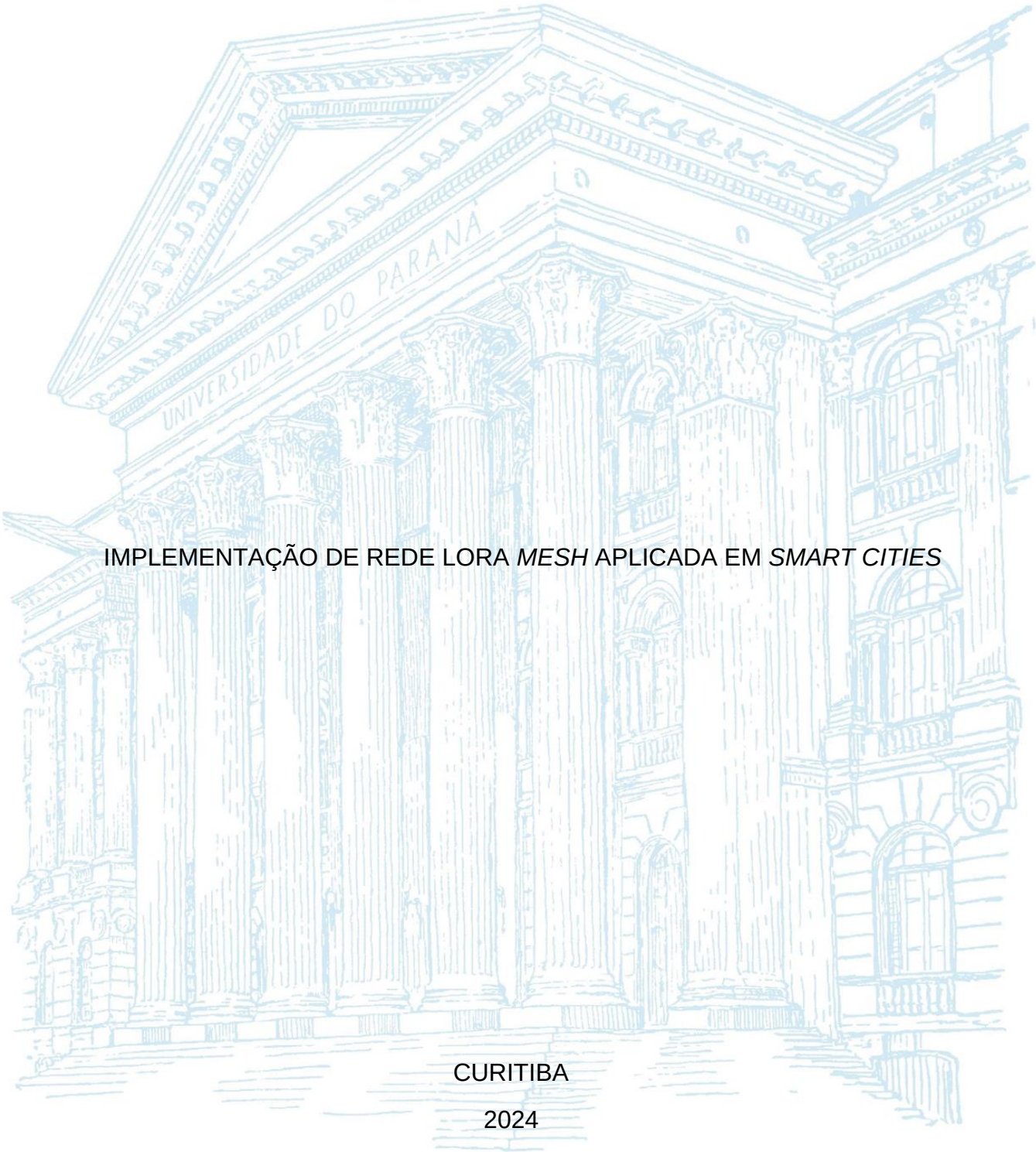
UNIVERSIDADE FEDERAL DO PARANÁ
CURSO DE ENGENHARIA ELÉTRICA

MYLENA DE MORAES EICH
PRISCILA AMI SUZUKI



IMPLEMENTAÇÃO DE REDE LORA *MESH* APLICADA EM *SMART CITIES*

CURITIBA
2024

MYLENA DE MORAES EICH
PRISCILA AMI SUZUKI

IMPLEMENTAÇÃO DE REDE LORA *MESH* APLICADA EM *SMART CITIES*

Trabalho de conclusão de curso apresentado ao curso de Engenharia Elétrica da Universidade Federal do Paraná como requisito parcial à obtenção do bacharelado em Engenharia Elétrica com ênfase em sistemas eletrônicos embarcados.

Orientador: Prof. Dr. Marcelo Eduardo Pellenz.

CURITIBA
2024

TERMO DE APROVAÇÃO

MYLENA DE MORAES EICH
PRISCILA AMI SUZUKI

IMPLEMENTAÇÃO DE REDE LORA MESH APLICADA EM SMART CITIES

Trabalho de conclusão de curso aprovado como requisito parcial para obtenção do grau de bacharel no Curso de Engenharia Elétrica da Universidade Federal do Paraná, pela seguinte banca examinadora:

_____

Orientador: Prof. Dr. Marcelo Eduardo Pellenz
Departamento de Engenharia Elétrica, UFPR

_____

Prof. Dr. Cláudio Bastos da Silva
Departamento de Engenharia Elétrica, UFPR

_____

Prof. Dr. Ândrei Camponogara
Departamento de Engenharia Elétrica, UFPR

Curitiba, 12 de dezembro de 2024.

AGRADECIMENTOS

Agradecemos, primeiramente, ao Professor Marcelo Eduardo Pellenz, pela orientação e incentivo no desenvolvimento deste trabalho, cuja contribuição foi essencial para a obtenção dos resultados aqui apresentados.

Expressamos também nossa gratidão aos colegas que estiveram presentes nessa trajetória acadêmica. Ao Henrique, pelo suporte em conhecimentos de software e pela disposição em colaborar; à Natália, pela amizade, pelas risadas e pelo companheirismo ao longo do curso; à Kathe e Hanna, pelas conversas descontraídas nos corredores, que tornaram a jornada mais leve e agradável. A todos os demais colegas que, de alguma forma, contribuíram para o nosso crescimento, deixamos nosso sincero reconhecimento.

Agradecemos, ainda, aos nossos namorados, cujo apoio, compreensão e incentivo foram fundamentais neste percurso, nos ajudando a enfrentar os desafios com serenidade e motivação.

Por fim, demonstramos nossa gratidão à família, cujo carinho, incentivo e paciência estiveram sempre presentes. Vocês foram o porto seguro que nos acolheu após cada etapa do caminho, demonstrando que, com amor e perseverança, é possível superar qualquer dificuldade.

RESUMO

Este trabalho tem como objetivo implementar e avaliar o desempenho de um protocolo de roteamento para redes LoRa *mesh* (malha), para aplicações com comunicação *multi-hop* (múltiplos saltos) sem fio para cidades inteligentes. Para isso, foi realizado um estudo e definição da placa de desenvolvimento, para então realizar a seleção do protocolo de roteamento. Então, foram realizados os ajustes necessários para a implementação do protocolo LoRaMesher com a função de múltiplos saltos, e desenvolvido um canal de comunicação TCP, para aplicar a implementação de um *gateway* (ponte entre camada da roteamento e a camada de aplicação) na rede. Além disso foram acrescentadas métricas para análise do desempenho do protocolo como a taxa de entrega de pacotes e a sobrecarga de controle. Em experimentos de campo, foi possível observar troca de mensagens entre os dispositivos e atualização da tabela de rotas de cada um com a quantidade de saltos entre eles. Os resultados confirmam o funcionamento proposto nos objetivos para cenários *multi-hop*, contribuindo para aplicações em cidades inteligentes.

Palavras-chave: LoRa. IoT. Cidade Inteligente.

ABSTRACT

This work aims to implement and evaluate the performance of a routing protocol for LoRa mesh networks, focusing on multi-hop wireless communication for smart cities applications. To achieve this, the study began by selecting and defining the development board, followed by choosing the appropriate routing protocol. Necessary adjustments were then made to implement the LoRaMesher protocol with multi-hop functionality, and a TCP socket was developed to apply the implementation of a gateway in the network. Additionally, metrics such as Packet Delivery Ratio and and Control Overhead were included for performance analysis. Field experiments confirmed that devices exchanged messages and updated their routing tables, including the number of hops between them. The results validate the proposed objectives for multi-hop scenarios, contributing to applications in smart cities.

Key words: LoRa. Multi-Hop. Smart City.

LISTA DE ILUSTRAÇÕES

FIGURA 1 – CAMADAS MODELO TCP/IP 24
FIGURA 2 – ESTRUTURA PADRÃO DE PACOTE LORA.....28
FIGURA 3 – ESTRUTURA DE PACOTE LORAMESHER 30
FIGURA 4 – ESTRUTURA DE PACOTE DE ROTEAMENTO 31
FIGURA 5 – ESTRUTURA DE PACOTE DE DADOS.....31
FIGURA 6 – ROTINA DE PROCESSAMENTO DE PACOTE DE ROTEAMENTO...32
FIGURA 7 – ROTINA DE PROCESSAMENTO DE PACOTE DE DADOS 33
FIGURA 8 – DIAGRAMA DE ROTAS COM NÚMERO DE SALTOS 33
FIGURA 9 – INTEGRAÇÃO ENTRE AS FILAS E AS TAREFAS.....36
FIGURA 10 – ARQUITETURA LORANET 39
FIGURA 11 – ESTRUTURA DE PACOTE DE APLICAÇÃO LORANET 39
FIGURA 12 – ESTRUTURA DE PACOTE DE ROTEAMENTO LORANET 40
FIGURA 13 – CASE EM IMPRESSÃO 3D.....41
FIGURA 14 – PLACA HELTEC WIFI LORA v2.....43
FIGURA 15 – FLUXOGRAMA APLICADO NA METODOLOGIA 46
FIGURA 16 – PLACA HELTEC MOSTRANDO RSSI 48
FIGURA 17 – MAPA DO CAMPUS POLITÉCNICO E POSIÇÕES DOS *NODES* 50
FIGURA 18 –POSICIONAMENTO DOS DISPOSITIVOS NO CAMPUS 51
FIGURA 19 – DIAGRAMA DE DISTÂNCIA ENTRE *NODES* E *GATEWAY* (8430) ..51
FIGURA 20 – FLUXOGRAMA PACOTE DE ROTAS ENVIADO.....53
FIGURA 21 – FLUXOGRAMA TABELA DE ROTAS RECEBIDA.....54
FIGURA 22 – FLUXOGRAMA ENVIA PACOTE DE DADOS.....54
FIGURA 23 – *GATEWAY* RECEBE DADOS E ENVIA *PAYLOAD* AO SERVIDOR..54
FIGURA 24 – GRÁFICO DE QUANTIDADE DE PACOTES POR *NODE* 55
FIGURA 25 – GRÁFICO RSSI POR *NODE* 57

LISTA DE TABELAS

TABELA 1 – COMPARATIVO DE TECNOLOGIAS 19
TABELA 2 – PARÂMETROS DE CONFIGURAÇÃO LORA.....27
TABELA 3 – PARÂMETROS DE QUALIDADE DE SINAL LORA.....27
TABELA 4 – COMPARATIVO CLASSE DE DISPOSITIVOS.....29
TABELA 5 – TAREFAS DE ROTEAMENTO LORAMESHER.....34
TABELA 6 – TAREFAS DE APLICAÇÃO LORAMESHER.....35
TABELA 7 – TAREFAS DE ROTEAMENTO LORANET.....37
TABELA 8 – COMPARATIVO DE BATERIAS.....42
TABELA 9 – COMPARATIVO DE PLACAS.....43
TABELA 10 – CUSTO POR DISPOSITIVO 44
TABELA 11 – PARAMETROS LORA.....45
TABELA 12 – RELAÇÃO ENTRE DISTÂNCIA E RSSI.....48
TABELA 13 – RESULTADO CALCULADO DE PDR 56

LISTA DE SÍMBOLOS

ADR – *Adaptative Data Range* (Faixa de Dados Adaptativa)
AODV – *Ad-Hoc Distance-Vector* (Vetorial de Distância Ad-Hoc)
API – *Application Programming Interface* (Interface de Programação de Aplicações)
BW – *Bandwidth* (Largura de Banda)
CR – *Coding Rate* (Taxa de Codificação)
CREA-PR – Conselho Regional de Engenharia e Agronomia do Paraná
CSS – *Chirp Spread Spectrum* (Espectro de Dispersão por "Chirp")
EEPROM – *Electrically Erasable Programmable Read-Only Memory* (Memória Somente de Leitura Programável e Apagável Eletricamente)
FAN – *Field Area Network* (Rede de Área de Campo)
FIFO – *First In First Ou*t (Primeiro a Entrar, Primeiro a Sair)
FreeRTOS – *Free Real-Time Operating System* (Sistema Operacional de Tempo Real Gratuito)
GSM – *Global System for Mobile Communications* (Sistema Global para Comunicações Móveis)
HTTP – *Hypertext Transfer Protocol* (Protocolo de Transferência de Hipertexto)
IANA – *Internet Assigned Numbers Authority* (Autoridade para Atribuição de Números da Internet)
IEEE – *Institute of Electrical and Electronics Engineers* (Instituto de Engenheiros Eletricistas e Eletrônicos)
IP – *Internet Protocol* (Protocolo de Internet)
IoT – *Internet of Things* (Internet das Coisas)
Li-Po – *Lithium-Polymer* (Lítio-Polímero)
LoRa – *Long Range* (Longo Alcance)
LoRaWAN – *Long Range Wide Area Network* (Rede de Longo Alcance em Área Ampla)
LPWAN – *Low Power Wide Area Network* (Rede de Área Ampla de Baixa Potência)
MAC – *Medium Access Control* (Controle de Acesso ao Meio)
MANET – *Mobile Ad-hoc Network* (Rede Ad-Hoc Móvel)

NB-IoT – *Narrowband Internet of Things* (Internet das Coisas em Banda Estreita)

PDR – *Packet Delivery Ratio* (Taxa de Entrega de Pacotes)

RSSF – *Relative Signal Strength Factor* (Fator de Força Relativa do Sinal)

RSSI – *Received Signal Strength Indicator* (Indicador de Intensidade do Sinal Recebido)

SF – *Spreading Factor* (Fator de Espalhamento)

SNR – *Signal-to-Noise Ratio* (Relação Sinal-Ruído)

TCP – *Transmission Control Protocol* (Protocolo de Controle de Transmissão)

TP – *Transmission Power* (Potência de Transmissão)

TTL – *Time to Live* (Tempo de Vida)

UDP – *User Datagram Protocol* (Protocolo de Datagrama do Usuário)

Wi-Fi – *Wireless Fidelity* (Fidelidade Sem Fio)

ZDO – *ZigBee Device Objects* (Objetos de Dispositivo ZigBee)

SUMÁRIO

1 INTRODUÇÃO 12

1.1 OBJETIVOS 13

1.1.1 Objetivo Geral 13

1.1.2 Objetivos Específicos 13

1.3 JUSTIFICATIVA 14

1.4 ESTRUTURA DA DISSERTAÇÃO 15

2 REVISÃO DA LITERATURA 16

2.1 CONCEITO DE CIDADES INTELIGENTES E IOT 16

2.2 TECNOLOGIAS E INFRAESTRUTURA DE REDE 16

2.2.1 Tecnologia LoRa 16

2.2.2 Tecnologia NB-IoT 17

2.2.3 Tecnologia Wi-SUN 18

2.2.4 Tecnologia ZigBee 18

2.3 CASOS DE USO 20

2.3.1 Belo Horizonte – MG 20

2.3.2 Londrina – PR 20

2.3.3 Cascavel – CE 21

2.3.4 Joinville – SC 21

2.3.5 Guaratuba – PR 22

3 FUNDAMENTAÇÃO TEÓRICA 23

3.1 IoT 23

3.2 ARQUITETURA DE REDE 23

3.2.1 Camada de Aplicação 24

3.2.2 Camada de Transporte 24

3.2.3 Camada de Rede 25

3.2.4 Camada de Enlace 25

3.2.5 Camada de Física 25
3.3 TECNOLOGIA LORA 26
3.3.1 Protocolo LoraWAN 28
3.3.2 Protocolo LoraMesher 29
3.3.3 Protocolo LoraNet 36
3.3.4 Escolha do protocolo 40
4 MATERIAIS E MÉTODOS 41
4.1 MATERIAIS 41
4.1.1 Heltec WiFi LoRa 32 (V2) 42
4.1.2 Custos 43
4.2 METODOLOGIA APLICADA 44
4.3 DESENVOLVIMENTO 46
4.4 FERRAMENTAS DE DESENVOLVIMENTO DE SOFTWARE 47
4.4.1 Git 47
4.4.2 PlatformIO 47
4.5 VALIDAÇÃO 48
4.5.1 Teste de Alcance 48
4.5.2 Teste com Protocolo LoraMesher 49
5 RESULTADOS E DISCUSSÃO 53
6 CONCLUSÕES E TRABALHOS FUTUROS 58
REFERÊNCIAS 60
APÊNDIE A – PARSER.PY 64
APÊNDIE B – MAIN.CPP 65

1 INTRODUÇÃO

Com o crescimento da popularidade das tecnologias IoT é inegável que o aumento da conectividade nas cidades acelerou o ritmo de fluxo das informações, e assim o conceito de cidades inteligentes está se tornando o foco para o avanço de muitos setores, abrangendo aplicações em ambientes urbanos, industriais e rurais.

