UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

ANTONIO DAVID TAVARES AMURIM

UMA SOLUÇÃO IOT VOLTADA À IDENTIFICAÇÃO DE EQUIPAMENTOS ELÉTRICOS COM O USO DA COMPUTAÇÃO DE BORDA

QUIXADÁ

2022

ANTONIO DAVID TAVARES AMURIM

UMA SOLUÇÃO IOT VOLTADA À IDENTIFICAÇÃO DE EQUIPAMENTOS ELÉTRICOS COM O USO DA COMPUTAÇÃO DE BORDA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientador: Prof. Me. Marcos Dantas Ortiz

QUIXADÁ

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

A549s Amurim, Antonio David Tavares.
Uma solução IoT voltada à identificação de equipamentos elétricos com o uso da Computação de Borda / Antonio David Tavares Amurim. – 2022.
90 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Engenharia de Computação, Quixadá, 2022.
Orientação: Prof. Me. Marcos Dantas Ortiz.

1. Internet das Coisas. 2. Aprendizado do computador. 3. Fiware. 4. Computação em Nuvem. I. Título.
CDD 621.39

ANTONIO DAVID TAVARES AMURIM

UMA SOLUÇÃO IOT VOLTADA À IDENTIFICAÇÃO DE EQUIPAMENTOS ELÉTRICOS COM O USO DA COMPUTAÇÃO DE BORDA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Aprovada em: ___/___/____

BANCA EXAMINADORA

Prof. Me. Marcos Dantas Ortiz (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Regis Pires Magalhães
Universidade Federal do Ceará (UFC)

Prof. Dr. Paulo Armando Cavalcante Aguilar
Universidade Federal do Ceará (UFC)

Dedico este trabalho a Deus. Pois, até aqui o Senhor me ajudou.

AGRADECIMENTOS

Gostaria de agradecer a Deus por ter me ajudado ao longo desta caminhada. Pois, Ele quem me deu forças e as condições necessárias para conseguir este feito.

Agradeço também ao meu estimado orientador, Prof. Me. Marcos Dantas, por todo incentivo, ensino, auxílio, confiança, suporte e dedicação no decorrer deste trabalho. De fato, este tempo de parceria contribuiu significativamente para a minha carreira profissional.

Agradeço ainda ao Engenheiro de laboratório Abdul-Hamid. Visto que, me ajudou consideravelmente na construção deste projeto, auxiliando e orientando no desenvolvimento das ideias.

Ademais, gostaria de agradecer a todos os professores que contribuíram profissionalmente para a minha carreira e me repassaram os conhecimentos necessários para o desenvolvimento deste trabalho.

Além disso, quero agradecer aos meus pais, Irenir Amurim e Carina Amurim, pelo incentivo e esforço que me proporcionaram a oportunidade de alcançar esta conquista. Bem como, toda minha família que sempre me apoiaram ao longo deste trajeto.

Ainda, quero agradecer a minha noiva Sara Aline pelo incentivo, força e compreensão durante esta jornada.

Agradeço ainda aos meus amigos e colegas, em especial aos do grupo Resistência EC, que compartilharam comigo muitos momentos excepcionais e me ajudaram ao longo deste curso.

Por fim, agradeço a todos que não foram citados, mas que contribuíram diretamente e indiretamente para a finalização deste ciclo.

“Se eu vi mais longe, foi por estar sobre ombros de gigantes.”

(Isaac Newton)

RESUMO

O mau gerenciamento dos recursos energéticos tem se tornado um grande problema nos centros urbanos, pois fere não apenas o meio ambiente, mas também os recursos financeiros, se agravando em grandes construções. Mais especificamente, a falta de gerenciamento de quais aparelhos elétricos estão em funcionamento pode acarretar gastos desnecessários de energia elétrica. Além disso, o uso descontrolado de alguns aparelhos pode diminuir sua vida útil causando ainda mais prejuízos financeiros. Todavia, a Internet das Coisas (*Internet of Things - IoT*) aliada à Inteligência Artificial traz a ideia de ambientes inteligentes que inserem a computação no mundo físico com o intuito de monitorar variáveis físicas, automatizar atividades corriqueiras e realizar previsões. Neste trabalho, é proposta uma solução *IoT* com foco na identificação de aparelhos elétricos através do monitoramento do sinal de consumo de corrente elétrica. Deste modo, são tratados os procedimentos de construção das diferentes partes projeto, como o desenvolvimento do dispositivo *IoT* responsável pela coleta do sinal elétrico, a implantação dos serviços de Nuvem com o uso da Computação de Borda utilizando a plataforma *FIWARE*, a construção do conjunto de dados com o treinamento e a avaliação dos modelos de classificação baseados em Aprendizado de Máquina, bem como o desenvolvimento do *dashboard* para visualização dos dados de monitoramento. Ademais, foram realizados testes e apresentados resultados promissores, indicando que a solução possui um monitoramento de qualidade com taxa de acertos variando entre 90% e 100% na identificação da maioria dos equipamentos elétricos monitorados.

Palavras-chave: Internet das Coisas. Aprendizado do computador. *Fiware*. Computação em Nuvem.

ABSTRACT

Poor management of energy resources has become a major problem in urban centers, as it harms not only the environment, but also financial resources, worsening in large constructions. More specifically, the lack of management of which electrical appliances are in operation can lead to unnecessary electrical energy expenditures. In addition, the uncontrolled use of some devices can reduce their useful life by causing even more financial losses. However, the Internet of Things (IoT) combined with Artificial Intelligence brings the idea of intelligent environments, which insert computing into the physical world in order to monitor physical variables, automate everyday activities and make predictions. In this work, an IoT solution is proposed focusing on the identification of electrical appliances by monitoring the electrical current consumption signal. In this way, the construction procedures of the different parts of the project are treated, such as the development of the IoT device responsible for collecting the electrical signal, the implementation of Cloud services using Edge Computing using the FIWARE platform, the construction of the set of data with the training and evaluation of classification models based on Machine Learning, as well as the development of the dashboard to visualize the monitoring data. In addition, tests were carried out and promising results were presented, indicating that the solution has a quality monitoring with a hit rate ranging between 90% and 100% in the identification of most of the monitored electrical equipment.

