UNIVERSIDADE FEDERAL DO ABC
CENTRO DE ENGENHARIA, MODELAGEM E
CIÊNCIAS SOCIAIS APLICADAS
PROGRAMA DE GRADUAÇÃO EM ENGENHARIA DE INFORMAÇÃO

RAFAEL MONTOIA DA SILVA

APLICAÇÃO DE UM SISTEMA EMBARCADO BASEADO EM ZIGBEE PARA MONITORAMENTO DE VIAS FÉRREAS

Santo André - SP

2022

RAFAEL MONTOIA DA SILVA

APLICAÇÃO DE UM SISTEMA EMBARCADO BASEADO EM ZIGBEE PARA MONITORAMENTO DE VIAS FÉRREAS

Trabalho de Graduação apresentado ao curso de graduação em Engenharia de Informação da Universidade Federal do ABC, como requisito parcial para a obtenção do título de Bacharel em Engenharia de Informação

Orientador: Prof. Dr. Ivan Roberto de Santana Casella

Santo André - SP

2022

Universidade Federal do ABC

ATA DE DEFESA DE TRABALHO DE GRADUAÇÃO EM ENGENHARIA DE INFORMAÇÃO

Ata de Defesa do Trabalho de Graduação em Engenharia de Informação da Universidade Federal do ABC

No dia **07 de dezembro de 2022** reuniu-se a banca examinadora do trabalho apresentado como Trabalho de Graduação em Engenharia de Informação de **RAFAEL MONTOIA DA SILVA**, intitulado: **APLICAÇÃO DE UM SISTEMA EMBARCADO BASEADO EM ZIGBEE PARA MONITORAMENTO DE VIAS FÉRREAS**. Após a exposição oral, o aluno foi arguido pelos componentes da banca que se reuniram reservadamente, e decidiram atribuir o conceito final **A**.

gov.br Documento assinado digitalmente
IVAN ROBERTO SANTANA CASELLA
Data: 07/12/2022 17:58:51-0300
Verifique em https://verificador.iti.br

_____

Orientador

Prof. Dr. Ivan R. S. Casella

gov.br Documento assinado digitalmente
JOAO HENRIQUE KLEINSCHMIDT
Data: 08/12/2022 08:20:35-0300
Verifique em https://verificador.iti.br

_____

Avaliador 1

Prof. Dr. João Henrique Kleinschmidt

gov.br Documento assinado digitalmente
RODRIGO REINA MUNOZ
Data: 07/12/2022 19:11:55-0300
Verifique em https://verificador.iti.br

_____

Avaliador 2

Prof. Dr. Rodrigo Reina Muñoz

Agradecimentos

Agradeço a todos da minha família que me ajudaram durante esse período, ao meu professor que foi de vital importância para o desenvolvimento desse trabalho e aos meus amigos da Companhia Paulista de Trens Metropolitanos (CPTM) que me ajudaram na realização dos testes do protótipo, sem os quais não conseguiria realizar sozinho.

Resumo

Este trabalho visa o estudo e construção de um sistema embarcado de comunicação para monitoramento e controle dos equipamentos de sinalização ferroviários baseado no padrão ZigBee considerando as topologias em estrela e *mesh*. Para isso foi realizado primeiro um estudo sobre o padrão ZigBee e sobre sistemas embarcados, para assim nortear a construção do mesmo e determinar os principais testes a serem realizados após a construção para validação do sistema proposto. O desempenho do sistema embarcado foi analisado através de simulações computacionais e testes em ambientes internos, externos e no próprio ambiente ferroviários para verificar seu comportamento nas mais diversas situações. O sistema embarcado de comunicação baseado em ZigBee tem o objetivo de substituir o atual sistema de sinalização ferroviária, onde o controle e monitoramento dos equipamentos ocorrem através do uso de cabos elétricos de cobre, que estão sujeitos a furtos, vandalismos e problemas de baixa isolação, assim diminuindo custos e prejuízos às empresas ferroviárias. Os resultados dos testes mostraram que o sistema proposto atende os requisitos necessários para as operações de monitoramento e controle dos equipamentos de sinalização ferroviários.

Palavras-chaves: ZigBee, Sinalização ferroviária, Arduino, XBee, Comunicação sem fio.

Abstract

This work aims at the study and construction of an embedded communication system based on the ZigBee standard for monitoring and controlling railway signaling equipment. For this, a study was first carried out on the ZigBee standard and on embedded systems, in order to guide its construction and determine the main tests to be carried out after construction for validation of the proposed system. The performance of the embedded system was analyzed through computer simulations and tests in internal and external environments and in the railway environment itself to verify its behavior in the most diverse situations. The embedded ZigBee-based communication system aims to replace the current railway signaling system, where the control and monitoring of equipment occur through the use of copper electrical cables, which are subject to theft, vandalism and low insulation problems, thus reducing costs and losses to railway companies. The results of the tests carried out with the proposed system were satisfactory, indicating that it met all the elaborated requirements, showing its validation.

Keywords: ZigBee, Railway signaling, Arduino, XBee, Wireless communication.

Lista de ilustrações

Figura 1 – Mapa das principais ferrovias do Brasil. 2
Figura 2 – Classificação e exemplos das tecnologias de rede sem fio. 5
Figura 3 – Componentes da via permanente. 10
Figura 4 – Exemplos de material rodante. 11
Figura 5 – Sinaleiro manual. 11
Figura 6 – (a) Sinaleiro plurifocal. (b) Sinaleiro unifocal. 12
Figura 7 – Sinaleiro anão (*Searchlight*). 13
Figura 8 – Mecanismo da unidade de sinal *Searchlight*. 13
Figura 9 – Esquema elétrico simplificado do mecanismo *Searchlight*. 14
Figura 10 – Exemplo de circuito de via com ocupação. 15
Figura 11 – Condição do circuito de via livre. 16
Figura 12 – Condição do circuito de via ocupado. 16
Figura 13 – Esquema de chave. 17
Figura 14 – Exemplos de arranjos de chaves. 18
Figura 15 – Exemplo de máquina de chave. 20
Figura 16 – Exemplo de IHM no CCO. 22
Figura 17 – Diagrama de conexão entre campo e CCO. 24
Figura 18 – Camadas do padrão ZigBee. 25
Figura 19 – Pacote da camada física. 27
Figura 20 – Faixas de frequências e canais utilizados. 28
Figura 21 – Tipos de modulação. 29
Figura 22 – Diagrama de blocos de modulação DSSS/OQPSK na frequência de 2.45 GHz. 29
Figura 23 – Sequências PN para cada símbolo de dados usadas pelo padrão ZigBee. 30
Figura 24 – Deslocamento dos *chips* entre I e Q. 31
Figura 25 – Exemplo de amostra de sequência de *chips* em um sinal passa-baixa com formatação de pulso filtrado com *half-sine*. 31
Figura 26 – Resumo das modulações de acordo com as frequências. 32
Figura 27 – Estrutura do superquadro. 34
Figura 28 – Gerenciamento dos *slots* GTS. 35
Figura 29 – Topologias do padrão ZigBee. 39
Figura 30 – Camada de aplicação e sua divisão. 39
Figura 31 – Exemplo de estrutura de um sistema embarcado. 41
Figura 32 – Placa Arduino Mega 2560. 42
Figura 33 – Características do Arduino Mega 2560. 43
Figura 34 – Plataforma de desenvolvimento do Arduino. 45

Figura 35 – Principais características do XBee-PRO S2. 46
Figura 36 – Fluxo de dados entre um dispositivo e XBee. 47
Figura 37 – Exemplo do envio de dados do pacote 0x1F (31 em decimal). 47
Figura 38 – Sequência de *frame* no modo AP = 1. 48
Figura 39 – Sequência de *frame* no modo AP = 2. 48
Figura 40 – Estrutura do *data frame*. 49
Figura 41 – Tipos de *frame*. 50
Figura 42 – Fluxo da UART no comando AT. 52
Figura 43 – Fluxo da UART na transmissão e recepção de dados via radio frequência. 52
Figura 44 – Fluxo da UART no comando AT remoto. 52
Figura 45 – Fluxo da UART na criação de rotas. 53
Figura 46 – *XBee* (esquerda) e adaptador (direita) utilizado. 56
Figura 47 – Módulo leitor de cartão micro SD. 57
Figura 48 – Vista frontal e traseira do módulo RTC DS3231. 57
Figura 49 – Esquema de conexões dos componentes ao Arduino Mega. 58
Figura 50 – Ilustração das conexões dos módulos. 59
Figura 51 – Montagem do protótipo A. 60
Figura 52 – Montagem do protótipo B. 61
Figura 53 – Montagem do protótipo C. 61
Figura 54 – Resumo das conexões dos dispositivos. 62
Figura 55 – Tabela com os principais parâmetros alterados no XBee. 63
Figura 56 – Simulação do padrão ZigBee no Simulink. 64
Figura 57 – Bloco TX. 64
Figura 58 – Bloco RX. 65
Figura 59 – Resultados da BER em função da SNR para canais AWGN no Simulink para o padrão Zigbee e a modulação OQPSK. 66
Figura 60 – Corredor do subsolo do bloco A da UFABC em Santo André. 67
Figura 61 – TX e RX em seus respectivos suportes e alinhados. 68
Figura 62 – Medidas de RSSI em função da distância. 69
Figura 63 – Tela do analisador de espectro a 10 m com TX ligado. 70
Figura 64 – Tela do analisador de espectro a 10 m com TX desligado (ruído). . . . 70
Figura 65 – Valores de SNR para cada distância. 71
Figura 66 – Exemplos de cálculo na base 256. 72
Figura 67 – Valores de PER para cada distância. 73
Figura 68 – Campus da FATEC vista pelo *software* Google Earth. 74
Figura 69 – Perfil de elevação vista pelo *software* Google Earth. 74
Figura 70 – Posição do TX. 75
Figura 71 – Posição do RX a 100 m. 76
Figura 72 – Posição do RX a 200 m. 76

Figura 73 – Posição do RX a 250 m. 77
Figura 74 – RSSI em função da distância. 77
Figura 75 – PER em função da distância. 78
Figura 76 – Percurso para teste de RSSI. 79
Figura 77 – Foto do posicionamento do RX. 80
Figura 78 – Foto do posicionamento do RX com visão para o percurso do teste. . . 81
Figura 79 – (a) Foto do posicionamento do TX com visão para o dispositivo RX. (b) Visão para o percurso do teste. 81
Figura 80 – Resultado do teste de RSSI. 82
Figura 81 – Percurso para teste de PER. 83
Figura 82 – Local do Coordenador. 84
Figura 83 – Local do Roteador A. 84
Figura 84 – Local do Roteador B. 85
Figura 85 – Visão do Coordenador e Roteador A. 86
Figura 86 – (a) Visão do Roteador A para o Roteador B. (b) Visão do Roteador B para o Roteador A. 86
Figura 87 – Visão reta alinhada com o Roteador B. 87
Figura 88 – Log gerado pelo Coordenador. 88
Figura 89 – Log gerado pelo Roteador A. 89
Figura 90 – Resultado do teste de PER no ambiente ferroviário. 90
Figura 91 – Arquitetura da rede obtida pelo *software* XCTU. 91
Figura 92 – *Hardware* ZENA. 92
Figura 93 – Pacote de transmissão com conteúdo do *payload* "Oi". 93
Figura 94 – Exemplo do uso de rotas na rede *mesh*. 93
Figura 95 – Exemplo do envio da chave de rede para ingressantes pelo Coordenador. 94
Figura 96 – Exemplo do envio de pacote com criptografia aplicada. 94
Figura 97 – Potência média medida no ambiente *indoor* em função da distância e a obtida pelo MPS correspondente. 97
Figura 98 – Potência média medida no ambiente *indoor* em função da distância e a potência instantânea estimada pelo MPS correspondente, considerando a dispersão causada pelo efeito de *shadowing*. 97
Figura 99 – Potência média medida no ambiente *outdoor* em função da distância e a obtida pelo MPS correspondente. 98
Figura 100 – Potência média medida no ambiente *outdoor* em função da distância e a potência instantânea estimada pelo MPS correspondente, considerando a dispersão causada pelo efeito de *shadowing*. 98
Figura 101 – Potência média medida no ambiente ferroviário em função da distância e a obtida pelo MPS correspondente. 99

Figura 102 – Potência média medida no ambiente ferroviário em função da distância e a potência instantânea estimada pelo MPS correspondente, considerando a dispersão causada pelo efeito de *shadowing*. 100
Figura 103 – (a) Constelação QPSK1. (b)Constelação QPSK2 107
Figura 104 – Densidade espectral de potência de um sinal QPSK para formatação de pulsos retangulares e pulsos filtrados com cosseno elevado 108
Figura 105 – Diagrama em bloco de um transmissor QPSK 109
Figura 106 – Diagrama em bloco de um receptor QPSK 109
Figura 107 – Diagrama de blocos do circuito de recuperação 110
Figura 108 – Formas de onda em fase e quadratura aplicadas a um modulador OQPSK. 111
Figura 109 – (a) Diagrama em blocos de um sistema DSSS com modulação BPSK transmissor. (b) receptor 112
Figura 110 – (a) Espectros do sinal recebido com interferência na saída do filtro. (b) saída do correlator após o desespalhamento 114
Figura 111 – Exemplo de transmissão e recepção de uma sequência de dados com o espalhamento. 114
Figura 112 – Circuito elétrico da placa Arduino Mega 2560. 116
Figura 113 – Circuito da parte de alimentação do Arduino Mega 2560 117
Figura 114 – Circuito de seleção de alimentação do Arduino Mega 2560 117
Figura 115 – Circuito de comunicação USB do Arduino Mega 2560 118
Figura 116 – Diagrama de blocos do microcontrolador da plataforma Arduino Mega 2560 119
Figura 117 – Diagrama de blocos da CPU AVR do microcontrolador da plataforma Arduino Mega 2560 120
Figura 118 – Diagrama de blocos do *Timer* 2 do microcontrolador da plataforma Arduino Mega 2560 121
Figura 119 – Modos de operação do *Timer* 2 do microcontrolador da plataforma Arduino Mega 2560 122
Figura 120 – Prescaler do *Timer* 2 do microcontrolador da plataforma Arduino Mega 2560 122
Figura 121 – Diagrama de comunicação do SPI. 124
Figura 122 – Diagrama de blocos do SPI no microcontrolador da plataforma Arduino Mega 2560 124
Figura 123 – Modos de operação do SPI 125
Figura 124 – Possibilidades de *clock* na comunicação SPI 125
Figura 125 – Diagrama de conexão do I2C 126
Figura 126 – Exemplo de comunicação de escrita no I2C 128
Figura 127 – Exemplo de comunicação de leitura no I2C 129

Figura 128 – Diagrama de blocos do I2C no microcontrolador da plataforma Arduino Mega 2560 130
Figura 129 – Prescaler do I2C no ATmega2560 130
Figura 130 – Exemplo de *frame* detalhado de comando AT 131
Figura 131 – Exemplo de *frame* detalhado de requisição de transmissão 133
Figura 132 – Exemplo de *frame* detalhado de resposta de comando AT 134
Figura 133 – Exemplo de *frame* detalhado de status da transmissão 135
Figura 134 – Exemplo de *frame* detalhado de recepção de pacotes via radio frequência 136

Lista de tabelas

Tabela 1 – Significado dos aspectos do sinaleiro luminoso. 14
Tabela 2 – Valores do modelo de propagação *Shadowing* no ambiente *indoor*. . . . 96
Tabela 3 – Valores do modelo de propagação *Shadowing* para o ambiente *outdoor*. 97
Tabela 4 – Valores do modelo de propagação *Shadowing* para o ambiente ferroviário. 99

Lista de abreviaturas e siglas

AMV	Aparelho de Mudança de Via
API	*Application Programming Interface*
APL	*Application Layer*
AWGN	*Additive White Gaussian noise*
BER	*Bit Error Rate*
BPSK	*Binary Phase Shift Keying*
CAP	*Contention Acess Period*
CCA	*Clear Channel Assesment*
CCO	Centro de Controle e Operação
CFP	*Contention Free Period*
CPTM	Companhia Paulista de Trens Metropolitanos
CRC	*Cyclic Redundancy Check*
CSMA-CA	*Carrier Sense Multiple Access with Collision Avoidance*
DSSS	*Direct Sequence Spread Spectrum*
ED	*Energy Detection*
FFD	*Full Function Device*
GTS	*Guaranteed Time Slots*
IDE	*Integrated Development Environment*
IEEE	*Institute of Electrical and Electronic Engineers*
IHM	Interface Homem-Máquina
IoT	*Internet of Things*
ISM	*Industrial, Scientific and Medical*
LQI	*Link Quality Indicator*

LSB	*Least Significant Bit*
MAC	*Media Acess Control*
MPS	Método de Propagação de *Shadowing*
MSB	*Most Significant Bit*
NWK	*Network*
OQPSK	*Offset Quaternary Phase Shift Keying*
OSI	*Open System Interconnection*
PC	*Personal Computer*
PDA	*Personal Digital Assistant*
PDU	*Protocol Data Unit*
PER	*Packet Error Rate*
PHR	*PHY Header*
PHY	*Physical*
PN	*Pseudo Noise*
PPDU	*PHY Protocol Data Unit*
PSDU	*PHY Service Data Unit*
PWM	*Pulse Width Modulation*
QPSK	*Quaternary Phase Shift Keying*
RAM	*Random Acess Memory*
RF	Rádiofrequência
RFF	Rede Ferroviária Federal
ROM	*Read Only Memory*
RSSI	*Received Signal Strength Indicator*
SCADA	*Supervisory Control and Data Acquisition*
SCTC	Sistema de Controle de Tráfego Centralizado
SD	*Secure Digital*

SFD	*Start of Frame Delimiter*
SHR	*Synchronization Header*
SIR	*Signal to Interference Ratio*
SNR	*Signal to Noise Rate*
SPR	São Paulo *Railway*
STD	Sistema de Transmissão de Dados
UART	*Universal Asynchronous Receiver/Transmitter*
UC	Unidade Central
UR	Unidade Remota
USB	*Universal Serial Bus*
UWB	*Ultra Wideband*
WiFi	*Wireless Fidelity*
WLAN	*Wireless Local Area Network*
WMAN	*Wireless Metropolitan Area Network*
WPAN	*Wireless Personal Area Network*
WWAN	*Wireless Wide Area Network*

Sumário

1 INTRODUÇÃO 1
1.1 Surgimento do transporte ferroviário 1
1.2 Comunicação sem fio 4
1.3 Objetivos 6
1.4 Motivação 6
1.5 Estrutura do trabalho 6
1.6 Revisão bibliográfica 7

2 SINALIZAÇÃO FERROVIÁRIA 10
2.1 Sinaleiros 10
2.1.1 Funcionamento elétrico dos sinaleiros luminosos 12
2.2 Circuito de via 15
2.2.1 Funcionamento 15
2.3 Aparelho de mudança de via 17
2.3.1 Chaves 17
2.3.2 Associações de chaves 18
2.3.3 Máquinas de chave 19
2.4 SCTC 20

3 ZIGBEE 25
3.1 Camada física 26
3.1.1 Pacotes de dados da camada PHY 26
3.1.2 Frequência e canais 27
3.1.3 Modulação e espalhamento 28
3.2 Camada MAC 32
3.2.1 Modo *Beacon* e *non-Beacon* 32
3.2.2 Estrutura do superquadro 33
3.2.3 CSMA-CA 35
3.3 Camada de rede 36
3.3.1 Dispositivos 37
3.3.2 Funções lógicas dos dispositivos 37
3.3.3 Topologia de rede 38
3.4 Camada de aplicação 38

4 PLATAFORMA DE SISTEMAS EMBARCADOS 40
4.1 Arduino 41

4.1.1 Arduino Mega 2560 . . . 42
4.1.2 Plataforma de desenvolvimento . . . 43
4.2 XBee . . . 44
4.2.1 Modo de operação . . . 47
4.2.1.1 Operação API . . . 48
4.2.1.2 Operação API *escaped* . . . 48
4.2.1.3 Comprimento . . . 49
4.2.1.4 *Frame data* . . . 49
4.2.1.5 Soma verificadora . . . 51
4.2.1.6 Fluxo de dados UART no modo API . . . 51
4.2.1.6.1 Comando AT . . . 51
4.2.1.6.2 Transmissão e recepção de dados RF . . . 51
4.2.1.6.3 Comandos AT remotos . . . 52
4.2.1.6.4 Roteamento . . . 53
4.2.1.7 XCTU . . . 53

5 PROTÓTIPOS DO SISTEMA EMBARCADO DE COMUNICAÇÃO BASEADO NO PADRÃO ZIGBEE . . . 55
5.1 Seleção . . . 55
5.2 Diagrama de conexão . . . 58
5.3 Montagem . . . 58
5.4 Configuração . . . 60
5.4.1 RTC DS3231 . . . 62
5.4.2 Leitor de cartão micro SD . . . 63
5.4.3 XBee . . . 63

6 ANÁLISES DO SISTEMA EMBARCADO DE COMUNICAÇÃO BASEADO NO PADRÃO ZIGBEE . . . 64
6.1 Análise do sistema ZigBee por simulações . . . 64
6.2 Análise do sistema embarcado de comunicação baseado no padrão ZigBee através de medidas no ambiente *indoor* . . . 66
6.2.1 RSSI . . . 67
6.2.2 SNR . . . 68
6.2.3 PER . . . 71
6.3 Análise do sistema embarcado de comunicação baseado no padrão ZigBee através de medidas no ambiente *outdoor* . . . 73
6.3.1 RSSI e PER . . . 75
6.4 Análise do sistema embarcado de comunicação baseado no padrão ZigBee através de medidas no ambiente ferroviário . . . 79
6.4.1 RSSI . . . 79

6.4.2 PER utilizando a topologia *mesh* . . . 82
6.4.2.1 Coordenador . . . 87
6.4.2.2 Roteadores . . . 88
6.4.2.3 Resultado . . . 90
6.5 Teste de *sniffer* . . . **92**
6.5.1 ZENA *Wireless Networtk Analyzer* . . . 92
6.6 Análise do modelo de propagação de *Shadowing* . . . **95**
6.6.1 Criação do modelo de propagação *Shadowing* . . . 95
6.6.1.1 Ambiente *indoor* . . . 96
6.6.1.2 Ambiente *Outdoor* . . . 96
6.6.1.3 Ambiente Ferroviário . . . 99

7 CONSIDERAÇÕES FINAIS . . . **101**

REFERÊNCIAS . . . **102**

ANEXOS 105

ANEXO A – MODULAÇÃO QPSK E OQPSK . . . **106**
A.1 QPSK . . . **106**
A.2 OQPSK . . . **110**

ANEXO B – ESPALHAMENTO ESPECTRAL DE SEQUÊNCIA DIRETA . . . **112**

ANEXO C – ARDUINO MEGA 2560 . . . **115**
C.1 Alimentação . . . **115**
C.2 Comunicação USB . . . **115**
C.3 Microcontrolador . . . **117**
C.3.1 CPU . . . 118
C.3.2 *Timers/Counter* . . . 120
C.4 SPI . . . **123**
C.5 I2C . . . **126**

ANEXO D – TIPOS DE *FRAMES* NO XBEE . . . **131**
D.1 Comando AT . . . **131**
D.2 Requisição de Transmissão . . . **131**
D.3 Resposta de comando AT . . . **132**
D.4 Status de Transmissão . . . **132**
D.5 Recepção de pacotes . . . **132**

1 Introdução

Esse capítulo apresentará uma breve introdução sobre o surgimento do transporte ferroviário no Brasil, sua evolução e alguns conceitos sobre comunicações sem fio que irão contextualizar os objetivos propostos desse trabalho, além de apresentar sua estruturação e capítulos.

1.1 Surgimento do transporte ferroviário

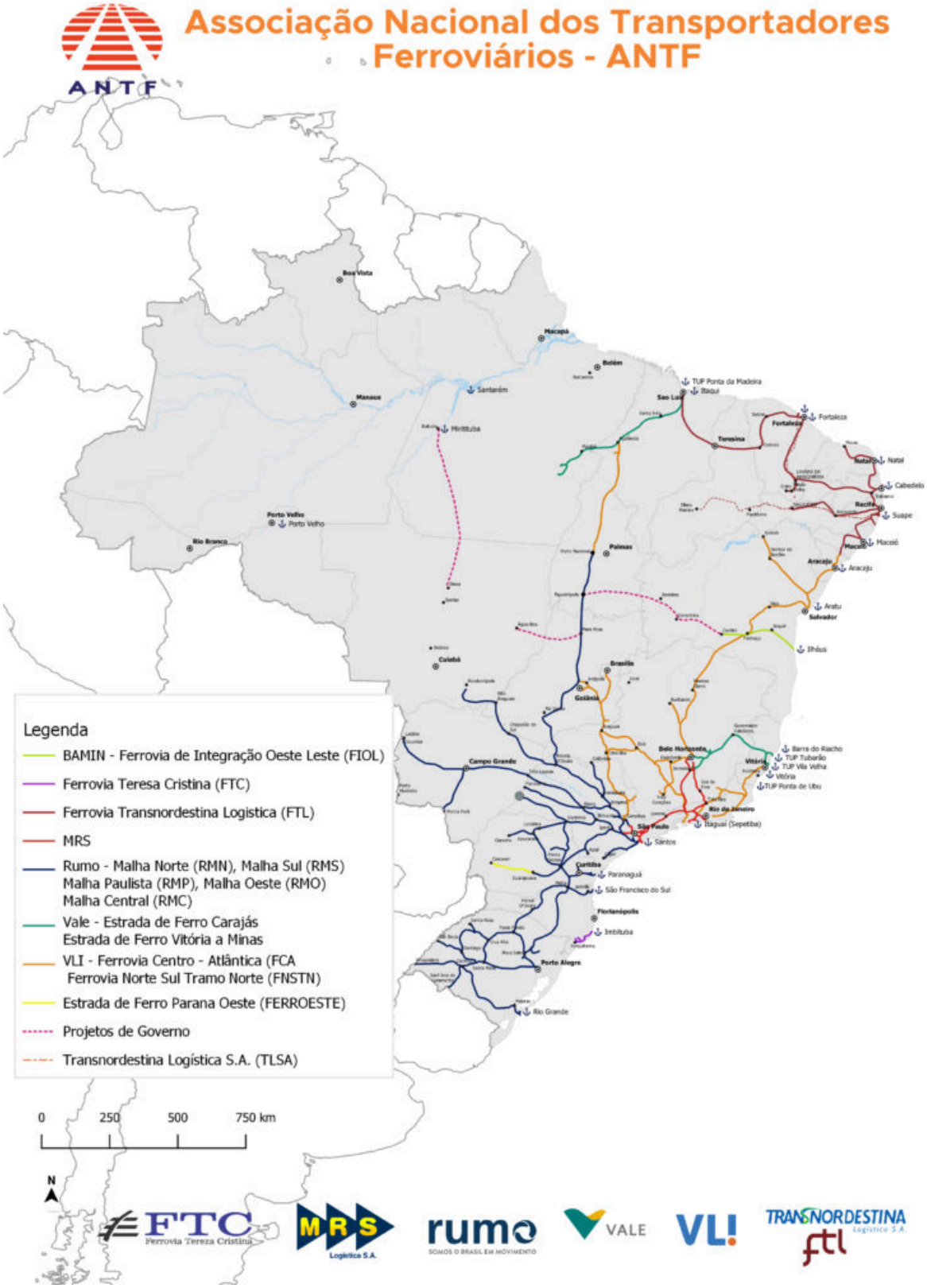
No início do século XVI, vários países europeus utilizavam o transporte sobre trilhos, e esses caminhos eram de uso exclusivo para o transporte de carvão e minérios de ferro extraídos de minas subterrâneas. Essas linhas eram compostas por trilhos de madeira e, sobre eles, circulavam carroças que se locomoviam puxadas por alguns cavalos (OLIVEIRA, 2013).

Já no século XVIII, algumas dessas empresas de mineração inglesas, começaram a revestir os trilhos de madeira com lâminas metálicas para aumentar a vida útil dos trilhos. Percebeu-se com isso que o deslocamento das carroças sobre os trilhos ficava mais fácil, o que gerava um aumento na produção, uma vez que o atrito entre as rodas e o trilho tinha diminuído, deixando as carroças mais fáceis de serem tracionadas. Os cavalos passaram a tracionar uma composição de carroças e não mais apenas uma. Segundo (FRANZÃO, 2018) com a Revolução industrial, vieram as máquinas a vapor e a primeira locomotiva, sendo criada em 1804 pelo inglês Richard Trevithick, proporcionando ainda mais volume e rapidez do transporte ferroviário. Em sua primeira viagem, a locomotiva criada por Richard conseguiu transportar 70 pessoas e 9 toneladas de carvão em cinco vagões, circulando a uma velocidade de 8 km/h por 15 km de trilhos.

Nos dias atuais, um dos principais meios de transporte, seja de passageiros ou de carga é o sistema ferroviário, principalmente no estado de São Paulo. A primeira ferrovia no estado de São Paulo foi construída devido à necessidade de se transportar o café produzido aqui no Brasil. Isso ocorreu em 1867, quando entrava em operação a SPR (São Paulo *Railway*) com uma ferrovia que ia desde o litoral até a cidade de Jundiaí. Porém ainda nessa época se expandiam as lavouras de café, sendo assim, foram surgindo pequenas ferrovias que completavam essa rede, onde se realizava o transporte de café até o porto de Santos (MIGUEL et al., 2018). Além disso, foram construídas várias ferrovias ao passar dos anos também interligando o país. Na Figura 1 pode-se visualizar o mapa ferroviário no Brasil em 2022.

O transporte ferroviário no Brasil ainda possui uma pequena participação no

Figura 1 – Mapa das principais ferrovias do Brasil.



Fonte: (ANTF, 2022).

volume total de cargas transportadas, devido principalmente à pequena extensão de sua malha, que é de apenas 29.320 km. Verifica-se que sua maior utilização é no transporte de cargas de baixo valor agregado, como por exemplo, os produtos agrícolas e minérios, pois correspondem a um melhor custo benefício, considerando-se o peso e o volume dos bens transportados (BURI et al., 2006).

Atualmente, depois da divisão e privatização das malhas ferroviárias nacionais, as empresas concessionárias que assumiram a administração destas vias, estão investindo em modernização e extensão para que cada vez mais se desenvolva um meio de transporte mais seguro e eficiente.

Além do transporte de cargas, ainda existe outro transporte ferroviário, o de passageiros. A extinta RFF (Rede Ferroviária Federal), possuía na maioria da sua malha, até o início dos anos 70, atendimento ao transporte de passageiro concomitante ao transporte de carga. Nos mesmos anos, com a modernização do país e a nítida preferência pelo transporte rodoviário, iniciou-se um declínio do transporte ferroviário de passageiros (FERREIRA; CANTARINO, 2006).

O transporte de passageiro acabou ficando em segundo plano nos investimentos, uma vez que o transporte de cargas eram mais vantajosos, porém com o aumento do número, peso e velocidade dos trens, foi acompanhado por um número cada vez maior de acidentes.

Para tentar garantir a segurança na circulação de trens e locomotivas, inicialmente, um homem a cavalo sinalizava para o maquinista com uma bandeira vermelha, indicando as condições à frente. Alguns anos mais tarde utilizavam-se sinaleiros fixos instalados ao longo da via, o posicionamento de uma bandeira vermelha no topo de um mastro indicava se a linha estava livre ou com problemas (MIGUEL et al., 2018).

Conforme o volume de tráfego aumentou, tornou-se necessário aumentar o número de cruzamentos obrigatórios entre trens de sentidos opostos. Esse aumento tornou a operação cada vez mais complexa e houve necessidade de maior controle sobre o tráfego. Com isso surgiu o SCTC (Sistema de Controle de Tráfego Centralizado), que é um sistema de supervisão que concentra o monitoramento e controle dos equipamentos em um único lugar. Para esta operação, é necessário ter um dispositivo central que irá controlar e monitorar todos os equipamentos remotamente (NUNES, 2012).

Esses sistemas vêm evoluindo cada vez mais, porém a base de monitoramento dos equipamentos nas vias, como sinaleiros e AMVs (Aparelho de Mudança de Via), continuam usando cabos elétricos de cobre para interligar o equipamento até o dispositivo centralizador, que às vezes pode estar até 1 km de distância. Com isso, há necessidade de lançamento de muitos cabos elétricos com vários condutores de cobre, que podem acabar sofrendo furtos. Esses furtos, paralisam e prejudicam o funcionamento do sistema, gerando

atrasos e necessidade de cada maquinista verificar a situação de cada equipamento. Além disso, gera prejuízo, pois têm-se a necessidade de realizar o lançamento de novos cabos elétricos no local. Esse problema abre um nicho de estudo para desenvolvimento do uso de sistemas de supervisão embarcado que utilizem comunicação sem fio para o monitoramento dos equipamentos de sinalização nas vias férreas.

1.2 Comunicação sem fio

A comunicação sempre é importante, seja em um sistema, seja entre pessoas ou entre pessoas e sistemas. Sem ela, seria muito difícil de realizar qualquer tarefa mesmo que simples. O uso de comunicações sem fio hoje é fundamental para grande parte dos sistemas e até mesmo para nós seres humanos, afinal o nosso som utiliza o ar para se propagar e chegar até o ouvido de outra pessoa, sendo assim, cada vez mais se faz necessário o estudo e desenvolvimento de comunicações sem fio.

Segundo (RAPPAPORT, 2009), a capacidade de nos comunicarmos com pessoas em movimentos evoluiu bastante desde que Guglielmo Marconi demonstrou a capacidade do rádio de fornecer contato contínuo com os navios navegando pelo canal inglês. Isso foi em 1897 e, desde então, novos métodos têm sido entusiasticamente adotados em todo o mundo. O setor de comunicação sem fio cresceu em uma ordem de grandeza, alimentado por melhorias na fabricação de circuitos digitais e de radiofrequência, bem como pela nova integração de circuitos de grande escala e outras tecnologias de miniaturização que tornaram os equipamentos portáteis de rádio ainda menores, mais baratos e mais confiáveis. As técnicas de comunicação digital facilitaram muito a implantação em larga escala de redes de comunicação de rádio, com facilidade de uso e preços bem acessíveis.

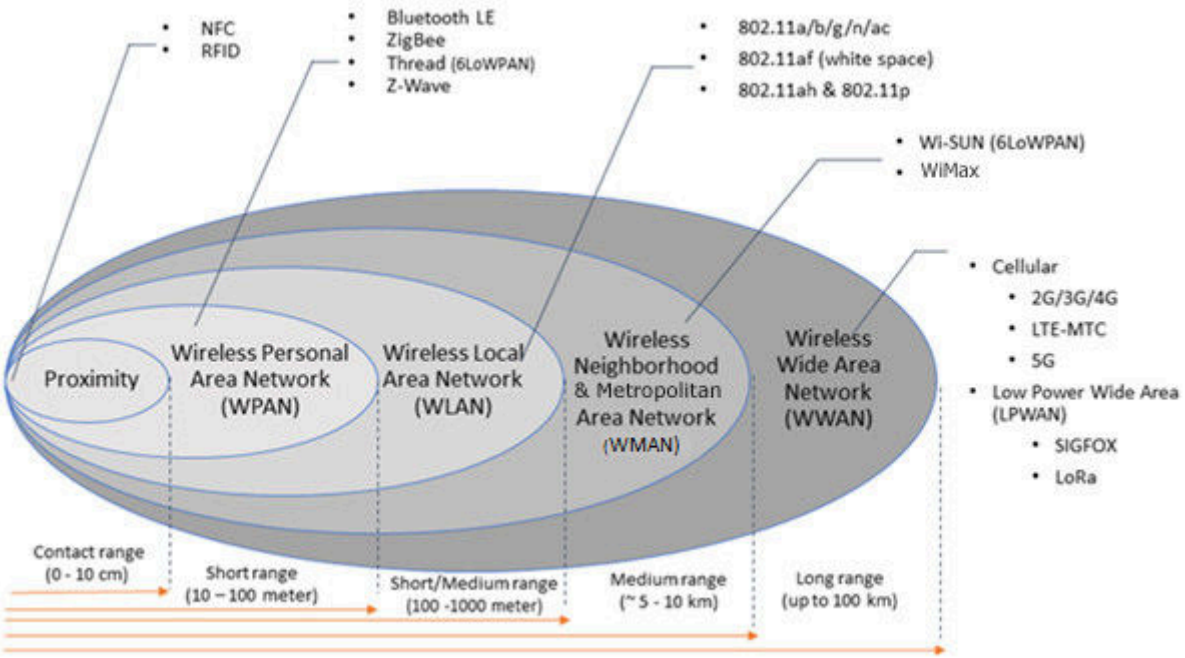
As redes sem fio podem ser classificadas em quatro grandes grupos em virtude do seu raio de alcance e aplicação:

- **WPAN** (*Wireless Personal Area Network*) - Reúne as tecnologias de rede sem fio de pequeno alcance (entre 10 e 100 m). Este escopo de rede gira em torno do indivíduo, mas efetua comunicação entre dispositivos móveis;
- **WLAN** (*Wireless Local Area Network*) – Reúne as tecnologias de rede sem fio destinadas à interligação de redes locais com alcance entre 100 e 300 m. Trata-se de padrão implementado como extensão ou alternativa para as redes com cabeamento convencional (par metálico ou fibra óptica);
- **WMAN** (*Wireless Metropolitan Area Network*) - Reúne as tecnologias que tratam dos acessos de banda larga para redes em áreas metropolitanas, com alcance em torno de 6 km;

- **WWAN** (*Wireless Wide Area Network*) – Neste grupo estão as tecnologias voltadas para redes de longa distância em telecomunicações, atendendo aos serviços de voz e alguns serviços de dados.

A Figura 2 ilustra a classificação dos grupos acima e exemplifica algumas das tecnologias associadas.

Figura 2 – Classificação e exemplos das tecnologias de rede sem fio.



Fonte: Adaptado de (GUERRERO, 2018).

Como apresentado na Figura 2, existem várias tecnologias de redes de comunicação sem fio que podem ser empregadas para a utilização no sistema proposto, porém para o sistema de sinalização ferroviário e a distância entre seus equipamentos, será utilizado nesse trabalho o padrão ZigBee. Esse padrão é utilizado em redes WPAN e em algumas WLAN, além de possuir uma topologia de rede *mesh*, o que permite que haja mais de uma rota para um mesmo destinatário e essa arquitetura não está presente em redes como por exemplo LoRa, SIGFOX e LTE. A topologia *mesh* é muito interessante pois permite aos dispositivos a utilização de outras rotas caso haja algum problema na rota principal, como por exemplo cruzamentos de trens ou que o destinatário não esteja dentro do campo de alcance de transmissão do remetente mas possua dispositivos próximos que criam uma rota até o destinatário. Com o estudo do padrão ZigBee, será realizado a prototipagem de um sistema supervisório para monitoramento e controle dos equipamentos de sinalização.

1.3 Objetivos

O objetivo principal deste trabalho é realizar o desenvolvimento de um sistema embarcado de comunicação baseado no padrão ZigBee para monitoramento e controle dos equipamentos de sinalização ferroviários e a realização de testes para validação do mesmo.

O objetivo secundário é realizar um estudo e análise sobre o padrão ZigBee, realizando as seguintes atividades:

- Realizar simulações computacionais do padrão ZigBee para entender seu comportamento e seus principais parâmetros;
- Encontrar um modelo de propagação representativo do ambiente estudado para o padrão ZigBee;
- Análise e estudo de plataformas embarcadas para definir o conjunto de dispositivos que serão utilizados para a montagem do sistema embarcado proposto
- Realizar testes práticos com o sistema desenvolvido utilizando a topologia *mesh* para análise de seu desempenho.

1.4 Motivação

De Janeiro a Agosto de 2018, Metrô-SP e CPTM (Companhia Paulista de Trens Metropolitanos) juntas tiveram mais de 15 km de cabos furtados, que geraram um prejuízo da ordem de R$ 900.000,00, dinheiro este que poderia ser investido em melhoria do próprio sistema ferroviário (BAZANI; MARQUES, 2018). Com o uso de um sistema de monitoramento que utilize comunicação sem fio, diminui-se consideravelmente a quantidade de cabos elétricos de cobre que devem ser lançados do concentrador até os equipamentos nas vias. Isso irá diminuir significativamente a quantidade de furto de cabos elétricos e assim aumentando a disponibilidade do transporte ferroviário.

1.5 Estrutura do trabalho

O presente trabalho foi estruturado em 7 capítulos, incluindo este introdutório. No segundo capítulo são apresentados os principais sistemas de uma ferrovia, o funcionamento do sistema de sinalização e será apresentado de forma breve os principais equipamentos e sistemas utilizados no SCTC.

No terceiro capítulo, é apresentado o padrão ZigBee, abordando suas camadas, topologias, dispositivos, arquitetura e seu funcionamento. No quarto capítulo, é desenvolvido um estudo e análise sobre plataformas embarcadas e em especial o funcionamento do sistema embarcado Arduino Mega 2560 e a plataforma XBee.

Já o quinto capítulo apresenta os requisitos necessários, seleção de periféricos, montagem e construção dos protótipos que irão formar o sistema de comunicação proposto. O sexto capítulo apresenta a simulação computacional e análise do padrão Zigbee, além de apresentar os resultados e análises dos testes realizados com os protótipos nos diversos tipos de ambientes para validação do sistema de comunicação proposto. O último capítulo apresenta as considerações finais sobre o trabalho.

1.6 Revisão bibliográfica

Em (NUNES; CAPPELLI; UMEZU, 2011), os autores realizaram um estudo para caracterização do ambiente de uma granja e as influências sobre a propagação de sinais de rede sem fio do padrão ZigBee. O objetivo dessa caracterização é a de utilização de sensores espalhados dentro da granja para realizar o monitoramento de temperatura, umidade relativa entre outras características que afetam diretamente a produção da granja. A metodologia adotada foi a utilização de dois módulos que comunicam entre si utilizando o padrão ZigBee para criar um tráfego de rede e foi-se alterando a distância entre esses dispositivos de forma a realizar as medidas de potência de transmissão, potência de ruído e intensidade de recepção RSSI (*Received Signal Strength Indicator*) para cada distância utilizada. Além dessas medidas, os autores desenvolveram um *software* para computador na linguagem Delphi para que ele enviasse várias vezes ao módulo principal (TX) uma "*String*"(através de uma conexão via cabo serial) e o módulo TX por sua vez enviaria para o segundo módulo (RX). O módulo RX fecharia o "*Loop*"enviando de volta a *String* recebida para o módulo TX, que por sua vez enviaria ao computador para analisar se a *String* recebida é a mesma que foi enviada.

Com essas medidas, foi possível estimar a propagação dos sinais de rede sem fio do padrão ZigBee dentro do ambiente da granja utilizando um modelo de propagação que leva em consideração os efeitos de sombreamento no ambiente analisado. Também foi possível verificar quantos pacotes foram perdidos ou apresentaram erros para cada distância. Isso ajuda a estimar uma estatística de perda de pacotes transmitidos. O conjunto dessas informações são de grande importância para demonstrar que é aplicável a implementação de sensores que utilizarão a rede sem fio do padrão ZigBee para comunicação para o ambiente analisado.

Já em (COSTA; MENDES, 2006), os autores realizaram um estudo comparativo entre as tecnologias de redes sem fio *Bluetooth* e ZigBee. Esse estudo mostra que o padrão ZigBee embora tenha uma taxa máxima de transmissão inferior ao *Bluetooth*, apresenta muitas outras vantagens. Uma das principais vantagens é o alcance, onde o padrão ZigBee pode chegar a mais de 100 metros de distância. Ele apresenta um consumo de corrente aproximadamente 25% menor, o que é ótimo para dispositivos IoT (*Internet of Things*)

que fará com que a bateria dure mais nesses dispositivos. Outra vantagem interessante é no quesito segurança, pois o ZigBee demonstra uma maior confiabilidade ao usuário, devido ao fato de seu protocolo tratar mais a segurança do que o protocolo do *Bluetooth*, uma vez que no *Bluetooth* ele deixa uma boa parte desse controle e segurança para o desenvolvedor da aplicação. No parâmetro quantidade de usuários e tempo de acesso a rede, mais uma vez o ZigBee foi vitorioso, tendo uma velocidade de acesso 100 vezes mais rápida e podendo ter até 65535 usuários na rede.

Em (VICENTINI, 2018), o autor desenvolveu um estudo sobre a interferência do WiFi (*Wireless Fidelity*) no padrão ZigBee, uma vez que existe sobreposições de canais nos padrões. Como o WiFi está cada vez mais presente em todos os ambientes, é muito importante o estudo e análise sobre a influência dele no ZigBee. No trabalho o autor realiza um teste para medir a interferência do WiFi. A metodologia aplicado foi a montagem de um conjunto composto por dois computadores e um roteador WiFi para se comunicarem constantemente e próximo a esse conjunto, um outro conjunto composto por dois dispositivos que trafegam o padrão ZigBee ligados a um terceiro computador. Esse terceiro computador tem a função de monitorar o tráfego dos dispositivos ZigBee e medir a potência do WiFi. Foram realizados esses testes para as distâncias de 1 a 8 m.

Após obter a medida de potência do WiFi no ZigBee, foi utilizado uma modelagem para descrever a potência do WiFi em função da distância e com esses dados o autor determinou a taxa de erro de bit (BER - *Bit Error Rate*) e a taxa de erro de pacotes (PER - *Packet Error Rate*) para alguns canais do ZigBee que apresentam diferença de frequência central do canal para o canal do WiFi na ordem de 2 a 7 MHz. Esses dados dão uma estimativa do quanto o WiFi atrapalha a comunicação entre os dispositivos ZigBee. Pode-se notar que quanto menor a diferença entre as frequências centrais, maior será a interferência do WiFi.

No artigo (ROMEIRO; COSTA, 2016), os autores mostraram um estudo e testes realizados com o padrão ZigBee usando o *software* de computador ZABBIX para realizar o monitoramento e controle de dispositivos em uma residência para sua automação. Sua metodologia foi a aplicação de duas plataformas embarcadas (Arduino no caso), onde um Arduino recebia os comandos do computador via *Ethernet* e transmitia ao segundo arduino via ZigBee. Esse por sua vez recebia e processava os comandos solicitados e enviava as indicações dos dispositivos controlados.

Esse estudou mostrou que o padrão *ZigBee* dentro de um ambiente residencial tem um comportamento satisfatório, mostrando que é possível realizar monitoramento e controle de vários dispositivos com o uso de um sistema embarcado, que é de baixo custo junto a outra plataforma que utiliza o padrão *ZigBee* que também é de baixo custo, podendo assim controlar vários equipamentos dentro de uma residência.

Já em (ROCHA et al., 2014), pode-se ver a aplicação do padrão *ZigBee* em um

campo agrícola, para monitoramento e controle de um sistema de irrigação e luminosidade. Os autores criaram um *software* na linguagem Java para computador, onde é possível escolher dias e períodos que irão ocorrer à irrigação do solo. Esse *software* envia através da internet as informações para um servidor instalado na propriedade do campo de irrigação. Esse por sua vez irá enviar para a plataforma de sistema embarcado através de dois dispositivos que operam no padrão ZigBee. A plataforma de sistema embarcado que controla a irrigação, realiza a medição de umidade relativa do solo, luminosidade e temperatura e envia pelo servidor para o software desenvolvido pelos autores.

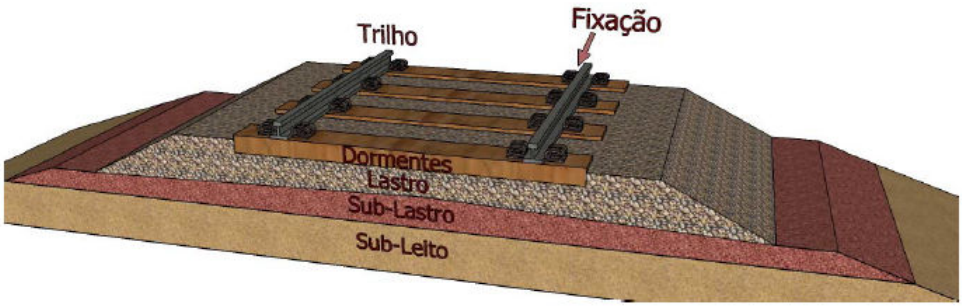
Com o protótipo criado pelos autores, foi possível mostrar que ao utilizar uma plataforma de sistemas embarcado, interligado a um servidor, é possível ter um resultado muito satisfatório nas transmissões, mesmo em um cenário bem diferente do estudado em (ROMEIRO; COSTA, 2016). E em ambos os casos, o ZigBee se mostrou robusto na transmissão dos dados, utilizando o ar como meio de transmissão evitando assim a utilização de condutores de cobre para interligar os dispositivos.

2 Sinalização ferroviária

Os principais sistemas de uma ferrovia são: Via permanente, Material rodante e Sinalização. Nesse capítulo será apresentada uma breve explicação sobre os conceitos de via permanente e material rodante. Após essa breve explicação, o restante do capítulo será focado no sistema de sinalização e seu funcionamento para entender como um sistema embarcado com comunicação sem fio pode substituir o sistema atual que depende em sua totalidade de comunicação que utiliza condutores elétricos.

Via permanente é a denominação utilizada para o conjunto de componentes e camadas que possibilitam a passagem de veículos ferroviários. São divididos em dois grupos: infraestrutura e superestrutura. A infraestrutura refere-se à camada inferior de terraplenagem que é chamada de sub-leito. Já a superestrutura é composta pelo sub-lastro, lastro, dormente, trilho e fixação. A Figura 3 apresenta os componentes da via permanente.

Figura 3 – Componentes da via permanente.



Fonte: (KLINCEVICIUS, 2011).

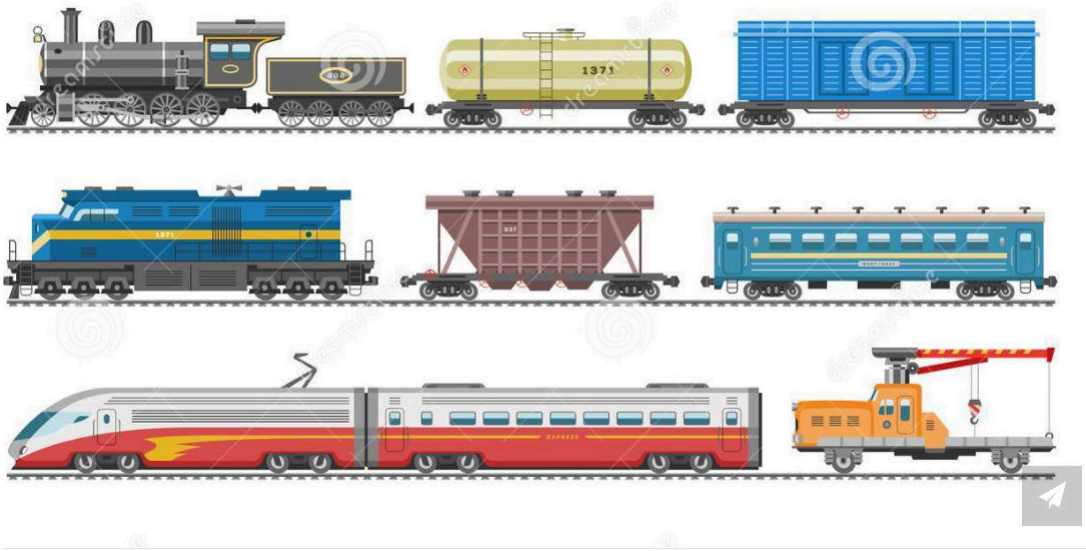
Material rodante é todo veículo que trafega através dos trilhos ferroviários, ou seja, veículos de manutenção, trens, locomotivas, vagões, veículos rodoferroviários (circulam tanto em rodovias quanto em ferrovias), entre outros. A Figura 4 apresenta alguns exemplos de material rodante.

