UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

AFONSO HENRIQUE KINGESKI

GUSTAVO PACOLA GONÇALVES

HENRIQUE LOPES SENGER

HYDRA, UM NOVO ECOSSISTEMA IOT EM *MESH*

CURITIBA

2024

AFONSO HENRIQUE KINGESKI
GUSTAVO PACOLA GONÇALVES
HENRIQUE LOPES SENGER

HYDRA, UM NOVO ECOSSISTEMA IOT EM *MESH*

Hydra, a new mesh IoT ecosystem

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Eletrônica do Curso de Bacharelado em Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Hermes Irineu Del Monego

Coorientador: Prof. Dr. Bruno Sens Chang

CURITIBA

2024

AFONSO HENRIQUE KINGESKI
GUSTAVO PACOLA GONÇALVES
HENRIQUE LOPES SENGER

HYDRA, UM NOVO ECOSSISTEMA IOT EM *MESH*

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Eletrônica do Curso de Bacharelado em Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 27/Junho/2024

Hermes Irineu Del Monego
Doutor
Universidade Tecnológica Federal do Paraná

Bruno Sens Chang
Doutor
Universidade Tecnológica Federal do Paraná

Daniel Rossato de Oliveira
Doutor
Universidade Tecnológica Federal do Paraná

Kleber Kendy Horikawa Nabas
Doutor
Universidade Tecnológica Federal do Paraná

CURITIBA
2024

RESUMO

Dado o grande crescimento no número de dispositivos no campo da Internet das Coisas e um potencial ainda maior para os próximos anos, necessitam-se de novas soluções que atendam as demandas deste crescimento. Partindo de um trabalho anterior feito pelos autores, o presente trabalho contempla a idealização e desenvolvimento de um novo ecossistema IoT, denominado Hydra. É explicitada a concepção da pilha de protocolos, *hardware*, do *firmware* e de um simulador de algoritmos *mesh*. Ao final, são expostos resultados de testes, como o da rede funcionando, do consumo energético do *hardware* e das distâncias máximas de comunicação obtidas.

Palavras-chave: internet das coisas; *mesh*; bluetooth *low energy*; lora; redes de comunicação sem fio.

ABSTRACT

Given the large growth in the number of devices in the field of the Internet of Things and an even greater potential for the coming years, new solutions are needed to meet the demands of this growth. Starting from a previous work done by the authors, this work contemplates the idealization and development of a new IoT ecosystem, called Hydra. The design of hardware, firmware and a mesh algorithm simulator are explained. At the end, test results, such as the network running, the energy consumption of the hardware and the maximum communication distances obtained are exposed.

Keywords: *internet of things*; *mesh*; bluetooth *low energy*; lora; wireless networks.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

Abreviaturas

art.	Artigo
cap.	Capítulo
sec.	Seção

Siglas

ATT	*Atributte Protocol*
BLE	*Bluetooth Low Energy*
BR/EDR	*Basic Rate/Enhaced Data Rate*
BW	*Bandwidth*
CAD	*Channel Activity Detection*
CDMA	*Code Division Multiple Access*
CPU	*Central Processing Unit*
CR	*Code Rate*
CSMA/CA	*Carrier-sense multiple access with collision avoidance*
CSRK	*Connection Signature Resolving Key*
CSS	*Chirp Spread Spectrum*
FEC	*Forward Error Correction*
GAP	*Generic Access Profile*
GATT	*Generic Attribute Profile*
GFSK	*Gaussian Frequency Shift Keying*
GSMA	*Global System for Mobile Communications*
HCI	*Host Controller Interface*
IDF	*IoT Development Framework*

IoT	*Internet of Things*
IP	*Internet Protocol*
IRK	*Identity Resolving Key*
ISM	*Industrial, scientific, and medical*
ISO	*International Organization for Standardization*
L2CAP	*Logical Link Control and Adaptation Protocol*
LoRa	*Long Range*
LPWAN	*Low Power Wide Area Networking*
LTK	*Long-Term Key*
MAC	*Medium Access Control*
MQTT	*Message Queue Telemetry Transport*
OSI	*Open Systems Interconnection*
PCB	*Printer Circuit Board*
PRR	*Packet Reception Ratio*
QoS	*Quality of Service*
QUCS	*Quite Universal Circuit Simulator*
RSSI	*Received Signal Strength Indicator*
SF	*Spreading Factor*
SIG	*Special Interest Group*
SMP	*Security Manager Protocol(SMP)*
SNR	*Signal to Noise Ratio*
SOC	*System-on-Chip*
STK	*Short-Term Key*
UART	*Universal asynchronous receiver-transmitter*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
UNB	*Ultra Narrow Band*

USB	*Universal Serial Bus*
UTFPR	Universidade Tecnologica do Paraná
VSWR	Voltage standing wave ratio

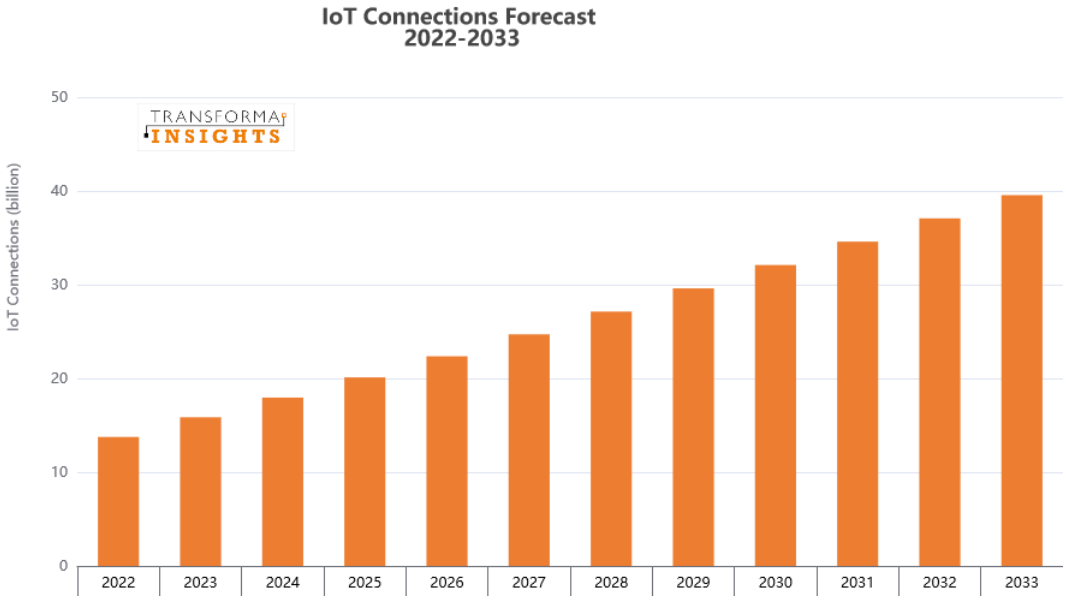
SUMÁRIO

1 INTRODUÇÃO

1.1 Justificativa

Dispositivos IoT(*Internet of Things*, do inglês, Internet das Coisas) vêm sendo utilizados nos mais diversos setores como Indústria 4.0, cidades inteligentes e automação residencial. De acordo com Transforma Connections (2024), o número desses dispositivos pode passar de 30 bilhões no ano de 2030 e chegar a quase 40 bilhões em 2033. Isto tem feito com que a densidade destes dispositivos aumente cada vez mais, exigindo soluções que atendam novas demandas.

Figura 1 – Previsão do Número de Dispositivos IoT Conectados



Fonte: Transforma Connections (2024).

A quantidade de novas soluções para o campo da Internet das Coisas cresce todos os anos. Um exemplo recente é o Amazon Sidewalk(Amazon, 2024), uma implementação da Amazon para interconectividade entre seus dispositivos, como o seu ecossistemas de campainhas inteligentes, o Amazon Ring. O Amazon Sidewalk cria uma rede privada, permitindo que usuários compartilham uma parte da largura de banda de Internet com outros dispositivos nas proximidades sem conectividade direta a uma rede Wi-Fi. No entanto, essa solução é proprietária, limitando o seu uso a dispositivos fornecidos pela Amazon.

Uma alternativa já existente é a The Things Network(The Things Network, 2024), uma rede global de IoT de código aberto, baseada na tecnologia LoRaWAN(Semtech, 2015). Ela oferece uma infraestrutura colaborativa e descentralizada que permite a troca de dados entre dispositivos IoT e é aberta para qualquer um que queira se utilizar da tecnologia. Porém, devido ao fato do LoRaWAN operar na topologia de rede estrela, ela depende que todos os dispositivos estejam na área de cobertura de um *gateway* LoRaWAN, que faz uso de *hardware* especializado

proprietário que possui custo elevado, dificultando a popularização e alcance da solução. Estes problemas são mitigados utilizando uma rede *mesh*, uma vez que o nó não precisa estar no alcance do *gateway*, mas sim no alcance de outros nós.

Dados essas limitações, a ideia do nosso projeto é desenvolver uma solução de implementação de rede sem-fio utilizando tecnologia LoRa(*Long Range*, do inglês, Longa Distância) e o Bluetooth *Low Energy*(BLE), buscando aplicar os conhecimentos adquiridos durante o curso de Engenharia Eletrônica para solucionar os problemas salientados acima.

1.2 Objetivos

1.2.1 Objetivo geral

Desenvolver uma solução para a comunicação entre dispositivos IoT, através de uma rede de topologia *mesh* híbrida, utilizando rádios LoRa e Bluetooth. O projeto engloba a parte de concepção da pilha de protocolos, *hardware* e *software* necessários para a criação de uma rede.

1.2.2 Objetivos específicos

- Projetar o *hardware* de um módulo microcontrolado com conectividade LoRa e Bluetooth e uma interface para dispositivos IoT
- Integrar diferentes dispositivos BLE por meio da tecnologia LoRa
- Projetar e implementar um simulador de eventos discretos com o intuito de testar e validar protocolos de roteamento *mesh*
- Implementar um protocolo de roteamento *mesh* entre os nós
- Desenvolver uma classe de dispositivo capaz de fazer a integração entre os módulos e a Internet utilizando o protocolo MQTT(*Message Queue Telemetry Transport*, do inglês, Transporte de Filas de Mensagem de Telemetria)
- Implementar uma aplicação de exemplo do uso da rede.

1.3 Estrutura do trabalho

O presente trabalho possui uma organização em seis capítulos, incluindo esta introdução. A seguir, serão apresentadas breves explicações dos próximos capítulos e suas peculiaridades.

O segundo capítulo, Referencial Teórico, trata da fundamentação teórica e revisão bibliográfica das tecnologias utilizadas na construção da rede Hydra. São abordadas noções básicas e conceitos técnicos necessários para um entendimento apropriado do projeto, tais como Internet das Coisas e sistemas embarcados. Também são discutidos especifidades de uma rede de comunicação em topologia *mesh* e *Low Power Wide Area Networks*(LPWAN, do inglês, Redes de Grande Àrea e Baixo Consumo), com o intuito de criar um embasamento teórico para compreender os parâmetros utilizados nos procedimentos experimentais e nas discussões dos resultados.

O terceiro capítulo, Análise Crítica de Trabalho Anterior, busca elucidar os conceitos iniciais, construção e resultados de um trabalho anterior dos autores. Neste capítulo, há um foco nas limitações do antigo trabalho que o presente projeto buscou aperfeiçoar.

O quarto capítulo, Desenvolvimento, apresenta os detalhes da concepção e produção dos elementos do projeto. São abordadas a definição do escopo do projeto, o processo de elaboração do *hardware* e exposição da estruturação do protocolo. Além disso, são explicitados os desenvolvimentos do simulador da rede Hydra e do *firmware* dos módulos.

No quinto capítulo, Resultados e Discussões, são expostos os diferentes resultados do projeto, desde os resultados de simulação dos protocolos implementados até as características do *hardware* do projeto, como nível de sinal e consumo energético. Todos os resultados são discutidos ao longo do capítulo.

Por fim, o sexto e último capítulo, Conclusão, destaca as inferências que puderam ser observadas ao longo do projeto como um todo, ressaltando a importância do seu desenvolvimento.

2 REFERENCIAL TEÓRICO

2.1 Sistemas Embarcados

Para que as informações que circulam na rede mundial possam interagir com o mundo real, é preciso que existam dispositivos capazes processar essas informações de forma eficiente e integrar a realidade física com a digital. Neste ponto entram os sistemas embarcados. Segundo Almeida, Moraes e Seraphim(2016), sistemas embarcados são sistemas eletrônicos microprocessados que possuem um função específica que geralmente, após serem programados, não pode ser alterada. São sistemas com recursos computacionais e interface bastante limitados e, desta forma, possuem técnicas de programação bastante distintas dos computadores convencionais.

Os sistemas embarcados podem ser divididos em duas grandes instâncias: o *hardware* e o *software*. O *hardware* se refere aos componentes físicos dos dispositivos que compõe um sistema embarcado. De acordo com Toniolo(2018), o principal componente é a CPU(*Central Processing Unit*, do inglês, Unidade Central de Processamento), responsável pelo cálculos e processos a serem executados. Além disso, existem todos os componentes que interagem com a CPU, como a memória, a interface com o usuário(botões e *displays*), barramentos para comunicação externa e outros, também chamados de periféricos do microcontrolador. Já o *software*, conhecido também como *firmware* no jargão de sistemas embarcados, é o conjunto de algoritmos e instruções que permitem que os periféricos trabalhem em conjunto para uma determinada aplicação.

2.2 Modelo OSI

Com o intuito de criar uma interoperabilidade entre diferentes sistemas e fabricantes, foi necessário o desenvolvimento de uma estrutura conceitual para entender e padronizar os protocolos de redes de computadores. Criada no final da década de 1970 pela ISO(*International Organization for Standardization*, do inglês, Organização Internacional de Padronização), o modelo OSI(*Open Systems Interconnection*, do inglês, Interconexão de Sistemas Abertos) é um padrão que engloba todas as facetas das comunicações de dados em redes(FOROUZAN, 2010).

De acordo com Kurose e Ross (2017), para estruturar o desenho de um protocolo de rede, os desenvolvedores geralmente organizam os protocolos em camadas, sendo que cada camada pode ser implementada em *hardware*, *software* ou ainda uma combinação dos dois. O modelo OSI se divide em sete camadas: camada de aplicação, camada de apresentação, camada de sessão, camada de transporte, camada de rede, camada de enlace de dados e a camada física. Estas camadas possuem uma ordem e uma numeração respectiva, o que pode ser observado na Figura 2.

Figura 2 – Camadas do Modelo OSI

Fonte: Forouzan (2010).

Cada camada se comunica somente com as suas camadas adjacentes e, assim, as informações trafegam por todas as camadas para ir de um meio físico(camada física, nível mais baixo) para uma aplicação(camada de aplicação, nível mais alto). A seguir, será descrito brevemente características de cada camada, de acordo com Forouzan (2010):

7. **Aplicação**: o papel da camada de aplicação é fornecer serviços ao usuário. É ela quem habilita o usuário final acesso à rede, fornecendo uma interface para os serviços como gerenciamento de banco de dados ou envios de mensagens.

6. **Apresentação**: a camada de apresentação assume a responsabilidade de proporcionar serviços que permitem que aplicações interpretem as informações transacionadas entre si(KUROSE; ROSS, 2017). São exemplos destes serviços a compactação e descompactação de dados, a tradução de padrões de caracteres e também a criptografia de dados.

5. **Sessão**: a camada de sessão trabalha como um gerenciador de conversação. Entre as responsabilidade desta camada têm-se controle de diálogo - permitindo que, por exemplo, a comunicação entre dois processos ocorra em um sentido ou simultaneamente - e também é responsável pela sincronização da troca de informações, adicionando pontos de sincronização em um fluxo de dados, por exemplo.

4. **Transporte**: a camada de transporte é responsável por transportar as informações advindas da camada de sessão, garantindo a entrega dos pacotes. A camada é encarregada da segmentação e remontagem da mensagem, bem como a supervisão do controle de erros.

3. **Rede**: a camada de rede é responsável pelo serviço de entrega dos pacotes de informação da origem ao destino. Entre suas responsabilidades, pode-se citar o endereçamento e o roteamento das mensagens.

2. **Enlace de dados**: a camada de enlace dados emprega diversos artifícios - como controle de erros, controle de acesso, controle de fluxo e empacotamento de *bits* em pacotes de dados gerenciáveis(quadros)- para que os dados da camada física aparentem estar livre de erros para a camada de rede.

1. **Física**: a camada física é responsável pela transmissão dos *bits* de dados de informação para o meio físico. Isto pode ocorrer de diversas manerias, seja através de ondas de rádio, sinais luminosos(como nas fibras ópticas) ou sinais elétricos.

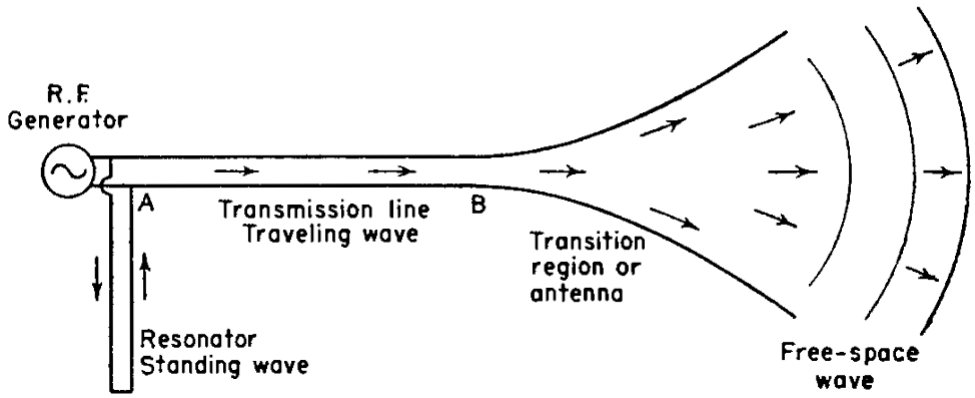
2.3 Antenas

2.3.1 Fundamentos

Muitos dispositivos se comunicam entre si através de cabos, sejam eles de cobre ou compostos por fibras de vidro que direcionam pulsos luminosos. Uma das formas de eliminar o uso de cabos, utilizando o próprio meio, como o ar, a água e até mesmo o vácuo para transmitir informações, é utilizando um componente passivo denominado antena. Segundo Kraus (1950), uma antena pode ser definida como o elemento ligado à região de transição entre um guia de onda e uma onda no espaço livre, e vice-versa.

Para um melhor entendimento deste conceito, pode-se observar a Figura 3. Conforme Kraus (1950), uma linha de transmissão é um equipamento para transmitir(ou guiar) energia de radiofrequência de um ponto a outro, buscando sempre a menor atenuação e perdas por radiação possível. Conectando um gerador em uma linha de transmissão sem perdas, surgirá uma onda uniforme ao longo da linha. Na Figura 3, a partir do ponto B, a linha de transmissão começa a se afastar, gerando diferentes comprimentos de onda e irradiando as ondas para o espaço livre. Esta é a região que caracteriza uma antena.

Figura 3 – Antena como região de transição



Fonte: Kraus (1950).

2.3.2 Conceitos

Em sistemas de comunicação sem fio por radiofrequência, as antenas são responsáveis por irradiar os sinais com as informações relevantes. Dado que existem diversos tipos de antenas com diferentes propósitos, alguns conceitos básicos são necessários para um melhor entendimento sobre o tema. A seguir, serão explicados alguns desses conceitos, segundo Cisco (2007):

- **Antenas Omnidirecionais**: antenas omnidirecionais referem-se a antenas cujo padrão de irradiação é circular, ou seja, irradiam para todas a direções. São exemplos de antenas omnidirecionais os dipolos e as antenas colineares.
- **Antenas Direcionais**: antenas direcionais são as antenas que irradiam sua energia em uma direção de maneira mais eficiente que outras. São exemplos de antenas direcionais as antenas *patch* e as antenas Yagi.
- **Antena Isotrópica**: a antena isotrópica se refere a uma antena teórica sem perdas, que irradia sua potência igualmente em todas a direções.
- **Ganho**: o ganho de uma antena é definido como a razão entre o seu ganho de potência em determinada direção e o ganho de potência de uma antena de referência na mesma direção. Geralmente usa-se o ganho de uma antena isotrópica como referência, cujo ganho é 1(ou 0dB).
- **Polarização**: antenas direcionam ondas eletromagnéticas que variam ao longo do tempo ao passo que trafegam pelo meio. A forma com que essa onda varia determina a sua polarização: se o campo elétrico varia verticalmente sempre no mesmo plano, é dito que a antena é polarizada linearmente; se a onda rotaciona enquanto trafega pelo

meio, é dito que a antena é polarizada elipticamente; ou seja, as antenas possuem sensibilidade maior para sinais com a mesma polarização que a sua própria.

- **VSWR**: a sigla VSWR vem de *voltage standing wave ratio*, do inglês, razão de onda estacionária de tensão. É definida como a razão entre a máxima e a mínima tensão do padrão da onda estacionária. Como uma onda estacionária surge quando potência é refletida pela carga, a VSWR determina quanto de potência é entregue para a carga e quanto é refletida.

2.4 Internet das Coisas

A Internet das Coisas, ou IoT(*Internet of Things*), se refere a um paradigma de dispositivos interconectados que tem se desenvolvido nas últimas duas décadas. De acordo com Patel *et al.* (2016), a IoT é uma rede de objetos físicos conectados sobre IP(*Internet Protocol*), não somente computadores, mas dispositivos de todos os tipos e tamanhos, desde carros e edifícios até brinquedos e animais. Todos esses objetos conectados sob protocolos pré-determinados, trocando informações para que se possa atingir um melhor controle de processos, logística, posicionamento e outros.

Segundo Buyya e Dastjerdi (2016), o termo ”Internet das Coisas” foi cunhado, em 1999, por Kevin Ashton, um diretor executivo do Instituto de Tecnologia de Massachusetts quando ele apresentava gerenciamento da cadeia de suprimentos. Ele acreditava que, devido aos avanços com computadores, Internet e geração de dados, a maneira de se interagir com as ”coisas” deveria ser reconsiderada. Desde então, o número de dispositivos conectados a rede IoT tem aumentado vertiginosamente: em 2022, foram estimados cerca de 14 bilhões de dispositivos, havendo previsões de, até 2030, haver mais de 30 bilhões dispositivos conectados à internet(Transforma Connections, 2024).

Este número se deve à enorme gama de aplicações que o IoT atinge: casas inteligentes com automação de luzes e temperatura ambiente ou, ainda, monitoramento remoto de câmeras e alarmes; agricultura inteligente, para monitoramento de parâmetros do solo e monitoramento climático; medicina, com acompanhamento constante de sinais vitais de pacientes e equipamentos clínicos; e muitos outros. Embora existam desafios para as empresas do ramo IoT, como escalabilidade, interoperabilidade e segurança (KAVRE; GADEKAR; GADHADE, 2019), cada vez mais vêm-se descobrindo outras aplicações, fazendo com que o número de dispositivos aumente ainda mais.

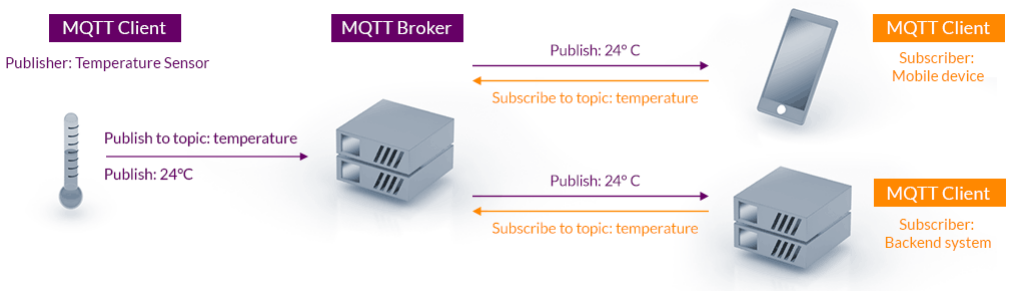
2.5 *Message Queue Telemetry Transport*(MQTT)

O MQTT(*Message Queue Telemetry Transport*, do inglês, Transporte de Filas de Mensagem de Telemetria) é um protocolo de mensagens construído em cima do protocolo TCP/IP.

Inventado e desenvolvido pela IBM no final dos anos 90, tinha como função original conectar sensores em oleodutos a satélites e hoje é um dos protocolos mais utilizados para comunicação IoT(YUAN, 2021). O protocolo é desenhado para ser um serviço de transporte de mensagens do gênero ”publicação/inscrição” extremamente leve, utilizando um mínimo de banda e, portanto, podendo escalar para milhões de dispositivos IoT(MQTT, 2022).

De acordo com Yuan (2021), o protocolo MQTT define dois tipos de entidade na rede: o MQTT *broker*(algo que se traduz como um intermediário de mensagens) e os clientes. O MQTT *broker* nada mais é do que um servidor responsável por receber todas as mensagens de todos os clientes e distribuí-las para todos os clientes relevantes. Um cliente é qualquer elemento que possa interagir com a rede para enviar e receber mensagens do *broker*. Por exemplo, sensores IoT ou aplicações de banco de dados que possam processar dados dos sensores IoT. A Figura 4 demonstra o funcionamento da arquitetura do protocolo MQTT. No exemplo, temos um sensor de temperatura que tem a função somente de publicar uma mensagem com o valor da temperatura. O MQTT *broker* então distribui esta mensagem para todos os dispositivos que estejam inscritos no tópico, no caso do exemplo, um dispositivo móvel e um sistema de *backend*.

Figura 4 – Arquitetura do protocolo MQTT



Fonte: MQTT (2022).

2.6 Redes *Mesh*

2.6.1 Fundamentos

Uma rede *mesh* é, de forma direta, uma topologia de rede onde cada ponto de comunicação, denominado nó, pode se conectar com qualquer outro nó e transmitir informações de forma não hierárquica de um ponto ao outro. De acordo com Methley(2009), redes de comunicação sem-fio em *mesh* possuem diversas implementações comerciais, que vão desde redes de sensores sem-fio para monitoramento de ambientes industriais até redes veiculares para prevenção de acidentes.

Entre as implementações comerciais, pode-se citar a tecnologia Zigbee(ALLIANCE, 2024a) - uma rede *mesh* otimizada energicamente, utilizada principalmente para automação residencial e controle industrial - e também a Iridium (INC., 2024), uma rede de satélites que,

juntos, oferecem uma cobertura *mesh* global para transmissão de dados, comunicação por voz e outros serviços.

Segundo Zhang(2006), uma rede de comunicação sem-fio em *mesh* possui como principais vantagens a rápida implementação, facilidade de manutenção, alta escalabilidade e confiabilidade a um baixo custo. Tudo isso devido a capacidade de auto-configuração, auto-organização e auto-manutenção da rede, o que ainda aumenta a resiliência, conectividade e capacidade da rede como um todo.

2.6.2 Desempenho da rede

Durante a implementação de uma rede sem-fio em *mesh*, existem alguns parâmetros que ajudam a mensurar o seu desempenho, de forma a validar requisitos de projeto ou ainda comparar diferentes soluções. A seguir, serão citados alguns destes parâmetros.

2.6.2.1 QoS (*Quality of Service*) e Largura de Banda

O *Quality of Service*(QoS, do inglês, Qualidade do Serviço) é uma métrica bastante utilizada para comparar diferentes tipo de abordagens em topologias *mesh*. Segundo Methley(2009), um usuário espera um QoS que abranja certos parâmetros, que geralmente são: largura de banda, latência, perdas de pacotes e disponibilidade.

Várias aplicações são demandadas pelas mais variadas necessidades. Cada vez mais, nota-se uma exigência para uma largura de banda mais simétrica de transmissão e recepção de dados. Aplicações voltadas para o compartilhamento de arquivos multimídia exigem isso, e estão levando a um aumento na largura de banda (METHLEY, 2009).

Assim, QoS e largura de banda são duas relevantes moedas de troca para redes em topologia *mesh*.

2.6.2.2 Conectividade *Mesh*

Um outro parâmetro de extrema importância para validação de uma rede *mesh* é a sua conectividade. Para atingir uma conectividade *mesh* eficiente, a auto-organização da rede é necessária (K.C, 2016).