Nesse cenário, é possível obter resultados com eficiência e produtividade através da integração de informações, contribuindo com a qualidade de serviços de mobilidade, segurança, sustentabilidade e produtividade através de uma tecnologia mais assertiva e otimizada.

Assim, para aplicar este conceito de cidades conectadas, uma das abordagens possíveis seria a aplicação de uma rede de malha que realiza uma interconexão dos diferentes dispositivos presentes no ambiente. E para tornar a comunicação possível e eficiente, são utilizados protocolos de comunicação, como: GSM, ZigBee, Wi-Fi, Bluetooth, entre outros. Entretanto, muitos protocolos consolidados no mercado não necessariamente são otimizados para aplicações de baixo custo, com pouco consumo de energia e com longo alcance, tornando menos adequado o uso dessa tecnologia em certos cenários.

Com isso, o protocolo de comunicação LoRa (*Long Range*) têm se destacado no mercado devido às suas características que se adaptam bem a essas aplicações. A tecnologia LoRa possibilita a implantação de redes sem fio com longo alcance e baixo consumo de energia, com potencial de inserção em ambientes em larga escala.

Dessa forma, esse trabalho tem como proposta estudar a tecnologia LoRa, otimizar seu potencial e analisar a sua viabilidade no uso em *smart cities* cidades inteligentes, com o desenvolvimento de uma rede *mesh* (malha) através um protocolo *open source* (código aberto). Através da motivação pela busca de melhorar a eficiência e custo-benefício de dispositivos e conectados, a escolha do tema vem da necessidade de adaptação das tecnologias de forma mais ágil e ampla.

Ao longo deste trabalho, será apresentado um estudo aprofundado sobre as tecnologias IoT já amplamente utilizadas, destacando as vantagens do uso do protocolo LoRa e comparando com outros protocolos existentes comumente usados para aplicações em *smart cities*. Além disso, serão abordadas questões relacionadas à implementação de uma rede *mesh* com este protocolo, considerando desafios,

benefícios e potenciais impactos na eficiência de dispositivos utilizando essa tecnologia.

Dessa forma, este trabalho visa contribuir para o avanço da pesquisa de soluções inovadoras IoT para *smart cities*, explorando o potencial da tecnologia LoRa na construção de uma rede *mesh* eficiente e adaptável à demanda urbana moderna.

1.1 OBJETIVOS

Com o desenvolvimento deste trabalho, procura-se estudar e compreender o funcionamento das tecnologias de rede Lora, a fim de obter tais conhecimentos e colaborar com o avanço das redes IoT no Brasil. Então espera-se que por meio dos experimentos e resultados obtidos através deste, possam ser relevantes à sociedade para impulsionar o crescimento e popularidade das tecnologias Lora.

1.1.1 Objetivo Geral

Este trabalho tem como objetivo implementar e avaliar o desempenho de um protocolo de roteamento para redes LoRa *mesh*, com foco em aplicações com comunicação *multi-hop* sem fio para *smart cities*.

1.1.2 Objetivos Específicos

Os objetivos específicos deste trabalho incluem:

- Realizar uma revisão bibliográfica sobre a tecnologia LoRa e aplicações já existentes no mercado, além dos principais protocolos de roteamento para topologias *mesh*.
- Selecionar e avaliar um protocolo de roteamento *open source* para redes LoRa *mesh.*
- Implementação e avaliação de desempenho de um cenário de teste em ambiente *outdoor* (ao ar livre) *de* uma rede LoRa *mesh*.

1.3 JUSTIFICATIVA

No contexto das grandes cidades que visam cada vez mais alcançar a modernização em conceitos como s*mart cities*, nada mais instintivo do que começar pensando nas melhorias nos serviços básicos do direito de todo cidadão. Assim vêm observando um crescimento exponencial do mercado em torno das inovações que se aplicam aos cenários urbanos, visando tanto otimizar o dia a dia nas cidades como melhorar aspectos como qualidade de vida e segurança pública.

Um exemplo disso, é ao analisar os dados do CREA-PR de 2016 que mostram que, no Brasil, a iluminação pública é responsável por uma parcela de aproximadamente 4,5% de toda a demanda do Sistema de Energia Elétrica Nacional e representa um consumo de 3% do total de energia elétrica do país, com uma distribuição de aproximadamente 15 milhões de pontos de luz. Tais dados mostram como a parcela de consumo de energia elétrica que demandam as redes de iluminação necessitam de otimizações para melhoria na eficiência energética das cidades (CREA-PR, 2016).

Com a inclusão das tecnologias IoT no mercado e as diversas aplicações que possam estar enriquecendo os ambientes urbanos, é preciso estudar as soluções disponíveis para encontrar as que melhor atendem as funcionalidades e que principalmente resolvam a questão da eficiência energética das redes de iluminação públicas.

Por essas razões, destaca-se a escolha da tecnologia LoRa por suas características de economia de energia e como o próprio nome diz *Long Range,* ou seja, trabalha com alcance a longas distâncias o que torna a tecnologia ideal para aplicações em áreas urbanas que precisam de capacidade de penetração em ambientes com muitos obstáculos como construções, prédios em geral, e também em ambientes rurais que não possuem muitos obstáculos porém por possuir áreas extensas, precisam de uma boa cobertura da região.

Apesar de tudo há desafios na escolha da tecnologia em questão, principalmente se quer utilizar em aplicações de protocolos de roteamento para redes em malha, uma vez que a tecnologia mais comumente utilizada atualmente é ponto-a-ponto.

Diante disso, a motivação deste trabalho é implementar e avaliar o protocolo de roteamento para o funcionamento de redes LoRa em malha, utilizando como

aplicação em *smart cities*. Após essa coleta e análise dos resultados, haverá a possibilidade de propor melhorias no roteamento e transmissão de pacotes de dados usando a rede LoRa.

1.4 ESTRUTURA DA DISSERTAÇÃO

A estrutura deste trabalho está organizada de maneira a introduzir os conhecimentos necessários para a implementação da rede e o detalhamento da aplicação prática para testes em campo, estando separados por capítulos.

No capítulo 2 apresentamos os conceitos das principais tecnologias utilizadas atualmente no contexto das aplicações em cidades inteligentes, exemplificando alguns casos de uso aplicados. No capítulo 3 são apresentados os conceitos das camadas de redes e são apresentadas as opções de protocolos que foram avaliadas para a implementação, justificando a escolha do protocolo a ser implementado.

A partir disso, no capítulo 4 é apresentado o hardware e componentes utilizados no desenvolvimento, a metodologia a ser aplicada em todo o trabalho, assim como o desenvolvimento para a implementação do protocolo e a validação da implementação com a aplicação de testes. Por fim, no capítulo 5 são apresentados os resultados, encerrando no capítulo 6 com os comentários e considerações finais.

2 REVISÃO DA LITERATURA

Neste capítulo será apresentada uma revisão dos conceitos de cidades inteligentes e abordadas as principais tecnologias existentes nesse tipo de rede. Serão discutidas as principais características e funcionalidades de cada uma com o intuito de pontuar as escolhas das autoras para a definição do protocolo de rede que será utilizado neste trabalho. Além disso, também serão contempladas as soluções encontradas que estão em desenvolvimento ou que já foram desenvolvidas para as aplicações em cidades inteligentes.

2.1 CONCEITO DE CIDADES INTELIGENTES E IOT

Uma cidade inteligente é descrita como uma cidade que abrange as aplicações de tecnologias da informação e comunicação e a execução de métodos que certamente afetam a vida dos cidadãos (HASSAN et al., 2023).

O objetivo de se construir uma cidade inteligente é melhorar a qualidade de vida dos habitantes por meio da gestão urbana, de forma sustentável e inovadora. As *smart cities* utilizam a Internet das Coisas (IoT) para coletar os dados e atuar em diversas áreas como mobilidade, energia, saúde e segurança por meio de dispositivos conectados à rede, de forma inteligente.

A rede LoRa tem sido amplamente adotada em soluções IoT *outdoor* devido ao seu longo alcance e eficiência energética. Espera-se que até 2024, 3,6 bilhões de conexões LPWAN sejam estabelecidas (DEVALAL e KARTHIKEYAN, 2018).

2.2 TECNOLOGIAS E INFRAESTRUTURA DE REDE

Nos próximos tópicos serão comentadas algumas das tecnologias aplicadas no contexto de cidades inteligentes e por fim é apresentada uma tabela comparativa técnica das tecnologias.

2.2.1 Tecnologia LoRa

É uma camada física com tecnologia de comunicação de baixo consumo de energia e longo alcance operando em bandas de rádio não licenciadas no nível sub-

gigahertz (DEVALAL e KARTHIKEYAN, 2018, p. 284-290). Essa tecnologia é transmitida através de *gateways* (ponte de comunicação) LoRa que atuam como pontes de dispositivos finais para um servidor de rede. Os dispositivos finais usam comunicação de rádio *single-hop* (um único salto) com *gateways*, que então se conectam ao servidor de rede por meio de conexões IP, por exemplo, Ethernet, 3G ou WiFi.

A tecnologia LoRa segue o padrão IEEE 802.15.4 e trabalha na faixa de frequências não licenciadas de 915 a 928 MHz no Brasil. Possui um alcance alto de até 5 km em ambiente urbano, entre *gateways*, e seu consumo econômico são as principais vantagens na sua escolha, como é possível verificar na Tabela 1.

O uso mais comum da tecnologia LoRa é em arquitetura estrela que faz a comunicação *single-hop* entre o *gateway* e os dispositivos finais. Porém o objetivo do presente trabalho é fazer o uso de uma comunicação *multi-hop*, criando uma arquitetura de malha e dando capacidade aos dispositivos finais de se comunicar entre si sem a interface pelo *gateway*.

2.2.2 Tecnologia NB-IoT

O NB-IoT é uma rede LPWAN que usa a tecnologia 3G e 4G de redes celulares. Essas redes não utilizam uma faixa de frequência livre de licença, são operadas por redes comercializadas com base em assinatura de dados e são controladas pelas operadoras móveis. Por conta disso, deve seguir as condições da operadora em termos de disponibilidade de cobertura, custo do serviço e número de dispositivos conectados, e não pode lidar com a forte onda de interferência de dispositivos IoT devido à densa população de dispositivos celulares (3GPP, 2016).

Por ser uma rede LPWAN possui capacidade de cobertura de grandes áreas e que não necessitem de um alto volume de transmissão de dados, e por se utilizar de redes móveis, o NB-IoT garante uma transferência de dados estável e confiável. Por essas características essa tecnologia é fortemente aplicada no agronegócio, tanto pela alta cobertura, mas como pela sua capacidade de transmissão serem apenas dados sensoriais locais (temperatura, umidade etc.).

O desenvolvimento dessa tecnologia foi feito pela 3GPP e dentre as faixas de frequências licenciadas que ele trabalha vão de 800 a 1.900 MHz. Possui um alcance médio alto, de até 5 km entre *gateways* para o cenário urbano, porém por utilizar de

outras redes já existentes e licenciadas, o seu custo é aumentado. Porém, por esse mesmo motivo, a sua rede se torna mais confiável e estável.

2.2.3 Tecnologia Wi-SUN

O protocolo de rede Wi-SUN é uma rede voltada a empresas de *utilities* (utilidades) e funciona na topologia *mesh*, ou em rede. Assim como a LoRa, são para baixo tráfego de dados e não competem com as redes Wi-Fi ou rede celular móvel, destinadas para elevado tráfego de dados (Wi-SUN Alliance, 2020).

Das especificações, a Wi-SUN é aberta e se baseia nos padrões IEEE, IETF e ANSI/TIA, e foi desenvolvida para ser uma rede FAN (*Field Area Network*) utilizando o padrão IEE802.15.4g para a camada física. Utiliza baixa potência e endereçamento IPv6 para os nós da rede. A arquitetura desse é no modelo de camadas OSI.

Além disso, a Wi-SUN pode trabalhar dois protocolos de roteamento, o RPL e o MHDS, que funciona na camada de link. Já para a camada de transporte funciona utilizando os protocolos de transmissão UDP/TCP. A faixa de frequências que a rede Wi-SUN trabalha é a mesma da LoRa, de 902MHz a 928MHz para o Brasil, como representado na Tabela 1.

De modo geral, as vantagens para o uso dessa tecnologia são a utilização de padrões abertos IEEE802.15.4g/IPv6, infraestrutura simples, topologia em malha, bom alcance de até 3 km no cenário urbano, e suporte para funcionar em frequências globais homologadas, além de ter a opção para funcionamento por baterias (consumo moderado/baixo).

2.2.4 Tecnologia ZigBee

O protocolo ZigBee é um protocolo de comunicação sem fio também padronizado nas normas IEEE 802.15.4, através de controle de acesso na camada física para WPANs de baixa taxa de transmissão. A tecnologia possui alcance menor que os demais citados, de 10 a 100 metros, com a possibilidade de alimentação por bateria devido ao seu foco em baixo consumo de energia.

Apesar da sua limitação de alcance, o ZigBee possui uma gama de aplicações possíveis por ser vantajoso em ambientes isolados, devido a existência da tecnologia ZDOs (*ZigBee Device Objects*) que controla os dispositivos e suas funções dentro da rede, entregando uma maior segurança e gerenciamento da rede. O sistema

opera em 3 faixas de frequência na banda ISM: 868MHz, 915MHz e 2,4GHz, com taxa de dados de 20Kb/s a 250Kb/s. Para 915 MHz, que é a faixa que utilizaremos para esse projeto, o sistema suporta uma taxa máxima de dados de 40 Kb/s.

Referente às topologias de rede suportadas, o ZigBee trabalha com modo estrela, árvore e em malha. A vantagem no uso da tecnologia é que sua topologia em malha possui confiabilidade e a capacidade de autocorreção da sua rede. Seu sistema pode criar uma rede auto-organizável com o gerenciamento dos dispositivos, com comunicação multicanal.

Abaixo na Tabela 1 estão listadas as tecnologias citadas, como comparativo de suas especificações técnicas.

TABELA 1 – COMPARATIVO DE TECNOLOGIAS

Parâmetros	**LoRa**	**NB-IoT**	**Wi-SUN**	**Zigbee**
Padrão	IEEE 802.15.4	3GPP	IEEE 802.15.4g	IEEE 802.15.4
Alcance típico	2 a 5 km	1 a 5 km	1 a 3 km	100 m
Faixa de Frequência	Não licenciada (902-928 MHz no BR)	Licenciado 800 a 1.900 MHz	Não licenciado 915-928 MHz	Não Licenciado 868 MHz, 915 MHz, 2,4 GHz
Taxa de Dados	50 Kb/s	200 Kb/s	300 Kb/s	250Kb/s
Largura de Banda	125 a 250 kHz	180 a 200 kHz	200 kHz	2 MHz
Topologia	Estrela/ *Mesh*/ Híbrida	Rede Celular/ Estrela	Estrela/ *Mesh*	Estrela/Árvore/ *Mesh*
Consumo	Muito baixo	Baixo/ Moderado	Moderado/ Alto	Muito Baixo
Custo	Baixo	Moderado/Alto	Moderado	Baixo

FONTE: As autoras (2024).

2.3 CASOS DE USO

Nos próximos tópicos serão apresentados alguns casos de uso com aplicação de redes *mesh* desenvolvidas ou que estão em desenvolvimento no mercado.

2.3.1 Belo Horizonte – MG

Este estudo de caso se trata apenas de um dos vários projetos que estão em andamento para tornar a cidade de Belo Horizonte uma cidade inteligente. A cidade passou pela modernização de aproximadamente 180 mil pontos de IP com lâmpadas de vapor de sódio para lâmpadas LEDs, e 33 mil lâmpadas LED com sistema de gestão inteligente nas principais vias e parques da cidade. A BHIP, concessionária de iluminação pública de iluminação de Belo Horizonte, em parceria com a Logicalis, proveu não somente a troca das luminárias dos postes de todo parque de BH mas também a instalação de tecnologias e de um sistema completo de telegestão que permite monitorar e transmitir informações em tempo real, possibilitando a tomada de ações preditivas, reduzindo os índices de falhas e melhorando a eficiência do processo (LOGICALIS, 2021).

Das tecnologias aplicadas a esse projeto, foram substituídos os sensores fotoelétricos dos postes por microcontroladores, responsáveis pela comunicação baseada em redes *mesh*, com auxílio de concentradores conectados em *backhaul* por *Ethernet* ou via LTE. Das funcionalidades aplicadas, têm-se o controle de intensidade da luz, podendo desligar/ligar em horários programados remotamente, e coletar dados e antecipar manutenções ou abastecimentos nos pontos de iluminação (LOGICALIS, 2021).

2.3.2 Londrina – PR

Ainda nesse ano de 2024 foi aprovado pela prefeitura de Londrina o projeto para a implementação de telegestão na iluminação pública da cidade. Em parceria com o Lactec e o Senai, o projeto pretende utilizar a tecnologia de rede Wi-SUN para o desenvolvimento de até 4,5 mil unidades de módulo de telegestão e 220 concentradores para conectar e gerenciar os 65 mil pontos do parque de iluminação da cidade (LACTEC, 2024).

