UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Solução integradora de monitoramento por imagem com processamento de borda.

Jean Wellington de Souza

Itajubá, 5 de outubro de 2024

UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Jean Wellington de Souza

Solução integradora de monitoramento por imagem com processamento de borda.

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica como parte dos requisitos para obtenção do Título de Mestre em Ciências em Engenharia Elétrica.

Área de Concentração: Microeletrônica

Orientador: Prof. Dr. Danilo Henrique Spadoti
Coorientador: Prof. Dr. Reinaldo Lima de Abreu

5 de outubro de 2024
Itajubá

UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Solução integradora de monitoramento por imagem com processamento de borda.

Jean Wellington de Souza

Dissertação aprovada por banca examinadora em 22 de Julho de 2024, conferindo ao autor o título de **Mestre em Ciências em Engenharia Elétrica.**

Banca Examinadora:
Prof. Dr. Décio Rennó de Mendonça Faria
Prof. Dr. Victoria Dala Pegorara Souto

Itajubá
2024

Jean Wellington de Souza

Solução integradora de monitoramento por imagem com processamento de borda/ Jean Wellington de Souza. – Itajubá, 5 de outubro de 2024-

107 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Danilo Henrique Spadoti

Dissertação (Mestrado)
Universidade Federal de Itajubá - UNIFEI
Programa de pós-graduação em engenharia elétrica, 5 de outubro de 2024.

1. Processamento de borda. 2. Reconhecimento de Imagens 2. NBIoT. 3. LoRa. 5. Consumo de Energia I. Spadoti, Danilo Henrique. II. Universidade Federal de Itajubá. III. Solução integradora de monitoramento por imagem com processamento de borda

CDU 07:181:009.3

Jean Wellington de Souza

Solução integradora de monitoramento por imagem com processamento de borda

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica como parte dos requisitos para obtenção do Título de Mestre em Ciências em Engenharia Elétrica.

Trabalho aprovado. Itajubá, 22 de Julho de 2024:

Prof. Dr. Danilo Henrique Spadoti
Orientador

Prof. Dr. Reinaldo Lima de Abreu
Coorientador

Prof. Dr. Décio Rennó de Mendonça Faria

Prof. Dr. Victoria Dala Pegorara Souto

Itajubá
5 de outubro de 2024

Agradecimentos

Gostaria de expressar minha gratidão a todas as pessoas que, direta ou indiretamente, contribuíram para a realização deste trabalho.

Primeiramente, agradeço aos meus pais, que sempre apoiaram minhas decisões e me proporcionaram um ambiente de honestidade e integridade. Sou grato por seu constante apoio e incentivo, que foram fundamentais para meu crescimento pessoal e profissional. Agradeço também ao meu irmão Vinicio e à minha irmã Rafaella, que sempre torceram pelo meu sucesso e celebram este momento comigo.

Aos amigos que estiveram ao meu lado desde o início desta jornada: Fabiano Marcos, Willen Simões, Richard Souza, Ítalo Renato, Raíssa, e o pessoal da RepUteiro: Willian, Edson, Diego "Didi"e Bruninho. Um agradecimento especial ao grande amigo Marciano pelas inúmeras ajudas e incentivos. A vocês, que foram meu lar durante um período muito especial da minha vida, expresso minha profunda gratidão pela amizade e pelo apoio incondicional.

Agradeço aos meus orientadores, Prof. Dr. Danilo Spadoti e ao Dr. Reinaldo Abreu, cujas orientações foram fundamentais para o desenvolvimento deste trabalho e para o meu crescimento acadêmico e profissional. Ao pessoal da IE Tecnologia e LabTel, em especial Caio, Vitória, Anderson "Bolha", Lucas e Fábio, pelas colaborações e suporte essenciais para a realização deste trabalho.

Por fim, agradeço a todos que, direta ou indiretamente, contribuíram para a realização deste trabalho. Cada palavra de incentivo e gesto de apoio fizeram a diferença e me ajudaram a chegar até aqui. A todos vocês, meu sincero e profundo agradecimento.

"Se desejamos um propósito cósmico, então é preciso encontrar para nós mesmos um objetivo digno"

(Carl Sagan)

Resumo

Este trabalho aplica um método de inteligência artificial em um dispositivo de transmissão de dados visando realizar o processamento de borda no reconhecimento dos dígitos de hidrômetros. A implementação prática foi realizada em dois diferentes transmissores de baixo consumo de energia, NBIoT e LoRa. Foram analisadas a eficiência de transmissão, a taxa de erros e o consumo de energia dos dispositivos, avaliando, assim, a viabilidade de soluções de baixo consumo em aplicações de processamento de borda.

Foram realizadas 387 medições de reconhecimento de imagem e processamento de borda com transmissão via NBIoT, e 241 medições com LoRa. Os resultados do processamento de borda, obtidos a partir do treinamento de uma rede neural desenvolvida neste trabalho, apresentaram um erro médio de 0,289 e uma acurácia de 93,52%, indicando uma alta confiabilidade na identificação dos valores numéricos do hidrômetro.

A transmissão de dados via NBIoT mostrou baixas perdas de pacotes (4,65%), enquanto a transmissão com LoRa não apresentou nenhuma perda, demonstrando alta precisão nos pacotes de transmissão. A aplicação de processamento de borda com bateria e transmissores de baixo consumo mostrou-se promissora para uso em hidrômetros com medições diárias, estimando uma autonomia de 623 dias para aplicações com LoRa e 631 dias para o NBIoT. Assim, comprova-se a viabilidade de soluções de baixo consumo em aplicações de processamento de borda.

Palavras-chaves: Processamento de borda, reconhecimento de imagens, NBIoT, LoRa, consumo de energia

Abstract

This work investigates an artificial intelligence-based method applied to edge processing for the recognition of water meter digits. The implementation includes two low-power transmitters, NBIoT and LoRa, for transmitting the recognized numbers, and estimates the device's energy consumption to evaluate the feasibility of low-power solutions in edge processing applications.

A total of 387 image recognition and edge processing measurements were conducted with transmission via NBIoT, and 241 measurements with LoRa. The edge processing results, obtained from the training of a neural network developed in this work, showed an average error of 0.289 and an accuracy of 93.52%, indicating high reliability in identifying the water meter's numerical values.

Data transmission via NBIoT showed low packet loss (4.65%), while transmission with LoRa showed no loss, demonstrating high precision in data transmission. The application of edge processing with battery and low-power transmitters proved promising for use in water meters with daily measurements, estimating an autonomy of 623 days for LoRa applications and 631 days for NBIoT. Thus, the feasibility of low-power solutions in edge processing applications is confirmed.

Key-words: Edge processing, image recognition, NBIoT, LoRa, energy consumption

Lista de ilustrações

Figura 1 – Características de um dispositivo de IoT, adaptado de [1] 18
Figura 2 – *Dashboard* de monitoramento de energia, figura do autor 27
Figura 3 – Representação das conexões sinápticas que compõem a saída de um neurônio artificial, adaptado de [33] 29
Figura 4 – Representação das conexões das camadas de nerônios artificiais em um modelo de MLP conectado, adaptado de [33] 30
Figura 5 – Representação de uma imagem com quatro *pixels* de altura por quatro *pixels* de largura e três canais RGB, adaptado de [35] 34
Figura 6 – Representação de uma CNN, adaptado de [35] 41
Figura 7 – Relação da largura de banda com o alcance nos protocolos utilizados em IoT, adaptado de [29] 42
Figura 8 – Sinal chirp, adaptado de [36] 44
Figura 9 – Tempo de transmissão por pacote de símbolo, adaptado de [36] 45
Figura 10 – Tipos de implementação do NBIoT no domínio da frequência, adaptado de [37] 47
Figura 11 – Formas de onda da tensão e corrente de alimentação em cada uma das etapas de funcionamento do ciclo de operação do dispositivo, adaptado de [38] 49
Figura 12 – Dispositivo principal de captura de imagens, figura do autor. 51
Figura 13 – Modulo transmissor LoRa E32-915T20D, figura do autor. 52
Figura 14 – LoRa E32-915T20D em transmissão fixa, adaptado de [39]. 53
Figura 15 – LoRa E32-915T20D em transmissão *broadcasting*, adaptado de [39]. . . 53
Figura 16 – Arquitetura de uma rede NBIoT, adaptado de [37]. 54
Figura 17 – MODEM NBIoT, figura do autor. 55
Figura 18 – Circuito MODEM NBIoT montado na bancada de testes, figura do autor. 56
Figura 19 – Circuito LoRa, montado na bancada de testes, figura do autor. 56
Figura 20 – Receptor LoRa, figura do autor. 57
Figura 21 – Diagrama da aplicação de recebimento dos dados, figura do autor. . . . 58
Figura 22 – Diagrama de funcionamento do *firmware*, figura do autor. 59
Figura 23 – Historiograma da base de dados, figura do autor. 60
Figura 24 – Amostras de valores utilizados no treinamento da Rede Neural, figura do autor. 61
Figura 25 – Relação de Erros por Épocas obtido com treinamento de 300 épocas, figura do autor. 62
Figura 26 – Relação de Acurácia por Épocas obtido com treinamento de 300 épocas, figura do autor. 63

Figura 27 – Diagrama das tarefas de inteligencia artificial para o reconhecimento de imagem, figura do autor. 64
Figura 28 – Estrutura de organização dos diretórios do projeto principal, figura do autor. 64
Figura 29 – Página principal do servidor WEB, figura do autor. 66
Figura 30 – Página de configuração da imagem de referência, figura do autor. . . . 67
Figura 31 – Página de configuração do alinhamento na imagem de referência com as regiões de interesse selecionadas para serem reconhecidas, figura do autor. 68
Figura 32 – Diagrama das tarefas para transmissão via Modem NBIoT, figura do autor. 69
Figura 33 – Diagrama das tarefas para transmissão via LoRa, figura do autor. . . . 70
Figura 34 – Bancada de testes, figura do autor. 71
Figura 35 – *Case* para acoplar o dispositivo de processamento de borda ao hidrometro, figura do autor. 71
Figura 36 – Distância da transmissão de dados via NBIoT, figura do autor 73
Figura 37 – Consumo de água do hidrômetro, figura do autor 74
Figura 38 – Erro médio absoluto na medição, figura do autor 75
Figura 39 – Valores transitório das amostras coletadas para o treinamento da rede neural, figura do autor 76
Figura 40 – Montagem dos equipamentos para realização do ensaio para medição de energia do módulo de transmissão NBIoT, figura do autor 77
Figura 41 – Gráfico da curva de corrente em função do tempo da transmissão por NBIoT, figura do autor 78
Figura 42 – Distância dos dados transmitidos com tecnologia LoRa, figura do autor 79
Figura 43 – Consumo do Hidrometro durante a transmissão via LoRa, figura do autor 80
Figura 44 – Erro médio absoluto na medição LoRa, figura do autor. 80
Figura 45 – Diagrama do circuito de coleta dos dados de consumo LoRa, figura do autor 81
Figura 46 – Gráfico da curva de corrente em função do tempo da transmissão por LoRa, figura do autor 82
Figura 47 – Diagrama do circuito de coleta dos dados do ESP32-CAM, figura do autor 83
Figura 48 – Gráfico da curva de corrente em função do tempo do ESP32-CAM, figura do autor 84
Figura 49 – Comandos AT realizados para conectar o transmissor Telit 92

Lista de tabelas

Tabela 1 – Comparativo dos trabalhos relacionados com esta pesquisa 24
Tabela 2 – Configuração de transmissões NBIoT. 49
Tabela 3 – Consumo do Transmissor NBIoT 79
Tabela 4 – Consumo do Transmissor LoRa 83
Tabela 5 – Consumo de diferentes etapas de um ESP32-CAM 84
Tabela 6 – Comparativo entre as tecnologias NBIoT e LoRa 85
Tabela 7 – Consumo do ESP32 + NBIoT 87
Tabela 7 – Consumo do ESP32 + LoRa 88

Lista de abreviaturas e siglas

3G	*Third Generation*	17
3GPP	*3rd Generation Partnership Project*	46
AMR	*Automatic Meter Reading*	22
ANATEL	Agência Nacional de Telecomunicações	85
ANN	*Artificial Neural Network*	28
AT	*Attention Commands*	67
BR	Banda Reduzida	47
chirp	*Compressed High Intensity Radar Pulse*	44
CNN	*Convolutional Neural Network*	34
CPU	*Central Processing Unit*	21
CR	*Coding Rate*	46
CRC	*Cyclic Redundancy Check*	45
CSS	*chirp spread spectrum*	44
FDD	*Frequency Division Duplex*	47
FEC	*Forward Error Correction*	69
GM	*Geometric Moments*	22
GPRS	*General Packet Radio Service*	22
GSM	*Global System for Mobile Communications*	47
HTTP	*Hypertext Transfer Protocol*	68
I2C	*Inter Integrated Circuit*	54
IA	*Inteligência Artificial*	19
IO	*Input Output*	21
IoT	*Internet of Things*	17
ISM	*Industrial, Scientific, and Medical*	85
JSON	*JavaScript Object Notation*	21
LoRa	*Long Range*	20
LoRaWAN	*Long Range Wide Area Network*	49
LPWA	*Low Power Wide Area*	54
LTE	*Long-Term Evolution*	43
MAC	*Medium Access Control*	47
MAE	*Forward Error Correction*	74
ML	*Machine Learn*	19
MLP	*Multiple Layers Perceptron*	30
MOCNN	*Multi-Objective Convolutional Neural Network*	23
MQTT	*Message Queuing Telemetry Transport*	21

NBIoT	*Narrowband Internet of Things*	20
NFC	*Near Field Communication*	41
OCR	*Optical Character Recognition*	22
PCB	*Printed Circuit Board*	90
PROCEL	Programa Nacional de Conservação de Energia Elétrica	28
PSRAM	*Pseudo Static Random Access Memory*	51
PTM	*Projection Template Matchine*	22
RAM	*Random Access Memory*	89
ReLU	*Rectified Linear Unit*	29
RFID	*Radio Frequency Identification*	41
RGB	*Red, Green, Blue*	34
RLC	*Radio Link Control*	47
RNA	Rede Neural Artificial	28
RNN	*Recurrent Neural Networks*	31
ROI	*Region of interest*	63
RRC	*Radio Resource Control*	47
RSSI	Received Signal Strength Indicator	86
RTOS	*Real-Time Operating System*	89
SCFDMA	*Single Carrier Frequency Division Multiple Access*	48
SF	*Spreading Factor*	44
SPI	*Serial Peripheral Interface*	54
SSID	*Service Set Identifier*	58
TM	*Template matching*	22
TTL	*Transistor-Transistor Logic*	56
TV	*Television*	17
UART	*Universal Asynchronous Receiver-Transmitter*	52
UR	Unidade de Recurso	48
URL	*Uniform Resource Locator*	59
USB	*Universal Serial Bus*	54
WEB	*World Wide Web*	57
WECA	*Wireless Ethernet Compatibility Alliance*	42
WLAN	*Wireless Local Area Network*	43
WPAN	*Wireless Personal Area Network*	42

Sumário

1 INTRODUÇÃO 17
1.1 Motivação 19
1.2 Trabalhos relacionados 20
1.2.1 Síntese 24
1.3 Objetivos 24
1.4 Estrutura do Trabalho 25
1.5 Publicações 25

2 REVISÃO TEÓRICA 27
2.1 Monitoramento em Tempo Real 27
2.2 Inteligência Artificial 28
2.2.1 Redes Neurais 28
2.2.2 Camadas de Redes Neurais 30
2.2.3 Treinamento da Rede Neural 31
2.2.4 Redes Neurais Convolucionais ou CNN 34
2.2.5 Filtros ou *Kernels* 34
2.2.6 Convolução 35
2.2.7 Agrupamento ou *Pooling* 37
2.2.8 Preenchimento ou *Padding* 38
2.2.9 Construção da CNN 39
2.3 Conectividade 41
2.3.1 Os protocolos de transmissão de dados para IoT 41
2.3.2 LoRa 44
2.3.3 LTE e o NBIoT 46
2.4 Medição de Energia 49

3 PROJETO EXPERIMENTAL 51
3.1 Dispositivo de processamento de imagem 51
3.2 Dispositivos de transmissão de dados 52
3.2.1 Transmissor LoRa 52
3.2.2 Transmissor LPWA NBIoT 54
3.3 A montagem dos transmissores e receptores 55
3.4 Apresentação e configuração da aplicação de processamento de imagem 57
3.4.1 Rede Neural criada para o reconhecimento de imagens 60
3.4.2 Rede Neural integrada ao *firmware* 62

3.4.3 Configuração do reconhecimento de imagem via servidor WEB 63
3.5 Adaptações no *firmware* da aplicação principal 67
3.5.1 Inclusão de rotina de envio via NBIoT 67
3.5.2 Inclusão de rotina de envio via LoRa 69
3.6 Montagem do projeto final em uma bancada de teste 70

4 RESULTADOS E ANÁLISES 73
4.1 Análise dos resultados obtidos dos dados transmitidos via NBIoT . 73
4.2 Analise dos resultados obtidos dos dados transmitidos via LoRa . . 79
4.3 Comparativo entre NBIoT e LoRa 84
4.4 Dimensionamento do banco de baterias para aplicação 87

5 CONCLUSÕES E TRABALHOS FUTUROS 89

APÊNDICES 91

APÊNDICE A – SEQUENCIA DE COMANDOS AT PARA CONEXÃO NBIOT 92

ANEXOS 96

ANEXO A – ARTIGO PUBLICADO 97

REFERÊNCIAS 104

1 Introdução

Muitas vezes, algumas palavras técnicas, são substituídas por jargões, no intuito de simplificar expressões ou termos para um determinado processo ou dispositivo. “Borda” ou “processamento na borda” não é diferente disto, são termos muito discutidos em sistemas embarcados e internet das coisas IoT (do inglês *Internet of Things*). Antes de entrar neste conceito, é interessante entender o que cada um destes termos representa.

Uma definição de Sistemas Embarcados é apresentada em [1], que define como “computadores que controlam a eletrônica de todos os dispositivos físicos”. São inúmeros estes dispositivos presentes na vida cotidiana, desde fones *bluetooth* à sistemas de controle para motores de carros, aviões, navios, entre outros. “Sistemas Embarcados” são sistemas eletrônicos microprocessados que, após serem programados, possuem uma função específica que geralmente não pode ser alterada. Uma impressora, por exemplo, mesmo possuindo um processador que poderia ser utilizado para qualquer tipo de atividade, tem sua funcionalidade restrita apenas para a impressão de páginas. Um computador de propósito geral, no entanto, pode ser utilizado num instante como um ambiente de entretenimento e, em outro, como uma estação de trabalho, ou até mesmo como um telefone [1]. Não há dúvidas que, de uma maneira geral, são “pequenos computadores” programáveis de propósito geral para tarefas específicas, que podem executar tarefas simples, como um relógio digital ou, ainda, em aplicações sofisticadas, como um sistema operacional Linux embarcado para uma *smart TV (do inglês Television)*, *laptop's* ou *smartphones*.

A IoT propõe, de forma abrangente, a interconexão de objetos de todos os tipos possibilitando a proliferação de dispositivos, que estão se tornando cada vez menores e mais acessíveis, permitindo processamento, detecção, atuação e comunicação sem fio [2].

Mas o que significa “conectar um objeto à Internet”? Inserir um soquete *ethernet* em uma cadeira ou um modem 3G (do inglês *Third Generation*) em uma máquina de costura não transforma esses objetos de maneira significativa. Em vez disso, deve haver algum fluxo de informações que conecte as características definidoras do objeto com o mundo dos dados e do processamento representado pela internet. O objeto está presente fisicamente no mundo real, em uma casa, no trabalho, no carro ou no corpo de uma pessoa. Isso significa que ele pode receber informações do ambiente e transformá-las em dados que são enviados à Internet para coleta e processamento. Assim, uma cadeira pode coletar informações sobre a frequência e a duração de uso, enquanto uma máquina de costura relata a quantidade de linha restante e o número de pontos costurados. A presença do objeto também significa que ele pode produzir resultados no ambiente por meio de “atuadores” acionados por dados coletados e processados na internet. Portanto, a cadeira

pode vibrar para avisar sobre a chegada de um *e-mail* [3].

Mas o que torna um dispositivo IoT diferente de qualquer outra aplicação eletrônica que coleta dados da internet é o consumo de energia, o processamento de dados, o tamanho do dispositivo e o preço. A Figura 1 apresenta um gráfico comparativo destas características, ou seja, em uma escala de 0 a 10, o quão alto deve ser o consumo, o tamanho, assim como o preço e o processamento dos dispositivos IoT. Portanto verifica-se que um dispositivo de IoT deve conter baixo consumo de energia, ter dimensões reduzidas, conter um baixo preço e não necessariamente um alto poder de processamento, embora os dispositivos modernos, como os processadores da família STMicroelectronics [4], Espressif [5], ATMEGA [6], Nordic [7], entre outros, possuem múltiplos núcleos, frequência de *clock* específica, armazenamento e memória [8].

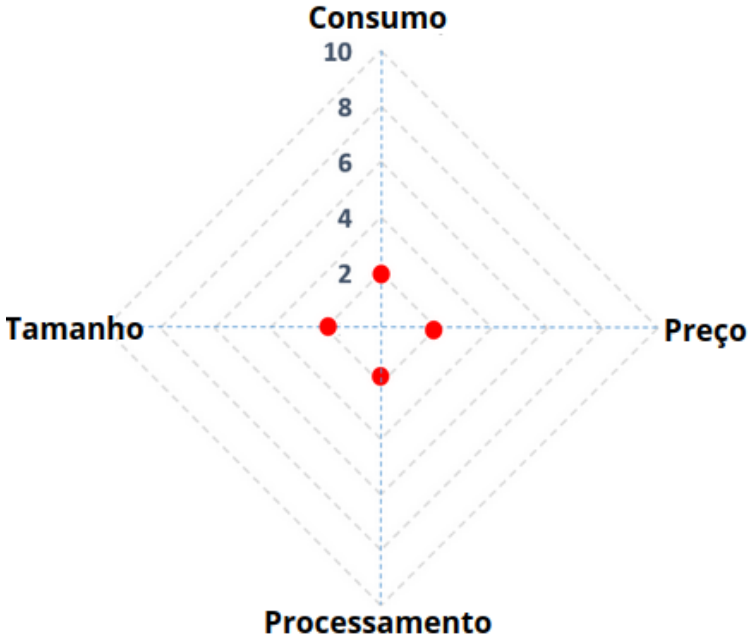

Figura 1 – Características de um dispositivo de IoT, adaptado de [1]

Existem alguns pontos em especial que chama atenção em IoT. Primeiro, com o aumento de dispositivos conectados, há uma grande quantidade de dados trafegando pela internet e sendo armazenados em *cloud's* e *datacenter's*. Estes dados quando enviados e armazenados criam um grande amontoado de dados denominado *data lake*, que não são usados [9]. O outro ponto se refere à energia que estes dispositivos tendem a consumir. Estima-se que até 2025 haverá 20,4 bilhões de dispositivos de IoT conectados à Internet, e isto leva a duas consequências: o aumento do processamento em nuvem e uma maior geração de energia [10].

O processamento em borda ou dispositivos em borda é uma solução que tem o intuito de aliviar este desafios, processar as informações nos dispositivos em tempo real e enviar o mínimo de dados, sem que haja a necessidade de acúmulo de dados para o

processamento [11].

Com o avanço da microeletrônica, os dispositivos embarcados de IoT se tornaram cada vez mais avançados em termos de desempenho computacional e economia de energia. Por outro lado, os problemas envolvendo o processamento de borda se tornaram cada vez mais complexos. Com isto, surge um novo conceito para solucionar problemas complexos na borda, problemas estes como, por exemplo, de reconhecimento, classificação e tomada de decisão, que podem ser solucionados com a ajuda de IA (*Inteligência Artificial*), ML (do inglês *Machine Learn*) e inteligência artificial de borda [12].

Portanto, ao considerar o que foi discutido anteriormente, têm-se três elementos principais: primeiro, o monitoramento de dispositivos embarcados utilizando o conceito de IoT; segundo, a necessidade de aplicações com maior poder computacional para resolver problemas complexos; e, por último, o uso de um ramo da IA, o ML, capaz de gerar algoritmos de aprendizagem para a tomada de decisões. Combinando esses elementos, obtém-se um cenário onde dispositivos eletrônicos de IoT auxiliam de maneira inteligente em diversas tarefas do dia a dia, coletando dados que são processados e oferecendo diagnósticos precisos através de algoritmos de IA.

Porém, com a grande imersão de dados, surgem muitos problemas: problemas de latência na transmissão de dados, acúmulo desnecessário de informação, gasto elevado de energia, necessidade de processadores extremamente potentes para realizar o processamento de grandes quantidades de dados, entre muitos outros [12].

A inteligência artificial de borda surge como solução para minimizar este problema. Os dispositivos de borda são dispositivos eletrônicos de IoT que coletam dados e utilizam modelos de IA reduzidos, treinados externamente para realizar inferência, ou seja, para fazer previsões ou tomar decisões com base nesses dados. Estes dispositivos podem não só exibir um resultado, mas também, podem atuar no ambiente ou em algum processo de maneira autônoma [12].

Dispositivos de borda são uma solução que vem sendo implementada gradualmente no cotidiano, e a grande maioria dos dispositivos de IoT são conectados à internet e realizam transmissão contínua de dados.

1.1 Motivação

Neste contexto de dispositivos inteligentes realizando medições, os medidores inteligentes são dispositivos responsáveis por obter medidas referentes ao consumo de energia, que pode não ser somente eletricidade, mas também, gás e água. Estes dispositivos devem ser capazes de realizar medições precisas de energia e possuir comunicação entre os vários nós de medição, monitorado a entrada e a saída do fluxo de energia [13]. De acordo com

a reportagem de "*Power and Renewables*" da revista *Wood Magazine*, até 2025 haverá um investimento de $30 bilhões de dólares para instalar mais de 300 milhões de *smart meters* [14]. Desta forma, a medição inteligente do consumo de energia é considerada essencial não somente para as companhias de fornecimento, mas também para os consumidores, que assim podem estar mais atentos e, consequentemente, ativos na melhoria de sua eficiência [15]. Um dos pontos importantes ressaltados pela filosofia dos *smart meters* é justamente a diminuição do desperdício de energia que os medidores convencionais apresentam. Segundo o Sistema de Gerenciamento de Recursos Hídricos do Estado de São Paulo (*SigRH*), o Brasil registrou 36,7% de perdas na distribuição de água potável, o que é suficiente para levar este recurso aos quase 35 milhões de brasileiros que ainda não possuem acesso a água para sequer lavarem as mãos [16].

Considerando o problema do desperdício de recursos hídricos, juntamente com a emersão do IoT integrado a IA e inteligência artificial de borda, a criação de dispositivos que utilizam essa nova tecnologia pode auxiliar na redução do desperdício, na tomada de decisões para evitar eventuais faltas, na redução de custos com a medição (já que no Brasil ainda é realizada com o deslocamento de funcionários), na melhoria dos hábitos energéticos, no aumento da economia de energia e na conscientização do usuário sobre o consumo e o desperdício. Além disso, permite a medição em tempo real do consumo e a análise relacionada devido aos algoritmos de eficiência [17].

1.2 Trabalhos relacionados

Trabalhos relacionados são estudos concluídos realizados pela comunidade científica que abordam os temas de processamento de borda, IA aplicada em microcontroladores, processamento de borda e envio de dados, LoRa (do inglês *Long Range*) integrado a dispositivos com processamento de borda, reconhecimento de números com microcontrolador e dispositivos integradores para a Indústria 4.0. Com isto, pretende-se identificar as técnicas utilizadas, a problemática envolvendo o processamento de borda com microcontroladores, bem como as técnicas e metodologias utilizadas buscando contribuir com o trabalho atual.

Todo o escopo da pesquisa é baseado em artigos de conferência e periódicos relevantes extraídos de bases científicas como *Google Scholar*, *IEEE eXplore*, *ScienceDirect* e *SciELO*. As pesquisas apresentadas foram realizadas entre janeiro de 2020 e dezembro de 2023 e teve como objetivo identificar trabalhos contendo palavras-chaves como "processamento de borda", "*TinyML*", "inteligência artificial de borda", "LoRa" e NBIoT (do inglês *Narrowband Internet of Things*). Os trabalhos relacionados passaram por 3 critérios de avaliação, revisão de título e resumo, remoção de duplicidade e análise do texto. Por fim, foram selecionados oito trabalhos que abordam o assunto.

A proposta em [18] baseia-se em um módulo de abstração de dispositivo de coleta de dados que reconhece diversos protocolos da fonte de dados, capturando os pacotes enviados e os disponibilizando de maneira padronizada para o processamento, garantindo a leitura de dados de várias fontes, formatos e protocolos numa arquitetura de *hardware* econômica em termos computacionais (memória, CPU (do inglês *Central Processing Unit*) e IO (do inglês *Input Output*)), além de utilizar técnicas de análise de dados na borda para a tomada de decisão. Segundo o autor, a problemática está ligada ao fato de que existem diversos protocolos de comunicação, onde muitos dos dispositivos atuais de borda enviam dados repetidos ou cíclicos. Portanto, o modelo de *framework STEAM* trata os dados de seis maneiras, extraindo o desvio padrão, média, incerteza, detecção de anomalias, reconhecimento de padrões e predição, enviando somente estes resultados por meio de um protocolo padrão MQTT (do inglês *Message Queuing Telemetry Transport*) padronizado num JSON (do inglês *JavaScript Object Notation*). Como resultados, o modelo de *framework* apresentou uma facilidade para o desenvolvimento de aplicações IoT autônomas, permitindo análise de dados em tempo real, enriquecimento de fluxo de dados em tomada de decisão na borda. O conjunto de funções da *STEAM* fornece um conjunto de classes prontas para funções integradas, tornando desnecessário o uso de código estruturado, lógica complexa, *loops* e instruções condicionais.

Em [19], o autor aborda a importância do uso do IoT nas *smart grids* e cita o grande aumento de dispositivos nos últimos anos, não só de medidores de energia para *smart grid*, mas dispositivos inteligentes em vários sistemas, como: militar, saúde, eletrodomésticos, transporte, vigilância, automação, entre outros. Além disso, aborda o principal problema envolvendo os milhares de dispositivos conectados, o envio de milhares de dados por segundo, o que causa indisponibilidade na largura de banda, pouca privacidade, baixa segurança e baixa resposta nos paradigmas convencionais de computação em nuvem. Para resolver este problema, o autor apresenta o paradigma de computação chamado "*edge*". Esse conceito se baseia em recursos de computação e armazenamento na borda da rede mais próxima dos dispositivos móveis e sensores, o que garante que os problemas como utilização excessiva de energia pela computação em nuvem, pressão na largura de banda da rede, atraso no tempo e altos custos na infraestrutura da rede possam ser resolvidos por este paradigma. E como solução, o autor apresenta uma arquitetura composta por camadas na geração, na distribuição e no consumo de energia, seguindo os conceitos de compatibilidade, sustentabilidade, confiabilidade, escalabilidade, gestão de recursos, flexibilidade, segurança e privacidade. Desta forma, o processamento deixa de ser centralizado, tomando uma resposta mais ágil e evitando os problemas citados anteriormente.

Quando se trata de processamento de imagens em microcontroladores de baixo custo, com o uso restrito de memória, processamento e bateria, existem poucos trabalhos relacionados como em [20]. Os autores apresentam uma solução para a detecção do uso de máscara em ambientes fechados utilizando um microcontrolador de baixo custo e baixo

consumo de energia com o uso de bateria. Para isto, utilizam o conceito de *TinyML*, uma integração da computação de borda entre dispositivos IoT e ML, que combinam o aspecto de baixo consumo de energia do IoT e algoritmos capazes de realizar tarefas complexas como a tomada de decisão, previsão e classificação. Com o uso da biblioteca *K-means*, os autores conseguiram criar o modelo de aprendizado e integrar em um microcontrolador STM32F411. Dito isso, o que se destaca no trabalho é a simplificação na métrica de distância, onde os autores retiram da métrica euclidiana quadrática (usada convencionalmente pelo algoritmo *K-means*) a parte quadrática, isto impactou diretamente no tempo de processamento e, consequentemente, da relação "processamento vs consumo". Como resultado, os autores alcançaram uma eficiência de 79% na detecção de rosto e 225 horas de operação com bateria de 2500mAh e 3,7V.

Um trabalho de AMR (do inglês *Automatic Meter Reading*) onde os autores projetam um algorítimo de OCR (do inglês *Optical Character Recognition*) para realizar a leitura de algarismos mecânicos de hidrômetros pode ser encontrado em [21]. OCR é uma técnica bastante conhecida para o reconhecimento de dígitos e palavras escritas à mão, utilizando visão computacional. Os métodos utilizados neste trabalho são TM (do inglês *Template matching*), que procura partes combinantes por meio da distância euclidiana; PTM (do inglês *Projection Template Matchine*), que compara a soma dos valores dos *pixels* das colunas verticais e horizontais; GM (do inglês *Geometric Moments*), também conhecido como momento cartesiano, que calcula a centroide das coordenadas, e ML, em que os testes foram realizados em paralelo com o *Matlab*. Como conclusão, os autores argumentam a dificuldade em fazer o reconhecimento nas transições dos números e recomendam o mínimo de intervenção possível, tendo em vista que utilizaram um *push button* para realizar a captura da imagem, causando ruído. Por outro lado, a precisão do modelo alcançou 98,1% e processamento de imagem de 0.00375 segundos por caractere.

Nas cidades inteligentes, o monitoramento de energia (eletricidade, água e gás) tem necessitado cada vez mais de dispositivos inteligentes de monitoramento e o envio de dados a longas distâncias com baixo consumo. No trabalho apresentado em [22] é desenvolvido um sistema remoto de leitura de medidores de água *multi-relay* com concentrador baseado em LoRa e GPRS (do inglês *General Packet Radio Service*). A estrutura apresentada é composta por 6 camadas, onde os medidores individuais se comunicam com relés; as residências e os concentradores via LoRa; o concentrador com a estação central (*Workstation*) e o *Data Center*, criando, assim, uma topologia de rede para o monitoramento de energia. O medidor de energia não utiliza reconhecimento de imagens ou adaptação de medidores analógicos, a leitura é realizada com módulo de *Reed switch* (contador de pulsos) já empregado nos medidores. A contribuição deste trabalho está na topologia apresentada e a conveniência em utilizar o *callback* do LoRa (*uplink*) para atuar desligando o fornecimento de água em situação de inadimplência.

