UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
PROGRAMA DE PÓS-GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Otimização na transmissão de dados em dispositivos IoT com foco na economia de energia.

Audrei Silva

Itajubá, 14 de abril de 2024

UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
PROGRAMA DE PÓS-GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Audrei Silva

Otimização na transmissão de dados em dispositivos IoT com foco na economia de energia.

Dissertação submetida ao Programa de Pós-Graduação em Ciência e Tecnologia da Computação como parte dos requisitos para obtenção do Título de Mestre em Ciência e Tecnologia da Computação.

Área de Concentração: Hardware e software básico

Orientador: Prof. Dr. Rodrigo M. A. de Almeida

14 de abril de 2024
Itajubá

UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
PROGRAMA DE PÓS-GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Otimização na transmissão de dados em dispositivos IoT com foco na economia de energia.

Audrei Silva

Dissertação aprovada por banca examinadora em 16 de Fevereiro de 2024, conferindo ao autor o título de **Mestre em Ciências em Engenharia Elétrica.**

Banca Examinadora:
Prof. Dr. Rodrigo Maximiano Antunes de Almeida
Prof. Dr. Carlos Henrique Valério de Moraes
Prof. Dr. Marcelo Barros de Almeida

Itajubá
2024

Audrei Silva
Otimização na transmissão de dados em dispositivos IoT com foco na economia de energia/ Audrei Silva. – Itajubá, 14 de abril de 2024-
82 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Rodrigo M. A. de Almeida

Dissertação (Mestrado)
Universidade Federal de Itajubá - UNIFEI
Programa de pós-graduação em Ciência e Tecnologia da Computação, 14 de abril de 2024.

1. Palavra-chave1. 2. Palavra-chave2. I. Orientador. II. Universidade xxx. III. Faculdade de xxx. IV. Título

CDU 07:181:009.3

Audrei Silva

Otimização na transmissão de dados em dispositivos IoT com foco na economia de energia

Dissertação submetida ao Programa de Pós-Graduação em Ciência e Tecnologia da Computação como parte dos requisitos para obtenção do Título de Mestre em Ciência e Tecnologia da Computação.

Trabalho aprovado. Itajubá, 16 de Fevereiro de 2024:

Prof. Dr. Rodrigo M. A. de Almeida
Orientador

Prof. Dr. Carlos Henrique Valério de Moraes

Prof. Dr. Marcelo Barros de Almeida

Itajubá
14 de abril de 2024

Agradecimentos

Primeiramente, gostaria de expressar minha profunda gratidão a Deus, por me permitir o dom da vida e sempre me guiar com sabedoria e paciência ao longo da vida, iluminando meus passos e fortalecendo meu espírito em todos os momentos.

Aos meus pais, Maria e José, pelo amor incondicional. Embora não tenham tido muitas oportunidades de estudo, sempre me incentivaram e foram exemplos com sua sabedoria, além de terem plantado em mim os valores de dedicação e perseverança. Eles foram essenciais para que eu alcançasse este marco em minha vida.

Um agradecimento especial à minha amada esposa, Bruna, por sempre acreditar no meu potencial. Seu amor e compreensão foram fundamentais para a realização deste trabalho. Sua presença foi a força que me impulsionou a seguir adiante, mesmo diante das dificuldades. Agradeço também ao meu filho João, que está prestes a vir ao mundo e completará ainda mais a nossa família.

Agradeço também aos meus irmãos, Elisângela, Reginaldo e Valdinei, por estarem sempre ao meu lado, oferecendo palavras de incentivo.

Aos meus colegas de trabalho e amigos, expresso minha gratidão pelos conselhos e pelos momentos de descontração, que foram essenciais para manter meu equilíbrio durante esta jornada.

Um reconhecimento especial ao meu orientador, Rodrigo, cuja paciência e orientação foram cruciais para o desenvolvimento e conclusão deste trabalho. Sua sabedoria, incentivo e os conhecimentos compartilhados foram fundamentais para o meu crescimento acadêmico e profissional.

"Na simplicidade, encontre a força. Na fé, encontre a esperança. Na caridade, encontre o amor."(Santa Rita de Cássia)

Resumo

O surgimento da Internet das Coisas viabilizou a conexão e o compartilhamento de informações entre objetos, possibilitando a execução de tarefas e a interação com outros dispositivos, com o intuito de melhorar a qualidade de vida das pessoas em diversas aplicações. Contudo, essa evolução tecnológica ainda enfrenta diversos desafios, entre eles a eficiência energética, sobretudo em dispositivos de borda, como os sensores, que em sua maioria, operam com alimentação de baterias e são instalados em regiões remotas de difícil acesso. Diante dessa realidade, propõe-se a criação de um driver destinado a gerenciar e otimizar a transmissão de dados em dispositivos da Internet das Coisas, com foco na economia de energia. Esse propósito é alcançado por meio da aplicação da técnica de agrupamento de dados, a qual permite a redução da frequência de ativação do rádio e a minimização do impacto do *overhead*, essencial nos protocolos de comunicação. A eficácia dessa solução é avaliada por meio de sua implementação prática.

De modo geral, observou-se concordância entre os resultados teóricos e experimentais. Contudo, para protocolos orientados à conexão, como no caso do Wi-Fi, a eficiência em termos de sobrecarga é comprometida pelas latências da rede. Mesmo nesse cenário, com a utilização do driver o consumo de corrente é 12,82% menor, além de aumentar a confiabilidade do sistema e evitar a transmissão de dados redundantes desnecessários.

Palavras-chaves: Internet das Coisas; Sistemas Embarcados; Economia de energia; Dispositivos de Baixo Consumo.

Abstract

The advent of the Internet of Things facilitates the connection and sharing of information between objects, enabling the performance of tasks and interaction with other devices, with the purpose of improving people's quality of life across various applications. However, this technological evolution still faces numerous challenges, among them energy efficiency, particularly in edge devices such as sensors, which largely operate on battery power and are installed in remote, difficult-to-access areas. Given this reality, it is proposed to create a driver aimed at managing and optimizing data transmission in IoT devices, with a focus on energy savings. This objective is achieved through the application of data clustering techniques, which allows for the reduction of radio activation frequency and minimizes the impact of overhead, crucial in communication protocols. The effectiveness of this solution is assessed through its practical implementation.

In general, there was agreement between the theoretical and experimental results. However, for connection-oriented protocols, such as Wi-Fi, efficiency in terms of overhead is compromised by network latencies. Even in this scenario, with the use of the driver, the current consumption is reduced by 12.82%, in addition to increasing the system's reliability and avoiding unnecessary redundant data transmission.

Key-words: Internet of Things; Embedded systems; Energy Save, Low-Power Devices.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AMQP	*Advanced Message Queuing Protocol*	70
ANT	*Adaptive Network Topology*	73
BLE	*Bluetooth Low Energy*	36
CoAP	*Constrained Application Protocol*	70
CPU	*Central Processing Unit*	27
CRC	*Cyclic Redundancy Check*	38
CSMA/CA	*Carrier Sense Multiple Access with Collision Avoidance*	74
DCPS	*Data Centric Publisher Subscriber*	72
DDS	*Data Distribution Service*	70
DPCM	*Differential pulse-code modulation*	42
DSSS	*Direct Sequence Spread Spectrum*	74
DTLS	*Datagram Transport Layer Security*	70
DWT	*Transformada Wavelet Discreta*	42
FFD	*Full Function Device*	74
GPS	*Global Positioning System*	34
HTTP	*Hypertext Transfer Protocol*	71
HTTPS	*Hypertext Transfer Protocol Secure*	71
IDE	*integrated development environment*	48
IoT	*Internet Of Things*	17
IP	*Internet Protocol*	24
IPv4	*Internet Protocol version 4*	24
IPv6	*Internet Protocol version 6*	24
ISA	*Instruction Set Architecture*	28
ISO	*Organization for Standardization*	21
LoRa	*Long Range*	36
LoRaWAN	*Long Range Wide Area Network*	10
LTE	*Long-Term Evolution*	33
MAC	*Media Access Control*	38
MIC	*Message Integrity Code*	38
MQTT	*Message Queuing Telemetry Transport*	70
MTU	*Maximum Transmission Unit*	26
mutex	*Mutual exclusion*	53
NFC	*Near Field Communication*	73
OSI	*Open Systems Interconnection*	19
PANs	*Personal Area Networks*	73

PDU	*Protocol Data Unit*	26
PIB	*Produto Interno Bruto*	33
RDC	*Radio Duty-Cycling*	40
RFD	*Reduced Function Device*	74
RFID	*Radio Frequency Identification*	33
RTOS	*Real Time Operating System*	30
RTOSs	*Real Time Operating Systems*	30
RTPS	*Real-Time Publisher Subscriber*	72
SDK	*Software Development Kit*	45
SO	*Sistema Operacional*	28
SSL	*Secure Socket Layer*	70
TCP	*Transmission Control Protocol*	25
TLS	*Transport Layer Security*	71
ToA	*Time on Air*	49
UDP	*User Datagram Protocol*	70
USB	*Universal Serial Bus*	46
WAMP	*Web Application Messaging Protocol*	72
Wi-Fi	*Wireless Fidelity*	27

Sumário

1 Introdução

Este capítulo tem como objetivo introduzir o tema abordado neste trabalho, expondo os objetivos, as principais contribuições e a relevância do estudo em análise. Ao final, é apresentada a organização estrutural do documento, detalhando a sequência dos tópicos e a maneira como as informações são distribuídas ao longo do texto.

1.1 Caracterização do Problema

A Internet das Coisas IoT (*Internet Of Things*) refere-se a uma abordagem que visa conectar diversos dispositivos físicos e virtuais, possibilitando a oferta de serviços avançados por meio de tecnologias de informação e comunicação [1]. Observa-se uma expectativa de crescimento significativo no mercado de IoT, com projeções estimando um valor de mercado de aproximadamente US$ 7,1 trilhões até 2025, além da previsão de cerca de 75 bilhões de dispositivos conectados [2]. O potencial de inovação e conectividade da IoT tem fomentado o surgimento de novas aplicações e serviços em diversos setores da sociedade, incluindo agricultura, saúde, indústria, transporte, entre outros [3].

No entanto, ainda existem diversos desafios a serem superados em termos de arquitetura, segurança, eficiência energética, compatibilidade e conectividade. Especificamente no que se refere à eficiência energética, a limitação da capacidade de bateria dos dispositivos IoT, em conjunto com o aumento do consumo de energia decorrente da expansão da tecnologia, representa um desafio significativo para a longevidade e sustentabilidade das redes [4].

Os dispositivos de borda IoT são utilizados tanto na coleta de informações do ambiente quanto na execução de tarefas específicas, sendo caracterizados por seu baixo custo e recursos limitados. Aspectos como a vida útil da bateria e a capacidade de memória impõem restrições significativas na operação e longevidade desses dispositivos [5]. Esta limitação é particularmente crítica, visto que muitos dispositivos são implantados em áreas remotas ou de difícil acesso, tornando inviável a substituição regular de baterias [6].

Neste contexto, torna-se necessário adotar estratégias eficazes para otimizar o uso da bateria e minimizar o consumo de energia nas operações de sensoriamento e comunicações [7]. A comunicação representa um dos principais fatores de consumo de energia, uma vez que transmitir dados frequentemente implica na ativação do transmissor de rádio, o qual demanda uma quantidade significativa de energia. Outro aspecto relevante é o protocolo de comunicação empregado. O desperdício de energia ocorre frequentemente devido à transmissão de dados desnecessários, sobrecarga dos protocolos e padrões de comunicação

ineficientes. Em dispositivos voltados para sensoriamento, é essencial identificar variações abruptas para permitir a atuação adequada do sistema. Assim, um gerenciamento eficaz dos dados, evitando a transmissão de informações repetitivas, contribui para a redução da frequência de uso do transmissor de rádio.

Portanto, o cenário projetado para os dispositivos IoT consiste em operar em modos de baixa energia quando não estão em processo de transmissão ativa de dados. Em tese, em vez de enviar pacotes pequenos e frequentes, é preferível que os dispositivos agreguem dados e transmitam pacotes maiores com menor frequência. Esta abordagem contribui para a redução do número total de transmissões, além de favorecer a utilização de protocolos de comunicação mais eficientes.

1.2 Objetivos

1.2.1 Objetivo Geral

O principal objetivo desta pesquisa é reduzir o consumo energético associado à transmissão de dados em dispositivos IoT, diminuindo a frequência das transmissões através do agrupamento de dados.

1.2.2 Objetivos Específicos

Os seguintes objetivos específicos foram estabelecidos para esta pesquisa:

- Investigar estudos correlatos, visando avaliar o estado da arte no campo de estudo.
- Realizar um estudo teórico detalhado sobre a inserção de *overhead* em protocolos de comunicação e seu impacto em dispositivos alimentados por bateria.
- Desenvolver um driver portável e confiável para o FreeRTOS, projetado para gerenciar de maneira eficiente os dados a serem transmitidos.
- Implementar um sistema de redução do volume de dados transmitidos, baseando-se na variabilidade entre as medidas.
- Desenvolver um sistema de gestão para avaliar a criticidade das medições realizadas.
- Aplicar o driver desenvolvido em um hardware real e proceder com a coleta de dados.
- Comparar os resultados obtidos com e sem o uso do driver proposto, com o objetivo de demonstrar a eficácia da abordagem em termos de economia de energia.

Estes objetivos específicos estão alinhados ao objetivo geral do estudo, contribuindo significativamente para o avanço do conhecimento na área de economia de energia em sistemas operacionais de tempo real e na sua aplicabilidade em dispositivos IoT alimentados por baterias.

1.3 Contribuições e Relevância da Pesquisa

Uma das principais contribuições deste trabalho é a efetiva redução do consumo energético em dispositivos IoT durante a transmissão de dados, através da diminuição da frequência de transmissões por meio do agrupamento de dados. A criação e implementação de um driver portável e confiável para o FreeRTOS visam esse propósito, gerenciando a criticidade das amostras e permitindo transmissões imediatas em situações atípicas, mas também reduzindo o número de transmissões em cenários típicos de funcionamento. Desta forma, desempenha um papel crucial na gestão eficiente dos dados a serem transmitidos, demonstrando uma estratégia eficaz para otimizar o consumo de energia na transmissão de dados.

Adicionalmente, a aplicação prática do driver em um hardware real, a coleta de dados e análise comparativa constituem uma etapa crucial deste estudo. Esta fase possibilitou a validação empírica da eficácia da abordagem proposta, bem como a comparação dos resultados obtidos com e sem o uso do driver. A demonstração prática do impacto positivo na economia de energia não apenas fortalece a base teórica, mas também evidencia a aplicabilidade real do driver desenvolvido, reforçando, assim, as contribuições desta pesquisa ao campo de sistemas embarcados e IoT.

1.4 Estrutura do Trabalho

Este trabalho está organizado em quatro capítulos. No Capítulo 2, realiza-se uma revisão sobre o modelo de camadas OSI (*Open Systems Interconnection*), abordando os protocolos e a necessidade de sobrecarga para a comunicação. Este capítulo também apresenta um estudo detalhado sobre a Internet das Coisas, contemplando as principais estruturas, as tecnologias e os protocolos atualmente em uso, além de discutir os desafios existentes. Adicionalmente, são expostas as características fundamentais e o princípio de funcionamento do sistema operacional de tempo real FreeRTOS.

No Capítulo 3, classifica-se a metodologia de pesquisa, considerando a abordagem, natureza e objetivos do estudo. Adicionalmente, este capítulo apresenta um cenário específico para dispositivos IoT de baixo consumo e introduz o driver desenvolvido no âmbito desta pesquisa.

No Capítulo 4, é relatada a análise de desempenho do driver proposto, em termos

de eficiência energética a partir da implementação prática. Os resultados coletados são minuciosamente discutidos. Por fim, o Capítulo 5 apresenta as conclusões e sugere direções para trabalhos futuros.

2 Revisão Teórica

Neste capítulo, são apresentados os conceitos teóricos essenciais para a compreensão do desenvolvimento deste trabalho.

Inicialmente, realiza-se uma revisão sobre o modelo de camadas OSI, protocolos e a necessidade de sobrecarga para a comunicação. Na sequência, é apresento um estudo sobre IoT, enfatizando as principais estruturas, protocolos e desafios inerentes. Adicionalmente, são expostas as principais características e o princípio funcional do sistema operacional em tempo real FreeRTOS, amplamente utilizado em sistemas embarcados.

2.1 Protocolos

Um protocolo consiste em um conjunto padronizado de regras para a formatação e processamento de dados. Existe uma ampla variedade de protocolos disponíveis, cada um oferecendo recursos específicos ou combinações de recursos, tornando-os mais adequados para diferentes aplicações. Esses protocolos facilitam a comunicação entre dispositivos, entre dispositivos e gateways, entre dispositivos e nuvem/data center, ou em uma combinação dessas comunicações. Fatores como localização geográfica, consumo de energia, presença de barreiras físicas, segurança e custo são determinantes na escolha do protocolo ideal para cada tipo de implantação [8].

Os sistemas em rede são desenvolvidos utilizando tecnologias em camadas, geralmente se baseiam em um modelo teórico como referência para descrever a comunicação entre as camadas. Um dos modelos mais reconhecidos é o OSI, desenvolvido pela ISO (*Organization for Standardization*) no final da década de 1970.

O modelo OSI representa um padrão aberto que possibilita a comunicação entre sistemas distintos, independentemente da arquitetura. Dessa forma, o modelo OSI atua como um framework essencial para entender e desenvolver uma arquitetura de rede flexível e robusta, que permite a operação conjunta. Este modelo é composto por um total de sete camadas, as quais são detalhadas na Figura 1.

As sete camadas do modelo OSI podem ser categorizadas em três subgrupos: suporte ao usuário nas três camadas superiores; suporte de rede nas três camadas inferiores; e transporte na camada central. Geralmente, as camadas superiores são implementadas via software, enquanto as camadas inferiores em níveis de hardware. O processo de transmissão de dados inicia-se na camada de aplicação e conclui-se na camada de enlace, percorrendo todas as camadas intermediárias [9].

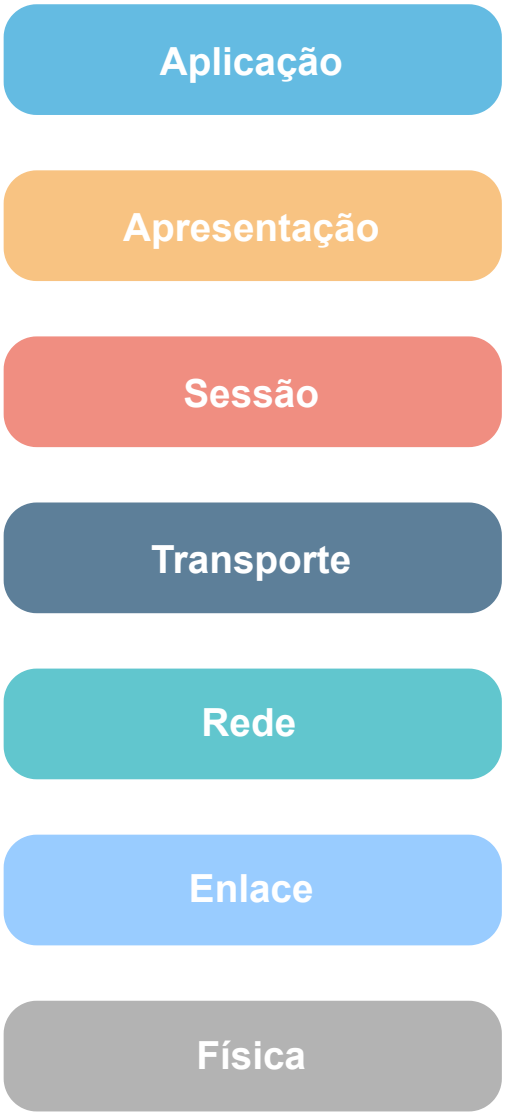

Figura 1 – Camadas do modelo OSI

FONTE: Elaboração Própria

O meio físico empregado para o transporte da informação é conhecido como meio de transmissão. Em sistemas IoT, é comum a utilização de meios de transmissão sem fio, onde a comunicação ocorre por meio de radiofrequência. A Figura 2 ilustra um exemplo de comunicação utilizando esse modelo em camadas. Os dados de D7 referem-se à unidade de dados na camada 7, os dados de D6 à unidade de dados na camada 6, e assim sucessivamente. O processo inicia-se na camada 7, a camada de aplicação, e, em seguida, procede-se à transferência da informação por cada camada, em sequência decrescente. Em cada camada, pode ocorrer a adição de um cabeçalho à unidade de dados. Especificamente na camada 2, é possível também a adição de um trailer. À medida que a unidade de dados formatada atravessa a camada física, camada 1, ela é convertida em um sinal eletromagnético e transmitida por meio físico. Ao alcançar o destino, o sinal é recebido pela camada 1 e convertido novamente em informação digital.

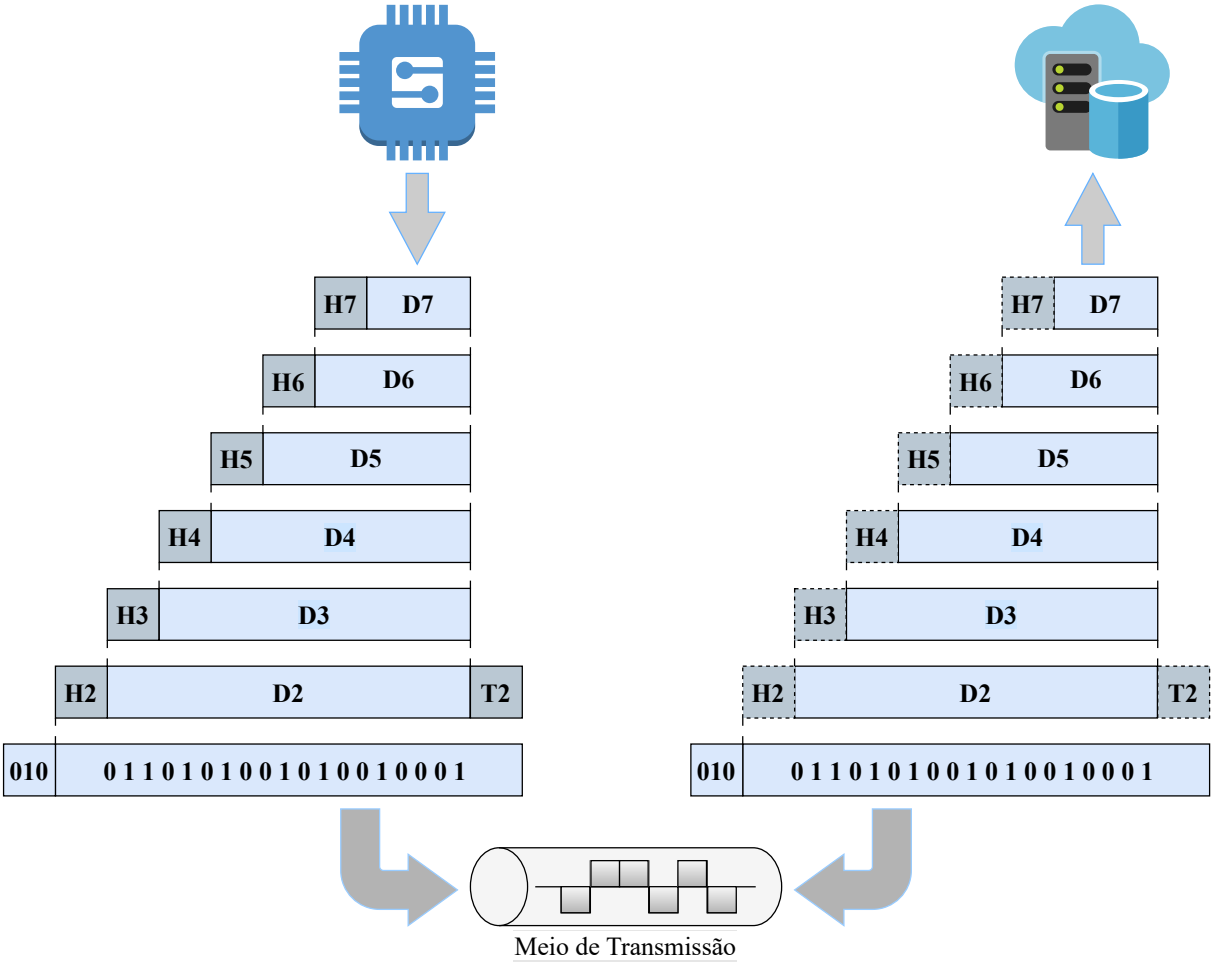

Figura 2 – Comunicação em camadas

FONTE: Adaptado de [9]

Em seguida, as unidades de dados são reencaminhadas pelas camadas subsequentes. À medida que cada bloco de dados alcança a próxima camada superior, os cabeçalhos e trailers adicionados na correspondente camada de envio são removidos, e medidas apropriadas são tomadas. Ao atingir a camada 7, a mensagem é reconstituída em um formato adequado para aplicação.

