UNIVERSIDADE FEDERAL DA PARAÍBA
Centro de Energias Alternativas e Renováveis
Programa de Pós-Graduação em Engenharia Elétrica

Rede de Sensores Sem Fio de Longo Alcance aplicada a Reconhecimento Facial por Imagem

VÍTOR JOSÉ COSTA RODRIGUES

João Pessoa - PB
Julho de 2022

VÍTOR JOSÉ COSTA RODRIGUES

Rede de Sensores Sem Fio de Longo Alcance aplicada a Reconhecimento Facial por Imagem

Dissertação apresentada como parte dos requisitos necessários para obtenção do título de Mestre em Engenharia Elétrica do Programa de Pós-graduação em Engenharia Elétrica da Universidade Federal da Paraíba.

Área de Concentração: Sistemas Eletrônicos e Automação

Orientador: Prof. Dr. Fabrício Braga Soares de Carvalho

João Pessoa - PB
Julho de 2022

Catalogação na publicação
Seção de Catalogação e Classificação

R696r Rodrigues, Vítor José Costa.
Rede de sensores sem fio de longo alcance aplicada a reconhecimento facial por imagem / Vítor José Costa Rodrigues. - João Pessoa, 2022.
85 f. : il.

Orientação: Fabrício Braga Soares de Carvalho.
Dissertação (Mestrado) - UFPB/CEAR.

1. Redes de sensores sem fio. 2. LoRa. 3. Reconhecimento facial. 4. Sistemas de monitoramento. I. Carvalho, Fabrício Braga Soares de. II. Título.

UFPB/BC CDU 004.722:528.8(043)

Elaborado por WALQUELINE DA SILVA ARAUJO - CRB-15/514

VÍTOR JOSÉ COSTA RODRIGUES

Rede de Sensores Sem Fio de Longo Alcance aplicada a Reconhecimento Facial por Imagem

Dissertação apresentada como parte dos requisitos necessários para obtenção do título de Mestre em Engenharia Elétrica do Programa de Pós-graduação em Engenharia Elétrica da Universidade Federal da Paraíba.

Aprovado em: 28/07/2022

BANCA EXAMINADORA

Fabrício Braga Soares de Carvalho, D.Sc.
Orientador - Universidade Federal da Paraíba

Waslon Terllizzie Araújo Lopes, D.Sc.
Avaliador - Universidade Federal da Paraíba

Ruan Delgado Gomes, D.Sc.
Avaliador - Instituto Federal da Paraíba

João Pessoa - PB
Julho de 2022

UNIVERSIDADE FEDERAL DA PARAÍBA – UFPB
CENTRO DE ENERGIAS ALTERNATIVAS E RENOVÁVEIS – CEAR
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA – PPGEE

A Comissão Examinadora, abaixo assinada, aprova a Dissertação

REDE DE SENSORES SEM FIO DE LONGO ALCANCE APLICADAS A RECONHECIMENTO FACIAL POR IMAGEM

Elaborada por

VÍTOR JOSÉ COSTA RODRIGUES

como requisito parcial para obtenção do grau de
Mestre em Engenharia Elétrica.

COMISSÃO EXAMINADORA

PROF. DR. FABRÍCIO BRAGA SOARES DE CARVALHO
Orientador – UFPB

PROF. DR. WASLON TERLLIZZIE ARAÚJO LOPES
Examinador Interno – UFPB

PROF. DR. RUAN DELGADO GOMES
Examinador Externo – IFPB

João Pessoa/PB, 28 de julho de 2022

"Beneath an unsinking black sun, through the boundless gloom, our journey continues."

Kentaro Miura

Resumo

Durante as últimas duas décadas, a aplicação de tecnologia na integração de diversas áreas do cotidiano humano tem se tornado mais comum e intensa, particularmente graças ao conceito de IoT, ou Internet das Coisas, que abrange diversas abordagens para essa integração. Uma dessas abordagens são as Redes de Sensores sem Fio (RSSF), com as quais é possível monitorar vários dispositivos simultaneamente, ou realizar leituras em etapas diferentes (e fisicamente separadas) de um processo ou serviço. Através desse tipo de automação, também pode-se aplicar "inteligência" aos ambientes, como casas, estradas, prédios, complexos fabris e até mesmo cidades ou regiões metropolitanas. Uma das grandes vantagens do uso das RSSFs em aplicações IoT é sua baixa demanda de infraestrutura, permitindo redução de custos tanto logísticos quanto financeiros. Sistemas de Monitoramento por Câmera e de Autenticação Biométrica Facial, ambos exemplos de aplicações de ambientes inteligentes, tradicionalmente utilizam tecnologias cabeadas e/ou meios de transmissão de curto alcance por conta da alta densidade de informação característica das imagens. Em contraponto a isto, este trabalho se baseia no uso das Redes de Sensores sem Fio com o objetivo de implementar uma Rede de Monitoramento e Autenticação Facial, utilizando a tecnologia de longo alcance LoRa como meio de comunicação entre seus dispositivos. A rede desenvolvida é composta por quatro estágios, em uma arquitetura modular, permitindo que a escala de cada estágio possa ser ajustada às necessidades do ambiente sem afetar os demais. Através da realização de testes e medições experimentais foi possível confirmar a viabilidade do uso de RSSFs para a transmissão de imagens estáticas, em particular, para aplicações de Reconhecimento Facial Remoto.

Palavras-Chave: Redes de Sensores Sem Fio, LoRa, Reconhecimento Facial e Sistemas de Monitoramento.

Abstract

During the last two decades, the application of technology in the integration of different areas of human daily life has become more common and intense, particularly due to the concept of IoT, or Internet of Things, which encompasses different approaches to such integration. One of these approaches are the Wireless Sensor Networks (WSNs), which make possible to monitor several devices simultaneously, or to perform readings at different (and physically separate) stages of a process or a service. Through this type of automation, you can also apply "intelligence" to environments, such as houses, roads, buildings, manufacturing sites and even cities or metropolitan regions. One of the great advantages of using WSNs in IoT applications is their low infrastructure demand, allowing both logistical and financial cost reductions. Camera Surveillance and Biometric Facial Authentication Systems are both examples of applications for smart environments which traditionally use wired technologies and/or short-range transmission media due to the high density of information characteristic in the images. In contrast to this, this work is based on the use of Wireless Sensor Networks with the goal of implementing a Monitoring and Facial Authentication Network, using the long-range LoRa technology as a means of communication among its devices. The proposed network is composed of four stages, in a modular architecture, allowing the scale of each stage to be adjusted to the needs of the environment without affecting the other stages. Through experimental tests and measurements, it was possible to confirm the feasibility of using WSNs for the transmission of static images, in particular, for Remote Facial Recognition applications.

Keywords: Wireless Sensors Networks, LoRa, Facial Recognition and Monitoring Systems.

Siglas

Chirp	Pulso de Radar Comprimido de Alta Intensidade (*Compressed High Intensity Radar Pulse*)
API	Interface de Programação de Aplicações (*Application Programming Interface*)
APS	Sensores de Pixels Ativos (*Active Pixel Sensors*)
CCD	Dispositivo de Carga Acoplada (*Charge-Coupled Device*)
CDMA	Acesso Múltiplo por Divisão em Código (*Code Division Multiple Access*)
CMOS	Semicondutores Complementar de Óxido Metálico (*Complementary Metal Oxide Semiconductors*)
CRC	Checagem de Redundância Cíclica (*Cyclic Redundancy Check*)
CSS	Espalhamento Espectral *Chirp* (*Chirp Spread Spectrum*)
DNN	Redes Neurais de Aprendizado Profundo (*Deep Neural Networks*)
DSSS	Espalhamento Espectral por Sequência Direta (*Direct Sequence Spread Spectrum*)
FDMA	Acesso Múltiplo por Divisão em Frequência (*Frequency Division Multiple Access*)
FHSS	Espalhamento Espectral por Salto em Frequência (*Frequency Hopping Spread Spectrum*)
FSCM	Modulação de Deslocamento de *Chirp* em Frequência (*Frequency Shift Chirp Modulation*)
HOG	Histogramas de Gradientes Orientados (*Histograms of Oriented Gradients*)
IoT	Internet das Coisas (*Internet of Things*)
LoRa	Rádio de Longo Alcance (*Long Range Radio*)
LPWAN	Redes de Grande Área e Baixo Consumo (*Low Power Wide Area Networks*)
MOS	Óxiodo Metálico Semicondutor (*Metal Oxide Semiconductor*)
NFC	Comunicação por Campo de Proximidade (*Near Field Communication*)
OSI	Interconexão de Sistema Aberto (*Open System Interconnection*)
PIL	Biblioteca de Imageamento do Python (*Python Imaging Library*)
PIR	Passivo Infravermelho (*Passive Infrared*)

RFID	Identificadores por Radiofrequência (*Radiofrequency Identifiers*)
RNA	Redes Neurais Artificiais
RSSF	Redes de Sensores sem Fio
SSL	Camada de Sockets Segura (*Secure Sockets Layer*)
SVM	Máquinas de Vetores de Suporte (*Support Vector Machines*)
TLS	Segurança de Camada de Transporte (*Transport Layer Security*)
UMC	Unidade Microcontrolada

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

O termo Internet das Coisas (IoT - *Internet of Things*) foi criado há mais de 20 anos como um chamariz publicitário para promover o conceito de uma linha de produção automática auto-suficiente. Segundo seu criador, as falhas humanas causadas por operadores e a dependência de haver funcionários realizando fisicamente o transporte de informações entre as máquinas eram dois grandes causadores de prejuízos, tanto do ponto de vista financeiro quanto do ponto de vista temporal (Ashton, 2009).

Desde então, a popularidade do termo e do conceito que ele carrega se tornaram tão populares que o que antes era um nome que só tinha significado dentro de um contexto, agora carrega o próprio significado dentro de si: Internet das Coisas, com *Internet* em seu sentido literal, de uma rede que interliga elementos; e *Coisas* como um descritor mais genérico possível, indicando que qualquer produto, processo ou serviço poderia integrar essa tal rede. Alguns estudiosos inclusive preferem utilizar a nomenclatura "Internet de Tudo" (*Internet of Everything*) porque torna mais claro o objetivo desse conceito (Buyya e Dastjerdi, 2016).

A popularidade e o interesse sobre esse assunto são tamanhos ao ponto de estimativas indicarem que, até o fim de 2021, o número de dispositivos já conectados em IoT tenha alcançado a marca de 12,3 bilhões e até meados de 2023 ultrapasse a marca dos 15 bilhões de dispositivos conectados (Sinha, 2021).

Formalmente, IoT não é uma tecnologia ou um conjunto de tecnologias em si, mas um princípio ou mesmo uma filosofia de engenharia. Esse princípio é amplo e, de certa maneira, abstrato, o que pode tornar difícil conceituá-lo de maneira específica. Atzori, Iera e Morabito (2010) descrevem IoT como toda e qualquer tentativa de integração pervasiva de tecnologia que permeia um ou mais aspectos do cotidiano.

A ideia de dar autonomia à comunicação entre processos e dispositivos se mostrou um grande avanço para a tecnologia e gerou inúmeras aplicações e sistemas que, de outro modo, não teriam como ser implementados ou, no melhor dos casos, seriam inviáveis economicamente. Este é o caso, por exemplo, das RSSF, em que vários sensores realizam medidas periódicas de grandezas de interesse e transmitem esses dados remotamente para uma central de processamento. Uma implementação dessas é completamente dependente da autonomia na comunicação entre esses dispositivos, especialmente considerando que algumas redes podem chegar a conter centenas ou até mesmo milhares de sensores conectados.

Como é o caso com todo tipo de sistema baseado em dispositivos embarcados, os projetos desses sistemas IoT, com as RSSF inclusas, sempre levam em consideração a relação entre desempenho e vida útil, já que sistemas embarcados inevitavelmente lidam com restrições em recursos. Uma área na qual o gerenciamento desses recursos limitados é crucial é a de comunicações sem fio, especialmente no caso das RSSF, nas quais elas são imprescindíveis.

Um dos recursos de grande importância em um dispositivo sem fio é sua fonte de energia, já que é ela que permite que os demais recursos operem. Seja o caso de baterias ou de dispositivos de geração própria, em ambos os casos, por razões diferentes, estas fontes de energia possuem capacidade limitada. Com isso, torna-se praticamente um pré-requisito para sistemas sem fio que estes otimizem e reduzam seu consumo o máximo possível. Dessa forma, as tecnologias de baixo consumo (*Low Power*) dominaram o mercado dos sistemas sem fio, por se mostrarem muito mais eficientes a longo prazo ao apresentarem uma maior vida útil, o que garante autonomia aos dispositivos por mais tempo.

Essas tecnologias têm por objetivo fornecer o melhor desempenho possível considerando um consumo bastante reduzido. Isso tradicionalmente foi atingido limitando o alcance de tais comunicações, como é o caso dos Identificadores por Radiofrequência (RFID - *Radiofrequency Identifiers*), da Comunicação por Campo de Proximidade (NFC - *Near Field Communication*) e de algumas tecnologias *Bluetooth*. Isso se dá pois o alcance de uma transmissão está diretamente relacionado com sua potência e, portanto, também relacionado com o consumo de energia. Essa abordagem é bastante eficiente no gerenciamento energético, porém restringe bastante o tipo de aplicação que a utiliza, já que os dispositivos precisam estar a distâncias muito curtas entre si, por vezes precisando serem colocados um ao lado do outro para se comunicarem.

Uma nova abordagem foi desenvolvida na última década buscando, ao abrir mão das altas taxas de transmissão, permitir uma transmissão de grande alcance sem que haja um consumo elevado. As Redes de Grande Área e Baixo Consumo (LPWAN - *Low Power Wide Area Networks*), como são chamadas, obtêm um alto alcance de transmissão em um contexto de baixo consumo ao sacrificarem sua capacidade de taxa de transmissão. A tecnologia *Sigfox*, por exemplo, investe fortemente no quesito de longa distância, possuindo um alcance nominal de até 50 km, mas com isso, sacrifica sua taxa de transmissão, limitada a apenas 100 bps. Por sua vez, a tecnologia *LoRa*, um segundo exemplo de LPWAN, possui uma abordagem de equilíbrio, balanceando tanto o interesse no alcance quanto na taxa de transmissão, com taxas de dados na ordem de dezenas de kbps e com um alcance nominal na faixa de 10 km (Medeiros, 2018).

Dessa forma, muito embora o *Sigfox* apresente um alcance muito maior, o *LoRa* se mostrou uma tecnologia eficiente no quesito de taxa de transmissão, no que compete às LPWAN. Esse tipo de tecnologia, menos restrita, abre a possibilidade de desenvolvimento de RSSF para aplicações mais diversas, em particular aquelas que possuam um fluxo de dados mais denso, como é o caso de sistemas de sensoriamento de imagem, visto que imagens são dados com alto potencial de atingir tamanhos elevados, considerando a característica quadrática/bidimensional destes.

1.1 Motivação

Considerando o potencial de aplicação do *LoRa* em sistemas com maior densidade informacional, sua aplicabilidade em um contexto de sistemas embarcados realizando transmissões de imagem via LPWAN pode ser verificada em Rodrigues (2019). A partir desse resultado, foi possível levantar a hipótese da possibilidade de implementação de uma RSSF, não somente contendo múltiplos dispositivos operando simultaneamente, mas também podendo essa quantidade de dispositivos não ser definida previamente, mas poder ser ajustada conforme o local na qual é utilizada. A hipótese leva em conta as mesmas condições de tecnologia utilizadas no sistema ponto-a-ponto visto em Rodrigues (2019), ou seja, levanta a aplicabilidade de um sistema de transmissão remota de imagens utilizando tecnologia LPWAN, além das características específicas das redes de sensores.

A expectativa inicial dessa hipótese era obter um resultado, minimamente análogo ao encontrado em Rodrigues (2019). Esta Dissertação, portanto, caracteriza-se como a verificação dessa hipótese em termos práticos, através do planejamento, desenvolvimento e validação da RSSF idealizada. Como são descritos em capítulos posteriores deste texto, os resultados desta verificação se mostraram bem superiores ao esperado (um resultado de desempenho similar), demonstrando uma menor ocorrência de imagens corrompidas e uma redução geral de cerca de 72% no tempo de transmissão.

1.1.1 Objetivo Geral

No presente trabalho, tem-se como objetivo geral o desenvolvimento e a implantação de uma Rede Remota de Monitoramento e Autenticação Facial baseada em Redes de Sensores sem Fio e na tecnologia de Baixo Consumo LoRa, projetada de maneira modular para poder ser ajustada ao ambiente no qual é implementada. A implementação inicial dessa rede se deu nas dependências do Campus I da Universidade Federal da Paraíba, na cidade de João Pessoa.

1.1.2 Objetivos Específicos

Como objetivos específicos aqui propostos, têm-se:

- Estruturar o fluxo de dados da RSSF proposta com base na aplicação de monitoramento remoto (Rodrigues, 2019);
- Gerenciamento de transmissão no *gateway* para controlar vários nós sensores;
- Projetar e desenvolver implementações de *software* com base no fluxo definido, considerando as necessidades básicas de uma RSSF;
- Validação do funcionamento da RSSF com a nova arquitetura;
- Produção de uma documentação detalhando o sistema.

1.1.3 Organização do Texto

O restante deste trabalho de Dissertação está organizado da seguinte maneira: no Capítulo 2 são apresentados alguns fundamentos teóricos relativos às redes de sensores sem fio, às redes LPWAN, à biometria facial e a alguns padrões e protocolos de rede, todos utilizados na elaboração deste trabalho; por sua vez o Capítulo 3 descreve a metodologia utilizada no desenvolvimento e implementação do sistema; o Capítulo 4 discute os resultados obtidos na implementação da rede proposta; por fim, o Capítulo 5 contém as considerações finais sobre o trabalho.

2 Fundamentação Teórica

Neste capítulo são apresentados algumas definições e conceitos pertinentes ao que foi desenvolvido no trabalho. Os temas aqui abordados são as redes de sensores sem fio, as redes LPWAN, a biometria facial e alguns padrões e protocolos de rede utilizados ao longo do sistema.

2.1 Redes de Sensores Sem Fio

De certo modo, a Instrumentação é o conhecimento que fundamenta toda criação e manufatura modernas. Para obter processos e produtos com qualidade otimizada e padronizada, o primeiro passo é poder verificar essa qualidade. E todo processo de verificação objetiva envolve, direta ou indiretamente, uma etapa de sensoriamento. É através dessa etapa que se pode observar e conhecer as características de um processo e/ou produto. E conhecer essas características é o pré-requisito fundamental para poder controlá-las.

Sensores, então, podem ser encontrados ao longo da criação de praticamente tudo, quase sempre em múltiplas instâncias. Uma problemática relativa a sensores, no entanto, é que estes tipicamente sempre estiveram associados a uma infraestrutura, já que os sensores só possuem utilidade prática se seus dados puderem ser lidos o que, para sistemas cabeados, implica na conexão física entre o elemento que realiza as medições e o elemento que as interpreta. Isso se torna um problema bastante incômodo especialmente em contextos onde vários sensores precisam ser utilizados em conjunto e de maneira simultânea. Nas últimas décadas, com o avanço das tecnologias de comunicação sem fio e com um interesse crescente no ramo da IoT, tornou-se viável o desenvolvimento das chamadas Redes de Sensores Sem fio (Burati et al., 2009).

De maneira geral, essas redes podem ser definidas como redes sem infraestrutura, capazes de monitorar determinadas condições em um determinado ambiente. Elas são compostas por um conjunto de sensores, tipicamente chamados de "nós sensores", realizando medições independentes em um determinado ambiente, que pode ser uma região delimitada e bem definida amostrada por pontos principais ou um conjunto de pontos de interesse independentes. De qualquer modo, os dados desse Campo de Sensores convergem a um ponto central, denominado de "nó sorvedouro", que concentra os dados e pode distribuí-los para as aplicações de acordo com a necessidade (Burati et al., 2009; Matin e Islam, 2012).

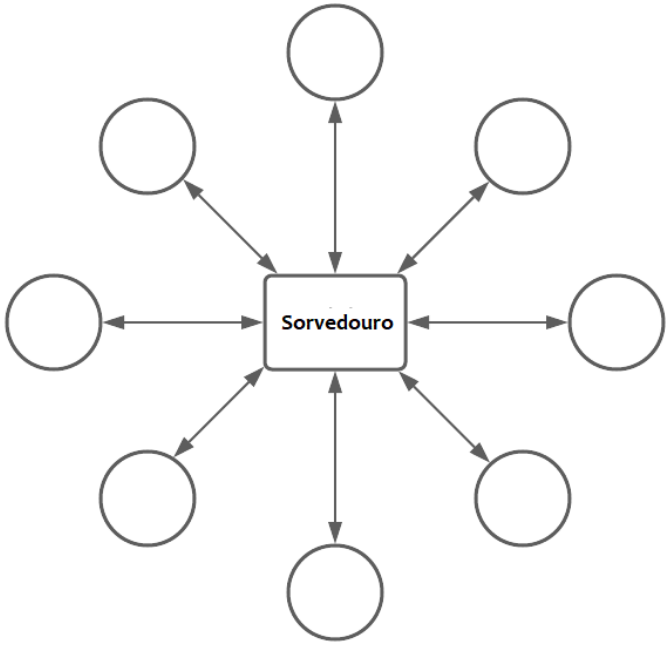
A definição de um nó sorvedouro é bastante vaga pois suas características podem variar entre cada implementação de RSSF, com a única característica fixa de que eles têm o papel de receber os dados dos nós sensores. Não é nem mesmo necessário que uma RSSF contenha apenas um nó sorvedouro, podendo redes complexas conterem vários destes nós e até mesmo camadas de nós sorvedouro encadeadas. Os nós sensores, por outro lado, apresentam uma constituição recorrente, podendo sempre serem definidos como um dispositivo composto por quatro elementos com funções bem definidas: *sensoriamento*, *processamento*, *comunicação* e *alimentação*. Esses nós podem ter mais de um elemento realizando cada função, ou elementos realizando mais de uma função, mas necessariamente terão elementos para realizar cada uma delas (Gupta e Sinha, 2014).

A hierarquia estrutural da RSSF pode variar dependendo da aplicação fim e da tecnologia empregada nos nós sensores. Por conta disso, diversas topologias são tipicamente empregadas na implementação dessas redes. Alguns dos exemplos de topologias mais comuns são as redes em árvore, em anel, em estrela e em malha (Sharma, Verma e Sharma,

2013). No desenvolvimento deste trabalho, utilizou-se a topologia de RSSF do tipo estrela.

A topologia em estrela (Figura 2.1) tem como característica principal a disposição dos nós sensores de maneira irradiante em relação ao nó sorvedouro. Apesar de similar à estratégia adotada pela topologia em árvore (que por vezes é descrita como um subtipo específico de topologia em estrela), a topologia em estrela não estabelece nenhum tipo de hierarquia entre nós sensores, estando todos eles conectados diretamente ao nó sorvedouro ou a algum concentrador intermediário (em caso da existência de múltiplos concentradores, estes também tipicamente são organizados em topologia estrela ao longo da rede).

Figura 2.1: Exemplo de topologia em estrela.



FONTE: O Autor (2022)

Dessa forma, todos os nós estão conectados diretamente ao nó sorvedouro (mesmo que por intermédio de um concentrador), o que elimina por completo o problema de queda parcial ou total da rede por falhas de comunicação observado em topologias como a árvore e o anel, visto que qualquer falha por comunicação que um nó sensor apresente irá afetar somente a sua própria comunicação (Sharma, Verma e Sharma, 2013). Outra vantagem associada com a topologia estrela é a menor dependência de dispositivos no fluxo de dados da rede em geral, por conta da comunicação direta entre sensores e receptores e da não necessidade de sensores intermediários realizarem papéis de repetidores.

Por outro lado, um aspecto negativo da topologia estrela é a necessidade de que todos os nós se comuniquem com um nó sorvedouro, sem a existência de nós intermediários. Isso aumenta os problemas associados com interferências de transmissões simultâneas, já que como todos os nós sensores concorrem pelo mesmo canal de transmissão, métodos de transmissão paralela não são possíveis de ser utilizados na rede.

2.1.1 LPWAN

Um dos critérios definidores de uma RSSF é a presença de um sistema de comunicação sem fio. A presença de tal tecnologia permite que uma rede de sensores torne-se significativamente menos complexa ao não necessitar de uma infraestrutura física conectando

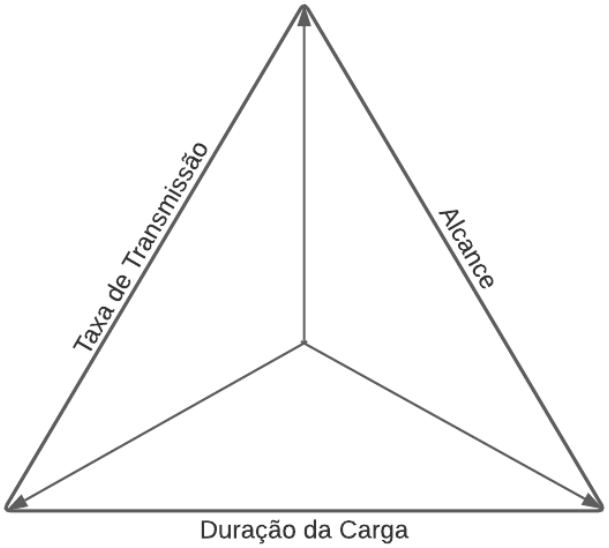
todos os seus elementos. A ausência dessa infraestrutura, no entanto, significa que os dispositivos remotos não poderão ser alimentados diretamente pelo sistema, de modo que isso exige uma certa auto-suficiência energética de cada dispositivo.

Existem múltiplas abordagens para esse problema, desde as mais complexas, como utilizar colheita de energia (Souza et al., 2016), até as mais simples, como simplesmente posicionar os nós sensores em locais com acesso à rede elétrica. Essas soluções, no entanto, têm sua viabilidade restrita a contextos específicos. A solução geral para a questão de alimentação de nós em uma RSSF é o uso de baterias.

Baterias são dispositivos de uso finito (por mais que sejam recarregáveis) e que ocupam um volume físico relativo à sua capacidade de armazenamento. Dessa forma, seu uso é um fator de restrição importante a ser considerado, tipicamente sendo o primeiro aspecto a ser levado em consideração ao se projetar tecnologias sem fio.

Por conta dessa quantidade limitada de energia disponível, dispositivos embarcados tipicamente são desenvolvidos considerando estas restrições. Desse modo, não é possível obter o melhor desempenho em todos os fatores de uma tecnologia visto que muitos deles disputam os recursos do sistema, como é o caso da taxa de transmissão, do alcance de sinal e da duração da carga de bateria. Nestes sistemas, com recursos limitados, o desenvolvimento leva inteiramente em consideração quanto investir em cada um dos fatores, já que aumentar um ou dois fatores indiscriminadamente irá afetar o quanto é possível dispor dos demais. Na Figura 2.2, é possível visualizar a ideia de que a restrição de recursos serve como uma forma de auto-balanceamento destes (Saad et al, 2014).

Figura 2.2: Três fatores concorrentes em sistemas de comunicação sem fio.



FONTE: O Autor (2022)

Dessa forma, não é viável em termos de economia energética investir tanto em alcance quanto em velocidade (já que ambos os fatores aumentam a potência de transmissão e, portanto, o consumo de energia da bateria). Como a questão energética é quase sempre prioridade nas RSSF, em situações onde um maior alcance se faz necessário, a restrição vai inevitavelmente estar associada com a velocidade da transmissão. As soluções de comunicação que buscam essa direção, priorizando tanto alto alcance de transmissão quanto baixo consumo de operação, são denominadas LPWAN.

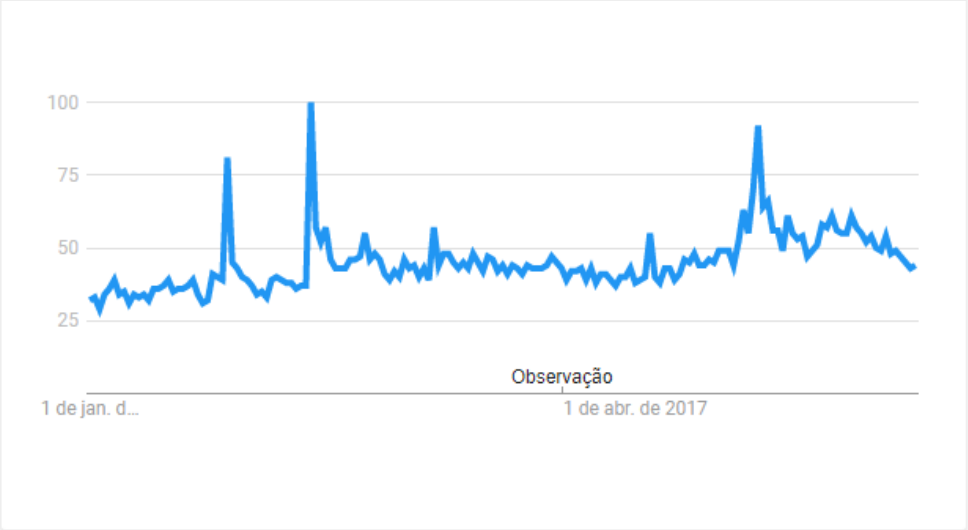
Tecnologias de comunicação com essa abordagem vêm surgindo na última década,

especialmente impulsionadas pelo aumento no interesse de IoT e seu potencial. Nessas tecnologias, em determinadas condições, é possível alcançar uma duração de carga de até 10 anos, ou um alcance de transmissão de até 40 km (Mekki et al., 2018). Dentre essas tecnologias, uma que tem tido sucesso em larga escala e uma grande popularidade é a tecnologia do Rádio de Longo Alcance (LoRa - *Long Range Radio*).

2.1.1.1 LoRa

A tecnologia LoRa foi desenvolvida em meados de 2008 pela empresa *Cycleo*, no polo tecnológico de Grenoble, França. Desde sua criação, sua relevância vem apenas aumentando. De acordo com estatísticas do Google (Figura 2.3), entre o intervalo de Janeiro de 2008 e Janeiro de 2022, vê-se três picos de interesse no termo de acordo com o gráfico apresentado. O mais recente, em Maio de 2019, época em que ocorreu a conferência IoT World 2019, seguido de picos menores nos anos seguintes, sugere que o LoRa ainda tem sido muito visado em equipamentos e soluções para IoT.