Inúmeros protocolos podem ser utilizados para que a rede seja autossuficiente e atue de forma independente. Assim, algoritmos de manutenção da rede e mecanismos que permitam que a rede performe de forma autônoma são bastante valiosos, permitindo não somente uma robustez na solução, como também uma facilidade no uso e implementação.

2.6.2.3 Escalabilidade

Outro parâmetro a ser analisado é a escalabilidade da rede. Redes sem-fio em *mesh* são especialmente afetadas quando a cobertura e número de nós na rede aumenta. Embora uma das vantagens desta topologia seja aumentar a extensão geográfica da rede para reduzir o custo total da infraestrutura, o número de saltos também aumenta e, consequentemente, esgota-se o recurso energético para performar a transmissão de rádio (ZHANG; LUO; HU, 2006). De maneira semelhante, o aumento do número de nós faz com que o rendimento da rede caia devido ao aumento do número de colisões durante as transmissões(ZHANG; LUO; HU, 2006).

Desta forma, diferentes tipos de protocolos de roteamento, hierarquia de nós e aplicações possuem diferentes maneiras de lidar com o desafio de aumentar a escalabilidade da rede.

2.6.2.4 Interoperabilidade

Por fim, uma característica bastante aferida quando estuda-se redes sem-fio *mesh* é a sua interoperabilidade. É desejável que os nós da rede possam ser integrados com outras formas de comunicação sem-fio para atingir um melhor desempenho(K.C, 2016).

2.7 *Low Power Wide Area Networks*(LPWAN)

2.7.1 Fundamentos

Como referido nas seções anteriores, o IoT criou uma necessidade extraordinária por sensores e atuadores nos mais diversos campos, reivindicando soluções que atingissem o máximo de indivíduos e sanassem a demanda energética. Uma vez que tecnologias convencionais, como o *Bluetooth*(SIG, 2024) e o Wi-fi (ALLIANCE, 2024b), cobrem uma área reduzida, um novo campo surgiu no ramo da Internet das Coisas: as tecnologias *Low Power Wide Area Networks*(LPWAN), que buscam a cobertura de uma grande área geográfica a um baixíssimo custo de energia(RAZA; KULKARNI; SOORIYABANDARA, 2017).

As tecnologias LPWAN prenunciam uma área de cobertura na escala de quilômetros, com baterias que podem durar anos e um baixo custo de implantação. Tudo isso ao custo de uma baixa taxa de transferência, na casa de dezenas de kbps(*kilobits per second*, do inglês, kilobits por segundo), e uma alta latência, tipicamente segundos ou até mesmo minutos(RAZA; KULKARNI; SOORIYABANDARA, 2017). Estes aspectos negativos podem limitar os casos de uso das tecnologias LPWAN, como monitoramento por vídeo, que necessitam de centenas de kbps e automações industriais ou monitoramentos críticos de infraestrutura que requerem operação em tempo real(ADELANTADO *et al.*, 2017). Ainda assim, LPWANs atingem dezenas

de aplicações para cidades inteligentes, automação residencial, eletrônicos vestíveis, logística e monitoramento ambiental que requerem baixa troca de informações e em frequência limitada(RAZA; KULKARNI; SOORIYABANDARA, 2017).

2.7.2 Características de longo alcance

Dispositivos LPWAN buscam uma área expandida de cobertura na escala de quilômetros, seja em ambientes urbanos ou rurais. Desta forma, eles foram desenhados para possuir uma excelente propagação de sinal nos mais remotos lugares. A seguir, são discutidas as formas como as LPWAN exploram esse objetivo.

2.7.2.1 Uso de banda sub-1GHz

Com a exceção de algumas tecnologias LPWAN, como, por exemplo a Ingenu(INGENU, 2024) e a WEIGTHLESS-W(OPENWEIGHTLESS, 2024), a maioria delas utiliza-se da banda abaixo de 1GHz. Segundo Raza, Kulkarni e Sooriyabandara(2017), existem três motivos principais para isto: primeiro, pode-se obter uma comunicação robusta e confiável a um baixo custo energético; segundo, quando comparado a banda de 2.4GHz, os sinais na banda sub-GHz possuem menor atenuação e sofrem menos problemas por multi-percurso; e por fim, em terceiro, a banda abaixo de 1 GHz é muito menos congestionada que a de 2.4GHz, que é disputada por tecnologias como *Bluetooth* e *Wi-fi*.

2.7.2.2 Técnicas de modulação

De acordo com Raza, Kulkarni e Sooriyabandara(2017), as tecnologias LPWAN conseguem alcançar a grande área de cobertura visada sacrificando altas taxas de transmissão e reduzindo as velocidades de transmissão, colocando mais energia no *bit* ou símbolo transmitido. Desta forma, os receptores podem decodificar mensagens bastante atenuadas corretamente, chegando a uma sensibilidade de até -130 dBm. As LPWAN basicamente adotaram duas classes diferentes de técnicas de modulação: técnicas de espalhamento espectral(do inglês, *Spread-Spectrum techniques*) e técnicas de banda estreita(do inglês, *Narrowband techniques*).

2.7.2.2.1 Técnicas de Espalhamento Espectral

Segundo Scholtz(1982), as técnicas de espalhamento espectral referem-se a um sistema que segue três condições: a portadora é um sinal de banda larga imprevisível(ou pseudo-aleatório); a largura de banda da portadora é muito maior que a da informação modulada; a

recepção é obtida por correlação cruzada do sinal de banda larga recebido com uma réplica gerada de forma síncrona da banda larga da portadora.

Os sistemas que se utilizam de técnicas de espalhamento espectral, devido as suas características intrínsecas, possuem ao menos quatro características importantes(SCHOLTZ, 1982):

1. **Baixa probabilidade de interceptação**: alto ganho no processamento e sinais de portadora imprevisíveis quando a potência é distribuída fina e uniformemente na frequência diminuem a probabilidade de interceptação, fazendo com que a detecção do sinal sobre o ruído seja complexa.

2. ***Antijam***: os sinais variáveis de portadora fazem com que um atacante não possa melhorar sua atuação através de repetidas observações do sinais, dependendo de outros meios para atacar.

3. **Pares de transmissor/receptor independentes**: se pares de transmissores e receptores diferentes usarem portadoras randômicas, eles podem utilizar a mesma largura de banda com baixa interferência entre si. Este sistema é conhecido como CDMA(*Code Division Multiple Access*, do inglês, Acesso Múltiplo por Divisão de Código).

4. **Capacidade criptográfica**: a portadora aleatória age como uma chave em um sistema de cifra, fazendo com que terceiros não possam identificar a diferença entre o que é informação e o que é portadora na modulação.

2.7.2.2.2 Técnicas de Banda Estreita

As técnicas de Banda Estreita são assim chamadas por utilizarem uma largura de banda de no máximo 25 kHz. Desta forma, todo o espectro pode ser compartilhado de forma eficiente, com cada sinal da portadora ocupando um pequeno espaço com baixíssimo ruído(PATEL, 2018). Existem, ainda, tecnologias LPWAN, como o Sigfox(SIGFOX, 2024), que se utilizam da *Ultra Narrow Band*(UNB), que limita a largura de banda dos sinais em 100 Hz, reduzindo ainda mais o efeito do ruído nas portadoras e aumentando a quantidade de dispositivos que podem compartilhar o espectro. Porém, a UNB restringe ainda mais a quantidade de informação que pode ser transmitida em cada canal(PATEL, 2018).

2.7.3 Características de baixo consumo

Ao projetar uma rede de comunicação, um dos fatores mais importantes a se considerar é o seu consumo energético, que afeta os custos de operação, de manutenção e confiabilidade da rede. No que se refere a redes com sensores, estes podem estar alocados em ambientes

hostis ou de difícil acesso, fazendo com que exiga-se o tempo de vida dos dispositivos seja da ordem de vários meses ou até mesmo anos(ANASTASI *et al.*, 2009).

Abaixo encontram-se algumas características que demonstram como as LPWANs obtém o baixo consumo.

2.7.3.1 *Duty-cycling*

Segundo Anastasi *et al.*(2009), medições experimentais demonstram que a transmissão de dados possui um custo energético bastante elevado quando comparado ao processamento de dados. Assim, o periférico de transmissão de dados deveria ser desligado assim que toda a informação foi transmitida, sendo acionado novamente somente quando existirem novas informações a serem transmitidas. Este comportamento é chamado de *Duty-cycling*, ao passo que *duty-cycle* é definido como o período de tempo em que os dispositivos estão ativos durante seu tempo de vida.

É importante lembrar que não só o transmissor, mas outros periféricos dos dispositivos, como o processador, consomem energia mesmo quando não estão sendo utilizados. Desta forma, existem duas abordagens diferentes e complementares para o *Duty-cycling*: primeiro, com o controle da topologia, pode-se explorar a redundância da rede, desativando os nós que não são necessários para manter a conectividade e economizar energia; segundo, com gerenciamento de energia, os nós ativos não precisam estar com os rádios sempre ligados, desativando-os quando não há atividade na rede(ANASTASI *et al.*, 2009).

2.7.3.2 Técnicas de MAC(*Medium Access Control*)

De acordo com Raza, Kulkarni e Sooriyabandara(2017), os protocolos MAC mais utilizados em redes celulares ou ainda em redes sem fio possuem um cabeçalho de controle muito extenso para as LPWANs. Estes protocolos são bastante dispendiosos para os dispositivos de baixo custo das redes LPWAN, que geralmente possuem uma comunicação bem menos frequente. Por esse motivo, as redes LPWAN optam por sistemas de acesso randomizados, como o ALOHA e o CSMA/CA(*Carrier Sense Multiple Access with Collision Avoidance*).

2.7.3.3 Complexidade dos dispositivos

Uma outra forma como as tecnologias diminuem seu custo operacional energético é retirando complexidade dos nós da rede, deixando as tarefas mais complexas para *gateways* ou para um servidor externo. Por exemplo, diferentes aplicações IoT não demandam um tráfego de dados constante, com sensores precisando se comunicar poucas vezes ao dia, reduzindo o custo de *hardware* dos nós e deixando o processamento para estações-base, que são menores em quantidade.

2.7.4 Características de baixo custo

Como as distâncias de transmissão costumam ser diversas vezes maiores que as tecnologias convencionais, diminui-se o custo de construção da infraestrutura, já que as tecnologias LPWAN não dependem de uma grande densidade de dispositivos.

Destaca-se também a redução da complexidade do *hardware* e da demanda por memória(KUHLINS *et al.*, 2020), o que reduz o custo por unidade de cada ponto da rede, podendo ser ainda mais reduzido com produção em larga escala. Outro ponto relevante é o uso de bandas não licenciadas, como é o caso do LoRa(Semtech, 2015) e do SigFox(SIGFOX, 2024), ou ainda o uso de bandas já licenciadas baseadas em redes celulares, como é o caso do NB-IoT(GSMA, 2024), de forma a diminuir ainda mais o custo de implantação de um rede LPWAN.

2.8 LoRa

Uma das tecnologias LPWAN mais conceituadas é o LoRa(*Long Range*, do inglês, Longa Distância), uma técnica de modulação proprietária da Semtech(CORPORATION, 2024). Neste tópico serão salientados alguns pontos importantes em relação à tecnologia.

2.8.1 Fundamentos

A LoRa é uma técnica de modulação de espalhamento espectral derivada da modulação CSS(*Chirp Spread Spectrum*), uma tecnologia desenvolvida nos anos 40 em radares. Utilizando-se de fatores de espalhamento ortogonais, a modulação LoRa implementa uma taxa de transmissão variável, que permite a troca de taxa de transmissão por alcance ou potência, de forma a aprimorar a comunicação em um canal com largura de banda fixa. Ela compreende a camada física no modelo OSI, sendo completamente indiferente às implementações das camadas superiores. Ou seja, ela consegue conviver e interagir com outas tecnologias já existentes(Semtech, 2015).

O espalhamento do espectro na modulação LoRa é obtido através de um sinal *chirp* variável em frequência. Desta forma, a complexidade do receptor é reduzida, uma vez que este método faz com que o deslocamento temporal e em frequência entre o transmissor e receptor sejam equivalentes(Semtech, 2015). A troca de potência ou cobertura por taxa de transmissão em uma mesma largura de banda na modulação LoRa deve-se especificamente por três parâmetros: o parâmetro SF(*Spreading Factor*), a largura de banda BW(*Bandwidth*) e o parâmetro CR(*Code Rate*). Com estes três parâmetros é possível definir a taxa de *bits* R_b, dada pela expressão:

$$R_b = SF * \frac{\frac{4}{4+CR}}{[\frac{2^{SF}}{BW}]} \tag{1}$$

O *Spreading Factor* SF é um parâmetro cujo valor varia de 7 a 12, e significa o número de *bits* modulados em um sinal *chirp*. Uma mensagem enviada com um SF maior possui uma maior latência e reduz a taxa de transmissão, mas melhora a resistência ao ruído(SUNDARAM; DU; ZHAO, 2019). Além disso, a modulação LoRa também inclui um código corretor de erro, melhorando a robustez do sinal transmitido ao custo de adicionar redundância de *bits* no sinal. A taxa de correção é dada pela expressão do denominador, $\frac{4}{4+CR}$, onde CR pode assumir os valores de 1 a 4. Por fim, a largura de banda do sinal BW pode assumir os valores de 125kHz, 250kHz e 500kHz.

2.8.2 Propriedades do LoRa

A tecnologia LoRa é uma das mais proeminentes no campo das LPWAN, isto devido a uma série de características de robustez e baixo consumo, que vão ao encontro das necessidades esclarecidas pelas LPWAN. Em relação ao alcance da comunicação, a LoRa consegue obter resultados na casa dos milhares de metros, com ou sem linha de visada. Segundo Sundaram, Du e Zhao(2019), com linha de visada, a comunicação LoRa entre dois pontos pode chegar até 9 quilômetros de distância usando um SF de valor 12, ao passo que um SF de valor 7 chega a 5 quilômetros de distância, ambos com PRR(*Packet Reception Ratio*) maior que 70%. Em cenários sem linha de visada, as distâncias se limitam a 2 quilômetros.

Pode-se citar, ainda, que a distância e qualidade de comunicação são bastante afetados pelos três parâmetros(SF, BW e CR) citados anteriormente. Segundo Angrisani *et al.*(2017), a alteração da largura de banda BW tem efeitos rigorosos nas perdas de pacotes e, aumentando-se o valor de SF, obtém-se uma comunicação muito mais robusta. Além disso, o valor de CR pode melhorar consideravelmente a qualidade da comunicação, aumentando ainda mais as distâncias que podem ser obtidas através da comunicação LoRa.

Para atingir certa qualidade de comunicação à grandes distâncias e com uma baixa potência, o sinal LoRa deve ser bastante resistente aos mais diversos tipos de ruídos. Devido à alta duração do sinal e sua largura espectral, além da natureza assíncrona, o sinal LoRa é resistente a diversos tipos de mecanismos de interferência, como por exemplo(Semtech, 2015):

- **Multipercurso/Desvanecimento**: pelo pulso *chirp* possuir banda relativamente larga, a LoRa oferece resistência contra os efeitos de multipercurso e desvanecimento, tornando-a excelente para áreas urbanas

- **Efeito Doppler**: o efeito Doppler causa um leve deslocamento de frequência no pulso LoRa, introduzindo um desvio temporal desprezível no sinal de banda base. Esta tolerância reduz o requisito de *clocks* altamente precisos.

- **Interferência interna da rede**: a modulação Lora aplica *spreading factors* ortogonais, fazendo com que seja possível transmitir diferentes sinais com SF diferentes no mesmo

canal ao mesmo tempo. Os sinais com SF diferentes são tratados como ruído pelos receptores LoRa.

Um dos fatores que explicam o destaque da LoRa entre as tecnologias LPWAN é o seu baixo consumo energético. A transmissão LoRa consome apenas de 120 a 150mW de potência e de 10 a 15mW de potência para operações do microcontrolador. Variando o *duty-cycle* entre 0,1% a 10%, isto pode levar a duração da bateria dos dispositivos durar de 2 a 5 anos(SUNDARAM; DU; ZHAO, 2019).

2.9 Bluetooth *Low Energy*

Bluetooth se refere a um conjunto de protocolos de comunicação sem fio de curta distância. Sua especifição se divide em duas grandes partes(CÄSAR *et al.*, 2022): o chamado Bluetooth *basic rate/enchanced data rate* (*BR/EDR*), também conhecido com Bluetooth *Legacy* e sua versão de baixo consumo, o Bluetooth *Low Energy*(BLE). Embora possuam similaridades, as duas tecnologias não são compatíveis. A seguir, serão discutidos alguns pontos relevantes referente a tecnologia BLE.

2.9.1 Fundamentos

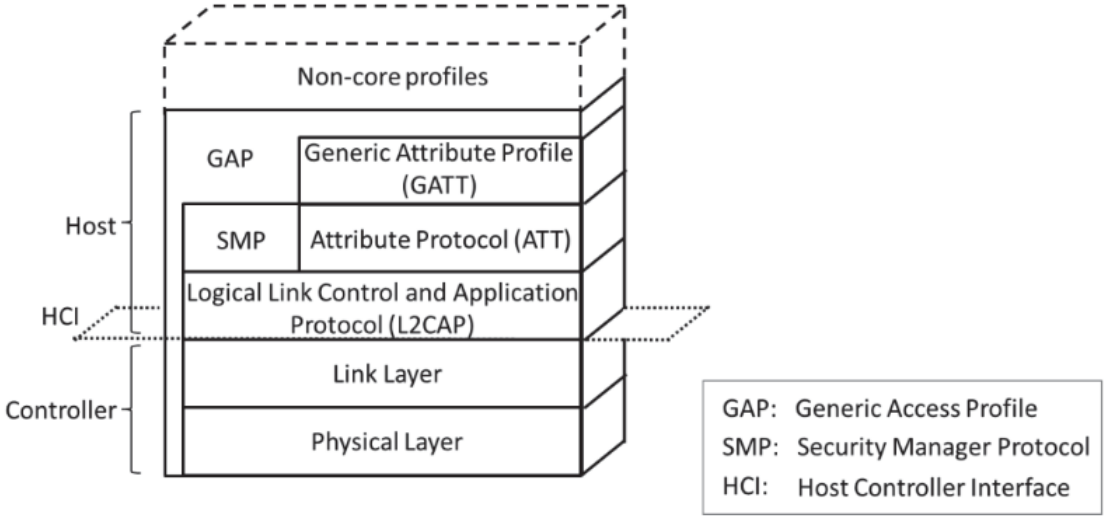
BLE é uma tecnologia de comunicação sem-fio de curta distância desenvolvida pela Bluetooth Special Interest Group(SIG), concebida como uma solução de baixo consumo para aplicações de monitoramento e controle(GOMEZ; OLLER; PARADELLS, 2012). Segundo Heydon (2013), enquanto o Bluetooth *Legacy* focava em obter taxas de transferência mais altas a cada geração, o protocolo BLE buscava economizar em custo energético e econômico tanto quanto possível. A tecnologia tecnologia BLE opera na banda ISM (*Industrial, Science, Medical*, do inglês, Indústria, Ciência e Medicina) de 2,4GHz a 2,48GHz. Esta faixa apesar de possuir limitações legais de potência, não exige licença para operar, sendo uma das poucas faixas de frequência não licenciadas globalmente. Baseado no Bluetooth BR/EDR, o BLE foi concebido para simplificar tanto a implementação do *hardware* quanto o protocolo do Bluetooth, maximizando sua eficiência energética.

2.9.2 Visão Geral dos Protocolos BLE

A pilha de protocolos BLE está ilustrada na Figura 5. Como pode-se observar, ela é formada por duas partes principais: o Controlador, que compreende as camadas física e de enlace, e o *Host*, que compreende as camadas superiores. De acordo com Gomez, Oller e Paradells (2012), o Controlador é comumente implementado em um pequeno *System-on-Chip*(SOC, do inglês, Sistema em um Chip) com um rádio integrado, ao passo que o *Host* é executado em pro-

cessador de aplicações como: *Logical Link Control and Adaptation Protocol*(L2CAP, do inglês, Controle de Ligação Lógica e Protocolo Adaptativo); *Atributte Protocol*(ATT, do inglês, Protocolo de Atributo); *Generic Attribute Profile*(GATT, do inglês, Perfil de Atributo Genérico); *Security Manager Protocol*(SMP, do inglês, Protocolo de Gerenciamento de Segurança); e, por fim, o *Generic Access Profile*(GAP, do ingles, Perfil de Acesso Genérico). Todas essas funcionalidades serão explanadas de forma sucinta neste capítulo. Além disso, pode-se citar o HCI(*Host Controller Interface*), que padroniza a comunicação entre o *Host* e o Controlador, e também as funcionalidades que não são definidas pela especificação do Bluetooth, mas podem ser utilizadas sob o *Host*(representadas por "*Non-core profiles*" na Figura 5).

Figura 5 – Pilha de Procolos BLE



Fonte: Gomez, Oller e Paradells (2012).

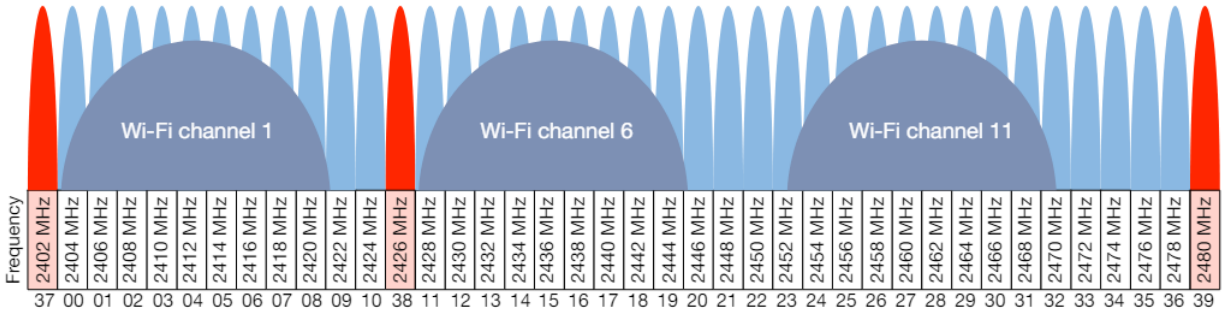
2.9.2.1 Camada Física e Camada de Enlace de Dados

A tecnologia BLE se utiliza da modulação GFSK(*Gaussian Frequency Shift Keying*, do inglês, Chaveamento de Mudança de Frequência Gaussiana) para transmitir e receber dados. Em termos simplificados, a modulação GFSK altera a frequência de transmissão do rádio para cima e para baixo para transmitir *bits* codificados e, como os pulsos de energia podem se espalhar pelo espectro de frequência, um filtro na forma da curva de Gauss é utilizado. Ademais, para coexistir com vários rádios na mesma área, o BLE divide a banda de 2.4GHz em 40 canais diferentes, com espaçamento espectral de 2MHz entre eles(HEYDON, 2013).

Conforme Gomez, Oller e Paradells (2012), existem 3 canais de divulgação, utilizados para descobrimento de dispositivos, estabelecimento de conexão e *broadcast*. Como pode ser observado na Figura 6, estes canais foram escolhidos para a não interferir com os canais 1, 6 e 11 do Wi-Fi, minimizando a sobreposição das transmissões. Os outros 37 canais são canais de dados, utilizados para comunicação bidirecional entre dois dispositivos conectados. Enfim, a

transmissão de dados pela camada física do BLE pode chegar a taxas de 1 Mbps e, em certas condições, possui área de cobertura na casa das dezenas de metros.

Figura 6 – Distribuição dos canais BLE



Fonte: Nikoukar *et al.* (2018).

Nas versões mais recentes da especificação do BLE, Bluetooth 5.0, existem algumas melhorias: uma nova maneira de se utilizar a camada física, denominada *LE 2M PHY*, utiliza-se de uma taxa de símbolo equivalente a 2 megabits por segundo; também uma nova função na camada física, denominada *LE Coded PHY*, se utiliza de uma técnica de correção de erro(FEC, do inglês, *Forward Error Correction*) para aumentar a distância de comunicação em até quatro vezes sem aumentar a potência de transmissão; por fim, existe ainda a possibilidade de se utilizar todos os 40 canais como canais de divulgação, para uma melhor eficiência espectral(WOOLLEY, 2021).

Segundo Heydon (2013), a camada de enlace é responsável pela divulgação, escaneamento, criação e preservação das conexões, assegurando que todos os pacotes estejam corretamente estruturados. Existem dois tipos de canais na camada de enlace: canais de divulgação, usado por dispositivos que não estão conectados, e canais de dados, usados pelos dispositivos durante o período de conexão. Para enviar dados nestes canais, são definidos pequenos pacotes que encapsulam os dados junto a informações como o endereçamento e um *checksum*.

2.9.2.2 *Logical Link Control and Adaptation Protocol*(L2CAP)

A camada do *Logical Link Control and Adaptation Protocol*(L2CAP) é a camada de multiplexação do BLE. Existem três canais bidirecionais fixos: um para o canal de sinalização, um para o *Security Manager Protocol*(SMP) e um para o *Atributte Protocol*(ATT)(HEYDON, 2013). O protocolo para o BLE é uma versão simplificada e otimizada do L2CAP do Bluetooth *Legacy*:as informações dos três canais são tratadas de forma a não usar retransmissão, mecanismos de controle de fluxo e segmentação - uma vez que o tamanho das mensagens são limitadas a um máximo de 23 bytes - (GOMEZ; OLLER; PARADELLS, 2012).

2.9.2.3 *Generic Access Profile*(GAP)

No topo da pilha de protocolo está a camada do *Generic Access Profile*(GAP), responsável por definir os papéis do dispositivo BLE, bem como os modos e procedimentos para o descobrimento de outros dispositivos e serviços.A seguir, serão descritos os quatro papéis que um dispositivo BLE pode desempenhar, de acordo com Gomez, Oller e Paradells (2012):

- ***Broadcaster***: neste papel, o dispositivo não suporta conexões com nenhum outro, apenas emite dados nos canais de divulgação.
- **Observador**: servindo como complemento ao papel de *broadcaster*, o papel do dispositivo BLE observador é de apenas receber dados, sem nunca estabelecer conexões com outros dispositivos.
- **Central**: o papel de central está relacionado ao dispositivo BLE incumbido da função de escanear outros dispositivos e iniciar o processo de conexão. Um dispositivos central pode gerenciar múltiplas conexões ao mesmo tempo.
- **Periférico**: o papel de um dispositivo periférico é a de se conectar com um dispositivo central para uma comunicação bidirecional. Quando o dispositivo BLE desempenha esta função, ele aceita apenas uma conexão.

Ainda de acordo com (GOMEZ; OLLER; PARADELLS, 2012), é importante notar que, quando é estabelecida uma conexão entre um dispositivo central e um periférico, é necessário que o controlador suporte a interação no modelo ”cliente/servidor”. Além disso, um dispositivo BLE pode suportar várias funções, mas apenas uma delas pode estar ativa em dado momento.

2.9.2.4 *Atributte Protocol*(ATT)

Segundo Heydon (2013), a camada *Atributte Protocol*(ATT) define as bases das regras para acesso aos dados de um dado dispositivo. Dados dois dispositivos, um agindo como servidor e outro como cliente, ambos podem iniciar a comunicação, exigindo uma resposta ou não. Por exemplo, o dispositivo agindo como cliente manda requisições ao dispositivo agindo como servidor, que devidamente as responde. Assim, o cliente utiliza as requisições para ler e escrever os ”atributos” do servidor. ”Atributos” são estruturas de dados que possuem: um identificador exclusivo; um tipo, que identificada a informação armazenadas pelo atributo; e um valor, que é o dado efetivamente. Por exemplo, um atributo do tipo ”Temperatura” pode possuir o valor ”20.5°C” no identificador ”0x01CE”.

As funções de cliente e servidor são independentes dos papéis de dispositivo central e dispositivo periférico definidos na camada GAP. Desta forma, um dispositivo periférico pode agir como cliente ou servidor, a depender apenas da aplicação(Nordic Semiconductor, 2024).