2.1.1 Sobrecarga de comunicação

A sobrecarga, do termo em inglês *overhead*, refere-se à quantidade de dados adicionais necessários para controlar, gerenciar e transmitir informações em uma rede ou sistema. Embora seja crucial para garantir a eficiência da comunicação, ela também pode impactar negativamente o desempenho do sistema. Em protocolos com pouco ou nenhum *overhead*, a comunicação é mais direta, porém, pode resultar em menor confiabilidade e adaptabilidade a diferentes cenários.

Por outro lado, redes com alta necessidade de *overhead* proporcionam uma comunicação mais robusta e adaptável, embora isso possa implicar em redução da taxa de

dados efetiva e aumento do consumo de recursos. Para dispositivos IoT que operam com baixo consumo energético, o *overhead* representa um desafio significativo devido às suas limitações de recursos. Considerando que a maioria desses dispositivos são alimentados por baterias, a energia é um recurso escasso e qualquer informação adicional nos pacotes de dados implica em consumo extra. Portanto, é essencial encontrar um equilíbrio entre *overhead* e eficiência para otimizar a troca de informações na rede.

2.1.2 Protocolo de Internet

O Protocolo de Internet, ou IP (*Internet Protocol*), é um dos protocolos mais empregados, devido à sua estrutura robusta que desempenha uma papel essencial na comunicação e troca de informações pela internet e em redes de computadores. Atualmente, as versões do protocolo IP em uso são o IPv4 (*Internet Protocol version 4*) e o IPv6 (*Internet Protocol version 6*). O IPv4 é a quarta versão, amplamente utilizada desde o início da internet, mas com um espaço de endereçamento limitado. Com isso, o protocolo IP versão seis, foi desenvolvido para atender, principalmente, às exigências de endereçamento.

A estrutura típica de um pacote IPv4 sem carga útil é ilustrada na Figura 3. O cabeçalho de um pacote IP, excluindo o campo de opções, possui um tamanho mínimo de 20 bytes. Essas cinco linhas, cada uma contendo 32 bits, são componentes obrigatórios em qualquer pacote IPv4.

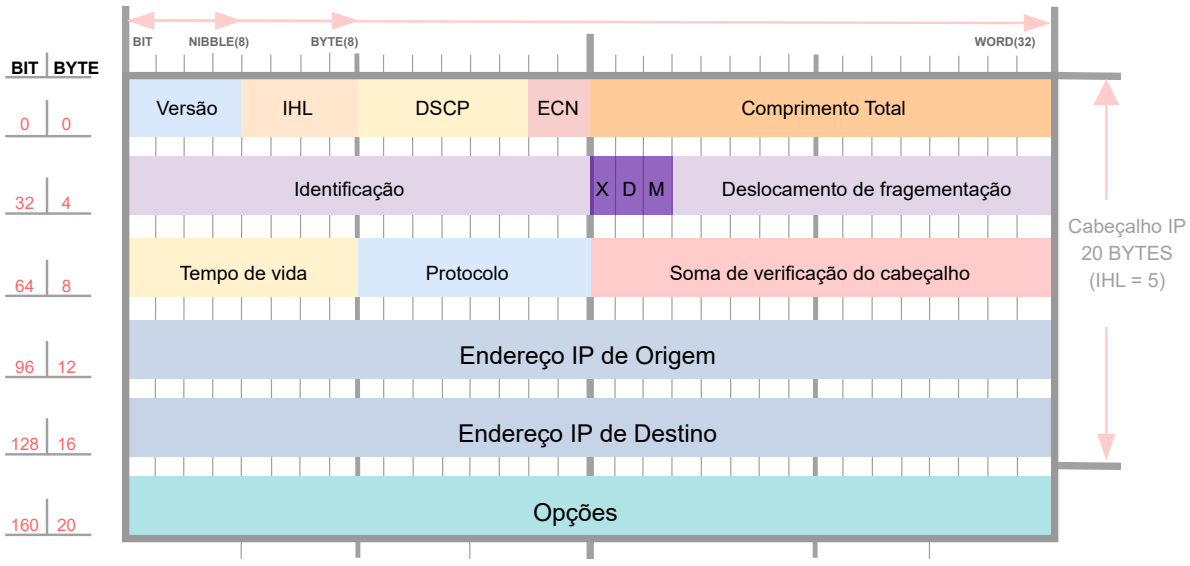

Figura 3 – Cabeçalho IPv4

FONTE: Adaptado de [10]

Os campos do cabeçalho IPv4 incluem a versão do protocolo, o tamanho do cabeçalho, o tipo de serviço, o comprimento total do pacote, a identificação, as flags de fragmentação, o deslocamento do fragmento, o tempo de vida, o protocolo da camada superior, o valor de verificação ou *checksum* do cabeçalho, além dos endereços IP de origem

e destino. Estes campos são essenciais para que os roteadores identifiquem os dispositivos finais e determinem a rota adequada para a entrega correta dos dados.

2.1.3 Protocolo de Controle de Transmissão

O protocolo TCP (*Transmission Control Protocol*) representa uma opção altamente confiável para a transmissão de dados, sendo comumente utilizado em conjunto com o protocolo IP. Essa combinação é eficaz, pois atende às diversas demandas da comunicação em rede. O IP é encarregado do roteamento e encaminhamento dos pacotes entre os dispositivos, enquanto o TCP é responsável pela fragmentação, retransmissão e reagrupamento destes pacotes, garantindo uma transmissão em sequência coerente e livre de erros.

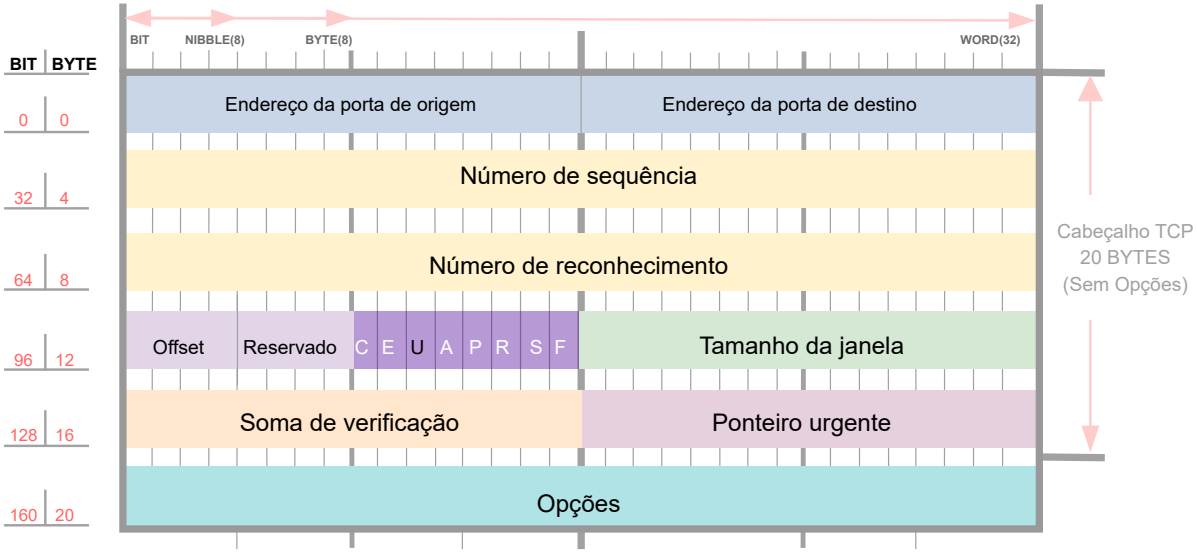

Figura 4 – Cabeçalho TCP

FONTE: Adaptado de [10]

A Figura 4 ilustra a estrutura típica de um cabeçalho TCP. O cabeçalho TCP possui um tamanho mínimo de 20 bytes e pode alcançar até 60 bytes, sendo 20 bytes obrigatórios e até 40 bytes opcionais. Dentre os campos mais relevantes, tem-se os endereços de porta de origem e destino, que identificam as aplicações envolvidas; os números de sequência e de reconhecimento, assegurando a ordem correta e a confiabilidade na entrega dos dados; os flags de controle, como SYN, ACK e FIN, que indicam o início, a confirmação e o término das conexões; o tamanho da janela de recepção, regulando o fluxo de dados entre os dispositivos; o *checksum*, responsável pela verificação da integridade dos dados no cabeçalho; e o campo de opções, utilizado para incluir informações adicionais e ajustar parâmetros da conexão.

2.1.4 Comunicação Ethernet

No contexto do modelo TCP/IP, a Unidade de Dados de Protocolo, ou PDU (*Protocol Data Unit*), representa a unidade básica de informação. Durante o processo de comunicação, a PDU atravessa diversas camadas, sendo em cada uma delas encapsulada com cabeçalhos específicos da respectiva camada, inserindo assim dados de controle e identificação. Na camada de aplicação, a PDU é conhecida como mensagem ou dados da aplicação. Quando avança para a camada de transporte, ela é denominada segmento, recebendo um cabeçalho TCP. Na camada de rede, a PDU é chamada de pacote e é acompanhada por um cabeçalho IP. Ao alcançar a camada de enlace, ela se transforma em um quadro, e o cabeçalho de enlace é adicionado. Finalmente, na camada física, a PDU é convertida em sinais físicos, adequados para a transmissão [11].

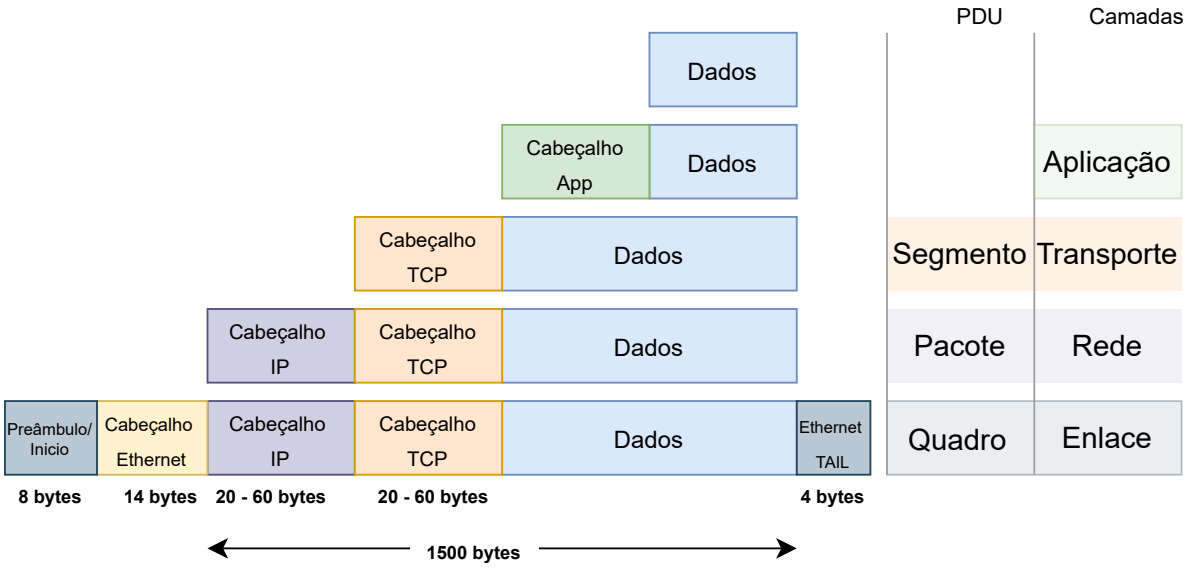

Figura 5 – Encapsulamento TCP/IP

FONTE: Adaptado de [12]

A Figura 5 ilustra o processo completo de encapsulamento. A Unidade Máxima de Transmissão, conhecida pela sigla em inglês MTU (*Maximum Transmission Unit*), determina o tamanho máximo que os dados úteis de um quadro na camada de enlace podem ter, excluindo-se os cabeçalhos e trailers.

A Tabela 2 apresenta a relação de overhead e carga útil transmitida [10]. Para transmissões pequenas, como 1 byte, o overhead é desproporcionalmente alto, atingindo 7.800%, o que significa que a quantidade de dados de controle é muito maior que os dados úteis. À medida que o tamanho da carga útil aumenta para 1 Kilobyte e 1 Gigabyte, o overhead se torna uma fração menor do total transmitido, caindo para 7,8% e 5,3%, respectivamente.

O cálculo baseia-se em um quadro Ethernet com uma MTU de 1500 bytes, resul-

tando em 1460 bytes de carga útil após a subtração de 20 bytes de cada cabeçalho IP e TCP. Na camada de enlace de dados, o padrão Ethernet adiciona sua própria sobrecarga de 26 bytes a cada PDU proveniente da camada de rede. Adicionalmente, no meio físico, 12 bytes são utilizados para cada pacote Ethernet como intervalo entre pacotes, também conhecido como tempo de silêncio. Portanto, a sobrecarga total em cada pacote Ethernet é de 78 bytes.

Tabela 2 – Relação de overhead e carga útil

Tamanho	Overhead
1 GB	5,3%
1 KB	7,8%
1 B	7.800%

Outras sobrecargas, tais como aquelas originadas pela aplicação do usuário, foram excluídas dos cálculos apresentados. Considerando a natureza teórica da análise, o *payload* mínimo de 46 bytes, especificado para o Ethernet, também foi desconsiderado. Este é apenas um exemplo de cálculo, porém, esta mesma análise pode ser aplicada a qualquer tecnologia de comunicação. É importante destacar que, para outros protocolos baseados no TCP/IP, o *overhead* pode variar e, em certas situações, ser significativamente maior. Isso ocorre inclusive com o protocolo Wi-Fi (*Wireless Fidelity*), utilizado na implementação deste projeto, que geralmente apresenta um *overhead* maior que o Ethernet. Esta diferença deve-se ao cabeçalho mais extenso, aos mecanismos adicionais de controle de acesso ao meio, necessários para gerenciar o ambiente de transmissão sem fio, e às retransmissões frequentes, que são necessárias devido a interferências.

2.2 Microcontroladores

Um microcontrolador é um dispositivo integrado que combina uma CPU (*Central Processing Unit*), memória, periféricos de entrada e saída e outros componentes essenciais em um único circuito integrado. Sua principal função é realizar tarefas específicas e controlar dispositivos ou sistemas. Os microcontroladores são amplamente utilizados em diversas áreas, como eletrodomésticos, automóveis, eletrônicos de consumo, dispositivos médicos e sistemas de controle industrial e automação [13].

Empregados para implementar sistemas ou dispositivos que necessitam de controle, monitoramento ou processamento de informações, os microcontroladores são frequentemente utilizados em interações com sensores, atuadores e outros dispositivos, onde o controle preciso é necessário para responder a eventos ou condições específicas.

O uso de microcontroladores apresenta várias vantagens. Uma delas é a capacidade de incluir diferentes componentes em um único encapsulamento, o que economiza espaço e energia. Isso torna os microcontroladores ideais para sistemas compactos e de baixo consumo energético. Além disso, eles são mais baratos em comparação com soluções que utilizam microprocessadores e componentes separados.

Os microcontroladores são projetados para programação em linguagens de alto nível, facilitando o desenvolvimento de software. Eles são capazes de trabalhar em tempo real, oferecendo respostas rápidas e previsíveis, fundamental em situações que exigem resposta imediata. A disponibilidade de microcontroladores com variadas capacidades e recursos permite atender a diferentes requisitos e aplicações. Contudo, suas limitações, como capacidade de processamento e memória, devem ser consideradas, pois podem não ser adequadas para aplicações mais complexas que demandam grande capacidade de processamento e armazenamento [14].

2.3 Sistemas Operacionais

Um sistema de computação é constituído fundamentalmente por hardware e software. Neste contexto, o SO (*Sistema Operacional*), sob uma perspectiva *Top-Down*, é compreendido como um conjunto de softwares que atua como uma camada de abstração entre o hardware e as aplicações de alto nível. Em uma abordagem *Bottom-Up*, o SO tem a responsabilidade de gerenciar a alocação ordenada e controlada dos recursos, como processadores, memórias e dispositivos de entrada/saída, distribuindo-os de forma eficiente entre os diversos programas que concorrem por esses recursos. Assim, o sistema operacional proporciona às aplicações um acesso uniforme aos dispositivos físicos. Isso permite que aplicações desenvolvidas há décadas, apesar de suas complexidades e diferenças tecnológicas, sejam compatíveis com dispositivos modernos [15]. Na computação, isso é possível através de camadas de abstração ou níveis de abstração. Que é uma maneira de ocultar os detalhes operacionais de um subsistema, permitindo a separação de preocupações para facilitar a interoperabilidade e a independência de plataforma. À medida que se avança para as camadas inferiores do modelo de abstração do sistema operacional, nota-se um aumento na complexidade e uma diminuição da abstração. Inversamente, ao atuar nas camadas superiores, como na camada de aplicação, ocorre uma maior abstração e simplificação das complexidades.

As camadas podem ser implementadas em hardware ou software. As camadas com maior abstração são implementadas em nível de software e as camadas com menor abstração em hardware. A camada que atua como interface entre o hardware e o software é o conjunto de instruções, conhecido como ISA (*Instruction Set Architecture*), que viabiliza a comunicação entre programas e sistemas operacionais com o processador.

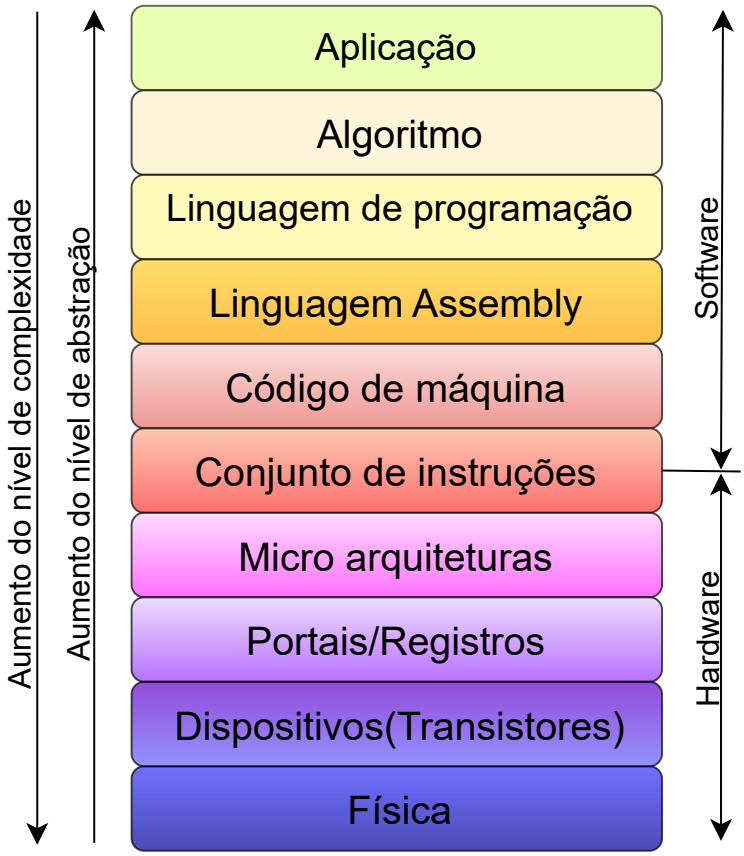

Figura 6 – Camadas de abstração

FONTE: Adaptado de [16]

Está camada pode ser implementada tanto em hardware quanto através de micro-código, que é uma forma intermediária entre o hardware puro e o software. O ISA especifica as operações básicas que o processador é capaz de realizar, incluindo aritmética, lógica, transferência de dados, controle de fluxo e operações de acesso à memória. Entre as arquiteturas mais conhecidas, destacam-se a x86, predominante em computadores pessoais, e a ARM, amplamente utilizada em dispositivos móveis e sistemas embarcados [17]. A Figura 6 apresenta as camadas de abstração de um sistema computacional.

O kernel é um componente central de um sistema operacional, com a função primordial de gerenciar a comunicação entre o software, incluindo aplicativos de nível de usuário, e os recursos de hardware, conforme ilustrado na Figura 7. Segundo [18], as principais responsabilidades do kernel incluem:

- Gerenciamento de Processos: Este envolve a criação, execução e encerramento de processos. Cada processo contém informações referentes à tarefa a ser executada. Múltiplos processos podem operar simultaneamente, e a gestão eficaz desses processos é essencial para evitar conflitos e assegurar o funcionamento adequado do sistema.