Figura 2.3: Interesse ao longo do tempo para o termo “LoRa”.



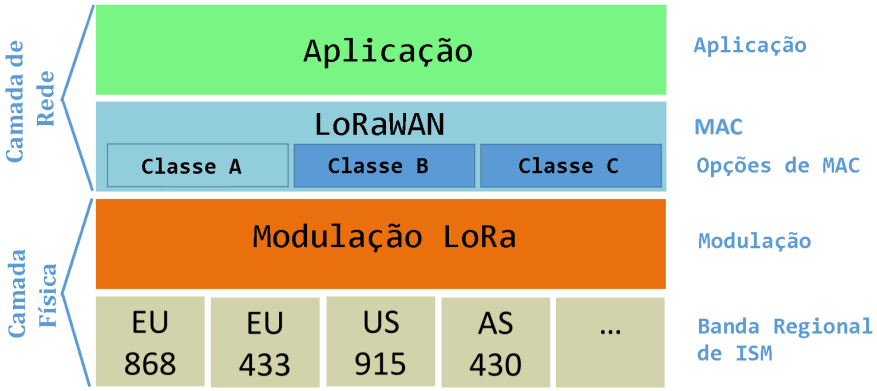
FONTE: Google Trends (2022)

Desde 2012, o LoRa tornou-se propriedade intelectual da empresa Semtech, uma das fundadoras da LoRa Alliance, uma organização sem fins lucrativos dedicada ao desenvolvimento e difusão das redes LPWAN para IoT (LoRa Alliance, 2019).

Segundo Augustin et al. (2016), pode-se definir um sistema LoRa como uma composição de duas camadas de operação: sendo uma camada física, a modulação LoRa (e seu *Hardware* associado), e uma camada de rede, a LoRaWAN. Na Figura 2.4, exemplifica-se a divisão proposta, na qual os elementos na metade inferior representam a camada física e os elementos na metade superior representam a camada de rede.

Em termos de uma analogia simplificada ao Modelo OSI, o que a LoRa Alliance define em sua documentação como “Camada Física” engloba tanto as camadas física quanto de enlace de dados do modelo OSI, enquanto que sua “Camada de Rede” engloba as cinco demais camadas do modelo.

Figura 2.4: Camadas de operação do LoRa.



FONTE: Adaptado de LoRa Alliance (2015)

2.1.1.1.1 Camada Física

A camada física, desenvolvida e patenteada pela Semtech, tem como objetivo uma comunicação sem fio de baixa potência e longas distâncias, baseada em uma técnica de modulação de espalhamento espectral denominada Espalhamento Espectral *Chirp* (CSS - *Chirp Spread Spectrum*).

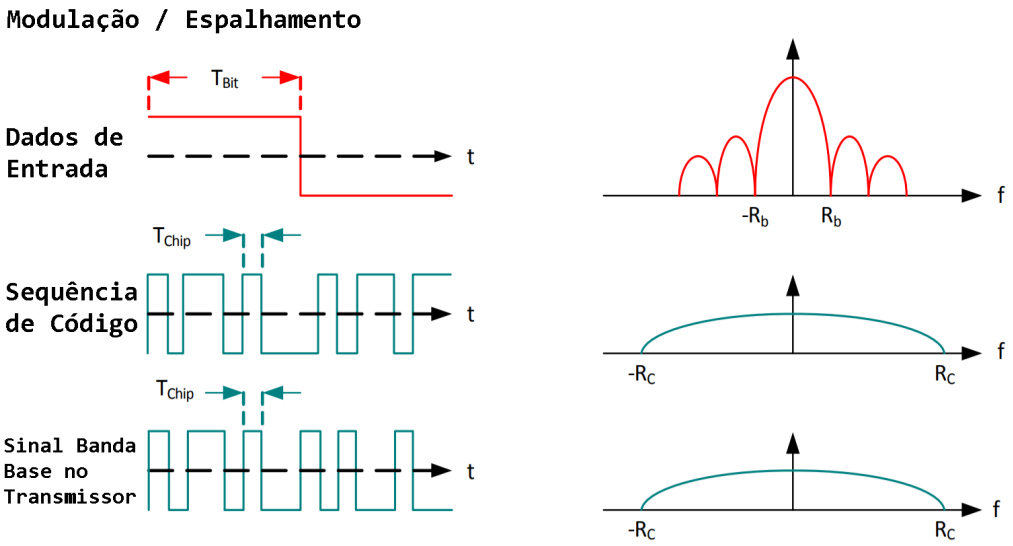
O CSS é uma técnica de espalhamento espectral baseada em ondas do tipo Pulso de Radar Comprimido de Alta Intensidade (*Chirp - Compressed High Intensity Radar Pulse*), que são essencialmente ondas de pulsos variantes em frequência ao longo do tempo. Dentre as vantagens de se utilizar ondas *Chirp* em conjunto com o espalhamento espectral, pode-se mencionar uma grande imunidade a ruído de canal, uma melhor eficiência energética e um maior alcance na transmissão (Elshabrawy e Robert, 2018).

O espalhamento espectral, por sua vez, é um tipo de técnica de modulação que é definida por expandir o sinal de interesse, com uma largura de banda de informação R_b de forma a ocupar uma largura de banda de transmissão R_c muito maior. Estas técnicas são a base do funcionamento de sistemas em Acesso Múltiplo por Divisão em Código (CDMA - *Code Division Multiple Access*). Comparado a um sistema em Acesso Múltiplo por Divisão em Frequência (FDMA - *Frequency Division Multiple Access*), em que uma largura de banda disponível R_c seria dividida em N canais de largura $R_b = R_c/N$ e cada usuário seria alocado a um canal R_b, em um sistema com espalhamento espectral cada usuário poderia ocupar qualquer parte do espectro (ou todo o espectro) durante uma transmissão, com mais de um usuário transmitindo simultaneamente (Haykin e Moher, 2008).

Ainda segundo Haykin e Moher (2008), duas das técnicas de espalhamento espectral mais utilizadas são o Espalhamento Espectral por Sequência Direta (DSSS - *Direct Sequence Spread Spectrum*) e o Espalhamento Espectral por Salto em Frequência (FHSS - *Frequency Hopping Spread Spectrum*). Em geral, técnicas de espalhamento espectral possuem maior tolerância à interferência, baixa probabilidade de detecção ou interceptação, maior tolerância a multipercursos e uma maior capacidade de alcance. Um exemplo de espalhamento espectral pode ser observado na Figura 2.5.

A quantidade de espalhamento obtida em um processo de modulação DSSS pode ser modelada em função da razão entre a taxa de variação da sequência de código (ou *chip sequence*) e a taxa de *bits* do sinal mensagem, como mostrado na Equação (2.1):

Figura 2.5: Processo de espalhamento de sinal.



FONTE: Adaptado de Semtech (2015)

$$G_p = 10 \cdot \log_{10}\left(\frac{R_c}{R_b}\right) \quad [dB], \tag{2.1}$$

em que G_p representa o ganho de processamento, expresso em dB, R_c é a taxa de variação do código (*Chip rate*), expressa em $chips/s$ e R_b é a taxa de *bits* da modulação (*Modulation bit rate*), expressa em $bits/s$.

A modulação CSS, apesar de ser baseada no espalhamento espectral, não é similar às suas principais técnicas, o DSSS e o FHSS. Diferente destas, que utilizam componentes de sinais aleatórios para codificar os sinais de mensagem, a CSS utiliza a variação linear de frequência inerente aos *chirps* como forma de codificação. Por conta disso, Vangelista (2017) propõe que uma melhor nomenclatura para a técnica seria Modulação de Deslocamento de *Chirp* em Frequência (FSCM - *Frequency Shift Chirp Modulation*) já que, na prática, esta técnica utiliza o *chirp* como uma espécie de portadora e codifica a mensagem na própria variação de frequência.

Segundo a Semtech (2015), sua modulação LoRa utiliza *chirps* no espalhamento espectral de modo a atuar como uma versão mais eficiente da modulação DSSS, tendo como grandes vantagens uma menor complexidade na construção de demoduladores de sinal, maior imunidade ao desvanecimento e ao efeito Doppler, maior robustez e maior alcance(Elshabrawy e Robert, 2018).

A taxa de *bits* da modulação LoRa, segundo descrito pela Semtech (2015), pode ser representada pela Equação:

$$R_b = SF \cdot \left(\frac{4}{4+CR}\right) \cdot R_s \quad [bits/s], \tag{2.2}$$

em que SF é o fator de espalhamento (*Spreading Factor*), CR é a taxa de codificação (*Coding Rate*) e R_s é a taxa de símbolos (*Symbol Rate*), expressa por:

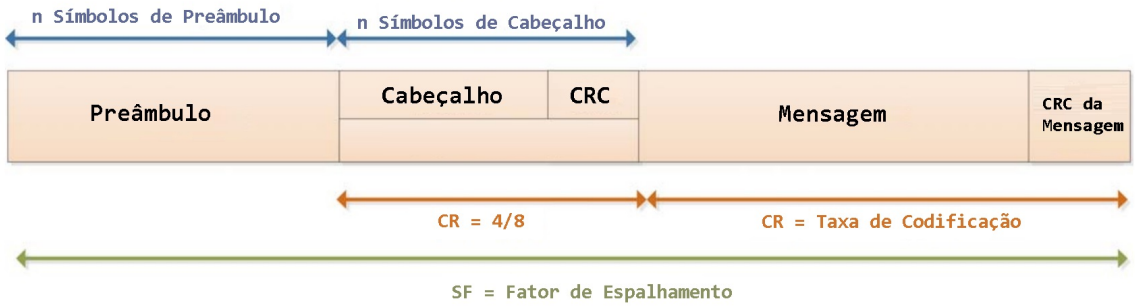
$$R_s = \frac{1}{T_s} = \frac{BW}{2^{SF}} \quad [simbolos/s], \tag{2.3}$$

em que BW é a largura de banda da modulação (*Bandwidth*). Já a taxa de variação de código R_c pode ser definida por:

$$R_c = R_s \cdot 2^{SF} = \frac{BW}{2^{SF}} \cdot 2^{SF} = BW \quad [chips/s], \tag{2.4}$$

Com relação aos pacotes de dados transmitidos pela modulação LoRa, apresenta-se, na Figura 2.6, sua estrutura básica.

Figura 2.6: Estrutura de pacotes LoRa.



FONTE: Adaptado de Semtech (2016)

O preâmbulo é um trecho da mensagem utilizado para indicar ao receptor a chegada de um novo pacote. Por padrão, possui 12 símbolos de comprimento, mas este pode ser alterado por software através dos valores inseridos em dois registradores de 8 *bits*, *RegPreambleMsb* e *RegPreambleLsb*, permitindo preâmbulos com comprimentos entre 6 e 65535 símbolos. Um receptor ocioso só entra em modo de leitura após detectar corretamente o preâmbulo no tamanho pré-configurado. Caso contrário, permanece em um laço periódico, tentando detectar o preâmbulo correto.

Por sua vez, o cabeçalho indica informações básicas a respeito da mensagem, como o comprimento de seu conteúdo e a presença ou não de uma Checagem de Redundância Cíclica (CRC - *Cyclic Redundancy Check*) de 16 *bits*. Ao final do cabeçalho, também há uma CRC própria. Quando a taxa de codificação (CR) e a CRC são conhecidas e fixas em todos os transceptores, é possível utilizar o modo de cabeçalho implícito, onde este trecho do pacote é omitido na transmissão.

A mensagem (*payload*) possui um comprimento variável de 1 a 255 *bytes* e um CRC próprio de 16 *bits*, de caráter opcional, que pode ser habilitado ou desabilitado nas configurações da transmissão.

A duração do envio de cada pacote T_{pacote}, também chamada de Período no Ar, pode ser definida através da Equação:

$$T_{pacote} = T_{preamb} + T_{msg} \quad [s], \tag{2.5}$$

na qual T_{preamb} representa o tempo necessário para enviar o preâmbulo do pacote e T_{msg} representa o tempo necessário para enviar a mensagem propriamente dita.

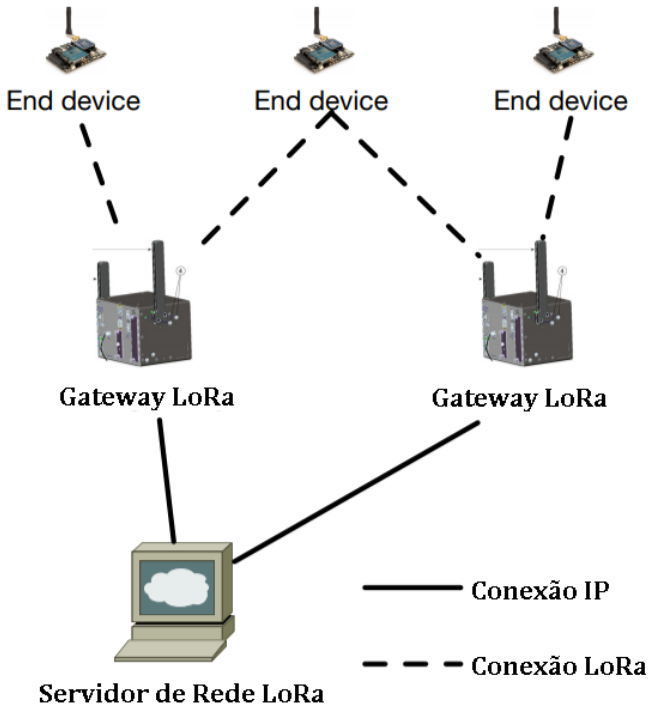
2.1.1.1.2 Camada de Rede (LoRaWAN)

Em LoRa Alliance (2015), descreve-se a LoRaWAN como um protocolo de rede otimizado para dispositivos, móveis ou fixos, alimentados a bateria. Sua disposição típica segue a topologia de uma rede estrela, na qual múltiplos nós dispositivos (também chamados de

end devices) são conectados a múltiplos *gateways* (aumentando a redundância na transmissão para diminuir a taxa de erro) que, por sua vez, repassam os pacotes ao servidor de rede, de onde são direcionados para os servidores de aplicação, onde é executada a camada de aplicação, mencionada na Figura 2.4. A topologia típica de uma LoRaWAN pode ser observada na Figura 2.7.

O *Gateway* é o dispositivo responsável por gerenciar os dados de um ou vários *end devices* e condicioná-los para serem enviados a um servidor de rede. Os servidores de rede, por sua vez, são dispositivos responsáveis por centralizar e gerenciar todas as informações do sistema, além de prover filtragem de redundâncias, checagem de segurança e verificações de erro mais robustas (Medeiros, 2018).

Figura 2.7: Topologia de rede LoRaWAN.



FONTE: Adaptado de Augustin et al. (2016)

Segundo a Lora Alliance (2015), a camada da rede LoRaWAN categoriza seus *End Devices* em três classes: A, B e C.

A Classe A permite ao dispositivo utilizar comunicações bi-direcionais em que, após cada transmissão *uplink* de um dispositivo, lhe são permitidos dois curtos intervalos de recepção *downlink*. O uso de *downlink* é restrito apenas a estes intervalos. A operação na Classe A é a que gera menor consumo e é indicada para dispositivos com aplicações que só requerem curtas (ou nenhuma) comunicações de *downlink* com o servidor (LoRa Alliance, 2017).

A Classe B permite ao dispositivo utilizar comunicações bi-direcionais da mesma forma que a Classe A, porém possui intervalos adicionais alocados para recepção *downlink*. Esta classe de dispositivos recebe um sinal (*Beacon*) a cada vez que seus intervalos adicionais se repetem. Este modo de operação é indicado para dispositivos que não se adequem nem à Classe A e nem à Classe C.

A Classe C permite ao dispositivo utilizar comunicações bi-direcionais com acesso contínuo a uma janela de recepção *downlink*, de modo que os dispositivos podem receber

informações do servidor a qualquer instante. Este modo de operação é o que gera maior consumo, porém é o que oferece menor latência para comunicação Servidor-Dispositivo.

2.2 Sensores de Movimento

Esta subclasse de sensores tem como aplicação a detecção, de maneira geral, de algum tipo de movimento em uma região de análise, como por exemplo objetos passando por uma esteira em uma linha de produção fabril ou pessoas passando por um corredor de um prédio. Sensores de movimento, também chamados de sensores de presença, são bastante utilizados como uma solução de automação e de eficiência energética, como é o caso de sensores de estacionamento, portas e luzes automáticas ou sensores de descarga e torneira em banheiros públicos (Guidino, 2018).

Em termos da tecnologia empregada, existem diversos sensores, tanto analógicos quanto digitais, capazes de realizar esta função. Tem-se como exemplos de sensores digitais os sensores baseados em tomografia e os sensores de movimento por vídeo (que se utilizam de sensores de imagem e processamento em tempo real para realizar o sensoriamento). Já como exemplos de sensores analógicos existem sensores ultrassônicos, sensores de micro-ondas e sensores infravermelhos (Bay Alarm, 2020).

Tipicamente, pode-se classificar os sensores com esta aplicação em dois grandes grupos quanto ao seu funcionamento: os sensores ativos e os sensores passivos. Sendo os ativos sensores que realizam o sensoriamento através de um dispositivo transmissor e de um dispositivo receptor, como é o caso dos sensores ultrassônicos. Já os sensores passivos contêm apenas um receptor e se baseiam em captar alguma grandeza física emitida pelos corpos em movimento, como é o caso de alguns tipos de sensores infravermelhos (Berry, 2020).

2.2.1 Sensores Piroelétricos (PIR)

Os Sensores do tipo Passivo Infravermelho (PIR - *Passive Infrared*) são uma classe de dispositivos capazes de detectar radiação no espectro infravermelho. Estes possuem tal nomenclatura para serem diferenciados dos sensores infravermelhos ativos e, ao contrário destes, não são capazes de emitir luz infravermelha.

O tipo mais comum de sensor PIR é o dos sensores piroelétricos e, por conta disso, o termo sensor piroelétrico acaba sendo incorretamente associado como um sinônimo a sensores PIR ou mesmo a Sensores de Presença. Estes sensores se baseiam no princípio físico da piroeletricidade e são muito utilizados comercialmente para detecção de movimento humano. Sua aplicação mais comum é o acionamento de lâmpadas automáticas, mas também podem ser encontrados em descargas de banheiro automáticas e em portas automáticas.

A piroeletricidade (em que o sufixo piro- deriva do grego *Pyr*, significando fogo) é uma propriedade física presente em alguns cristais ferroelétricos e pode ser descrita como a capacidade de um material gerar uma pequena tensão quando sofre uma mudança de temperatura. Sendo mais específico, o fenômeno da piroeletricidade é a geração de um potencial eletroquímico em um material cristalino ao ter sua estrutura alterada por irradiação térmica (Fraden, 1999).

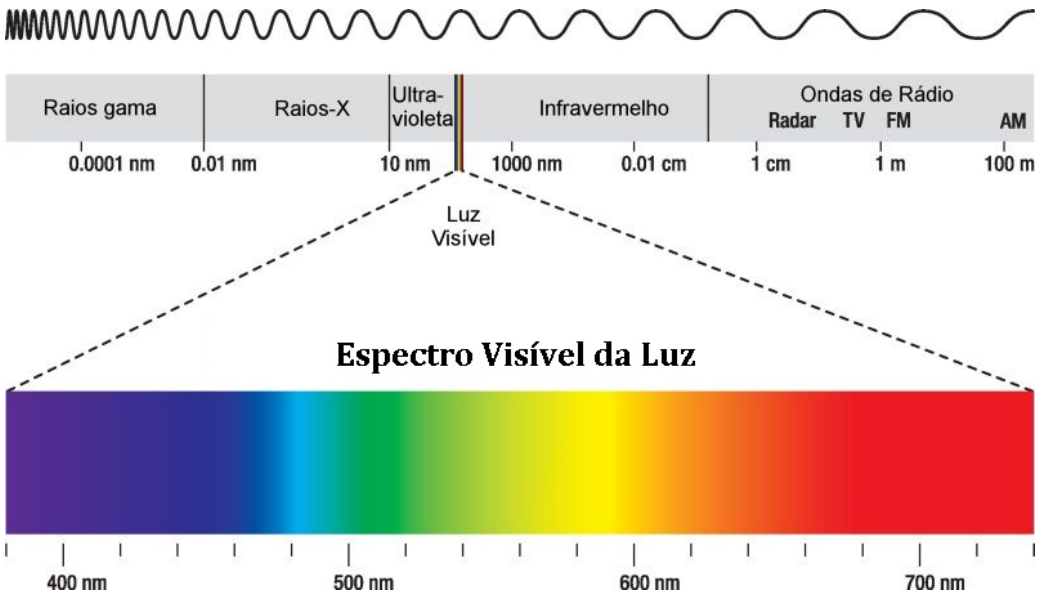
A nível atômico, a radiação, ou seja, um feixe de fótons se movendo a determinada frequência, colide com as moléculas cristalinas do material e transfere sua energia cinética ao mesmo, no processo que é descrito pela física como transferência de calor. Dessa

forma, o fenômeno piroelétrico pode ser visto não como luminoso ou térmico, mas como uma transferência de energia das partículas de radiação (fótons) para o material cristalino (Fraden, 1999).

Ainda segundo Fraden (1999), o movimento natural dos átomos ocorre não somente no núcleo, mas também em seu campo elétrico. Este campo elétrico em movimento, ou campo elétrico variante, por sua vez, produz um campo magnético variante. Com o campo magnético variante, os átomos adjacentes são induzidos e produzem campos elétricos variantes em resposta, propagando o campo eletromagnético da partícula em movimento. Esta propagação recebe o nome de radiação térmica. Em termos práticos, pode-se afirmar que todo corpo que possui massa, ou seja, que é composto por átomos, pode ser visto como um gerador de radiação térmica.

A radiação térmica emitida por uma partícula ou objeto, por ser um fenômeno eletromagnético, pode ser expresso como um sinal periódico com determinado comprimento de onda. Objetos em altíssimas temperaturas tendem a emitir radiação com comprimentos de onda entre 380 nm e 780 nm, compondo o denominado espectro de luz visível (Figura 2.8) e tornam-se incandescentes, emitindo um brilho colorido. O aumento de temperatura é inversamente proporcional ao comprimento da onda irradiada, de modo que objetos menos quentes emitem ondas de maior comprimento (Fraden, 1999).

Figura 2.8: Espectro visível da luz.



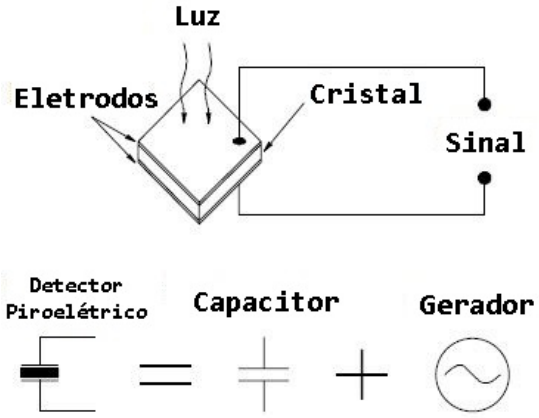
FONTE: Info Escola (2019)

O corpo humano é capaz de emitir radiação térmica dentro de uma faixa de comprimentos de onda de 5 μm a 15 μm, nas regiões denominadas de Espectros Infravermelho Médio e Longo (Fraden, 1999). Estas emissões, apesar de não detectáveis pelos olhos, são utilizadas pelos sensores piroelétricos para detectar a presença humana.

A construção desses sensores é baseada na ligação simétrica entre dois cristais piroelétricos. De acordo com Fraden (1999), este arranjo simétrico é feito de modo a atenuar o efeito piezoelétrico causado pela dilatação térmica dos cristais. Além disso, este arranjo gera um defasamento de 180 graus de modo que, quando irradiados pelo mesmo feixe, ambos os cristais produzam tensões elétricas que se anulem. Desta forma, o sensor detecta a radiação infravermelha de maneira diferencial.

O cristal piroelétrico tipicamente é utilizado acoplado a dois eletrodos em suas extremidades. Ao conjunto se dá a nomenclatura de Detector Piroelétrico. Tomando cada eletrodo como um terminal elétrico, seu comportamento ao receber radiação pode ser modelado como um capacitor em paralelo a uma fonte de tensão controlada por radiação térmica, como mostrado na Figura 2.9.

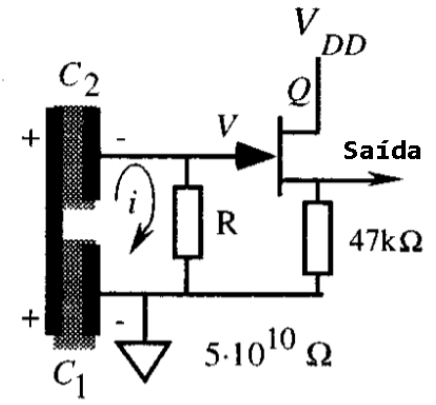
Figura 2.9: Analogia elétrica do detector piroelétrico irradiado.



FONTE: Adaptado de RF Wireless World (2019)

Um circuito de condicionamento deve ser utilizado para se amplificar a tensão gerada pelo cristal irradiado. A topologia mais utilizada é a de um transistor JFET polarizado atuando como desacoplador de impedâncias para a tensão produzida pelo cristal, como mostra a Figura 2.10.

Figura 2.10: Circuito de condicionamento para o cristal piroelétrico.



FONTE: Adaptado de Fraden (1999)

Dessa forma, quando um dos cristais receber irradiação, produzirá uma corrente elétrica proporcional à energia da irradiação absorvida. O valor típico de corrente para uma irradiação proveniente do corpo humano é de 1 pA, de modo que, para se obter uma tensão de saída típica de 5 V, de acordo com a Primeira Lei de Ohm, seria necessário acoplar um resistor R com impedância de valor:

$$R = \frac{V}{I} = \frac{5V}{1pA} = 5 \cdot 10^{12} \quad [\Omega], \tag{2.6}$$

Por conta desse valor elevado, o desacoplamento de impedâncias promovido pelo JFET é de extrema importância, já que a equivalência de impedâncias no circuito pode diminuir severamente o valor de R e, por consequência, a tensão de saída obtida do sensor.

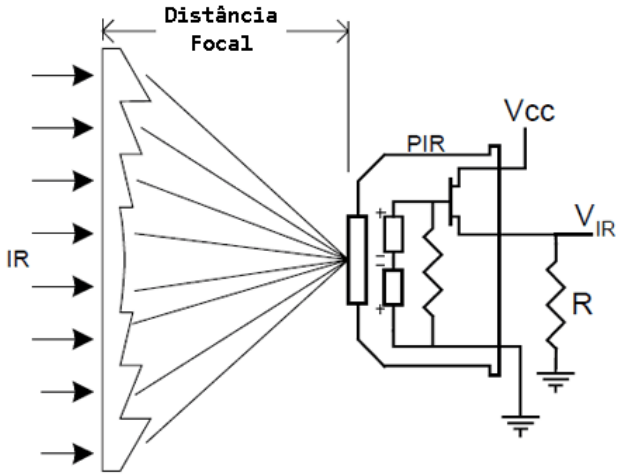
Este circuito é usualmente encapsulado em metal, o que permite uma maior isolação da radiação presente no ambiente. A única abertura no encapsulamento encontra-se diretamente acima do par de cristais e é coberta por um filme plástico transparente à faixa de Infravermelho Médio e Longo (Figura 2.11).

Figura 2.11: Encapsulamento metálico de um sensor PIR.

FONTE: Global Sources (2019)

Outro elemento importante em um sensor piroelétrico é sua lente. Por se tratar de um elemento sensível a ondas eletromagnéticas, seus resultados podem sofrer interferências baseadas na forma como os feixes infravermelhos colidem com o sensor. Para evitar que isso aconteça, um tipo especial de lente, denominada Lente de Fresnel (Figura 2.12), é utilizada. Esta lente permite que um feixe infravermelho muito maior do que o próprio sensor seja focalizado em um único ponto.

Figura 2.12: Lentes de Fresnel em um circuito piroelétrico.



FONTE: Adaptado de Adafruit (2019)

2.3 Sensores de Imagem

Sensores de imagem, de maneira geral, são quaisquer tipo de dispositivo capaz de capturar imagens na forma de informações codificadas, que podem ser utilizadas para reconstruir a imagem em um meio distinto. Estes sensores existem tanto como dispositivos analógicos quanto como dispositivos digitais. Nos sensores de imagem analógicos, o processo fundamental para captura da imagem é uma reação fotoquímica. Já nos sensores de imagem digitais, o processo fundamental é uma reação fotoelétrica.

Segundo Nakamura et al. (2006), uma imagem pode ser definida como a variação de intensidade luminosa ou taxa de reflexão em função de sua posição em um plano finito. O conceito de captura de imagem é, então, a habilidade de armazenar, como informação, a radiação luminosa refletida por uma determinada região.

Gonzalez e Woods (2001), por sua vez, define imagem sob a perspectiva matemática, sendo vista como uma função de duas variáveis $f(x, y)$, em que x e y representam as coordenadas (horizontal e vertical, respectivamente) em um plano finito e f, dado qualquer par de coordenadas (x, y), representa a intensidade luminosa da imagem naquele ponto. Com isso, a representação matemática se dá pela forma:

$$P_{xy} = \{I_\nu(x, y)\}; \quad x, y \subset \mathbb{R}, \tag{2.7}$$

em que I_ν é a intensidade luminosa observada em um ponto do plano finito determinado pelo par de coordenadas (x, y), expressa na unidade *cd* (Candelas).

Ao lidar com sistemas e sensores digitais, no entanto, as representações digitais de uma imagem são conjuntos de dados discretos, ou seja, amostrados. Desse modo, podem ser modelados pelo caso específico da Equação (2.7):

$$\hat{P}_{mn} = \hat{I}_\nu[m, n]; \quad m, n \subset \mathbb{Z}, \tag{2.8}$$

Os valores de P são conhecidos como Elementos de Imagem, Elementos de Figura ou *Pixels* (do inglês *Picture Element*).