Até o momento foram estabelecidas apenas as funcionalidades de ligar, desligar, realizar a dimerização da intensidade luminosa, e realizar o monitoramento dos dados de tensão, corrente, temperatura, etc para identificar problemas mais rapidamente e prever manutenções de forma remota. Porém o projeto deixa claro que tem objetivos no futuro de estender para incluir serviços 5G, Wi-Fi e outras funcionalidades.

2.3.3 Cascavel – CE

O projeto *Meshconnect* desenvolvido pelo Instituto Atlântico, utiliza a tecnologia LoRa em topologia *mesh* para ampliar o acesso à internet em áreas remotas e desconectadas. A solução é baseada na criação de uma rede *mesh* composta por dispositivos de baixo custo que utilizam LoRa e Wi-Fi (802.11) para roteamento de pacotes IP, permitindo comunicação em longas distâncias com baixo consumo energético, alimentados por energia solar (INSTITUTO ATLÂNTICO, 2023).

A escolha da aplicação em topologia *mesh*, tornou possível a infraestrutura de interconexão da rede, já que os dispositivos são responsáveis por colaborar uns com os outros dentro do alcance no meio físico. Assim, os usuários são capazes de acessar a Internet e serviços básicos por meio de pontos de acesso Wi-Fi que garantem a interoperabilidade com os dispositivos dos usuários.

O projeto também avalia a possibilidade de aplicação em cenários de desastres naturais, para situações onde a infraestrutura da rede principal esteja comprometida ou em casos de falhas que gerem um comprometimento na conectividade (INSTITUTO ATLÂNTICO, 2023).

2.3.4 Joinville – SC

Na cidade de Joinville atuou a Smart Green, empresa que já é responsável por atuar também em outras cidades com a implementação dos seus sistemas de telegestão de iluminação pública. Sua solução utiliza tecnologia de comunicação RF *mesh*, realizando gerenciamento através de redes *mesh* permitindo múltiplas rotas e que permitem o processo de contingência em casos de falhas de pontos. Segundo a empresa, financeiramente esse tipo de rede possui vantagem por não gerar custos recorrentes ou infraestruturas adicionais como torres e repetidores (SMART GREEN, 2023).

O sistema de telegestão de Joinville possui como ponto focal os *gateways* para gerenciamento de tráfegos dos dados dos pontos remotos, e em seguida processa e envia para um servidor remoto através de um *Broker MQTT*. A solução também utiliza protocolos de roteamento dinâmico, como BGP, RIP ou OSPF e pode operar com os protocolos ZigBee, Thread, Bluetooth 5.0 LE. Entre as funcionalidades implementadas na telegestão fazem parte o controle de consumo para melhoria da eficiência, através da dimerização (SMART GREEN, 2023).

2.3.5 Guaratuba – PR

Semelhante à Joinville, em Guaratuba também foi implementado pela empresa *SmartGreen* um projeto de uma rede inteligente de multisserviços que realiza a integração e o monitoramento de diversos serviços como: câmeras de segurança pública, câmeras de reconhecimento de placas de veículos, rastreamento de frotas, medidores de energia e água de prédios públicos e a telegestão da iluminação pública. Todos os serviços são interligados por uma rede *mesh* que atende um Centro de Controle de Operações (SMART GREEN, 2023).

3 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão contemplados os conceitos estudados para compreender a estrutura de redes sem fio, bem como a tecnologia escolhida e o estudo dos principais protocolos encontrados, considerados mais adequados para a aplicação deste projeto. Ao final é comentado algumas considerações e as justificativas para a escolha do protocolo.

3.1 IoT

A Internet das Coisas (IoT) é uma rede de objetos físicos – “coisas” – incorporadas com sensores, software e outras tecnologias para conectar e trocar dados com outros dispositivos e sistemas através da Internet (Hussain et al., 2024).

Essa tecnologia proporciona a capacidade de um sistema ser integrado e se comunicar entre si. A IoT ajudará a proporcionar uma operação mais eficiente, econômica e segura criando um sistema conectado entre si através de uma rede global. A Internet das Coisas consiste principalmente em objetos reais - que podem ser medidos, inferidos e compreendidos -, amplamente dispersos, com baixa capacidade de armazenamento e processamento (Arasteh et al. 2016)

De acordo com a Cisco, a previsão é de que 500 bilhões de aparelhos estejam conectados à Internet das Coisas até 2030 (CISCO, 2011). A tecnologia IoT a ser usada nesta presente trabalho é a LoRa, cujo funcionamento se dá pelo uso de sensores os quais são chamados de nós. Esses têm a capacidade de se comunicar através de um protocolo de comunicação.

3.2 ARQUITETURA DE REDE

Para que os dispositivos possam realizar a comunicação entre eles é necessária a utilização de protocolos, que são responsáveis pela codificação e decodificação das mensagens que trafegam na rede. Nas redes de computadores, esses protocolos são implementados em camadas, sendo os dois principais modelos que definem a construção das camadas, o modelo OSI e o modelo TCP/IP. O modelo TCP/IP é definido por uma pilha de protocolos em 5 camadas: camada de aplicação,

camada de transporte, camada de rede, camada de enlace e a camada física, conforme representado na Figura 1.

FIGURA 1 – CAMADAS MODELO TCP/IP

FONTE: As autoras (2024).

3.2.1 Camada de Aplicação

A camada de aplicação é a camada superior da pilha de protocolos e é responsável pela comunicação entre os serviços da rede externa, como a internet, com os aplicativos locais e softwares. As mensagens de dados que são enviadas incluem um cabeçalho, com as informações do pacote de controle, e o *payload* (carga de dados) que inclui o pacote de dados. Um exemplo de protocolo que opera nessa camada é o protocolo HTTP, utilizado na transferência de páginas web.

3.2.2 Camada de Transporte

A camada de transporte é responsável por garantir a efetividade da comunicação, atuando no envio e recepção dos pacotes na rede para as aplicações. A sua base possui dois protocolos principais: TCP e UDP. A principal diferença entre os dois é que o protocolo TCP é orientado à conexão, o que faz com que ele primeiro estabeleça uma conexão e somente após isso, realiza a transmissão de dados. Isso torna muito mais confiável o processo de entrega de pacotes e uma melhor garantia na integridade dos dados transferidos. Já o UDP não é orientado à conexão, sendo

mais simples ideal para aplicações que precisem de velocidade de transferência (aplicações em tempo real), uma vez que não possui garantia nenhuma de entrega ou ordenação dos pacotes.

3.2.3 Camada de Rede

A camada de rede é a camada que abriga os protocolos de roteamento e o principal é o protocolo de internet. O IP é responsável pelo endereçamento dos dispositivos da rede (IPv4 ou IPv6) e garante que os pacotes cheguem corretamente aos seus destinos. É na camada de rede que é também feito o cálculo das melhores rotas de dados entre a origem e o destinatário.

Por meio do gerenciamento de endereços e roteamento de dados, os protocolos de roteamento na camada de rede identificam os dispositivos disponíveis através de endereços únicos, e conseguem determinar o caminho que o pacote irá seguir, avaliando fatores como distâncias e tráfego de rede.

3.2.4 Camada de Enlace

A camada de enlace é responsável pela comunicação direta entre os dispositivos conectados na rede, fazendo com que um pacote seja transferido para o próximo dispositivo sem que haja troca de informações referentes à origem/destino dos dados. Em outras palavras, a camada de enlace organiza e administra o single-hop entre dois *node*s (nós), enviando os pacotes em quadros de maneira sincronizada.

Aqui, para a implementação da rede LoRa *mesh*, pretende-se utilizar os nós da rede para atuar como roteadores interligados entre si, de maneira a aplicar o *multi-hop* dos dados. A vantagem de os dados trafegarem por caminhos alternativos dentro da rede local é justamente para ampliar a escala de alcance da rede, sendo a distância um parâmetro muito valioso para as aplicações IoT.

3.2.5 Camada de Física

A camada física é a última camada da pilha e é nela onde a transmissão bit a bit é realizada. Através da utilização de protocolos como o LoRa, são definidos os

parâmetros fundamentais, como frequência, velocidade de transmissão e modulação, entre outros, para gerar e transformar os sinais em dados. Dessa forma, a camada física constrói a base para a operação das outras camadas.

3.3 TECNOLOGIA LORA

A tecnologia LoRa é uma tecnologia de transmissão de rádio LPWAN de propriedade da Semtech. Em seu nível físico, o LoRa utiliza modulação derivada do *Chirp Spread Spectrum* (CSS), o que lhe confere a capacidade de cobrir longas distâncias com baixa interferência. O CSS opera em faixas de frequência ISM não licenciadas (SEMTECH, 2024). Em 2017, a ANATEL regulamentou a tecnologia LoRa no Brasil através do ato 14448, estabelecendo o plano de frequência para a América Latina baseado no padrão Australiano de 923 MHz (915 MHz a 928MHz) (ANATEL, 2017).

Um pulso chirp, também conhecido como "amostra", é um sinal cuja frequência varia linearmente. Se a frequência aumenta, é chamado de "*up-chirp*"; se diminui, é chamado de "*down-chirp*". Os chirps são ciclicamente deslocados, ou seja, após atingir a frequência mais alta, retornam à frequência mais baixa. A informação transportada por qualquer sinal é determinada pelos saltos de frequência. Cada salto, quando decodificado, representa um símbolo, o qual pode representar um ou mais bits de dados. Um símbolo possui 2^{SF} valores possíveis, onde SF é o fator de espalhamento que representa o número de bits na modulação (Vangelista, 2017).

Na maioria das aplicações que utilizam modulação CSS, o sinal que carrega informação é submetido a correção de erro direta, onde uma sequência de pulsos chirp é modulada antes de ser transmitida. Os “chirps” em uma determinada largura de banda (BW) podem usar diferentes fatores de espalhamento (SF), resultando em diferentes ganhos de processamento ($N = 2^{SF}$) e eficiências espectrais. O período de uma amostra é dado pela seguinte equação:

$$Ts = \frac{2^{SF}}{BW}$$

Em um determinado BW de canal na mesma frequência, cada *chirp* (ou amostra) com o mesmo SF é diferenciado pelo início da sua frequência dessa forma, os dados podem ser codificados combinando o BW do canal selecionado e o fator de

espalhamento, e podem ser modulados na frequência central selecionada (Gkotsiopoulos et al., 2021).

Na tabela 2, são listados os parâmetros que devem ser configurados no firmware a ser programado na placa que emite o sinal LoRa são eles:

TABELA 2 – PARÂMETROS DE CONFIGURAÇÃO LORA

Parâmetro	Descrição
Fator de Espalhamento (SF)	Controla a modulação do sinal, ajustando alcance e taxa de dados. SF5 oferece maior velocidade e menor alcance, enquanto SF12 proporciona maior alcance e menor velocidade.
Largura de banda (BW)	Define a largura de banda e a duração dos *chirps*. Valores comuns são 125 kHz, 250 kHz e 500 kHz. Maior largura de banda aumenta a velocidade, mas reduz a sensibilidade.
Coding Rate (CR)	Baseado no Código de Hamming, adiciona bits de paridade para corrigir erros. Representado como 4/x (x entre 5 e 8), melhora a confiabilidade ao custo de maior overhead.
Potência de Transmissão (*Tx Power*)	Determina a força do sinal transmitido. Potências mais altas ampliam o alcance e melhoram o cálculo da potência entre pacotes enviados e recebidos, mas aumentam o consumo de energia.
Tamanho de prêambulo	Indica o número de símbolos enviados antes do *payload*, permitindo ao receptor detectar e sincronizar o pacote de dados.

FONTE: As autoras (2024).

Na tabela 3 são explicados dois parâmetros que são usados para verificar a qualidade do sinal:

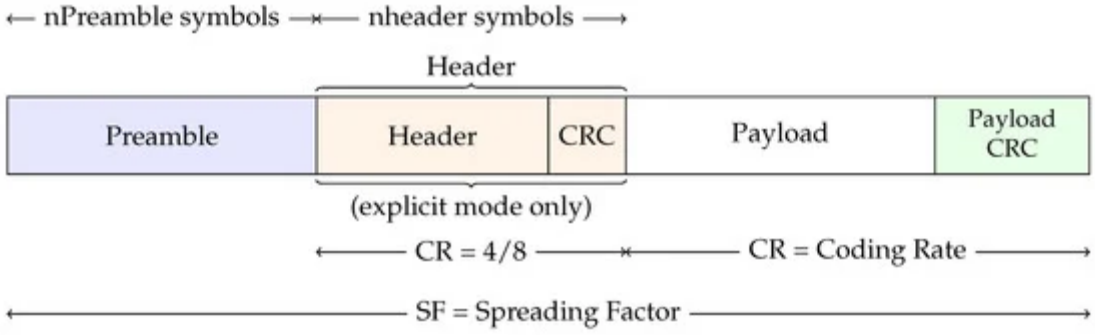
TABELA 3 – PARÂMETROS DE QUALIDADE DE SINAL LORA

Parâmetro	Descrição
RSSI (Received Signal Strength Indicator)	Mede a força do sinal recebido. Valores mais próximos de zero indicam sinais mais fortes.
SNR (Signal-to-Noise Ratio)	Compara o nível do sinal útil ao ruído. Valores altos permitem taxas de transmissão elevadas, enquanto valores baixos aumentam o alcance.

FONTE: As autoras (2024).

A seguir na Figura 2 é apresentada a estrutura de um pacote de dados padrão LoRa, no qual pode-se ver alguns dos parâmetros definidos anteriormente:

FIGURA 2 – ESTRUTURA PADRÃO DE PACOTE LORA



FONTE: Cecílio, Ferreira e Casimiro (2020).

3.3.1 Protocolo LoraWAN

A topologia na qual o LoRa é mais aplicado é a topologia estrela, na qual existe um *gateway* central que irá comunicar com dispositivos finais. Para que os dispositivos possam se comunicar, eles devem necessariamente passar pelo *gateway*. Para a rede LoRa o protocolo LoRaWAN é recomendado.

LoRaWAN é o protocolo de controle de acesso ao meio (MAC) para a camada de enlace de dados e camada de rede em LoRa, apoiado e desenvolvido como padrão pela LoRa Alliance. A comunicação LoRaWAN é bidirecional e oferece vantagens como: baixa taxa de bits e de consumo de energia, porém têm sido apontados algumas limitações em cenários de grande escala (Almuhaya et al, 2023).

O protocolo MAC da LoRaWAN baseado em ALOHA limita sua escalabilidade e representa um obstáculo para a LoRa liderar outras tecnologias LPWAN como principal representante na implantação em larga escala de aplicações IoT. LoRaWAN é ideal para circunstâncias em que as transferências de dados são pouco frequentes (vários pacotes por dia) e o tamanho da carga útil é de cerca de 10 a 50 bytes (Jouhar et al, 2023).

ALOHA em redes de computadores é conhecido como um protocolo de acesso múltiplo usado para transmitir dados por meio de canais de rede compartilhados.

Cada nó e estação transmitem quadros sem detectar se o canal de transmissão está ocupado ou ocioso. Quando o canal está ocioso, os frames são transmitidos com sucesso, e quando dois frames tentam ocupar o canal simultaneamente, ocorre colisão de frames e eles são descartados.

A rede LoRaWAN é composta por dispositivos finais e *gateways*. Os dispositivos finais podem ser definidos em três classes, detalhado na Tabela 4 (Li, Raza e Khan, 2019):

TABELA 4 – COMPARATIVO CLASSE DE DISPOSITIVOS

Classe	Tipo de Dispositivo	Operação	Sincronização	Consumo de Energia	Limitações
Classe A	Sensores	Assíncrona, transmite dados e abre janela de comunicação com o *gateway*. Entra em hibernação após transmissão.	Não há, comunicação ocorre quando há dados para transmitir.	Baixo, entra em hibernação após transmitir.	Comunicação só após transmissão inicial.
Classe B	Atuadores	Sincronizada com beacons enviados pelo *gateway*, abrindo janelas de comunicação em períodos específicos.	Com beacon enviado pelo *gateway*, sincroniza o dispositivo e servidor.	Moderado, devido à sincronização periódica com o beacon.	Depende da periodicidade dos beacons para comunicação.
Classe C	Diversos	Janela de comunicação aberta por longo período, permitindo envio de mensagens a qualquer momento.	Sem sincronização, janelas abertas continuamente, monitoramento em tempo real.	Alto, janela aberta continuamente e estado ativo constante.	Alto consumo de energia e limitação do LoRaWAN para pacotes únicos.

FONTE: As autoras (2024).

Outra especificação da LoRaWAN é o Adaptative Data Range (ADR). Seu objetivo principal é ajustar as variáveis Spreading Factor (SF) e Transmission Power (TP). Se um dispositivo final detecta que não está recebendo informações depois de uma série de transmissões uplink, ele aumenta o valor de seu Transmission Power para o máximo e, em seguida, caso ainda não haja resposta, aumenta seu Spreading Factor até o máximo, da mesma forma (Li, Raza, Khan, 2019).