Keywords: Internet of Things. Computer learning. Fiware. Cloud Computing.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ILM	*Intrusive Load Monitoring*
NILM	*Non-Intrusive Load Monitoring*
IoT	*Internet of Things*
CoT	*Cloud of Things*
GEs	*Generic Enablers*
GPIO	*General Purpose Input/Output*
SoC	*System on Chip*
SCT	*Split-core Current Transformer*
ADC	*Analog-to-Digital Converter*
SDA	*Serial Data*
SCL	*Serial Clock*
SPS	*Samples per Second*
PGA	*Programmable Gain Amplifier*
A	Ampère
SI	Sistema Internacional
KNN	*K-Nearest Neighbor*
SVM	*Support Vector Machine*
GBM	*Gradient Boosting Machine*
VP	Verdadeiro Positivo
VN	Verdadeiro Negativo
FN	Falso Negativo
FP	Falso Positivo
RNAs	Redes Neurais Artificiais
REDD	*Reference Energy Disaggregation Data Set*
LoRa	*Long Range*
Trimpot	*Trimmer Potentiometer*
PCB	*Printed Circuit Board*
CSV	*Comma Separated Values*
MLP	*Multilayer Perceptron*

SUMÁRIO

1 INTRODUÇÃO

Cerca de 60% do consumo mundial de eletricidade é representado pelos edifícios comerciais e residenciais (ROCHA *et al.*, 2019). Logicamente, grandes construções tendem a ter maiores consumos devido ao grande número de dispositivos elétricos ligados, grande parte desse consumo sendo desnecessário, consequência do mau gerenciamento dos recursos energéticos, como mostra o estudo de caso feito no trabalho de Ferreira *et al.* (2018). Nesse contexto, o mau gerenciamento dos recursos energéticos tem impactos significantes na sustentabilidade e nos custos financeiros com energia elétrica.

Os autores Salhofer *et al.* (2019) exibem uma ideia de cidade inteligente onde diferentes ambientes urbanos são monitorados de modo a coletar dados de temperatura, umidade e concentração de partículas finas (PM2.5 e PM10) para análise de qualidade do ar. Nesse raciocínio, se elevou o interesse nas chamadas construções inteligentes que monitoram e regulam dinamicamente a utilização da energia elétrica como, por exemplo, com o controle automático dos dispositivos, alerta de consumo, definição de horários de funcionamento e análise das propriedades de um equipamento específico, buscando constatar anomalias de modo a reduzir o consumo e otimizar o uso de aparelhos elétricos.

No trabalho de Tundis *et al.* (2019) é descrita a identificação de equipamentos elétricos como um dos caminhos para constatar anomalias no consumo de energia, onde é possível comparar as variáveis reais de consumo com as informadas pelo fabricante do dispositivo reconhecido. Contudo, o problema de identificação das assinaturas de cargas elétricas possui algumas ramificações como identificar apenas o tipo do dispositivo elétrico, ou seja, conhecer suas características resistivas, indutivas e capacitivas para classificá-los em aparelhos puramente resistivos, que possuem motores, eletrônicos, etc. Ademais, pode ser feita a classificação mais aguçada, identificando de fato qual dispositivo está consumindo energia elétrica, como mostrado em Barker *et al.* (2014). Ainda, é possível distinguir dispositivos aparentemente idênticos, como tratado em Paixao *et al.* (2016). Vale ressaltar, que o custo e a complexidade de implantação da solução influencia na aceitação do projeto por parte dos responsáveis pelo gerenciamento de energia dos edifícios.

Como mostrado em Abeykoon *et al.* (2016) e Khan *et al.* (2019), algoritmos de Aprendizado de Máquina (*Machine Learning*) podem ser usados para reconhecer padrões no sinal elétrico objetivando classificar o equipamento monitorado. Entretanto, o número de amostras capturadas precisam ser suficientes para extrair os recursos (*features*) necessários para treinar

o modelo de classificação e posteriormente rotular o sinal elétrico do dispositivo que será identificado.

Segundo Abubakar *et al.* (2017), existem dois modelos de monitoramento do consumo energético, empregados de modo a realizar a captura de amostras do sinal. São eles o modelo intrusivo ou *Intrusive Load Monitoring* (*ILM*) e o não intrusivo ou *Non-Intrusive Load Monitoring* (*NILM*). O *ILM* corresponde a medir o consumo de eletricidade dos aparelhos usando um dispositivo de baixo custo sendo necessário invadir o ambiente elétrico para realizar a medição. Por outro lado, o método *NILM*, também chamado medição de um sentido, consiste em medir o consumo energético utilizando um medidor inteligente baseando-se em um único ponto de medição não invadindo o ambiente elétrico.

Contudo, em uma construção inteligente vários desafios podem ser encontrados, principalmente no que diz respeito à latência, cobertura, disponibilidade, escalabilidade e custo. No entanto, paradigmas recentes como a computação em névoa, também chamada computação de borda, que estende o processamento em nuvem até a borda de uma rede, torna projetos voltados a *Internet* das Coisas ou *Internet of Things* (*IoT*) de baixa latência por disponibilizar um núcleo de processamento dedicado próximo ao dispositivo *IoT* (FRAGA-LAMAS *et al.*, 2019). Neste raciocínio, há diversas plataformas *IoT* que propõem melhorias na escalabilidade e no gerenciamento dos dados capturados, como as listadas em Farahzadi *et al.* (2018), Alberti *et al.* (2019) e Batista *et al.* (2018). Tais ferramentas implementam conceitos de Nuvem das Coisas ou *Cloud of Things* (*CoT*), que traz a ideia de dispositivos *IoT* conectados a Nuvem, ainda, algumas delas oferecem serviços de Computação de Borda.

Dito isso, este trabalho busca a identificação eficiente de dispositivos através de padrões existentes no sinal de corrente elétrica. Para isso, amostras são capturadas do sinal empregando o modelo *NILM* e utilizados algoritmos de Aprendizado de Máquina para classificar o equipamento elétrico. Ademais, o trabalho recorre ao *middleware FIWARE* (CIRILLO *et al.*, 2019) para o gerenciamento e armazenamento dos dados de contexto. Especialmente, é empregada a ferramenta *FogFlow* (CHENG *et al.*, 2017) disponibilizada pelo *FIWARE* para implementação dos paradigmas de Computação de Borda. Assim, este trabalho é destinado aos gestores de recursos energéticos que visam reduzir o consumo por meio da identificação de eventuais dispositivos ligados indevidamente. Ainda, este trabalho possui relevância para pesquisas que procuram aplicar o uso de *middlewares* e conceitos de Computação de Borda em um contexto de construção inteligente.

O presente trabalho relata os aspectos principais para a execução deste projeto. No Capítulo 2, é explanada a fundamentação teórica, que aborda os principais conceitos referentes aos objetivos deste trabalho. No Capítulo 3, são listados e detalhados os trabalhos relacionados a este projeto, bem como, feito um comparativo entre eles. No Capítulo 4, são abordados os procedimentos metodológicos realizados. No Capítulo 6, são apresentados e explicados os resultados obtidos. Por fim, no Capítulo 7 é descrita a conclusão e relatados os trabalhos futuros.

1.1 Objetivos

O objetivo geral deste trabalho é monitorar e identificar equipamentos elétricos por meio da análise do sinal de consumo utilizando conceitos de *IoT* e Computação de Borda. Como objetivos específicos estão:

1. Projetar e construir um dispositivo *IoT* eficiente na aquisição das amostras do sinal elétrico.
2. Criar um conjunto de dados com as amostras capturadas pelo dispositivo *IoT*.
3. Utilizar tecnologias e ferramentas *IoT* para prover serviços de Nuvem e Borda.
4. Treinar e avaliar modelos de classificação e executá-los no nó de Borda, com o intuito de identificar aparelhos elétricos dadas as amostras do sinal de corrente.
5. Validar a solução proposta realizando testes reais de identificação com diversos aparelhos elétricos.