A camada ATT também define que alguns atributos possuem permissões, permitindo que um cliente leia ou escreva no valor do atributo. Não é possível descobrir de forma explícita as permissões de um atributo mas sim apenas implicitamente, mandando uma requisição e recebendo um erro de reposta, indicando que a requisição não foi realizada (HEYDON, 2013).

2.9.2.5 *Generic Attribute Profile*(GATT)

A camada do *Generic Attribute Profile*(GATT) fica logo acima da camada ATT, definindo os tipos de atributos e como eles são utilizados(HEYDON, 2013). Segundo Gomez, Oller e Paradells (2012), o GATT define uma estrutura para a descoberta de ”serviços” e a troca de ”características” de um dispositivo a outro. Para entender o conceito de ”serviços” e ”características”, pode ser utilizado um exemplo em conjunto com a Figura 7.

Assumindo o exemplo de um dispositivo BLE cuja função principal seja a de medir a temperatura de um ambiente. O valor físico da temperatura, por exemplo ”20.5°C”, poderia ter a grandeza salva como um atributo denominado ”Valor”, cujo valor seria ”20.5”, e a unidade de medida como um atributo denominado ”Unidade”, cujo valor seria ”°C”. Os dois atributos juntos formam a chamada ”característica”. Neste caso, a característica poderia ser ”Medida Atual de Temperatura”.

Todas as características são encapsuladas em um ”serviço”. ”Serviços” contém inúmeras características. Neste exemplo, poderia haver um serviço denominado ”Medição de Temperatura”, que conteria a característica ”Medida Atual de Temperatura”, bem como outras características, como ”Localização do Termômetro”.

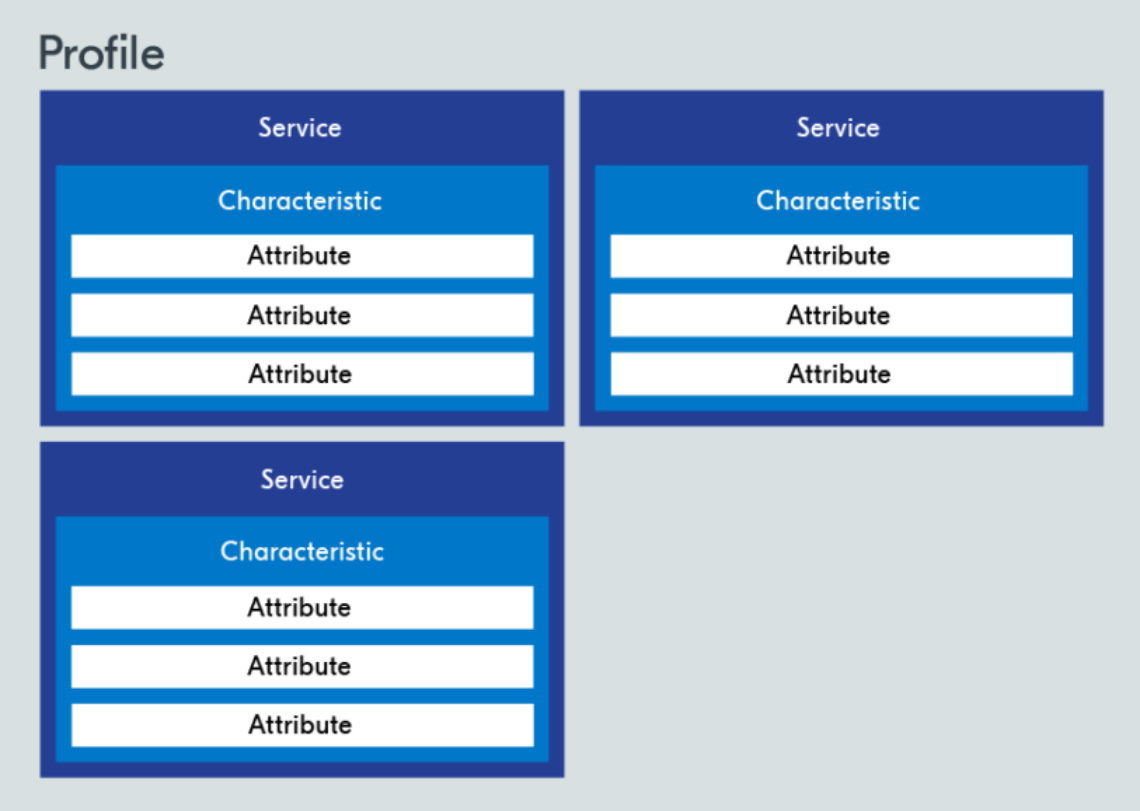
Encapsulando tudo isto, existe um ”perfil”, que contém um ou mais serviços da aplicação. O perfil ”Medir Tempratura do Quarto” possui o serviço ”Medição de Temperatura”, mas pode conter também o serviço ”Informações do Termômetro”. Este, por sua vez, conteria características como ”*Firmware*” e ”Fabricante”.

2.9.2.6 *Security Manager Protocol*(SMP)

De acordo com Gomez, Oller e Paradells (2012), a camada do *Security Manager Protocol*(SMP) é responsável pela troca de mensagens durante as três fases do pareamento entre dispositivos, quando utilizados os recursos de segurança do BLE. O pareamento é um procedimento em que os dispositivos geram e distribuem as chaves de segurança:

- **Primeira Fase**:os dispositivos anunciam suas capacidades de *input/output* e, baseados nisso, escolhem o método de troca de chaves para a próxima fase
- **Segunda Fase**:a segunda fase tem como objetivo gerar a *Short-Term Key*(STK, do inglês, Chave de Curto Prazo), usada na terceira fase para assegurar a distribuição de chaves

Figura 7 – Estrutura do GATT



Fonte: Nordic Semiconductor (2024).

- **Terceira Fase**: a terceira fase é responsável por permutar a *Long-Term Key*(LTK, do inglês, Chave de Longo Prazo), usada para encriptação e autenticação na camada de enlace, a *Connection Signature Resolving Key*(CSRK, do inglês, Chave de Resolução da Assinatura de Conexão), utilizada para assinatura de dados no protocolo ATT e, por fim, a *Identity Resolving Key*(IRK, do inglês, Chave de Resolução de Identidade), usada para gerar um endereço privado na base do endereço público do dispositivo.

A maioria dos recursos de segurança podem ser expressados em via de dois modos de segurança: *LE Security Mode 1* e *LE Security Mode 2*. Estes recursos são mutuamente exclusivos e adicionam funcionalidades de segurança para a camada de enlace e para a camada ATT, respectivamente. O BLE também possui uma funcionalidade de privacidade, que permite que o dispositivo possua um endereço privado e o altere frequentemente, mitigando a possibilidade de um terceiro rastrear o dispositivo(GOMEZ; OLLER; PARADELLS, 2012).

2.10 Modelos de Propagação

Os sistemas de comunicação sem fio trafegam as informações por um meio imprevisível: o transmissor e o receptor podem estar um à vista do outro(caso chamado de ”linha de visada”) ou podem haver diversos obstáculos entre eles, como prédios, árvores ou mesmo pessoas

transitando, o que afeta o sinal recebido no receptor. Existem três mecanismos de propagação que afetam um sistema de comunicação sem fio(RAPPAPORT, 2002):

- **Reflexão**: uma onda eletromagnética reflete em objetos cuja dimensão seja maior que o seu comprimento de onda, como paredes e o chão.
- **Difração**: ocorre quando os obstáculos entre o transmissor e o receptor possuem formatos irregulares, fazendo com que a onda se flexione por volta do obstáculo.
- **Dispersão**: dispersão ocorre quando o meio em que a onda trafega consiste de objetos cuja dimensão é pequena em relação ao comprimento de onda, ou quando há uma grande densidade de obstáculos. As ondas eletromagnéticas então dispersam e podem chegar ao receptor por múltiplos caminhos.

Dada esta característica randômica do meio de propagação dos sinais em um sistema de comunicações sem fio, vários modelos matemáticos foram desenvolvidos de forma a tentar prever o nível de sinal recebido no receptor à uma certa distância do transmissor.

2.10.1 Modelo de Propagação no Espaço Livre

De acordo com Rappaport (2002), o Modelo de Propagação no Espaço Livre é utilizado para antever o nível do sinal recebido quando o transmissor e o receptor possuem linha de visada, ou seja, não há nenhum obstáculo entre eles. O valor da potência P_r recebida pela antena de um receptor a uma distância d do transmissor é dada pela equação de Friis:

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L} \tag{2}$$

onde P_t é a potência de transmissão, G_t é o ganho da antena de transmissão, G_r é o ganho da antena de recepção, λ é o comprimento de onda e L é um fator de perda do sistema, não relacionado à propagação($L >= 1$). As potência devem ser dadas em Watt, enquanto os ganhos das antenas e o fator L são adimensionais É importante lembrar que λ está relacionada com a frequência da transmissão:

$$\lambda = \frac{c}{f} = \frac{2\pi c}{\omega_c} \tag{3}$$

onde f é a frequência da portadora, em Hertz, ω_c é a frequência da portadora em radianos por segundo e c é a velocidade da luz, em metros por segundo.

Um parâmetro bastante utilizado na análise de sistemas de comunicação sem fio é o denominado ”Perda de Percurso”. Segundo Rappaport (2002), perda de percurso é definida como a diferença, em decibéis(dB), entre a potência transmitida e a potência recebida de um

sinal. A perda de percurso PL é dada por:

$$PL(dB) = 10log\frac{P_t}{P_r} = -10log[\frac{G_t G_r \lambda^2}{(4\pi^2)d^2}] \quad (4)$$

Por fim, vale ressaltar que a equação de Friis para o modelo de espaço livre só é válida para valores de d que estão na chamada região de campo distante, uma região além de uma distância d_f da antena transmissora(RAPPAPORT, 2002). Esta distância está relacionada às dimensões físicas da antena transmissora, e é dada por:

$$d_f = \frac{2D^2}{\lambda} \quad (5)$$

onde D é a maior dimensão linear da antena, em metros. De forma complementar, para estar na região de campo distante, $d_f >> D$ e $d_f >> \lambda$.

2.10.2 Modelo de Perda de Percurso Log-Distância

De acordo com Rappaport (2002), tanto os modelos teóricos como empíricos indicam que a média da potência do sinal recebidos decaem de forma logarítmica com a distância. Assim, o modelo Log-Distância define a perda de percurso $\overline{PL}$ pode ser expressa como:

$$\overline{PL}(dB) = \overline{PL}(d_0) + 10nlog(\frac{d}{d_0}) \quad (6)$$

onde n é um expoente de perdas que indica a taxa em que a perda de percurso ocorre com o aumento da distância, d_0 é uma referência de distância determinada por medições próximas ao transmissor e d é a distância entre o receptor e o transmissor. O valor de n depende do ambiente sendo avaliado, cujos exemplos de valores podem ser observado no Quadro 1.

Quadro 1 – Exemplos de expoentes n para o modelo Log-Distância

Ambiente	**Expoente n**
Espaço Livre	2
Área urbana	2,7 a 3,5
Área urbana sombreada	3 a 5
Linha de visada em ambiente interno	1,6 a 1,8
Obstruído em prédios	4 a 6
Obstruído em fábricas	2 a 3

Fonte: Rappaport (2002).

3 ANÁLISE CRÍTICA DE TRABALHO ANTERIOR

O presente Trabalho de Conclusão de Curso teve como ponto de partida um trabalho anterior, denominado ”Medição e Controle de Sistemas de Potência”, desenvolvido para a disciplina de Redes Avançadas do curso de Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná. A disciplina, ministrada pelo Professor Doutor Hermes Irineu Del Monego, tinha como objetivo final a apresentação de um sistema de comunicação completo, que abrangesse todas as camadas do modelo OSI. Neste capítulo serão brevemente explicadas as ideias iniciais e os resultados obtidos, com um grande enfoque em suas limitações, uma vez que estas seriam metas de melhoria para a rede Hydra.

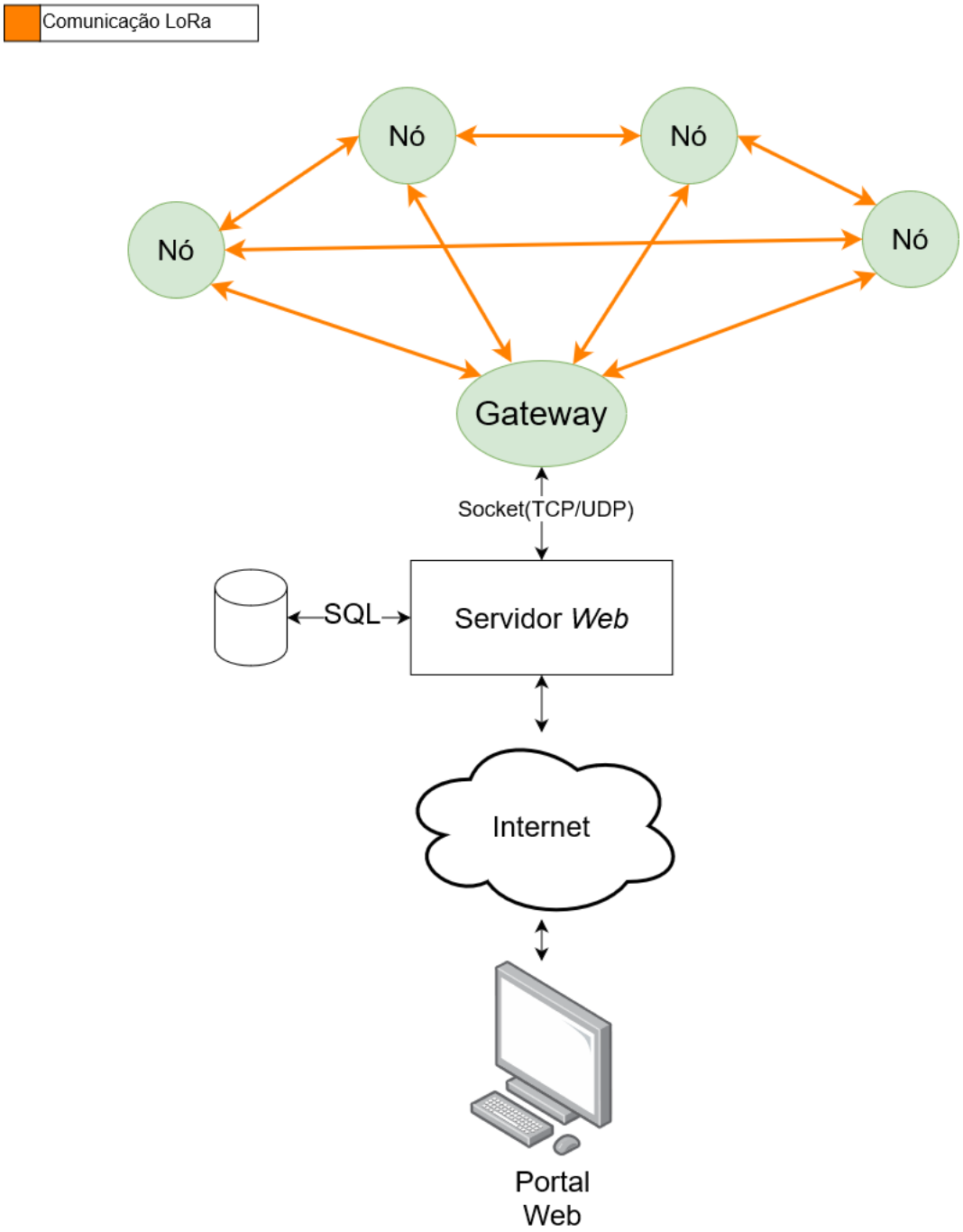
3.1 Conceito Inicial

O objetivo do trabalho era criar uma rede de dispositivos capazes de, individualmente, medir a potência e o consumo energético em uma tomada e que, através de comunicação LoRa, pudessem enviar estes dados para uma interface *Web* para controle e manutenção de um histórico destas informações.

A Figura 8 mostra a solução proposta. Cada ”nó” seria um dispositivo composto por um microcontrolador, uma rádio LoRa, um relé, um medidor de tensão e um medidor de corrente. Assim, com uma entrada de tomada, o dispositivo seria capaz de medir a potência instantânea e o consumo de algum aparelho conectado a ele e, além disso, poderia ativar ou desativar sua alimentação comutando seu relé interno. Todos os nós seriam equipados com um rádio LoRa e, através de um algoritmo de roteamento em *mesh*, transmitiriam suas informações para um *gateway*.

O *gateway* seria um dispositivo diferente do nó. Ele possuiria um rádio LoRa para recepção das informações dos nós, mas também seria capaz de hospedar um servidor *web* para habilitar a criação de um aplicação. A aplicação teria como objetivo principal o controle todos os relés, mostrar a medição de cada nó e um histórico de medições.

Figura 8 – Arquitetura do sistema de Medição de Controle de Sistemas de Potência



Fonte: Autoria própria (2024).

3.2 Resultado

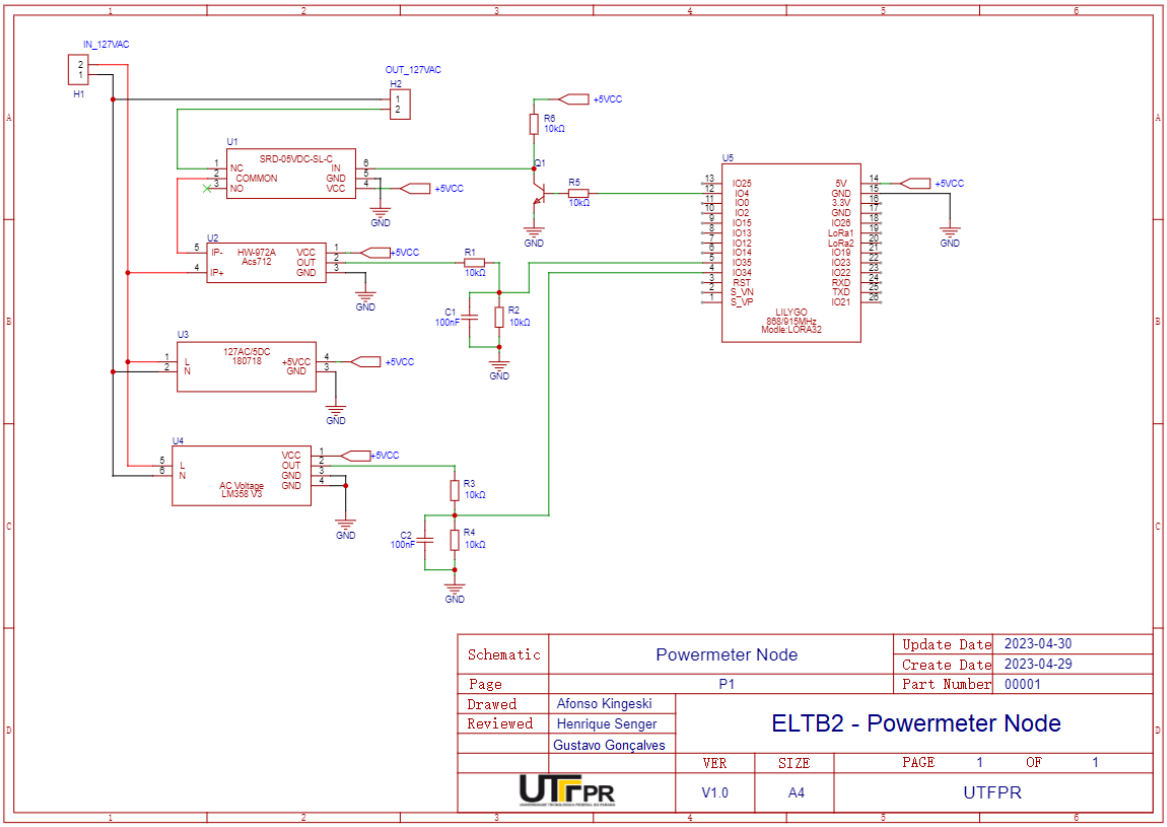
O esquemático final do nó pode ser observado na Figura 9. Para a medição de potência, foram utilizados módulos prontos: um sensor de corrente por efeito *hall* ACS712(ALLEGRO MICROSYSTEMS, 2007) e um módulo amplificador operacional LM358(TEXAS INSTRUMENTS, 2022) para a medição de tensão. O relé utilizado foi o SRD-05VDC0SL-C(SONGLE RELAY, 2007). Todo o sistema era alimentado por uma tomada de tensão nominal 127Volts, com um

transformador retificador que fazia a conversão da tensão alternada para 5Volts contínuos para a alimentação dos módulos.

Para facilitar o desenvolvimento, foram adquiridos módulos LILYGO® TTGO LoRa32 V2.1(LilyGO, 2024), que contém um microcontrolador ESP32 e um rádio LoRa integrado. Foi implementado um protocolo *mesh* denominado *flooding*. Sempre que um nó recebia uma mensagem, ele verificava se a mensagem era para ele próprio. Se sim, ele processava a mensagem. Caso contrário, ele incrementava um contador na mensagem e retransmitia. Isto acontecia até que o contador na mensagem atingisse certo limite.

Para o *gateway*, foi escolhido um Raspberry Pi 3(RASPBERRYPI LTD, 2023) para fazer a ponte entre a rede e a aplicação. O Raspberry Pi 3 foi conectado à um módulo LILYGO®, que recebia as informações dos nós e repassava ao Raspberry através de comunicação serial. Além disso, no Raspberry executava um MQTT *broker* e, desta forma, comunicava-se com a aplicação. Para a aplicação, foi desenvolvida uma interface *web* em que se podia consultar todos os nós, ligar e desligar os aparelhos a eles conectados e verificar o histórico de medições. O produto final desenvolvido pode ser observado na Figura 10.

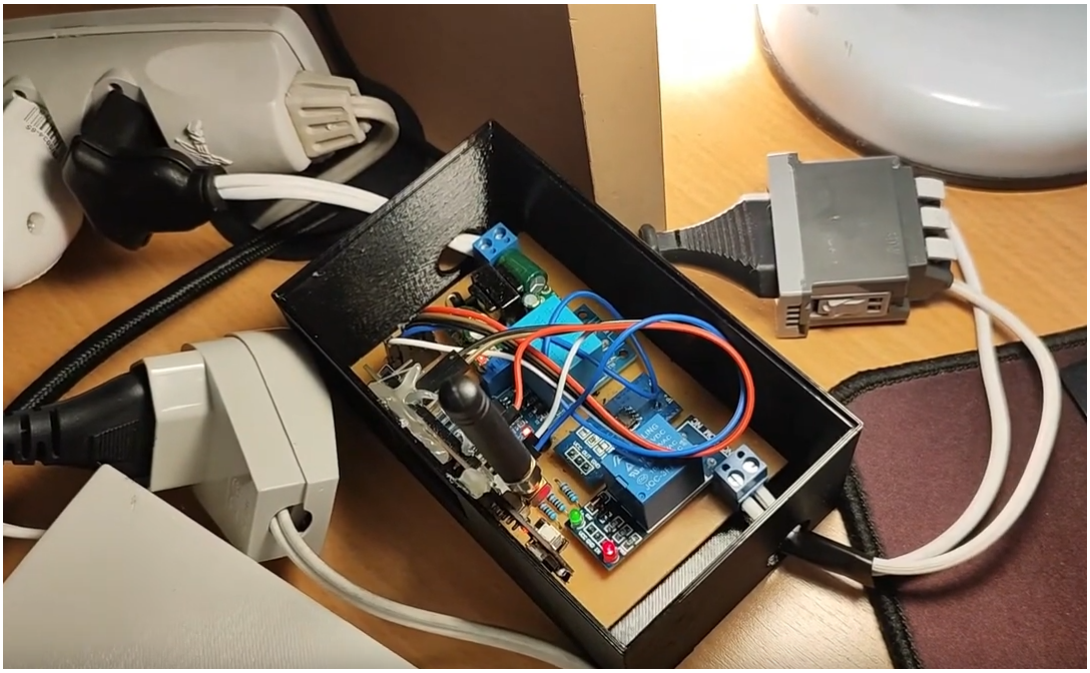
Figura 9 – Esquemático do nó



Fonte: Autoria própria (2024).

Alguns testes foram conduzidos para a validação do projeto. Como pode ser observado na Figura 11, a rede foi distribuída pela UTFPR Campus Curitiba, sede Centro. O *gateway* ficou no Bloco Q, enquanto um nó ficou no Bloco L e outro nó no Bloco J. Com os rádios LoRa

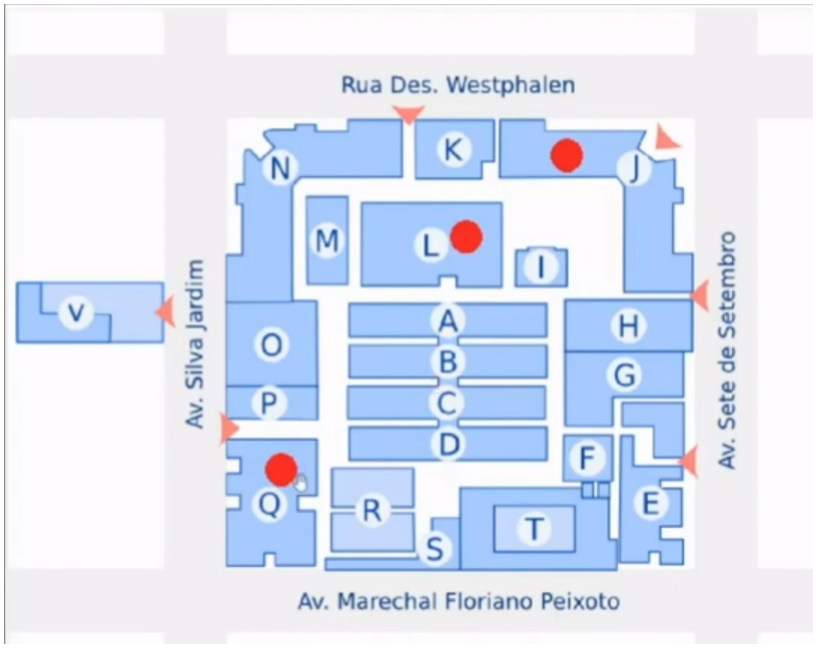
Figura 10 – Resultado final do Medidor



Fonte: Autoria própria (2024).

configurados para a mínima distância, o *gateway* não conseguia se comunicar diretamente com o nó do Bloco J. Entretanto, com o nó do Bloco L ativo e, portanto, retransmitindo pacotes, foi possível enviar uma mensagem do *gateway* para o nó do Bloco J.

Figura 11 – Teste de validação da rede



Fonte: Autoria própria (2024).

3.3 Limitações

Embora o projeto funcionasse para o que havia sido proposto, diversas limitações foram encontradas durante o desenvolvimento. Em primeiro lugar, o algoritmo de *flooding* é bastante simples e possui algumas desvantagens. Sem um roteamento bem definido, as mensagens acabam por sendo retransmitidas diversas vezes sem necessidade, gerando uma redundância desnecessária. A transmissão é o processo de maior consumo energético do rádio, logo o consumo total é aumentado. Além disso, enquanto transmitem, os nós não podem receber mensagens, logo, aumenta-se o congestionamento e o número de colisões na rede. Isto posto, um novo algoritmo *mesh* seria muito benéfico para a rede.

Em segundo lugar, o número de medidores do projeto é limitado pelo número de rádios LoRa disponíveis. A tecnologia LoRa, devido ao seu licenciamento, possui muito menos fornecedores que tecnologias mais difundidas, como o Bluetooth. Desta forma, os rádios LoRa limitam a quantidade de medidores devido ao seu custo mais elevado.

Outro problema é que a solução é totalmente fechada, com mudanças de aplicação sendo bastante difíceis e custosas. Por exemplo, caso houvesse a necessidade de acrescentar a medição de temperatura do ambiente, todo o projeto do nó deveria ser refeito do zero, refazendo a placa de circuito impresso e refatorando todo o código do microcontrolador.

Estes foram os principais problemas que serviram de pontapé inicial para a formulação da rede Hydra, cujo desenvolvimento será descrito nos próximos capítulos.