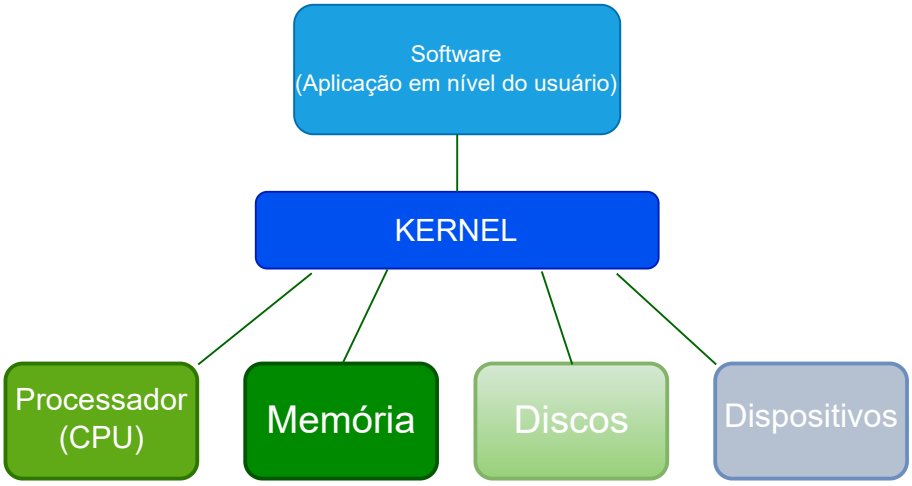

Figura 7 – Kernel Básico

FONTE: Elaboração Própria

- Gerenciamento de Memória: O kernel administra a alocação de memória para processos em execução e sua subsequente liberação. O controle de quais partes da memória estão ocupadas e quais estão disponíveis é uma função chave do kernel.
- Gerenciamento de dispositivos: O kernel também gerencia todos os diferentes dispositivos conectados ao sistema.
- Gerenciamento de Interrupções: O kernel é responsável por suspender processos em execução para tratar interrupções que surjam durante sua execução.
- Comunicação de entrada e saída: Como gestor de todos os dispositivos conectados, o kernel também lida com os diversos tipos de entradas e saídas realizadas por esses dispositivos, processando todas as informações provenientes do usuário e as saídas direcionadas a ele através de diferentes aplicativos.

2.3.1 Sistemas Operacionais em Tempo Real

O RTOS (*Real Time Operating System*) é um tipo de sistema operacional desenvolvido especificamente para atender a aplicações que possuem requisitos rigorosos em relação ao tempo de resposta. A principal função do RTOS é garantir que as tarefas sejam executadas de maneira determinística e previsível, cumprindo prazos estabelecidos. Essa característica é de extrema importância em cenários onde a precisão temporal é um fator crítico, como em sistemas de controle, automação industrial, dispositivos médicos e veículos autônomos, onde a previsibilidade e o cumprimento de prazos rigorosos são fundamentais.

Os RTOSs (*Real Time Operating Systems*) administram as tarefas atribuindo-lhes prioridades, garantindo que as tarefas mais críticas recebam atenção prioritária em rela-

ção às menos críticas. Além disso, esses sistemas oferecem mecanismos de sincronização e comunicação entre as tarefas, assegurando uma interação segura e ordenada. Uma característica fundamental dos RTOSs é a otimização para baixa latência, minimizando o tempo entre a ocorrência de eventos e as respostas correspondentes das tarefas. Em resumo, os RTOSs são soluções essenciais para aplicações que requerem um ambiente de tempo real, fornecendo a precisão e a previsibilidade necessárias para o funcionamento eficiente desses sistemas. Dentro do espectro dos sistemas operacionais em tempo real, o FreeRTOS emerge como um exemplo notável, alinhando-se perfeitamente com os requisitos e características mencionados acima.

2.3.2 FreeRTOS

O FreeRTOS, exemplificando a aplicação prática dos princípios dos RTOS, foi desenvolvido originalmente por Richard Barry, a partir de um de seus projetos de consultoria, e gradualmente se consolidou como um dos RTOS mais utilizados mundialmente [19]. Diversos fatores contribuem para a popularidade do FreeRTOS, incluindo sua eficiência, alta frequência de atualizações e manutenção, e uma política de licenciamento de software flexível, que o torna uma opção atraente tanto para projetos acadêmicos quanto comerciais sem a necessidade de custos com licença.

A compatibilidade do FreeRTOS com uma ampla variedade de microcontroladores e plataformas amplia sua aplicabilidade em diferentes contextos, fazendo dele uma escolha versátil para a implementação de sistemas exigentes em termos de tempo real [20]. Desenvolvido e continuamente aprimorado em colaboração com as principais fabricantes de microcontroladores, o FreeRTOS foi adaptado para mais de 40 arquiteturas de hardware, abrangendo microcontroladores de 8 a 64 bits e arquiteturas como ARM, AVR, x86, MIPS, RISC-V, entre outras. Suas principais vantagens incluem portabilidade, escalabilidade e simplicidade, com um kernel central compacto e eficiente, composto por apenas três ou quatro arquivos em linguagem C, resultando em uma imagem binária com tamanho reduzido [21].

2.4 Internet das coisas

O conceito de conectar objetos físicos à internet surgiu com [22] em 1999, com o objetivo de descrever a capacidade dos sensores de se conectarem a novos serviços na internet, dando origem ao termo "Internet das Coisas", do inglês *Internet of things*. Assim, a IoT viabiliza que objetos não apenas coletem e processem dados, mas também os compartilhem e interajam por meio da internet, utilizando tecnologias como computação pervasiva, dispositivos integrados, tecnologias de comunicação e redes de sensores. Essa integração possibilita a realização de tarefas e a interação com outros dispositivos

em diversos contextos e ambientes, trazendo impactos significativos para a vida cotidiana. Exemplos desses avanços incluem a automação residencial em casas inteligentes, o aumento da produtividade no agronegócio e a melhoria da eficiência nas indústrias 4.0 [23].

Com isso, atrai considerável atenção e cresce rapidamente em diversas aplicações. Entre os principais domínios de aplicação da IoT podemos citar transporte, saúde, agricultura, casas inteligentes, veículos, educação, mercado e indústria. Essas aplicações podem se interconectar, formando um mercado horizontal, e dentro de cada domínio de aplicação existem diversos serviços independentes, formando um mercado vertical.

Cada setor específico do mercado vertical utiliza a IoT para atender as suas necessidades e desafios. Por exemplo, na agricultura, são utilizados sensores para monitorar condições do solo, clima, e saúde das plantas, otimizando a irrigação e o uso de fertilizantes. Em contrapartida, o mercado horizontal refere-se a produtos, serviços ou soluções desenvolvidos para serem utilizados em uma ampla variedade de indústrias e setores. Um exemplo disso é uma plataforma de análise de dados que pode ser usada tanto na agricultura para monitorar as condições do solo, quanto no setor de saúde para monitorar sinais vitais de pacientes. A Figura 8 sintetiza os principais domínios de aplicação e a representação dos mercados verticais e horizontais.

Figura 8 – Aplicações IoT

FONTE: Adaptado de [24]

A atuação diversificada da IoT, aliada ao avanço tecnológico, que inclui hardware mais acessível, armazenamento em nuvem, maior velocidade e redução dos custos de conectividade, sugere um potencial econômico significativo. Estima-se que a IoT poderá gerar um impacto econômico anual entre US$ 3,9 e US$ 11,1 bilhões até 2025, correspondendo a cerca de 4 a 11% do PIB (*Produto Interno Bruto*) global [25].

2.4.1 Elementos da IoT

De acordo com [26], a IoT é constituída por elementos fundamentais que possibilitam seu funcionamento adequado. Dentre esses elementos, tem-se:

- Identificação: Consiste na atribuição de uma identidade única a cada objeto na rede, dividindo-se em dois processos: nomeação e endereçamento. A nomeação refere-se ao nome do objeto, enquanto o endereçamento ao seu endereço exclusivo. Embora objetos possam ter o mesmo nome, os endereços devem ser diferentes e únicos.
- Sensoriamento: Processo de coleta de informações dos objetos, utilizando-se de Tags RFID (*Radio Frequency Identification*), sensores de pressão, temperatura, entre outros.
- Comunicação: Fundamental para a interconexão e troca de mensagens entre dispositivos. Para facilitar essa comunicação, são empregadas diversas tecnologias, tais como Bluetooth, Wi-Fi e LTE (*Long-Term Evolution*).
- Computação: Utilizada para processar, filtrar e remover informações desnecessárias coletadas pelos sensores. Pode ser implementada tanto em plataformas de hardware, como Raspberry Pi e Arduino, quanto em sistemas operacionais avançados, como Tiny OS e Android.
- Serviços: Existem quatro tipos principais de serviços. O serviço de identidade, que determina a identidade dos objetos solicitantes; o de agregação de informações, que coleta e processa dados dos objetos; o colaborativo, que toma decisões com base nos dados coletados e comunica respostas aos dispositivos; e o ubíquo, que responde instantaneamente aos dispositivos, independentemente de tempo ou localização.
- Semântica: Responsável por facilitar as tarefas dos usuários, atuando como o "cérebro"do sistema. Ele obtém informações, toma decisões e envia respostas apropriadas aos dispositivos para cumprir suas responsabilidades.

2.4.2 Arquiteturas

A arquitetura da IoT é fundamental para fornecer uma estrutura organizada que permite a interconexão e funcionamento eficiente dos dispositivos conectados. Ela busca

estabelecer uma comunicação inteligente entre objetos físicos e sistemas de informação, lidando com a coleta, processamento e troca de informações de forma eficaz.

A estrutura é composta por várias camadas que descrevem diferentes aspectos e funcionalidades do sistema. Para lidar com desafios como alta taxa de tráfego e demanda por armazenamento, as arquiteturas devem considerar aspectos como segurança, confiabilidade, integridade e qualidade de serviço [27].

A configuração mais básica, composta por três camadas, foi introduzida nos estágios iniciais de pesquisas nessa área [28, 29, 30]. No entanto, devido ao contínuo desenvolvimento da IoT, diversas outras arquiteturas em camadas foram propostas na literatura para atender às demandas emergentes, incluindo as de quatro camadas e cinco camadas[31, 32]. A Figura 9 apresenta uma ilustração das camadas para esses três tipos de arquitetura. Estas arquiteturas são descritas a seguir.

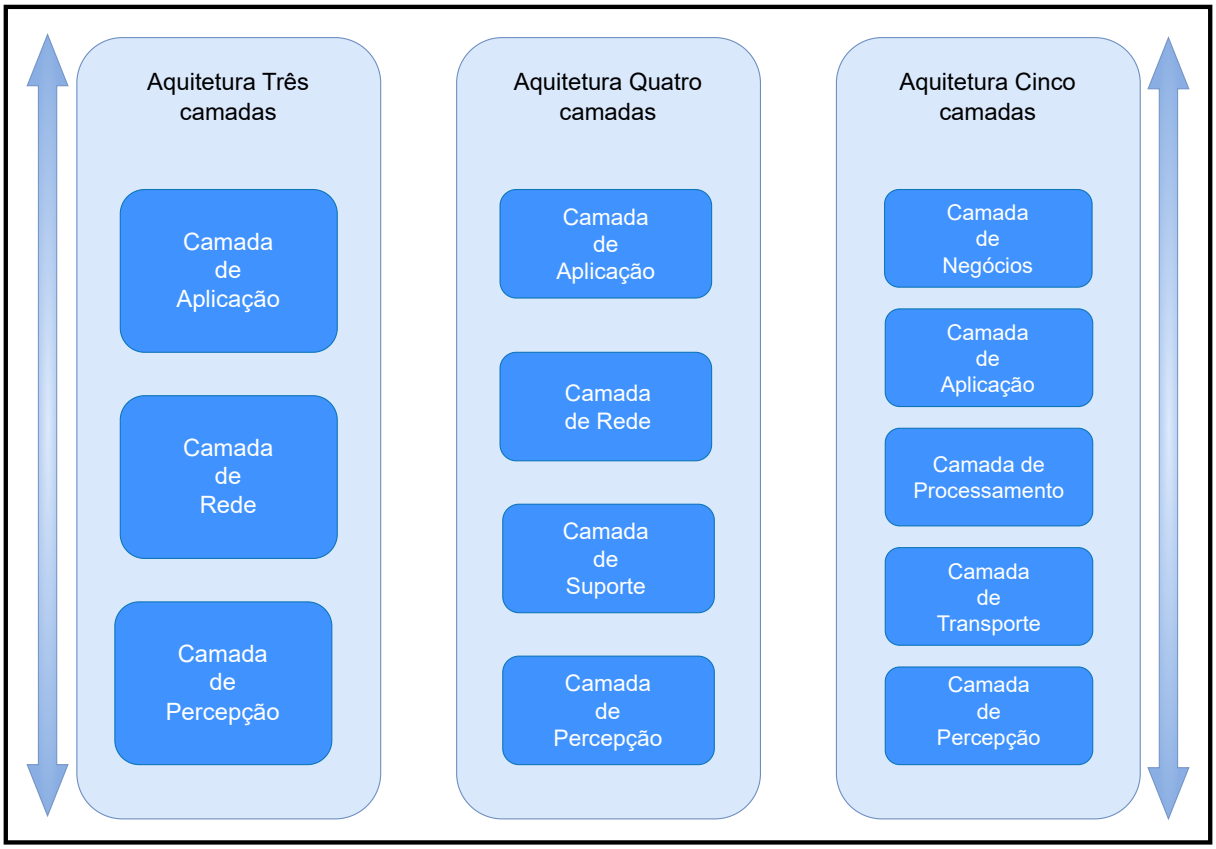

Figura 9 – Comparação entre as arquiteturas de IoT de 3, 4 e 5 camadas.

FONTE: Adaptado de [33]

A. Arquitetura de três camadas

(I) Camada de Percepção: Essencial na identificação e coleta de informações, a camada de percepção atua como a interface física entre o mundo real e o sistema IoT. Utilizando dispositivos como tags RFID, sensores, câmeras e GPS (*Global Positioning System*), ela é responsável por identificar objetos e captar dados relevantes, além de re-

conhecer outros objetos inteligentes no ambiente, facilitando o funcionamento das demais camadas [28, 29, 30].

(II) Camada de Rede: Camada fundamental para transmissão e processamento das informações coletadas pela camada de percepção. Sua principal função é realizar a interface e possibilitar a interconexão e comunicação eficiente entre os elementos do sistema, que incluem dispositivos sensores, dispositivos de rede e servidores.

(II) Camada de Aplicação: O objetivo desta camada é fornecer serviços ao usuário final. Ela define diversas aplicações para a IoT, como casas, cidades e saúde inteligentes, baseando-se nos dados coletados pela camada de percepção.

B. Arquitetura de Quatro Camadas

Além das camadas presentes na arquitetura de três camadas, a arquitetura de quatro camadas introduz uma nova camada focada em aprimorar a segurança. As três primeiras camadas possuem funções similares às da arquitetura anterior, sendo a quarta camada denominada camada de suporte.

(III) Camada de Suporte: Conhecida como camada de segurança, suas principais responsabilidades são a autenticação do remetente e a integridade das informações, frequentemente utilizando senhas ou chaves compartilhadas para autenticação. Essa camada também envia informações para a camada de rede e é crucial para aumentar a segurança, embora não seja suficiente para proteger completamente as demais camadas [33].

C. Arquitetura de cinco camadas

A arquitetura de cinco camadas visa superar alguns desafios do modelo de quatro camadas. Neste modelo, duas camadas adicionais são propostas para expandir o modelo de três camadas, melhorando a segurança, principalmente no que diz respeito à privacidade do usuário, processamento de dados, e a definição de como são realizados negocios no âmbito da IoT

(I) Camada de Negócios: Essa camada é responsável pelo controle e gerenciamento dos aplicativos, preservação da privacidade do usuário e administração do sistema. Ela também desempenha um papel vital no gerenciamento e cobrança das aplicações da IoT, além de conduzir pesquisas relacionadas.

(III) Camada de Processamento: A função principal desta camada é a coleta de dados, extração de informações relevantes e filtragem de ruídos. Ela é crucial para otimizar o processamento de grande volume de dados, atuando como uma camada intermediária. As principais tecnologias empregadas nesta camada incluem computação em nuvem e computação ubíqua.

2.4.3 Protocolos e tecnologias

A seleção apropriada da tecnologia e do protocolo é fundamental em aplicações de IoT, dada a ampla variedade disponível. A escolha é influenciada por diversos fatores, incluindo o caso de uso específico, requisitos operacionais, topologia da rede e limitações tecnológicas. Devido à natureza remota e à necessidade de conexões sem fio para os dispositivos inteligentes, os sistemas devem lidar eficientemente com redes de acesso que podem ser não confiáveis, intermitentes e com baixa largura de banda, assegurando a conectividade dos dispositivos [34].

Assim, a utilização de protocolos é crucial para garantir a correta leitura e interpretação das informações transmitidas entre dispositivos, permitindo a troca de informações em uma linguagem padronizada pelo ecossistema. A comunicação em IoT pode ocorrer através de diferentes tecnologias e protocolos. A seguir, são discutidas as tecnologias Wi-Fi e LoRaWAN. O Wi-Fi será utilizado durante a implementação prática do driver proposto, e o LoRaWAN é abordado devido ao uso do protocolo LoRa (*Long Range*) nos cálculos teóricos; ambas as tecnologias são complementares, uma vez que o LoRa está na camada física do LoRaWAN. No Apêndice A, serão detalhados outros protocolos e tecnologias também empregados em IoT [35].

Wi-Fi

O Wi-Fi, é amplamente utilizado para comunicação sem fio, opera nas bandas de frequência de 2,4 GHz e 5 GHz. Permite a criação de redes para transmissão de dados e conexão de variados dispositivos, como smartphones, computadores e roteadores, fornecendo conectividade à internet em um alcance de até 100 metros. O Wi-Fi é especialmente útil em aplicações que requerem altas taxas de transferência de dados e estão localizadas próximas a pontos de acesso, sendo amplamente utilizado em ambientes como casas e cidades inteligentes. Sua principal vantagem reside na alta velocidade de transferência de dados, que permite uma comunicação rápida e eficiente entre dispositivos.

Contudo, é importante considerar que o Wi-Fi pode não ser a escolha ideal para dispositivos com limitações rigorosas de energia ou para locais remotos, devido ao seu consumo energético relativamente alto e alcance limitado. Em comparação com protocolos de baixo consumo energético, como LoRaWAN ou BLE (*Bluetooth Low Energy*), o Wi-Fi pode ser menos adequado nessas situações. Portanto, em casos que demandam eficiência energética e extenso alcance, a escolha de tecnologias mais adaptadas a essas necessidades específicas pode ser mais benéfica, possibilitando uma implementação eficaz de dispositivos IoT nessas condições.

LoRaWan

Desenvolvido pela LoRa Alliance, o LoRaWAN constitui uma camada de comunicação baseada na tecnologia LoRa, projetada para comunicações de longa distância com

baixo consumo energético. Esta tecnologia é ideal para redes sem fio escaláveis com um grande número de dispositivos, oferecendo suporte a mobilidade, segurança e comunicação bidirecional. Apesar de apresentar vantagens como ampla cobertura, custo reduzido e baixo consumo de energia, o LoRaWAN possui algumas limitações, incluindo taxas de dados que variam de 0,3 a 50 Kbps e uma capacidade de carga útil restrita. Contudo, destaca-se como uma opção eficiente para aplicações que requerem um alcance estendido e eficiência energética [36][8].

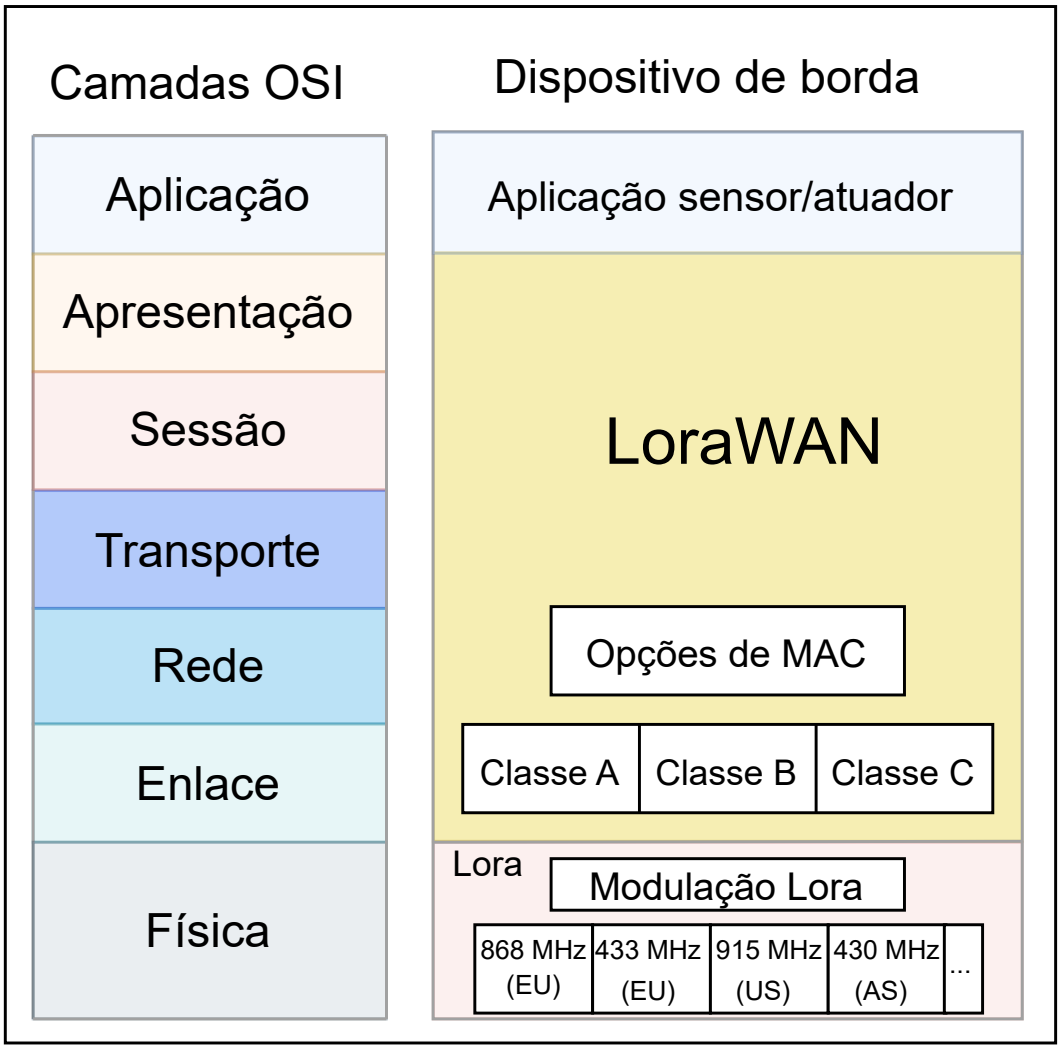

Figura 10 – Pilha do protocolo LoraWan simplificada no modelo OSI

FONTE: Adaptado de [37]

A pilha de protocolos LoRaWAN, ilustrada na Figura 10, formada pela camada física LoRa, uma subcamada de acesso ao meio e a camada de aplicação. Esta última representa os dados de uma aplicação LoRaWAN ou os dados de um gateway que permite a comunicação dessa tecnologia com o padrão TCP/IP. O LoRaWAN classifica os dispositivos em três categorias: classes A, B e C, cada uma com capacidades distintas. Os dispositivos da classe A empregam o método ALOHA puro para o uplink e, após enviar um quadro, aguardam por uma resposta durante duas janelas de recepção de downlink.

Esta classe é reconhecida por seu baixo consumo de energia [38]. As classes B e C diferem em seus mecanismos de downlink e na duração de escuta do canal, resultando em um consumo energético mais elevado. Dado o enfoque deste trabalho na eficiência energética, as classes B e C não serão detalhadas.