1.2 Principais contribuições

A solução proposta possui contribuições em diferentes áreas da computação e no gerenciamento de recursos energéticos. Como principais contribuições estão:

1. Placa de circuito impresso utilizada no dispositivo *IoT* de captura do sinal de corrente.
2. Conjunto de dados com amostras de sinais de corrente de diversos dispositivos elétricos.
3. Arquitetura de comunicação e processamento em Nuvem utilizando Computação de Borda.
4. Execução de modelos de inteligência artificial, voltados a classificação, em um nó de Borda *Raspberry Pi*.
5. *Dashboard* de monitoramento energético.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são detalhados os pontos mais importantes para o entendimento deste trabalho. Na Seção 2.1, é tratado o conceito de ambiente inteligente onde são abordadas as principais ideias e definições. Na Seção 2.2, são detalhados os paradigmas da *IoT*, onde é explicado o modelo de comunicação em camadas *IoT*. Na Seção 2.3, é introduzida a ideia de Nuvem das Coisa e esplanada a técnica de Computação de Borda. Na Seção 2.4, é tratado o *middleware FIWARE* e os serviços utilizados neste trabalho. Na Seção 2.5, são descritas as placas de desenvolvimento utilizadas neste projeto. Na Seção 2.6, são introduzidas as características dos sensores em geral e do sensor *SCT-013-000* em específico. Semelhantemente, na Seção 2.7, são detalhados os parâmetros de um conversor analógico-digital e do conversor *ADS1115* em especial. Na Seção 2.8, são explicados os conceitos de assinatura de carga, utilizados para identificar os aparelhos elétricos. Por fim, na Seção 2.9, são introduzidos os paradigmas de Aprendizado de Máquina, mais especificamente, os algoritmos utilizados para classificação.

2.1 Ambientes Inteligentes

Um exemplo de cidade inteligente possui infraestruturas tecnológicas em quase todas as suas partes. Além disso, apresenta soluções tecnológicas para os cidadãos em uma série de áreas como energia renovável, defesa cibernética, resistência tecnológica, pontos *Wi-Fi* (*IEEE* 802.11), aplicativos de governo eletrônico e municipalidade eletrônica, aplicativos móveis e outros campos (ÖZCAN *et al.*, 2017).

Nesse raciocínio, os conceitos de cidades inteligentes podem ser adaptados para construções inteligentes que empregam soluções tecnológicas em seus diversos setores. Praticamente, qualquer espaço físico pode se tornar inteligente. Um ambiente inteligente é um espaço que emprega tecnologia para conectar coisas ao mundo virtual, pretendendo aumentar o nível de consciência do que acontece em ambientes físicos e com as pessoas, além disso, ambientes inteligentes agregam valores às pessoas, aos negócios, melhorando a sustentabilidade, fazendo uso eficiente e eficaz dos recursos (ALBERTI *et al.*, 2019).

Atualmente, os serviços de *IoT* são soluções essenciais para fornecer ambientes inteligentes em residências, edifícios e cidades (FARAHZADI *et al.*, 2018). Nesse viés, outros conceitos relacionados à *IoT* surgem como soluções para a comunicação entre as coisas e o gerenciamento dos dados compartilhados. Dito isso, pode ser citada a ideia de Nuvem das Coisas

que traz artifícios de comunicação e armazenamento em um ambiente *IoT*, além da computação de borda, que aproxima a Nuvem dos dispositivos *IoT* criando um núcleo de processamento mais próximo. Ainda, o emprego de *middlewares*, protocolos leves, comunicação sem fio e dispositivos de baixo consumo e custo acessível facilitam a troca de mensagens e a construção de um ambiente inteligente.

Neste trabalho, os conceitos de ambiente inteligente se torna indispensável, pois a ideia de identificação de dispositivos elétricos está amplamente ligada à concepção de uma construção capaz de identificar quais aparelhos estão conectados em sua rede elétrica de modo a calcular o consumo, além de prever e diminuir gastos. Além disso, o propósito de monitorar a rede elétrica para capturar amostras do sinal de corrente e realizar a classificação é bastante facilitado com o emprego de noções de *IoT* e computação em Nuvem para o transporte, armazenamento e processamento dos dados.

2.2 *Internet* das Coisas

O uso da tecnologia em diferentes áreas e objetos está em constante crescimento, o que implica diretamente na necessidade de tudo estar conectado para proporcionar o relacionamento entre as coisas. Com isso, a *IoT* é uma das inovações em destaque que tem o potencial de trazer vários benefícios nos ambientes em que vivemos (ATLAM *et al.*, 2018), como a redução de custos, a acessibilidade e o desenvolvimento sustentável.

Na arquitetura de comunicação *IoT*, cada camada é definida por suas funções e pelos dispositivos e tecnologias usadas nessa camada. Contudo, a *IoT* implementa principalmente três camadas denominadas Percepção, Rede e Aplicação (MAHMOUD *et al.*, 2015; LIN *et al.*, 2017), ilustradas na Figura 1. Cada camada de *IoT* possui seus devidos desafios e tecnologias que auxiliam na solução dos problemas encontrados. Vale ressaltar que, existem derivações nesta arquitetura de três camadas acrescentando novas funções às camadas ou variando o número de níveis, como exemplificado em Wang *et al.* (2016). A seguir, as camadas são detalhadas, mostrando os papéis de cada uma e os dispositivos que nela atuam.