3.3.2 Protocolo LoraMesher

LoraMesher é uma biblioteca *open source* que propõe um protocolo de roteamento *multi-hop* utilizando a tecnologia Lora (FREITAG et al, 2022). Ao contrário

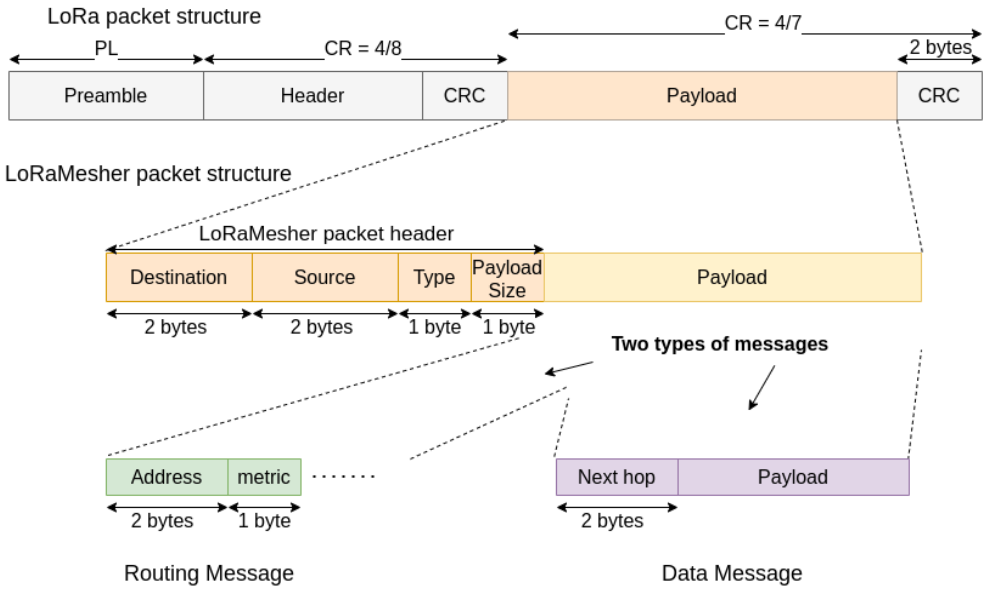
da biblioteca LoraWAN, a proposta da LoraMesher é que a rede possa ter um maior alcance, implementando uma comunicação entre os dispositivos finais. Dessa maneira os *nodes* não precisam ficar a apenas um pulo de distância do *gateway*.

Além da própria biblioteca, também usa a biblioteca RadioLib que é direcionada para aplicações de comunicação sem fio e é compatível com modems da Semtech da linha 127x. Também foi usada a biblioteca FreeRTOS para gerenciamento de tarefas concorrentes, permitindo uma maior eficiência para o microcontrolador.

Essa biblioteca implementa um protocolo de roteamento proativo distance-vector, ou seja, dando preferência pela distância. Esse protocolo irá manter as informações de roteamento atualizadas. Tendo isso em vista, o protocolo manda dois tipos de mensagem: mensagem de roteamento e mensagem de dados. Na primeira o *node* (nó) enviará uma mensagem em *broadcast* para que cada um possa identificá-lo e o adicioná-lo em sua tabela de rotas. Já a segunda envia apenas para destino final e se este não estiver em sua tabela de rotas, colocará como *next hop* o *node* que tem acesso ao mesmo.

Na Figura 3 é possível ver a estrutura de pacotes padrão LoRa e dentro do *payload* será incluída a estrutura definida no protocolo de roteamento LoraMesher, que é separado em *header* (cabeçalho) e *payload*. Para ambos os tipos de mensagem, o cabeçalho será o mesmo, mas o *payload* terá informações diferentes.

FIGURA 3 – ESTRUTURA DE PACOTE LORAMESHER



FONTE: Freitag et al. (2022).

A mensagem de rotas será incluída na tabela de rotas de cada dispositivo, em forma de lista. Cada rota contém o endereço do nó e número de saltos de distância até ele. Como a tabela de rotas é atualizada dinamicamente, os *nodes* compartilham entre si sua tabela de rotas.

Nas figuras 4 e 5 estão detalhadas as estruturas de um pacote de roteamento e de um pacote de dados. O cabeçalho de ambas tem a mesma estrutura e o *payload* as diferencia. No *payload* da mensagem de rotas consta os endereços contidos em sua tabela de rotas e suas devidas métricas, que significa o número de saltos até o npo que recebeu essa tabela de rotas. E na mensagem de dados o *payload* contém o próximo salto (se ele existir), o qual vai conter o endereço do nó intermediário que deve encaminhar este pacote para o endereço de destino, e o *payload* que deve conter os dados e ter até 255 bytes.

FIGURA 4 – ESTRUTURA DE PACOTE DE ROTEAMENTO

Pacote de Roteamento					
Cabeçalho				Mensagem de Rotas	
Destino	Origem	Tipo do pacote	Tamanho do *payload*	Endereço do nó	Métrica
8430	06D4	Roteamento	32	8314, 8A34, 830C	2, 3, 3
2 bytes	2 bytes	1 byte	1 byte	2 bytes	1 byte

FONTE: As autoras (2024)

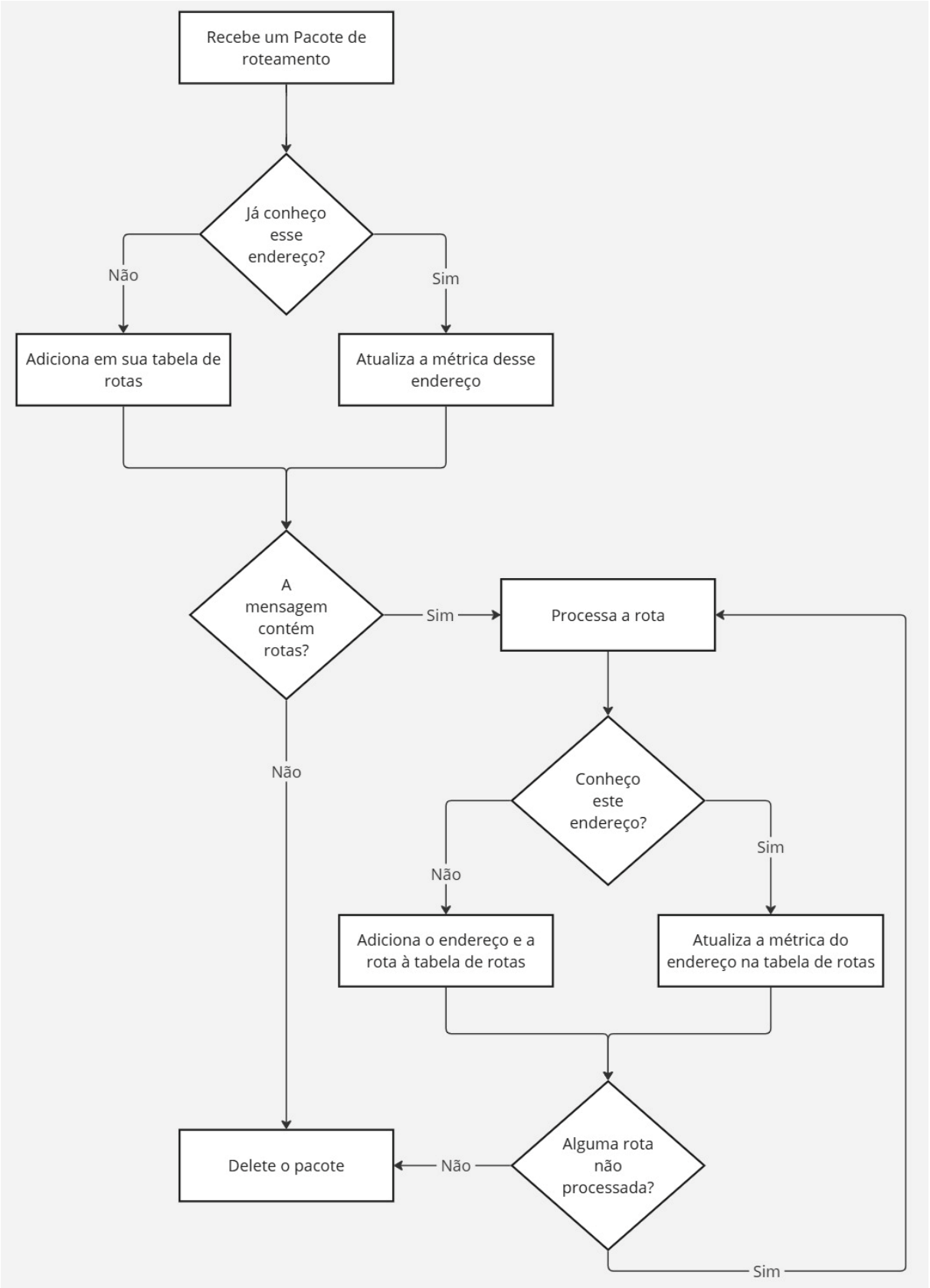
FIGURA 5 – ESTRUTURA DE PACOTE DE DADOS

Pacote de Dados					
Cabeçalho				Mensagem de Dados	
Destino	Origem	Tipo do pacote	Tamanho do *payload*	Próximo salto	*Payload*
8430	8E14	Dados	49	06D4	
2 bytes	2 bytes	1 byte	1 byte	2 bytes	Até 255

FONTE: As autoras (2024)

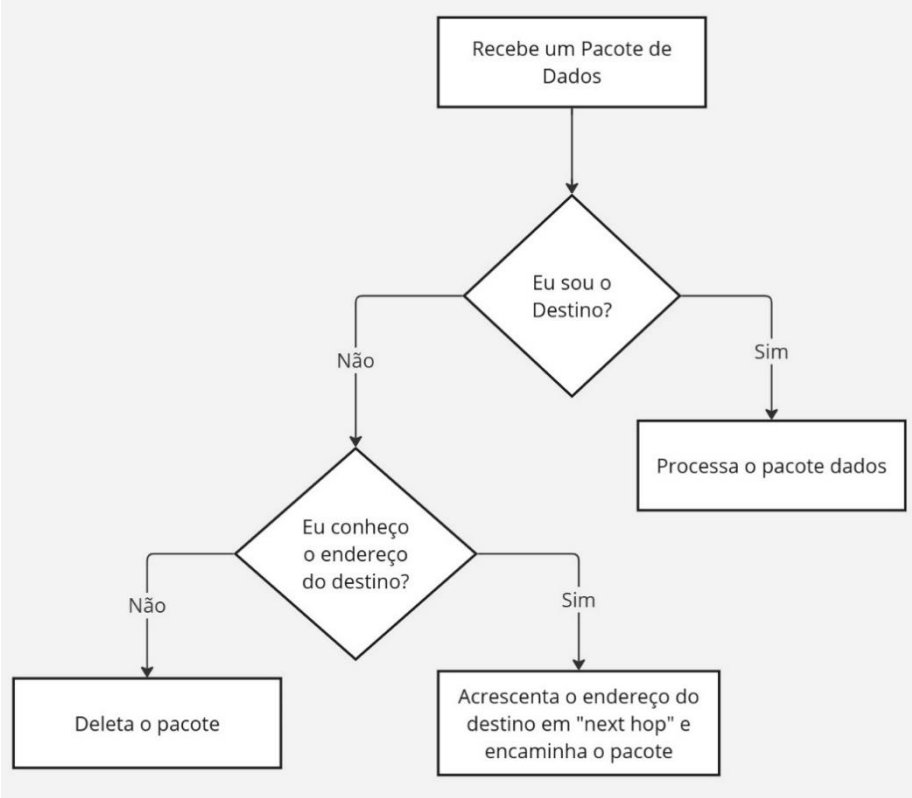
Nas Figuras 6 e 7 estão explicados em diagrama de blocos da rotina de processamento de um pacote de roteamento e de um pacote de dados. Uma informação importante é que quando um nó atualiza a métrica dos nós que constam na tabela de rotas, ele grava o SNR (relação sinal-ruído) da mensagem enviada por tal nó. Esse valor será usado para ajustar a preferência na tabela rotas, ou seja, quanto menor a relação sinal-ruído, maior a preferência.

FIGURA 6 – ROTINA DE PROCESSAMENTO DE PACOTE DE ROTEAMENTO



FONTE: As autoras (2024)

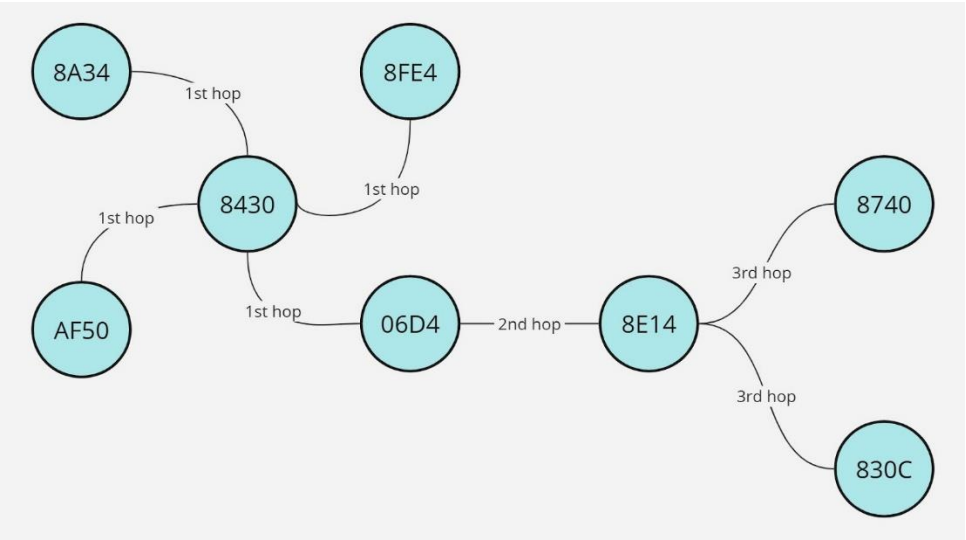
FIGURA 7 – ROTINA DE PROCESSAMENTO DE PACOTE DE DADOS



FONTE: As autoras (2024)

Na figura 8 foi feito um diagrama que mostra a tabela de rotas de um nó. Neste exemplo o endereço 8430 é o *gateway*.

FIGURA 8 – DIAGRAMA DE ROTAS COM NÚMERO DE SALTOS



FONTE: As autoras (2024).

O protocolo usa um esquema de filas para administras as tarefas a serem processadas são elas:

- Receive packets (Q_RP): armazena pacotes recebidos até que possam ser processados.
- Send packets (Q_SP): armazena pacotes até serem enviados.
- User Received Packets (Q_URP): armazena os pacotes de dados a serem enviados na camada de aplicação.

Como o código usa FreeRTOS, que faz o gerenciamento e priorização de tarefas, foram implementadas seis tarefas responsáveis por implementar cada rotina. Nas tabelas 5 e 6 estão explicadas cada uma delas.

TABELA 5 – TAREFAS DE ROTEAMENTO LORAMESHER

Tarefa	**Função Principal**	**Ações Realizadas**	**Prioridade**	**Ativação**
Receive Task	Receber e preparar pacotes LoRa para processamento.	Recebe pacote, processa *payload*, transforma em pacote LoRaMesher, adiciona à fila Q_RP e notifica a *Process Task*.	Mais alta	ISR (acionada ao receber um pacote LoRa).
Send Task	Enviar pacotes da fila Q_SP com controle de colisão (CSMA/CA).	Obtém pacote da fila Q_SP, verifica tabela de roteamento, realiza atraso *duty cycle* (ciclo de trabalho) e envia pacote.	Segunda mais alta	Notificada quando um pacote é adicionado à Q_SP.
Routing Protocol Task	Construir e compartilhar tabela de roteamento entre nós vizinhos.	Cria mensagens de roteamento, atualiza tabela de roteamento com base no número de saltos e periodicidade configurada.	Média (abaixo da *Send Task*)	Executada periodicamente.

Process Task	Processar pacotes recebidos (dados ou roteamento).	Verifica o tipo de mensagem, atualiza rotas, entrega mensagens locais ou encaminha/descarta pacotes conforme a tabela de roteamento.	Mais baixa	Notificada pela *Receive Task* ao receber um pacote.

FONTE: As autoras (2024).