4 DESENVOLVIMENTO

4.1 Hydra

4.1.1 Fundamentos

Hydra é o nome do novo ecossistema IoT em *mesh* proposto pelo presente trabalho. As limitações do trabalho anterior, descrito no último capítulo, guiaram as decisões do projeto deste ecossistema. A Hydra se propõe a ser uma solução IoT genérica e integral. Genérica por que é completamente indiferente aos dados que trafegam por ela, podendo ser utilizadas nos mais diversos campos de aplicação: agricultura, indústrias, medicina e outros. E integral pois abrange desde os sensores até a interface com o usuário final de uma dada aplicação.

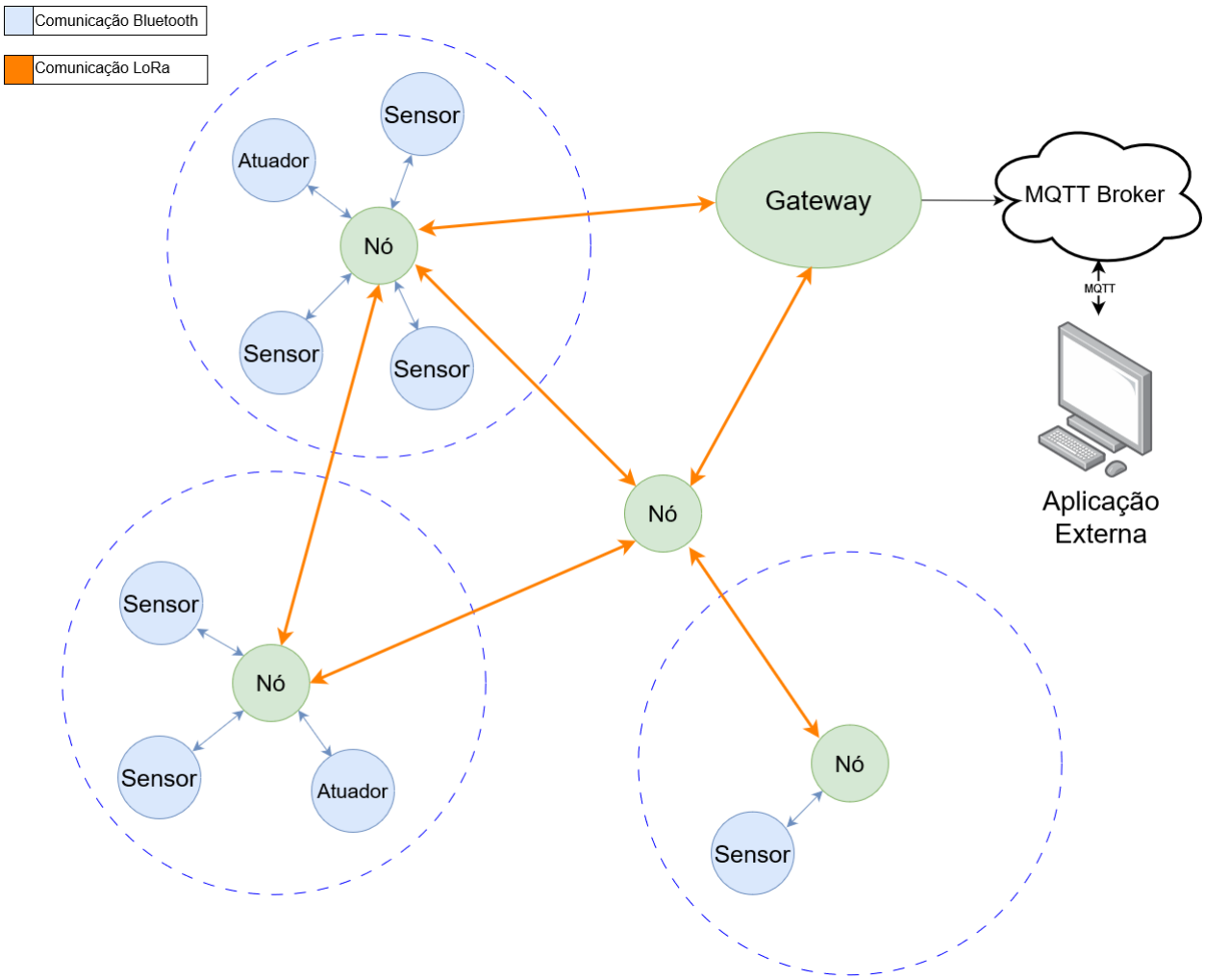
A Figura 12 mostra um exemplo do ecossistema Hydra. O *gateway* Hydra é o agente centralizador da rede, responsável por receber todas as informações da rede e enviá-las para a aplicação. Ele é munido de acesso a Internet e comunica-se com a aplicação através de um MQTT *broker*. É interessante mencionar que o ecossistema Hydra suporta múltiplos *gateways*.

Os nós Hydra são os pontos de acesso à rede Hydra. Eles se comunicam entre si através de um algoritmo *mesh* para que as informações cheguem ao *gateway*. Desta forma, um nó não precisa estar no alcance do *gateway* e sim no alcance de outro nó para integrar-se a rede. A comunicação é bidirecional, ou seja, tanto os nós podem transmitir informações para a aplicação quanto a aplicação pode transmitir dados aos nós.

Todos os nós são dotados de um rádio BLE, podendo se comunicar a múltiplos dispositivos BLE, como sensores e atuadores. O nó Hydra age como uma central BLE, procurando por dispositivos que anunciam um serviço dedicado e conectando-se a eles de tempos em tempos. No caso de sensores IoT, o nó utiliza o tempo conectado para angariar suas informações e, para o caso de atuadores, envia comandos de controle.

Existe apenas um *hardware* Hydra, que pode ser programado tanto como nó quanto como *gateway*. Ele foi projetado de forma a ser compacto e versátil, podendo ser apenas um agregador de dados de vários dispositivos, um ponto de entrada na rede ou ele mesmo um gerador de informação - uma vez que é possível acoplar sensores e atuadores ao nó.

Figura 12 – Exemplo do ecossistema Hydra



Fonte: Autoria própria (2024).

4.1.2 Protocolo *mesh*

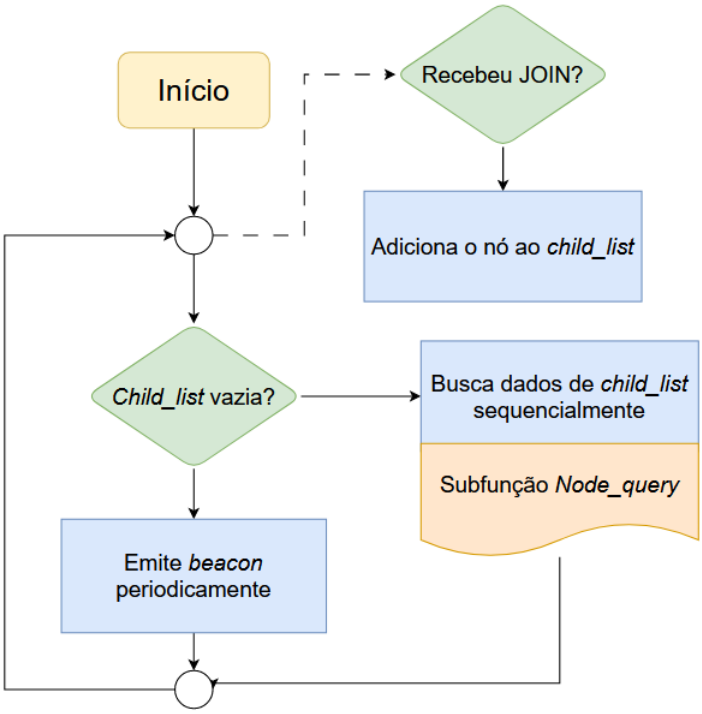
O protocolo *mesh* foi desenvolvido com base na implementação de Lee e Ke (2018), em ”*Monitoring of Large-Are IoT Sensors Using a LoRa Wireless Mesh Network System: Design and Evaluation*”(do inglês, Monitoramento de Sensores IoT de Grande Área Usando um Sistema de Rede *mesh* Lora: Projeto e Avaliação). No artigo, os autores desenvolvem e mensuram uma rede *mesh* gerenciável, com nós visando a coleta de dados de vários sensores IoT de uma região.

Este algoritmo foi escolhido para a rede Hydra pois possui várias características interessantes. Por exemplo, toda a comunicação é iniciada no *gateway*, diminuindo o número de colisões(dois nós transmitindo ao mesmo tempo) e o número de transmissões desnecessárias quando comparado ao algoritmo de *flooding*. Além disso, a rede *mesh* construída é montada como uma ”árvore”, facilitando a estruturação do roteamento.

O *gateway* inicializa o processo de construção da rede emitindo periodicamente um sinal, denominado *beacon*. Este *beacon* é como um convite para que nós que o escutarem entrem

na rede. Se um nó captar este *beacon*, ele pode enviar uma requisição JOIN para o *gateway*, que torna-se seu ”pai”. A estrutura *child_list* contém uma lista de todos os nós mapeados na rede e, inicialmente, está vazia. Como demonstrado na Figura 13, quando a estrutura *child_list* possui nós adicionados, o *gateway* começa a recolher os dados disponíveis de maneira sequencial através da subfunção *Node_query*.

Figura 13 – Diagrama de funcionamento do *gateway mesh* Hydra

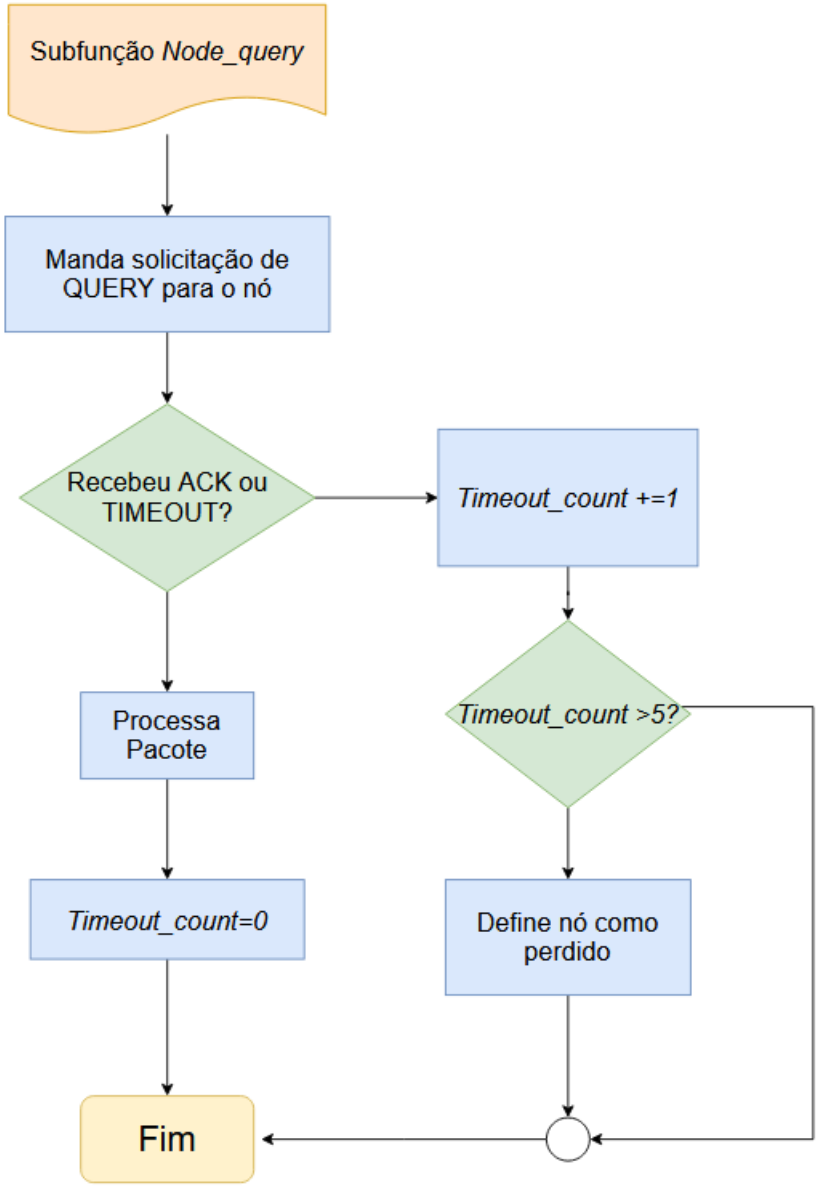


Fonte: Adaptado de Lee e Ke (2018).

A Figura 14 exemplifica a subfunção *Node_query*, chamada pelo *gateway* para cada nó inserido na rede. A função assume o papel de requisitar informações para um nó que está em *child_list*. O *gateway* envia uma requisição de QUERY para o nó selecionado e aguarda pela resposta. Caso não a receba, o *gateway* começa a incrementar um contador de tempo de espera, o *timeout_count*. Após determinada quantidade de tentativas de QUERY sem sucesso, o nó é dado como perdido. Caso o nó responda à solicitação, com um pacote ACK, por exemplo, o *gateway* processa a mensagem obtida e depois zera *timeout_count*.

Um ponto a se acentuar é que, como mostrado na Figura 13, assim que *child_list* possui seu primeiro nó, o *gateway* para de enviar o *beacon*, utilizando as solicitações de QUERY para notificar outros nós para participarem da rede.

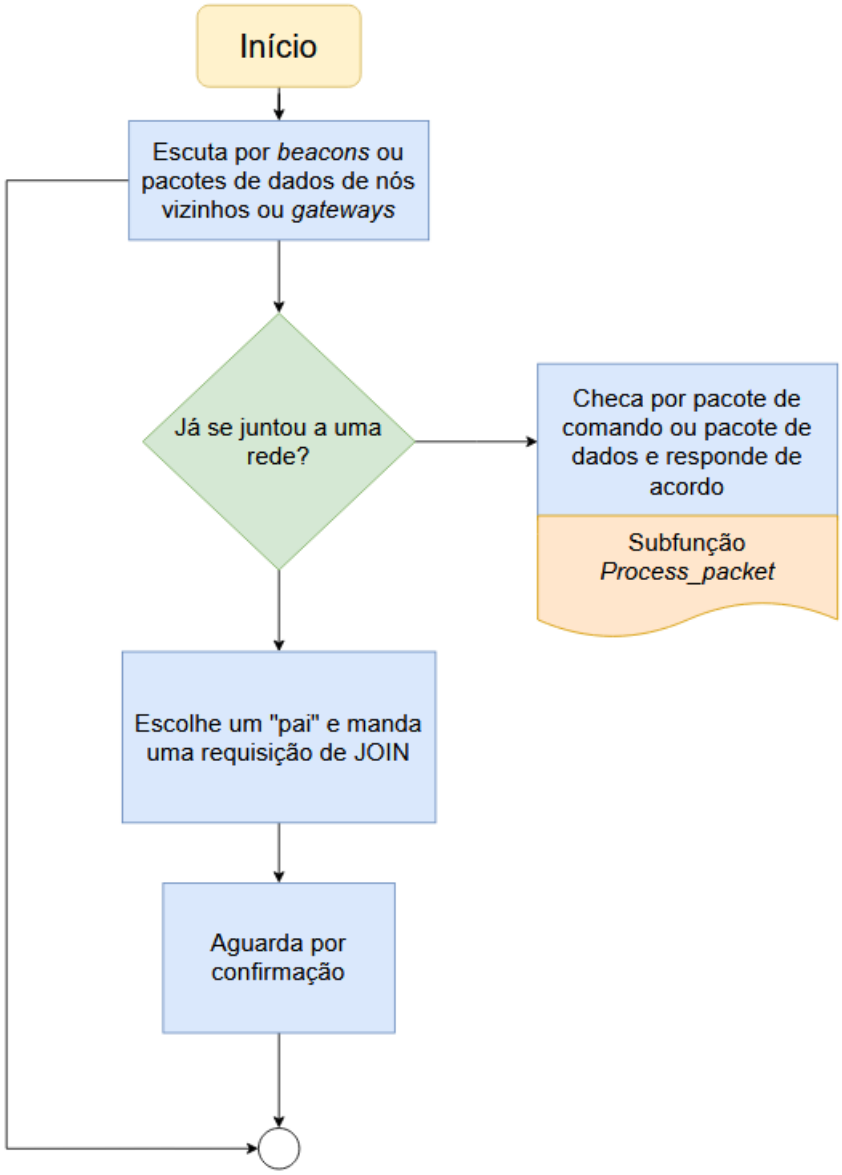
Figura 14 – Diagrama de funcionamento da subfunção *node_query* de um *gateway*



Fonte: Adaptado de Lee e Ke (2018).

A Figura 15 exibe o diagrama de funcionamento de um nó Hydra para com a rede *mesh*. Inicialmente, o nó abre sua comunicação para escutar *beacons* ou pacotes de dados de nós vizinhos ou *gateways*. Em caso de recebimento de um ou mais pacotes, caso o nó ainda não esteja associado a um rede *mesh*, ele escolhe o melhor ”pai”, envia uma requisição JOIN e aguarda pela resposta. No momento, esta escolha é puramente pelo sinal com maior intensidade. Quando o nó recebe um pacote mas já está inserido em uma rede, é chamada a função *Process_packet*, que avalia se o pacote contém dados ou um comando(requisição JOIN, por exemplo) e responde de acordo.

Figura 15 – Diagrama de funcionamento de um nó *mesh* Hydra

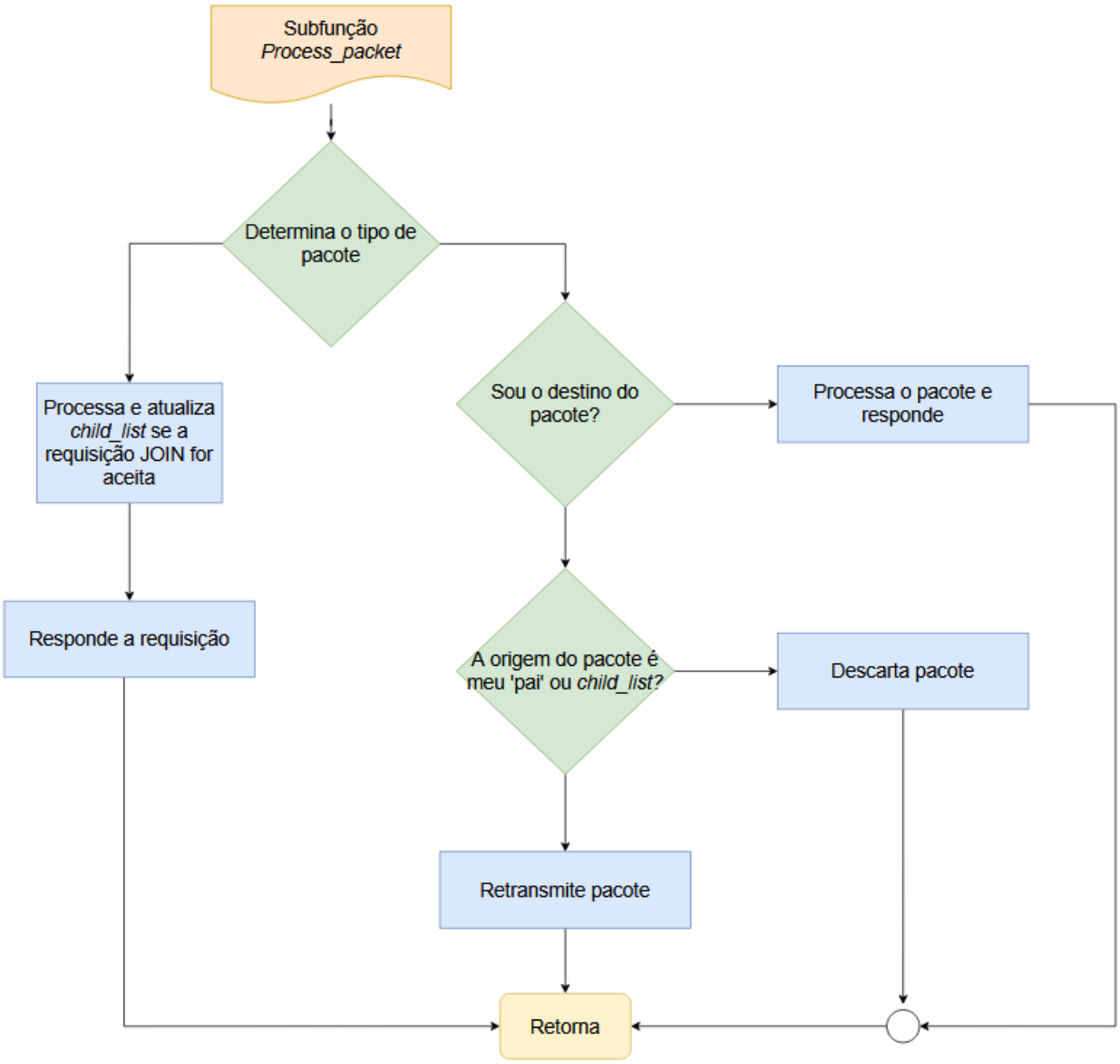


Fonte: Adaptado de Lee e Ke (2018).

A função *Process_packet* está descrita no diagrama da Figura 16. Ao receber um pacote, o nó verifica se é um pedido de JOIN. Em tal cenário, o nó processa o pacote, verificando se é possível adicionar o nó requisitante a sua *child_list* - a requisição JOIN pode ser rejeitada se a *child_list* do nó ”pai” estiver cheia, por exemplo. Aceitando a requisição, o nó envia a resposta para o nó que requisitou JOIN.

Em caso de uma pacote de dados ou de comando, o nó verifica se ele próprio é o destino daquele pacote. Em caso afirmativo, ele processa o pacote e responde. Caso contrário, ele retransmite o pacote caso verifique que o destino do pacote é para seu ”pai” ou para algum nó de sua *child_list*. Se o pacote não foi indicado para nenhum nó em sua memória, o nó descarta o pacote.

Figura 16 – Diagrama de funcionamento da subfunção *process_packet* de um nó *mesh* Hydra

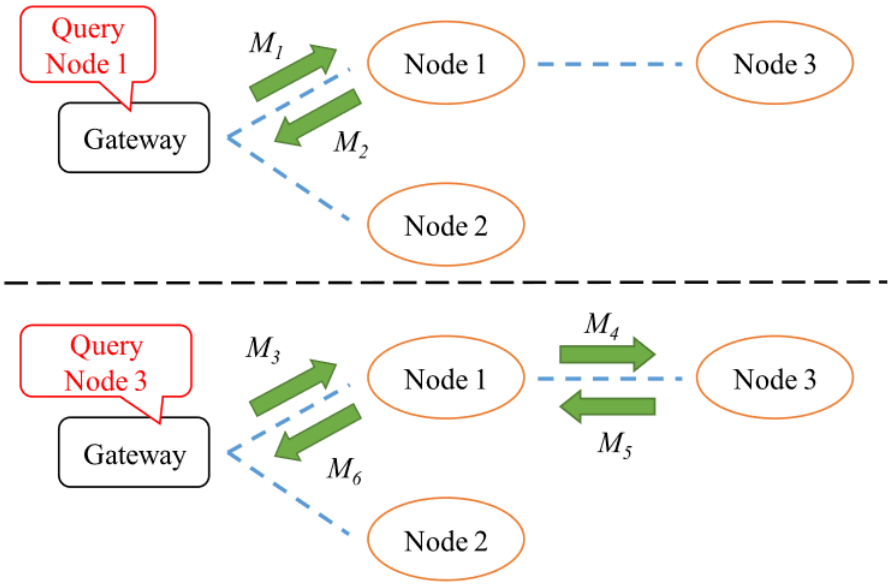


Fonte: Adaptado de Lee e Ke (2018).

Para exemplificar, a Figura 17 mostra um cenário onde o nó 1 se integra a rede *mesh* colocando o *gateway* como seu ”pai”. Este, por sua vez, adiciona o nó 1 à sua *child_list*. Então, o *gateway* faz uma requisição de QUERY, denominada M1, para o nó 1, que responde com M2. A transmissão M2 age como *beacon* para o nó 3, que é adicionado à rede com o nó 1 sendo seu ”pai”. O nó 1 adiciona o nó 3 a sua própria *child_list* e avisa o *gateway* da presença do nó 3. Desta forma, a rede *mesh* é criada.

Na próxima rodada de aquisição de dados, o *gateway* faz uma requisição de QUERY para o nó 3 através da mensagem M3, que é retransmitida pelo nó 1 através de M4. O nó 3 responde a requisição com M5 e, por fim, a resposta chega ao *gateway* com a transmissão M6. Tendo esta sequência em vista, todos os nós podem alcançar a rede e o *gateway* pode se comunicar com todos eles.

Figura 17 – Processo para um nó ingressar na rede e consulta do *gateway*

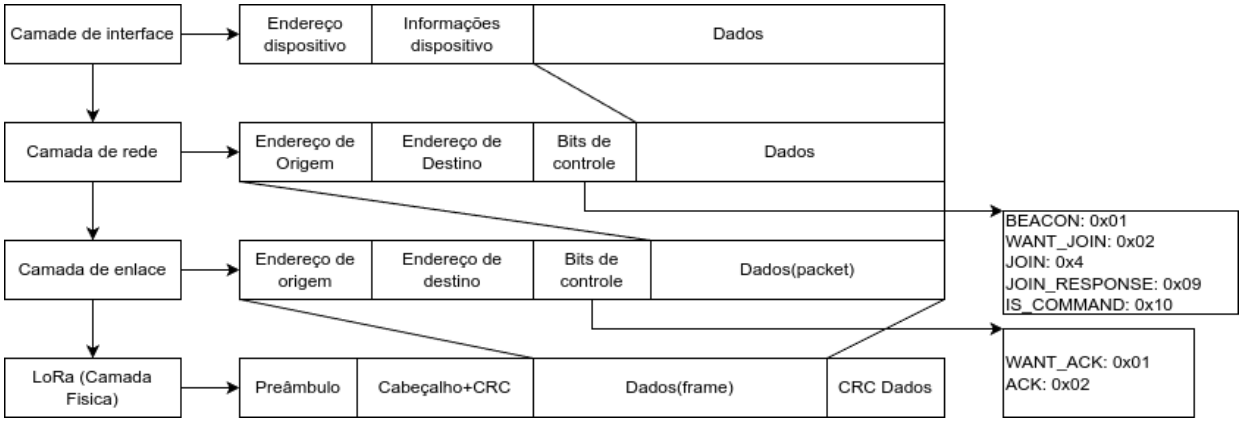


Fonte: Lee e Ke (2018).

4.1.3 Pilha de Protocolos Hydra

A pilha de protocolos da rede é mostrada na figura Figura 13. Ela é composta por quatro camadas, que são descritas em mais detalhes em seguida.

Figura 18 – Pilha de camadas da rede



Fonte: Autoria própria (2024).

4.1.3.1 Camada Física

A camada física é a camada implementada em *hardware* diretamente no rádio *LoRa*. Ela é composta por um preâmbulo, utilizado para a detecção de pacotes, sincronismo e ajuste de ganho. Um cabeçalho opcional que contém o tamanho dos dados no pacote, seguidos pelos dados e seu respectivo *checksum*

4.1.3.2 Camada de enlace

A camada de enlace é responsável pela comunicação ponto a ponto na rede. Ela é composta por um cabeçalho contendo o endereço único de *hardware* da origem e o endereço único do destino e bits de controle. Esses bits de controle são utilizados no caso de transmissões com confirmação, onde uma transmissão com o bit *WANT_ACK* habilitado, exige a resposta do receptor com o bit *ACK* habilitado, caso contrario é considerado que a mensagem não foi recebida.

4.1.3.3 Camada de rede

A camada de rede é responsável pela formação e roteamento de mensagens na mesh. Ela é composta por um cabeçalho contendo o endereço de rede do nó origem, endereço de rede do nó destino e bits de controle. Neste camada, os bits de controle *BEACON, WANT_JOIN, JOIN e JOIN_RESPONSE* são utilizados para auxiliar na formação da rede. E o bit *IS_*-

COMMAND indica se o conteúdo daquela mensagem é uma comando de configuração de rede ou uma mensagem contendo dados.

4.1.3.4 Camada de interface

A camada de interface é um protocolo essencial que atua como uma ponte entre as tecnologias LoRa, BLE e MQTT, encaminhando os dados para o ponto de saída da rede apropriado. Ela é formada por um cabeçalho que contem o endereço e meta dados sobre a informação contida no pacote.