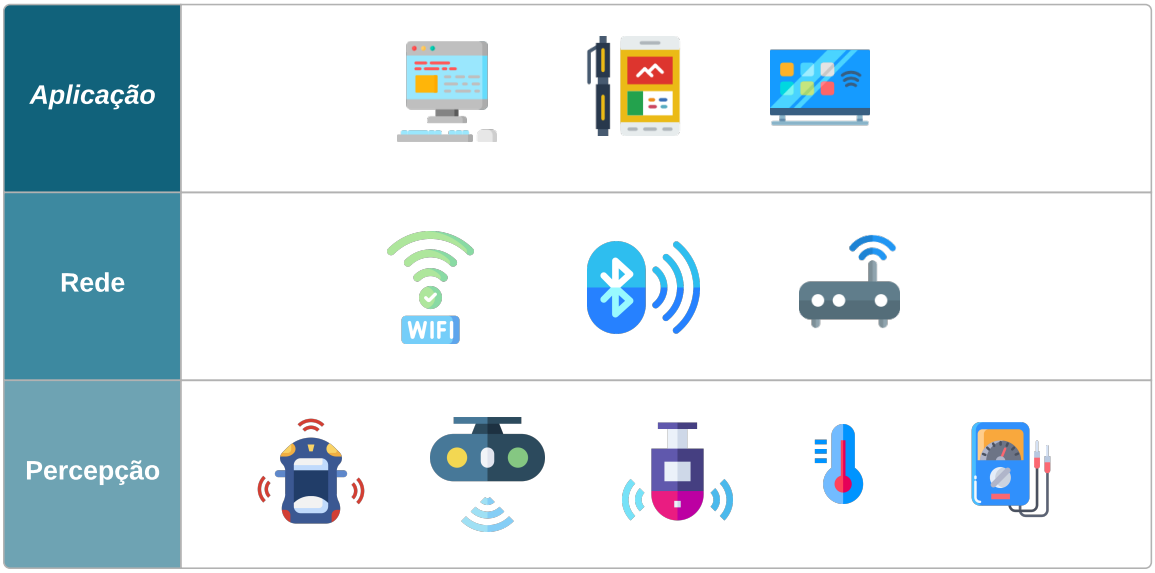
– **Percepção:** É a camada de nível mais baixo, também conhecida como camada dos sensores. Seus principais objetivos são conectar objetos na rede *IoT* realizando coleta e medição dos dados além de realizar processamentos através de *hardwares* inteligentes denominados dispositivos *IoT*. Assim, a camada de percepção interage com ambientes físicos e componentes através de sensores e atuadores, transmitindo as informações

processadas para a camada superior (LIN *et al.*, 2017).

– **Rede:** A camada de rede, também chamada transmissão ou *gateway*, atende à função de roteamento de dados para diferentes nós *IoT* pela arquitetura de comunicação. É arquitetada como camada intermediária na estrutura de comunicação *IoT*. Nesta camada, plataformas de Computação em Nuvem, *gateways* de *internet*, dispositivos de comutação e roteamento operam usando algumas das tecnologias mais recentes, como *Wi-Fi*, *Bluetooth*, *3G*, *Zigbee*, entre outras (MAHMOUD *et al.*, 2015). A camada de *gateway* é um conjunto de nós com capacidades de processamento relativamente altas buscando agregar, filtrar e transmitir dados entre aplicação e dispositivos *IoT*.

– **Aplicação:** A camada de aplicação tem como função receber os dados transmitidos da camada de transmissão para realizar operações necessárias e fornecer serviços. Por exemplo, disponibilizar gráficos ou tabelas resultantes da análise das informações recebidas, correlação de dados, armazenamento das informações em um banco de dados, geração de alerta condicionada pelas variáveis recebidas, envio de comandos aos dispositivos atuadores, entre outros serviços (LIN *et al.*, 2017).

Figura 1 – Camadas IoT



Fonte: Elaborada pelo próprio autor

Nesse contexto, o conceito de comunicação dividida em camadas *IoT* é utilizado neste trabalho buscando simplificar a troca de informações entre as diferentes partes do projeto. Além disso, o meio de comunicação sem fio é empregado objetivando facilitar a implantação dos vários dispositivos *IoT* que fazem a captura das amostras de corrente. Ainda, outras

caracterizações de *IoT* como ferramentas de gerenciamento de dados de contexto e modelos de transporte eficientes de dados também são vistas neste trabalho.

2.3 Nuvem das Coisas

Devido ao grande número de coisas conectadas à *internet*, além da alta taxa de crescimento, alguns desafios foram levantados como gerenciamento, agregação e armazenamento de grandes dados produzidos. Para resolver alguns desses problemas, a Computação em Nuvem surgiu para a *IoT* como Nuvem das Coisas, que fornece serviços virtuais ilimitados para aprimorar as plataformas *IoT* em grande escala (FARAHZADI *et al.*, 2018). Nesse contexto, se tornou possível os projetos de ambientes inteligentes que vão desde cidades a diferentes construções com inúmeros dispositivos conectados a Nuvem, compartilhando dados de modo a melhorar o convívio humano e utilizar os recursos de maneira eficiente.

A *IoT* diz respeito a dispositivos amplamente distribuídos e com recursos de *hardware* limitados que capturam e compartilham dados. Por outro lado, a Computação em Nuvem compreende um alto poder de computação e armazenamento teoricamente ilimitado, o que complementa, em parte, a *IoT* (ATLAM *et al.*, 2017). Assim, a *IoT* e a Computação em Nuvem crescem em conjunto e evoluem cada vez mais com o surgimento de novas ideias e ferramentas que facilitam a comunicação e o processamento das informações obtidas pelos dispositivos *IoT*.

Todavia, apenas a Computação em Nuvem não supre todos os problemas encontrados em ambientes inteligentes, devido à necessidade de baixa latência e tempo rápido de resposta. Ainda, há a agravante dos custos de largura de banda e armazenamento serem substancialmente altos se todos os dados forem enviados para a Nuvem (CHENG *et al.*, 2017). Então, surgiu a ideia de transferir parte da computação da Nuvem e dos dispositivos *IoT* para uma camada intermediária, sendo *gateways* de *IoT*, conhecidos como Borda de Nuvem ou Nevoeiro. Assim, parte do processamento e armazenamento em Nuvem permanece próximo aos dispositivos *IoT*, garantindo uma comunicação mais rápida e menos acesso à Nuvem.

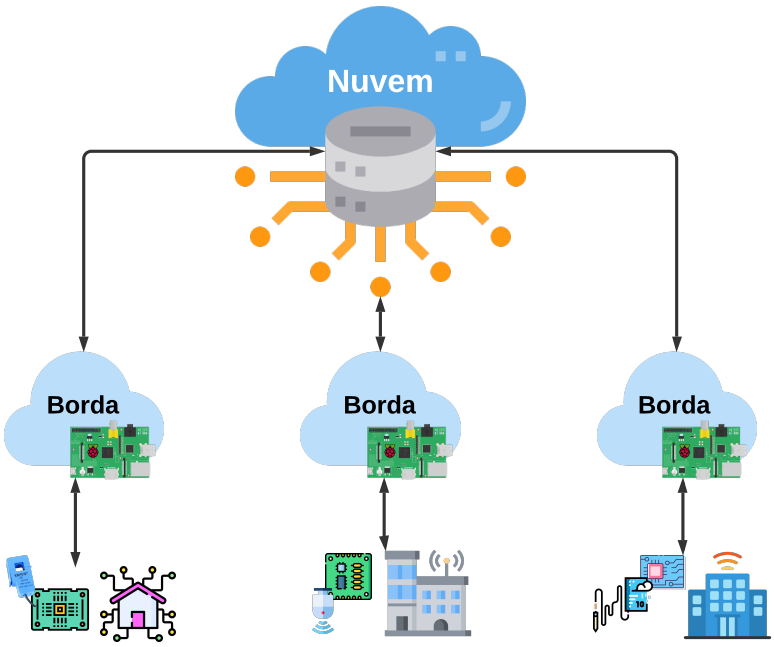
Neste trabalho, os conceitos de Computação em Nuvem, bem como, de Computação de Borda são empregados utilizando a plataforma *FIWARE*, que facilita a integração da Nuvem e da Borda com as demais camadas do projeto. Nesse viés, procura-se aplicar as principais vantagens de computar dados em Nuvem, adicionando o artifício de Computação de Borda, buscando diminuir a latência no transporte e no processamento das amostras de corrente.