O trabalho em [23] propõem um método de leitura automática com uma *Raspberry Pi 4B*. O método determina a posição do ângulo do ponteiro que se encontra no medidor de água capturado por câmera. Para reconhecer os ponteiros, os autores aplicam um filtro gaussiano de duas dimensões que transforma a imagem em escala de cinza e realça os ponteiros da foto. Em seguida, é aplicado um método de segmentação de imagem baseado na informação da posição para apagar as informações redundantes da área restando apenas os ponteiros. Após isto, é aplicada a transformada de *Hough* que transforma os círculos (contido nos ponteiros) em parâmetros de espaço no plano da imagem. Com o centro do círculo na imagem é possível desenhar um ponteiro em direção ao ponto mais extremo da imagem utilizando o método de *Bresenham*, o ponteiro é obtido calculando o ângulo em que a linha central girou no sentido anti-horário em comparação com a linha de escala de zero. Assim, é possível obter ângulos e mapear os valores em uma escala subdividida de 0 a 9. Como resultado, o autores discutem a precisão da medição, como se trata de subdivisões entre 0 e 9, uma precisão de 36º é suficiente para realizar a leitura, porém, o sistema apresenta a precisão do reconhecimento entre 2º e 4º.

O reconhecimento de números em sistema mecânicos girantes, como de hidrômetros, não é uma tarefa fácil. Embora eficientes para a classificação destes algorítimos, muitas arquiteturas de redes neurais não são adequadas para projetos de IoT devido a limitação de memória e processamento. Um trabalho que propõe uma rede neural com memória otimizada é apresentado em [24]. A metodologia aplicada pelos autores consiste em criar uma configuração de rede neural convolucional chamada de MOCNN (do inglês *Multi-Objective Convolutional Neural Network*), que reduz a quantidade de pesos e não afeta o desempenho do reconhecimento da imagem. Os resultados experimentais de MOCNN apresentam 100% de eficiência no reconhecimento de imagens em condições adequadas de luz.

Existe um sensor específico para medir energia por meio da variação de um líquido no sensor chamado de *gauge*. Este nível é alterado de acordo com o que se deseja medir (pressão, temperatura ou nível de água). Embora preciso, não se trata de um dispositivo que seja integrado à Internet, onde a verificação e o controle devem ser realizados de maneira visual. Neste contexto, o trabalho de [25] é realizar a integração das leituras do nível do equipamento por meio do processamento de imagem por correspondência de características e reconhecimento com rede neural *backpropagation* a fim de comparar a eficácia dos métodos. Ambos os métodos realizam a captura da imagem e aplicam o filtro (*grayscale* gaussiano) para transformar a imagem em escala de *pixels* (0-255). O método de correspondência divide o algoritmo em quatro segmentos e faz a comparação bit a bit com um banco de imagem. O intuito de segmentar a imagem em quatro partes é tornar o processamento e o reconhecimento mais rápido. Por outro lado, é utilizado uma rede neural densa que faz a classificação das imagens pelo método de regressão, sendo convertida em uma matriz de 240 x 1 entrada da rede neural, e a saída correspondente

ao valor daquela entrada. Como conclusão, o método de correspondência foi classificado como o melhor, com taxa de 100% de acertos, enquanto o método que utiliza redes neurais apresentou muitos problemas devido devido ao método de descida gradiente produzir muitos problemas de mínimos locais. Os autores também citam o tempo de resposta, o algoritmo de correspondência leva 1,2 segundos para realizar o reconhecimento, enquanto a rede neural leva 10,6 segundos.

1.2.1 Síntese

A síntese dos trabalhos relacionados aborda diversos aspectos que estão em consonância com a proposta deste estudo. A Tabela 1 apresenta um resumo conciso das principais contribuições de cada trabalho relacionado, fornecendo uma visão geral clara das conexões entre as pesquisas existentes e a abordagem deste estudo. A análise dessas contribuições ajuda a fundamentar teoricamente e contextualizar a abordagem proposta.

Autor	Dispositivo de Borda	Baixo Consumo	Otimiza Informações	Técnica de Transmissão
[18]	Sim	Não	Sim	WiFi
[19]	Sim	Sim	Não	WiFi
[20]	Sim	Sim	Sim	Não transmite
[21]	Não	Não	Sim	Não transmite
[22]	Sim	Sim	Sim	Não transmite
[23]	Sim	Não	Sim	Não transmite
[24]	Sim	Não	Sim	Não transmite
[25]	Sim	Não	Sim	Não transmite

Tabela 1 – Comparativo dos trabalhos relacionados com esta pesquisa

Conclui-se que as pesquisas na área de processamento de borda, focam na redução e otimização de informações, mas não se atentam à transmissão eficiente dos dados e à redução do consumo de energia. Portanto, este trabalho busca preencher essa lacuna ao propor uma abordagem integrada que não apenas otimiza o processamento de dados na borda, mas também considera a transmissão eficiente e o consumo de energia, contribuindo assim para um avanço significativo no campo do processamento de borda.

1.3 Objetivos

Este trabalho busca explorar a interseção entre o processamento de borda com IA, redes de IoT e a eficiência energética. Logo, o objetivo principal deste estudo é realizar a análise prática da implementação de soluções de processamento de borda em ambientes IoT, com o objetivo de otimizar o consumo em hidrômetros analógicos.

Serão considerados aspectos como a distribuição inteligente de tarefas, o uso de algoritmos eficientes e o impacto na vida útil dos dispositivos, que apresentam-se como objetivos específicos:

- Desenvolver uma aplicação de IoT baseada em IA e processamento de borda que faz o reconhecimento de números de um hidrômetro;
- Implementar dois dispositivo transmissores de dados de tecnologia de comunicação sem fio NBIoT e LoRa e apresentar uma conclusão do dispositivo que ofereça melhor eficiência;
- Realizar uma análise de eficiência energética.

1.4 Estrutura do Trabalho

Esta dissertação está estruturada em cinco capítulos. O Capítulo 1 introduz a temática de processamento de borda, sistemas embarcados e IoT, apresenta a motivação do tema e trabalhos relacionados. O Capítulo 2 explora a teoria de monitoramento em tempo real, IA, conectividade e interface de dados, estes temas são relevantes para o projeto, que é descrito no Capítulo 3. Os resultados originados da aplicação do modelo proposto são apresentados no Capítulo 4, e as conclusões são resumidas no Capítulo 5.

1.5 Publicações

Durante o desenvolvimento deste trabalho, resultados prévios foram publicados no WCNPS'23, no artigo de título *"Analysis of Energy Consumption in Smart Edge Processing Device"* e autores Jean W. Souza, William H. P. Costa, Reinaldo L. Abreu e Danilo H. Spadoti. O artigo foi publicado em congresso internacional do IEEE na cidade de Brasília-DF.

Outras publicações foram realizadas durante o período do mestrado, relacionadas a projetos em execução no Laboratório de Telecomunicações, listados abaixo em ordem cronológica:

- COSTA, Caio T. B.; SOUZA, Jean W; ABREU, Reinaldo L; SPADOTI, Danilo H. ANÁLISE DO RISCO DE TRANSMISSÃO DE DOENÇAS RESPIRATÓRIAS UTILIZANDO SENSORES DE CO2 CONECTADOS EM REDE LORAWAN. 15º Congresso Brasileiro de Eletromagnetismo. Anais... Florianópolis(SC) Online, 2022.
- COSTA, CAIO TÁCITO BORGES DA ; SPADOTI, DANILO ; SOUZA, JEAN WELLINGTON DE ; ABREU, REINALDO . LoRaWAN integrated CO2 sensing applied in COVID-19 transmission risk assessment. In: XL Simpósio Brasileiro de

Telecomunicações e Processamento de Sinais, 2022. Anais do XL Simpósio Brasileiro de Telecomunicações e Processamento de Sinais.

2 Revisão Teórica

Neste capítulo, é empregada uma abordagem abrangente dos conceitos teóricos que estabelecem as bases para uma compreensão deste trabalho. Dentre os tópicos abordados, destacam-se: o monitoramento em tempo real, que permite a coleta contínua e imediata dos dados que serão reconhecidos e transmitidos; a IA, que aborda o método de convolução para o reconhecimento de imagens; a conectividade, com ênfase nos protocolos LoRa e NBIoT; e a medição de energia, que demonstra o método usado para dimensionar baterias em um dispositivo por meio do cálculo do consumo.

2.1 Monitoramento em Tempo Real

Sistemas de monitoramento em tempo real são amplamente utilizados em indústrias e empresas, o monitoramento está geralmente associado ao controle de processos, produção, ao monitoramento de grandezas e ao consumo de energia. Empresas e indústrias costumam seguir cronogramas de tarifas de energia e políticas de penalização para o consumo excessivo de energia, um exemplo é a penalização para cargas com baixo fator de potência. Para reduzir custos e evitar penalidades, torna-se necessário o monitoramento em tempo real para implementar políticas de eficiência e economia de consumo [26].

A Figura 2 mostra um *Dashboard* para o monitoramento do consumo de energia. *Dashboard* são plataformas integradas a equipamentos de medição, utilizados para o monitoramento em tempo real de alguma grandeza.

Figura 2 – *Dashboard* de monitoramento de energia, figura do autor

Analisando a Figura 2, pode-se observar o monitoramento de energia da produção de uma empresa onde são monitorados o consumo mensal geral, a demanda, o consumo acumulado e o fator de potência. É possível observar que existe um fator limitador indicado

por uma linha amarela, o fator limitador indica o controle para a tomada de decisão, que pode ser desde a conscientização dos usuários de energia à cortes programados de energia.

Diferente do setor industrial, o setor residencial é composto por vários usuários unitários de energia, como casas, apartamentos, lojas e comércio. Diferente das indústrias, atualmente existem poucos meios de monitoramento de energia em tempo real para usuários residenciais ou consumidores cativos, estes são informados do seu consumo apenas no fechamento mensal das faturas de cobrança de energia, o que torna a tomada de decisão para a redução do consumo dificultada [27].

A redução do consumo de energia no setor residencial está associada integralmente à conscientização dos seus usuários. Medidas simples como apagar todas as luzes ao sair de casa e a compra de eletrodomésticos com o selo PROCEL (Programa Nacional de Conservação de Energia Elétrica), que indica menor consumo de energia, ajudam a reduzir o consumo de energia. Embora o presente estudo tenha focado na medição de água, o sistema desenvolvido pode ser utilizado também para aplicações de energia, seguindo o mesmo princípio. A escolha pela medição de água deve-se ao fato de que a maior parte dos hidrômetros instalados em residências ainda são analógicos.

2.2 Inteligência Artificial

A IA busca fazer com que um computador faça o que a mente humana é capaz de fazer como, por exemplo, o raciocínio de percepção, associação, previsão e planeamento, habilidades que permitem os seres humanos e animais a atingirem seus objetivos. A inteligência não é uma dimensão única, mas um espaço estruturado de diversas capacidades de informação [11]. Resumidamente, a IA possibilita com que uma máquina, por meio de algoritmos, possua a capacidade cognitiva – semelhante ao do ser humano, para realizar tarefas que antes só eram realizadas pelo homem [28].

2.2.1 Redes Neurais

Uma RNA (Rede Neural Artificial), ou ANN (do inglês *Artificial Neural Network*), é um modelo computacional inspirado no funcionamento do cérebro humano, desenvolvido com o propósito de replicar a maneira de como o cérebro executa uma atividade ou função específica [29]. Desde a sua origem e difusão, esse conceito tem sido utilizado para abordar desafios relacionados ao reconhecimento de fala e escrita, visão computacional, previsão de tendências de mercado, detecção de anomalias, entre outras aplicações [30].

A arquitetura básica de uma ANN, responsável por tratar as entradas do modelo e obter uma saída, é baseada na conexão entre neurônios artificiais, expressos por equações matemáticas que definem suas conexões sinápticas [31]. A saída y_j do neurônio arbitrário j relaciona-se com os valores de ativação $x_{i,i} \in [1, n]$ dos n dados de entrada ou neurônios

conectados à sua entrada a partir de pesos $w_{i,j}$ e do limiar de ativação b_j [32], conforme o diagrama da Figura 3.

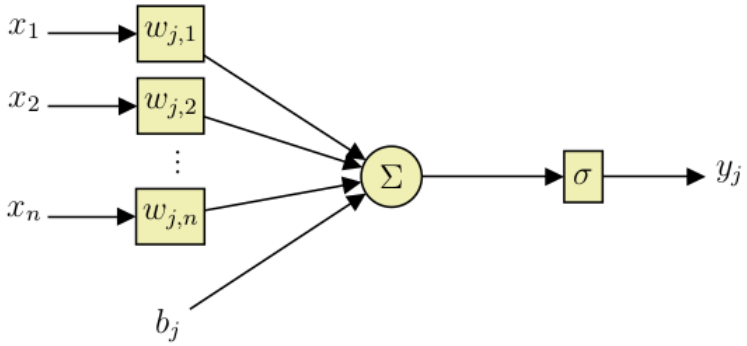

Figura 3 – Representação das conexões sinápticas que compõem a saída de um neurônio artificial, adaptado de [33]

A combinação de entradas do neurônio artificial é aplicada ao bloco σ, que representa a função de ativação que ajusta a saída gerada dentro do intervalo definido, seguindo o comportamento de um neurônio biológico. Entre as funções de ativação usadas na literatura, se destacam [31]:

1. **Sigmoide:** A função sigmoide, também chamada de função logística, mapeia uma entrada real para o intervalo (0, 1). Sua representação é dada pela seguinte equação:

$$\sigma_{sigm}(x) = \frac{1}{1 + exp(-x/\beta)} \tag{2.1}$$

em que $\beta \in (0, \infty)$ representa a elevação da função sigmoide em seu ponto de inflexão [30].

2. **Tangente hiperbólica:** Trata-se de uma variante escalonada da função sigmoide, cujo conjunto de valores está contido no intervalo (-1, 1). Sua expressão matemática pode ser apresentada da seguinte forma:

$$\sigma_{tanh}(x) = \frac{1 - exp(-\beta x)}{1 + exp(-\beta x)} \tag{2.2}$$

3. **ReLU:** A ReLU (do inglês *Rectified Linear Unit*) é uma função monótona que zera os valores negativos do domínio, sendo representada por:

$$\sigma_{relu}(x) = \begin{cases} 0, & \text{se } x < 0 \\ x, & \text{caso contrário.} \end{cases} \tag{2.3}$$

Neste trabalho, foi utilizado a função de ativação ReLU. Segundo [31], esta tem se mostrado altamente eficaz em permitir que as redes neurais aprendam e performem bem em uma variedade de tarefas, incluindo reconhecimento de imagem e processamento de linguagem natural.

2.2.2 Camadas de Redes Neurais

A estrutura dos neurônios em uma RNA pode ser subdividida em três partes, conhecidas como:

- **Camada de Entrada:** (*Input Layer*): camada encarregada de receber informações da entrada, que podem incluir dados, sinais e medições provenientes do ambiente externo. As entradas passam por um processo de normalização, considerando os limites das funções de ativação adotadas no modelo, buscando assim, obter uma precisão numérica aprimorada para as operações matemáticas executadas pela rede [31];
- **Camadas Ocultas:** (*Hidden Layers*): consistem em neurônios encarregados de identificar padrões associados ao processo ou sistema em análise. Essas camadas desempenham a maior parte do processamento em uma RNA [31];
- **Camada de Saída:** (*Output Layer*): também formada por neurônios, esta camada assume a responsabilidade de gerar e apresentar as saídas da rede. Ela combina os resultados do processamento das camadas anteriores [31].

Enquanto a estrutura das camadas de entrada e camadas de saída varia de acordo com a aplicação específica da RNA, o tamanho e as conexões das camadas ocultas dependem da arquitetura do modelo escolhido. Uma das arquiteturas mais comuns é a MLP (do inglês *Multiple Layers Perceptron*). A MLP é uma classe de RNA composta por pelo menos três camadas de nós: uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Cada nó, ou neurônio, em uma camada é completamente conectado a todos os neurônios da camada seguinte, permitindo que o modelo capture relações complexas não lineares nos dados. A título de ilustração, a Figura 4 mostra um exemplo de uma MLP completamente conectada.

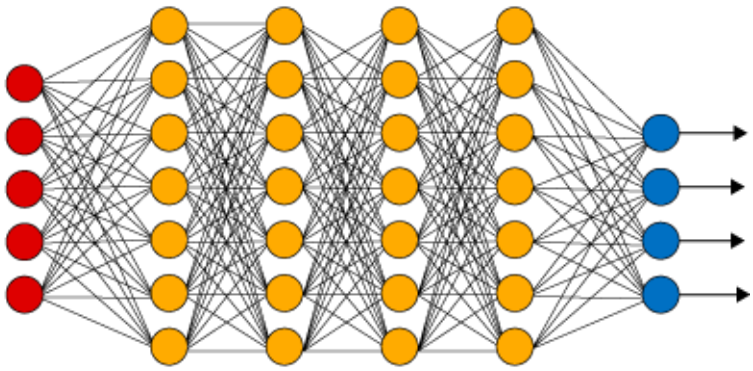

Figura 4 – Representação das conexões das camadas de nerônios artificiais em um modelo de MLP conectado, adaptado de [33]

Cada neurônio artificial nas camadas ocultas calcula as conexões sinápticas, conforme apresentado na Figura 3, de maneira sequencial até alcançar a camada de saída, que

converte essas conexões em saídas do modelo. Ao considerar o conjunto de neurônios que compõem uma camada do modelo e a representação da conexão sináptica que expressa suas ativações, é possível representar as ativações de uma camada arbitrária, denotada como $Y^{(j)}$, composta por n neurônios, em relação à camada anterior $Y^{(j-1)}$, que possui m neurônios, da seguinte forma matricial [31]:

$$Y^{(j)}_{n\times 1} = f^{(j)}\left(Y^{(j-1)}_{m\times 1}\right) = \sigma\left(W^{(j)}_{n\times m}\cdot Y^{(j-1)}_{m\times 1} + B^{(j)}_{n\times 1}\right) \tag{2.4}$$

em que:

- $Y^{(j)}_{n\times 1}$: representa o vetor de saída da j-ésima camada da rede neural. A notação $(n \times 1)$ mostra que se trata de um vetor de n linhas e 1 coluna.
- $f^{(j)}\left(Y^{(j-1)}_{m\times 1}\right)$: representa a função de ativação aplicada à saída da camada anterior $j-1$. Esta função pode ser, por exemplo, a função ReLU denotada por σ em (2.4).
- $\sigma\left(W^{(j)}_{n\times m}\cdot Y^{(j-1)}_{m\times 1} + B^{(j)}_{n\times 1}\right)$: representa a aplicação da função de ativação σ à soma ponderada das entradas da camada anterior mais o vetor de bias.
- $W^{(j)}_{n\times m}$: representa a matriz de pesos da j-ésima camada com dimensão $n \times m$, onde n é o número de neurônios na camada j e m o número de neurônios na camada $j-1$.
- $Y^{(j-1)}_{m\times 1}$: representa o vetor de saída da $(j-1)$-ésima camada que possui m elementos, onde m é o número de neurônios na camada $(j-1)$.
- $B^{(j)}_{n\times 1}$: representa o vetor de bias da j-ésima camada com n elementos correspondentes ao número de neurônios na j-ésima camada.

A Equação 2.4 é utilizada na conexão de todas j camadas que compõem a rede neural, partindo da conexão da camada de entrada $Y^{(0)}$ com a primeira camada oculta $Y^{(1)}$, até a ultima conexão de camada oculta $Y^{(j-1)}$, com a camada de saída $Y^{(j)}$. Desta forma, os parâmetros ajustáveis são compostos pela matriz $W^{(j)}$ e $B^{(j)}$ para cada camada $j \in [1, l]$ onde l representa o número total de camadas na rede neural. As matrizes compõem um conjunto Θ que representa os parâmetros treináveis de uma RNN (do inglês *Recurrent Neural Networks*), podendo assim, representá-la como uma função $\Re(x;\Theta)$ onde x é a entrada $Y^{(0)}$ e $\Re(\cdot) = f^{(l)}\left(f^{(l-1)}\left(\ldots f^{(2)}\left(f^{(1)}\left(\cdot\right)\right)\ldots\right)\right)$, gerando uma saída y a partir da entrada x e do conjunto de parâmetros Θ [31].

2.2.3 Treinamento da Rede Neural

A otimização da RNN $\Re(x;\Theta)$ consiste no treinamento da rede, que envolve aplicar uma série de passos para ajustar Θ e generalizar a solução para os valores desejados. Para minimizar a função de custo $J(\Theta)$, que é a soma dos erros entre a saída da rede $\Re(x_k;\Theta)$ e

os valores esperados y_k para todas as amostras k em um lote de treinamento k, utiliza-se a seguinte equação [31]:

$$J(\Theta) = \sum_{k} \pounds(\Re(x_k; \Theta), y_k) \tag{2.5}$$

onde:

- $J(\Theta)$: Representa a função de perda (ou função de custo) em ML. Esta função mede a performance ao comparar as previsões com os valores reais e tem como objetivo alcançar o menor valor possível ajustando os parâmetros de Θ.
- Θ : Os parâmetros treináveis em uma rede neural referem-se aos elementos que são ajustados durante o processo de treinamento para minimizar a função de custo, estes parâmetros incluem os pesos das conexões $W^{(j)}$ e Bias $B^{(j)}$.
- $\pounds(\Re(x_k; \Theta), y_k)$: Representa a função de perda que mede a discrepância ou desigualdade entre a predição da rede neural $\Re(x_k; \Theta)$ e o valor alvo y_k. Em aplicações de classificação de números é comum utilizar a função de perda *Categorical Cross-Entropy*. Esta função é definida como:

  $$\pounds(\Re(x_k; \Theta), y_k) = -\sum_{c=1}^{C} y_{k,c} \log(\Re(x_k; \Theta)_c) \tag{2.6}$$

  onde:

  - k : Representa o índice do treinamento. k nas equações indica que a perda é calculada para a k-ésimo parte do treinamento.
  - c : O índice da classe. Classes em um problema de classificação refere-se aos grupos de nos quais as amostras de dados podem ser classificados. Um exemplo, para reconhecimento de dígitos escritos a mão, cada imagem poderá ser classificada em 10 dígitos possíveis, estes dígitos são as classes.
  - C : Refere-se ao numero total de classes.
  - $y_{k,c}$ é um valor binário que indica se o exemplo k pertence à classe c (1 se pertence, 0 se não pertence).
  - $\Re(x_k; \Theta)_c$ é a probabilidade predita pela rede para o exemplo k pertencer à classe c.

A usual minimização da função de custo $J(\Theta)$ ocorre tipicamente por meio do algoritmo de descida do gradiente, ou suas variações. O algoritmo original envolve a atualização do conjunto de parâmetros Θ utilizando o negativo do gradiente da função de custo em relação a esses parâmetros. Em outras palavras, o processo consiste em iterativamente ajustar os parâmetros na direção oposta ao gradiente da função de custo [31]:

$$\Theta \leftarrow \Theta - \eta \nabla_{\Theta} J(\Theta), \tag{2.7}$$

em que:

- Θ : representa o conjunto de parâmetros treináveis da rede neural (W e bias B). O objetivo do treinamento é ajustar Θ para minimizar a função de custo $J(\Theta)$.
- $\leftarrow$: Indica a atualização dos parâmetros Θ. A seta aponta que o novo valor de Θ é obtido pela expressão à direita.
- $\Theta - \eta \nabla_{\Theta} J(\Theta)$: Expressão que atualiza os parâmetros Θ subtraindo o gradiente da função de custo multiplicado pela taxa de aprendizado η.
- η: Taxa de aprendizado que controla o tamanho do passo na direção do gradiente. É um parâmetro importante no treinamento.
- $\nabla_{\Theta} J(\Theta)$: Gradiente da função de custo $J(\Theta)$ em relação aos parâmetros Θ. Indica a direção de maior aumento da função de custo.
- $I_v(m, n)$: Elemento do kernel I_v na posição (m, n). Cada elemento do *kernel* é multiplicado pelo elemento correspondente de I_h na região selecionada.

Em (2.7) é utilizada em cada lote de treinamento para o ajuste gradativo do conjunto de Θ, este procedimento é chamado de épocas de treinamento e representa uma iteração de processos cíclicos do treinamento de uma RNN [31].

Para verificar se a rede neural é capaz de gerar uma solução eficaz, um conjunto de validação de dados é aplicado à rede, sendo geralmente diferente dos dados utilizados no treinamento. A eficiência da rede pode ser avaliada pela função de custo descrito em (2.5), comparando-se os custos associados aos conjuntos de treinamento e validação. Se o resultado apresentar um alto valor de custo tanto no conjunto de treinamento quanto no conjunto de validação, isso indica que a rede não conseguiu capturar padrões relevantes nos dados, caracterizando um problema de *Underfitting*. O *Underfitting* ocorre quando o modelo é excessivamente simples, falhando em aprender a estrutura subjacente dos dados de treinamento. Por outro lado, se o custo for baixo no conjunto de treinamento, mas elevado no conjunto de validação, a rede neural pode ter memorizado detalhes específicos do conjunto de treinamento que não são generalizáveis para novos dados. Esse fenômeno é conhecido como *Overfitting*. O *Overfitting* ocorre quando o modelo se ajusta demasiadamente aos dados de treinamento, incluindo ruídos e variações irrelevantes, resultando em uma baixa capacidade de generalização para novos conjuntos de dados. Em resumo, enquanto o *Underfitting* indica que o modelo não é suficientemente complexo para capturar os padrões dos dados, o *Overfitting* revela que o modelo é excessivamente complexo, ajustando-se a detalhes específicos dos dados de treinamento e falhando em generalizar para dados não vistos [31].

A configuração ideal se dá quando o processo evita o *Underfitting* e o *Overfitting*, ou seja, apresenta baixo valor para ambos os conjuntos amostrais.

2.2.4 Redes Neurais Convolucionais ou CNN

No contexto de ML, uma CNN (do inglês *Convolutional Neural Network*) é uma das classes de redes neurais utilizadas no processamento de imagens digitais. A ideia principal é filtrar linhas, curvas e bordas de uma imagem de forma a destacar padrões que possam ser reconhecidos pela técnica de aprendizagem.

Estas imagens que serão reconhecidas ou classificadas compõem a entrada da rede neural, e normalmente são representadas por matrizes tridimensionais, que possuem altura, largura e profundidade. A profundidade está relacionadas à quantidade de canais de uma imagem. Por exemplo, uma imagem colorida possui usualmente três canais correspondente as cores RGB (do inglês *Red, Green, Blue*), enquanto uma imagem "preto e branco", ou em escala de cinza do inglês *grayscale*, possui apenas um canal. A Figura 5 ilustra a representação de uma imagem com quatro *pixels* de altura por quatro *pixels* de largura – um pixel é a menor unidade de uma imagem. Além disso, a imagem possui três canais de cores RGB [33].

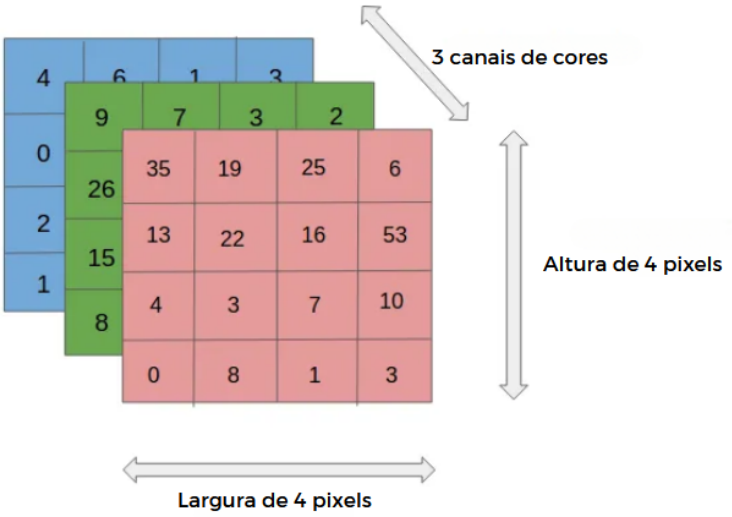

Figura 5 – Representação de uma imagem com quatro *pixels* de altura por quatro *pixels* de largura e três canais RGB, adaptado de [35]

As CNN atuam na imagem com o auxílio de filtros, ou *kernels*, para extrair características que auxiliam na classificação da imagem.

2.2.5 Filtros ou *Kernels*

O principal componente das redes neurais convolucionais são os filtros, que são matrizes quadradas de dimensões n_k x n_k, onde n_k é um numero inteiro. Estes filtros,

também conhecidos como *kernels*, são amplamente utilizados no processamento digital de sinais. Esse conceito é bem estabelecido e aplicado em programas de manipulação de imagens, como *Photoshop*, que utiliza filtros para *sharpening* (realce de nitidez), *blurring* (desfoque) e *embossing* (realce de relevo). Operações como corte e redimensionamento de imagem são, na verdade, a aplicação de um *kernel* na imagem original.

2.2.6 Convolução

Para compreender as CNN, é fundamental entender primeiro o conceito de convolução, que é feito entre dois tensores. Um tensor é uma generalização de escalas, vetores e matrizes para qualquer número de dimensões, sendo uma estrutura matemática que pode representar dados em espaços multidimensionais. A operação de convolução entre dois tensores sempre resulta em outro tensor e usualmente sua operação é indicada pelo operador *. Considerando dois tensores I_v e I_h de dimensão 3x3, a operação de convolução é feita aplicando a seguinte equação (2.8) de [33]:

$$I_v = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, I_h = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix}$$

$$I_v * I_h = C(i,j) = \sum_{m=1}^{3} \sum_{n=1}^{3} I_h(i+m-1, j+n-1) \cdot I_v(m,n) \tag{2.8}$$

em que:

- $C(i,j)$: Representa o valor da convolução na posição i,j (linha e coluna da matriz resultante C) no tensor resultante C. Cada posição i,j em C é o resultado da aplicação do *kernel*.
- $\sum_{m=1}^{3} \sum_{n=1}^{3}$: Somas duplas que percorrem os índices, como o *kernel* é 3x3, m e n variam de 1 a 3.
- O *kernel* I_v é aplicado a uma região específica de I_h é o resultado da multiplicação é somado para gerar o valor em $C(i,j)$.
- $I_h(i+m-1, j+n-1)$: Elemento do tensor maior I_h na posição $(i+m-1, j+n-1)$, este índice é ajustado para garantir que o *kernel* "deslize" sobre I_h.
- Para cada posição de (i,j) no tensor C o *kernel* I_v é multiplicado por uma submatriz de I_h centrada em (i,j).
- O termo $i+m-1$ e $j+n-1$ desloca os índices para percorrer a região de I_h correspondente ao tamanho do *kernel*.
- $I_v(m,n)$: Elemento do *kernel* I_v na posição (m,n). Cada elemento do *kernel* é multiplicado pelo elemento correspondente de I_h na região selecionada.

A ideia fundamental é que cada elemento do tensor seja multiplicado pelo elemento correspondente do *kernel*, e todos os valores resultantes sejam somados para obter um resultado. Neste contexto, o tensor de menor dimensão é chamado de *kernel*. Tipicamente, os *kernels* são pequenos, com dimensões de 3x3 ou 5x5, enquanto o tensor ao qual são aplicados é considerado grande. No caso de imagens, por exemplo, com dimensões de 1024x1024 *pixels*, a imagem possui 1.048.576 elementos.

Para compreender melhor a ideia de convolução, considera-se a matriz R_{4x4} e um *kernel* S_{3x3}. Para realizar a convolução nos seguintes passos:

$$R_{4x4} = \begin{bmatrix} r_1 & r_2 & r_3 & r_4 \\ r_5 & r_6 & r_7 & r_8 \\ r_9 & r_{10} & r_{11} & r_{12} \\ r_{13} & r_{14} & r_{15} & r_{16} \end{bmatrix}, S_{3x3} = \begin{bmatrix} s_1 & s_2 & s_3 \\ s_4 & s_5 & s_6 \\ s_7 & s_8 & s_9 \end{bmatrix}$$

01 - Seleciona-se a quantidade de elementos do *kernel* s na matriz r:

$$\begin{bmatrix} r_1 & r_2 & r_3 & r_4 \\ r_5 & r_6 & r_7 & r_8 \\ r_9 & r_{10} & r_{11} & r_{12} \\ r_{13} & r_{14} & r_{15} & r_{16} \end{bmatrix} \rightarrow \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix}$$

02 - Calcula-se a convolução dos valores selecionados de r com o *kernel* s:

$$C_1 = r_1.s_1 + r_2.s_2 + r_3.s_3 + r_4.s_4 + r_5.s_5 + r_6.s_6 + r_7.s_7 + r_8.s_8 + r_9.s_9$$

03 - Depois de calculado o valor de C_1, para o cálculo de C_2 considera-se um deslocamento 1 passo à direita do *kernel* como a seguir:

$$\begin{bmatrix} r_1 & r_2 & r_3 & r_4 \\ r_5 & r_6 & r_7 & r_8 \\ r_9 & r_{10} & r_{11} & r_{12} \\ r_{13} & r_{14} & r_{15} & r_{16} \end{bmatrix} \rightarrow \begin{bmatrix} r_2 & r_3 & r_4 \\ r_6 & r_7 & r_8 \\ r_{10} & r_{11} & r_{12} \end{bmatrix}$$

$$C_2 = r_2.s_1 + r_3.s_2 + r_4.s_3 + r_6.s_4 + r_7.s_5 + r_8.s_6 + r_{10}.s_7 + r_{11}.s_8 + r_{12}.s_9$$

04 - Quando se chega à borda, não é possível realizar mais passos. Porém, se a matriz fosse maior, o *kernel* continuaria deslizando sobre a tabela principal até encontrar o final da borda. Ao final, devemos deslocar uma linha abaixo e repetir o processo de deslizamento até encontrar a borda novamente. Esse deslizamento é chamado de *stride*, denominado por s, ou seja, s=1 quer dizer que deslocamos de uma em uma coluna, s=2 de duas em duas e assim por diante.

$$\begin{bmatrix} r_1 & r_2 & r_3 & r_4 \\ r_5 & r_6 & r_7 & r_8 \\ r_9 & r_{10} & r_{11} & r_{12} \\ r_{13} & r_{14} & r_{15} & r_{16} \end{bmatrix} \rightarrow \begin{bmatrix} r_5 & r_6 & r_7 \\ r_9 & r_{10} & r_{11} \\ r_{13} & r_{14} & r_{15} \end{bmatrix}$$

$$C_3 = r_5.s_1 + r_6.s_2 + r_7.s_3 + r_9.s_4 + r_{10}.s_5 + r_{11}.s_6 + r_{13}.s_7 + r_{14}.s_8 + r_{15}.s_9$$

$$\begin{bmatrix} r_1 & r_2 & r_3 & r_4 \\ r_5 & r_6 & r_7 & r_8 \\ r_9 & r_{10} & r_{11} & r_{12} \\ r_{13} & r_{14} & r_{15} & r_{16} \end{bmatrix} \rightarrow \begin{bmatrix} r_6 & r_7 & r_8 \\ r_{10} & r_{11} & r_{12} \\ r_{14} & r_{15} & r_{16} \end{bmatrix}$$

$$C_4 = r_6.s_1 + r_7.s_2 + r_8.s_3 + r_{10}.s_4 + r_{11}.s_5 + r_{12}.s_6 + r_{14}.s_7 + r_{15}.s_8 + r_{16}.s_9$$

05 - O resultado da convolução, tem-se um tensor T_{2x2} formado pelas convoluções da matriz R_{4x4}:

$$T_{2x2} = \begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix}$$

A operação de convolução aplicada no processo de reconhecimento de imagem extrai características relevantes, como bordas e texturas, de uma imagem. A matriz resultante da convolução, assim como no exemplo apresentado, é uma versão filtrada da imagem original, essencial para a detecção e classificação de objetos.