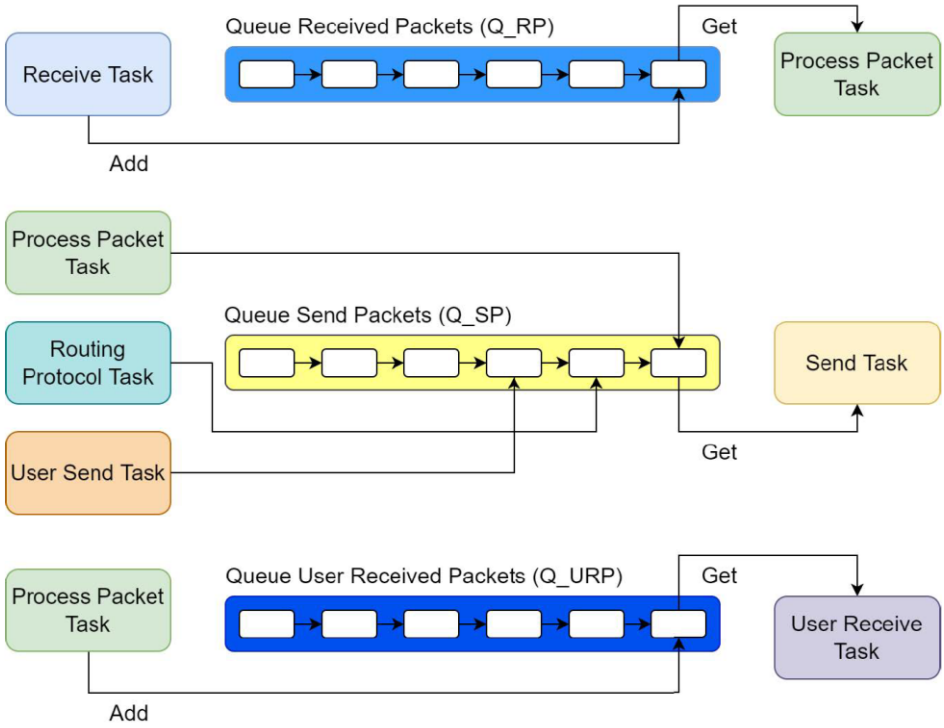
TABELA 6 – TAREFAS DE APLICAÇÃO LORAMESHER

Tarefa	**Função Principal**	**Ações Realizadas**	**Ativação**
User Send Task	Enviar mensagens de dados pela biblioteca, implementadas no nível da aplicação.	Chama *createPacketAndSend*, específico destino e *payload*, e explora tabela de roteamento para selecionar nós disponíveis.	Executada sob demanda pelo código da aplicação.
User Receive Task	Processar mensagens de dados destinadas ao nó e gerenciar a fila Q_URP.	Notificada pela *Process Task*, processa pacotes destinados ao nó, e exclui mensagens da fila após o processamento.	Notificada pela *Process Task* ao receber uma mensagem de dados.

FONTE: As autoras (2024).

A figura 9 está mostrando o funcionamento das tarefas em conjunto com o sistema de filas do protocolo.

FIGURA 9 – INTEGRAÇÃO ENTRE AS FILAS E AS TAREFAS



FONTE: Freitag et al (2022)

Para validar a biblioteca, Freitag et al. (2022) propõe uma validação pelas seguintes métricas:

1. PDR [%]: relação entre o número total de pacotes recebidos e enviados.

$$PDR = \frac{n\ Pacotes\ Recebidos}{n\ Pacotes\ enviados}(100)[\%]$$

2. *Control Overhead* (Controle de Sobrecarga) [%]: relação entre a quantidade de pacotes de controle enviados e a quantidade de pacote de dados recebidos.

$$Control\ Overhead = \frac{n\ Pacotes\ de\ Controle}{n\ Pacotes\ de\ Dados}$$

3.3.3 Protocolo LoraNet

O LoRaNet é um projeto de uma biblioteca protótipo, onde foi proposta uma pilha de protocolos para implementação de rede *mesh* baseada em LoRa, utilizando o protocolo AODV (*On-Demand Distance Vector Routing*) para permitir comunicação ponto a ponto entre dispositivos IoT, sem a necessidade de *gateways* como no LoRaWAN (CUNHA et al, 2022).

A biblioteca foi desenvolvida em C/C++ para a placa ESP32 utilizando a IDE Arduino, e é *open source* para fins de pesquisa.

A fim de evitar a dependência de *gateways*, como é o caso do LoRaWAN, e já que a tecnologia LoRa define apenas a camada física da comunicação, o LoRaNet implementa a comunicação *multi-hop* através dos dispositivos finais LoRa, possibilitando a retransmissão de dados e com isso demonstrou a viabilidade do uso da tecnologia para expansões cobertura de rede LPWAN.

O protocolo AODV é um protocolo de roteamento reativo (apenas cria as rotas sob demanda) para redes ad hoc sem fio, com foco em redes móveis. Para a implementação da rede *mesh* através do protocolo AODV, foi preciso utilizar outro modelo de operação do 802.11, para o modo ad hoc. Nesse modelo de operação, os dispositivos se comunicam entre si, sem utilizar como base a infraestrutura da rede existente, tornando a rede descentralizada e flexível e passível de implementar *multi-hops*. Assim é descrito como o MANET - rede ad hoc móvel.

O protocolo LoRaNet implementa três tarefas principais responsáveis por cada rotina: Solicitação de rota (RREQ), Resposta de Rota (RREP) e Erro de Rota (RERR), que estão explicadas mais detalhadamente na Tabela 7 abaixo.

TABELA 7 – TAREFAS DE ROTEAMENTO LORANET

Tarefa	**Função Principal**	**Ações Realizadas**	**Prioridade**	**Ativação**
Solicitação de Rota (RREQ)	Descobrir rotas para destinos desconhecidos	Transmissão de pacotes RREQ em broadcast; configuração de rotas reversas nos nós intermediários; uso de TTL expansivo	Alta para descoberta inicial	Ativada sob demanda pelo nó originador
Resposta de Rota (RREP)	Confirmar e estabelecer rotas válidas para a comunicação.	Geração de RREP pelo destino ou nó com rota ativa; envio unicast de RREP pela rota reversa; atualização de tabelas de roteamento	Alta para rotas recém-descobertas	Ativada ao receber um RREQ válido

Erro de Rota (RERR)	Notificar falhas de rota e iniciar reparos.	Identificação de falhas; envio de RERR (unicast ou broadcast); invalidação de rotas inválidas; reparo via retransmissão de RREQ	Alta para rotas críticas	Ativada ao detectar falha de rota.

FONTE: As autoras (2024).

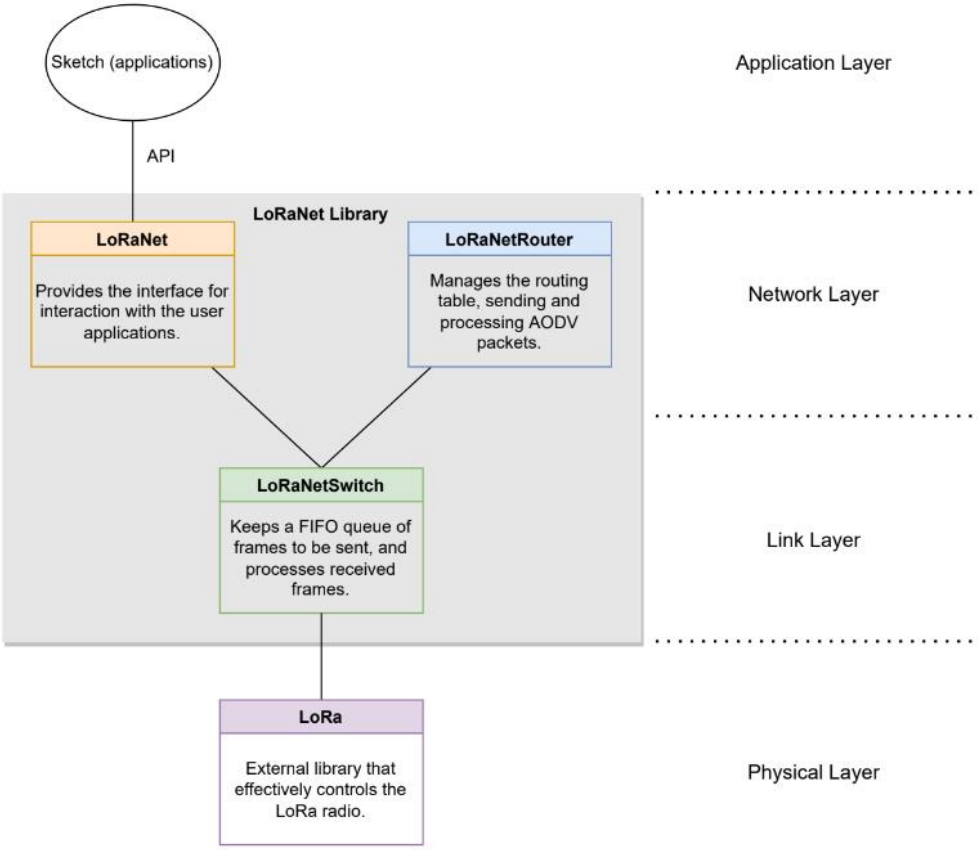
Como já comentado anteriormente, o LoRaNet é baseado em rede com topologia *mesh*, e não existe hierarquias entres os *nodes*. Os dispositivos atuam tanto como receptores quanto fonte de mensagens, ou ainda como nós intermediários, podendo ser utilizados para o envio de pacotes quando não há alcance da fonte para o destino.

Os dispositivos LoRaNet operam de forma a verificar se já houve uma configuração inicial ao serem ligados, lendo a memória EEPROM para validar essa verificação. Caso contrário, geram novas credenciais e um endereço IP aleatório, que são armazenados para reinicializações futuras. Essa abordagem garante um identificador único para cada dispositivo na rede. Os dados a serem transmitidos são organizados em uma fila FIFO (First In First Out) e enviados apenas quando uma rota ativa está disponível, otimizando o uso da rede.

O endereçamento ocorre quando cada dispositivo na rede seleciona um endereço IPv4 aleatório para se identificar, utilizando a faixa de endereços 10.0.0.0, reservada para redes privadas pela Internet Assigned Numbers Authority (IANA).

Abaixo está ilustrada na Figura 10 a arquitetura proposta para o protocolo LoRaNet, e na sequência será comentado sobre os módulos implementados dentro de sua biblioteca, que definem a rotina de tarefas na camada de rede e na camada de enlace.

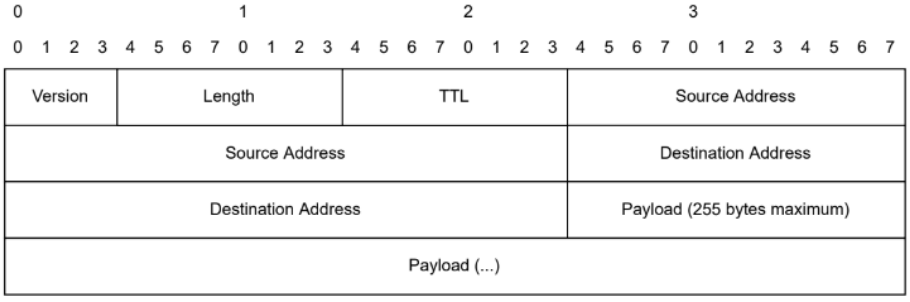
FIGURA 10 – ARQUITETURA LORANET



FONTE: Bruno Cunha (2022).

O módulo LoRaNet, está presente na camada de rede e é responsável pelo gerenciamento da comunicação entre as aplicações e o usuário, além de iniciar o campo do cabeçalho e do *payload*. Ele realiza o processamento de todos os pacotes transmitidos e exige que seja registrada na função de *callback* (retorno) na aplicação final do usuário como garantia de recebimento das mensagens. Na figura 11 mostra a estrutura do pacote de aplicação:

FIGURA 11 – ESTRUTURA DE PACOTE DE APLICAÇÃO LORANET

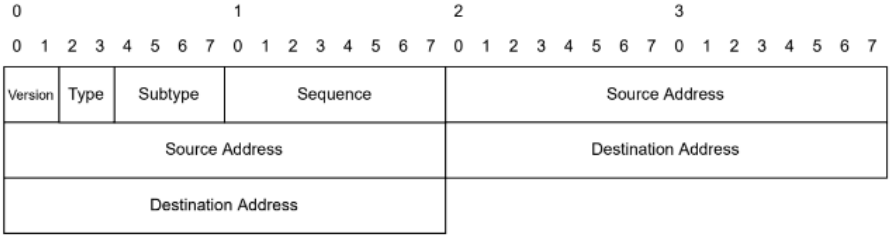


FONTE: Bruno Cunha (2022).

O módulo LoRaNetRouter é responsável pela tabela de roteamento e realiza o gerenciamento e processamento das mensagens do protocolo AODV. Os cabeçalhos aqui implementados seguem o padrão do protocolo AODV de roteamento.

Por último, o módulo LoRaNetSwitch é responsável pelo gerenciamento da camada de enlace, fazendo o controle da fila de quadros, mensagens de confirmação de pacotes e replicar quadros. O cabeçalho implementado aqui foi baseado no padrão IEEE 802.11 e modificado para fins de redução de sobrecarga. Na figura 12 está representada a estrutura do pacote de roteamento.

FIGURA 12 – ESTRUTURA DE PACOTE DE ROTEAMENTO LORANET



FONTE: Bruno Cunha (2022).

3.3.4 Escolha do protocolo

A partir dos estudos realizados nos protocolos LoRaMesher e LoRaNet apresentados nos tópicos anteriores, decidiu-se por implementar o protocolo LoRaMesher para a aplicação deste trabalho. O motivo para essa escolha é pela estrutura do projeto. Apesar da biblioteca LoRaNet ser mais simples na compressão do protocolo, este possui menos funcionalidades. O protocolo LoraMesher possui uma documentação mais completa, o que contribuiu no desenvolvimento e teste da aplicação.

4 MATERIAIS E MÉTODOS

4.1 MATERIAIS

Após pesquisas e comparação das especificações de diferentes microcontroladores e modem compatíveis com a tecnologia LoRa, foi escolhida a placa da Heltec que conta com um microcontrolador ESP-32 acoplado com modem Sx1276 compatível com a frequência de 915 MHz, que é a frequência reservada para esse tipo de tecnologia no Brasil.

O motivo para a escolha desse modelo, foi a alta compatibilidade com diferentes protocolos disponíveis em redes de código aberto, além de ser a opção de placa ESP-32 + LoRa de menor custo entre as opções pesquisadas. Na tabela 8 é possível comparar o preço de diferentes dispositivos embarcados com a mesma proposta.

Foram feitas cases por impressão 3D no formato da placa. O motivo do uso da case é pela proteção do módulo e estabilização da antena para melhor qualidade de sinal durante os testes. Segue imagem do módulo já na case na Figura 13:

FIGURA 13 – CASE EM IMPRESSÃO 3D

FONTE: As autoras (2024).

A placa foi alimentada com uma bateria íon-lítio modelo 18650 Samsung. O motivo da escolha desta bateria foi por uma das colegas já ter comprado para uso em projetos anteriores. Foi analisada outra opção de bateria Li-Po que poderia ser

encaixada dentro da case, porém o custo era mais elevado. Segue tabela 8 com as comparações das baterias pesquisadas:

TABELA 8 – COMPARATIVO DE BATERIAS

Modelo	Química	Dimensão [mm]	Tensão nominal [V]	Capacidade [mAh]	Preço [R$]
18650 Samsung	Íon-Lítio	18x65	3,7	2200	23
602035 Rontek	Li-Po	35x20x5	3,7	400	45

FONTE: As autoras (2024).

A seguir será comentado com mais detalhes o hardware escolhido e as opções avaliadas. Foram comparados módulos que eram usados em projetos *mesh*/ *multi-hop* encontrados no github, inclusive o protocolo LoraMesher, que foi o protocolo escolhido pela equipe, utiliza a placa TTGO T-Beam. Porém essa não foi a placa escolhida por conta do preço. O ponto positivo é que essa também usa como microcontrolador ESP-32, portanto foram necessárias algumas mudanças no código, mas a placa da Heltec é compatível com o protocolo

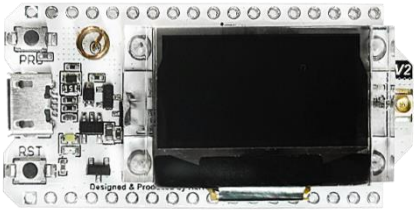
Por fim, foram descartadas as escolhas de hardwares que tivessem que adquirir o módulo LoRa separadamente da placa de desenvolvimento pela preferência na praticidade que as opções integradas oferecem, além das vantagens de custo menor e maior facilidade na implementação prática.

4.1.1 Heltec WiFi LoRa 32 (V2)

O módulo WiFi LoRa 32 foi desenvolvido pela Heltec Automation, e é um hardware que integra o ESP32 como chip master e com o chip LoRa SX127x como chip *node*, também oferecendo funções de Wi-Fi e BLE. Das demais especificações, trabalha nas frequências de 470~510MHz e 863~923MHz (depende da região da aplicação), taxa máxima de potência de transmissão Tx de 19±1dBm, e sensibilidade de -135dBm de sinal de recebimento.

É uma placa construída para desenvolvedores e devido as suas especificações, é uma ótima escolha para trabalhar em aplicações IoT e *smart cities*. Segue imagem da placa utilizada na Figura 14.

FIGURA 14 – PLACA HELTEC WIFI LORA v2



FONTE: Heltec (2024)

Um dos principais motivos que influenciaram na escolha do hardware foi a integração do chip SX1276 na placa. Os transceptores SX1276/77/78/79 da Semtech possuem o modem LoRa embutido, com RSSI limitado a -127 dBm. Por possuir o chip LoRa integrado ao hardware do ESP32, isso tornou a solução mais adequada para a aplicação do projeto. O modelo SX1276 foi o escolhido por ser compatível com a banda de frequência de 915MHz, que é frequência usada para LoRa no Brasil.

Além disso, foi considerada uma opção versátil, levando em conta também a compatibilidade do módulo com a LoRaWAN que permitiria utilizar os módulos para futuros trabalhos e a maior abrangência de aplicações. A seguir na Tabela 9, é feito um comparativo com as características que influenciaram na escolha do hardware.