A Figura 11 ilustra a estrutura de um quadro LoRaWAN. O tamanho máximo da carga útil pode variar de acordo com a região e os parâmetros regionais específicos. A especificação do LoRaWAN estabelece diferentes configurações para a taxa de dados, o fator de espalhamento e a largura de banda do canal. Fatores de espalhamento menores fornecem uma taxa de bits mais alta para uma largura de banda e taxa de codificação fixa. Permitindo que a mesma quantidade de dados seja enviada em menos tempo, favorece a eficiência de transmissão e reduz o consumo de energia com o compromisso de um alcance reduzido principalmente em ambientes ruidosos. Por outro lado, valores maiores do fator de espalhamento reduzem a taxa efetiva de dados, isso significa que para transmitir a mesma carga útil, o sistema precisará de mais tempo resultando em uma menor eficiência energética. No entanto, ao aumentar o fator de espalhamento, cada bit de dado é representado por mais chipes, o que torna o sinal mais robusto contra ruído e interferência. Isso é benéfico para alcançar distâncias maiores ou garantir uma comunicação mais confiável em ambientes desafiadores [39]. A estrutura de um quadro LoRaWAN é segmentado em três camadas principais: Camada Física, Camada MAC e Camada de Aplicação, cada uma servindo a propósitos distintos dentro do protocolo.

- Camada Física: Esta camada base consiste em um preâmbulo de oito bytes, um cabeçalho com um CRC (*Cyclic Redundancy Check*) acompanhante de 20 bits, um *payload* da camada física e um CRC para o payload de dois bytes. O preâmbulo facilita a sincronização e detecção do início do quadro, enquanto o cabeçalho e seu CRC garantem a integridade e a decodificação adequada da estrutura do quadro.

- Camada MAC (*Media Access Control*): Situada acima da camada física, esta camada inclui um cabeçalho MAC de um byte, um MIC (*Message Integrity Code*) de quatro bytes e o *payload* da camada MAC. O cabeçalho MAC dita o tipo de quadro e outras informações de controle e o MIC fornece um mecanismo robusto para garantir a integridade e autenticidade da mensagem.

- Camada de Aplicação: A camada mais alta inclui um cabeçalho que pode variar 7-22 bytes, uma porta de um byte e o payload da aplicação. O cabeçalho neste nível inclui informações críticas como o endereço do dispositivo com quatro bytes, informações de controle com um byte, um contador com dois bytes e, informações opcionais de 0-15 bytes. A porta especifica o ponto final da aplicação.

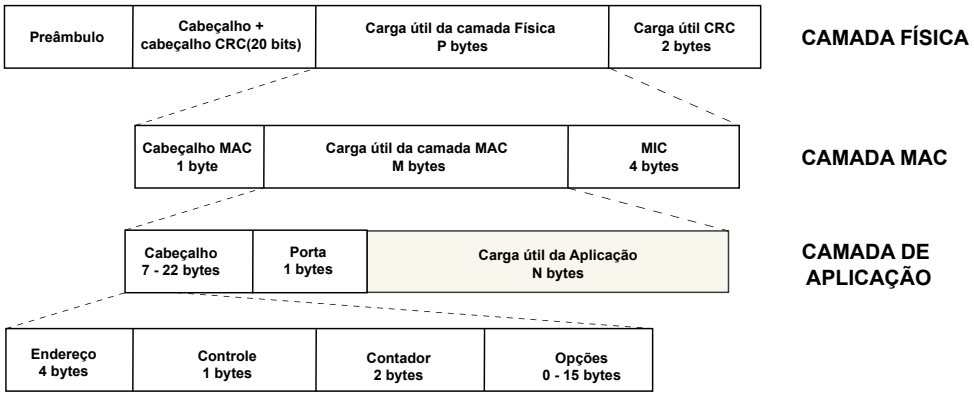

Figura 11 – Quadro LoRaWAN

FONTE: Adaptado de [40]

A Tabela 6, apresentada no Apêndice B, sintetiza as tecnologias e protocolos no contexto das aplicações de IoT, destacando suas taxas de comunicação e alcances de transmissão. O objetivo desta tabela é oferecer um panorama das principais alternativas disponíveis, proporcionando suporte na escolha da tecnologia mais apropriada para cada contexto específico de aplicação.

2.4.4 Desafios

A IoT configura-se como um cenário inovador e com relevante potencial de transformação. Entretanto, enfrenta diversos desafios que requerem abordagens em suas distintas camadas e aspectos. Conforme ilustrado na Figura 12, os desafios podem ser categorizados em quatro grupos principais: arquitetura, entidades, tecnologia e recursos. Os desafios de arquitetura referem-se a aspectos específicos relacionados às camadas da IoT e à sua integração em uma estrutura unificada. Os desafios de entidades estão relacionados à implementação em larga escala de hardware, software e dados. Já os desafios de tecnologia dizem respeito às tecnologias suporte e questões específicas da IoT. Por fim, os desafios de recursos abrangem questões como consumo de energia, compatibilidade, conectividade e segurança [41].

Assim, é fundamental criar um sistema eficaz que dê nomes e gerencie adequadamente a identificação de objetos e sensores. Além disso, a interoperabilidade e a padronização são cruciais, pois melhoram a forma como as informações podem ser acessadas e mantêm a privacidade. Para garantir a segurança, é importante proteger contra acessos não autorizados e usar criptografia para proteção dos dados. Outro desafio significativo é a sustentabilidade, considerando o aumento no consumo de energia associado ao crescimento dos dispositivos conectados. Soluções sustentáveis e energeticamente eficientes são indispensáveis para assegurar o sucesso e a contínua evolução da tecnologia [32].

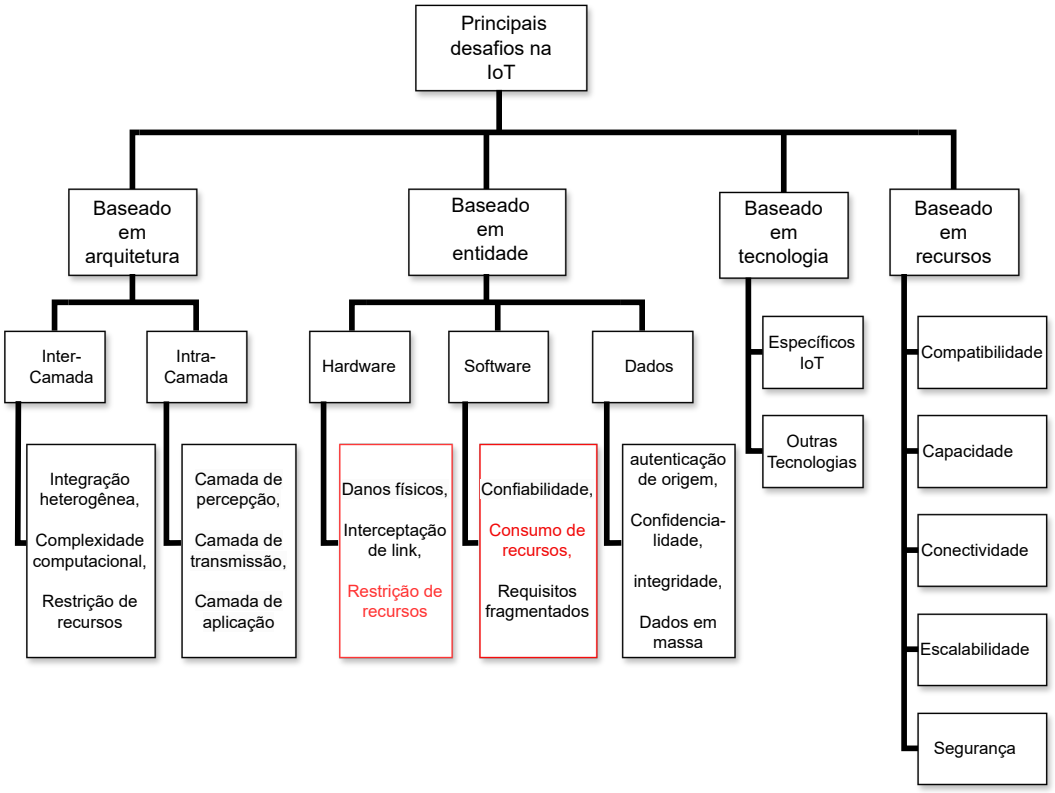

Figura 12 – Desafios da IoT

FONTE: Adaptado de [41]

Na IoT, além da eficiência energética do sistema como um todo, muitos dispositivos operam alimentados por baterias. Por essa razão, é essencial empregar técnicas que reduzem o consumo de energia e, consequentemente, prolonguem a vida útil das baterias. Entre as principais estratégias, destacam-se a utilização de hardware de baixo consumo e implementação de modos de repouso. Em algumas aplicações, a coleta de energia também pode ser uma opção viável. Vale ressaltar que, em comunicação sem fio, o transceptor de rádio é geralmente um dos componentes que mais consome energia, tornando as técnicas de Controle de Redução de Consumo RDC (*Radio Duty-Cycling*) fundamentais para minimizar esse gasto energético excessivo [42].

Nos dispositivos IoT que necessitam transmitir apenas pequenas quantidades de dados, o problema de *overhead* pode ser particularmente relevante devido à proporção entre o tamanho reduzido dos dados úteis e a quantidade de informações adicionais exigidas para a transmissão. Isso ocorre porque a inclusão de cabeçalhos, metadados e informações de controle pode representar uma parte significativa do tamanho total do pacote de dados. Portanto, do ponto de vista energético, é necessário escolher o protocolo de comunicação sem fio a ser utilizado com base no contexto específico de uso. Além disso, medidas para evitar retransmissões e gerenciar eficientemente o tempo de atividade do dispositivo são fundamentais para garantir a operação correta e energeticamente eficiente do sistema.

2.5 Estado da arte

O foco deste estudo são dispositivos IoT cujo recurso de energia é limitado. Quando este requisito é atendido, uma consequência direta é a extensão da vida útil das baterias, o que implica diretamente na minimização da intervenção humana para substituí-las. Além disso, ocorre a redução do consumo total da rede IoT, contribuindo para um sistema mais sustentável. No entanto, atender a essa especificação é bastante desafiador durante a fase de projeto de um dispositivo sem fio, uma vez que é obrigatório incluir um módulo para transmissão de dados. Este componente é um dos mais exigentes em termos de energia. Portanto, é necessário adotar alguma abordagem para contornar esse problema e minimizar o consumo.

Para obter um sistema com requisitos de baixo consumo energético, é necessário assegurar que tanto o hardware quanto o software estejam em conformidade com tais requisitos. No âmbito do hardware, de forma simples, é essencial garantir que os circuitos utilizados apresentem características de baixo consumo e que os periféricos não utilizados possam ser desligados. No que se refere ao software, abordagens para minimizar o consumo durante a execução da aplicação, isso inclui a coleta de dados, processamento e a transmissão de dados.

Em [43], o autor apresenta uma técnica custo-efetiva, baseada em redes petri, para otimizar o consumo de energia em sistemas DRE(*distributed real-time embedded*). Denominado de modelo HDRE-net (*Hierarchical Distributed Real-time Embedded-net*), o consumo de energia é otimizado baseado na técnica DVS (*Dynamic Voltage Scaling*). No entanto, apresenta complexidade de implementação e introduz *overhead* para realizar a comutação dinâmica.

Em [44], é apresentado um dispositivo para realizar o monitoramento de lixeiras. Utiliza-se uma técnica para economia de energia que desabilita a alimentação dos principais periféricos. As baterias fornecem alimentação para um contador, que a estoura a cada 57 minutos. Com isso, o circuito do microcontrolador é alimentado, executa um firmware para realizar o sensoriamento da lixeira e habilita a alimentação do rádio para transmitir o estado atual. Após a transmissão, o contador é reiniciado e o microcontrolador e o rádio permanecem desligados até o próximo estouro. Apesar da economia de energia, remover a alimentação do microcontrolador reduz a confiabilidade e, em caso de falha, não é possível restaurar o sistema.

Em [45], o autor apresenta uma técnica de amostragem acionada por evento, que consiste em esquemas de amostragem aperiódicos ou assíncronos para atualizar as informações apenas quando uma mudança relevante na medição é detectada. O mecanismo de disparo pode ser ativado no dispositivo sensor, medição baseada em limite, e utiliza chaves eletrônicas analógicas de baixo consumo para desabilitar o rádio quando não está

em utilização. Além disso, apresenta um nível de complexidade maior que uma aplicação com amostragem periódica, incluindo uma latência variável no sistema. Pode ocorrer a perda de eventos rápidos e apresentar dificuldade para identificar eventos relevantes.

Uma estratégia para economia de energia através da redução de dados baseado na técnica de amostragem adaptativa usando nível de risco é apresentada em [46]. A amostragem adaptativa com nível de risco é uma técnica em que o processo de coleta de dados é ajustado dinamicamente, com base na avaliação contínua do risco associado às observações ou resultados obtidos, baseando-se nas informações das amostras anteriores. Ao contrário das estratégias de amostragem fixa, nas quais o plano de amostragem é estabelecido de forma prévia e permanece inalterado, independentemente dos resultados obtidos, a amostragem adaptativa modifica sua metodologia com base em novas informações ou resultados adquiridos ao longo do processo. E devido a aplicação ser a coleta de informação sobre sinais vitais, foi aplicado as técnicas de DWT (*Transformada Wavelet Discreta*) e DPCM (*Differential pulse-code modulation*) para filtragem de ruído, que é uma estratégia bem específica para algumas aplicações especiais.

Em [47], o autor propõe um esquema de gerenciamento de energia que engloba três estratégias principais. A primeira estratégia envolve técnicas que reduzem a quantidade de dados que são transmitidos ou recebidos pelos nós baseando-se em energia. A segunda estratégia compreende uma metodologia de trabalho para economizar a energia consumida em cada nó IoT. A terceira estratégia compreende um cenário de tolerância a falhas. A minimização de dados inclui três processos: priorização de dados, compactação de dados e ajuste de dados. A abordagem funciona em nós com alto poder de processamento, visto que precisa de algoritmos para tomada de decisão. Uma outra técnica para melhorar a eficiência energética na transmissão de dados é ajustando o nível de potência de transmissão dinamicamente, como apresentado em [48]. É uma técnica muito eficiente quando tem-se uma rede de sensores, no entanto não é eficaz em aplicações com dispositivos distantes fisicamente.

Quando se fala em eficiência, principalmente em locais remotos, uma das técnicas óbvias é a captação de energia, em [49] foi desenvolvido um estudo que analisa os resultados mais recentes na área de captação de energia de IoT e explora sua viabilidade para diversas aplicações. O tipo de coletor de energia e o recurso energético varia de acordo com a aplicação e o sistema. Que podem ser realizadas através de diferentes fontes de energia, como por exemplo energia de radiofrequência em dispositivos RFID, através do calor corporal ou bioenergia em aplicações de monitoramento de saúde, ou através da luz solar utilizando células fotovoltaicas em aplicações ao ar livre. No entanto, a coleta de energia não é o foco deste trabalho.

3 Método Proposto

Este capítulo apresenta a implementação detalhada de um driver confiável e portátil para o FreeRTOS, projetado para gerenciar a transmissão de dados em dispositivos de IoT de baixo consumo energético. A metodologia adotada para alcançar este objetivo baseia-se na técnica de agrupamento de dados. Além disso, serão fornecidos cálculos teóricos relativos ao consumo energético, possibilitando a avaliação comparativa da eficácia da solução proposta em relação a duas tecnologias de comunicação frequentemente utilizadas em IoT. Finalmente, serão descritas as ferramentas utilizadas na implementação prática. Os códigos fonte, estão disponíveis para consulta e uso público no repositório do GitHub e podem ser acessados diretamente através do link: IoT Edge Data Efficiency Repository. A inclusão deste repositório visa promover a transparência, replicabilidade e continuidade da pesquisa.

3.1 Procedimentos Metodológicos

Este tópico tem como finalidade classificar a metodologia de pesquisa adotada neste trabalho em termos de abordagem, natureza e objetivo. Definir o método de pesquisa é um passo essencial no início de uma pesquisa científica, pois as escolhas metodológicas influenciam significativamente a confiabilidade e a aplicabilidade dos resultados obtidos.

A abordagem quantitativa foi selecionada para este estudo, com o intuito de mensurar, analisar e quantificar o impacto da proposta de redução do consumo energético na transmissão de dados em dispositivos IoT. Essa abordagem é adequada para gerar dados numéricos sobre variáveis específicas, como tempo de transmissão e consumo de energia, permitindo identificar padrões, medir a eficiência e validar estatisticamente os resultados, a fim de avaliar a redução do consumo energético.

Trata-se de uma pesquisa de natureza aplicada, com foco na solução de um problema prático e concreto do mundo real: a diminuição do consumo energético na transmissão de dados em dispositivos IoT alimentados por bateria. O objetivo do estudo é explicativo, visando compreender a relação causal entre o agrupamento de dados e a redução do consumo energético.

3.2 Materiais Utilizados

Esta seção descreve os materiais, equipamentos e softwares utilizados ao longo da execução da pesquisa para a realização da implementação prática, detalhando suas

especificações, procedência e, quando aplicável, as configurações empregadas. A seguir, apresentam-se detalhadamente os recursos empregados, subdivididos em categorias conforme sua natureza e função no contexto do estudo.

3.2.1 ESP32

O ESP32 é um microcontrolador amplamente adotado em projetos IoT, destacando-se como uma solução versátil para uma variedade de aplicações, graças às suas funcionalidades integradas de Wi-Fi e *Bluetooth*. A placa de desenvolvimento utilizada, denominada ESP32-DEVKITV1, é retratada na Figura 13 e inclui o módulo ESP32-WROOM-32, um regulador de tensão, uma interface USB destinada à programação e depuração, bem como conectores barra pinos que facilitam o acesso aos pinos de entrada e saída do microcontrolador [50].

Adicionalmente, o ESP32 é equipado com um microprocessador Tensilica Xtensa LX6 de duplo núcleo, proporcionando significativo poder de processamento, o que favorece a multitarefa e a execução eficaz de tarefas complexas. Ele é compatível com múltiplos padrões Wi-Fi, incluindo 802.11 b/g/n, além de suportar conectividade *Bluetooth Classic* e BLE. Possui modos de "*sleep*"e funcionalidades de gestão de energia que contribuem para a minimização do consumo energético [51].

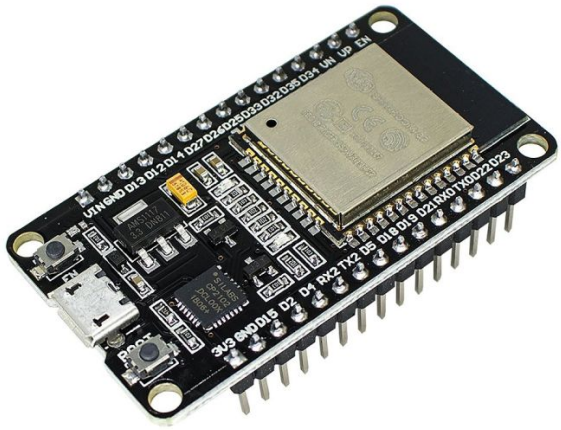

Figura 13 – Módulo ESP32

FONTE: [52]

Assim, a natureza de código aberto do ESP32, aliada ao amplo suporte da comunidade, consolidou-o como uma escolha primordial para inúmeros desenvolvedores. A agregação de seus recursos configura-o como uma ferramenta de grande valor para uma diversidade de projetos nos âmbitos de IoT e sistemas embarcados [53].

3.2.2 Visual Studio Code

O Visual Studio Code, conhecido também como VSCode, é um editor de código amplamente utilizado no desenvolvimento de software. Sua grande aceitação na comunidade de desenvolvedores deve-se à versatilidade, capacidade de personalização e interface de fácil utilização. Um dos benefícios significativos do uso do VSCode para o desenvolvimento de software embarcado é seu ecossistema de extensões, com diversas delas disponíveis para o trabalho com o ESP32, incluindo SDK (*Software Development Kit*) e bibliotecas específicas da plataforma [54]. Essas extensões oferecem funcionalidades e ferramentas adicionais, auxiliando o processo de desenvolvimento e facilitando a interação com o ESP32.

Além disso, o VSCode conta com uma comunidade de desenvolvedores grande e ativa, que proporciona acesso a uma ampla quantidade de recursos e suporte. Os desenvolvedores encontram tutoriais, documentação e fóruns onde podem buscar assistência e orientação para seus projetos envolvendo o ESP32. A comunidade em torno do VSCode assegura o acesso a informações recentes e melhores práticas para trabalhar com o ESP32.

O VSCode também permite o trabalho com o ESP-IDF (Espressif IoT Development Framework), por meio de uma extensão dedicada, que oferece integração completa com o framework, facilitando o desenvolvimento. Com uma interface intuitiva e forte apoio da comunidade, o Visual Studio Code se estabelece como escolha preferida para desenvolvedores em projetos com o ESP32, com um ambiente ideal para tarefas de desenvolvimento firmware. Por essas razões, foi selecionado para o desenvolvimento do driver proposto [55].

3.2.3 PyCharm

O PyCharm, um ambiente de desenvolvimento integrado dedicado ao Python [56], destaca-se pelo seu vasto conjunto de ferramentas de desenvolvimento, que incluem depuração de código, gerenciamento de projetos e conclusão inteligente de código [57]. Tais recursos elevam a eficiência e a produtividade na programação em Python, otimizando a escrita, o teste e a depuração de códigos.

A integração do PyCharm com bibliotecas Python, como NumPy e NiBabel, facilita o trabalho com dados complexos e a realização de tarefas avançadas de processamento [56]. Essa funcionalidade é particularmente preciosa em contextos que requerem uma intensiva manipulação de dados. Assim, devido a essas características, o PyCharm foi selecionado para o desenvolvimento de um servidor TCP destinado à recepção de dados do dispositivo.

3.2.4 Setup

Para a realização dos testes, foi utilizado um roteador TP-Link, modelo EC220-G5 [58], ideal para uso em redes domésticas ou em pequenos comércios. Suas principais características incluem:

- **Padrões sem-fio:** IEEE 802.11ac/n/a 5 GHz, IEEE 802.11n/b/g 2.4 GHz;
- **Velocidade sem-fio:** Dual-band com velocidades de até 1350 Mbps (450 Mbps em 2.4GHz e 867 Mbps em 5GHz);
- **Portas:** 1 Porta WAN Gigabit Ethernet e 4 Portas LAN Gigabit Ethernet;
- **Antenas:** 3 Antenas externas fixas;
- **Criptografia:** WPA-PSK/WPA2-PSK, WPA/WPA2, que protege a rede contra acessos não autorizados.

Uma rede local de teste foi configurada, conectando apenas o ESP32-DEVKITV1 e o servidor. O servidor utilizado era uma máquina com processador Core™ i5-3230M de 2,6 GHz, 4 GB de DDR3-SDRAM, SSD de 500 GB, compatível com o padrão Wi-Fi 4 802.11n, placa-mãe Intel HM77 Express e sistema operacional Ubuntu 22.04.4 LTS.

3.3 Metodologia utilizada para a medição do consumo

A medição do consumo de corrente na placa ESP32-DEVKITV1, foi efetuada por meio da configuração apresentada na Figura 14.

