interfaces. These interfaces make it easy to connect to a wide variety of external peripherals. The MICAz (MPR2400) IEEE 802.15.4 radio offers both high speed (250 kbps) and hardware security (AES-128).

Sensor Boards

MEMSIC offers a variety of sensor and data acquisition boards for the MICAz Mote. All of these boards connect to the MICAz via the standard 51-pin expansion connector. Custom sensor and data acquisition boards are also available. Please contact MEMSIC for additional information.

Processor/Radio Board	MPR2400CA	Remarks
Processor Performance		
Program Flash Memory	128K bytes	
Measurement (Serial) Flash	512K bytes	> 100,000 Measurements
Configuration EEPROM	4K bytes	
Serial Communications	UART	0-3V transmission levels
Analog to Digital Converter	10 bit ADC	8 channel, 0-3V input
Other Interfaces	Digital I/O, I2C, SPI	
Current Draw	8 mA	Active mode
	< 15 µA	Sleep mode
RF Transceiver		
Frequency band[1]	2400 MHz to 2483.5 MHz	ISM band, programmable in 1 MHz steps
Transmit (TX) data rate	250 kbps	
RF power	-24 dBm to 0 dBm	
Receive Sensitivity	-90 dBm (min), -94 dBm (typ)	
Adjacent channel rejection	47 dB	+ 5 MHz channel spacing
	38 dB	- 5 MHz channel spacing
Outdoor Range	75 m to 100 m	1/2 wave dipole antenna, LOS
Indoor Range	20 m to 30 m	1/2 wave dipole antenna
Current Draw	19.7 mA	Receive mode
	11 mA	TX, -10 dBm
	14 mA	TX, -5 dBm
	17.4 mA	TX, 0 dBm
	20 µA	Idle mode, voltage regular on
	1 µA	Sleep mode, voltage regulator off
Electromechanical		
Battery	2X AA batteries	Attached pack
External Power	2.7 V - 3.3 V	Molex connector provided
User Interface	3 LEDs	Red, green and yellow
Size (in)	2.25 x 1.25 x 0.25	Excluding battery pack
(mm)	58 x 32 x 7	Excluding battery pack
Weight (oz)	0.7	Excluding batteries
(grams)	18	Excluding batteries
Expansion Connector	51-pin	All major I/O signals

Notes
[1] 5 MHz steps for compliance with IEEE 802.15.4/D18-2003.
Specifications subject to change without notice

MIB520CB Mote Interface Board

Base Stations

A base station allows the aggregation of sensor network data onto a PC or other computer platform. Any MICAz Mote can function as a base station when it is connected to a standard PC interface or gateway board. The MIB510 or MIB520 provides a serial/USB interface for both programming and data communications. MEMSIC also offers a stand-alone gateway solution, the MIB600 for TCP/IP-based Ethernet networks.

Ordering Information

Model	Description
MPR2400CA	2.4 GHz MICAz Processor/Radio Board
WSN-START2400CA	2.4 GHz MICAz Starter Kit
WSN-PRO2400CA	2.4 GHz MICAz Professional Kit