A sinalização ferroviária tem o objetivo de garantir o máximo de trens circulando dentro de uma mesma estrutura, ao mesmo tempo garantindo que dois veículos não cheguem ao mesmo lugar ao mesmo tempo, gerando colisões entre os mesmos (NUNES, 2012). A sinalização possui subsistemas e será apresentado a seguir o funcionamento de alguns dos principais subsistemas da sinalização.

2.1 Sinaleiros

O Sinaleiro é uma forma de transmitir, por seus aspecto e significado, informações aos maquinistas que estão circulando com as composições (material rodante), com a

Figura 4 – Exemplos de material rodante.

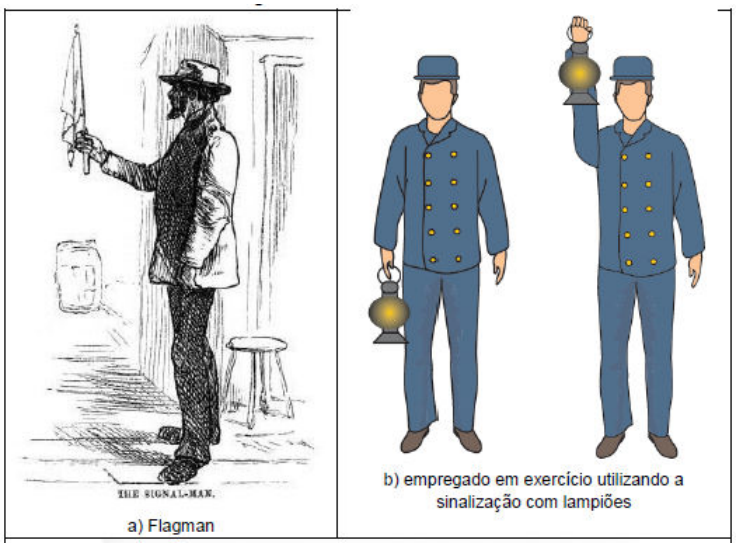


Fonte: (ILUSTRAÇÃO. . . , 2018).

finalidade de autorizar ou não a circulação dos trens.

Nos primórdios da ferrovia, era utilizado o sinaleiro manual que era realizado através de uma pessoa que fazia a indicação para as composições através de uma bandeira ou algum lampião. A Figura 5 exemplifica isso. Com o passar dos anos foram surgindo várias formas de sinaleiros conforme avançavam as tecnologias. Hoje o sinaleiro mais utilizado em ferrovia é o sinaleiro luminoso.

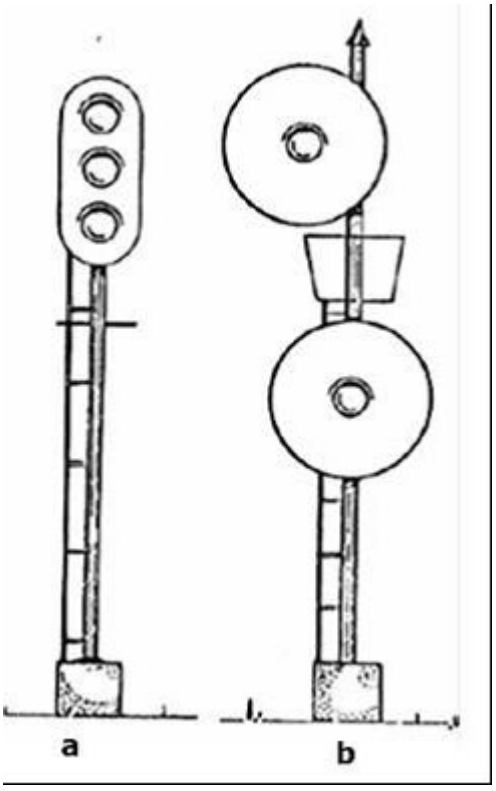
Figura 5 – Sinaleiro manual.



Fonte: (MIGUEL et al., 2018).

O sinaleiro luminoso é aquele cujo aspecto é fornecido pela cor de um ou mais focos luminosos. Este sinaleiro pode ser unifocal, quando os diferentes aspectos são mostrados no mesmo foco ou plurifocal, onde cada foco apresenta um aspecto (MIGUEL et al., 2018).A Figura 6 apresenta exemplos de sinaleiros luminosos.

Figura 6 – (a) Sinaleiro plurifocal. (b) Sinaleiro unifocal.



Fonte: (MIGUEL et al., 2018).

2.1.1 Funcionamento elétrico dos sinaleiros luminosos

Um dos principais sinaleiros luminoso utilizado é o sinaleiro do tipo anão (mais conhecido como *Searchlight*). Ele é um sinaleiro unifocal onde de acordo com a cor do seu foco, será interpretado pelos maquinistas se podem prosseguir ou não e a indicação sobre o trecho a frente se está livre ou ocupado. A Figura 7 apresenta uma foto dele visto pela frente e por trás. O sinaleiro anão possui uma estrutura de ferro fundido, essa estrutura possui uma lente, uma viseira e base para fixação. Dentro dessa estrutura, é instalado um mecanismo que controla o sinaleiro.

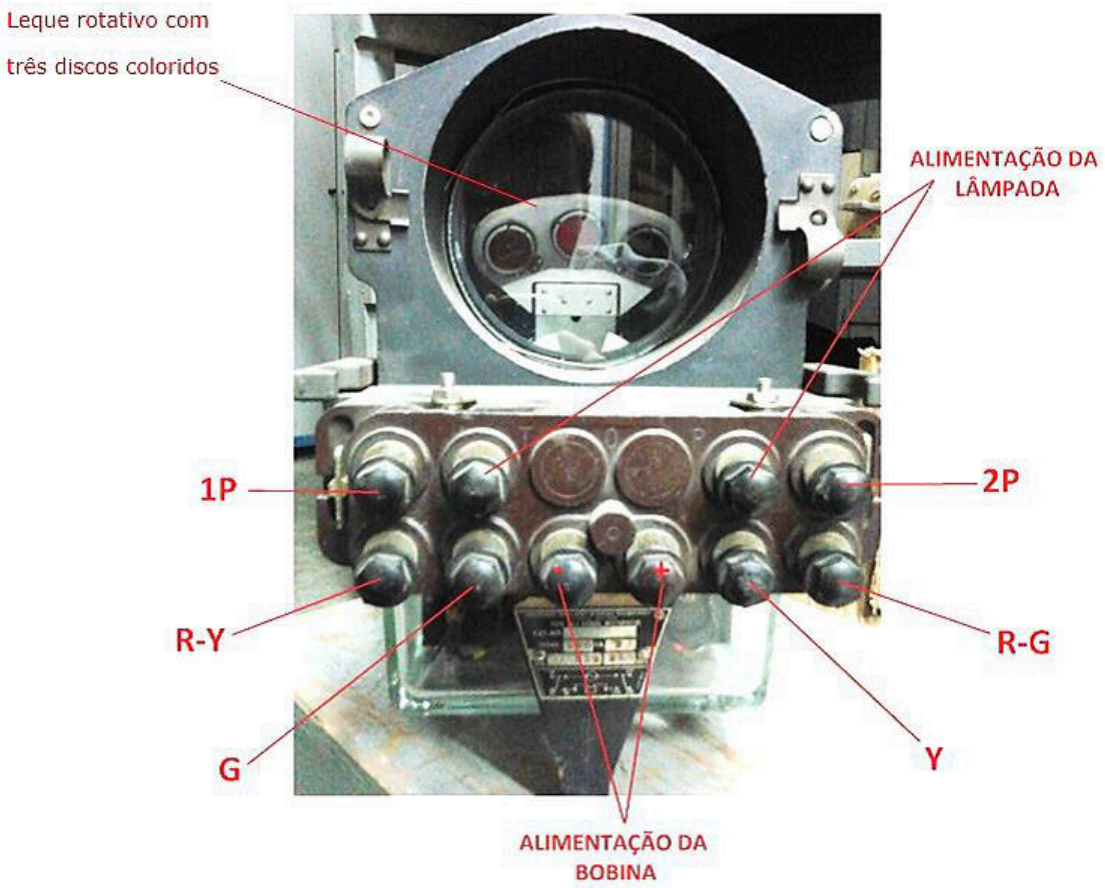
Esse sinaleiro é conhecido como *Searchlight* pois dentro da estrutura de ferro fundido, existe um mecanismo chamado *Searchlight*, que é o mecanismo que altera a cor do foco do sinal e controla a indicação de qual cor está sendo mostrada no sinaleiro. O mecanismo *Searchlight* é apresentado na Figura 8. No mecanismo, aloja-se um relê de duas posições, o qual movimenta um leque com três discos coloridos e controla dois contatos internos que indicam a posição do leque do sinaleiro (MIGUEL et al., 2018).

O sinaleiro tem a cor do seu aspecto modificada através do *Searchlight* que move um leque rotativo com três discos coloridos que projeta uma das indicações de aspecto vermelho, amarelo ou verde por meio de um sistema de lente e lâmpada, dando um foco luminoso constante. A polaridade aplicada no rele do mecanismo *Searchlight* que virá do circuito de controle determina qual das cores será posicionada no foco luminoso,

Figura 7 – Sinaleiro anão (*Searchlight*).

Fonte: (MIGUEL et al., 2018).

Figura 8 – Mecanismo da unidade de sinal *Searchlight*.



Fonte: (MIGUEL et al., 2018).

podendo ser amarelo ou verde. Quando o rele estiver desenergizado, o disco vermelho será posicionado no foco luminoso através da gravidade, assim no caso de falta de energia, o aspecto vermelho será sempre mostrado, impedindo a circulação, gerando uma condição de falha segura no sistema. A tabela 1 exemplifica a interpretação do foco do sinaleiro luminoso.

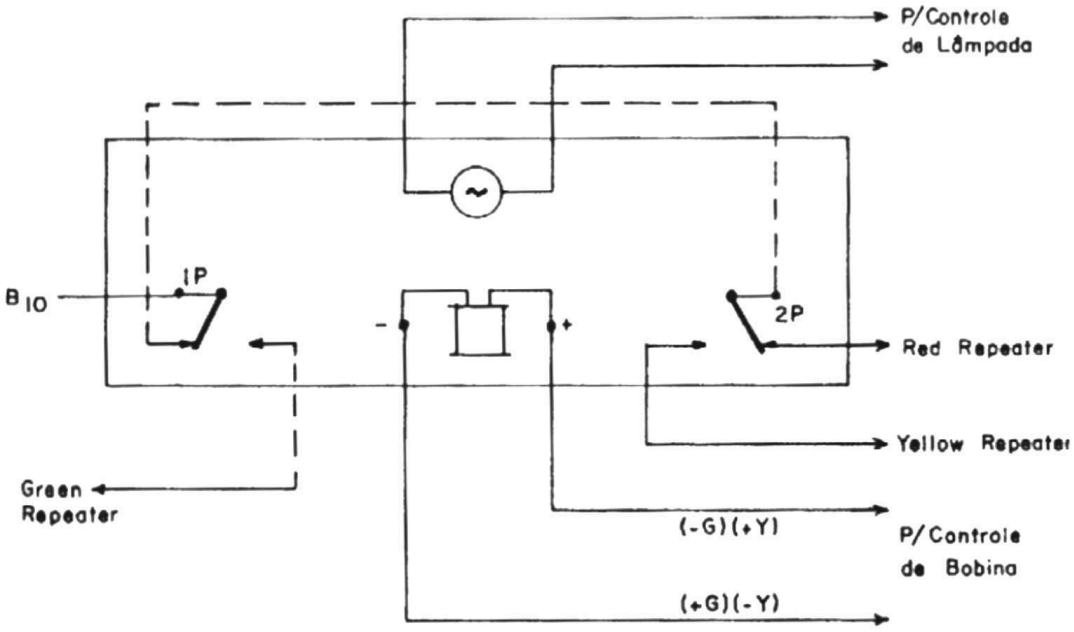
Tabela 1 – Significado dos aspectos do sinaleiro luminoso.

Aspecto	Indicação
Verde	Prossiga em velocidade máxima autorizada
Amarelo	Prossiga em velocidade Limitada
Vermelho	Pare
Vermelho piscante	Prossiga em velocidade restrita, preparado para parar.

Fonte: (BUENO, 2006).

A Figura 9 ilustra de forma simplificada o esquema elétrico do mecanismo *Searchlight*.

Figura 9 – Esquema elétrico simplificado do mecanismo *Searchlight*.



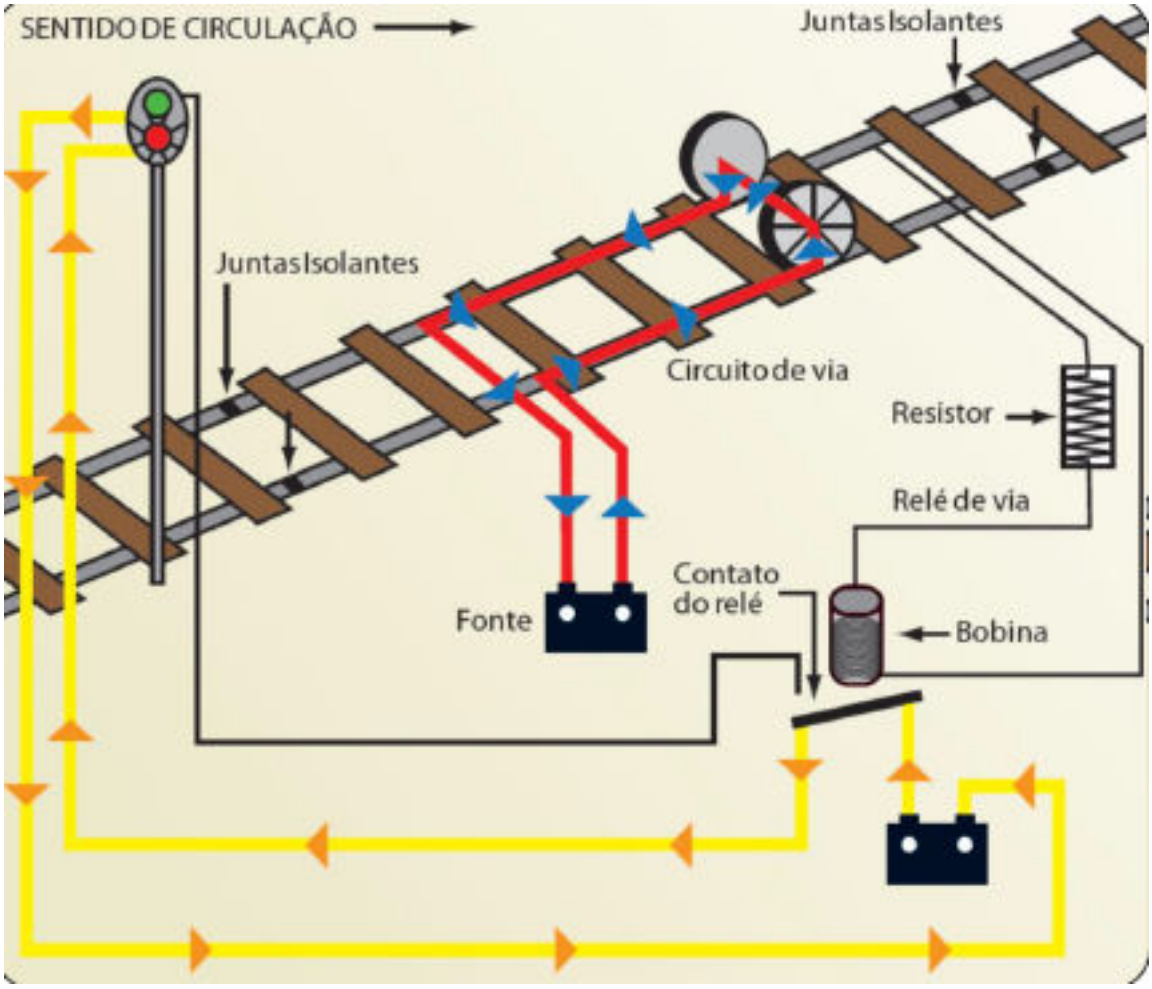
Fonte: (MIGUEL et al., 2018).

Pela figura 9, pode-se perceber que para o controle completo do sinal, são necessários 8 condutores elétricos. Como é importante possuir condutores de reserva para o caso de falha de rompimento do cabo ou baixa isolação, o ideal é ter pelo menos 10 condutores elétricos que cheguem nesse sinaleiro. Caso o concentrador (local onde se concentra o controle e indicações dos equipamentos em campo) esteja a 100 m de distância desse sinal, já será necessário aproximadamente 1 km de cabo de condutores elétricos para controle de apenas um único sinal, além de quanto maior for a distância, provavelmente maior deverá ser a seção desses condutores devido queda de tensão nos mesmos.

2.2 Circuito de via

Circuito de via é o elemento básico de qualquer sistema de sinalização. É a forma pela qual os trens serão detectados, sendo usados para construir a lógica de controle ao longo da linha de um sistema de sinalização. Os trilhos, neste contexto são divididos em seções por meio de juntas isoladas (talas isolantes). Um dos lados desta seção está conectado a uma fonte de alimentação em série com um resistor e o outro está ligado ao relê de via. A Figura 10 ilustra o circuito de via e especificamente representa uma ocupação realizada por um trem.

Figura 10 – Exemplo de circuito de via com ocupação.

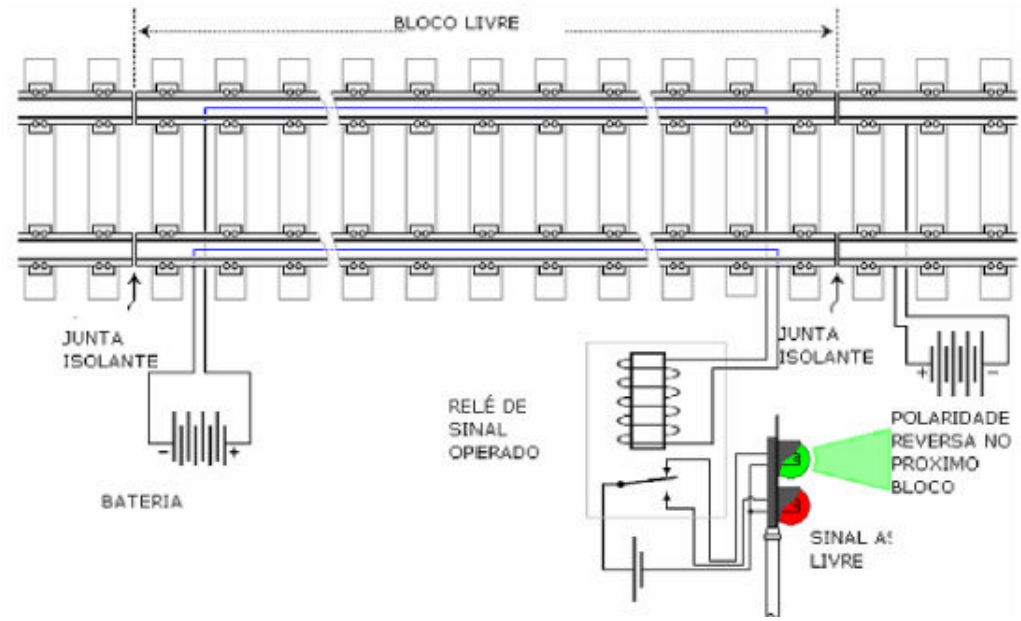


Fonte: (MIGUEL et al., 2018).

2.2.1 Funcionamento

Quando o circuito está desocupado (sem trem no circuito), a corrente da fonte de alimentação circula através do resistor limitador para os trilhos, percorrendo o caminho indicado na Figura 11 e consequentemente alimentando o rele de via fazendo com que seus contatos fiquem fechados (MIGUEL et al., 2018).

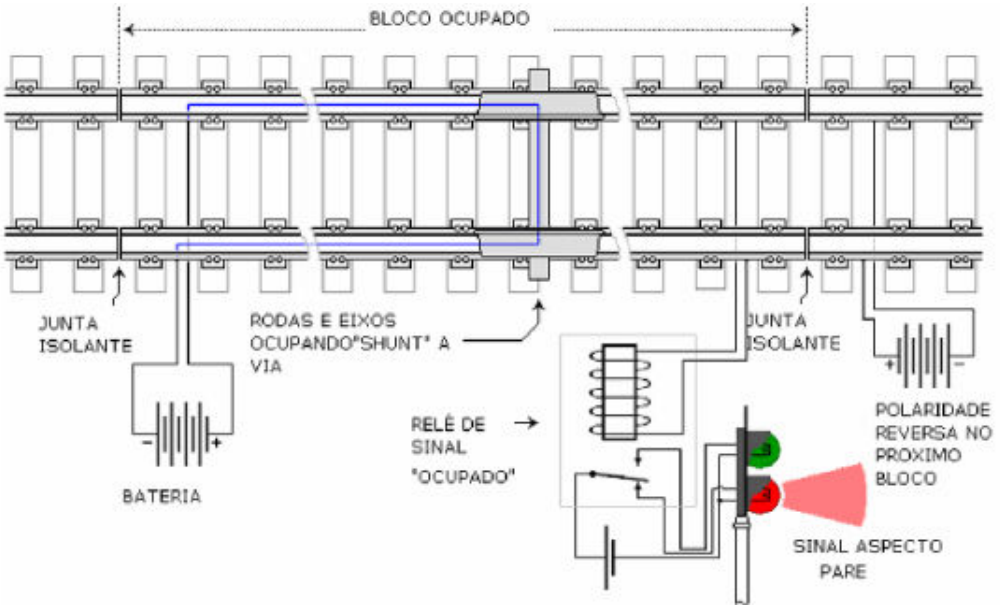
Figura 11 – Condição do circuito de via livre.



Fonte: (MIGUEL et al., 2018).

Na presença de um veículo ferroviário, em qualquer parte da seção de via, a corrente do relê deixa de passar através da bobina do relê, para então passar pelos rodeiros e consequentemente para o eixo do trem. A bobina agora não recebe energia suficiente, permitindo assim que o relê passe para o estado de "queda", causando abertura de seus contatos. A Figura 12 ilustra essa condição.

Figura 12 – Condição do circuito de via ocupado.



Fonte: (MIGUEL et al., 2018).

Fazendo uma análise, qualquer desconexão nesse circuito fechado como fonte, resistor regulador, bobina do rele ou a descontinuidade do caminho natural da corrente pelos condutores e ou suas conexões aos trilhos, por qualquer razão (por exemplo trilho

partido), provocará a desenergização do relê. Essa situação é uma simulação de presença de um veículo ferroviário, sendo a condição mais restritiva do circuito de via, podendo-se dizer que passou a estar na condição de falha segura.

Conforme o esquema de ligação apresentado na Figura 10, pode-se perceber que para a ligação de um circuito de via é necessário pelo menos 4 condutores (sendo ligado um par em cada extremidade do circuito). Para controle de apenas um único circuito de via que esteja a uma distância de aproximadamente 50 m do concentrador, será necessário pelo menos 200 m de cabo de condutores elétricos.

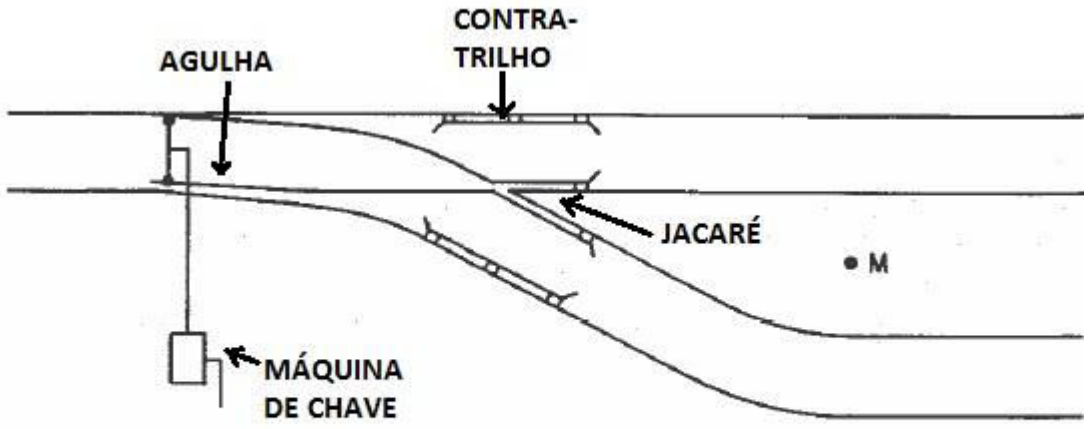
2.3 Aparelho de mudança de via

Os AMVs são elementos que possuem partes móveis. Podem ser utilizadas para transferir o veículo ferroviário de uma via para outra, fazer o cruzamento de vias e proteger que veículos façam movimentos não autorizados através do descarrilamento dos mesmos.

2.3.1 Chaves

Uma chave é um AMV que faz interface somente entre duas vias. A figura 13 representa a estrutura de chave e suas principais partes.

Figura 13 – Esquema de chave.



Fonte: (NUNES, 2012).

A máquina de chave é a parte responsável pelo dispositivo de movimentação da chave. Nela estão presentes os tirantes de operação, que irão causar a movimentação das agulhas e, quando existentes, os tirantes de indicação, que indicam a correspondência da

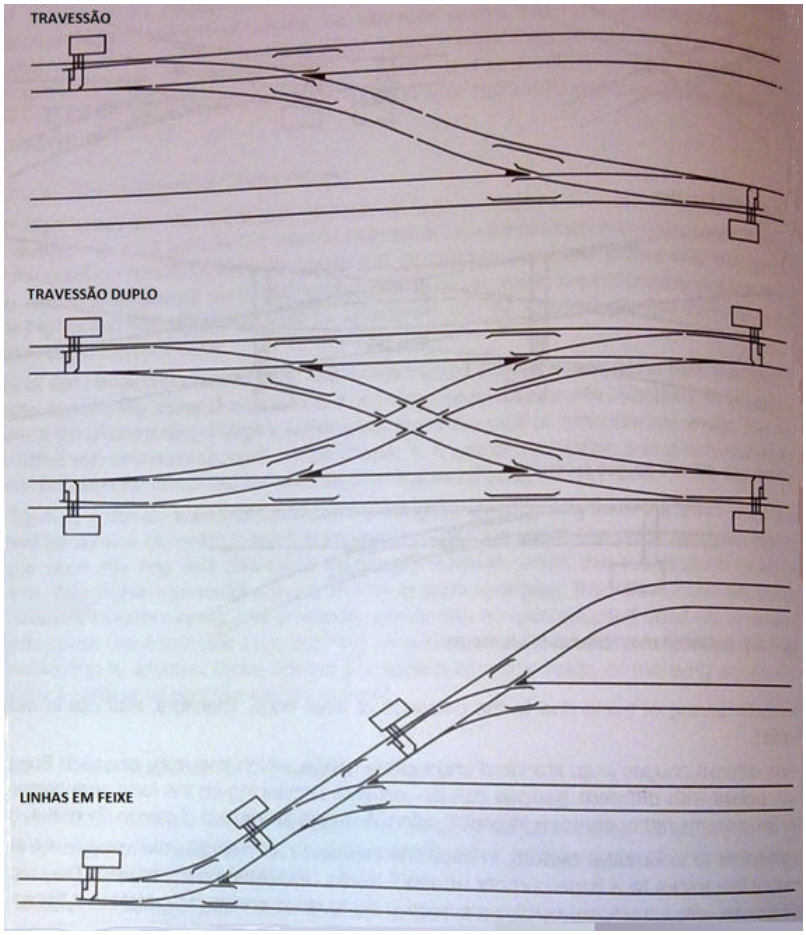
chave em uma posição: normal ou reversa. As agulhas são partes efetivamente móveis de uma chave, elas são responsáveis por fechar o caminho a ser percorrido pelos rodeiros de um veículo ferroviário, conduzindo para uma determinada via.

Segundo (NUNES, 2012), os contra-trilhos são os trilhos que impedem as rodas ao passarem pelo jacaré, entrem na via errada e ocasionem um descarrilamento. O jacaré é a região de intersecção entre duas vias. Neste ponto é onde podem ocorrer desvios de uma roda, causando descarrilamento do veículo. Desse fato surge então a necessidade do uso de contra-trilhos.

2.3.2 Associações de chaves

De acordo com a necessidade, diferentes layouts podem ser formados a partir da associação de duas ou mais chaves. A Figura 14 apresenta as mais facilmente encontradas.

Figura 14 – Exemplos de arranjos de chaves.



Fonte: (NUNES, 2012).

Um travessão é um arranjo simples entre duas chaves e uma solução típica de conexão entre duas linhas paralelas. Podem ser tanto operadas em conjunto quanto individualmente, porém o alinhamento de rotas depende de as duas estarem em uma posição favorável a essa rota.

Um travessão duplo ou travessão universal é a junção de dois travessões, podendo ser juntos ou separados. Dessa forma, todas as possibilidades de rota podem ser realizadas. A fim de garantir a segurança, movimentos retos nas linhas podem ser realizados independentemente, mas qualquer rota com mudança de linha bloqueia outras rotas.

Linhas em feixe são formadas por duas ou mais chaves com mesmo ângulo para separar uma linha em três ou mais. São utilizados normalmente em pátios com diversas linhas, pois sua configuração permite maior densidade de malha, ou seja, melhor aproveitamento do espaço (NUNES, 2012).

2.3.3 Máquinas de chave

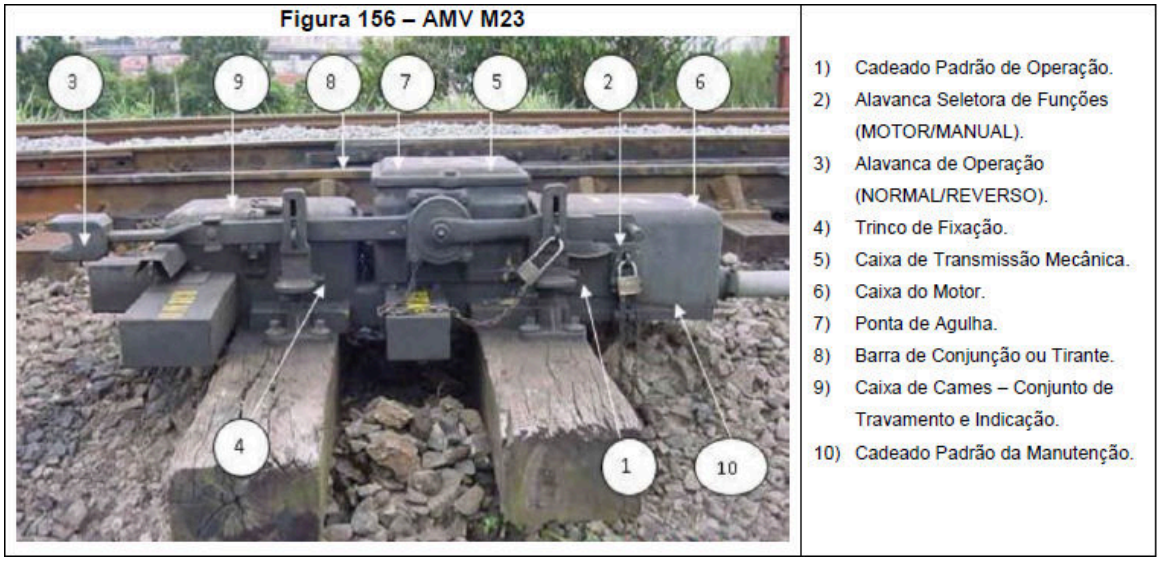
Dá-se o nome de máquina de chave, aos dispositivos responsáveis pela movimentação das agulhas de um AMV. Sendo então, três as funções associadas ao movimento: Alterar a posição da chave, aplicando força sobre as agulhas; Travar a chave, segurando as agulhas na posição apos o término do movimento; Supervisionar a chave, indicando a posição atual das lâminas às pessoas de interesse e ao sistema de sinalização.

A indicação, em especial, é muito importante por questões de segurança. Um trem que passe sobre uma chave que não está corretamente posicionada e indicada, pode gerar destruição da máquina de chave e, no pior dos casos, descarrilamentos ou colisões com veículos que estão na linha adjacente. As máquinas de chave segundo (NUNES, 2012) podem ser talonáveis ou não talonáveis. As primeiras permitem que os rodeiros do trem, ao passarem por sobre as agulhas em posição contrária, movimente-as a fim de permitir sua passagem e a não quebra da chave. As últimas, em contrapartida, não permitem esse movimento das agulhas, sendo elas travadas mecanicamente. Os tipos mais comuns de máquina de chave são:

- **Manual**: São operadas local e manualmente através da movimentação de uma alavanca;
- **Eletro-pneumáticas**: O movimento das agulhas é governado pela força do ar armazenado em reservatórios;
- **Eletromagnéticas**: A energia elétrica é transformada em energia mecânica através de solenóides;
- **Eletromecânicas**: A energia elétrica é transformada em mecânica através de motores conectados a engrenagens;
- **Eletro-hidráulicas**: O movimento das agulhas é governado por motores hidráulicos.

A Figura 15 ilustra um modelo de máquina de chave e destaca suas principais partes.

Figura 15 – Exemplo de máquina de chave.



Fonte: (MIGUEL et al., 2018).

O esquema de ligação varia bastante para cada modelo de máquina de chave, porém na média sempre será necessário 2 condutores para ligar o motor da chave, 2 condutores para o comando da máquina de chave e pelo menos 4 condutores de indicação da posição da máquina de chave. Caso seja um travessão completo (duas máquinas de chave), ainda será necessário lançar cabos de uma máquina até a outra e dessa por sua vez até o concentrador. Em média para um travessão completo, são utilizados 12 condutores elétricos que percorrem o caminho concentrador - maq1 - maq2 - concentrador. Para controle e indicação de apenas um travessão completo que está apenas 100 m do concentrador, será necessário aproximadamente 1,5 km de cabos de cobre.

2.4 SCTC

Segundo (BUENO, 2006) o sistema de sinalização é controlado a partir de CCOs (Centro de Controle e Operação), localizado em um ponto estratégico. Este sistema, denominado SCTC, é composto de uma rede de microcomputadores do tipo *Workstation* nos sistemas mais modernos e engloba todos os equipamentos necessários para o controle e supervisão do tráfego de trens. Esse sistema dispõe de recursos que irão permitir a automatização de operações, bem como para gerenciamento de atividade de manutenção dos sistemas de sinalização e controle.

Os equipamentos de IHM (Interface Homem-Máquina) fornecem aos controladores, os meios apropriados para permitir a execução das seguintes tarefas:

- Requisição de comandos simples e memorizados;
- Requisição de funções avançadas;

- Tratamento de alarmes;
- Emissão de relatórios operacionais.

Elas também permitem aos controladores, a visualização das seguintes indicações operacionais:

- Posicionamento e prefixação das composições ferroviárias;
- Identificação e rastreamento automático dos trens;
- Correspondência das máquinas de chave;
- Estado operacional dos sinais a margem da via;
- Estado operacional das funções avançadas e de controle;
- Alarmes.

O sistema é preparado para permitir a operação totalmente automática, de acordo com programas pré-estabelecidos de movimentação de trens. A arquitetura de *hardware* do SCTC proporciona uma elevada disponibilidade e autonomia dos diversos subsistemas que o compõe. As funções consideradas como críticas, tais como banco de dados, controle de tráfego de trens e comunicação são dualizadas e incorporam mecanismos de chaveamento automático em caso de falhas.

Segundo (BUENO, 2006) o SCTC é responsável pelo controle e gerenciamento do tráfego, a partir de dados recebidos do campo através de URs ou de comandos gerados no CCO e enviados ao campo pelo STD central. No CCO, os dados são processados por equipamentos informatizados e visualizados pelo controlador. O SCTC é composto pelos seguintes subsistemas:

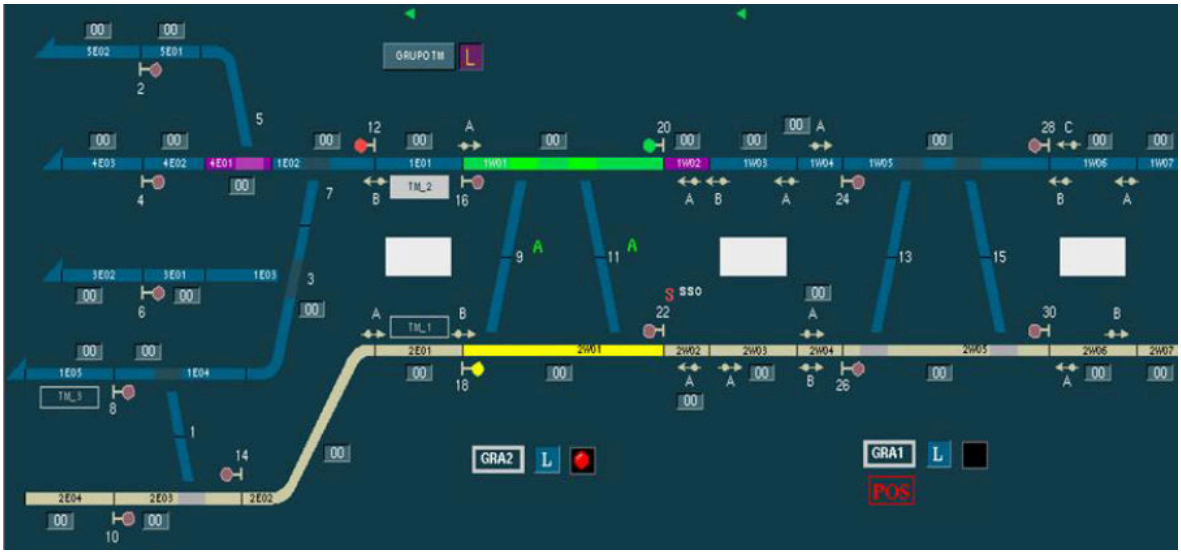
- IHM constituídas de consoles de operação, console de supervisão, console de manutenção e engenharia e painéis sinóticos;
- Servidor SCADA (*Supervisory Control and Data Acquisition*) dualizado que agrega as funções de banco de dados e controle de tráfego de trens;
- Unidades de acionamento de painéis sinóticos;
- Rede local de comunicação padrão ethernet;
- Subsistema de transmissão de dados, composto de STD (Sistema de Transmissão de Dados) central e UR (Unidade Remota);

- Subsistema de alimentação ininterrupta de *No-Break* e banco de baterias.

A interação entre Banco de dados e IHM é feita segundo a arquitetura Cliente/Servidor, sendo os equipamentos pertencentes à IHM e os clientes de bancos de dados residente no servidor.

A Figura 16 ilustra um exemplo da IHM do painel sinótico usado no CCO.

Figura 16 – Exemplo de IHM no CCO.



Fonte: (MIGUEL et al., 2018).

O STD é responsável pela comunicação que permite o controle remoto dos equipamentos de campo e a supervisão dos seus estados operacionais. O STD recebe do SCTC, as seguintes informações principais:

- Comando de posição de máquina de chave;
- Comando de requisição de sinal;
- Comando de chamada de manutenção;
- Comando de interface: seleção de estação;

O STD envia ao SCTC as seguintes informações principais:

- Indicação de posicionamento de máquina de chave: reversa ou normal;
- Indicação de ocupação de circuito de via;
- Indicação de sentido de rota alinhada;
- Indicação de aspecto de sinais: fechamento ou aberto;

- Alarmes e diagnósticos;

O STD possibilita a transmissão e a recepção das informações entre a UC (Unidade Central) e a UR. A UC instalada no CCO é capaz de conseguir codificar e transmitir mensagens de comando, receber, decodificar e registrar as mensagens de indicação. Segundo (BUENO, 2006) as UR's, instaladas em locações de campo, são capazes de receber, decodificar e registrar mensagens de comando, possibilitando o acionamento de circuitos elétricos que acionam equipamentos de sinalização e de codificar e transmitir mensagens de indicação quando acionados os circuitos de interface com os equipamentos de sinalização. São empregados circuitos à base de microprocessadores para conseguir efetuar o processamento e gerenciamento de todas as mensagens trocadas entre o CCO e as UR's.

Os módulos de microprocessadores e modems da UC são duplicados, de forma a manter uma disponibilidade em tempo integral das funções do CCO comutando-se automaticamente para o módulo reserva em casos de falha no principal.

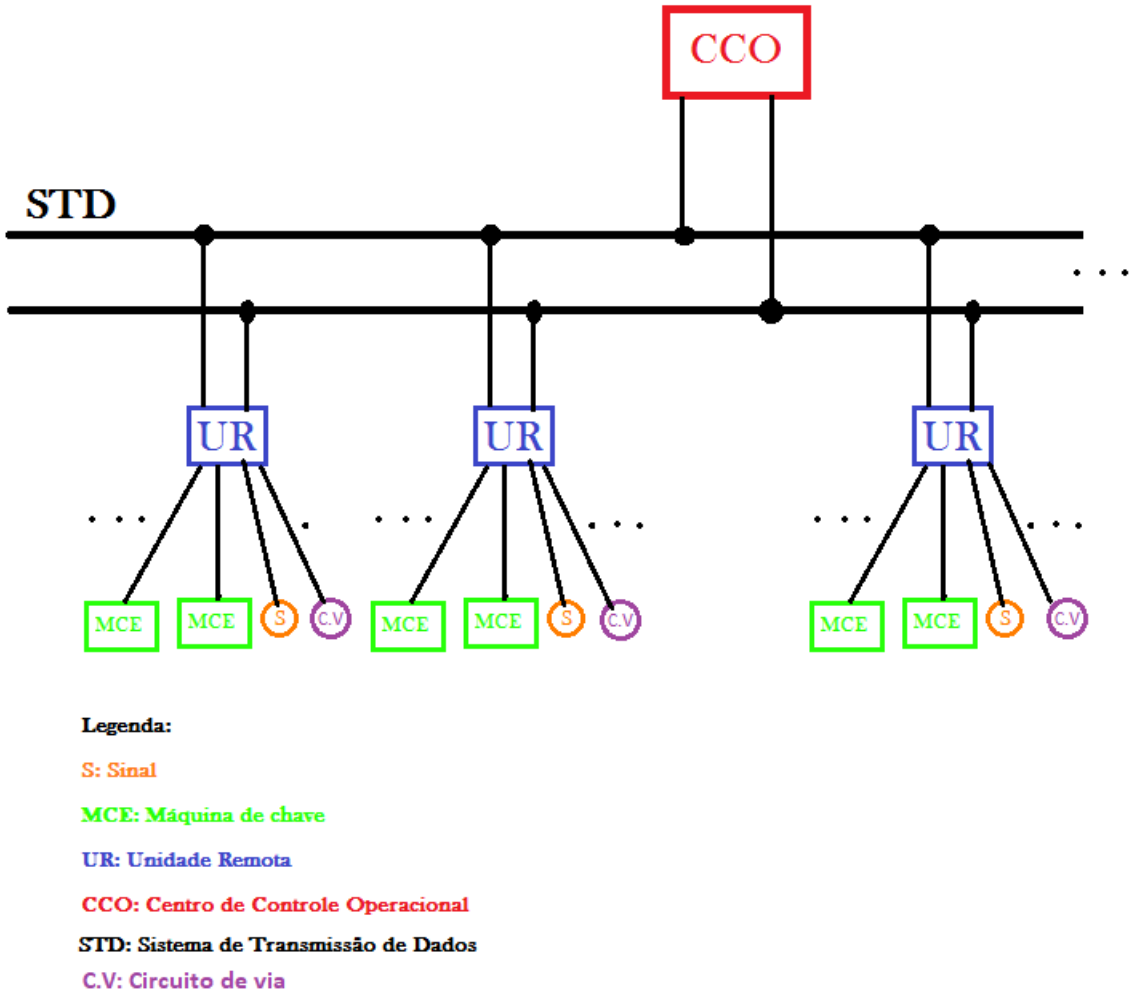
A Figura 17 ilustra o diagrama simplificado das conexões entre equipamentos de campo, UR (as vezes denominado concentrador), o STD e o CCO. Analisando a figura pode-se verificar que cada equipamento em campo será ligado a UR, que por sua vez irá processar as indicações e enviar via o STD as informações para o CCO, para que o UC possa processar isso e atualizar a IHM para os controladores.

Pode-se perceber pelas informações apresentadas nesse capítulo, que para controlar e supervisionar o sistema de sinalização, uma vez que cada equipamento deve ser ligado até a UR, quanto mais longe o equipamento, maior deverá ser a seção dos condutores e a metragem necessária. Uma estação de pequeno porte com 3 travessões completos (duas máquinas de chave em cada travessão), mais os sinais para proteger os travessões, somado aos circuitos de via, facilmente se gasta 10 km de cabo de condutores elétricos.

O fato das linhas ferroviárias serem extensas e abertas dificulta a proteção desse patrimônio, que fica sujeita a vandalismos e furtos, furtos esses que trazem um prejuízo grande, uma vez que será necessária a compra do cabo furtado, além do uso da mão de obra de funcionários que poderiam estar realizando manutenções preventivas no sistema de sinalização.

Visto esse problema, se faz necessário uma mudança na forma do controle e monitoramento dos equipamentos da sinalização. A utilização de rede sem fio é uma ótima opção para a comunicação das UR's com os equipamentos controlados pela mesma. No próximo capítulo, será apresentado um padrão de comunicação sem fio que pode atender os requisitos para controle e monitoramento dos equipamentos.

Figura 17 – Diagrama de conexão entre campo e CCO.



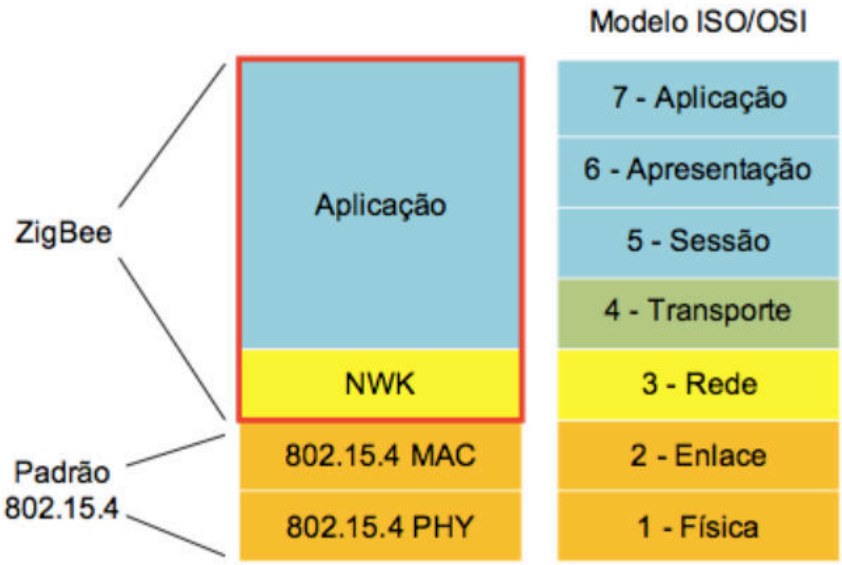
Fonte: Autor.

3 ZigBee

Em 2004, uma associação de empresas, que foi denominada *ZigBee Alliance* (hoje é conhecida como *Connectivity Standards Alliance* segundo (ALLIANCE, 2022)), desenvolveu um padrão chamado ZigBee que adiciona duas camadas de alto nível de rede no modelo OSI para a estrutura estabelecida pelo padrão IEEE 802.15.4 (IEEE, 2016). O objetivo do ZigBee era ser um padrão de comunicação sem fio que provê uma rede de curto alcance e boa relação custo benefício, sendo otimizado para aplicações com baixo custo alimentadas por baterias, tais como automação predial, controle industrial e comercial entre outros (OLIVEIRA, 2015).

A arquitetura do ZigBee foi desenvolvida em camadas. Cada uma com suas funções específicas que são utilizadas pelas camadas superiores. Como mencionado anteriormente as duas primeiras camadas são determinadas pelo padrão 802.15.4, já as outras duas camadas (rede e aplicação) foram desenvolvidas pela *ZigBee Alliance*. A Figura 18 apresenta as camadas do padrão ZigBee e as camadas do modelo OSI (*Open System Interconnection*). Analisando a figura pode-se verificar que as camadas de transporte, sessão e apresentação do modelo OSI, foram incorporadas dentro da camada de aplicação do ZigBee. A seguir será apresentada cada uma das camadas do padrão ZigBee.

Figura 18 – Camadas do padrão ZigBee.



Fonte: (KAORU, 2013).

3.1 Camada física

A camada física ou camada PHY (*Physical*) do padrão 802.15.4 é a responsável pelo envio das PDU (*Protocol Data Unit*). Também é responsabilidade da camada PHY indicar a qualidade da conexão, identificar canais livres e detectar suas potências. As tarefas executadas pela camada PHY são:

- Ativação e desativação do rádio transceptor
- Detecção de energia (ED - *Energy Detection*) dentro do canal atual
- Indicador de qualidade do link (LQI - *Link Quality Indicator*) para pacotes recebidos
- Avaliação de canal claro (CCA - *Clear Channel Assesment*) para acesso múltiplo com detecção de portadora com prevenção de colisão (CSMA-CA - *Carrier Sense Multiple Access with Collision Avoidance*)
- Seleção de canal de frequência
- Transmissão e recepção de dados
- Faixa de precisão para modulações de banda ultra larga (UWB - *Ultra-Wide Band*)

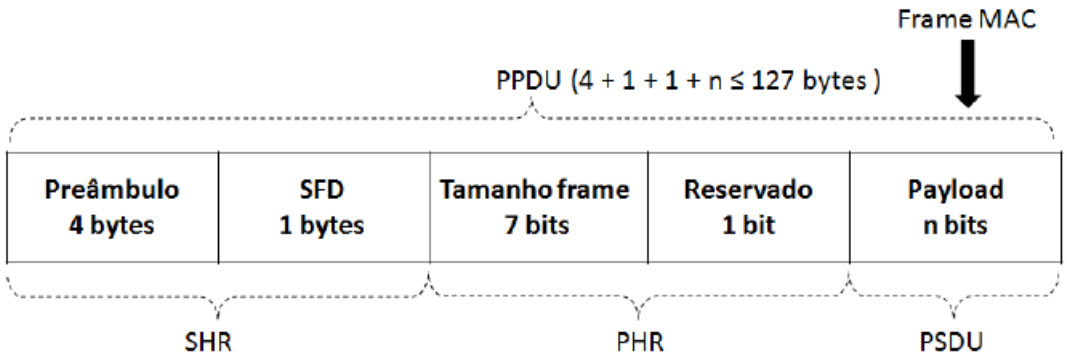
3.1.1 Pacotes de dados da camada PHY

O pacote de dados da camada PHY é chamada PPDU (*PHY Protocol Data Unit*). Após o cabeçalho ser adicionado ao *frame* da camada superior MAC (*Media Acess Control*), o pacote será transmitido ao meio. O PPDU pode ser fragmentado em três partes: O cabeçalho de sincronização (SHR - *Synchronization Header*), cabeçalho de informação do tamanho do frame (PHR - *PHY Header*) e a parte de dados que contém o *frame* vindo da camada MAC (PSDU - *PHY Service Data Unit*). A Figura 19 ilustra o pacote de dados da camada PHY.

O cabeçalho SHR é formado pelos campos preâmbulo e pelo SFD (*Start of Frame Delimiter*). O preâmbulo é formado por uma sequência de bits usada para sincronizar o dispositivo receptor com o pacote de dados que foi recebido. O campo SFD só delimita o fim dos *bytes* de sincronização e o início dos campos de dados.