TABELA 9 – COMPARATIVO DE PLACAS

Equipamento	**Preço Médio (R$)**	**Vantagens**	**Aplicações**	**Restrições**
Heltec WiFi LoRa 32 (V2)	189,9	Integra Wi-Fi, Bluetooth e LoRa Display OLED de 0,96" Gerenciamento de bateria Li-Po integrado	Projetos IoT, cidades inteligentes, automação residencial	- Limite no range do RSSI de -127 dBm.
TTGO T-Beam	394,0	Inclui GPS integrado Suporte para bateria 18650 Conectividade Wi-Fi, Bluetooth e LoRa	Rastreamento de ativos, projetos de geolocalização	- Maior consumo de energia devido ao GPS - Tamanho físico maior, podendo ser uma limitação em projetos compactos

FONTE: As autoras (2024).

4.1.2 Custos

Na Tabela 10 foram declarados os custos dos materiais para cada *node*. São apresentados os valores por unidade e os valores totais, considerando que para o presente trabalho foram utilizados 8 *nodes.*

TABELA 10 – CUSTO POR DISPOSITIVO

Equipamento	Custo Unitário [R$]	Custo Total [R$]
Heltec WiFi LoRa 32 (V2)	189,90	1.519,20
Bateria 18650 Samsung 2200 Mah	23,00	184,00
Case	8,71	69,70
Total	**224,90**	**1.772,9**

FONTE: As autoras (2024).

4.2 METODOLOGIA APLICADA

Inicialmente foram feitas pesquisas acerca da compatibilidade da tecnologia LoRa com protocolo de comunicação *multi-hop*, visto que o protocolo de comunicação mais difundido é o LoRaWAN, que é single-hop.

Para isso foi usada a plataforma Github, que é uma rede aberta colaborativa na qual foi possível encontrar diversos repositórios com protocolos de comunicação LoRa *multi-hop*. Nessa rede foi possível encontrar tanto códigos de firmware para ESP-32 com o modem Sx 1276 quanto protocolos de comunicação *mesh* que serão responsáveis pela camada de roteamento LoRa entre os *nodes.*

Foram estudadas algumas opções de protocolos de roteamento para que a equipe pudesse analisar e escolher a mais compatível com nosso objetivo. Sendo assim, a equipe chegou a duas opções finais, entre o protocolo LoRaNet e o protocolo LoRaMesher, e optou-se por escolher a segunda opção - que inclusive dispõe de um artigo do qual foram coletadas informações valiosas para melhor entendimento acerca do seu funcionamento.

Após a escolha do protocolo de roteamento, a equipe realizou a aquisição dos equipamentos necessários para a implementação da rede. Com isso, inicia-se a etapa de implementação do firmware base, a adaptação e modificações do código para o

atendimento aos objetivos do trabalho, a execução do protocolo e os testes para validação do funcionamento.

Como o protocolo LoraMesher utilizou uma placa diferente da escolhida nesse trabalho, foi preciso fazer algumas alterações no código para que fosse compatível com a placa Heltec Wifi Lora v2.

A respeito da parametrização utilizada na aplicação LoRa, foram usados os mesmos parâmetros já definidos na biblioteca, exceto a frequência que foi ajustada para o padrão utilizado no Brasil. Segue a Tabela 11 com os valores de parâmetros:

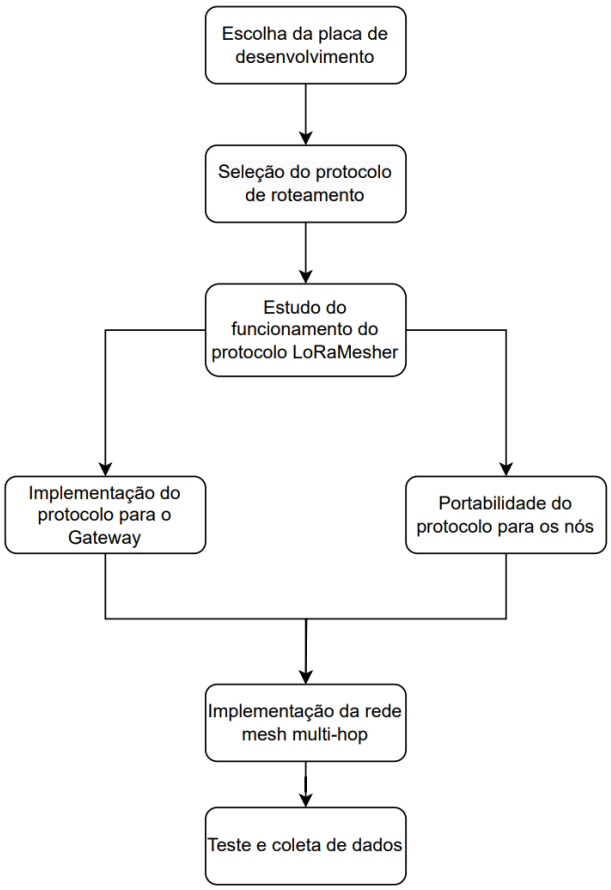
TABELA 11 – PARAMETROS LORA

Frequency	915 MHz
Bandwidth	125 kHz
Spreading Factor	7
Preamble length	8
Transmission Power	14
Coding rate	4/7

FONTE: As autoras (2024).

Segue ilustrado na Figura 15 o diagrama de blocos representando os processos aplicados na metodologia:

FIGURA 15 – FLUXOGRAMA APLICADO NA METODOLOGIA



FONTE: As autoras (2024).

4.3 DESENVOLVIMENTO

Primeiramente foi adaptado um exemplo de uso disponibilizado na biblioteca para esta aplicação. Apesar do código ter uma lógica para diferenciar *nodes* como g*ateway* e sensor, foi feita uma proposta diferente da usada na biblioteca.

Para o módulo escolhido para o papel de *gateway*, foi definido o seu endereço MAC em um firmware diferente dos demais nós da rede. Nele, foi implementado um parser TCP para enviar o *payload* dos pacotes de dados do *gateway* para um servidor IP. O parser TCP é uma ferramenta que filtra e rotula pacotes TCP.

Além disso, foi definido que o destino de todos os pacotes de dados seria o endereço do g*ateway*, dessa forma todos os *nodes* irão direcionar seus pacotes de dados para o mesmo destino. Caso o *node* não tenha acesso ao *gateway*, ele irá encaminhar o pacote para um *node* que o tenha em sua tabela de rotas.

Além disso, foram usados os dados coletados no monitor serial para realizar uma análise do PDR e *Control Overhead*.

Antes de começarem os testes, foram definidos os parâmetros LoRa com valores compatíveis com essa aplicação e definido que os pacotes de rotas serão enviados a cada 30 segundos e os pacotes de dados a cada 5s.

Após o desenvolvimento do código foram feitos testes em bancada para verificar o comportamento para em seguida prosseguir com os testes em campo. Foram programados os firmwares em todos os nós e no *gateway* para validar a implementação e conforme necessário, realizar os ajustes na lógica do protocolo. Esses, foram realizados em ambiente indoor.

4.4 FERRAMENTAS DE DESENVOLVIMENTO DE SOFTWARE

4.4.1 Git

Git é um sistema de controle de versão e gerenciamento de projetos, amplamente utilizado para o desenvolvimento de software (SHACON, S e STRAUB, B, 2014) Esta ferramenta permite que desenvolvedores colaborem em um mesmo projeto simultaneamente, além de manter as versões armazenadas de forma segura e documentada. Esses foram os principais fatores que incentivaram o uso do git para controle das versões de software. A versão de git usada no projeto foi a 2.47.0 lançada no dia 7 de outubro de 2024.

4.4.2 PlatformIO

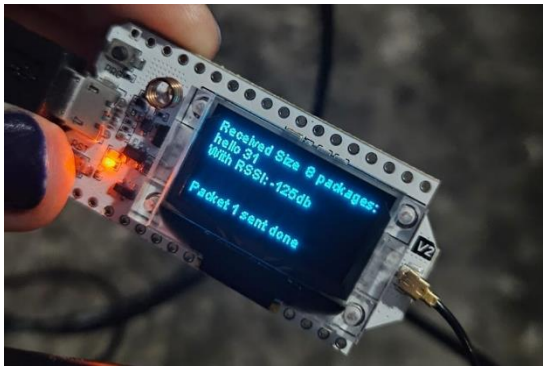
PlatformIO é um ambiente de desenvolvimento integrado (IDE) e um sistema de construção de código aberto, projetado especialmente para o desenvolvimento de sistemas embarcados. Esta IDE é compatível com os *frameworks* Arduino e ESP-IDF e é usada como uma extensão dentro do editor de código VS Code. A facilidade de usar o platformio é em poder migrar projetos entre placas e frameworks sem a necessidade de instalar as IDEs tradicionais, economizando também a memória do seu sistema. Além disso, ao criar um projeto no platformio, as bibliotecas são gerenciadas automaticamente e também é possível acompanhar logs pelo monitor serial como o clássico arduino. A versão usada foi a 6.1.16 lançada no dia 26 de setembro de 2024.

4.5 VALIDAÇÃO

4.5.1 Teste de Alcance

Após a aquisição dos módulos, foram realizados testes em campo para verificar o alcance real da comunicação, bem como perceber suas limitações. Para isso, foi utilizada a própria biblioteca da Heltec que possui exemplos para implementar uma comunicação básica de *Send* (aquele que envia) e *Receive* (aquele que recebe), com uma contagem de mensagens enviadas e a potência do sinal. A Figura 16 mostra esse em andamento.

FIGURA 16 – PLACA HELTEC MOSTRANDO RSSI



FONTE: As autoras.

Para a realização dos testes experimentais de alcance foi selecionado um local aberto, com o objetivo visualizar e obter os dados sem obstáculos como estruturas ou afins que pudessem atenuar o sinal.

Para esse teste foi utilizado apenas dois dispositivos LoRa configurados um como *sender* e outro como *receiver*. Com conexão direta, foi possível concluir as medidas descritas na Tabela 12.

TABELA 12 – RELAÇÃO ENTRE DISTÂNCIA E RSSI

Medição	Distância (m)	RSSI (dBm)
1	60	-110
2	100	-120
3	170	-129
4	450	-129

FONTE: As autoras.

Considerando que o SX1276 foi projeto para um limite de range do RSSI de -127 dBm, foi possível notar que após passar de 100m o sinal já chega praticamente no seu limite, porém vale ressaltar mesmo após o sinal indicar tais valores, dependendo da distância (como é o caso da posição 4) a comunicação ainda era efetiva e a contagem de mensagens entregues continuou operando.

Uma ressalva para o caso 4, que a comunicação manteve-se operando com uma distância máxima testada pela equipe de 450 metros e o RSSI no limite de -120 dBm, a equipe poderia ter ido mais longe, porém por falta de espaço aberto sem que houvessem infraestruturas no caminho não foi possível. Nesse caso, o teste foi realizado em uma rua reta.

Com isso, a equipe concluiu algumas análises iniciais como a limitação da potência do sinal RSSI com a distância, porém por outro lado pôde-se perceber que a comunicação não se restringia aos mesmos limites, e que em espaços abertos como campos e aplicações agrícolas, o módulo LoRa utilizado tem grande potencial de alcance.

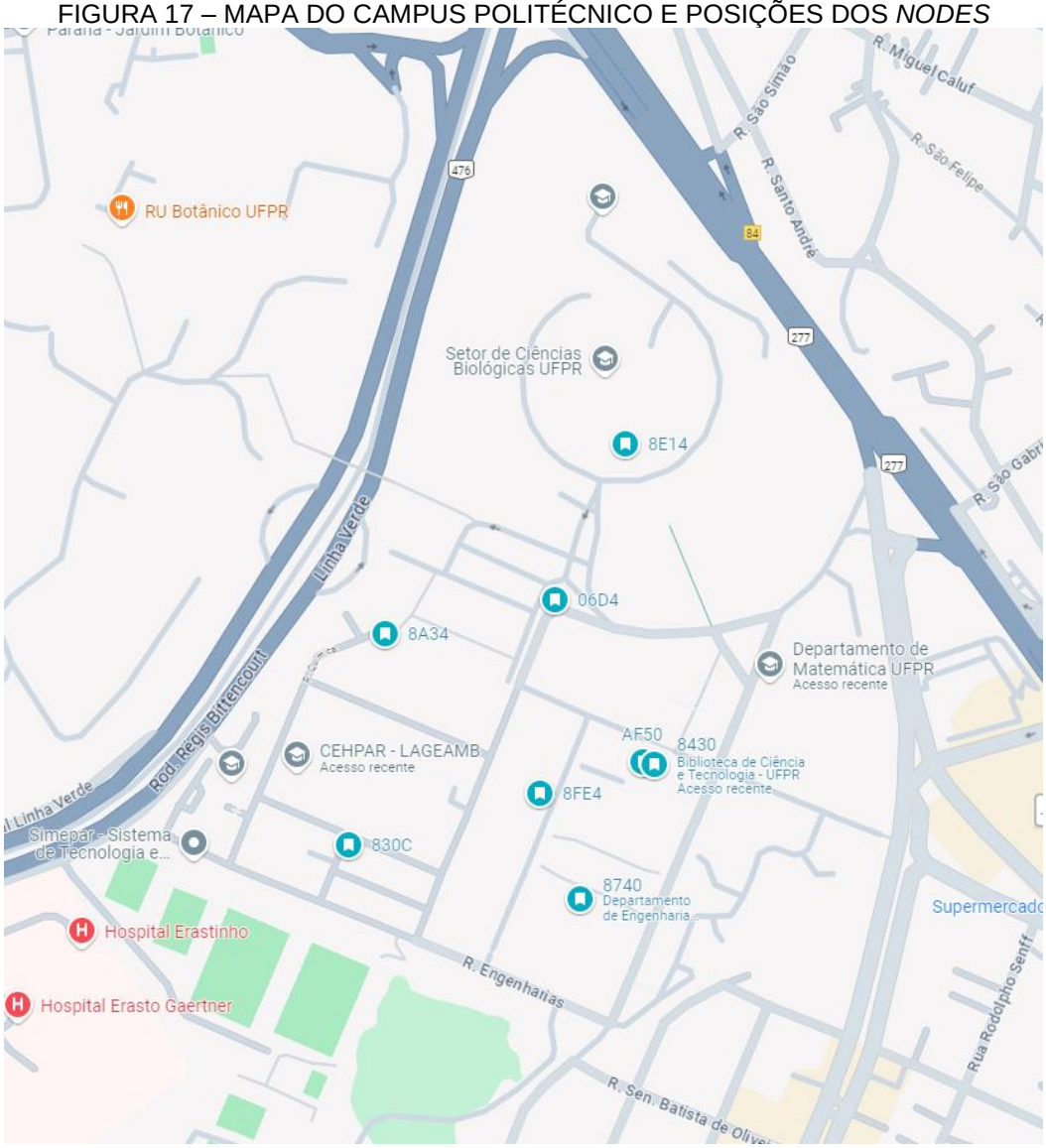
4.5.2 Teste com Protocolo LoraMesher

Foi definido que o teste em campo do protocolo seria feito no campus Centro Politécnico da Universidade Federal do Paraná. Desse modo, com base nos resultados obtidos no teste inicial, foram definidas as posições que os *nodes* seriam posicionados dentro do campus. Os *nodes* foram posicionados entre 130m a 320m do g*ateway*, o qual foi definido como a placa de endereço 8430. A distância é um fator importante para garantir que os nós fizessem saltos entre si até que as informações chegassem no *gateway*.

A equipe então avaliou quais pontos do mapa do campus seriam estratégicos para posicionar os *nodes* de forma que a rede realizasse a melhor cobertura possível dos blocos, e que os nós mais distantes tivessem pelo menos um outro nó ao alcance da rede.

Segue abaixo na Figura 17 que mostra o mapa do centro politécnico com a marcação da posição dos nós e em seguida na Figura 19 um diagrama mostrando a distância entre cada *node* e o *gateway*.

FIGURA 17 – MAPA DO CAMPUS POLITÉCNICO E POSIÇÕES DOS *NODES*



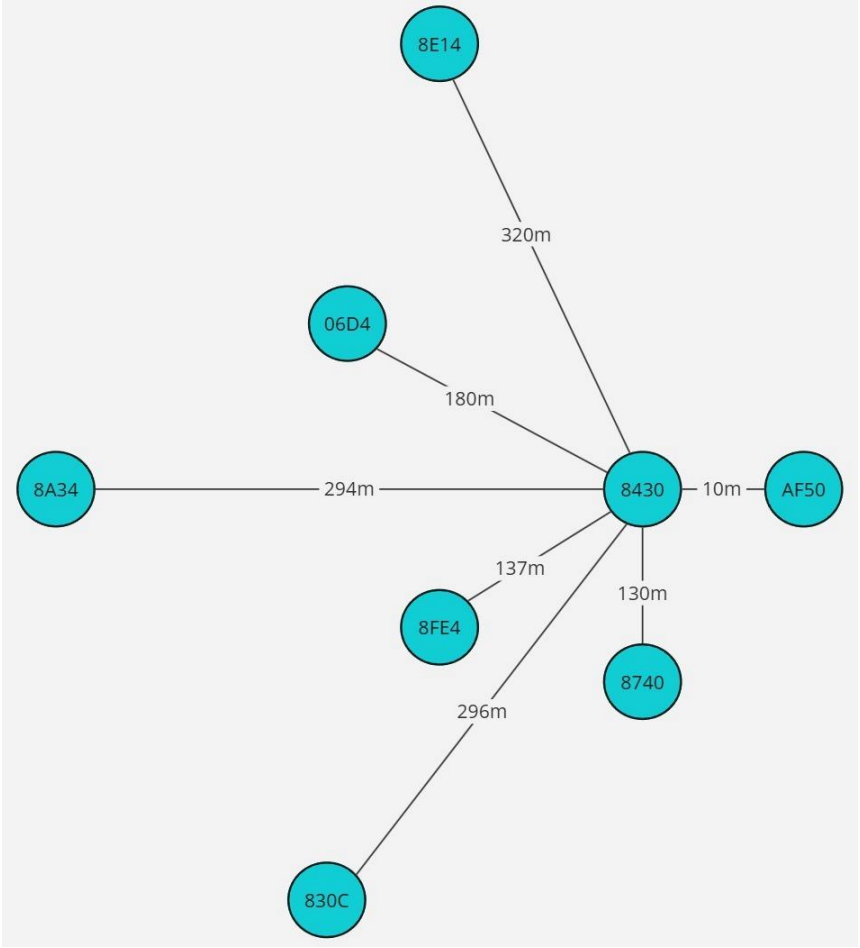
FONTE: As autoras.