2.3.1 Computação de Borda

A Computação de Borda estende os atributos da Computação em Nuvem para próximo dos dispositivos sensores e atuadores que agem diretamente na intersecção das tecnologias computacionais com o ambiente físico. Assim, a Computação de Borda oferece enormes facilidades para armazenamento, processamento e rede entre sensores ou usuários finais, possuindo características de baixa latência, reconhecimento de localização, distribuição geográfica e suporte para mobilidade e interação em tempo real (VERMA *et al.*, 2016).

Na Figura 2 é mostrada a arquitetura básica de uma rede de comunicação que integra a Computação de Borda. Como já descrita e ilustrada, a Borda está localizada em uma camada intermediária, realizando processamentos e armazenamentos mais próximos dos dispositivos inteligentes. Também chamados nós de névoa, os nós de borda são normalmente executados em microcomputadores como o *Raspberry Pi*, trocando informações com o nó de Nuvem apenas quando necessário (KANANI; PADOLE, 2020; XU; ZHANG, 2019; BATTULGA *et al.*, 2020).

Figura 2 – Arquitetura de comunicação com Computação de Borda



Fonte: Elaborada pelo próprio autor

Segundo Atlam *et al.* (2018) e Yi *et al.* (2015), as propriedades da Computação em Névoa podem ser sintetizadas nos seguintes pontos:

– **Percepção de localização e distribuição geográfica:** A Computação de Borda possui o serviço de detecção de localização, onde diferentes nós de Névoa podem ser implantados

em locais distintos e registradas as coordenadas da sua posição.

- **Baixa latência:** Além disso, como a Borda está fisicamente mais próxima dos dispositivos *IoT*, há uma menor latência na comunicação. Bem como, uma quantidade de acesso menor à Nuvem.
- **Escalabilidade:** A névoa também oferece recursos de processamento e armazenamento distribuídos que supre a demanda de uma rede de dispositivos inteligentes de grande escala.
- **Interações em tempo real:** As aplicações de Computação de Borda fornecem interações em tempo real entre nós de Borda, em vez do processamento em lote empregado no nó de Nuvem.
- **Heterogeneidade:** Ainda, a Névoa consegue tratar diferentes dispositivos finais, projetados por fabricantes distintos que precisam ser implantados de acordo com suas plataformas.
- **Interoperabilidade:** Os componentes do nó de Borda podem se comunicar implementando protocolos variados entre diferentes prestadores de serviços, podendo atender dispositivos *IoT* que utilizam protocolos específicos.

Como visto, a Computação de Borda agrega uma série de vantagens para um ambiente *IoT* no que diz respeito a computação, comunicação em rede e armazenamento. Por isso, neste trabalho procura-se aplicar essa ideia. Além disso, existem diversas plataformas *IoT* que oferecem serviços de Computação de Borda já encorporado e com a arquitetura de comunicação já definida para facilitar a implementação como, por exemplo, a plataforma *FIWARE* que possui inúmeros serviços de Nuvem e Névoa voltados a IoT.

2.4 *FIWARE*

O *FIWARE* é uma plataforma *IoT* de código aberto que oferece vários microsserviços denominados de *Generic Enablers* (*GEs*) (CIRILLO *et al.*, 2019), que proporcionam funções reutilizáveis e comumente compartilhadas atendendo a várias áreas de uso em vários setores de um ambiente inteligente. Os *GEs* são partes dessa tecnologia e incluem serviços de hospedagem em Nuvem, computação em Névoa, gerenciamento de dados/contexto, arquitetura de aplicativos, ativação de serviços de *IoT*, interface para redes e dispositivos, segurança, *middleware* moderno e interface de usuário baseada na *web* (PLAMANESCU *et al.*, 2019).

No Quadro 1 são listados alguns dos serviços oferecidos pela plataforma *FIWARE* (MUNOZ-ARCENTALES *et al.*, 2020; CIRILLO *et al.*, 2019). Como pode ser notado, há várias opções de *GEs* que proporcionam serviços em diferentes setores, em que alguns atuam em

áreas semelhantes, cada um com vantagens dependendo do contexto em que são inseridos. Vale ressaltar que, mesmo os *GEs* que atuam em áreas próximas, podem ser utilizados em conjunto para aproveitar suas características específicas e suprir as necessidades do sistema.

Quadro 1 – Alguns dos serviços oferecidos pelos *GEs* do *FIWARE*

Serviço	***GEs***
Gerenciamento de contexto central	*Orion, Scorpio* e *Stellio*
Persistência e análise de dados	*STH Comet, Cygnus, Draco, Cosmos* e *QuantumLeap*
Interface *IoT* e sistemas de terceiros	*IDAS, OpenMTC, FAST DDS, Firos* e *Oliot*
Processamento, análise e visualização de contexto	*Wirecloud, Kurento, FogFlow, Perseo* e *OpenVidu*
Segurança e privacidade	*Keyrock, Wilma* e *AuthZForce*

Fonte: Elaborado pelo próprio autor.

Neste trabalho, são empregados três *GEs* do *FIWARE*. Deste modo, é utilizado o *IDAS* para integração entre o *middleware FIWARE* e os dispositivos de captura das amostras, o *FogFlow* para o processamento dos dados capturados na Borda e o *Orion* para gerenciar os dados resultantes do processamento. Na sequência, são descritos o funcionamento e as propriedades de cada um desses microsserviços, e detalhadas funções que exercem no projeto, além de como se relacionam com as diferentes partes da infraestrutura de comunicação.

2.4.1 IDAS

O *GE IDAS* (*Backend Device Management*) disponibiliza vários tradutores de protocolo de comunicação denominados *IoT Agent* que transformam protocolos como *MQTT*, *HTTP*, *LWM2M*, entre outros para o padrão *NGSI* utilizado pelo *FIWARE* (MARTÍNEZ *et al.*, 2016). Ainda, permite o gerenciamento de entidades de contexto associadas a dispositivos em conjunto com outros *GEs*, podendo criar entidades com atributos dinâmicos e estáticos para armazenar os dados de contexto atuais dos sensores, além de ser possível registrar comandos nesses objetos virtuais enviados ao dispositivo *IoT* sempre que requisitado.