A frequência do oscilador é quase sempre diferente de dispositivo para dispositivo. Essa diferença, apesar de pequena, causa um descompasso na temporização de eventos. Um descompasso entre receptor e transmissor pode prejudicar a recepção de um pacote de dados, mesmo que a frequência de oscilação entre eles seja ligeiramente diferente. Caso este problema ocorra, ele será corrigido por meio da sequência de *bits* do preâmbulo, que é o primeiro campo a ser enviado pelo transmissor e contém 32 *bits* nulos. Já o campo SFD

Figura 19 – Pacote da camada física.



Fonte: (BRONZATTI, 2013).

deve ser setado com a sequência de *bits* 11100101, onde nessa sequência o *bit* LSB (*Least Significant Bit* está localizado o mais à esquerda (BRONZATTI, 2013).

O campo PHR descreve o tamanho do campo *payload* ou PSDU, que varia de 0 a 127 *bytes*. O *payload* é o campo que contém o *frame* da camada MAC. Para um comprimento de 5 *bytes*, o PSDU será formado por um *frame* da camada MAC de reconhecimento. Para um comprimento maior que 8 *bytes*, ele será formado por outros tipos de *frame* da camada MAC.

3.1.2 Frequência e canais

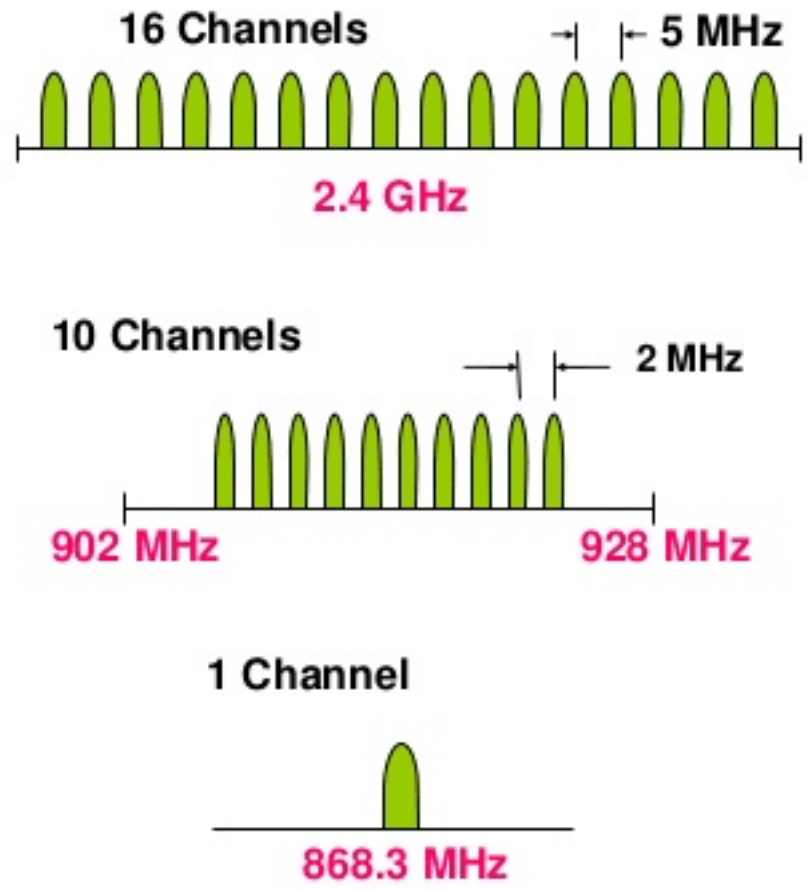
O padrão IEEE 802.15.4 permite o uso de três faixas de frequência de operação. A Figura 20 ilustra a divisão do espectro, conforme os canais.

Cada uma dessas faixas de frequência de operação é dividida em faixas menores para formar os diversos canais de comunicação. Somando os canais das três faixas de frequência de operação, o padrão ZigBee possibilita o uso de 27 canais de comunicação enumerados de $k = 0$ *a* 26. Na faixa de 868 MHz há apenas 1 canal de transmissão/recepção, em 915 MHz, há 10 canais e em 2,4 GHz existem 16 canais à disposição. Nesse trabalho será apresentado apenas a faixa de frequência de operação de interesse de 2,4 GHz. A frequência central de cada canal k ($f_{central}(k)$) pode ser calculada em função de k utilizando a equação 3.1 (BRONZATTI, 2013):

$$f_{central}(k) = 2405 + 5 \times (k - 11)\ MHz,\ 11 \leq k \geq 26 \tag{3.1}$$

Para a faixa de frequência de operação de 2,4 GHz, a largura de cada canal é de 2 MHz segundo (IEEE, 2016).

Figura 20 – Faixas de frequências e canais utilizados.



Fonte: (MELO, 2017).

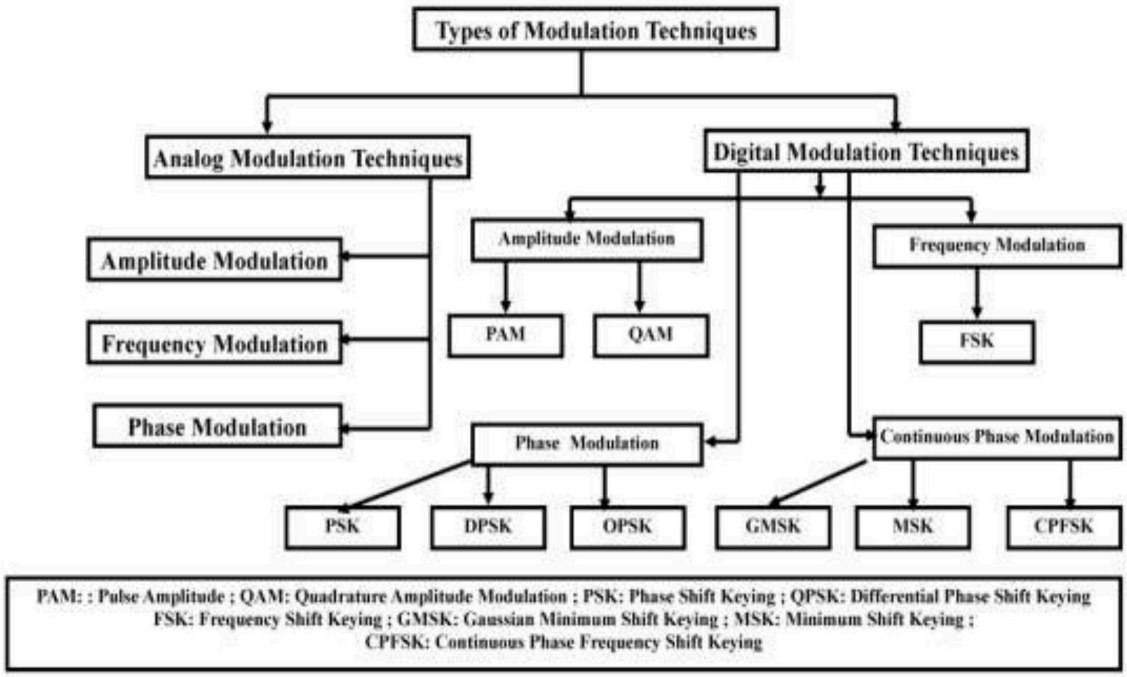
3.1.3 Modulação e espalhamento

Segundo (HAYKIN; MOHER, 2011), a modulação constitui-se da técnica empregada para modificar um sinal com a finalidade de transportar uma informação através de um canal de comunicação e recuperar o sinal original no destino. Uma modulação pode ser classificada pela sua portadora (podendo ser analógica ou digital) e quanto o tipo de informação (analógica ou digital). A Figura 21 exemplifica alguns tipos de modulação.

Já o espalhamento espectral é uma técnica onde a energia média do sinal transmitido é espalhada sobre uma largura de banda que é muito maior que a largura de banda que contém a informação. Esses sistemas compensam uma maior largura de banda de transmissão, por uma menor densidade espectral de potência e proporcionam uma melhora na rejeição aos sinais interferentes operando na mesma faixa de frequência

De acordo com o padrão Zigbee, a modulação e o espalhamento a serem aplicados, variam de acordo com a faixa de frequência de operação. Conforme mencionado anteriormente, será apenas apresentado a modulação e espalhamento para a faixa de frequência

Figura 21 – Tipos de modulação.

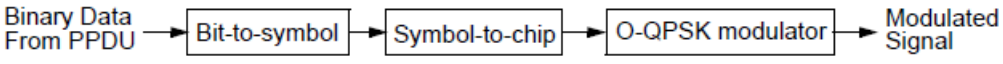


Fonte: (JANGRA; KUMAR, 2020).

de operação em 2,4 GHz. Nessa faixa de frequência de operação, segundo (IEEE, 2016), o tipo de modulação escolhido é o OQPSK (*Offset Quaternary Phase Shift Keying*) e a técnica de espalhamento utilizada é a DSSS (*Direct Sequence Spread Spectrum*). Para mais informações sobre a modulação e seu funcionamento, consultar o anexo A e para mais informações sobre a técnica de espalhamento espectral consultar o anexo B.

Durante cada período de símbolo de dados, quatro bits de informação são usados para selecionar uma de dezesseis sequências PN (*Pseudo Noise*) quase ortogonais a serem transmitidas. As sequências PN para os símbolos de dados sucessivos são concatenadas e então modulada na portadora, usando OQPSK. A Figura 22 mostra um diagrama de blocos simplificado do transmissor.

Figura 22 – Diagrama de blocos de modulação DSSS/OQPSK na frequência de 2.45 GHz.



Fonte: (IEEE, 2016).

A taxa de transmissão de *bits* é de 250 kb/s. Os *bits* de dados que vem do PPDU, são aqueles montados conforme a Figura 19, onde nesse caso, o preâmbulo será composto por 8 símbolos (isto é, 4 octetos) e os *bits* devem ser todos "zeros". Já o campo SFD segue o padrão descrito anteriormente de "11100101"em binário e o PHR segue o formato da Figura 19.

Todos os dados binários no PPDU devem ser codificados usando a modulação e espalhamento descrito na Figura 22. Os 4 *bits* menos significativos (LSBs) b0, b1, b2 e b3 de cada octeto devem ser mapeados em um símbolo de dados. Cada octeto da PPDU é processado através da modulação e espalhamento da Figura 22 sequencialmente, se iniciando pelo Preâmbulo até o último octeto da informação do PSDU. Em cada octeto, o símbolo menos significativo (b0, b1, b2 e b3) é processado primeiro e o símbolo mais significativo (b4, b5, b6 e b7) é processado depois.

Cada símbolo de dados deve ser mapeado dentro de uma sequência PN de 32 *chips*. Essas sequências são especificadas na Figura 23. Essas sequências PN estão relacionadas entre si por meio de deslocamentos cíclicos e/ou conjugação (ou seja, inversão de valores de chip com índice ímpar).

Figura 23 – Sequências PN para cada símbolo de dados usadas pelo padrão ZigBee.

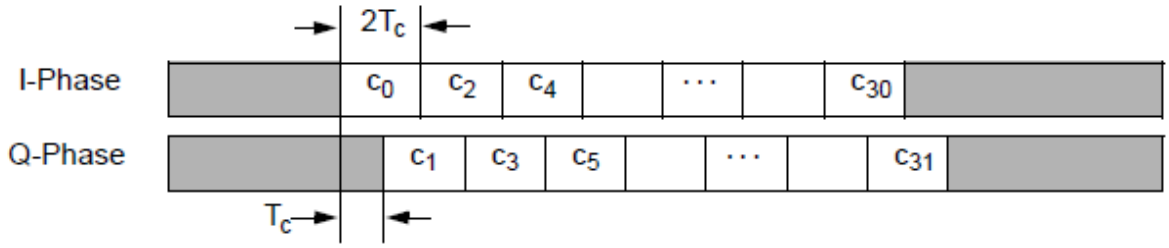
Data symbol	Chip values (c_0 c_1 … c_{30} c_{31})
0	1 1 0 1 1 0 0 1 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0
1	1 1 1 0 1 1 0 1 1 0 0 1 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0
2	0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0
3	0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 1 0 0 0 0 1 1 0 1 0 1
4	0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 1 0 0 0 0 1 1
5	0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0
6	1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1
7	1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1
8	1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1
9	1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1
10	0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1
11	0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0
12	0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0
13	0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1
14	1 0 0 1 0 1 1 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0
15	1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0

Fonte: (IEEE, 2016).

As sequências de *chips* que representam cada símbolo de dados são moduladas na portadora usando OQPSK utilizando uma formatação de pulsos filtrados com *half-sine*. Os chips com índices pares são modulados na portadora I (*in phase*) e os chips com índice ímpar são modulados na portadora Q (*quadrature*). Cada símbolo de dados é representado

por uma sequência de 32 *chips* e, portanto, a taxa de *chips* é 32 vezes a taxa de símbolo, ou seja, uma taxa de 2000 kchips/s. Para formar o deslocamento entre a modulação do *chip* da fase I e Q, os *chips* da fase Q serão atrasados por T_c em relação aos *chips* da fase I, conforme ilustrado na Figura 24 onde T_c é o inverso da taxa de *chip*.

Figura 24 – Deslocamento dos *chips* entre I e Q.



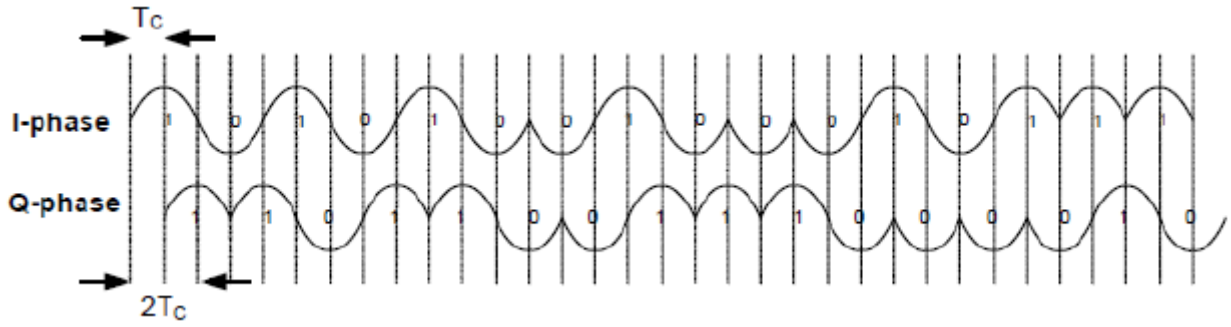
Fonte: (IEEE, 2016).

A formatação de pulso filtrado com *half-sine* utilizada para representar cada chip em um sinal passa-baixa é a apresenta na equação 3.2.

$$p(t) = \begin{cases} \sin\left(\pi \frac{t}{2T_c}\right), & 0 \leq t \leq 2T_C \\ 0, & Outro\ valor \end{cases} \tag{3.2}$$

A Figura 25 mostra uma amostra de sequência de *chips* (sequência zero) em um sinal passa-baixa com formatação de pulso filtrado com *half-sine*.

Figura 25 – Exemplo de amostra de sequência de *chips* em um sinal passa-baixa com formatação de pulso filtrado com *half-sine*.



Fonte: (IEEE, 2016).

Durante cada período de símbolo, o *chip* menos significativo, c_0, é transmitido primeiro e o *chip* mais significativo, c_{31} por último.

A sensibilidade mínima do receptor deve ser de pelo menos -85 dBm ou melhor e deve apresentar uma rejeição mínima de 30 dB para canais alternados e 0 dB para canais adjacentes. O dispositivo também deve ser capaz de mensurar a LQI. A medida LQI é a caracterização da força e/ou a qualidade do pacote recebido. A medida pode ser implementada usando um receptor ED como uma estimativa da razão sinal-ruído ou a

combinação desses métodos. O uso do resultado LQI pela rede ou camadas de aplicação não é especificado no padrão 802.15.4 (IEEE, 2016).

A medição LQI deve ser realizada para cada pacote recebido. Os valores de LQI mínimo e máximo (0x00 e 0xFF) devem ser associados ao sinais compatíveis de qualidade baixa e mais alta detectáveis pelo receptor, e os valores LQI intermediários devem ser uniformemente distribuídos entre esses dois limites. Devem ser usados pelo menos oito valores únicos de LQI.

A Figura 26 mostra um resumo dos tipos de modulações e taxas em cada uma das três faixas de frequência de operação.

Figura 26 – Resumo das modulações de acordo com as frequências.

PHY (MHz)	Banda de frequência (MHz)	Parâmetros de espalhamento		Parâmetros dos dados		
		Taxa de espalhamento (kchip/s)	Modulação	Taxa de bit (kb/s)	Taxa de símbolo	Símbolos
868/915	868-868.6	300	BPSK	20	20	Binário
	902-928	600	BPSK	40	40	Binário
2450	2400-2483.5	2000	O-QPSK	250	62.5	16 símbolos

Fonte: (UFRJ, 2016).

3.2 Camada MAC

A camada MAC controla o acesso ao canal de rádio compartilhado. Ela irá gerar e reconhecer os endereços e verifica as sequências das estruturas de controle. Ela também é responsável por sincronizar as transmissões dos *frames* no modo *non-beacon*, usando o método CSMA-CA, onde um nó verifica a possibilidade de transmissões simultâneas. Depois que o canal é encontrado e determinando como livre, o nó inicia a transmissão. Quando o nó suspeita de uma colisão, ele imediatamente para de transmitir e espera um tempo aleatório.

3.2.1 Modo *Beacon* e *non-Beacon*

A rede ZigBee, funcionando em seu modo *beacon*, utiliza a estrutura de superquadro para o acesso dos dispositivos ao canal de transmissão. Nessa estrutura, o acesso é limitado dentro de períodos de tempo pré-determinados, parecido com o que ocorre em uma multiplexação por divisão de tempo. Um dispositivo desejando transmitir terá a oportunidade de fazê-lo somente dentro do período ativo do superquadro, que ocorre após a sinalização do coordenador. O coordenador transmite um sinal *Beacon* aos outros dispositivos, marcando o início do período de troca de dados dentro da rede. Quando recebem o Beacon, os dispositivos estão livres para transmitir os seus dados, caso o tenham

durante o período ativo do superquadro. Os dispositivos sabem o intervalo de tempo que tem para trocarem seus dados, pois após esse intervalo, eles entrarão em modo de suspensão para preservarem o máximo de energia possível (IEEE, 2016).

Já no modo *non-Beacon*, não existe o superquadro ou transmissão de um sinal *Beacon* pelo coordenador, assim não há também, um período específico para se transmitir. Qualquer dispositivo desejando transmitir, o poderá fazer a qualquer momento, bastando apenas que o canal esteja livre.

3.2.2 Estrutura do superquadro

O superquadro ou *superframe* do modo *Beacon*, é dividido em duas partes, sendo a primeira a parte ativa e a segunda a parte inativa.

Segundo (IEEE, 2016), a parte ativa compreende o período do superquadro liberado para transmissões. Ele começa a partir da emissão do coordenador de um *frame Beacon* difundido na rede periodicamente.

A parte inativa começa após o término do período ativo e é o momento onde a rede fica "silenciosa", sem transações, e a maioria dos dispositivos aproveitam para entrar em modo de suspensão, reduzindo seu consumo de energia. O período de tempo t, das partes ativa e inativa variam. Em redes com grande volume de tráfego, a parte inativa do superquadro deve ser reduzida e nas redes com pouco tráfego, em que há uma margem para suspensão dos dispositivos, a parte inativa pode aumentar.

O período total do superquadro (BI) e da parte ativa (SD) são fixados pelos atributos *macBeaconOrder* (BO) e *macSuperframeOrder* (SO), respectivamente. O período inativo corresponderá ao período do superquadro menos o período ativo. As equações de 3.3 a 3.6 determinam o tempo do superquadro, da parte ativa, parte inativa e *slot*, ressaltando que a unidade desses resultados é em unidade de símbolos, ou seja, para uma mesma condição de BO e SO terão tempo em segundo diferente para o OQPSK e BPSK (*Binary Phase Shift Keying*) (IEEE, 2016). *aBaseSUperframeDurantion* e *aBaseSlotDurantion* são constantes definidas no padrão 802.15.4 (IEEE, 2016).

$$t_{superframe} = BI = aBaseSuperframeDurantion \times 2^{BO} \quad 0 \leq BO \leq 14 \tag{3.3}$$

$$t_{ativa} = SD = aBaseSuperframeDuration \times 2^{SO} \quad 0 \leq SO \leq 14 \tag{3.4}$$

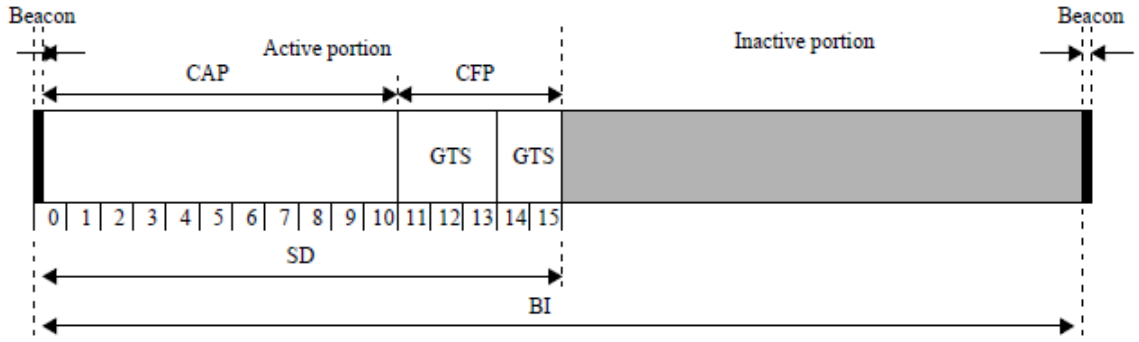
$$t_{slot} = aBaseSlotDurantion \times 2^{SO} \tag{3.5}$$

$$t_{inativo} = aBaseSuperframeDurantion \times (2^{BO} - 2^{SO}) \tag{3.6}$$

Para as redes no modo *non-Beacon* BO e SO deverão ser iguais a 15. Sendo assim, o coordenador não irá transmitir os *Beacons* e o acesso ao meio será feito através do *unslotted* CSMA-CA

A Figura 27 demonstra como é dividido o superquadro.

Figura 27 – Estrutura do superquadro.



Fonte: (IEEE, 2016).

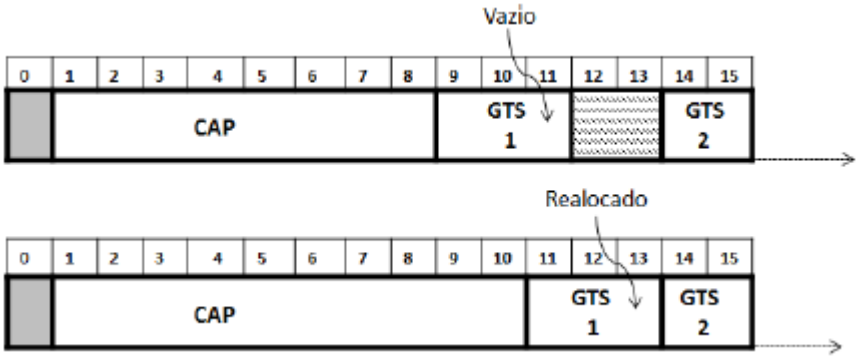
Toda a parte ativa é dividida em 16 *slots* de igual período t_{slot}, que também varia de acordo com o valor de SO. O primeiro *slot* é reservado para o sinal *Beacon* e os restantes são distribuídos entre o período de disputa pelo acesso (CAP - *Contention Acess Period*) e o período livre de disputa (CFP - *Contention Free Period*). O sinal *Beacon* é considerado pertencente à parte ativa do superquadro. Dentre outras coisas, o *frame Beacon* contém informações de sincronização, de identificação da rede WPAN e de descrição da estrutura do superquadro(BRONZATTI, 2013).

O CAP deve iniciar imediatamente seguido do *Beacon* e deve terminar antes do início do CFP. No período do CAP, o canal não é reservado por um intervalo de tempo para um único dispositivo. Para acessar o canal, será preciso verificar se o mesmo se encontra livre pelo mecanismo *slotted* CSMA-CA. Um dispositivo transmitindo durante o período de disputa de acesso ao canal deverá certificar-se de que a transferência de dados seja finalizada antes do fim do período do CAP.

O CFP deve começar em um limite de *slot* imediatamente após o CAP e deve ser concluído antes do final da parte ativa do superquadro. No período do CFP não há disputa pelo acesso do canal de comunicação, uma vez que são reservados períodos GTSs (*Guaranteed Time Slots*). Durante um GTS o canal será alocado exclusivamente para um dispositivo e, portanto estes não precisarão disputar o acesso ao canal. Ressaltando também que os dispositivos que alocaram GTS também estão liberados para transmitir no período do CAP. Para gerência da alocação de espaços dentro do CFP, os coordenadores detêm informações como: *slot* de início e comprimento do GTS e endereço dos dispositivos que alocam espaço entre outros dados (BRONZATTI, 2013).

Se após alguns períodos de superquadro um dispositivo que aloca GTS não usá-lo para transmissão, terá seu espaço GTS liberado para outros dispositivos. Se após p superquadros não houver transmissões dentro do GTS, o coordenador retirará o GTS alocado para o dispositivo, liberando mais espaço CAP, como mostrado na Figura 28.

Figura 28 – Gerenciamento dos *slots* GTS.



Fonte: (BRONZATTI, 2013).

A equação 3.7 define o valor de p.

$$p = \begin{cases} 2 \times 2^{(8-macBeaconOrder)} & 0 \leq macBeaconOrder \leq 8 \\ 2 & 8 < macBeaconOrder \end{cases} \tag{3.7}$$

3.2.3 CSMA-CA

É um protocolo utilizado para permitir o acesso ao meio por vários usuários e prevenir o uso simultâneo do canal por mais de um dispositivo, em um mesmo momento.

No protocolo CSMA-CA, será esperado um período aleatório antes de checar se o canal está livre. Somente após o final desse período aleatório, o dispositivo irá verificar se o canal está desocupado e, caso esteja, transmitirá. Entretanto, se o canal estiver ocupado, o dispositivo terá que aguardar novamente por um novo período aleatório e, só então, fazer a checagem de canal novamente. Esse procedimento irá se repetir até o dispositivo querendo transmitir conseguir encontrar o canal livre. O período aleatório t_{delay} a se esperar é quantizado em unidade de $aUnitBackoffPeriod$, que é uma constante de tempo definida no padrão 802.15.4 (IEEE, 2016).

Cada dispositivo deve manter três variáveis para cada tentativa de transmissão, sendo elas NB, CW e BE.

NB é o número de vezes que o algoritmo CSMA-CA teve que recuar enquanto tentava a transmissão atual. Seu valor deve ser inicializado para zero antes de cada nova tentativa de transmissão.

CW é o comprimento da janela de contenção, que define o número de períodos de *backoff* que precisam estar livres de atividade no canal antes que a transmissão possa começar. O valor de CW deve ser inicializado para dois antes de cada tentativa de retransmissão e redefinido para $CW0$ cada vez que o canal é avaliado como ocupado. A variável CW é usada apenas para o *slotted* CSMA-CA.

BE é o expoente de *backoff*, que está relacionado a quantos períodos de *backoff*

um dispositivo deve esperar antes de tentar avaliar o canal. O valor mínimo de BE pode variar entre 0 a 3 e o valor máximo é 5. O mínimo é definido no atributo $macMinBE$ e o máximo, na constante $macMaxBE$, especificados no padrão (IEEE, 2016). O mecanismo de avaliação do canal será feito com o dispositivo no modo recepção.

Cada um dos modos (*Beacon* e *non-Beacon*) utiliza o algoritmos CSMA-CA ligeiramente diferentes, sendo o *slotted* CSMA-CA e *unslotted* CSMA-CA.

Segundo (BRONZATTI, 2013) *unslotted* CSMA-CA é usado para acessar o canal em redes no modo *non-Beacon*, onde não há estrutura de superquadro nem sinais *Beacons* enviados pelo coordenador. Os dispositivos poderão tentar ocupar o canal de comunicação a qualquer momento. Já o *slotted* CSMA-CA é o mecanismo empregado por dispositivos tentando acessar o canal no período CAP do superquadro, quando há disputa pelo acesso ao canal. Nesse método, é preciso fazer uma checagem dupla da disponibilidade do canal antes de se transmitir.

3.3 Camada de rede

A camada de rede ou camada NWK (*Network*) está entre a camada de aplicação (APL - *Application*) e a camada MAC e tem o papel de vincular corretamente uma aplicação aos níveis IEEE 802.15.4. Suas principais funções segundo (BRONZATTI, 2013) são:

- **Começar uma nova rede:** Habilidade de um coordenador de formar uma nova rede;
- **Configuração da rede:** A camada NWK de um coordenador tem a permissão de reconfigurar a rede alternando, por exemplo, a profundidade da rede, número de nós "filhos por roteador "pai", se necessário;
- **Juntar-se ou sair de uma rede:** A habilidade de um dispositivo decidir deixar ou entrar em uma rede, ou a de um coordenador ou roteador de pedir a saída de algum elemento da rede.
- **Descoberta de vizinhos:** A habilidade de descobrir informações sobre os nós vizinhos e armazenar estas informações na memória não volátil;
- **Endereçamento:** A habilidade de um roteador ou coordenador em atribuir um endereço de rede para um novo dispositivo que acabou de se juntar na rede;
- **Roteamento:** A habilidade de descobrir e armazenar diferentes rotas para a entrega de uma mensagem até o destinatário final. A camada de rede exibe mecanismos inteligentes de roteamento em malha, estendendo o alcance da rede;

- **Transmissão *multicast*:** A camada de rede possibilita que uma mensagem seja enviada a apenas um grupo de nós dentro da rede. Isto é uma funcionalidade adicionada pelo ZigBee, já que no 802.15.4 só é possível à transmissão para um único dispositivo ou para todos (*broadcast*);
- **Segurança:** A capacidade de adicionar segurança ao *payload* do *frame* NWK encriptando a mensagem.

3.3.1 Dispositivos

O padrão ZigBee define para as redes, dois tipos de dispositivos, os de função reduzida (RFD - *Reduced Function Device*), e os de função completa (FFD - *Full Function Device*).

Os dispositivos FFD são aqueles aptos a funcionarem em qualquer um dos modos de operação do padrão: Coordenador, roteador ou dispositivo final. Podem se comunicar tanto com os outros FFD quanto com dispositivos RFD.

Os dispositivos RFD são dispositivos que só podem se comunicar com dispositivos FFD. Dessa forma fica claro que esses dispositivos poderão atuar apenas como dispositivos finais (*End-pointings*) da rede. São dispositivos mais simples e de menor custo, visando um consumo de energia mais reduzido.

3.3.2 Funções lógicas dos dispositivos

De acordo com a disponibilidade de funções dos dispositivos (FFD ou RFD) e sua posição na rede, os nós podem ser classificados como: Coordenadores, roteadores ou dispositivos finais.

- **Coordenador:** O coordenador é o nó inicial da rede. Um dispositivo ao ser ligado pela primeira vez como coordenador iniciará sua rede selecionando um identificador PAN único no seu raio de influência. Na inicialização, todos os canais da frequência de operação são escaneados até esse PAN ID único ser encontrado. O coordenador opera em estado ativo para efetuar o controle da rede e costuma ser alimentado diretamente reduzindo o risco de falha no nó centralizado da rede.
- **Roteador:** Os dispositivos roteadores são usados em topologia em malha (*mesh*) e *cluster* para dar maior robustez à rede. Eles possuem tabelas de roteamento e, por serem FFD, permitem encontrar o menor caminho para se chegar ao destino. Caso o roteador não possua o endereço de destino requisitado, este fará o *broadcast* de uma requisição de rota (*Route Request*) e receberá do destino a rota mais eficaz atualizando sua tabela. Este mecanismo dá à rede a característica de auto regeneração caso ocorra a queda das funcionalidades de outros nós roteadores na rede.

- **Dispositivo final:** São os nós folhas das topologias estrelas e cluster. Por serem dispositivos RFD, não fazem função de roteamento nem coordenam a rede. Eles se comunicam diretamente com o roteador pai e podem ser implementados com microcontroladores ainda menores (em termos de memória e potência) passando quase todo o tempo em estado inativo. Um dispositivo RFD é a comum localização de sensores, atuadores e sistemas de controle.

3.3.3 Topologia de rede

A topologia é a maneira em que os dispositivos se agrupam em uma rede para se comunicarem. As topologias possíveis no ZigBee são: em estrela, em árvore e em malha segundo (BRONZATTI, 2013)

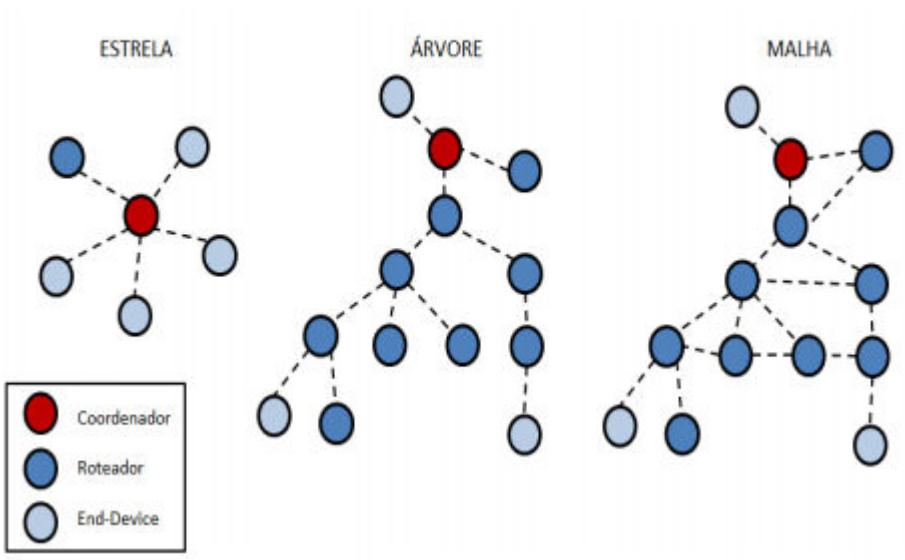
- **Topologia Estrela:** Nesta topologia um nó central (no ZigBee será o coordenador) é conectado com todos os outros nós da rede. Uma mensagem só chegará a um destinatário mediante passagem pelo coordenador. O esquema em estrela é mais simples, porém pouco dinâmico, no sentido em que todos os dados fluem para o nó central, muitas vezes sobrecarregando-o. Uma falha no nó central fará todas as comunicações da rede serem cortadas, já que não há rotas alternativas que não passem pelo coordenador.
- **Topologia em Árvore (*cluster*):** Na topologia em árvore, o coordenador é conectado a vários outros roteadores e *End-devices* em uma topologia, inicialmente, em estrela. Mas estes roteadores também acrescentarão outros *End-devices* ou outros roteadores à rede, gerando diferentes níveis hierárquicos acima.
- **Topologia em Malha (*mesh*):** Estrutura similar a topologia em árvore, porém os roteadores podem estabelecer comunicação entre si. Assim, criam caminhos redundantes entre dois nós e a estrutura torna-se resiliente com rotas alternativas em caso de falha de algum dispositivo. O roteamento, porém, torna-se mais complexo. A descoberta de rota é uma das características da rede em malha, na qual é verificada a melhor rota entre dois dispositivos distantes.

A Figura 29 ilustra as possíveis topologias no padrão Zigbee.

3.4 Camada de aplicação

A camada de aplicação ou camada APL, segundo (OLIVEIRA, 2015) é a camada de nível mais alto do sistema de comunicação sem fio ZigBee, esta camada é subdividida em três seções: Suporte à Aplicação (*Application Support*), *ZigBee Device Objects* e *Application Framework*.

Figura 29 – Topologias do padrão ZigBee.



Fonte: (BRONZATTI, 2013).

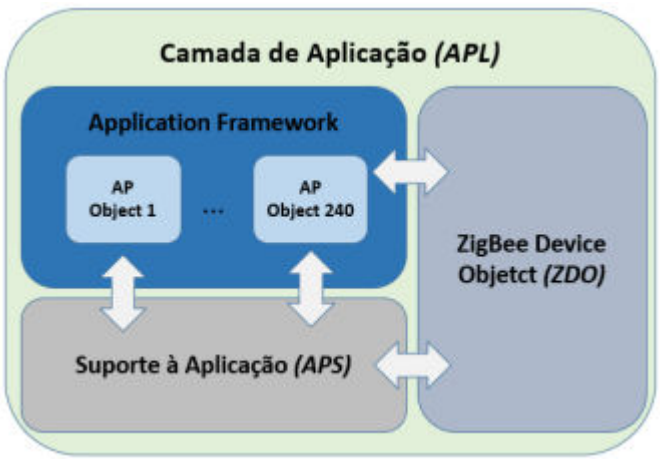
A seção de Suporte à Aplicação basicamente oferece uma interface entre a camada de rede (NWK) e a camada de aplicação (APL).

A *Application Framework* é a seção onde se encontram os objetos de aplicação armazenados para o controle e gerenciamento dessa camada em um dispositivo ZigBee.

Por fim, a seção *ZigBee Device Object* é responsável por intermediar a atuação da camada de aplicação promovendo uma melhor integração entre esta camada e as camadas inferiores. Essa por sua vez é determinada e construída pelo fabricante do equipamento, as demais continuam seguindo o padrão Zigbee.

A Figura 30 ilustra a divisão da camada APL em suas 3 subcamadas.

Figura 30 – Camada de aplicação e sua divisão.



Fonte: (OLIVEIRA, 2015).

4 Plataforma de sistemas embarcados

Os sistemas embarcados se caracterizam por serem mais limitados, tanto em *software*, quanto em *hardware*, em relação a um sistema computacional de propósito geral como computadores pessoais e *Smartphones*. Do ponto de vista do *hardware*, significa dizer que possuem menor capacidade de processamento, consumo de energia e memória, pois são projetados com o propósito específico. Quanto ao *software*, refere-se à ter um número de aplicações reduzidas, um sistema operacional reduzido ou nem possuir um (NOERGAARD, 2012).

Os projetos de sistemas embarcados são criados para as mais diversas finalidades, sendo um nicho com muitos aspectos a serem explorados. A presença de sistemas embarcados tem crescido muito nos últimos anos, tanto no comércio, indústrias e mesmo em residências. Os sistemas embarcados estão presentes no nosso cotidiano e a previsão é que aumentarão em grande escala nos próximos anos.

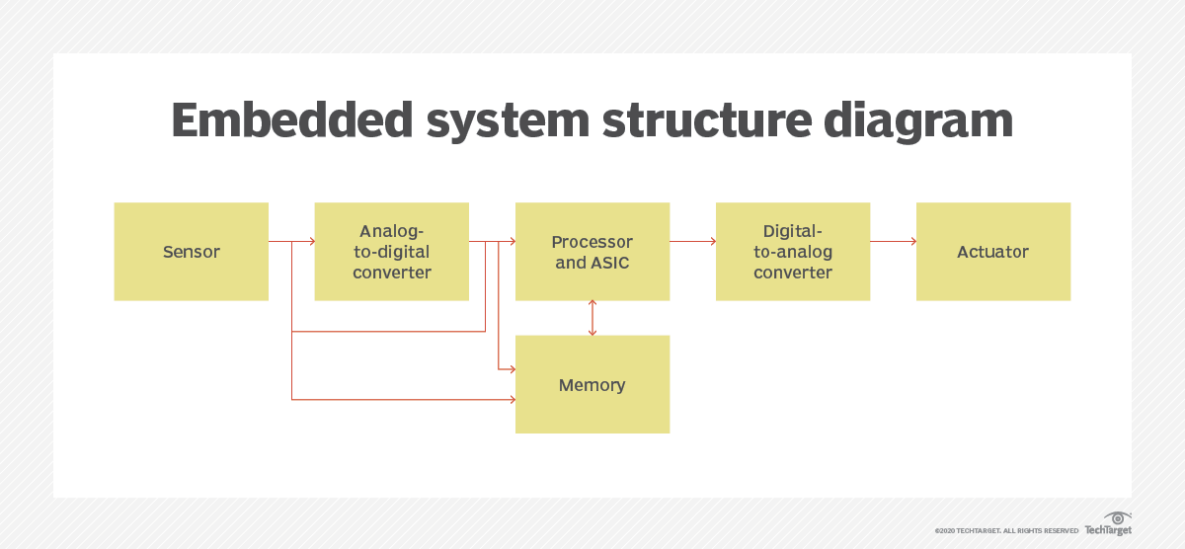
Os exemplos de aplicações dos sistemas embarcados, comumente remetem aos sistemas computacionais específicos, como por exemplo: uma calculadora, um roteador para internet ou um monitor cardíaco. Contudo, os sistemas embarcados atravessem um momento de profunda mudança.

A Figura 31 mostra um diagrama de estrutura de um sistema embarcado. Nela pode-se ver que as condições mínimas para termos um sistema embarcado é:

- **Conversor Analógico-Digital:** Esse conversor serve para realizar a discretização das variáveis de interesse do processo que virão pelos sensores e/ou transdutores. Essa discretização é necessária para que o processador possa ler e interpretar as variáveis de entrada do processo, transformando assim da natureza analógica das variáveis em digital.
- **Memória:** Será utilizada pelo processador para armazenar e ler variáveis e onde ficara armazenado o programa (sequências de instruções que o processador deverá executar).
- **Processador:** É o conjunto de *hardware* que irá realizar as instruções do programa de acordo com as variáveis de entrada
- **Conversor Digital-Analógico:** Conversores que tem a função de converter as grandezas digital de saída do processador para grandezas analógicas.

Hoje em dia com o avanço das tecnologias, é possível encontrar vários equipamentos pequenos que já possuem todos esses itens, como por exemplo, os microcontroladores,

Figura 31 – Exemplo de estrutura de um sistema embarcado.



Fonte: (LUTKEVICH, 2020).

que são hardwares que possuem dentro de si o processador (ou processadores), memórias ROM (*Read Only Memory*) e RAM (*Random Acess Memory*) e alguns periféricos como por exemplo, conversores AD e DA, temporizadores, *hardwares* de alguns padrões de comunicação, interrupção entre outros.

Existem os mais diversos fabricantes de microcontroladores como MCC, Atmel, Microchip, Motorola, Intel, STMicroeletronics, Texas Instruments entre outras, onde as características dos microprocessadores podem varias significativamente, como tamanho de memória, portas de entrada e saída, e a quantidade de *bits* que trabalha o processador.

Uma das plataformas que vem ganhando destaque e crescimento no mercado é o Arduino, contando com vários modelos de sistemas embarcados que atendem a um grande número de aplicações, além de várias bibliotecas disponibilizadas que facilitam a utilização dos mesmos. A seguir será apresentada uma introdução sobre essa plataforma que será utilizada na montagem do sistema de comunicação proposto

4.1 Arduino

O Arduino foi criado em 2005 por um grupo de 5 pesquisadores, Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino e David Mellis. O objetivo era elaborar um dispositivo que fosse ao mesmo tempo barato, funcional e de fácil programação para que dessa forma, seja acessível a estudantes e projetistas amadores. Além disso, foi adotado o conceito de *hardware* livre, o que significa que qualquer um pode montar, modificar, melhorar e personalizar o Arduino, partindo do mesmo *hardware* básico.

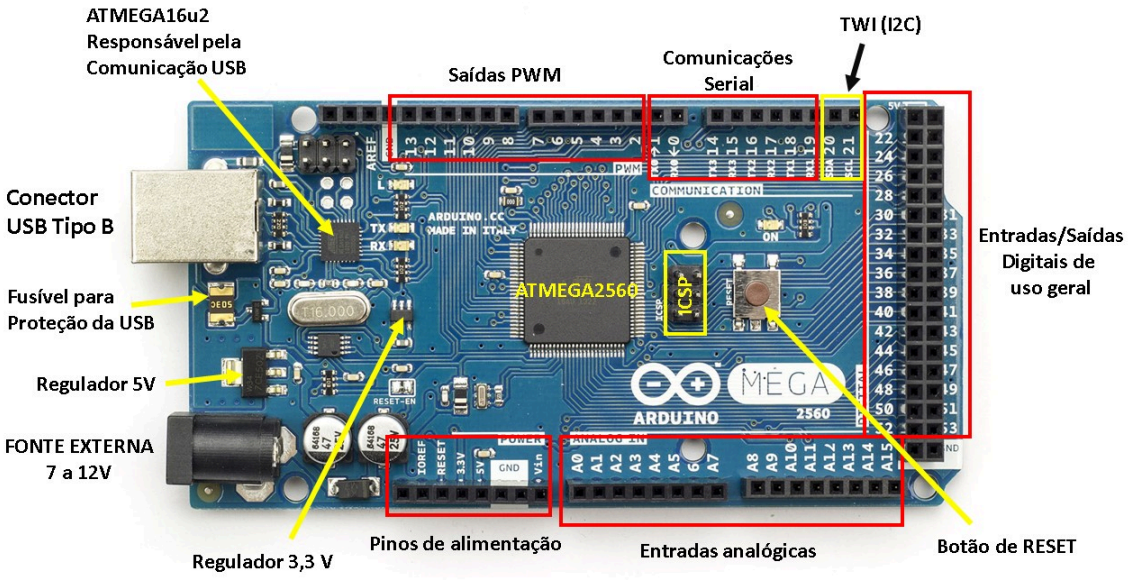
Assim, foi criada uma placa composta por um microcontrolador Atmel, circuitos

de entrada/saída que podem ser facilmente conectada a um computador e programada via IDE (*Integrated Development Environment*) utilizando uma linguagem baseada em C/C++, sem a necessidade de equipamentos extras além de um cabo USB (*Universal Serial Bus*). Depois de programado, basta alimentá-lo e ele irá realizar as instruções de acordo com o programa nele salvo.

4.1.1 Arduino Mega 2560

A plataforma Arduino Mega 2560 possui pequenas dimensões (10,16 cm X 5,08 cm) podendo ser instalado em vários locais. A Figura 32 apresenta a foto da placa do Arduino Mega, mostrando os recursos disponíveis na placa.

Figura 32 – Placa Arduino Mega 2560.



Fonte: (SOUZA, 2014).

A placa Arduino Mega 2560 é uma placa da plataforma Arduino que possui recursos bem interessantes para prototipagem e projetos mais elaborados. Baseada no microcontrolador ATmega2560, possui 54 pinos de entradas e saídas digitais onde 15 destas podem ser utilizados como saídas PWM (*Pulse Width Modulation*) e possui 16 entradas analógicas.

Possui também alguns hardwares embarcados dentro da plataforma ou dentro do microcontrolador. Ele possui 4 portas de comunicação serial (UART - *Universal Asynchronous Receiver/Transmitter*) o que permite que ele possa se comunicar diretamente com outros microcontroladores ou outros dispositivos que possuam UART. Possui um hardware para converter a comunicação serial para comunicação USB, permitindo comunicação direta com um computador, notebook ou dispositivos que tenham comunicação USB.

Ainda falando em comunicação, ele possui uma porta para comunicação SPI (*Serial Peripheral Interface*) e uma porta para comunicação I2C (*Inter-Integrated Circuit*), que irá permitir ingressar em uma rede com esses protocolos de comunicação sem a necessidade de hardwares adicionais, o que irá facilitar o seu uso.

Possui também 6 *Timers/Counter* que podem ser utilizados para a contagem de tempo ou até mesmo contagem de eventos externos. Esses *Timers/Counter* ainda podem gerar interrupções para desviar o código para execução de outras tarefas e voltar após a execução delas.

Além da quantidade de pinos maior, ela conta também com maior quantidade de memória que o Arduino UNO, sendo uma ótima opção para projetos que necessitem de muitos pinos de entradas e saídas além de memória de programa com maior capacidade. A Figura 33 apresenta as principais características dessa placa

Figura 33 – Características do Arduino Mega 2560.

Micro controlador	Atmega 2560
Tensão de funcionamento	5V
Tensão de entrada (recomendado)	7 a 12V
Tensão de entrada (máxima)	6 a 20V
Pinos de entrada e saída digital	54 (dos quais 14 podem ser saídas PWM)
Pinos de entradas analógicas	16
Valor máximo de corrente fornecida por pino	40mA
Valor de corrente para pino 3,3V	50mA
Memória flash	256KB
SRAM	8KB
EEPROM	4KB
Velocidade de clock	16MHz

Fonte: (SOUZA, 2014).

Mais informações sobre arquitetura, componentes da plataforma embarcada, microcontrolador e *Timers* podem ser consultadas no anexo C. Ainda no mesmo anexo nas seções C.4 e C.5 é apresentado sobre o funcionamento, arquitetura e seleção dos parâmetros do protocolo de comunicação SPI e I2C respectivamente para a plataforma do Arduino Mega 2560.

4.1.2 Plataforma de desenvolvimento

As operações realizadas pelo Arduino se resumem em essência a sinais elétricos. A lógica por trás dos impulsos e sinais transmitidos e interpretados pela plataforma dependem de programas inseridos em seu microcontrolador.

De acordo com (MARGOLIS, 2011), comandos e programas são escritos no computador utilizando um ambiente de desenvolvimento integrado (IDE), que permite escrever, alterar códigos e convertê-los em instruções que o Arduino consiga compreender e executar,

além de permitir a sua gravação no microcontrolador. Em outras palavras, tudo o que será realizado pelo Arduino, seja através do próprio microcontrolador ou de periféricos a ele conectados, deverá ser programado em scripts inseridos por intermédio da IDE, onde são definidas as orientações necessárias para cada execução. A interface da IDE do Arduino é intuitiva. No entanto, como depende da programação de comandos e execuções a mesma exige do usuário certos conhecimentos de algoritmos e linguagem de programação, em geral, C ou $C++$.

A Figura 34 apresenta a interface da plataforma de desenvolvimento para *Arduino*. Basicamente o código a ser escrito, resume-se em duas funções, sendo a primeira "setup()"que será executada uma única vez quando o Arduino for energizado, e após a execução dessa função, ele irá executar em "*loop* infinito"a função "loop()". No exemplo da Figura 34, o código irá fazer o LED da placa ligado ao pino D13 piscar a cada 1 segundo, ou seja, ele irá permanecerá desligado por 1 segundo, e ficará ligado por 1 segundo. Assim como toda linguagem de programação, o compilador possui comandos próprios, palavras reservadas, constantes e etc.

Para transferir o programa criado, primeiro deve indicar a plataforma, qual modelo de Arduino será utilizado e qual porta de comunicação USB (porta COM) está conectada o dispositivo. No exemplo pode-se verificar no canto direito inferior, que foi configurado para um Arduino Mega ligado na porta COM3. Após a configuração, deve-se compilar o código e se o programa não apresentar erros, poderá ser transferido para o Arduino. Caso haja algum problema na compilação ou na transferência do código, a plataforma irá apresentar um *log* sobre o ocorrido.