A imagem abaixo (Figura 18) foram fotos tiradas no dia dos testes, e são dois dispositivos que foram posicionados no campus, indicados no mapa como nó 8A34 e nó 8740.

FIGURA 18 –POSICIONAMENTO DOS DISPOSITIVOS NO CAMPUS

FONTE: As autoras.

FIGURA 19 – DIAGRAMA DE DISTÂNCIA ENTRE *NODES* E *GATEWAY* (8430)



FONTE: As autoras.

Após a escolha do posicionamento, foram programados todos os dispositivos com o mesmo protocolo, exceto o *gateway* que possui uma configuração alternativa que implementa o WI-FI para a comunicação. Para isso, foi implementada na lógica do protocolo uma condição que, caso definido o uso do Wi-Fi pelo programador, será implementado como *gateway* o dispositivo que está sendo gravado o protocolo.

Na sequência, a equipe se dividiu para distribuir os *nodes*, que foram posicionados a céu aberto, nos locais definidos anteriormente, apenas durante a realização dos testes, após isso foram recolhidos.

Ao inicializar o teste, os logs transmitidos no monitor serial foram gravados usando a API do PlatformIO, na qual é possível gravar as placas e salvar o texto do monitor serial em um arquivo .txt, para serem usados na análise posteriormente.

Três dos 8 *nodes* foram lidos na serial: o *gateway* 8430, o AF50 que estava bem próximo do *gateway* como forma de controle e o 8FE4 que estava posicionado no departamento de Engenharia Elétrica.

Além dos logs do monitor serial, também foram salvos os pacotes de dados enviados pelo *gateway* para o servidor e usados para análise. O servidor nesse caso foi configurado para ser o computador, que paralelamente estava coletando logs.

5 RESULTADOS E DISCUSSÃO

Após a coleta dos dados realizados no experimento em campo, a equipe reuniu as variáveis de interesse e as métricas para avaliação do desempenho e comprovação do funcionamento da rede com o protocolo implementado.

A configuração experimental consistiu em aplicar o protocolo *mesh* ao cenário real e validar sua métrica de taxa de entrega de pacotes (PDR) e se a comunicação *multi-hop* foi implementada com sucesso.

Com o objetivo de simular um cenário real com um alto tráfego de dados na rede, foi configurado um tempo de 5 segundos de delay para o envio de pacotes. Esse tempo é considerado bem curto e escolhido assim justamente para estressar a rede e incitar a perda de pacotes.

Vale citar também que os nós não foram iniciados ao mesmo tempo, o que poderia gerar um pior cenário de entrega de pacotes, mas como os experimentos foram posicionados fisicamente no cenário, a distância inviabilizou tal ação.

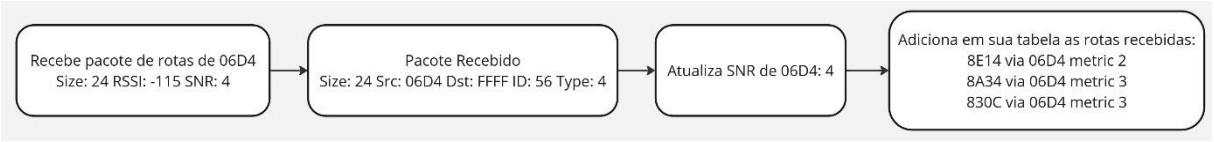
Serão mostradas nas Figuras 20 a 24 o processo observado no *gateway* (8430) pelo monitor serial e transcrevido em modo de fluxograma. O objetivo é mostrar a rotina em loop de processos que o *gateway* realizava. Vale explicar que o ID dos pacotes é incrementado a cada pacote enviado, então se o ID é 11, esse endereço já enviou 11 pacotes, tanto de roteamento quanto de dados. E com relação aos tipos de pacote foi definido como 0 um pacote de dados e como 4 um pacote de roteamento.

FIGURA 20 – FLUXOGRAMA PACOTE DE ROTAS ENVIADO

FONTE: As autoras (2024).

Após o envio de sua mensagem de roteamento o *gateway* recebe mensagens de roteamento de outros *nodes* e atualiza sua tabela de rotas, como pode ser visto na Figura 21.

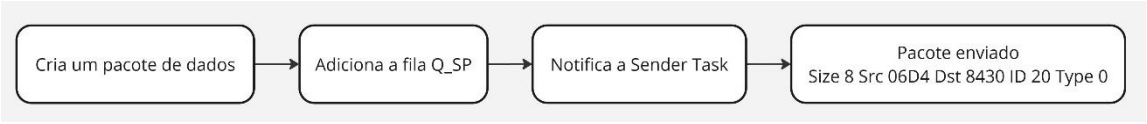
FIGURA 21 – FLUXOGRAMA TABELA DE ROTAS RECEBIDA



FONTE: As autoras.

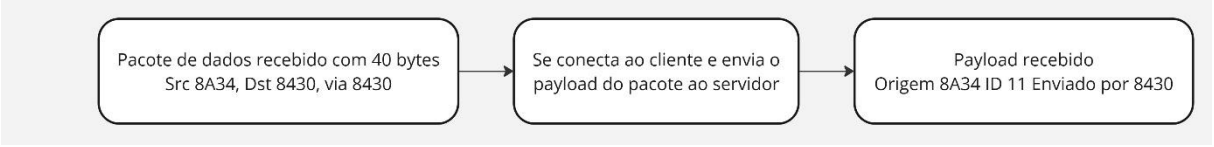
Tendo a tabela de rotas, então o *gateway* começa a receber mensagens de dados direcionadas a ele, pois é o destino. Ao receber a mensagem, o *payload* da mensagem de dados é enviado pelo *gateway* para o servidor. As Figuras 22 e 23 mostram o processo de um pacote de dados sendo enviado por um nó e o *payload* deste pacote sendo enviado ao servidor.

FIGURA 22 – FLUXOGRAMA ENVIA PACOTE DE DADOS



FONTE: As autoras.

FIGURA 23 – *GATEWAY* RECEBE DADOS E ENVIA *PAYLOAD* AO SERVIDOR



FONTE: As autoras.

Analisando as métricas aquisitadas durante o teste:

- *Packet Delivery Ratio* (PDR)

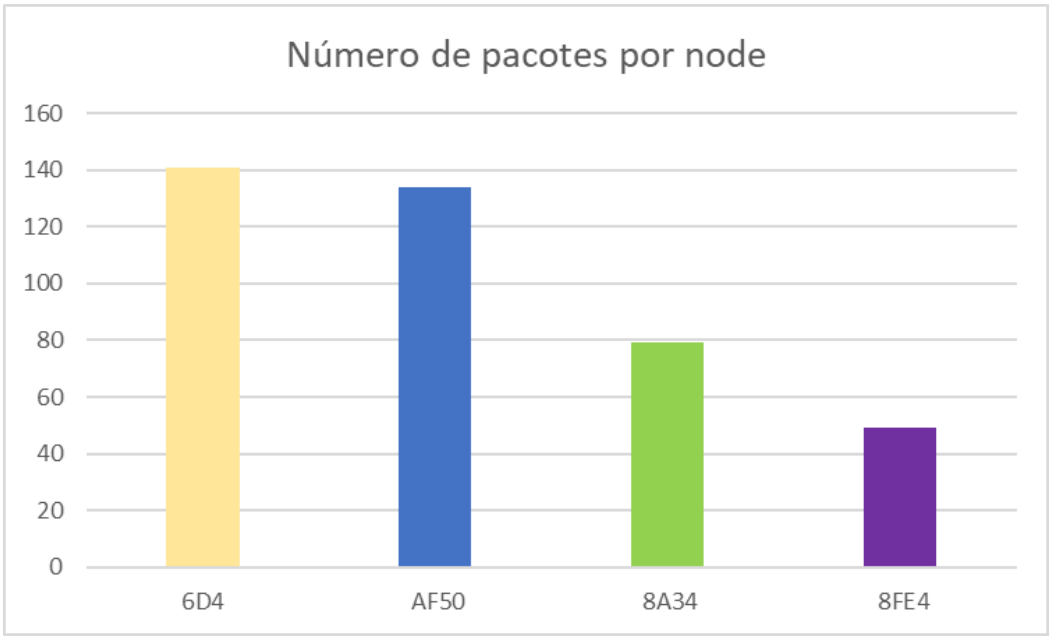
Para calcular o PDR, foi estimada a quantidade de pacotes que os nós devem ter enviado de acordo com o tempo especificado no código. Cada nó envia um pacote de rotas a cada 30s e um pacote de dados a cada 5s. Como o experimento durou 27 minutos no total, foi feita uma estimativa da quantidade de pacotes que foram enviados por cada nó. Para o cálculo da quantidade de pacotes de rotas, foi multiplicada duração do teste por 2 – pois a cada minuto eram enviados 2 pacotes. E para o cálculo do pacote de dados foi multiplicado por 12 – pois eram enviados 12 pacotes a cada minuto. Totalizando assim em 378 pacotes enviados durante o tempo de teste por cada dispositivo, exceto o *gateway*.

$$n\ Pacotes\ de\ rotas = 27 * 2 = 54$$
$$n\ Pacotes\ de\ dados = 27 * 12 = 324$$
$$n\ Pacotes\ totais = 54 + 324 = 378$$

Pelo fato de o protocolo fazer o roteamento de forma proativa, a tabela de rotas mudava conforme a informação de relação sinal-ruído de um certo nó era atualizada. Dessa maneira, o *gateway* atualiza a tabela de rotas sempre que um *node* apresenta uma relação sinal ruído melhor, colocando este como via. Nos testes performados por Freitag et al. As tabelas de rotas foram pré-definidas, portanto, não variavam, como nesse trabalho.

Sendo assim, coletar as medidas de cada dispositivo neste caso foram mais complicadas por este fator. Devido a isso foram calculadas essas medidas para os nós que tinham métrica 1, ou seja, se comunicavam diretamente com o *gateway*. Na Figura 24 está presente o gráfico com o número de pacotes por nó recebidos pelo *gateway* no tempo total do experimento. Os dados apresentados foram baseados nos dados coletados pelo monitor serial.

FIGURA 24 – GRÁFICO DE QUANTIDADE DE PACOTES POR *NODE*



FONTE: As autoras.

Na Tabela 13 está a relação de pacotes enviados e pacotes recebidos pelo *gateway*.

TABELA 13 – RESULTADO CALCULADO DE PDR

Node	Nº Pacotes recebidos pelo *gateway*	Nº de pacotes enviados	PDR [%]
06D4	141	378	37%
AF50	134	378	35%
8A34	79	378	21%
8FE4	49	378	13%

- *Control Overhead*

O parâmetro de *Control Overhead* (Sobrecarga de Controle) para essa aplicação foi de 16%. Esse resultado é consistente, já que a quantidade enviada de pacotes de dados e a quantidade de pacotes de roteamento para todos os nodes é o mesmo.

Esse valor foi calculado considerando a taxa de pacotes de dados e de roteamento enviados por minuto. Para o pacote de rotas, a taxa é de 2 pacotes/min e para o pacote de dados é de 12 pacotes/min. Fazendo o cálculo, resulta em um *Control Overhead* de 16%.

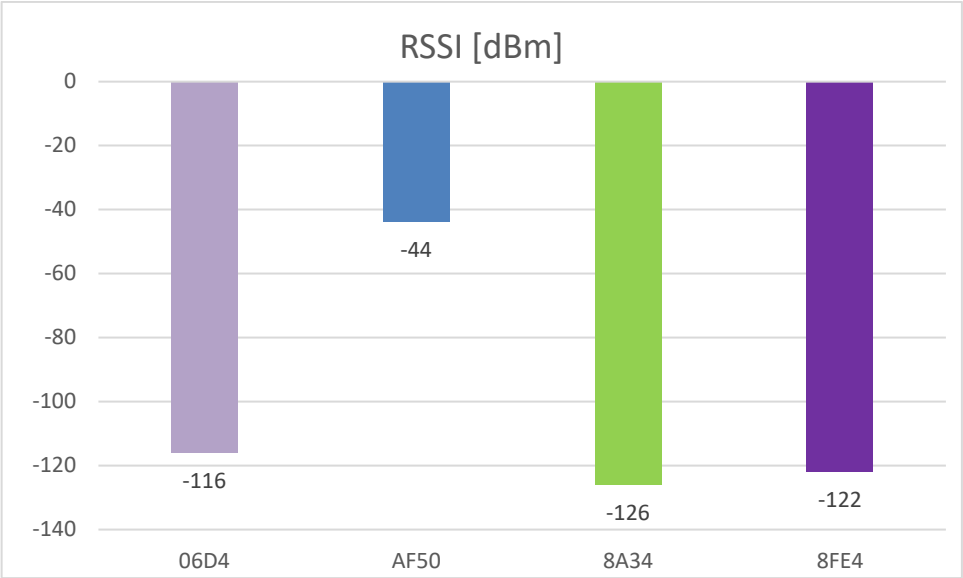
$$Control\ Overhead = \frac{2}{12} * 100 = 16\%$$

O *Control Overhead* indica um aumento nas operações de roteamento de controle, sendo que quanto maior a sua porcentagem, maior é essa sobrecarga. Nesse caso, como o protocolo implementado na aplicação deste trabalho é proativo, o resultado indica que a rede não tem tantos problemas no roteamento de controle, já que a tabela dinâmica auxilia a operação.

- *Received Signal Strength Indicator* (RSSI)

Analisando, pode-se concluir que os *nodes* mais próximos ao *gateway* tiveram valores melhores que os mais distantes, o melhor caso foi do AF50 que estava a 10m de distância. Na Figura 25 está mostrando o valor médio de RSSI por *node*.

FIGURA 25 – GRÁFICO RSSI POR *NODE*



FONTE: As autoras.

6 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho teve como objetivo implementar e avaliar o desempenho de um protocolo de roteamento para redes LoRa *mesh*, além de estudar e compreender o funcionamento da tecnologia Lora e suas diversas aplicações.

A equipe utilizou como base a biblioteca LoraMesher na implementação da rede de comunicação sem fio LoRa, permitindo a realização de alterações que fossem mais compatíveis com a aplicação proposta. Essas alterações não provocaram um prejuízo na qualidade da comunicação, mas sim enriqueceram os resultados coletados para avaliação da rede.

Durante os testes em campo, realizado no campus Centro Politécnico da UFPR, foi concluído de forma satisfatória o objetivo de implementar uma rede *multi-hop*, os dispositivos finais criaram suas tabelas de rotas, dando preferência ao *nodes* com melhor qualidade de sinal (RSSI) na recepção das mensagens, e as compartilharam entre si. Os *nodes* fizeram saltos entre si com a intenção de enviar pacotes ao seu destino final, que foi sempre definido como o *gateway*. Foram observadas as atualizações nas tabelas de rotas conforme eram encontradas melhores rotas, provando que é um protocolo de roteamento pró-ativo. Com relação a cobertura de rede, pode-se concluir que a cobertura com um protocolo *multi-hop* é superior a um protocolo single-hop, tendo em vista que com aquela topologia um pacote de dados fez até três saltos para chegar ao *gateway*, portanto o alcance estaria mais limitado no caso de uma topologia estrela.

Apesar desse trabalho ter utilizado a biblioteca LoraMesher como base, os experimentos realizados por Freitag et al. simulam as topologias testadas via software, portanto os nós tinham uma tabela de rotas pré definidas e ignoravam as verdadeiras mensagens de roteamento recebidas. Neste presente trabalho os experimentos foram feitos em ambiente outdoor, aplicando a topologia *mesh* entre os módulos e trocando mensagens efetivas de roteamento. Isso faz com que as métricas tenham valores diferentes das observadas no experimento do LoRaMesher (FREITAG et al, 2022).

Vale a pena destacar também que ao implementar as métricas dos dados de roteamento, foi possível avaliar que para os casos *multi-hop* não foi possível calcular os dados individuais dos *nodes*, uma vez que não há a implementação de rastreamento dos endereços que passaram pelo pacote. Essa é uma das melhorias

que a equipe avaliou como uma oportunidade para permitir uma análise mais completa do roteamento.

Além disso, também são recomendadas como melhorias para trabalhos futuros a inclusão dos valores das métricas calculadas dentro do *payload* de dados e a implementação de um banco de dados para salvamento dos pacotes recebidos pelo *gateway*, para facilitar na visualização dos parâmetros e pela praticidade na coleta de dados.