A primeira câmera eletrônica foi registrada em 1972, pela *Texas Instruments* (Adcock, 1977), mas foi na década de 1980 que a Kodak popularizou as câmeras digitais, baseadas em dispositivos eletrônicos. Estas lentamente dominariam o mercado ao longo das duas décadas seguintes, até que finalmente as grandes empresas fabricantes de câmeras fotográficas descontinuariam a produção das câmeras baseadas em filme.

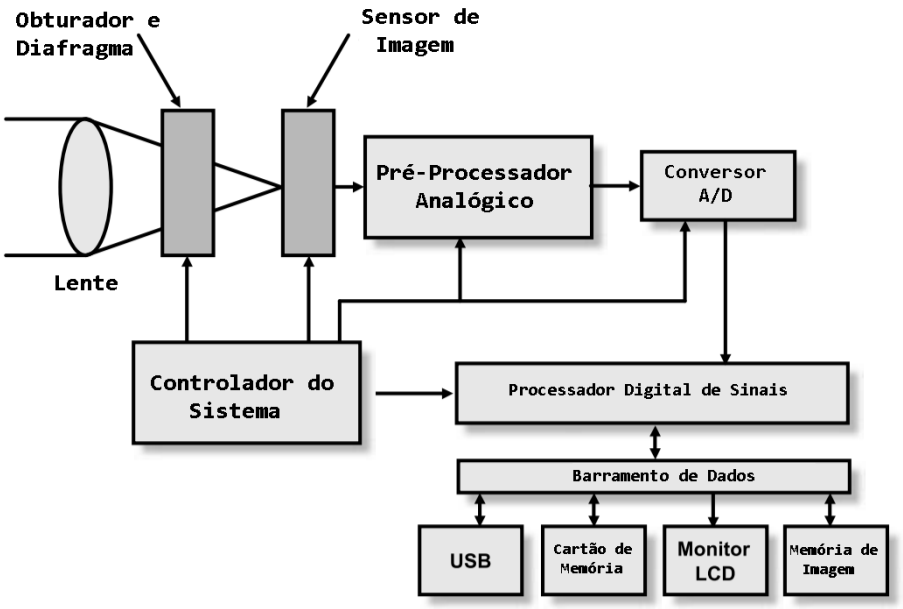
A grande vantagem no armazenamento eletrônico é que permitia às câmera digitais uma capacidade de armazenamento muito maior do que as câmeras de filme da época. A Figura 2.13 mostra a estrutura típica de uma câmera digital, em diagrama de blocos.

2.3.1 Sensores CCD

Os sensores de imagem nas câmeras digitais, os dispositivos que de fato realizavam a captura de imagem, eram inicialmente todos baseados na tecnologia de Dispositivo de Carga Acoplada (CCD - *Charge-Coupled Device*), desenvolvido pela AT&T (Boyle e Smith, 1973).

De acordo com Nakamura et al. (2006), o princípio de funcionamento do CCD baseia-se no conceito de utilizar a dopagem eletrônica de materiais semicondutores como forma de armazenamento e a deriva eletrônica ocorrida em junções PN como forma de transferência de dados.

Figura 2.13: Estrutura típica de uma câmera digital.



FONTE: Adaptado de Nakamura et al. (2006)

A camada sensora do CCD é composta por uma matriz de fotodiodos (elementos semicondutores sensíveis à luz) e capacitores de deslocamento do tipo Óxiodo Metálico Semicondutor (MOS - *Metal Oxide Semiconductor*). Com a exposição dos fotodiodos à radiação da luz visível, estes produzem uma corrente elétrica, que acumula nos capacitores acoplados a eles uma determinada quantidade de elétrons proporcional às intensidades luminosas a que os fotodiodos foram submetidos. Esta determinada quantidade de elétrons em cada capacitor forma o que é chamado de "Poço de Potencial Elétrico". A porção da imagem que o poço de potencial detecta é armazenada em um *pixel*.

Em cada linha de capacitores MOS, o conjunto de poços de potencial pode ser transferido aos capacitores adjacentes ao se aplicar tensões nas portas de cada capacitor de maneira sincronizada, produzindo um efeito similar ao de um registrador de deslocamento analógico. Dessa forma, torna-se possível obter as informações armazenadas em cada capacitor de maneira serial em cada coluna, de modo que estas podem ser condicionadas na forma de sinais digitais e armazenadas em registradores de memória (Nakamura et al., 2006).

2.3.2 Sensores CMOS

No começo da década de 1990, surgiu uma tecnologia que se provaria uma confiável alternativa ao CCD: o Sensor de Imagem baseado nos Semicondutores Complementar de Óxido Metálico (CMOS - *Complementary Metal Oxide Semiconductors*). Estes sensores também são conhecidos como Sensores de Pixels Ativos (APS - *Active Pixel Sensors*). Cada *pixel* é capturado por um conjunto de um fotodiodo e um capacitor chaveado CMOS.

Sua construção apresenta três grandes vantagens em relação aos sensores de imagem CCD. A primeira era o consumo reduzido, ao se utilizar elementos de alta impedância (transistores complementares) no lugar de capacitores, reduzindo o consumo de corrente, além de uma menor tensão de alimentação necessária (tipicamente 2,5 V ou 3,3 V em relação aos 8 V ou 15 V do CCD). A segunda era uma maior miniaturização, já que os

CMOS com mesma capacidade de armazenamento ocupavam um menor espaço. A terceira era a capacidade de leitura de *pixels* individuais por comando. No entanto, os primeiros modelos de sensores CMOS possuíam um alto ruído no barramento de leitura e uma considerável interferência de corrente escura no valor transferido do fotodiodo ao CMOS. Estas desvantagens na qualidade da imagem fariam com que, inicialmente, os sensores CMOS fossem restritos ao uso em câmeras mais simples, como *Webcams*, Câmeras de Celular e *Toy Cameras* (Nakamura et al., 2006).

Porém, muitas desvantagens dos sensores CMOS foram compensadas com o passar dos anos, especialmente por meio de tecnologias inicialmente pensadas para CCDs. Atualmente, os sensores de imagem CMOS constituem a maioria dos dispositivos fotográficos, sendo os CCD utilizados apenas em câmeras profissionais ou dispositivos que capturam resoluções muito grandes (Nakamura et al., 2006).

Em sensores CMOS, a forma típica de leitura é conhecida como Leitura X-Y, ou *Rolling Shutter*. Com a capacidade de operar cada CMOS individualmente, cada *pixel* é capturado e lido, elemento por elemento, linha por linha.

Toda a varredura dos sensores dura uma fração de segundo e este procedimento consome quantidades consideravelmente menores de energia em relação ao CCD, que deve manter alimentada toda a matriz de capacitores a todo instante para poder capturar os *pixels* da imagem. No entanto, apesar da relativa rapidez de toda a varredura da matriz de *pixels*, objetos muito velozes frequentemente são distorcidos na captura da imagem, por conta do movimento linear de leitura. Este problema não é visto na tecnologia CCD, que captura toda a imagem simultaneamente, serializando apenas a etapa de leitura dos dados.

2.4 Biometria Facial

A Biometria, ou Autenticação Biométrica, como seu nome dá a entender (*Bio*, "Vida" e *Metria*, "Medição"), é o conjunto de técnicas responsáveis por mensurar características de um indivíduo que lhe são únicas e pelas quais ele pode ser identificado. As tecnologias biométricas podem ser classificadas em *identificatórias* e *verificatórias* (Jain, Ross e Prabhakar, 2004; Boulkenafet, 2017).

Na biometria identificatória, os dados coletados pelo sistema são comparados à totalidade de seu banco de dados, classificando cada comparação através de uma pontuação e escolhendo como resultado a melhor pontuação que adentre os critérios de aceitabilidade (e.g. reconhecimento facial ou de íris). Esta categoria de biometria tipicamente emprega o *reconhecimento negativo*, que define quem a pessoa não é, de modo que uma pessoa jamais será identificada por mais de uma credencial (Jain, Ross e Prabhakar, 2004).

Na biometria verificatória, os dados são recebidos com uma atribuição prévia a um indivíduo e o sistema tem a função de confirmar se os dados recebidos correspondem à credencial alegada (e.g. cartão magnético, RFID ou NFC). Nesta categoria normalmente se emprega o *reconhecimento positivo*, que define quem a pessoa é, de modo que uma credencial jamais será usada para identificar mais de uma pessoa (Jain, Ross e Prabhakar, 2004). De maneira geral, esta é a maneira mais segura de garantir que o acesso não supervisionado a determinadas informações, elementos, experiências ou o que for, seja restrito a um indivíduo ou grupo de pessoas específico.

Apesar dos fundamentos da biometria serem bastante homogêneos em termos conceituais, suas implementações são bastante diversas em termos de execução. Dentre as mais populares estão o reconhecimento de impressão digital, o reconhecimento de voz,

o reconhecimento de íris e o reconhecimento facial (Jain, Flynn e Ross, 2008). Com o interesse de incrementar a segurança e a robustez desses sistemas de autenticação e com o surgimento de novas tecnologias de sensoriamento e medição, com o passar dos anos novas soluções de biometria vão surgindo e as já existentes vão sendo aprimoradas.

Dentre as soluções de biometria disponíveis na atualidade, a biometria através do reconhecimento facial tem-se mantido em um patamar de grande importância e de vasta aplicabilidade por conta de seus pontos fortes: um baixíssimo custo na implementação de infraestrutura (sendo necessário apenas uma câmera e um processador digital de imagens qualquer), uma grande densidade de características utilizáveis como forma de autenticação e sua pouca invasividade durante a medição (Boulkenafet, 2017; Kortli et al., 2020).

A constituição básica dos sistemas de reconhecimento facial se dá, quase sempre, em dois passos: a detecção do rosto e a identificação do rosto. Em ambas as etapas, é crucial manter uma alta coerência nas características mensuradas, considerando o grande número de fatores a interferirem na captura de rostos por imagem (como ângulo, distorção, oclusão, iluminação, foco, ruído, entre outros.). Por conta disso, a maioria dos sistemas de biometria facial emprega técnicas de reconhecimento negativo, garantindo que várias possíveis leituras de um mesmo rosto sejam associadas sempre com a mesma credencial (Jain, Ross e Prabhakar, 2004).

Em termos de implementação da detecção facial, três métodos compõem a quase totalidade das soluções presentes atualmente no mercado (Kortli et al., 2020; Neethu e Ãnoop, 2014). São eles:

- Os métodos de Análise Matricial, como os classificadores *Cascade*;
- Métodos baseados em Máquinas de Vetores de Suporte (SVM - *Support Vector Machines*), como os Histogramas de Gradientes Orientados (HOG - *Histograms of Oriented Gradients*);
- e métodos que utilizam as Redes Neurais de Aprendizado Profundo (DNN - *Deep Neural Networks*).

Os Algoritmos *Cascade* surgiram a partir de 2001, com o algoritmo Viola-Jones, popularmente conhecido como *Haar Cascade*. Estes algoritmos fazem uso de um caso particular da Transformada Wavelet, conhecida como Transformada Haar ou Transformada Wavelet de Haar (Viola e Jones, 2001).

Os algoritmos HOG, apesar de já existentes desde a década de 1980, tiveram seu uso difundido como solução para detecção facial apenas após Dalal and Triggs (Dalal e Triggs, 2005). Estes algoritmos utilizam campos vetoriais obtidos através do treinamento de estruturas SVM para reconhecer contornos faciais. Essas estruturas SVM, por sua vez, podem ser definidas como algoritmos de computador que realizam aprendizado de máquina através de técnicas de classificação entre dois grupos e fazem isso atribuindo rótulos a determinados trechos de dados e comparando esses dados entre os conjuntos (Noble, 2006).

Algoritmos DNN fazem uso de uma técnica mais recente e moderna e têm se tornado a escolha dominante para aplicações de detecção facial, especialmente com o surgimento de computadores acessíveis com alta taxa de processamento gráfico. Este tipo de algoritmo começou a se tornar popular por volta de 2014, com o lançamento da biblioteca de desenvolvimento *Deepface*, estruturada no uso de redes neurais convolucionais para detecção facial. O processo de treinamento nestes algoritmos, em média, é considerado mais lento

do que o obtido nas técnicas previamente mencionadas por necessitar de um alto número de iterações de treino. No entanto, os resultados obtidos quase sempre são superiores aos demais em termos de precisão e acurácia, de modo que seu custo-benefício relativo ainda é bastante positivo (Balaban, 2015; Zhao et al., 2019).

2.4.1 *Spoofing*

Assim como são inúmeras as técnicas de biometria existentes na atualidade, também são diversas as técnicas para fraudar esses mecanismos, considerando que assim como existe o interesse de utilizar a biometria como uma forma de segurança, ao restringir e proteger o acesso a determinadas informações, também existe o interesse em burlar essas restrições e acessar dados sigilosos.

Em termos de eficiência, as maiores ameaças aos sistemas de biometria são as fraudes por meio de *cracking* destes sistemas (utilizando algoritmos para derrubar o sistema ou para encontrar falhas de segurança) ou mesmo pelo roubo físico de dispositivos armazenando dados de seus bancos de dados. Em termos de ocorrência, porém, as tentativas de fraude mais comuns são as tentativas de falsificação de identidade, mais conhecidas pelo termo em inglês "*spoofing*" (Galbally, Marcel e Fierrez, 2014).

A razão principal para que tentativas de fraude por *spoofing* sejam tão comuns é a relativa facilidade com a qual elas podem ser efetuadas, sendo tipicamente necessário apenas possuir a(s) característica(s) utilizada(s) na autenticação de um usuário válido. Isto pode ser significativamente mais complexo de se obter em sistemas biométricos que utilizem tecnologias como o reconhecimento de formato manual ou reconhecimento venal, mas pode se tornar particularmente simples e intuitivo em biometrias pouco invasivas, como o reconhecimento facial ou de voz. Dessa forma, a facilidade com que o *spoofing* pode ser praticado em um sistema biométrico específico e, portanto, a dificuldade inerente em combatê-lo com técnicas anti-fraude está diretamente ligada com o nível de exposição da característica individual utilizada na autenticação daquele sistema (Galbally, Marcel e Fierrez, 2014).

Um ponto comum entre as técnicas de reconhecimento facial previamente mencionadas é que a alta flexibilidade na identificação das características de um rosto, apesar de aumentar consideravelmente a robustez do sistema biométrico, também traz consigo uma vulnerabilidade inerente. Essa vulnerabilidade se expressa no fato do sistema não ter nenhuma forma própria para distinguir uma pontuação suficientemente classificatória obtida por um rosto verdadeiro em condições de fotografia não-favoráveis, como baixa luminosidade ou ângulos atípicos, de uma pontuação obtida por uma cópia do rosto em alguma tentativa de fraude (Boulkenafet, 2017).

No entanto, apesar desta grande vulnerabilidade a tentativas de fraude por *spoofing*, esta não é uma questão irremediável. Com a crescente popularização destas tecnologias e o eventual aumento no número de tentativas de fraudes, várias técnicas *anti-spoofing* foram desenvolvidas nos últimos anos e se mostrado bastante efetivas, especialmente quando utilizadas em conjunto, para gerar redundâncias na segurança (Souza et al., 2017).

Podem ser encontradas técnicas que se utilizam de múltiplos quadros de vídeo para identificar um rosto em movimento (Li et al., 2004), ou técnicas para imagens estáticas, analisando um ou vários padrões, como cor (Boulkenafet, Komulainen e Hadid, 2016), textura (Kim et al., 2012; Komulainen, Hadid e Pietikäinen, 2013; Määttä, Hadid e Pietikäinen, 2012) e nitidez (Siddiqui e Park,2019; Kim et al., 2012), que apresentam variações bastante coerentes por se tratarem da maioria das tentativas de *spoofing* facial.

As técnicas *anti-spoofing* de vídeo são eficazes não somente para as tentativas de fraude estáticas (imagens digitais ou impressas do rosto da vítima apresentadas à câmera) mas também para as fraudes dinâmicas (maquiagem e máscaras 3D), porém trazem restrições estruturais por não poderem ser implementadas em sistemas de captura única, podendo exigir uma troca completa de *hardware* no sistema para implementar esta medida de segurança (Li et al., 2004).

As técnicas de imagem estática também costumam ser utilizadas em conjunto com as técnicas de vídeo para uma robustez ainda maior, mas podem muito bem serem utilizadas sozinhas, no caso de sistemas mais simples, sem captura de vídeo. Isso pode ser feito sem sacrificar em demasiado a segurança do sistema, pois a complexidade e o alto custo investido na prática das fraudes faciais dinâmicas acaba sendo um obstáculo natural por si, diminuindo significativamente sua ocorrência em relação às fraudes estáticas (Määttä, Hadid e Pietikäinen, 2012).

2.4.2 Redes Neurais Artificiais

As Redes Neurais Artificiais (RNA), segundo Haykin (2008), são máquinas projetadas para modelar a forma como um cérebro humano realiza determinadas funções ou tarefas de interesse. Pode-se descrevê-las então, como processadores massivamente paralelizados, compostos de simples unidades de processamento, denominados "neurônios", que têm propensão natural a armazenar conhecimento adquirido por experiência e torná-lo disponível para uso posterior.

Essas RNA, têm sua concepção inspirada no funcionamento do Cérebro Humano, cujo processo de aprendizado consiste em armazenar uma informação ao associá-la a uma determinada "rota", que consiste em um determinado conjunto de neurônios, interconectados por sinapses, disparando em uma sequência específica. Para as RNA, essa associação é feita iterativamente através do ajuste de certos pesos matemáticos, em um processo denominado de treinamento, que determinam qual caminho percorrer nas estruturas denominadas neurônios, organizadas em conjuntos denominados camadas, em função de uma determinada entrada. Dessa forma, assemelham-se a um cérebro físico em dois aspectos:

1. O conhecimento é adquirido pela rede advindo do meio através de um processo de aprendizado;

2. As conexões entre neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido.

Ainda utilizando da analogia do cérebro humano, as unidades básicas de processamento das RNA, os neurônios, podem ser modelados como contendo, essencialmente, três características:

- Um conjunto de ligações (sinapses) com outros neurônios, onde cada um é caracterizado por um peso próprio. Em que um sinal x_j na entrada da sinapse j é conectado a um neurônio k e, por isso, multiplicado pelo peso sináptico w_{kj};

- Um somador de sinais de entradas, ponderados pelos pesos sinápticos de seus respectivos neurônios;

- Uma função de ativação para limitar a amplitude da saída do neurônio.

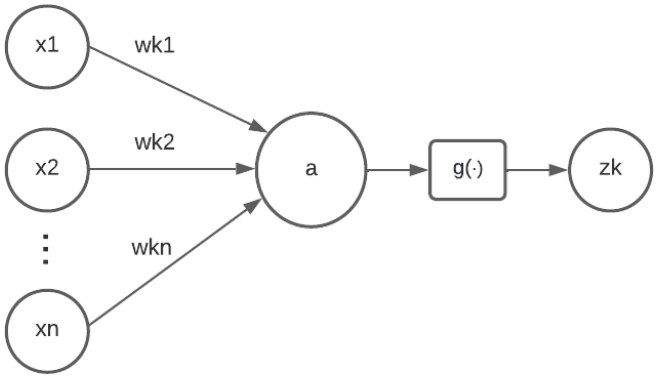
Na prática, um neurônio (Figura 2.14) é um valor arbitrado pela máquina no momento do treinamento da RNA, que depende das conexões com os neurônios da camada anterior e de uma função de ativação pré-determinada. Matematicamente, isso pode ser descrito pela Equação (2.9).

$$z_k = g(a) = g\left(\sum_{j=1}^{n} w_{kj}x_j + \theta_k\right), \tag{2.9}$$

onde $g(\cdot)$ é a função de ativação do neurônio, x_j é uma entrada do neurônio, w_{kj} é seu respectivo peso e θ_k é um valor constante que, assim como os pesos de cada entrada, pode ser ajustado para produzir determinada saída.

Existem múltiplas funções que podem ser utilizadas como função de ativação $g(\cdot)$ como, por exemplo, funções de degrau unitário, funções de tangente hiperbólica ou funções sigmóides (Sharma, Sharma e Athaiya, 2020).

Figura 2.14: Diagrama básico de um neurônio em uma RNA.



FONTE: O Autor (2022)

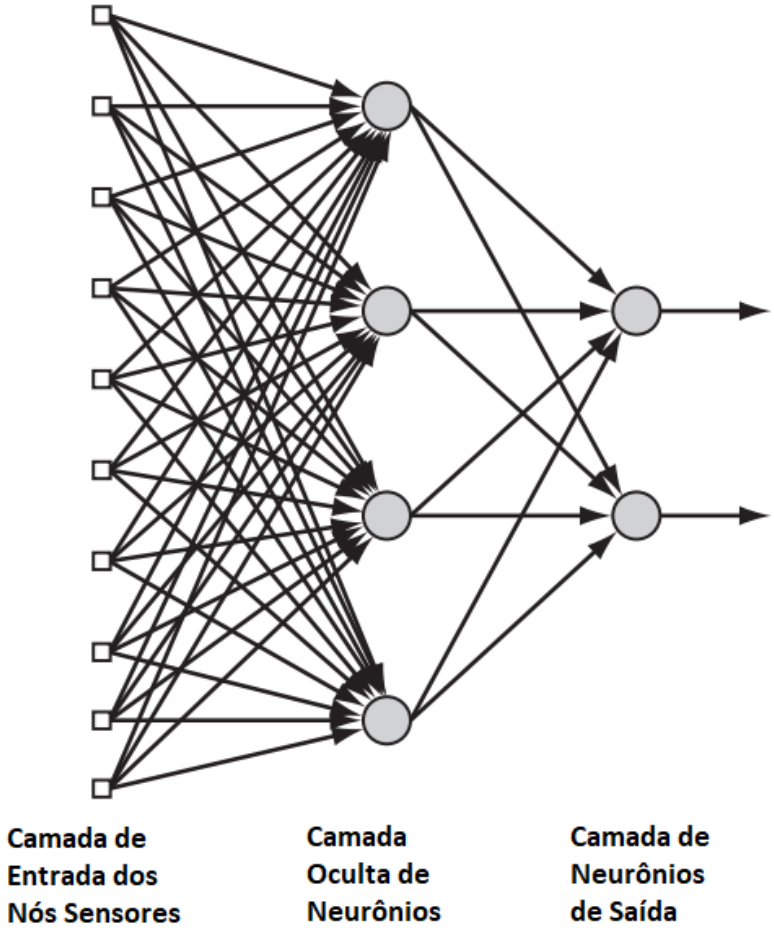
A arquitetura das RNA tipicamente segue a estrutura de camadas, onde uma camada inicial define os parâmetros de entrada, contendo um neurônio para cada um desses parâmetros. Na sequência, existem as camadas intermediárias, denominadas camadas ocultas, que são compostas por arranjos específicos de neurônios e sinapses. Não existe um método definitivo para determinar o formato ideal do conjunto de camadas ocultas de uma RNA para cada aplicação, sendo o método de tentativa e erro tipicamente utilizado na hora de encontrar o arranjo mais otimizado. Por fim, a rede contém uma última camada contendo um neurônio para cada parâmetro de saída. Na Figura 2.15 mostra-se um exemplo de rede neural de alimentação positiva com uma única camada oculta e dois neurônios de saída.

Considerando esta estrutura, é possível formular um equacionamento geral para a saída de uma camada de uma rede neural, com base na Equação (2.9):

$$\left[Y^{(p)}\right]_n = g\left([A]_n\right) = g\left([W]_{n\times m} \cdot \left[Y^{(p-1)}\right]_m + [\Theta]_n\right), \tag{2.10}$$

onde o vetor $Y^{(p)}$ são os valores de saída da p-ésima camada e o vetor $Y^{(p-1)}$ são os valores de saída da camada imediatamente anterior, utilizados nesta camada como valores de entrada ($X^{(p)} = Y^{(p-1)}$).

Figura 2.15: Estrutura básica de uma RNA com uma camada oculta.



FONTE: Adaptado de Haykin (2008)

O procedimento de aprendizado da RNA é denominado treinamento. Existem diversas técnicas de treinamento, como Aprendizado Supervisionado, Aprendizado Reforçado, Associação ou Reconhecimento de Padrões, Malha de Controle, dentre outras (Schmidhuber, 2015). O conceito por trás destas pode ser sintetizado como uma otimização iterativa dos parâmetros de peso sináptico w_k e da constante θ_k, para obter uma resposta de interesse.

Após treinada com um conjunto de dados, uma RNA é capaz de efetuar os procedimentos "aprendidos" em um novo conjunto de dados. Este novo conjunto de dados pode, no entanto, ser completamente diferente do conjunto de treinamento, necessitando ao menos de algumas características em comum. Isso ocorre porque o processo de treinamento é inteiramente baseado no reconhecimento de padrões entre um determinado conjunto de entradas e um determinado conjunto de saídas. Desse modo, quanto mais complexo o conjunto de parâmetros de entrada de uma RNA e quanto maior a quantidade de dados utilizados no treinamento, maior é o alcance da RNA.

Aplicações típicas de RNAs são sistemas de predição e sistemas de classificação. Ambas as aplicações se baseiam em treinar a RNA para reconhecer determinados padrões e então, no caso do primeiro, determinar resultados futuros com base nos resultados já existentes e, no caso do segundo, determinar dentre um conjunto de categorias pré-definidas a qual os dados apresentados são pertencentes.

Em sua maioria, técnicas de reconhecimento facial baseadas em RNA são considera-

das sistemas de classificação, pois o processo execução da RNA envolve a extração de um conjunto pré-determinado de características faciais e a comparação destas com as apresentadas nas Categorias existentes, ou seja, nos rostos de referência com os quais se deseja comparar o rosto detectado. O resultado dessas RNA de reconhecimento facial é determinado pela classificação das entradas (rostos) de acordo com o conjunto de referência (banco de dados), tipicamente feito através da tomada de decisão subjetiva em cima de um conjunto de valores percentuais, indicando a taxa de proximidade do rosto avaliado com cada um dos rostos conhecidos do banco de dados.

Como anteriormente mencionado, nos dias de hoje, o tipo de RNA mais comumente utilizado na biometria facial é o das DNN, que consistem em redes neurais com uma alta densidade de camadas ocultas, de modo que estas possuem um desempenho melhor lidando com um grande conjunto de parâmetros de entrada, que é o caso com o processamento das características faciais extraídas.

2.4.3 Pacote de Ferramentas Dlib

Dlib é um pacote de ferramentas (*toolkit*) independente e de código aberto, criado em 2002 e desenvolvido a partir da linguagem C++, integrando recursos que promovem soluções para as mais diversas aplicações (King, 2017).

Dentre as áreas abordadas por este *toolkit*, temos aprendizado de máquina, compressão de dados, interface de redes, paralelismo, processamento de imagem, métodos numéricos, além de diversas aplicações menores em temas variados (King, 2017).

Para esta RSSF em específico, foi utilizada a ferramenta de reconhecimento facial disponibilizada no *toolkit*. O reconhecimento facial no Dlib possui desempenho equiparável ao de sistemas de reconhecimento facial já bem estabelecidos, como o clássico sistema baseado na biblioteca OpenCV utilizando o método *Haar Cascade* apresentado em Viola e Jones (2004).

Em termos de arquitetura, trata-se de uma Rede Neural Residual (ResNet) com 29 camadas de convolução. O algoritmo obteve um nível de acurácia de 99,38% em treinamentos com *benchmarks* conhecidos, como o *Labeled Faces in the Wild* ou LFW (Learned-Miller et al., 2016).

O modelo utilizado no Dlib, apesar de possuir alguns contrapontos, como uma menor área de delimitação das faces, geralmente omitindo partes da testa ou do queixo. Além disso, seu treinamento garante reconhecimento confiável de faces com pelo menos 80x80 *pixels* (Gupta, 2019). O algoritmo apresenta importantes vantagens em relação ao já mencionado modelo *Haar* do OpenCV:

- Tem um menor tempo de processamento;
- Possui otimização para paralelismo de máquina;
- Tem maior tolerância à obstrução de faces;
- É capaz de identificar rostos com pequena inclinação;
- É muito mais robusto em relação a falsas detecções de faces.

Estas diferenças são proporcionadas pelo método utilizado, denominado *Deep Metric Learning* (King, 2017). Antes de 2017, o sistema de reconhecimento facial do Dlib se baseava na técnica de HOG, que utiliza um conjunto de campos vetoriais treinados para

detectar bordas e, com isso, encontrar rostos indiretamente através de suas silhuetas (King, 2014). Este sistema baseado em HOG também apresentava muitas vantagens encontradas no sistema baseado em DML. Mas com o surgimento de técnicas mais eficazes de reconhecimento facial envolvendo *Deep Learning* e RNA nos últimos anos, como as apresentadas pelo sistema OpenCV-DNN (sucessor do OpenCV-Haar), também o Dlib precisou ser atualizado, para se adequar novamente ao estado da arte.

2.4.4 Linguagem Python

Python é uma linguagem de programação criada em 1989 por Guido van Rossum, no Instituto de Pesquisa para Matemática e Ciência da Holanda, baseada em diversas linguagens de programação da época, como ABC, Modula-3 e C++, tendo sido originalmente planejada para ser utilizada por físicos e engenheiros em seus projetos (Borges, 2010).

A linguagem possui uma sintaxe clara e concisa, com estruturas muito próximas às utilizadas em linguagens humanas, motivo pelo qual é classificada como uma linguagem de programação de altíssimo nível (*Very High Level Language*).

É uma linguagem orientada a objeto e interpretada. Orientada a objeto pois é capaz de definir estruturas de dados (denominadas "objetos") com não somente atributos próprios, mas também funções próprias. Interpretada pois pode ser executada pelo sistema operacional diretamente na forma de algoritmo, sem a necessidade de passar por um processo de compilação, que a traduza para linguagem de máquina (Lutz, 2009).

Ela conta com uma quantidade numerosa de contribuidores e vem crescendo em popularidade nos últimos anos, estando entre as cinco linguagens de programação mais utilizadas por usuários nas comunidades de desenvolvedores, como GitHub, Meetup e StackOverflow, além de ser uma das aptidões mais demandadas e mais bem pagas no ramo de tecnologia (Srinath, 2017).