4.1.4 Serviço BLE

Foi necessário projetar o próprio perfil BLE para que os nós consigam interagir de maneira uniforme com qualquer periférico que faça parte de rede. No contexto desta seção, um periférico se refere a qualquer sensor ou atuador que contenha apenas conectividade BLE. Este perfil é composto por um único serviço, demonstrado no Quadro 2.

A característica *Device ID* é um identificador único de um dispositivo periférico. A característica Network ID identifica a qual rede *Hydra* aquele dispositivo pertence, permitindo que múltiplas redes atuem na mesma areá com periféricos distintos.

Já a característica *Data* é utilizada para realizar toda troca de dados entre o periférico e o nó da rede. Embora a utilização de uma única característica para a representação de múltiplos tipos de dados vá contra a filosofia do BLE de usar características distintas para dados distintos, essas foi a melhor solução encontrada para criar um sistema genérico o suficiente de modo que fosse possível integrar múltiplos dispositivos diferentes a rede sem a necessidade da existência de serviços específicos para eles.

No periférico, optamos por utilizar os 27 bytes disponíveis no pacote de *advertisement* para transmitir que o dispositivo tem suporte ao serviço BLE, o ID do dispositivo e o ID da rede a qual ele pertence. Isso além de facilitar a filtragem de dispositivos próximos sem a necessidade de conexões ou *scans*, também permite que dispositivos apenas com a capacidade de transmitir, isso é, *beacons*, também possam ser integrados a rede.

Quadro 2 – Serviço Hydra BLE

Tipo	Nome	Permissões	UUID
Serviço	Hydra	Leitura	87ac62c8-be1a-42cd-afc3-59f3d72eb71c
Característica	Device ID	Leitura	94380b23-eac8-42c0-a4c4-d472a35a5378
Característica	Network ID	Leitura	8c006d13-1d0b-43ab-9d70-1402345e6eb8
Caracteristica	Data	Leitura Escrita Inscrição	fec0e8c0-247f-4012-8f3b-440b623e5c8a

Fonte: Autoria própria (2024).

4.2 *Hardware*

Para conseguir implementar uma rede de comunicação é preciso que existam dispositivos físicos capazes de comunicar-se entre si. Com o protocolo e ambiente de desenvolvimento definidos, foi necessário desenvolver um *hardware* que acomodasse tanto a transmissão quanto a recepção de LoRa e Bluetooth, com um microcontrolador poderoso o suficiente lidar com os protocolos de ambas as tecnologias somadas ao protocolo da Hydra.

Alguns pontos críticos foram levantados antes de começar o desenvolvimento do *hardware* do nó Hydra, pontos estes que guiaram sua concepção e elaboração:

- **Custo**: o preço total de um nó deve ser tão barato quanto possível, para diminuir os custos totais do projeto e conciliá-lo com os preceitos de uma rede LPWAN, tornando praticável a implementação de uma rede Hydra com dezenas de nós.
- **Tamanho**: as dimensões da placa de circuito impresso não podem passar de 50 milímetros de altura por 50 milímetros de altura, por conta das limitações do fornecedor. Além de tornar a fabricação mais econômica, isto também aumenta a versatilidade do nó Hydra que, por ter um tamanho reduzido, pode ocupar diferentes espaços de forma inexpressiva.
- **Acessível**: para não haver dificuldade na obtenção dos componentes, todos os elementos utilizados devem se de ampla difusão.

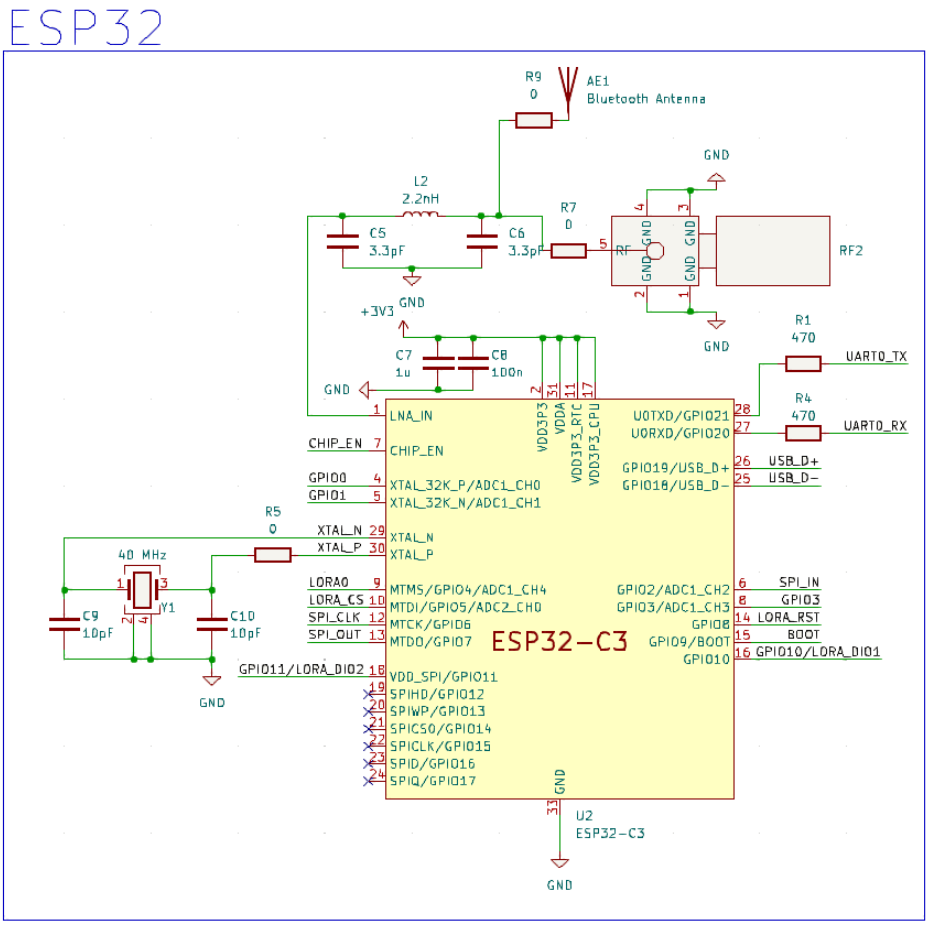
Dados todos esses objetivos, o primeiro passo foi a escolha do microcontrolador, que deve ser barato mas potente o suficiente para rodar as pilhas de protocolo. Foi escolhido o ESP32-C3, da Espressif(SYSTEMS, 2024), um microcontrolador de baixa potência que suporta as tecnologias Wi-fi de 2,4GHz e Bluetooth Low Energy(ESPRESSIF SYSTEMS, 2024). Além disso, ele possui 4MB de memória *flash* integrada e um conversor USB para Serial também integrado no próprio chip, diminuindo o número de componentes necessários. Todos os inte-

grantes do grupo já haviam tido contato com microcontroladores da mesma família, acelerando o desenvolvimento do projeto.

O ESP32-C3 possui todas as características buscadas inicialmente no projeto: ele possui um preço bastante acessível e é possível encontrá-lo à venda nas grandes lojas de produtos eletrônicos. O seu encapsulamento QFN32 possui apenas 5 milímetros de altura e de largura, ocupando pouco espaço na placa de circuito impresso. Além disso, economiza-se em componentes para integração com a tecnologia BLE e Wi-fi.

O esquemático utilizado para o microcontrolador pode ser observado na Figura 19. Foram seguidos criteriosamente as recomendações da fabricante na escolha dos componentes. Pode-se observar que, para a entrada de radiofrequência, foram colocadas duas opções de antenas: uma antena de micro-fita que seria construída na própria PCB; e um conector SMA para encaixe de uma antena externa. Elas podem ser comutadas colocando e/ou retirando os resistores de 0 Ω R7 e R9.

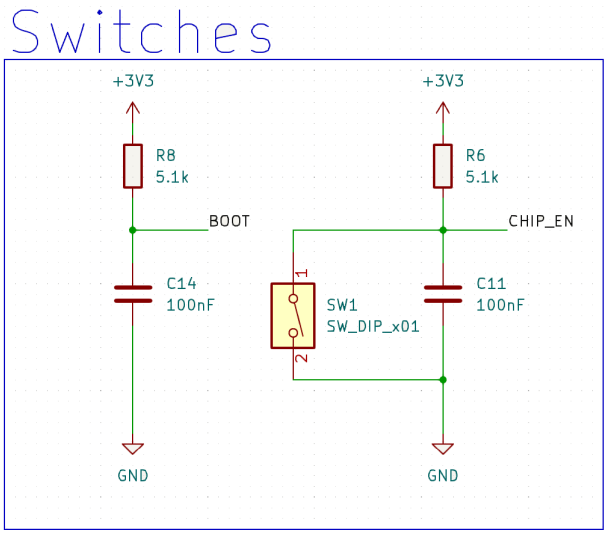
Figura 19 – Esquemático do microcontrolador ESP32-C3



Fonte: Autoria própria (2024).

A Figura 20 mostra o esquemático dos botões auxiliares do microcontrolador, um botão para sinal de *reset* e um *pull-up* para sinalizar o *bootloader*.

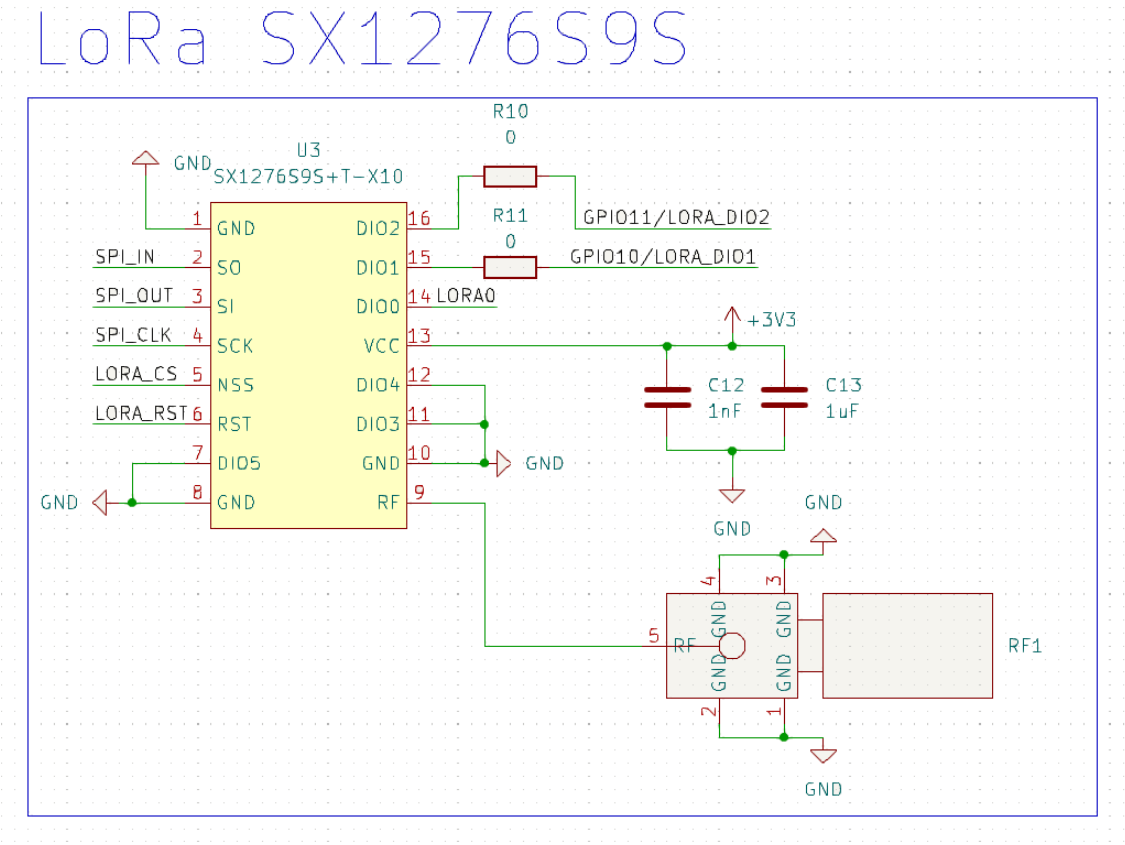
Figura 20 – Esquemático dos botões do módulo



Fonte: Autoria própria (2024).

Para a comunicação com a tecnologia LoRa foi escolhido o SX1276S9S+T-X1, um módulo de rádio frequência integrado com o *chip* SX1276(VOLLGO TECHNOLOGY, 2024). O *chip* se comunica através de uma interface SPI, retirando parte da complexidade da comunicação LoRa do microcontrolador principal. O esquemático pode ser observado na Figura 21, onde foram seguidas as recomendações do fabricante. Para a entrada de radiofrequência do módulo foi escolhido um conector SMA para encaixe de uma antena externa.

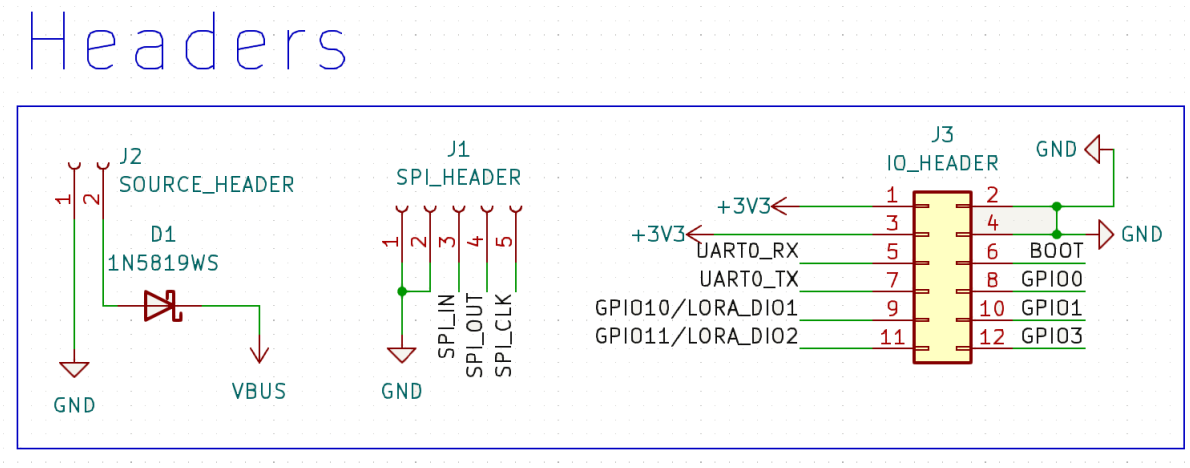
Figura 21 – Esquemático do rádio LoRa do módulo



Fonte: Autoria própria (2024).

Foram escolhidas algumas saídas para o módulo, como pode ser observado na Figura 22. Foram expostos os pinos da comunicação SPI, assim como cinco pinos de uso geral, para flexibilizar o uso de diferentes sensores e atuadores com cada nó Hydra. Também foram expostos os pinos da comunicação serial UART para comunicação e os pinos de alimentação. Ainda, foi exposta uma entrada para alimentação anterior ao regulador de tensão, para acoplamento de uma bateria.

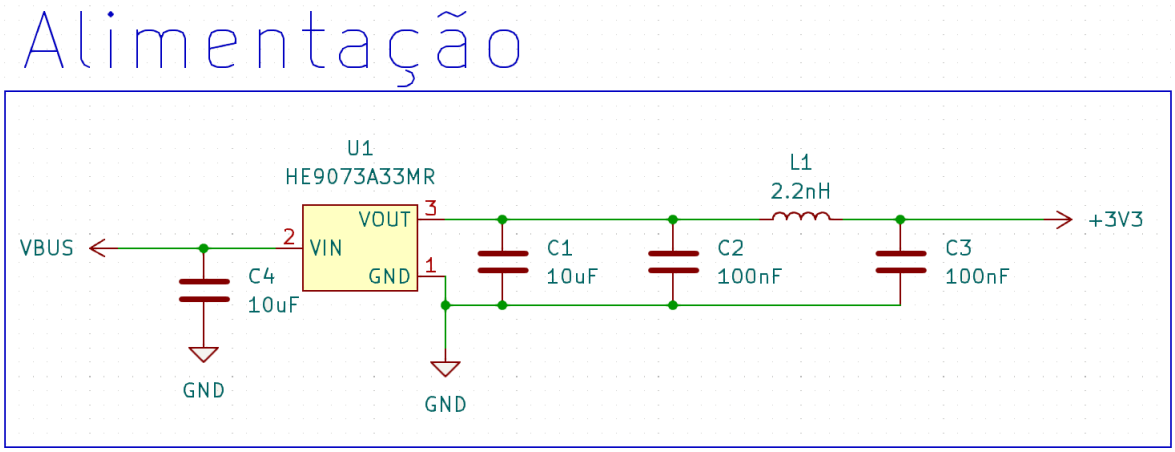
Figura 22 – Esquemático das saídas do módulo



Fonte: Autoria própria (2024).

Na Figura 23, pode-se observar o circuito de alimentação do microcontrolador ESP32-C3, cujo regulador de tensão linear escolhido foi o HE9073A33MR (HEERMICR, 2020), pela sua baixa queda de tensão e corrente quiescente. Na Figura 24, observa-se o conector USB tipo C, usado tanto para a alimentação do nó Hydra como para gravação do *firmware*.

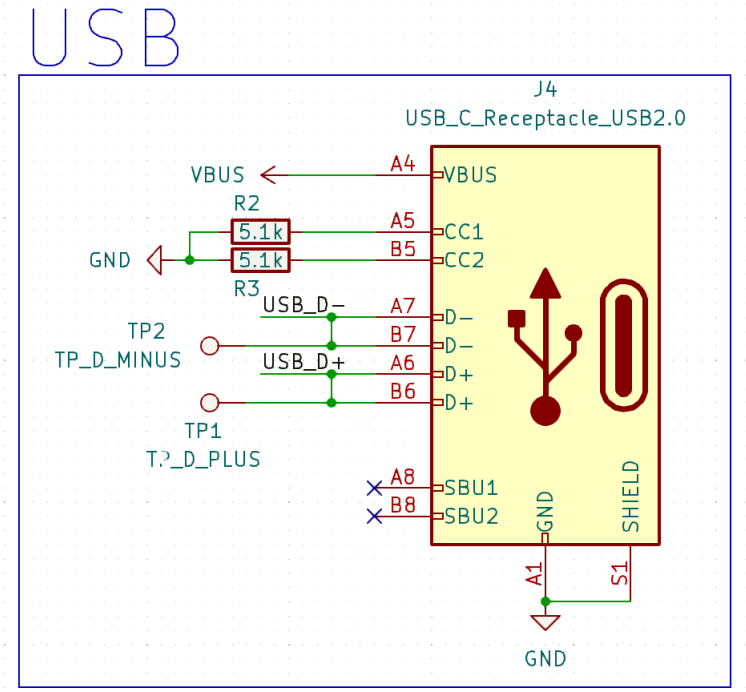
Figura 23 – Esquemático da alimentação do módulo



Fonte: Autoria própria (2024).

Com o esquemático definido, iniciou-se o processo de roteamento da placa de circuito impresso. O programa escolhido para o roteamento foi o KiCAD(KICAD, 2024), por todos os membros da equipe já terem conhecimento prévio sobre como utilizá-lo. Como o fornecedor

Figura 24 – Esquemático da entrada USB do módulo

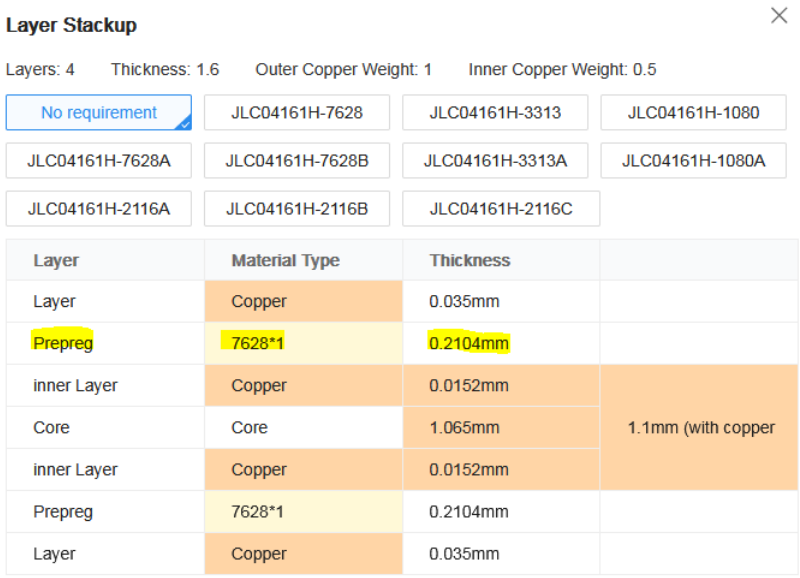


Fonte: Autoria própria (2024).

selecionado foi a chinesa JLCPCB(JLCPCB, 2024), a placa foi limitada às dimensões de 50 milímetros de largura por 50 milímetros de altura.

É interessante acrescentar que as trilhas em que percorrem o sinal de radiofrequência das antenas foram planejadas para terem impedância característica de 50 Ω. Com base nas informações fornecidas pela fabricante JLCPCB, que podem ser observadas na Figura 25 e na Figura 26, foi possível calcular a espessura da trilha utilizando o *software* QUCS(QUCS, 2017). Os resultados para a antena de 2,4GHz podem ser observados na Figura 27, ao passo que os resultados para a antena de 915MHz podem ser observados na Figura 28.

Figura 25 – Tamanho das camadas da PCB



Layer Stackup

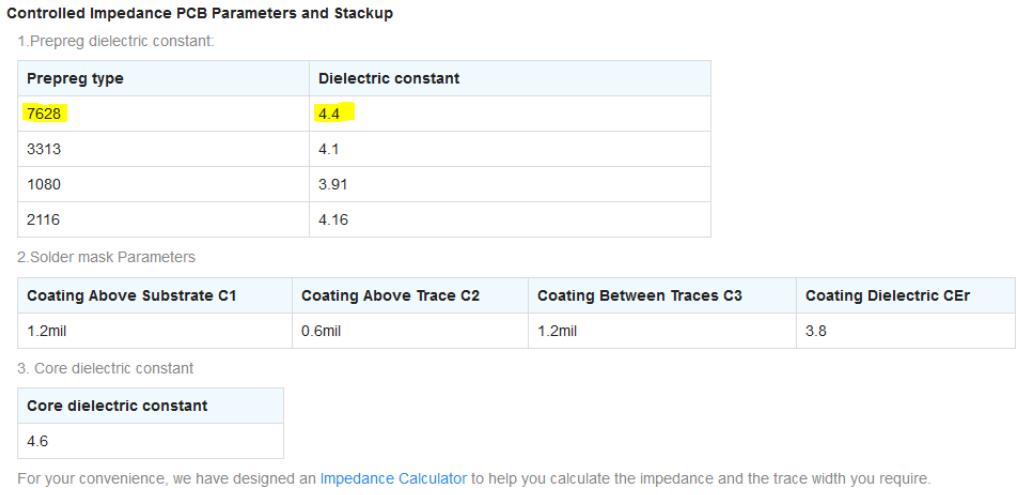
Layers: 4 Thickness: 1.6 Outer Copper Weight: 1 Inner Copper Weight: 0.5

No requirement | JLC04161H-7628 | JLC04161H-3313 | JLC04161H-1080
JLC04161H-7628A | JLC04161H-7628B | JLC04161H-3313A | JLC04161H-1080A
JLC04161H-2116A | JLC04161H-2116B | JLC04161H-2116C

Layer	Material Type	Thickness	
Layer	Copper	0.035mm	
Prepreg	7628*1	0.2104mm	
inner Layer	Copper	0.0152mm	1.1mm (with copper
Core	Core	1.065mm	
inner Layer	Copper	0.0152mm	
Prepreg	7628*1	0.2104mm	
Layer	Copper	0.035mm	

Fonte: JLCPCB (2024).

Figura 26 – Valor do dielétrico das camadas da PCB



Controlled Impedance PCB Parameters and Stackup

1.Prepreg dielectric constant:

Prepreg type	Dielectric constant
7628	4.4
3313	4.1
1080	3.91
2116	4.16

2.Solder mask Parameters

Coating Above Substrate C1	Coating Above Trace C2	Coating Between Traces C3	Coating Dielectric CEr
1.2mil	0.6mil	1.2mil	3.8

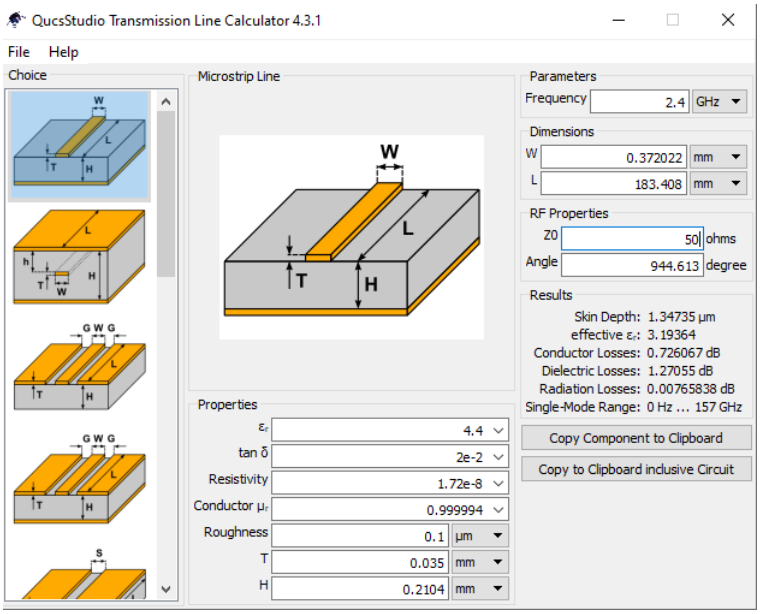
3. Core dielectric constant

Core dielectric constant
4.6

For your convenience, we have designed an Impedance Calculator to help you calculate the impedance and the trace width you require.

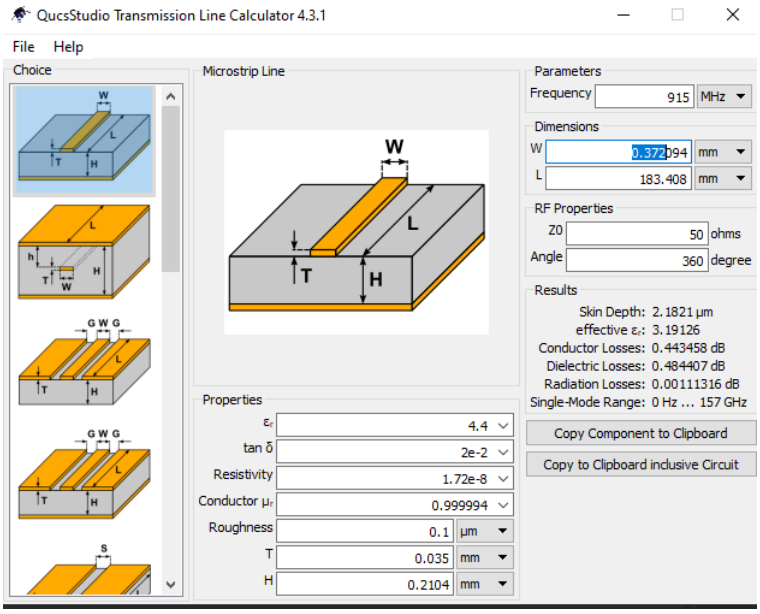
Fonte: JLCPCB (2024).

Figura 27 – Cálculo da Trilha de 2.4GHz no QUCS



Fonte: QUCS (2017).