2.2.7 Agrupamento ou *Pooling*

Pooling é a segunda operação fundamental de uma CNN. Essa operação é semelhante a convolução, para compreender sua aplicação, exemplifica-se com a aplicação do método de *max pooling*, considerando a matriz P_{4x4} definida como [33]:

$$P = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \\ p_{13} & p_{14} & p_{15} & p_{16} \end{bmatrix} \tag{2.9}$$

Para a performance do *pooling*, é necessário definir uma região de interesse. Assim como na convolução, neste exemplo utiliza-se uma matriz $n_k = 2$, a qual será posicionada no topo esquerdo da matriz:

$$\begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \\ p_{13} & p_{14} & p_{15} & p_{16} \end{bmatrix} \rightarrow \begin{bmatrix} p_1 & p_2 \\ p_5 & p_6 \end{bmatrix} \tag{2.10}$$

Para encontrar o valor resultante B_1 usa-se o maior valor "max" da região de interesse como apresentado em (2.16):

$$B_1 = max(p_1, p_2, p_5, p_6) \tag{2.11}$$

A diferença é que, agora, em vez de deslizar o *kernel*, o deslocamento ocorre no sentido direito (destacado em negrito na matriz A) repetindo o procedimento na linha de baixo, até o final da matriz:

$$\begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \\ p_{13} & p_{14} & p_{15} & p_{16} \end{bmatrix} \rightarrow \begin{bmatrix} p_3 & p_4 \\ p_7 & p_8 \end{bmatrix} \rightarrow B_2 = max(p_3, p_4, p_7, p_8) \tag{2.12}$$

$$\begin{bmatrix} p_1 & p_2 & p_3 & a_4 \\ p_5 & p_6 & p_7 & a_8 \\ p_9 & p_{10} & p_{11} & p_{12} \\ p_{13} & p_{14} & p_{15} & p_{16} \end{bmatrix} \rightarrow \begin{bmatrix} p_9 & p_{10} \\ p_{13} & p_{14} \end{bmatrix} \rightarrow B_3 = max(p_9, p_{10}, p_{13}, p_{14}) \tag{2.13}$$

$$\begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \\ p_{13} & p_{14} & p_{15} & p_{16} \end{bmatrix} \rightarrow \begin{bmatrix} p_{11} & p_{12} \\ p_{15} & p_{16} \end{bmatrix} \rightarrow B_4 = max(p_{11}, p_{12}, p_{15}, p_{16}) \tag{2.14}$$

Como resultado do exemplo, obtém-se uma matriz V_{2x2} com o valor máximo de cada janela da região de interesse:

$$\mathbf{V} = \begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix} \tag{2.15}$$

O *pooling* é essencial no processo de reconhecimento de imagens em uma rede neural convolucional porque reduz a dimensionalidade dos mapas de características, mantendo as informações mais relevantes. Isso não apenas diminui o custo computacional, mas também ajuda a controlar o *overfitting*, tornando o modelo mais generalizável. Ao extrair os valores máximos de regiões específicas, o *max pooling* preserva as características dominantes e importantes da imagem, contribuindo para uma detecção mais eficaz e eficiente de padrões durante o treinamento da rede neural [33].

2.2.8 Preenchimento ou *Padding*

O *Padding* é uma técnica utilizada para otimizar o resultado da convolução, dimensionando uma imagem, acrescentando à ela *pixels* na parte superior, inferior, à esquerda e

à direita da imagem, para que as matrizes resultantes tenham o mesmo tamanho da matriz original [33]. O *padding* oferece várias vantagens importantes no contexto das CNNs:

- Preservação da Informação nas Bordas: Sem *padding*, as operações de convolução reduzem o tamanho da imagem a cada camada, o que pode resultar na perda de informações importantes, especialmente nas bordas
- Manutenção do Tamanho da Imagem: Ao adicionar *pixels* de *padding*, o tamanho da matriz de características resultante pode ser mantido igual ao da matriz original. Isso é particularmente útil em arquiteturas profundas de redes neurais, onde múltiplas camadas de convolução são aplicadas.
- Melhoria da Precisão e Generalização: O *padding* ajuda a melhorar a precisão dos modelos de CNN ao permitir que a rede capture padrões de forma mais eficaz em toda a imagem, incluindo nas bordas
- Controle do *Overfitting*: Ao manter o tamanho da matriz de características constante, o *padding* permite que a rede utilize um número maior de parâmetros de forma eficiente, o que pode ajudar a controlar o *overfitting*.

Em resumo, *padding* é uma técnica essencial no processamento de imagens com CNN. Esta técnica preserva informações importantes nas bordas da imagem, mantém o tamanho das matrizes de características, melhora a precisão e generalização dos modelos e ajuda a controlar o *overfitting*.

2.2.9 Construção da CNN

Redes neurais baseadas em CNN utilizam operações de convolução e *pooling* para construir as camadas usadas nas CNN. Uma camada convolucional toma como entrada uma matriz tridimensional Y (imagem) com dimensões $n_b \times n_b \times c$ e aplica um certo numero de *kernels*, *bias* e uma função de ativação para induzir a não linearidade ao resultado da convolução, resultando em uma matriz X. Ao aplicar n_c *kernels* a uma matriz Y, obtém-se uma matriz X com dimensões $n_b \times n_b \times n_c$, onde n_c é o numero de *kernels* aplicados. Isto significa que cada *kernel* gera uma matriz de saída correspondente, formando as camadas de X:

$$X_{i,j,k} \quad \forall i,j \in [1, n_b], \quad k \in [1, n_c] \tag{2.16}$$

em que:

- $X_{i,j,k}$: Representa o valor da matriz de saída X na posição i, j da camada k.
- i, j: Índices que percorrem as dimensões espaciais (largura e altura) da matriz X, variando de 1 a n_b.

- k: Índice que percorre as diferentes camadas da matriz X, variando de 1 a n_c, onde n_c é o numero de *kernels* aplicados.

A saída da convolução da imagem de entrada com o primeiro *kernel* resulta em:

$$X_{i,j,2} \quad \forall i,j \in [1, n_b] \tag{2.17}$$

em que:

- $X_{i,j,1}$: Representa o valor da matriz de saída X na posição i,j da primeira camada ($k = 1$), resultante da aplicação do primeiro *kernel*
- i,j: Índices que percorrem as dimensões espaciais (largura e altura) da matriz X, variando de 1 a n_b.

A saída da convolução com o segundo *kernel* é dada por:

$$X_{i,j,2} \quad \forall i,j \in [1, n_b] \tag{2.18}$$

em que:

- $X_{i,j,2}$: Representa o valor da matriz de saída X na posição i,j da segunda camada ($k = 2$), resultante da aplicação do segundo *kernel*
- i,j: Índices que percorrem as dimensões espaciais (largura e altura) da matriz X, variando de 1 a n_b.

E assim por diante, até o n_c-ésimo *kernel*. Os pesos e parâmetros de prendizagem que a rede aprende durante a fase de treinamento são os elementos dos *kernels*.

Nas CNN, as camadas convolucionais e de *pooling* são empilhadas, geralmente com uma camada de *pooling* após cada camada de convolução. Embora as camadas de *pooling* não possuam parâmetros de aprendizagem, elas simplificam a informação da camada anterior, reduzindo a dimensionalidade da matriz. Ao final, é introduzida a estrutura de neurônios de uma RNA convencional, com neurônios e função de ativação para realizar classificação. A Figura 6 mostra a representação de uma CNN

Na figura 6, a "imagem de entrada" representa a camada de entrada da CNN, seguida da extração de características, composta por 3 camadas de convolução com função de ativação ReLU e *pooling*, concluindo a etapa de convolução. Sequencialmente, é aplicada a estrutura de uma RNA composta por 3 camadas ocultas com função de ativação *Softmax*. Ao final, obtém-se a classificação dos dígitos.

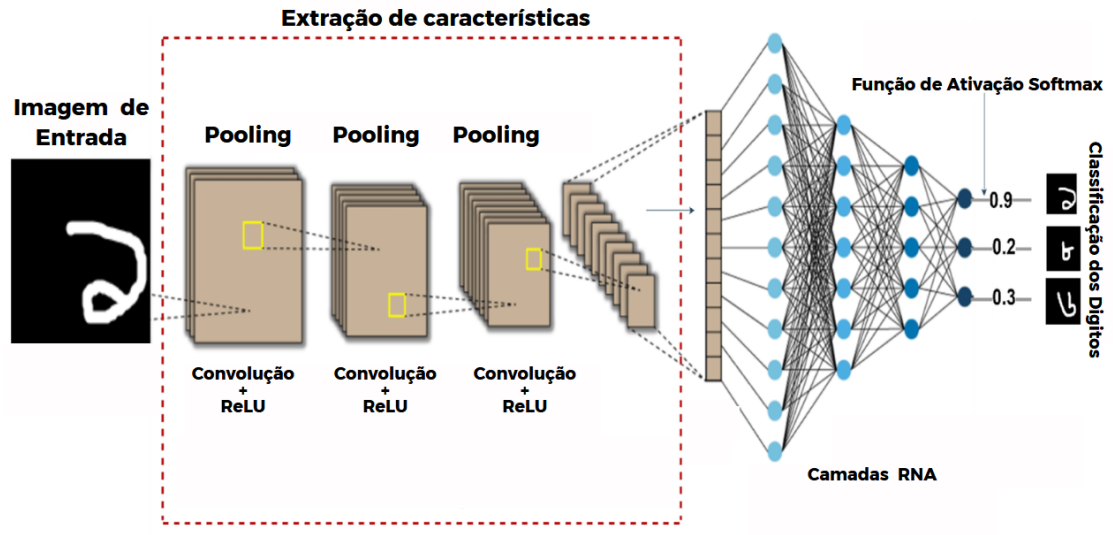

Figura 6 – Representação de uma CNN, adaptado de [35]

2.3 Conectividade

A transmissão de dados é um recurso importante e característico da IoT, com vários protocolos disponíveis para sua execução. A escolha do protocolo de transmissão geralmente está associada à distância de envio, à largura de banda e ao nível de potência requerido para a transmissão [27].

2.3.1 Os protocolos de transmissão de dados para IoT

Existem diferentes protocolos de comunicação amplamente utilizados em aplicações de IoT, como *Bluetooth, Zigbee, Wi-Fi, NFC (do inglês Near Field Communication)* entre outros. É comum classificar esses protocolos de acordo com a largura de banda e o alcance, como na Figura 7.

Na Figura 7, os protocolos posicionados no canto inferior direito do gráfico, além prover o maior alcance na transmissão de dados, também são caracterizados por baixo consumo de energia, sendo frequentemente escolhidos para aplicações da IoT [27]. Os protocolos mais comumente utilizados são:

- *NFC*: Protocolo de comunicação de curto alcance com baixo consumo de energia, baseado na tecnologia de indução magnética, o *NFC* é considerado uma evolução simplificada do RFID (do inglês *Radio Frequency Identification*). A comunicação em *NFC* envolve um dispositivo iniciador (leitor) e um dispositivo alvo, o iniciador gera um campo de ondas de rádio de baixa frequência, tipicamente 13,56 MHz, permitindo uma conexão bidirecional. As taxas de transferência de dados variam entre 106 kbs, 212 kbs, 424 kbs e 848 kbs, dependendo da aplicação. Devido à natureza

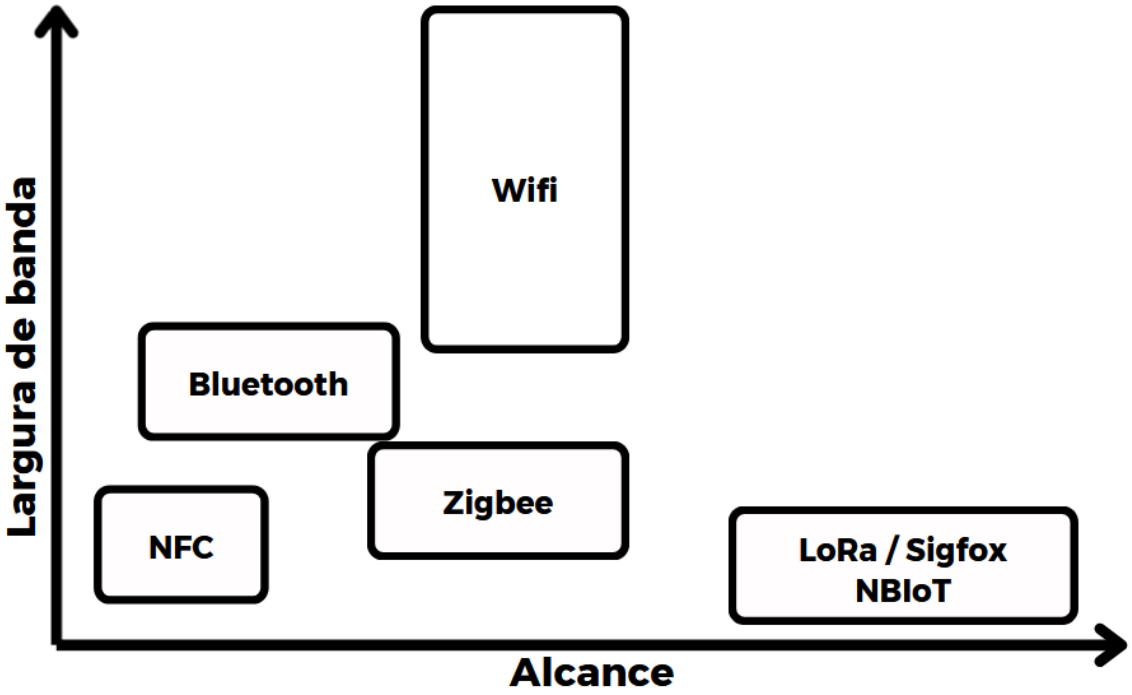

Figura 7 – Relação da largura de banda com o alcance nos protocolos utilizados em IoT, adaptado de [29]

da tecnologia, as conexões *NFC* requerem proximidade física entre os dispositivos, normalmente em uma distância de até 20 cm [34].

- O *Bluetooth*, utiliza ondas de rádio de baixa potência (1 mW) para transmitir dados em curtas distâncias, funciona na faixa de 2,4 GHz a 2,485 GHz e usa "saltos adaptativos em frequência" para minimizar interferências. Com alcance de 1 a 100 metros e taxa de transferência de 3 Mbps, o *Bluetooth* é eficiente e ideal para dispositivos interconectáveis de curto alcance [35].

- *ZigBee* é um padrão que define uma pilha protocolar simplificada, caracterizada por baixo custo, baixo consumo de energia. Com alcance de 30 metros, funciona na faixa de 2,4 GHz, possui uma taxa de transferência limitada a 250 kbps, devido a essas características, o *ZigBee* é amplamente utilizado em monitoramento e controle industrial, automação residencial e sistemas de energia [36].

- *WiFi* é uma tecnologia de rede local sem fio, padronizada pelo *IEEE 802.11*, amplamente conhecida pela abreviação de *wireless fidelity*. O termo *WiFi* é uma marca registrada da WECA (do inglês *Wireless Ethernet Compatibility Alliance*), hoje conhecida como *Wi-Fi Alliance*. As redes *wireless* funciona na faixa de 2,4 GHz a 5,0 GHz e taxa de transmissão de 500 Mbps, seu alcance pode variar entre 15 e 100 metros, dependendo das condições e do equipamento utilizado. Com base no alcance, as redes sem fio podem ser classificadas em WPAN (do inglês *Wireless Personal*

Area Network), que cobre uma área pessoal restrita, ou WLAN (do inglês *Wireless Local Area Network*), que abrange uma área maior, como uma casa, escritório ou campus [37].

- LoRa é uma forma de modulação sem fio projetada para permitir a comunicação de longo alcance e baixo consumo de energia entre dispositivos conectados ao IoT. Esta tecnologia opera em diferentes bandas de frequência, dependendo da região: 868 MHz na Europa, 915 MHz na América do Norte e 923 MHz na Ásia, graças a essas características, LoRa pode alcançar distâncias de até 15 km em áreas rurais e até 5 km em áreas urbanas, com uma taxa de dados de até 50 kbps, tornando-a ideal para aplicações que exigem comunicação de longo alcance e eficiência energética [38].

- O *Sigfox* é composta por estações base e antenas estrategicamente distribuídas para fornecer cobertura de longo alcance para dispositivos IoT. Operando em frequências de 868 MHz na Europa e 902 MHz nos EUA, o *Sigfox* utiliza uma largura de banda estreita de 100 Hz ("Ultra Narrow Band"), o que minimiza colisões e maximiza o alcance. A taxa de transmissão é de 100 bps, adequado para transmissões de dados leves. Em áreas urbanas, o alcance varia de 1 a 10 km, podendo chegar a até 50 km em áreas rurais ou com menos obstáculos, proporcionando uma solução eficiente em termos de energia e comunicação para aplicações IoT de médio/longo alcance [39].

- NBIoT é uma tecnologia projetada para otimizar o uso da banda passante e o consumo de energia em comunicações IoT, posicionando-se como uma alternativa a tecnologias como *Sigfox*, *ZigBee* e LoRa. O NBIoT pode ser implantado de três maneiras: *In-band*, onde compartilha recursos com LTE (do inglês *Long-Term Evolution*); *Guard-band*, utilizando blocos de recursos não utilizados entre operadoras LTE e *Standalone*, operando em espectro dedicado. Opera em várias faixas de frequência que são geralmente reutilizadas de redes celulares existentes, dependendo da região e da operadora, usualmente na América são utilizadas as bandas 20 de 800 MHz, banda 5 de 850 Mhz, banda 28 de 700 Mhz, banda 26 de 850 Mhz e banda 3 de 1800 Mhz. A taxa transmissão de dados é de 66bps, e o alcance da aplicação é de até 10 km [40].

Neste trabalho, foram utilizados dois protocolos de comunicação, LoRa e NBIoT. O protocolo LoRa devido seu baixo consumo de energia e longo alcance, permitindo a comunicação eficiente entre dispositivos em áreas remotas ou de difícil acesso, onde a duração da bateria é crítica. Já o protocolo NBIoT devido sua capacidade de operar em redes celulares existentes, além de suportar uma longa duração de bateria e transmissão de dados em alta velocidade com alta confiabilidade. Os tópicos seguintes abordam detalhadamente o funcionamento destas duas tecnologias.

2.3.2 LoRa

O LoRa é uma modulação patenteada pela empresa desenvolvedora de semicondutores *Semtech*® com base na modulação de espectro CSS (do inglês *chirp spread spectrum*). O LoRa é caracterizado por fornecer longo alcance na transmissão de dados, baixo consumo de energia, baixa taxa de dados e o envio seguro de informação [41].

O LoRa utiliza de bandas de radiofrequências de sub gigahertz sem licença como, por exemplo, 433 MHz, 868 MHz na Europa, 915 MHz na Austrália, Brasil e Norte dos Estados Unidos e 433 MHz na Ásia. Essa tecnologia permite a transmissão de dados de longo alcance (mais de 10 km com visada) com um baixo consumo de energia [42].

A modulação LoRa utiliza chirp (do inglês *Compressed High Intensity Radar Pulse*) ou modulação chirp de amplo espectro para ter várias transmissões ao mesmo tempo para um mesmo canal de comunicação. Ao emitir um sinal LoRa modulado, cada símbolo possui uma forma básica, como apresentado na Figura 8 [42].

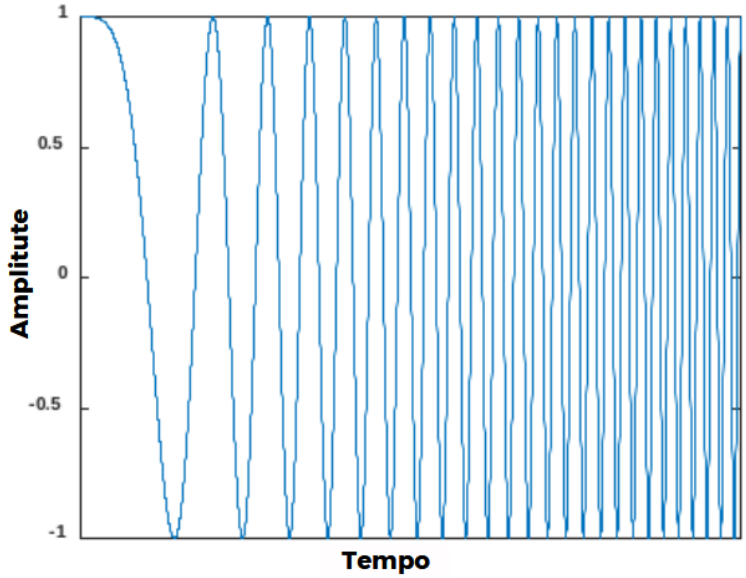

Figura 8 – Sinal chirp, adaptado de [36]

Cada símbolo leva um tempo para ser transmitido (*Tsymbol*) e esse tempo depende diretamente do fator de espalhamento SF (do inglês *Spreading Factor*). O fator de espalhamento SF controla a quantidade de tempo que cada símbolo leva para ser transmitido, determinando quantos chips (sinais) são usados para representar cada byte de dados. Quanto maior o SF maior o número de chips, o que torna a transmissão mais robusta contra ruídos, mas também mais lenta. Por exemplo, considerando a mesma largura de banda, o tempo de transmissão com um SF de 8 (SF8) isto significa que 2^8 ou 128 chips são usados para cada byte, enquanto um SF de 12 (SF12) onde 2^{12} ou 4096 chips são usados para cada byte, isto implica que com o mesmo canal de largura de banda um (SF12) levará mais tempo para transmitir o mesmo número de bytes comparado com SF8. A Figura 9 apresenta um gráfico do tempo de transmissão por pacote de símbolos [42].

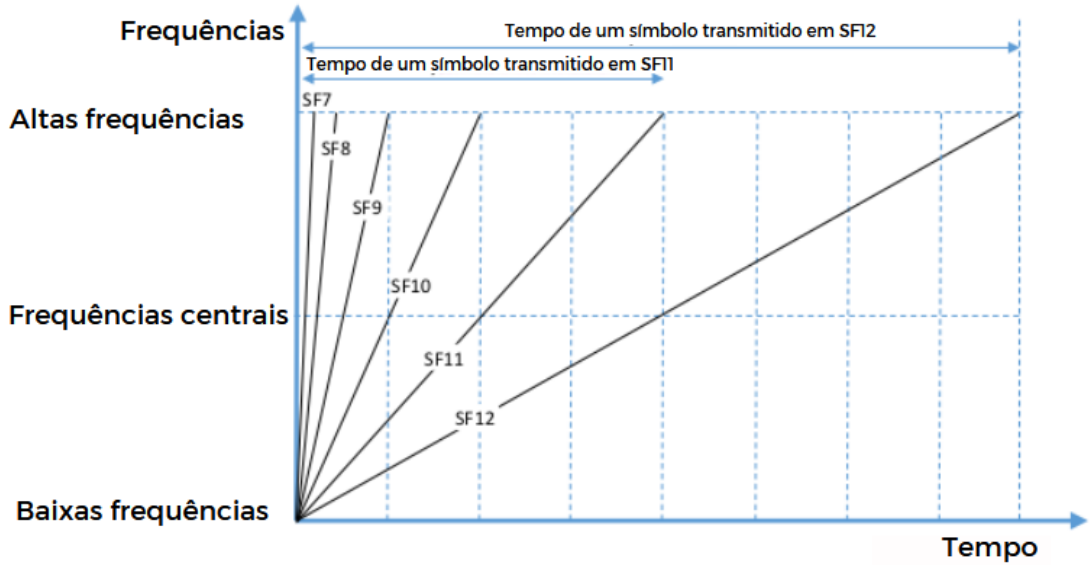

Figura 9 – Tempo de transmissão por pacote de símbolo, adaptado de [36]

O tempo de transmissão por símbolo pode ser expresso por meio da largura de banda:

$$Tsymbol = \frac{2^{SF}}{BW} \tag{2.19}$$

em que:

- $Tsymbol$ representa a duração de um único símbolo durante a transmissão em uma comunicação LoRA, expressa em segundos (s).
- F representa o fator de espalhamento.
- BW representa a largura de banda do canal, expressa em hertz (Hz)

A razão de símbolos por segundo é expressa por $Fsimbol$:

$$\frac{1}{Tsymbol} = Fsimbol = \frac{BW}{2^{SF}} \tag{2.20}$$

Portanto, quanto maior a largura de banda, maior a taxa de símbolo.

O tempo total que um pacote de dados leva para ser transmitido de um ponto A até um ponto B é denominado tempo no ar. Esse tempo depende da quantidade de símbolos presentes no quadro *LoRa*, incluindo o cabeçalho, a mensagem, o preâmbulo e o CRC (do inglês *Cyclic Redundancy Check*). Em *LoRa*, cada símbolo representa múltiplos bits, e a relação entre o fator de espalhamento SF e a largura de banda BW influencia a taxa de bits efetiva. A taxa de bits é dada pela equação:

$$Taxa\,de\,bits = \frac{SF \times BW}{2^{SF}} \tag{2.21}$$

Porém, os bits não são enviados individualmente ou em pacotes encadeados, pois isso poderia causar problemas relacionados à transmissão de pequenos pacotes. Para solucionar esse problema e garantir a robustez da comunicação, existe a taxa de codificação CR (do inglês *Coding Rate*) A taxa de codificação aumenta o número de bits transmitidos com o objetivo de permitir a detecção e a correção de erros.

No LoRa CR é definido pela razão entre os bits de dados e o número total de bits transmitidos. Existem quatro opções de CR no LoRa (4/5, 4/6, 4/7, 4/8), onde, por exemplo, CR = 4/8 significa que para cada quatro bits de dados são transmitidos oito bits no total, ou seja, quatro bits adicionais são usados para correção de erros. O cálculo para determinar o tempo no ar é dado pela equação:

$$Tempo\,no\,ar = Nsymbols.Tsymbols \tag{2.22}$$

Onde $Nsymbols$ representa o número total de símbolos que compõem o pacote a ser transmitido. O termo $Nsymbols$ depende dos parâmetros apresentado na equação 2.23:

$$n_{symbols} = (n_p + 4,25) + 8 + max\left[ceil\left(\frac{8 * mensagem - 4.SF + 28 + 16 - 20.H}{4(SF - 2DE)}\right)\right] \tag{2.23}$$

em que:

- n_p: representa o número de símbolos do preâmbulo configurado.
- A função *max* é usado para garantir que o número de símbolos da mensagem não seja negativo
- A função *ceil* é usado para garantir que o valor resultante seja sempre um numero inteiro com tendencia a arrendondar para o maior valor.
- mensagem: O comprimento da mensagem em bytes.
- SF: O valor de *Spreading Factor* utilizado na transmissão.
- H: acrônimo do termo em inglês *HeaderEnabled* é uma constante que constitui 0 se o cabeçalho estiver habilitado e 1 caso contrário.
- DE: acrônimo do termo em inglês *Data Optimization*, representa uma constante com o valor 1 se esta funcionalidade for habilitada e 0 caso contrário.

2.3.3 LTE e o NBIoT

O NBIoT é descrito como uma inovação na 3GPP (do inglês *3rd Generation Partnership Project*), integrado ao padrão LTE, faz parte da tecnologia 4G, com o propósito de aproveitar a infraestrutura de *hardware* e parte da pilha de protocolos LTE. Essa tecnologia visa atender as demandas de aplicações IoT de baixo custo, integrando-se de maneira suave à sistemas já existentes, requerendo uma faixa de frequência restrita [43]. Embora

compartilhe os protocolos das camadas superiores da rede LTE (RRC (do inglês *Radio Resource Control*), RLC (do inglês *Radio Link Control*) e MAC (do inglês *Medium Access Control*)), o NBIoT apresenta distinções notáveis, incluindo o uso de acesso ao meio de forma aleatória e alterações nos canais físicos para se adaptar à banda reduzida, conforme explicado por [44].

Esse padrão recente opera no modo FDD (do inglês *Frequency Division Duplex*) e estabelece portadoras com uma largura de banda de 200 kHz, correspondente a uma largura de BR (Banda Reduzida) no LTE. Essas frequências podem estar contidas tanto na faixa do LTE quanto fora dela, dependendo da configuração específica da instalação do NBIoT.

Como ilustrado na Figura 10, o 3GPP define três formatos para a implantação do NBIoT: dentro da banda, banda de guarda e o NBIoT independente [43].

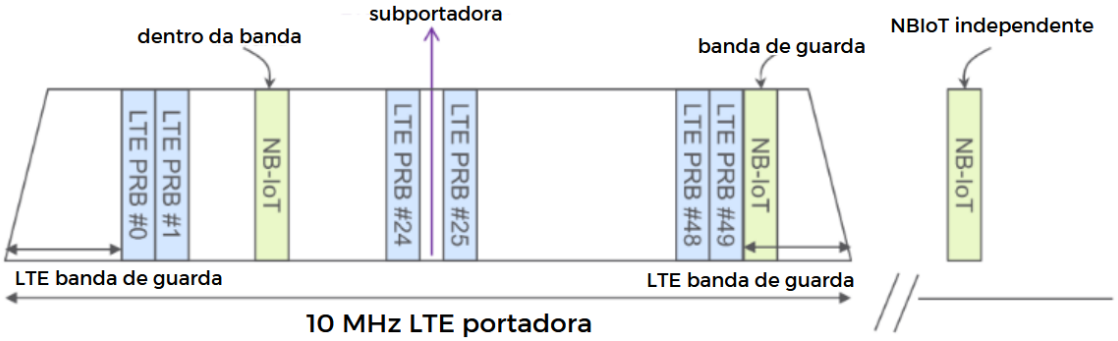

Figura 10 – Tipos de implementação do NBIoT no domínio da frequência, adaptado de [37]

Na modalidade dentro da banda, do inglês *in-band*, as portadoras do NBIoT utilizam a faixa de frequência correspondente a uma largura de banda reduzida BR em uma portadora LTE. Isso implica em uma restrição dos recursos disponíveis para outros tipos de dispositivos, já que ambas as tecnologias compartilham a mesma faixa de frequência. Essa forma de multiplexação possibilita uma operação mais econômica, uma vez que as duas tecnologias coexistem na mesma faixa de frequência [43].

Na configuração de banda de guarda, do inglês *guard band*, a portadora NBIoT utiliza parte do espectro de frequência que não está sendo aproveitado na faixa de guarda do LTE. Essa abordagem permite que o NBIoT opere em uma faixa adjacente à do LTE, utilizando eficientemente o espaço de frequência disponível na banda de guarda. Essa estratégia minimiza a interferência entre as tecnologias e otimiza a utilização do espectro, proporcionando uma coexistência mais eficaz entre o NBIoT e o LTE [43].

No modo NBIoT independente, do inglês *standalone*, o NBIoT opera em frequências independentes, ou seja, fora de uma portadora LTE. Essa configuração é especialmente útil para a substituição de tecnologias mais antigas, como o GSM (do inglês *Global System*

for Mobile Communications). Nesse cenário, o NBIoT não compartilha a mesma faixa de frequência com o LTE, proporcionando uma solução autônoma para atender as demandas específicas de dispositivos IoT, sem a necessidade de competir por recursos com outras tecnologias [43].

A alta capacidade do padrão NBIoT é, em grande parte, derivada da capacidade de multiplexar sua estreita faixa de frequência para acomodar numerosos dispositivos. Essa habilidade é alcançada por meio de transmissões conhecidas como *single-tone* e *multi-tone*. É importante observar que esses tipos de transmissões são viáveis apenas no *uplink*. No *downlink*, o NBIoT opera de maneira semelhante ao *downlink* do LTE, mas com a distinção de ser composto por apenas um bloco de recursos BR [43].

Similar ao padrão LTE, o espaço de frequência de uma portadora NBIoT é subdividido em subportadoras. Contudo, neste novo padrão, são definidas duas configurações distintas de espaçamento entre subportadoras: 3,75 kHz e 15 kHz. A transmissão *multi-tone* é exclusiva para a configuração de espaçamento de subportadoras de 15 kHz. Essas opções proporcionam flexibilidade na configuração do NBIoT, permitindo a adaptação às necessidades específicas de diferentes cenários de implementação [43].

O espaçamento de 15 kHz é idêntico ao utilizado no padrão LTE, possibilitando a existência de 12 subportadoras por portadora NBIoT. Em contraste, o espaçamento de 3,75 kHz permite a existência de 48 subportadoras, mas pode apresentar o desafio de causar interferência com portadoras LTE. A escolha entre essas configurações dependerá das necessidades específicas da implementação, ponderando a capacidade desejada, possíveis interferências e outros requisitos do sistema [43].

As transmissões *single-tone* possibilitam a transmissão utilizando apenas uma subportadora, permitindo a realização de 12 e 48 transmissões simultâneas, dependendo do espaçamento escolhido (15 kHz ou 3,75 kHz, respectivamente). Por outro lado, as transmissões *multi-tone* efetuam a transmissão utilizando 3, 6 e 12 subportadoras, proporcionando uma abordagem mais ampla para a comunicação simultânea de dados em ambientes NBIoT. Essa flexibilidade em termos de número de subportadoras utilizadas pode ser ajustada conforme as exigências específicas do cenário de implementação [43].

A utilização de transmissões *multi-tone* oferece a vantagem significativa de reduzir a duração da transmissão. A Tabela 2 exibe as diversas configurações de transmissões, suas durações e a ocupação das subportadoras, destacando as variações na eficiência temporal dessas transmissões. Essa característica pode ser crucial para otimizar o tempo de comunicação e maximizar a eficiência espectral em ambientes NBIoT [43].