Atualmente pode-se encontrar exemplos de seu uso em diversos níveis e ramos da indústria mundial. Empresas no ramo de gestão de dados na internet, como Google e Yahoo, utilizam massivamente o Python para desenvolvimento *web* (Borges, 2010). Empresas de animação, como Industrial Light & Magic, Pixar e Disney, utilizam a linguagem no processo de criação de filmes de animação 3D (Lutz, 2009). Desenvolvedoras de eletrônicos, como Hewlett & Packard (HP), Qualcomm e IBM utilizam o Python em suas rotinas de teste de *hardware* (Srinath, 2017).

2.4.4.1 Módulo *face_recognition* para Python

O projeto de nome "A API de Reconhecimento Facial mais simples do mundo para Python e Linhas de Comando" (*The world's simplest facial recognition api for Python and the command line*), ou simplesmente conhecido pelo nome genérico de "Face Recognition", criado por Adam Geitgey em meados de 2017 é uma extensão (ou módulo) de Python, compatível com as versões 2.7 e de 3.3 em diante, que pode ser utilizado tanto integrado a um algoritmo como em execução direta de API por linha de código. Na Figura 2.16, apresenta-se um dos exemplos de aplicação mostrados na documentação do projeto, onde a rotina de reconhecimento é aplicada em uma imagem com dois rostos (um conhecido e um desconhecido), retornando na saída ambos os rostos, que posteriormente seriam classificados (o rosto conhecido receberia o rótulo correspondente do banco de dados e o rosto desconhecido receberia o rótulo "unknown").

Trata-se de um algoritmo modularizado, encapsulado em um módulo *Python* (equivalente às bibliotecas da linguagem C), que se baseia nos recursos de codificação facial e

Figura 2.16: Exemplo de aplicação em Python do módulo *Face_Recognition*.


```
import face_recognition
image = face_recognition.load_image_file("your_file.jpg")
face_locations = face_recognition.face_locations(image)
```

FONTE: Adaptado de Geitgey (2017)

reconhecimento facial contidos na biblioteca Dlib. Por conta dessa modularização, apresenta duas vantagens em relação aos próprios algoritmos-exemplos de reconhecimento facial disponibilizados no Dlib. Em primeiro lugar, a rede neural residual já se encontra configurada e treinada, o que poupa o tempo que seria despendido treinando a rede e o esforço que seria feito para estabelecer um banco de imagens de treinamento. Além disso, sua estrutura é planejada para ser utilizada como apenas algumas definições de classes e chamadas de funções, de modo que as aplicações que utilizem este módulo sejam transparentes ao código que é executado nele em segundo plano.

2.4.4.2 Flask

O Flask é um *framework* minimalista desenvolvido para *Python* com propósito de criar aplicações de rede simples. Este *framework* é baseado no kit de ferramentas *Werkzeug* e no padrão de rede para Python, o WSGI (Pallets, 2010).

Uma das vantagens associadas com o Flask é sua curva de aprendizado ser relativamente baixa em comparação com alternativas no Python, bem como o fato de sua arquitetura ser muito explícita, o que melhora a legibilidade e, também dessa forma, facilita a compreensão do código.

2.5 Padrões e Protocolos de Rede

Nesta seção, estão descritos alguns dos padrões e protocolos de rede utilizados na RSSF desenvolvida, em especial, no trecho pertinente ao servidor de rede da RSSF (descrito no próximo capítulo).

2.5.1 JSON

O JSON (*JavaScript Object Notation* ou Notação de Objetos JavaScript) é um estilo de formatação para troca de dados baseado no padrão para JavaScript ECMA-262.

Uma de suas características mais notáveis é seu alto nível de legiblidade, o que facilita tanto na leitura quanto na escrita neste formato pelo usuário sem sacrificar as capacidades de geração e análise feitas por *software*. Outro ponto forte do JSON é independer da linguagem de programação e, não somente isso, utilizar convenções de sintaxe comuns a diversas linguagens da família C, o que facilita sua incorporação nestas linguagens (Crockford, 2001).

A sintaxe do JSON é descrita como uma composição de uma ou mais iterações de duas estruturas específicas (ECMA, 2017):

- Um objeto, definido como um conjunto envolvido por chaves e separado por vírgula de pares de nomes (chamados por vezes de chaves) e valores separados por dois pontos;
- Um vetor, definido como uma lista ordenada envolvida por colchetes e separada por vírgula de valores.

Um nome ou chave no JSON precisa ser, necessariamente, uma *string* que seja única no contexto de seu conjunto (chaves repetidas podem ser utilizadas em escopos diferentes de um mesmo JSON, mas não no mesmo escopo).

Um valor no JSON pode ser dos tipos nulo, booleano (falso ou verdadeiro), numérico, *string*, *array* ou objeto. Por conta desses dois últimos tipos, a estrutura do JSON se torna encadeável.

2.5.2 Protocolos HTTP e HTTPS

O Protocolo de Transferência por Hipertexto (HTTP, *Hypertext Transfer Protocol*) é um protocolo de comunicação muito utilizado na transferência de dados em rede. Sua aplicação mais conhecida é a transferência de dados entre cliente e servidor para a exibição de páginas de *sites* na rede.

Sua estrutura é baseada em requisições e respostas, cada uma com seu respectivo formato (WDN Web Docs, 2022).

As requisições são compostas pelos seguintes elementos:

- Um método HTTP, geralmente um verbete curto como GET, POST, OPTIONS HEAD;
- O caminho do recurso a ser acessado;
- A versão do protocolo HTTP;
- *Headers* com informações adicionais para o servidor (opcional);

- Um corpo contendo os dados a serem enviados (para métodos que assim o fazem, como o POST).

Por sua vez, as respostas dessas requisições são compostas por:

- A versão do protocolo HTTP;
- Um código de estado único, de três dígitos, que indica se a requisição foi bem sucedida ou não e caso não, indica a causa da falha;
- Uma mensagem de estado, descrevendo de maneira sintética o comportamento representado pelo código de estado retornado;
- *Headers* com informações adicionais para o cliente (opcional);
- Um corpo contendo os dados a serem retornados (para métodos que assim o fazem, como o GET).

Um problema associado com a segurança do HTTP nesta estrutura de requisições é que o hipertexto enviado entre cliente e servidor é explícito no corpo da requisição. Isso permite que usuários maliciosos manipulem externamente requisições para acessar dados indevidos ou inserir dados indesejados no servidor.

Uma solução para isso é o HTTPS, que utiliza criptografia, tipicamente via certificados com tecnologia Camada de Sockets Segura (SSL - *Secure Sockets Layer*) ou Segurança de Camada de Transporte (TLS - *Transport Layer Security*), para proteger os dados transferidos entre cliente e servidor, dificultando o acesso externo de requisições e respostas. Isso é particularmente interessante para sistemas que lidam com dados sensíveis, como é o caso de sites que realizam transações financeiras (Gaspar, 2021).

2.5.3 Interfaces de Programação de Aplicações

Uma Interface de Programação de Aplicações (API - *Application Programming Interface*) é um mecanismo que permite que dois componentes de software se comuniquem usando um conjunto de definições e protocolos.

Neste contexto, o termo "Aplicação" pode se referir a qualquer tipo de software que desempenhe determinado papel e a "interface" é uma estrutura de comunicação que garanta que esses softwares possam comunicar-se entre si sem a necessidade de protocolos de comunicação próprios para cada conexão específica entre dois destes (AWS, 2019).

Como essa interface tem por objetivo ser um padrão para integração entre vários tipos de aplicações, as características desse padrão precisam estar bastante explícitas, tipicamente apresentadas em uma documentação pública mantida pelos desenvolvedores (Red Hat, 2017).

2.5.4 Arquiteturas REST

No que se refere a APIs de rede, é preferível desenvolver/utilizar arquiteturas de sistema que seguem padrões certificados. Uma das especificações mais utilizadas é a REST (do inglês *Representational State Transfer* ou Transferência Representacional de Estado).

Essa especificação define uma arquitetura de API que se baseia na comunicação via HTTP através de certas funções de comunicação, dentre as quais existem, pelo menos,

as funções GET, PUT, DELETE, com as quais as aplicações-clientes podem acessar os dados da aplicação-servidor (AWS, 2019).

Às APIs que se conformam com a arquitetura REST dá-se no nome de API RESTful. Segundo Fielding (2000), uma API pode ser objetivamente considerada RESTful ao cumprir as seguintes seis restrições de projeto:

- Uma arquitetura cliente-servidor baseada em solicitações via HTTP;
- Não há persistência de dados do cliente no servidor entre as solicitações. Em vez disso, esses dados são mantidos com o cliente;
- Capacidade de armazenamento de *cache* diminuindo o número de interações entre o cliente e o servidor em uma determinada requisição;
- Uma interface uniforme, visto que o objetivo dessa arquitetura é ser o mais abrangente possível. São sugeridas quatro restrições de projeto para atingir este critério: a identificação de recursos, a manipulação de recursos por meio de representações, as mensagens auto-descritivas e as hipermídias como plataforma do estado das aplicações;
- Sistema em camadas, onde as camadas de cliente e servidor podem ser intercaladas por camadas intermediárias que podem oferecer recursos extras;
- Opcionalmente, os servidores também podem ampliar a funcionalidade de um cliente por meio da transferência de códigos executáveis sob demanda.

Além dos critérios que determinam se uma API é RESTful ou não, o projetista também deve levar em consideração um ou mais métodos de segurança para proteger o acesso aos recursos do servidor. Dentre os métodos principais usados para este fim, citam-se dois (AWS, 2019,):

1. Tokens de autenticação: Um valor aleatório gerado no momento em que o cliente solicita acesso ao servidor e tem seu acesso autorizado. Tipicamente são valores criptografados;
2. Chaves de API: Um valor único e específico relacionado a determinada aplicação que a identifica. São menos seguras do que os tokens de autenticação, mas fornecem um controle maior sobre os acessos ao servidor.

3 Metodologia

Descritos neste capítulo encontra-se o sistema proposto, apresentado em detalhes, contendo a RSSF e a aplicação de reconhecimento facial remoto desenvolvidas.

Tanto a RSSF quanto a aplicação são baseadas no sistema proposto em Rodrigues (2019). A arquitetura desenvolvida em tal trabalho era significativamente mais simples por conta de sua aplicação ponto-a-ponto. Em relação a este trabalho original, diversas melhorias foram realizadas tanto no âmbito da otimização do funcionamento como de ajustes necessários para que ele se tornasse uma rede de sensores de fato.

A abordagem principal em torno desta nova arquitetura é a expansibilidade, ou seja, a RSSF foi desenvolvida pensando em ter um desempenho similar independente da escala e que essa variação de tamanho não seja custosa para o usuário da rede.

Optou-se por manter o fluxo de transmissões simples do sistema ponto-a-ponto, ou seja, sem dispositivos intermediários, mesmo em uma estrutura de rede. Esta abordagem não é atípica, no entanto, sendo um princípio relativamente comum em redes de topologia estrela, como é o caso da RSSF desenvolvida. O objetivo dessa escolha é aumentar essa escalabilidade ao, removendo as interações entre dispositivos da mesma etapa, também remover a necessidade de haver um controle de quantos dispositivos próximos compõem a mesma etapa. Dessa forma, na rede desenvolvida, os dispositivos de uma mesma etapa são não precisam se preocupar em gerenciar dispositivos semelhantes ao seu redor. Isso ajuda a permitir com que a quantidade de dispositivos alocados numa mesma função ao longo da rede não interfira em seu desempenho, já que não realizando esse controle, a única coisa que limita a capacidade real de uma rede é a capacidade do *hardware* utilizado.

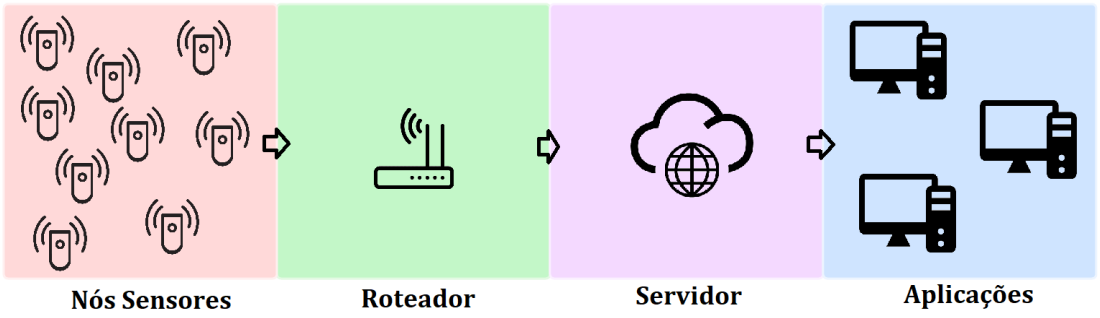
A arquitetura de rede desenvolvida para este trabalho foi denominada arquitetura em camadas e muito se inspira na estrutura típica da arquitetura LoRaWAN, porém se diferencia nos protocolos de comunicação utilizados entre os dispositivos de cada camada. Outra diferença notável é a ausência de um protocolo MAC para comunicação entre os dispositivos da RSSF, tendo sido utilizado um método de validação de endereços atrelado à estrutura dos próprios pacotes transmitidos.

Dentre as implementações desenvolvidas com base no sistema original de Rodrigues (2019), incluiu-se um servidor dedicado para gerenciar os dados capturados pela RSSF e direcioná-lo à interface da aplicação. Dessa forma, o sistema desenvolvido não é mais composto por três estágios, com um nó sensor, um receptor e uma aplicação interligada, mas um sistema composto por quatro estágios com nós sensores, *gateways* e um servidor de rede interligados e uma aplicação acessando remotamente as informações publicadas no servidor, como é típico em RSSFs.

3.1 Descrição da Rede

Em princípios gerais, a ideia da RSSF desenvolvida se baseia no conceito clássico deste tipo de rede, onde um conjunto de nós sensores propaga seus dados a um nó sorvedouro (*sink node*), que transmite estes dados remotamente para a aplicação através da Internet, como ilustrado na Figura 3.1. A arquitetura da rede, no que se refere ao arranjo dos nós sensores, utilizou a topologia de rede Estrela, a topologia mais comumente utilizada com a tecnologia LoRa (LoRa Alliance, 2019).

Figura 3.1: Diagrama da RSSF proposta.



FONTE: O Autor (2022)

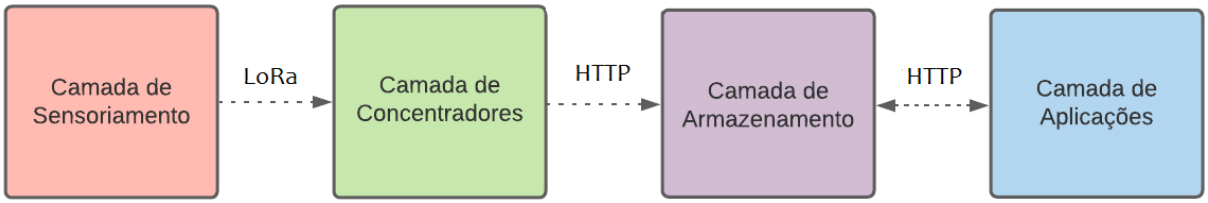
Seu processo de operação é autônomo, sendo a coleta de dados ativada por um gatilho na forma de um sensor piroelétrico de movimento. O nó sensor deve ter em suas configurações tanto o endereço do *gateway* com o qual irá se comunicar através da modulação LoRa quanto o identificador de aplicação, que corresponde ao local no servidor de rede onde os dados serão publicados para serem adquiridos pela aplicação.

As comunicações com o servidor de rede da RSSF são feitas através de requisições HTTP tanto do lado dos *gateways*, que fornecem os dados, quanto da aplicação, que os adquire. Por conta disso, apesar do servidor ser um componente integrante da RSSF, sua interação com os *gateways* não precisa necessariamente de uma conexão física, assim como ocorre com a interação os nós sensores e os *gateways*. Essa integração remota permite o acesso múltiplo e simultâneo tanto de *gateways* quanto da aplicação ao servidor.

A aplicação desenvolvida, bem como toda a interface de rede e a etapa de processamento do *gateway* foram todas escritas na linguagem Python. Por sua vez, os algoritmos dos nós sensores e da parte embarcada do *gateway* foram projetados na linguagem C.

No que diz respeito à sua estrutura, a RSSF foi subdividida em estágios que foram denominados de "camadas" (Figura 3.2). Essas camadas podem ser definidas como um conjunto de dispositivos realizando uma mesma função, que não interagem entre si, mas comunicando-se com as camadas adjacentes através de tecnologias de comunicação específicas. Tal definição está de acordo com a escalabilidade proposta para a rede. As quatro camadas desenvolvidas são, pela ordem do fluxo de dados: a camada de sensoriamento, contendo os nós sensores; a camada de concentradores, contendo o(s) *gateway*(s); a camada de armazenamento, contendo o servidor de rede; e a camada de aplicações, contendo as aplicações atribuídas à rede.

Figura 3.2: Estrutura de camadas da RSSF proposta.



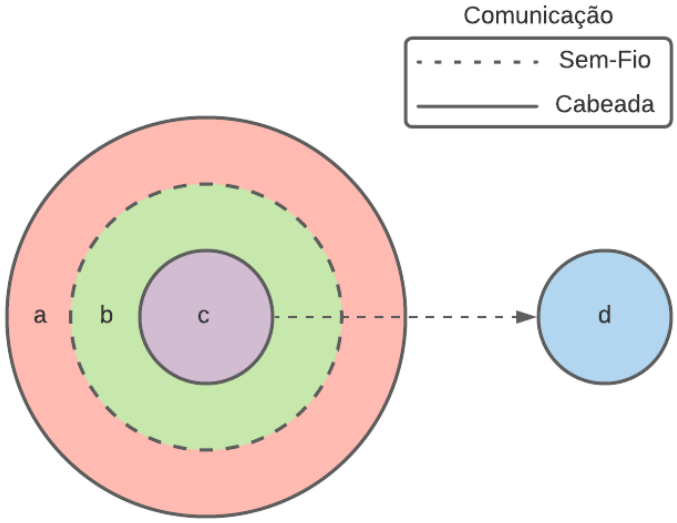
FONTE: O Autor (2022)

A quantidade de elementos em cada camada está diretamente relacionada com o tamanho da rede que se deseja implementar. Tipicamente, a camada de sensoriamento é a

que possui mais elementos, seguida da camada de concentradores e por fim da camada de armazenamento. A única camada que não segue esta tendência é a camada de aplicações, em que o número de elementos nesta representa a quantidade de aplicações associadas a uma determinada rede, não havendo relação direta com o tamanho da mesma.

Dessa forma, é possível visualizar a disposição organizacional da rede como um conjunto concêntrico das três primeiras camadas interligado à quarta e última camada (d), como observado na Figura 3.3. Nesse tipo de disposição, vários elementos da camada mais externa (a) podem estar ligados a um único elemento da camada intermediária (b), ao passo que vários elementos da camada intermediária se comunicam com um único elemento da camada interna (c), de modo que o fluxo de dados da rede se concentra nesta, ou seja, na camada de armazenamento.

Figura 3.3: Forma alternativa de visualizar as camadas da rede proposta.



FONTE: O Autor (2022)

Este tipo de visualização corresponde a uma forma abstrata de observar a aplicação da já mencionada topologia em estrela, na qual todos os elementos externos transmitem dados de maneira convergente a um elemento central.

Na implementação da RSSF desenvolvida, bem como em sua respectiva validação, apresentada nos capítulos posteriores deste trabalho, observa-se uma versão em escala mínima (em estágio de produto viável mínimo), composta por dois nós sensores conectados a um único *gateway*, ligado por sua vez a um único servidor, que recebe solicitações da aplicação de monitoramento remoto por imagem.

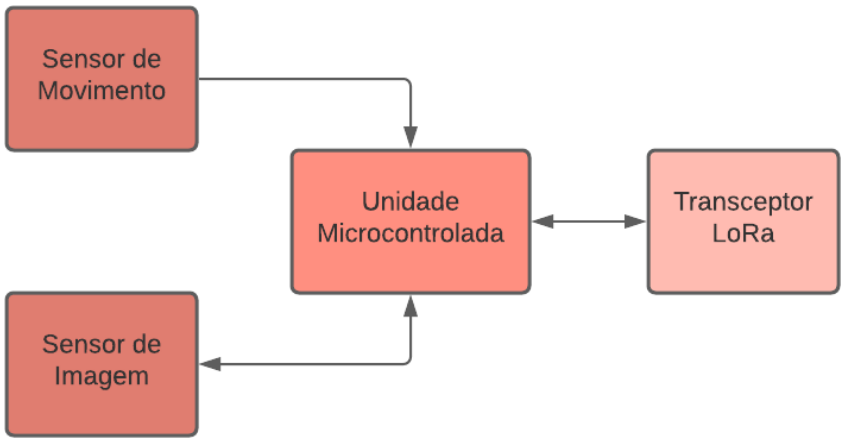
3.1.1 Camada de Sensoriamento

Esta camada tem como elemento definidor o dispositivo conhecido como nó sensor. É o nó sensor que de fato dá sentido ao nome de "rede de sensores". Cada nó sensor tem o papel de adquirir dados em um conjunto de condições específico. Em RSSF clássicas, geralmente essas condições são geo-posicionais, com sensores realizando a mesma medição em posições distintas do ambiente e/ou elemento a ser sensoreado. Não é incomum que uma rede de sensores realize medidas também com configurações distintas e/ou em intervalos de tempo distintos, a depender do comportamento que se deseja capturar.

O objetivo desta camada é coletar os dados brutos a serem utilizados, muito embora seja possível implementar estágios de pré-processamento no próprio nó sensor, já que, por definição, estes dispositivos possuem capacidade de processamento própria. De fato, nós sensores não são pura e simplesmente elementos sensores, mas um dispositivo criado da combinação entre um ou mais sensores, uma interface de processamento e uma ou mais interfaces de comunicação sem fios.

Na RSSF desenvolvida, o nó sensor presente nesta camada é um dispositivo composto por quatro componentes, como ilustrado no diagrama da Figura 3.4.

Figura 3.4: Diagrama de componentes do nó sensor proposto.

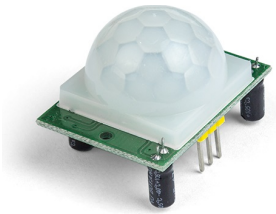


FONTE: O Autor (2022)

3.1.1.1 Sensor de Movimento

O Sensor de movimento utilizado foi o módulo eletrônico modelo HC-SR501 (Figura 3.5). Este módulo é baseado no sensor piroelétrico LHI778. O tempo de reaquecimento mínimo do sensor, ou seja, o menor tempo possível entre duas detecções, segundo a documentação, é de cerca de 5 s (MPJA, s.d.).

Figura 3.5: Módulo de sensor piroelétrico HC-SR501.



FONTE: RoboCore (2019)

O sensor possui um alcance de até 7 m em um ângulo de 120^o e uma entrada de alimentação de 5 a 20 V e uma saída lógica de 3,3 V, sendo assim compatível com a maioria dos dispositivos embarcados.

3.1.1.2 Sensor de Imagem

O sensor de imagem utilizado foi o módulo eletrônico modelo ArduCAM Mini 2MP (Figura 3.6), baseado no sensor de imagem OV2640, com resolução de até 2 MP e conversão interna para JPEG.

Figura 3.6: Módulo do sensor de imagem ArduCAM.

FONTE: ArduCAM (2019)

O módulo conta com interfaces de dados e controle independentes, para garantir uma captura de imagem ininterrupta, sendo a primeira interface baseada em SPI e a segunda baseada em I2C. Com um total de 8 pinos, os quatro primeiros são relativos interface SPI (pinos 1 a 4), seguidos de 2 pinos de alimentação (pinos 5 e 6) e, por fim, de dois referentes à interface I2C (pinos 7 e 8).

3.1.1.3 Unidade Microcontrolada

A Unidade Microcontrolada (UMC) é responsável por todo o processo de integração entre as demais unidades. A UMC de escolha foi a placa de prototipagem Arduino Mega2560 (Figura 3.7), baseada no microcontrolador ATMega2560. A placa possui 256 kB de armazenamento *Flash* e uma memória SRAM de 8 kB, além de contar com 54 pinos digitais (sendo 16 capazes de gerar PWM) e 16 pinos analógicos. Essas características, bem como sua relativa disponibilidade, são um atrativo para a implementação neste projeto, em especial a alta quantidade de pinos disponíveis na placa, o que permitiu a conexão com todos os módulos utilizados e ainda manteve capacidade para expansão, caso a utilização de outros módulos fossem necessários. Em contrapartida, comparativamente a modelos mais simples de Arduino, como o modelo UNO, esta capacidade maior de processamento e de interface de entrada/saída é refletida no custo de mercado da placa, o que impacta diretamente no custo total do nó sensor e da RSSF como um todo.

A rotina de execução do algoritmo do nó sensor, executada pela UMC, pode ser visualizada no fluxograma apresentado na Figura 3.8.

A arquitetura do algoritmo é organizada de modo que o nó sensor sempre se encontra em uma de seis etapas de operação; são eles, em ordem: estado ocioso (*idle*), capturando quadro de imagem, solicitando transmissão, calculando pacotes, transmitindo pacotes e finalizando transmissão.

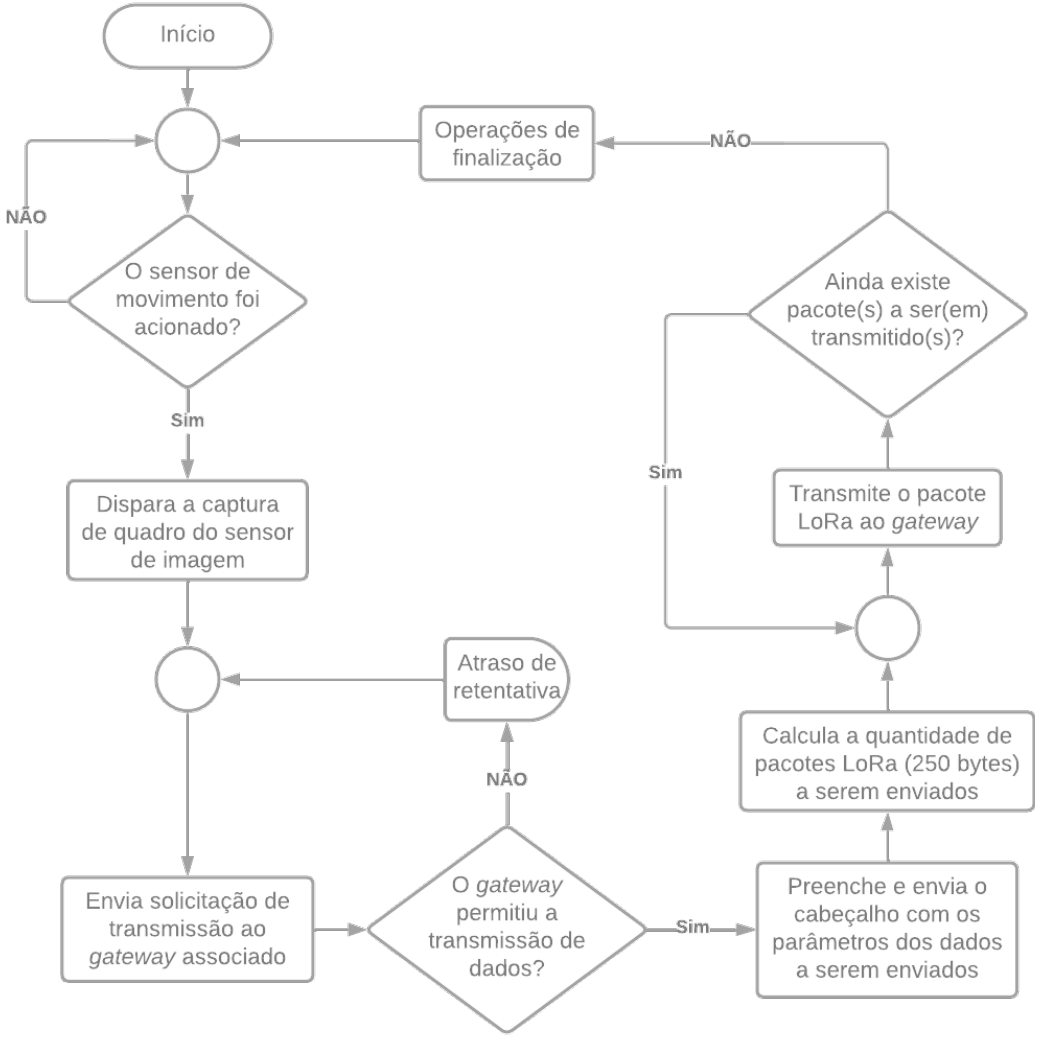
Por padrão, após ser inicializado ou logo após uma transmissão, a UMC encontra-se sempre na primeiro etapa, que se caracteriza por um laço ocioso, interrompido assim que o sensor de presença envia qualquer sinal de nível lógico `1` à UMC.

Imediatamente após receber esse sinal, a UMC entra na segunda etapa, onde envia através do barramento I2C um comando de captura de quadro ao sensor de imagem e

Figura 3.7: Placa de prototipagem Arduino Mega2560 Rev3.

FONTE: Arduino Company (2019)

Figura 3.8: Fluxograma do algoritmo do nó sensor desenvolvido.



FONTE: O Autor (2022)

aguarda inoperante até que o sensor de imagem retorne uma *flag* indicando o resultado do processo de captura (seja sucesso ou falha). O processo de captura de quadro em si tem uma latência estimada de 66,63 ms para a máxima resolução do sensor, de modo que essa etapa tem uma duração tipicamente inferior a 70 ms.

Caso o resultado da captura seja um sucesso, o nó sensor entra em sua terceira etapa, composto por um segundo laço perpétuo, onde o nó sensor realiza uma transmissão de pacote de solicitação ao *gateway* em que se encontra associado e aguarda uma resposta. Caso dentro de um intervalo de tempo configurado o nó sensor não receba uma resposta do *gateway* associado ou receba dele uma resposta que não seja uma autorização, o nó sensor reinicia o laço enviando mais uma solicitação de transmissão. Somente ao receber uma resposta autorizando a transmissão (descrito em 3.1.2), ele prossegue para a etapa seguinte.

Na quarta etapa, o nó sensor monta o pacote de cabeçalho da transmissão, contendo informações dos dados a serem transmitidos que serão usados nas camadas seguintes. Após montado, o pacote é enviado ao *gateway* e, em seguida, o nó sensor calcula quantos pacotes de mensagem serão necessários para enviar o conteúdo serializado do quadro capturado pelo sensor de imagem.