O kit ESP32 DEVKITV1 utilizado pode ser alimentado tanto pelo conector USB (*Universal Serial Bus*), quanto diretamente através dos pinos de alimentação externos no conector de barras de pinos. O módulo ESP32 opera com uma alimentação de 3,3 VCC, mas a placa inclui um regulador que permite a alimentação com 5 VCC no pino VIN. Para a medida de corrente, empregou-se um resistor, denominado Rsense, de 1 ohm. A corrente é calculada conforme a Lei de Ohm, que estabelece que a corrente elétrica diretamente proporcional à diferença de potencial aplicada, expressa pela equação:

$$I = \frac{V}{R} \tag{3.1}$$

Onde I representa a corrente elétrica, V a diferença de potencial e R a resistência. Considerando que o valor do resistor Rsense é de 1 ohm, a corrente se torna diretamente proporcional à tensão. A medida da tensão é realizada por meio de um osciloscópio digital.

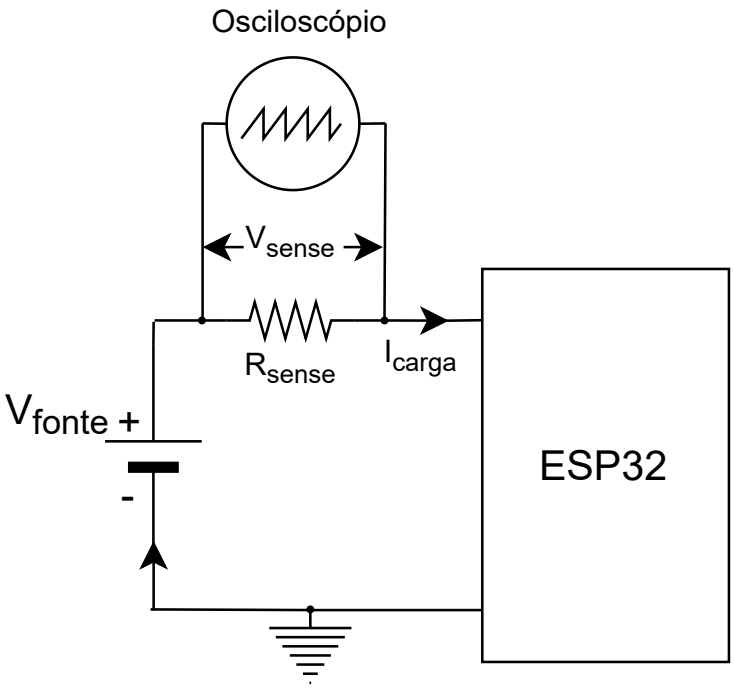

Figura 14 – Cenário utilizado para medida de consumo

FONTE: Elaboração Própria

3.4 Comunicação cliente/servidor

O driver proposto foi implementado no ESP32 com o objetivo de avaliar a solução em um cenário real. A implementação considera o uso de um sensor que coleta dados ambientais e os transmite a um servidor. Os dados de teste incluem o número de identificação do dispositivo, o tipo de medição e o valor medido, resultando em uma carga útil de 12 bytes. A comunicação entre o sensor e o servidor é realizada via rede Wi-Fi, conforme representado na Figura 15. O sensor se autentica na rede e, quando necessário, estabelece um socket TCP na função de cliente para o envio dos dados.

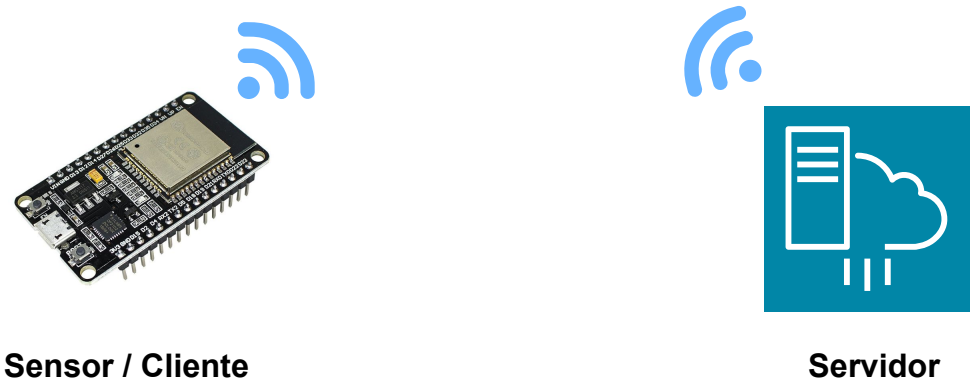

Figura 15 – Comunicação cliente/servidor

FONTE: Elaboração Própria

No lado do servidor, implementou-se um servidor TCP em Python, utilizando a IDE (*integrated development environment*) PyCharm, com o propósito de receber e exibir os dados transmitidos pelo sensor.

3.5 Cenário IoT para dispositivos baixo consumo

Os dispositivos de borda em IoT têm como função principal coletar informações do ambiente ou realizar tarefas específicas, operando, na maioria dos casos, alimentados por baterias.

Um exemplo típico de aplicação desses dispositivos é o seu uso em redes de sensores destinadas ao monitoramento e alerta de incêndios ou queimadas florestais. Nesse contexto, os sensores têm o objetivo de coletar dados ambientais, como aumento da temperatura e presença de fumaça. Quando essas variáveis atingem valores críticos, os dispositivos são programados para emitir notificações em tempo real, possibilitando que equipes de combate a incêndios atuem imediatamente para controlar e extinguir as chamas.

Em condições normais, as variáveis ambientais geralmente não sofrem alterações bruscas ao longo do dia, e a transmissão contínua de dados repetitivos pode levar à redução da vida útil da bateria. Assim, um sistema adequado para esse tipo de cenário deve ser capaz de filtrar e eliminar informações redundantes desnecessárias. Para minimizar o impacto do *overhead*, é essencial agrupar a maior quantidade possível de informações antes de efetuar a transmissão, o que, por sua vez, contribui para diminuir o uso do rádio.

No entanto, o emprego de um filtro, pode resultar em longos períodos sem a transmissão de dados pelo dispositivo, levando o sistema a presumir a perda de comunicação. Portanto, torna-se fundamental que o sistema seja capaz de emitir sinais periódicos de *"keep-alive"*, agrupar os dados com um filtro eficaz para evitar a acumulação de informações desnecessárias e notificar o servidor imediatamente ao detectar uma condição crítica.

Considerando como exemplo um sensor para detecção de incêndios florestais, as informações a serem captadas são: temperatura, umidade e detecção de fumaça. Levando em conta, que esses dados sejam transmitidos em um *payload* de tamanho fixo, cada informação ocuparia 5 bytes. Adicionando, um byte extra para o caractere de terminação de pacote, a carga útil total para esse quadro seria de 16 bytes. A Figura 16 ilustra a composição da carga útil de um pacote conforme essa especificação.

A. Ethernet

Com base nos cálculos fornecidos em [10] e as informações sobre os protocolos apresentadas na seção 2.1, é possível calcular a porcentagem de sobrecarga, para o exemplo de detecção de queimadas, da seguinte forma:

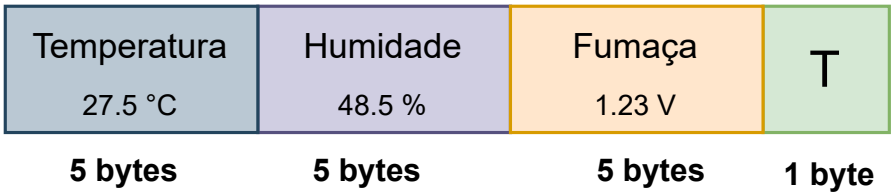

Figura 16 – Exemplo de dados a serem transmitidos

FONTE: Elaboração Própria

$$\text{Percentual\_sobrecarga} = \left(\frac{\text{Sobrecarga\_eth}}{\text{Sobrecarga\_eth} + \text{Carga Útil}} \right) \times 100 \tag{3.2}$$

Onde:

$$\text{Sobrecarga\_eth} = 78 \text{ bytes}$$
$$\text{Carga Útil} = 16 \text{ bytes}$$

Substituindo os valores na equação:

$$\text{Percentual\_sobrecarga} = \left(\frac{78}{78 + 16} \right) \times 100$$
$$\approx 82,98\%$$

Dessa forma, considerando uma carga útil de 16 bytes, a porcentagem de sobrecarga calculada para o quadro é de aproximadamente 82,09%.

B. LoRa

De maneira similar, é possível calcular o percentual de overhead para um pacote LoRa. A carga útil máxima de um pacote LoRa é definida pelo máximo ToA (*Time on Air*) ou tempo máximo de transmissão permitido pela regulamentação. O tempo máximo que o pacote permanece no ar é definido por vários fatores, incluindo largura de banda, código de correção de erros e fator de espalhamento. Para este cálculo, considerou-se uma carga útil máxima de 242 bytes, que é uma configuração comum com fator de espalhamento de 7, largura de banda de 125 kHz e um preâmbulo padrão de 8 bytes [59]. Assim, para este cenário, e somando o CRC e cabeçalho apresentado na Figura 11 obtemos o seguinte resultado:

$$\text{Sobrecarga\_LoRa} = 12,5 \text{ bytes}$$
$$\text{Carga Útil} = 16 \text{ bytes}$$

Substituindo os valores na equação:

$$\text{Percentual\_sobrecarga} = \left(\frac{12,5}{12,5+16}\right) \times 100$$
$$\approx 43,86\%$$

Portanto, considerando uma carga útil de 16 bytes, a sobrecarga percentual para o envio de um pacote é de aproximadamente 43,86%. É importante destacar que esses cálculos não levam em consideração retransmissões e outras potenciais fontes de sobrecarga no processo de transmissão.

3.6 Solução proposta

Como discutido nos capítulos anteriores, para dispositivos que operam com baterias, a otimização do consumo de energia é importante para prolongar sua vida útil e garantir a eficiência operacional. Neste contexto, a etapa de transmissão de dados é um desafio crítico na gestão eficiente do consumo energético.

Um dos fatores principais que contribui para o consumo de energia é a necessidade de ativação do módulo de rádio para a comunicação. Este consumo envolve a alimentação de diversos componentes internos do rádio, como transmissores, receptores e amplificadores. Portanto, o uso frequente do rádio pode levar a uma rápida descarga da bateria.

Adicionalmente, a comunicação via rádio apresenta desafios inerentes. As transmissões são frequentemente intermitentes, exigindo que o dispositivo saia do modo de baixo consumo para estabelecer uma conexão, transmitir ou receber dados e, posteriormente, retornar ao estado de baixa energia. Essas transições recorrentes entre modos ativos e inativos intensificam o consumo de energia devido à sobrecarga associada à mudança de estados [60].

Gerenciar eficientemente esses desafios requer a implementação de estratégias cuidadosas de gerenciamento de energia. Dispositivos de baixo consumo devem equilibrar entre os benefícios de manter a conectividade e os custos energéticos relacionados à transmissão de dados e à operação do rádio.

Para abordar essa questão, é proposto um algoritmo na forma de um driver portátil, que tem como finalidade gerenciar os dados a serem transmitidos. O objetivo principal

é agrupar dados, minimizando a frequência de ativação do rádio, especialmente em cenários de transmissão de dados redundantes. A Figura 17 apresenta o fluxograma da solução proposta, descrita a seguir.

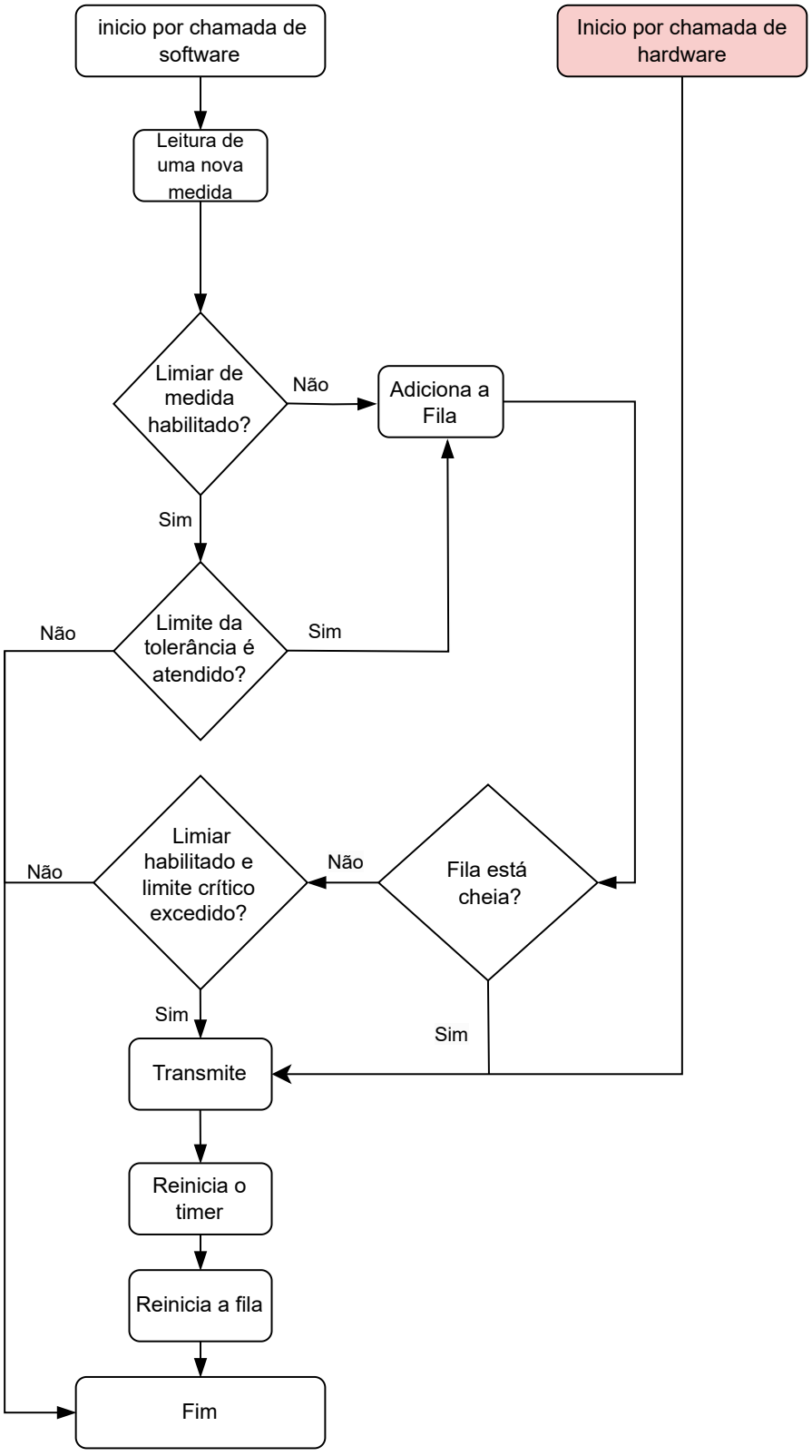

Figura 17 – Fluxograma do driver proposto

FONTE: Elaboração Própria

Utiliza-se uma fila para realizar o agrupamento dos dados. Com o intuito de suprir as necessidades de uma ampla gama de aplicações, o sistema possibilita a ativação ou a desativação de determinadas funcionalidades. Entre essas funcionalidades, destaca-se o limiar de estabilidade, cuja função é eliminar redundâncias desnecessárias, e o limiar crítico, que tem a capacidade de indicar variações abruptas na variável observada. O valor em porcentagem dos limiares deve ser configurado pelo usuário. Dependendo da configuração estabelecida, é possível identificar três situações distintas:

1. **Modo Normal:** Neste modo, todas as medidas são adicionadas à fila. A transmissão ocorre quando a fila atinge sua capacidade máxima ou quando excede o tempo limite, sendo a transmissão realizada de forma imediata. O principal objetivo deste modo é organizar os dados de maneira eficiente, visando a economia de energia.

2. **Limiar de estabilidade:** Somente serão adicionadas à fila as medidas em que a variância, em relação à medida anterior, seja igual ou superior ao limiar estabelecido. Medidas que não atendam a este critério serão descartadas. A transmissão ocorre quando a fila estiver cheia ou quando exceder o tempo limite. A eliminação de informações redundantes contribui para a redução do consumo de energia, tornando mais provável a transmissão devido ao tempo limite excedido.

3. **Limiar Crítico:** No caso de a variância crítica, em relação à medida anterior, ser ultrapassada, a medida é adicionada à fila, e os dados nela presentes são transmitidos imediatamente. Esse sistema implica uma penalidade em termos de consumo de energia, mas, em contrapartida, proporciona um aumento na confiabilidade.

O acesso ao driver pode ser efetuado de duas maneiras distintas: por meio de chamada de software ou por chamada de hardware. A chamada via software deve ser realizada sempre que a aplicação registrar uma nova medida. Em contrapartida, a chamada por hardware é acionada quando o tempo limite, previamente estipulado pelo usuário, é excedido. Este teste, executado via hardware, é ilustrado na Figura 17 em vermelho. Este procedimento tem como objetivo proporcionar previsibilidade e monitorar o desempenho do sistema. Desta forma, mesmo que não se verifiquem as condições como fila cheia ou atingimento do limiar crítico, o dispositivo enviará periodicamente um sinal de "keep-alive"contendo os dados presentes na fila até o momento.

Além das configurações apresentadas anteriormente, é necessário realizar outras configurações, como o comprimento máximo da fila, o tamanho do buffer de transmissão, o tamanho e a prioridade do processo de transmissão. A estrutura que armazena os dados também pode ser personalizada de acordo com as necessidades específicas de cada aplicação.

A Figura 18 apresenta o diagrama de sequência para a inicialização do driver proposto. O processo inicia-se com a tarefa de aplicação `App_task`, que executa a função `driver_init()`. Esta função representa o ponto de entrada para o processo de inicialização do driver. No componente `driver`, o primeiro passo é a chamada do método `Task_transmission_handler_create()`, que cria o processo responsável pela transmissão dos dados. A segunda etapa envolve a criação de uma fila por meio do método `xQueueCreate()`, que será utilizada para armazenar os dados até o momento da transmissão. Finalmente, ocorre a inicialização de um temporizador `xTimer_init()`, destinado à supervisão do dispositivo.

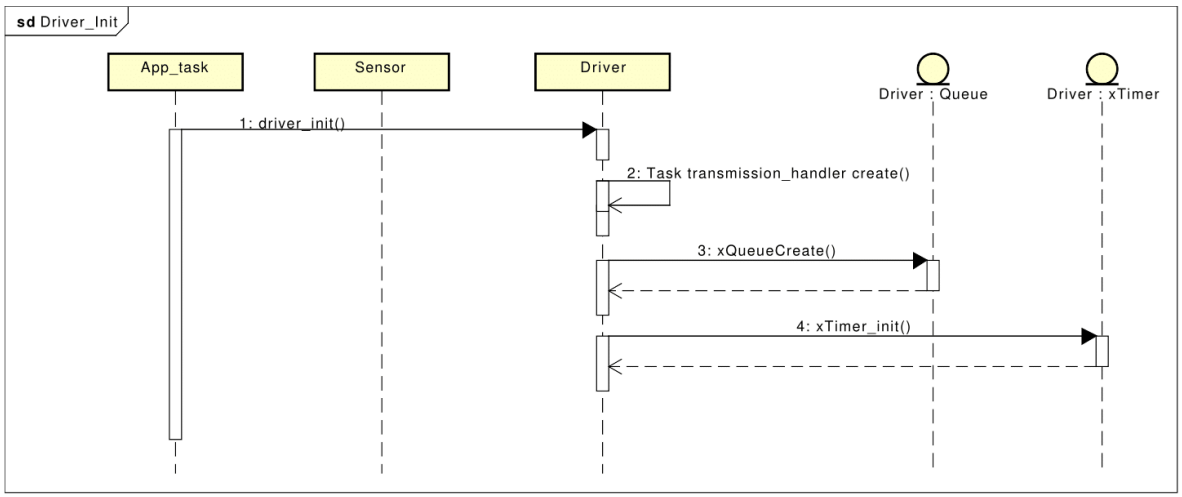

Figura 18 – Diagrama de sequência da inicialização

FONTE: Elaboração Própria

A Figura 19 ilustra a interação das funções do driver, destacando o fluxo de controle e comunicação entre os componentes do sistema. Este é representado por meio de um diagrama de sequência para os três modos nos quais uma transmissão pode ocorrer. A primeira situação representa uma transmissão devido a fila estar sem espaços disponíveis. A segunda, uma transmissão em função do estouro do temporizador. O terceiro caso ocorre quando há uma variação brusca na medida, ultrapassando o limite crítico. Este limite é muito importante, pois indica ao sistema que ocorreu uma grande variação em um curto intervalo de tempo, podendo sinalizar um evento relevante no contexto de sensores. Devido à possibilidade de a fila ser acessada de diferentes pontos do driver, sua proteção foi implementada utilizando um mutex (*Mutual exclusion*). O mutex é uma estrutura de controle de concorrência empregada para garantir que apenas um processo por vez possa acessar um recurso compartilhado, sendo fundamental para prevenir conflitos de acesso que poderiam corromper os dados ou causar resultados inesperados. A área sob a proteção do mutex é indicada em vermelho.