Cada configuração gera diferentes tipos de UR (Unidade de Recurso), cada uma com características distintas de ocupação de subportadoras e duração. Apesar dessas variações, todas as configurações de UR resultam na mesma quantidade de símbolos SCFDMA

Tabela 2 – Configuração de transmissões NBIoT.

Transmissão	Espaçamento subportadoras	Subportadoras	Duração
Single-tone	3,75 MHz	1	32 ms
	15 MHz	1	8 ms
Multi-tone	15 MHz	3	4 ms
	15 MHz	6	2 ms
	15 MHz	12	1 ms

(do inglês *Single Carrier Frequency Division Multiple Access*), mantendo, assim, a consistência na quantidade de dados transmitidos por UR. Essa uniformidade na quantidade de símbolos SCFDMA contribui para uma gestão eficiente do espectro e padronização na transmissão de dados, independentemente da configuração específica utilizada [43].

2.4 Medição de Energia

Em [45] é apresentado um trabalho que aborda o "modelo de consumo de energia para dispositivos LoRaWAN (do inglês *Long Range Wide Area Network*)". Esse trabalho divide a tarefa de obter o consumo em duas etapas, onde a primeira etapa consiste em montar um *setup*, que entre a alimentação e o dispositivo medidor é inserido um elemento *shunt* – um resistor de 1 ohm, em série com o equipamento. Com auxilio de um osciloscópio, é obtida a forma de onda da corrente em relação ao tempo. A Figura 11 apresenta um exemplo do resultado obtido por esse trabalho. Nota-se que há demarcações de tempo, definidas como: pré-transmissão (t_0 a t_1), transmissão (t_1 a t_2), espera para o intervalo de recepção (t_2 a t_3), recepção do ACK (t_3 a t_4) e pós-recepção (t_4 a t_5).

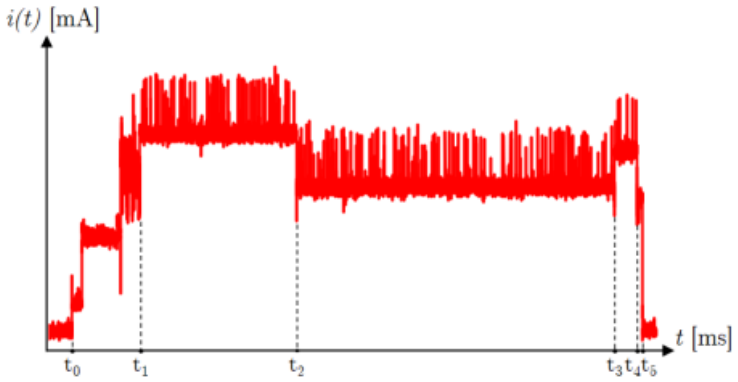

Figura 11 – Formas de onda da tensão e corrente de alimentação em cada uma das etapas de funcionamento do ciclo de operação do dispositivo, adaptado de [38]

Uma segunda etapa do trabalho visa identificar o tamanho do pacote de *uplink* - dados de recepção (S), taxa de transmissão (R) e potência de transmissão configurada

(P). A partir desse modelo é possível obter a equação geral de consumo, dada por [45]:

$$E_n = E_N1 + E_N2(S, R, P) + E_N3 + E_N4(R) + E_N5 \tag{2.24}$$

Onde E_n representa a energia consumida pelo nó em um ciclo de operação completo e E_NK representa a energia consumida pelo nó na etapa K. Assim, a partir do *setup* montado na etapa 1, a energia consumida na etapa de funcionamento do dispositivo ENK pode ser calculada subtraindo a energia sobre o resistor *shunt* da energia total do circuito, ou seja [45]:

$$E_NK = E_TK - E_SK = \int_{t_{k-1}}^{t_k} \frac{v(t).v_s(t)}{R_s}\,dt - \int_{t_{k-1}}^{t_k} \frac{v_s^2(t)}{R_s}\,dt \tag{2.25}$$

Dado em Joule, o índice K varia de 1 a 5, representando as etapas de funcionamento do dispositivo. Na equação (2.25), é dada em função da tensão de alimentação do circuito ($v(t)$) e da tensão sobre o resistor *shunt* ($v_s(t)$). Finalmente, a energia consumida pelo nó em todo o ciclo de operação é dada pela somatória das energias em cada uma das etapas de funcionamento [45].

3 Projeto Experimental

Este Capítulo trata da implementação da solução de processamento de borda com dispositivos IoT (LoRa e NBIoT). Será apresentado o dispositivo de processamento de imagem, dispositivos de transmissão de dados, montagem dos transmissores e receptores, apresentação e configuração da aplicação de processamento de imagem, adaptações no *firmware* da aplicação principal e a montagem do projeto final em uma bancada de teste.

3.1 Dispositivo de processamento de imagem

O microcontrolador principal utilizado para o processamento e reconhecimento de imagem foi um ESP32-CAM, da empresa Espressif. As características de funcionamento, dados do processador, IO e alimentação são descritos no *datasheet* disponível pelo fabricante [5]. Esse dispositivo foi escolhido devido à sua alta capacidade de processamento, além de possuir uma antena integrada de 2,4 GHz para comunicação *WiFi* e hardware dedicado ao funcionamento desse protocolo. O ESP32-CAM possui câmera modelo OV2640, de 2 megapixels de resolução; um *LED Flash*, para auxiliar na captura de imagens em ambientes com pouca ou nenhuma luminosidade; e entrada para cartão de memória, que disponibiliza a extensão de memória não volátil do microcontrolador, permitindo salvar imagens em formatos codificados. Possui PSRAM (do inglês *Pseudo Static Random Access Memory*) para salvar em memória temporária valores voláteis, como fotos, e executar tarefas complexas de reconhecimento de imagem. A Figura 12 mostra o modelo de ESP32-CAM utilizado neste trabalho. No entanto, o ESP32-CAM não possui comunicação IoT como LoRa ou NBIoT. Para superar essa limitação, dispositivos de transmissão de dados LoRa e NBIoT foram integrados ao sistema. O capítulo subsequente aborda em detalhes esses dispositivos de transmissão de dados e explica como eles foram integrados ao ESP32-CAM.

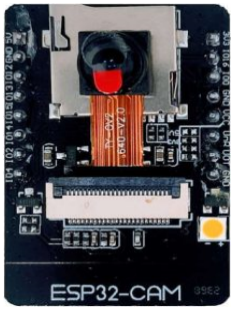

Figura 12 – Dispositivo principal de captura de imagens, figura do autor.

Este microcontrolador foi utilizado para embarcar o *firmware* de processamento e reconhecimento de imagem. O *firmware* de processamento e reconhecimento de imagem é

o *software* embutido no ESP32-CAM que controla todas as operações do dispositivo. Este *firmware* é responsável por capturar as imagens através da câmera OV2640, processá-las em tempo real e realizar análises de reconhecimento de imagem utilizando algoritmos de IA. Ele gerencia a comunicação entre os componentes de *hardware*, como o *LED Flash*, a memória PSRAM e o cartão de memória, otimizando o uso dos recursos disponíveis para garantir um desempenho eficiente. Essa integração de *hardware* e *software* faz do ESP32-CAM um dispositivo potente e versátil para aplicações de monitoramento e automação que requerem capacidades avançadas de visão computacional.

3.2 Dispositivos de transmissão de dados

Este sub-tópico aborda em detalhes os módulos de comunicação LoRa E32-915T20D da empresa Ebyte [46] e o módulo NBIoT da empresa Telit [47], utilizados para adicionar ao dispositivo de reconhecimento de imagens as duas novas tecnologias de comunicação, ampliando assim suas capacidades de comunicação para aplicações de IoT.

3.2.1 Transmissor LoRa

O E32 é um transmissor UART (do inglês *Universal Asynchronous Receiver-Transmitter*) - *wireless*, baseado no chip SEMTECH's SX1276 RF, que trabalha no espectro de frequência de 915 MHz, uma faixa licenciada no Brasil para aplicações de comunicação IoT. Este dispositivo possui potência de transmissão de 20 dBm e uma taxa de dados pelo ar padrão de 0.3 kbps a 19.2 kbps, tornando-o ideal para atuação de média distância e o uso em locais remotos. A Figura 13 mostra o dispositivos físico utilizado neste trabalho.

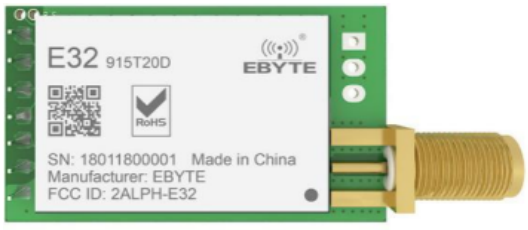

Figura 13 – Modulo transmissor LoRa E32-915T20D, figura do autor.

Este dispositivo opera de duas formas: transmissão fixa e *broadcasting*. Na transmissão fixa, ocorre a comunicação direta entre dois dispositivos, enquanto no *broadcasting* ocorre a comunicação entre vários dispositivos. A Figura 14 mostra um exemplo da transmissão fixa.

Neste modo de operação, configura-se o equipamento transmissor com endereço e canal semelhante ao receptor, neste caso somente o equipamento que esteja com o mesmo endereço e canal irá receber a informação. A transmissão fixa minimiza a interferência

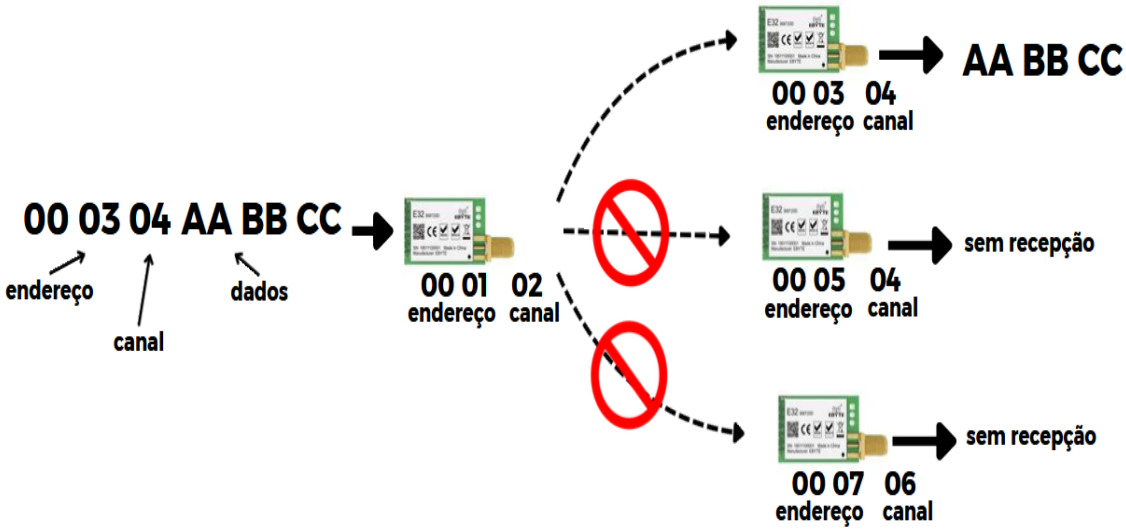

Figura 14 – LoRa E32-915T20D em transmissão fixa, adaptado de [39].

com outros dispositivos, garantindo que os dados sejam entregues de maneira confiável e eficiente. Este método é amplamente utilizado em aplicações onde um dispositivo precisa se comunicar exclusivamente com outro, sem a necessidade de uma rede mais complexa.

Na transmissão *broadcasting*, um dispositivo transmite e vários recebem a mensagem. A Figura 15 mostra um exemplo da transmissão *broadcasting*.

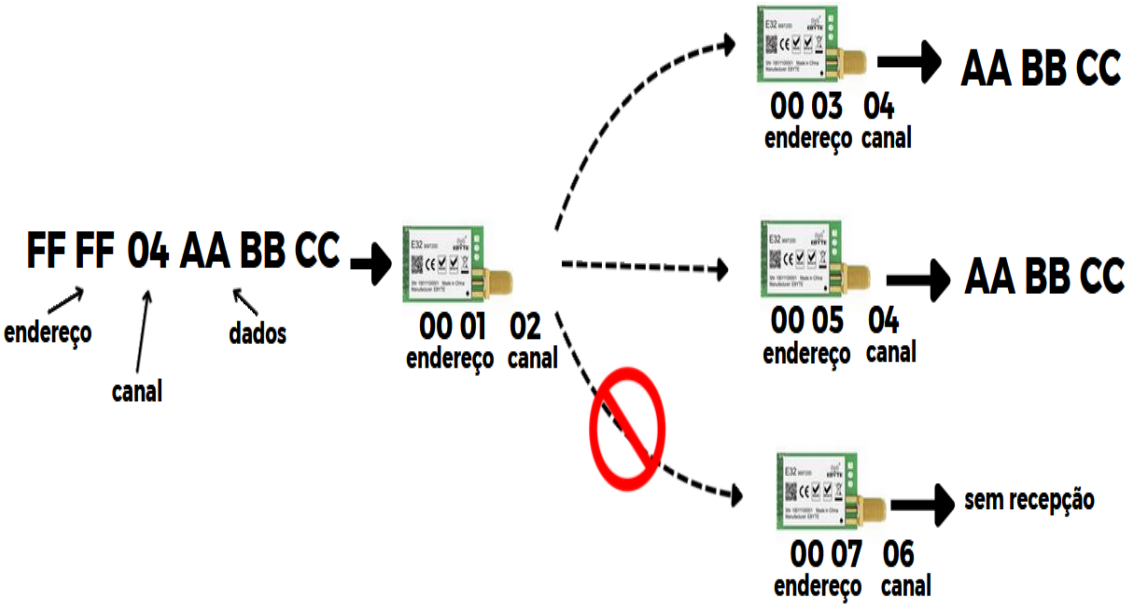

Figura 15 – LoRa E32-915T20D em transmissão *broadcasting*, adaptado de [39].

Na transmissão *broadcasting*, configura-se o equipamento transmissor com qualquer endereço e um canal, neste caso todos os equipamentos receptores que estejam com o mesmo canal do transmissor receberá as informações. Este modo é útil para enviar mensagens para múltiplos dispositivos simultaneamente, facilitando a disseminação de

informações para uma rede maior de dispositivos. A transmissão *broadcasting* é especialmente benéfica em aplicações que requerem atualizações simultâneas ou notificações para todos os dispositivos dentro do alcance, aumentando a eficiência e a abrangência da comunicação.

3.2.2 Transmissor LPWA NBIoT

O módulo Telit ML85G1WW opera com conectividade LPWA (do inglês *Low Power Wide Area*) NBIoT projetado para aplicações de baixa latência e alto rendimento, com foco em segurança e flexibilidade. Ideal para aplicações *machine to machine*, possui uma faixa de alimentação de 2.1 V a 3.4 V. Suas interfaces de comunicação incluem UART, USB (do inglês *Universal Serial Bus*), SPI (do inglês *Serial Peripheral Interface*) e I2C (do inglês *Inter Integrated Circuit*) e baixo consumo de energia. O módulo Telit ML85G1WW trabalha com comandos AT [48], que facilitam a comunicação entre o dispositivo, estação base e servidores de rede, direcionando os dados aos servidores de aplicação.

A Figura 16 mostra um exemplo de vários dispositivos finais de IoT com transmissores NBIoT conectados à estação base de celular, aos servidores de rede e aos servidores de aplicação.

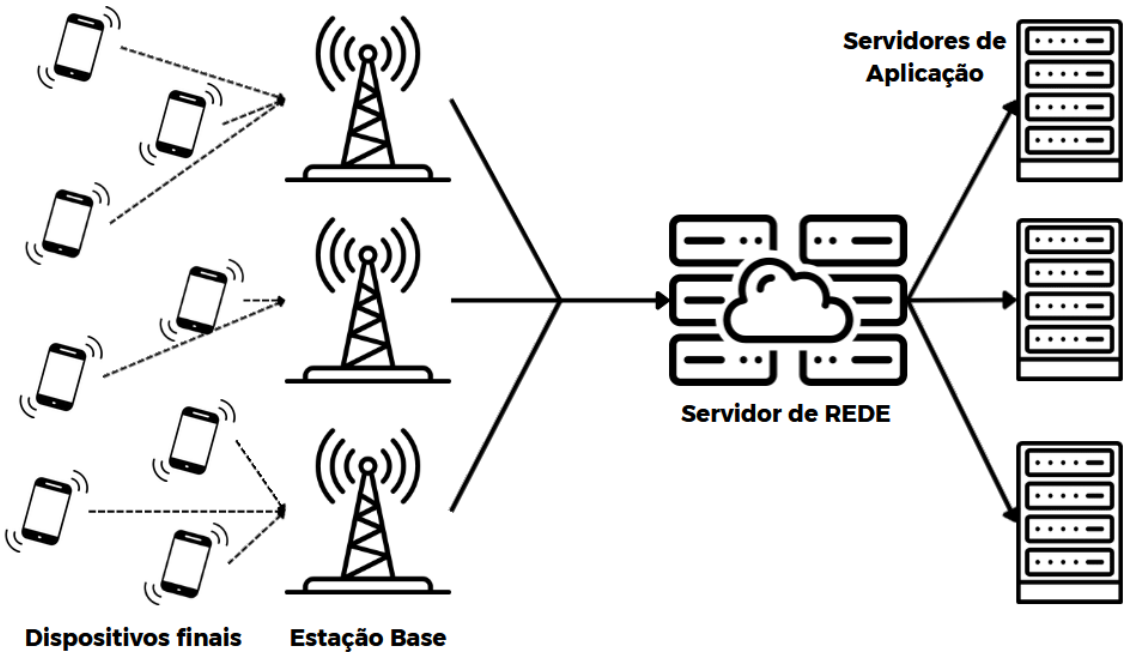

Figura 16 – Arquitetura de uma rede NBIoT, adaptado de [37].

Para o dispositivo Telit, foi utilizado um modem da empresa *CCK* Automação que atuou como um transmissor UART *wireless*, de forma similar ao dispositivo LoRa. O modem possui como núcleo principal de comunicação o módulo multibanda Telit ML85G1WW. Na Figura 17 é apresentado o modem NBIoT.

Figura 17 – MODEM NBIoT, figura do autor.

O modem NBIoT possui entrada para *Simcard* e conector *SMA* para conectar uma antena externa, além de comunicação serial UART. Ele suporta múltiplas bandas de frequência, garantindo ampla cobertura e compatibilidade com diversas operadoras de telecomunicações. O design do módulo facilita a comunicação com o ESP32-CAM através do barramento de comunicação, permitindo uma integração eficiente e versátil à aplicação.

3.3 A montagem dos transmissores e receptores

Os transmissores foram montados para realizar a transmissão dos dados reconhecidos pela aplicação principal. Toda a comunicação entre o dispositivo principal e os transmissores foi feita utilizando o protocolo serial UART. O primeiro transmissor desenvolvido utilizou o modem Telit ML85G1WW, a Figura 18 mostra o modem com o dispositivo principal de reconhecimento de imagem montados.

O segundo transmissor desenvolvido utilizou o módulo LoRa E32915T20D pra realizar a transmissão dos dados reconhecidos pela aplicação principal. Toda comunicação entre o dispositivo principal e os transmissores também foi feita utilizando o protocolo serial UART.

O funcionamento do modem NBIoT não necessita de um receptor separado, uma vez que a aplicação LTE realiza a conexão com a estação base de telefonia, que atua como receptor. Todo o trajeto de mensagens é descrito na Seção 3.1.3. Diferentemente do NBIoT, o dispositivo LoRa foi implementado de maneira transparente, ou seja, são necessários dois módulos, um transmissor e um receptor. A mensagem é transmitida e recebida, para só então ser processada e enviada a aplicação final.

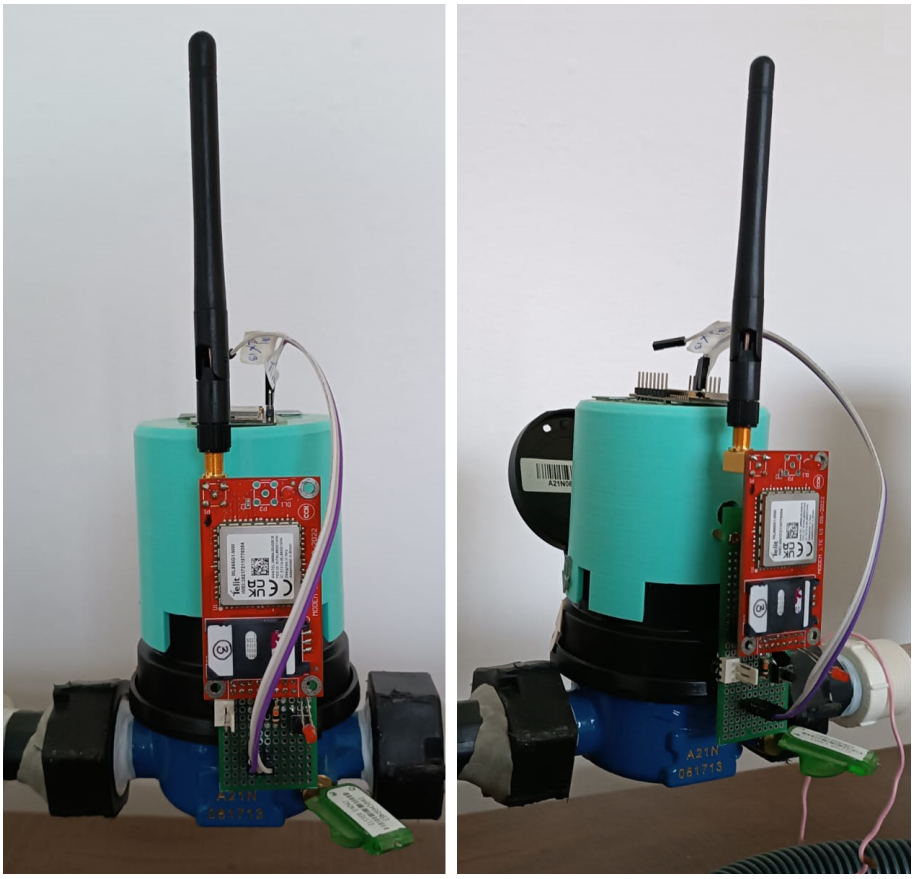

Figura 18 – Circuito MODEM NBIoT montado na bancada de testes, figura do autor.

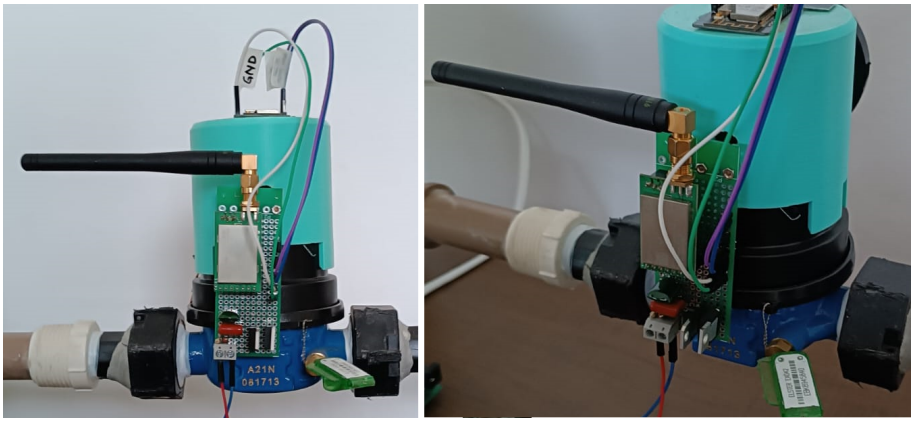

Figura 19 – Circuito LoRa, montado na bancada de testes, figura do autor.

O receptor foi conectado a um conversor serial TTL (do inglês *Transistor-Transistor Logic*) para transmitir os dados a uma aplicação local. A Figura 20 ilustra a montagem do receptor conectado ao conversor serial TTL. O conversor serial TTL foi escolhido devido à sua compatibilidade com os dispositivos de comunicação e sua capacidade de converter sinais de níveis lógicos em sinais seriais compatíveis.

A aplicação responsável por interpretar os dados recebidos das transmissões foi desenvolvida utilizando a ferramenta *Node-RED* e está armazenada em um servidor local. Essa aplicação possui três funcionalidades principais: recepção dos dados, validação das

Figura 20 – Receptor LoRa, figura do autor.

informações e armazenamento. A Figura 21 apresenta o diagrama da aplicação:

Na recepção de dados, existem dois meios disponíveis: um *socket*, que possui a funcionalidade de escutar todas as recepções provenientes da estação base das mensagens NBIoT, e uma porta serial, onde está conectado o receptor LoRa. Ambas as mensagens recebidas são armazenadas em um vetor que identifica o início e o fim da mensagem. Separadamente, cada valor é testado e deve estar entre 0 e 9; qualquer valor fora dessa faixa é classificado como erro, sendo descartado e contabilizado. Após essa etapa, os valores são agrupados para formar um número inteiro que representa o consumo instantâneo do hidrômetro. Para fins estatísticos, os valores corretos também são contabilizados. Por fim, os valores são armazenados em dois bancos de dados distintos. Dessa forma, é possível avaliar o consumo de água, a precisão dos valores reconhecidos e a eficiência da transmissão.

3.4 Apresentação e configuração da aplicação de processamento de imagem

O *firmware* utilizado no processamento de borda foi desenvolvido por [49]. Este *firmware* possui licença GNU, o que permite a utilização e alteração do *firmware* conforme a necessidade da aplicação. Sob estas condições, foi desenvolvida uma versão com funcionalidades adaptadas para este projeto. Nesta versão foram feitas mudanças no *layout* das paginas WEB (do inglês *World Wide Web*), criação de um modelo de aprendizagem personalizado e adição de duas tecnologias de comunicação, LoRa e NBIoT. O funcionamento

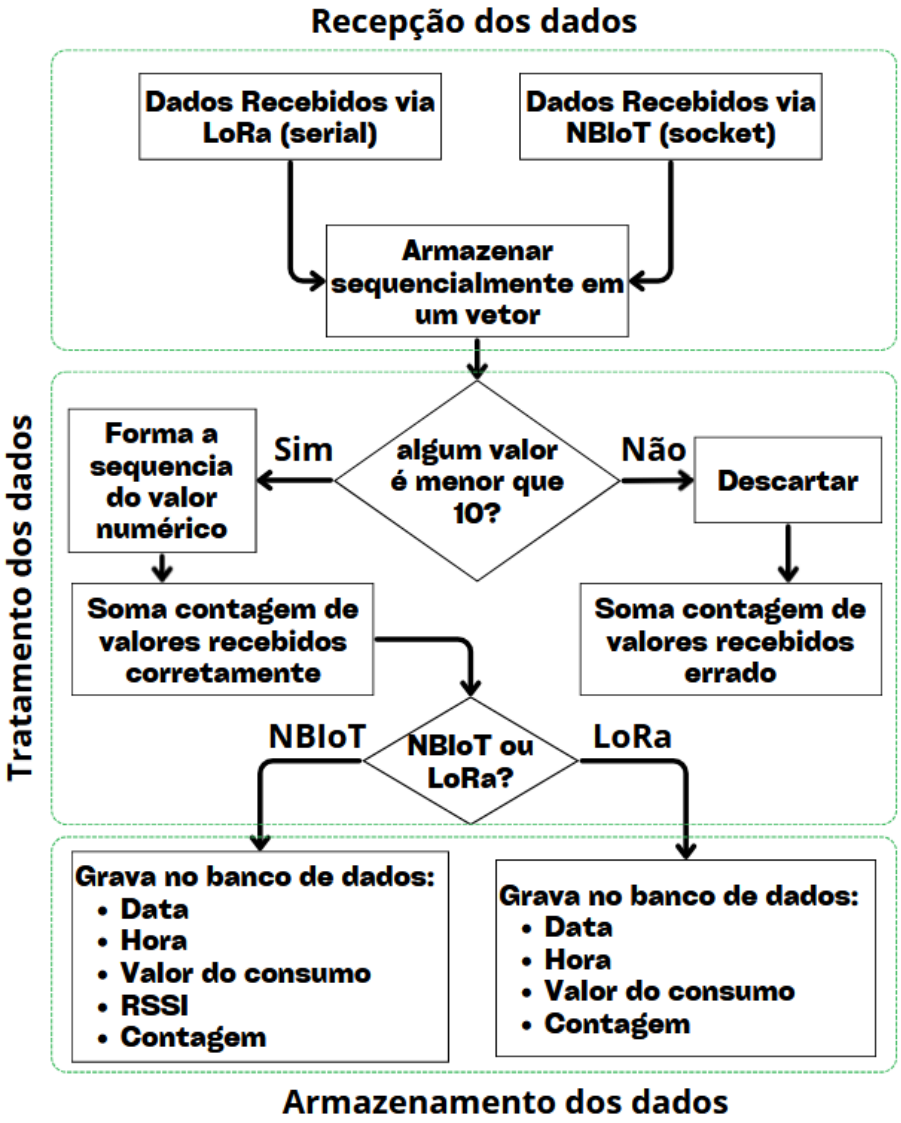

Figura 21 – Diagrama da aplicação de recebimento dos dados, figura do autor.

do *firmware* é apresentado no diagrama da Figura 22.

O diagrama ilustra o processo de funcionamento do *firmware*: Inicialmente a câmera é iniciada de maneira a garantir o funcionamento adequado do equipamento. Uma imagem é capturada e comparada com o máximo tamanho em bytes suportado pelo equipamento. Caso haja alguma inconsistência, uma rotina de alerta é iniciada, com indicação luminosa via Led do equipamento, neste caso é necessário a reinicialização manual do equipamento. Em seguida, é executada uma rotina de verificação das informações contidas no *SD-CARD*, caso esteja correto, são extraídas informações que compõem a conexão de rede SSID (do inglês *Service Set Identifier*), *senha*, *hostname*, *gateway* e *netmask* para iniciar a rotina de conexão *Wifi*, caso contrário a aplicação é interrompida e o estado de alerta é acionado.

As rotinas de servidor WEB trabalha em paralelo, sincronizado ciclicamente ao serviço de captura, tratamento e reconhecimento de imagens de acordo com um tempo

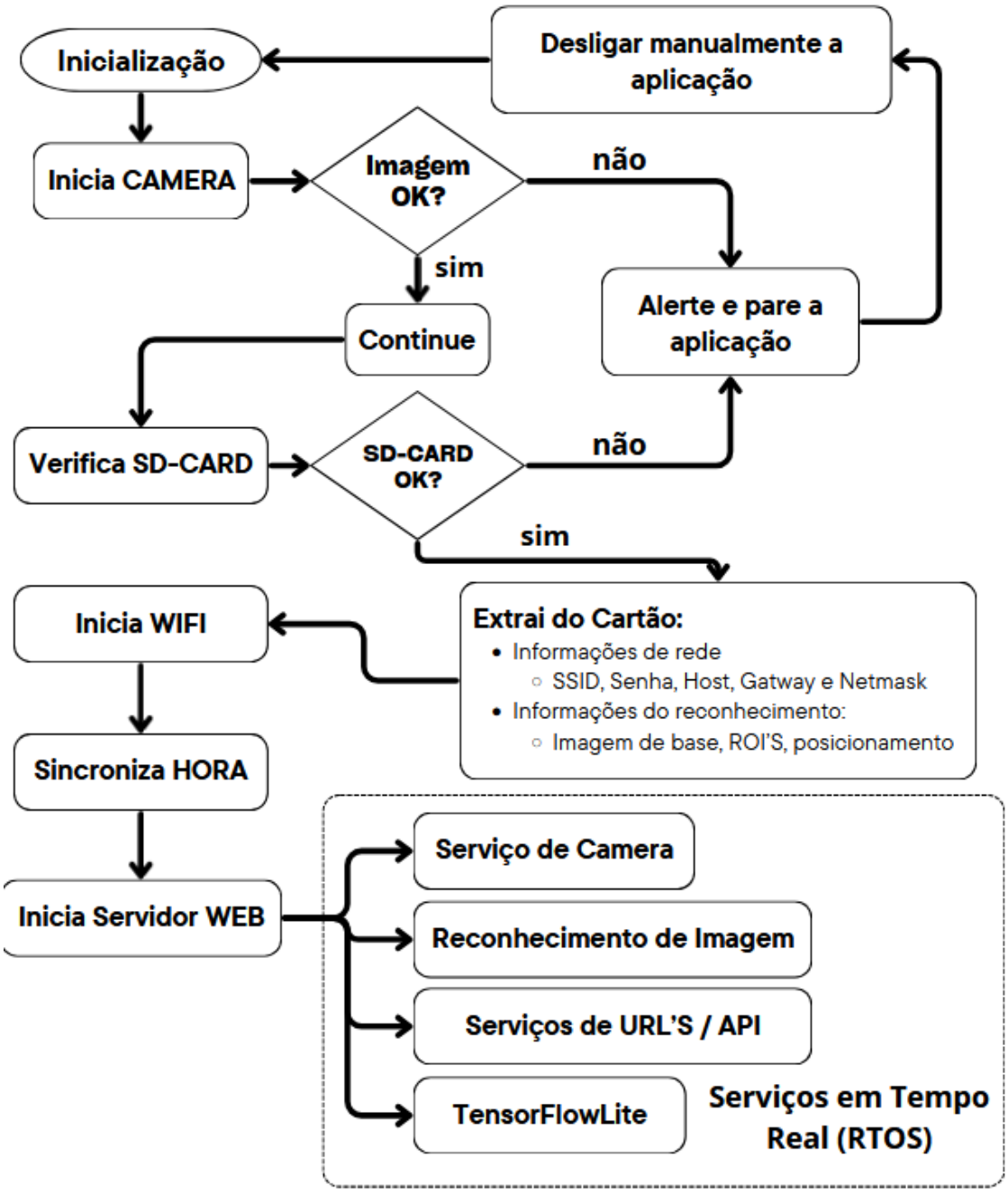

Figura 22 – Diagrama de funcionamento do *firmware*, figura do autor.

programado. Na Figura 22 a área demarcado por linhas pontilhadas, representa a etapa de serviço de câmera: rotinas que tiram as fotos; reconhecimento de imagem e *tensorFlowLite* trabalham em conjunto para reconhecer os algorítimos e os serviços de URL (do inglês *Uniform Resource Locator*)'S/*API* são usados para configurar o reconhecimento de imagem.

3.4.1 Rede Neural criada para o reconhecimento de imagens

Neste trabalho, foi desenvolvida uma RNA utilizando a ferramenta *Google Colaboratory* e a biblioteca de IA *TensorFlowLite*, com programação em *Python*. A base de dados foi composta por 1071 amostras de imagens, conforme ilustrado no histograma da Figura 23. Nesse gráfico, as categorias de 1 a 10 representam as imagens contendo os números de 1 a 9, sendo que a categoria 10 refere-se a ruídos, isto é, imagens capturadas durante a transição entre um número e outro. O eixo vertical do histograma exibe a quantidade de imagens em cada categoria. Cada imagem foi devidamente categorizada e será utilizada pela RNA no processo de reconhecimento.