Após calculada a quantidade de pacotes, o nó sensor entra em sua quinta etapa, entrando em um terceiro laço perpétuo onde realiza o processo de preencher sequencialmente os pacotes de mensagem com o conteúdo da imagem, armazenado no *buffer* do sensor de imagem e enviá-los ao *gateway* associado. O nó sensor realiza essa leitura e envio em iterações de 250 bytes de conteúdo até que ou a quantidade de pacotes calculada previamente seja ultrapassada ou seja identificado no conteúdo da mensagem a sequência que define o fim do arquivo de imagem (`EOF`), o que ocorrer primeiro.

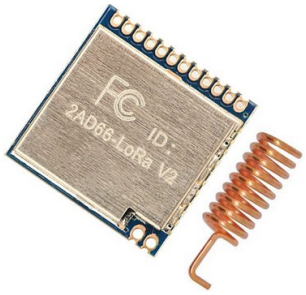
Ao sair da condição de transmissão dos pacotes de mensagem, o nó sensor entra em sua sexta e última etapa, onde realiza as operações necessárias para que possa ocorrer uma nova captura e transmissão, como limpar o armazenamento temporário dos pacotes e as estruturas de dados utilizadas, enviar comandos para os módulos LoRa e de Imagem para limpar seus respectivos armazenamentos internos e acionar novamente a detecção de sinal do sensor de movimento. Ao final dessa etapa, o nó sensor é novamente colocado em seu laço ocioso da primeira etapa, onde pode aguardar por uma nova detecção piroelétrica e iniciar um novo ciclo.

No estado atual de desenvolvimento do sistema, o nó sensor não verifica se houve falhas na transmissão de pacotes, de modo que não existe um fluxo de retentativas a nível de cada pacote individual, somente da imagem capturada como um todo. Uma única verificação dos dados é realizada após a transmissão de todos os pacotes, pelo *gateway*, checando se o conjunto de dados é um arquivo JPEG válido. Isso significa que perdas tanto de bit quanto de pacote podem comprometer a integridade transmissão da imagem de duas formas: se o primeiro e/ou o último pacote forem perdidos, invalidando a verificação de formato de arquivo e se algum pacote intermediário for perdido, distorcendo o conteúdo da imagem reconstruída. Este problema pode ser corrigido futuramente com a adição de um sistema de verificação a nível de pacotes e o uso de técnicas de checagem de mensagem, como o CRC.

3.1.1.4 Transceptor LoRa

A comunicação do sistema foi baseada na modulação LoRa. O módulo escolhido para esta função foi o 2ad66-LORAV2 (Figura 3.9), da empresa chinesa NiceRf Wireless Technology.

Figura 3.9: Módulo de comunicação LoRa 2ad66 da NiceRf.

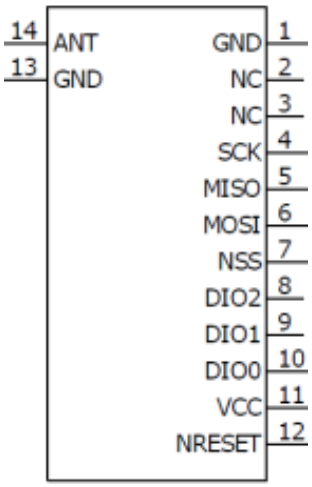


FONTE: Saravati (2022)

Este dispositivo é baseado no chip LoRa SX1276, da Semtech, operando na faixa de frequência móvel de 915 MHz. Realiza a recepção de dados através de uma interface de comunicação SPI de 3,3 V, podendo armazenar até 256 *bytes* de dados em seu *buffer* interno. Segundo a documentação do módulo, seu alcance máximo é atingido em visada direta utilizando um consumo de 120 mA de corrente a 20 dBm de potência.

O módulo conta com catorze pinos (Figura 3.10), sendo três de alimentação (pinos 1, 11 e 13), dois pinos não utilizados (pinos 2 e 3), um pino de de interrupção (pino 12), um pino de conexão com a antena (pino 14), três pinos de entrada/saída digital (pinos 8, 9 e 10) e quatro pinos para a interface SPI (pinos 4 a 7).

Figura 3.10: Diagrama de pinos do 2ad66-LORAV2.



FONTE: NiceRF (2020)

Na RSSF implementada, uma vez que o *buffer* do sensor de imagem tem uma saída serial, o acesso ao *buffer* de dados do módulo LoRa foi realizado utilizando um único pino (pino 10 ou DIO0), para não interferir na sequência dos bytes transferidos de um *buffer* a outro. Dessa forma, bem como os pinos 2 e 3, os pinos 8 e 9 também não foram utilizados.

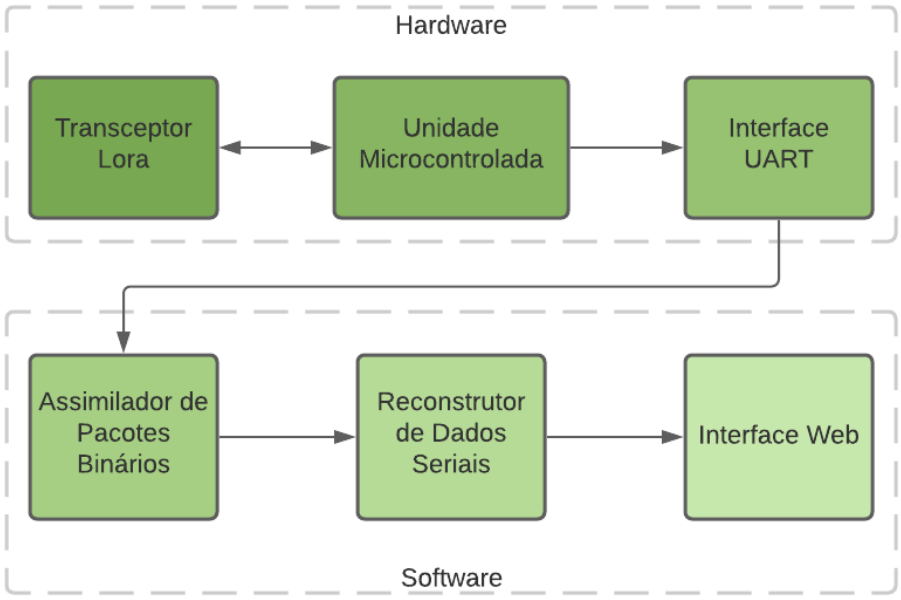
3.1.2 Camada de Concentradores

Esta camada tem como elemento característico o *gateway*. Seguindo a topologia de RSSF do tipo estrela, o *gateway* é um dispositivo que pode receber dados de múltiplos

nós sensores e tem como função concentrar os vários fluxos de dados desses nós sensores em um único caminho. Dependendo da escala de tamanho da rede, pode haver um único *gateway* gerenciando todos os nós sensores ou múltiplos *gateways* gerenciando cada conjunto de nós sensores. Em redes mais elaboradas também é possível observar um roteamento inteligente, em que alguns *gateways* podem compartilhar nós sensores e o Nó tem a opção de escolher por qual *gateway* enviar seus dados de modo a otimizar a transmissão. Também existe a possibilidade de *gateways* secundários ou mesmo de ordens maiores, cujo papel é concentrar os dados de outros *gateways*, em contato com os nós sensores, sempre com o objetivo de convergir todas as rotas para um único destino final (uma central de armazenamento e/ou processamento).

O *gateway* desenvolvido para a RSSF proposta (Figura 3.11) consiste de um elemento físico (hardware) e um elemento abstrato (software). Aqui, o elemento físico é responsável por realizar a comunicação com os nós sensores da camada de sensoriamento, o que é feito através dos dispositivos embarcados que compõem o *gateway*, ao passo que o elemento abstrato é responsável por se comunicar com o servidor de rede da camada de armazenamento, através de um *script* Python sendo executado em um computador ou microcomputador.

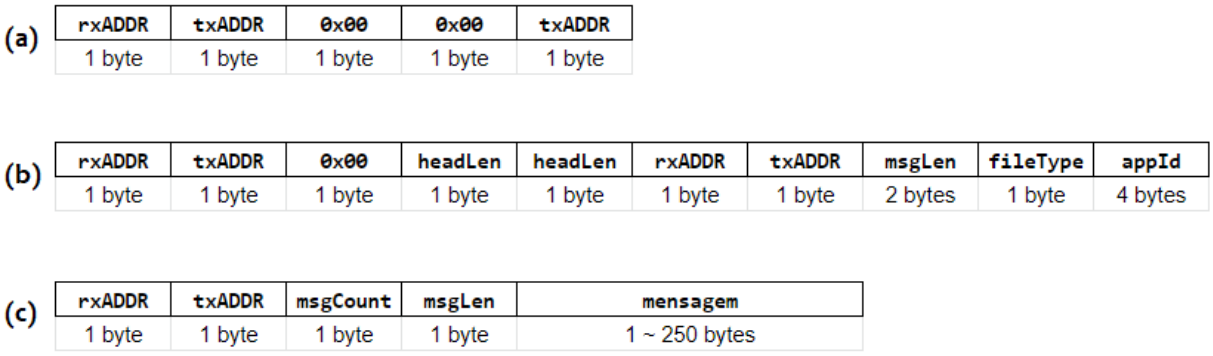
Figura 3.11: Diagrama de componentes do *gateway* desenvolvido.



FONTE: O Autor (2022)

O elemento físico realiza a recepção dos dados enviados pela camada de sensoriamento utilizando dois componentes: um módulo LoRa e uma UMC. Os dados recebidos pelo elemento físico são enviados diretamente ao elemento abstrato através de uma interface serial UART, pela qual a UMC se comunica com o elemento abstrato. Durante o processo de comunicação, três tipos de pacotes são enviados para o *gateway*: o primeiro tipo é o dos pacotes de solicitação de transmissão (Figura 3.12.(a)), enviados somente quando um nó sensor realizou uma captura e ainda não a enviou a um *gateway*; o segundo tipo é o dos pacotes de cabeçalho (Figura 3.12.(b)), enviados uma vez antes de cada transmissão de dados; o terceiro e último tipo é o dos pacotes de mensagem (Figura 3.12.(c)), que são pacotes com um conteúdo de até 250 bytes que podem conter uma parte ou a totalidade do conteúdo em bytes da imagem transmitida.

Figura 3.12: Tipos de pacotes recebidos pelo *gateway*.



FONTE: O Autor (2022)

Para realizar a comunicação LoRa entre o *gateway* da camada de concentradores e os nós sensores da camada de sensoriamento, o módulo LoRa utilizado foi exatamente o mesmo utilizado nos nós sensores (visto na Seção 3.1.1.4), sendo também configurado da mesma forma. O padrão utilizado na transmissão de pacote interage com o algoritmo de recepção de dados presente na UMC em um processo que foi denominado de "Rotina de Gerenciamento de Transmissão".

Essa rotina se dá através de uma sequência de laços encadeados controlados por funções de leitura distintas, que são compatíveis apenas com seus respectivos tipos de pacote. Esta arquitetura permite que um nó sensor bloqueie o *gateway* em seu processo de transmissão até que este seja concluído (ou caso alguma falha ocorra na transmissão, até que o tempo de estouro seja excedido), impedindo que outros nós sensores transmitam a este *gateway* específico e, mesmo que por alguma exceção específica um segundo transmissor consiga interferir na transmissão do primeiro com algum pacote (que quase certamente seria um pacote de solicitação), a UMC simplesmente ignora o pacote e não o transmite ao elemento abstrato do *gateway*.

Por conta dessa rotina, foi possível otimizar o uso da UMC como receptor, já que elimina a necessidade de um gestor de transmissões baseado em sistema operacional ou mesmo de algum tipo de pré-processamento e/ou armazenamento temporário (*buffering*) da transmissão no dispositivo o que, tratando-se de um dispositivo embarcado, poderia implicar em um sério gargalo computacional, dada suas restritas capacidades tanto de armazenamento quanto de processamento.

Ainda são necessários, no entanto, testes com um número elevado de nós sensores transmitindo simultaneamente para determinar o comportamento do gerenciador neste caso extremo, visto que, apesar da estrutura de leitura do *gateway* ignorar os pacotes incorretos no sentido de não realizar nenhum processamento naqueles dados, estritamente, o *gateway* ainda recebe cada um dos pacotes que lhe é enviado, de modo que existe algum atraso na operação do mesmo relacionado à quantidade de transmissões paralelas de nós sensores solicitantes e que cuja influência ainda precisa ser estudada.

De todo modo, os dados recebidos pelo elemento físico são repassados através de comunicação UART ao elemento abstrato do *gateway*, que consiste em um algoritmo que recebe o conjunto de dados binários transmitido através dos pacotes de mensagem e os processa. Esse processamento consiste em anexar as mensagens de cada pacote em um único vetor de dados e recuperar seu formato de arquivo original (o padrão é JPEG, utilizado pela aplicação, porém o *gateway* também processa arquivos do tipo PNG) através

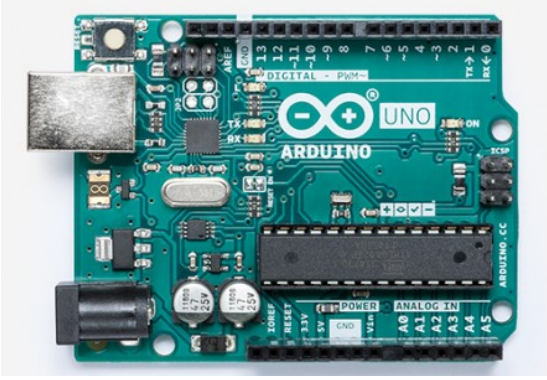
dos dados de controle enviados no pacote de cabeçalho.

Também através do pacote de cabeçalho, são obtidas as informações utilizadas na geração do pacote de transmissão para a camada de armazenamento, no formato JSON, como o nome identificador do arquivo, o tipo e formato de arquivo, a data que foi transmitido, as aplicações que o utilizam e o endereço de destino (*endpoint*) pelo qual poderá ser acessado pela aplicação.

3.1.2.1 Unidade Microcontrolada

Para o papel da UMC, escolheu-se a placa de prototipagem Arduino UNO (Figura 3.13). Sua escolha ocorreu pela ampla disponibilidade do dispositivo e sua fácil utilização. Todavia, qualquer dispositivo com comunicação UART e compatível com o módulo 2ad66 (possua interface SPI) poderia ter sido utilizado como UMC do *gateway*.

Figura 3.13: Placa de prototipagem Arduino Uno Rev3.



FONTE: Arduino Company (2019)

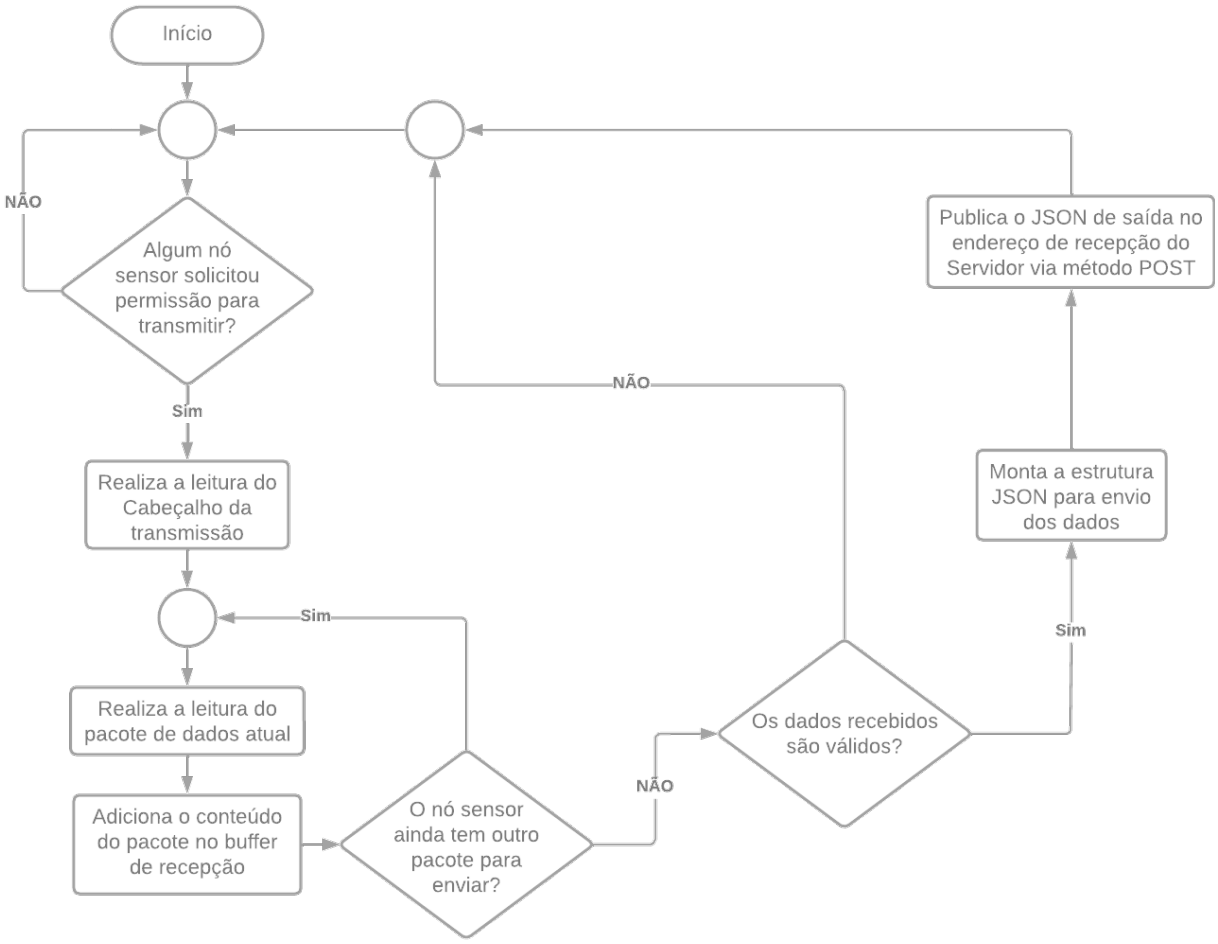
O fluxo do algoritmo executado pelo *gateway* pode ser observado na Figura 3.14.

Durante a execução do algoritmo do *gateway*, dois códigos são executados em laço simultaneamente: o primeiro código é responsável por receber os dados e é executado na UMC, ao passo que o segundo código é responsável por processar os dados recebidos e é executado em um computador ou microcomputador compatível com linguagem Python (neste caso, um computador pessoal foi utilizado). Estes códigos interagem entre si através de uma interface serial UART.

No primeiro código, o estado padrão da UMC é um laço de escuta passiva, que permanece "livre" para receber transmissões de outros dispositivos. Neste primeiro laço, a UMC aguarda algum pacote LoRa enviado a ela. Assim que recebe um pacote, a UMC identifica se o pacote é válido (está no formato adequado e se o endereço de destino corresponde a seu próprio endereço) e se contém uma requisição para transmitir.

Caso ambos os critérios sejam atingidos, a UMC repassa a requisição para o *script* de processamento e inicia o processo de autorização, verificando se o endereço do nó sensor solicitante está incluído na lista de aceite daquele *gateway*. Caso seja este o caso, o *script* de processamento retorna uma afirmativa e a UMC repassa essa afirmativa na forma de uma mensagem que o nó sensor interpreta como um aceite. Caso o endereço solicitante não esteja incluído na lista do *script* de processamento, a UMC transmite uma mensagem de recusa ao nó sensor. Por fim, caso um ou ambos dos critérios não sejam atingidos na verificação pela UMC, esta simplesmente não transmite uma resposta ao nó sensor.

Figura 3.14: Fluxograma do algoritmo do *gateway* desenvolvido.



FONTE: O Autor (2022)

Se a UMC enviou uma resposta à solicitação de um nó sensor e essa resposta foi afirmativa, a UMC entra em um segundo laço para recepção de pacotes de cabeçalho e de dados (a UMC não distingue e nem precisa distinguir entre estes dois pacotes).

Neste segundo laço, a UMC transmite os dados diretamente ao *script* de processamento, até que uma sequência que indique o fim do arquivo seja encontrada, momento no qual a UMC retorna ao seu estado ocioso original. Durante a execução deste laço, a UMC ignora pacotes no formato de solicitação de transmissão.

3.1.2.2 Script de Processamento

O elemento abstrato do *gateway* tem um papel secundário no primeiro laço de execução (escuta de solicitações) e um papel principal no segundo laço (recepção e processamento dos pacotes).

Em seu primeiro laço, o *script* aguarda o primeiro byte a ser enviado no barramento da porta UART. Este byte necessariamente é o endereço de um nó sensor solicitante, enviado pela UMC após verificar que a solicitação é válida. Ao receber este valor, o *script* de processamento o compara com uma lista de valores hexadecimais correspondente a todos os endereços de nós sensores permitidos para aquele gateway. Esta comparação usa uma lógica booleana minimamente suficiente, ou seja, retorna um valor verdadeiro (*true*) caso pelo menos um dos valores da lista corresponda ao valor lido e um valor falso (*false*)

caso nenhum dos valores da lista tenha correspondência.

Caso o retorno da comparação seja falso, o *script* escreve no barramento serial a *string* "NO", que a UMC interpreta como uma recusa da solicitação. Caso seja verdadeiro, o *script* escreve "OK", que é interpretado pela UMC como um aceite.

Em seguida, o *script* de processamento entra em seu segundo laço, lendo o barramento serial pelos próximos 10 bytes. Esses 10 bytes são relativos ao primeiro pacote de dados repassado pela UMC, que corresponde ao cabeçalho da transmissão enviado pelo nó sensor.

Através desse cabeçalho, o *script* de processamento determina o tamanho da mensagem a ser lida e realiza uma leitura contínua do barramento serial até atingir uma quantidade de bytes lidos igual a esse tamanho ou até encontrar a sequência de fim de arquivo, o que ocorrer primeiro.

Em seguida, o *script* de processamento verifica, baseado no formato esperado de arquivo enviado no cabeçalho, se os dados recebidos da UMC são válidos. Em caso afirmativo, os dados são formatados em uma estrutura JSON, com a adição de alguns atributos pertinentes, como mostrado na Figura 3.15. Em caso negativo, os dados são descartados e o fluxo de processamento é abortado.

Figura 3.15: JSON contendo dados enviados à camada de armazenamento.

```
{
  "publisher": senderName,
  "filename":  fileName,
  "format":    fileFormat,
  "timestamp": creationTime,
  "dataGroup": dataGroup,
  "appId":     appId,
  "fileSize":  length,
  "data":      data
}
```

FONTE: O Autor (2022)

Esse JSON então é enviado ao servidor de rede presente na camada de armazenamento através de sua rotina de publicação, baseada no método HTTP POST.

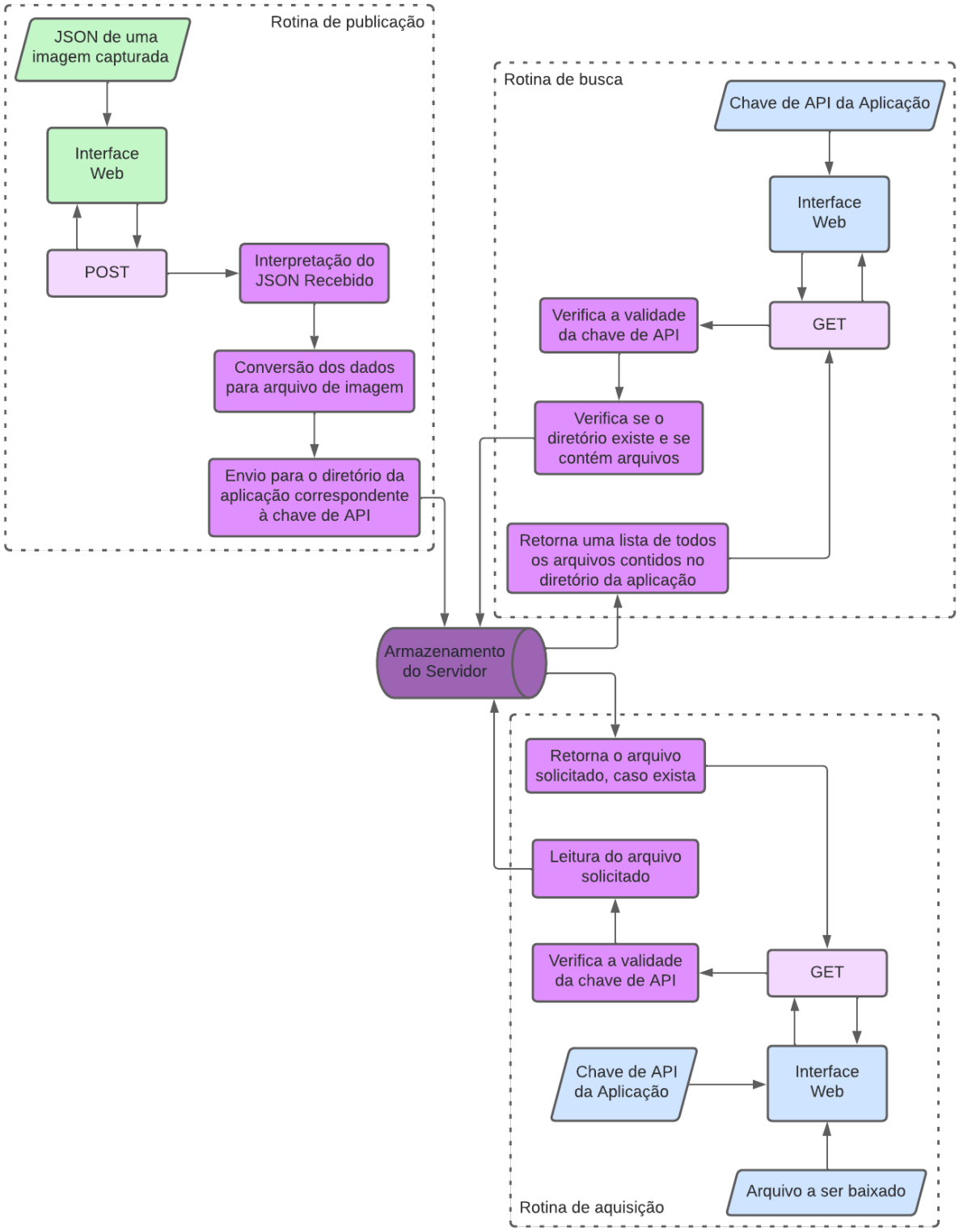
Após o envio do JSON à camada de armazenamento, o *script* de processamento retorna a seu laço inicial e aguarda o recebimento de um novo endereço de nó sensor solicitante.

3.1.3 Camada de Armazenamento

A camada de armazenamento tem como elemento característico o servidor de rede. Esse servidor tem o propósito de armazenar os dados capturados pela RSSF e manter o sistema desacoplado das aplicações. Isso permite que múltiplas aplicações tenham acesso concorrente aos dados sem a necessidade de um algoritmo de gerenciamento mais robusto ou sem a necessidade de um canal direto de comunicação (seja cabeada ou sem fio) entre a RSSF e cada aplicação individualmente, o que permite que a rede possa ser utilizada por uma quantidade virtualmente infinita de aplicações.

O servidor de rede desenvolvido consiste em uma API RESTful de rede desenvolvida na linguagem *Python*, através do *framework* Flask. Esse servidor possui três rotinas de execução: a rotina de publicação, baseada no método HTTP POST, a rotina de busca e a rotina de aquisição, ambas baseadas no método HTTP GET, como visto na Figura 3.16.

Figura 3.16: Fluxo de dados no servidor de rede.



FONTE: O Autor (2022)

3.1.3.1 Rotina de Publicação

A rotina de publicação é iniciada por um *gateway* da camada de concentradores para armazenar a imagem recebida em um fluxo de transmissão no servidor de rede. Através

de uma requisição POST, os dados a serem armazenados são enviados em formato JSON.

Além dos dados contidos no campo ‘‘data’’, são enviados alguns atributos da imagem em outros campos do JSON, como o nome/identificador do nó sensor que capturou a imagem em ‘‘publisher’’, o nome do arquivo em ‘‘filename’’, o formato de arquivo no qual os dados serão salvos em ‘‘format’’, a data de recepção da imagem no *gateway* (já que o nó sensor não possui relógio próprio) em ‘‘timestamp’’, o grupo ao qual os dados fazem parte (para esta aplicação, o grupo é 1, correspondente a imagens, mas o sistema pode ser expandido para suportar dados de outros tipos) em ‘‘dataGroup’’, a chave de API da aplicação em ‘‘appId’’ e o tamanho em bytes da imagem em ‘‘fileSize’’.

Com essas informações, o servidor extrai os dados recebidos, converte-os para o formato de arquivo adequado e os armazena no diretório associado à aplicação que irá utilizá-lo. Cada aplicação possui uma chave de API única e um diretório associado no servidor, de modo que a própria chave indica o destino de armazenamento. Caso o diretório associado com a chave de API não exista, este é criado antes do arquivo ser armazenado.

3.1.3.2 Rotina de Busca

A rotina de busca é uma das duas rotinas realizadas por uma aplicação da camada de aplicações. Através de uma requisição GET, esta rotina tem o propósito de gerar uma lista com novos arquivos a serem adquiridos pela aplicação. Como parâmetros extras dessa requisição, a Chave de API da Aplicação é enviada.

No servidor de rede, a chave de API é validada e, caso válida, é gerada uma lista ordenada de pares, composta pelos valores das datas de criação e os respectivos nomes de todos os arquivos presentes em um diretório dedicado exclusivamente àquela aplicação. Caso não existam arquivos no diretório da aplicação, esta lista é gerada vazia.

Caso a requisição e a rotina sejam bem sucedidas, o servidor retorna à aplicação um código `HTTP 200` (indicando sucesso) e uma lista contendo todos os arquivos presentes no diretório da aplicação como conteúdo.

3.1.3.3 Rotina de Aquisição

Por fim, a rotina de aquisição é iniciada pela aplicação iterativamente, quando a rotina de busca retorna uma lista não vazia de pares. Através de uma requisição GET, esta rotina faz o papel de adquirir um único arquivo do diretório correspondente da aplicação.