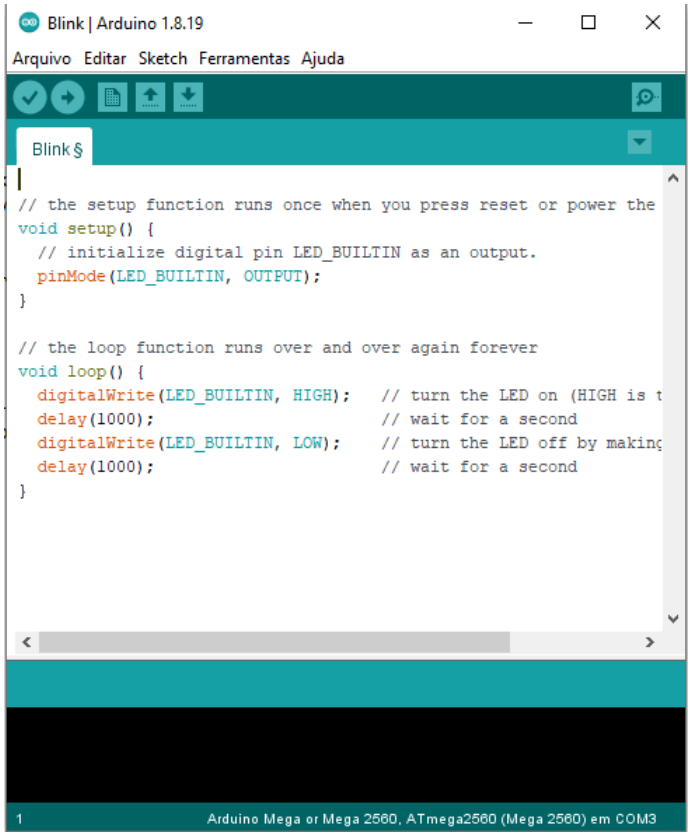
A IDE pode ser obtida no site (https://www.arduino.cc/en/software) do desenvolvedor .

4.2 XBee

XBee é a marca de uma família popular de módulos de conectividade sem fio da empresa DIGI. Os primeiros módulos XBee foram introduzidos sob a marca MaxStream em 2005 e foram baseados no padrão IEE 802.15.4 de 2003, projetado para comunicação ponto a ponto e estrela. Desde a introdução inicial, a família XBee cresceu e criou uma gama completa de módulos sem fio, *gateways*, adaptadores e *softwares*. Segundo o fabricante, os rádios XBee podem ser usados com o número mínimo de conexões, sendo alimentação (VCC e GND) e entrada e saída de dados (UART). Além disso, a maioria dos dispositivos da família XBee possui algum outro controle de fluxo, entradas e saídas digitais, conversor analógico-digital (A/D) e linhas indicadoras integradas.

O XBee é uma plataforma composta por um microcontrolador e um *transceiver* que aplica o padrão ZigBee de comunicação sem fio. A empresa DIGI desenvolveu vários

Figura 34 – Plataforma de desenvolvimento do Arduino.



Fonte: Autor.

modelos de XBee que podem comunicar em várias frequências e em outros padrões como *Bluetooth* por exemplo, porém nesse trabalho será apresentado apenas sobre o modelo XBee-PRO S2. A Figura 35 apresenta as principais características da plataforma XBee-PRO S2.

Os módulos RF XBee, fazem interface com um dispositivo por meio de uma porta serial assíncrona. Através dessa porta, o módulo pode se comunicar com qualquer UART compatível com os níveis de tensão. Os dispositivos que possuem interface UART podem se conectar diretamente aos pinos do módulo RF. A Figura 36 apresenta a conexão entre os dispositivos e o fluxo de dados.

Os dados entram no módulo UART através do pino DIN (pino 3 do XBee) como sinal serial assíncrono. O sinal deve ficar inativo em nível alto (VCC) quando nenhum dado está sendo transmitido. Cada *byte* de dados consiste em um *bit* de *Start*, 8 *bits* de dados (iniciando pelo LSB) e um *bit* de *Stop*. A Figura 37 exemplifica esse padrão.

A comunicação serial depende das duas UARTs (microcontrolador e módulo RF) estarem configurados de forma compatível com os parâmetros de *baud rate, parity, start bits, stop bits* e *data bits*.

Os pinos $\overline{RTS}$ e $\overline{CTS}$ podem ser usados para prover um controle de fluxo dos dados da UART. O pino de controle $\overline{CTS}$, permite um sinal de indicação para o dispositivo

Figura 35 – Principais características do XBee-PRO S2.

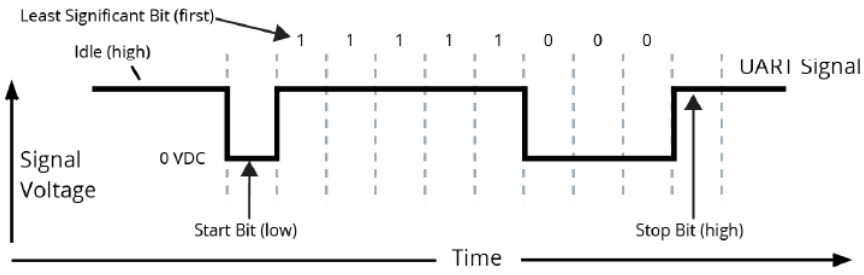
Specification	XBee-PRO (S2)
Performance	
Indoor/urban range	Up to 300 ft. (90 m), up to 200 ft (60 m) international variant
Outdoor RF line-of-sight range	Up to 2 miles (3200 m), up to 5000 ft (1500 m) international variant
Transmit power output	50 mW (+17 dBm) 10 mW (+10 dBm) for International variant
RF data rate	250,000 b/s
Data throughput	up to 35000 b/s (see Transmission, addressing, and routing)
Serial interface data rate (software selectable)	1200 b/s - 1 Mb/s (non-standard baud rates also supported)
Receiver sensitivity	-102 dBm
Power Requirements	
Supply voltage	3.0 - 3.4 V
Operating current (transmit, max output power)	295 mA (@3.3 V) 170 mA (@3.3 V) international variant
Operating current (receive))	45 mA (@3.3 V)
Idle current (receiver off)	15mA
Power-down current	3.5 µA typical @ 25°C
General	
Operating frequency band	ISM 2.4 GHz
Dimensions	0.960 x 1.297 (2.438 cm x 3.294 cm)
Operating temperature	-40 to 85° C (industrial)
Antenna options	Integrated whip antenna, embedded PCB antenna, RPSMA or U.FL connector
I/O interface	3.3 V CMOS UART (not 5 V tolerant), DIO, ADC
Networking and Security	
Supported network topologies	Point-to-point, point-to-multipoint, peer-to-peer, and mesh
Number of channels	14 Direct sequence channels
Channels	11 to 24
Addressing options	PAN ID and addresses, cluster IDs and endpoints (optional)
Agency Approvals	
United States (FCC Part 15.247)	FCC ID: MCQ-XBEEPRO2
Industry Canada (IC)	IC: 1846A-XBEEPRO2
Europe (CE)	Yes
Australia	RCM/R-NZ
Japan	R201WW08215142 (international variant) Wire, chip, RPSMA, and U.FL versions are certified for Japan. PCB antenna version is not.
Brazil	ANATEL: 2256-14-1209
RoHS	Compliant

Fonte: (DIGI, 2018).

Figura 36 – Fluxo de dados entre um dispositivo e XBee.

Fonte: (DIGI, 2018).

Figura 37 – Exemplo do envio de dados do pacote 0x1F (31 em decimal).



Fonte: (DIGI, 2018).

parar de enviar dados ao módulo RF. Se habilitado, quando o *buffer* da serial estiver a 17 *bytes* de estar cheio, o módulo RF desativa o pino $\overline{CTS}$ (colocando em nível alto). Ele irá ativar o pino (colocando em nível baixo), quando o *buffer* de recepção tiver pelo menos 34 *bytes* de espaço. Já o pino de controle $\overline{RTS}$ permite uma indicação do dispositivo para o módulo não enviar os dados no *buffer* de transmissão. Se habilitado, os dados não serão enviados até que este pino esteja em nível baixo. Caso o módulo continue recebendo dados para enviar ao dispositivo, porém o pino $\overline{RTS}$ continue em nível lógico alto, os pacotes que chegam após encher o *buffer*, serão descartados.

4.2.1 Modo de operação

O módulo XBee pode operar em dois modos, sendo o modo Transparente e o modo API (*Application Programming Interface*). Nesse trabalho será apresentado o modo API que foi utilizado no desenvolvimento do sistema proposto que será apresentado no capítulo 5. A operação API requer que a comunicação com o módulo seja feito por meio de uma interface estruturada (os dados são comunicados em *frames* em uma ordem definida). O API especifica como comandos, respostas de comandos e mensagem de status do módulo são enviadas e recebidas pelo módulo usando a UART.

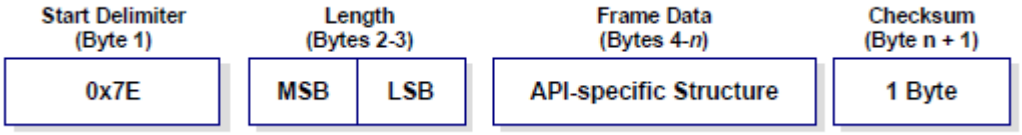
Dois modos de API são suportados e ambos podem ser ativados usando o comando AP (API *Enable*). A seguir são apresentados os valores de parâmetro AP para configurar o módulo para operar um modo específico:

- AP = 1: Operação API
- AP = 2: Operação API (com caracteres de *escape*)

4.2.1.1 Operação API

Quando esse modo de API está habilitado (AP = 1), a estrutura do *frame* de dados na UART é definido conforme a Figura 38.

Figura 38 – Sequência de *frame* no modo AP = 1.



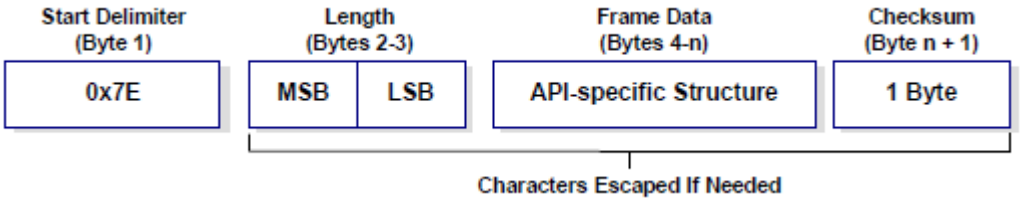
Fonte: (DIGI, 2018).

Quaisquer dados recebidos antes do delimitador de início são descartados silenciosamente. Se o *frame* não for recebido corretamente, ou se o *checksum* falhar, o módulo responderá com um *frame* de status indicando a natureza da falha.

4.2.1.2 Operação API *escaped*

Quando esse modo de API está habilitado (AP = 2), a estrutura do *frame* de dados da UART é definida conforme a Figura 39

Figura 39 – Sequência de *frame* no modo AP = 2.



Fonte: (DIGI, 2018).

Ao enviar ou receber um *frame* de dados, valores de dados específicos devem ser *escaped* (sinalizados) para que que não interfiram no sequenciamento do *frame* de dados. Para escapar de uma interferência de *byte* de dados, insira 0x7D e siga-o com o *byte* que resulta da operação XOR entre o byte a ser escapado e 0x20. Note que, se não houver *escape*, 0x11 e 0x13 serão enviados com estão.

Bytes de dados que precisam ser escapados:

- 0x7E - Delimitador de Quadro
- 0x7D - Fuga

- 0x11 - XON
- 0x13 - XOFF

A seguir é exemplificado essa ação. Abaixo é apresentado o *frame* de dados da UART bruto (antes de escapar bytes de interferência):

0x7E 0x00 0x02 0x23 0x11 0xCB

O byte 0X11 precisa ser escapado, o que resulta no seguinte *frame*:

0x7E 0x00 0x02 0x23 0x7D 0x31 0xCB

No exemplo acima, o comprimento dos dados brutos (excluindo o byte *checksum*) é 0x0002 e a soma de verificação de dados sem *escape* (excluindo delimitador de *frame* e comprimento) são calculados como:

0xFF - (0x23 + 0x11) = (0xFF - 0x34) = 0xCB

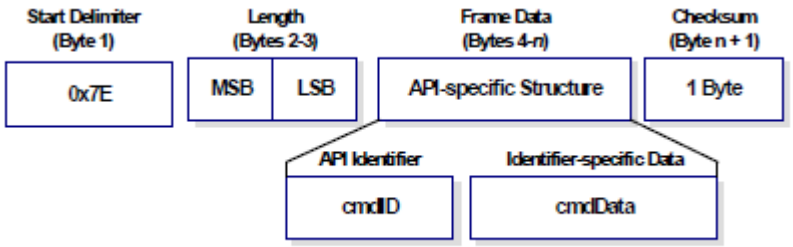
4.2.1.3 Comprimento

O campo de comprimento (*Length*) tem um valor de dois *byte* que especificam o número de *bytes* que estarão contidos no campo *frame data*.

4.2.1.4 *Frame data*

A estrutura do *Frame data* é apresentada na Figura 40. O *frame* cmdID (API-*identifier*) indica quais mensagens de API estão contidas no *frame cmdData* (dados específicos do identificador). Observe que os valores de vários *bytes* são enviados como *big endian*. A Figura 41 apresenta os tipos de *frame* de API que os módulos suportam.

Figura 40 – Estrutura do *data frame*.



Fonte: (DIGI, 2018).

Figura 41 – Tipos de *frame*.

API Frame Names	API ID
AT Command	0x08
AT Command - Queue Parameter Value	0x09
Zigbee Transmit Request	0x10
Explicit Addressing Zigbee Command Frame	0x11
Remote Command Request	0x17
Create Source Route	0x21
AT Command Response	0x88
Modem Status	0x8A
Zigbee Transmit Status	0x8B
Zigbee Receive Packet (AO=0)	0x90
Zigbee Explicit Rx Indicator (AO=1)	0x91
Zigbee IO Data Sample Rx Indicator	0x92
XBee Sensor Read Indicator (AO=0)	0x94
Node Identification Indicator (AO=0)	0x95
Remote Command Response	0x97
Over-the-Air Firmware Update Status	0xA0
Route Record Indicator	0xA1
Many-to-One Route Request Indicator	0xA3

Fonte: (DIGI, 2018).

4.2.1.5 Soma verificadora

Para testar a integridade dos dados, uma soma de verificação (*checksum*) é calculada e verificada em dados sem *escape*.

- **Calcular:** Não incluindo o delimitador de *frame* e o campo *length*, adicione todos os *bytes* mantendo apenas os 8 bits mais baixo do resultado e subtraia de 0xFF.
- **Verificar:** Adicione todos os *bytes* (inclua *checksum*, mas não o delimitador e o comprimento). Se a soma de verificação estiver correta, a soma será igual a 0xFF.

A seguir é apresentado um exemplo real de um *frame* completo com comando AT para configurar o XBee para permitir ingressos na rede (definindo o parâmetro NJ como 0xFF). O *frame* completo ficaria assim:

0x7E 0x00 0x05 0x08 0x01 0x4E 0x4A 0xFF 0x5F

Onde:

0x7E = Delimitador de início do *frame*

0x00 = MSB do campo *length*

0x05 = LSB do campo *length*

0x08 = Tipo de *frame* (no caso é do tipo comando AT)

0x01 = *Frame* ID

0x4E4A = Comando AT ('NJ')

0xFF = Valor a ser colocado no parâmetro

0x5F = *Checksum*

4.2.1.6 Fluxo de dados UART no modo API

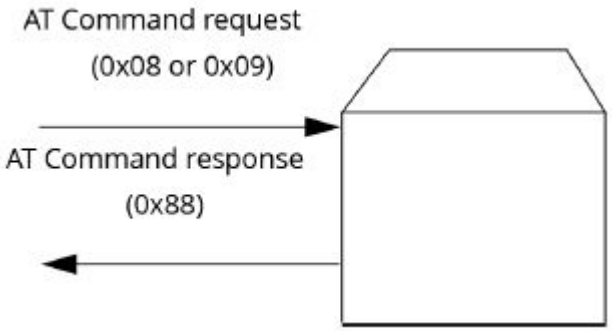
4.2.1.6.1 Comando AT

A Figura 42 mostra a troca de *frames* da API que ocorre na UART ao enviar uma solicitação de comando AT para ler ou definir um parâmetro de módulo. A resposta pode ser desabilitada definindo o *Frame* ID como zero na solicitação.

4.2.1.6.2 Transmissão e recepção de dados RF

A Figura 43 mostra os fluxo da API que ocorrem na UART ao enviar dados de RF para outro módulo. O *frame* de status de transmissão é sempre enviado no final de uma

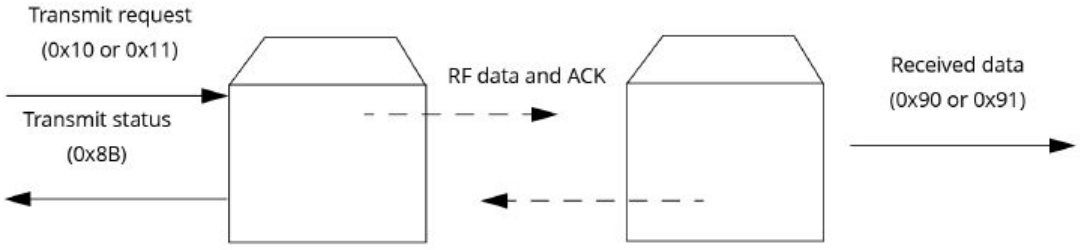
Figura 42 – Fluxo da UART no comando AT.



Fonte: (DIGI, 2018).

transmissão de dados, a menos que o *frame* ID seja definido como zero na solicitação da transmissão. Se o pacote não puder ser entregue ao destino, o *frame* de status indicará a causa da falha. O *frame* de dados recebido (0x90 ou 0x91) é definido pelo comando AP.

Figura 43 – Fluxo da UART na transmissão e recepção de dados via radio frequência.

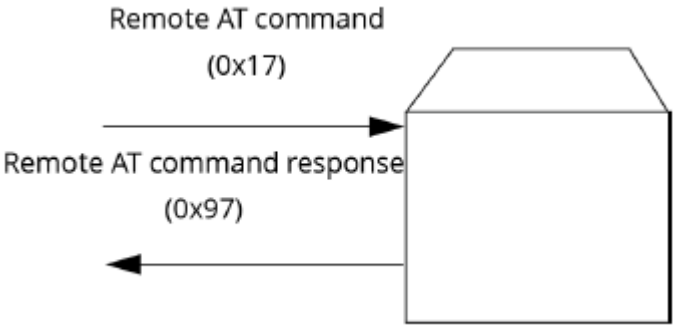


Fonte: (DIGI, 2018).

4.2.1.6.3 Comandos AT remotos

A Figura 44 mostra o fluxo que ocorrem na UART ao enviar um comando AT remoto para um outro módulo. Um *frame* de resposta de comando remoto não é enviado pela UART se o dispositivo não receber o comando remoto.

Figura 44 – Fluxo da UART no comando AT remoto.

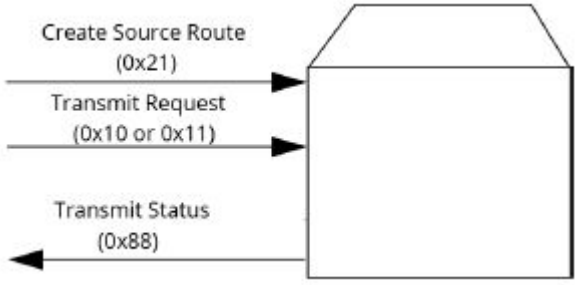


Fonte: (DIGI, 2018).

4.2.1.6.4 Roteamento

A Figura 45 mostra o fluxo que ocorrem na UART ao enviar uma transmissão de criação de rota.

Figura 45 – Fluxo da UART na criação de rotas.



Fonte: (DIGI, 2018).

Mais informações sobre os principais tipos, comandos, parâmetros e exemplos de *frames* são apresentadas no anexo D.

4.2.1.7 XCTU

O XCTU é um aplicativo multiplataforma gratuito e projetado para permitir que os desenvolvedores interajam com os módulos RF da Digi por meio de uma interface gráfica simples de usar. Inclui novas ferramentas que facilitam a instalação, configuração e teste dos módulos *XBee*.

O XCTU inclui todas as ferramentas que um desenvolvedor precisa para começar a trabalhar rapidamente com o XBee. Recursos exclusivos como visualização de rede gráfica, que representa graficamente a rede XBee juntamente com a intensidade do sinal de cada conexão, e o construtor de *frames* API para XBee sendo usados no modo API, combinam-se para tornar o desenvolvimento na plataforma XBee mais fácil do que nunca. Outros destaques do XCTU incluem os seguintes recursos:

- Gerenciamento e configuração de vários dispositivos RF, mesmo os dispositivos conectados remotamente (*over-the-air*)
- O processo de atualização do *firmware* restaura perfeitamente as configurações do seu módulo, manipulando automaticamente as alterações de modo e taxa de transmissão.
- Dois consoles API e AT específicos foram projetados do zero para se comunicar com seus dispositivos de rádio.
- É possível salvar suas sessões de console e carregá-las em um PC (*Personal Computer*) diferente executando o XCTU.

- O XCTU inclui um conjunto de ferramentas incorporadas que podem ser executadas sem ter nenhum módulo de RF (Rádio Frequência) conectado:
  - **Interpretador de *frames*:** Decodifique um *frame* de API e veja seus valores de *frames* específicos.
  - **Recuperação:** Recupere os módulos de rádio com *firmware* danificado ou em modo de programação.
  - **Carregar sessão de console:** Carrega uma sessão de console salva em qualquer PC executando XCTU.
  - **Teste de alcance:** Faça um teste de alcance entre 2 módulos de rádio da mesma rede.
  - **Explorador de *firmware*:** Navegue pela biblioteca de *firmware* do XCTU
- Permite que seja atualizado automaticamente o próprio aplicativo e biblioteca de *firmware* do rádio sem a necessidade de arquivos extras.
- O XCTU contém documentação completa e abrangente, que pode ser acessada a qualquer momento.

No próximo capítulo será apresentado como foi construído o sistema proposto, partindo das informações apresentada nesse e nos capítulos anteriores.

5 Protótipos do sistema embarcado de comunicação baseado no padrão ZigBee

Nesse capítulo, serão apresentados os detalhes do desenvolvimento dos protótipos para implementação do sistema de comunicação proposto e como foram realizadas as etapas de seleção, montagem e configuração.

Esses protótipos foram desenvolvido para validar o sistema de comunicação proposto que tem por objetivo a substituição de condutores elétricos por um sistema embarcado de comunicação sem fio baseado no padrão ZigBee nos equipamentos de sinalização em sistemas ferroviários. Uma vez que o ambiente ferroviário é dinâmico com circulação de material rodante de diversos tamanhos e que a distância entre os próprios equipamentos podem variar consideravelmente, se faz necessário a construção e a realização de testes para determinar a validade do sistema proposto, além de determinar se os dispositivos conseguem se comunicar mesmo quando houver material rodante entre eles, ou seja, sem visada direta e com a interferência que podem sofrer de sinais que estão no ambiente na mesma faixa de frequência de operação.

5.1 Seleção

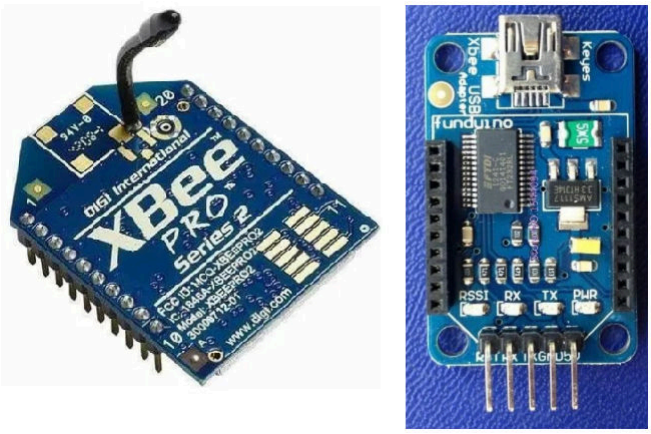
Nessa etapa, primeiramente foi realizado o levantamento de requisitos que o protótipo deveria atender. A seguir serão apresentados os requisitos levantados que devem ser atendidos pelo protótipo.

1. O dispositivo deve conseguir trabalhar em uma rede sem fio no padrão ZigBee;
2. O dispositivo deve ser capaz de montar e gerenciar a rede do padrão ZigBee na topologia *Mesh*;
3. O dispositivo será capaz de enviar e receber dados e executar tarefas de acordo com a informação recebida;
4. O dispositivo deve ser capaz de armazenar cada uma das informações recebidas e enviadas para controle e revisão caso seja necessário;
5. O dispositivo deve possuir um relógio interno que funcione mesmo quando o dispositivo estiver desenergizado, para geração de *logs* com horário de cada evento;
6. O dispositivo deve conseguir acionar cargas e conseguir ler indicações dos sistemas de sinalização ferroviária;

7. O dispositivo deve ser capaz de funcionar com uma alimentação de 5 VCC.

Após o levantamento de requisitos, o próximo passo foi selecionar o conjunto de dispositivos e circuitos que poderiam executar essas tarefas. Para atender os requisitos nº 1 e 2, foi escolhido a plataforma XBee-Pro S2, a qual foi apresentada no capítulo 4.2, pois essa plataforma atende os requisitos necessários. Para facilitar o manuseio e utilização do XBee, foi utilizada um *gadget* adaptador de USB para XBee, para facilitar a configuração e comunicação com o XBee, além de converter os níveis de tensão nos pinos, uma vez que o XBee é alimentado com 3,3 VCC. A Figura 46 ilustra o modelo de XBee e o adaptador utilizado.

Figura 46 – *XBee* (esquerda) e adaptador (direita) utilizado.



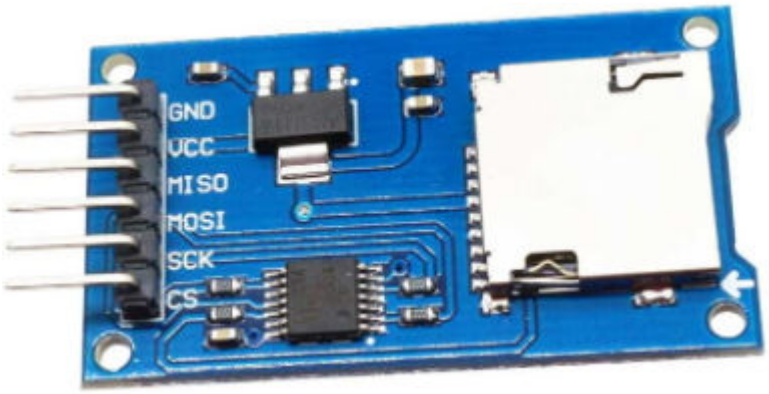
Fonte: Autor.

Após isso, foi realizado a seleção da plataforma que iria atender os requisitos nº 3, 6 e 7, além de conseguir operar com o XBee. O sistema escolhido, foi a plataforma Arduino Mega que foi apresentada no capítulo 4.1.1, que irá centralizar o controle dos demais dispositivos.

Para atender o requisito nº 4, foi escolhido um módulo leitor de cartão micro SD (*Secure Digital*), onde é possível inserir um cartão micro SD formatado em FAT32 e armazenar ou ler dados desse cartão. Esse módulo utiliza comunicação SPI, deve ser alimentado com 5 VCC e pode operar com tensões de 3,3 VCC ou 5 VCC, o que o torna versátil e compatível com a maioria do microcontroladores, incluindo nesse caso o ATmega 2560. A Figura 47 apresenta o módulo leitor de cartão micro SD. Nela é possível visualizar seus pinos para conexão de alimentação (VCC e GND) e os pinos de comunicação SPI (MISO, MOSI, SCK e CS), conforme apresentado no anexo C.4.

Para atender o requisito nº 5, foi escolhido o módulo RTC DS3231. O módulo RTC DS3231 é um relógio de tempo real de alta precisão, baixo consumo de energia e baixo custo, podendo ser alimentado com 3,3 VCC ou 5 VCC. Este módulo é capaz de trabalhar com informações como segundos, minutos, horas, dias, meses, anos e ainda é capaz de

Figura 47 – Módulo leitor de cartão micro SD.



Fonte: (CIA, 2020).

ajustar-se automaticamente em casos de meses com menos de 31 dias e em situações de ocorrência de anos bissextos. Além disso, o módulo pode operar tanto no formato 12 horas (AM/PM) como no formato 24 horas. Para comunicação com o módulo, o mesmo utiliza o protocolo de comunicação I2C, possui um compartimento para que seja inserida uma bateria externa, para manter o funcionamento mesmo quando não estiver sendo alimentado, possui também um sensor de temperatura embutido e dois alarmes programáveis. A Figura 48 apresenta o módulo, onde é possível verificar os pinos de alimentação (VCC e GND), além dos pinos de comunicação (SCL e SDA) conforme apresentado no anexo C.5. Possui também o pino 32 kHz (que gera uma saída de 32 kHz no pino) e o pino SQW (que pode ser usado como saída de interrupção ou para emitir uma onda quadrada de frequência desejada entre 4 possíveis frequências (1, 1024, 4096 e 8192 Hz)).

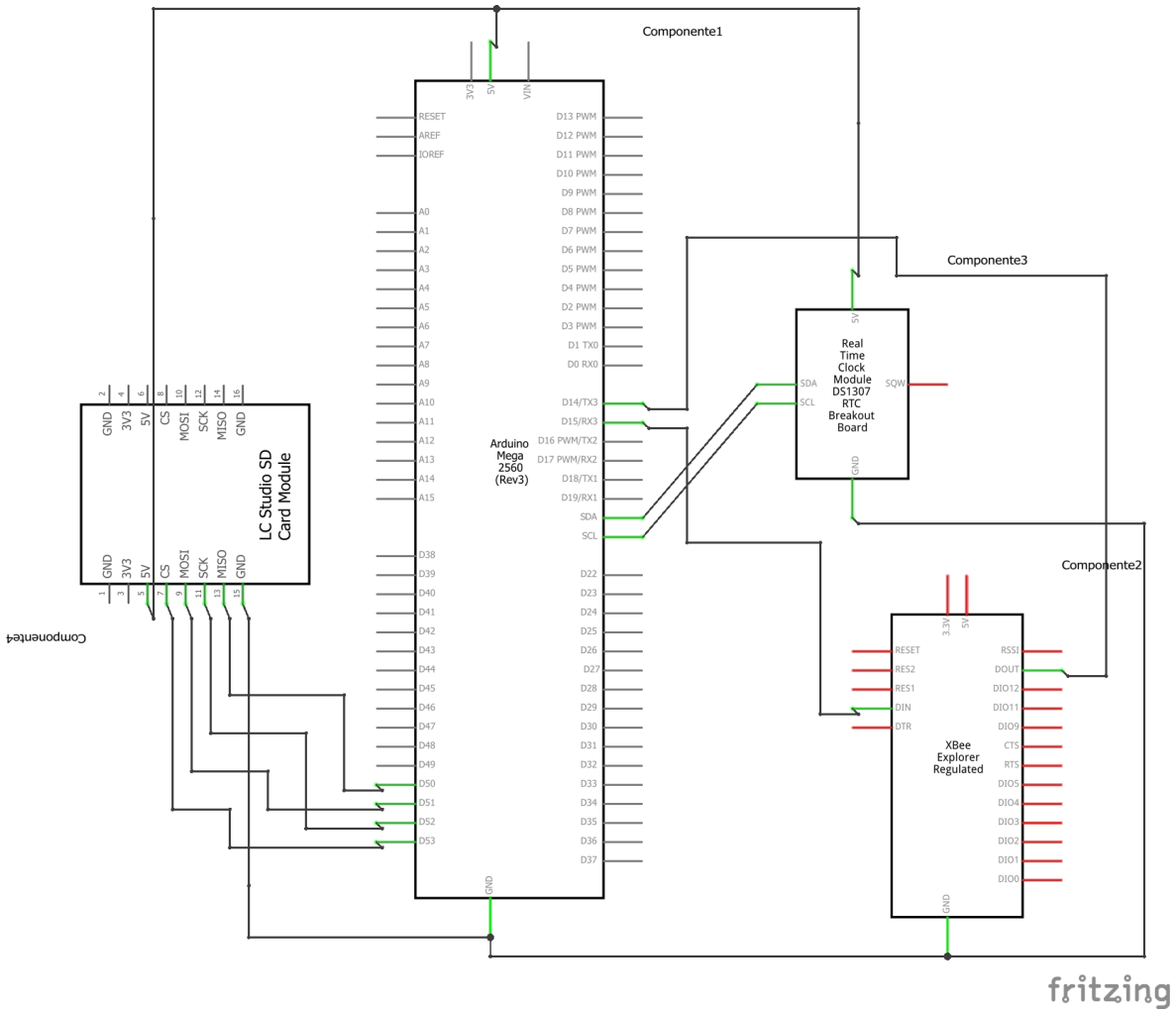
Figura 48 – Vista frontal e traseira do módulo RTC DS3231.

Fonte: (MADEIRA, 2017).

5.2 Diagrama de conexão

Após a seleção de todos os componentes, foi realizado um estudo e leitura dos *datasheets* para realizar a interconexão de todos os dispositivos. A Figura 49 apresenta a ligação dos componentes ao Arduino Mega.

Figura 49 – Esquema de conexões dos componentes ao Arduino Mega.



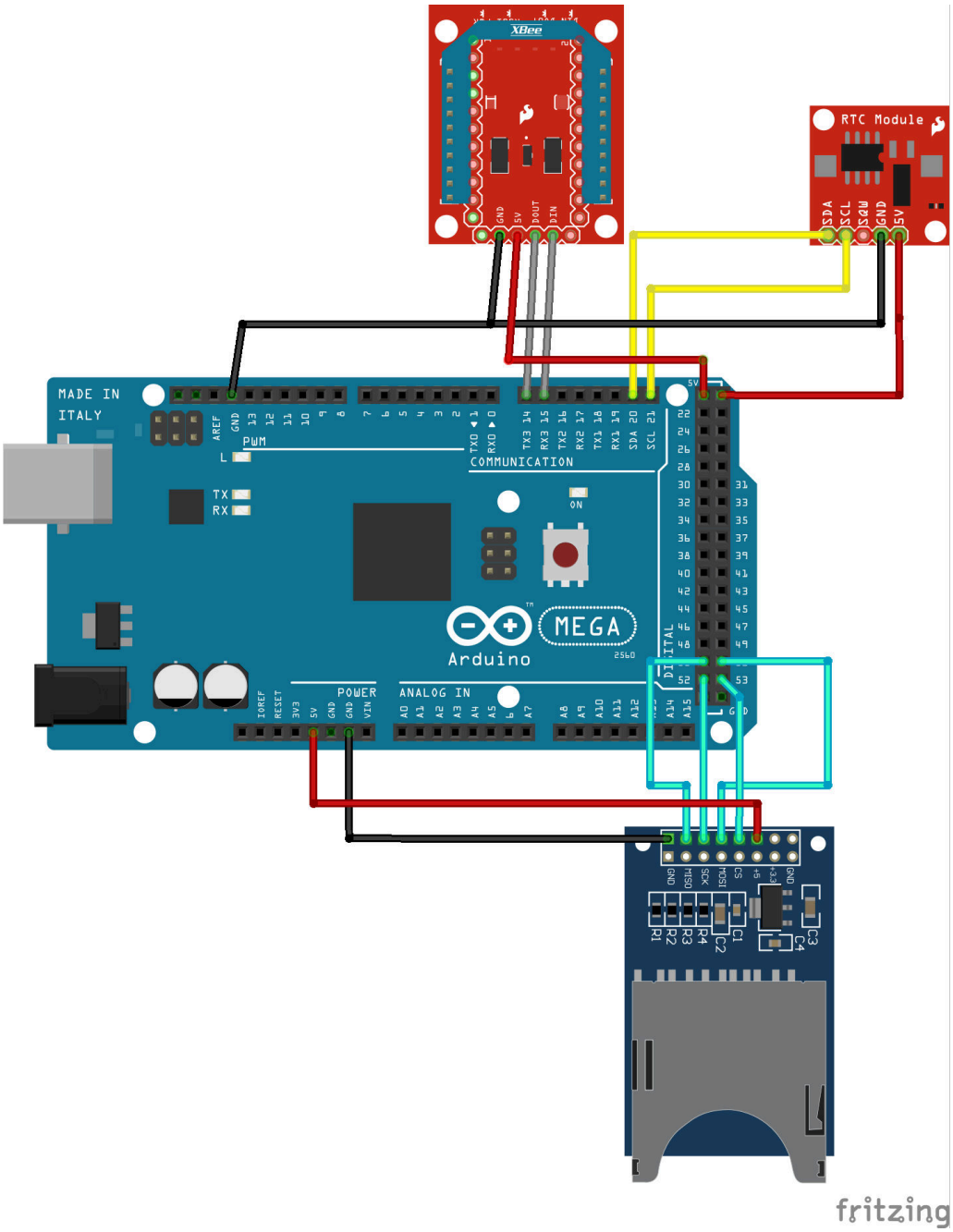
Fonte: Autor.

Para facilitar a interpretação da Figura 49, foi gerada no *software* Fritizing a Figura 50 que apresenta as ligações mais próximas do real de acordo com os formatos e disposição dos pinos dos módulos. Na Figura 50, pode-se visualizar as conexões ao GND (preto), VCC (vermelho), conexões Serial TX e RX (cinza), I2C (amarelo) e SPI (cyano).

5.3 Montagem

Para realizar a montagem, foi utilizado um pedaço de placa de acrílico como base de todos os componentes. Foram montados 3 protótipos para realização dos testes. Devido a falta de possuir 3 Arduino Mega, foi utilizado um Arduino Uno e um Arduino Nano na

Figura 50 – Ilustração das conexões dos módulos.

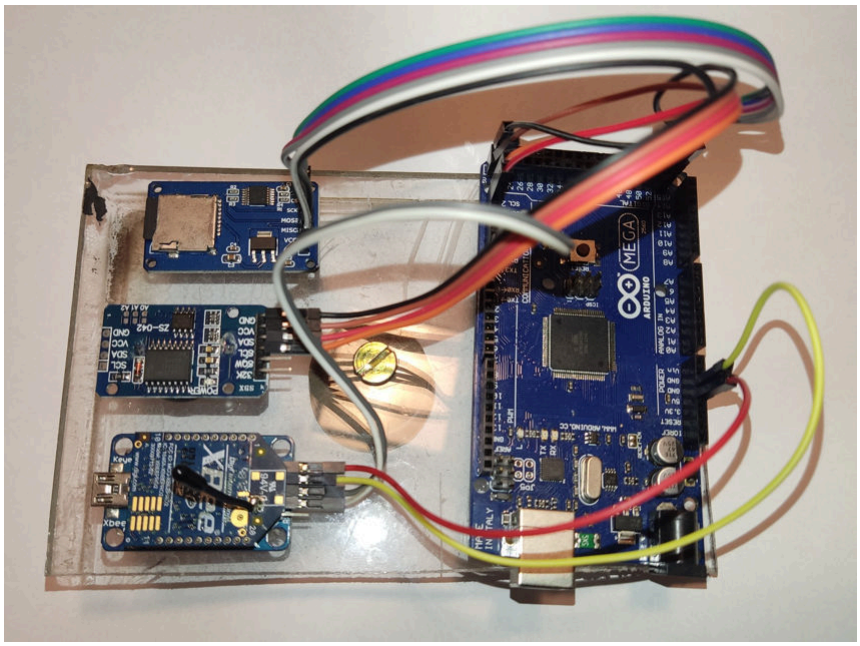


Fonte: Autor.

montagem do segundo e terceiro protótipo. Tanto o Arduino Uno quanto o Nano, possuem o mesmo microcontrolador (ATmega 328P). Esse microcontrolador possui arquitetura igual e todos os componentes de hardware apresentado no capítulo 4.1.1, sendo que a principal diferença para o ATmega2560 é na quantidade de *Timers* e na quantidade de memória *Flash* e *ROM*, além de possuírem bem menos pinos de entrada/saída.

A Figura 51 apresenta a montagem final do primeiro protótipo com o Arduino Mega. Abaixo da base de acrílico, foi parafusado uma tampa de tubo de 3/4", a qual será utilizada para acoplar o sistema ao suporte de fixação para realizar os testes.

Figura 51 – Montagem do protótipo A.



Fonte: Autor.

A Figura 52 apresenta a montagem final do segundo protótipo com o Arduino Uno. Assim como o anterior, foi parafusada uma tampa de tubo para acoplar ao suporte.

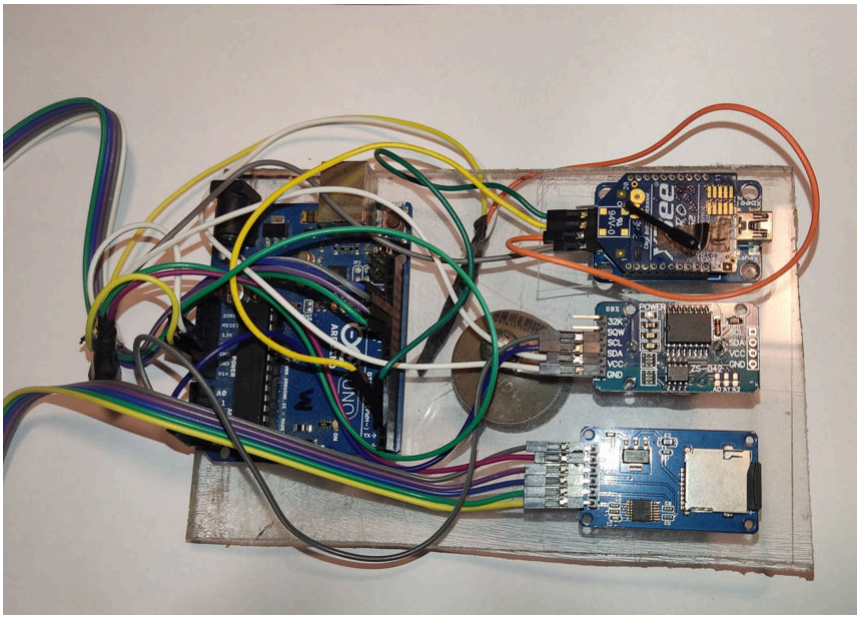
A Figura 53 apresenta a montagem final do terceiro protótipo com o Arduino nano. Diferente dos anteriores, nesse não foi fixado a tampa de tubo, pois não será necessário acoplar ao suporte para testes.

Como cada Arduino possui uma variação na posição dos pinos, a Figura 54 apresenta resumidamente as ligações entre os dispositivos e cada um dos pinos dos respectivos Arduinos.

5.4 Configuração

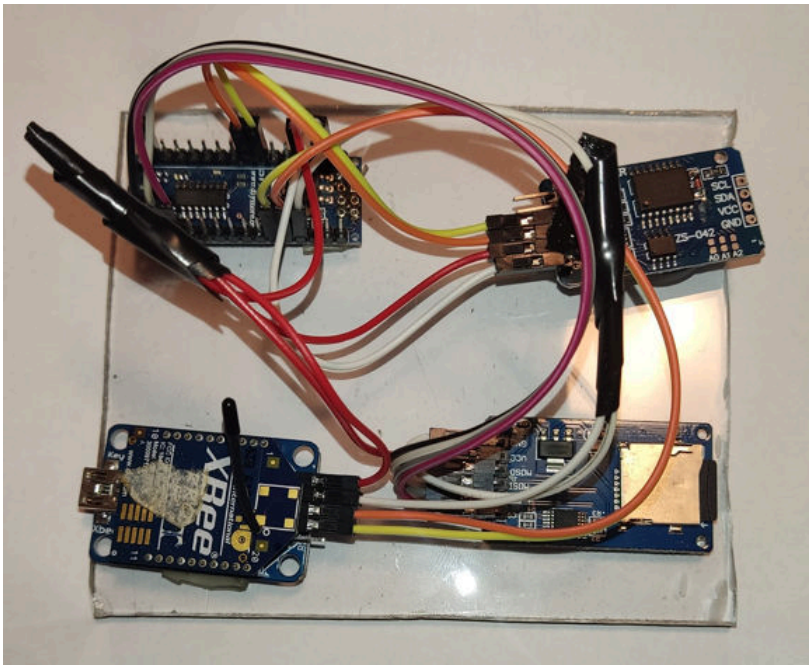
A seguir serão apresentadas as configurações que foram realizadas nos dispositivos para o funcionamento desejado. Os Arduinos (Mega, Uno e Nano), não necessitam de

Figura 52 – Montagem do protótipo B.



Fonte: Autor.

Figura 53 – Montagem do protótipo C.



Fonte: Autor.

Figura 54 – Resumo das conexões dos dispositivos.

RTC			
Pino	**MEGA**	**UNO**	**Nano**
SCL	21	A5	A5
SDA	20	A4	A4
VCC	5V	5V	5V
GND	GND	GND	GND
SD CARD			
CS/SS	53	10	D10
SCK	52	13	D13
MOSI	51	11	D11
MISO	50	12	D12
VCC	5V	5V	5V
GND	GND	GND	GND
XBEE			
5V	5V	5V	5V
GND	GND	GND	GND
TX	14	3	3
RX	15	2	2

Fonte: Autor.

configuração, a não ser a alteração do programa gravado na memória dele, que é feita pelo software apresentado no capítulo 4.1.2.

5.4.1 RTC DS3231

Para configurar o RTC, primeiro foi colocada uma bateria do modelo CR2032, para que após a desenergização ele não perdesse os parâmetros configurados. Para informar o horário ao módulo, foi utilizada a biblioteca DS3231 da IDE, com o exemplo "DS3231_set" em que é possível enviar através do console via comunicação serial, a hora e data para o Arduino no seguinte formato:

YYMMDDwHHMMSSx

onde as letras representam o seguinte:

- YY: Ano atual com 2 dígitos.
- MM: Mês atual.
- DD: Dia com 2 dígitos
- w: Caractere que indica a separação entre data e hora, não deve ser alterado.

- HH: Hora atual
- MM: Minuto atual
- SS: Segundo atual
- x: Caractere que indica a finalização da *string*, não deve ser alterado.

Foi então transferido esse programa para o Arduino e ligado o módulo RTC a ele e foi informado via console serial a hora e data atual, formato de 24h e desligado os alarmes.

5.4.2 Leitor de cartão micro SD

A única configuração necessária para o funcionamento do módulo, foi a utilização de cartão micro SD e a formatação dele no formato FAT32, tarefa essa que foi realizado com o auxílio de um computador.

5.4.3 XBee

Para configurar cada um dos *XBee*, foi necessários escolher primeiro a arquitetura da rede e o modo de operação deles. A arquitetura de rede escolhida foi a *Mesh*, e o modo de operação foi o API 2 (com *escaping*).

Após isso foi definido qual dispositivo seria o Coordenador da rede, e os demais seriam roteadores (Roteador A e Roteador B) da rede. Após essas escolhas, foram instalados os *firmware* neles com a ferramenta *Firmware Explores* através do *software* XCTU e então foram configurados os parâmetros necessários para utilização da rede. A Figura 55 apresenta os principais parâmetros que foram alterados e seus respectivos valores. Os demais parâmetros de cada *firmware*, ficaram com os valores padrões.

Figura 55 – Tabela com os principais parâmetros alterados no XBee.

Parâmetro	Dispositivo			Descrição do parâmetro
	Coordenador	Roteador A	Roteador B	
ID	1234	1234	1234	ID da rede PAN
SC	1	1	1	Canais que podem ser utilizado. Nesse caso utiliza apenas o canal 11.
SH	13A200	13A200	13A200	MSB do endereço do dispositivo de 64 bits (os 32 bits mais significativos)
SL	409BD04B	409E87A5	406B4547	LSB do endereço do dispositivo de 64 bits (os 32 bits menos significativos)
DH	0	0	0	MSB do endereço de destino. No modo API não é utilizado e deve ficar com o valor zero
DL	0	0	0	LSB do endereço de destino. No modo API não é utilizado e deve ficar com o valor zero
NI	Coordenador	Roteador A	Roteador B	Identificação do nó na rede.
NH	2	2	2	Quantidade máxima de saltos
BD	115200	9600	9600	Velocidade de comunicação serial (Baud Rate)
AP	2	2	2	Modo de operação API 2 (com escaping)
VR	21A7	23A7	23A7	Versão do firmware instalado no dispositivo

Fonte: Autor.

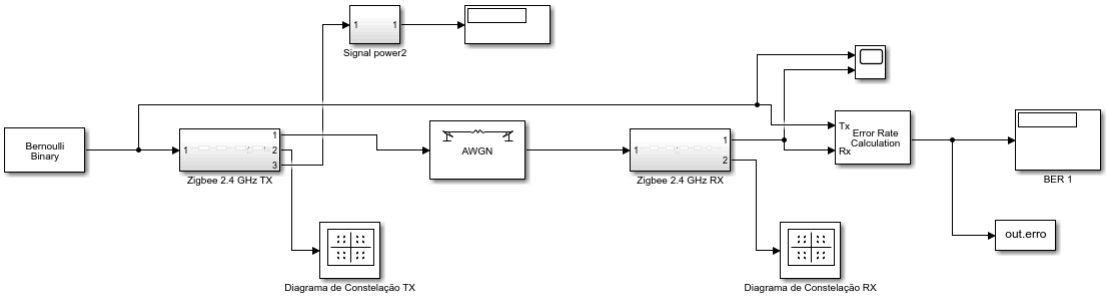
Após a finalização da montagem e configuração de cada um dos três protótipos, eles estão prontos para a realização dos testes que irão validar o sistema proposto para o controle e monitoramento de equipamentos ferroviários. Os testes e seus resultados serão apresentado no capítulo a seguir.

6 Análises do sistema embarcado de comunicação baseado no padrão ZigBee

6.1 Análise do sistema ZigBee por simulações

Um passo importante antes do desenvolvimento de um sistema é a análise de seu desempenho através de simulações computacionais. Para esse trabalho, foi desenvolvido uma simulação do padrão ZigBee de comunicação através do *software* Simulink. A Figura 56 apresenta a montagem do padrão ZigBee dentro da plataforma do Simulink.

Figura 56 – Simulação do padrão ZigBee no Simulink.



Fonte: Autor.

A montagem se divide em dois blocos principais, o “TX” e o “RX”. Existe um gerador de *bits* aleatórios, que alimenta o bloco TX. Bloco esse que reproduz o que foi apresentado na Figura 22. A Figura 57 apresenta o conteúdo desse bloco TX.

Figura 57 – Bloco TX.

Fonte: Autor.

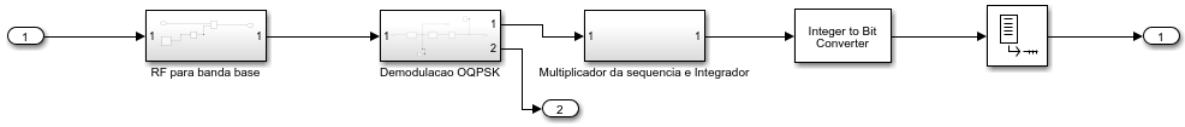
O bloco “Bits-Símbolo”, tem a função de juntar 4 *bits*, para formar um símbolo, uma vez formado esse símbolo, ele é convertido para inteiro e vai como entrada para o bloco “Simbolo - Chips”. Esse bloco por sua vez recebe o valor inteiro e através de um *mux*, seleciona a sequência de *chips* correspondente ao símbolo gerado.

Essa sequência de *chips* irá entrar no bloco “Modulação OQPSK”, onde será criado um *buffer*, para a cada dois *bits*. Esse *buffer* irá transformar uma sequência em linha de bits em duas, ou seja, os bits de índice ímpar ficarão todos alinhados em sequência em

uma linha e na outra linha os bits com índices pares.Essas duas linhas de bits irão para o modulador OQPSK. Sua saída é passada por um filtro *raised cosine*, que vai para o bloco "TX Banda passante", onde será multiplicado por um sinal de portadora para transformar em um sinal passa-faixa.

Após sair do bloco TX, ele passa pelo bloco AWGN (*Additive white Gaussian noise*), que representa um canal com ruído aditivo branco, que irá gerar um ruído proporcional à relação SNR (*Signal to Noise Rate*) desejada. Saindo desse bloco, ele entra no bloco de recepção RX. A Figura 58 apresenta os blocos que constituem o bloco RX.

Figura 58 – Bloco RX.