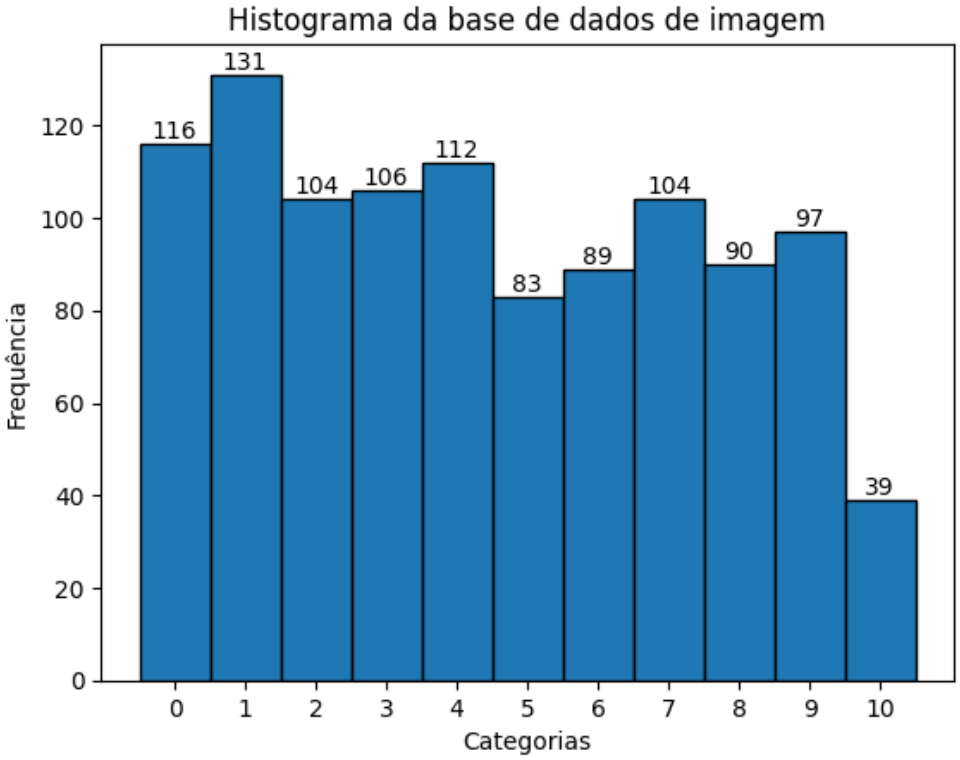

Figura 23 – Historiograma da base de dados, figura do autor.

Nesta base de dados, as imagens possuem tamanhos variados, devido a coleta aleatória de diferentes fontes. Foi necessário padronizar o tamanho das imagens para que fossem compatíveis com as processadas pelo dispositivo. Portanto, foi adotado um padrão em que todas as imagens foram dimensionadas para 32x20x3 *pixels*, conforme é mostrado na Figura 24. A escolha deste valor foi determinado após verificar que a redução não descaracterizava a imagem, reduzia 1440 vezes o seu tamanho em bytes e auxiliava no processamento da rede neural, resultando em uma melhor identificação dos dígitos.

Com a base de dados formada, foi criada a primeira camada da rede neural baseada no tamanho da imagem. Segundo [30], este é um método padrão para o reconhecimento de imagem. Portanto, a primeira camada, chamada de camada de entrada, espera receber sempre $x_i \in \mathbb{R}^{32\times20\times3}$, esta notação é usada para indicar que o vetor x_i tem dimensões

Figura 24 – Amostras de valores utilizados no treinamento da Rede Neural, figura do autor.

$32 \times 20 \times 3$ e cada elemento deste tensor pertence ao conjunto dos reais $\mathbb{R}$. Em seguida, a imagem passa por 6 camadas de convolução. Estas camadas foram construídas com convolução 2D, 32 *kernels* de 3x3, uma camada de agrupamento *Pooling* e função de ativação *ReLu*.

Complementando a arquitetura da rede neural, após a camada de convolução, há uma camada de *Max Pooling* com passos 2x2, uma camada de *Flatten* que realiza a verticalização da imagem, uma camada oculta com 128 neurônios e função de ativação *ReLu*, e uma camada final com 11 neurônios e função de ativação *softmax*. Ao todo, foram criados 53.719 pesos que disponibilizam uma saída de valor unitário da rede neural, de valores reconhecidos de 0 a 9, com o valor 10 indicando um erro de não reconhecimento.

O treinamento realizado com estes parâmetros foi de 300 épocas, resultando em uma precisão do modelo em de 0,2152 e acurácia de 93,52%. A Figura 25 mostra o gráfico dos resultados do treinamento por época da arquitetura mencionada anteriormente. Neste gráfico, é possível observar a evolução da acurácia e a redução do erro ao longo das épocas, demonstrando a eficiência do modelo desenvolvido.

Nesta imagem é possível observar que a margem de erro nos primeiros treinamentos ultrapassam os 200%, mas, a medida que as épocas avançam, o percentual reduz exponencialmente, estabilizando a partir de 200 épocas. O mesmo ocorre com a acurácia a Figura 26 mostra a acurácia do modelo desenvolvido

Nas primeiras inferências a margem de acerto é próxima de zero e, conforme as épocas são calculadas, a taxa de acerto se estabiliza, aproximando-se de uma margem

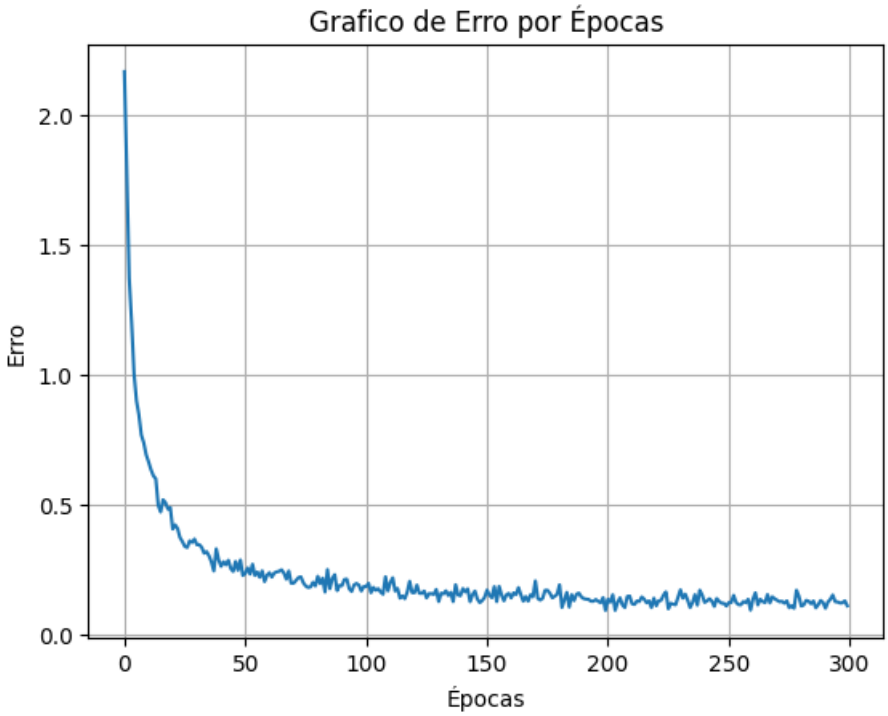

Figura 25 – Relação de Erros por Épocas obtido com treinamento de 300 épocas, figura do autor.

constante por volta das 200 épocas.

Ao final, todo o trabalho foi convertido em um modelo *.tflite*. Esse modelo contém uma estrutura binária com toda a arquitetura da rede e os pesos da aprendizagem, que serão incorporados ao *firmware* de processamento de borda para realizar o reconhecimento das imagens capturadas pelo dispositivo.

3.4.2 Rede Neural integrada ao *firmware*

Durante o processo de inicialização do *firmware*, uma tarefa específica da biblioteca *TensorFlowLite* para microcontroladores é dedicada exclusivamente ao reconhecimento de imagens. A Figura 27 apresenta um diagrama que detalha as etapas envolvidas nessa tarefa.

O modelo é uma estrutura fundamental do projeto, ele contém informações sobre a rede neural elaborada, desde a sua estrutura básica, como a quantidade de neurônios, os pesos utilizados, o tipo de rede neural, as camadas de convolução, os filtros, o *kernel*, a quantidade de entradas e saídas [50].

Provindas dessas informações, um objeto interpretador no *firmware* replica a rede neural criada para realizar o reconhecimento dos valores numéricos de acordo com o modelo, deixando pré-definidas todas as camadas de convolução, filtros, *kernel*, pesos, entradas e saídas. Portanto, é importante que os dados a serem reconhecidos contenham

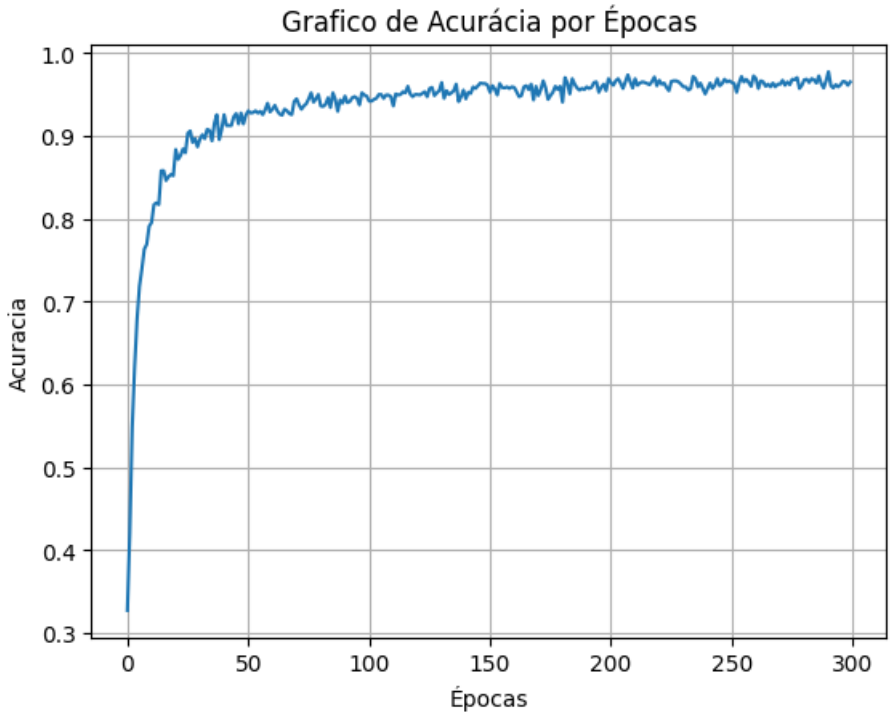

Figura 26 – Relação de Acurácia por Épocas obtido com treinamento de 300 épocas, figura do autor.

exatamente a mesma entrada. Em seguida, foi realizado um processo de tratamento da imagem capturada, no qual foram recortadas todas as ROI (*Region of interest*). Esse processo transforma uma imagem de 720x1023x3 *pixels* em 7 imagens de 32x20x3 *pixels* contendo apenas os números a serem reconhecidos. Por fim, uma instrução de *Flatten* é aplicada à matriz de imagem, verticalizando os valores.

Com as imagens formatadas, o interpretador realiza a tarefa de reconhecimento e obtém a saída esperada, um valor numérico de 0 a 9. Porém, imprevistos, como a má qualidade da imagem, podem afetar o reconhecimento, que causa um não entendimento do interpretador, e consequentemente gera um valor de erro como resultado.

Neste trabalho, o erro é representado pelo valor "10"e neste caso o valor é descartado, que significava a não interpretação da imagem reconhecida. Esse processo se repete para a quantidade de ROI definidos na configuração. Ao finalizar o reconhecimento, os valores reconhecidos são mostrados nas páginas do WEB *server* e enviados para a tarefa de transmissão de dados. Por padrão, o projeto realiza a transmissão via *Wifi*.

3.4.3 Configuração do reconhecimento de imagem via servidor WEB

Para configurar o reconhecimento de imagem, foram necessários quatro etapas: a preparação do cartão de memória, criação de uma imagem de referência, alinhamento da imagem de referência e marcação dos ROI.

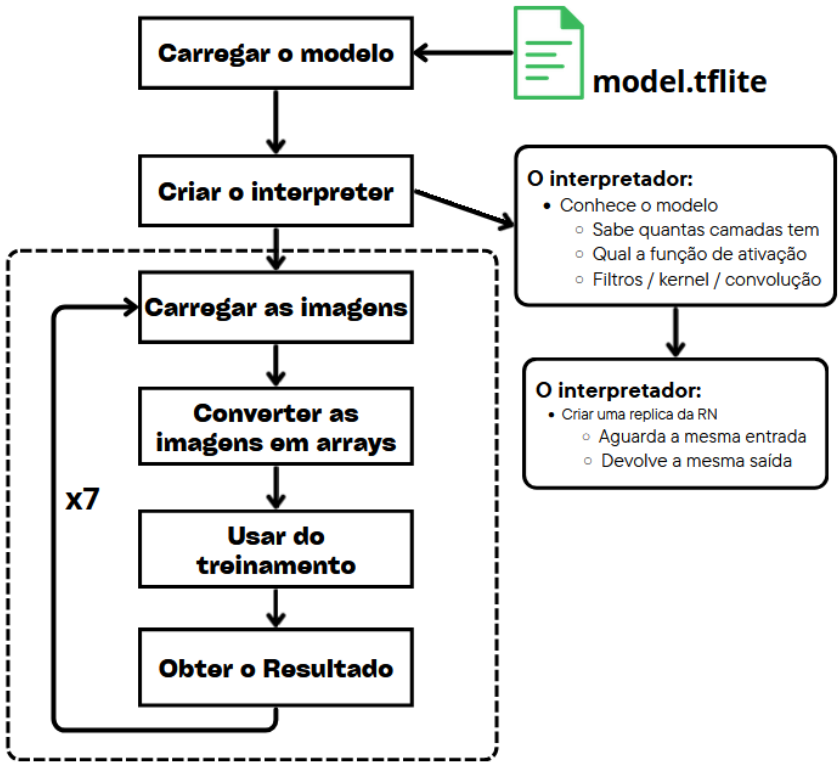

Figura 27 – Diagrama das tarefas de inteligencia artificial para o reconhecimento de imagem, figura do autor.

No projeto oficial [49], há instruções de configuração do reconhecimento de imagem que requerem uma estrutura específica no cartão de memória. Portanto, antes de iniciar a aplicação, foi necessário realizar a preparação do cartão de memória com a estrutura apresentado na Figura 28. Essa organização permite que o dispositivo principal gerencie o armazenamento de informações de forma adequada.

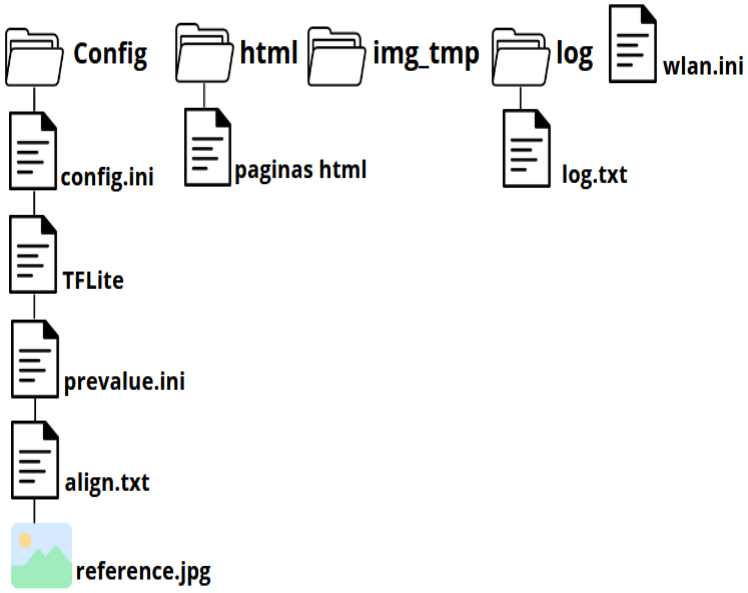

Figura 28 – Estrutura de organização dos diretórios do projeto principal, figura do autor.

A estrutura do cartão de memória segue uma hierarquia bem definida, conforme descrito abaixo:

- **Arquivo wlan.ini**: Contém SSID, senha, *hostname*, *gateway* e *netmask* da rede local, usados para iniciar o servidor WEB.
- **Diretório Config**: Contém parâmetros de inicialização da imagem, como posição e tamanho da imagem
- **Arquivo TFLite**: Arquivo do modelo gerado pelo treinamento da CNN.
- **Arquivo prevalue.ini**: Armazena a configuração dos valores já reconhecidos.
- **Arquivo align.txt**: Contém os valores das posições usadas para cortar as imagens no reconhecimento.
- **Arquivo reference.jpg**: Imagem de referência usada.
- **Diretório HTML**: Contém as páginas do servidor WEB.
- **Diretório img_tmp**: Usado pelo *firmware* para armazenar imagens temporárias e evitar gastos de memória RAM durante o processamento.
- **Diretório log**: Contém o arquivo log.txt, utilizado como registro de erros.
- **Configuração na interface WEB**: Todos esses parâmetros (exceto os de rede) são configurados com o dispositivo ligado e conectado à rede local.

Ao iniciar o dispositivo de reconhecimento de imagem (ESP32-CAM), foi possível acessar o servidor WEB através de um *browser* local, por meio do endereço de *IP* configurado previamente no cartão *SD*, para realizar a configuração das imagens que servira de base para o reconhecimento dos dígitos. A Figura 29 mostra a página principal de reconhecimento de imagem com layout adaptado.

Nesta figura é apresentada a página principal do servidor WEB para o reconhecimento dos dígitos. Na figura, é mostrada uma foto do hidrômetro que foi utilizado nos testes para coleta de dados. Os valores selecionados para serem reconhecidos pela CNN estão contornados com bordas azuis, enquanto as bordas vermelhas, têm a função de alinhar a imagem. No lado direito da Figura, são apresentados *Previous Value* (valor prévio), que é utilizado como base do consumo total, o *Raw Value* (valor reconhecido atualmente) e o cálculo do erro em "Error", que mostra a diferença entre o *Previous Value*.

Para iniciar a configuração, foi necessário acessar "Configuração"no menu suspenso da Figura 29. A primeira configuração realizada foi a imagem de referência, que tem como objetivo servir de base na seleção dos dígitos. A Figura 30 mostra a página de configuração da imagem de referência do equipamento.

Esta configuração é realizada por meio do clique no botão 'Criar nova referência'. Neste momento, o equipamento captura uma imagem do hidrômetro para verificar se o

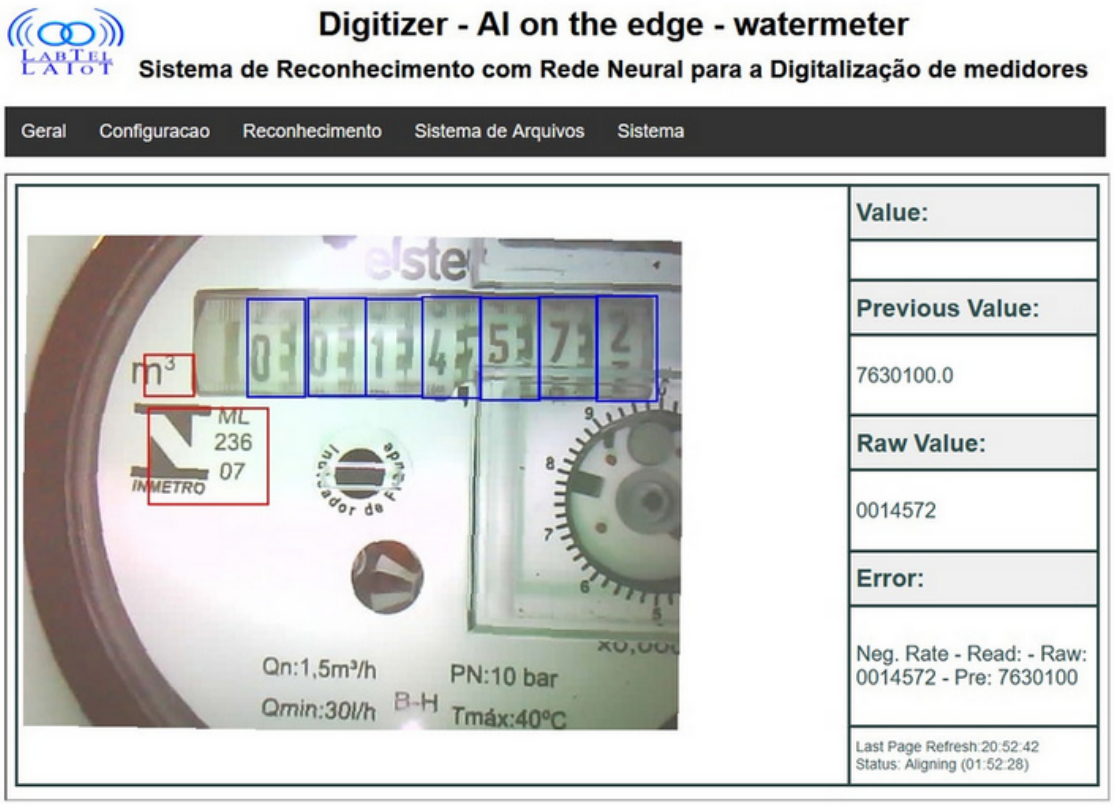

Figura 29 – Página principal do servidor WEB, figura do autor.

alinhamento e a qualidade da imagem estão adequados. Esta é uma etapa importante para obter uma boa classificação dos dígitos. Nesta página, existem ferramentas para aprimorar a qualidade da imagem, disponíveis acima da foto do hidrômetro.

A configuração dos ROI's definem a seleção dos dígitos a serem reconhecidos pela CNN. Internamente, o *firmware* usa a estrutura de alinhamento na imagem de referência para cortar a região de interesse e obter imagens menores apenas com os dígitos. Segundo [49], isto facilita o reconhecimento e diminui a complexidade da arquitetura da rede neural para reconhecer o valor. A Figura 31 mostra as regiões de interesse selecionadas para serem reconhecidas.

Durante esta etapa, foram ajustadas manualmente as bordas das ROI's para assegurar que todos os dígitos sejam incluídos corretamente e que nenhuma parte irrelevante da imagem interfira no reconhecimento.

Com a configuração completa, o sistema de reconhecimento de imagem está preparado para operar de acordo com [49]. A estrutura específica do cartão de memória e as etapas detalhadas de configuração, como a criação da imagem de referência e a definição das regiões de interesse, garantem que o dispositivo possa gerenciar e processar as informações corretamente.

Através do acesso ao servidor WEB, é possível monitorar e ajustar as configurações

Figura 30 – Página de configuração da imagem de referência, figura do autor.

conforme necessário. Essa abordagem não só facilita o reconhecimento dos dígitos, mas também reduz a complexidade da arquitetura da rede neural, permitindo um desempenho otimizado do sistema. Com estas preparações, o dispositivo está pronto para realizar medições.

3.5 Adaptações no *firmware* da aplicação principal

As adaptações do *firmware* são uma extensão da Figura 22 onde foram incluídas rotinas de transmissão sincronizadas ao reconhecimento de imagem, e incluídas novas comunicações como o NBIoT e LoRa.

3.5.1 Inclusão de rotina de envio via NBIoT

Sincronizadas as tarefas de reconhecimento de imagem e *TensorFlow Lite*, uma nova tarefa de transmissão é criada. O diagrama da Figura 32 mostra as etapas dessa tarefa.

A tarefa chamada *xTelit* inicia com a configuração da saída serial do dispositivo principal em 115200 bps 8N1. Em seguida, é chamada a rotina de comandos AT (do inglês

Figura 31 – Página de configuração do alinhamento na imagem de referência com as regiões de interesse selecionadas para serem reconhecidas, figura do autor.

Attention Commands). O manual [48] descreve a sequência de comandos e respostas que devem ser seguidos para estabelecer uma comunicação estável com a estação base da operadora. No Apêndice A deste trabalho, são apresentados a sequência de comandos e as respostas esperadas para que a comunicação seja estabelecida. Com a comunicação estável, um envio de teste é realizado ao servidor com a mensagem "OK", aguardando que o servidor retorne o valor 200 no cabeçalho HTTP (do inglês *Hypertext Transfer Protocol*). Caso isso não ocorra, é necessário reiniciar os comandos AT.

Uma vez estabelecida a comunicação, a tarefa aguarda que a fila *xTelitQueue* seja preenchida pela tarefa do *TensorFlow Lite* com os valores do reconhecimento para transmissão. Enquanto não há dados nesta fila, a cada 1 segundo verifica-se a qualidade do sinal com o comando "CSQ". Geralmente, a intensidade do sinal NBIoT varia de -50 dBm (ótimo sinal) a -120 dBm (sinal extremamente ruim). Caso a comunicação esteja nos limites de -120 dBm uma rotina de interrupção acontece e o modem é reiniciado, reestabelecendo os comandos AT.

Uma vez que há valores na fila de transmissão e o sinal é estável para comunicação, os dados são transmitidos um a um para o servidor. Por fim, a fila é limpa e o processo

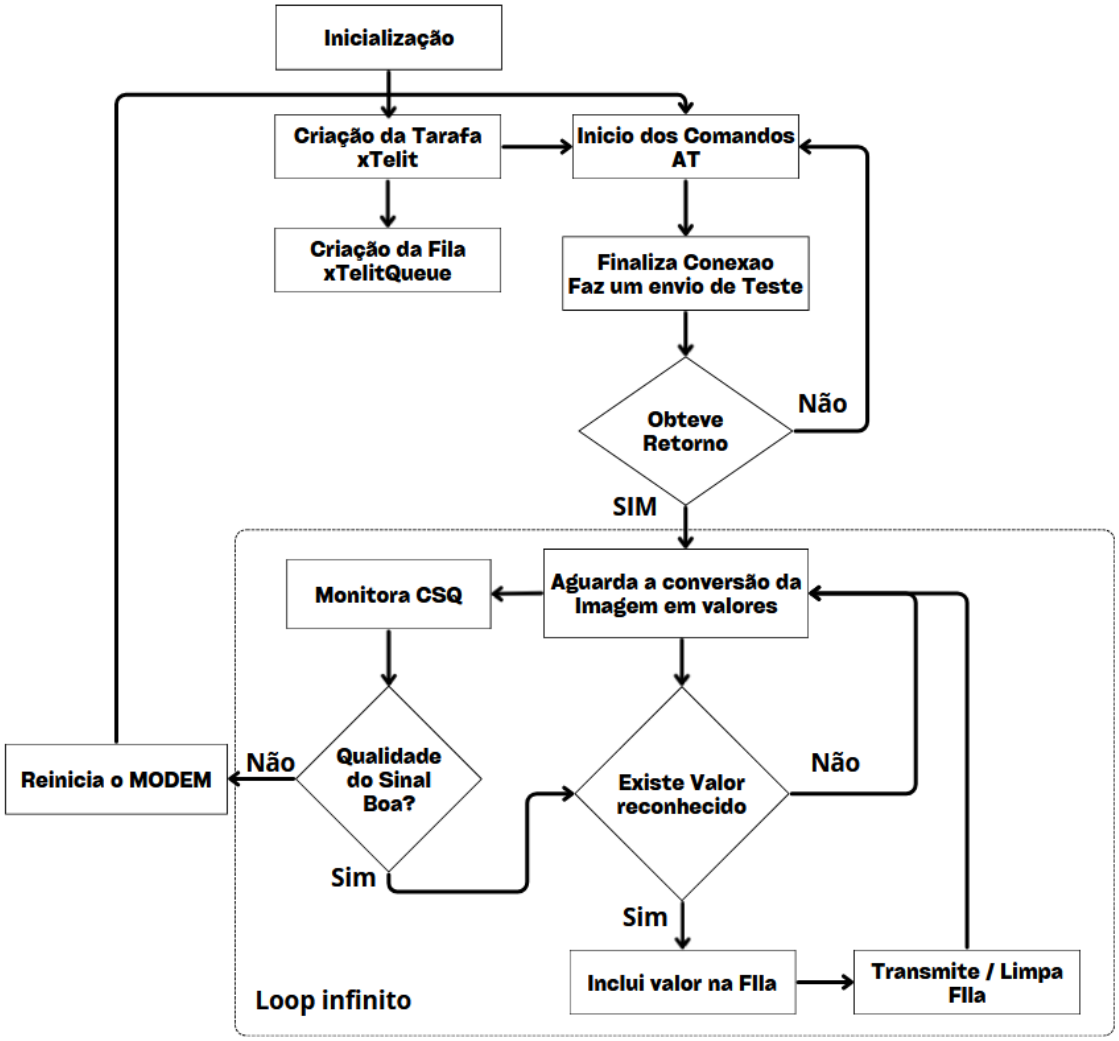

Figura 32 – Diagrama das tarefas para transmissão via Modem NBIoT, figura do autor.

se repete, aguardando a chegada de novos dados.

3.5.2 Inclusão de rotina de envio via LoRa

Sincronizadas as tarefas de reconhecimento de imagem e *TensorflowLite*, uma nova tarefa de transmissão é criada, o diagrama da Figura 33 mostra as etapas desta tarefa.

A inicialização do módulo *xLora* começa configurando a taxa de transmissão do dispositivo. Neste trabalho, foi utilizada a velocidade de 9600 bps 8N1 e uma taxa de transmissão aérea de 0.3 kbps. Conforme especificado no *datasheet*, uma taxa de *air rate* mais baixa permite transmissões de longo alcance e melhor desempenho anti-interferência. Essa configuração permite a transmissão de até 58 bytes. Em seguida, a potência do transmissor é configurada para a potência máxima de 20 dBm, conforme indicado no *datasheet*. A opção de FEC (do inglês *Forward Error Correction*) do dispositivo é ativada devido à sua capacidade anti-interferência.

O modo de transmissão é configurado como *broadcast*, com o canal fixo em 15.

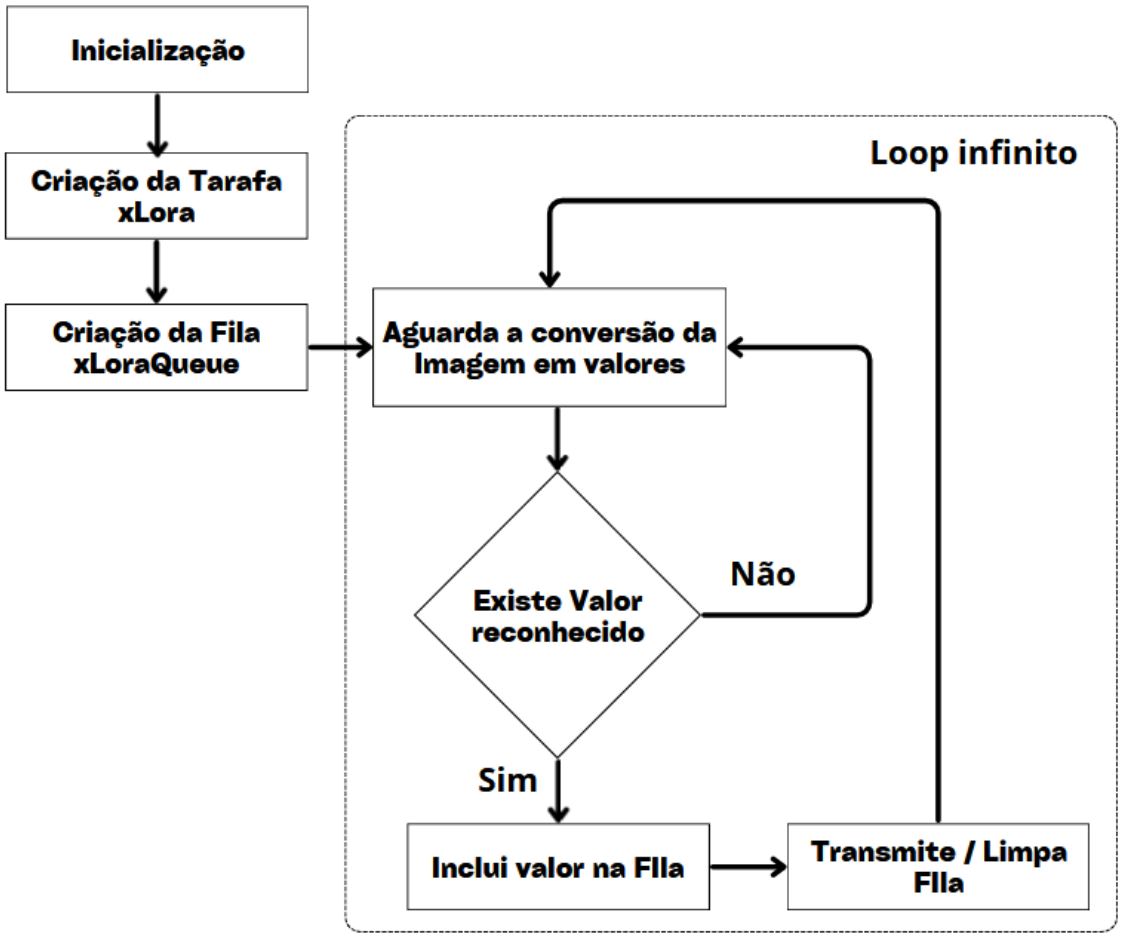

Figura 33 – Diagrama das tarefas para transmissão via LoRa, figura do autor.

Dessa forma, qualquer dispositivo na mesma faixa de frequência de 915 MHz com o canal 15 está apto a receber os dados transmitidos por este dispositivo.

Para concluir a inicialização do dispositivo, a tarefa chamada *xLora* configura a saída serial do dispositivo principal em 9600 bps 8N1. Em seguida, é criada uma fila para aguardar a recepção de dados da tarefa do *TensorFlow Lite*. Quando um valor é reconhecido, ele é inserido na fila de transmissão do *LoRa*, que é verificada pela tarefa e transmitida ao receptor.