Apêndice B – Código RS implementado em C++

```
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
#include <time.h>
#include <math.h>
#include <vector>

#include <itpp/itcomm.h>
#include <itpp/base/vec.h>


#include "my_rs.h"
#include "my_rand.h"
#include "rs_utils.h"
#include "boolchar.h"

#include <iostream>
#include <fstream>
#include <string>

using namespace itpp;
using namespace std;


int main(int argc, char ** argv) {

    int K = 239;
    int N = 255;
    int m = 8;
    int seed = 5;

    my_rand_init_with_time();
    Random* r = my_rand_init_with_time();

    RS* rs = my_rs_new(N,K);

    int EbN0dBsize = 310;
    float* mean_BER_vec = (float*) calloc(EbN0dBsize,sizeof(float));
    float* BLER_vec = (float*) calloc(EbN0dBsize,sizeof(float));
    float* SNR_vec = (float*) calloc(EbN0dBsize,sizeof(float));


    int frame = 1000;
    int cont = 0;
    int bler = 0;
    int err = 0;

    time_t tstart, tend;
    tstart = time(0);

    float target_ebno_db;
    for (target_ebno_db=0; target_ebno_db<250; target_ebno_db+=3) { //SNR1
```

```
    float* BER = (float*) calloc(frame,sizeof(float));
    float ebno2_db = target_ebno_db*pow(0.333333,3);

for(int z=0; z < frame; z++){

    unsigned char* msg = (unsigned char*) malloc(K*sizeof(unsigned char));
    unsigned char* cdwrd = (unsigned char*) malloc(N*sizeof(unsigned char));
    unsigned char* decoded_msg = (unsigned char*) malloc(K*sizeof(unsigned char));

int i;
for(i=0;i<K;i++){
    msg[i] = my_rand_next_int_ranged(r, 256);
}

/* PASSA PRA BIT */
bool* msg_bool = (bool*) malloc(K*m*sizeof(bool));

for(int j=0;j<K;j++){
    char2bool(msg[j], msg_bool + j*8);
 }


////////////////// ENCODE /////////////////


my_rs_encode(rs, cdwrd, msg);

bool* cdwrd_bool = (bool*) malloc(N*m*sizeof(bool));
int j;
for(j=0;j<N;j++){
    char2bool(cdwrd[j], cdwrd_bool + j*8);
}


////////////////// CHANNEL /////////////////

PSK bpsk(4);


Vec<bool> v (cdwrd_bool, N*m);
bvec cdwrd_v = to_bvec(v);
cvec tx_symbols = bpsk.modulate_bits(cdwrd_v);

double EbN0dB = ebno2_db;
double stddev = sqrt(pow(10,(-EbN0dB/10))); /* dB pra linear */
double N0 = pow(stddev,2); /* Variância */

AWGN_Channel awgn_channel;
awgn_channel.set_noise(N0);

cvec rx_symbols = awgn_channel(tx_symbols);
bvec decbits = bpsk.demodulate_bits(rx_symbols);
```

```
////////////////// DECODE /////////////////


bool* demodulate_bool = (bool*) malloc(N*m*sizeof(bool));

for (int l=0; l<N*m; l++) {
    demodulate_bool[l] = (bool) decbits[l]; }

/* VOLTA PRA CHAR */
unsigned char* demodulate_cdwrd = (unsigned char*) malloc(N*sizeof(unsigned char))

for(int i=0;i<N;i++){
    demodulate_cdwrd[i] = bool2char(demodulate_bool + i*8);}


int c = my_rs_decode(rs, decoded_msg, demodulate_cdwrd);

/* Passa a mensagem codificada para bits para contar a quantidade de erros */
bool* decoded_msg_bool = (bool*) malloc(K*m*sizeof(bool));

for(int j=0;j<K;j++){
    char2bool(decoded_msg[j], decoded_msg_bool + j*8);
}

err = count_err(msg_bool,decoded_msg_bool,K*m);

if(err > 0){
    bler++;
}

BER[z] = err/(float)(K*m);

free(decoded_msg);
free(cdwrd);
free(msg);
free(msg_bool);
free(cdwrd_bool);
free(decoded_msg_bool);
free(demodulate_cdwrd);
free(demodulate_bool);
}



float sum_ber = sum_array(BER, frame);
float mean_BER = sum_ber/frame;

mean_BER_vec[cont] = mean_BER;
```

```
    float BLER = bler / (float) (frame);
    BLER_vec[cont] = BLER;
    SNR_vec[cont] = (target_ebno_db * 100) / 100;
    bler = 0;
    }


    ofstream myfile;
    myfile.open ("1hop.tsv");

    myfile << "SNR \n";
        for (int i = 0; i<cont; i++) {
        myfile << SNR_vec[i] << "\n";
        }

    myfile << "BER Média \n";
        for (int i = 0; i<cont; i++) {
        myfile << mean_BER_vec[i] << "\n";
        }


    myfile.close();

    tend = time(0);
    cout << "It took "<< difftime(tend, tstart) <<" second(s)."<< endl;
    return (EXIT_SUCCESS);

}
```

Apêndice C – Código LDPC implementado em C++.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>
#include <math.h>
#include <vector>


extern "C" {
#include "encode.h"
#include "decode.h"
}

#include"my_rand.h"
#include"punct.h"
#include"depunct.h"
#include"encode.h"
#include"decode.h"
#include"pchk_reader.h"
#include"util.h"

#include <itpp/itcomm.h>
#include <itpp/base/vec.h>

#include <iostream>
#include <fstream>
#include <string>


using namespace itpp;
using namespace std;


int** read_pchk(char filename[], int* k,int* n,int* L);
float rate_dmt_sim_ber(int n_frames, int N_cdwrd, double* SNRdB, int* bitload, int N_sub, int** H, int m,
int n, int L, int pp, int n_hops);



int main(int argc, char ** argv) {


  int n_frames = atof(argv[1]);  // 1º argumento: Nº de frames
  int n_hops = atoi(argv[2]);  // 2º argumento: Nº de hops

 time_t tstart, tend;
 tstart = time(0);

 srand(time(NULL));

 int k,L,m,n;
    char filename[] = "ldpc(2_3)L"; /* Escolher matriz do G.hn */
    int** H = read_pchk(filename,&k,&n,&L);
```