Neste trabalho, é empregado o protocolo *MQTT* na comunicação dos dispositivos *IoT* por ser um protocolo de transporte leve que usa com eficiência a largura de banda da rede com um cabeçalho fixo de 2 *bytes* (KODALI; SORATKAL, 2016). O *MQTT* é um protocolo baseado em publicação/assinatura, funciona em *TCP* e garante a entrega de mensagens do nó para o servidor. Por ser um protocolo de troca de informações orientado a mensagens, o *MQTT* é ideal para os nós *IoT* que possuem capacidades e recursos limitados. Vale ressaltar que, qualquer conexão *MQTT* geralmente envolve duas categorias de agentes: clientes *MQTT* e *broker MQTT*. Dito isso, neste projeto é utilizado o *Mosquitto*, por ser o *broker* padrão do *MQTT*.

Nesse raciocínio, as amostras capturadas pelos dispositivos inteligentes são enviadas ao *GE IDAS* através do *broker Mosquitto*, utilizando o protocolo *MQTT*. Ainda, é utilizado o protocolo *Ultralight 2.0* para representação de dados externos, por ser um protocolo simples de texto plano, ideal para dispositivos com poucos recursos. Então, o *IDAS* traduz esses dados para o padrão *NGSI* para serem processados pelos demais *GEs* do *FIWARE*.

2.4.2 FogFlow

O *GE FogFlow* é uma ferramenta de execução distribuída para suportar fluxos de processamento dinâmico tanto na Nuvem quanto nas Bordas (CHENG *et al.*, 2017). Nesse contexto, o *FogFlow* é composto, essencialmente, por um nó de Nuvem onde é possível registrar inúmeros nós de Borda, além disso, são cadastrados operadores responsáveis pelo processamento dos dados. Então, os operadores podem ser implementados e executados na Nuvem ou na Borda. Logo, o *GE Fogflow* orquestra o fluxo de dados, direcionando-o para os nós em que estão localizados os operadores necessários para o processamento das informações.

Este *Enabler* também utiliza o padrão *NGSI* e um modelo de entidades compatível com os *GEs IDAS* e *Orion*, o que facilita a integração. Com isso, neste trabalho o *FogFlow* recebe as amostras do sinal de corrente já no padrão *NGSI* traduzidos pelo *IDAS*, realiza o processamento para identificar o dispositivo e envia os resultados para entidades no *GE Orion*. Assim, o *Orion* pode gerenciar os resultados e disponibilizá-los para camada de aplicação.

2.4.3 Orion Context Broker

O *GE Orion Context Broker*, ou somente *Orion*, é um núcleo de gerenciamento de dados (DETZNER; SALHOFER, 2020). Nele é possível criar entidade com atributos para armazenar dados atuais, para isso, utiliza a base de dados *MongoDB* para auxiliar na persistência. Ainda, é possível relacionar as entidades, organizando-as como objetos reais. Ademais, o *Orion* consegue registrar as chamadas *subscriptions* que envia os dados de um determinado atributo para um endereço sempre que acontece uma atualização.

Desta forma, neste trabalho o *Orion* é utilizado para criar entidades que representam os dispositivos *IoT*. Assim, as entidades armazenam os dados resultantes do processamento de cada pontos de monitoramento utilizando seus atributos, alertando e disponibilizando os resultados para a camada de aplicação sempre que as informações são atualizadas. Ainda, com o *Orion* é possível a comunicação com outros bancos de dados integrando outros *GEs*, facilitando

o armazenamento e a análise de dados históricos.

2.5 Placas de Desenvolvimento

Atualmente, várias placas de desenvolvimento são propostas pela comunidade e pelos fabricantes. Todavia, nota-se um maior interesse em placas desenvolvidas pela *Texas Instruments* (*BeagleBoard* e *BeagleBone*), *Raspberry Pi*, *Arduino* e *Espressif*. Na escolha da placa ideal para um projeto *IoT*, é necessário considerar algumas variáveis como memória (*RAM* e *Flash*), pinos de entrada/saída de propósito geral ou *General Purpose Input/Output* (*GPIO*), número de *bits* do núcleo de processamento (microprocessador ou microcontrolador), sistemas operacionais, conectividade, suporte e compatibilidade com sensores/atuadores (GAMESS; SMITH, 2020). Além disso, o custo é suma importância para a escalabilidade do projeto.

Neste projeto, é utilizado a placa microcontroladora *ESP-01*, fabricada pela *Espressif* que possui o microcontrolador *ESP8266* como principal núcleo de processamento. A *ESP-01* tem a função de ler as amostras de corrente capturadas pelo sensor e enviá-las ao *middleware FIWARE* para serem processadas. Além disso, é empregado o microcomputador *Raspberry Pi* que implementa o nó de Borda referente a *CoT*, de modo a realizar o processamento de classificação do sinal.

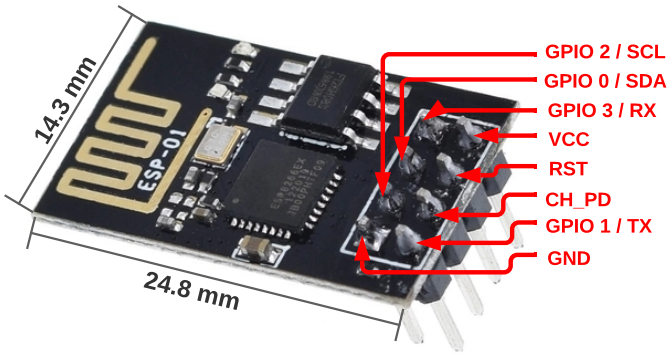
2.5.1 Microcontrolador ESP8266

O *ESP8266* é um microcontrolador de baixo custo e com baixo consumo de energia, conta ainda com um módulo *Wi-Fi*, o que o deixa bastante apropriado para projetos voltados a *IoT* (MESQUITA *et al.*, 2018). O *ESP8266* é eficiente para hospedar um aplicativo ou remover todas as funções de rede *Wi-Fi* de outro microcontrolador. Este módulo tem capacidade de processamento e armazenamento bastante robusto. Isso permite que ele se integre por portas *GPIO* a outros microcontroladores ou realize tarefas com o auxílio de sensores e atuadores (SRIVASTAVA *et al.*, 2018).

Existem inúmeras placas que possuem o *ESP8266* como microcontrolador (ARJADI *et al.*, 2018). Todavia, a *ESP-01* se destaca pelo seu tamanho reduzido, baixo custo e poucos pinos de expansão, portando não mais que o suficiente para desempenhar as atividades a ela atribuídas neste projeto. Além dos pinos de alimentação e configuração, a *ESP-01* possui quatro pinos de *GPIO* que podem ser multiplexados para exercer funções de *PWM*, *UART*, *I2C*, entre

outras, além da capacidade de processamento e armazenamento *onboard* que permite a integração com sensores e outros dispositivos voltados ao uso da *IoT* (ROSLI *et al.*, 2018). Neste trabalho, a *ESP-01* é empregada no dispositivo *IoT* que captura as amostras de corrente. A placa *ESP-01* é mostrada na Figura 3 com as informações de tamanho e pinagem.