3.6 Montagem do projeto final em uma bancada de teste

A bancada de testes foi elaborada com intuito de validar o funcionamento da aplicação com as tecnologias de comunicação e gerar dados para análise e resultados. Trata-se de uma simulação de consumo de água residencial com um hidrômetro 3/4"do fabricante Elster. A Figura 34 mostra a bancada de teste

A bancada é composta por dois reservatório de água de 7 litros cada (1), conectadas por tubos de PVC de 1' (2), entre os reservatórios foi instalado um hidrometro de 3/4"

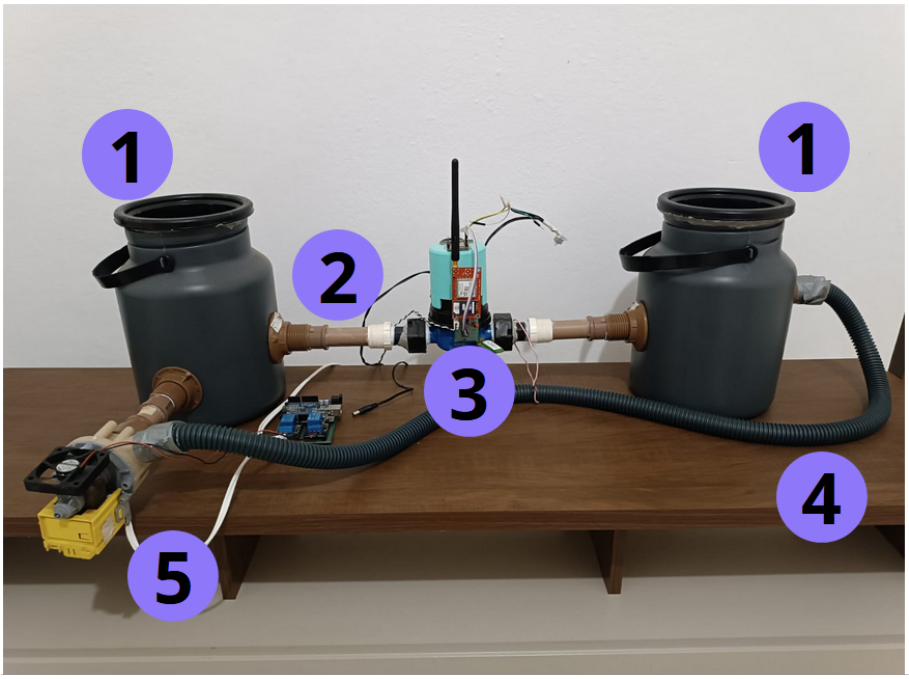

Figura 34 – Bancada de testes, figura do autor.

do fabricante Elster (3) que contabiliza por meios mecânicos o consumo de água, uma mangueira realiza o retorno ao reservatório (4), forçado por uma bomba hidráulica modelo inter de 12 W de potência (5).

Para que o consumo fosse semelhante ao consumo de uma residência, foi adicionado um controlador que gerencia o acionamento da bomba hidráulica. A bomba realiza um ciclo aonde permanece desligada por 10 minutos e ligada durante 5 minutos, desta forma há um fluxo de água pelos reservatórios que é contabilizado pelo hidrômetro. Para acoplar o hidrômetro ao dispositivo de processamento de borda foi elaborado um *case* como mostrado na Figura 35.

Figura 35 – *Case* para acoplar o dispositivo de processamento de borda ao hidrometro, figura do autor.

Este dispositivo tem como objetivo auxiliar a captura de imagens, ajustar o posicionamento da câmera, dispositivo de comunicação e impedir que não haja interferência

nas fotos.

4 Resultados e Analises

Este Capítulo aborda os resultados obtidos com experimentos realizados na bancada de testes do dispositivo de processamento de borda integrado às tecnologias IoT. São apresentados os dados transmitidos em gráficos de consumo, a análise do erro médio absoluto dos dados recebidos em comparação com o valor esperado e a análise da qualidade do sinal durante as transmissões.

Além disso, é discutida a relação de perda dos valores transmitidos pelas duas tecnologias, a eficácia do treinamento comparado ao resultado previsto do treinamento da rede neural e a análise de consumo do dispositivo, com o objetivo de dimensionar um banco de baterias para a aplicação.

4.1 Análise dos resultados obtidos dos dados transmitidos via NBIoT

A transmissão de dados foi realizada na cidade de Santa Rita do Sapucaí - Minas Gerais, nos pontos latitudinal e longitudinal (-22.23568514176665, -45.70711147011611) até a estação base da operadora TIM, situada em orientações latitudinal e longitudinal (-22.243972871534194, -45.69833447028311) da mesma cidade, a distância ponta a ponta da aplicação foi de 1,3 km conforme é apresentado na Figura 36

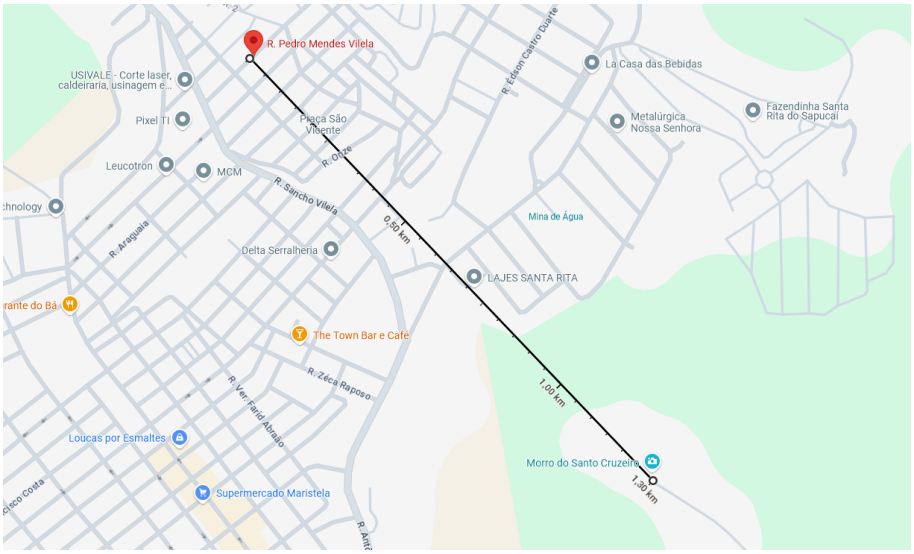

Figura 36 – Distância da transmissão de dados via NBIoT, figura do autor

Entre os dias 13/04/24 e 14/04/24, foram realizadas 387 transmissões da bancada de testes com o dispositivo de reconhecimento de imagem conectado ao transmissor de dados com tecnologia NBIoT. Comumente, dispositivos com inteligência embarcada, diferente de dispositivos de telemetria, costumam realizar cerca de uma ou duas transmissões

diárias, a fim de economizar bateria e evitar redundância de informação. No entanto, para fins de análise de transmissão em um curto período de tempo, o dispositivo utilizado neste trabalho foi programado para realizar a transmissão dos dados de consumo de água em um intervalo de 5 minutos. Na Figura 37 é apresentado os valores transmitidos de consumo de água, registrado pelo hidrômetro neste intervalo.

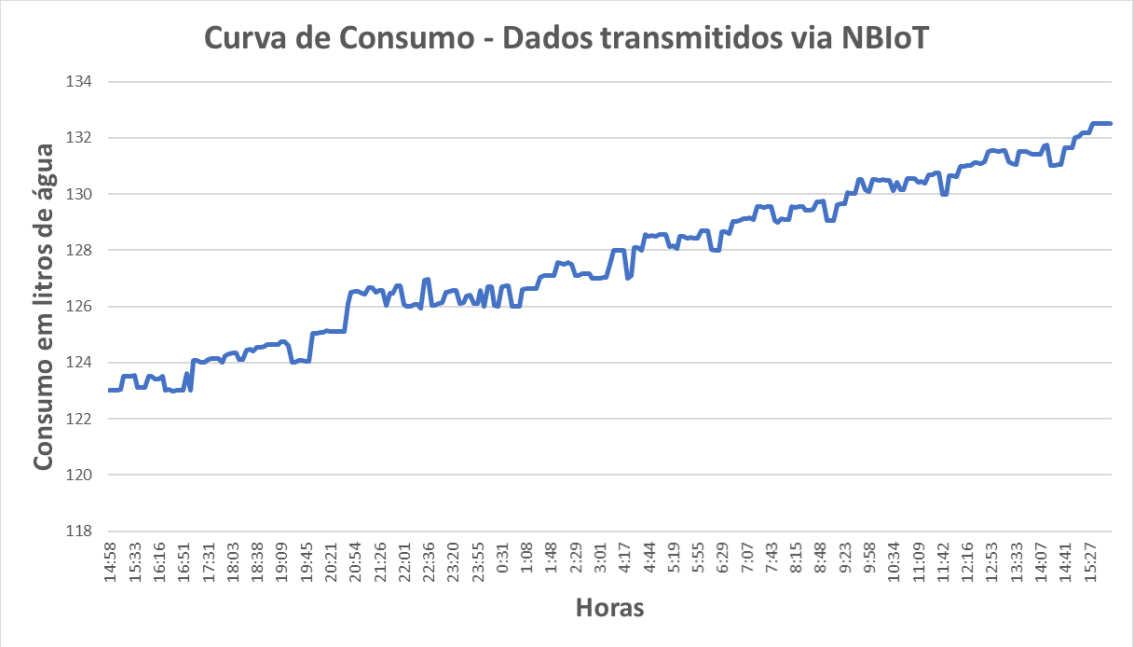

Figura 37 – Consumo de água do hidrômetro, figura do autor

A fim de verificar a exatidão dos valores reconhecidos pelo dispositivo de processamento de imagem recebido pelo transmissor, foi determinado que o sentido do fluxo de água do hidrômetro seria sempre positivo. Dessa forma, não poderiam haver valores menores que os anteriores. No entanto, como se trata de uma aplicação que utiliza recursos de IA, devem ser considerados os parâmetros empregados no reconhecimento de imagem, como acurácia e erro absoluto, e relacionar os valores reconhecidos com o consumo de água, que é determinado por uma equação linear. Assim, observa-se a diferença entre o valor esperado e o valor reconhecido. A Figura 38 apresenta ambas as curvas: os valores reconhecidos, chamados de valores reais, e os valores esperados, denominados no gráfico como reta simulada.

A imagem apresenta duas curvas características, em azul a medição real realizada pelo equipamento, em laranja a reta linear do consumo esperado determinado pela equação y=0,032x+123,09 apresentado no gráfico. Com os valores reais da medição e os valores da curva de consumo esperado é possível analisar o erro absoluto da medição, ao qual pode ser avaliado pela equação do MAE (do inglês *Forward Error Correction*):

$$MAE = \frac{1}{N} \sum_{i=0}^{N-1} |y_i - y'_i| \,. \tag{4.1}$$

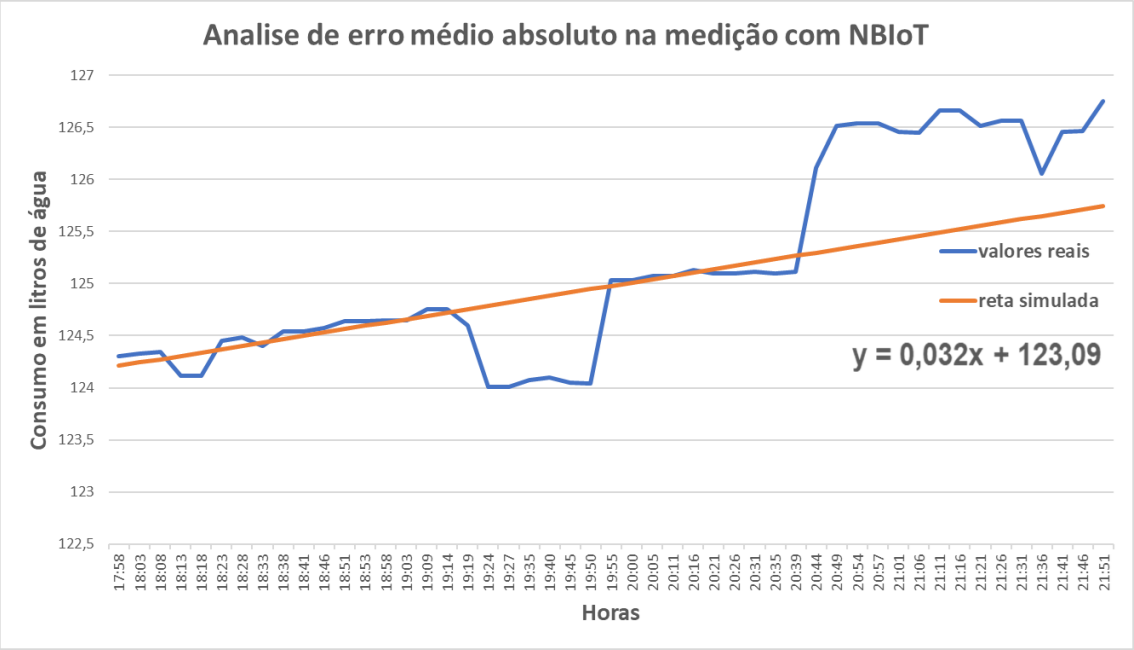

Figura 38 – Erro médio absoluto na medição, figura do autor

Na equação 4.1, $1/N$ será utilizado para calcular a média da quantidade total N, n é o numero de amostras total y_i' o valor medido e y_i o valor esperado, desta forma a substituição dos valores resulta em:

$$MAE = 1/387(|123,12 - 123,01| + |123,154 - 123,01| \dots).$$

O resultado do erro médio absoluto neste trabalho foi de 0,289997 o que significa que em média a diferença absoluta entre os valores previstos e observados são de 0,289997 litros de água consumidos, este resultado provém do erro resultante do treinamento citado em 3.3.1 na Figura 39 onde foi obtido 0,2152.

Além da transmissão dos dados de consumo de água, a potência do sinal foi monitorada durante a transmissão. Em média, a potência do sinal recebido foi de -60 dBm. Vale mencionar que o NBIoT, apesar de fazer parte da tecnologia 4G, possui uma sensibilidade maior devido às suas características de banda estreita, o que permite um alcance maior em cenários de baixa potência. Segundo a folha de dados do fabricante do módulo *Telit*, sinais menores que -100 dBm são classificadas como péssimas, enquanto sinais entre -50 dBm e -60 dBm são classificadas como excelentes. No entanto, ocorreram 18 perdas durante a transmissão, representando 4,65% dos dados transmitidos.

Além das perdas durante a transmissão, outras inconsistências ocorreram nos dados enviados, entre as 387 amostras enviadas, 32 valores não foram totalmente identificados, este problema foi causado devido ao treinamento realizado com os dados. Conforme mencionado anteriormente na Seção 3.3.1, a Figura 26 apresenta um gráfico da relação de acurácia por época, com um resultado de 93,52% de acurácia, 32 valores errôneos dos dados transmitidos resulta em uma acurácia de 91,73%, o que está próximo do resultado

esperado. Esta taxa de erro está associada ao chamado "número transitório", onde um valor não pode ser identificado pois está na transição entre dois valores.

Esses erros ocorrem principalmente devido à dificuldade do modelo em lidar com valores de transição que não foram previamente vistos durante o treinamento. Demonstra-se, assim, uma limitação do modelo ao enfrentar situações que envolvem valores intermediários e não estáveis. Na Figura 39, apresenta-se um exemplo dessa situação.

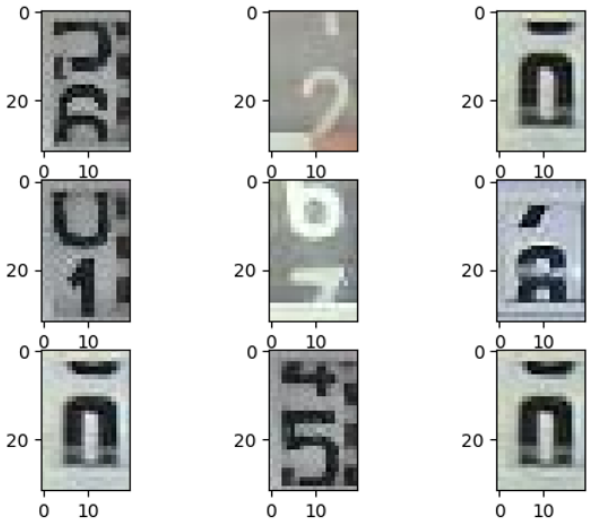

Figura 39 – Valores transitório das amostras coletadas para o treinamento da rede neural, figura do autor

A imagem ilustra várias transições de algarismos coletadas do hidrômetro. Ressalta-se que trata-se de um hidrômetro com indicadores analógicos, onde a troca de dígitos dos algarismos é totalmente mecânica: de 5 para 6, de 1 para 2, de 9 para 0, de 0 para 1, de 6 para 7, de 7 para 8, de 9 para 0, de 4 para 5 e de 9 para 0. Como esses valores não fazem parte do treinamento e tal situação ocorreu apenas nos testes práticos, a previsão desses valores previamente não foi possível, o que ocasionou erros de reconhecimento.

Em complemento a análise dos dados transmitidos, foi avaliado consumo de energia do equipamento de transmissão NBIoT afim de dimensionar um banco de bateria para a aplicação. Para isto, utilizou-se o método de monitoramento do consumo durante a ação do equipamento, que incluí transmissão, recepção e modo de espera. O objetivo foi capturar com precisão os padrões de consumo ao longo do tempo de operação. Assim, foi realizado um ensaio onde, inicialmente, o dispositivo foi isolado da aplicação principal e configurado com fonte controlada, resistência de carga *shunt* e osciloscópio para medição. A Figura 40 ilustra a montagem dos equipamentos para o ensaio.

O dispositivo de reconhecimento de imagem utilizado neste trabalho é um ESP32-CAM. O transmissor é um módulo proprietário da *Telit*, modelo ML865G. A fonte de alimentação de bancada utilizada para alimentar o sistema de medição foi uma Minipa, modelo ML-3303M, com uma saída regulada de 3,3 V e proteção contra surtos de corrente

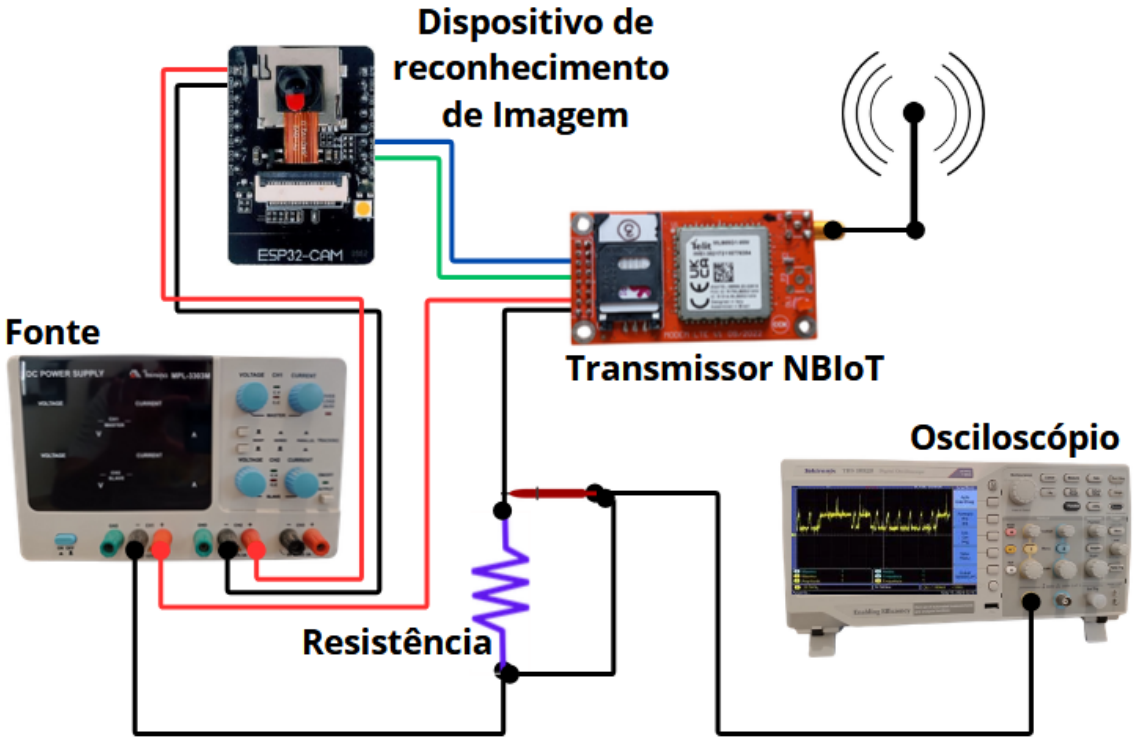

Figura 40 – Montagem dos equipamentos para realização do ensaio para medição de energia do módulo de transmissão NBIoT, figura do autor

de 500 mA. O osciloscópio utilizado para realizar as amostragens da tensão foi um equipamento da Tektronix, modelo TBS 1052B, com uma taxa de amostragem de 1 GS/s. Por fim, foi utilizada uma resistência de 1 ohm com uma precisão de 0,1% como resistência de carga *shunt*.

O consumo de energia foi determinado a partir da curva de tensão amostrada pelo osciloscópio. Considerando que a resistência de *shunt* é de 1 Ohm, a Lei de Ohm permite interpretar essa curva de tensão como uma curva de corrente em relação ao tempo. A Figura 41 ilustra a curva de corrente ao longo do tempo, obtida pelo osciloscópio durante a transmissão de 7 bytes pelo transmissor NBIoT. Esta representação fornece uma visão clara dos padrões de consumo de energia do dispositivo durante a operação.

Nesta curva, destacam-se três comportamentos distintos do dispositivo: (1) a transmissão dos dados, com um período de 0,898 segundos; (2) a recepção dos dados, com 0,337 segundos; e (3) o período no qual o dispositivo aguarda o envio ou recebimento de mensagens, com aproximadamente 1,678 segundos.

Para realizar uma comparação de consumo com o transmissor LoRa, foi adotado um padrão composto por envio e modo de aguardo. Desta forma, a transmissão de dados, destacada na Figura 41 como (1), e a recepção como (2), foram somadas para compor o período de "RX/TX", que representa a transmissão. O restante até a próxima recepção de dados foi denominado de aguardo. É importante notar que o intervalo de tempo entre

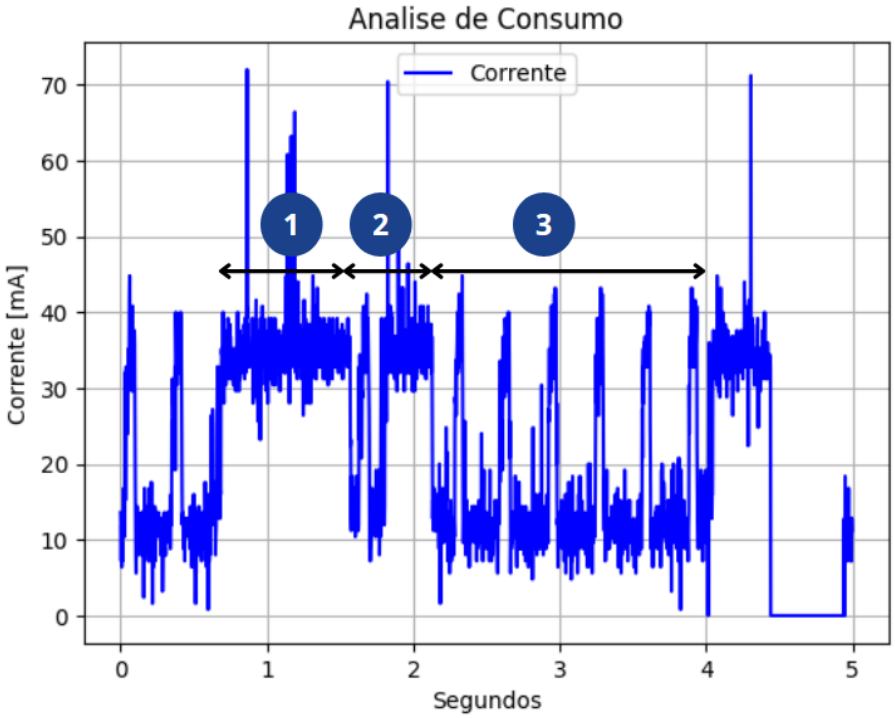

Figura 41 – Gráfico da curva de corrente em função do tempo da transmissão por NBIoT, figura do autor

.

os envios de dados foi ajustado de 5 minutos para aproximadamente 3 segundos, para se adequar à escala de amostragem do osciloscópio.

Para o cálculo do consumo, empregou-se a integração utilizando o método de Simpson [51]. Este método consiste em uma aproximação retangular ponto a ponto, sendo computacionalmente integrável e resultando em uma precisão superior em comparação com as aproximações manuais de funções. Como resultado, durante os períodos 1 e 2 da Figura 41, a área foi calculada como sendo $47,178$ mAs.

Comercialmente, os equipamentos não trabalham com a notação "As", sendo mais comumente expressos em W ou mW/h. Para determinar este valor, é necessário dividir o valor encontrado pelo tempo de uma hora para obter a razão horária do consumo, conforme apresentado na equação 4.2. Posteriormente, multiplica-se o valor encontrado pela tensão utilizada no teste conforme a equação 4.3.

$$Consumo = \left(\frac{47,178.10^{-3}}{3600}\right) = 13,105\ \mu Ah \tag{4.2}$$

$$13,055.10^{-6}.3,3 = 43,246\ \mu Wh \tag{4.3}$$

A Tabela 3 apresenta um resumo do consumo de ambas etapas do transmissor NBIoT

Tabela 3 – Consumo do Transmissor NBIoT

Etapa	Duração calculado	consumo em μAh	consumo em μWh
RX/TX	1,234 s	$13,055\ \mu Ah$	$43,017\ \mu Wh$
Aguardo	1,678 s	$7,270\ \mu Ah$	$23,992\ \mu Wh$
Soma	2,912 s	$20,325\ \mu Ah$	$67,009\ \mu Wh$

Os resultados indicam que durante a transmissão e recebimento, descritos na tabela como "RX/TX" tem-se o maior consumo, o que é esperado devido atuação do equipamento. Em comparação as informações da folha de dados [48], observa-se que o consumo tipico do dispositivo é de 20 μA em modo de aguardo e de até 50 mA para transmissões de dados. Desta forma, foi possível validar que a medição de energia do dispositivo NBIoT está coerente com os valores apresentados na folha de dados.

4.2 Analise dos resultados obtidos dos dados transmitidos via LoRa

As transmissões de dados via LoRa foram realizados na cidade de Santa Rita do Sapucaí - Minas Gerais, utilizando a mesma configuração geográfica empregada na transmissão via NBIoT, descrita na Seção anterior. A distância ponta a ponta da aplicação é de 6,23km, A Figura 42 apresenta o mapa onde a reta destacada em preto representa a distancia da transmissão.

Figura 42 – Distância dos dados transmitidos com tecnologia LoRa, figura do autor

Entre os dias 08/04/24 e 09/04/24, foram realizadas 241 transmissões utilizando a mesma configuração do dispositivo de reconhecimento de imagem conectado ao transmissor de dados via LoRa, com o intervalo de 5 minutos entre as transmissões para análise

em curto período de tempo. Na Figura 43, são apresentados os valores de consumo de água registrados pelo hidrômetro durante esse período.

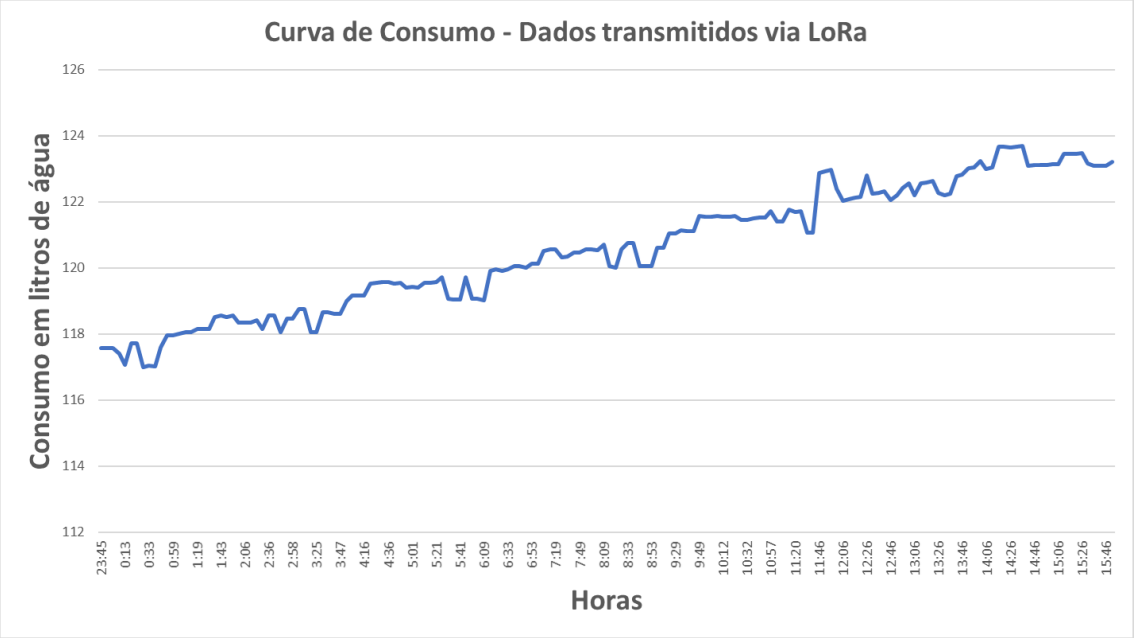

Figura 43 – Consumo do Hidrometro durante a transmissão via LoRa, figura do autor

Para verificar a exatidão dos valores reconhecidos pelo dispositivo de processamento de imagem via LoRa, foi adotado o mesmo critério utilizado na transmissão via NBIoT, garantindo que o fluxo de água do hidrômetro fosse sempre positivo. Dessa forma, a diferença entre os valores esperados e reconhecidos foi analisada, considerando a acurácia e o erro absoluto, conforme ilustrado na Figura 44, que apresenta uma janela dos dados transmitidos entre 02:00 e 06:48.

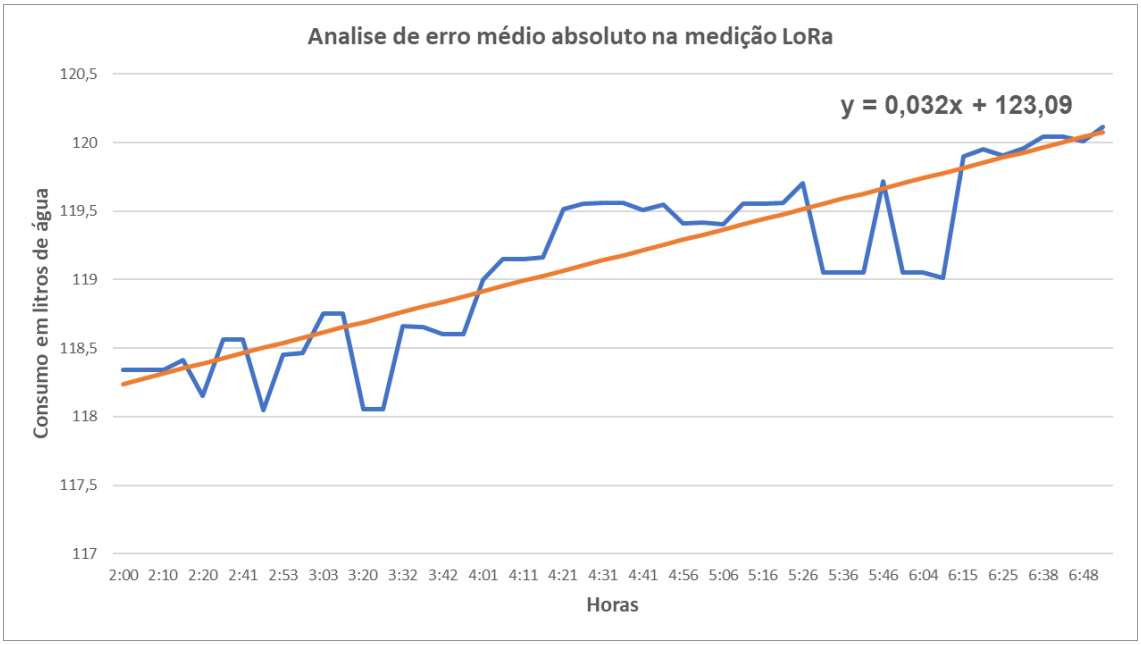

Figura 44 – Erro médio absoluto na medição LoRa, figura do autor.

O valor do erro médio absoluto para os dados transmitidos em LoRa foi de 0,2496, isto indica que, em média, a diferença absoluta entre os valores previstos e observados é de 0,2496 litros de água consumidos comparado ao treinamento mencionado na Sessão 3.3.1, onde o erro do treinamento é de 0,2152. Diferente dos dados transmitidos, a qualidade do sinal LoRa é fixa para o transmissor, configurada na programação com um valor fixo em 20 dBm.

De forma semelhante aos resultados prévios, de um total de 241 amostras enviadas, 28 amostra não foram totalmente identificadas, o que resulta em 92,76% de acurácia, um valor próximo da acurácia do treinamento de 93,52%.

A análise do consumo de energia do equipamento de transmissão com LoRa, utilizou a mesma metodologia da medição de consumo com NBIoT. O dispositivo foi isolado da aplicação principal e configurado com uma fonte controlada, uma resistência de carga *shunt* de 1 Ohm e 0,1% de precisão, e um osciloscópio para medição. A Figura 45 ilustra a disposição dos dispositivos na montagem.

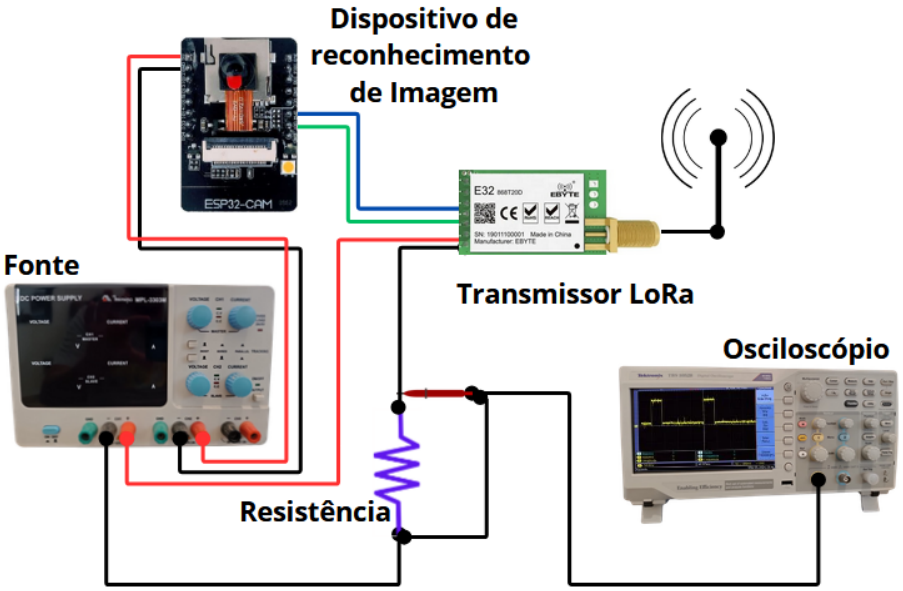

Figura 45 – Diagrama do circuito de coleta dos dados de consumo LoRa, figura do autor

O transmissor foi um módulo proprietário da Ebyte modelo E32-915T20D. O dispositivo de reconhecimento de imagem, a fonte de bancada e o osciloscópio foram os mesmos empregados na medição do NBIoT, mantendo a mesma configuração e calibração.