```
    m = n-k;

    int b = 2;  /* Numero de bits, ordem da modulacao 2^b */

    int cont = 0;
    int snr_length = 200;

  int N_cdwrd = 1;
  int N_sub = N_cdwrd * n* L / b;
  double* mean_SNRdB = (double*) calloc(snr_length, sizeof(double));
  double* mean_ber = (double*) calloc(snr_length, sizeof(double));
  double* SNRdB = (double*) calloc(N_sub, sizeof(double));
  int* bitload = (int*) calloc(N_sub,sizeof(int));
  int pp = 0;

   for (double snr= 0; snr<95; snr+=1) {
        for (int jj = 0; jj< N_sub; jj++){
        SNRdB[jj] = snr;
        bitload[jj] = b;
        }

        printf("%d %d %d\n", k*L, n*L, m);

        double ber = rate_dmt_sim_ber(n_frames, N_cdwrd, SNRdB, bitload, N_sub, H, m, n, L, pp, n_hops);

        mean_ber[cont] = ber;
        mean_SNRdB[cont] = snr;
}


   ofstream myfile;
    myfile.open ("2hops.tsv");

    myfile << "SNR \n";
        for (int i = 0; i<cont; i++) {
        myfile << mean_SNRdB[i] << "\n";
        }

    myfile << "BER Média \n";
        for (int i = 0; i<cont; i++) {
        myfile << mean_ber[i] << "\n";
        }

    myfile.close();


tend = time(0);
cout << "It took "<< difftime(tend, tstart) <<" second(s)."<< endl;

    return (EXIT_SUCCESS);
}
```

```
}


void qam_awgn_llr_ghn_approx(Random* r,float* llr, bool* encoded, int size,int M_QAM, double SNRdB, int
hops){
        //------------------------------------------------------------
        //This function simulates the path between LDPC encoder and decoder,
        //IT USES THE APPROXIMATED GHN LLR GENERATION (Developed at LaPS)
        //  it receives a vector of previously encoded bools, [encoded]
        //  and returns a vector of resulting llr's, [llr]
        //  both vectors have size [size],
        //  all bits will be mapped in a [M_QAM] constellation
        //  and the gausian noise added have a standart deviation [stddev]
        //------------------------------------------------------------

        //initial parameters
        int i;
        int bits_per_symbol = (int)log2(M_QAM);
        int nb_mapped_symbols = size/bits_per_symbol;
        double stddev = sqrt(pow(10,(-SNRdB/10)));

        double SNR2 = SNRdB*pow(0.5,3);;
        double stddev2 = sqrt(pow(10,(-SNR2/10)));

        double SNR3 = SNRdB*pow(0.333333,3);
        double stddev3 = sqrt(pow(10,(-SNR3/10)));


        //initial allocation
        int *condensed_bools= (int*)malloc(nb_mapped_symbols*sizeof(int));
        float **mapped_symbols = (float**)malloc(nb_mapped_symbols*sizeof(float*));
        for (i=0; i<nb_mapped_symbols; i++)
                mapped_symbols[i] = (float*)malloc(2*sizeof(float));
        float noise_I, noise_Q;
        double *aproximate_llr = (double*)malloc(size*sizeof(double));

        condense_bools(encoded, size ,bits_per_symbol, condensed_bools);

        ghn_mapper(condensed_bools, nb_mapped_symbols, bits_per_symbol, mapped_symbols);

#ifdef DEBUG
        printf("\n\nAPPROX: symbols: stddev: %f\n", stddev);
        for (i=0; i<nb_mapped_symbols; i++)
                printf("%d ", condensed_bools[i]);

        printf("\nmodulated sym: %d\n", nb_mapped_symbols);
        for (i=0; i<nb_mapped_symbols; i++)
                printf("%f+%fi\n", mapped_symbols[i][0], mapped_symbols[i][1]);

        printf("noise:\t\tnoisy symbol:\n");
#endif
```

```
        switch(hops) {
                case 1 : // 1 hop

               /////// RUÍDO 1 ///////

                for(i=0; i<nb_mapped_symbols; i++){
                        #if 1
                        double* c = (double*) malloc(2*sizeof(double));
                        my_rand_next_complex_gaussian(r,c);
                        noise_I = stddev3*c[0];
                        noise_Q = stddev3*c[1];
                        free(c);
                        //cout << stddev2 << endl;
                        #else
                        noise_I = noise[i].real();
                        noise_Q = noise[i].imag();
                        #endif
                        mapped_symbols[i][0] += noise_I;
                        mapped_symbols[i][1] += noise_Q;
                        #ifdef DEBUG
                                printf("%f+j%f\t\t%f+j%f\n", noise_I, noise_Q, mapped_symbols[i][0],
mapped_symbols[i][1]);
                        #endif
                }