Dessa forma, o fluxo iterativo no qual essa rotina é chamada faz com que ela possa ser chamada múltiplas vezes em sequência. Tem como parâmetros de entrada a chave de API da aplicação e o nome do arquivo que se deseja adquirir do servidor, que são sempre diferentes para cada posição da iteração (já que a lista retornada da busca não contém elementos repetidos).

Ao se verificar a validade da chave de API, verifica-se se o diretório associado existe. Se não existir, a requisição é retornada com um erro `HTTP 404` (Não Encontrado). Caso exista, o servidor verifica se o arquivo existe, retornando um erro `HTTP 404` (Não Encontrado) em caso negativo e, em caso afirmativo, um código de sucesso (`HTTP 200`) e o arquivo solicitado na forma de um vetor de bytes como conteúdo.

3.1.4 Camada de Aplicação

Nesta camada, o elemento central, como o nome já deixa a entender, refere-se às aplicações da RSSF. Dentre as quatro camadas, esta é a que possui o elemento mais

abstrato, pois uma aplicação pode ser potencialmente implementada de qualquer maneira com as únicas restrições existentes sendo o formato dos dados de entrada e a forma de captura dos mesmos, já que estes são definidos pela camada de armazenamento.

Enquanto o papel da RSSF é o de capturar os dados de interesse nas condições desejadas, a aplicação tem o papel de atuar sobre esses dados capturados para cumprir uma determinada função. No geral, aplicações podem tipicamente ser observadas contendo dois estágios: o processamento e a atuação.

Na fase de processamento, a aplicação condiciona e/ou interpreta os dados fornecidos pela RSSF como, no caso da aplicação de monitoramento, é realizado o reconhecimento facial para identificar a existência de rostos desconhecidos.

Já na fase de atuação, a aplicação de fato executa uma determinada tarefa baseada no que foi obtido pelos dados processados, o que poderia ser, usando do mesmo exemplo, uma notificação ou SMS em um dispositivo do usuário ou mesmo o disparo de um alerta sonoro, ao identificar um rosto desconhecido em uma área monitorada.

A aplicação implementada neste trabalho consiste em um algoritmo desenvolvido em linguagem *Python* que lê imagens em formato JPEG e realiza uma operação de reconhecimento facial através da API de Geitgey (2017), buscando detectar todos os rostos presentes na imagem recebida e verificar se ao menos um deles é desconhecido, como uma forma de exemplo. O propósito desta aplicação é criar uma interface de monitoramento remoto que pode ser utilizada tanto em um contexto de vigilância quanto em um contexto de biometria, visto que a aplicação não somente reconhece rostos identificados em um banco de dados facial, mas também tem uma rotina especial executada sempre que uma imagem recebida contém rostos não identificados.

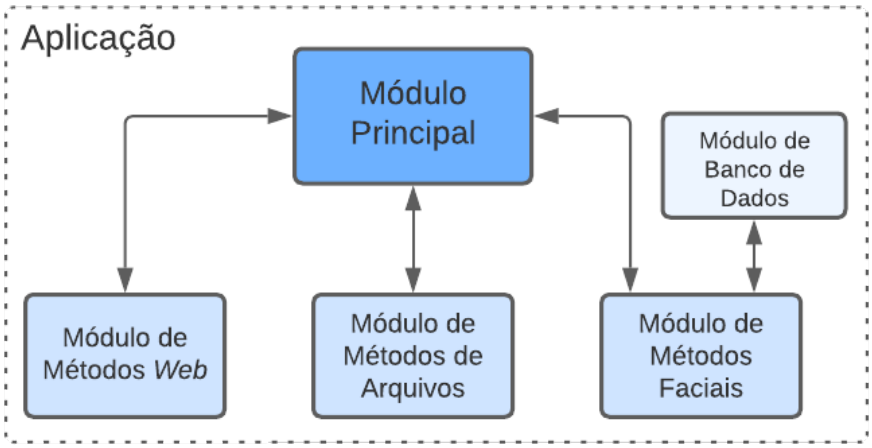
A fase de atuação em si desta aplicação é simplificada, visto que o foco desta aplicação concentrou-se na fase de processamento. Os rostos processados são reconhecidos e este reconhecimento é registrado em *log* textual, ao passo que os rostos desconhecidos não são apenas registrados em *log*, mas armazenados na forma de arquivos e catalogados seguindo o padrão estabelecido de nomenclatura dos arquivos de imagem.

A fase de processamento, por sua vez, compõe a maior parte da aplicação, indo desde o processo de checagem de *endpoint* no servidor de rede da camada de armazenamento até o processo de codificação e classificação de rostos presentes nas imagens recebidas.

A arquitetura do algoritmo desta aplicação, assim como realizado com o *gateway*, seguiu uma estrutura atômica, ou seja, sem nenhum tipo de paralelismo de processos. Esta característica de projeto foi escolhida intencionalmente como uma das maneiras de evitar conflitos com o acesso ao servidor, bem como minimizar o tempo de operação na execução da API. Isto se dá, pois, mesmo na situação de arquivos relativamente reduzidos, como é o caso de imagens JPEG, a rotina de execução da rede neural é relativamente custosa e o acúmulo de múltiplas instâncias de execução em paralelo da mesma pode facilmente ocasionar travamentos na máquina utilizada.

Mantendo a estrutura apresentada em Rodrigues (2019), esta aplicação está organizada em módulos (Figura 3.17), de maneira que o módulo principal contém a rotina de execução da aplicação, que chama os módulos secundários, responsáveis por lidar com uma seção específica da fase de processamento. São eles: o módulo de métodos *web*, o módulo de métodos de arquivos e o módulo de métodos faciais, contendo este último um submódulo para os métodos do banco de dados facial.

Figura 3.17: Diagrama da arquitetura modular da aplicação.



FONTE: O Autor (2022)

3.1.4.1 Módulo Principal

O módulo principal ou *script* principal é o núcleo do algoritmo da aplicação e de fato o que é executado quando se inicializa a aplicação. Este código pode ser descrito como uma rotina de quatro passos, repetidos sequencialmente por um tempo indefinido.

O primeiro passo é a busca, onde a aplicação varre os arquivos em seu diretório pessoal, listando cada um destes em um par de valores que corresponde à sua data de criação original (que é diferente d data em que foi salvo no diretório da aplicação) e seu nome de arquivo. Esse diretório é verificado no começo da execução da aplicação, antes mesmo desta entrar em seu laço de repetição e, caso diretório de arquivos não exista na raiz da aplicação, ele é imediatamente criado.

Com a lista de pares dos arquivos do diretório da aplicação, o módulo principal chama um dos métodos *web* para iniciar a rotina de busca do servidor de rede, obtendo como resultado, caso tudo ocorra como devido, uma segunda lista contendo a mesma estrutura de dados para o diretório remoto da aplicação localizado no servidor de rede.

Com essas duas listas, é feita uma comparação entre os arquivos presentes no diretório local e remoto, através de uma subtração de conjuntos. O resultado dessa comparação é uma lista contendo os arquivos presentes no diretório remoto que ainda não estão presentes no diretório local, ou seja, uma lista de arquivos a serem adquiridos.

O segundo passo é a aquisição, que é realizada iterativamente, de modo que cada iteração corresponde a um elemento da lista de arquivos a serem adquiridos. A cada iteração, a aplicação chama um método *web* para iniciar a rotina de aquisição do servidor de rede, utilizando como parâmetros adicionais a chave de API da aplicação e o nome do arquivo correspondente à iteração.

Sendo bem sucedida essa requisição, a aplicação obtém como retorno um vetor de bytes correspondente à imagem que foi adquirida. Essa imagem é tratada, processada e armazenada no diretório local da aplicação com métodos *web* e de arquivo.

O terceiro passo, realizado em um segundo laço iterável, é o reconhecimento facial, propriamente. Este passo utiliza a lista obtida ao final do primeiro passo e métodos faciais para realizar o reconhecimento facial nas imagens adquiridas, uma por uma, e tratar as informações obtidas desse reconhecimento. Ao final desse laço, a aplicação terá identificado todos os rostos conhecidos nas imagens e apontado todos os rostos desconhecidos.

O quarto passo é o estado ocioso. Esse passo ocorre após a aplicação ter sincronizado os arquivos no seu diretório local e aplicado reconhecimento facial em todas as imagens novas. Nesse passo, a aplicação apenas aguarda por um intervalo de tempo pré-determinado pelo usuário para então realizar uma nova busca no servidor. Caso a lista retornada no primeiro passo seja vazia ou nula, o segundo e terceiro passos são ignorados, o que indica que o diretório local já se encontra sincronizado com o diretório remoto.

3.1.4.2 Módulo de Métodos *Web*

Este módulo tem como função se comunicar com o servidor de rede através de requisições HTTP. Para isso, utiliza-se essencialmente o método GET, para realizar duas rotinas, uma de busca e uma de aquisição.

Para isso, este módulo contém a função de envio de dados, utilizada para gerar a requisição GET com um campo opcional de parâmetros, de modo que a mesma função pode ser utilizada para ambas as rotinas da aplicação. A diferença entre as rotinas está, então, no *endpoint* que é utilizado na chamada e nos parâmetros adicionais inseridos.

Além dessa função de envio, o módulo também conta com algumas funções de tratamento dos dados específicas pra cada uma das rotinas.

3.1.4.3 Módulo de Métodos de Arquivos

Este módulo é responsável por transformar os pacotes de *bytes* recebidos do servidor em arquivos com seus formatos adequados e em seus respectivos diretórios destino de interesse. Para isso, utilizam-se os recursos de *pipeline* de sistemas POSIX para acesso a arquivos, recursos estes já integrado à própria linguagem Python.

Adicionalmente, este módulo também contém outros métodos de manipulação de arquivo e/ou dados utilizados ao longo da execução do software de aplicação.

Ao se definir um nome para o arquivo, a escolha de um nome fixo faz com que o arquivo atual sempre substitua o arquivo criado anteriormente. Para evitar isso, utilizou-se mais uma vez o módulo *time*, para criar um nome de arquivo que sempre dependa do tempo de relógio naquele determinado instante, de modo a sempre criar nomes únicos.

Como as imagens transmitidas pelo sensor de imagem são todas no formato .jpg, este módulo conta com funções específicas para checagem de integridade de arquivos JPEG. No entanto, o módulo também possui funções genéricas que realizam a leitura e escrita de arquivos sem checagem, que possibilitam manipular qualquer tipo de dado em qualquer formato de arquivo.

Para a escrita dos dados adquiridos no arquivo, é necessário primeiramente chamar a função *open()* novamente, passando o parâmetro "wb+" para definir uma escrita de dados em formato binário (serial) e criar o arquivo, caso não exista. Em seguida, a função *write()* realiza a escrita dos dados no *pipeline* e a função *close()* encerra a conexão.

3.1.4.4 Módulo de Métodos Faciais

Este módulo contém as funções e métodos responsáveis por detectar, extrair e reconhecer os rostos dos indivíduos presentes na imagem recém armazenada na unidade de processamento. Ele é baseado no módulo Python *face recognition* (Geitgey, 2017) que, por sua vez, é baseado da biblioteca de detecção facial do *Dlib* (King, 2017). O algoritmo ainda roda em segundo plano os módulos *NumPy* e *PIL*, além do compilador de C++.

Ainda na etapa de importação de *scripts*, o módulo de métodos faciais chama seu submódulo, o módulo de banco de dados. Este submódulo é responsável por carregar o

banco de imagens contendo rostos conhecidos da aplicação e codificá-lo de uma forma que seja interpretável pelo *face_recognition*. Isso é feito automaticamente, já que os métodos do módulo de banco de dados são executados no próprio *script*. Este banco de dados codificado é o que a rede neural do *face_recognition* utiliza para aplicar técnicas de reconhecimento negativo nas imagens avaliadas.

Os métodos presentes neste módulo consistem nas cinco etapas realizadas pelo reconhecimento facial ao longo da aplicação. Primeiro, a imagem recebida da camada de armazenamento passa por um mapeamento de detecção facial, onde se verifica se existem rostos humanos e em quais coordenadas da imagem eles se encontram. O arquivo é carregado como uma matriz multi-dimensional e são detectados todos os rostos existentes na imagem, de modo a extrair suas coordenadas em um *array* de *arrays*.

Com a matriz da imagem e as coordenadas de todos os rostos, é possível codificar cada rosto na imagem separadamente para o padrão utilizado na rede neural. Isso é feito utilizando a Biblioteca de Imageamento do Python (PIL - *Python Imaging Library*), através da geração de sub-*arrays* a partir do *array* principal, utilizando as coordenadas previamente extraídas como limiares. Esta etapa é opcional, por não ser essencial, podendo ser omitida pelo usuário. Porém, seu propósito é permitir que o operador do sistema possa avaliar os rostos obtidos durante o processo de reconhecimento, caso seja do seu interesse.

Em seguida, já com estes rostos codificados bem definidos, um processo iterativo bidimensional é realizado para comparar cada rosto detectado na imagem, se houver, com cada rosto presente no banco de dados (esse procedimento é denominado *Face Matching*). O *Face Matching* é um processo iterativo de dois níveis que retorna, para cada rosto detectado na imagem, uma lista de valores booleanos para cada rosto no banco de dados (ou seja, retorna uma lista de listas).

Cada valor booleano representa o resultado de uma comparação entre um dos rostos conhecidos e um dos rostos detectados. A API avalia a proximidade entre os traços faciais detectados em ambos os rostos e retorna um valor entre 0 e 1, onde 0 significa rostos perfeitamente idênticos e 1 significa rostos perfeitamente distintos. A rede neural toma então a decisão se ambos os rostos são a mesma pessoa ou não, baseado na magnitude do nível de proximidade.

O limiar de tolerância para tomada de decisão é definida por padrão de software em 0,6, onde valores abaixo deste limiar definem que os rostos são da mesma pessoa e retornam valor booleano *True* e valores acima deste limiar definem que são rostos de pessoas diferentes, retornando valor booleano *False*. A documentação do algoritmo de reconhecimento facial não explicita se existe alguma razão em particular para a escolha deste valor padrão. Entretanto, o limiar de tolerância pode ser modificado para cada iteração ao se inserir o valor de tolerância como parâmetro na chamada de função. Neste projeto, o limiar de tolerância foi ajustado para 0,5.

O motivo para este ajuste, como já explorado em Rodrigues (2019), é a existência de classificações falso-positivas com resultados na faixa de 0,52 a 0,6. Duas situações de falso-positivos ocorreram com a tolerância original: um rosto não cadastrado no banco de dados podia ser falsamente reconhecido como um rosto semelhante cadastrado, gerando uma brecha para fraudes; ou um rosto cadastrado no banco de dados poderia ser reconhecido como mais de um rosto se houvessem dois rostos parecidos, produzindo um resultado inválido para um sistema biométrico classificatório. Estas situações ocorriam somente no que diz respeito a rostos muito semelhantes porém não idênticos, como é o caso de sósias e irmãos parecidos. Este ajuste não cobre, no entanto, o caso de irmãos gêmeos idênticos ou tentativas intencionais de fraude biométrica.

Com o resultado da comparação, a etapa seguinte é a classificação do rosto com base nos resultados obtidos, identificando-o com o rótulo correspondente do banco de dados, caso seja conhecido, ou como "desconhecido" caso não esteja presente no banco.

Por fim, a quinta etapa ocorre somente se foram detectados rostos na imagem e pelo menos um deles não está cadastrado no banco de dados. Esta etapa é, essencialmente, a fase de atuação da aplicação, onde a mesma realiza o registro em arquivo dos rostos desconhecidos em uma pasta separada.

3.1.4.5 Módulo do Banco de Dados

Este submódulo tem como papel pré-carregar as imagens de referência utilizadas na API de reconhecimento facial para definir os rostos conhecidos. Por isso, ele não contém métodos, mas somente uma execução sequencial, chamada durante a inicialização do módulo de métodos faciais.

O submódulo inicia sua execução varrendo o diretório de imagens de referência e incluindo os nomes de todos os arquivos de imagem encontrados em uma lista ordenada.

Esta lista é utilizada como um conjunto iterável de elementos no qual cada rosto é carregado no formato adequado para a API e depois codificado na estrutura de dados utilizada pela rede neural. Ao final de cada iteração, duas listas são incrementadas na mesma sequência, com uma contendo o rosto codificado e a outra contendo seu rótulo correspondente, gerado a partir do nome do arquivo de imagem de referência (que seguem uma convenção de nomenclatura onde recebem apenas o primeiro nome da pessoa).

Essas duas listas geradas ao final do laço de iteração são os conjuntos de dados de referência utilizados pelo módulo de métodos faciais para realizar o reconhecimento e classificação dos rostos detectados nas imagens recebidas pela aplicação. Estas listas são, respectivamente, a lista de rostos conhecidos e a lista de rótulos de rostos.

4 Resultados e Discussões

Nesta seção, são apresentados os resultados obtidos nas etapas de avaliação e validação da RSSF e aplicação propostas ao longo de seu desenvolvimento.

4.1 Avaliação do Desempenho da Comunicação LoRa

Para garantir o funcionamento adequado da RSSF, era preciso conhecer as limitações de uso do *hardware* de comunicação sem fio. Dessa forma, uma avaliação de campo foi realizada com um par de transceptores, com um atuando como receptor e o outro atuando como transmissor. Esse cenário busca identificar os fatores de interferência inerentes à comunicação e, portanto, outros fatores relativos ao fluxo de dados, como o conteúdo dos pacotes e a quantidade destes, são conhecidos e mantidos constantes.

Foram avaliados 48 pontos ao longo das redondezas do campus da UFPB (Figura 4.1), percorridos pelo transmissor enquanto o receptor permaneceu estático em um local conhecido. Esses pontos buscaram simular possíveis localizações de nós sensores em uma RSSF.

Figura 4.1: Mapa dos pontos avaliados.

FONTE: Google Maps (2021)

A relação das distâncias entre cada ponto de transmissão apresentado na Figura 4.1 pode ser observada na Tabela 4.1. Todas as transmissões foram realizadas a uma altura

constante de cerca de 1,5m do solo, utilizando antenas helicoidais de cobre a uma potência de 100 mW, na faixa de frequência de 915 MHz.

Tabela 4.1: Tabela de distâncias relativas a cada ponto utilizado na avaliação.

Ponto	Dist. (m)	Ponto	Dist. (m)	Ponto	Dist. (m)
RX	0,000	17	62,390	34	34,452
01	21,226	18	63,358	35	63,377
02	33,200	19	56,565	36	99,161
03	47,434	20	24,413	37	139,527
04	58,376	21	33,772	38	173,806
05	73,309	22	41,448	39	204,314
06	98,378	23	64,816	40	233,277
07	119,699	24	104,521	41	249,781
08	138,821	25	83,841	42	153,435
09	162,105	26	76,420	43	174,727
10	170,110	27	119,305	44	156,313
11	175,574	28	125,195	45	130,696
12	156,880	29	138,643	46	118,672
13	139,246	30	188,950	47	87,119
14	109,977	31	209,479	48	56,045
15	90,071	32	245,706		
16	76,865	33	206,567		

O procedimento de transmissão consistiu na realização de 20 transmissões espaçadas em intervalos de 2 segundos de modo que não fossem enviadas em conjunto. Após o grupo das 20 transmissões, o transmissor era deslocado para um novo ponto, a menos que dessas 20 mensagens enviadas menos da metade fosse recebida pelo receptor. Nessa situação, uma nova leva de 20 mensagens seria transmitida. Se o desempenho de recepção continuasse, mais uma terceira leva de 20 mensagens seria transmitida. Só então o transmissor seria deslocado para o ponto seguinte, independente do resultado da terceira transmissão.

A transmissão em si consistiu em uma *string* constante de texto de 255 bytes, enviada a uma frequência de 915MHz, em uma potência de 17dBm e utilizando como configurações de modulação CSS uma largura de banda BW de 125kHz e um fator de espalhamento SF de 7, resultando, de acordo com a equação (2.3) em uma taxa de símbolos R_s de aproximadamente 976,56 símbolos por segundo.

Estas configurações tinham como objetivo priorizar a taxa de transferência das mensagens e não o alcance da transmissão, o que pode explicar as ordens de grandezas de distância apresentadas na Tabela 4.1. Utilizando uma largura de banda menor e um fator de espalhamento maior, seria possível ampliar o alcance da transmissão, embora isso prejudicasse diretamente a latência da transferência de dados, de crucial importância se tratando de conjunto de dados extensos, como é o caso neste trabalho.

Esta estratégia foi adotada para garantir que se pudesse distinguir as falhas na recepção por causas temporárias das falhas por conta da distância de transmissão. Assim, os resultados de um ponto que não apresentou problemas são a média de 20 transmissões, para um ponto que apresentou problemas raros é a média de até 40 transmissões e para um ponto que apresentou problemas graves de transmissão a média é de até 60 transmissões.

Foram avaliados alguns parâmetros de transmissão para cada ponto, que podem ser

observados nos gráficos a seguir. A taxa de erro de pacote (Figura 4.2) indica a razão entre pacotes não recebidos e pacotes enviados, indicando a porcentagem de falha no recebimento naquele determinado ponto.

Nos pontos 43 e 44, indicados em vermelho no mapa da Figura 4.1, o receptor foi considerado fora da região de cobertura do sinal transmitido, pois não apresentaram sucesso na transmissão de nenhum pacote de dados, mesmo ao longo de três iterações. É importante salientar que isso não pode ser explicado unicamente levando-se em conta o alcance entre transmissor e receptor, visto que pontos como o 40 e 41 apresentam distâncias maiores que estes, como visto na Tabela 4.1. Esta ocorrência também tem influência da qualidade do percurso em termos de obstruções, visto que a região dos pontos 43 e 44 é a que apresenta maior densidade de prédios no caminho para o receptor dentre os pontos avaliados no teste.

Deste modo, a Figura 4.2 reflete o comportamento dos pontos 43 e 44 com uma taxa de erro de 100%, visto que nenhum dos pacotes enviados por estes pontos foram recebidos pelo receptor.

Do ponto de vista de transmissão, a relação sinal-ruído (SNR) média de cada ponto pode ser observada na Figura 4.3.

Também observa-se a ocorrência das falhas nos pontos 43 e 44 expressas pelo valor nulo de SNR média, visto que a razão da SNR calculada é diretamente proporcional à potência do sinal. Como não existe nenhum dado recebido, para o receptor a potência média do sinal destes pontos é nula e, portanto, também é sua SNR média.

Por sua vez, o RSSI médio de cada ponto se encontra na Figura 4.4.

Dentre os gráficos apresentados até então, é na Figura 4.4 que o comportamento dos pontos 43 e 44 se mostra mais destoante. Assim como no caso da SNR média, a RSSI média depende da potência do sinal recebido e, como não houve efetiva recepção destes pontos, a RSSI é considerada nula.

Por fim, a taxa de erro de bit (Figura 4.5) busca avaliar a qualidade da mensagem recebida, contabilizando a quantidade de bits diferentes entre as mensagens que foram transmitidas e as que foram recebidas.

Aqui, diferentemente dos gráficos apresentados até então, não são os pontos 43 e 44 que particularmente chamam atenção. Os valores não nulos apresentados neste gráfico indicam que os pontos em questão apresentaram interferências no conteúdo recebido em relação ao conteúdo transmitido. Dessa forma, a BER dos pontos 43 e 44 foi intencionalmente desconsiderada por estar significativamente fora de escala em relação aos demais valores não nulos e não trazer nenhuma informação numericamente relevante.

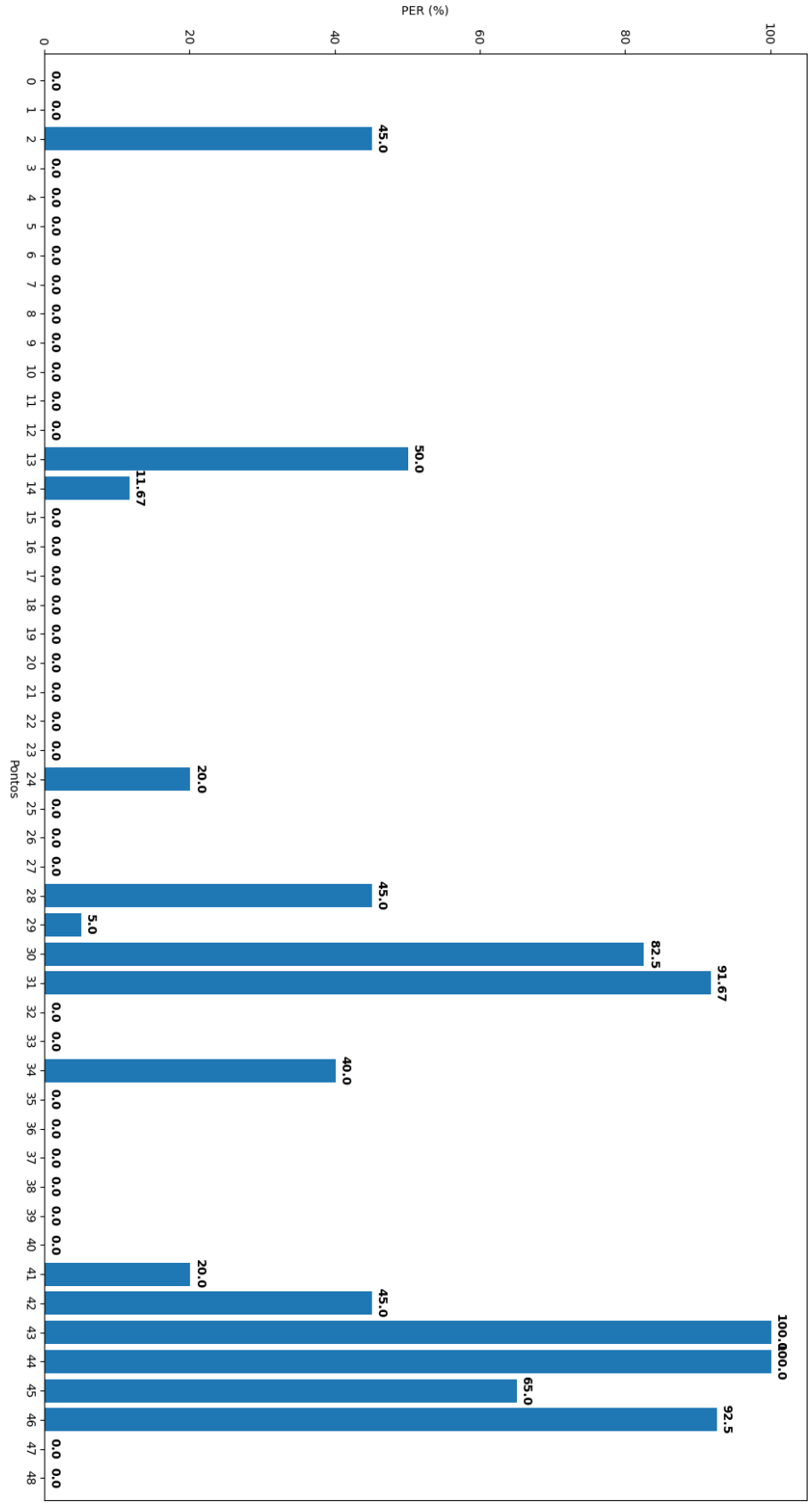
A partir da análise dos demais valores não nulos, em conjunto com o mapa da Figura 4.1 e a Tabela 4.1, observa-se que a taxa de erro de bit ocorre com maior frequência particularmente em caminhos com alguma obstrução, seja ela por vegetação (pontos 13 e 27) ou por construções (pontos 30, 31, 41 e 42), porém não tão elevada como no caso dos pontos 43 e 44.

Desta forma, deduz-se que a presença de erros de bit na transmissão é um tipo de comportamento limítrofe entre a completa falha na transmissão como ocorre em 43 e 44 e a transmissão bem sucedida dos demais pontos.

4.2 Validação do Gerenciamento de Dispositivos

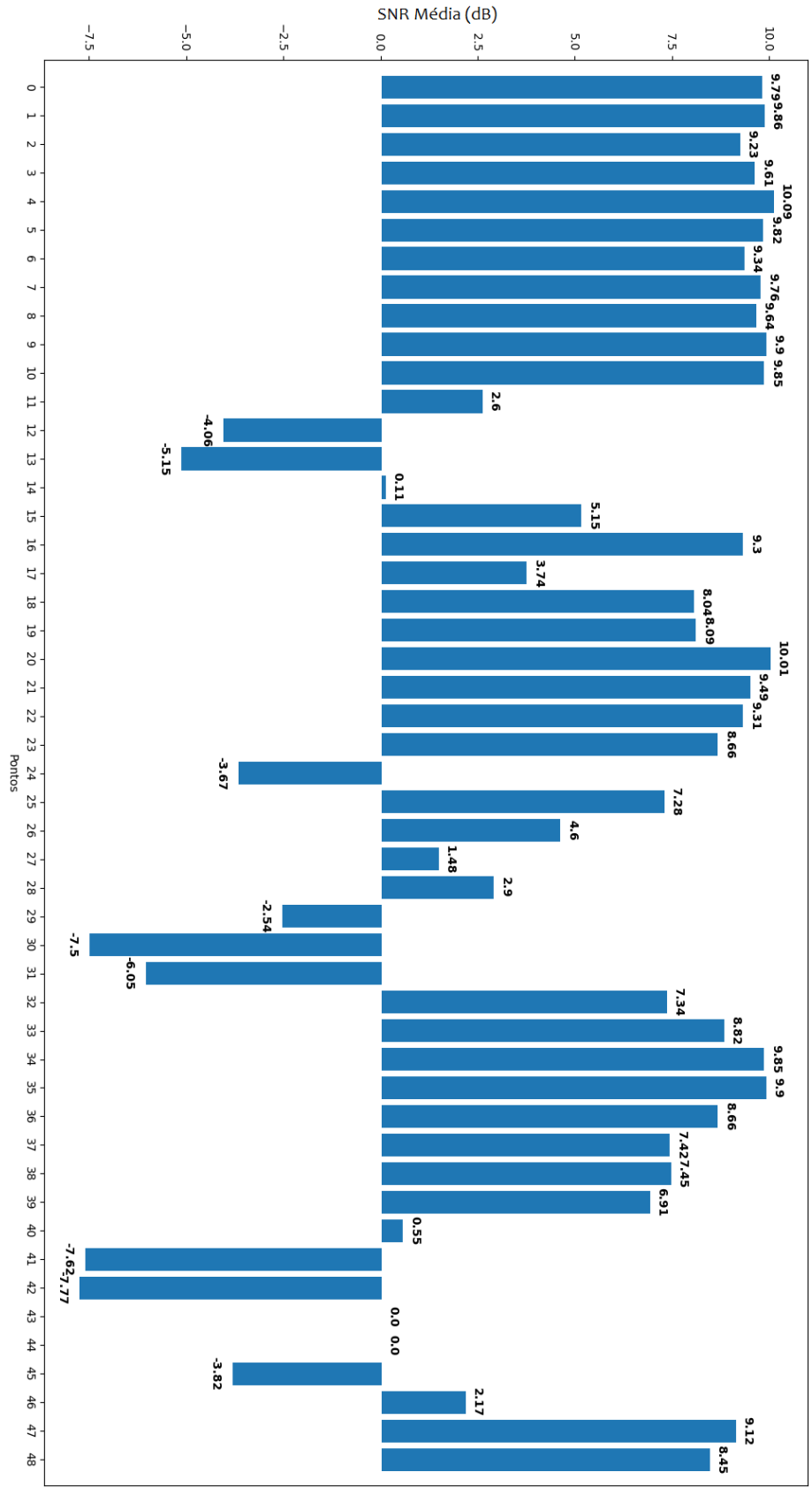
Pelo fato da rede desenvolvida não utilizar nenhum tipo de protocolo MAC, foi necessário se estabelecer um método de identificação e gerenciamento de dispositivos, utili-

Figura 4.2: Taxa de erro de pacote (PER) em cada ponto.