Figura 28 – Cálculo da Trilha de 915MHz no QUCS

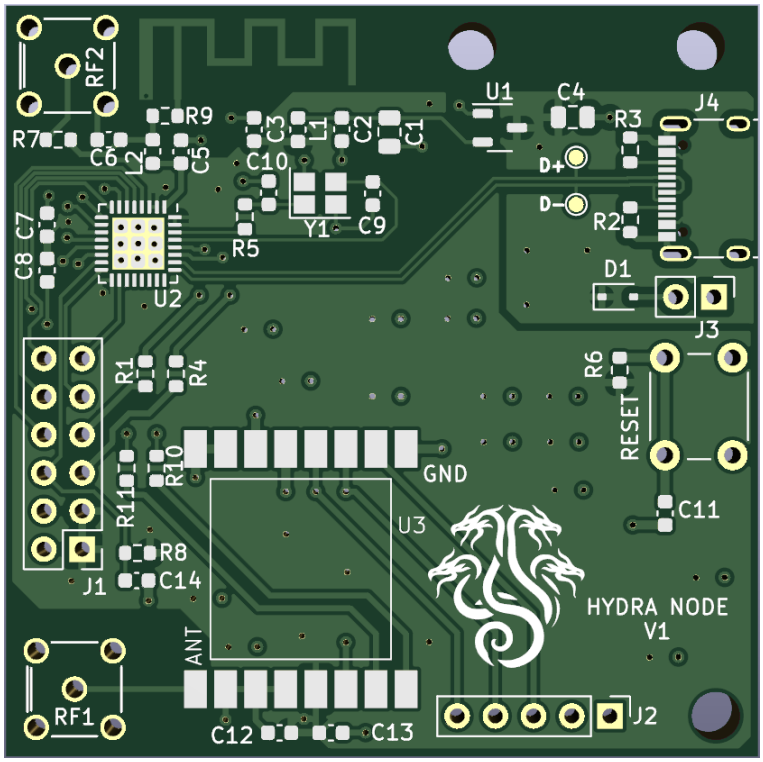


Fonte: QUCS (2017).

A Figura 29 mostra a visão frontal da PCB ao passo que a Figura 30 mostra a visão traseira. A placa possui 4 camadas, com as duas camadas interiores sendo a referência. Além disso, foram adicionados três furos para fixação.

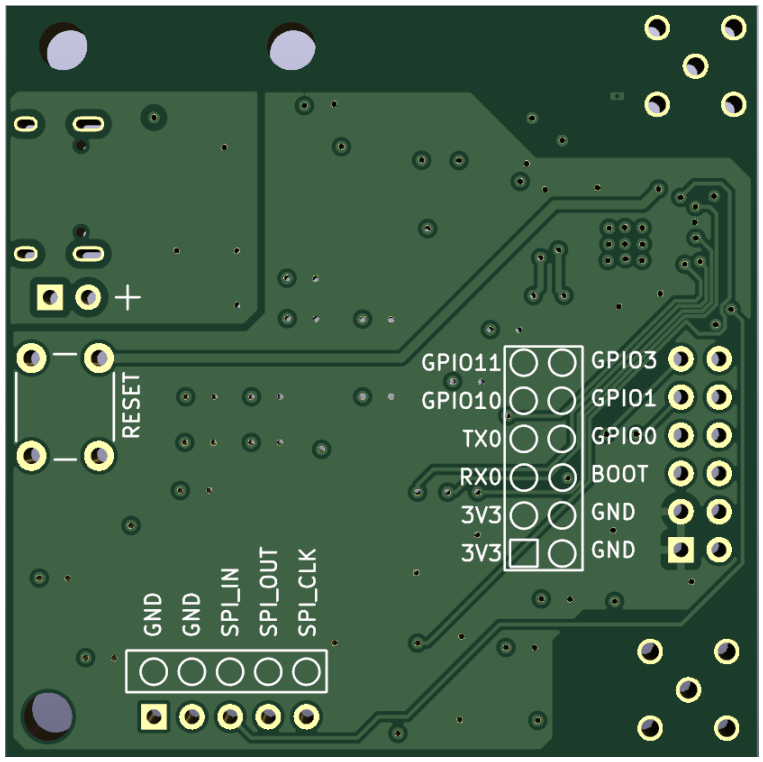
A lista completa dos componentes pode ser observada na Tabela 1.

Figura 29 – Visão frontal da PCB



Fonte: Autoria própria (2024).

Figura 30 – Visão traseira da PCB



Fonte: Autoria própria (2024).

Tabela 1 – Lista de Materiais do nó Hydra

#	Referência	Quantidade	Valor	Encapsulamento
1	C1, C4	2	10uF	0805
2	C2, C3, C11, C14	4	100nF	0603
3	C5, C6	2	3.3pF	0603
4	C7	1	1uF	0603
5	C8	1	100nF	0603
6	C9, C10	2	10pF	0603
7	C12	1	1nF	0603
8	C13	1	1uF	0603
9	D1	1	1N5819WS	SOD-323
10	J1	1	IO_HEADER	2x06 P2.54mm
11	J2	1	SPI_HEADER	1x05 P2.54mm
12	J3	1	SOURCE_HEADER	1x02 P2.54mm
13	J4	1	USB_C_2.0	-
14	L1, L2	2	2.2nH	0603
15	R1, R4	2	470 Ohm	0603
16	R2, R3, R6, R8	4	5.1k Ohm	0603
17	R5, R7, R9, R10, R11	5	0 Ohm	0603
18	RF1, RF2	2	-	BWSMA-KWE-Z001
19	SW1	1	-	Button_Switch_THT_6mm
20	U1	1	-	HE9073A33MR
21	U2	1	ESP32-C3	QFN-32-1EP
22	U3	1	SX1276S9S+T-X1	-
23	Y1	1	40 MHz	TZ0308D

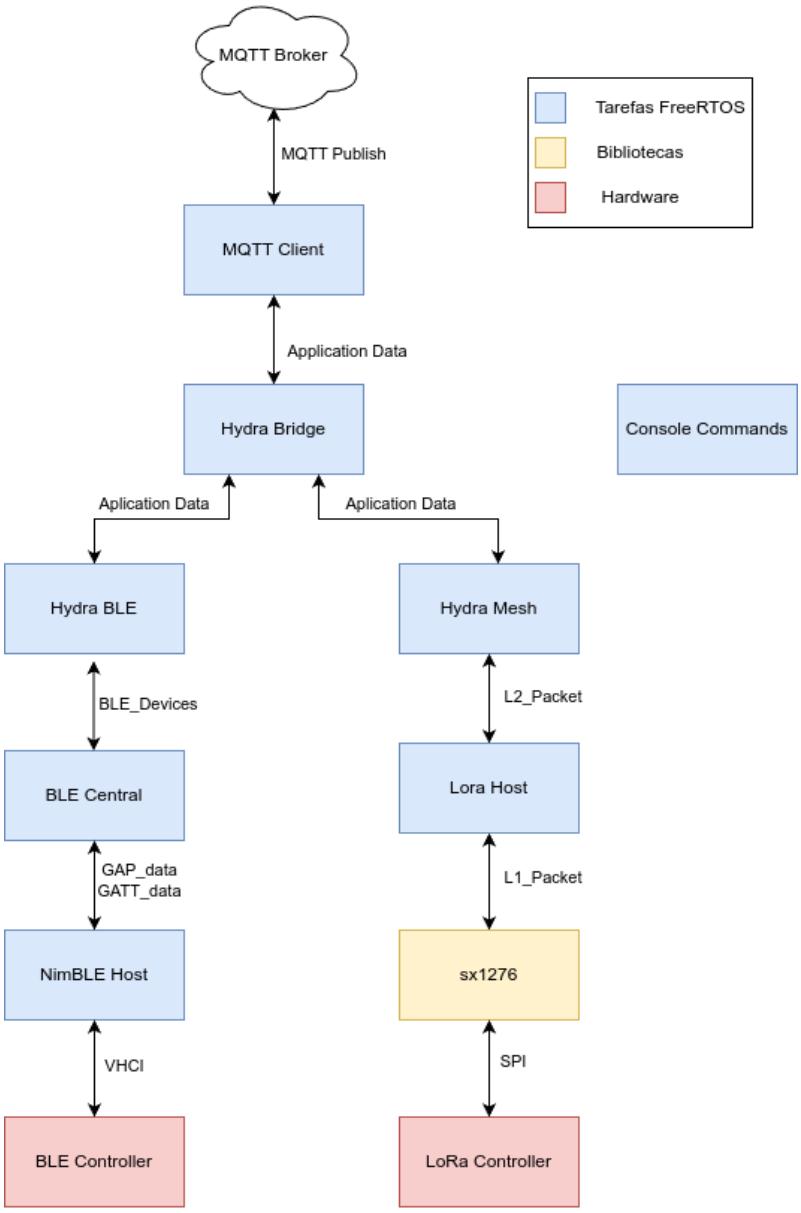
Fonte: Autoria própria (2024).

4.3 *Firmware*

A implementação do *firmware* foi feita utilzando a *framework* oficial recomendada para o desenvolvimento da linha de produtos ESP32 da Espressif, o *IoT Device Framework*(IDF). Pela natureza assíncrona da nossa aplicação, foi utilizado também extensivamente as funcionalidades do *FreeRTOS*(FreeRTOS, 2024), um sistema operacional de tempo real.

A implementação é composta por diversos subsistemas, como ilustrado na Figura 33. A maioria destes subsistemas operam em tarefas concorrentes diferentes e, utilizando estruturas de sincronização, recebem e enviam informações para outros subsistemas. Nesta seção são explicados o funcionamento de cada subsistema.

Figura 31 – Visão geral da implementação



Fonte: Autoria própria (2024).

4.3.1 *LoRa Host*

LoRa Host é a tarefa responsável por interagir com o *driver* do rádio LoRa e implementar a camada de enlace da rede. Como o rádio LoRa utilizado é *half-duplex*, essa tarefa é implementada como uma máquina de estados, ilustrada na Figura 32. O rádio pode estar em 4 estados: *receiving*(recebendo), *transmiting*(transmitindo), CAD e ocioso. O estado CAD corresponde a *Channel Activity Detection*(Detecção de Atividade no Canal), esse estado em especial tem a função de detectar se alguma informação está sendo transmitida no canal utilizado do rádio, uma funcionalidade necessária para a implementação do CSMA/CA.

Este subsistema interage através de filas com a camada superior da rede Hydra Mesh, o que é feito através de estruturas *L1_packet*, descritas no Quadro 3.

Quadro 3 – Estrutura de pacote L1

Campo	**Tamanho (bytes)**	**Descrição**
sender_mac	2	MAC do nó de origem
dest_mac	2	MAC do nó destino
packet_id	2	ID unico do pacote
len_bytes	2	Numero de bytes de data
flags	1	Flags da camade de enlace
data	192	Conteudo do pacote

Fonte: Autoria própria (2024).

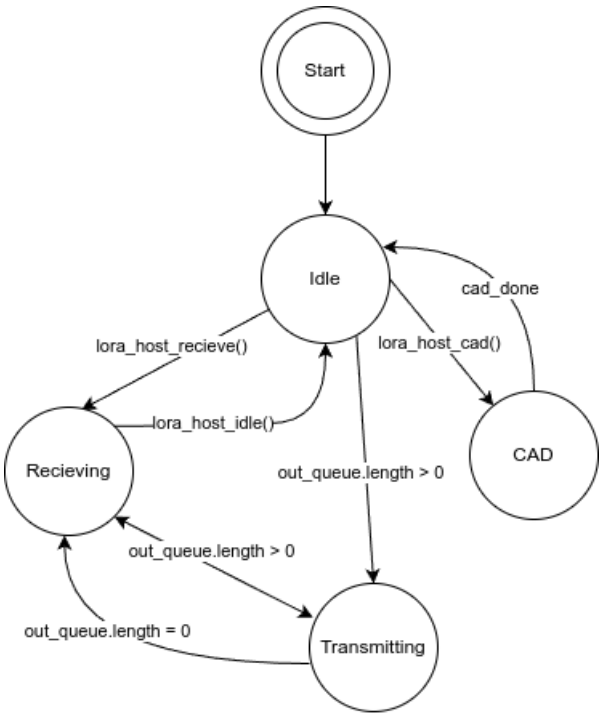
4.3.2 Hydra Mesh

O subsistema Hydra Mesh é responsável pela implementação da camada de rede *mesh*. Esta camada implementa o protocolo *mesh* descrito na seção 4.1. Sua função é tanto de formar a rede quanto de rotear os pacotes para o nó destino. O pacote *L2_packet* utilizando nesta camada é mostrado no Quadro 4. Esta estrutura é encapsulada nos campos de dados do pacote L1.

4.3.3 NimBLE Host

NimBLE é uma implementação da pilha de protocolos BLE desenvolvida pela *Apache Foundation*. Originalmente feita para o sistema operação de tempo real *MyNewt*, o NimBLE hoje em dia é disponibilizado independentemente para ser utilizado em outros sistemas e é oferecido como um dos módulos de BLE pelo IDF. Sua principal responsabilidade é implementar o *host* Bluetooth e fornecer uma API para acesso as funcionalidades do GATT e do GAP.

Figura 32 – Máquina de estados do LoRa Host



Fonte: Autoria própria (2024).

Quadro 4 – Estrutura de pacote L2

Campo	Tamanho (bytes)	Descrição
src_addr	2	Endereço do nó de origem
dest_mac	2	Endereço do nó destino
len_bytes	2	Numero de bytes de data
flags	1	Flags da camada de rede
data	186	Conteúdo do pacote

Fonte: Autoria própria (2024).

4.3.4 Hydra BLE Central

Esse subsistema é responsável por configurar e inicializar o nó como um dispositivo central de BLE, além de tratar as mensagens recebidas via GATT e GAP. Além disso, ele fornece funções e estruturas de dados para o subsistema Hydra BLE.

4.3.5 Hydra BLE

Responsável por realizar a varredura periódica de dispositivos próximos que anunciem suportar o serviço BLE Hydra. Eventualmente se conecta a esses dispositivos, obtendo os dados e informações mais atualizadas sobre esse dispositivo. Também disponibiliza essas infor-

Quadro 5 – Flags do pacote L2

Nome	Descrição
beacon	Indica que essa mensagem é um *beacon* do *gateway*
join	Mensagem é uma requisição de um nó para se juntar a rede
join_response	Resposta de um *join*, indica se nó foi aceito ou não na rede
network_cmd	Indica se o conteúdo dessa mensagem é um comando da rede ou data

Fonte: Autoria própria (2024).

mações para serem eventualmente encaminhadas para a rede Hydra ao receber um *probe* do *gateway*.

4.3.6 Hydra Bridge

É o subsistema responsável por fazer a ponte entre as três tecnologias utilizadas pela rede. No caso do *gateway*, ele é responsável por fazer a ponte entre o subsistema Hydra *Mesh* e o cliente MQTT. Já no caso do nó, ele é responsável pela ponte entre o Hydra BLE e o Hydra Mesh.

4.3.7 *Console Commands*

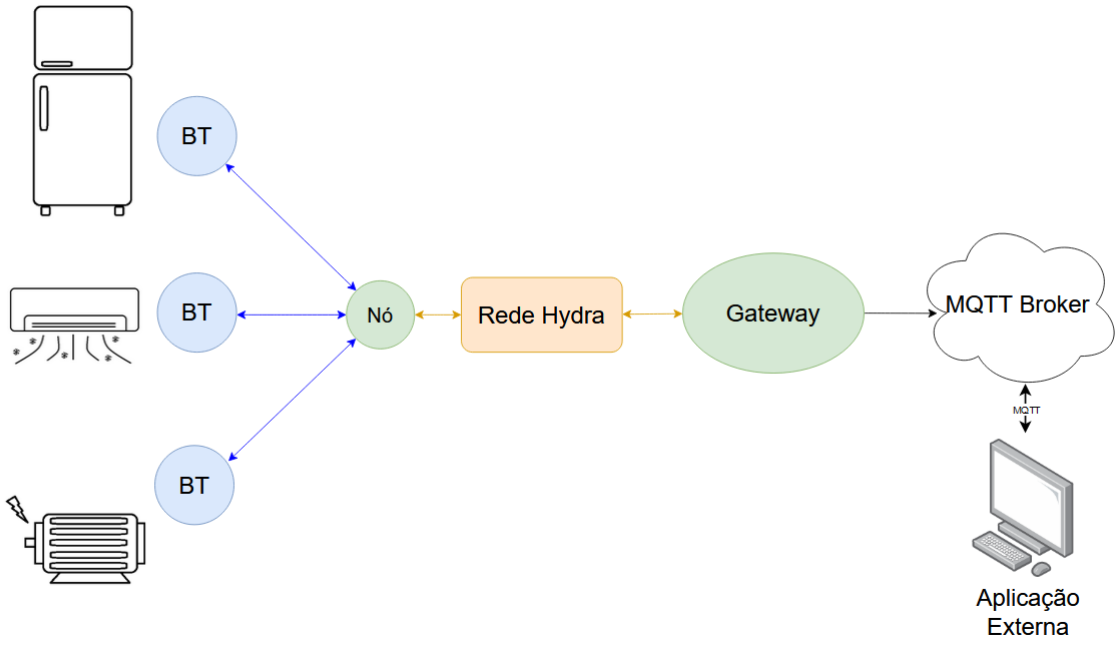
Responsável por fornecer um console utilizando UART via USB que permite que o usuário configure, depure e monitore a rede quando conectado a um computador.

4.4 Aplicação

Muito embora a rede Hydra seja completamente indiferente aos dados que por nela trafegam, foi necessária uma solução para melhor apresentar o seu funcionamento. Assim, como plano de fundo do projeto, foi escolhida uma rede de medição e controle de sistemas de potência.

O sistema seria composto de um dispositivo capaz de medir a tensão e a corrente de um aparelho conectado a uma fonte de energia, como uma tomada. Este dispositivo possuiria comunicação BLE, enviando os dados de tensão e corrente para o nó Hydra periodicamente. Além disso, o dispositivo também poderia receber um comando de desligar o aparelho conectado a ele através da comutação de um relé. A Figura 33 exemplifica o conceito da aplicação. Como pode-se observar, faz-se a comunicação do *gateway* da rede Hydra com a aplicação externa através de um *MQTT Broker*.

Figura 33 – Exemplo de aplicação Hydra



Fonte: Autoria própria (2024).

O *gateway* Hydra envia as informações da rede através dos tópicos descritos no Quadro 6. Ele publica periodicamente, em formato JSON, a última atualização dos valores de cada medidor atrelado a cada nó através do tópico 1, cuja estrutura está definida no Quadro 7. Através do tópico 2, é enviado a configuração de ”pais e filhos” da rede. Além disso, o *gateway* subscreve-se no tópico de todos os nós da rede e, através do formato do tópico 3, toma a ação de enviar para o nó a função de ligar ou desligar o relé do medidor escolhido pelo usuário na aplicação.

Quadro 6 – Lista dos tópicos MQTT para a aplicação

#	Tópico	Descrição	Formato
1	measures/[nodeID]	Medida das grandezas do nó	[structMedidas]
2	mesh/[GatewayID]	Formação de nós atual da rede	[ParentID][Child1][Child11],[Child2]
3	[nodeID]/[meterID]	Ativa ou desativa o relé medidor	[state](true = ON e false = OFF)

Fonte: Autoria própria (2024).

Com base nessas idealizações, foi construída uma aplicação externa a rede Hydra, em Python(FOUNDATION, 2024). Ao ligar a aplicação, ela automaticamente se conecta ao MQTT *broker*, se inscrevendo nos tópicos 1 e 2 do Quadro 6. Todas as informações colocadas nos tópicos são adicionadas à um banco de dados, cujas tabelas estão descritas na Figura 34. A tabela *Mesh* contém os identificadores de todos os nós presentes na rede naquele momento, junto com o ”pai” do nó. Sabendo essas informações, é possível montar a rede desde o *gateway* até o último nó disponível.

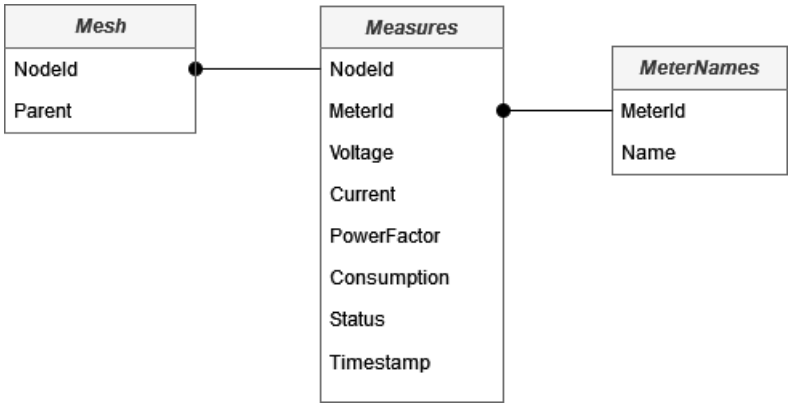
Quadro 7 – Estrutura de dados das medições

Campo	Unidade	Descrição	Formato
ID do medidor	N/A	ble_addr_t(6 bytes)	0xD2EB79F95560
Tensão	Volts	float(4 bytes)	127.11V
Corrente	Ampere	float(4 bytes)	3.2A
Fator de Potência	N/A	float(4 bytes)	0.92
Consumo	quilowatt-hora	float(4 bytes)	5.4kWh
Estado	N/A	uint8_t(1 byte)	0=Desligado, 1 = Ligado

Fonte: Autoria própria (2024).

A tabela *Measures* contém todas as medidas recebidas pela aplicação. Cada vez que recebe as medidas pelo servidor MQTT, a aplicação adiciona o horário(*timestamp*), obtendo um histórico cronológico de todos os medidores da rede. Desta forma, a aplicação pode filtrar as medidas por nó ou medidor. Por fim, a tabela *MeterNames* contém um nome para cada medidor, definido pelo próprio usuário na aplicação, para uma melhor administração da rede. Ao invés de se lembrar que o medidor que está ligado à geladeira tem o ID ”0xAD”, o usuário pode digitar o nome ”Fridge”, por exemplo.

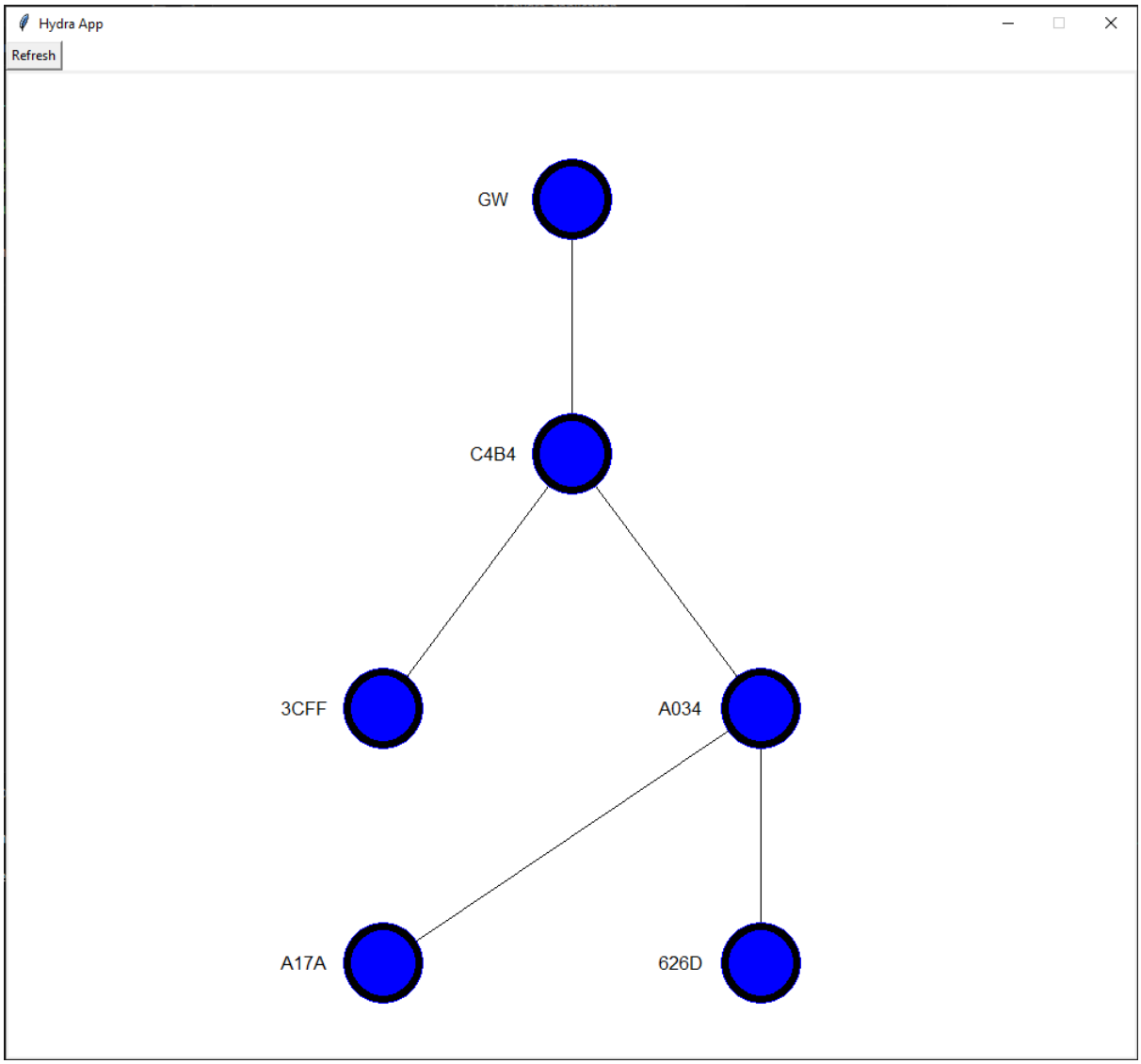
Figura 34 – Banco de Dados da Aplicação Hydra



Fonte: Autoria própria (2024).

A seguir, será descrito o fluxo de telas da aplicação. A tela inicial está representada pela Figura 35. Nela, pode-se observar um exemplo de rede Hydra com 5 nós, todos conectados ao mesmo *gateway*. A tela é gerada dinamicamente, com base no que está disponível na tabela *Mesh* do banco de dados.

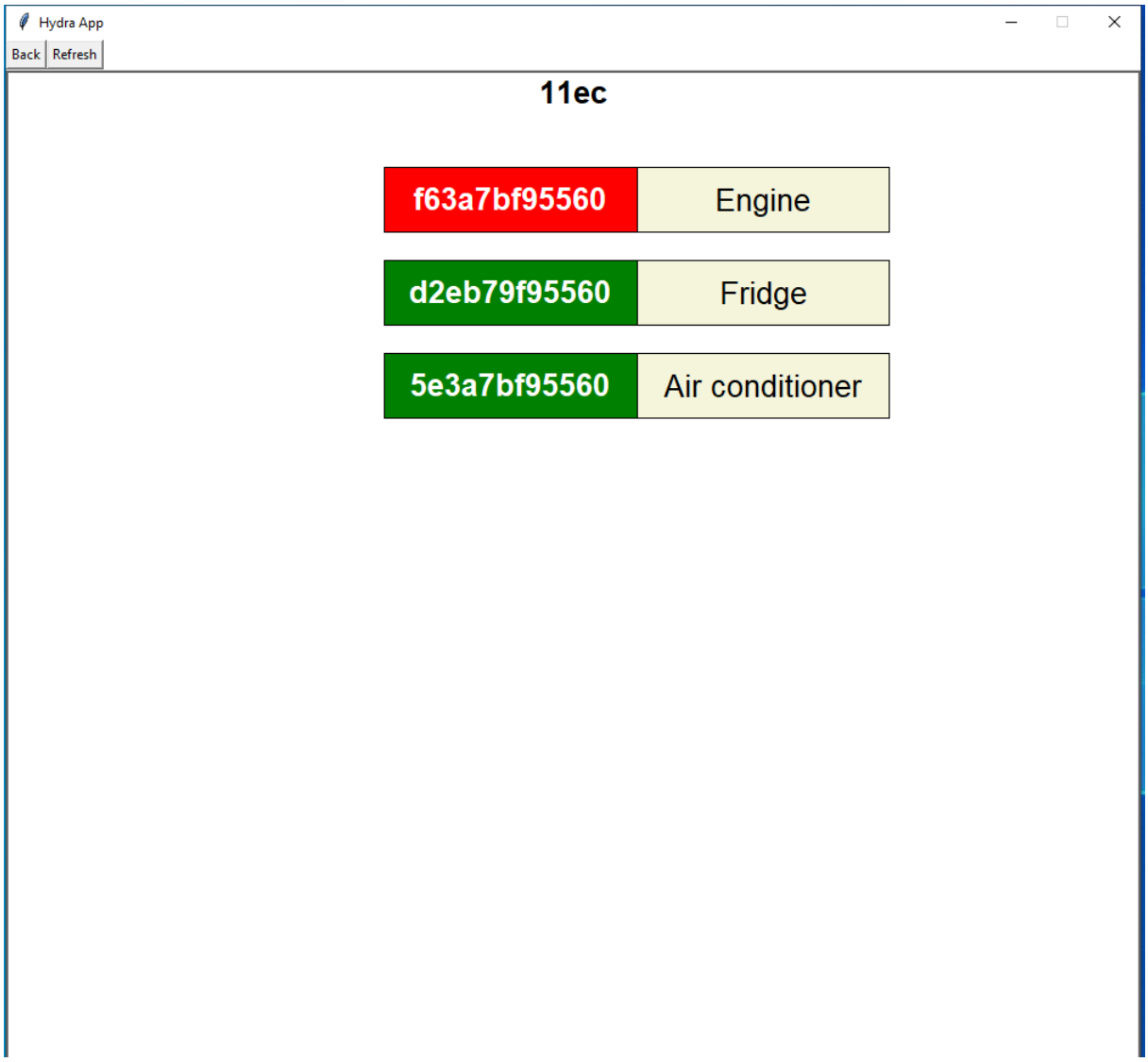
Figura 35 – Exemplo de Menu Inicial da Aplicação Hydra



Fonte: Autoria própria (2024).