                        break;
                case 2 : // 2 hop

               /////// RUÍDO 1 ///////

                for(i=0; i<nb_mapped_symbols; i++){
                        #if 1
                        double* c = (double*) malloc(2*sizeof(double));
                        my_rand_next_complex_gaussian(r,c);
                        noise_I = stddev*c[0];
                        noise_Q = stddev*c[1];
                        free(c);
                        #else
                        noise_I = noise[i].real();
                        noise_Q = noise[i].imag();
                        #endif
                        mapped_symbols[i][0] += noise_I;
                        mapped_symbols[i][1] += noise_Q;
                        #ifdef DEBUG
                                printf("%f+j%f\t\t%f+j%f\n", noise_I, noise_Q, mapped_symbols[i][0],
mapped_symbols[i][1]);
                        #endif
                }
```

```
                ////// RUÍDO 2 //////

                for(i=0; i<nb_mapped_symbols; i++){
                        #if 1
                        double* c = (double*) malloc(2*sizeof(double));
                        my_rand_next_complex_gaussian(r,c);
                        noise_I = stddev2*c[0];
                        noise_Q = stddev2*c[1];
                        free(c);
                        #else
                        noise_I = noise[i].real();
                        noise_Q = noise[i].imag();
                        #endif
                        mapped_symbols[i][0] += noise_I;
                        mapped_symbols[i][1] += noise_Q;
                        #ifdef DEBUG
                                printf("%f+j%f\t\t%f+j%f\n", noise_I, noise_Q, mapped_symbols[i][0],
mapped_symbols[i][1]);
                        #endif
                }

                        break;
                case 3 : // 3 hop

                ////// RUÍDO 1 //////

                for(i=0; i<nb_mapped_symbols; i++){
                        #if 1
                        double* c = (double*) malloc(2*sizeof(double));
                        my_rand_next_complex_gaussian(r,c);
                        noise_I = stddev*c[0];
                        noise_Q = stddev*c[1];
                        free(c);
                        #else
                        noise_I = noise[i].real();
                        noise_Q = noise[i].imag();
                        #endif
                        mapped_symbols[i][0] += noise_I;
                        mapped_symbols[i][1] += noise_Q;
                        #ifdef DEBUG
                                printf("%f+j%f\t\t%f+j%f\n", noise_I, noise_Q, mapped_symbols[i][0],
mapped_symbols[i][1]);
                        #endif
                }
```

```
                ////// RUÍDO 2 //////

                for(i=0; i<nb_mapped_symbols; i++){
                        #if 1
                        double* c = (double*) malloc(2*sizeof(double));
                        my_rand_next_complex_gaussian(r,c);
                        noise_I = stddev*c[0];
                        noise_Q = stddev*c[1];
                        free(c);
                        #else
                        noise_I = noise[i].real();
                        noise_Q = noise[i].imag();
                        #endif
                        mapped_symbols[i][0] += noise_I;
                        mapped_symbols[i][1] += noise_Q;
                        #ifdef DEBUG
                                printf("%f+j%f\t\t%f+j%f\n", noise_I, noise_Q, mapped_symbols[i][0],
mapped_symbols[i][1]);
                        #endif
                }

                ////// RUÍDO 3 //////

                for(i=0; i<nb_mapped_symbols; i++){
                        #if 1
                        double* c = (double*) malloc(2*sizeof(double));
                        my_rand_next_complex_gaussian(r,c);
                        noise_I = stddev*c[0];
                        noise_Q = stddev*c[1];
                        free(c);
                        #else
                        noise_I = noise[i].real();
                        noise_Q = noise[i].imag();
                        #endif
                        mapped_symbols[i][0] += noise_I;
                        mapped_symbols[i][1] += noise_Q;
                        #ifdef DEBUG
                                printf("%f+j%f\t\t%f+j%f\n", noise_I, noise_Q, mapped_symbols[i][0],
mapped_symbols[i][1]);
                        #endif
                }

                        break;
        }



        ghn_demapper_llr_approx(mapped_symbols, nb_mapped_symbols, bits_per_symbol, stddev*stddev,
aproximate_llr);

        for(i=0;i<size;i++){
                llr[i] = (float)aproximate_llr[i];
        }

        free(aproximate_llr);
        free(condensed_bools);
        for (i=0; i<nb_mapped_symbols; i++)
                free(mapped_symbols[i]);
        free(mapped_symbols);
}
```