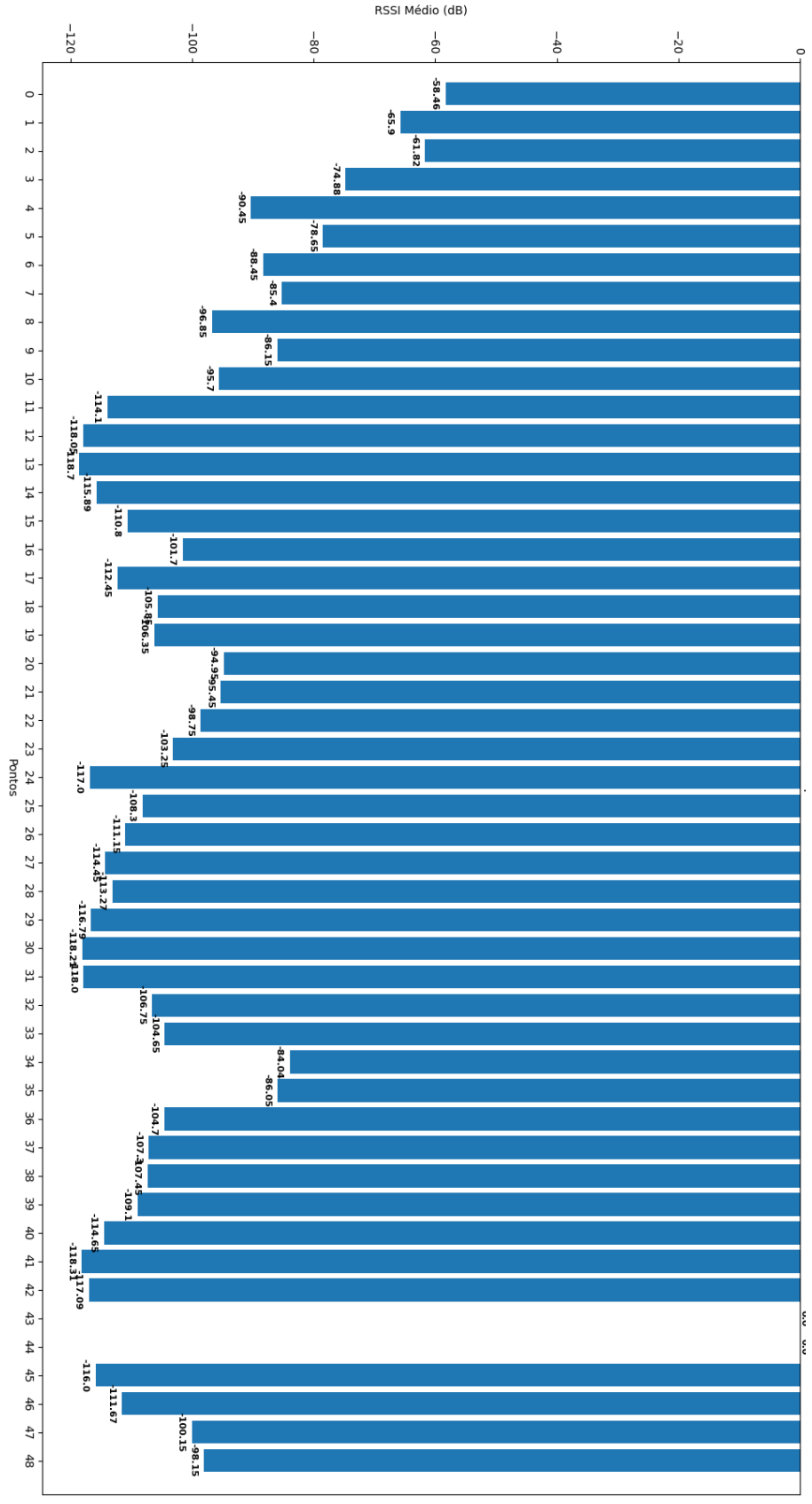
FONTE: O Autor (2021)

Figura 4.3: SNR média em cada ponto.



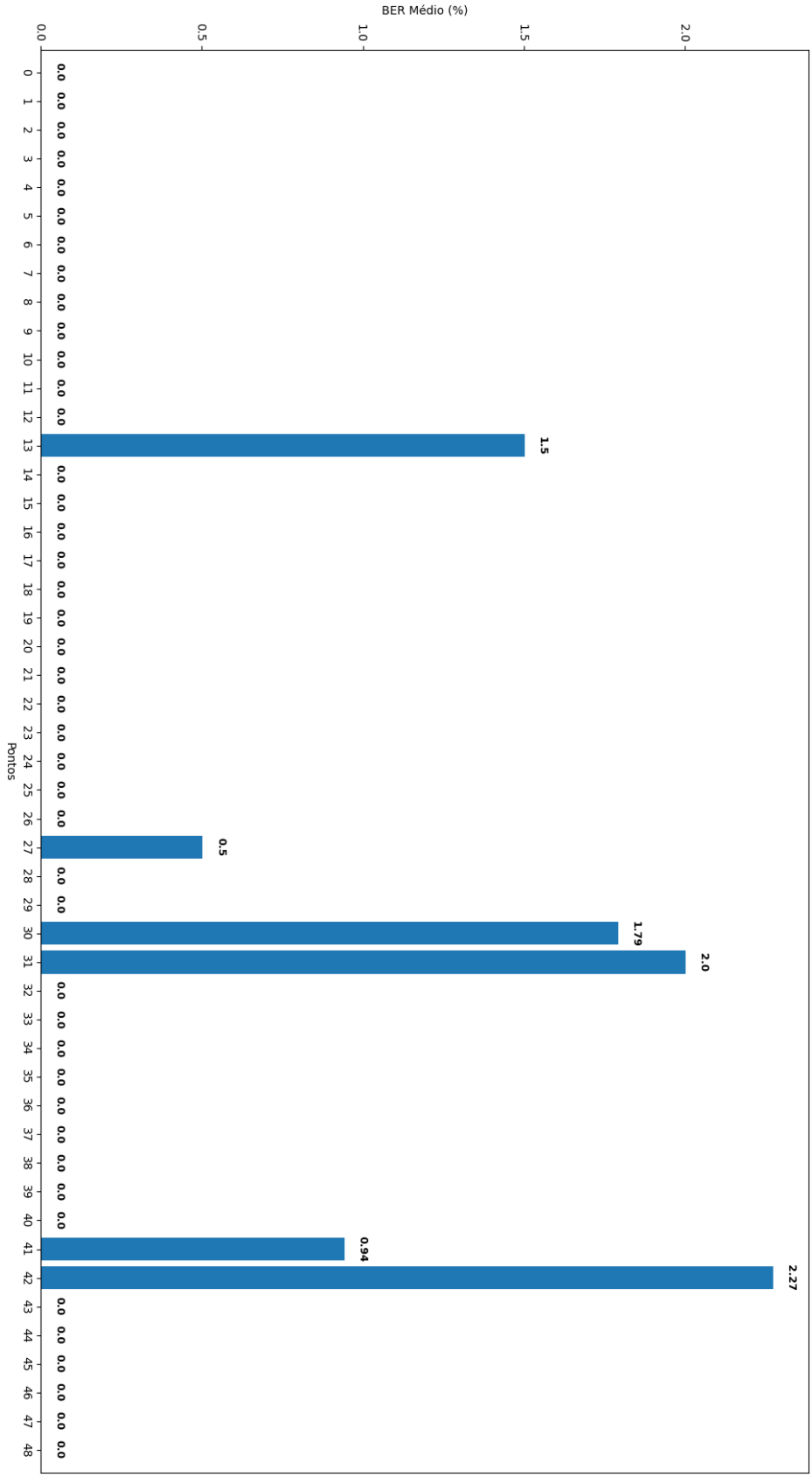
FONTE: O Autor (2021)

Figura 4.4: RSSI médio em cada ponto.



FONTE: O Autor (2021)

Figura 4.5: Taxa de erro de bit (BER) média em cada ponto.



FONTE: O Autor (2021)

zando endereços, de modo a evitar transmissões simultâneas de nós sensores a um mesmo *gateway*.

Para validar este recurso do sistema, uma rotina de transmissões foi realizada de modo que dois nós sensores distintos executassem transmissões em situações distintas de maneira controlada, sem que houvesse conflito entre eles. Nessa rotina, o gerenciador desenvolvido deve garantir a manutenção de três regras básicas que o definem:

- Um transmissor não captura uma nova imagem até que tenha enviado a última imagem capturada;
- O receptor, enquanto estiver lendo os dados de um transmissor, deverá ignorar quaisquer solicitações ou tentativas de transmissão de outros transmissores;
- Um transmissor permanece solicitando permissão para transmitir até que a última imagem capturada seja devidamente enviada.

Para garantir que essas regras foram devidamente cumpridas, uma sequência de transmissões com condições específicas foi realizada, com o objetivo de verificar cada possibilidade de interação entre dois nós sensores independentes, denominados de Nó A e Nó B para referência. As condições verificadas foram:

1. Ou o Nó A ou o Nó B detectam movimento enquanto o receptor entrou no modo ocioso pela primeira vez;
2. Ou o Nó A ou o Nó B detectam movimento enquanto ainda estão transmitindo ao receptor;
3. Ou o Nó A ou o Nó B detectam movimento imediatamente após terem transmitido;
4. O Nó A detecta movimento após o Nó B ter transmitido;
5. O Nó B detecta movimento após o Nó A ter transmitido;
6. Nó A detecta movimento enquanto o Nó B transmite;
7. Nó B detecta movimento enquanto o Nó A transmite.

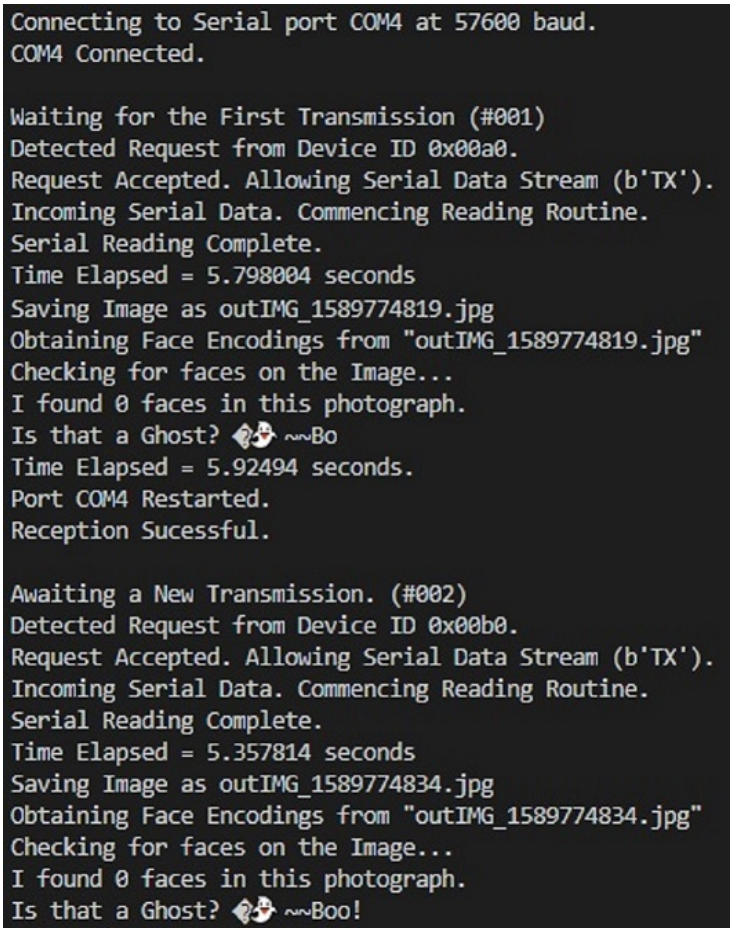
Cada situação foi observada 10 vezes para garantir menor variabilidade, mas todas as situações foram totalmente consistentes em seus resultados. O trecho de uma das sequências realizadas pode ser observado na Figura 4.6, em que uma transmissão do nó sensor B ocorre imediatamente após a transmissão do nó sensor A.

É possível observar, no trecho de texto, cada passo sequencial do algoritmo sendo executado. A sequência de transmissões descrita acima foi realizada repetidamente, variando a ordem na qual os casos enumerados foram reproduzidos. Os resultados demonstraram que não somente o sistema sempre obedece suas restrições, mas também cada transmissão aparenta ser completamente indiferente às iterações de transmissões anteriores.

Isso significa que, para uma rede com dois nós sensores (A e B), cada possível sequência combinacional de n transmissões entre elas já é suportada pelo sistema de gerenciamento. Esta conclusão pode, razoavelmente, ser extrapolada para uma rede com m nós sensores.

O valor máximo para m ou, em outras palavras, o número máximo de nós que podem ser conectados na rede é limitado por software para 255 dispositivos, compondo os

Figura 4.6: Resultado de uma das rotinas da validação do gerenciador.



FONTE: O Autor (2020)

endereços de valores `0x0001` a `0x00FF`, sendo `0x0000` reservado para o modo *broadcast* utilizado pelo *gateway*.

Caso seja necessário, a limitação de endereçamento de nós sensores atribuídos a um *gateway* pode ser ampliada indefinidamente ao se aumentar a quantidade de bytes necessária para representar cada endereço, utilizando a relação:

$$m_{max} = 2^{8 \cdot B} - 1 \tag{4.1}$$

onde m_{max} é o limite máximo de m definido por software e B é o número de bytes utilizado para representar endereços dos nós. Apesar dessa adição de novos bytes afetar diretamente o tamanho de cada pacote transmitido entre dispositivos, por conta da relação entre os dois fatores ser exponencial, uma aplicação real dificilmente necessitará aumentar em muito o tamanho do endereçamento visto que um endereçamento de 5 bytes, por exemplo, já permite pouco menos do que 1,1 trilhões de endereços únicos.

Todavia, apesar da limitação de *software* ser intencional, deve-se considerar as diversas limitações físicas que, praticamente todas, não o são. O limite definido de 255 nós conectados por *gateway* ou ainda a relação exponencial de expansão desse limite (m_{max}) não definem a capacidade física de simultaneidade de cada dispositivo.

Limitações de *hardware*, ou seja, a capacidade real dos dispositivos utilizados ainda precisam ser mensuradas para determinar um limite objetivo para a RSSF, lidando com

fatores como latência entre transmissões, cruzamentos de sinais e interferências entre transmissões paralelas e geradas por ruídos do meio. Outras limitações relativas ao meio e à implementação da rede também podem ser investigadas, como a dispersão dos dispositivos e a interferência causada por fatores ambientais. Diferente das limitações de *hardware* que são relativas à construção dos dispositivos utilizados, algumas das limitações do meio podem ser contornadas com um reprojeto da rede, como a redistribuição de nós sensores, inclusão de outros *gateways* e no uso de medidas para atenuar as perdas por fatores ambientais.

4.3 Validação de Desempenho da Aplicação

Um dos primeiros testes realizados na RSSF foi verificar o desempenho da aplicação em reconhecer rostos desconhecidos em uma imagem e executar a rotina correspondente a esta situação. Para isso, utilizou-se como imagem de teste uma fotografia de um evento técnico realizado na Universidade Federal da Paraíba (Figura 4.7). Nenhum dos indivíduos presentes na foto teve seu rosto incluído no banco de dados, de modo que, para a Aplicação, todos eram rostos desconhecidos.

Figura 4.7: Imagem de teste utilizada na validação.

FONTE: O Autor (2020)

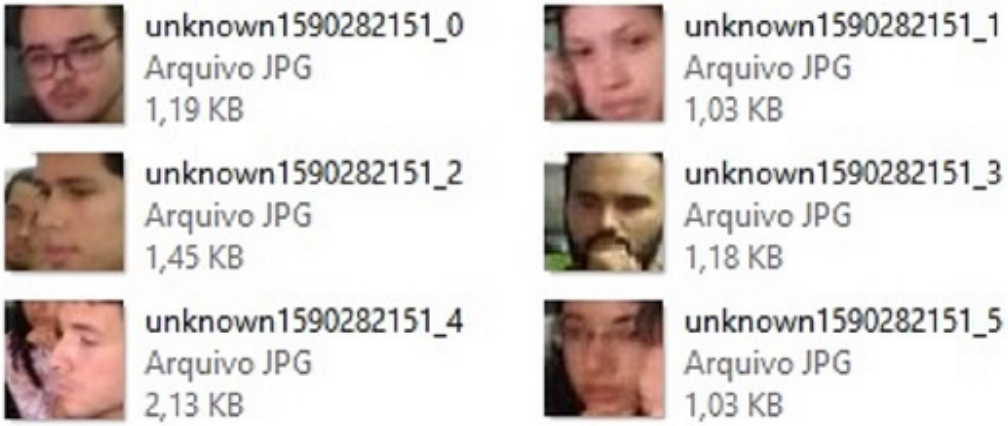
A rotina esperada da aplicação é que, ao detectar pelo menos um rosto desconhecido na imagem, a aplicação salve em um diretório separado todos os rostos desconhecidos encontrados naquela imagem, seguindo uma convenção de nomenclatura de arquivo onde o nome dos arquivos de rostos desconhecidos é o mesmo nome da imagem original acrescentado de um índice (que corresponde à posição do rosto na lista de rostos detectados durante o reconhecimento facial), separados por um caracter *underscore*.

Na imagem apresentada, nove rostos são observáveis em posições e ângulos distintos. Ao executar o algoritmo, a aplicação detectou seis dos nove rostos observáveis na imagem. No que diz respeito aos três rostos que não foram detectados, dois deles apresentam um alto índice de obstrução (aparecendo apenas parcialmente) e um estava posicionado em

um ângulo acentuado demais para poder ser detectado pela RNA (estando praticamente perpendicular ao plano da imagem). Ambas as situações são descritas na documentação da API de reconhecimento facial como limitações conhecidas.

Durante o processo de reconhecimento dos seis rostos que foram detectados, a aplicação categorizou todos como "desconhecido" e ao final do rotulamento armazenou todos os seis rostos em arquivos de imagem próprios, como observado na Figura 4.8, que demonstra quais rostos foram de fato detectados e também demonstra de forma prática a convenção de nomenclatura utilizada na criação desses arquivos de imagem.

Figura 4.8: Rostos desconhecidos extraídos da imagem na Figura 4.7.



FONTE: O Autor (2021)

4.4 Avaliação de Desempenho do Ciclo de Transmissão

Por se tratar de uma RSSF voltada para processamento de imagens, é relevante mensurar para esta rede seu desempenho quanto à resolução da imagem capturada pelo sensor de imagem.

Por se tratar de um conjunto de dados bidimensional, a relação de variação de grandezas entre duas imagens de tamanhos diferentes é quadrática. Ou seja, considerando duas imagens sem compressão, uma com o dobro da altura e comprimento da outra, haverá uma relação de tamanho de quatro vezes e, por consequência, um conjunto de dados será quatro vezes maior que o outro.

Considerando o uso das técnicas de compressão, a diferença no tamanho do conjunto de dados causada pelo diferença de tamanho da imagem é drasticamente reduzida, mas isso não significa que ela deixe de ser relevante.

Do ponto de vista de otimizar a taxa de transmissão, o cenário ideal é utilizar a menor resolução possível. Já do ponto de vista de otimizar a aplicação, o cenário ideal é utilizar a maior resolução possível. Esse impasse gera a necessidade de encontrar um ponto ótimo de custo-benefício que não é trivial e precisa ser testado.

Levando isso em consideração, foi levantado o desempenho da RSSF quanto ao tempo de publicação de uma imagem capturada no servidor, variando a resolução de captura do sensor de imagem. A Tabela 4.2 apresenta os tempo de transmissão entre nó sensor e *gateway* e entre nó sensor e servidor de rede, estimados pela média de 20 amostras cada.

A distância de reconhecimento ideal depende da distância do rosto em relação à lente da câmera, o que impede a obtenção de um resultado absoluto. Entretanto, durante os tes-

Tabela 4.2: Tabela de tempos de transmissão para cada resolução do sensor de imagem.

Resolução (px.)	Tempo Nó × *Gateway* (ms)	Tempo Nó × Servidor (ms)
160 x 120	6690,82	6869,68
320 x 240	9297,82	9381,83
640 x 480	33384,65	33444,89
800 x 600	41826,37	41916,82
1024 x 768	69426,99	69495,47
1280 x 1024	100742,63	100810,17

tes realizados, foi considerado o intervalo de alcance do sensor de movimento, com valores entre 0,2 e 3,0 m de distância para o rosto em relação à lente. Nesse intervalo, as resoluções de 320 × 240 e 640 × 480 apresentaram os melhores resultados de custo-benefício no que compete à razão taxa de transmissão por qualidade da imagem transmitida.

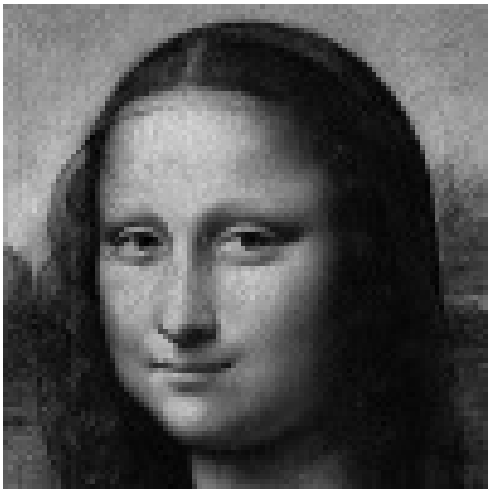
Outra possível abordagem para reduzir a duração do ciclo de transmissão foi a de pré-processamento. Integrando a capacidade de processamento de imagem no próprio nó sensor, seria possível reduzir o tamanho dos conjuntos de dados ao se enviar somente os rostos detectados na imagem. Essa abordagem, no entanto, não foi considerada por aumentar em muito o custo computacional do nó sensor, o que por sua vez aumentaria diretamente tanto seu consumo de energia quanto o próprio custo com dispositivos, já que uma MCU típica não seria capaz de realizar esse tipo de processamento.

4.5 Avaliação de Desempenho Rede x Aplicação

Um último teste de campo buscou avaliar a qualidade da imagem recebida pela aplicação em função da distância e localização de um nó sensor em relação ao *gateway*.

Para isso, manteve-se um nó sensor estático em uma localidade, transmitindo uma imagem de teste sempre que seu sensor piroelétrico fosse ativado e deslocou-se o *gateway* através de 8 localidades na Universidade Federal da Paraíba. A imagem de teste utilizada pode ser observada na Figura 4.9

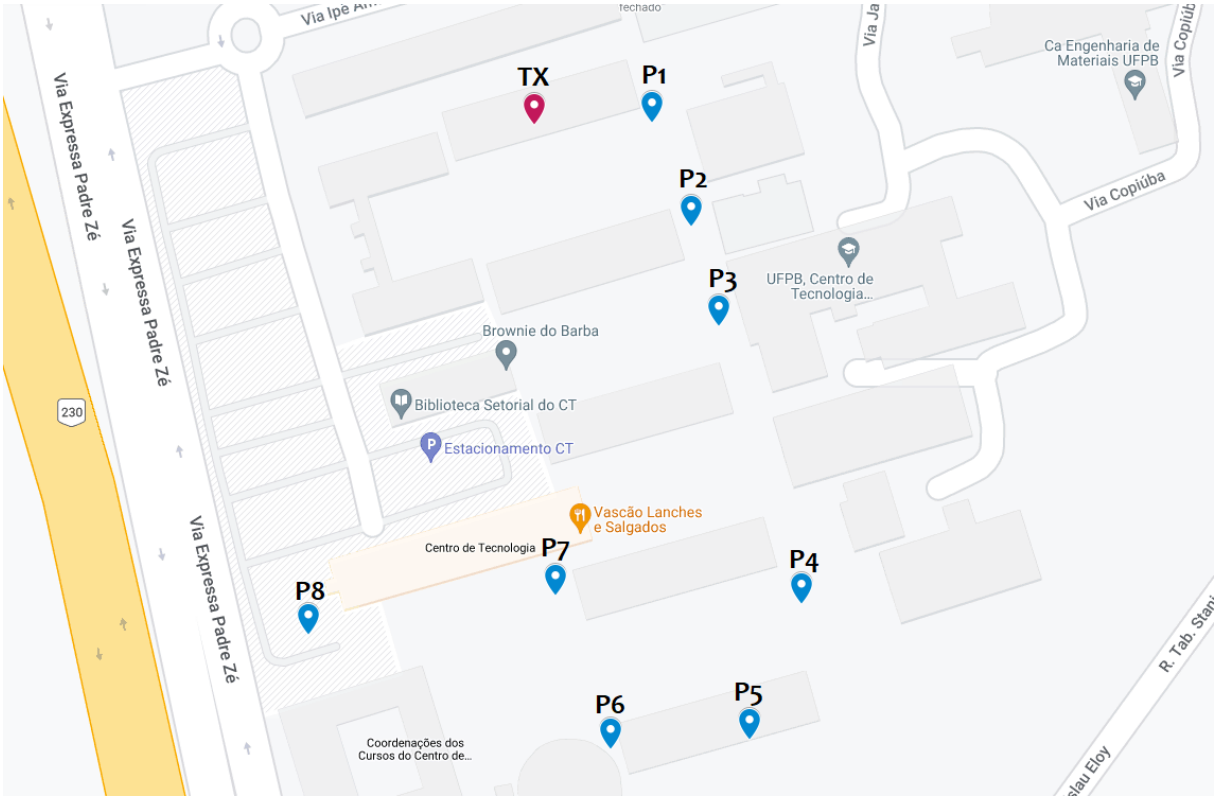
Figura 4.9: Imagem utilizada no teste.



FONTE: Adaptado de da Vinci, 1506

Na Figura 4.10, são apresentados, em um mapa, os pontos nos quais foram realizadas as recepções. Para cada ponto, três recepções foram realizadas.

Figura 4.10: Mapa dos pontos avaliados.



FONTE: Google My Maps (2022)

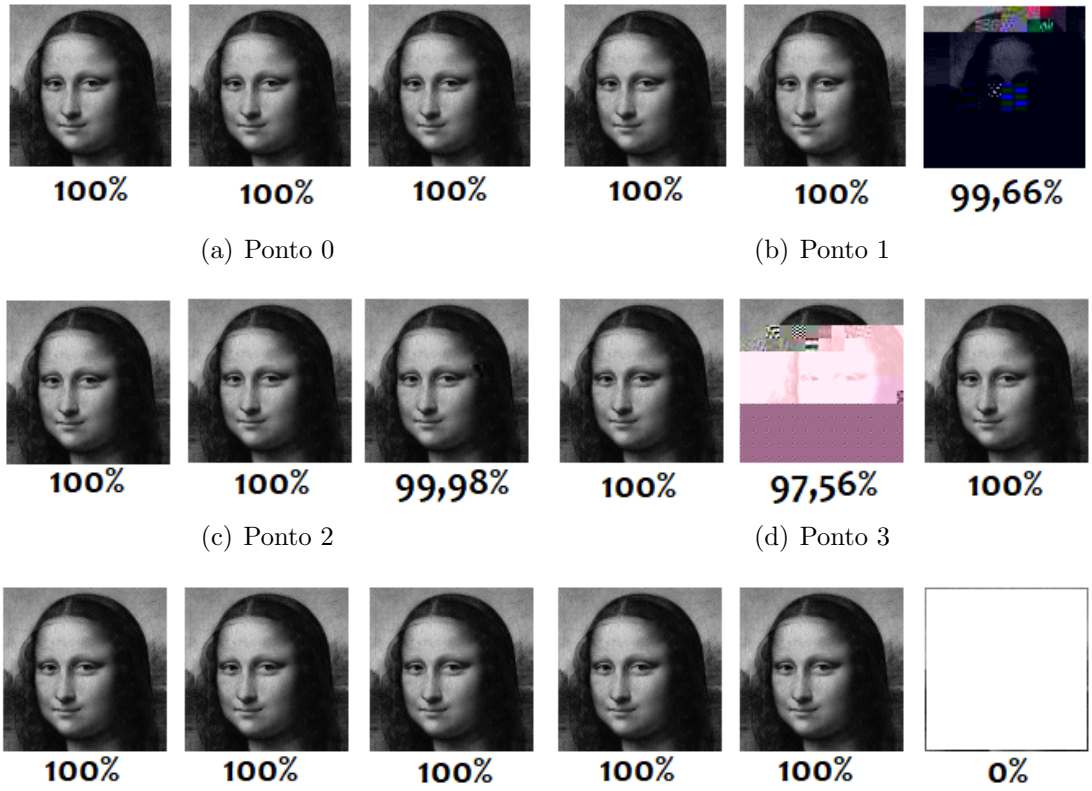
As distâncias dos pontos 1 a 8 em relação ao transmissor (ponto 0) podem ser observadas na Tabela 4.3. Os pontos onde não houve nenhuma recepção durante o teste foram grifados em vermelho.