A equipe considera importante o desenvolvimento das tecnologias sem fio de longo alcance, por conta do seu potencial para inovação do mercado de *smart cities*.

Os resultados obtidos foram considerados aceitáveis para o trabalho proposto, tendo em vista a implementação e validação da rede *mesh*/*multi-hop* aplicada em campo. O presente trabalho realizado, proporcionou para a equipe um aprimoramento dos conhecimentos referentes a sistemas eletrônicos embarcados aplicados a redes de comunicação sem fio.

REFERÊNCIAS

SEMTECH. **What is LoRa?** Disponível em: <https://www.semtech.com/lora/what-is-lora>. Acesso em: 1 de abr de 2024.

HASSAN, M et al. **Intelligent Transportation Systems in Smart City: A Systematic Survey.** International Conference on Robotics and Automation in Industry (ICRAI), Peshawar, Pakistan, 2023, pp. 1-9, doi: 10.1109/ICRAI57502.2023.10089543.

HUSSAIN, I et al. **Smart city solutions: Comparative analysis of waste management models in IoT-enabled environments using multiagent simulation.** Sustainable Cities and Society, Volume 103, 2024, doi:10.1016/j.scs.2024.105247.

DEVALAL, S. KARTHIKEYAN, A. **LoRa Technology - An Overview.** ICECA, Coimbatore, India, 2018, pp. 284-290, doi: 10.1109/ICECA.2018.8474715.

ARASTEH, H et al. **Iot-based smart cities: A survey.** IEEE 16th EEEIC, Florence, Italy, 2016, pp. 1-6, doi: 10.1109/EEEIC.2016.7555867.

GKOTSIOPOULOS, P. ZORBAS, D. DOULIGERIS, C. **Performance Determinants in LoRa Networks: A Literature Review.** IEEE Communications Surveys & Tutorials, vol. 23, no. 3, pp. 1721-1758, 2021, doi: 10.1109/COMST.2021.3090409.

VANGELISTA, L. **Frequency Shift Chirp Modulation: The LoRa Modulation.** IEEE Signal Processing Letters, vol. 24, no. 12, pp. 1818-1821, Dec. 2017, doi: 10.1109/LSP.2017.2762960.

JOUHARI, M et al. **A Survey on Scalable LoRaWAN for Massive IoT: Recent Advances, Potentials, and Challenges.** IEEE Communications Surveys & Tutorials, vol. 25, no. 3, pp. 1841-1876, 2023, doi: 10.1109/COMST.2023.3274934.

LI, S. RAZA, U and KHAN, A. **How Agile is the Adaptive Data Rate Mechanism of LoRaWAN?**. IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 2018, pp. 206-212, doi: 10.1109/GLOCOM.2018.8647469.HUH, H. KIM, Y.

LoRa-based Mesh Network for IoT Applications. IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 2019, pp. 524-527, doi: 10.1109/WF-IoT.2019.8767242.

LUNDELL, D et al. **A Routing Protocol for LoRA Mesh Networks.** IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), Chania, Greece, 2018, pp. 14-19, doi: 10.1109/WoWMoM.2018.8449743.

FREITAG, F et al. **Implementation of a LoRa Mesh Library**. IEEE Access, vol. 10, pp. 113158-113171, 2022, doi: 10.1109/ACCESS.2022.3217215.

CUNHA, B. **LoRaNet: Implementation of a LoRa Mesh Network.** Departamento de Ciência da Computação da Universidade de São Paulo, SP- São Paulo, 2022.

PRADE, L et al. **Multi-radio and multi-hop LoRa communication architecture for large scale IoT deployment**. Computers and Electrical Engineering, Volume 102, 2022, 108242, ISSN 0045-7906, doi: 10.1016/j.compeleceng.2022.108242.

GAGLIARDI, G. Et al. **Advanced Adaptive Street Lighting Systems for Smart Cities.** MDPI Smart Cities, Rende, Italy, 2020, 3(4), 1495-1512, doi: 10.3390/smartcities3040071.

QUATTROCCHI, A et al. **Designing a Low-Cost System to Monitor the Structural Behavior of Street Lighting Poles in Smart Cities.** Sensors 2023, 23, 6993, doi: 10.3390/s23156993.

SCHULZ, F. **Iluminação Pública.** CREA-PR Série de Cadernos Técnicos da Agenda Parlamentar, 2016. Disponível em: <https:// www.crea-pr.org.br>.

LACTEC. **Smart city: Lactec implementa telegestão na iluminação de Londrina.** Disponível em: <https://lactec.com.br/smart-city-lactec-implementa-telegestao-na-iluminacao-de-londrina/>. 28 de fev de 2024.

MACHADO, Oldon. **Modernização da iluminação pública em BH gera economia anual de R$25mi.** Disponível em:

<https://www.canalenergia.com.br/noticias/53113412/modernizacao-da-iluminacao-publica-em-bh-gera-economia-de-r-25-milhoes>. 27 de set de 2019.

SMARTGREEN. **Telegestão da iluminação pública.** Disponível em: <https://smartgreen.net/solucoes/telegestao-de-iluminacao-publica/>. Acesso em: 10 de jun de 2024.

CECILIO, J. FERREIRA, P. CASIMIRO, A. **Evaluation of a Lora Technology in Flooding Prevention Scenarios.** Universidade de Lisboa, 2020. doi:10.3390/s20144034

SHACON, S e STRAUB. **Pro Git – Everything you need to know about it**. Apress, 2014. Disponível em < https://git-scm.com/book/en/v2 >. Acesso em 3 de out de 2024.

PLATFORMIO. **Your Gateway to Embedded Software Development Excellence.** Disponível em < https://platformio.org/ >. Acesso em: 27 de set de 2024.

SOLÉ, J et al. **LoRaMesher Library.** Disponível em: <https://github.com/LoRaMesher/LoRaMesher>. Acesso em 5 de ago de 2024.

WI-SUN ALLIANCE. **Field Area Networks: Wi-SUN FAN Specification Overview.** 2020. Disponível em: https://www.wi-sun.org. Acesso em: 8 abr. 2024.

3GPP. **Release 13: Narrowband Internet of Things (NB-IoT).** Technical Specification Group Radio Access Network, 2016. Disponível em: https://www.3gpp.org. Acesso em: 11 abr. 2024.

SEMTECH. **LoRa® and LoRaWAN®.** Disponível em: https://www.semtech.com/uploads/technology/LoRa/lora-and-lorawan.pdf. Acesso em: 15 mai. 2024.

AGÊNCIA NACIONAL DE TELECOMUNICAÇÕES (ANATEL). **Ato nº 14.448, de 4 de dezembro de 2017**. Disponível em: <https://www.anatel.gov.br/legislacao/atos-de-2017/1161-ato-14448> Acesso em: 15 mai. 2024.

APÊNDIE A – PARSER.PY

```
import socket

print("Creating server...")
s = socket.socket()
s.bind(('192.168.XX.XXX', 10000))
s.listen(0)

while True:
        client, addr = s.accept()
        while True:
                content = client.recv(1024)
                if len(content) == 0:
                        break
                else:
                        print(content)
        print("Closing connection")
```

APÊNDIE B – MAIN.CPP

```
#include <Arduino.h>
#include "LoraMesher.h"
#include "display.h"
#include <WiFi.h>
#include <inttypes.h> // Adicionado para usar PRIx16
#include <map>
#include "src\entities\packets\DataPacket.h"

const char *ssid = "WIFI";
const char *password = "password";

#define BOARD_LED 25
#define LED_ON LOW
#define LED_OFF HIGH
#define GATEWAY_ADDR 0x8430
// #define WIFI 1

LoraMesher &radio = LoraMesher::getInstance();

uint32_t dataCounter = 0;
uint32_t controlBytes = 0;      // Bytes de controle enviados
uint32_t dataBytes = 0;         // Bytes de dados enviados

struct dataPacket
{
    uint32_t payload[10] = {0, 0, 2, 3, 4, 5, 6, 7, 8, 9};
};

dataPacket *helloPacket = new dataPacket;

struct PacketTiming {
    uint32_t sendTime;
    uint32_t receiveTime;
};

std::map<uint32_t, PacketTiming> packetTimings;  // Map para armazenar os tempos de
envio e recebimento

/**
 * @brief Flash the lead
 *
 * @param flashes number of flashes
 * @param delaymS delay between is on and off of the LED
 */
void led_Flash(uint16_t flashes, uint16_t delaymS)
{
    uint16_t index;
    for (index = 1; index <= flashes; index++)
    {
        digitalWrite(BOARD_LED, LED_ON);
        vTaskDelay(delaymS / portTICK_PERIOD_MS);
        digitalWrite(BOARD_LED, LED_OFF);
        vTaskDelay(delaymS / portTICK_PERIOD_MS);
    }
}

/**
 * @brief Print the counter of the packet
 *
 * @param data
 */
void printPacket(dataPacket *data, uint16_t sourceAddress)
{
    char text[32];
```

```
    snprintf(text, 32, ("%" PRIx16 "-> %d\n"), sourceAddress, data->payload[0]);

    Screen.changeLineThree(String(text));
    Serial.printf("Received data n° %d\n", data->payload[0]);
}

/**
 * @brief Iterate through the payload of the packet and print the counter of the
packet
 *
 * @param packet
 */
void printDataPacket(AppPacket<dataPacket> *packet)
{
    Serial.printf("Packet arrived from %" PRIx16 " with size %d bytes\n", packet-
>src, packet->payloadSize);

    // Debug para verificar se o PacketTiming está sendo atualizado corretamente
    Serial.printf("Packet  %d:  sendTime=%d,  receiveTime=%d\n",  packet->payload-
>payload[0],                    packetTimings[packet->payload->payload[0]].sendTime,
packetTimings[packet->payload->payload[0]].receiveTime);

    // Get the payload to iterate through it
    dataPacket *dPacket = packet->payload;
    size_t payloadLength = packet->getPayloadLength();
    #ifdef WIFI
    WiFiClient client;
    if (!client.connect(IPAddress(192,168,XX,XXX), 10000))
    { // <<<<<< fixed
        // if (!client.connect("192.168.XX.XXX", 10000)) {            // <<< or like
this
        Serial.println("Connection to host failed");
        delay(1000);
        return;
    }
    Serial.println("client connected sending packet"); // <<< added

    printPacket(&dPacket[0], packet->src);

    Serial.printf("---- Payload ---- Payload length in dataP: %d \n", payloadLength);

    for (size_t i = 0; i < payloadLength; i++)
    {
        Serial.printf("Received data n° %d", i);

        for (size_t j = 0; j < 10; j++)
        {
            Serial.printf("%d, ", dPacket[i].payload[j]);
            client.printf("%d, ", dPacket[i].payload[j]);
        }
        Serial.println();
        client.printf("Id: %X\r\n", radio.getLocalAddress());
        client.println();
    }


    Serial.println("---- Payload Done ---- ");
    client.println("---- Payload Done ---- ");
    #endif
}

/**
 * @brief Function that process the received packets
 *
 */
void processReceivedPackets(void *)
{
    for (;;)
```

```
    {
        /* Wait for the notification of processReceivedPackets and enter blocking */
        ulTaskNotifyTake(pdPASS, portMAX_DELAY);
        led_Flash(1, 100); // one quick LED flashes to indicate a packet has arrived

        // Iterate through all the packets inside the Received User Packets Queue
        while (radio.getReceivedQueueSize() > 0)
        {
            Serial.println("ReceivedUserData_TaskHandle notify received");
            Serial.printf("Queue        receiveUserData        size:        %d\n",
radio.getReceivedQueueSize());

            // Get the first element inside the Received User Packets Queue
            AppPacket<dataPacket> *packet = radio.getNextAppPacket<dataPacket>();

            // Print the data packet
            printDataPacket(packet);

            // Delete the packet when used. It is very important to call this function
to release the memory of the packet.
            radio.deletePacket(packet);
        }
    }
}

TaskHandle_t receiveLoRaMessage_Handle = NULL;

/**
 * @brief Create a Receive Messages Task and add it to the LoRaMesher
 *
 */
void createReceiveMessages()
{
    int res = xTaskCreate(
        processReceivedPackets,
        "Receive App Task",
        4096,
        (void *)1,
        2,
        &receiveLoRaMessage_Handle);
    if (res != pdPASS)
    {
        Serial.printf("Error: Receive App Task creation gave error: %d\n", res);
    }
}

/**
 * @brief Initialize LoRaMesher
 *
 */
void setupLoraMesher()
{
    // Get the configuration of the LoRaMesher
    LoraMesher::LoraMesherConfig config = LoraMesher::LoraMesherConfig();

    // Set the configuration of the LoRaMesher (TTGO T-BEAM v1.1)
    config.loraCs = 18;
    config.loraRst = 14;
    config.loraIrq = 26;
    config.loraIo1 = 35;

    config.module = LoraMesher::LoraModules::SX1276_MOD;

    // Init the loramesher with a configuration
    radio.begin(config);

    // Create the receive task and add it to the LoRaMesher
    createReceiveMessages();
```

```
    // Set the task handle to the LoRaMesher
    radio.setReceiveAppDataTaskHandle(receiveLoRaMessage_Handle);

    // Start LoRaMesher
    radio.start();

    Serial.println("Lora initialized");
}

/**
 * @brief Displays the address in the first line
 *
 */
void printAddressDisplay()
{
    char addrStr[15];
    snprintf(addrStr, 15, "Id: %X\r\n", radio.getLocalAddress());

    Screen.changeLineOne(String(addrStr));
}

/**
 * @brief Print the routing table into the display
 *
 */
void printRoutingTableToDisplay()
{

    // Set the routing table list that is being used and cannot be accessed (Remember
to release use after usage)
    LM_LinkedList<RouteNode> *routingTableList = radio.routingTableListCopy();

    routingTableList->setInUse();

    Screen.changeSizeRouting(radio.routingTableSize());

    char text[15];
    for (int i = 0; i < radio.routingTableSize(); i++)
    {
        RouteNode *rNode = (*routingTableList)[i];
        NetworkNode node = rNode->networkNode;
        snprintf(text, 15, ("|%X(%d)->%X"), node.address, node.metric, rNode->via);
        Screen.changeRoutingText(text, i);
    }

    // Release routing table list usage.
    routingTableList->releaseInUse();

    // Delete routing table list
    delete routingTableList;

    Screen.changeLineFour();
}

/**
 * @brief Every 20 seconds it will send a counter to a position of the dataTable
 *
 */
void sendLoRaMessage(void *)
{
    int dataTablePosition = 0;

    for (;;)
    {
        if (radio.routingTableSize() == 0)
        {
            vTaskDelay(5000 / portTICK_PERIOD_MS);
```

```
            continue;
        }

        if (radio.routingTableSize() <= dataTablePosition)
            dataTablePosition = 0;

        LM_LinkedList<RouteNode> *routingTableList = radio.routingTableListCopy();


        uint16_t addr = (*routingTableList)[dataTablePosition]->networkNode.address;

        if(addr != GATEWAY_ADDR){
            addr = GATEWAY_ADDR;
        }

        Serial.printf("Send data packet n° %d to %X (%d)\n", dataCounter, addr,
dataTablePosition);

        dataTablePosition++;

        // Atualizar bytes de controle e dados
        controlBytes += sizeof(dataPacket);   // Considerando que todo pacote de
controle tem o tamanho da estrutura dataPacket
        dataBytes += sizeof(helloPacket->payload); // Atualizar para um tamanho
definido do payload

        // Create packet and send it.
        helloPacket->payload[0] = (uint32_t) radio.getLocalAddress();
        helloPacket->payload[1]++;

        radio.createPacketAndSend(addr, helloPacket, 1);

        // Print second line in the screen
        Screen.changeLineTwo("Send " + String(dataCounter));

        // Increment data counter
        // helloPacket->counter[0] = dataCounter++;
        dataCounter++;

        // Print routing Table to Display
        printRoutingTableToDisplay();

        // Release routing table list usage.
        delete routingTableList;

        // Wait 20 seconds to send the next packet
        vTaskDelay(5000 / portTICK_PERIOD_MS);
    }
}

/**
 * @brief Setup the Task to create and send periodical messages
 *
 */
void createSendMessages()
{
    TaskHandle_t sendLoRaMessage_Handle = NULL;
    BaseType_t res = xTaskCreate(
        sendLoRaMessage,
        "Send LoRa Message routine",
        4098,
        (void *)1,
        1,
        &sendLoRaMessage_Handle);
    if (res != pdPASS)
    {
        Serial.printf("Error: Send LoRa Message task creation gave error: %d\n", res);
        vTaskDelete(sendLoRaMessage_Handle);
```

```
    }
}

void createSocket()
{
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.println("...");
    }
    Serial.print("WiFi connected with IP:");
    Serial.println(WiFi.localIP());
}

void setup()
{
    Serial.begin(115200);
    pinMode(BOARD_LED, OUTPUT); // setup pin as output for indicator LED

    led_Flash(2, 125); // two quick LED flashes to indicate program start
    #ifdef WIFI
    createSocket();
    #endif
    setupLoraMesher();
    createSendMessages();

}

void loop()
{
    vTaskPrioritySet(NULL, 1);
    vTaskDelay(30000/portTICK_PERIOD_MS);
}
```