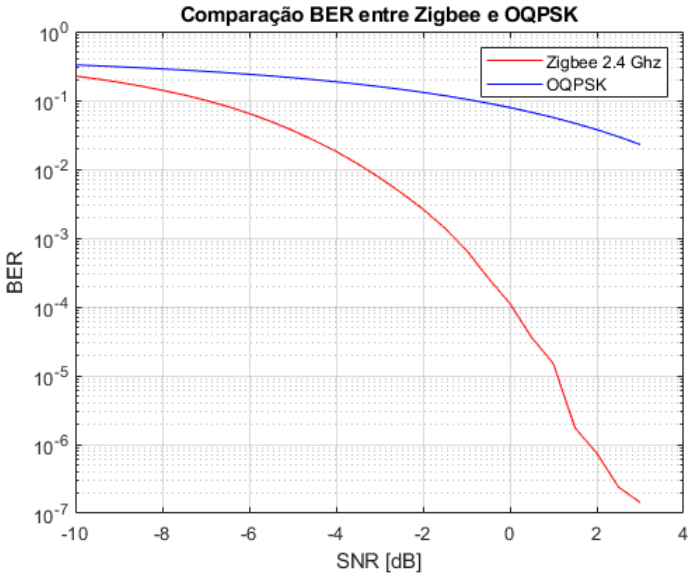
Fonte: Autor.

O bloco “RF para banda base”, tem a função de pegar o sinal enviado pelo bloco AWGN e multiplicar pela mesmo sinal de portadora que foi usado no TX, para transformar o sinal em um sinal passa-baixa. O bloco “Demodulacao OQPSK”, tem a responsabilidade de aplicar o filtro de recepção *raised cosine* e realizar a demodulação OQPSK do sinal resultante. Após isso, ele irá unificar as duas sequências de *bits*, transformando os dados em uma única fila, que será a entrada do bloco “Multiplicador da sequencia e Integrador”. Nesse bloco, o sinal que chega é comparado com todas as 16 sequências. A sequência que for mais próxima do sinal que chegou (apresentou menos erros), através do demultiplexador irá passar o número de sua entrada (que pode ser de 0 a 15) como um valor inteiro. Após isso, o valor inteiro será convertido para *bits* em grupos de 4 e depois será transformado em uma fila única de *bits*. Essa fila irá entrar no bloco comparador que irá comparar se o bit transmitido é igual ao final do processo de recepção para calcular a BER.

Para o modelo construído, foi realizada uma simulação para obter valores de BER com a variação do valor de SNR entre -10 e 3 (passos de 0,5). Para cada valor de SNR, foram gerados pelo menos 1 milhão de *bits* aleatoriamente. Para os casos em que a BER chega a aproximadamente um erro a cada dez milhões, foram gerados bits até a geração de pelo menos 5 erros. A Figura 59 apresenta o resultado obtido da simulação e em comparação apresenta também a curva de BER teórica para uma modulação OQPSK. Cabe ressaltar aqui que com o espalhamento espectral, ele se torna uma modulação "quase-ortogonal"(não é totalmente ortogonal pois ele utiliza 16 sequências de 32 chips. Se utilizasse 32 sequências seria considerado ortogonal), logo a comparação entre o padrão ZigBee simulado e OQPSK tem apenas o intuito de apresentar as diferenças que o espalhamento espectral pode apresentar.

Pode-se perceber pela figura 59, que o padrão ZigBee apresentou um desempenho

Figura 59 – Resultados da BER em função da SNR para canais AWGN no Simulink para o padrão Zigbee e a modulação OQPSK.



Fonte: Autor.

bem melhor que o OQPSK. Embora no início os valores de BER estejam próximos, para o valor de SNR igual a 0 dB (que é o valor onde a potência do ruído é igual a potência do sinal), a diferença fica grande. Para a modulação OQPSK pode-se verificar uma BER de 0,08, ou seja, 8 erros a cada 100 *bits*, já para o padrão ZigBee, uma BER de 0,0001, ou seja, 1 erro a cada 10000 *bits*. Para uma mesma situação o conjunto espalhamento espectral e modulação apresentou um desempenho muito melhor, o que permite o padrão ZigBee operar até em condições de SNR negativas (condições onde a potência do ruído é maior que a potência do sinal transmitido) e apresentar resultados satisfatórios.

Se considerarmos uma BER de 10^{-6}, ou seja, um erro a cada um milhão de *bits* como parâmetro satisfatório a ser alcançado na comunicação sem fio, pode-se verificar que o padrão ZigBee irá precisar de uma SNR de aproximadamente 2 dB.

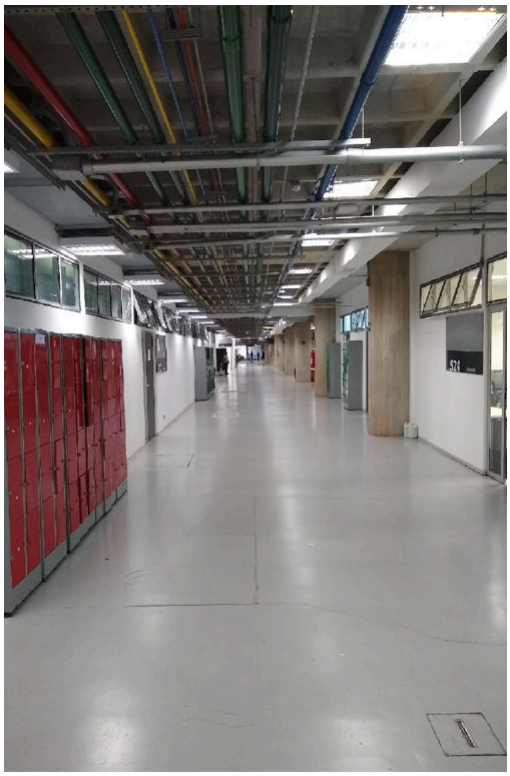
Após as simulações e uma compreensão melhor do comportamento do padrão ZigBee, serão realizados testes do sistema propostos para a verificação de alguns parâmetros importantes do sistema. Os testes serão realizados em ambiente *indoor* (interno), *outdoor* (externo) e ferroviário, além da realização de testes com a utilização de um *Sniffer*. Os resultados serão apresentado nas próximas seções.

6.2 Análise do sistema embarcado de comunicação baseado no padrão ZigBee através de medidas no ambiente *indoor*

Os testes em ambiente *indoor* foram realizados no subsolo do bloco A do campus da Universidade Federal do ABC (UFABC) em Santo André. A Figura 60 apresenta o

corredor utilizado.

Figura 60 – Corredor do subsolo do bloco A da UFABC em Santo André.



Fonte: Autor.

Nesse ambiente, foram realizados os testes de PER, SNR e RSSI e os resultados obtidos serão apresentados a seguir.

6.2.1 RSSI

O primeiro teste realizado foi o de RSSI para verificar a intensidade do sinal recebido conforme é alterado a distância do receptor em relação ao transmissor. Para realizar o teste, foram feitas marcações no chão de 10 em 10 m, totalizando uma distância de 100 m. Após as marcações, foram utilizados os suportes com 2 m de altura em relação ao solo para fixar os protótipos. Foi então fixado o transmissor (TX) na primeira marcação e o receptor era movido entre as marcações mantendo sempre visada e alinhamento. A Figura 61 apresenta como foi feito o alinhamento dos suportes.

Para realizar as medidas, foi necessário criar um programa para o Arduino TX (Coordenador), utilizando a biblioteca "*xbee-arduino-master*". Esse programa enviava um pacote para o endereço do Roteador A a cada 500 ms continuamente.

Já o programa para o Arduino RX (Roteador A), com auxílio da mesma biblioteca, ficava sempre lendo o *buffer* da serial, quando chegava alguma informação (chegada de pacote via RF), ele comparava se o tipo de *frame* recebido era o esperado, caso positivo, ele enviava ao *XBee* o comando AT "DB" que é o comando para verificar a RSSI do último

Figura 61 – TX e RX em seus respectivos suportes e alinhados.

Fonte: Autor.

pacote recebido. Após receber a resposta do *XBee*, ele verificava se o tipo do *frame* do pacote que chegou era do tipo resposta de comando AT, se sim, ele extraia do pacote a informação do RSSI e enviava para o computador através do Monitor Serial da IDE de programação do Arduino.

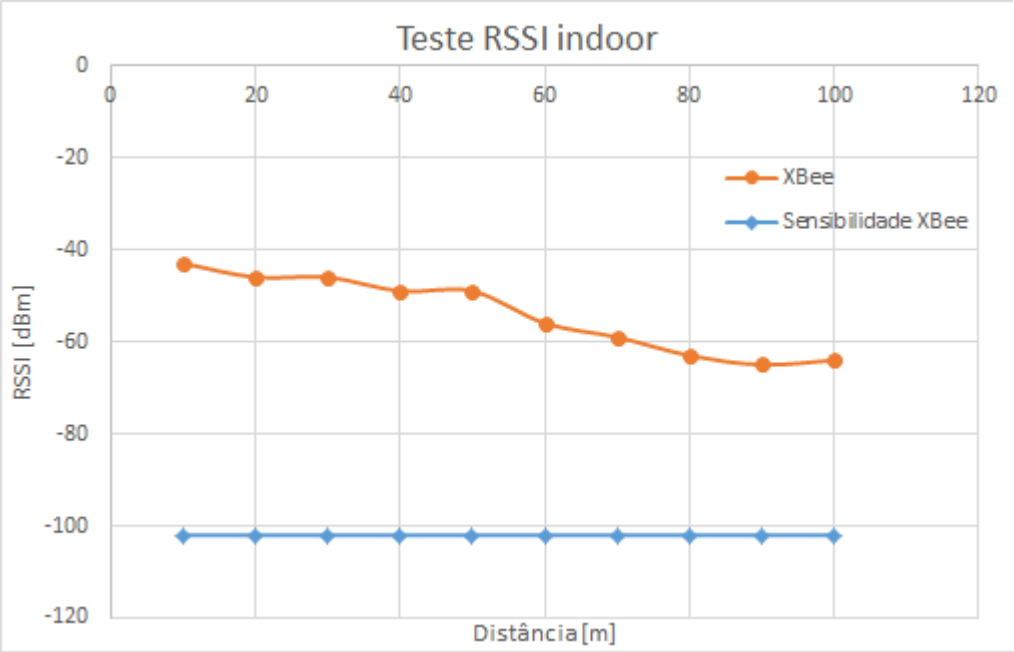
Colocado o RX em cada uma das posições, eram esperados 30 segundos para as medidas estabilizarem, após esse tempo foi anotado a medida para esse ponto. Após obter as dez medidas, obtivemos o resultado apresentado na figura 62. Lembrando que este corredor está sujeito a sinais WiFi presente no ambiente estudado, além de outras interferências na faixa de frequência de operação de 2,4 GHz.

Pode-se perceber que até com 100 m de distância, com visada mantida e mesmo altura, o valor de RSSI ficou bem acima do limite de sensibilidade, o que indica que poderíamos aumentar a distância significativamente que ainda poderíamos ter recepção dos dados.

6.2.2 SNR

Para realizar o teste de SNR foi necessária a utilização de um analisador de espectro uma vez que o módulo XBee não é capaz de informar esse valor. Para obter a medida, primeiro utilizamos o suporte do RX a altura de 1 m do solo e fixamos uma antena nele. Após isso ligamos essa antena na entrada do analisador de espectro. O analisador foi

Figura 62 – Medidas de RSSI em função da distância.



Fonte: Autor.

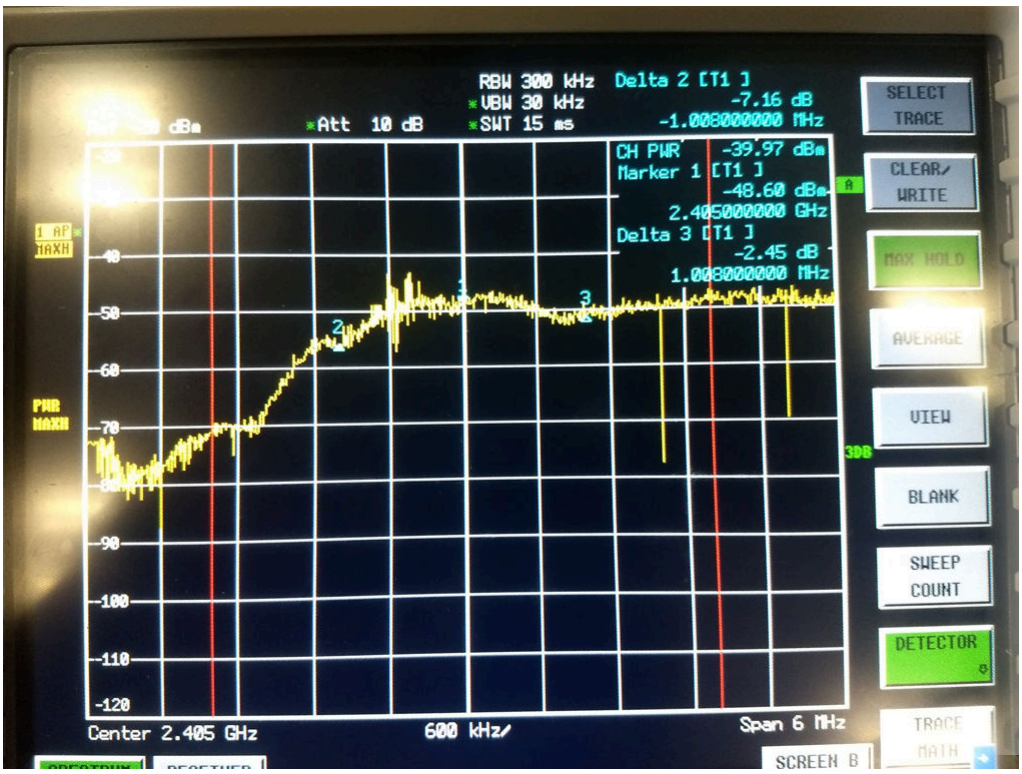
configurado para a frequência central de 2,405 GHz com um *span* de 6 MHz e foi ligado a função *Max Hold* (para manter na tela o máximo obtido em cada frequência), isso foi necessário porque o XBee não ocupa o canal constantemente, ele ocupa o canal apenas durante a transmissão, que é bem mais rápido do que a varredura em tempo real do analisador. O suporte com antena foi posicionado na primeira medida.

Após isso, o XBee do Coordenador foi configurado para operar no canal 11 apenas. Utilizando o mesmo programa de teste de RSSI, apenas alterando o tempo de envio para 100 ms e a configuração para não precisar de resposta ACK (*Acknowledgement*) após o envio do pacote. Então foi posicionado o TX em cada uma das medidas e deixado transmitir por 1 minuto e obtivemos a potência com o transmissor ligado. Depois desligamos o TX, limpamos as medidas do analisador e deixamos o analisador ligado por 1 minuto, para obter a potência do canal (ruído).

A Figura 63 apresenta o exemplo de medida de potência com TX ligado a uma distância de 10 m. Já a Figura 64 apresenta o exemplo de medida com o TX desligado (ruído) para a distância de 10 m. Na Figura 63 não é possível visualizar o espectro do sinal de transmissão de forma clara, isso ocorre devido a presença de outros sinais na faixa de frequência de 2,4 GHz.

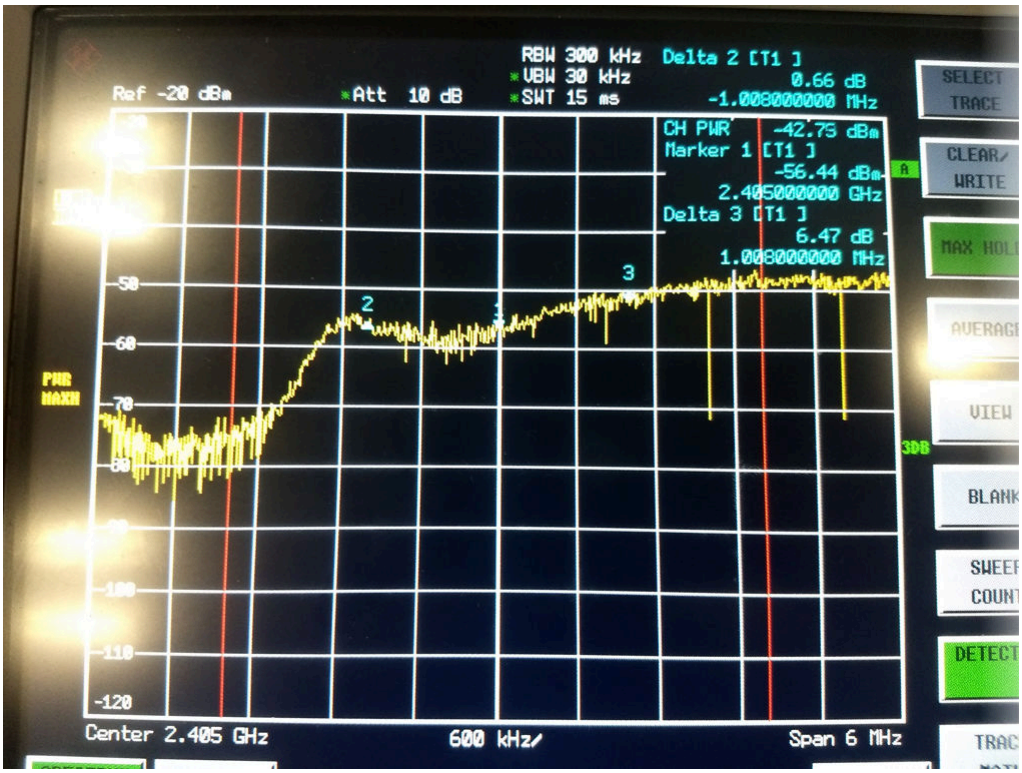
Após a leitura das duas medidas, utilizamos a equação 6.1 para calcular a SNR, onde $SNR(d)$ é o valor de SNR para a distância d em dBm, $P_{TX}(d)$ é o valor da potência medida com TX ligado para distância d em dBm e $P_{Ruido}(d)$ é o valor da potência medida com TX desligado para distância d em dBm.

Figura 63 – Tela do analisador de espectro a 10 m com TX ligado.



Fonte: Autor.

Figura 64 – Tela do analisador de espectro a 10 m com TX desligado (ruído).

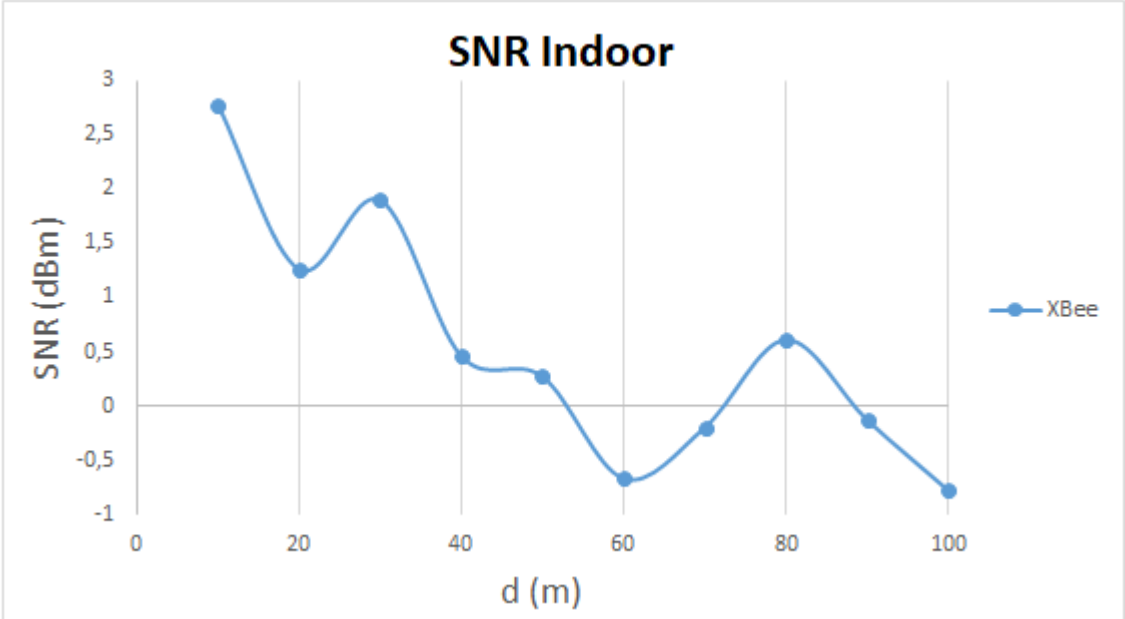


Fonte: Autor.

$$SNR(d) = P_{TX}(d) - P_{Ruido}(d) \tag{6.1}$$

Após o cálculo para cada distância, obtivemos o resultado apresentado na Figura 65. Analisando-a, pode-se perceber que o nível de ruído está bem alto em relação ao nível de transmissão, uma vez que os valores de SNR estão bem próximos de 0 dBm ou até mesmo negativo (potência do ruído é maior do que a potência de transmissão).

Figura 65 – Valores de SNR para cada distância.



Fonte: Autor.

6.2.3 PER

Para realizar o teste de PER, utilizamos o mesmo conjunto do teste de RSSI. A ideia principal é transmitir pacotes até que o receptor receba um total de 10 mil pacotes, após isso contabilizar quantos pacotes foram enviados e quantos foram recebidos para determinar a taxa de pacotes perdidos.

O programa do TX, enviava pacotes a cada aproximadamente 500 ms. O conteúdo do *payload* do pacote era de 4 *bytes*, esses 4 *bytes* foram divididos em dois grupos, sendo cada grupo com 2 *bytes*. O primeiro grupo irá contabilizar a perda de pacotes independentemente se houve ou não retransmissão, ou seja, se um pacote foi enviado e precisou de retransmissão para chegar ao destinatário será considerado apenas um pacote enviado pelo TX e um pacote recebido pelo RX (ou um pacote perdido caso não chegue mesmo com as retransmissões). Já o segundo grupo irá contabilizar cada retransmissão como um evento de perda de pacote, ou seja, se o transmissor enviou o pacote e teve 3 retransmissões para chegar ao destinatário, será considerado que o TX enviou 4 pacotes (o

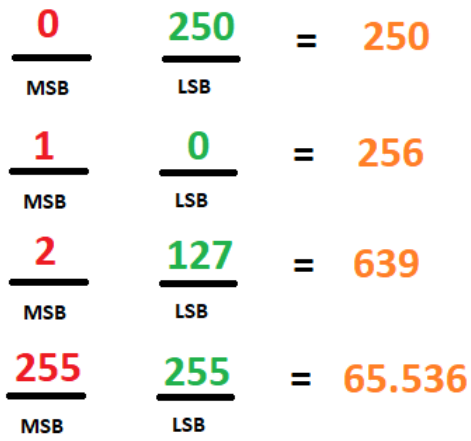
primeiro mais 3 retransmissões) e no RX será considerado o recebimento de 1 pacote e perda de 3 pacotes (pois o pacote é transmitido 4 vezes até ser corretamente recebido).

Para facilitar a contagem até 10 mil, no *payload* foi utilizado cada byte como um digito de um sistema de base 256, ou seja, dos dois *bytes*, o primeiro *byte* é o MSB e o segundo seria o LSB, sendo assim, é possível contar até $256 \times 256 = 65.536$. Para calcular na base 256, é utilizado a equação 6.2, onde MSB (*Most Significant Bit*) e LSB são respectivamente os *bytes* enviados no *payload*.

$$Total = 256^1 \times MSB + 256^0 \times LSB \tag{6.2}$$

A Figura 66 apresenta alguns exemplo de cálculos na base 256.

Figura 66 – Exemplos de cálculo na base 256.



Fonte: Autor.

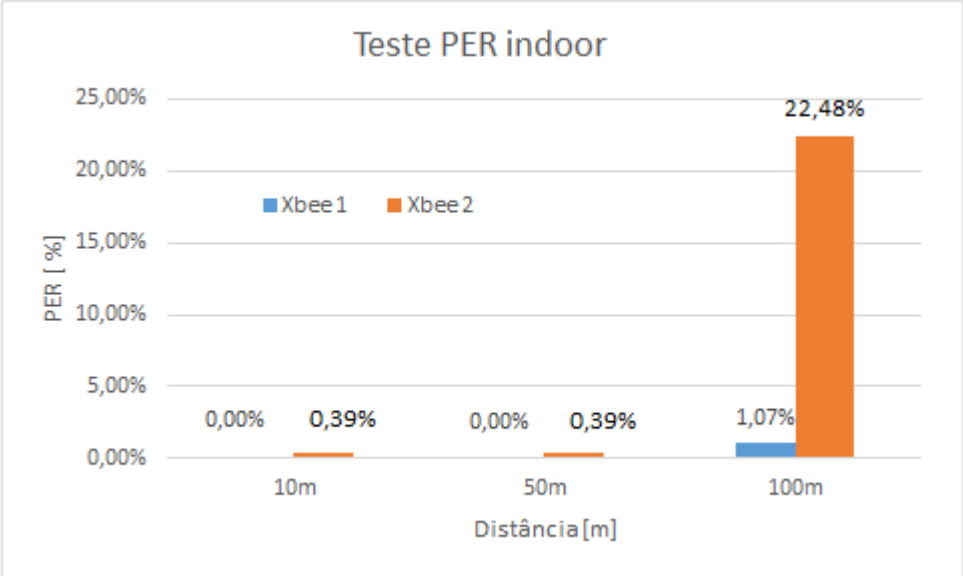
Para realizar o teste, o programa do TX iniciou os 2 *bytes* na base 256 com 0 e 1 e os outros 2 *bytes* também com 0 e 1. A cada pacote enviado, ele adiciona um no primeiro grupo (sem contabilizar retransmissão). Para atualizar o contador do segundo grupo, o transmissor aguarda o recebimento do pacote do tipo *status* de transmissão e quando recebe ele é possível extrair a informação de quantas vezes o pacote foi retransmitido até receber o ACK. De posse dessa informação, o Arduino adiciona ao contador do segundo grupo, o número de retransmissão mais um. Segundo (DIGI, 2018) no XBee, o número máximo de retransmissões é 4. Então o TX vai realizando esses procedimentos para cada pacote enviado, indefinidamente.

Já para o programa do RX, o programa verifica o *buffer* da serial, aguardando a chegada de pacotes. Ao receber um pacote, ele verifica se o tipo de *frame* do pacote é do tipo *receiver*. Se for o primeiro pacote, ele atualiza as variáveis internas de 16 bits para cada grupo de contadores. Se não for o primeiro pacote, ele olha os valores dos contadores do pacote que acabou de chegar, e subtrai os contadores internos desses valores recebidos. Se o resultado foi 1, sabe-se que não houve perda de pacotes. Caso o resultado seja diferente

de 1, houve perdas de pacotes, assim é contabilizado as perdas de pacote. A cada iteração, o RX obtinha o valor da RSSI de cada pacote da mesma forma que nos testes de RSSI, e enviava para o Serial Monitor o valor de RSSI e os valores dos dois grupos de contadores. O programa rodava até que o receptor recebesse 10 mil pacotes.

Esse teste foi realizado apenas nas distâncias de 10, 50 e 100 m. A Figura 67 apresenta os resultados obtidos. Na figura a legenda "XBee 1" representa o primeiro grupo (o qual não contabiliza as retransmissões) e a legenda "XBee 2" representa o segundo grupo (o qual considera cada retransmissão com um evento de perda de pacote).

Figura 67 – Valores de PER para cada distância.



Fonte: Autor.

Pelos resultados apresentado, para distância de 10 e 50 m, o XBee teve ótimos resultados, mesmo considerando as retransmissão como perda de pacotes. Já para 100 m, pode-se ver que houve 22,48% de perda nos pacotes. É um resultado interessante, uma vez que a essa distância, a SNR era -1 dB, ou seja a potência do ruído era maior do que a transmissão. Isso demonstra que o DSSS dá uma robustez a transmissão, porque mesmo em casos assim, ele perdeu apenas 1 pacote a cada 4 aproximadamente. Se considerarmos ainda o cenário em que o objetivo fosse que o RX recebesse o pacote, independente de ter retransmissões ou não, pode-se verificar que se perderam apenas 1% dos pacotes, ou seja, 100 pacotes em 10 mil transmitidos.

6.3 Análise do sistema embarcado de comunicação baseado no padrão ZigBee através de medidas no ambiente *outdoor*

Os testes de ambiente *outdoor*, foram realizados no campus da Faculdade de Tecnologia (FATEC) em São Bernardo do Campo. O local foi escolhido por apresentar

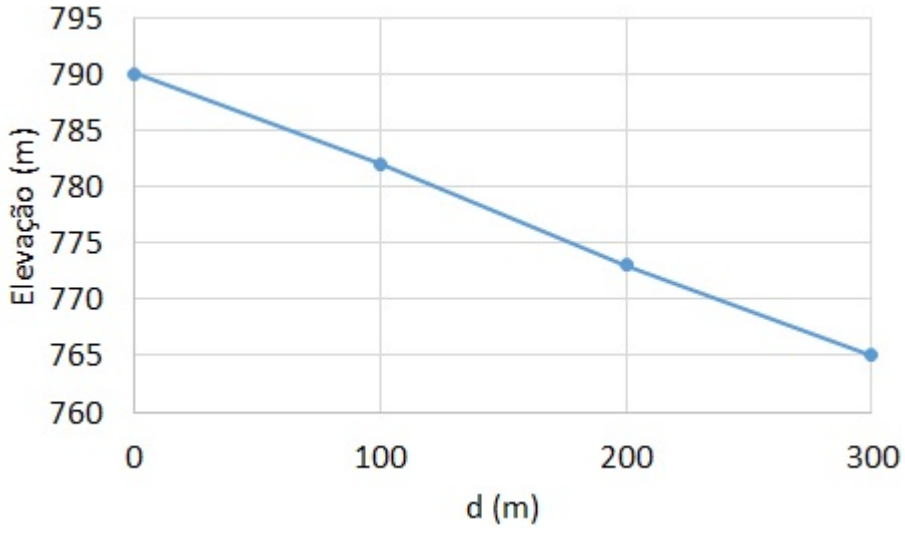
um terreno bem aberto e conseguir alcançar distância de até 300 m dentro do campus. A Figura 68 apresenta uma imagem do *software* Google Earth com os pontos onde ocorreram as medidas.

Figura 68 – Campus da FATEC vista pelo *software* Google Earth.

Fonte: Autor.

A Figura 69 apresenta uma estimativa do perfil de elevação do terreno, onde as medidas foram realizadas. Os dados foram obtidos através da utilização do *software* Google Earth. Pode-se verificar que o terreno é relativamente inclinado, apresentando uma diferença de elevação máxima entre TX e RX de aproximadamente 25 m.

Figura 69 – Perfil de elevação vista pelo *software* Google Earth.



Fonte: Autor.

Nesse ambiente, foram realizadas as medidas de RSSI e PER. Não foi possível realizar a medida de SNR devido falta de um analisador de espectro no local.

6.3.1 RSSI e PER

Nesse ambiente, os testes de RSSI e PER foram realizados simultaneamente. Para realizar os testes, fixamos o TX no primeiro ponto (ponto mais a direita na Figura 68), conforme apresentado na Figura 70.

Figura 70 – Posição do TX.

Fonte: Autor.

Após isso, o RX foi colocado nas distância de 100, 200 e 250 metros (respectivamente as Figuras 71, 72 e 73). A distância de 300 m não foi possível realizar comunicação entre TX e RX, uma vez que o RX não conseguiu receber nenhum pacote a essa distância.

Os programas utilizadas para os testes foram os mesmos utilizados para o teste de PER no ambiente *Indoor*, uma vez que esse programa já apresenta a RSSI para cada pacote. Para definir o valor de RSSI para cada medida, primeiro é esperado a transmissão estabilizar e após isso, é realizado uma média entre 10 medidas de RSSI. A Figura 74 apresenta os resultados obtidos para cada medida.

Pelos resultados obtidos, pode-se inferir que o não recebimento de pacotes a distância de 300 m, ocorreu por causa da elevação do terreno (tornando a comunicação sem visada) e a interferência presente no local, pois a 250 m obtivemos uma RSSI de -83 dBm, ou seja, ainda havia uma margem de aproximadamente 20 dBm pela sensibilidade

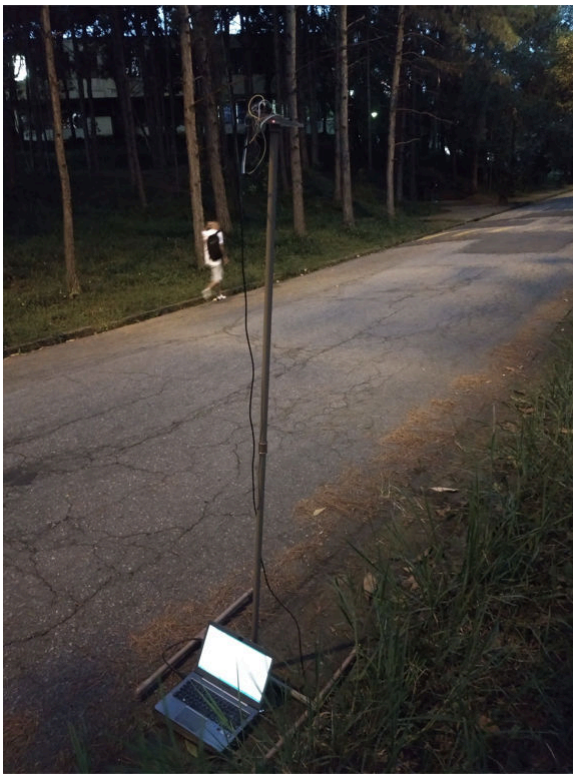
Figura 71 – Posição do RX a 100 m.

Fonte: Autor.

Figura 72 – Posição do RX a 200 m.

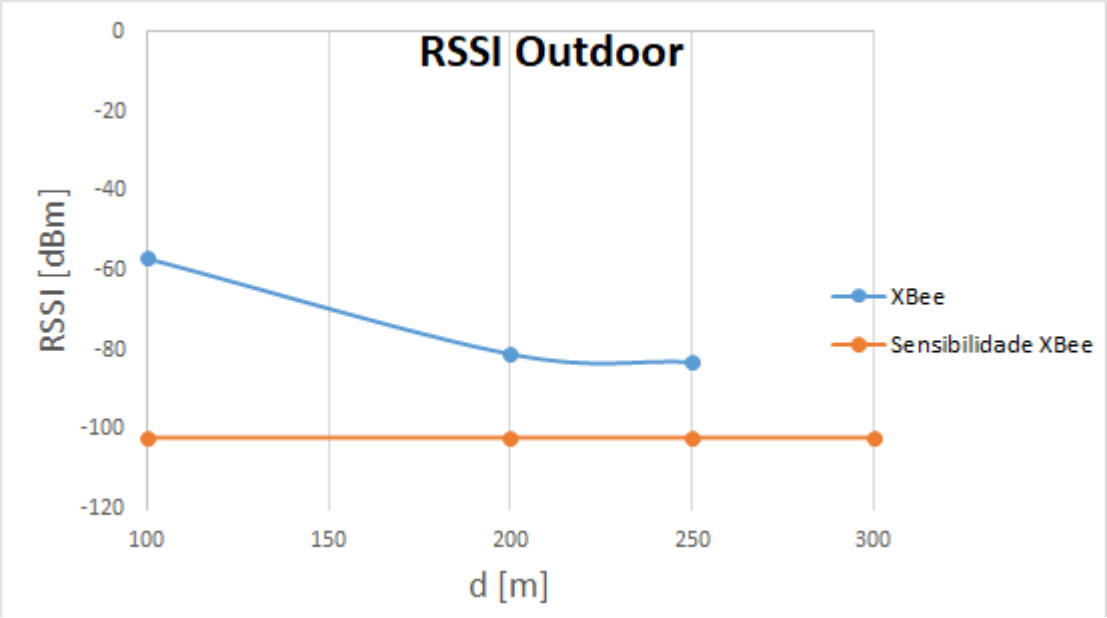
Fonte: Autor.

Figura 73 – Posição do RX a 250 m.



Fonte: Autor.

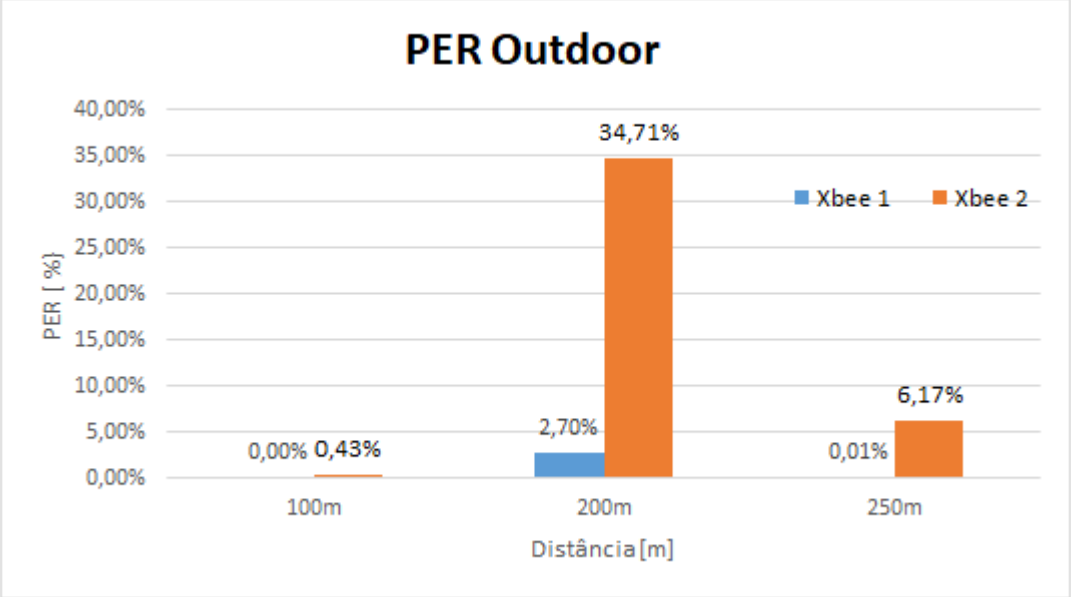
Figura 74 – RSSI em função da distância.



Fonte: Autor.

do XBee (-102 dBm). Já a Figura 75, apresenta o resultado das medidas de PER obtidas. Na figura as legendas seguem o padrão do teste de PER indoor.

Figura 75 – PER em função da distância.



Fonte: Autor.

Pode-se perceber que em 100 m, houve um ótimo resultado, com apenas algumas pouquíssimas retransmissões. Em 250 m, pode-se verificar que houve um aumento nas retransmissões, uma vez que houve aproximadamente 6% de perdas considerando as retransmissões como um evento de perda. Porém se for considerado o objetivo do pacote chegar ao destinatário, houve 0,01% de perdas o que é um bom resultado.

Já em 200 m nota-se que houve um aumento significativo no número de pacotes perdidos. Como não foi possível medir a SNR nesse ponto e se baseando nos resultados de RSSI e PER de 250 metros, pode-se inferir que em 200 m, havia uma interferência muito grande para que houvessem tantas retransmissões. Essas interferências podem ter sido causadas pelas árvores que cobrem o local ou interferências de sinais na faixa de frequência de operação como WiFi entre outros. Embora houve quase 35% de perdas considerando as retransmissões como pacotes perdidos, pode-se verificar que se desconsiderarmos as retransmissões, ou seja, que o objetivo era que o pacote chegasse independentemente de retransmissões, se perderam apenas 2,7% dos pacotes, ou seja, aproximadamente 270 pacotes em 10 mil.

6.4 Análise do sistema embarcado de comunicação baseado no padrão ZigBee através de medidas no ambiente ferroviário

Após a realização de testes *indoor* e *outdoor*, serão realizados testes no ambiente ferroviário, para analisar o comportamento do sistema embarcado frente aos problemas que o ambiente pode gerar, como circulação de material rodante, campo magnético e elétrico gerado pela corrente de tração dos trens além de outras interferências que podem ocorrer na faixa de frequência de operação de 2.4 GHz, uma vez que o ambiente ferroviário normalmente é totalmente aberto, cruzando ruas e avenidas.

6.4.1 RSSI

O primeiro teste a ser realizado foi o de RSSI em função da distância, porém esse teste será realizado para medições de longa distância. Para realizar essas medidas, foi escolhido o trecho entre o final da plataforma da Luz, até a região do Pari, próximo ao Brás. Esse trecho foi escolhido, pois apresenta um trecho de reta de 1 km, onde é possível realizar os testes com visada direta entre os dispositivos.

A Figura 76 apresenta o percurso utilizado para o teste de RSSI. Ela foi obtida através de uma imagem extraída do *software* Google Earth.

Figura 76 – Percurso para teste de RSSI.

Fonte: Autor.

Após a escolha do local, foram realizadas marcações nos trilhos a cada 100 m, até obter a distância de 1 km. Não foi possível realizar medida para distâncias maiores porque os dispositivos ficariam sem visada, uma vez que existe uma curva grande nessa região do Pari.

O teste foi realizado da seguinte maneira, primeiro foi fixado o RX no final da plataforma central da estação da Luz sentido a estação do Brás. Foi determinado que a altura utilizada no teste seria de 3 m em relação ao trilho, para manter visada direta e não ter no caminho alguns obstáculos que existem abaixo dessa altura. Como o RX ficou posicionado na plataforma e a mesma possui 1,5 m de altura em relação ao trilho, seu suporte tinha então 1,5 m de altura. As Figuras 77 e 78 apresentam como ficou posicionado

o RX. A Figura 77 apresenta o suporte com o dispositivo ligado ao *notebook*, já a Figura 78 apresenta o dispositivos sentido o percurso do teste.

Figura 77 – Foto do posicionamento do RX.

Fonte: Autor.

Já para o dispositivo TX, foi utilizado um suporte com altura de 3 metros, para manter o alinhamento entre os dispositivos. A Figura 79 apresenta uma foto do TX, sendo que a figura 79(a) apresenta a foto sentido dispositivo RX e a 79(b) apresenta sentido o percurso do teste.

Foi movido o TX de 100 em 100 m, de acordo com as marcações realizadas no trilho. Uma vez posicionado o TX, ele transmitia pacotes continuamente para o RX. No RX o dispositivo obtinha a RSSI a cada novo pacote que chegava e realizava uma média móvel de 10 medidas. Aguardou-se um tempo de 1 minuto para que as medidas obtidas da média móvel apresentassem uma estabilidade e foi anotada essa medida. Após isso, o TX era movido para a próxima medida até chegar à última medida de 1 km. A Figura 80 apresenta o resultado obtido dessas medidas.

Importante ressaltar que esse teste foi realizado no sentido paralelo a circulação de trens, ou seja, em nenhum momento houve circulação de trem que cruzasse o caminho entre os dispositivos. Pode-se verificar pelo resultado, que os dispositivos ainda conseguiriam transmitir a uma distância maior, pois os valores estavam acima da sensibilidade máxima do XBee, porém devido a existência de curva na região, não seria possível manter a visada entre os dispositivos. Na distância de 1 km, ainda havia uma diferença de aproximadamente -10 dBm entre a medida e a sensibilidade máxima do dispositivo. Empiricamente foi percebido

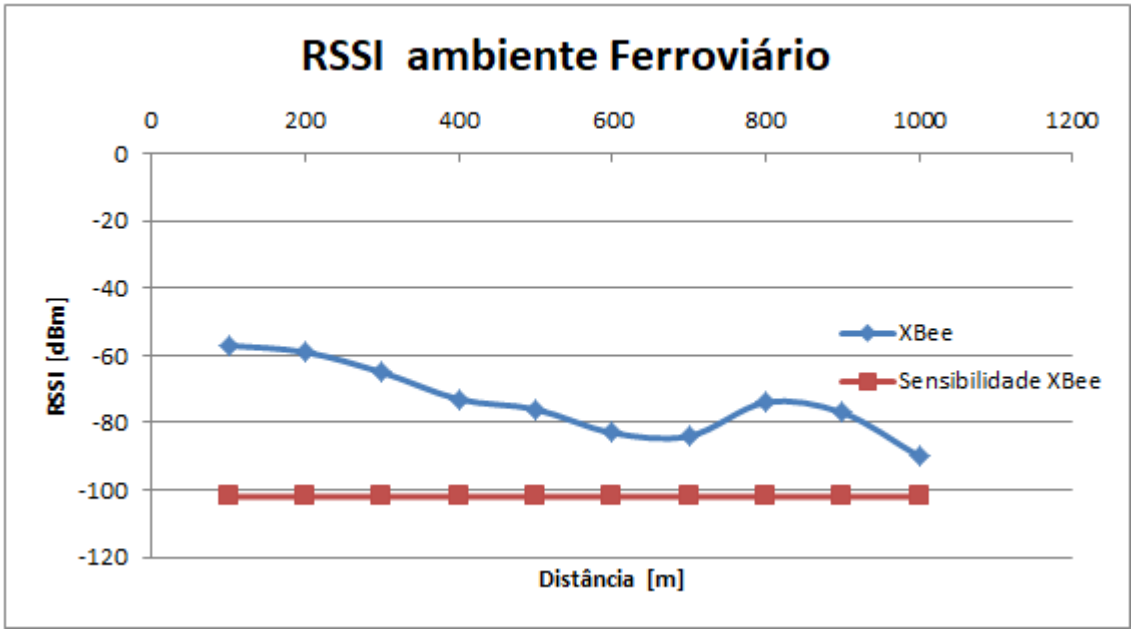
Figura 78 – Foto do posicionamento do RX com visão para o percurso do teste.

Fonte: Autor.

Figura 79 – (a) Foto do posicionamento do TX com visão para o dispositivo RX. (b) Visão para o percurso do teste.

Fonte: Autor.

Figura 80 – Resultado do teste de RSSI.



Fonte: Autor.

que quando havia circulação de trens na via 2 (via ao lado direito do suporte de TX na figura 79(b), havia um aumento significativo na recepção do RX, muitas vezes chegava até 10 dBm de ganho, porém todas as medidas foram realizadas com ausência de trens nessa via, nas demais (2 vias paralelas a esquerda do RX e uma a direita) circulavam normalmente. Uma fato importante foi que em 800 m houve um aumento da RSSI, essa região fica em uma ponte ferroviária que passa por cima da Av. dos Estados e é um lugar totalmente aberto lateralmente, ou seja, não existem muros de pedra ou tijolo em suas laterais, além da circulação de veículos por baixo dessa ponte. As medidas de 800 e 900 m ficam dentro dessa ponte sem a existência de muros laterais.

6.4.2 PER utilizando a topologia *mesh*

O segundo teste realizado foi o teste de PER, porém esse teste foi realizado de forma diferente aos realizados no ambiente *indoor* e *outdoor*, pois com a realização desse teste, além da PER, serão verificados outros parâmetros e comportamento dos protótipos.

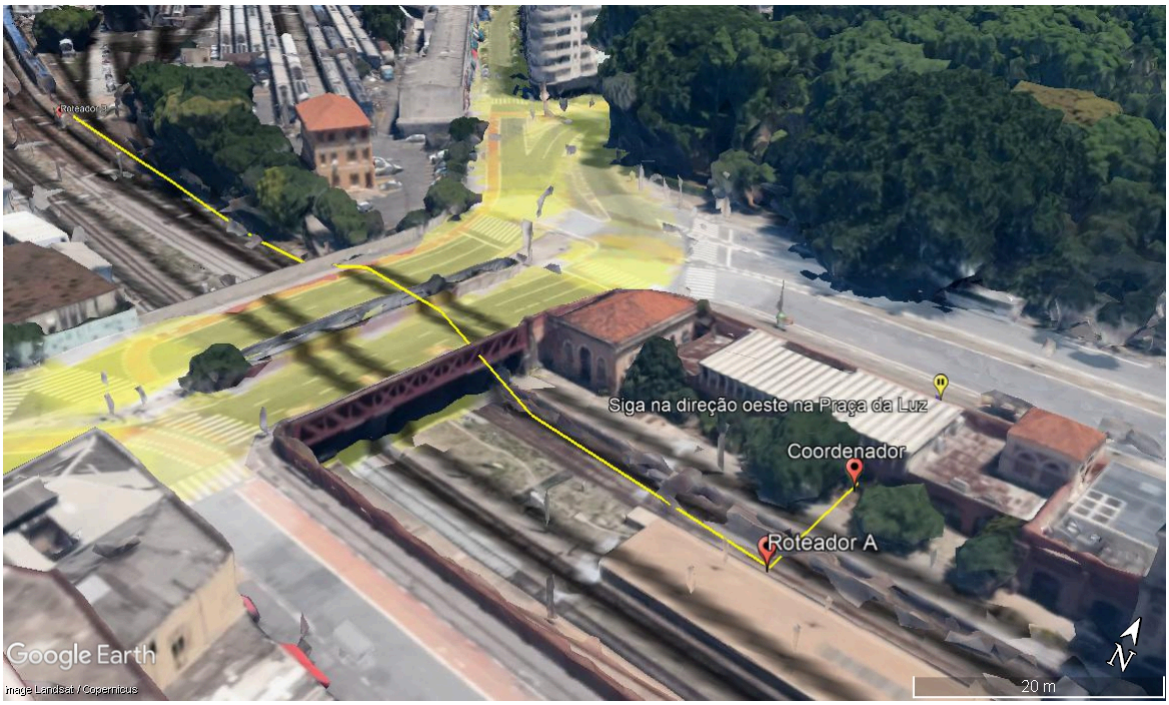
Em todos os testes anteriores, sempre foi utilizado apenas dois dispositivos, ou seja, só existia uma única opção de rota entre os dispositivos, isso representa basicamente a topologia do tipo estrela, porém esse trabalho visa apresentar as vantagens da topologia *mesh*, onde um dispositivo normalmente terá mais de uma opção de rota para chegar no seu destinatário. Nesse teste será utilizado a topologia *mesh*, com a utilização de 3 dispositivos para a comunicação.

O teste foi realizado com o intuito de envio do máximo de pacotes entre os dispositivos durante as 24 h consecutivas, para avaliar o comportamento do protótipo

perante o ambiente ferroviário real, ou seja, circulação de trens em horário de pico, horário de vale e quando a circulação comercial está interrompida, além de avaliar circunstâncias em que não existe visada direta e cruzamento de trens entre os dispositivos. Também será avaliado o comportamento dos dispositivos, uma vez que eles terão que gerar um log para cada pacote, sendo esse log composto pela data e hora da chegada do pacote, RSSI, conteúdo do pacote e a atualização dos contadores de pacotes atuais, além de processar cada tipo de pacote que chega, verificando qual é o seu tipo e se é o esperado no momento.

O primeiro passo foi determinar em quais locais seriam colocados os dispositivos, e dessa vez, seriam utilizados os três dispositivos durante o teste. A Figura 81 apresenta em vermelho, os três pontos onde ficaram os dispositivos (Coordenador, Roteador A e Roteador B) e o trajeto amarelo apresenta o caminho entre eles. Foi realizada a medida entre o Roteador B e Roteador A, e entre Roteador A e Coordenador, e obtivemos respectivamente 205 e 20 m. A distância em linha reta entre Coordenador e Roteador B também é de aproximadamente 205 m.

Figura 81 – Percurso para teste de PER.



Fonte: Autor.

A Figura 81 foi obtida através do *software* Google Earth. Pela figura, pode-se perceber que entre os percursos, não existe visada direta, uma vez que eles estão posicionados em alturas diferentes, além da existência de concreto, viadutos, plataformas e outras construções que podem atenuar significativamente o sinal. O conjunto de figuras apresentado entre a Figura 82 e 87, apresentam fotos dos locais.

A Figura 82, 83 e 84 apresentam respectivamente, o local onde foi colocado o Coordenador, o Roteador A e o Roteador B.