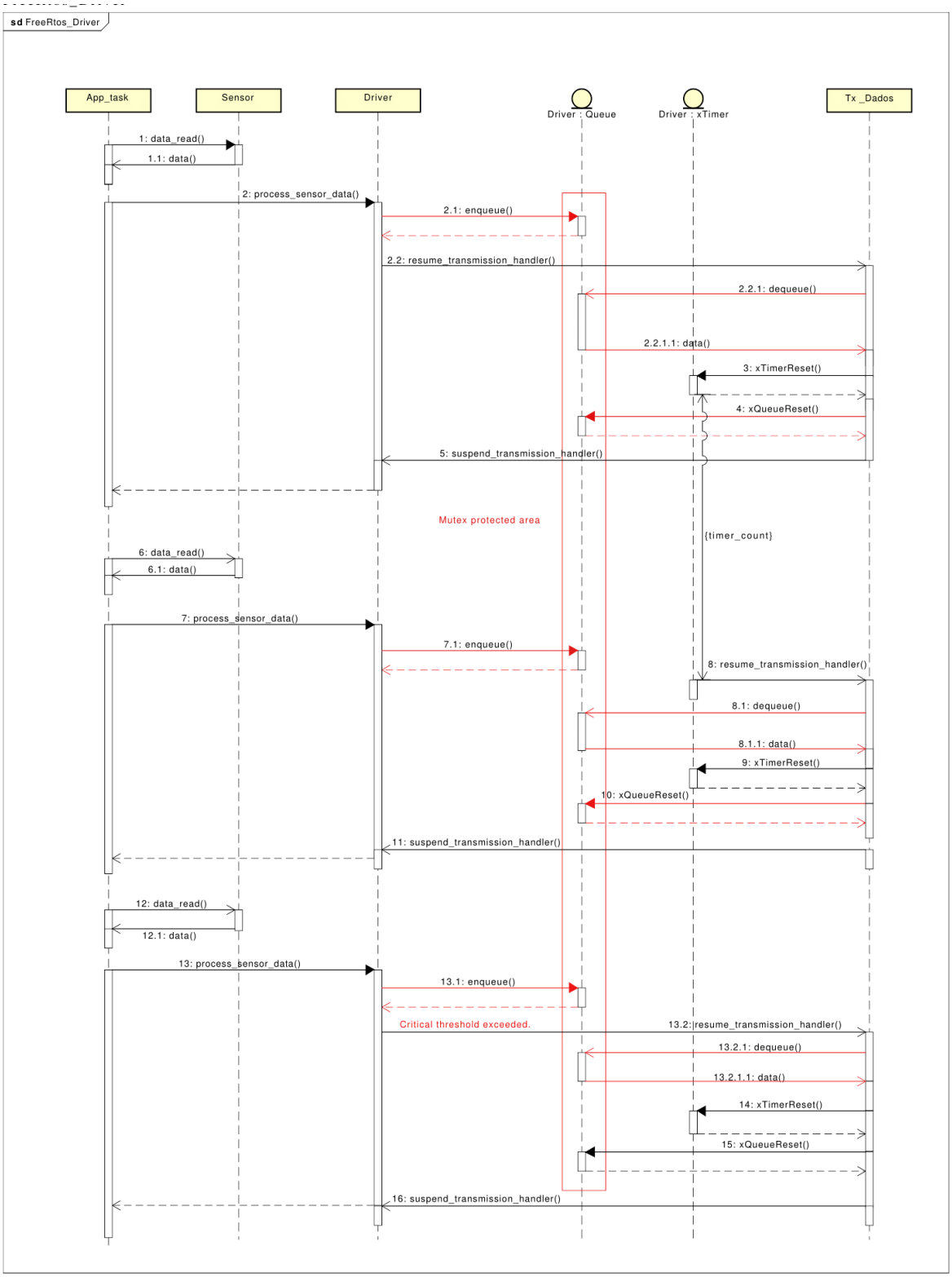

Figura 19 – Diagrama de sequência do modo de operação

FONTE: Elaboração Própria

Na Figura 20, é apresentado um diagrama de classes que auxilia no entendimento do funcionamento do modelo proposto. No entanto, sua implementação exata não é viável, uma vez que os códigos são desenvolvidos em linguagem C e, portanto, não suportam diretamente os conceitos de orientação a objetos. Neste diagrama, identificam-se três componentes principais: `Main`, `Driver` e `Interface de Transmissão`, cada um com diferentes responsabilidades e interações. A classe `Main` implementa os métodos da aplicação do usuário e estabelece comunicação com a classe `Driver`. Na classe `Driver`, encontram-se os métodos e atributos específicos do driver proposto. A seguir, segue a descrição de cada um desses métodos:

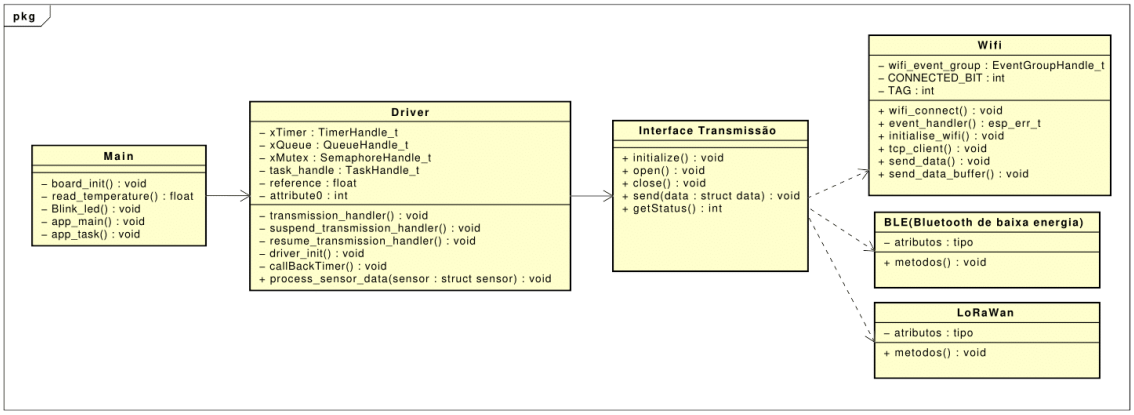

Figura 20 – Diagrama de classes

FONTE: Elaboração Própria

- **`transmission_handler(void)`**: Processo encarregado de remover os elementos da fila e efetuar a transmissão.
- **`suspend_transmission_handler(void)`**: Responsável pela suspensão do processo `transmission_handler()`.
- **`resume_transmission_handler(void)`**: Método utilizado para reiniciar o processo `transmission_handler()`.
- **`driver_init(void)`**: Método destinado à inicialização do driver.
- **`process_sensor_data(struct sensor)`**: Método encarregado do tratamento dos dados provenientes do sensor, incluindo a avaliação dos limiares e a determinação do momento para a transmissão. Recebe como argumento uma estrutura denominada sensor. Deve ser chamado na aplicação do usuário.

Considerando que o objetivo do driver é ser genérico, independentemente do tipo de tecnologia de transmissão empregada, a classe `Interface de Transmissão` disponibi-

liza métodos genéricos para gerenciar a abertura, o fechamento e o envio de dados. Essa abordagem promove a portabilidade do driver, facilitando seu uso com diferentes tecnologias de comunicação. Como exemplos de tecnologias compatíveis com o driver, tem-se Wi-Fi, BLE e LoRaWAN.

A Tabela 3 apresenta uma comparação das porcentagens de sobrecarga com e sem o uso do driver. Para o cálculo, considerou-se que a transmissão será realizada quando a fila estiver completa. Para minimizar a quantidade de sobrecarga, deve-se utilizar a máxima carga útil disponível. As capacidades máximas de carga útil para os protocolos Ethernet e LoRa são de 1460 e 242 bytes, respectivamente.

Tabela 3 – Comparação entre as porcentagens de sobrecarga.

Protocolo	Sobrecarga sem Driver	Sobrecarga com Driver
Ethernet	82,98%	5,07%
Lora	43,86%	4,91%

Portanto, a quantidade de overhead no protocolo Ethernet foi reduzida de 82,98% para 5,07%, o que representa uma diminuição de 93,89%. No caso do protocolo LoRa, observou-se uma redução de 43,86% para 4,91%, equivalente a uma diminuição de 88,81%.

4 Resultados e Discussões

Este capítulo é dedicado à avaliação crítica, tanto em aspectos teóricos quanto práticos, do uso do driver proposto para economia de energia em sistemas embarcados que operam sob o FreeRTOS. A análise foca no consumo de energia e é realizada por meio da implementação prática de uma comunicação cliente-servidor utilizando a tecnologia Wi-Fi. A metodologia adotada para analisar a eficiência energética inclui medições do consumo de energia por parte do sensor, atuando como cliente, durante a transmissão de dados para o servidor.

4.1 Resultados comunicação cliente/servidor

Visto que a MTU é de 1460 bytes, optou-se por uma estrutura de pacote de 12 bytes para a transmissão, contendo o número de identificação do dispositivo, o tipo de medida e o valor medido, cada um com 4 bytes. O tamanho máximo do pacote, respeitando a MTU, foi estabelecido em 1200 bytes. Assim, para a transmissão de 120 bytes, o tamanho da fila no driver é configurado para um máximo de 10, e a transmissão ocorre apenas quando a fila estiver completamente cheia. O tempo limite também foi ajustado para que não seja excedido antes do preenchimento total da fila.

Com o ESP32 operando o driver e realizando uma tarefa de medição de temperatura, o consumo médio de corrente é de 26 mA. No entanto, quando o Wi-Fi está conectado, ocorrem varreduras periódicas. Conforme ilustrado na Figura 21, o período de cada varredura e o consumo de corrente associado são, respectivamente, de 100 ms e 120 mA. A duração de cada varredura é aproximadamente 5 ms.

Este consumo pode ser minimizado ao empregar a função de economia de energia do Wi-Fi. No entanto, a ativação dessa função resulta na desconexão do dispositivo do ponto de acesso. Neste cenário, o tempo necessário para a reconexão pode variar devido a vários fatores da rede, como latência, força do sinal, processos de varredura, tempo de autenticação e ocorrência de colisões. Assim, na implementação, optou-se por manter a conexão Wi-Fi ativa desde a inicialização do hardware, e estabelecendo a conexão com o servidor, via *socket*, apenas quando necessário para transmitir dados. Após a transmissão, a conexão é encerrada. Essa estratégia simplifica a implementação, considerando que o driver é genérico e a manutenção da conexão ativa é uma condição específica do protocolo Wi-Fi.

O melhor resultado de taxa de transferência de dados alcançado pelo ESP32 utilizando o Wi-Fi, conforme testes realizados em laboratório, é de 20 Mbit/s [61]. Assim,

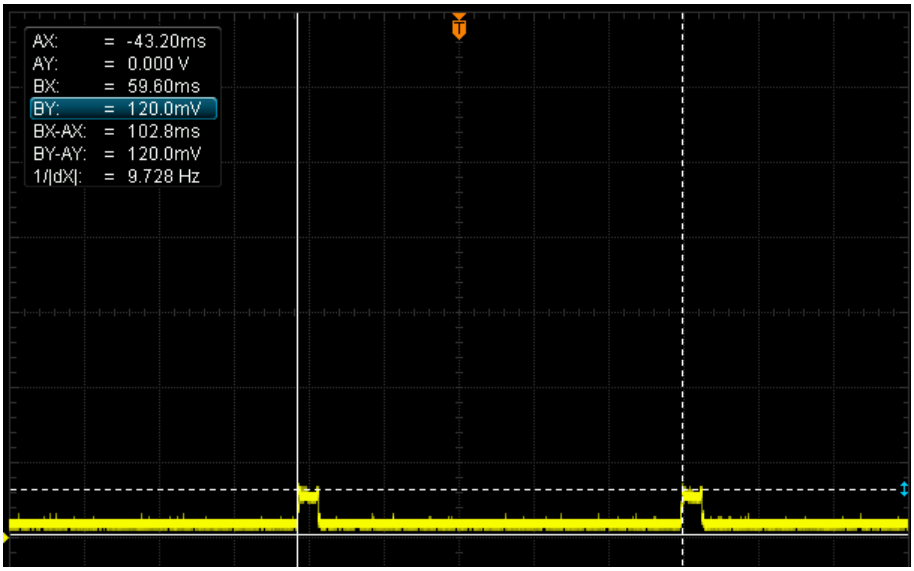

Figura 21 – Tempo e Consumo da varredura do Wifi

FONTE: Elaboração Própria

a Tabela 4 apresenta os valores teóricos para o tempo necessário à transmissão de um pacote em dois cenários distintos: o Cenário 1, que envolve a transmissão de um pacote com 12 bytes; e o Cenário 2, que envolve a transmissão de um pacote com 1200 bytes.

Tabela 4 – Tempo para transmissão de dados

Informação	**Cenário 1**	**Cenário 2**
Carga útil	12 bytes	1200 Bytes
Cabeçalhos	78 bytes	78 Bytes
Comprimento do pacote	720 Bits	10224 Bit
Taxa de transmissão	20Mbit/s	20Mbit/s
Tempo para envio de um pacote	**36µs**	**511,2µs**

Portanto, para transmitir a mesma quantidade de carga útil, são necessários 100 pacotes do cenário 1 em comparação com o cenário 2. Isso resulta em um tempo com o transmissor ligado de 3,6 ms, o que equivale a um período sete vezes maior do que na transmissão de um único pacote do cenário 2. Como discutido anteriormente, em dispositivos de baixo consumo de energia, o tempo em que o rádio permanece ligado representa a principal fonte de gasto energético.

No entanto, o tempo médio para a transmissão de um pacote é influenciado por muitas variáveis da rede Wi-Fi. Entre essas variáveis, destacam-se o congestionamento da rede, retransmissões, ocupação do canal, configurações de qualidade de serviço e segurança. A Figura 22 apresenta o tempo necessário para a transmissão de um pacote com 1200 bytes de carga útil, com um tempo total de 96 ms para concluir a transmissão do pacote.

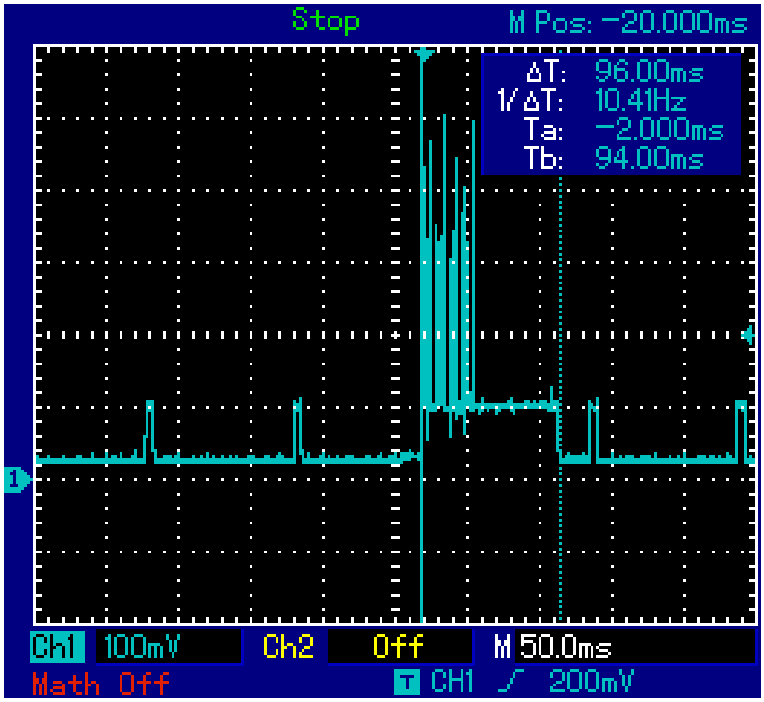

Figura 22 – Transmissão de 1 Pacote com 1200Bytes

FONTE: Elaboração Própria

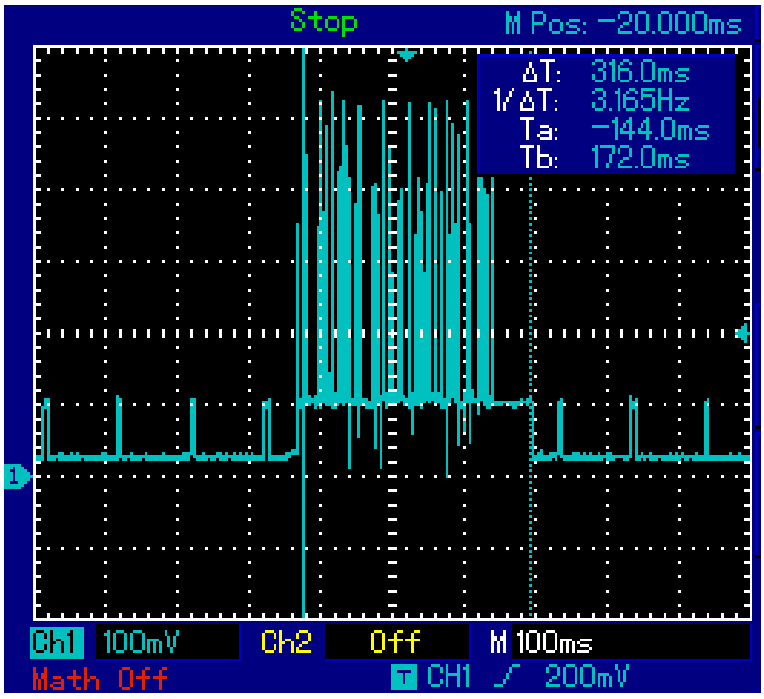

Figura 23 – Transmissão de 10 Pacotes com 120 Bytes

FONTE: Elaboração Própria

Devido às características do protocolo 802.11, as vantagens da economia de energia em relação à *overhead* existem, porém são relativamente irrelevantes quando comparadas aos tempos relacionados à latência da própria rede.

Tabela 5 – Dados das amostragens em milissegundos

	Amostragem 1	Amostragem 2	Amostragem 3
1	68	312	2600
2	114	280	3200
3	108	256	3200
4	90	364	1920
5	84	316	2280
6	68	260	2600
7	92	336	2960
8	96	400	2440
9	68	356	2240
10	84	416	2320
11	136	268	2160
12	72	468	2360
13	70	284	2520
14	98	264	2560
15	178	296	2240
16	84	248	2500
17	108	400	2220
18	86	284	2280
19	90	252	2360
20	138	252	2480
21	114	252	2100
22	68	364	2240
23	84	260	2440
24	92	276	2210
25	158	356	2080
26	96	256	3110
27	84	304	1980
28	98	252	2440
29	92	273	2600
30	90	416	2230
Média	96,93	310,70	2429
Mediana	91	284	2360
Desvio padrão	26,40	61,99	328,16

A Tabela 5 apresenta a variação dos dados, em milissegundos, referentes à transmissão de um pacote com 1200 bytes, 10 pacotes de 120 bytes e 100 pacotes de 12 bytes, representados pelas amostragens 1, 2 e 3, respectivamente. Na transmissão de 10 e 100 pacotes, o *socket* foi aberto e fechado após o envio de cada pacote.

Devido às variações no meio de transmissão do Wi-Fi, foram realizadas 30 amostragens dos sinais para determinar o tempo médio de transmissão em cada cenário. A Figura 22 representa a medição 26 da amostragem 1. A Figura 23 apresenta a quinta medição da amostragem 2, enquanto a Figura 24 refere-se à medição 17 da amostragem

3. As imagens selecionadas foram aquelas que mais se aproximaram da média observada.

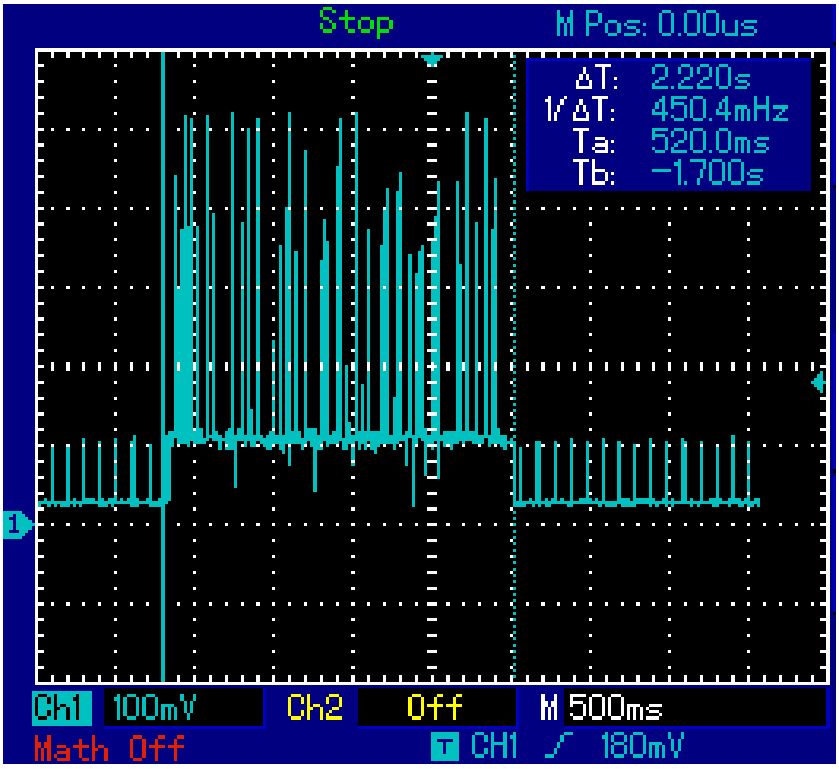

Figura 24 – Transmissão de 100 Pacotes com 12 Bytes

FONTE: Elaboração Própria

Na amostragem 1, a média dos tempos de transmissão foi de 96,93 ms, a mediana foi de 91 ms e desvio padrão foi de 26,40 ms. Na amostragem 2, a média foi de 310,70 ms, a mediana foi de 284 ms e o desvio padrão foi de 61,99 ms. Já na amostragem 3 a média foi de 2,43 s, a mediana foi de 2,36 s e o desvio padrão foi de 328 ms.

Devido ao comportamento não determinístico da rede Wi-Fi, é notável que o desvio padrão aumenta à medida que o número de pacotes enviados é maior. Essa variação também pode ser constatada através da representação da distribuição dos dados através dos gráficos boxplots apresentados nas Figuras 25, 26 e 27.

Portanto, a transmissão de pacotes com pouca informação requer o envio de uma maior quantidade de dados pela rede, o que aumenta a probabilidade de retransmissão de pacotes e, consequentemente, eleva o consumo de energia.