Figura 3 – Placa *ESP-01*



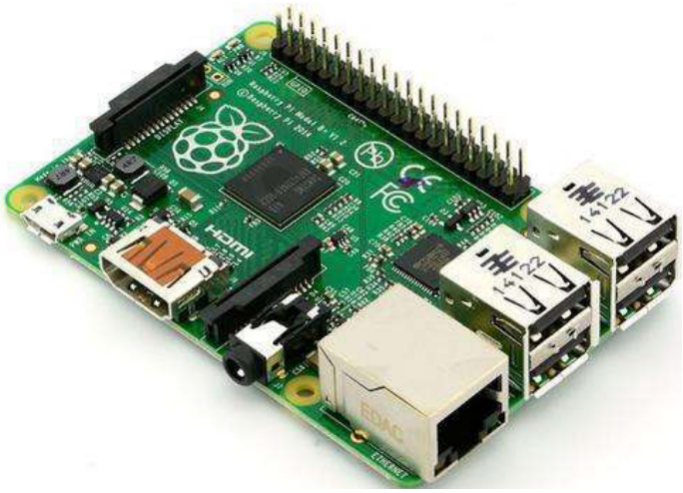
Fonte: Adaptado de Rosli *et al.* (2018)

2.5.2 Microcomputador Raspberry Pi

O *Raspberry Pi* é um *System on Chip* (*SoC*), um computador onde todos os circuitos necessários como núcleo central de processamento, unidade de processamento gráfico, módulos de entrada/saída e circuitos auxiliares estão integrados em uma única placa. Esse microcomputador pode ser usado em diferentes contextos por disponibilizar recursos como *GPIOs*, *Bluetooth* e *Wi-Fi* que torna possível a comunicação com microcontroladores, bem como sensores e atuadores (YAMANOOR; YAMANOOR, 2017). Ainda, possui uma entrada para cartão *MicroSD* que permite a instalação e uso de diversas distribuições *Linux*, incluindo o *Raspbian OS*, o sistema operacional padrão disponibilizado pela *Raspberry Pi Foundation*. O microcomputador *Raspberry Pi* pode ser visualizado na Figura 4.

Nesse contexto, o *Raspberry Pi* é bastante recomendado como nó de borda em arquiteturas que implementam a Computação em Nuvem, por possuir recursos suficientes para armazenar e processar dados de contexto e trocar informações facilmente com outros dispositivos das camadas *IoT* por meio da comunicação sem fio (KANANI; PADOLE, 2020). Além disso,

Figura 4 – *Raspberry Pi*



Fonte: Adaptado de Li *et al.* (2020)

possui um baixo custo comparado a um servidor em Nuvem que faria todo o processamento e armazenamento em um ambiente sem borda. Neste trabalho, o *Raspberry Pi* é utilizado como nó de Borda, responsável por analisar e classificar as amostras de corrente.

2.6 Sensores

Os sensores desempenham um papel importante na aquisição de dados em diferentes contextos, medindo e processando as informações capturadas para detectar mudanças em ambientes físicos. Assim, o sensor produz uma resposta, como um sinal elétrico, sempre há mudanças no ambiente monitorado. Atualmente, existem várias categorias de sensores adaptados para diversos projetos voltados a IoT, desta forma, os sensores são classificados com base em suas especificações, seu método de conversão, modelo de material usado, seu fenômeno físico de detecção, propriedades que medem e o campo de aplicação (SEHRAWAT; GILL, 2019).

Precisamente, o termo sensor é utilizado para referenciar elementos que detectam variáveis naturais. Entretanto, existem os chamados transdutores que convertem os dados capturados pelo sensor em um sinal detectável como elétrico, mecânico, óptico e outros (SOLOVEV; SHADRIN, 2017). Contudo, ambos os termos possuem o mesmo significado geralmente, com o termo sensor sendo empregado na maioria das vezes para as duas definições. Neste trabalho é utilizado o sensor de corrente não intrusivo *SCT-013-000*, com a finalidade de capturar amostras de corrente através do fio que conecta o equipamento, a ser identificado, a rede elétrica.

2.6.1 Sensor SCT-013-000

Os sensores do tipo *Split-core Current Transformer* (*SCT*) utilizam os princípios do eletromagnetismo para medir a corrente elétrica. Especificamente, o sensor *SCT-013-000* (*YHDC*, *Beijing*, China) é constituído de dois núcleos magnéticos e uma bobina de 2000 espiras. Além disso, diferente de outros sensores que utilizam os mesmos princípios, seu núcleo magnético é dividido dando a vantagem de ser não intrusivo, ou seja, não é necessário interferir no sistema elétrico que será monitorado.

O funcionamento desse sensor parte da Lei de Ampère, onde é declarado que um fio condutor de corrente induz um campo magnético ao seu redor proporcional a essa corrente. Neste caso, por ser uma corrente alternada, o campo magnético gerado por essa corrente é variante no tempo (DIAS; DIAS, 2015). Dito isso, pode ser utilizado outro princípio físico, a Lei de Faraday, onde é definido que um campo magnético variante no tempo induz uma força eletromotriz em uma espira que, gera uma corrente elétrica proporcional à intensidade do campo magnético (HESSEL *et al.*, 2015).

Considerando os princípios utilizados pelo sensor, a primeira metade do núcleo magnético possui apenas uma espira formado pelo fio onde o sensor será acoplado, já a segunda metade do núcleo contém uma bobina de 2000 espiras. Essa estrutura pode ser melhor entendida visualizando a Figura 5. Seguindo essa lógica, o valor de corrente capturado pelo sensor é inversamente proporcional às 2000 espiras da bobina secundária e obedece equação 2.1.

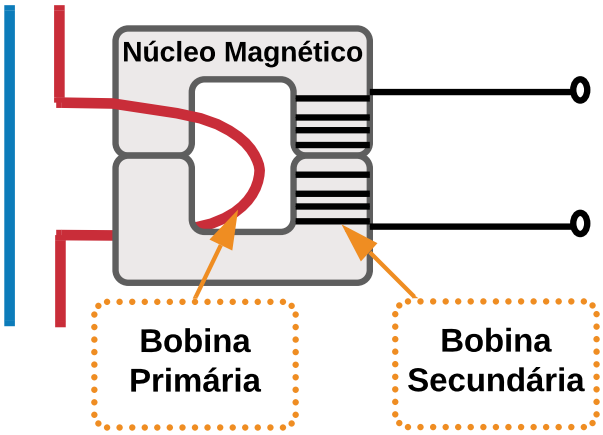
$$\frac{N_1}{N_2} = \frac{I_2}{I_1} \tag{2.1}$$

Onde N_1 é o número de espiras na bobina primária, ou seja, uma espira apenas e N_2 é o número de espiras na bobina secundária, ou seja, 2000 voltas. Já I_1 é o valor de corrente no fio e I_2 é o valor capturado pelo sensor. O sensor *SCT-013-000* foi projetado para medir um valor máximo de corrente de 100 A RMS. Nesse contexto, o valor máximo informado pelo sensor é de 50 mA RMS.