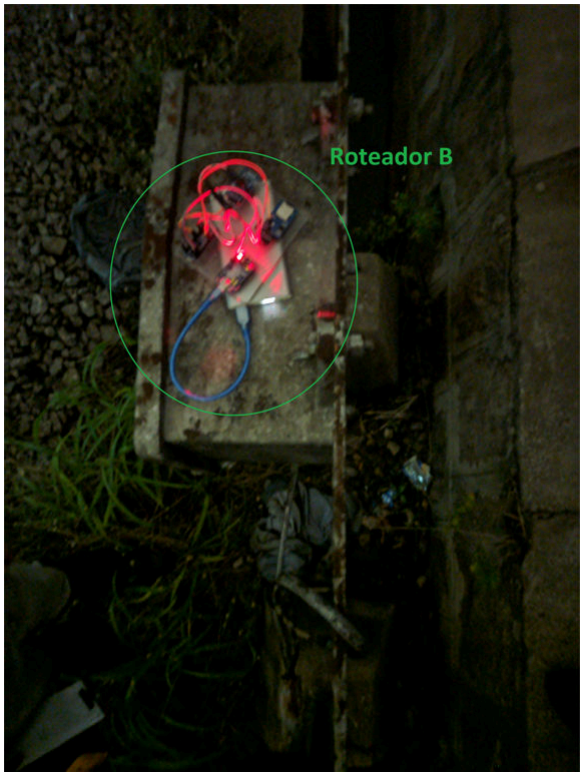
Figura 82 – Local do Coordenador.

Fonte: Autor.

Figura 83 – Local do Roteador A.

Fonte: Autor.

Figura 84 – Local do Roteador B.



Fonte: Autor.

A Figura 85 apresenta a visão entre os dispositivos Coordenador e Roteador A. Pode-se visualizar que existem paredes de concreto e a plataforma entre os dispositivos, além de estarem em alturas diferentes, ou seja, eles não possuem visada direta e o Roteador A está posicionado em um local que a sua parte superior é fechada, apenas as laterais são abertas.

Já a Figura 86(a) e 86(b), apresentam a visão entre os dispositivos Roteador A e Roteador B, onde pode-se visualizar que o viaduto está interrompendo a visão direta deles, além de termos outras estruturas no direcionamento deles. Outro ponto é que estão em alturas diferentes e o trecho apresenta uma curva significativa. A Figura 87 apresenta a vista reta alinhada com a posição do Roteador B.

Esses locais foram escolhidos estrategicamente, pois foram posicionados de acordo com a posição de equipamentos da sinalização existentes. O coordenador foi colocado dentro da base da Sinalização da Luz, pois em um sistema real, o concentrador de todas as informações seria posicionado nesse local, uma vez que ali já é um concentrador de equipamentos ligados utilizando condutores elétricos de cobre. Para a posição do Roteador A, foi escolhido o equipamento de sinalização mais próximo do concentrador, que no caso foi o sinaleiro luminoso que pode ser visualizado na figura 86(a), esse sinal protege três travessões na Luz. Já o roteador B, foi colocado no local que é paralelo ao sinal contrário de proteção dos travessões, para manter a posição e distâncias entre os equipamentos de sinalização, além de ser um local que frequentemente ocorre travessia de trens, que

Figura 85 – Visão do Coordenador e Roteador A.

Fonte: Autor.

Figura 86 – (a) Visão do Roteador A para o Roteador B. (b) Visão do Roteador B para o Roteador A.

Fonte: Autor.

Figura 87 – Visão reta alinhada com o Roteador B.

Fonte: Autor.

podem interferir no sinal e pelo fato de ser um local que não tem nenhuma visada para o Coordenador e o Roteador A. Outro ponto importante, é que a distância entre esses sinais, não é muito comum, normalmente um travessão simples, tem metade dessa distância entre sinais, com isso pode-se verificar o comportamento dos dispositivos para distâncias mais extremas sem nenhum dispositivo no meio.

Essas escolham tornam o teste bem completo, pois não será fácil encontrar cenários piores que esse, uma vez que pela metodologia do sistema de monitoramento e controle, seriam instalados protótipos em todos os equipamentos, o que ajudariam nas rotas de tráfego dos pacotes, caso uma rota esteja com interferência.

Após a escolha dos locais, foi gerado o programa para cada um dos dispositivos. Após uma análise de taxa máxima de envio de pacotes, foi estipulado que seriam enviados pelo menos 100 mil pacotes do Coordenador para o Roteador A e outros 100 mil pacotes entre o Coordenador e o Roteador B. Para que isso ocorresse em 24 h, foi necessário determinar um intervalo fixo para o envio dos pacotes para manter a regularidade. A seguir será detalhado sobre os códigos desenvolvidos para cada um dos dispositivos.

6.4.2.1 Coordenador

O programa desenvolvido para utilização no Coordenador, se assemelha muito ao utilizado no teste de PER nos ambientes *indoor* e *outdoor*, com algumas modificações.

A alteração principal foi a utilização do *Timer* 2 do Arduino, apresentado no anexo C.3.2. Foi utilizado esse recurso para sincronizar o tempo de envio dos pacotes. Foi utilizado o *Timer* de forma a contar intervalos de aproximadamente 432 ms, pois a cada intervalo desse, o *Timer* 2 gera uma interrupção, para enviar um pacote e alterar a variável de controle para enviar o próximo pacote para o outro roteador, ou seja, ele manda um pacote para o Roteador A, espera 432 ms, envia um pacote para o Roteador B, espera novamente 432 ms e envia novamente um pacote para o Roteador A. Resumidamente, leva-se aproximadamente 864 ms entre os pacotes para um mesmo Roteador. Dado o tempo total de 24 h e dividir pelo tempo de cada pacote, será obtido o envio de exatamente 100 mil pacotes.

Outra alteração importante foi nos contadores de pacotes, diferente dos outros testes, onde foram utilizados 4 *bytes* (2 *bytes* para cada grupo de contador), aqui foram utilizados 6 *bytes* (3 *bytes* para cada grupo). Isso porque com 2 *bytes*, pode-se contar apenas até 65.536 pacotes. Já com 3 *bytes*, pode-se contar mais de 16 milhões.

Então após o envio do pacote pelo Coordenador e contagem das retransmissões, o Coordenador obtém a data e hora atual do envio do pacote, ou seja, ele obtém o dia, mês, ano, hora, minuto e segundo do módulo de RTC. Após isso, eram obtidos os dados do conteúdo do pacote e contadores (valores dos 6 *bytes* enviados, contadores de pacotes totais enviados, número de retransmissões, o código de status de envio de cada pacote e o endereço para quem foi enviado o pacote). Após obter todos esses dados, o Coordenador utilizando o módulo leitor de cartão SD, cria um arquivo com a extensão CSV, onde para cada linha ele armazena os valores obtidos. A Figura 88 apresenta o resultado do *log* gerado pelo Coordenador.

Figura 88 – Log gerado pelo Coordenador.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Endereço	Data	Hora	Pacote	Pacote 2	Pacote 3	Total Pacotes	Cod_Envio	Retransmissoes	Pct_ret	Pct_ret2	Pct_ret3	Total com retransmissoes
2	13a200409e87a5	23/3/2021	1:59:48	1	0	0	1	0	0	1	0	0	1
3	13a200406b4547	23/3/2021	1:59:49	1	0	0	1	24	0	1	0	0	1
4	13a200409e87a5	23/3/2021	1:59:49	2	0	0	2	0	0	2	0	0	2
5	13a200406b4547	23/3/2021	1:59:50	2	0	0	2	0	4	2	0	0	2
6	13a200409e87a5	23/3/2021	1:59:51	3	0	0	3	0	4	3	0	0	3
7	13a200406b4547	23/3/2021	1:59:51	3	0	0	3	0	0	7	0	0	7
8	13a200409e87a5	23/3/2021	1:59:51	4	0	0	4	0	0	8	0	0	8
9	13a200406b4547	23/3/2021	1:59:51	4	0	0	4	0	0	8	0	0	8
10	13a200409e87a5	23/3/2021	1:59:52	5	0	0	5	0	0	9	0	0	9
11	13a200406b4547	23/3/2021	1:59:52	5	0	0	5	0	4	9	0	0	9
12	13a200409e87a5	23/3/2021	1:59:53	6	0	0	6	0	4	10	0	0	10
13	13a200406b4547	23/3/2021	1:59:53	6	0	0	6	0	0	14	0	0	14
14	13a200409e87a5	23/3/2021	1:59:54	7	0	0	7	0	0	15	0	0	15
15	13a200406b4547	23/3/2021	1:59:54	7	0	0	7	0	0	15	0	0	15

Fonte: Autor.

6.4.2.2 Roteadores

O programa desenvolvido para utilização nos roteadores também se assemelha bastante ao utilizado nos testes de PER no ambiente *indoor* e *outdoor*. As principais diferenças são o uso de 6 *bytes* nos contadores ao invés de 4 e a sequência de passos na recepção de pacotes.

Para cada pacote recebido, primeiramente o Arduino irá checar se é um pacote do tipo recepção de dados RF, se sim, ele já se comunica com o módulo RTC e obtém data e hora atual da chegada do pacote. Após isso, ele analisa o pacote para verificar se os dados estão corretos e dentro do esperado, atualiza seus contadores e já envia uma solicitação de comando AT, para obter o valor da RSSI daquele pacote. Realizadas essas tarefas, ele irá entrar em um *loop*, verificando os próximos pacotes, pois o próximo pacote esperado é o do tipo resposta de comando AT, porém pode acontecer de ele receber outro pacote do coordenador antes da resposta do comando AT. Nesse *loop*, ele verifica qual é o tipo do pacote que chegou, e, caso não seja o esperado, ele armazena esse pacote em uma *struct* para não perder os dados e continua aguardando a chegada do pacote de resposta de comando AT. Ele consegue armazenar na *struct*, até 3 pacotes enviados pelo coordenador, sem perder as informações enquanto aguarda a resposta do comando AT, porém os valores de RSSI dos 2 primeiros pacotes irão se perder.

Uma vez que ele recebe a resposta do comando AT, verifica se houve erros na resposta, caso não, ele irá armazenar esse valor junto com as informações obtidas do conteúdo do pacote mais os contadores atuais. Para salvar os dados, ele criará um arquivo de extensão CSV com o módulo leitor de cartão SD e preencherá cada linha com as informações de RSSI, dia, mês, ano, hora, minuto, segundo, conteúdo do pacote e seus contadores atuais. A Figura 89 apresenta o *log* gerado por um dos Roteadores.

Figura 89 – Log gerado pelo Roteador A.

	A	B	C	D	E	F	G	H	I	J	K	L	N
1	RSSI	Data	Hora	Pacote	Pacote 2	Pacote 3	Pct_ret	Pct_ret2	Pct_ret3	Código de erro	Pacotes recebidos	Pct Perdidos	Pct Perdidos com retr.
2	91	23/03/2021	02:00:04	1	0	0	1	0	0	0	1	0	0
3	93	23/03/2021	02:00:04	2	0	0	2	0	0	0	2	0	0
4	92	23/03/2021	02:00:06	3	0	0	3	0	0	0	3	0	0
5	90	23/03/2021	02:00:07	4	0	0	8	0	0	0	4	0	4
6	93	23/03/2021	02:00:07	5	0	0	9	0	0	0	5	0	4
7	93	23/03/2021	02:00:08	6	0	0	10	0	0	0	6	0	4
8	90	23/03/2021	02:00:09	7	0	0	15	0	0	0	7	0	8
9	91	23/03/2021	02:00:10	8	0	0	16	0	0	0	8	0	8
10	93	23/03/2021	02:00:11	9	0	0	17	0	0	0	9	0	8
11	91	23/03/2021	02:00:12	10	0	0	18	0	0	0	10	0	8
12	89	23/03/2021	02:00:13	11	0	0	19	0	0	0	11	0	8
13	92	23/03/2021	02:00:13	12	0	0	20	0	0	0	12	0	8
14	96	23/03/2021	02:00:16	13	0	0	21	0	0	0	13	0	8
15	95	23/03/2021	02:00:16	14	0	0	26	0	0	0	14	0	12

Fonte: Autor.

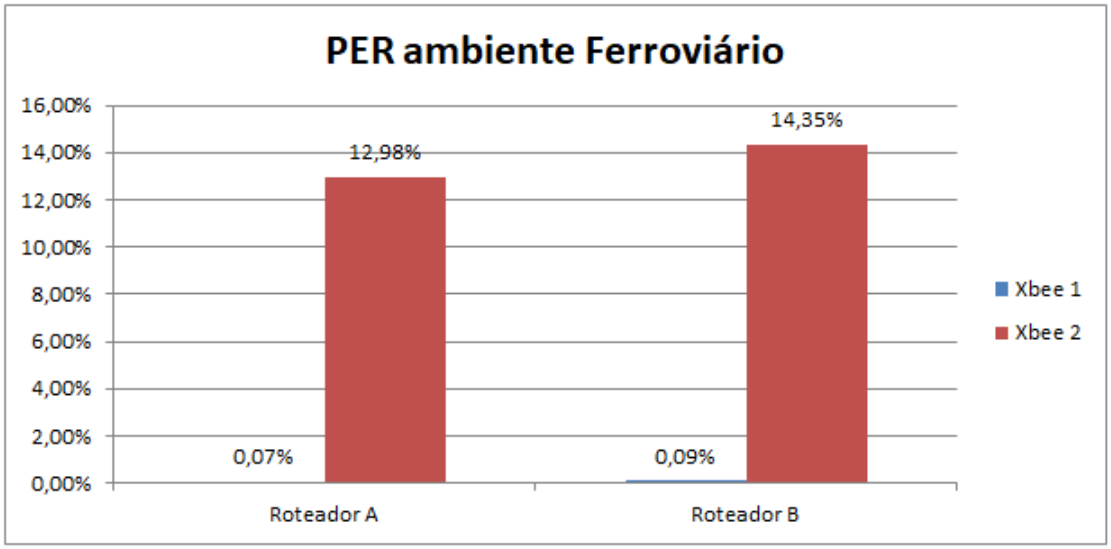
Após a geração do *log* no cartão SD, ele verifica se existem pacotes na *struct*, caso exista, ele irá refazer todos os passos citados anteriormente até que não existam dados na *struct*. Então ele volta aguardar a chegada de um novo pacote. O código para ambos os roteadores foi o mesmo, apenas com as mudanças de pinos indicados dentro do código, conforme foi apresentado na Figura 55.

Para execução dos testes, os roteadores foram alimentado por um carregador portátil, e o coordenador, foi ligado a um *notebook* que estava sendo alimentado por um carregador ligada a uma tomada.

6.4.2.3 Resultado

O teste foi realizado por um pouco mais de 24 h (aproximadamente 27 h), porém os resultados foram analisados até o momento que os roteadores recebessem 100 mil pacotes (aproximadamente 25 h). Após análise dos *logs*, foi obtido o seguinte resultado apresentado na Figura 90. As legendas da figura seguem o padrão explicado no teste de PER *indoor*.

Figura 90 – Resultado do teste de PER no ambiente ferroviário.



Fonte: Autor.

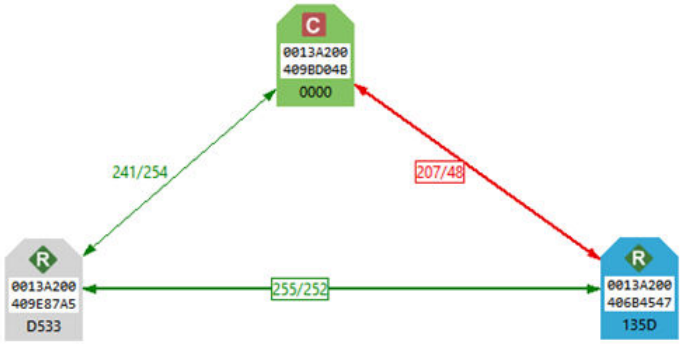
Pode-se verificar pela figura 90 que se desconsiderarmos as retransmissões, ou seja, que o objetivo era que os pacotes chegassem até o receptor, independente de retransmissões, para o Roteador A foram perdidos apenas 70 pacotes em 100 mil, e o Roteador B perdeu apenas 90 pacotes em 100 mil. Já considerando cada retransmissão como um pacote perdido, houve uma perda de quase 13% e 14% respectivamente.

Foi observado empiricamente, que quando havia cruzamento de trens entre os dispositivos, ocorriam muitas retransmissões, ou seja, quando havia trens entre o Coordenador e o Roteador A, as retransmissões para o Roteador A aumentaram significativamente. Na ausência de trens, pouquíssimas retransmissões ocorriam. Outro fator que isso gerava era um aumento nas retransmissões para o Roteador B, uma vez que a arquitetura *Mesh* possibilita várias rotas. A rota Coordenador - Roteador B estava bem prejudicada devido às construções e nenhuma visada entre os dispositivos. O Roteador B estava apresentando uma RSSI média de aproximadamente -85 dBm, porém como o Roteador A estava em uma posição melhor que o Coordenador, a rota utilizada pelo Coordenador era Coordenador - Roteador A - Roteador B. Com essa rota, a RSSI média no Roteador B (O valor de RSSI é apenas referente ao último salto da rota, ou seja, o trecho Roteador A - Roteador B) era de aproximadamente -75 dBm. Logo quando havia trem entre o Coordenador e o Roteador A, o Coordenador usava sua própria rota (Coordenador - Roteador B) para enviar para o Roteador B.

Isso demonstra a importância da arquitetura *Mesh* para essa configuração dos sistemas de sinalização, pois um trem tem em média quase 200 m, logo para o nosso teste implica que existem momentos em que o trem pode estar interrompendo o caminho entre Coordenador - Roteador A, Coordenador - Roteador B e o caminho Roteador A - Roteador B. Mesmo sobre essas circunstâncias que ocorrem em média de 5 minutos, houve um bom resultado. Com esses resultado e a análise junto ao teste de RSSI, pode-se inferir que um dispositivo conseguirá se comunicar com outro dispositivo a 2 km de distância, desde que exista um dispositivo no meio a uma distância de 1 km entre os dispositivos (considerando que os dispositivos na extremidade tenham visada direta para o dispositivo do centro). Essa distância é muito maior que a média de distância longa entre dispositivos do sistema de sinalização.

A Figura 91 apresenta a arquitetura da rede obtida pelo *software* XCTU. Nela é possível visualizar os 3 dispositivos dentro da rede, e suas possíveis rotas para cada dispositivo. Os dois valores próximos de cada *link*, indica o valor de LQI obtido por cada dispositivo para essa rota. O valor de LQI varia entre 0 e 255, sendo 0 o pior e 255 a melhor condição. Na Figura o coordenador está no centro, à esquerda o Roteador A e a direita o Roteador B. Pode-se visualizar que a rota Coordenador - Roteador A está boa, com os valores de LQI próximos de 255 em ambos os sentido. O mesmo vale para a rota entre Roteador A e Roteador B com ótimos valores de LQI em ambos os sentidos. Já a rota entre Coordenador e Roteador B, pode-se verificar que em um dos sentidos está bem ruim com o valor de LQI aproximadamente de 48. O Coordenador ao perceber isso, irá mudar sua rota utilizando o Roteador A para chegar até o Roteador B. Isso reforça as vantagens de se utilizar uma rede com topologia *mesh*. Cabe ressaltar que essa figura foi obtida sem circulação de trens, ou seja, não havia interferência provenientes de cruzamento de trens ou circulação deles.

Figura 91 – Arquitetura da rede obtida pelo *software* XCTU.



Fonte: Autor.

Não foi possível realizar a medida de SNR devido a falta de um analisador de espectro no local dos testes.

6.5 Teste de *sniffer*

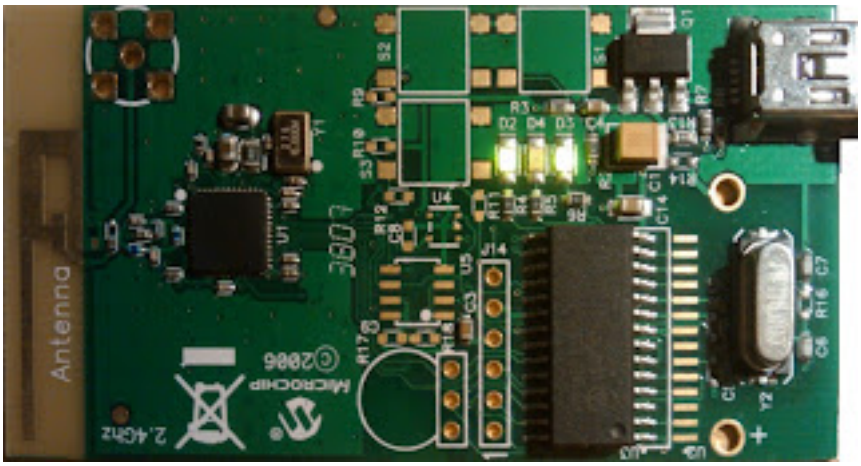
O teste *sniffer* tem a função de visualizar os pacotes que estão sendo transmitidos no ar, visualizar a montagem dos *frames* e a disposição do pacote completo de RF. Para realizar essa análise, utilizamos o kit ZENA *Wireless Networtk Analyzer*.

6.5.1 ZENA *Wireless Networtk Analyzer*

Segundo o fabricante, o analisador ZENA fornece três ferramentas principais para desenvolver soluções IEEE 802.15.4 de forma rápida e eficiente com o Microchip *Stack* gratuito para o protocolo ZigBee. O analisador ZENA permite que os desenvolvedores modifiquem e adaptem rapidamente o *Microchip Stack* para o protocolo ZigBee para atender aos requisitos da aplicação.

O analisador ZENA também é um analisador de pacotes IEEE 802.15.4 que atualmente suporta o espectro de 2,4 GHz. O analisador ZENA é capaz de decodificar pacotes do protocolo ZigBee v1.0. O analisador ZENA também fornece suporte à análise de rede do protocolo ZigBee. O analisador desenha a topologia de rede do protocolo ZigBee à medida que ela é formada, permite que os usuário observem as transações de pacotes e reproduzam esses pacotes em velocidades variáveis. A Figura 92 apresenta o *hardware* utilizado para os testes.

Figura 92 – *Hardware* ZENA.



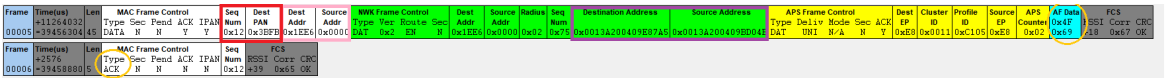
Fonte: (DESBONNET, 2013).

O primeiro teste realizado com o *sniffer*, foi a de visualização de pacotes enviados de um XBee, para outro XBee. Para facilitar a realização do teste, foram conectados os 3 dispositivos (XBees) pela porta USB (*Universal Serial Bus*) ao computador e foi conectado também o *hardware* do ZENA ao computador pela USB. Usamos o *software* XCTU para controlar os pacotes enviados pelos dispositivos e o ZENA para monitorar o tráfego no ar. Importante destacar, que o ZENA, em nenhum momento ingressa na rede dos XBees, ele

apenas consegue capturar os pacotes no ar e por conhecimento do protocolo, indicar os dados dentro do pacote.

A Figura 93 apresenta o pacote capturado quando houve uma transmissão do Coordenador para o Roteador A (*frame* 5), cujo *payload* (AF Data) é "Oi"(0x4F (O) e 0x69 (i) em hexadecimal destacado no círculo laranja no *frame* 005) e em seguida houve o envio do ACK pelo Roteador A (*frame* 006 com o destaque laranja no tipo ACK).

Figura 93 – Pacote de transmissão com conteúdo do *payload* "Oi".

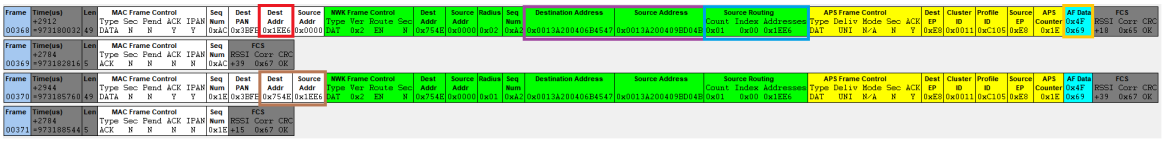


Fonte: Autor.

Pode-se observar que dados interessantes da rede são facilmente capturados pelo ar. Dados como endereço 64 *bits* do destino e remetente (destacado em roxo), endereço de 16 *bits* do remetente e destinatário (destacado em rosa), endereço da rede PAN e sequência do *frame* (destacado em vermelho).

Outro teste interessante, é o envio do mesmo pacote, só que agora do Coordenador para o Roteador B, porém utilizando a rota Coordenador - Roteador A - Roteador B, para mostrar a arquitetura *mesh* da rede. A Figura 94 apresenta os pacotes capturados pelo ZENA.

Figura 94 – Exemplo do uso de rotas na rede *mesh*.



Fonte: Autor.

Pela Figura 94, pode-se visualizar no *frame* 368 que a origem do pacote cujo o endereço de 64 *bits* origem é o Coordenador (final D04B) e o destino final é o Roteador B (final 4547) (ambas destacada em roxo), porém mais para o início do *frame*, pode-se verificar que o endereço de destino de 16 *bits* é o 0x1EE6 (destacado em vermelho). Esse endereço de 16 *bits* pertence ao Roteador A (o endereço do roteador B é 0x754E). Outra observação importante também está na parte *Source Routing* (destacado em azul), onde existe um contador e um índice, que são as informações da rota, ele já nos indica que o primeiro "salto"da rota é o endereço 0x1EE6 (Roteador A). Pode-se perceber que o *payload* (AF Data) que é 0x4F e 0x69 ("Oi") (destacado em laranja). Após o envio desse *frame*, pode-se visualizar o ACK enviado pelo Roteador A no *frame* 369. Sabe-se que esse ACK é para o *frame* 368, pois é um *frame* de ACK, cujo a sequência numérica é igual a do pacote enviado.

O Roteador A, após receber o pacote do Coordenador e enviar a ele o ACK, abre o pacote, verifica a integridade do pacote e verifica o próximo salto. Nesse caso, não há outro

salto, pois o próximo endereço já é o destino final. Então ele envia um novo pacote para o destino final, que é o *frame* 370, onde pode-se visualizar que os dados em "verde", são os mesmos do *frame* 368, porém os dados de endereço de 16 *bits* são diferente (destacados em marrom), sendo que o remetente é 0x1EE6 (Roteador A) e o destino é 0x754E (Roteador B). Quando o Roteador B recebe esse pacote, ele emite um ACK para o Roteador A (*frame* 371), finalizando a transmissão.

Como foi apresentado acima, é fácil capturar os pacotes, mesmo não estando dentro da rede. Para melhorar a segurança da rede, os XBee podem aplicar criptografia no envio dos dados. Para realizar isso, é necessário colocar o parâmetro EE em 1 (Criptografia AES 128 bits habilitada), após realizar isso em todos, é possível escolher o valor em hexadacimal de 128 *bits* da chave de rede (parâmetro NK). Essa opção só é possível escolher no coordenador. Caso o valor seja "0", o coordenador vai gerar uma senha aleatória (modo recomendado pelo fabricante), caso o valor seja diferente de "0", será utilizado o valor informado.

Realizado as configurações, o Coordenador enviará a chave de rede para todos que ingressarem na rede. A Figura 95 apresenta esse envio da chave (destacado em marrom). A chave utilizada nesse exemplo foi do valor 0x30000000000000000000000000000000 em hexadecimal.

Figura 95 – Exemplo do envio da chave de rede para ingressantes pelo Coordenador.

Frame	Time(us)	Len	MAC Frame Control						Seq Num	Dest PAN	Dest Addr	Source Addr	NWK Frame Control						Dest Addr	Source Addr	Radius	Seq Num	APS Frame Control					APS Counter	APS Command	Key Type	Key	SN	Destination Address	Source Address	FCS	
			Type	Sec	Pend	ACK	IPAN						Type	Ver	Route	Sec							Type	Deliv	Mode	Sec	ACK								RSSI	Corr	CRC
00087	+4944 =3545584432	56	DATA	N	N	Y	Y		0x9C	0x000F	0x32F2	0x0000	DAT	0x2	SUP	N			0x32F2	0x0000	0x02	0x20	CMD	UNI	N/A	N	N	0x05	TransKey	NWK	0x30000000000000000000000000000000	0x00	0x0013A20040B4547	0xFFFFFFFFFFFFFFFF	+19	0x60	OK
Frame	Time(us)	Len	MAC Frame Control						Seq Num	FCS																									
			Type	Sec	Pend	ACK	IPAN			RSSI	Corr	CRC																							
00088	+3126 =3545615685	5	ACK	N	N	N	N		0x9C	+08	0x66	OK																							

Fonte: Autor.

Para contornar esse problema em que é possível visualizar a chave de rede enviada, existe um parâmetro dentro dos dispositivos XBee chamado KY. Caso esse valor seja diferente de zero, esse valor deve ser configurado igual em todos os dispositivos, assim, o coordenador continuará enviando a chave de rede, porém agora encriptado utilizando o valor do parâmetro KY, isso impede que qualquer um consiga identificar a chave de rede e ingressar na rede. A figura 96 apresenta o envio do pacote com *payload* "Oi"(0x4F 0x69) do Coordenador para o Roteador A, porém com a criptografia aplica. Pode-se verificar que não é possível obter os dados do *payload*, sem saber da chave de rede.

Figura 96 – Exemplo do envio de pacote com criptografia aplicada.

Frame	Time(us)	Len	MAC Frame Control						Seq Num	Dest PAN	Dest Addr	Source Addr	NWK Frame Control						Dest Addr	Source Addr	Radius	Seq Num	Destination Address	Source Address	Security Control					Frame Counter	Source Address	Key SN	Encrypted Data								FCS	
			Type	Sec	Pend	ACK	IPAN						Type	Ver	Route	Sec									ExtN	Key	Lvl													RSSI	Corr	CRC
00003	+13802096 =2019680	63	DATA	N	N	Y	Y		0x40	0xDBC3	0xA801	0x0000	DAT	0x2	ER	Y			0xA801	0x0000	0x02	0xFF	0x0013A200409E8745	0x0013A200409ED04B	Y	NWK	0	0x0000000B	0x0013A200409ED04B	0x00	0x56	0x69	0xA0	0x23	0xE7	0x5A	0xCC	0x25	0xFB	0x0A	0x9A	0x99	0xD6	0xD3	+20	0x60	OK
Frame	Time(us)	Len	MAC Frame Control						Seq Num	FCS																																
			Type	Sec	Pend	ACK	IPAN			RSSI	Corr	CRC																															
00004	+20200400	5	ACK	N	N	N	N		0x40	+39	0x66	OK																															

Fonte: Autor.

6.6 Análise do modelo de propagação de *Shadowing*

6.6.1 Criação do modelo de propagação *Shadowing*

Os modelos de propagação teóricos e baseados em medição indicam que a potência média de um sinal recebido em dB, diminui linearmente com o inverso do logaritmo da distância, seja em ambientes externos ou internos. A perda de caminho média em grande escala para uma separação entre transmissor e receptor qualquer é expressa com uma função da distância usando um expoente de perda de caminho γ (RAPPAPORT, 2009). A equação 6.3 apresenta essa ideia.

$$\overline{PL}(dB) = \overline{PL}(d_0) - 10\gamma \log(\frac{d}{do}) \tag{6.3}$$

Onde γ é o expoente de perda de caminho que indica a taxa com a qual a perda aumenta em relação à distância, d_0 é a distância de referência que é a distância medida mais próxima do transmissor, e d é a distância entre o transmissor e receptor. As barras na equação 6.3 indicam a média conjunta de todos os valores possíveis de perda de caminho para determinado valor de d. Quando desenhada em uma escala log-log, a perda de caminho modelada é uma linha reta com inclinação de 10γ por dezena. Já o valor de γ depende do ambiente de propagação específico, por exemplo, no espaço livre, γ é igual a 2, quando existem obstruções, γ terá uma valor maior.

Segundo (RAPPAPORT, 2009), é importante que a escolha da distância de referência seja apropriada para o ambiente de propagação, pois a distância de referência sempre deve estar no campo distante da antena, de modo que os efeitos de campo próximo não alterem a perda do caminho de referência. O modelo apresentado na equação 6.3 não considera o fato de que o ruído ambiental ao redor pode ser muito diferentes em dois locais distintos tendo a mesma separação entre transmissor e receptor. Isso levará a sinais medidos que são muito diferentes do valor médio previsto pela equação 6.3. As medições têm mostrado que para qualquer valor de d, a perda de caminho $PL(d)$ em determinado local é aleatória e distribuída log-normalmente em torno do valor médio dependente da distância. A equação 6.4 mostra isso, onde X_σ é uma variável aleatória com distribuição gaussiana de média zero (em dB) com desvio padrão σ (também em dB).

$$PL(d)[dB] = PL(d_0) - 10\gamma \log(\frac{d}{d_0}) + X_\sigma \tag{6.4}$$

A distribuição log-normal descreve os efeitos aleatórios do sombreamento, que ocorrem em um grande número de locais medidos que possuem a mesma separação entre transmissor e receptor, mas com diferentes níveis de ruído no caminho de propagação.

Segundo (RAPPAPORT, 2009) a distância de referência d_0, o expoente de perda

de caminho γ descrevem estatisticamente o modelo de perda de caminho para um local qualquer tendo separação entre transmissor e receptor e esse modelo pode ser usado em simulações para fornecer níveis de potência recebida para locais aleatórios no projeto e análise do sistema de comunicação. Na prática, os valores de γ e σ são calculados a partir dos dados medidos, usando a regressão linear de modo que a diferença entre as perdas de caminho medida e estimada são minimizadas para a média do erro quadrático em relação a grande faixa de locais medidos e separações de transmissor e receptor. O valor de $\overline{PL}(d_0)$ é baseado nas medições próximas ou em uma suposição de espaço livre a partir do transmissor até d_0.

Será aplicado o modelo apresentado nesta seção para os dados obtidos de medições nos ambientes *indoor, outdoor* e ferroviário e após a criação do modelo, serão comparados os dados obtidos e os do modelo gerado. Para isso foi desenvolvido um código no *software* Matlab, que irá aplicar o algoritmo descrito anteriormente com as medidas de cada ambiente. Então será obtida uma estimativa do valor de γ e σ para cada ambiente.

6.6.1.1 Ambiente *indoor*

Foram utilizados os dados obtidos para RSSI no ambiente *indoor* e o algoritmo nos retornou os valores para γ e σ apresentado na tabela 2.

Tabela 2 – Valores do modelo de propagação *Shadowing* no ambiente *indoor*.

γ	σ
$1,7992$	$3,8950$

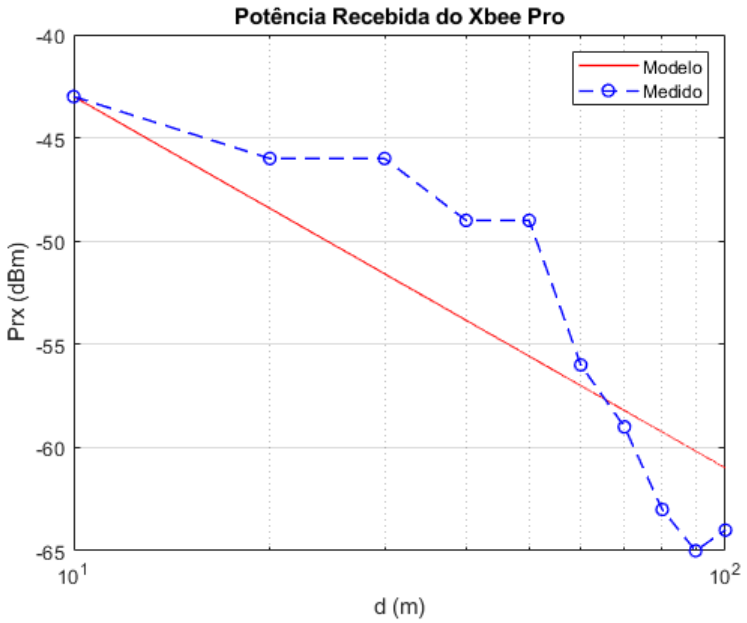
A Figura 97 apresenta o resultado obtido para a P_{RX} medida e para a P_{RX} média obtida pelo MPS (Método de Propagação de *Shadowing*). Analisando os resultados, é possível verificar que a P_{RX} medida apresenta algumas variações ao longo de d, fundamentalmente causadas pelos cancelamentos das componentes de multipercurso da ordem de múltiplos de meio comprimento de onda do sinal transmitido. Já a figura 98 mostra os efeitos aleatórios de *Shadowing* sobre a curva da P_{RX} média obtida pelo MPS (ajustado a partir dos valores de P_{RX} medidos) para o ambiente. Como não foram realizadas medidas com distâncias significativamente maiores que a distância crítica (para esse caso 128 metros), não é possível uma estimativa precisa da distância máxima para o ambiente Indoor.

6.6.1.2 Ambiente *Outdoor*

Foram utilizados os dados obtidos para RSSI no ambiente *outdoor* e o algoritmo nos retornou os valores para γ e σ apresentado na tabela 3.

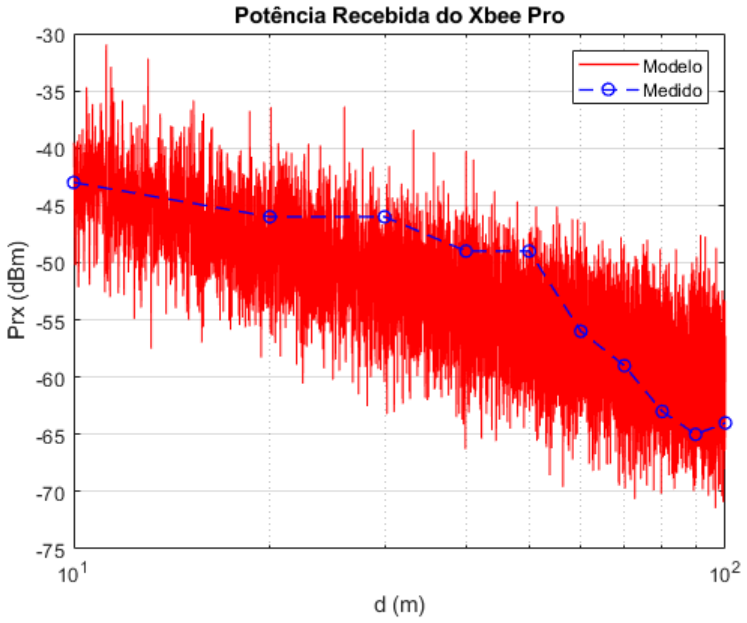
A figuras 99 apresenta o resultado obtido para a P_{RX} medida e para a P_{RX} média obtida pelo MPS. Pode-se ver que o ajuste se aproximou bastante das medições realizadas.

Figura 97 – Potência média medida no ambiente *indoor* em função da distância e a obtida pelo MPS correspondente.



Fonte: Autor.

Figura 98 – Potência média medida no ambiente *indoor* em função da distância e a potência instantânea estimada pelo MPS correspondente, considerando a dispersão causada pelo efeito de *shadowing*.



Fonte: Autor.

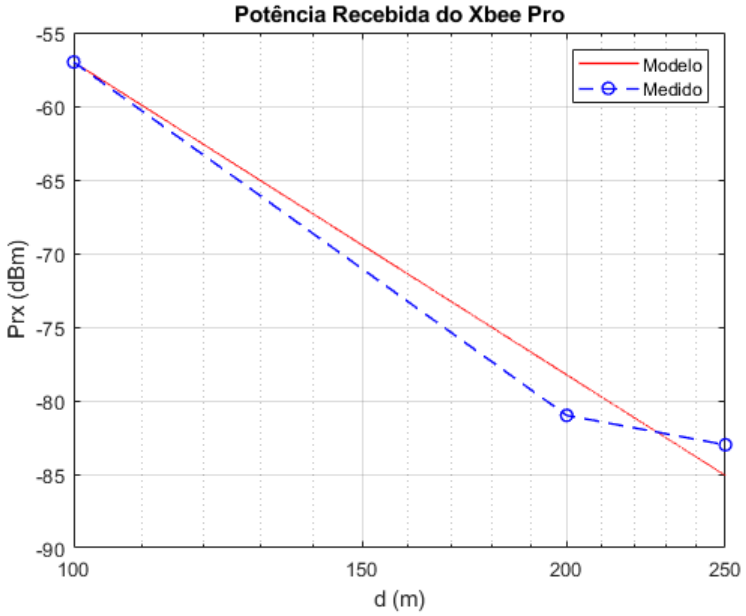
Tabela 3 – Valores do modelo de propagação *Shadowing* para o ambiente *outdoor*.

γ	σ
$7,0574$	$1,9945$

Utilizando o MPS, pode-se obter uma estimativa da distância para a sensibilidade máxima do *XBee* (-102 dBm) de aproximadamente 414 m, o que é mais um indicativo do fato de

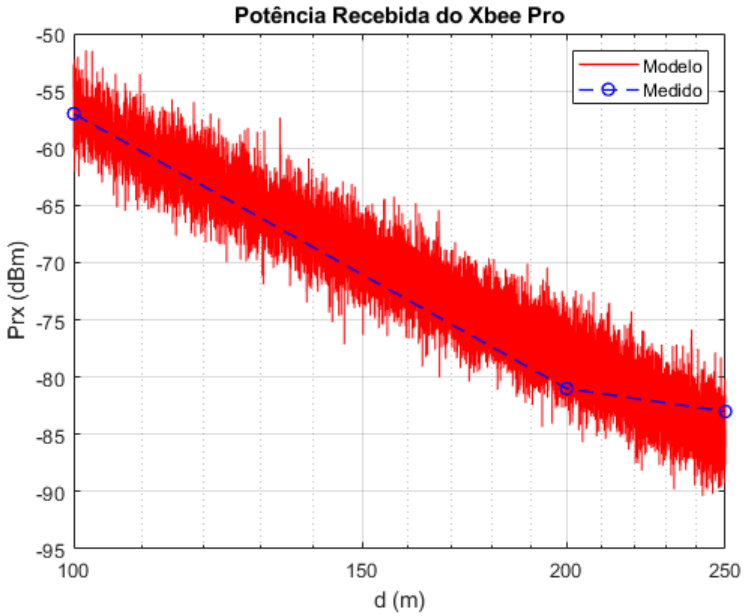
não conseguir obter pacotes a distância de 300 m foi devido a falta de visada. Já a figura 100 mostra os efeitos aleatórios de *Shadowing* sobre a curva da P_{RX} média obtida pelo MPS (ajustado a partir dos valores de P_{RX} medidos) para o ambiente.

Figura 99 – Potência média medida no ambiente *outdoor* em função da distância e a obtida pelo MPS correspondente.



Fonte: Autor.

Figura 100 – Potência média medida no ambiente *outdoor* em função da distância e a potência instantânea estimada pelo MPS correspondente, considerando a dispersão causada pelo efeito de *shadowing*.



Fonte: Autor.

6.6.1.3 Ambiente Ferroviário

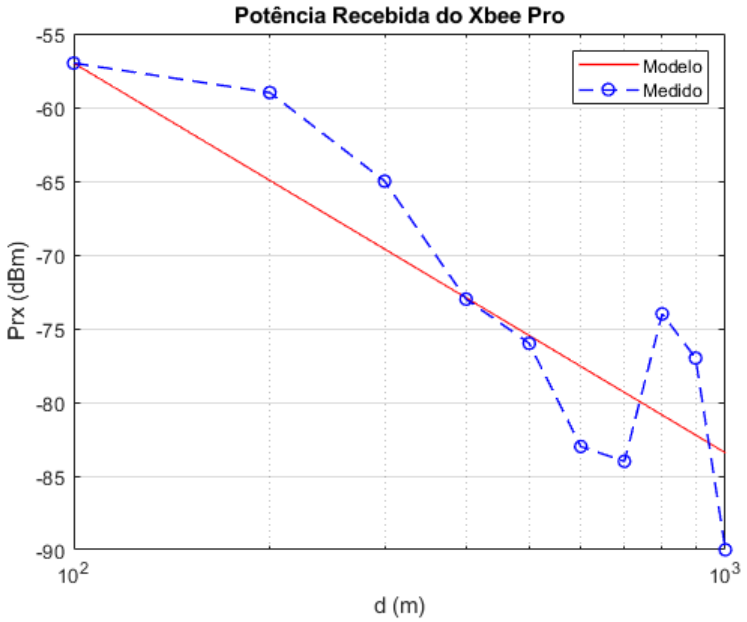
Foram utilizados os dados obtidos para RSSI no ambiente ferroviário e o algoritmo nos retornou os valores para γ e σ apresentado na tabela 4.

Tabela 4 – Valores do modelo de propagação *Shadowing* para o ambiente ferroviário.

γ	σ
$2,6427$	$4,7528$

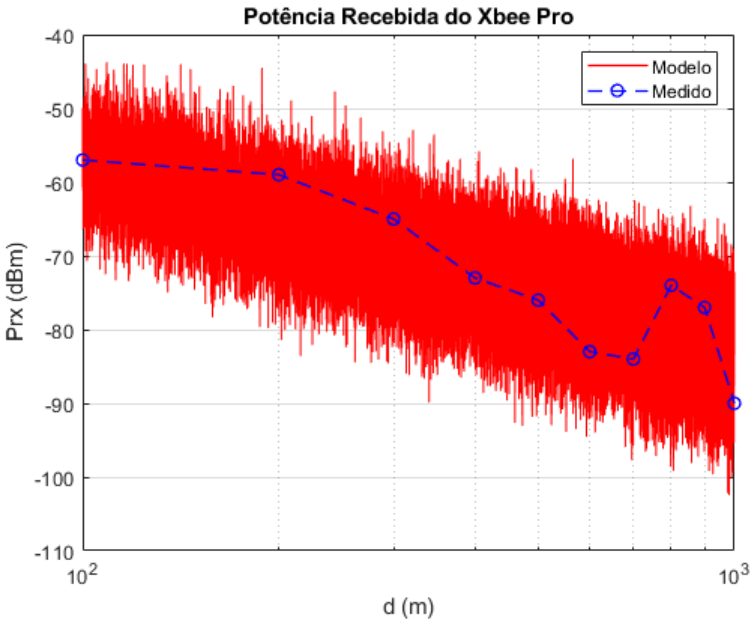
A figuras 101 apresenta o resultado obtido para a P_{RX} medida e para a P_{RX} média obtida pelo MPS. Pode-se ver que o ajuste não se aproximou muito das medições realizadas, pois é possível perceber que a atenuação pela distância é bem variável, sendo bem alta em alguns pontos (como -8 dB entre 300 e 400 m e -1 dB entre 600 e 700 m) . Utilizando o MPS, pode-se obter uma estimativa da distância para a sensibilidade máxima do *XBee* (-102 dBm) uma vez que a distância crítica é de 280 metros e foram realizadas pelo menos 8 medidas acima dessa distância. A distância estimada pelo MPS foi de aproximadamente 5 km. Muito provavelmente ele não chegará a essa distância, pois o próprio fabricante informa que a distância máxima com visada direta é de 3,2 km. Porém a distância de 1 km já é muito significativa para o ambiente ferroviário, devido às disposições dos equipamentos de sinalização. Já a Figura 102 mostra os efeitos aleatórios de *Shadowing* sobre a curva da P_{RX} média obtida pelo MPS (ajustado a partir dos valores de P_{RX} medidos) para o ambiente.

Figura 101 – Potência média medida no ambiente ferroviário em função da distância e a obtida pelo MPS correspondente.



Fonte: Autor.

Figura 102 – Potência média medida no ambiente ferroviário em função da distância e a potência instantânea estimada pelo MPS correspondente, considerando a dispersão causada pelo efeito de *shadowing*.



Fonte: Autor.

7 Considerações finais

Após a análise dos resultados, o sistema embarcado de comunicação baseado em ZigBee proposto nesse trabalho demonstrou que atendeu os requisitos previstos, pois ele foi capaz de realizar o envio e recepção de dados, mesmo em cenário críticos com muita interferência. Foi capaz de processar os pacotes, verificando sua integridade e o tipo de pacote, o que demonstra que mesmo sem realizar testes práticos nos equipamentos de sinalização, ele facilmente poderia controlar e verificar posicionamento dos equipamentos, uma vez que só seria necessária a utilização de interfaces como optoacopladores para indicação, isolando eletricamente os equipamentos do nosso protótipo e a utilização de relês para acionamento de cargas. Assim só seria necessária a aplicação do protocolo de comunicação utilizado pela sinalização. Ele realizaria essas tarefas, pois o mais difícil seria a leitura e interpretação de cada pacote que chega, e isso foi executado de forma precisa.

A topologia *mesh* utilizada demonstrou que pode melhorar significativamente a recepção de um dispositivo, pois durante os testes foi possível verificar uma melhora de até 10 dB na RSSI, demonstrando sua importância na utilização da melhor rota, além de permitir a expansão da rede, pois permite comunicação entre dois dispositivos que não estão no campo de transmissão um do outro, mas possuem vizinhos que podem gerar uma rota entre eles. Entretanto sua utilização aumenta a latência da rede pois quanto mais dispositivos na rota, maior será o tempo para chegada do pacote enviado. Essa latência para a aplicação na sinalização não é tão crítica uma vez que os dados são transmitidos para o CCO em média entre 8 a 10 segundos.

O sistema proposto demonstrou que pode gerar economia de dezenas de quilômetros de cabo de cobre, evitando assim problemas com furtos e vandalismos, além de problemas como baixa isolação e interrupção dos condutores. Permite uma fácil expansão de novos equipamentos, uma vez que só é necessário levar até o equipamento os cabos de alimentação que podem ser comum a todos os equipamentos de sinalização.

Seria interessante para trabalhos futuros, a análise e projeto de antenas que possam atender esses dispositivos da melhor forma, pois a antena embutida na plataforma *XBee*, é uma antena com ganho menor que unitário, e não houve nenhuma alteração dessas durante os testes. Com um projeto de antena adequada, é possível melhorar significativamente a comunicação entre os dispositivos, melhorando a recepção e diminuindo a quantidade de pacotes perdidos e retransmissões.

Referências

ALLIANCE, C. S. **CSA - IOT - *Connectivity Standards Alliance***. [S.l.], 2022. Disponível em: <https://csa-iot.org>. Acesso em: 22 out 2022.

ANTF, A. dos T. F. **Mapa ferroviário no Brasil**. [S.l.], 2022. Disponível em: <https://www.antf.org.br/mapa-ferroviario/>. Acesso em: 06 ago 2022.

ARDUINO.CC. **Circuito elétrico Arduino MEGA 2560 Rev. 3**. [S.l.], 2018. Disponível em: <https://content.arduino.cc/assets/MEGA2560_Rev3e_sch.pdf>. Acesso em: 28 fev 2022.

ATMEL. ***Datasheet*: ATmega640/V-1280/V-1281/V-2560/V-2561/V**. [S.l.], 2014.

BAZANI, A.; MARQUES, J. **Furtos e roubos de cabos da CPTM e do Metrô**. [S.l.], 2018. Disponível em: <https://diariodotransporte.com.br/2018/10/17/furtos-e-roubos-de-cabos-da-cptm-e-do-metro-causaram-prejuizos-de-r-900-mil-em-oito-meses/>. Acesso em: 20 jan 2020.

BRONZATTI, L. F. C. **Análise sobre a tecnologia de rede sem fio Zigbee/IEEE 802.15.4**. Monografia (Graduação) — Universidade de São Paulo, São Carlos, São Paulo, Brasil, Junho 2013.

BUENO, S. **Sinalização Ferroviária CPTM**. São Paulo, São Paulo, Brasil, 2006.