Para analisar o consumo energético, é essencial levar em consideração a variação temporal da corrente do dispositivo, a qual depende da atividade executada pelo dispositivo em cada instante de tempo. Portanto, o valor médio de um sinal de corrente variável no tempo pode ser determinado pela seguinte equação:

$$I_{\text{média}} = \frac{1}{T} \int_0^T i(t)\, dt \tag{4.1}$$

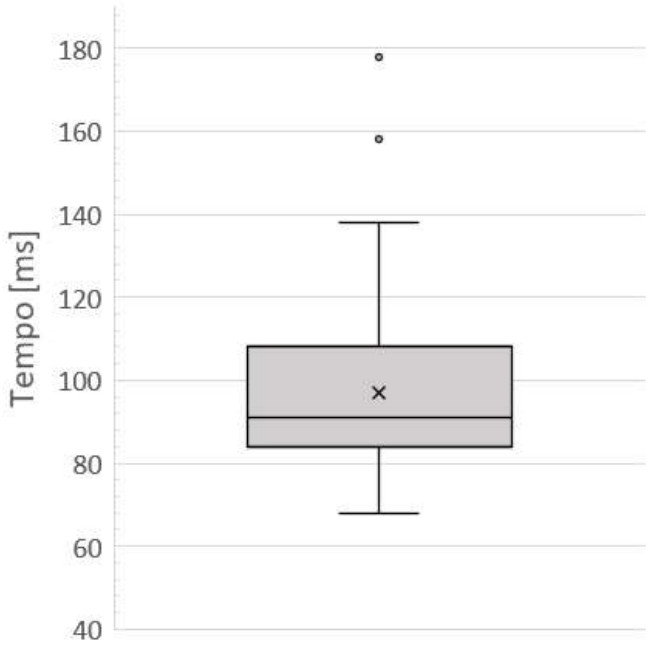

Figura 25 – Boxplot da representação da distribuição dos dados da Amostragem 1

FONTE: Elaboração Própria

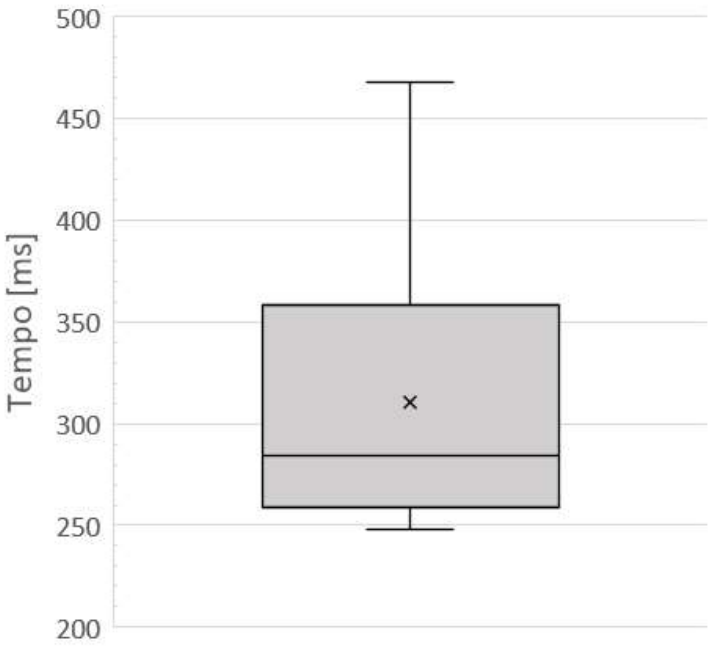

Figura 26 – Boxplot da representação da distribuição dos dados para Amostragem 2

FONTE: Elaboração Própria

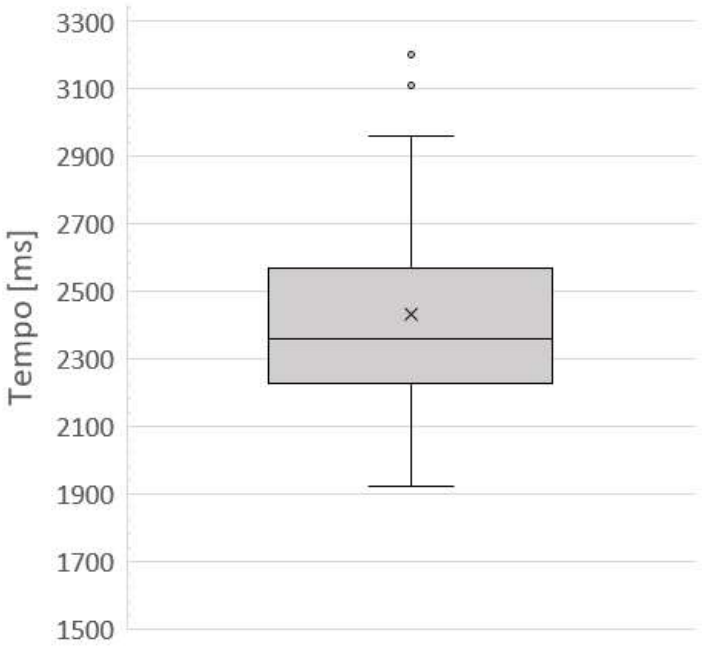

Figura 27 – Boxplot da representação da distribuição dos dados para Amostragem 3

FONTE: Elaboração Própria

No entanto, o sinal de corrente pode apresentar discontinuidade, a equação 4.1 pode ser reescrita na forma de um somatório, conforme apresentado na equação 4.2.

$$I_{\text{média}} = \frac{1}{N}\sum_{k=1}^{N} i_k \tag{4.2}$$

Assim, para avaliar a corrente média nos casos das amostragens 1 e 3 da Tabela 5, é necessário determinar as correntes tanto quando o dispositivo está em processo de transmissão quanto quando está apenas realizando o sensoriamento do ambiente. Com isso, o cálculo é dado por:

$$I_{\text{média}} = \frac{1}{N}\sum_{k=1}^{N} C_t + C_s \tag{4.3}$$

Onde C_t é a corrente na transmissão e C_s é a corrente média no sensoriamento. Através da Figura 29, observa-se que o consumo médio da corrente para realizar o sensoriamento é de 26 mA. Na Figura 28, constata-se que o consumo médio de corrente de transmissão é de aproximadamente 125 mA, desconsiderando os picos de corrente. Como esses picos ocorrem em intervalos curtos de tempo e, consequentemente, é difícil medir o seu valor médio, eles serão desconsiderados em uma análise inicial. Para um intervalo de tempo de $60s$, considerando o tempo de transmissão de 100 pacotes com 120 bytes, temos:

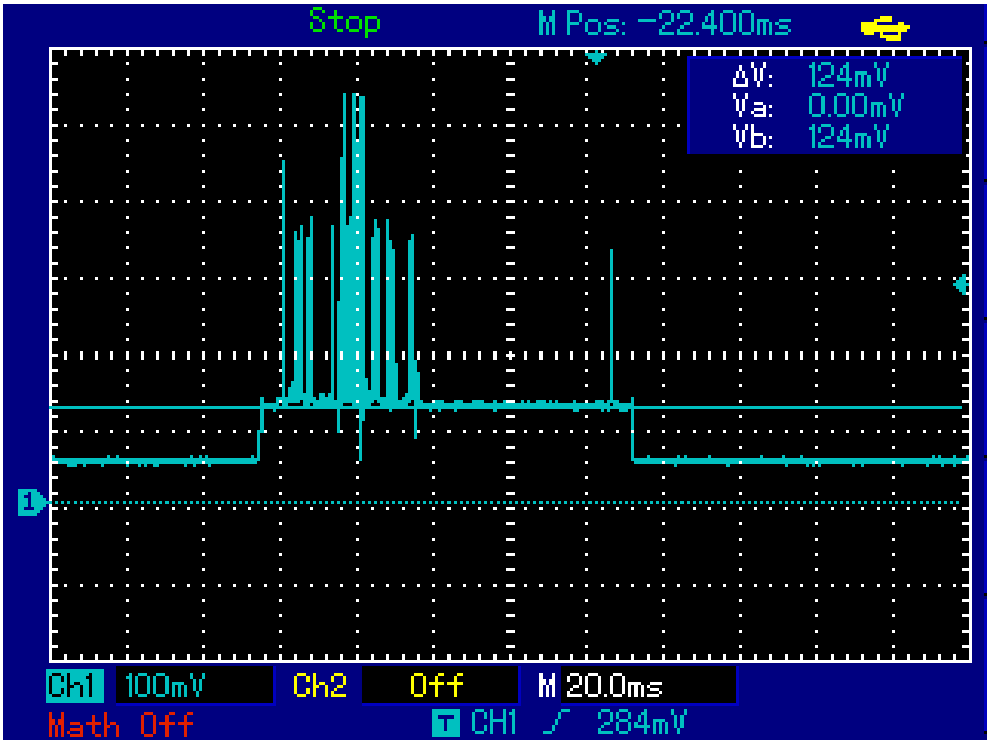

Figura 28 – Corrente na transmissão

FONTE: Elaboração Própria

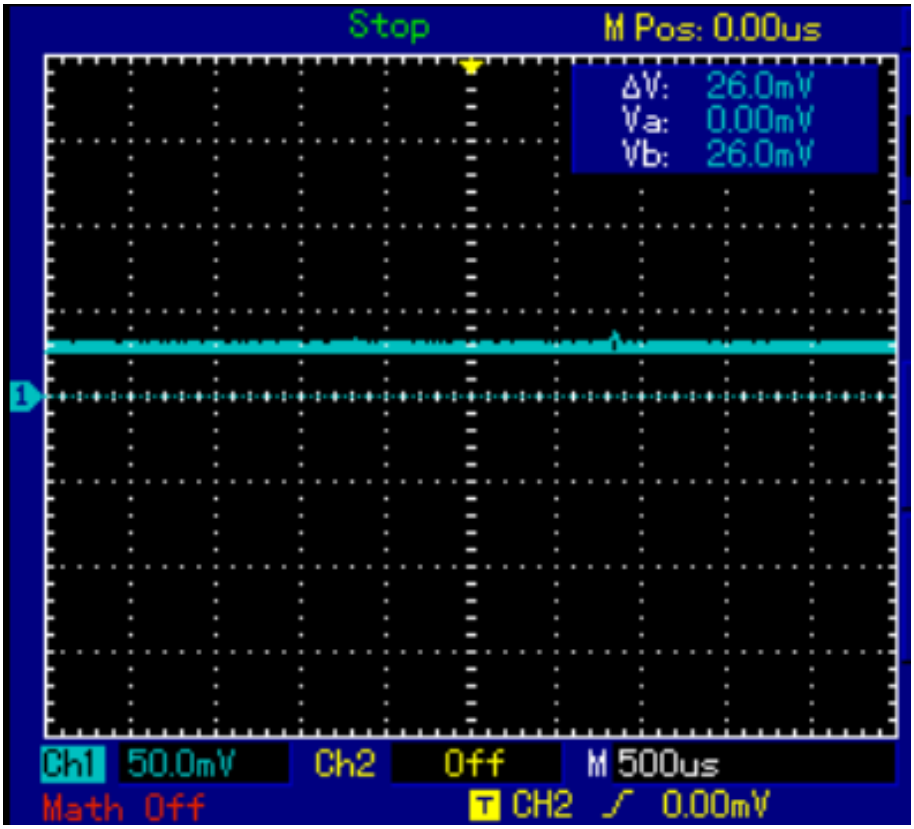

Figura 29 – Corrente sensoriamento

FONTE: Elaboração Própria

$$I_{\text{média}} = \frac{1}{60}(2,429 * 125) + (60 - 2,429) * 26 \tag{4.4}$$

$$I_{\text{média}} \approx 30,01mA \tag{4.5}$$

De forma análoga, para o valor médio da amostragem 3 com a transmissão de um pacote com 1200 bytes, tem-se:

$$I_{\text{média}} = \frac{1}{60}(0,096933 * 125) + (60 - 0,096933) * 26 \tag{4.6}$$

$$I_{\text{média}} \approx 26,16mA \tag{4.7}$$

Portanto, em uma análise inicial e conservadora, o consumo de corrente quando os dados são agrupados é aproximadamente 12,82% menor.

5 Conclusão

A dissertação fundamentou-se na otimização da transmissão de dados em dispositivos IoT de borda, com foco na economia de energia por meio da redução da frequência de transmissão utilizando técnica de agrupamento de dados. Essa abordagem é justificada pelo esperado crescimento no mercado de IoT e com isso o surgimento de novas aplicações. O consumo de energia representa um desafio significativo em dispositivos de borda, uma vez que possuem capacidade de bateria limitada e são frequentemente instalados em áreas remotas.

O estudo abordou as características essenciais dos protocolos de comunicação comumente empregados em IoT. Além disso, foi realizada uma análise da introdução de *overhead* em duas tecnologias amplamente utilizadas: Ethernet e LoRa. Com isso, foi possível constatar que, ao enviar um pacote com 16 bytes no protocolo ethernet, a porcentagem de sobrecarga é de aproximadamente $82,98\%$. Já ao utilizar o protocolo LoRa para enviar a mesma quantidade de carga útil, o percentual de sobrecarga é de $43,86\%$.

No decorrer deste trabalho, foi desenvolvido um algoritmo na forma de um driver portátil destinado a gerenciar os dados a serem transmitidos. O objetivo é agrupar as mensagens para reduzir a frequência de ativação do rádio para transmissão, organizar os dados de forma estruturada e segura, evitar o armazenamento de dados redundantes desnecessários, identificar variações abruptas e enviar "keep-alive".

O armazenamento de dados redundantes é evitado através de um limiar de estabilidade. Grandezas físicas frequentemente monitoradas, como temperatura, geralmente não apresentam variações bruscas em curtos intervalos de tempo. É possível definir a porcentagem de variação para que uma amostragem seja armazenada, contribuindo para a redução da quantidade de memória necessária e, consequentemente, do custo do produto final.

O limiar crítico informa o sistema sobre uma brusca variação na grandeza sendo monitorada, enviando os dados armazenados até o momento imediatamente. Isso permite que o sistema atue de forma rápida e eficiente. Embora o sistema possa sofrer uma penalidade em relação ao consumo, há um aumento na confiabilidade. Dependendo da aplicação, o dispositivo pode permanecer muito tempo sem comunicação. Portanto, para manter uma conexão ativa entre o dispositivo e o sistema, é possível configurar um tempo limite, auxiliando o sistema a manter a estabilidade e detectar falhas de comunicação, ou seja, realizar o supervisionamento dos dispositivos.

O driver foi implementado na placa de desenvolvimento ESP32-DEVKITV1, que integra um microcontrolador e comunicação sem fio Wi-Fi. O código fonte está disponível

em um repositório no GitHub. Adotou-se o cenário de comunicação cliente/servidor para a transmissão de dados. A análise do tempo necessário para transmitir cargas úteis de 12 e 1200 bytes, com base na vazão máxima observada em laboratório para o ESP32, indicou que o tempo requerido para enviar a mesma quantidade de dados é aproximadamente sete vezes maior para transmissões de pacotes agrupados em comparação com pacotes não agrupados.

No entanto, devido à característica orientada a conexão do protocolo Wi-Fi, o tempo de propagação de dados pela rede possui alta latência devido a vários fatores como congestionamento, retransmissões, ocupação do canal, configurações de segurança e qualidade do serviço. Portanto, para a rede Wi-Fi, as vantagens da economia de energia em relação à sobrecarga existem, porém são relativamente irrelevantes quando comparadas aos tempos relacionados à latência da própria rede. Em termos de consumo energético, utilizando a técnica de agrupamento de dados, a corrente média é $12,82\%$ menor. Isso ao longo do tempo reflete uma maior autonomia do dispositivo até a necessidade de troca da bateria.

5.1 Trabalhos Futuros

Com base nos resultados apresentados, há diversas oportunidades para futuros trabalhos e aprimoramentos na área da dissertação. Algumas sugestões de trabalhos futuros incluem:

- **Estudo de Protocolos Alternativos:** Implementar e comparar o desempenho de outros protocolos de comunicação para dispositivos IoT, especialmente os protocolos orientado a mensagens. Avaliar considerando os critérios de eficiência energética, latência e sobrecarga.
- **Avaliar o Timer Crítico:** Comparar o driver com diferentes intervalos de *keep-alive* para identificar o ponto crítico em que os benefícios do uso do driver deixam de ser perceptíveis.
- **Otimização do algoritmo e implementação de novas funcionalidades:** Aprimorar os algoritmos empregados no driver com o objetivo de reduzir ainda mais o consumo de energia, de modo que, além de gerenciar a transmissão de dados, o driver seja capaz de determinar momentos ociosos e entrar em modos de baixo consumo. Uma outra melhoria seria implementar no driver a possibilidade de realizar uma amostragem adaptativa. Pensando em portabilidade, deixar as funções mais agnosticas através de estruturas de ponteiro de função.
- **Otimização do Algoritmo e Implementação de Novas Funcionalidades:** Aprimorar os algoritmos utilizados no driver com o objetivo de minimizar ainda

mais o consumo de energia. Além de gerenciar a transmissão de dados, espera-se que o driver possa identificar períodos ociosos e ativar modos de baixo consumo. Uma melhoria adicional seria implementar no driver a capacidade de realizar amostragem adaptativa. Visando a portabilidade, pretende-se tornar as funções mais agnósticas por meio de estruturas de ponteiros de função.

- **Análise de Consumo em Larga Escala:** Avaliar o consumo de energia e desempenho do driver em ambientes com um grande número de dispositivos IoT interconectados, considerando possíveis impactos na rede.
- **Estudo de Casos Específicos:** Aplicar o driver a cenários específicos de IoT, como agricultura inteligente, monitoramento de queimadas ou cidades inteligentes, para compreender melhor seu desempenho em contextos específicos.

Estas sugestões de trabalhos futuros podem contribuir significativamente para o avanço da área, consolidando as descobertas da dissertação e estendendo suas aplicações para cenários diversificados.

Apêndices

APÊNDICE A – Protocolos e Tecnologias de comunicação

A.1 Protocolos

Os protocolos aplicados em IoT são desenvolvidos para prover funcionalidades específicas, como baixo consumo de energia, eficiência de rede, suporte a dispositivos com recursos limitados, segurança, escalabilidade e interoperabilidade. Alguns dos protocolos notáveis nesse contexto incluem MQTT (*Message Queuing Telemetry Transport*), CoAP (*Constrained Application Protocol*), AMQP (*Advanced Message Queuing Protocol*) e DDS (*Data Distribution Service*). A seguir, é apresentada uma descrição breve de cada um desses protocolos.

AMQP

Desenvolvido em 2003 como um protocolo aberto, o Advanced Message Queuing Protocol (AMQP) tem como objetivo principal promover a interoperabilidade funcional entre clientes e *middleware* de mensagens. Amplamente utilizado em aplicações distribuídas, o AMQP oferece recursos fundamentais, como confiabilidade, eficiência e comunicação assíncrona entre sistemas. Suas vantagens incluem a garantia de transferência segura de dados críticos, interoperabilidade entre diferentes sistemas, comunicação ponto a ponto eficiente, flexibilidade de adaptação a diversos padrões e capacidade de estabelecer conexões seguras utilizando SSL (*Secure Socket Layer*). No entanto, o protocolo apresenta desvantagens, como incompatibilidade com versões mais antigas, maior complexidade se comparado a outros protocolos, exigência de mais largura de banda e falta de suporte para descoberta automática de recursos. Essas limitações devem ser avaliadas na escolha do AMQP para aplicações específicas, principalmente naquelas que demandam compatibilidade com sistemas mais antigos ou enfrentam restrições severas de recursos de rede [62][33][63].

CoAP

O CoAP é projetado para ser extremamente eficiente em termos de uso de recursos, como largura de banda e capacidade da rede. Possui uma comunicação rápida e assíncrona por meio do UDP (*User Datagram Protocol*), ideal em redes com grande número de dispositivos. Embora utilize UDP, o CoAP assegura integridade, segurança e privacidade das comunicações através do uso de DTLS (*Datagram Transport Layer Security*). Por utilizar UDP apresenta a desvantagem de não garantir confiabilidade na confirmação de entrega das mensagens [33][63].

MQTT

É um protocolo de mensagens de padrão aberto, desenvolvido pela IBM em 1999 e padronizado pela OASIS em 2013. Reconhecido por sua simplicidade, eficiência em economia de energia e largura de banda. Sua arquitetura, baseada no modelo de publicação/assinatura, permite que dispositivos denominados publicadores, enviem dados para um *broker* central, que os distribui apenas para os aplicativos interessados, os assinantes. As vantagens do MQTT, destacam-se sua leveza, ideal para dispositivos com recursos limitados; a rápida e confiável conectividade entre dispositivos e sistemas; e a flexibilidade do modelo assíncrono de comunicação. Entretanto, algumas desvantagens requerem atenção especial: a segurança pode ser vulnerável, mesmo com mecanismos de autenticação e criptografia, devido a falhas de configuração em algumas implementações. O gerenciamento adequado do *broker* é essencial para evitar riscos de segurança e falhas de desempenho. Em cenários de sobrecarga, a latência pode aumentar devido à falta de mecanismos robustos de controle de acesso. Para mitigar riscos, o MQTT oferece mecanismos de segurança, mas é fundamental que os desenvolvedores os implementem corretamente.

HTTP

O HTTP (*Hypertext Transfer Protocol*), um protocolo de comunicação baseado em texto e web, oferece recursos organizados e eficientes para realizar solicitações e obter respostas. Utiliza o TCP como transporte e os protocolos de segurança TLS (*Transport Layer Security*)/SSL. Seu uso em dispositivos IoT oferece vantagens, como facilidade de implementação, independência de plataforma e amplo suporte, permitindo troca estruturada de dados com servidores através da arquitetura RESTful, além de possibilitar comunicação segura por meio do HTTPS (*Hypertext Transfer Protocol Secure*). Contudo, suas limitações devem ser consideradas: o protocolo tem alta sobrecarga e pode apresentar latência, o que impacta negativamente em dispositivos com recursos limitados e em aplicações que exigem respostas em tempo real. Além disso, não é eficiente em termos de consumo energético, o que requer cautela em sua aplicação em dispositivos alimentados por bateria. A escalabilidade também pode ser um desafio em cenários com alto volume de tráfego [36].

TCP

O TCP é um protocolo de transmissão de dados amplamente utilizado e reconhecido por sua confiabilidade. Ele assegura a entrega ordenada e a integridade dos dados por meio da confirmação e retransmissão automática de pacotes perdidos. Além disso, gerencia o fluxo de dados para regular as taxas de transmissão e inclui mecanismos de verificação de erros para garantir precisão. Sua abordagem orientada à conexão oferece um processo previsível e consistente de troca de dados. No entanto, a confiabilidade do TCP pode resultar em uma sobrecarga que afeta o desempenho da rede, enquanto sua natureza orientada à conexão pode levar a latências, tornando-o menos adequado para

aplicações que exigem respostas em tempo real. Adicionalmente, a complexidade do TCP e a retransmissão automática podem não ser ideais para cenários que toleram perdas de dados. Por outro lado, o UDP pode ser uma escolha mais apropriada para esses contextos. Portanto, a decisão entre TCP e UDP deve ser baseada nas necessidades específicas da aplicação ou serviço em questão.

WebSocket

O protocolo WebSocket, desenvolvido como parte da iniciativa HTML5, é uma tecnologia de comunicação que aprimora canais sobre o TCP. Ao contrário de outros protocolos, o WebSocket estabelece uma sessão full-duplex assíncrona, permitindo a troca de mensagens entre cliente e servidor. Uma vez estabelecida a sessão, os cabeçalhos HTTP são descartados, o que contribui para a redução da sobrecarga. Pesquisas demonstram que o WebSocket apresenta menor latência em comparação com a técnica de polling HTTP. Embora não tenha sido originalmente projetado para dispositivos com recursos limitados, é seguro e eficiente. O subprotocolo WAMP (*Web Application Messaging Protocol*) oferece sistemas de mensagens de publicação/assinatura. Em resumo, o WebSocket é uma opção competitiva para diversos cenários de comunicação, especialmente aqueles que requerem troca de dados em tempo real [64].

A.2 Tecnologias de comunicação

Uma tecnologia de comunicação em IoT refere-se ao conjunto de meios, métodos e padrões utilizados para permitir a troca de informações entre dispositivos conectados. Cada uma das tecnologias é fundamentada em um ou mais protocolos que definem como a comunicação entre dispositivos ocorre. Portanto, essa relação intrínseca é fundamental para um sistema IoT. A seguir, são apresentadas as principais tecnologias de comunicação amplamente adotadas no contexto atual.

DDS

O DDS é uma tecnologia de comunicação em tempo real, baseada no modelo de mensagens de publicação e assinatura. Funciona como um *middleware* de padrão aberto, facilitando a troca de informações entre dispositivos sem a necessidade de um broker central, minimizando assim potenciais falhas por gargalos. O DDS utiliza métodos de descoberta, como o DCPS (*Data Centric Publisher Subscriber*) ou RTPS (*Real-Time Publisher Subscriber*), permitindo a comunicação direta entre os participantes sem conhecimento prévio entre eles. As principais vantagens do DDS incluem a oferta de qualidade de serviço e confiabilidade. Além disso, o DDS implementa medidas de segurança, como o TLS e DTLS, assegurando a autenticidade e integridade das informações trocadas. No entanto, desafios de segurança, especialmente vulnerabilidades no protocolo de handshake, podem comprometer a confidencialidade das conexões, exigindo a implementação cuidadosa de

medidas de segurança para proteger os ambientes que utilizam o protocolo [8][65].