O consumo de energia é representado de forma semelhante ao transmissor com NBIoT, por meio de uma curva de corrente em relação ao tempo. A Figura 46 ilustra a curva de corrente em função do tempo, obtida pelo osciloscópio durante a transmissão de um pacote de dados de 7 bytes pelo transmissor LoRa.

Nesta curva, destacam-se dois comportamentos distintos do dispositivo: (1) a trans-

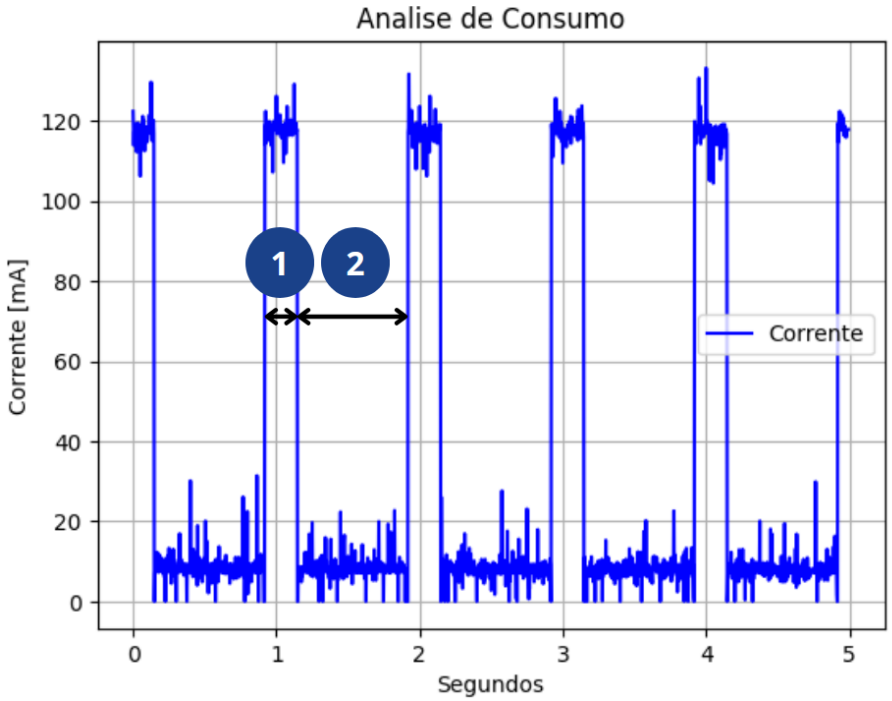

Figura 46 – Gráfico da curva de corrente em função do tempo da transmissão por LoRa, figura do autor

.

missão dos dados, com um período de 0,229 segundos e (2) o período oscilatório, no qual o dispositivo aguarda o envio ou recebimento de mensagens com 0,752 segundos. Durante a etapa de medição de energia, foi necessário alterar o tempo de transmissão dos dados do LoRa devido ao curto período de amostragem do envio de dados. Portanto, o tempo de envio foi ajustado para aproximadamente 1 segundos, visando uma melhor amostragem do período de envio.

Os consumos foram calculados em duas partes: durante a transmissão e em modo de espera. Os resultados foram obtidos por meio do método de Simpson. Durante o período de transmissão, conforme apresentado como (1) na Figura 46, a área foi calculada como sendo 37,772 mAs e convertida em μWh:

$$Consumo = \left(\frac{37,772 \times 10^{-3}}{3600}\right).3,3 = 34,579\ \mu Wh \tag{4.4}$$

A Tabela 4 apresenta um resumo do consumo de ambas etapas do transmissor LoRa

Quando comparadas às informações da folha de dados [46] é citado que a corrente de transmissão do dispositivo é de 120 mA e modo de recepção de até 14 mA. Desta forma, foi possível validar que a medição de energia do dispositivo LoRa está coerente com os valores apresentados na folha de dados.

Para concluir a análise dos consumos de energia, foi realizada uma análise do

Tabela 4 – Consumo do Transmissor LoRa

Etapa	Duração calculado	consumo em Ah	consumo em Wh
Transmissão	0,229 s	$10,478\ \mu$Ah	$34,579\ \mu$Wh
Aguardo	0,752 s	$2,455\ \mu$Ah	$8,103\ \mu$Wh
Soma	0.981 s	$12,933\ \mu$Ah	$42,682\ \mu$Wh

consumo do dispositivo de reconhecimento de imagens (ESP32-CAM), seguindo o mesmo procedimento utilizado para os dispositivos transmissores. A Figura 47 ilustra o ensaio de consumo de energia para o dispositivo de reconhecimento de imagem.

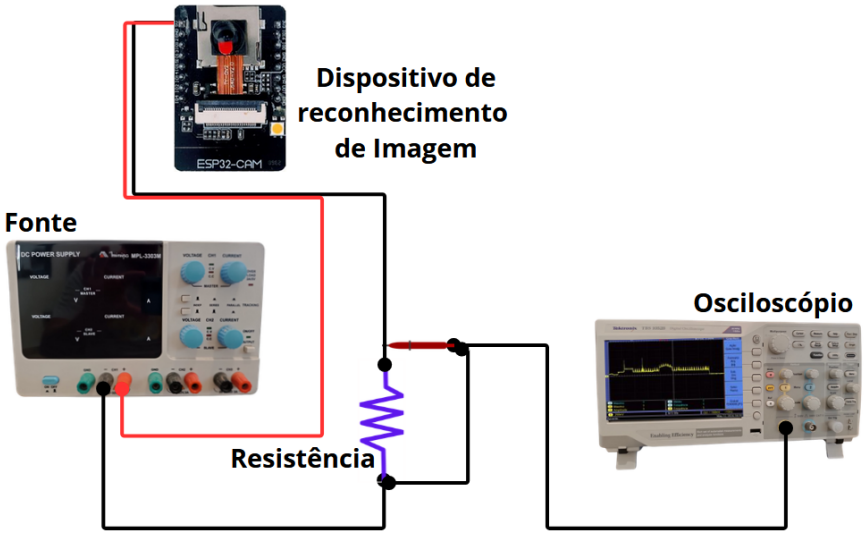

Figura 47 – Diagrama do circuito de coleta dos dados do ESP32-CAM, figura do autor

Análogo as medições anteriores, o gráfico da Figura 48 mostra a corrente em função do tempo durante a medição.

Diferentemente dos dispositivos de transmissão, o consumo do dispositivo de reconhecimento de imagens é dividido em seis etapas: (1) inicialização do microcontrolador, (2) inicialização dos periféricos da câmera, (3) operação dos periféricos de *firmware*, (4) flash, (5) captura de foto, processamento e reconhecimento, e (6) modo de espera. Utilizando das mesmas técnicas mencionadas anteriormente para o cálculo do consumo, a Tabela 5 mostra o consumo das diferentes etapas do dispositivo

A análise detalhada dessas etapas permite identificar os períodos de maior consumo de energia e otimizar o desempenho do dispositivo de reconhecimento de imagens. Em particular, a etapa de captura de foto e acionamento do LED demanda mais energia devido às exigências do *hardware*. Compreendendo o consumo específico em cada etapa do dispositivo de reconhecimento de imagens e dos transmissores, é possível dimensionar adequadamente o banco de baterias para a aplicação.

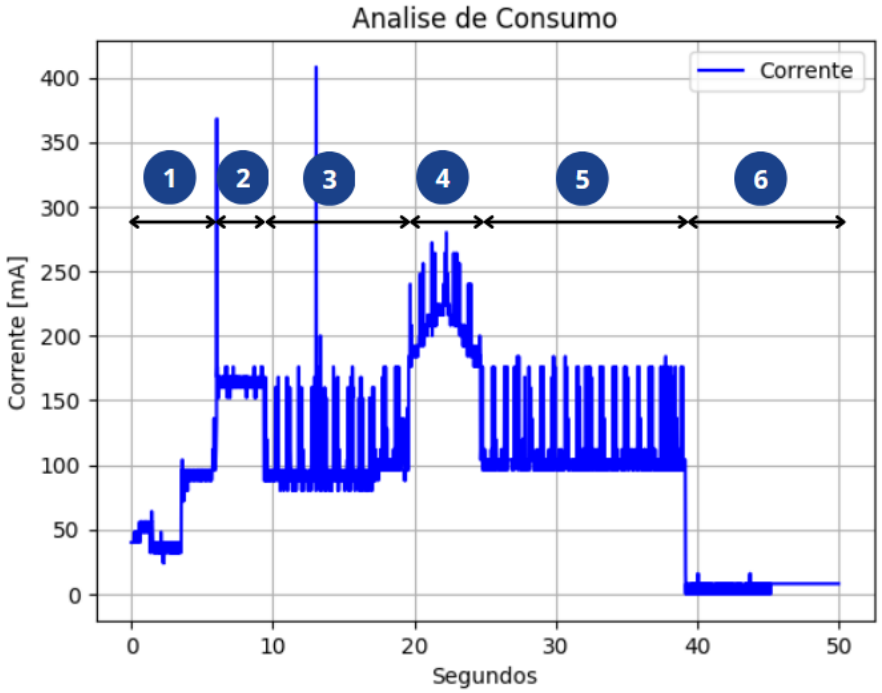

Figura 48 – Gráfico da curva de corrente em função do tempo do ESP32-CAM, figura do autor

Tabela 5 – Consumo de diferentes etapas de um ESP32-CAM

Etapa	Duração calculado	consumo em Ah	consumo em Wh
1	6,020 s	$104,63\ \mu$Ah	$345,29\ \mu$Wh
2	3,300 s	$149,77\ \mu$Ah	$494,26\ \mu$Wh
3	10,26 s	$290,87\ \mu$Ah	$959,87\ \mu$Wh
4	5,580 s	$307,17\ \mu$Ah	1,013 mWh
5	14,52 s	$437,98\ \mu$Ah	1,445 mWh
6	10,82 s	$17,369\ \mu$Ah	$57,29\ \mu$Wh
Soma	50,53 s	1,307 mAh	4,314 mWh

4.3 Comparativo entre NBIoT e LoRa

De forma resumida, a Tabela 6 apresenta um comparativo detalhado entre as tecnologias NBIoT e LoRa aplicadas ao monitoramento por imagem com processamento de borda. Este comparativo explora os resultados obtidos com ambas as tecnologias de transmissão, destacando suas vantagens e desvantagens. Ao longo desta sessão, serão analisados aspectos como eficiência energética, alcance, capacidade de transmissão de dados e adequação para diferentes cenários de aplicação. O objetivo é fornecer uma visão abrangente e comparativa que auxilie na escolha da tecnologia mais apropriada para aplicações específicas de monitoramento por imagem com processamento de borda.

A Tabela 6 fornece um comparativo abrangente entre as tecnologias NBIoT e LoRa, destacando diversos parâmetros críticos para o monitoramento por imagem com

Tabela 6 – Comparativo entre as tecnologias NBIoT e LoRa

Procedimento	NBIoT	LoRa
Distancia de transmissão	1,3 km	6,23 km
Frequência de transmissão	728 MHz	915 MHz
Erro médio absoluto (MAE)	0,289	0,2496
Perdas por conta da qualidade do sinal	18	0
Perdas por conta do reconhecimento de imagem	32	28
Tempo de transmissão durante o teste de envio	1,25 s	0,229 s
Consumo de energia durante a transmissão	$43,017\ \mu Wh$	$34,579\ \mu Wh$
Tempo em modo de aguardo	1,67 s	0,7525 s
Consumo de energia durante o modo de aguardo	$23,992\ \mu Wh$	$8,103\ \mu Wh$

processamento de borda. A análise dos resultados permite observar as seguintes considerações:

- Distância de Transmissão: Em termos de cobertura de longo alcance, tanto o LoRa quanto o NBIoT apresentam características semelhantes. De acordo com [3], o NBIoT cobre uma distância estendida de 18 a 20 km, superando o LoRa, que suporta de 12 a 15 km conforme [8]. Embora o NBIoT ofereça melhor desempenho em áreas urbanas, seu desempenho é apenas mediano em subúrbios e áreas rurais, onde o LoRa se destaca por operar eficientemente em áreas abertas com linha de visada. Nos testes realizados neste trabalho, o LoRa demonstrou uma capacidade superior de transmissão, porém, é importante ressaltar que o local dos testes estava situado em um subúrbio com alta elevação em relação à cidade local, o que pode ter influenciado os resultados.

- Frequência de Transmissão: O LoRa opera na banda ISM (do inglês *Industrial, Scientific, and Medical*) livre não licenciada. A banda ISM é uma faixa de frequência de rádio que foi originalmente reservada para o uso de equipamentos industriais, científicos e médicos, mas que hoje também é amplamente utilizada para comunicação sem fio de curto alcance, como em dispositivos que utilizam tecnologias como *WiFi*, *Bluetooth* e LoRa. O grande benefício do uso dessa banda está na ausência de custos para sua utilização, desde que sejam seguidas as regulamentações de potência e interferência estabelecidas pela ANATEL (Agência Nacional de Telecomunicações). Já o NBIoT opera em bandas de frequência licenciadas e pode ser implementado de três maneiras diferentes: de forma independente, na banda de guarda ou dentro da própria banda. A maioria das operadoras ao redor do mundo utiliza a faixa de 900 MHz para o NBIoT, com algumas implementações ocorrendo na faixa de 800 MHz. Por pertencer a uma banda licenciada, semelhante às usadas para 2G, 3G e 4G,

o NBIoT faz o uso de uma frequência planejada que enfrenta relativamente pouca interferência. Além disso, essa tecnologia pode ser integrada às estações base das redes celulares já existentes o que acarreta custos a implementação.

- Erro Médio Absoluto (MAE): Neste trabalho o MAE fornece uma medida direta de quão grandes são os erros em média, sem considerar a direção dos erros (se são positivos ou negativos). O MAE é uma métrica fácil de entender e interpretar, uma vez que está na mesma unidade de medida que os dados originais, é uma medida que não é sensitiva que não penaliza grandes discrepâncias. Durante a transmissão do LoRa o MAE foi de 0,2496 e durante a transmissão NBIoT 0,289. A avaliação do MAE não tem relação com a transmissão, o intuito é avaliar se um método de transmissão não vai interferir nos dados recebidos, qualquer erro na transmissão resultaria em um maior MAE entre as medições reais e as previstas.

- Perdas devido à Qualidade do Sinal e Reconhecimento de Imagem: A qualidade do sinal no NBIoT pode ser comprometida por interferências de outros dispositivos ou por condições ambientais, como obstáculos físicos (prédios, árvores, etc.), condições meteorológicas adversas, ou até mesmo a movimentação de pessoas e veículos. Quando o RSSI (do inglês *Received Signal Strength Indicator*) é baixo, a probabilidade de perda de pacotes aumenta significativamente. Devido à localização do transmissor NBIoT e ao indicativo de baixo RSSI, ocorreram perdas consideráveis durante a transmissão (18 perdas). Por outro lado, o LoRa, estando melhor localizado e com uma linha de visão clara para o receptor, mesmo à distância, não apresentou perdas. Isso sugere que a localização da instalação pode ter uma influência direta na ocorrência de perdas de pacotes.

- Tempo e Consumo de Energia durante Transmissão e Modo de Aguardo: Em termos de eficiência energética, o LoRa mostrou-se mais vantajoso, consumindo menos energia tanto durante a transmissão quanto no modo de aguardo. O tempo de transmissão do LoRa também foi maior, o que pode ser um fator a ser considerado dependendo da aplicação específica.

Em resumo, a escolha entre NBIoT e LoRa depende dos requisitos específicos da aplicação. O LoRa se destaca pelo modo proprietário que pode alcançar longas distancias em boas situações e possui melhor eficiência energética, enquanto o NBIoT pode ser mais adequado para cenários que exigem uma maior densidade de mensagens transmitidas. A decisão final deve considerar o equilíbrio entre alcance, consumo de energia, robustez do sinal e a quantidade de dados a serem transmitidos.

4.4 Dimensionamento do banco de baterias para aplicação

Com base no consumo de todos os equipamentos, foi possível dimensionar um banco de baterias para os dispositivos. Para isto, é necessário simular dois cenários de consumo, nos quais todos equipamentos permanecem em modo de baixo consumo e realiza uma transmissão por dia.

Para o dispositivo NBIoT, deve-se determinar inicialmente o tempo em que o dispositivo deve permanecer em modo de baixo consumo por um dia. O tempo de operação deve ser encontrado somando-se todos os tempos de consumo das etapas do dispositivo de reconhecimento de imagem ao tempo de envio do transmissor, resultando em 53,442 segundos ou 0,04 horas. Esse valor de tempo de operação implica que o dispositivo deve permanecer em modo de baixo consumo por 23,96 horas. Ainda segundo o *datasheet* [47] o modulo NBIoT em baixo consumo opera com 3 μA, em um dia de operação subtraindo o envio, resulta em 72 μAh.

Segundo o *datasheet* [5] o consumo do ESP-32 em modo de baixo consumo é de 10 μA, ao multiplicar o valor pelo tempo que o dispositivo deve permanecer em baixo consumo, estima-se 240 μAh. Finalmente, somando-se o consumo total do dispositivo de reconhecimento de imagem, o tempo em que o equipamento deve permanecer em modo de baixo consumo, e o consumo da transmissão do NBIoT, obtém-se:

Tabela 7 – Consumo do ESP32 + NBIoT

Etapa	consumo em Ah
Consumo do ESP-32 em modo de operação	1,307 mAh
Transmissão do NBIoT	13,055 μAh
Consumo do ESP-32 em modo de baixo consumo	240 μAh
NBIoT em modo de baixo consumo	72 μAh
Somatório das correntes	1,632 mAh
Consumo	5,385 mWh

O mesmo calculo pode ser empregado ao consumo do dispositivo LoRa, considerando o consumo total do dispositivo de reconhecimento de imagem, O mesmo cálculo pode ser aplicado ao consumo do dispositivo LoRa, O mesmo cálculo pode ser aplicado ao consumo do dispositivo LoRa, considerando tanto o consumo total do dispositivo de reconhecimento de imagem quanto o tempo em que o equipamento permanece em modo de baixo consumo, além do consumo adicional durante a transmissão utilizando LoRa:

Com os consumos totais calculados, deve-se utilizar uma bateria comercial, encontrar o valor em mW/h e dividir os valores para estimar o tempo em horas, dias e anos. O cálculo apresentado abaixo foi realizado com uma bateria comercial da I2G de 12 V 6800

Tabela 7 – Consumo do ESP32 + LoRa

Etapa	consumo em Ah
Consumo do ESP-32 em modo de operação	1, 307 mAh
Transmissão do LoRa	10, 478 μAh
Consumo do ESP-32 em modo de baixo consumo	240 μAh
LoRa em modo de baixo consumo	96 μAh
Somatório das correntes	1, 653 mAh
Consumo	5, 4564 mWh

mAh para o dispositivo com transmissor NBIoT:

$$\begin{aligned} &Bateria : 12\ V \times 6800\ mAh = 81600\ mWh \\ &Duração\ em\ horas : \left(\frac{81600\ mWh}{5,385\ mW}\right) = 15153,2\ horas \\ &Duração\ em\ dias : \left(\frac{15153,2\ horas}{24\ horas}\right) = 631,38\ dias \end{aligned} \tag{4.5}$$

O mesmo cálculo pode ser aplicado ao consumo do dispositivo LoRa para estimar a duração de uma bateria comercial de 12 V e 6800 mAh:

$$\begin{aligned} &Duração\ em\ horas : \left(\frac{81600\ mWh}{5,4564\ mW}\right) = 14954,9\ horas \\ &Duração\ em\ dias : \left(\frac{14954,9\ horas}{24\ horas}\right) = 623,12\ dias \end{aligned} \tag{4.6}$$

Portanto, devido ao baixo consumo inerente às duas tecnologias investigadas, bem como ao número de envios limitados por dia, tanto o NBIoT quanto o LoRa demonstram ser opções viáveis para o monitoramento por imagem com processamento de borda. Os cálculos realizados indicam que ambas as tecnologias podem operar por longos períodos com uma única carga de bateria comercial de 12V e 6800 mAh, com o dispositivo NBIoT alcançando aproximadamente 631 dias de operação e o dispositivo LoRa alcançando cerca de 623 dias.

Esses resultados ressaltam a importância de considerar o consumo energético na escolha da tecnologia de transmissão para aplicações IoT, evidenciando que tanto o NBIoT quanto o LoRa oferecem eficiência energética e sustentabilidade, com o NBIoT apresentando uma pequena vantagem em termos de duração da bateria.

5 Conclusões e Trabalhos Futuros

Neste trabalho, foi explorado um método baseado em ML aplicado ao processamento de borda para reconhecimento das imagens dos dígitos de um hidrômetro. Essa abordagem visa atender aos requisitos do IoT como transmissões a longas distâncias e baixo consumo de energia acrescido de alto processamento.

Ao longo da pesquisa, foram elaboradas várias alterações no código da aplicação principal para sincronizar o processamento, o reconhecimento de imagem e a transmissão. Isso resultou em dois métodos distintos que trabalham com processamento em paralelo e transmitem os dados reconhecidos, demonstrando a viabilidade de utilizar transmissores de baixo consumo nesta aplicação de processamento de borda. Tal método pode ser empregado para outras aplicações, desde que suporte processamento em paralelo com RTOS (do inglês *Real-Time Operating System*) e tenha memória RAM (do inglês *Random Access Memory*) disponível.

No contexto do processamento de borda, conforme citado em [12], é importante compreender que não se trata apenas de coletar dados. Diferentemente das aplicações de telemetria, os dados transmitidos pelo processamento de borda representam os resultados finais. Quando se utiliza aprendizado de máquina para encontrar esses resultados, é essencial considerar os conceitos da inteligência artificial que abordam a eficácia dos resultados. Como mencionado em [28], existe o conceito de *overfitting*, no qual o modelo de aprendizado de máquina não pode alcançar uma margem de erro absoluto (MAE) e acurácia de 100%, o que significa que o modelo não apresentará resultados precisos em testes práticos. Segundo a literatura [28], é aceitável que a MAE esteja dentro de 1% e que a acurácia seja superior a 90%. Neste trabalho, foram obtidos resultados com MAE de aproximadamente 300 mL de água quando os dados foram transmitidos via NBIoT e 0,2496 litros de água nas transmissões realizadas via LoRa, o que demonstrou uma margem de erro média de 0,267 litro de água, considerada aceitável para aplicações em que o hidrômetro de 3/4 de polegada é comumente utilizado.

Em termos das tecnologias utilizadas para a transmissão dos dados, é importante destacar que cada aplicação possui suas vantagens e desvantagens. Em relação ao NBIoT, uma vantagem destacada é o uso de apenas um dispositivo para a transmissão de dados, diferentemente do LoRa, que necessita de transmissor e receptor. Outra grande vantagem do NBIoT está no escalonamento de dispositivos, uma vez que a recepção é um serviço exclusivo da estação base de telefonia, embora isso resulte em custos adicionais. Algumas desvantagens do NBIoT incluem a perda de pacotes e o consumo de energia, onde o LoRa se mostrou mais eficiente.

Em relação ao LoRa, destacam-se o longo alcance e a baixa potência de transmissão em comparação com o NBIoT. No entanto, a desvantagem do LoRa está relacionada ao escalonamento da aplicação, pois o uso de múltiplos dispositivos requer a criação ou a aquisição de um *gateway* proprietário para tratar corretamente as mensagens recebidas de cada dispositivo.

A contribuição deste trabalho reside na demonstração da viabilidade e eficácia da utilização de transmissores de baixo consumo de energia, como NBIoT e LoRa, em aplicações de processamento de borda para o reconhecimento de imagens em tempo real. Além disso, este trabalho demonstrou que os resultados de reconhecimento de imagem em um hidrômetro estão diretamente associados ao aprendizado, proveniente do treinamento dos dados. Os parâmetros utilizados para mensurar os resultados obtidos são os mesmos comumente empregados em análises de dados com inteligência artificial, como MAE e acurácia.

Embora os resultados sejam promissores, é importante ressaltar existem limitações desta aplicação, principalmente associadas à instalação do dispositivo. É necessário alinhar manualmente os valores da imagem para que a captura dos ROIS seja feita corretamente. Uma vez instalada a aplicação, qualquer movimento brusco no dispositivo acoplado à câmera pode resultar no desalinhamento da imagem e na perda de dados. Uma solução para este problema seria a fixação de um suporte universal com PCB (do inglês *Printed Circuit Board*) e câmera acoplada.

Como sugestão para pesquisas futuras, sugere-se a criação de um circuito dedicado em vez de utilizar placas de desenvolvimento, como foi feito neste trabalho. O uso de microcontroladores de alto desempenho, como os da Família M1 ou F4 da *STM-Microcontroler*, pode ser explorado para melhorar ainda mais o desempenho e a eficiência do sistema. Essas abordagens podem proporcionar maior integração e otimização dos componentes, contribuindo para um sistema mais eficiente e robusto para aplicações de processamento de borda. Em termos de melhorias dos resultados é sugerido trabalhar com uma base de dados mais balanceada, pois no trabalho fora retiradas muitas amostras ruidosas e isto influenciou diretamente na acurácia e erro do treinamento, o nivelamento das amostras pode resultar em resultados mais precisos.

Apêndices

APÊNDICE A – Sequencia de Comandos AT para Conexão NBIoT

Comandos AT são uma série de instruções que são usadas para controlar modems. Eles foram introduzidos inicialmente pela empresa *Hayes*, mas hoje são um padrão amplamente aceito e usado para configurar, controlar e se comunicar com modems e outros dispositivos de comunicação, como módulos GSM, GPRS, NBIoT e LTE. A Figura 49 mostra a sequencia de comandos utilizados para conexão do modem NBIoT

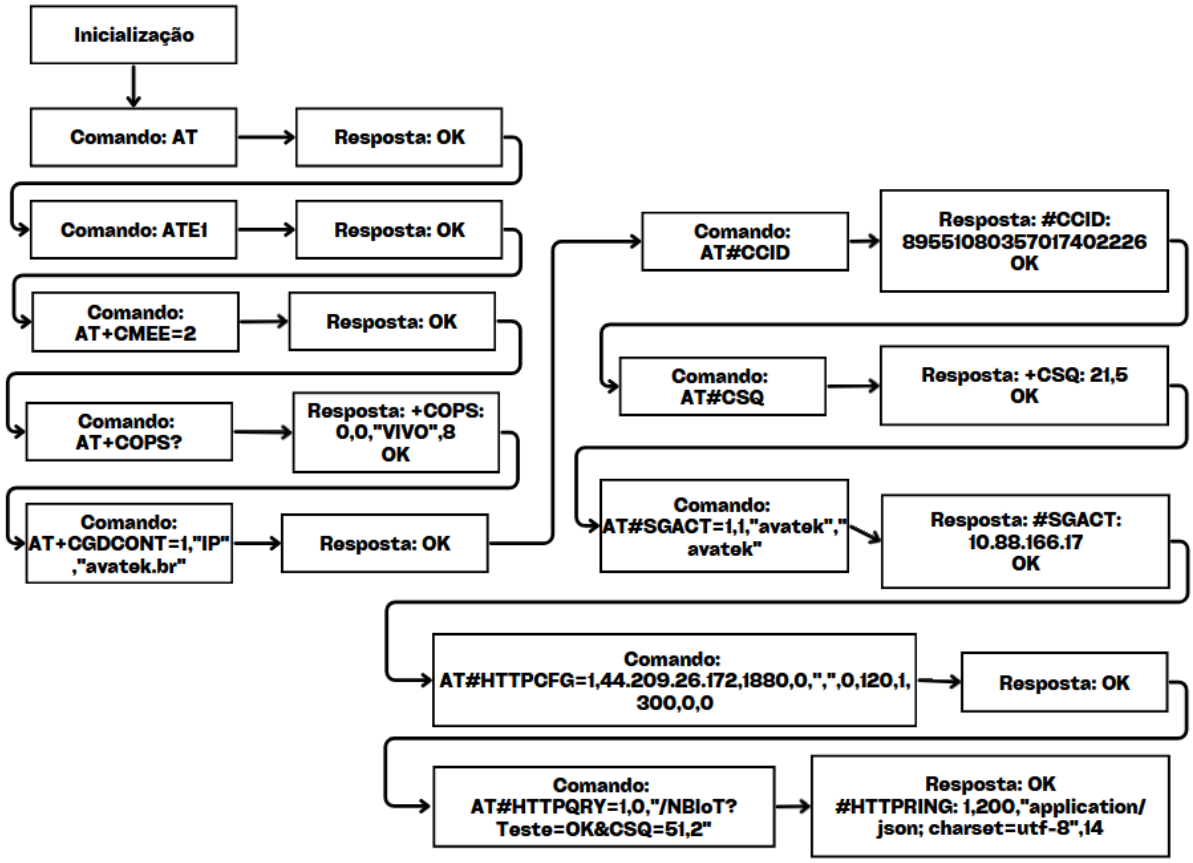

Figura 49 – Comandos AT realizados para conectar o transmissor Telit

A descrição dos comandos e suas respectivas respostas são listadas abaixo:

- Comando: **AT**: Comando de inicialização, indica atenção

  Resposta: OK
- Comando: **ATE1**: Ativa o eco dos comandos AT, ou seja, os caracteres digitados serão ecoados de volta ao terminal. Isso é útil para verificar o que está sendo digitado e enviado ao modem.

Resposta: OK

- Comando: **AT+CMEE=2**: Ativa a apresentação detalhada das mensagens de erro em formato numérico e textual. Quando um erro ocorre, o modem retorna um código numérico e uma descrição textual do erro, facilitando a identificação e a resolução de problemas.

  Resposta: OK

- Comando: **AT+COPS?**: Retorna a operadora de rede atualmente selecionada e outras informações relacionadas à rede.

  Resposta: +COPS:0,0,"VIVO",8 OK. A resposta indica: 0 = Modo de seleção automática, 0 = formato da operadora, "VIVO"= operadora, 8 = tipo de acesso a rede NBIoT

- Comando: **AT+CGDCONT=1,"IP","Avatek.br"**: Define um contexto de PDP *Packet Data Protocol* para uma conexão de dados, o perfil desta conexão é definida por:

  - **1**: O identificador do contexto PDP. Este número (CID - Context Identifier) é usado para referenciar este contexto específico.
  - **"IP"**: O tipo de PDP. Neste caso, especifica que o tipo de conexão é baseada em IP (Internet Protocol).
  - **"Avatek.br"**: O APN (Access Point Name), que é o nome do ponto de acesso fornecido pela operadora de rede para conectar à internet.

  Resposta: OK

- Comando: **AT+CCID**: Retorna o número de identificação do cartão SIM inserido no modem, conhecido como ICCID *Integrated Circuit Card Identifier*.

  Resposta: #CCID: 895510803570174102226

- Comando: **AT#CSQ**: Retorna a qualidade do sinal recebida pelo modem em termos de intensidade de sinal e bit error rate (BER).

  A resposta típica inclui dois valores:

  - RSSI *Received Signal Strength Indication*: Indica a força do sinal recebido. Este valor varia de 0 a 31, onde 0 significa sinal muito fraco e 31 sinal muito forte. O valor 99 indica que a intensidade do sinal é desconhecida ou não detectada.
  - BER *Bit Error Rate*: Indica a taxa de erro de bits, que representa a qualidade do sinal. Este valor varia de 0 a 7, onde 0 é a melhor qualidade (menor taxa de erro) e 7 é a pior. O valor 99 indica que a taxa de erro de bits é desconhecida ou não detectada.

- Comando: **AT#SGACT=1,1,"Avatek","Avatek"**: Utilizado para ativar ou desativar um contexto PDP (Packet Data Protocol) em um modem GSM/UMTS/LTE.

Especificamente, o comando AT#SGACT=1,1,"Avatek","Avatek"ativa um contexto PDP utilizando as credenciais fornecidas. Vamos detalhar os componentes deste comando:

  - **1:** O identificador do contexto PDP CID - *Context Identifier*. Este número é usado para referenciar este contexto específico.
  - **1:** A ação a ser realizada. Neste caso, 1 indica que o contexto PDP deve ser ativado. O valor 0 seria utilizado para desativar o contexto PDP.
  - **"Avatek":** Nome de usuário para autenticação.
  - **"Avatek":** Senha para autenticação.

  Resposta: #SGACT: 10.88.166.17 OK a resposta indica que o contexto PDP foi ativado com sucesso e o modem recebeu o endereço IP "10.88.166.17"

- Comando: **AT#HTTPCFG=1,44.209.26.172,1880,,,0,120,1,300,0,0**: utilizado para configurar uma conexão HTTP em um modem. Especificamente, o comando AT#HTTPCFG=1,44.209.26.172,1880,,,0,120,1,300,0,0 configura um perfil HTTP com vários parâmetros. Vamos detalhar os componentes deste comando:

  - **1:** O identificador do perfil HTTP *Profile Identifier*. Este número é usado para referenciar este perfil específico.
  - **44.209.26.172:** O endereço IP do servidor HTTP ao qual o modem se conectará.
  - **1880:** A porta do servidor HTTP.
  - **:** O nome de usuário para autenticação HTTP. Neste caso, está vazio, indicando que não há nome de usuário necessário.
  - **:** A senha para autenticação HTTP. Neste caso, está vazio, indicando que não há senha necessária.
  - **0:** Tipo de conexão (0 para HTTP, 1 para HTTPS).
  - **120:** Tempo máximo de espera para a conexão em segundos *timeout*.
  - **1:** Método de conexão (1 para conexão persistente, 0 para não persistente).
  - **300:** Tempo de inatividade da conexão persistente em segundos *idle timeout*.
  - **0:** Desativa o *proxy*.
  - **0:** Desativa a autenticação *proxy*.