BURI, M. R.; FABRETI, M. P.; OLIVEIRA, E. R. de; SILVA, M. M. da. Transporte ferroviário de cargas no brasil - aproveitamento da malha. **XII SIMPEP**, Bauru, São Paulo, Brasil, 2006.

CIA, A. e. **Como usar módulo cartão micro SD Arduino**. [S.l.], 2020. Disponível em: <https://www.arduinoecia.com.br/como-usar-modulo-cartao-micro-sd-arduino/>. Acesso em: 02 abr 2022.

COSTA, R. A. A.; MENDES, L. A. M. **Evolução das Redes Sem Fio:**: Um estudo comparativo entre bluetooth e zigbee. Barbacena, Minas Gerais, Brasil, 2006.

DESBONNET, J. ***Using the Microchip ZENA ZigBee/802.15.4 network analyzer with Linux***. [S.l.], 2013. Disponível em: <http://jdesbonnet.blogspot.com/2011/02/using-microchip-zena-zigbee802154.html>. Acesso em: 05 abr 2022.

DIGI. **XBee/XBee-PRO Zigbee RF *Modules User Guide***. [S.l.], 2018.

FERREIRA, M. P.; CANTARINO, A. Transporte de passageiro no brasil: Análise e comentário de um estudo de caso à luz da responsabilidade social corporativa. **Estudos tecnológicos**, Niterói, Rio de Janeiro, Brasil, v. 2, n. 2, p. 113–130, Julho 2006.

FRANZÃO, A. A. **Análise estatística das ocorrências de acidentes ferroviários na região Centro-sudeste Paulista, causa e consequências**. Monografia (Graduação) — Universidade Federal de Uberlândia, Uberlândia, Minas Gerais, Brasil, 2018.

GAMMON, N. ***Introdution on SPI***. [S.l.], 2015. Disponível em: <http://www.gammon.com.au/spi>. Acesso em: 30 mar 2022.

GUERRERO, J. C. ***Caminar con éxito hacia la Industria* 4.0**. [S.l.], 2018. Disponível em: <https://ticnegocios.camaravalencia.com/servicios/tendencias/caminar-con-exito-hacia-la-industria-4-0-capitulo-11-infraestructuras-i-redes-inalambricas/>. Acesso em: 7 dez 2022.

GUIMARÃES, F. **I2C**. [S.l.], 2018. Disponível em: <https://mundoprojetado.com.br/i2c/>. Acesso em: 20 out 2021.

HAYKIN, S.; MOHER, M. **Sistemas de Comunicação**. 5. ed. [S.l.]: Bookman Editora, 2011.

IEEE. Ieee standard for low-rate wireless networks. **IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)**, p. 1–709, 2016.

ILUSTRAÇÃO do transporte da locomotiva ou do vagão e do metro. [S.l.], 2018. Disponível em: <https://pt.dreamstime.com/treine-o-grupo-da-ilustra\%C3\%A7\%C3\%A3o-do-transporte-locomotiva-ou-vag\%C3\%A3o-e-metro-railway-vetor-de-transport\%C3\%A1vel-image119304051#_>. Acesso em: 30 ago 2022.

JANGRA, S.; KUMAR, R. *Digital Modulation Techniques Using MATLAB Script file and Graphical Approach*. In: . [S.l.: s.n.], 2020.

KAORU, M. **ZigBee usa agora 6loWPAN**. [S.l.], 2013. Disponível em: <https://ipv6.br/post/zigbee-usa-agora-6lowpan-sua-proxima-lampada-tera-ipv6/>. Acesso em: 13 fev 2022.

KLINCEVICIUS, M. G. Y. **Estudo de propriedades, de tensões e do comportamento mecânico de lastros ferroviários**. 12-16 p. Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo, São Paulo, 2011.

LUTKEVICH, B. ***Embedded System***. [S.l.], 2020. Disponível em: <https://www.techtarget.com/iotagenda/definition/embedded-system>. Acesso em: 24 fev 2022.

MADEIRA, D. **RTC DS3231 - Registrando data e hora com arduino**. [S.l.], 2017. Disponível em: <https://portal.vidadesilicio.com.br/real-time-clock-rtc-ds3231/>. Acesso em: 02 abr 2022.

MARGOLIS, M. **Arduino cookbook: recipes to begin, expand, and enhance your projects**. [S.l.]: O'Reilly Media, 2011.

MELO, P. **Padrão IEEE 802.15.4 - A base para as especificações Zigbee, WirelessHart e MiWi**. [S.l.], 2017. Disponível em: <https://www.embarcados.com.br/padrao-ieee-802-15-4>. Acesso em: 15 fev 2022.

MIGUEL, C. D.; COSTA, P. F. N. da; MORALES, R. T.; KRONEMBERGER, W. **Sistemas de Sinalização**. 3. ed. São Paulo, São Paulo, Brasil, 2018.

NOERGAARD, T. **Embedded systems architecture: a comprehensive guide for engineers and programmers**. Oxford: Newnes, 2012. 672 p.

NUNES, C. **Utilização de métodos de análise de falhas em um sistema de sinalização ferroviária**. Monografia (Especialização) — Instituto Militar de Engenharia, Rio de Janeiro, Brasil, 2012.

NUNES, E. F.; CAPPELLI, N. L.; UMEZU, C. K. Avaliação da propagação de sinais de radiofrequência para tecnologia zigbee em granja de frango de corte. **Revista Brasileira de Engenharia Agrícola e Ambiental**, SciELO Brasil, v. 15, n. 1, p. 102–107, 2011.

OLIVEIRA, R. W. H. **Caracterização da escória de ferro silício-manganês para a aplicação como agregado em pavimentação ferroviária.** Dissertação (Mestrado) — Universidade Federal de Ouro Preto, Ouro Preto, 2013.

OLIVEIRA, T. de A. **Redes dinâmicas de sensores sem fio Zigbee para aplicações de monitoramento e controle**. Dissertação (Mestrado) — Faculdade de Engenharia de Bauru Unesp, Bauru, SP, 2015.

RAPPAPORT, T. S. **Comunicações sem Fio: Princípios e Práticas.** 2. ed. [S.l.]: Pearson, 2009.

ROCHA, F. B.; SILVA, R. S.; AVELINO, A. M.; COSTA, C. M. Plataforma de comunicação sem fio aplicada a sistemas de irrigação. **Holos**, Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, v. 5, p. 260–273, 2014.

ROMEIRO, W.; COSTA, F. Monitoramento residencial utilizando o zabbix e o padrão ieee 802.15. 4. **Holos**, Instituto Federal do Rio Grande do Norte (IFRN), v. 2016, n. 1, p. 253–263, 2016.

SOUZA, F. **Arduino MEGA 2560**. [S.l.], 2014. Disponível em: <https://embarcados.com.br/arduino-mega-2560/>. Acesso em: 24 fev 2022.

STALLINGS, W. ***Wireless Communications and Networks***. 2. ed. [S.l.]: Prentice Hall, 2005.

UFRJ. **Zigbee**. [S.l.], 2016. Disponível em: <https://www.gta.ufrj.br/grad/10_1/zigbee/padrao.html>. Acesso em: 18 fev 2022.

VICENTINI, R. M. Medidas, modelagem e análise computacional de sistemas zigbee sob interferência de redes wifi. **INDUSCON**, v. 13, p. 1595–1600, 2018.

Anexos

ANEXO A – Modulação QPSK e OQPSK

A.1 QPSK

O chaveamento por deslocamento de fase quartenário (*Quaternary Phase Shift Keying* - QPSK) tem o dobro da eficiência de largura de banda do BPSK, pois dois *bits* são transmitidos em um único símbolo de modulação. No QPSK, a fase da portadora assume um de quatro valores possíveis como, por exemplo, 0, $\frac{\pi}{2}$, π, e $\frac{3\pi}{2}$, porém podem ser utilizados outros valores de fase. Cada valor de fase corresponde a um par exclusivo de *bits* do sinal de informação. Utilizando os valores de fase mencionados nesse parágrafo, o sinal QPSK ($s_{QPSK}(t)$) pode ser definido como o apresentado na equação A.1, onde T_s é a duração do símbolo e é igual ao dobro do período de *bits*, ou seja, igual a $2T_b$, E_s é a energia de símbolo e f_c é a frequência da portadora

$$\begin{aligned} s_{QPSK}(t) &= \sqrt{\frac{2E_s}{T_s}} \cos\left(2\pi f_c t + (i-1)\frac{\pi}{2}\right) \\ &= 0 \leq t \leq T_s \quad i = 1, 2, 3, 4 \end{aligned} \tag{A.1}$$

Usando identidades trigonométricas, a equação A.1 pode ser rescrita para o intervalo $0 \leq t \leq T_s$ como apresentado na equação A.2.

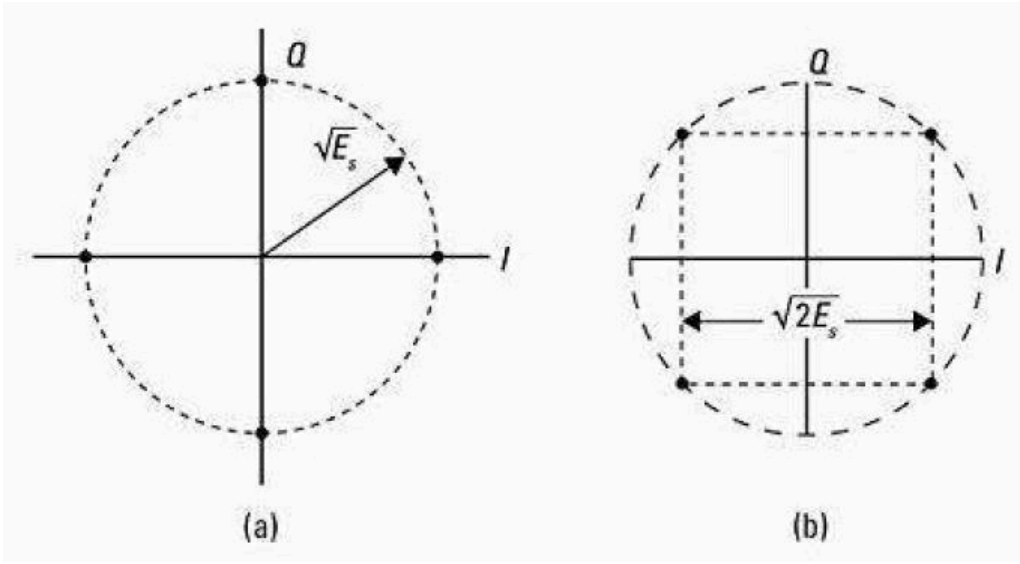
$$s_{QPSK}(t) = \sqrt{\frac{2E_s}{T_s}} \cos\left[(i-1)\frac{\pi}{2}\right] \cos\left(2\pi f_c t\right) - \sqrt{\frac{2E_s}{T_s}} \sin\left[(i-1)\frac{\pi}{2}\right] \sin\left(2\pi f_c t\right) \tag{A.2}$$

Se as funções de base $\phi_1(t) = \sqrt{\frac{2}{T_s}} \cos\left(2\pi f_c t\right)$ e $\phi_2(t) = \sqrt{\frac{2}{T_s}} \sin\left(2\pi f_c t\right)$ forem definidas sobre o intervalo $0 \leq t \leq T_s$ para o conjunto de sinais QPSK, então os quatro sinais no conjunto podem ser expressos em termos dos sinais base como apresentado na equação A.3.

$$\begin{aligned} s_{QPSK}(t) = &\left\{ \sqrt{E_s} \cos\left[(i-1)\frac{\pi}{2}\right] \phi_1(t) - \sqrt{E_s} \sin\left[(i-1)\frac{\pi}{2}\right] \phi_2(t) \right\} \\ &i = 1, 2, 3, 4 \end{aligned} \tag{A.3}$$

Com base nessa representação, um sinal QPSK pode ser ilustrado usando-se um diagrama de constelação bi-dimensional com quatro pontos, conforme a Figura 103(a) (QPSK1). Deve-se observar que diferentes conjuntos de sinais QPSK podem ser derivados simplesmente girando-se a constelação. Como um exemplo, a Figura 103(b) mostra outro

Figura 103 – (a) Constelação QPSK1. (b)Constelação QPSK2



Fonte: (RAPPAPORT, 2009)

sinal QPSK (QPSK2), onde os valores de fase são $\pi/4$, $3\pi/4$, $5\pi/4$ e $7\pi/4$ (RAPPAPORT, 2009).

A partir do diagrama de constelação de um sinal QPSK apresentado na Figura 103, pode se verificar que a distância entre dois símbolos adjacentes é $\sqrt{2E_s}$. Como cada símbolo corresponde a dois *bits*, então $E_s = 2E_b$, de modo que a distância entre dois pontos adjacentes na constelação QPSK é igual a $2\sqrt{E_b}$. Temos então que a probabilidade de erro de *bit* considerando um canal AWGN é calculada através da equação A.4, onde N_0 é a densidade espectral de potência unilateral do ruído.

$$P_{e,QPSK} = Q\left(\sqrt{\frac{2Eb}{N_0}}\right) \tag{A.4}$$

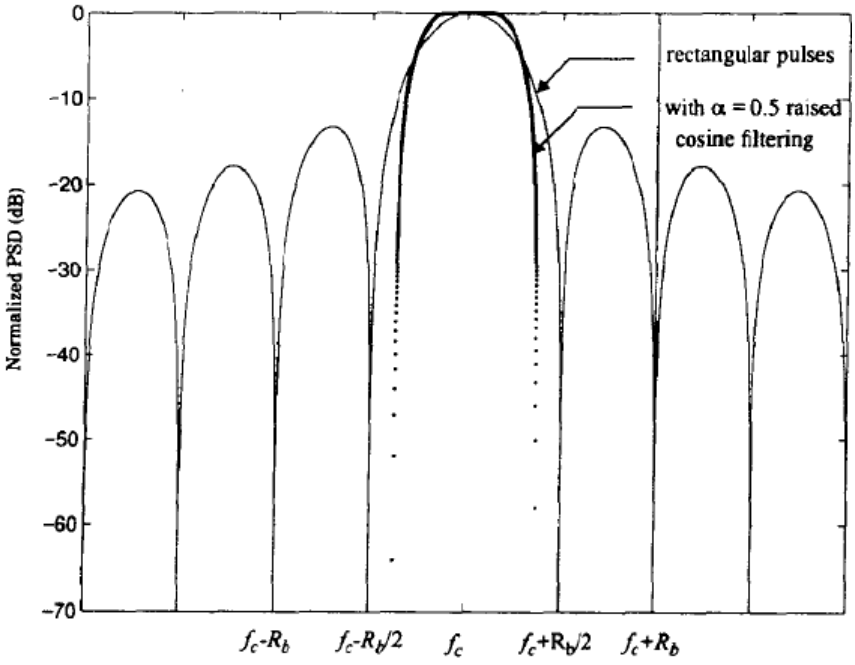
Um resultado interessante é que a probabilidade de erro de bit do QPSK é igual à da modulação BPSK, mas o dobro de dados pode ser enviado na mesma largura de banda. Importante ressaltar também que se considerarmos uma mesma relação E_b/N_0, a relação sinal-ruído (SNR) do QPSK será metade do BPSK.

A PSD de um sinal QPSK usando um sinal de informação com formatação de pulsos retangulares pode ser expressa como apresentado na equação A.5

$$\begin{aligned} P_{QPSK}(f) &= \frac{E_s}{2}\left[\left(\frac{\sin\pi(f-f_c)T_s}{\pi(f-f_c)T_s}\right)^2 + \left(\frac{\sin\pi(-f-f_c)T_s}{\pi(-f-f_c)T_s}\right)^2\right] \\ &= E_b\left[\left(\frac{\sin\pi(f-f_c)T_s}{\pi(f-f_c)T_s}\right)^2 + \left(\frac{\sin\pi(-f-f_c)T_s}{\pi(-f-f_c)T_s}\right)^2\right] \end{aligned} \tag{A.5}$$

A PSD de um sinal QPSK para o sinal de informação com formatação de pulsos retangulares e com formatação de pulsos filtrados com cosseno elevado com coeficiente $\alpha = 0,5$ é representada graficamente na Figura 104. A largura de banda do sinal passa-faixa para formatação de pulsos retangulares é igual à taxa de bits R_b que é a metade daquela de um sinal passa-faixa BPSK.

Figura 104 – Densidade espectral de potência de um sinal QPSK para formatação de pulsos retangulares e pulsos filtrados com cosseno elevado

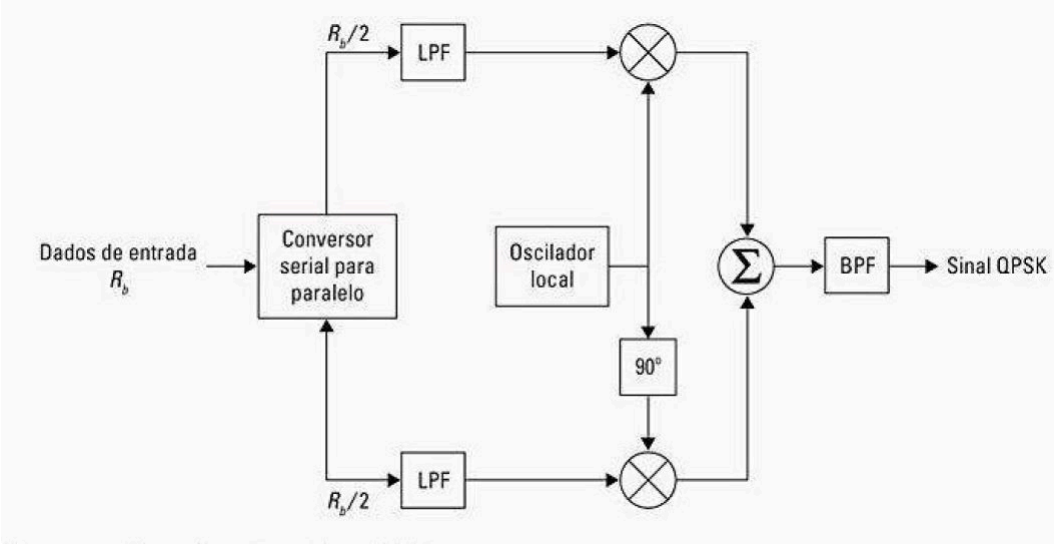


Fonte: (RAPPAPORT, 2009)

A Figura 105 mostra um diagrama de blocos de um transmissor QPSK típico. O fluxo de mensagens binárias unipolares tem taxa de *bits* R_b e esse fluxo de *bits* é convertido para um sinal polarizado sem retorno a zero (NRZ). O fluxo de *bits* $m(t)$ é então dividido em dois fluxos de bits $m_I(t)$ e $m_Q(t)$. Os índices "I"e "Q"representam respectivamente fase (*in phase*) e quadratura (*quadrature*), cada um desses fluxos tendo uma taxa de símbolos de $R_S = R_b/2$. O fluxo de *bits* $m_I(t)$ é chamado fluxo "par", e o $m_Q(t)$ é chamado fluxo "ímpar". As duas sequências binárias são moduladas separadamente por duas portadoras, $\phi_1(t)$ e $\phi_2(t)$, que estão em quadratura. O filtro passa-faixas na saída do modulador confina o espectro de potência do sinal QPSK dentro da banda alocada, o que impede que a potência do sinal transmitido alcance os espectros dos canais adjacentes, além de remover sinais espúrios fora da banda gerados durante o processo de modulação (RAPPAPORT, 2009).

A Figura 106 mostra um diagrama de blocos de um receptor QPSK coerente. O filtro passa-faixas remove o ruído fora da banda e a interferência do canal adjacente. A saída filtrada é dividida em duas partes, e cada parte é demodulada coerentemente usando-se as portadoras em fase e de quadratura. As portadoras coerentes usadas para

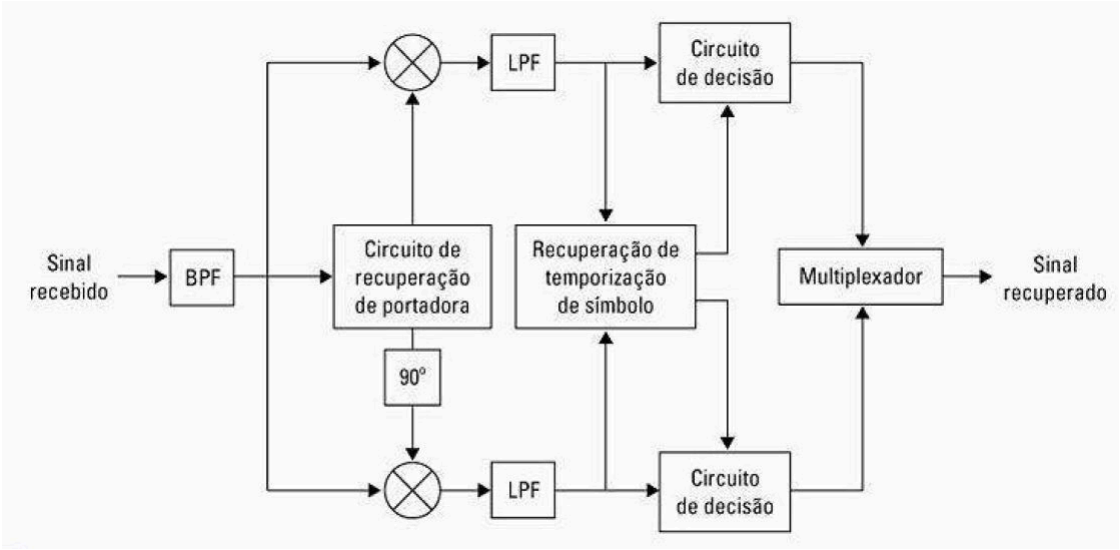
Figura 105 – Diagrama em bloco de um transmissor QPSK



Fonte: (RAPPAPORT, 2009)

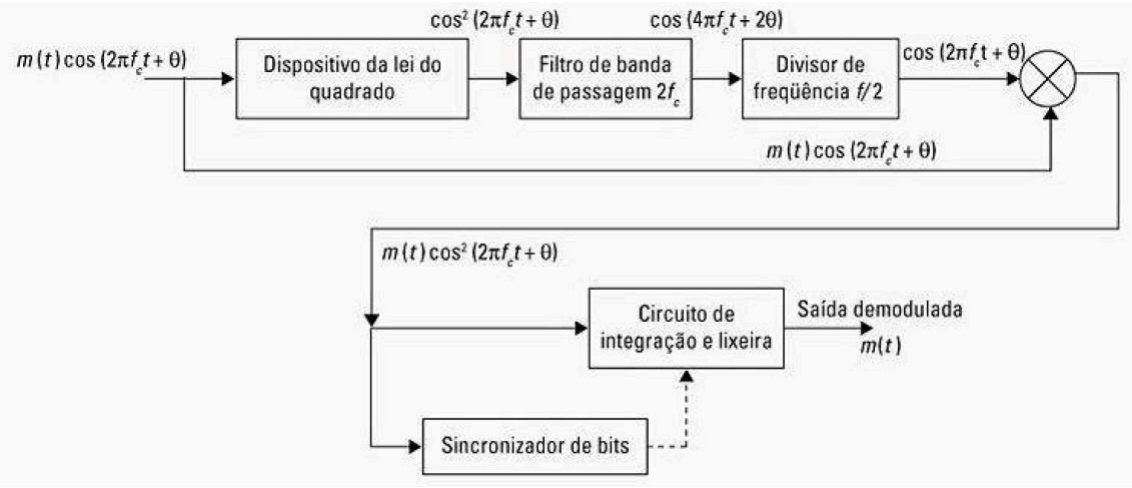
demodulação são recuperadas a partir do sinal recebido usando-se circuitos de recuperação de portadora do tipo descrito na figura 107. As saídas dos demoduladores são passadas por circuitos de decisão que geram os fluxos binários em fase e de quadratura. Os dois componentes são então multiplexados para reproduzir a sequência binária original.

Figura 106 – Diagrama em bloco de um receptor QPSK



Fonte: (RAPPAPORT, 2009)

Figura 107 – Diagrama de blocos do circuito de recuperação



Fonte: (RAPPAPORT, 2009)

A.2 OQPSK

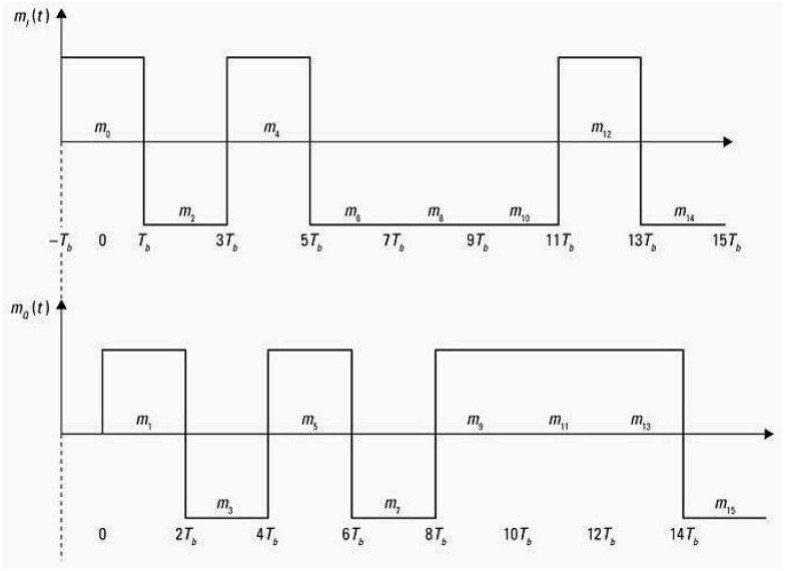
A amplitude de um sinal QPSK é idealmente constante. Porém, quando os sinais de informação da modulação QPSK são em formatação de pulsos, a modulação QPSK perde a propriedade de envelope constante. Caso ocorra uma variação no sinal de informação que gere um deslocamento de fase de π radianos, irá fazer com que o envelope de sinal tenha um cruzamento por zero e isso irá gerar não linearidades. Qualquer tipo de limitação ou amplificação não linear de cruzamento de zero do sinal traz de volta os lóbulos laterais filtrados, pois a fidelidade do sinal em pequenos níveis de tensão é perdida na transmissão. Para impedir essa regeneração dos lóbulos laterais e o alargamento espectral, é imperativo que os sinais QPSK que usam formatação de pulsos sejam amplificados apenas por amplificadores lineares, que são menos eficientes.

Uma forma modificada do QPSK chamada de QPSK Deslocado (*Off-set* QPSK (OQPSK)) ou QPSK Disperso (*Staggered* QPSK) é menos suscetível a esses efeitos nocivos e admite uma amplificação mais eficiente, ou seja, OQPSK garante que ocorram transições de fase limitadas independente das transições do sinal de informação. Isso ajuda a eliminar as não linearidades no processo e o aumento de largura de banda após a amplificação (RAPPAPORT, 2009).

Os sinais da modulação OQPSK são semelhantes aos sinais da modulação QPSK. O OQPSK segue a equação A.2, exceto pela diferença no alinhamento no tempo dos *bits* pares e ímpares do sinal de informação. No OQPSK, os fluxos de *bits* pares e ímpares, $m_I(t)$ e $m_Q(t)$, são deslocados em seu alinhamento relativo por um período de *bit* (meio período de símbolo). Isso é mostrado nas formas de onda da Figura 108.

Devido ao alinhamento no tempo de $m_I(t)$ e $m_Q(t)$ no QPSK padrão, as transições de fase ocorrem apenas uma vez a cada $T_s = 2T_b$ segundos, e serão no máximo 180° se

Figura 108 – Formas de onda em fase e quadratura aplicadas a um modulador OQPSK.



Fonte: (RAPPAPORT, 2009)

houver uma mudança no valor de $m_I(t)$ e $m_Q(t)$. Porém, na sinalização do OQPSK, as transições de *bits* (e, portanto, as transições de fase) ocorrem a cada T_b segundos. Como os instantes de transição de $m_I(t)$ e $m_Q(t)$ são deslocados, em qualquer momento que ocorrer uma transição, somente um dos dois fluxos de *bits* irá mudar de valor. Isso implica que o deslocamento de fase máximo do sinal transmitido em determinado momento é limitado a $\pm 90°$. Logo, alternando as fases com mais frequência (ou seja, a cada T_b segundos em vez de $2T_b$ segundos), a sinalização OQPSK elimina as transições de fase 180° (RAPPAPORT, 2009).

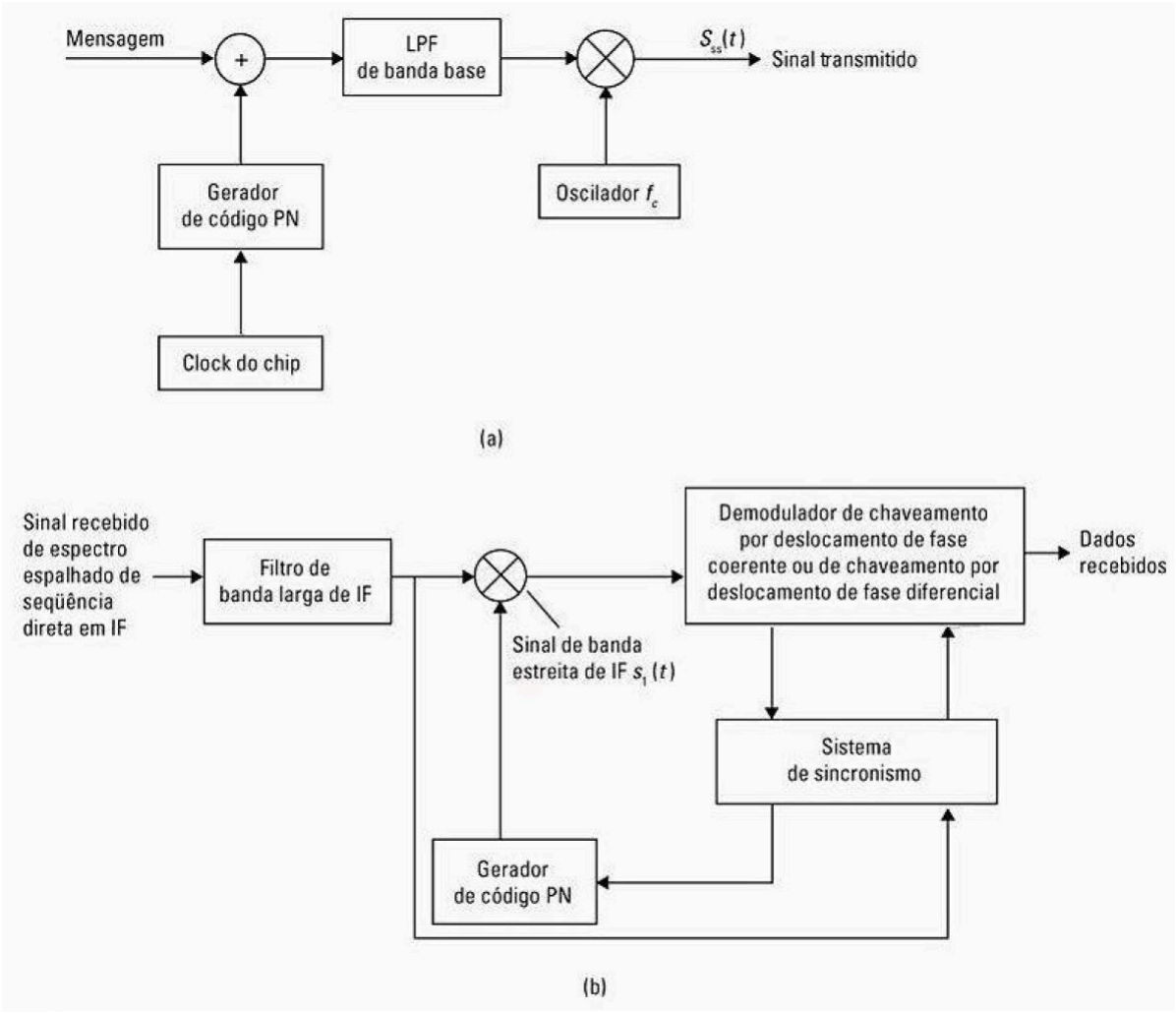
Como as transições de fase 180° foram eliminadas, a limitação de banda (ou seja, a formatação de pulsos) do OQPSK não irá fazer com que o envelope do sinal tenha um cruzamento por zero. Obviamente, haverá alguma quantidade de ISI causada pelo processo de transição de fase de 90°. Mas as variações de envelope são consideravelmente menores, daí a limitação ou amplificação não-linear dos sinais OQPSK não irão gerar o aumento no espectro com os lóbulos laterais de alta frequência tanto quanto no QPSK. Assim, a ocupação espectral é significativamente reduzida, enquanto permite a amplificação mais eficiente.

O espectro de um sinal OQPSK é idêntico ao de um sinal QPSK, portanto, os dois sinais ocupam a mesma largura de banda. O desalinhamento dos fluxos de *bits* pares e ímpares não altera o seu espectro. O OQPSK mantém sua banda limitada mesmo depois da amplificação não-linear, portanto, é muito atraente para sistemas de comunicações móvel, onde a eficiência da largura de banda e amplificadores não-lineares eficientes são críticos para o baixo dreno de potência (RAPPAPORT, 2009).

ANEXO B – Espalhamento Espectral de Sequência Direta

Um sistema de espalhamento espectral de sequência direta (*Direct Sequence Spread Spectrum* (DSSS)) espalha os dados de um sinal passa-baixa por uma sequência de pseudo-ruído (PN) que é produzida por um gerador de código de PN. Um único pulso ou símbolo da forma de onda de PN é chamado chip. A Figura 109 mostra um diagrama de blocos funcional de um sistema DSSS com modulação BPSK.

Figura 109 – (a) Diagrama em blocos de um sistema DSSS com modulação BPSK transmissor. (b) receptor



Fonte: (RAPPAPORT, 2009)

Esse sistema é uma das implementações de sequência direta mais utilizadas (RAPPAPORT, 2009). Os *bits* do sinal de informação são modificados através da operação "Ou exclusivo"com os chips da sequência antes de serem modulados em fase. Uma demodulação

de chaveamento por deslocamento de fase (PSK) coerente ou diferencialmente coerente pode ser usada no receptor.

O sinal do espalhamento espectral recebido (s_{SS}) pode ser representado como apresentado na equação B.1, onde $m(t)$ são os *bits* do sinal de informação e $p(t)$ são os *bits* da sequência PN.

$$s_{SS}(t) = \sqrt{\frac{2E_s}{T_s}} m(t)p(t)\cos(2\pi f_c t + \theta_c) \tag{B.1}$$

A forma de onda dos dados do sinal de informação é uma sequência no tempo de pulsos retangulares não sobrepostos, cada um com uma amplitude igual a $+1$ ou -1. Cada símbolo em $m(t)$ representa um símbolo de dados e tem duração T_s. Cada pulso em $p(t)$ representa um chip , normalmente é retangular com uma amplitude igual a $+1$ ou -1 e tem duração de T_c. As transições dos símbolos de dados e chips coincidem de modo que a razão entre T_s e T_c é um inteiro. Se B_{SS} é a largura de banda de s_{SS} e B é a largura de banda de um sinal $m(t)\cos(2\pi f_c t)$ modulando convencionalmente, o espalhamento devido a $p(t)$ gera $B_{SS} \gg B$ (RAPPAPORT, 2009).

A Figura 109(b) ilustra um receptor de DSSS, supondo que o sincronismo de código foi alcançado no receptor. O sinal recebido passa pelo filtro passa-faixas e é multiplicado por um réplica local da sequência de código $p(t)$. Se $p(t) = \pm 1$, então $p^2(t) = 1$, e essa multiplicação gera o sinal de desespalhamento $s_1(t)$ dado pela equação B.2 na entrada do demodulador. Como s_1 tem a forma de um sinal BPSK, a demodulação correspondente extrai $m(t)$.

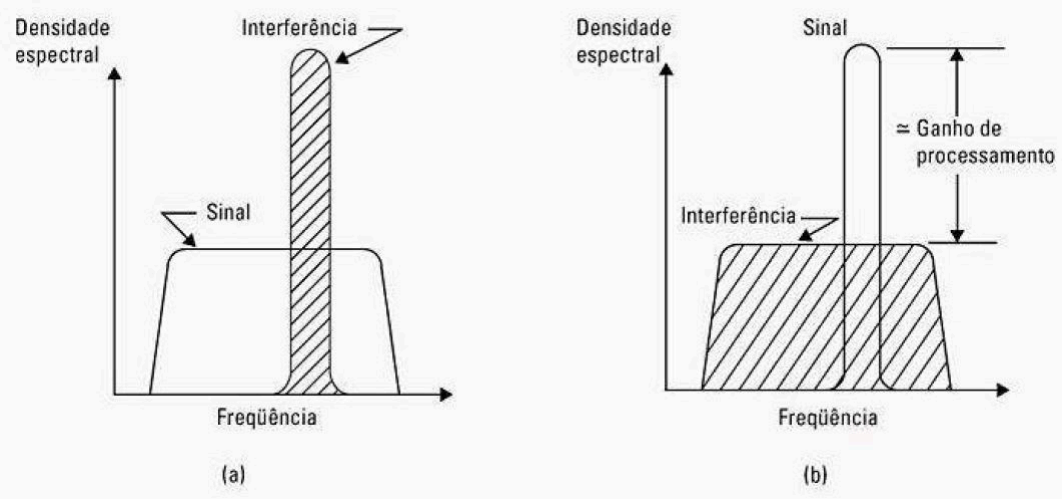
$$s_1(t) = \sqrt{\frac{2E_s}{T_s}} m(t)\cos(2\pi f_c t + \theta_c) \tag{B.2}$$

A Figura 110 mostra os espectros recebidos do sinal e a interferência na saída do filtro passa-faixas do receptor. A multiplicação pela forma de onda de espalhamento produz os espectros da figura 110(b) na entrada do demodulador.

Segundo (RAPPAPORT, 2009), a largura de banda do sinal é reduzida a B enquanto a energia de interferência é espalhada por uma largura de banda excedente B_{SS}. Essa ação de filtragem do demodulador remove a maioria do espectro de interferência que não se sobrepõe ao espectro do sinal, sendo assim, a maior parte da energia de interferência é eliminada pelo espalhamento e afeta ao mínimo o sinal desejado. Uma medida aproximada pela razão B_{SS}/B, que é igual ao ganho de processamento definido como na equação B.3, considerando que B é largura do sinal modulado sem o espalhamento que é igual a $2R_s$.

$$PG = \frac{T_s}{T_c} = \frac{R_c}{R_s} = \frac{B_{SS}}{2R_s} \tag{B.3}$$

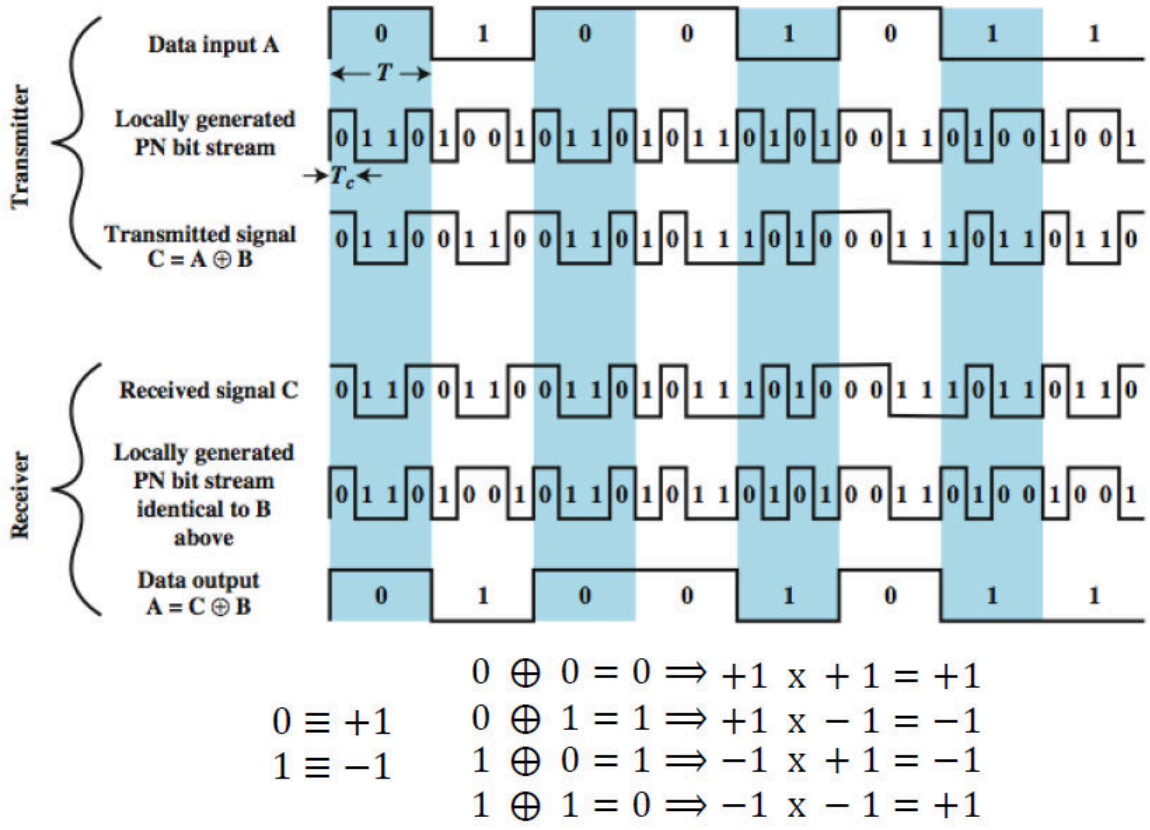
Figura 110 – (a) Espectros do sinal recebido com interferência na saída do filtro. (b) saída do correlator após o desespalhamento



Fonte: (RAPPAPORT, 2009)

Quanto maior o ganho de processamento do sistema, maior será sua capacidade de suprimir a interferência na banda. A Figura 111 mostra um exemplo de transmissão e recepção de uma sequência de dados utilizando a técnica de espalhamento espectral.

Figura 111 – Exemplo de transmissão e recepção de uma sequência de dados com o espalhamento.



Fonte: Adaptado de (STALLINGS, 2005).

ANEXO C – Arduino Mega 2560

A Figura 112 apresenta o circuito elétrico completo da plataforma do *Arduino* Mega 2560. Será destacado e apresentado sobre as principais partes dos circuitos dessa plataforma.

C.1 Alimentação

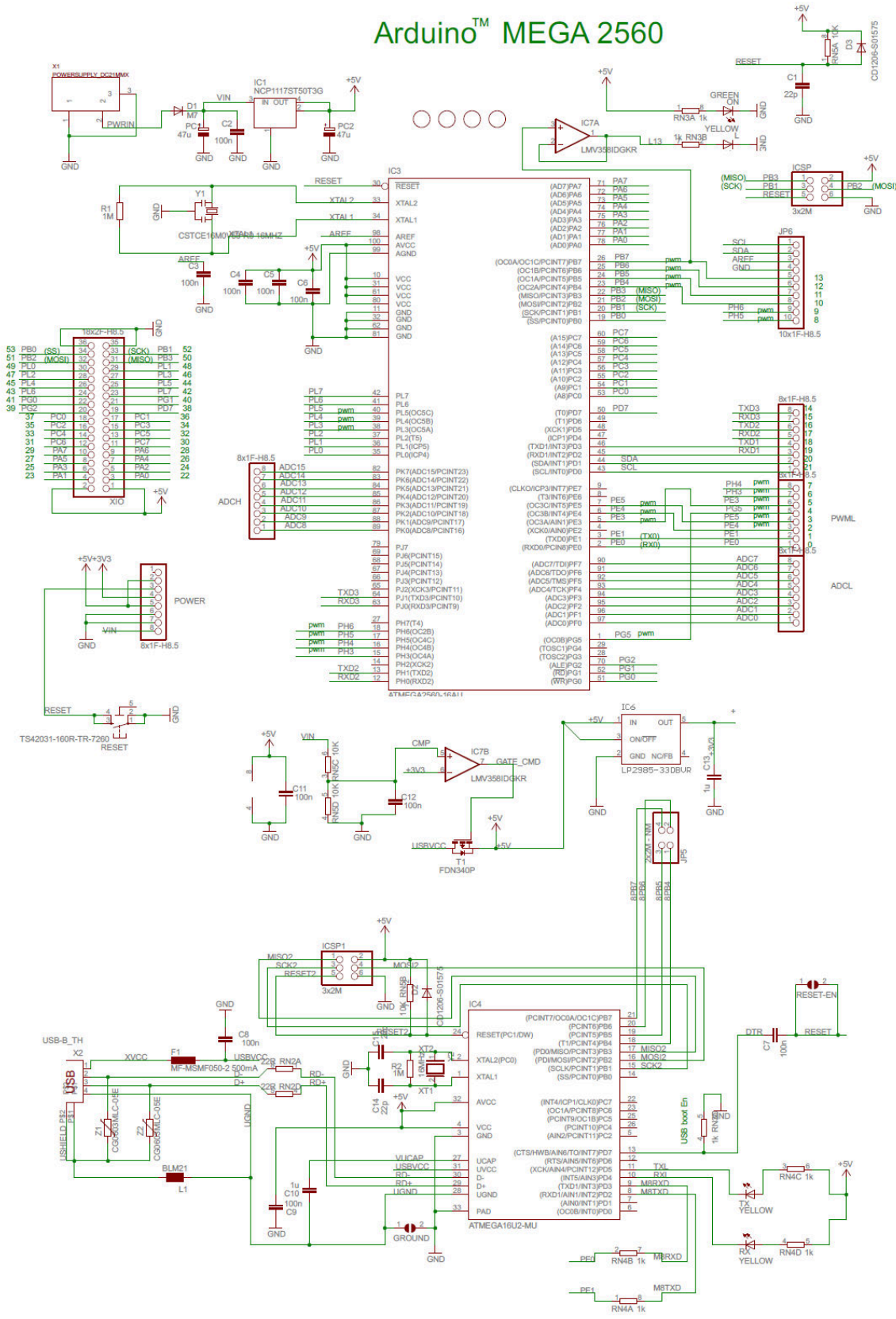
O *Arduino* pode ser alimentado por uma fonte externa, utilizando o plugue de fonte do tipo P4, ou inserir diretamente essa alimentação no pino V_{in} no conector da placa. Ele suporta uma tensão máxima de 20 V. Conforme a figura 113, pode-se visualizar que a alimentação passará por um regulador de 5 V que alimenta o resto da plataforma, incluindo um regulador de 3.3 V que é utilizado no sistema de seleção de alimentação, uma vez que a placa pode ser alimentada diretamente pelo conector USB da mesma. A Figura 114 apresenta esse sistema, onde há um circuito comparador que verifica se existe uma alimentação externa maior que 7 V, caso exista, ela alimentará a plataforma mesmo que o cabo USB esteja conectado. Caso não seja conectada alimentação externa, o comparador irá acionar o *mosfet* para que a alimentação da plataforma seja feita pela alimentação oriunda do cabo USB. A plataforma possui um diodo de proteção para evitar que seja alimentado com polaridade invertida.

C.2 Comunicação USB

O microcontrolador principal da plataforma não possui condições direta de estabelecer uma comunicação USB. Para não ter a necessidade de utilizar gravador de microcontrolador todas as vezes que for realizar um *upload* de um novo código binário, os autores da plataforma criaram uma interface para fazer a comunicação USB entre o PC e o microcontrolador principal da placa. A Figura 115 apresenta essa interface que é constituída de um microcontrolador da ATMEL 16U2, que possui comunicação USB interna. Esse microcontrolador por sua vez irá fazer a programação do microcontrolador principal (ATmega 2560). Com isso para programar a plataforma é necessário apenas o software do fabricante e um cabo USB.

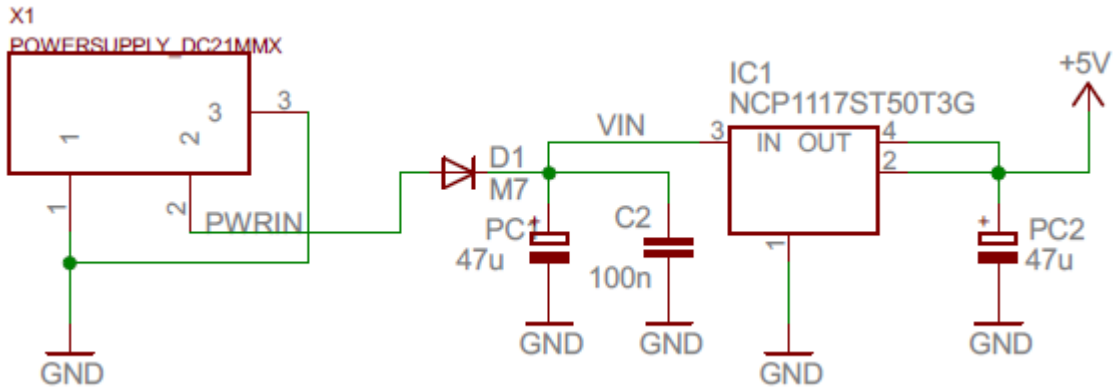
A interface ainda possui uma proteção de fusível (F1) de 500 mA para proteger a conexão USB de um curto-circuito e um ferrite (L1) na linha do GND para eliminar ruídos na alimentação, além de dois varistores nas linhas de dados (D+ e D-) para proteger contra surtos nesses condutores.

Figura 112 – Circuito elétrico da placa Arduino Mega 2560.