Tabela 4.3: Tabela de distâncias de cada ponto de recepção.

Ponto	Distância (m)	Recepções
P0	0	3/3
P1	33	3/3
P2	52	3/3
P3	76	3/3
P4	153	0/3
P5	182	0/3
P6	176	3/3
P7	132	2/3
P8	156	0/3

Na Figura 4.11, vemos os resultados obtidos nas recepções de cada ponto. Os pontos onde não foram obtidas recepções foram omitidos. O tempo médio de recepção entre as imagens recebidas foi de aproximadamente 16499 ms.

Figura 4.11: Imagens recebidas pela aplicação nos pontos de recepção.



(a) Ponto 0 (b) Ponto 1

(c) Ponto 2 (d) Ponto 3

(e) Ponto 6 (f) Ponto 7

FONTE: Adaptado de da Vinci, 1506

É possível observar um comportamento já notado em testes anteriores com transmissões LoRa, onde somente a variação de distância, em intervalos razoáveis, não apresenta uma influência significativa na qualidade dos dados recebidos ao longo de toda sua área de alcance. Por outro lado, observa-se uma relação direta entre a incidência de erros presentes nas imagens recebidas pela aplicação e o nível de obstrução no caminho percorrido pelo ar da transmissão.

Uma possível solução para esse tipo de comportamento é o posicionamento dos *gateways* da RSSF em locais de elevação.

5 Considerações Finais

Este trabalho buscou expandir, baseado no cenário já estabelecido previamente, o escopo de atuação do sistema de um sistema ponto-a-ponto para uma RSSF propriamente dita.

Alguns obstáculos foram enfrentados no processo de desenvolvimento e implementação da RSSF proposta. Um desses obstáculos se deu no que diz respeito ao uso de servidores em nuvem abertos como forma de camada de armazenamento do sistema, o que se provou bastante difícil, visto que praticamente todos os servidores LoRaWAN disponíveis possuíam restrição de acesso a *hardwares* de determinados fabricantes, por mais que todos estes sejam baseados nos chips SX1276 e SX1278. Dessa forma, foi necessário criar uma abordagem própria como solução para esta camada que não envolvesse o uso de LoRaWAN mas ainda se utilizasse da arquitetura clássica de RSSFs com as aplicações separadas da rede. Para tal, a solução escolhida foram os servidores *web*, baseados em http e linguagens de alto nível.

Essa solução alternativa acabou também trazendo uma maior liberdade para o controle de dados tanto na entrada quanto na saída do servidor, o que permite garantir que o protocolo de comunicação padronizado sirva como uma espécie de canal direto entre o nó sensor e a aplicação, já que o próprio servidor pode ser projetado levando em conta as abstrações nesse tipo de protocolo de comunicação.

Com a mudança do *hardware* dos nós sensores da rede, em relação ao que foi apresentado na versão anterior, um processo de reestruturação de todo o *software* presente na RSSF foi necessário. Isso, no entanto, acabou trazendo benefícios, pois boa parte dos trechos de *software* que sofreram atualizações o fizeram não somente no aspecto de suportar os novos dispositivos mas também no aspecto de torná-los mais legíveis e otimizar seu funcionamento no que tange o custo computacional. Isso se justifica pelo fato de que, em muitos casos, os conhecimentos de programação adquiridos ao longo dos quase dois anos e meio entre Rodrigues (2019) e o trabalho atual foram um diferencial para encontrar melhores implementações do ponto de vista de gastos computacionais, de consumo de memória ou até mesmo do ponto de vista organizacional para operações já funcionais. No geral, foi possível observar uma redução entre 20 e 25% no tempo de processamento dos algoritmos embarcados (Nó sensor e *gateway*) e de cerca de 80% no tempo de processamento do algoritmo da aplicação.

O desempenho obtido com a RSSF foi significativamente melhor do que o que foi originalmente obtido em Rodrigues (2019), o que pode se explicar pelas diversas melhorias e otimizações implementadas, descritas neste trabalho.

Considerando que ainda há espaço para melhorias mais refinadas, é razoável concluir que o tipo de aplicação proposta, ou seja, um sistema de monitoramento remoto baseado em LPWAN, não somente é viável como pode se mostrar vantajoso em determinados contextos, com a maioria das limitações observadas podendo ser contornadas com uma maior disponibilidade de dispositivos compondo a rede e/ou um estudo aprofundado do ambiente para projetar adequadamente onde cada dispositivo deve estar localizado. No que compete ao alcance da transmissão LoRa entre nós sensores e *gateways*, outra possível observação é realizar adequadamente o condicionamento de antenas, visto que a capacidade de potência delas é um fator não trivial que pode interferir diretamente no alcance de sinal em uma implementação da RSSF.

5.1 Trabalhos Futuros

Apesar do conceito da RSSF e da aplicação em si já terem sido desenvolvidos, esta pesquisa ainda apresenta o potencial para novos resultados serem obtidos no futuro. A seguir, encontra-se uma lista de possíveis tópicos de interesse considerados relevantes a partir do que já foi desenvolvido e que ainda não foram abordados:

- Avaliar o consumo de bateria do nó sensor (Nikolić et al., 2017);
- Avaliar a influência de fatores ambientais (luminosidade, umidade, temperatura, etc.) na sensibilidade do nó sensor e na qualidade dos dados (Sahmarani, Simeu-Abazi e Ladret, 2006);
- Avaliar o risco de fraude na RSSF associado a *spoofing* e possíveis soluções (Siddiqui e Park, 2019);
- Simulação de rede com alta densidade de nós para avaliar desempenho (Martinez-Sala et al., 2005);
- Comparativo com redes de monitoramento baseadas em outras tecnologias.

5.2 Publicações Acadêmicas

Nesta seção estão listadas, em ordem cronológica, as publicações realizadas ao longo do desenvolvimento desta pesquisa, na forma de artigos em periódicos, patente e registro de *software*.

- RODRIGUES, V., MEDEIROS, D., CARVALHO, F., & LOPES, W. (2021). LoRa System for Monitoring and Facial Recognition. Journal of Communication and Information Systems, 36(1), 1-16. DOI: https://doi.org/10.14209/jcis.2021.1
- RODRIGUES, V. J. C.; CARVALHO, F. B. S. "Software para Monitoramento baseado em Reconhecimento Facial e Transmissão a Longas Distâncias". Registro de Software - Programa de Computador. Número do Registro: BR512021000893-6. Instituição de Registro: INPI - Instituto Nacional da Propriedade Industrial. Data do Registro: 04/05/2021.
- RODRIGUES, Vítor José Costa; DE CARVALHO, Fabrício Braga Soares. Cenário Atual, Perspectivas e Aplicações de IoT. Revista de Tecnologia da Informação e Comunicação, [S.l.], v. 10, n. 1, p. 50-56, dez. 2021. ISSN 2237-5104. Disponível em: <https://rtic.com.br/index.php/rtic/article/view/120>.
- RODRIGUES, V. J. C.; CARVALHO, F. B. S. "Sistema de Monitoramento e Reconhecimento Facial para Transmissão a Longas Distâncias". Patente: Privilégio de Inovação. Número de Registro: BR1020220020086. Instituição de Registro: INPI - Instituto Nacional da Propriedade Industrial. Depósito em: 02/02/2022.

Referências

[1] A. Ross e S. Prabhakar. A. K. Jain. "An Introduction to Biometric Recognition". Em: *IEEE Transactions on Circuits and Systems for Video Technology* 14 (jan. de 2004), pp. 4–20. DOI: 10.1109/TCSVT.2003.818349.

[2] W. A. Adcock. *Electronic Photography System. United States Patent*. Texas Instruments, Inc. Dallas, TX, nov. de 1977. URL: https://patentimages.storage.googleapis.com/42/c5/65/f8f4d850406b62/US4057830.pdf.

[3] L. Souza et al. "How far did we get in face spoofing detection?" Em: *Engineering Applications of Artificial Intelligence* 72 (out. de 2017), pp. 368–381. ISSN: 0952-1976. DOI: 10.1016/j.engappai.2018.04.013.

[4] Y. Kortli et al. "Face Recognition Systems: A Survey". Em: *Sensors* 20 (jan. de 2020). DOI: 10.3390/s20020342.

[5] Z. Zhao et al. "Object Detection with Deep Learning: A Review". Em: *IEEE Transactions on Neural Networks and Learning Systems* 1 (jan. de 2019), pp. 1–21. DOI: 10.1109/TNNLS.2018.2876865.

[6] G. Kim et al. "Face liveness detection based on texture and frequency analyses". Em: *IAPR International Conference on Biometrics* 5 (ago. de 2012). DOI: 10.1109/ICB.2012.6199760.

[7] J. Li et al. "Live Face Detection Based on the Analysis of Fourier Spectra". Em: *Biometric Technology for Human Identification* 5404 (ago. de 2004), pp. 296–303. DOI: 10.1117/12.541955.

[8] Bay Alarm. *How do motion sensors work: A guide*. Dez. de 2020. URL: https://www.bayalarm.com/commercial/burglar-alarm-systems-commercial/how-do-motion-sensors-work-a-guide/ (acesso em 30/06/2022).

[9] LoRa Alliance. *About LoRa Alliance*. URL: https://lora-alliance.org/about-lora-alliance (acesso em 04/09/2019).

[10] LoRa Alliance. *White Paper. LoRaWAN^{TM} 1.1 Specification*. 2017.

[11] Anatel. *Atribuição de Faixas de Frequência no Brasil (2014)*. URL: https://www.anatel.gov.br/Portal/verificaDocumentos/documento.asp?numeroPublicacao=314474&assuntoPublicacao=null&caminhoRel=null&filtro=1&documentoPath=314474.pdf (acesso em 30/08/2019).

[12] ArduCAM. *Arducam Mini 2MP Plus – OV2640 SPI Camera Module for Arduino UNO Mega2560 Board & Raspberry Pi Pico*. 2019. URL: https://www.arducam.com/product/arducam-2mp-spi-camera-b0067-arduino/ (acesso em 16/01/2022).

[13] K. Ashton. *That 'Internet of Things' Thing*. 2009. URL: https://www.rfidjournal.com/articles/pdf?4986 (acesso em 11/08/2019).

[14] L. Atzori, A. Iera e G. Morabito. "The internet of things: a survey". Em: *Computer Networks, Elsevier.* (2010).

[15] A. Augustin et al. "A Study of LoRa: Long Range & Low Power Networks for the Internet of Things". Em: *Sensors* (2016).

[16] K. E. Berry. *Passive and Active Sensors: Demystifying Satellite-Based Leak Detection*. Jul. de 2020. URL: https://asterra.io/resources/passive-and-active-sensors-demystifying-satellite-based-leak-detection/ (acesso em 30/06/2022).

[17] L. E. Borges. *Python para Desenvolvedores*. 2ª ed. Rio de Janeiro: Edição do Autor, 2010.

[18] W. S. Boyle e G. E. Smith. *Buried Channel Charge Coupled Devices. United States Patent*. Bell Labs. New Jersey, abr. de 1973. URL: https://patentimages.storage.googleapis.com/90/6e/28/e1f0a9b89d5110/US3792322.pdf.

[19] C. Burati et al. "An Overview on Wireless Sensor Networks Technology and Evolution". Em: *Sensors* 9 (ago. de 2009), pp. 6869–6896. ISSN: 1424-8220. DOI: 10.3390/s90906869.

[20] R. Buyya e A. V.. Dastjerdi. *Internet of Things. Principles and Paradigms*. 1ª ed. An optional note. Elsevier Inc., jul. de 2016.

[21] V. J. C. Rodrigues. *Sistema Embarcado de Monitoramento e Identificação Pessoal Baseado na Tecnologia LoRa. Monografia (Graduação)*. Publicação Eletrônica. João Pessoa, 2019.

[22] C2RMF e L. S. P. da Vinci. *IMAGEM: Mona Lisa*. 1506. URL: https://en.wikipedia.org/wiki/File:Mona_Lisa,_by_Leonardo_da_Vinci,_from_C2RMF_retouched.jpg (acesso em 20/06/2022).

[23] *CMOS VGA (640x480) CameraChip With OmniPixel Technology. Datasheet: OV7670 / OV7171*. 1ª ed. Publicação Eletrônica. OmniVision. Jul. de 2005.

[24] Arduino Company. *Arduino Mega 2560 Rev3*. URL: https://store.arduino.cc/usa/mega-2560-r3 (acesso em 30/08/2019).

[25] Arduino Company. *Arduino Uno Rev3*. URL: https://store.arduino.cc/usa/arduino-uno-rev3 (acesso em 30/08/2019).

[26] Semtech Corporation. *AN1200.22. LoRa^{TM} Modulation Basics*. Mai. de 2015.

[27] D. Crockford. *Introducing JSON*. 2001. URL: https://www.json.org/json-en.html (acesso em 11/07/2022).

[28] N. Dalal e B. Triggs. "Histograms of Oriented Gradients for Human Detection". Em: *International Conference on Computer Vision & Pattern Recognition* (jun. de 2005).

[29] NumPy Developers. *NumPy*. 2019. URL: https://numpy.org (acesso em 31/08/2019).

[30] MDN Web Docs. *An overview of HTTP*. Versão 1.09. Mai. de 2022. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview (acesso em 11/07/2022).

[31] *Dual Element Detector. Datasheet: LHI 778*. 1ª ed. Publicação Eletrônica. PerkinElmer. Abr. de 2001.

[32] T. Elshabrawy e J. Robert. "Closed-Form Approximation of LoRa Modulation BER Performance". Em: *IEEE Communication Letters* (set. de 2018).

[33] Info Escola. *Espectro Eletromagnético*. URL: https://www.infoescola.com/fisica/espectro-eletromagnetico/ (acesso em 04/04/2019).

[34] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. University of California, Irvine, 2000.

[35] J. Fraden. *The measurement, instrumentation, and sensors handbook*. 1ª ed. CRC Press LLC, 1999. ISBN: 978-0-8493-8347-2.

[36] J. Galbally, S. Marcel e J. Fierrez. "Biometric Antispoofing Methods: A Survey in Face Recognition". Em: *IEEE Access* 2 (jan. de 2014), pp. 1530–1552. DOI: 10.1109/ACCESS.2014.2381273.

[37] L. Gaspar. *Protocolo HTTP: entenda o que é e para que serve!* Versão 1.1. Jun. de 2021. URL: https://www.hostgator.com.br/blog/o-que-e-protocolo-http/ (acesso em 11/07/2022).

[38] A. Geitgey. *The world's simplest facial recognition api for Python and the command line. Documentation. Release 1.2.3*. 2017. URL: https://buildmedia.readthedocs.org/media/pdf/face-recognition/latest/face-recognition.pdf (acesso em 10/08/2019).

[39] A. Geitgey. *The world's simplest facial recognition api for Python and the command line*. Mar. de 2017. URL: https://github.com/ageitgey/face_recognition (acesso em 08/08/2019).

[40] R. C. Gonzalez e R. E. Woods. *Digital Image Processing*. 2ª ed. Upper Saddle River, NJ: Prentice Hall, 2001. ISBN: 0201180758.

[41] M. Guidino. *How Do Motion Sensors Work? Types & Applications*. Fev. de 2018. URL: https://www.arrow.com/en/research-and-events/articles/how-motion-sensors-work (acesso em 30/06/2022).

[42] S. K. Gupta e P. Sinha. "Overview of Wireless Sensor Network: A Survey". Em: *International Journal of Advanced Research in Computer and Communication Engineering* 3 (jan. de 2014). ISSN: 2278-1021.

[43] V. Gupta. *Face Detection – OpenCV, Dlib and Deep Learning (C++ / Python)*. URL: https://www.learnopencv.com/face-detection-opencv-dlib-and-deep-learning-c-python/ (acesso em 08/09/2019).

[44] Red Hat. *O que é uma API?* 2017. URL: https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces (acesso em 11/07/2022).

[45] S. Haykin. *Neural Networks and Machine Learning*. 3ª ed. Upper Saddle River, NJ: Pearson, 2008. ISBN: 0131471392.

[46] S. Haykin e M. Moher. *Sistemas Modernos de Comunicações Wireless*. 1ª ed. São Paulo, SP: Bookman Editora, 2008. ISBN: 0130224723.

[47] ECMA International. *Standard ECMA-404. The JSON Data Interchange Syntax*. Dez. de 2017. URL: https://www.ecma-international.org/wp-content/uploads/ECMA-404_2nd_edition_december_2017.pdf (acesso em 11/07/2022).

[48] A. Hadid e M. Pietikäinen J. Komulainen. "Face Spoofing Detection Using Dynamic Texture". Em: *Lecture Notes in Computer Science* 7728 (jan. de 2013), pp. 146–157. DOI: 10.1007/978-3-642-37410-4_13.

[49] A. Hadid e M. Pietikäinen J. Määttä. "Face spoofing detection from single images using texture and local shape analysis". Em: *IET Biometrics* 1 (abr. de 2012), pp. 3–10. DOI: 10.1049/iet-bmt.2011.0009.

[50] A. K. Jain, P. Flynn e A. A. Ross. *Handbook of biometrics*. Springer Science & Business Media, 2007. ISBN: 978-0-387-71040-2.

[51] M. Jamshed et al. "Significant HOG. Histogram of Oriented Gradient Feature Selection for Human Detection". Em: *International Journal of Computer Applications* (dez. de 2015).

[52] D. E. King. *A toolkit for making real world machine learning and data analysis applications in C++*. 2017. URL: https://github.com/davisking/dlib (acesso em 08/08/2019).

[53] D. E. King. *Dlib 18.6 released. Make your own object detector!* 2014. URL: http://blog.dlib.net/2014/02/dlib-186-released-make-your-own-object.html?m=1 (acesso em 08/08/2019).

[54] D. E. King. *dlib vs OpenCV face detection*. 2014. URL: https://www.youtube.com/watch?v=LsK0hzcEyHI (acesso em 08/08/2019).

[55] D. E. King. *High Quality Face Recognition with Deep Metric Learning*. 2017. URL: http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html (acesso em 08/08/2019).

[56] D. E. King. "Max-Margin Object Detection". Em: *Cornell University Archive* (jan. de 2015).

[57] Adafruit Explore & Learn. *How PIRs Work*. URL: https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/how-pirs-work (acesso em 05/04/2019).

[58] E. Learned-Miller et al. "Labeled Faces in the Wild: A Survey". Em: abr. de 2016, pp. 189–248. ISBN: 978-3-319-25956-7. DOI: 10.1007/978-3-319-25958-1_8.

[59] F. Lundh. *Python Imaging Library*. 2011. URL: http://www.pythonware.com/products/pil/ (acesso em 31/08/2019).

[60] M. Lutz. *Learning Python. Powerful Object-Oriented Programming*. 4ª ed. Sebastopol, CA: O'Reilly Media, 2009.

[61] A. Martinez-Sala et al. "An accurate radio channel model for wireless sensor networks simulation". Em: *Journal of Communications and Networks* 7.4 (2005), pp. 401–407. DOI: 10.1109/JCN.2005.6387982.

[62] University of Massachussetts. *Labeled Faces in the Wild*. URL: http://vis-www.cs.umass.edu/lfw/ (acesso em 08/09/2019).

[63] M. A. Matin e M. M. Islam. "Overview of Wireless Sensor Network". Em: set. de 2012. Cap. 1. DOI: 10.5772/49376.

[64] E. L. de Medeiros. *Desenvolvimento de um Sistema de Aquisição de Dados em Rede Híbrida de Comunicação de uma Planta Hidráulica focado em LoRa. Monografia (Graduação)*. Publicação Eletrônica. João Pessoa, 2018.

[65] K. Mekki et al. "Overview of Cellular LPWAN Technologies for IoT Deployment: Sigfox, LoRaWAN, and NB-IoT". Em: *International Workshop on Mobile and Pervasive Internet of Things* (2018).

[66] A Survey On Face Detection Methods. Em: *ICATET* 1 (2014).

[67] MPJA. *HC-SR501 PIR Motion Detector. Datasheet*. Publicação Eletrônica. MPJA.

[68] J. Nakamura et al. *Image Sensors and Signal Processing for Digital Still Cameras*. 4ª ed. Boca Raton, FL: CRC Press LLC, 2006.

[69] Ltd. NiceRf Wireless Technology Co. *100mW LoRa Wireless Transceiver Module LoRa1276-915*. URL: https://www.nicerf.com/products/detail/100mw-lora-wireless-transceiver-module-lora1276-915.html (acesso em 16/01/2022).

[70] G. Nikolić et al. "Battery capacity estimation of wireless sensor node". Em: *2017 IEEE 30th International Conference on Microelectronics (MIEL)*. 2017, pp. 279–282. DOI: 10.1109/MIEL.2017.8190121.

[71] W. S. Noble. "What is a support vector machine?" Em: *Nature Biotechnology* 24 (mai. de 2006), pp. 1565–1567. URL: https://www.ifi.uzh.ch/dam/jcr:00000000-7f84-9c3b-ffff-ffffc550ec57/what_is_a_support_vector_machine.pdf.

[72] Pallets. *Welcome to Flask - Flask Documentation v2.1*. 2010. URL: https://flask.palletsprojects.com/en/2.1.x/ (acesso em 11/07/2022).

[73] RoboCore. *Sensor de Presença PIR - HC-SR501*. URL: https://www.robocore.net/loja/sensores/sensor-de-presenca-pir-hc-sr501 (acesso em 31/08/2019).

[74] V. Rodrigues et al. "LoRa System for Monitoring and Facial Recognition". Em: *Journal of Communication and Information Systems* 36 (2021), pp. 1–16. DOI: https://doi.org/10.14209/jcis.2021.1.

[75] V. J. C. Rodrigues e F. B. S. de Carvalho. "Cenário Atual, Perspectivas e Aplicações de IoT". Em: *Revista de Tecnologia da Informação e Comunicação* 10 (dez. de 2021), pp. 50–56. ISSN: 2237-5104. URL: https://rtic.com.br/index.php/rtic/article/view/120.

[76] A. Rosebrock. *An interview with Davis King, creator of the dlib toolkit*. PyImageSearch. Mar. de 2017. URL: https://www.pyimagesearch.com/2017/03/13/an-interview-with-davis-king-creator-of-the-dlib-toolkit/ (acesso em 08/09/2019).

[77] S. Balaban. "Deep learning and face recognition: the state of the art". Em: *SPIE Defense + Security* (mai. de 2015). DOI: 10.1117/12.2181526.

[78] C. Saad et al. "Comparative Performance Analysis of Wireless Communication Protocols for Intelligent Sensors and Their Applications". Em: *International Journal of Advanced Computer Science and Applications* 5 (set. de 2014). DOI: 10.14569/IJACSA.2014.050413.

[79] K. El Sahmarani, Z. Simeu-Abazi e P. Ladret. "Vision system for Command and Fault detection". Em: *2006 International Conference on Service Systems and Service Management*. Vol. 2. 2006, pp. 1648–1652. DOI: 10.1109/ICSSSM.2006.320793.

[80] Saravati. *Módulo LoRa 915MHz 100mW - LoRa 1276 V2.0*. Nov. de 2020. URL: https://www.saravati.com.br/Modulo-LoRa-915MHz-100mW-LoRa1276-V2 (acesso em 16/01/2022).

[81] J. Schmidhuber. "Deep learning in neural networks: An overview". Em: *Neural Networks* 61 (2015), pp. 85–117. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2014.09.003. URL: https://www.sciencedirect.com/science/article/pii/S0893608014002135.

[82] Amazon Web Services. *O que é uma API?* 2019. URL: https://aws.amazon.com/pt/what-is/api/ (acesso em 11/07/2022).

[83] D. Sharma, S. Verma e K. Sharma. "Network Topologies in Wireless Sensor Networks: A Review". Em: *International Journal of Eletronics & Communication Technology* 4 (jun. de 2013). ISSN: 2230-7109.

[84] S. Sharma, S. Sharma e A. Athaiya. "Activation Functions in Neural Networks". Em: *International Journal of Engineering Applied Sciences and Technology* 4 (abr. de 2020), pp. 310–316.

[85] S. Sinha. *State of IoT 2021: Number of connected IoT devices growing 9% to 12.3 billion globally, cellular IoT now surpassing 2 billion.* 2021. URL: https://iot-analytics.com/number-connected-iot-devices/ (acesso em 20/12/2021).

[86] Global Sources. *Attractive PIR Sensor, PIR Motion Sensor, Infrared Sensor, Replacement for Nicera RE200B.* URL: https://www.globalsources.com/si/AS/Eagle-Power/6008821063503/pdtl/Attractive-PIR-Sensor/1129250277.htm (acesso em 03/04/2019).

[87] C. P. Souza et al. "On Harvesting Energy from Tree Trunks for Environmental Monitoring". Em: *International Journal of Distributed Sensor Networks* 12 (jun. de 2016). DOI: https://doi.org/10.1155%2F2016%2F9383765.

[88] K. R. Srinath. "Python – The Fastest Growing Programming Language". Em: (dez. de 2017).

[89] *SX1276/77/78/79 - 137MHz to 1020MHz Low Power Long Range Transceiver. Datasheet: SX1276/77/78.* 1ª ed. Publicação Eletrônica. Semtech. Ago. de 2016.

[90] B. Traversy. *Examples for Python Face Recognition library.* URL: https://github.com/bradtraversy/face_recognition_examples (acesso em 08/08/2019).

[91] Google Trends. *LoRa. Interesse ao longo do tempo.* URL: https://trends.google.com.br/trends/explore?date=2008-01-01%202022-01-08&q=LoRa (acesso em 08/01/2022).

[92] Z. A. Siddiqui e U. Park. "Face spoofing attack detection using spatial frequency and gradient-based descriptor". Em: *KSII Transactions on Internet and Information Systems* 13 (fev. de 2019), pp. 892–911. DOI: 10.3837/tiis.2019.02.022.

[93] L. Vangelista. "Frequency Shift Chirp Modulation: The LoRa Modulation". Em: *IEEE Signal Processing Letters* (dez. de 2017).

[94] P. Viola e M. Jones. "Rapid Object Detection using a Boosted Cascade of Simple Features". Em: *Conference on Computer Vision and Pattern Recognition* (2001). URL: http://www.merl.com/publications/docs/TR2004-043.pdf (acesso em 13/09/2019).

[95] P. Viola e M. Jones. "Robust Real-time Object Detection". Em: *Second International Workshop on Statistical and Computational Theories of Vision - Modeling, Learning, Computing and Sampling* (jul. de 2001).

[96] M. Wang e W. Deng. "Deep Face Recognition: A Survey". Em: *arXiv* (abr. de 2018).

[97] K. Gromaszek e M. Junisbekov W. Wójcik. Em: *Face Recognition: Issues, Methods and Alternative Applications* (jul. de 2016). ISSN: 978-953-51-2421-4. DOI: 10.5772/62950.

[98] Wikipédia. *Aprendizado de máquina*. URL: https://pt.wikipedia.org/wiki/Aprendizado_de_m%C3%A1quina.

[99] RF Wireless World. *What is Infrared Pyroelectric Detector-Function,operation*. URL: https://www.rfwireless-world.com/Terminology/What-is-Infrared-Pyroelectric-Detector.html (acesso em 05/04/2019).

[100] J. Komulainen e A. Hadid Z. Boulkenafet. "Face Spoofing Detection Using Colour Texture Analysis". Em: *IEEE Transactions on Information Forensics and Security* 11 (jul. de 2016). DOI: 10.1109/TIFS.2016.2555286.

[101] Z. Boulkenafet. "Face Anti-spoofing in Biometric Systems". Em: *Biometric Security and Privacy* 1 (nov. de 2017), pp. 299–321. DOI: 10.1007/978-3-319-47301-7_13.

A Tabelas de Custos dos Protótipos

Na Tabela A.1 está apresentada a relação de custos dos componentes do nó sensor que compõe a camada de sensoriamento, baseados em preços listados em *sites* de compras *online*, com valores de Julho de 2022.

Tabela A.1: Tabela de custos estimados da camada de sensoriamento (nó sensor).

Componente	Qtde.	*AliExpress*	*Banggood*	*Mercado Livre*	*Amazon*
HC-SR501	1	R$4,49	R$22,57	R$13,28	R$20,90
Arduino Mega 2560	1	R$70,62	R$130,05	R$124,00	-
ArduCAM 2MP	1	-	-	-	US$25,99
2ad66-LORAV2	1	R$76,46 -	-	R$61,50	-

Com um custo médio para o sensor de presença (HC-SR501) de R$15,31; para o Arduino Mega, o preço médio de R$108,23; para o sensor de imagem (ArduCAM), o preço aproximado de R$138,77 e, para o módulo LoRa (2ad66), o preço médio de R$68,98. Dessa forma, o valor estimado para cada nó sensor é de aproximadamente R$331,29.

Este valor não leva em consideração taxas e serviços associados à aquisição dos componentes e nem o valor de conectores e adaptadores.

Já na Tabela A.2 vê-se a relação de custos dos componentes do *gateway* que compõe a camada de concentradores, também baseados em preços de *sites* de compras, com valores de Julho de 2022.

Tabela A.2: Tabela de custos estimados da camada de concentradores (*gateway*).

Componente	Qtde.	*AliExpress*	*Banggood*	*Mercado Livre*	*Amazon*
Arduino Uno	1	R$37,73	45,20	R$99,99	145,90
2ad66-LORAV2	1	R$76,46 -	-	R$61,50	-

Com um custo médio para o Arduino Uno de R$82,20 e, para o módulo LoRa (2ad66), o preço médio é de R$68,98. Dessa forma, o valor estimado para cada *gateway* é de aproximadamente R$151,18.

Este valor não leva em consideração taxas e serviços associados à aquisição dos componentes e nem o valor de conectores e adaptadores ou o valor da máquina executando a etapa de *software* da camada de concentradores.

Considerando o protótipo da RSSF utilizado na etapa de validação, contendo um *gateway* e dois nós sensores, o valor estimado para os componentes da RSSF é de R$813,76.

Este custo refere-se somente aos dispositivos embarcados utilizados na validação, não incluindo outros dispositivos que integraram direta ou indiretamente o sistema, como computadores pessoais para o reconhecimento facial ou *modems* para o acesso à internet.