Ao clicar em qualquer um dos Nós, é aberto o Menu do Nó, cujo exemplo pode ser observado na Figura 36. São mostrados todos os medidores que já se conectaram àquele nó. As cores que preenchem o identificador do medidor representam o estado atual do medidor, se está ligado(verde) ou se está desligado(vermelho). É neste menu em que o usuário pode alterar o nome do medidor, para identificar em que aparelho ele está ligado.

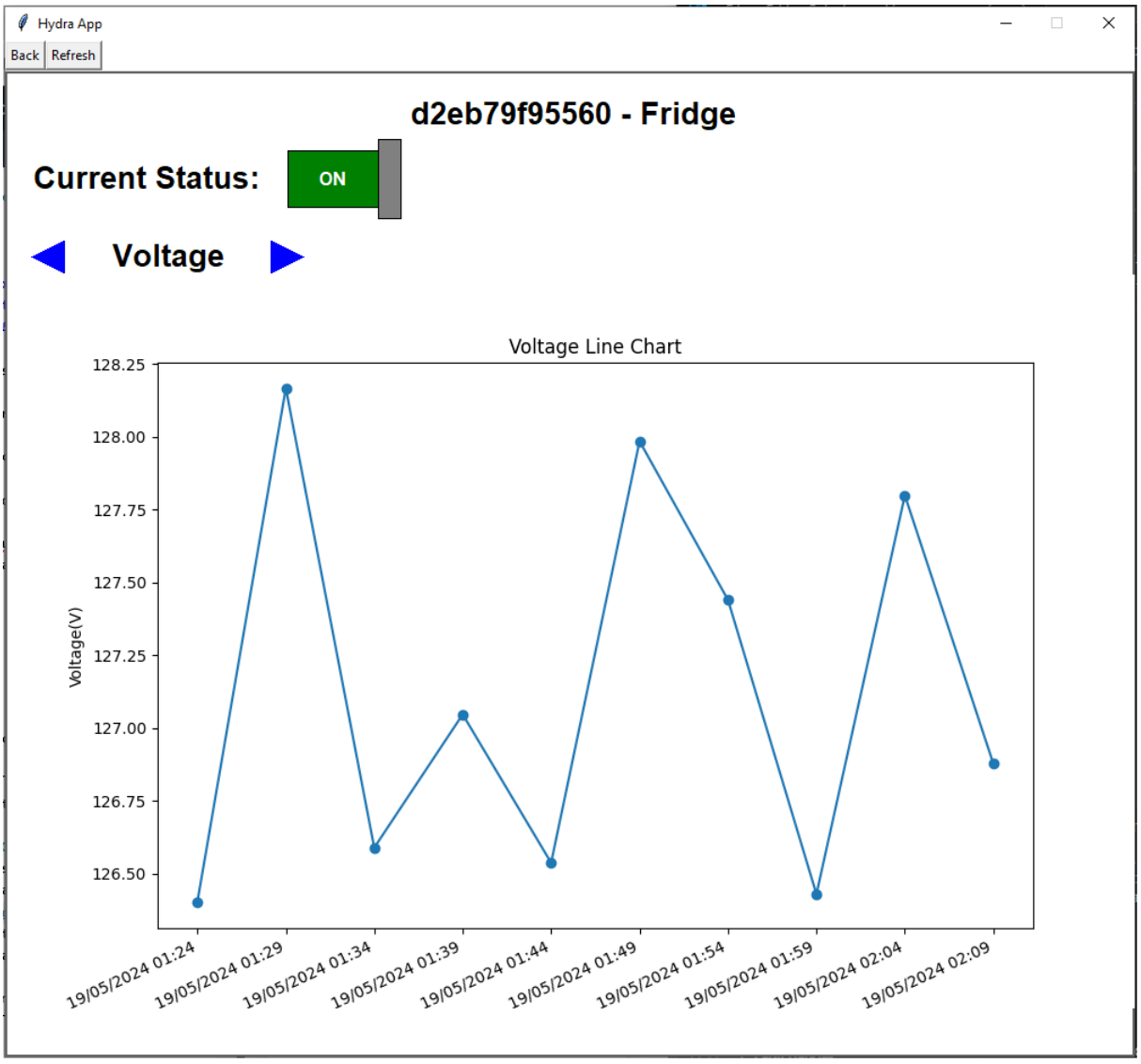
Figura 36 – Exemplo de Menu do Nó na Aplicação Hydra



Fonte: Autoria própria (2024).

Por fim, ao se clicar em um dos medidores, o usuário é redirecionado para o Menu do Medidor, exemplificado pela Figura 37. Nele é possível consultar as medidas através dos gráficos de todas as grandezas no tempo. Também é possível alterar o estado do relé do medidor em questão: ao clicar no botão no topo da tela, a aplicação publica no tópico 3 do Quadro 6 para que o *gateway* envie uma mensagem ao medidor.

Figura 37 – Exemplo de Menu do Medidor na Aplicação Hydra



Fonte: Autoria própria (2024).

4.5 Simulador da Rede Hydra

O teste de redes sem fio, principalmente para sistemas com alto alcance, é uma tarefa complexa. Pode haver a necessidade de grandes áreas geográficas para a realização dos testes, principalmente para testes de instabilidade. Por fim, em situações onde problemas ocorrem, há uma dificuldade de determinar a sequência de eventos que causou o problema, consequência de vários sistemas operando independentemente entre si.

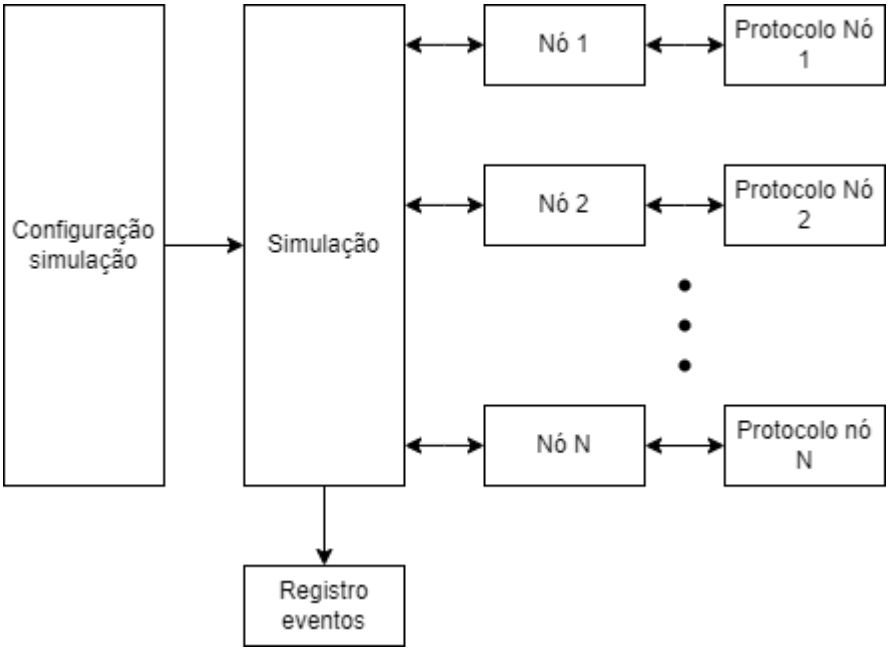
Assim, foi implementado um simulador que permite uma análise prévia do sistema de forma a resolver problemas antes que eles se manifestem no sistema real.

A obtenção de dados estatísticos sobre o funcionamento do sistema também é facilitado pela simulação, já que não necessita de vários equipamentos físicos que podem ter um alto custo, para testar a escalabilidade da rede. Sua reconfiguração também é facilitada por não necessitar de acesso físico aos dispositivos.

4.5.1 Arquitetura do simulador

O Simulador consiste em uma lista de Canais, que pode receber eventos de vários nós. Tais eventos permitem a interação dos nós com o Canal, possibilitando o uso do Canal para comunicação entre si. A Figura 38 mostra de forma geral a interação dos nós com o Simulador.

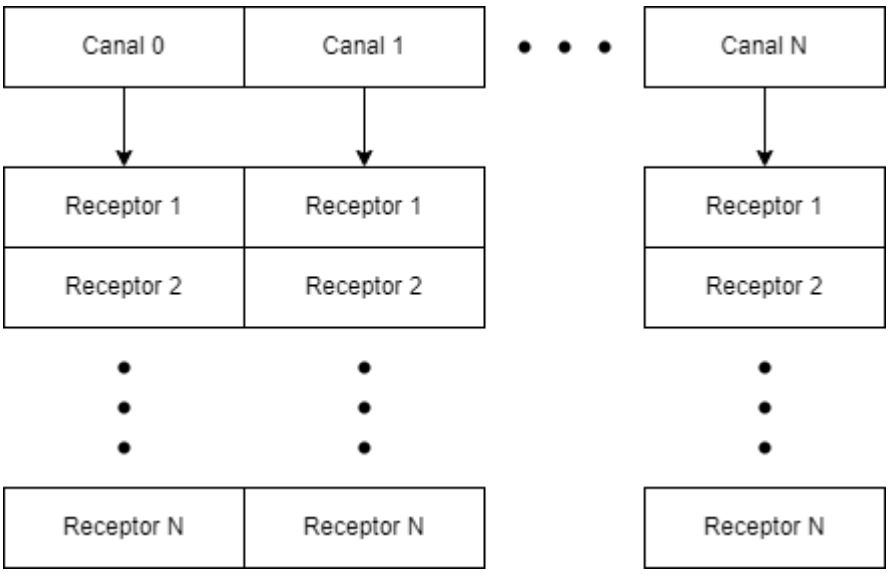
Figura 38 – Arquitetura Do Simulador Da Rede Hydra



Fonte: Autoria própria (2024).

Cada Canal mantém uma lista de nós em escuta, apresentada na Figura 39, que é atualizada com os eventos de acesso. Cada nó na lista também possui uma lista com os possíveis transmissores, que também é atualizada com os eventos de acesso ao canal.

Figura 39 – Dados internos do simulador



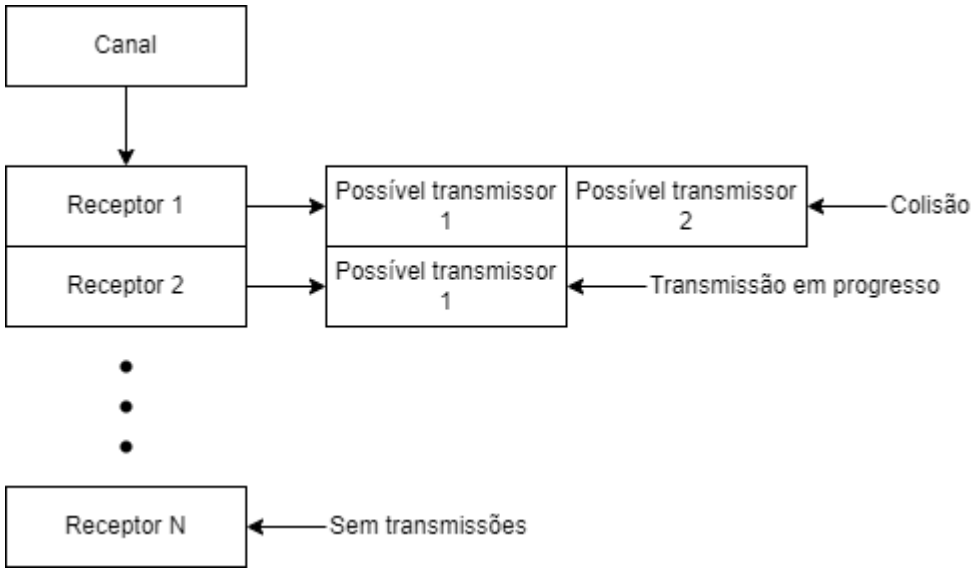
Fonte: Autoria própria (2024).

4.5.2 Eventos de acesso ao canal

Um nó pode gerar quatro tipos de eventos: iniciar transmissão, finalizar transmissão, iniciar escuta, finalizar escuta.

Tais eventos são tratados pelo Canal correspondente, atualizando seu estado interno, apresentado na Figura 40, e notificando nós que receberam mensagens. Situações de colisão durante uma transmissão também são devidamente tratadas.

Figura 40 – Informações armazenadas nos receptores dentro da simulação



Fonte: Autoria própria (2024).

4.5.2.1 Iniciar escuta

Um nó que envia um evento de iniciar escuta a um canal é adicionado à lista de nós escutando tal canal. Inicialmente, a lista de candidatos a transmissores está vazia.

4.5.2.2 Finalizar escuta

Um nó que envia um evento de finalizar escuta a um canal é removido da lista de nós escutando tal canal.

4.5.2.3 Iniciar transmissão

Ao receber um evento de iniciar transmissão, o Simulador irá adicionar o nó que está iniciando a transmissão à lista de candidatos a transmissores de cada nó em escuta que esteja dentro da distância máxima de transmissão do Canal.

4.5.2.4 Finalizar transmissão

Ao receber um evento de finalizar transmissão, o Simulador irá verificar cada nó escutando o canal. Caso a lista de possíveis transmissores contenha unicamente o nó que finalizou a transmissão, é considerado que houve uma transmissão bem sucedida, gerando uma notificação ao nó em escuta.

Na lista de possíveis transmissores, a presença de outros nós junto do nó que finalizou a transmissão indica que houve uma colisão, gerando a destruição da mensagem para aquele receptor.

4.5.3 Limitações da simulação

Devido à complexidade de simulação de todas as características de um transmissor em um ambiente sem fio, o simulador apresenta as seguintes limitações.

4.5.3.1 Canais perfeitamente ortogonais

Durante a simulação, é considerado que mensagens enviadas em um canal não afetam de maneira alguma mensagens enviadas em outros canais.

4.5.3.2 Recepção de mensagem depende apenas da distância

A única condição para envio correto de uma mensagem entre nós é a distância entre tais nós. Não são consideradas na simulação variações no meio ou outros efeitos que podem afetar a entrega de mensagens. Assim, caso uma mensagem seja entregue entre dois nós, há uma garantia de que, a menos que outro nó cause a destruição da mensagem, os dois nós sempre poderão comunicar entre si.

4.5.4 Registro do resultado da simulação

A simulação é executada e, a cada evento relevante, salva algumas informações relevantes ao evento. Após a finalização da simulação, os eventos são salvos em um arquivo de texto para análise posterior.

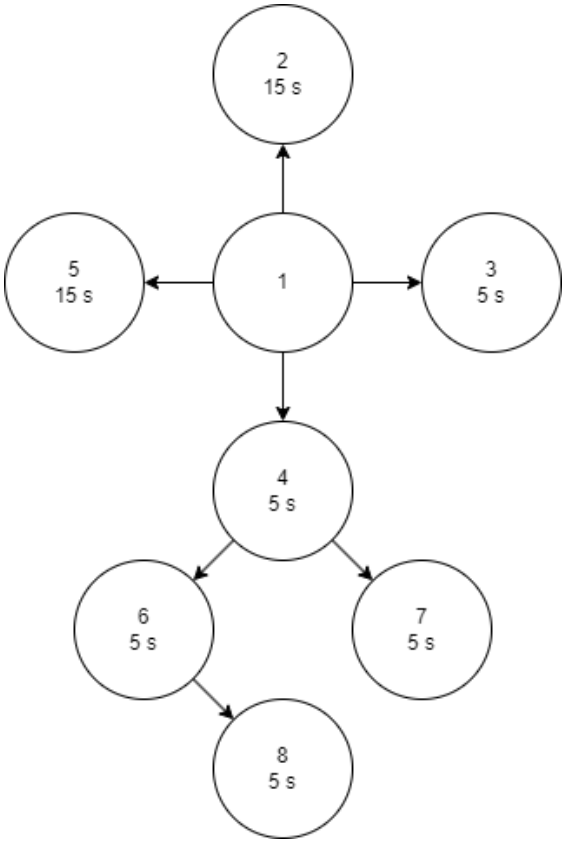
5 RESULTADOS E DISCUSSÕES

5.1 Comparação de algoritmos por simulador

O simulador, além de auxiliar no desenvolvimento da rede física, também foi usado para gerar alguns dados comparando a performance do algoritmo implementado na Hydra com o algoritmo utilizado no trabalho anterior, o *flooding*.

Para efeitos de comparação, será utilizada a rede apresentada na Figura 41. Os nós, numerados de 1 a 8, geram periodicamente informações novas que devem ser enviadas para o *Gateway*, nó de número 1. O tempo para a geração de informação em cada nó está marcado no nó correspondente.

Figura 41 – Rede simulada para comparação entre algoritmos



Fonte: Autoria própria (2024).

Também foi feita uma simulação comparando o efeito da adição de nós gerando informações com uma frequência excessiva, possivelmente sobrecarregando a rede. Para isso, o período de geração de informações do nodo 8 foi alterado de 5 segundos para 0,25 segundo.

Dessa forma, com as informações apresentadas na Tabela 2, é possível observar que o algoritmo utilizado pela Hydra é capaz de coletar informações de maneira satisfatória mesmo com um nodo gerando uma quantidade excessiva de informações. Também é possível observar que, devido à forma que as informações são coletadas, o número de transmissões e tempo

Tabela 2 – Comparação de resultados de simulações

	Número de transmissões	Tempo escutando (s)	Informações coletadas	Informações geradas	Taxa de entrega (%)
Hydra	3057	6130,75	1019	1034	98,5
Flooding	1433	7641,75	333	1034	32,2
Hydra (sobrecarga)	3057	6130,75	4811	4834	99,5
Flooding (sobrecarga)	10600	5350,00	2099	4834	43,4

Fonte: Autoria própria (2024).

escutando total da rede são constantes. Assim, o consumo de energia para manter a rede em funcionamento também é constante e pode ser reduzido significativamente aumentando o tempo entre ciclos de coleta de informações.

Comparando a Hydra com o trabalho anterior, que usou o algoritmo *flooding*, é possível observar que a taxa de entrega de mensagens aumentou de maneira significativa. Também, em situação de sobrecarga, é possível observar que o consumo energético, que é indicado pelo número de transmissões, aumenta consideravelmente, enquanto a rede Hydra se mantém constante e sem prejudicar sua taxa de entrega.

5.2 Testes com a rede Hydra

5.2.1 Consumo energético

A Figura 42 mostra os nós Hydra montados. Foram montados quatro nós para os testes. Dois nós foram equipados com antenas externas de 2,4GHz para o BLE e dois nós foram utilizados as antenas roteadas.

Figura 42 – Nós Hydra montados

Fonte: Autoria própria (2024).

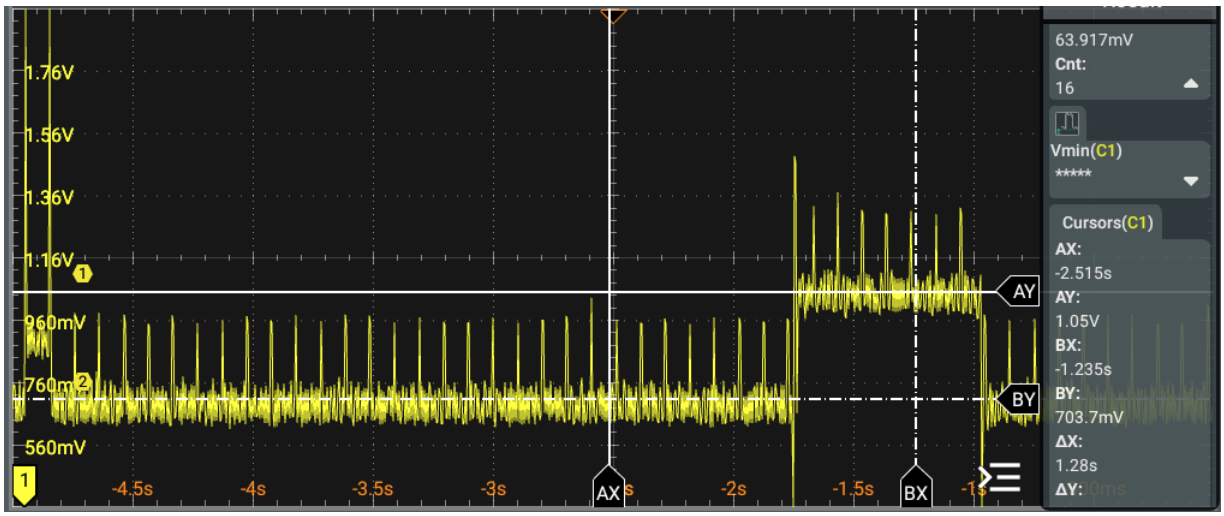
Para a medição da corrente consumida por um nó Hydra, foi utilizado um resistor *shunt* de 5Ω em série com a alimentação pela porta USB. Como pode ser observado na Figura 43, a tensão sobre o *shunt* estava em 703,7mV quando o nó permanecia em estado ocioso e, enquanto transmitia, o valor da tensão sobre o *shunt* subia para 1,05V. Além disso, de acordo com o *datasheet* do ESP-C3(ESPRESSIF SYSTEMS, 2024), quando o módulo entra no modo de *deep sleep*, ele consome apenas 5μA, um valor muito baixo para ser medido com os instrumentos disponíveis. Com estes valores, foi possível montar a Tabela 3, com os valores de consumo de corrente do nó Hydra.

Tabela 3 – Consumo de corrente do nó Hydra

Estado do Nó Hydra	Corrente
Dormindo	5μA
Ocioso	140,74 mA
Transmitindo	210 mA

Fonte: Autoria própria (2024).

Figura 43 – Medição de tensão em um resistor *shunt* em um nó Hydra



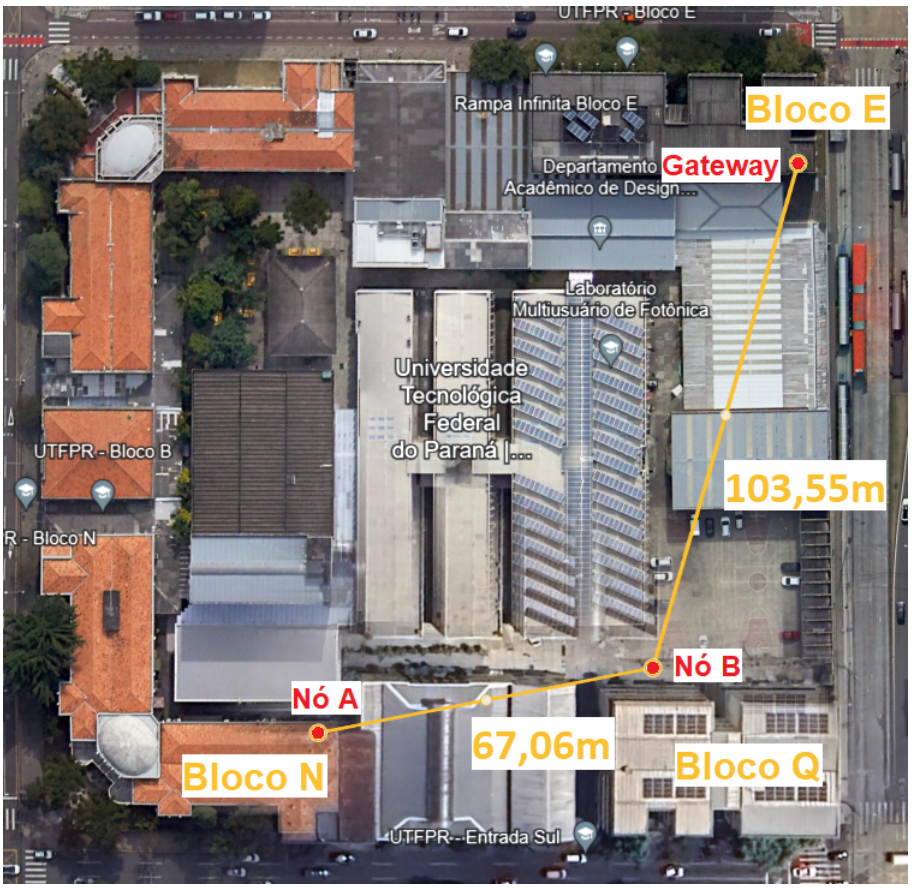
Fonte: Autoria própria (2024).

5.2.2 Teste do protocolo *mesh*

Para validar se o *firmware* do nó estava funcionando corretamente, foi preparado um teste para ser executado no ambiente da UTFPR. Um dispositivo foi configurado como *gateway* e posicionado na sala CE-208. Depois um nó, denominado ”Nó A”, foi posicionado dentro do bloco N, longe o bastante para que não conseguisse comunicação direta com o *gateway*. Por fim, um segundo nó, denominado ”Nó B”, foi posicionado na frente do bloco Q. Ao lado de cada nó, foi ligado um dispositivo BLE que simula envio de medições elétricas da aplicação.

As posições dos nós e do *gateway* podem ser observadas na Figura 44. Assim que o nó B foi ligado, ele mandou uma requisição de JOIN para o *gateway* e entrou na rede. Logo após, o nó A enviou uma requisição de JOIN para o *gateway* através do nó B, que foi aceito. Então, o *gateway* fez uma requisição de informações, primeiro para o nó B, depois para o nó A. Ambos responderam a requisição corretamente e o *gateway* pode fazer a publicação das medições no MQTT *broker*.

Figura 44 – Teste da *mesh* na UTFPR



Fonte: Autoria própria (2024).

5.2.3 Teste da distância

Para fazer uma avaliação da perda de percurso da comunicação entre nós, foi realizado um teste no Parque São Lourenço, em Curitiba. Um *gateway* foi modificado para enviar o valor da RSSI da mensagem recebida no MQTT *broker*. Com um nó alimentado por uma bateria, conectado ao *gateway*, foi medida a RSSI em diversas distâncias. Em todos os casos, o *gateway* possuia linha de visada com o nó. Vale ressaltar que os rádios LoRa estavam configurados para a menor distância possível, com o valor de $SF = 7$. O resultado das medições pode ser verificado na Tabela 4.