BLE

O BLE, também conhecido como "Bluetooth Smart", é uma tecnologia de comunicação desenvolvida pelo *Bluetooth Special Interest Group*. Apresenta um alcance mais limitado e menor consumo de energia em comparação a outros protocolos concorrentes. A pilha de protocolos do BLE guarda semelhanças com a do bluetooth clássico, porém apresenta diferenças significativas, como o suporte a transferências rápidas de pequenos pacotes de dados, em vez de transmissão contínua. O BLE é amplamente aplicado em cenários de IoT, onde dispositivos móveis operam de forma eficiente em termos de energia, despertando periodicamente para receber dados [66]. Ele suporta requisitos de baixo ciclo de trabalho, com uma taxa de dados de 1 Mbps em 40 canais, cada um com largura de banda de 2 MHz. Pesquisas indicam que o BLE é significativamente mais eficiente em termos de consumo energético comparado a outros certos protocolos, tornando-o uma escolha adequada para comunicações em tempo real que exigem baixo consumo de energia. Por sua vez, BLE representa uma evolução do Bluetooth original, destinando-se a aplicações em que o baixo consumo de energia é primordial. Ele possui uma distância de cobertura de até 400m na versão 5.0 e técnicas de criptografia e autenticação avançadas. Diferente do Bluetooth clássico, o BLE suporta transferências rápidas de pequenos pacotes de dados, que contribui para uma menor latência e maior eficiência energética [67][36].

NFC

O NFC (*Near Field Communication*) é uma tecnologia de comunicação sem fio de curto alcance que permite a interação entre dispositivos móveis em distâncias de apenas alguns centímetros. Baseado em RFID, opera na faixa de frequência de 13,56 MHz. O NFC possui dois modos de operação: ativo, onde ambos os dispositivos geram campos magnéticos, e passivo, onde apenas um dispositivo gera o campo e o outro modula a carga para transferência de dados. Essa tecnologia é aplicável em transações seguras, como pagamentos e suporta comunicação bidirecional, sendo amplamente suportada por smartphones devido à sua facilidade de uso. Além disso, o NFC pode ser utilizado em conjunto com protocolos como ANT (*Adaptive Network Topology*) e ISA100.11a para comunicação de curto alcance. O NFC opera de forma ad-hoc, permitindo uma comunicação rápida e direta entre dispositivos. O ANT, por sua vez, é um protocolo sem fio mestre-escravo, especialmente utilizado em redes de sensores sem fio, operando nas frequências de 2,4 GHz e apresentando semelhanças com o *Bluetooth Low Energy* [26][67].

ZigBee

O Zigbee, uma tecnologia de comunicação sem fio baseada no padrão IEEE 802.15.4, desenvolvida pela Zigbee Alliance para redes de área pessoal PANs (*Personal Area Networks*). Operando em frequências como 868 MHz, 915 MHz e 2,4 GHz, o Zigbee abrange distâncias

de 10 a 100 metros, oferecendo taxas de dados de 20 kbps, 40 kbps e 250 kbps, respectivamente. Utiliza a técnica de modulação DSSS (*Direct Sequence Spread Spectrum*) para a transmissão de dados e CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*) para previnir colisões no meio de comunicação compartilhado.

Em uma rede Zigbee, são suportados três tipos de dispositivos: dispositivos totalmente funcionais FFD (*Full Function Device*), dispositivos com funcionalidade reduzidas RFD (*Reduced Function Device*) e um coordenador. Entre suas principais vantagens estão o baixo consumo de energia, o baixo custo, a rápida resposta, a auto-organização, a capacidade de suportar múltiplas topologias e alta segurança. Contudo, suas desvantagens incluem a baixa taxa de dados e limitações de memória. A rede Zigbee pode ser organizada em topologias estrela, árvore ou malha. A evolução para o padrão Zigbee PRO ampliou suas funcionalidades, incluindo gerenciamento de dispositivos filhos, segurança reforçada e topologias de rede alternativas. Isso torna o Zigbee uma escolha adequada para sistemas de IoT que requerem baixo consumo de energia e comunicação confiável [36][67].

Z-Wave

O Z-Wave é um padrão de comunicação de baixo consumo de energia, adequado para automação residencial, incluindo casas inteligentes e ambientes comerciais. Ele opera em distâncias que variam 30 a 100 metros, oferecendo comunicação ponto a ponto e capacidade de enviar mensagens curtas. O protocolo utiliza o método CSMA/CA para acesso à mídia e confirmações de mensagens para garantir uma transmissão confiável. O Z-Wave adota uma arquitetura de mestre-escravo, onde o mestre controla os dispositivos escravos e gerencia o agendamento das comunicaçãos. Para a segurança dos dispositivos conectados, emprega criptografia AES de 128 bits e suporta até 232 dispositivos em uma única rede [8][36].

APÊNDICE B – Tabela com as tecnologias e protocolos IoT sem fio

Tabela 6 – Tecnologias e Protocolos IoT sem fio

Tecnologia Sem Fio	**Taxa de Dados**	**Distância**
5G	5G: Faixa baixa 5G (600 - 700 MHz) velocidade de download um pouco mais altas que o 4G no momento: 30-250 Mbps. Faixa média 5G (2,5-3,7 GHz) atualmente permite velocidades de 100-900 Mbps. faixa alta 5G (25 - 39 GHz e frequências mais altas de até 80 GHz) alcança, no momento	5G de baixa banda possui uma cobertura similar ao 4G (dezenas de quilômetros), o 5G de banda média possui alcance de vários quilômetros. O 5G de alta banda tem alcance de centenas de metros até 1,5 km.
ANT+ Alliance	12.8 Kbit/s - 60kbit/s	≈ 30m
Bluetooth	≈ 2Mbps	≈ 50m
BLE (Bluetooth Low Energy) or Bluetooth Smart (Bluetooth 5, 4.2)	< 1Mbps ≈ (n x 100kbps)	≈ 100m
GSM/GPRS/EDGE (2G), UMTS/HSPA (3G), LTE (4G)	Typical download: 35-170kps (GPRS), 120-384kbps (EDGE), 384Kbps-2Mbps (UMTS), 600kbps-10Mbps (HSPA), 3-10Mbps (LTE)	≈ 35km max for GSM e ≈ 200km max for HSPA.
IEEE 802.15.4	≈ 20 kbps e 40 kbps (BPSK), ≈ 250 kbps (O-QPSK with DSSS)	≈ 100m
ISA100.11a	≈ 250 kbps	≈ 100m
6LoWPAN	≈ 250 kbps	≈ 100m
LoRaWAN	≈ 0.3-50 kbps	≈ 15km
EC-GSM-IoT	70 kbps (GSMK), 240 kbps (8PSK)	≈ 15km
LTE-MTC Cat 0	≈ 1 Mbps	≈10 km.
LTE-M (Cat M1, Cat M2) - eMTC	LTE-M Cat M1 ≈ 1 Mbps LTE-M Cat M2 ≈ 4 Mbps DL / ≈ 7 Mbps UL	≈ 10 km.
NB-IoT - Narrowband-IoT (LTE Cat NB1 e LTE Cat NB2)	LTE Cat NB1 ≈ 66 kbps (multi-tone) e ≈ 16.9 Kbit/s (single-tone) LTE Cat NB2 ≈ 159kbps	Melhor que acobertura LTE-M.
Neul	Até 100kbps	≈ 10km
NFC	106kbps, 212kbps, 424kbps	≈ 10cm
RPMA (Random Phase Multiple Access)	100kbs	≈ 70km

Tecnologia Sem Fio	**Taxa de Dados**	**Distância**
IEEE 802.11a/b/g/n/ac	Velocidades teóricas é de 11 Mbps (IEEE 802.11b), 54 Mbps (IEEE 802.11a e IEEE 802.11g), 100 Mbps (IEEE 802.11n) ou 300 Mbps (IEEE 802.11ac).	≈ 60m
Wi-Fi HaLow	347Mbps	≈900m
IEEE 802.16 (WiMax)	40Mbit/s móvel, 1 Gbit/s fixo	≈50Km
HART	250 kbps	≈200Km
Z-Wave	40kbps(915MHz) and 20kbps(868MHz)	30-100m
ZigBee	250 kbps (2.4GHz) 40kbps (915MHz) 20kbps (868MHz)	30-100m
Thread	250kbps	≈30m
DigiMesh	250 kbps (2.4) 40kbps (915) 20kbps (868)	Data 72
MiWi	20kbps	≈100m
EnOcean	125kbps	≈30m (outdoor 300m)
Weightless (W, N, P)	≈600bps-100kbps	≈2km (P), 5km (W, N)
mcThings	50kbps	≈200m
LoRa	50kbps	≈30km
SIGFOX	600bps	≈40km
DECT ULE	1Mbps	≈300m
Insteon	38400bps - via RF 13165bps - via powerlines	≈50m
RFID	4kbps - 640kbps	0.01m-10m
WAVIoT (NB-Fi - Narrowband Fidelity)	10-100bps	≈50km
DASH7 Alliance Protocol (D7A or D7AP)	167kbps	≈2km
Wi-SUN	300kbps	≈1000m
Wavenis	9.6kbps (433 e 868MHz) / 19.2kbps (915MHz)	≈1000m
MiOTY	407 bps	≈ 15km

Referências

1 DONNO, M. D.; TANGE, K.; DRAGONI, N. Foundations and evolution of modern computing paradigms: cloud, iot, edge, and fog. *IEEE Access*, v. 7, p. 150936–150948, 2019. 17

2 LASSO, F. F. J. et al. A survey on software-defined wireless sensor networks: current status, machine learning approaches and major challenges. 2021. 17

3 ASIF, R.; GHANEM, K.; IRVINE, J. Proof-of-puf enabled blockchain: concurrent data and device security for internet-of-energy. *Sensors*, v. 21, p. 28, 2020. 17

4 CHOI, G. H.; NA, T. Analysis of state-of-the-art spin-transfer-torque nonvolatile flip-flops considering restore yield in the near/sub-threshold voltage region. *Electronics*, v. 9, p. 2118, 2020. 17

5 ABUTAHA, M. et al. Secure lightweight cryptosystem for iot and pervasive computing. *Scientific Reports*, v. 12, 2022. 17

6 SHIRVANIMOGHADDAM, M. et al. Towards a green and self-powered internet of things using piezoelectric energy harvesting. *Ieee Access*, v. 7, p. 94533–94556, 2019. 17

7 KIM, W. et al. An indoor light-powered sensor system integrated with organic photovoltaics. *Advanced Materials Technologies*, v. 8, 2023. 17

8 SALMAN, T.; JAIN, R. A survey of protocols and standards for internet of things. *arXiv preprint arXiv:1903.11549*, 2019. 21, 37, 73, 74

9 FOROUZAN, B.; FEGAN, S. *Protocolo TCP/IP - 3.ed.* McGraw Hill Brasil, 2009. ISBN 9788563308689. Disponível em: <https://books.google.com.br/books?id=fNvIgp3kkyQC>. 21, 23

10 KRISHNA, M. B. V. *Economics of Computer Network Overheads*. 2018. Acesso em: 13/08/2023. Disponível em: <https://www.linkedin.com/pulse/economics-computer-network-overheads-mohan-bvk/>. 24, 25, 26, 48

11 TANENBAUM, A. S.; WETHERALL, D. J. *Redes de Computadores*. [S.l.]: Artmed, 2011. 26

12 TANENBAUM, A. S. *Sistemas operacionais modernos*. [S.l.]: Pearson Educación, 2010. 26

13 ALMEIDA, R. et al. *Programação de Sistemas Embarcados*. [S.l.]: Elsevier, 2023. ISBN 9788595159105. 27

14 VALVANO, J. W. *Embedded systems: introduction to ARM® Cortex (TM)-M microcontrollers*. [S.l.]: Jonathan W. Valvano, 2014. 28

15 MAZIERO, C. A. Sistemas operacionais: conceitos e mecanismos. *Livro aberto*, 2014. 28

16 SECPLICITY. *Understanding the Layers of a Computer System*. 2018. <https://www.secplicity.org/2018/09/19/understanding-the-layers-of-a-computer-system/>. 29

17 DRAGOS, C.; CHIRITA, A.; NELUș-CONSTANTIN, B. Understanding the differences between arm and x86 isa's. *ResearchGate*, Nov 2021. Disponível em: <https://www.researchgate.net/figure/ISA-as-an-interface-between-hardware-and-software_fig3_361792962>. 29

18 SANTANA, M. Chapter 6 - eliminating the security weakness of linux and unix operating systems. In: VACCA, J. R. (Ed.). *Network and System Security (Second Edition)*. Second edition. Boston: Syngress, 2014. p. 155–178. ISBN 978-0-12-416689-9. Disponível em: <https://www.sciencedirect.com/science/article/pii/B978012416689900006X>. 29

19 BARRY, R. *FreeRTOS-a free RTOS for small embedded real time systems*. 2006. 31

20 GUAN, F. et al. Open source freertos as a case study in real-time operating system evolution. *Journal of Systems and Software*, Elsevier, v. 118, p. 19–35, 2016. 31

21 INAM, R. et al. Support for hierarchical scheduling in freertos. In: IEEE. *ETFA2011*. [S.l.], 2011. p. 1–10. 31

22 ASHTON, K. et al. That 'internet of things' thing. *RFID journal*, Hauppauge, New York, v. 22, n. 7, p. 97–114, 2009. 31

23 SALIH, K. O. M. et al. A comprehensive survey on the internet of things with the industrial marketplace. *Sensors*, MDPI, v. 22, n. 3, p. 730, 2022. 32

24 AL-FUQAHA, A. et al. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE communications surveys & tutorials*, IEEE, v. 17, n. 4, p. 2347–2376, 2015. 32

25 EDQUIST, H.; GOODRIDGE, P.; HASKEL, J. The internet of things and economic growth in a panel of countries. *Economics of Innovation and New Technology*, Taylor & Francis, v. 30, n. 3, p. 262–283, 2021. 33

26 BURHAN, M. et al. Iot elements, layered architectures and security issues: A comprehensive survey. *sensors*, MDPI, v. 18, n. 9, p. 2796, 2018. 33, 73

27 DATTA, P.; SHARMA, B. A survey on iot architectures, protocols, security and smart city based applications. In: IEEE. *2017 8th international conference on computing, communication and networking technologies (ICCCNT)*. [S.l.], 2017. p. 1–5. 34

28 WU, M. et al. Research on the architecture of internet of things. In: IEEE. *2010 3rd international conference on advanced computer theory and engineering (ICACTE)*. [S.l.], 2010. v. 5, p. V5–484. 34, 35

29 MASHAL, I. et al. Choices for interaction with things on internet and underlying issues. *Ad Hoc Networks*, Elsevier, v. 28, p. 68–90, 2015. 34, 35

30 SAID, O.; MASUD, M. Towards internet of things: Survey and future vision. *International Journal of Computer Networks*, v. 5, n. 1, p. 1–17, 2013. 34, 35

31 DARWISH, D. Improved layered architecture for internet of things. *Int. J. Comput. Acad. Res.(IJCAR)*, Citeseer, v. 4, n. 4, p. 214–223, 2015. 34

32 KHAN, R. et al. Future internet: the internet of things architecture, possible applications and key challenges. In: IEEE. *2012 10th international conference on frontiers of information technology*. [S.l.], 2012. p. 257–260. 34, 39

33 TALAL, H.; ZAGROUBA, R. Mads based on dl techniques on the internet of things (iot): Survey. *Electronics*, MDPI, v. 10, n. 21, p. 2598, 2021. 34, 35, 70

34 CHEN, Y.; KUNZ, T. Performance evaluation of iot protocols under a constrained wireless access network. In: IEEE. *2016 International Conference on Selected Topics in Mobile & Wireless Networking (MoWNeT)*. [S.l.], 2016. p. 1–7. 36

35 DEEPSEADEV. *Most Used IoT Protocols*. 2023. Disponível em: <https://www.deepseadev.com/en/blog/most-used-iot-protocols/>. 36

36 MANSOUR, M. et al. Internet of things: A comprehensive overview on protocols, architectures, technologies, simulation tools, and future directions. *Energies*, MDPI, v. 16, n. 8, p. 3465, 2023. 37, 71, 73, 74

37 KUNTKE, F. et al. Lorawan security issues and mitigation options by the example of agricultural iot scenarios. *Transactions on Emerging Telecommunications Technologies*, Wiley Online Library, v. 33, n. 5, p. e4452, 2022. 37

38 ADELANTADO, F. et al. Understanding the limits of lorawan. *IEEE Communications Magazine*, v. 55, n. 9, p. 34–40, 2017. 38

39 LEONARDI, L. et al. Industrial lora: A novel medium access strategy for lora in industry 4.0 applications. In: IEEE. *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*. [S.l.], 2018. p. 4141–4146. 38

40 MENDES, A. S. et al. Multi-agent approach using lorawan devices: An airport case study. *Electronics*, v. 9, n. 9, 2020. ISSN 2079-9292. Disponível em: <https://www.mdpi.com/2079-9292/9/9/1430>. 39

41 GUPTA, B. B.; QUAMARA, M. An overview of internet of things (iot): Architectural aspects, challenges, and protocols. *Concurrency and Computation: Practice and Experience*, Wiley Online Library, v. 32, n. 21, p. e4946, 2020. 39, 40

42 HAHM, O. et al. Low-power internet of things with ndn & cooperative caching. In: *Proceedings of the 4th ACM Conference on Information-Centric Networking*. [S.l.: s.n.], 2017. p. 98–108. 40

43 CHEN, L.; FAN, G.; LIU, Y. Analyzing schedulability of energy-oriented distributed real-time embedded software. *Journal of Software*, v. 6, n. 9, p. 1771–1778, 2011. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-80051723818&doi=10.4304%2Fjsw.6.9.1771-1778&partnerID=40&md5=f5f6bf5caef40b81b4db02d2b19c5e79>. 41

44 CERCHECCI, M. et al. A low power iot sensor node architecture for waste management within smart cities context. *Sensors*, MDPI, v. 18, n. 4, p. 1282, 2018. 41

45 SANTOS, C.; JIMÉNEZ, J. A.; ESPINOSA, F. Effect of event-based sensing on iot node power efficiency. case study: Air quality monitoring in smart cities. *IEEE Access*, IEEE, v. 7, p. 132577–132586, 2019. 41

46 AZAR, J. et al. Using adaptive sampling and dwt lifting scheme for efficient data reduction in wireless body sensor networks. In: IEEE. *2018 14th international conference on wireless and mobile computing, networking and communications (WiMob)*. [S.l.], 2018. p. 1–8. 42

47 SAID, O.; AL-MAKHADMEH, Z.; TOLBA, A. Ems: An energy management scheme for green iot environments. *IEEE access*, IEEE, v. 8, p. 44983–44998, 2020. 42

48 ZHOU, G. et al. Adaptive transmission power control for wireless sensor networks. University of Virginia, Department of Computer Science, 2006. 42

49 ZEADALLY, S. et al. Design architectures for energy harvesting in the internet of things. *Renewable and Sustainable Energy Reviews*, Elsevier, v. 128, p. 109901, 2020. 42

50 MAIER, A.; SHARP, A.; VAGAPOV, Y. Comparative analysis and practical implementation of the esp32 microcontroller module for the internet of things. *2017 Internet Technologies and Applications (ITA)*, 2017. 44

51 HERCOG, D. et al. Design and implementation of esp32-based iot devices. *Sensors*, v. 23, n. 15, 2023. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/23/15/6739>. 44

52 SYSTEMS, E. *ESP32 - ESPRESSIF - Modules, Chips & Development Boards*. 2023. Disponível em: <https://www.espressif.com/en/products/modules/esp32>. 44

53 WIBOWO, F. et al. Design and implementation of iot-based smart laboratory using esp32 and thingsboard to improve security and safety in polnep informatics engineering laboratory. *Jurnal ELIT*, v. 3, p. 13–21, 2022. 44

54 TURRAHMAN, F.; FAIZAH, N.; KORYANTO, L. Tourism search application within the bima city locale of west nusa tenggara utilizing the android-based location-based benefit strategy utilizing visual studio code. *Journal Mobile Technologies (JMS)*, v. 1, p. 18–26, 2023. 45

55 Espressif Systems. *ESP-IDF Programming Guide*. [S.l.], 2024. Available online: <https://docs.espressif.com/projects/esp-idf/en/stable/esp32/index.html> (Accessed on [20/02/2024]). Disponível em: <https://docs.espressif.com/projects/esp-idf/en/stable/esp32/index.html>. 45

56 CARTER, L. M. et al. Paradim: a phits-based monte carlo tool for internal dosimetry with tetrahedral mesh computational phantoms. *Journal of Nuclear Medicine*, v. 60, p. 1802–1811, 2019. 45

57 LI, H.; SUN, X. Design and implementation of internet of things technology in college students' innovation and entrepreneurship integration system. *Computational Intelligence and Neuroscience*, v. 2022, p. 1–10, 2022. 45

58 TP-Link. *TP-Link EC220-G5 Specifications*. [S.l.], 2024. Available: <https://www.tp-link.com/br/service-provider/wifi-router/ec220-g5>. Disponível em: <https://www.tp-link.com/br/service-provider/wifi-router/ec220-g5>. 46

59 The Things Network. *LoRaWAN Regional Parameters*. 2023. Disponível em: <https://www.thethingsnetwork.org/docs/lorawan/regional-parameters/>. 49

60 SANTOS, D. A. de S. *Desenvolvimento e Aplicação de uma Metodologia para Avaliação e Melhoria de Maturidade em Gerenciamento de Projetos*. Dissertação (Dissertação de Mestrado) — Universidade Federal de Itajubá, Itajubá, 2017. Disponível em: <https://repositorio.unifei.edu.br/xmlui/bitstream/handle/123456789/699/dissertacao_santos_2017.pdf?sequence=1&isAllowed=y>. 50

61 Espressif Systems. *ESP32 Wi-Fi Throughput*. Acessado em 2023. Acessado em 31 de maio de 2023. Disponível em: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/wifi.html#esp32-wi-fi-throughput>. 57

62 NAIK, G. P.; BAPAT, A. U. A brief comparative analysis on application layer protocols of internet of things: Mqtt, coap, amqp and http. *Int. J. Comput. Sci. Mob. Comput*, v. 9, n. 9, p. 135–141, 2020. 70

63 TIGHTIZ, L.; YANG, H. A comprehensive review on iot protocols' features in smart grid communication. *Energies*, MDPI, v. 13, n. 11, p. 2762, 2020. 70

64 KARAGIANNIS, V. et al. A survey on application layer protocols for the internet of things. *Transaction on IoT and Cloud computing*, v. 3, n. 1, p. 11–17, 2015. 72

65 NEBBIONE, G.; CALZAROSSA, M. C. Security of iot application layer protocols: Challenges and findings. *Future Internet*, MDPI, v. 12, n. 3, p. 55, 2020. 73

66 NAGRARE, T.; SINDHWAD, P.; KAZI, F. Ble protocol in iot devices and smart wearable devices: Security and privacy threats. *arXiv preprint arXiv:2301.03852*, 2023. Disponível em: <https://ar5iv.org/pdf/2301.03852>. 73

67 SETHI, P.; SARANGI, S. R. Internet of things: Architectures, protocols, and applications. *Journal of Electrical and Computer Engineering*, Hindawi, v. 2017, p. 9324035, Jan 2017. ISSN 2090-0147. Disponível em: <https://doi.org/10.1155/2017/9324035>. 73, 74