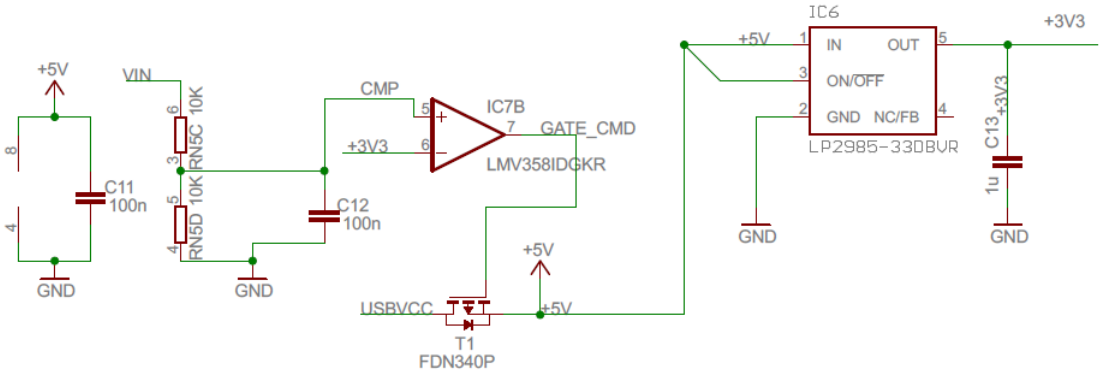
Fonte: (ARDUINO.CC, 2018)

Figura 113 – Circuito da parte de alimentação do Arduino Mega 2560



Fonte: (ARDUINO.CC, 2018)

Figura 114 – Circuito de seleção de alimentação do Arduino Mega 2560



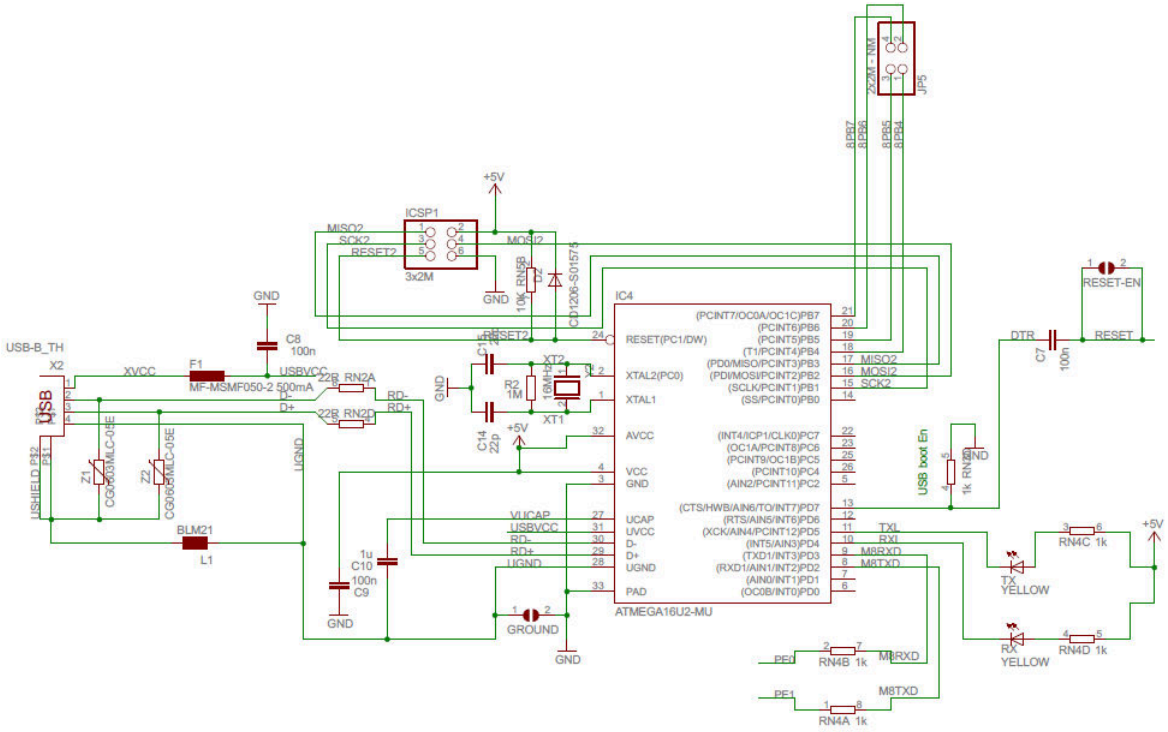
Fonte: (ARDUINO.CC, 2018)

Esse microcontrolador (16U2) possui um cristal de 16 MHz para sua operação e um conector ICSP na placa caja haja necessidade de atualizar ou alterar o programa do mesmo, além de possuir também dois leds na placa que indicam que a comunicação via USB está acontecendo.

C.3 Microcontrolador

O ATmega2560 é um microcontrolador CMOS (*Complementary metal-oxide-semiconductor*) de 8 *bits* de baixa potência baseado no processador AVR aprimorado sendo a sua arquitetura RISC (*Reduced Instruction Set Computer*). Ao executar instruções poderosas em um único ciclo de clock, o ATmega2560 atinge taxas de transferência que se aproximam de 1 MIPS (Milhão de Instruções por segundo) por MHz permitindo o sistema de designer otimizar o consumo de energia em relação a velocidade de processamento. A Figura 116 apresenta o diagrama de blocos da construção do microcontrolador da empresa ATMEL.

Figura 115 – Circuito de comunicação USB do Arduino Mega 2560



Fonte: (ARDUINO.CC, 2018)

C.3.1 CPU

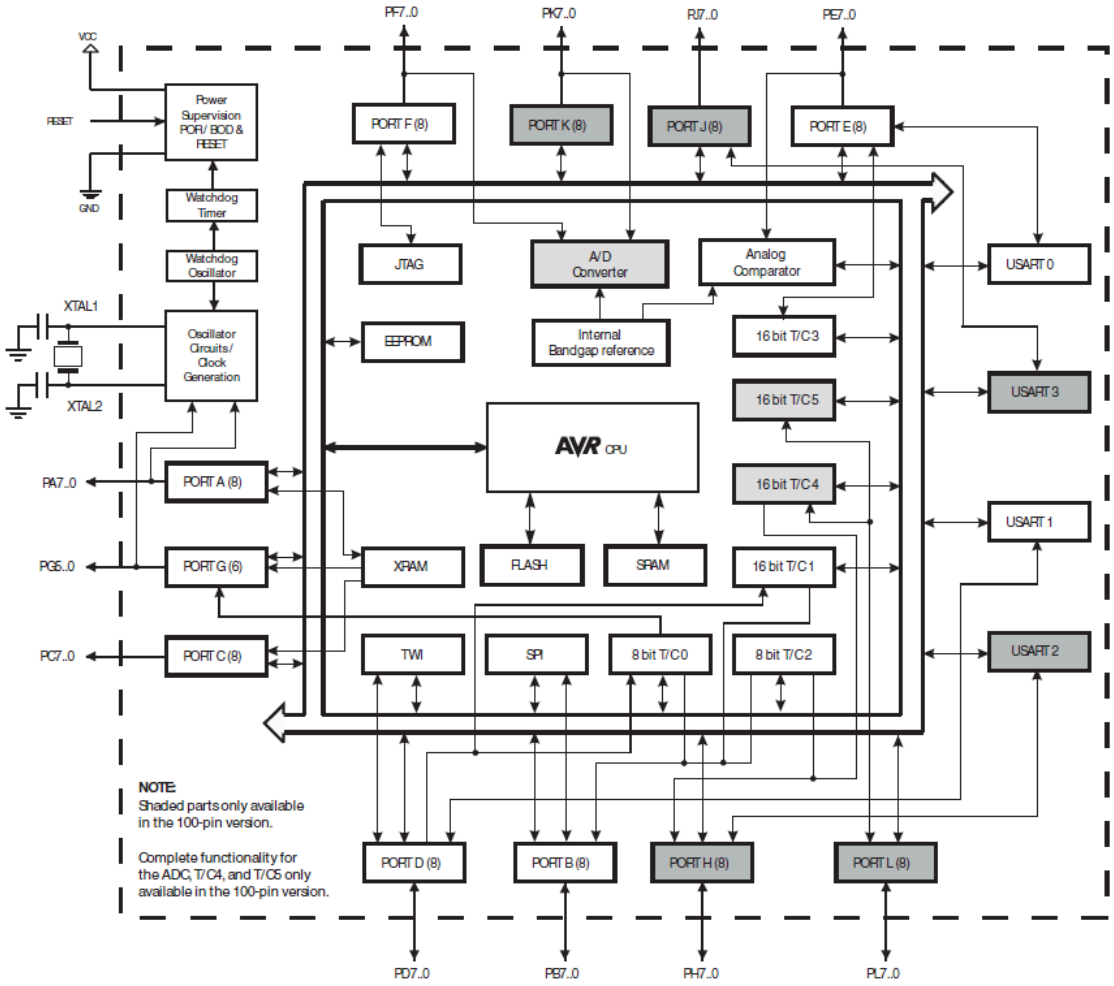
Segundo (ATMEL, 2014), para maximizar a *performace* e paralelismo, o AVR utiliza um arquitetura Harvard, com memórias separadas para programa e dados. As instruções na memória de programa são executadas com um pipeline de nível único, ou seja, enquanto uma instrução está sendo executada, a próxima instrução é pré-buscada na memória do programa. Esse conceito permite que as instruções sejam executadas em cada ciclo de *clock*. A memória do programa é reprogramável no sistema da memória *flash*. A Figura 117 apresenta o diagrama de blocos da CPU AVR.

Ele possui um arquivo de registro de acesso rápido que contém 32 registradores de uso geral sendo cada um deles de 8 *bits* que podem ser acessados com um único ciclo de *clock*. Isso permite que a operação da ULA (Unidade Lógica Aritmética) seja de ciclo único.

Em sua arquitetura, 6 dos 32 registradores podem ser usados como três ponteiros de registro de endereço indireto de 16 *bits* para endereçamento de espaço de dados, permitindo cálculos de endereço eficientes.

O fluxo do programa é fornecido por instruções de saltos condicionais, incondicionais e chamada de instruções capazes de endereçar todo o espaço de endereçamento. A maioria das instruções AVR tem um único formato de 16 bits. Cada endereço de memória do programa contém uma instrução de 16 ou 32 bits.

Figura 116 – Diagrama de blocos do microcontrolador da plataforma Arduino Mega 2560

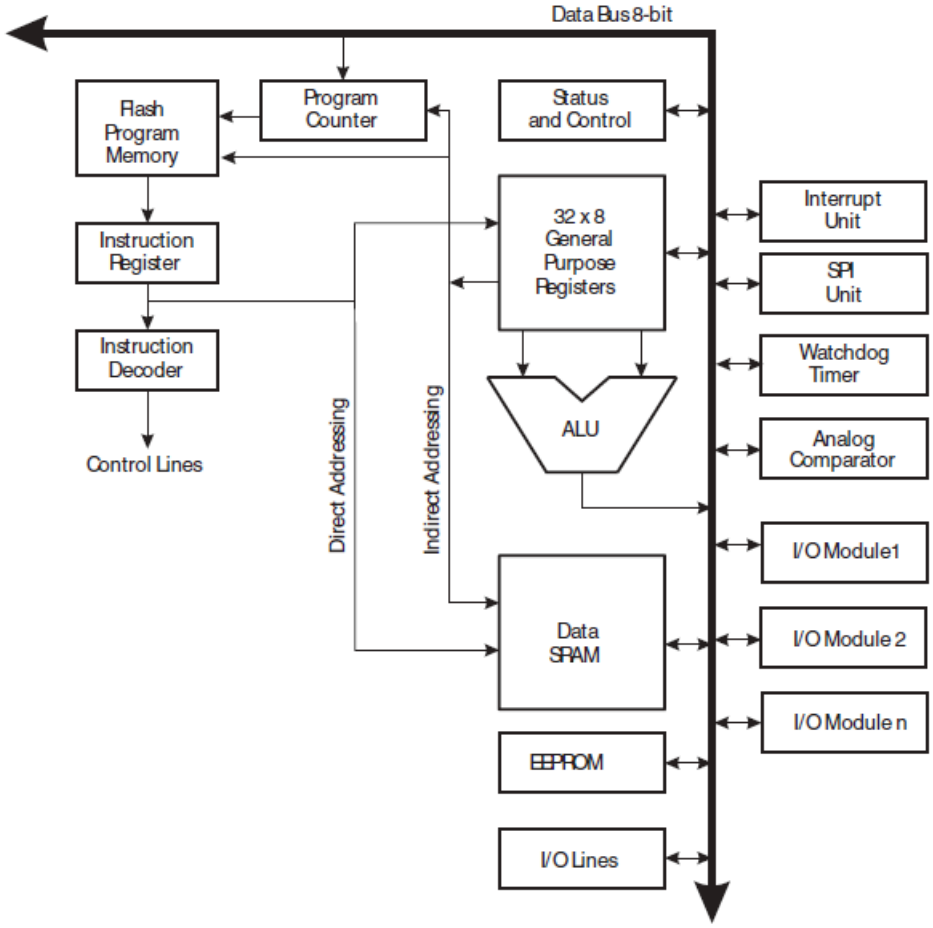


Fonte: (ATMEL, 2014)

O espaço de memória *Flash* do programa é dividido em duas seções, a seção *Boot Program* e a seção *Application Program*. Ambas as seções possuem *bits* de bloqueio dedicados para proteção contra gravação e leitura/gravação.

Durante interrupções e chamadas de sub-rotinas, o endereço de retorno do registrador *Program Counter* é armazenado na pilha. A pilha é efetivamente alocada na SRAM (*Static Random Acess Memory*) de dados gerais e consequentemente, o tamanho da pilha é limitado apenas pelo tamanho total da SRAM e pelo uso da SRAM. Todos os programas do usuário devem inicializar o registrador *Stack Pointer* na rotina *Reset* (antes que sub-rotinas ou interrupções sejam executadas). O *Stack Pointer* é acessível para leitura/gravação no espaço de I/O. A SRAM de dados pode ser facilmente acessada através dos cincos modos de endereçamento diferentes suportados na arquitetura AVR.

Figura 117 – Diagrama de blocos da CPU AVR do microcontrolador da plataforma Arduino Mega 2560



Fonte: (ATMEL, 2014)

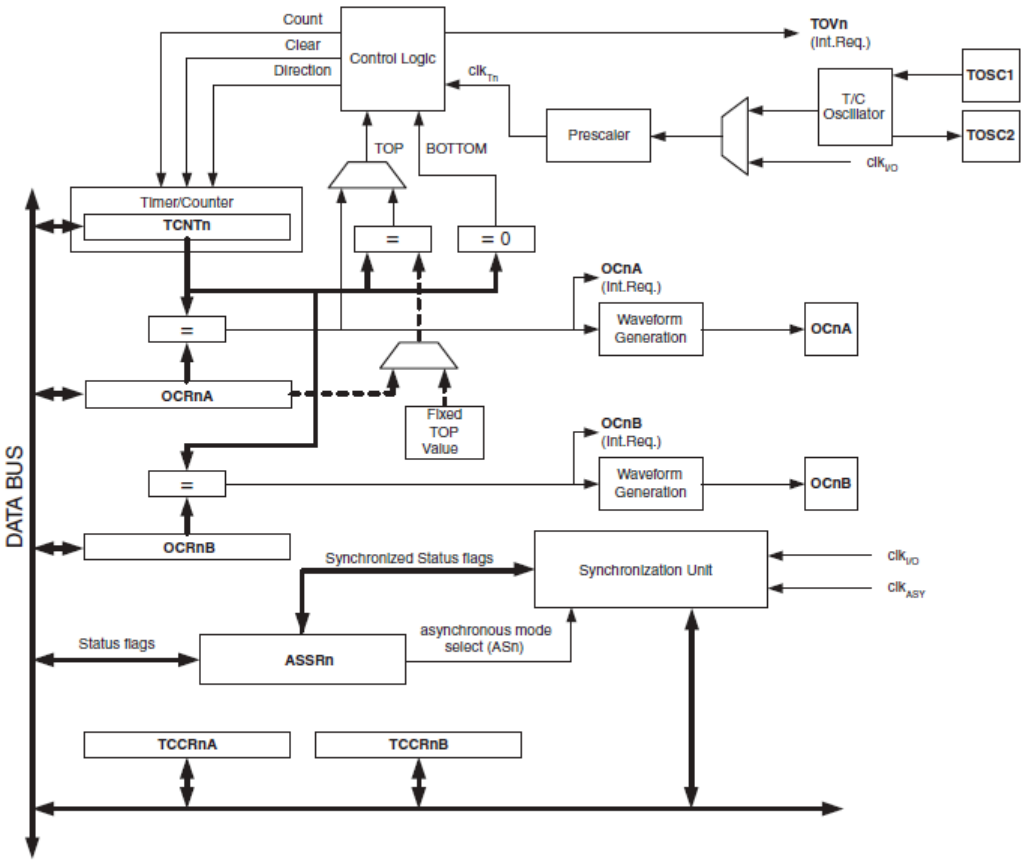
C.3.2 *Timers/Counter*

Segundo o (ATMEL, 2014), o microcontrolador ATmega2560 possui seis *Timers* interno, sendo dois *Timers* de 8 *bits* (*Timers* 0 e 2) e quatro de 16 *bits* (*Timers* 1, 3, 4 e 5). Esses *Timers* permitem um tempo preciso na execução do programa e geração de ondas. Eles são importantes para diversas funcionalidades, tais como:

- Temporização;
- Contagem de eventos externos;
- Geração de sinais PWM
- Interrupções periódicas;
- Medida de intervalos de pulsos.

Cada temporizador possui características próprias e são utilizados conforme os recursos disponíveis em cada um. A seguir será apresentado algumas informações do *Timers* 2. A Figura 118 apresenta o diagrama de blocos do *Timers* 2 do microcontrolador.

Figura 118 – Diagrama de blocos do *Timer* 2 do microcontrolador da plataforma Arduino Mega 2560



Fonte: (ATMEL, 2014)

Esse *Timer* possui 3 modos de operação:

- **Normal**: Neste modo, o contador simplesmente conta até o valor máximo (que nesse caso de 8 *bits* será de 0 a 255);
- **Contagem para comparação (CTC)**: Conta de 0 ao valor definido no registrador de comparação OCRnA ou OCRnB e em seguida reseta o contador para zero.
- **Modulação de largura de pulso (PWM)**: Normalmente usado para fornecer uma saída analógica controlada digitalmente como por exemplo controle de motores, LEDs ou geração de forma de onda.

A Figura 119 apresenta uma tabela com as possíveis configurações e variações dos modos de operação do *Timer* 2 e os devidos valores para os registradores.

No *Timer* 2 é possível operar de forma síncrona (sincronizada com o *clock* do microcontrolador) ou de forma assíncrona com a injeção de um sinal de oscilação externo nos pinos TOSC1 e TOSC2 (pinos PG4 e PG3 respectivamente do microcontrolador). Em ambos os casos, é possível utilizar o *Prescaler* do *Timer* que pode ou não dividir o sinal de

Figura 119 – Modos de operação do *Timer* 2 do microcontrolador da plataforma Arduino Mega 2560

Table 20-8. Waveform Generation Mode Bit Description

Mode	WGM2	WGM1	WGM0	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on(1)(2)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Notes: 1. MAX = 0xFF.
2. BOTTOM= 0x00.

Fonte: (ATMEL, 2014)

clock em 1, 8, 64, 256 ou 1024 vezes para alcançar tempos diferentes e ser mais preciso. A Figura 120 apresenta uma tabela com as possíveis configurações do *prescaler* do *Timer* 2.

Figura 120 – Prescaler do *Timer* 2 do microcontrolador da plataforma Arduino Mega 2560

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	clkI/O/1 (No prescaling)
0	1	0	clkI/O/8 (From prescaler)
0	1	1	clkI/O/64 (From prescaler)
1	0	0	clkI/O/256 (From prescaler)
1	0	1	clkI/O/1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

Fonte: Adaptado de (ATMEL, 2014)

Segundo o (ATMEL, 2014), o *Timer* 2 pode gerar diferentes tipos de interrupção. Essas interrupções podem ser habilitadas ou desabilitadas nos registradores do *Timer*. Existem três (considerando que para uma forma existem 2 interrupções independentes) formas de gerar uma interrupção por causa do *Timer* e quando esses eventos ocorrem, eles geram uma interrupção para o microcontrolador que pode desviar a execução do código atual para uma outra parte do código e após a execução disso, ele retornará onde estava. As formas de interrupção são:

- ***Timer Overflow*:** Estouro do temporizador significa que o temporizador ultrapassou o valor limite de contagem (255). Quando ocorre isso, o conteúdo do *Timer* é zerado

e é sinalizado o bit TOV2 que indica ao microcontrolador que houve um interrupção por *overflow*.

- **Correspondência de comparação de saída:** Ocorre quando o conteúdo do *Timer* é igual ao conteúdo do OC2A ou OC2B, quando isso ocorre, o conteúdo do *Timer* é zerado e o bit OCF2x (onde x pode ser o A ou o B) é sinalizado indicando que houve uma interrupção por comparação.

C.4 SPI

A interface SPI (*Serial Peripheral Interface*) é usada para comunicação entre vários dispositivos em curtas distâncias e em altas velocidades. Na rede, normalmente existe um único dispositivo "mestre", que irá iniciar as comunicações e fornecer o *clock* que controla a taxa de transferência de dados. Podem existir um ou mais escravos na rede. Um sistema de SPI completo possui quatro linhas de sinais:

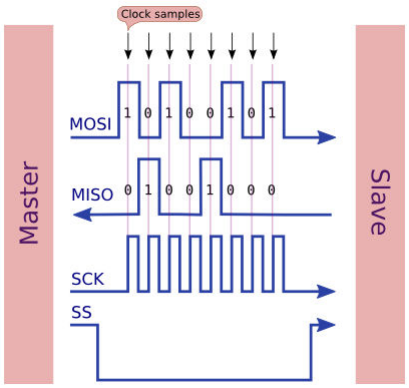
- ***Master Out, Slave In* (MOSI):** São os dados que vão no sentido do fluxo do mestre para escravo;
- ***Master In, Slave Out* (MISO):** São os dados que vão no sentido do fluxo do escravo para mestre;
- **Relógio Serial (SCK):** Linha de *clock* para sincronizar as transmissões e leitura dos dados;
- **Seleção de Escravo (SS):** Linha utilizada pelo mestre para selecionar a qual escravo está sendo direcionado a transmissão de dados

Quando vários escravos são conectados ao sinal MISO, espera-se que eles estejam com esse pino em *tri-state* (alta impedância), até que sejam selecionados pelo pino SS. Para selecionar um escravo, o pino SS deve ter o valor "zero", indicando ao escravo que as informações são para ele. A Figura 121 apresenta um diagrama simplificado da comunicação entre mestre e escravo.

Como os dados são enviados e recebidos no mesmo pulso do *clock*, não é possível que o escravo responda imediatamente ao mestre. Os protocolos de SPI geralmente esperam que o mestre solicite dados em uma transmissão e obtenha a resposta na transmissão seguinte.

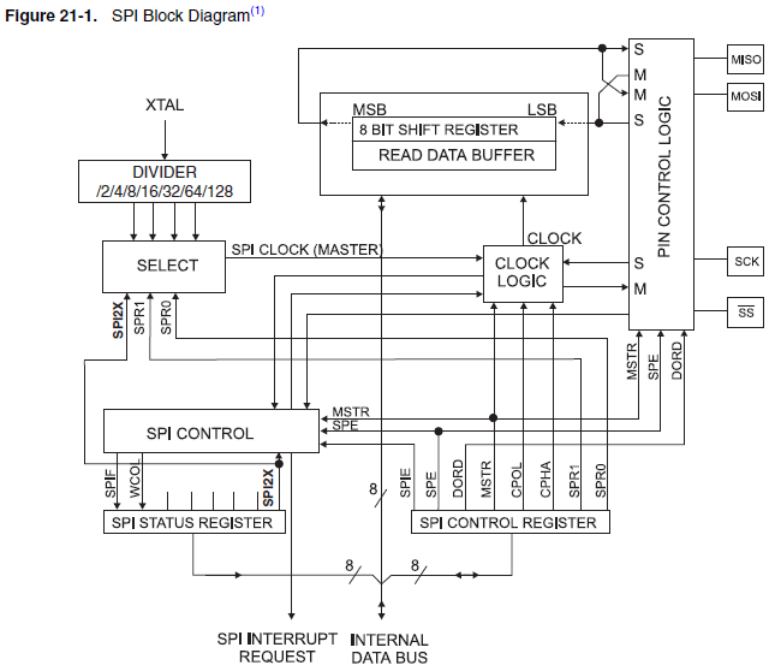
A Figura 122 apresenta um diagrama de blocos do módulo de SPI no microcontrolador ATmega2560.

Figura 121 – Diagrama de comunicação do SPI.



Fonte: (GAMMON, 2015)

Figura 122 – Diagrama de blocos do SPI no microcontrolador da plataforma Arduino Mega 2560



Fonte: (ATMEL, 2014)

A ordem dos dados tanto no pino MOSI quanto no pino MISO, normalmente segue do MSB para o LSB, porém é possível alterar a ordem de envio dos *bit* para ser do LSB para o MSB através do bit DORD do registrador SPCR.

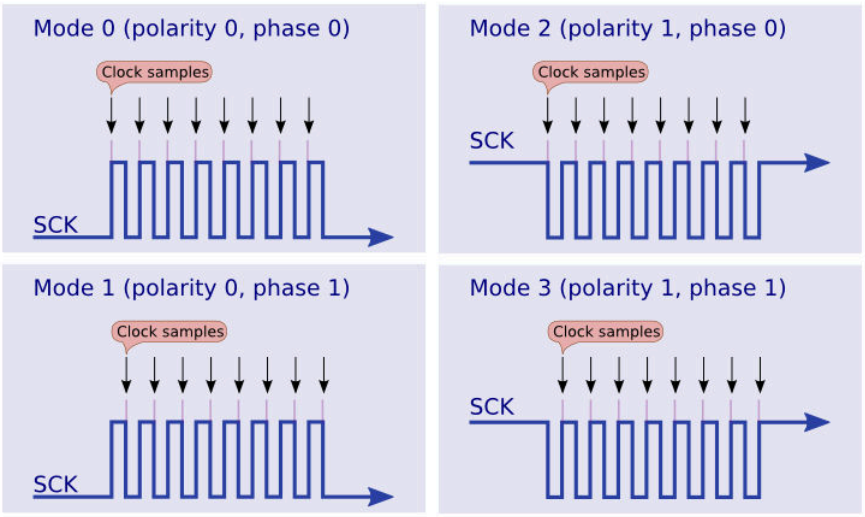
O *hardware* do SPI permitem 4 modos de operação, modos esses que variam na polaridade dos pulsos (CPOL) e a fase (CPHA) do *clock*. A seguir será descrito simplificadamente os modos de funcionamento:

- **Modo 0 (padrão):** O *clock* normalmente fica em nível zero (CPOL = 0) e os dados serão amostrados na transição de zero para um (borda superior (CPHA=0));
- **Modo 1:** O *clock* normalmente fica em nível zero (CPOL = 0) e os dados serão amostrados na transição de um para zero (borda inferior (CPHA=1));

- **Modo 2:** O *clock* normalmente fica em nível um (CPOL = 1) e os dados serão amostrados na transição de um para zero (CPHA=0);
- **Modo 3:** O *clock* normalmente fica em nível um (CPOL = 1) e os dados serão amostrados na transição de zero para um (CPHA=1);

A Figura 123 ilustra cada um dos modos de operação do SPI.

Figura 123 – Modos de operação do SPI



Fonte: (GAMMON, 2015)

É possível alterar a taxa de oscilação do *clock* no pino SCK, desde de que este esteja atuando no modo Mestre. A Figura 124 apresenta os possíveis valores de *clock*, onde f_{osc} representa a frequência de oscilação de entrada no microcontrolador (pinos XTAL1 e XTAL2). Para alterar o valor, basta alterar os bits SPI2X, SPR1 e SPR0.

Figura 124 – Possibilidades de *clock* na comunicação SPI

Table 21-5. Relationship Between SCK and the Oscillator Frequency

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

Fonte: (ATMEL, 2014)

C.5 I2C

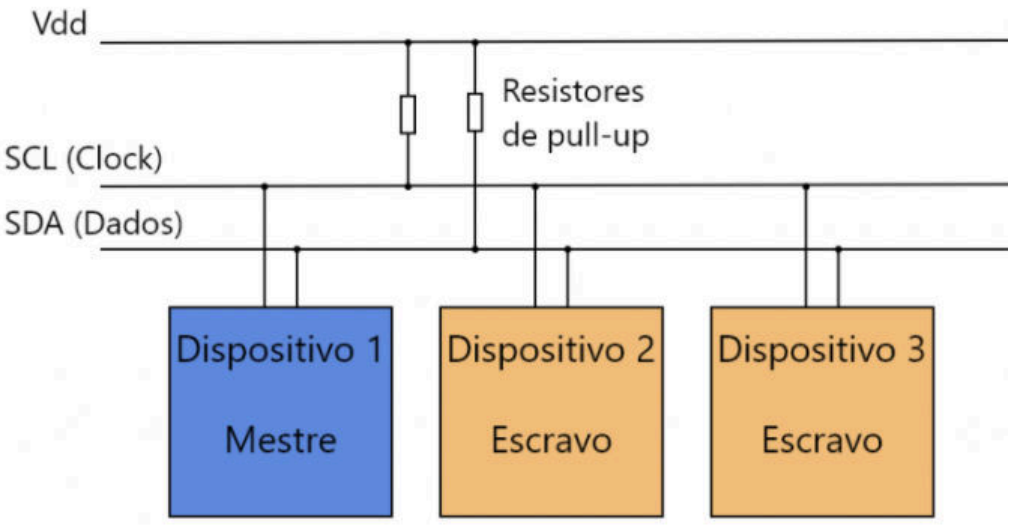
O I2C (*Inter-Integrated Circuit*) é um tipo de interface serial utilizada para comunicar com diversos dispositivos. Assim como na comunicação assíncrona, o I2C utiliza dois canais para comunicação. A diferença é que nesse caso, um canal é para transmissão e recepção de dados e o outro será para sincronização dos mesmos, ou seja, ele é um tipo de comunicação síncrona. O I2C foi criado pela Philips, tendo como vantagem o baixo custo e simplicidade e como desvantagem a sua velocidade (GUIMARÃES, 2018).

Em relação aos canais de comunicação, existe o canal de dados seriais que é chamado de *Serial Data* (SDA) e o canal de sincronização chamado de *Serial Clock* (SCL). Outra informação importante é que ambos os canais são de comunicação bidirecionais, ou seja, os dispositivos dos dois lados podem utilizar os canais.

Para funcionar a rede, ela precisa necessariamente de pelo menos um mestre e um escravo. O mestre será responsável pela coordenação da comunicação, gerando o *clock* e iniciando a comunicação. Já o escravo apenas recebe os dados e responde quando é chamado. Cada escravo possui um endereço de 7 bits que é utilizado para um mestre conseguir comunicar individualmente com ele. Dependendo da forma utilizada na rede, os dispositivos podem ser mestres ou escravos em momentos diferentes, embora normalmente eles assumam um papel fixo.

As linhas de comunicação do I2C, funcionam em coletor aberto, então se faz necessário a utilização de resistores de *pull-up* para evitar erros na comunicação. A Figura 125 apresenta o esquema de ligação da rede.

Figura 125 – Diagrama de conexão do I2C



Fonte: (GUIMARÃES, 2018)

Segundo (GUIMARÃES, 2018), a desvantagem do I2C é a relativa baixa velocidade. Isso acontece por dois motivos, o primeiro é que o procedimento de transmissão de dados

é bem "burocrático"e o segundo é que uma única linha transmitem e recebem dados, ou seja, a comunicação é *half-duplex*. Isso impede que ocorra transmissão e recepção ao mesmo tempo. Um valor médio de velocidade é de 100 kbit/s. Apesar desse problema, o barramento permite a comunicação entre diversos dispositivos com apenas dois fios, sendo assim, podem existir diversos escravos e diversos mestres no barramento. O número máximo de dispositivos ligados pode depender da capacitância máxima do barramento (400 pF).

Em um barramento com dois ou mais mestres, podem ocorrer conflitos de comunicação. Normalmente, um mestre sempre espera o outro terminar a comunicação quando o barramento está sendo utilizado. Entretanto, dois mestres podem tentar transmitir dados ao mesmo tempo. Para isso, existe uma arbitragem que evita o erro na comunicação. O que ocorre é que cada mestre verifica o nível da linha SDA e compara com o nível que é esperado. Se o nível do SDA é diferente do esperado, então este mestre perde o direito de transmissão e esperar até o outro mestre terminar o procedimento. Caso contrário, ele pode transmitir todos os dados necessários. Assim o protocolo tenta evitar conflitos de comunicação

A seguir será descrito as etapas de funcionamento do I2C, levando em consideração que as duas linhas (SCL e SDA) estão em nível alto devido os resistores de *pull-up* e que a transmissão dos *bits* ocorre do MSB para o LSB. A transmissão ocorrerá na seguinte ordem:

1. Para ser detectado um início de transmissão, a linha de dados SDA precisa passar de nível alto para nível baixo enquanto a linha de *clock* SCL estiver em nível alto. Isso é chamado de *bit* de *Start*.

2. Em seguida, a linha SCL fica oscilando na frequência estipulada.

3. Enquanto isso, a linha SDA deve enviar 7 *bits* correspondente ao endereço do dispositivo alvo. Esses *bits* são enviados um por vez a cada intervalo que a linha SCL estiver em nível baixo, ou seja, o *clock* fica em nível baixo e um bit é enviado, depois o *clock* fica em nível alto e, em seguida, fica novamente em nível baixo, então outro *bit* é enviado.

4. Após os 7 *bits* de endereço, um *bit* deve ser enviado para determinar se deseja escrever ou ler um registrador do dispositivo alvo. Enviar 0 indica escrita e 1 indica leitura. Para esse início da comunicação devemos enviar 0.

5. Na sequência do primeiro *byte*, o escravo deve informar se ele recebeu as informações. Ele faz isso mantendo a linha SDA em nível baixo e o mestre deve deixar a linha SDA livre. Esse *bit* é chamado de *Acknowledge* ou ACK. Se o escravo não recebeu corretamente os dados, ele mantém a linha em nível alto e este *bit* recebe o nome

de *Not Acknowledge* ou NACK. O NACK pode ocorrer por outros motivos como, por exemplo, se nenhum escravo com o endereço existir, mas de qualquer forma a comunicação será abortada com um NACK.

6. Caso exista o *bit* ACK, o mestre envia o endereço do registrador que ele deseja ler ou gravar os dados. Este endereço é constituído de um *byte* e deve ser enviado da mesma forma que foi enviado o endereço.

7. Em seguida o escravo irá novamente enviar um bit de ACK.

A partir desse ponto, o que ocorre a seguir irá depender se a comunicação foi de leitura ou de escrita. Caso seja de escrita e o mestre recebeu o segundo ACK e então:

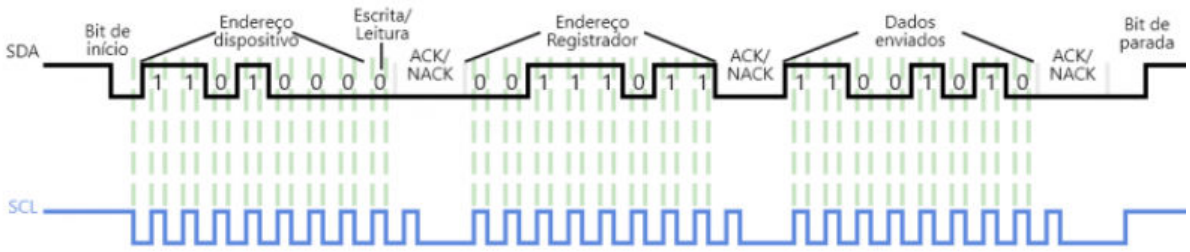
1. O mestre enviará o *byte* com os dados que ele deseja gravar no registrador. Pode ser que o registrador possua mais de um *byte* de espaço. Se for esse caso, esse passo e o ACK se repetem até todos os dados serem enviado.

2. A penúltima informação enviada deve ser um ACK pelo escravo.

3. Por último, quando não há mais nada a ser enviado, a linha SDA passa de nível baixo para nível alto enquanto a linha SCL está em alta. Isso é chamado de *Stop bit*.

A Figura 126 mostra o exemplo de uma escrita, onde o endereço do escravo é 0x68 (1101000), o registrador é 0x3B (00111011) e os dados enviados são 11001010.

Figura 126 – Exemplo de comunicação de escrita no I2C



Fonte: (GUIMARÃES, 2018)

Já se for de leitura irá ocorrer o seguinte:

1. O mestre envia um novo *bit* de *Start* (SCL em alto e SDA passa de alto para baixo).

2. Após este segundo bit de *Start*, o mestre deve enviar novamente o endereço do escravo assim como enviou anteriormente.

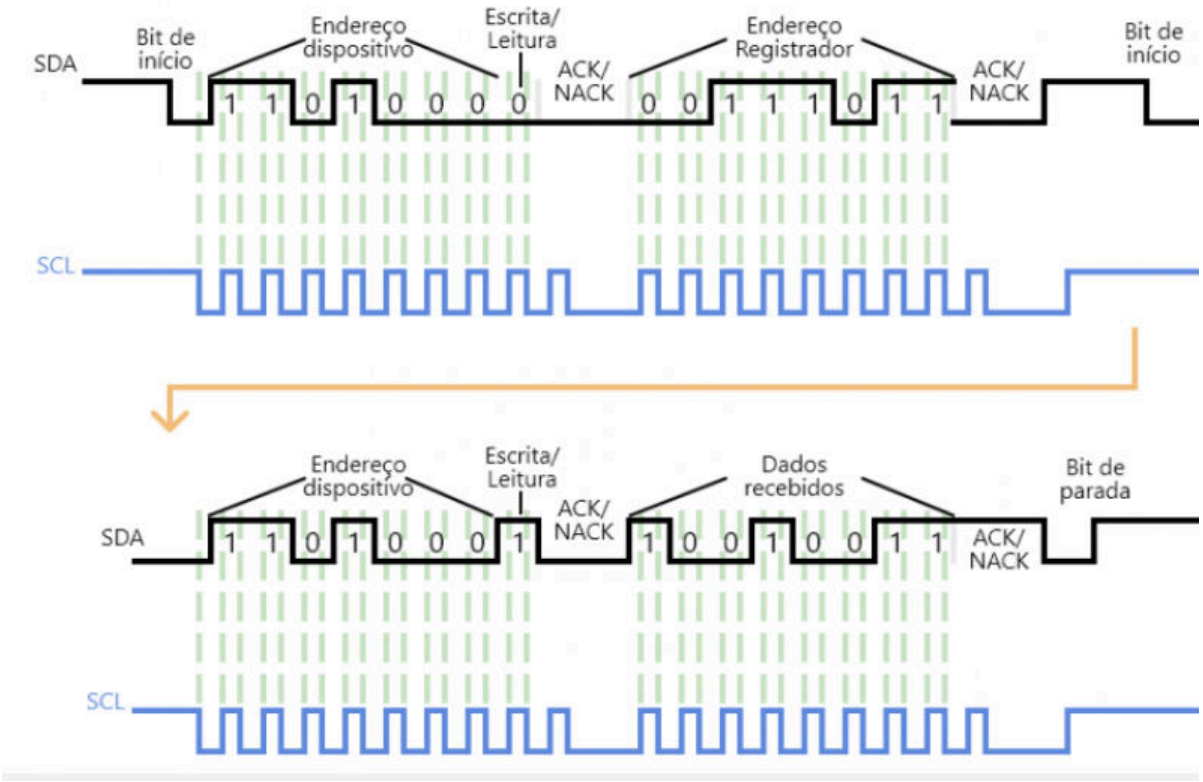
3. Ao fim do endereço do escravo, o *bit* de leitura 1 deve ser enviado.

4. Em seguida o escravo deve enviar um *bit* de ACK.

5. Se este último ACK deu certo, o escravo finalmente envia os dados que estão gravados no registrador. Em sequência, agora será o mestre que deve enviar o ACK. Pode ser que o registrador possua mais de um *byte* de espaço. Se for o caso, este passo e o ACK (do mestre) se repetem até todos os dados serem enviados pelo escravo.
6. A penúltima informação enviada deve ser um ACK do mestre.
7. Por último, quando não há mais nada a ser recebido pelo mestre, ocorre um *Stop bit* (a linha SDA passa de nível baixo para nível alto enquanto a linha SCL está em alta.

A Figura 127 apresenta um exemplo de leitura. O endereço do escravo é 0x68 (1101000), o registrador é 0x3B (00111011) e os dados enviados pelo escravo é 10010011.

Figura 127 – Exemplo de comunicação de leitura no I2C



Fonte: (GUIMARÃES, 2018)

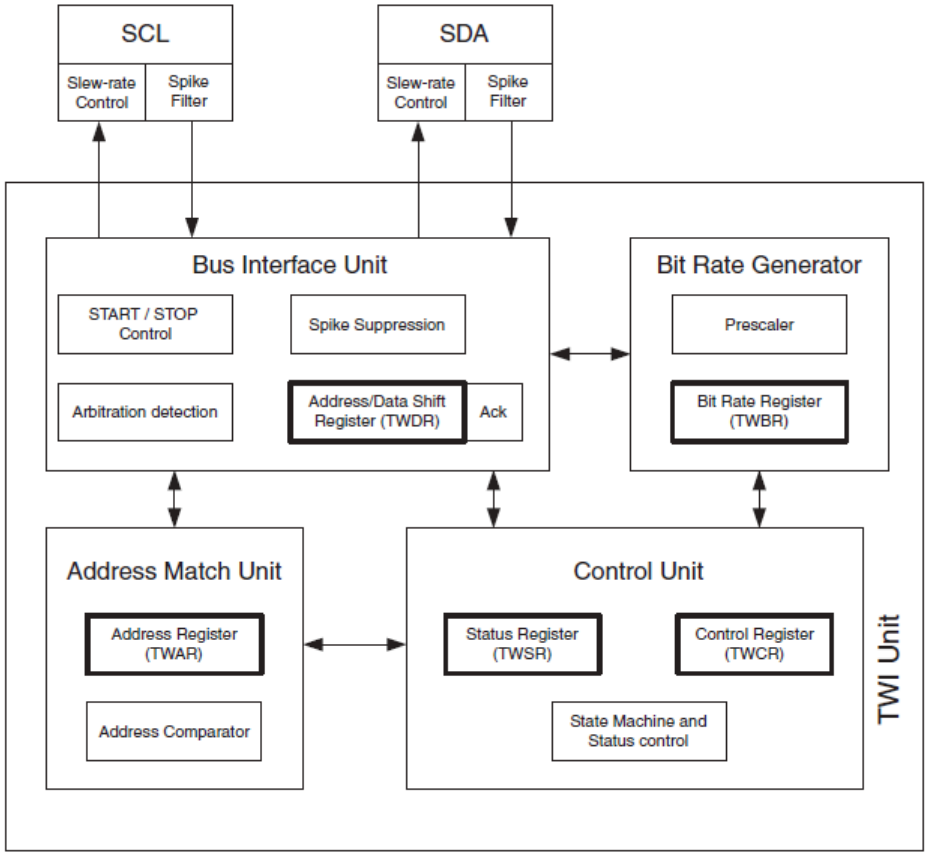
A Figura 128 apresenta o diagrama de bloco do módulo I2C dentro do ATmega2560.

A equação C.1 apresenta a equação de seleção da frequência do *clock*, isso quando ele é utilizado como mestre.

$$SCLfrequency = \frac{CPU\ Clock\ frequency}{16 + 2(TWBR) \times 4^{TWPS}} \tag{C.1}$$

Onde TWBR é o valor do registrador TWI *Bit Rate* e TWPS é o valor dos *bits* de *prescaler* no registrador TWI *Status Register*. O *prescaler* pode ser configurado conforme a figura 129.

Figura 128 – Diagrama de blocos do I2C no microcontrolador da plataforma Arduino Mega 2560



Fonte: Datasheet

Figura 129 – Prescaler do I2C no ATmega2560

Table 24-7. TWI Bit Rate Prescaler

TWPS1	TWPS0	Prescaler Value
0	0	1
0	1	4
1	0	16
1	1	64

Fonte: (ATMEL, 2014)

ANEXO D – Tipos de *Frames* no XBee

A seguir será apresentado detalhadamente alguns tipos de *frame* e sua construção.

D.1 Comando AT

Usado para consultar ou definir os parâmetros do módulo no dispositivo local. Este comando de API aplica as alterações após a execução do comando. As alterações feitas nos parâmetros do módulo entram em vigor assim que as alterações são aplicadas. A Figura 130 apresenta o *frame* em detalhes e um exemplo de comando AT para consulta do parâmetro NJ do módulo.

Figura 130 – Exemplo de *frame* detalhado de comando AT

Frame Fields		Offset	Example	Description
Start Delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and the checksum
		LSB 2	0x04	
Frame-specific Data	Frame Type	3	0x08	
	Frame ID	4	0x52 (R)	Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgment). If set to 0, no response is sent.
	AT Command	5	0x4E (N)	Command Name - Two ASCII characters that identify the AT command.
		6	0x4A (J)	
	Parameter Value (optional)			If present, indicates the requested parameter value to set the given register. If no characters present, register is queried.
Checksum		7	0x0D	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Fonte: (DIGI, 2018)

D.2 Requisição de Transmissão

Um *frame* de API de solicitação de transmissão faz com que o módulo envie dados como um pacote de RF para o destino especificado.

O endereço de destino de 64 *bits* deve ser definido como 0x000000000000FFFF para uma transmissão de *broadcast* (envio para todos os dispositivos). O coordenador pode ser endereçado configurando o endereço de 64 *bits* para todos os *bits* como zero (0x0000000000000000) e o endereço de 16 *bits* para 0xFFFE, ou configurando o endereço de 64 *bits* e o endereço de 16 *bits* todos como zero. Para outras transmissões, configurar o endereço de 16 *bits* para o endereço de 16 *bits* correto pode ajudar a melhorar o desempenho

ao transmitir para vários destinos. Se um endereço de 16 *bits* não for conhecido, este campo deve ser definido como 0xFFFE (desconhecido). O *frame* Status de transmissão (0x8B) indicará o endereço de 16 *bits* descoberto se for bem sucedido.

O raio de transmissão pode ser definido de 0 a NH. Se definido como zero, o raio de transmissão será o máximo possível. Este parâmetro é usado apenas para transmissões broadcast. O número máximo de **bytes** de *payload* pode ser lido com o comando NP. Se for utilizada criação de rotas, o *payload*, do RF será reduzido em dois *bytes* por salto intermediário na rota de origem.

A Figura 131 apresenta em detalhes o *frame* de Requisição de transmissão e apresenta o exemplo de transmissão para o módulo de endereço de 64 *bits* 0x0013A200400A0127, sem conhecimento do endereço de 16 *bits* (0xFFFE), cujo o conteúdo da mensagem é "TxData1B".

D.3 Resposta de comando AT

Em resposta a uma mensagem de comando AT, o módulo enviará uma mensagem de resposta de comando AT. Alguns comandos enviarão de volta vários *frames* (por exemplo, o comando ND (*Node Discover*). A Figura 132 apresenta uma mensagem de resposta de comando AT, supondo que foi realizado a alteração bem sucedida do parâmetro BD no dispositivo local.

D.4 Status de Transmissão

Quando uma solicitação de transmissão é concluída, o módulo envia uma mensagem de status de transmissão. Esta mensagem indicará se o pacote foi transmitido com sucesso ou se houve uma falha. A Figura 133 apresenta em detalhes o *frame*, com a suposição de que foi realizado uma transmissão de dados *unicast* para um dispositivo de destino com endereço de 16 *bits* de 0x7D84.

D.5 Recepção de pacotes

Quando o módulo recebe um pacote RF, ele é enviado pela UART usando este tipo de mensagem. A Figura 134 apresenta o *frame* detalhadamente e com exemplo supondo que o dispositivo com endereço de 64 *bits* de 0x0013A20040522BAA e endereço de 16 *bits* 0x7D84 enviou uma transmissão de dados unicast para um dispositivo remoto com o *payload* "RxData". Se o parâmetro AO é igual à zero no dispositivo receptor, a mensagem será igual a da figura 134.

Figura 131 – Exemplo de *frame* detalhado de requisição de transmissão

Frame Fields		Offset	Example	Description
Start Delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and the checksum
		LSB 2	0x16	
Frame-specific Data	Frame Type	3	0x10	
	Frame ID	4	0x01	Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgment). If set to 0, no response is sent.
	64-bit Destination Address	MSB 5	0x00	Set to the 64-bit address of the destination device. The following addresses are also supported: 0x0000000000000000 - Reserved 64-bit address for the coordinator 0x000000000000FFFF - Broadcast address
		6	0x13	
		7	0xA2	
		8	0x00	
		9	0x40	
		10	0x0A	
		11	0x01	
		LSB 12	0x27	
	16-bit Destination Network Address	MSB 13	0xFF	Set to the 16-bit address of the destination device, if known. Set to 0xFFFE if the address is unknown, or if sending a broadcast.
		LSB 14	0xFE	
	Broadcast Radius	15	0x00	Sets maximum number of hops a broadcast transmission can occur. If set to 0, the broadcast radius will be set to the maximum hops value.
	Options	16	0x00	Bitfield of supported transmission options. Supported values include the following: 0x01 - Disable retries and route repair 0x20 - Enable APS encryption (if EE=1) 0x40 - Use the extended transmission timeout Enabling APS encryption presumes the source and destination have been authenticated. I also decreases the maximum number of RF payload bytes by 4 (below the value reported by NP). The extended transmission timeout is needed when addressing sleeping end devices.It also increases the retry interval between retries to compensate for end device polling. See Transmission timeouts for a description. Unused bits must be set to 0.
	RF Data	17	0x54	Data that is sent to the destination device
		18	0x78	
		19	0x44	
		20	0x61	
		21	0x74	
		22	0x61	
		23	0x30	
		24	0x41	
Checksum		25	0x13	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Fonte: (DIGI, 2018)

Figura 132 – Exemplo de *frame* detalhado de resposta de comando AT

Frame Fields		Offset	Example	Description
Start Delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and the checksum
		LSB 2	0x05	
Frame-specific Data	Frame Type	3	0x88	
	Frame ID	4	0x01	Identifies the UART data frame being reported. Note If Frame ID = 0 in AT Command Mode, no AT Command Response will be given.
	AT Command	5	'B' = 0x42	Command Name - Two ASCII characters that identify the AT Command.
		6	'D' = 0x44	
	Command Status	7	0x00	0 = OK 1 = ERROR 2 = Invalid Command 3 = Invalid Parameter 4 = Tx Failure
	Command Data			Register data in binary format. If the register was set, then this field is not returned, as in this example.
Checksum		8	0xF0	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Fonte: (DIGI, 2018)

Figura 133 – Exemplo de *frame* detalhado de status da transmissão

Frame Fields		Offset	Example	Description
Start Delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and the checksum
		LSB 2	0x07	
Frame-specific Data	Frame Type	3	0x8B	
	Frame ID	4	0x01	Identifies the UART data frame being reported. Note: If Frame ID = 0 in AT Command Mode, no AT Command Response will be given.
	16-bit address of destination	5	0x7D	16-bit Network Address the packet was delivered to (if successful). If not successful, this address will be 0xFFFD: Destination Address Unknown.
		6	0x84	
	Transmit Retry Count	7	0x00	The number of application transmission retries that took place.
	Delivery Status	8	0x00	0x00 = Success 0x01 = MAC ACK Failure 0x02 = CCA Failure 0x15 = Invalid destination endpoint 0x21 = Network ACK Failure 0x22 = Not Joined to Network 0x23 = Self-addressed 0x24 = Address Not Found 0x25 = Route Not Found 0x26 = Broadcast source failed to hear a neighbor relay the message 0x2B = Invalid binding table index 0x2C = Resource error lack of free buffers, timers, and so forth. 0x2D = Attempted broadcast with APS transmission 0x2E = Attempted unicast with APS transmission, but EE=0 0x32 = Resource error lack of free buffers, timers, and so forth. 0x74 = Data payload too large
	Discovery Status	9	0x01	0x00 = No Discovery Overhead 0x01 = Address Discovery 0x02 = Route Discovery 0x03 = Address and Route 0x40 = Extended Timeout Discovery
Checksum		10	0x71	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Fonte: (DIGI, 2018)

Figura 134 – Exemplo de *frame* detalhado de recepção de pacotes via radio frequência

Frame Fields		Offset	Example	Description
Start Delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and the checksum
		LSB 2	0x11	
Frame-specific Data	Frame Type	3	0x90	
	64-bit Source Address	MSB 4	0x00	
		5	0x13	64-bit address of sender. Set to 0xFFFFFFFFFFFFFFFF (unknown 64-bit address) if the sender's 64-bit address is unknown.
		6	0xA2	
		7	0x00	
		8	0x40	
		9	0x52	
		10	0x2B	
		LSB 11	0xAA	
	16-bit Source Network Address	MSB 12	0x7D	16-bit address of sender
		LSB 13	0x84	
	Receive Options	14	0x01	0x01 - Packet Acknowledged 0x02 - Packet was a broadcast packet 0x20 - Packet encrypted with APS encryption 0x40 - Packet was sent from an end device (if known) Note Option values can be combined. For example, a 0x40 and a 0x01 will show as a 0x41. Other possible values 0x21, 0x22, 0x41, 0x42, 0x60, 0x61, 0x62.
	Received Data	15	0x52	Received RF data
		16	0x78	
		17	0x44	
		18	0x61	
		19	0x74	
		20	0x61	
Checksum		21	0x0D	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Fonte: (DIGI, 2018)