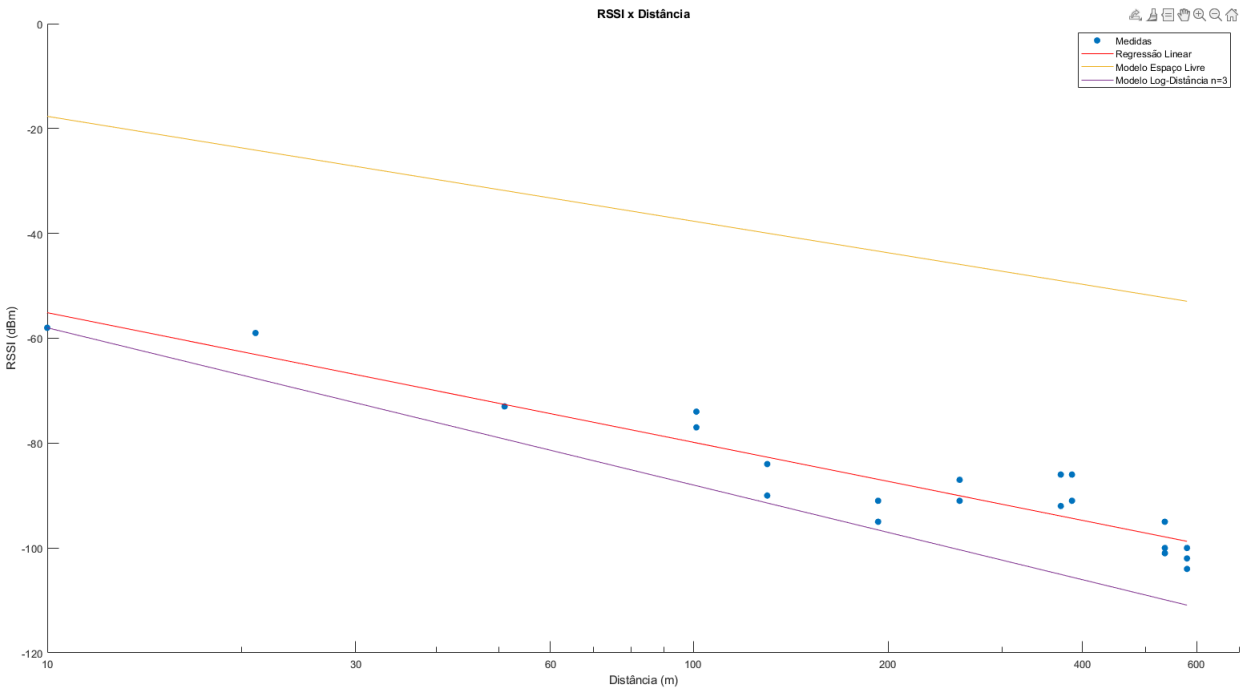
Tabela 4 – Medidas do RSSI por distância na comunicação LoRa

Distância(m)	RSSI(dBm)
10	-58
21	-59
51	-73
101	-74
101	-77
130	-90
130	-84
193	-91
193	-95
258	-87
258	-91
370	-86
370	-92
385	-91
385	-86
536	-101
536	-95
536	-100
580	-100
580	-104
580	-102

Fonte: Autoria própria (2024).

Para fazer uma comparação, também foram calculados os valores teóricos para o modelo de propagação no espaço livre e também o modelo de Log-Distância com o fator $n = 3$. O gráfico com estes resultados, junto às medições reais, pode ser observado na Figura 45.

Figura 45 – Medidas da Distância do LoRa



Fonte: Autoria própria (2024).

Na Figura 45 também pode se observar uma linha vermelha, que representa a regressão linear das medidas, obtendo-se a seguinte equação:

$$PL(dBm) = -0.05d - 71.01 \tag{7}$$

onde d é a distância em metros.

Pelo *datasheet* do rádio LoRa(SEMTECH, 2016), o RSSI mínimo para recepção, quando $SF = 7$, é de -116 dBm. Sendo assim, aplicando na fórmula, pode-se assumir que a distância máxima que poderia ser atingida seria de 899,8 metros.

O mesmo foi feito para testar a distância máxima do BLE. O *gateway* foi programado para enviar ao MQTT o RSSI de um dispositivo BLE, que foi medida em várias distâncias. O resultado das medições pode ser observado na Tabela 5.

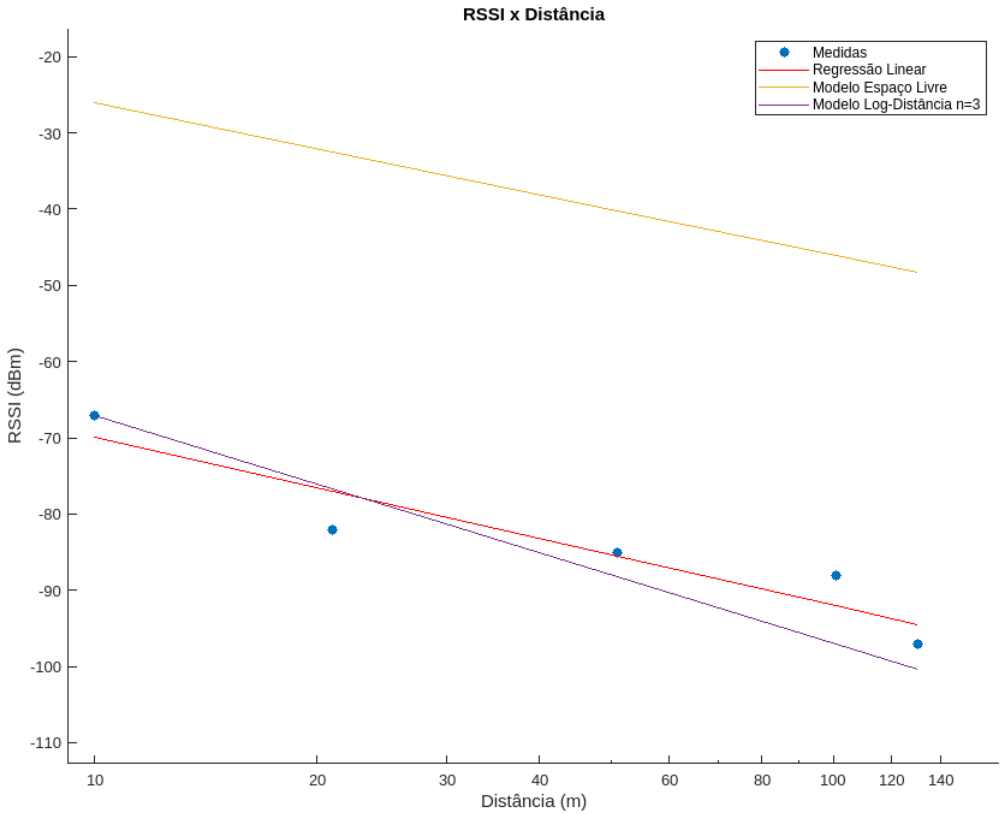
Tabela 5 – Medidas do RSSI por distância na comunicação Bluetooth Low Energy

Distância(m)	RSSI(dBm)
10	-67
21	-82
51	-85
101	-88
130	-97

Fonte: Autoria própria (2024).

Como anteriormente, para fazer uma comparação, também foram calculados os valores teóricos para o modelo de propagação no espaço livre e também o modelo de Log-Distância com o fator $n = 3$. O gráfico com estes resultados, junto às medições reais, pode ser observado na Figura 46.

Figura 46 – Medidas da Distância do BLE



Fonte: Autoria própria (2024).

Na Figura 46 também pode ser observado uma linha vermelha, que representa a regressão linear das medidas, obtendo-se a seguinte equação:

$$PL(dBm) = -0.19d - 72.071 \tag{8}$$

onde d é a distância em metros. Neste caso, após 130 metros, não foi mais possível se conectar com o dispositivo BLE, indicando que esta é a distância máxima me linha de visada.

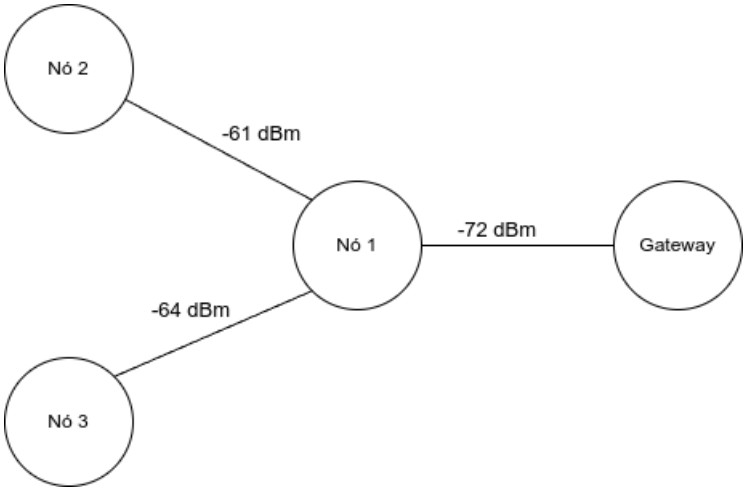
5.3 Teste de estabilidade

Um segundo teste da rede Hydra foi conduzido de forma a verificar a estabilidade da rede. A ideia do teste era avaliar o desempenho da rede por longos períodos de tempo. Desta forma, utilizando três nós e o *gateway* Hydra dentro da casa de um dos membros, onde a rede

ficou funcionando por cerca de 23 horas. A Figura 47 mostra a disposição dos nós na rede, bem como o RSSI entre eles. Os nós foram configurados para um $SF = 9$ e $BW = 500kHz$.

Para uma comparação com o simulador, a mesma configuração de rede da Figura 47 foi simulada, de modo a obter uma noção dos resultados e avaliar a efetividade do simulador.

Figura 47 – Teste de estabilidade da *mesh* Hydra



Fonte: Autoria própria (2024).

Os resultados podem ser observados na Tabela 6. Na rede Hydra real, durante todo o período de 23 horas, foram enviadas 12549 requisições de informações para os nós, das quais 12114 foram respondidas, totalizando uma eficácia de 96,52%.

Existem alguns pontos interessantes em relação ao teste. Observando o histórico de mensagens, notou-se que, durante a noite, o Nó 2 se desconectou da rede devido à falta de alimentação, mas não teve problemas em juntar-se a rede novamente e responder as requisições do *gateway*. Além disso, por um erro no código, o tempo de espera do nó para desistir da rede era menor que o tempo do *gateway* de reenviar as requisições, aumentando as falhas de comunicação. Mesmo com os eventos adversos, a rede conseguiu se regenerar e continuar o teste normalmente.

Quanto ao simulador, dos 14325 *queries* enviados, foram recebidos 14324, equivalente a uma eficácia de 99,99%. A simulação foi executada com o equivalente a 23h de funcionamento. Vale ressaltar que o *query* perdido ocorreu pois a simulação foi finalizada antes do recebimento da resposta do último query.

Tabela 6 – Comparação de resultados de simulações

	***Queries* recebidos**	***Queries* executados**	**Taxa sucesso**
Simulador	14324	14325	99,99%
Rede real	12114	12549	96,52%

Fonte: Autoria própria (2024).

5.4 Discussões

Ao final do projeto, com os *hardwares* prontos e os testes realizados, foi possível fazer uma análise mais aprimorada dos objetivos iniciais e seus resultados. Antes disso, vale ressaltar o grande desafio que foi implementar e testar uma rede *mesh* utilizando um *hardware* personalizado.

Quaisquer mudanças no *firmware* geravam a necessidade de se reprogramar todos os nós e, para a realização de testes realistas da rede, esses nós precisavam estar separados por uma distância física considerável, complicando bastante a iteração e testes dos *firmwares*.

Mesmo na configuração de rádio que limita o seu alcance, a distância entre o nó e o *gateway* onde a comunicação apresentava desempenho insatisfatório é da ordem de centenas de metros. Isto tornava pouco prático deslocar um segundo nó para uma região que esteja fora do alcance do *gateway*, mas ao alcance do primeiro nó. Se adicionarmos mais nós nesta lista, o trabalho fica múltiplas vezes mais custoso. Isto ainda considerando que modificações no *firmware* tenham sido bem-sucedidas pois, caso contrário, todos os nós deveriam ser levados ao ambiente de desenvolvimento para serem reprogramados.

Sobre o novo algoritmo *mesh*, os resultados foram bastante satisfatórios. Em comparação ao *flooding*, há um controle muito maior sobre os processos da rede, devido ao *gateway* iniciar todo o processo de comunicação. Apesar de adicionar uma grande latência ao processo de consulta dos dados dos nós, isto permite a criação de mecanismos de sincronização por tempo e também permite armazenamento da topologia completa da rede no *gateway*. Além disso, geram-se muito menos transmissões desnecessárias no algoritmo *mesh* da Hydra que no *flooding*, tornando a solução mais eficiente energicamente.

No que se diz respeito ao simulador, é preciso, em primeiro lugar, avaliar a sua finalidade. A construção de um simulador de redes em que se avaliassem todos os fatores e nuanças que compõe um sistema de comunicações real é impraticável no escopo deste trabalho. Tais fatores - como a interferência, desvanecimento de canais, atenuação de sinal e perdas de pacote - exigiriam a implementação de modelos matemáticos complexos e estudos teóricos que demandariam muito mais tempo que o disponível. Isto posto, o simulador compreende uma parcela pequena, mas significativa, do sistema de comunicação Hydra: o algoritmo de roteamento *mesh*. Com o simulador, foi possível averiguar as vantagens da *mesh* Hydra em relação ao *flooding* e também abre espaço para, em trabalho futuros, avaliar outros algoritmos para outras aplicações. Dadas as dificuldades encontradas na implementação do *firmware*, outro trabalho futuro de grande utilidade seria a implementação de uma maneira de simular a interface dos rádios via simulador, de modo que fosse possível utilizar as funções implementadas dos nós com uma rede simulada. Isto não só facilitaria a depuração do *firmware* como também permitiria que fosse simulada a distância entre os nós sem a necessidade de movê-los fisicamente.

Em relação ao trabalho anterior da equipe, descrito no Capítulo 3, pode-se afirmar que a Hydra é uma grande evolução. Durante os testes de funcionamento da *mesh*, a Hydra se

mostrou muito mais eficaz: no ambiente da UTFPR, enquanto os nós com o algoritmo *flooding* precisavam enviar os pacotes várias vezes para que a mensagem chegasse ao destino, os nós Hydra, uma vez estabelecida conexão com a rede, perdiam poucos pacotes e o *gateway* conseguia fazer o *query* das informações sem grandes dificuldades.

Por fim, é importante ressaltar a semelhança dos resultados obtidos com o simulador e os resultados obtidos experimentalmente com a rede Hydra montada fisicamente. Mesmo com todas as interferências do mundo real, a rede Hydra obteve um resultado de 96,52%, muito próximo dos 99,99% obtidos pelo simulador.

6 CONCLUSÃO

Os resultados obtidos ao longo deste trabalho foram bastante satisfatórios. O desenvolvimento de um ecossistema IoT partindo do zero proporcionou uma oportunidade única de aplicar diversos conhecimentos adquiridos ao longo do curso de Engenharia Eletrônica. A efetiva implementação da rede não apenas consolidou tais conhecimentos, como gerou uma solução viável de comunicação sem fio.

Ao comparar-se com o trabalho ”Medição e Controle de Sistemas de Potência”, desenvolvido anteriormente pelos autores, a rede Hydra é uma solução muito mais robusta. A adição da comunicação Bluetooth *Low Energy* ao nó possibilita aumentar o número de medidores, uma que vez que os custos de produção e manutenção dos medidores é reduzido. Além disso, permite que a aplicação seja pivotada para outras áreas que não sejam a de medição de potência, tendo em vista que outros tipos de sensores ou atuadores BLE são facilmente adicionadas no nó. Desta forma, a rede Hydra se torna uma solução genérica para os mais diversos campos, de medicina à agricultura, por exemplo.

Um ponto de melhora foi o algoritmo *mesh* utilizado. Em comparação ao *flooding*, o algoritmo *mesh* do protocolo da Hydra gera muito menos transmissões desnecessárias para coletar informações dos sensores. Isto gera uma economia de consumo, visto que transmissões são custosas energicamente, e também diminui a chance de congestionamento com o aumento do número de nós. Além disso, por conta do *gateway* ser sempre quem inicia a comunicação, o algoritmo permite um maior controle sobre a rede. Por exemplo, em uma aplicação em que se necessita fazer a varredura dos sensores apenas uma vez por dia, podem ser adicionados sincronizações por tempo para que todos os nós entrem em hibernação, acordando apenas no horário que o *gateway* varre os sensores, economizando energia.

O desenvolvimento da Hydra também originou um simulador de redes *mesh*. Através dele, foi possível comprovar a eficácia do algoritmo *mesh* da Hydra e verificar suas vantagens em relação ao *flooding*. Ainda, o simulador pode ser usado para analisar outros algoritmos e compará-los entre si.

Para trabalhos futuros, existem muitos aspectos da solução proposta que podem ser aprimorados. A implementação focou em um protótipo minimamente praticável e, assim, não foram implementados algumas características indispensáveis para um rede IoT comercial. Por exemplo, não foram exploradas nenhuma implementação de uma camada de segurança.

O algoritmo da *mesh* é outra área que ainda pode ser bastante explorada. Nossa rede atualmente atua em uma única frequência central e com parâmetros LoRa fixos. Melhorias nessa área incluem explorar o uso de múltiplas frequências e canais ortogonais, o que garantiria uma maior resiliência a rede e facilitaria a coexistência com outras redes que podem estar usando o meio.

REFERÊNCIAS

ADELANTADO, F. *et al.* Understanding the limits of lorawan. **IEEE Communications Magazine**, v. 55, n. 9, p. 34–40, 2017.

ALLEGRO MICROSYSTEMS. **ACS712**: Fully integrated, hall effect-based linear current sensor with 2.1 kvrms voltage isolation and a low-resistance current conductor. [*S.l.*], 2007. Revision 7. Disponível em: https://www.sparkfun.com/datasheets/BreakoutBoards/0712.pdf. Acesso em: 26 maio 2024.

ALLIANCE, C. S. **Zigbee**. [*s.n.*], 2024. Disponível em: https://csa-iot.org/all-solutions/zigbee/. Acesso em: 01 fev. 2024.

ALLIANCE, W.-F. **Wi-fi**. [*s.n.*], 2024. Disponível em: https://www.wi-fi.org/. Acesso em: 05 mar. 2024.

Amazon. **Amazon Sidewalk**. 2024. Online. Disponível em: https://www.amazon.com/Amazon-Sidewalk/b?ie=UTF8&node=21328123011. Acesso em: 30 maio 2024.

ANASTASI, G. *et al.* Energy conservation in wireless sensor networks: A survey. **Ad Hoc Networks**, v. 7, n. 3, p. 537–568, 2009. ISSN 1570-8705. Disponível em: https://www.sciencedirect.com/science/article/pii/S1570870508000954.

BUYYA, R.; DASTJERDI, A. V. **Internet of Things**: Principles and paradigms. [*S.l.*]: Elsevier, 2016. ISBN 9780128053959.

CÄSAR, M. *et al.* A survey on bluetooth low energy security and privacy. **Computer Networks**, Elsevier BV, v. 205, p. 108712, mar. 2022. ISSN 1389-1286. Disponível em: http://dx.doi.org/10.1016/j.comnet.2021.108712.

Cisco. **Antenna Patterns and Their Meaning**. California, EUA, 2007. Disponível em: https://www.industrialnetworking.com/pdf/Antenna-Patterns.pdf. Acesso em: 10 abril 2024.

CORPORATION, S. **Semtech**. [*s.n.*], 2024. Disponível em: https://www.semtech.com/. Acesso em: 17 mar. 2024.

ESPRESSIF SYSTEMS. **ESP32-C3 Series**. [*S.l.*], 2024. Version 1.6. Disponível em: https://www.espressif.com/sites/default/files/documentation/esp32-c3_datasheet_en.pdf. Acesso em: 23 mar. 2024.

FOROUZAN, B. A. **Comunicação de Dados e Redes de Computadores**. 4. ed. [*S.l.*]: AMGH, 2010. ISBN 9788563308474.

FOUNDATION, P. S. **Python**. [*s.n.*], 2024. Disponível em: https://www.python.org/. Acesso em: 15 maio 2024.

FreeRTOS. **FreeRTOS - Market leading RTOS (Real Time Operating System) for embedded systems with Internet of Things extensions**. 2024. Online. Disponível em: https://freertos.org/index.html. Acesso em: 6 junho 2024.

GOMEZ, C.; OLLER, J.; PARADELLS, J. Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. **Sensors**, MDPI AG, v. 12, n. 9, p. 11734–11753, ago. 2012. ISSN 1424-8220. Disponível em: http://dx.doi.org/10.3390/s120911734.

GSMA. **NB-IoT**. [*s.n.*], 2024. Disponível em: https://www.gsma.com/iot/narrow-band-internet-of-things-nb-iot/. Acesso em: 05 mar. 2024.

HEERMICR. **HE9073 0.3uA IQ Series High Speed Low Noise LDO**. [*S.l.*], 2020. Version 1.2. Disponível em: https://www.lcsc.com/datasheet/lcsc_datasheet_2201242130_HEERMICR-HE9073A33MR_C723793.pdf. Acesso em: 30 mar. 2024.

HEYDON, R. **Bluetooth Low Energy**: The developer's handbook. [*S.l.*]: Prentice Hall, 2013. ISBN 978-0-13-288836-3.

INC., I. C. **Iridium**. [*s.n.*], 2024. Disponível em: https://www.iridium.com/. Acesso em: 01 fev. 2024.

INGENU. **Ingenu**. [*s.n.*], 2024. Disponível em: https://www.ingenu.com/. Acesso em: 05 fev. 2024.

JLCPCB. **JLCPCB**. [*s.n.*], 2024. Disponível em: https://jlcpcb.com/. Acesso em: 30 mar. 2024.

KAVRE, M.; GADEKAR, A.; GADHADE, Y. Internet of things (iot): A survey. *In*: **2019 IEEE Pune Section International Conference (PuneCon)**. [*S.l.*: *s.n.*], 2019. p. 1–6.

K.C, K. Wireless mesh network: A survey. *In*: **2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)**. [*S.l.*: *s.n.*], 2016. p. 1966–1970.

KICAD. **KiCad EDA**. [*s.n.*], 2024. Disponível em: https://www.kicad.org/. Acesso em: 30 mar. 2024.

KRAUS, J. D. **Antennas**. [*S.l.*]: McGraw-Hill, 1950. ISBN 070354103.

KUHLINS, C. *et al.* **Cellular networks for Massive IoT**. [*S.l.*], 2020. Disponível em: https://www.ericsson.com/48ff1f/assets/local/reports-papers/white-papers/massive_iot_whitepaper.pdf. Acesso em: 05 mar. 2024.

KUROSE, J. F.; ROSS, K. W. **Computer Networking**: A top-down approach. 7. ed. [*S.l.*]: Pearson, 2017. ISBN 9780133594140.

LEE, H.-C.; KE, K.-H. Monitoring of large-area iot sensors using a lora wireless mesh network system: Design and evaluation. **IEEE Transactions on Instrumentation and Measurement**, v. 67, n. 9, p. 2177–2187, 2018.

LilyGO. **LoRa32 V2.1_1.6**. 2024. Disponível em: https://www.lilygo.cc/products/lora3. Acesso em: 26 maio 2024.

METHLEY, S. **Essentials of Wireless Mesh Networking**: Cambridge wireless essentials series. [*S.l.*]: Cambridge University Press, 2009. ISBN 052187680X,9780521876803.

MQTT. **MQTT**. [*s.n.*], 2022. Disponível em: https://mqtt.org/. Acesso em: 19 maio 2024.

NIKOUKAR, A. *et al.* Empirical analysis and modeling of bluetooth low-energy (ble) advertisement channels. *In*: **2018 17th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)**. [*S.l.*: *s.n.*], 2018. p. 1–6.

Nordic Semiconductor. **Bluetooth Low Energy Fundamentals**. 2024. Online. Disponível em: https://academy.nordicsemi.com/courses/bluetooth-low-energy-fundamentals/lessons/lesson-1-bluetooth-low-energy-introduction. Acesso em: 25 maio 2024.

OPENWEIGHTLESS, C. **Weightless**. [*s.n.*], 2024. Disponível em: https://www.openweightless.org/. Acesso em: 05 fev. 2024.

PATEL, D. Low power wide area networks (lpwan) technology reviews and experimental study on mobility effect. **Electronic Theses and Dissertations**, n. 2667, 2018. Disponível em: https://openprairie.sdstate.edu/etd/2667/. Acesso em: 03 fev. 2024.

PATEL, K. *et al.* Internet of things-iot: Definition, characteristics, architecture, enabling technologies, application & future challenges. **International Journal of Engineering Science and Computing**, v. 6, n. 5, p. 6123–6131, 2016.

QUCS. **Quite Universal Circuit Simulator**. [*s.n.*], 2017. Disponível em: https://qucs.sourceforge.net/. Acesso em: 30 mar. 2024.

RAPPAPORT, T. S. **Wireless Communications**: Principles and practice. [*S.l.*]: Prentice Hall, 2002. ISBN 978-0130422323.

RASPBERRYPI LTD. **Raspberry Pi 3 Model B+**. [*S.l.*], 2023. Disponível em: https://datasheets.raspberrypi.com/rpi3/raspberry-pi-3-b-plus-product-brief.pdf. Acesso em: 23 mar. 2024.

RAZA, U.; KULKARNI, P.; SOORIYABANDARA, M. Low power wide area networks: An overview. **IEEE Communications Surveys & Tutorials**, v. 19, n. 2, p. 855–873, 2017.

SCHOLTZ, R. The origins of spread-spectrum communications. **IEEE Transactions on Communications**, v. 30, n. 5, p. 822–854, 1982.

Semtech. **AN1200.22 LoRa™ Modulation Basics**. [*S.l.*], 2015. Disponível em: https://www.frugalprototype.com/wp-content/uploads/2016/08/an1200.22.pdf. Acesso em: 05 mar. 2024.

SEMTECH. **SX1276/77/78/79 - 137 MHz to 1020 MHz Low Power Long Range Transceiver**. [*S.l.*], 2016. Disponível em: https://semtech.my.salesforce.com/sfc/p/E0000000JelG/a/2R0000001Rbr/6EfVZUorrpoKFfvaF_Fkpgp5kzjiNyiAbqcpqh9qSjE. Acesso em: 23 mar. 2024.

SIG, B. **Bluetooth**. [*s.n.*], 2024. Disponível em: https://www.bluetooth.com/. Acesso em: 01 fev. 2024.

SIGFOX. **Sigfox**. [*s.n.*], 2024. Disponível em: https://www.sigfox.com/. Acesso em: 05 fev. 2024.

SONGLE RELAY. **Relay ISO9002**. [*S.l.*], 2007. Disponível em: https://www.circuitbasics.com/wp-content/uploads/2015/11/SRD-05VDC-SL-C-Datasheet.pdf. Acesso em: 26 maio 2024.

SUNDARAM, J. P. S.; DU, W.; ZHAO, Z. A survey on lora networking: Research problems, current solutions and open issues. **IEEE Communications Surveys & Tutorials**, PP, p. 1–1, 10 2019.

SYSTEMS, E. **Espressif**. [*s.n.*], 2024. Disponível em: https://www.espressif.com/. Acesso em: 23 mar. 2024.

TEXAS INSTRUMENTS. **LMx58-N Low-Power, Dual-Operational Amplifiers**. [*S.l.*], 2022. Disponível em: https://www.ti.com/lit/ds/symlink/lm158-n.pdf. Acesso em: 26 maio 2024.

The Things Network. **The Things Network**. 2024. Online. Disponível em: https://www.thethingsnetwork.org/. Acesso em: 30 maio 2024.

Transforma Connections. **Current IoT Forecast Highlights**. 2024. Online. Disponível em: https://transformainsights.com/research/forecast/highlights. Acesso em: 30 maio 2024.

VOLLGO TECHNOLOGY. **SX1276SXS+T-X1 Wireless Module**. [*S.l.*], 2024. Disponível em: https://xonstorage.blob.core.windows.net/pdf/vollgo_sx1276s9stx1_xonjuly20_20_nocatlink.pdf. Acesso em: 23 mar. 2024.

WOOLLEY, M. **Bluetooth® Core Specification Version 5.0 Feature Enhancements**. [*S.l.*], 2021. Disponível em: https://www.bluetooth.com/wp-content/uploads/2019/03/Bluetooth_5-FINAL.pdf. Acesso em: 30 maio 2024.

YUAN, M. **Getting to know MQTT**. [*s.n.*], 2021. Disponível em: https://developer.ibm.com/articles/iot-mqtt-why-good-for-iot/.

ZHANG, Y.; LUO, J.; HU, H. **Wireless Mesh Networking**: Architectures, protocols and standards. [*S.l.*]: Auerbach Publications, 2006. ISBN 0849319137, 084932212X, 0849329604,0849331579.