  Resposta: OK

- Comando: **AT#HTTPQRY=1,0,"/NBIoT?Teste=OK&CSQ=51,2"**: Realiza uma solicitação HTTP (query) utilizando um perfil HTTP previamente configurado em um modem. Especificamente, o comando AT#HTTPQRY=1,0,"/NBIoT?Teste=OK&CSQ= uma solicitação HTTP GET. Vamos detalhar os componentes deste comando:

- **1:** O identificador do perfil HTTP *Profile Identifier* que foi configurado anteriormente com o comando AT#HTTPCFG.
- **0:** O tipo de solicitação HTTP. Neste caso, 0 indica uma solicitação GET.
- **"/NBIoT?Teste=OK&CSQ=51,2":** O recurso e os parâmetros da solicitação HTTP. Aqui, o recurso é /NBIoT, e os parâmetros da query string são Teste=OK e CSQ=51,2.

Resposta: #HTTPRING: 1,200,"application/json; charset=utf-8", 14. Neste exemplo, 1 é o identificador do perfil, 200 é o código de status HTTP (indicando sucesso), e "application/json; charset=utf-8"é o tipo do conteúdo recebido.

Anexos

ANEXO A – Artigo publicado

8th Workshop on Communication Networks and Power Systems (WCNPS 2023)

Analysis of Energy Consumption in Smart Edge Processing Device

Jean W. Souza *†,William H. P. Costa †,
Reinaldo L. Abreu ‡, Danilo H. Spadoti §
Institute of Systems Engineering and Information Technology,
Universidade Federal de Itajubá UNIFEI
Email: jeandesouza1211@gmail.com*, william_henrique@live.com†,
reinaldolabreu@gmail.com‡,spadoti@unifei.edu.br§

***Abstract*—This work presents an analysis of the energy consumption for a setup developed with embedded artificial intelligence to recognize the characters of an analogic hydrometer and transmitting the data via LoRa. The results showed that the developed device with image edge computing is capable of capturing the image, recognizing the specific characters in the image and transmitting. An estimated battery lifetime is 5 years and 3 months using a battery with a capacity of 10,000 [mAh], based on a daily transmission of 5 characters from the hydrometer.**

***Keywords*—Embedded Artificial Intelligence, Energy measurement, Batteries Dimensioning, Smart measurement of hydrometers.**

I. Introduction

In recent years, computing has evolved from a centralized, cloud-based model to a distributed edge model. While cloud computing has been widely adopted, it faces challenges related to security, high rates, regulation, and latency. Edge processing, an emerging paradigm that enables local processing at the network's edge before data transmission, emerges as a technology to address the challenges posed by existing models.

Edge processing offers several advantages, such as reduced data transmission rates and the preservation of information privacy. Additionally, facial recognition applications can achieve a significant decrease in response time by shifting processing from the cloud to the edge. In [1], a model was presented with an approximate reduction of 700 ms using edge processing. Furthermore, devices like security cameras can perform real-time video analysis at the edge, identifying intrusions or suspicious behaviors without the need to send video data to a central server [2].

In the context of the Internet of Things (IoT), the number of deployed edge devices has been growing exponentially, bringing significant challenges related to energy efficiency [3]. Many of these devices are deployed in remote or hard-to-reach locations, making battery maintenance or replacement a costly task. Furthermore, environmental concerns are becoming increasingly prominent, increasing the demand for devices that are more energy-efficient.

Thus, power consumption in edge processing devices remains a developing research field [4]. Researchers have been exploring optimization techniques and strategies to manage and reduce energy consumption during data transmission. They have also implemented machine learning (ML) algorithms to minimize the energy consumption of these devices.

Regarding energy consumption, studies indicate that the most significant power consumption of an edge device is associated with the use of the transceiver [5], [6], responsible for wireless communication, including data transmission and reception. Therefore, effective management of this resource can have a positive impact on reducing the energy consumption of the devices.

In this context, the present study aims to analyze the data transmission energy consumption of an edge image processing device with LoRa connectivity applied to a water meter. Moreover we compared the consumption with a pulse counter for a LoRa-connected water meter.

II. Edge Computing and LoRa Communication

The edge processing device consists of two prototyping microcontrollers from Espressif®, the DOIT-Kit ESP32 [7] and ESP32-CAM [8], along with a LoRa data transmission module, the EBYTE® E32-915T20D [9]. Figure 1 presents the prototype created using the aforementioned devices.

The objective of this prototype was to capture images of a water meter, process them using embedded artificial intelligence, extract consumption values, and transmit the recognized values via LoRa.

The software contained in ESP32-CAM is designed for taking photos and recognizing digits within the

979-8-3503-4357-1/23/$31.00 ©2023 IEEE

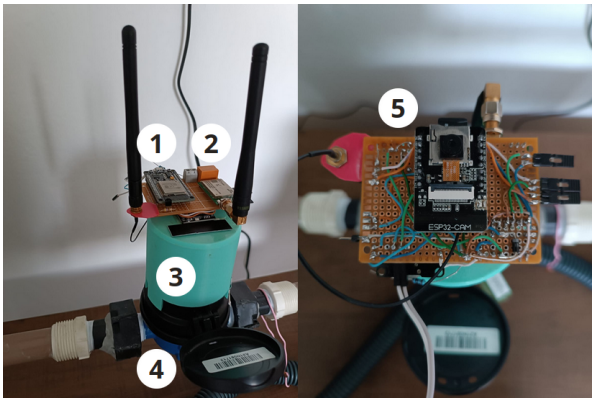

Figure 1: Prototype of the edge processing device. (1) - DOIT-Kit ESP32, (2) - LoRa E32-915T20D, (3) - Plastic adapter for camera mounting and positioning, (4) - Elster UR S120 water meter, (5) - ESP32-CAM.

images using artificial intelligence. This software was based on the [10] project, which employs the TensorflowLite tool for embedded devices [11]. Another feature of the ESP32-CAM's software is the integrated web server. It includes a website for configuring image extraction and displays in real-time the values recognized through artificial intelligence.

Figure 2 displays the diagram show the operational steps of the ESP32-CAM.

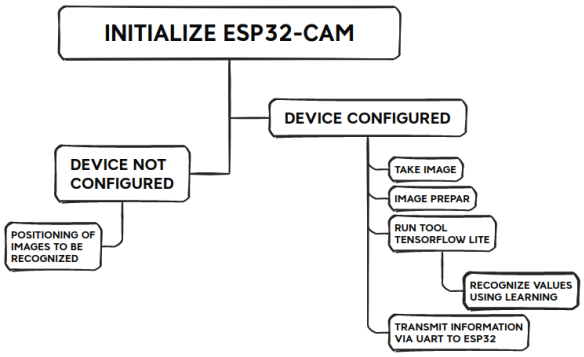

Figure 2: Diagram of the ESP32-CAM operation.

Configuration on the website is necessary to set up the image for recognition, indicating the location of the digits to be recognized. Figure 3 displays the configuration of the image's characters in order to make the digits recognizable through artificial intelligence.

Each blue square that surrounds the numbers was cropped by the software and transformed into a numerical matrix with dimensions of 16x16 pixels and 3 chanel RGB. Using the Convolutional Neural Network (CNN) technique [12], integrated into the TensorFlow Lite tool, patterns were highlighted, simplifying the

Figure 3: Configuration of the numbers to be recognized by artificial intelligence.

recognition process through embedded learning [13].

After the initial configuration, all photos were captured in the same position. Subsequently, the images were processed, including cropping and treatment, using Convolutional Neural Network (CNN) techniques, and the resulting numerical values were transmitted via UART to the second ESP32 microcontroller.

The software on the second microcontroller, DOIT-Kit ESP32, was employed for tasks related to managing the power consumption of the ESP32-CAM device and the LoRa transmitter, as well as processing and storing data.

Figure 4 displays a diagram illustrating the operational steps of this device.

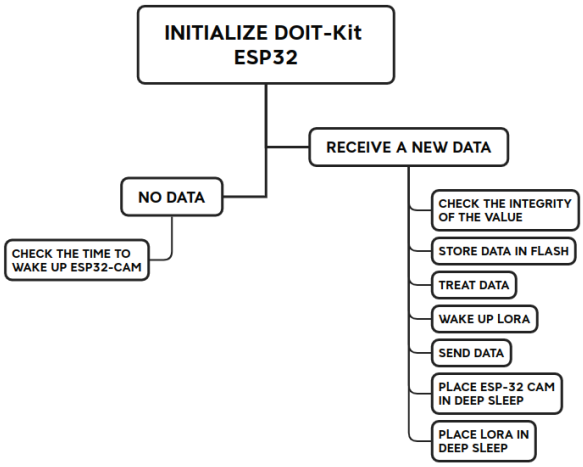

Figure 4: Diagram of the operation of the second DOIT-Kit ESP32 device.

When the device is initialized, it waits for the arrival of a new value via UART, which is recognized by the ESP32-CAM. In this study, we assumed that the data should follow an ascending order of values. Therefore, the integrity verification procedure consisted of evaluating whether the current value is greater than the previous one. If not, the value is discarded. In cases in which the value proved to be valid, it was stored in the device's EEPROM memory. Subsequently, the values are encapsulated and transmitted using LoRa technology. At the end of the transmission process, the

ESP32-CAM and LoRa modules are put into deep sleep mode.

III. Power Consumption Model

To reduce the application's power consumption, the following strategies were adopted:

1) Disable the web server functionality on the device responsible for image capture and processing.
2) Transmit data via LoRa.
3) Place the device responsible for image capture and processing, as well as the LoRa transmitter, into deep sleep mode through the data processing device.
4) Put the device that handles data processing into deep sleep mode.

To calculate the device's transmission power consumption, we adopted the methodology of parameterizing the entire application at a single common connection point for the entire system. Subsequently, we distinguished the operational phases based on the current-time relationship.

In this way, the operational stages of the device were divided into:

1) Operating Stage: When the ESP32-CAM initializes the *web server*, takes a photo, and performs image processing to recognize the values in the image, the second ESP32 initializes data capture and transmission;
2) Deep Sleep Mode: This is the stage where the second ESP32 turns off the ESP32-CAM and the LoRa module and enters a deep sleep state to conserve energy.

A. Data Acquisition

To device data collection, the circuit was assembled as shown in the block diagram in Figure 5.

The circuit was constructed using a 5 [V], 1 [A], 5 [W] DC power supply and a 1 [Ohm] shunt resistor with 0.1% accuracy in series with the device. With this configuration, it is observed that the voltage drop is equivalent to the current in the circuit.

Data was collected using an oscilloscope, generating a curve that relates the voltage (V_s) across the shunt resistor. The same voltage value can be related to the current, since the resistor has a value of 1 [Ohm].

Figure 6 displays the data transmission graph. The values on the y-axis represent the instantaneous current in Amperes on the device, and on the x-axis is the sampling time, with each sampling interval being 200 ms.

The equation that converts the sampling time into seconds per sample can be given by:

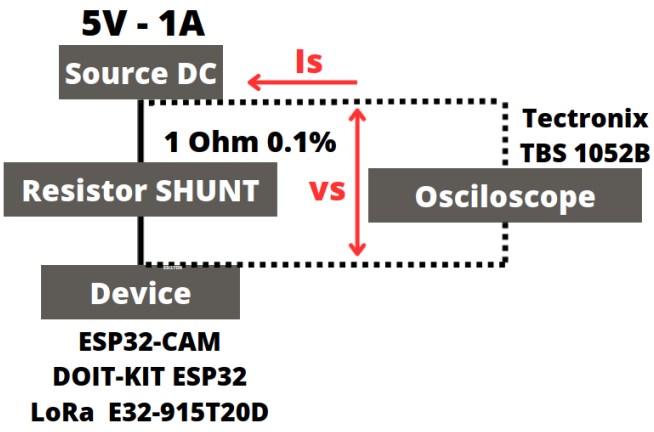

Figure 5: Configuration for voltage and current data acquisition that power the system, figure by the author.

$$TIME = \Delta_{samples} \times 0.2 \quad (1)$$

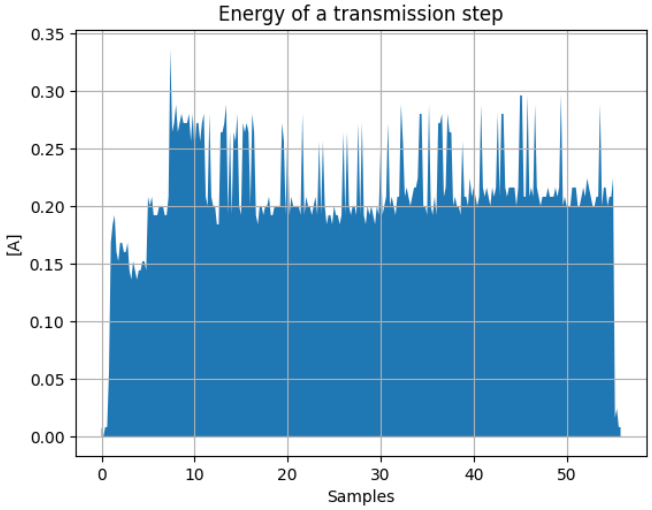

Figure 6: Power consumption of the edge processing device.

From the circuit illustrated in Figure 5, the energy of the device could be calculated by integrating the current curve and multiplying this value by the supply voltage. For this purpose, the numerical integration method of Simpson's Rule [14] was employed through code developed in the Python programming language. Simpson's Rule is a method that performs point-to-point approximation to find an integrable function, where the points of the vector are taken as the function itself, and the sum of least squares is calculated to determine the area.

Thus, to determine the area under the curve of a function, the sampled values were used in vector form. In the present study, the integration step was set to be equal to the sampling period (0.2 seconds), which corresponds to the same value used on the oscilloscope scale.

The calculation of the device's power consumption was performed in milliampere-hours [mAh]. The choice of this unit of measurement was adopted considering that most batteries available in the market use this unit of measurement, instead of Watts, used by the International System of Units (SI). The first step consisted of determining the area under the curve obtained in the system measurement as represented in Figure 5. The resulting value of this area was expressed in Amperes multiplied by seconds. Specifically, the integral of Figure 6 resulted in a value of 11.78 [A.s], divided by 3600 [s], resulting in $3.27 \cdot 10^{-3}$ [Ah], which is equivalent to 3.27 [mAh].

Table I presents the results of tests conducted with different data transmissions:

Table I: Consumption for different data processing and transmission

Quantity	Time[s]	Capacity [mAh]	Consumption [mWh]
1	59,0	3,52	0,618
2	54,6	3,13	0,627
3	56,6	3,24	0,649
4	55,8	3,27	0,654
5	54,4	3,29	0,659

The first row presents the time of transmission and the energy consumed in milliampere-hours and milliwatt-hours to capture a photo of the water meter, perform numerical value recognition from the image via artificial intelligence, and transmit the result of a single character via LoRa. The energy consumption at this instant is lower because only one character is being sent, but the time is longer because there are more functionalities involved than in the subsequent transmissions of 2, 3, 4, and 5 characters, where photo capture is not performed, only recognition, processing, and transmission. This time varies slightly between 54.4 [s] and 56.6 [s], but the consumption increases as more characters are transmitted.

B. Battery Lifetime Estimation

In the calculation of the device's lifespan estimate, different batteries were considered, each with its respective energy capacity expressed in [mAh]. This metric allows evaluating the amount of current each battery can provide over a one-hour period.

By considering the energy capacities of the batteries, it was possible to conduct an analysis to determine how long the device can operate before requiring recharging or battery replacement.

Table II: Specification of the batteries used in this study

Bat.	Modelo ou Marca	Capacidade [mAh]	Ref.
1	DC-05680	6800	[15]
2	I2GO-Pro	10000	[16]

In Table 7, two lithium-ion batteries were presented, which were used as a reference to evaluate the device's power consumption. The battery in line 1, model DC-05680, has a capacity of 6800 [mAh], and in line 2, a capacity of 10000 [mAh]. Both have charge control circuitry and can be charged via USB."

The calculation of the battery's estimated lifespan was performed by relating the device's power consumption to the battery's capacity in equivalent physical quantities. To do this, the battery's capacity was divided by the device's power consumption. This division resulted in the estimated time during which the battery can provide energy to the device before requiring recharging or replacement.

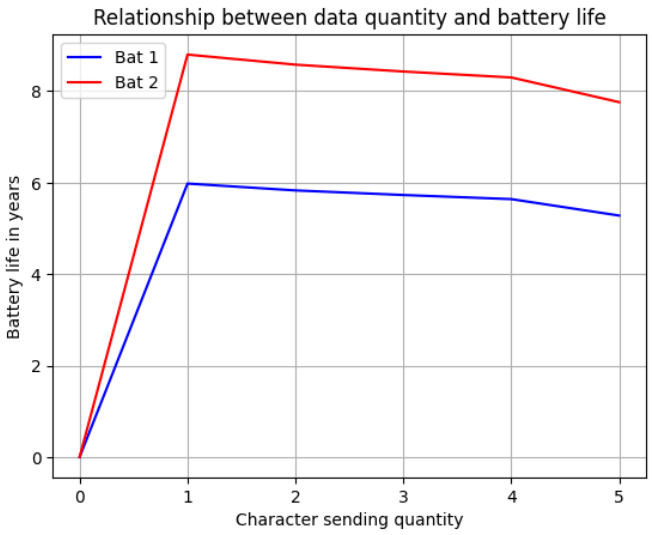

Figure 7: Data transmission rate and battery lifespan in years, author's own figure

The graph presented in Figure 7 illustrates the relationship between the amount of data sent (x-axis) and the time in years, considering one transmission per day.

To perform a comparison with a device that has a similar function, the "KHOMP ITC 100 IoT Water and Gas Meter" from KHOMP [17] was selected. Based on the data provided in the datasheet [17], it was found that this device has a transmission current of 172 [mA] and an average operating power of 0.26 [W].

One of the advantages of the edge processing device in this work is the flexibility to install and configure the equipment in various existing models of water meters.

The edge processing device has an average current of 206 [mA] during transmission, which is 19.7% higher than the current of the "KHOMP ITC 100 IoT Water and Gas Meter." However, the device in this paper can be applied to a common water meter and perform image capture and processing using artificial intelligence, as well as data transmission via LoRa. In comparison to the ITC100, which requires a pre-equipped water meter for telemetry and a "red-switch" sensor accessory to perform pulse counting.

IV. Conclusion

The smart edge processing device offers several advantages when compared to commercial meters like the ITC100. It allows for the utilization of the existing water meter and, through image capture, reads the meter's register and transmits the data via the LoRaWAN protocol.

The objective of this work was to conduct an analysis of the energy consumption of the device, considering the interval during which the device woke up from sleep mode, captured an image, processed the image using a convolutional neural network, obtained the reading from the water meter's register, and transmitted the result via the LoRa protocol."

The current consumption obtained during transmission was 206 [mA], which, although 19.7% higher than the current of a commercial meter like the KHOMP ITC100, proved to be quite promising due to the advantages that the device offers by avoiding the need to replace the water meter with a model suitable for pulse reading.

The duration period of two commercial batteries with capacities of 6800 [mAh] and 10000 [mAh] was analyzed, resulting in approximately 5.28 years (around 5 years and 3 months) for the 6800 [mAh] battery with a 5-character transmission and up to approximately 8.8 years (around 8 years and 9 months) for the 10000 [mAh] battery with a 1-character transmission. These are quite reasonable results for IoT device use cases.

The experiments conducted in this study were based on prototype boards, which include extra components that are not used and consume power, thus reducing the energy efficiency of the device and, consequently, affecting the battery sizing.

For future work, it is recommended to use low-power and high-precision microcontrollers, such as the *STM32 L0* family of microcontrollers, and employ *Very Low Drop (LDO)* voltage regulators. These technologies enable a significant reduction in energy consumption during the device's periods of inactivity, and further contribute to energy efficiency and battery life.

References

[1] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *2015 Third IEEE workshop on hot topics in web systems and technologies (HotWeb)*. IEEE, 2015, pp. 73–78.

[2] J. Barthélemy, N. Verstaevel, H. Forehead, and P. Perez, "Edge-computing video analytics for real-time traffic monitoring in a smart city," *Sensors*, vol. 19, no. 9, p. 2048, 2019.

[3] T. Leppänen and J. Riekki, "Energy efficient opportunistic edge computing for the internet of things," in *Web Intelligence*, vol. 17, no. 3. IOS Press, 2019, pp. 209–227.

[4] G. Kaur and R. S. Batth, "Edge computing: Classification, applications, and challenges," in *2021 2nd International Conference on Intelligent Engineering and Management (ICIEM)*. IEEE, 2021, pp. 254–259.

[5] X. Zhou and H. Mo, "Evaluation of individual demands for parking lot based on type-2 fuzzy sets," in *2017 Seventh International Conference on Information Science and Technology (ICIST)*, 2017, pp. 345–350.

[6] D. C. Harrison, D. Burmester, W. K. Seah, and R. Rayudu, "Busting myths of energy models for wireless sensor networks," *Electronics Letters*, vol. 52, no. 16, pp. 1412–1414, 2016.

[7] Espressif, "Doit kit esp32 series user manual," Datasheet, Available in: https://www.espressif.com/sites/2/datasheet/en.pdf. Accessed in: 25 Maio 2023.

[8] ——, "Esp32 series user manual," Datasheet, Available in: https://www.espressif.com/sites/2/datasheet/en.pdf. Accessed in: 18 Maio 2023.

[9] E. I. A. Expert, "E32-915t20d user manual," Available in: https://github.com/pksuf07/E32-915T-20D. Accessed in: 14 Abril 2023.

[10] J. Müller, "Water meter measurement system," AI on the Edge Device, Available in: https://github.com/jomjol/water-meter-measurement-system. Accessed in: 18 Maio 2023.

[11] S. Main, "Tensorflow lite micro with ml acceleration," Blog, Available in: https://blog.tensorflow.org/2023/02/tensorflow-lite-micro-with-ml-acceleration.html. Accessed in: 30 Agosto 2023.

[12] P. P. Mohit Sewak, Md. Rezaul Karim, *Practical Convolutional Neural Networks: Implement advanced deep learning models using Python*. Packt Publishing, 2018, vol. 1.

[13] C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*. Springer, 2012, vol. 2.

[14] R. L. Burden and J. D. Faires, *Numerical Analysis*. Ninth Edition, 2011, vol. 9.

[15] BHGate, "Lítio-íon bateria dc 5v 6800mah," Datasheet. Available in: https://pt.dhgate.com/product/portable-super-capacity-rechargeable-lithium/386843407.html. Accessed in: 13 Abril de 2023.

[16] i2GO, "Bateria recarregável i2go-pro 10000mah," Datasheet. Available in: https://www.i2go.com.br/produto/carregador-portatil-power-bank-i2go-10000mah-2-saidas-usb-preto-i2go-plus-90567. Accessed in: 13 Abril de 2023.

[17] Khomp, "Endpoint lora contador para leitura de medidores de água e gás," Datasheet, Available in: https://www.khomp.com/wp-content/uploads/2020/06/ITC_100_-_PT-v17.pdf. Accessed in: 23 de Maio de 2023.

[18] P. Ruiz, "Tensorflow lite para microcontroladores," Tensorflow Org, Available in: https://www.tensorflow.org lite/microcontrollers?hl=pt-br. Accessed in: 01 Maio 2023.

[19] S. MONTAGNY, *LoRa - LoRaWAN and Internet of Things for beginners*. Université Savoie Mont Black, 2021, vol. 1.

[20] R. Sanchez-Iborra, J. Sanchez-Gomez, J. Ballesta-Viñas, M.-D. Cano, and A. F. Skarmeta, "Performance evaluation of lora considering scenario conditions," *MDPI*, vol. 1, no. 1, 2018.

[21] C. H. T, H. Y. F, E. P. M. M, F. S. L. G, and S. H. D, "Análise de consumo de energia de dispositivo classe a em rede lorawan," Master's thesis, Universidade Federal de Itajubá - UNIFEI, Iajubá, Minas Gerais, Brasil, 2022.

[22] M. Eletronics, "Ldfm datasheet," Datasheet, Available in: https://br.mouser.com/datasheet/2/389/dm00063302-1797750.pdf. Accessed in: 13 de Maio de 2023.

[23] ——. (2023) Stmicroelectronics ldf & ldfm very low drop voltage regulators. [Online]. Available: https://br.mouser.com/datasheet/2/389/dm00063302-1797750.pdf

[24] M. Yahuza, M. Y. I. B. Idris, A. W. B. A. Wahab, A. T. S. Ho, S. Khan, S. N. B. Musa, and A. Z. B. Taha, "Systematic review on security and privacy requirements in edge computing: State of the art and future research opportunities," *IEEE Access*, vol. 8, pp. 76 541–76 567, 2020.

[25] J. Zawadzki. (2018) The state of self-driving cars for everybody. [Online]. Available: https://towardsdatascience.com/the-state-of-self-driving-cars-for-everybody-29446c1c2e2c

Referências

1 ALMEIDA, R. M. A. M.; SERAPHIM, C. H. V. *Programação de sistemas embarcados – Desenvolvendo software para microcontroladores em linguagem C*. [S.l.]: Elsevier,, 2016. v. 1. 17

2 SHANG, W. et al. Named data networking of things (invited paper). In: *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*. [S.l.: s.n.], 2016. p. 117–128. 17

3 MCEWEN, A.; CASSIMALLY, H. *Designing the Internet of Thigs*. [S.l.]: Wiley, 2014. v. 1. 18, 85

4 STMICROELECTRONICS. *STM32F205xx datasheet*. STMicroelectronics, 2020. Disponível em: <https://www.alldatasheet.com/datasheet-pdf/pdf/1179054/STMICROELECTRONICS/STM32.html>. 18

5 SYSTEMS, E. *ESP32 Series Datasheet*. Espressif Systems, 2023. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>. 18, 51, 87

6 ATMEL. *ATmega32 datasheet*. ATMEL, 2011. Disponível em: <https://ww1.microchip.com/downloads/en/DeviceDoc/doc2503.pdf>. 18

7 ATMEL. *Arduino Nano 33 BLE datasheet*. ATMEL, 2023. Disponível em: <https://docs.arduino.cc/resources/datasheets/ABX00030-datasheet.pdf>. 18

8 MONTAGNY, S. *LoRa - LoRaWAN and Internet of Things for beginners*. [S.l.]: Université Savoie Mont Black, 2021. v. 1. 18, 85

9 A., G. *Building a Data Lake: A Modern Data Architecture for Big Data Analytics*. [S.l.]: O'Reilly Media, 2015. 21–28. 31, 28, 26, 45 p. 18

10 SMITH, J. Impacto da internet das coisas no consumo de energia. *Revista de Tecnologia e Inovação*, v. 5, n. 2, p. 30–45, 2020. 18

11 PATRICK, M. *The Rise of Edge Computing ans Mouser Eletronics*. Embedded World, 2018. Disponível em: <http://files.iccmedia.com/magazines/basfeb18/basfeb18-p06.pdf>. 19, 28

12 DANIEL, S.; JENNY, P. *AI at the Edge – Solving Real World Problems with Embedded Machine Learning*. [S.l.]: O'Reilly Media, Inc, 2023. v. 1. 19, 89

13 ESKANDARNIA, E. M.; AL-AMMAL, H. M. A taxonomy of smart meter analytics: Forecasting, knowledge discovery, and power management. *nternational Journal of Computing and Digital Systems*, v. 1, n. 1, 2022. 19

14 MACKENZIE, W. *Power & renewables, Tackling the opportunities and challenges surrounding decarbonisation, energy decentralisation and the transformation of the power sector*. Wood Mackenzie, 2022. Disponível em: <https://www.woodmac.com/industry/power-and-renewables/>. 20

15 YOU, Y. *Intelligent System Designs: Data-driven Sensor Calibration and Smart Meter Privacy*. 133 p. Dissertação (Mestrado) — KTH Royal Institute of Technology, Universitetsservice US-AB, Sweden, 2022. 20

16 SIGRH. *Brasil desperdiça quase quarenta por cento de toda a água potável que detém, diz estudo*. SigRH, 2021. Disponível em: <https://www.sigrh.sp.gov.br/pageitems/450/news/11781>. 20

17 MARTINELLI, F.; MERCALDO, F.; SANTONE, A. Water meter reading for smart grid monitoring. *MDPI Journal*, v. 1, n. 1, 2022. 20

18 GOMES, M. M. *STEAM: Um modelo para processamento de eventos e enriquecimento de fluxos de dados IoT na borda da Rede*. 118 f. Monografia (Tese (Doutorado)) — Universidade do Vale dos Sinos - UNISINOS, Sao Leopoldo, 2022. 21, 24

19 MEHMOOD, M. Y. et al. Edge computing for iot-enabled smart grid. hindawi - security and communication networks. *Hindawi Security and Communication Networks*, v. 1, n. 1, 2021. 21, 24

20 D, P.; K, W. Tinyml – based concept system used to analyze whether the face mask is worn properly in battery-operated condition. *MDPI Journal*, v. 1, n. 1, 2021. 21, 24

21 JUN, L. X.; YI., G. Digitisation of conventional water meters using automated number recognition. *TENCON 2021 - 2021 IEEE Region 10 Conference (TENCON)*, v. 1, n. 1, 2021. 22, 24

22 LI, Y. et al. Research on water meter reading system based on lora communication. *IEEE International Conference on Smart Grid and Smart Citie*, v. 1, n. 1, 2017. 22, 24

23 MENG, L.; CHENG, J. Research on the visual recognition method of pointer water meter reading. *IAEAC 2021 – Advanced Information Tecnology, Eletronic and Automation Control conference*, v. 1, n. 1, 2022. 23, 24

24 HAN, D.; KIM, H. A number recognition system with memory optimized convolutional neural network for smart metering devices. *Department of Electronics Engineering, College of Electrical and Computer Engineering, Chungbuk National University, Cheongju*, v. 1, n. 1, 2020. 23, 24

25 BIN, S. et al. Research on water level measurement based on image recognition for industrial boiler. *Faculty of Information Engineering and Automation*, v. 1, n. 1, 2020. 23, 24

26 T., P. M. et al. *Development of a real time energy monitoring platform user-friendly for buildings*. [S.l.]: Elsevier Ltda, 2012. v. 1. 27

27 SILVA, E. O. A. S. C. Monitoramento de energia em tempo real para usuários residenciais: uma abordagem para a redução do consumo. *Revista de Energia Sustentável*, v. 10, n. 2, p. 45–58, 2020. 28, 41

28 BODEN, M. A. *Artificial Intelligence: A Very Short Introduction*. [S.l.]: Oxford, 2018. v. 1. 28, 89

29 S., H. *Neural networks and learning machines*. [S.l.]: Pearson Education India, 2010. 21 p. 28

30 I.N., S. et al. *Introduction. In: Artificial Neural Networks*. [S.l.]: Springer Cham, 2010. 28, 29, 60

31 D., S. I. N. *Artificial neural network architectures and training processes*. [S.l.]: Springer Cham, 2017. 21–28. 21, 23, 24, 25 p. 28, 29, 30, 31, 32, 33

32 D., G. *Principles of artificial neural networks*. [S.l.]: World Scientific, 2013. v. 7. 21 p. 29

33 JOHN, S. *Fundamentos de Redes Neurais Artificiais*. [S.l.]: Editora XYZ, 2018. 50-55 p. 34, 35, 37, 38, 39

34 WANT, R. Near field communication. *IEEE Pervasive Computing*, v. 10, n. 3, p. 4–7, 2011. 42

35 BONATTO, A.; CANTO, D. O. do. Bluetooth technology (ieee 802.15). *Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)*, 2007. 42

36 H., S. E.; D., F. S. *Protocolo Zigbee*. 2018. <https://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2017_2/802154/zigbee.html>. Acessado em: 01 Setembro de 2024. 42

37 TELECO. *Wi-Fi e WiMAX I: Características do Wi-Fi*. <https://www.teleco.com.br/tutoriais/tutorialww1/pagina_4.asp>. Acessado em: 01 Setembro de 2024. 43

38 M., M. *LoRa: Saiba tudo sobre essa tecnologia de radiofrequência!* 2024. <https://victorvision.com.br/blog/lora/>. Acessado em: 01 Setembro de 2024. 43

39 E., F. *Wireless solutions #5 – SigFox. Fulfilling an IoT communications need*. 2018. <https://my.element14.com/fulfilling-an-iot-communications-need>. Acessado em: 01 Setembro de 2024. 43

40 MWAKWATA, C. B. et al. Narrowband internet of things (nb-iot): From physical (phy) and media access control (mac) layers perspectives. *NIH - National Library of Medicine*, 2019. Accessed in: 01 Setembro 2024. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6603562/>. 43

41 SENEVIRATNE MULLERIYAWA, S. L. P. *Beginning LoRa Radio Networks with Arduino: Build Long Range, Low Power Wireless IoT Networks*. [S.l.]: Apress Editora, 2019. v. 1. 44

42 SANCHEZ-IBORRA, R. et al. Performance evaluation of lora considering scenario conditions. *MDPI*, v. 1, n. 1, 2018. 44

43 J., X. et al. Narrowband internet of things: Evolutions, technologies and open issues. *IEEE Internet of Things Journal*, p. 1–1, 99, 2017. 46, 47, 48, 49

44 R., R. et al. Nb-iot system for m2m communication. *IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, p. 428–432, 2016. 47

45 T, C. H. et al. *Análise de Consumo de Energia de Dispositivo Classe A em Rede LoRaWAN*. 6 p. Dissertação (Mestrado) — Universidade Federal de Itajubá - UNIFEI, ajubá, Minas Gerais, Brasil, 2022. 49, 50

46 EBYTE. *E32-915T20D User Manual*. Ebyte, 2023. Disponível em: <https://www.f1depo.com/class/INNOVAEditor/assets/E32-915T30D_Usermanual_EN_v1.7.pdf>. 52, 82

47 SIMCOM. *SIM7020 Series AT Command Manual*. SIMCOM, 2023. Disponível em: <https://m5stack.oss-cn-shenzhen.aliyuncs.com/resource/docs/datasheet/unit/nbiot/SIM7020%20Series_AT%20Command%20Manual_V1.05.pdf>. 52, 87

48 IBM. *AT Commands*. IBM, 2023. Disponível em: <referenciarhttps://www.ibm.com/docs/pt-br/aix/7.3?topic=troubleshooting-commands>. 54, 68, 79

49 MüLLER, J. *Water Meter Measurement System*. AI on the Edge Device, Available in: https://github.com/jomjol/water-meter-measurement-system. Accessed in: 18 Maio 2023. 57, 64, 66

50 RUIZ, P. *TensorFlow Interpreter*. Tensorflow Org, Available in: https://www.tensorflow.org/api_docs/python/tf/lite/Interpreter. Accessed in: 01 Maio 2024. 62

51 BURDEN, R. L.; FAIRES, J. D. *Numerical Analysis*. [S.l.]: Ninth Edition, 2011. v. 9. 78