2.7 Conversores Analógicos-Digitais

Os conversores analógicos-digitais ou *Analog-to-Digital Converter* (*ADC*) possuem o propósito de converter os sinais de tempo contínuo em tempo discreto, mais especificamente, na forma de codificação binária. Normalmente, são utilizados para converter dados analógicos

Figura 5 – Arquitetura do sensor *SCT-013-000*



Fonte: Elaborada pelo próprio autor

provenientes de sensores, de modo a serem lidos por um microcontrolador. Vale ressaltar que, a aquisição de dados digitais é uma função essencial de qualquer instrumento analítico (ITTERHEIMOVÁ *et al.*, 2021). Logo, o sinal analógico (normalmente representado por uma tensão ou corrente elétrica) do sensor é primeiro processado em um *ADC* e convertido em valores digitais. Assim, os dados podem ser repassados para os dispositivos responsáveis pelo processamento e armazenamento dos dados.

Ainda, na construção de um sistema de aquisição de dados é fundamental considerar a resolução do *ADC*, bem como a faixa de tensões permitida e a frequência de amostragem (ITTERHEIMOVÁ *et al.*, 2021). Esses parâmetros são essenciais para definir a precisão do conversor e se ele será suficiente para o projeto em que está inserido. A resolução determina o número de níveis de voltagem que podem ser produzidos ao longo da faixa de valores analógicos permitida, em outras palavras, a resolução designa o número de valores discretos ao longo de uma faixa contínua. Assim, considerando um *ADC* de 10 *bits* de resolução, o número de níveis discretos é 1024, enquanto para um *ADC* de 16 *bits* há 65.536 níveis possíveis. Então, para uma faixa de medição de $\pm$ 4 V os níveis de tensão de um *ADC* de 10 *bits* e 16 *bits* variam em cerca de 8 mV e 122 μV respectivamente.

Neste trabalho, o *ADC ADS1115* é empregado de modo a converter para digital os dados analógicos oriundos do sensor de corrente. Dessa forma, o microcontrolador se comunica com o *ADC* para realizar a leitura dos dados, fazer o pré-processamento e enviar as amostras com destino ao nó de Borda. Assim, o *middleware*, via *FogFlow*, pode seguir com o processamento e

identificação do aparelho elétrico.

2.7.1 Conversor ADS1115

O *ADS1115* converte sinais analógicos em dados digitais. Este conversor usa o protocolo de comunicação *I2C* através dos pinos *Serial Data* (*SDA*) e *Serial Clock* (*SCL*) que controlam o fluxo de dados e a frequência de aquisição respectivamente (SALIM *et al.*, 2016). Ademais, esse conversor disponibiliza 4 pinos para leitura analógica, além de possuir 16 *bits* de resolução podendo ler dados tanto com sinal quanto sem sinal e realizar conversões em taxas de dados de até 860 amostras por segundo ou *Samples per Second* (*SPS*) (GOWTHAM *et al.*, 2020).

Ainda, o conversor *ADS1115* possui amplificador de ganho programável ou *Programmable Gain Amplifier* (*PGA*) integrado, que pode ser controlado dinamicamente, permitindo a medição de sinais grandes e pequenos com alta resolução (SAADATZI *et al.*, 2016). Nesse contexto, o *ADS1115* é ideal para ler diferentes sinais que podem variar bastante dependendo do equipamento monitorado.

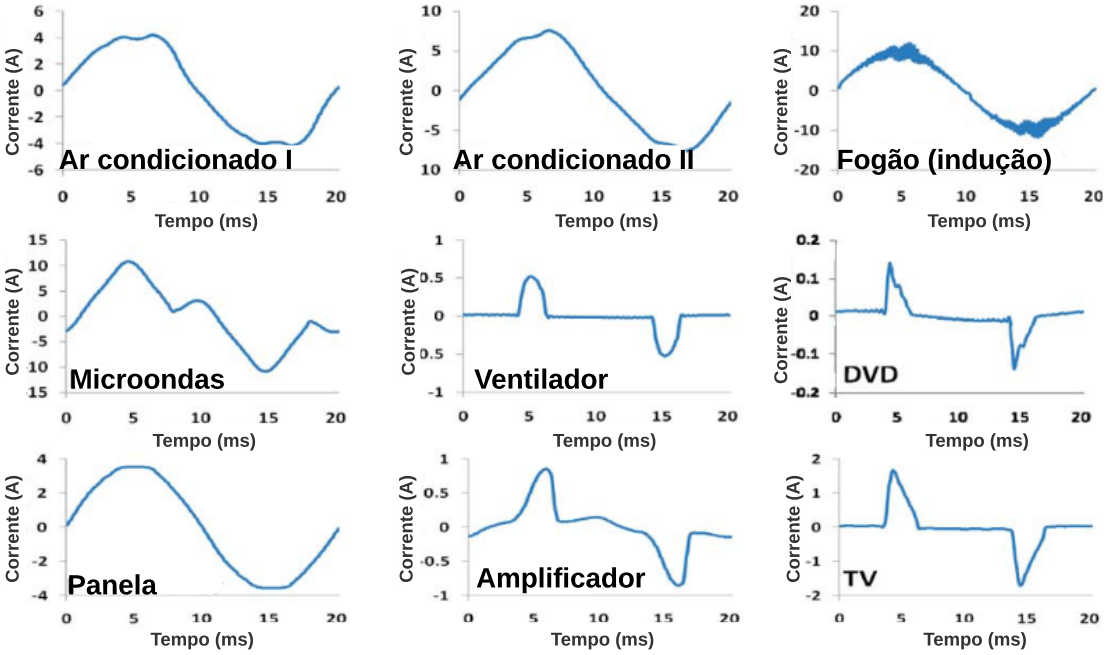
2.8 Assinatura de Carga

Segundo Liang *et al.* (2009), o conceito de assinatura de carga é definido como um atributo específico e mensurável no padrão de consumo de um aparelho elétrico ou peça de um equipamento individual. Nessa óptica, as características exclusivas observadas normalmente incluem as variáveis de corrente elétrica, tensão e potência, onde podem ser avaliados os valores de pico, variância, média, formato de onda, entre outros (LIANG *et al.*, 2009).

A corrente elétrica pode ser definida como o movimento ordenado de portadores de cargas ou elétrons (FEITOSA *et al.*, 2021). Em outras palavras, a corrente é expressa como o número de elétrons por unidade de tempo passando através de uma seção condutora de eletricidade. Além disso, tem sua unidade de medida calculada em Ampère (A), conforme o Sistema Internacional (SI). A Figura 6 mostra o sinal de corrente de vários equipamentos elétricos em funcionamento, onde é possível perceber os diferentes valores e formatos de onda.

Como é possível perceber na Figura 6, há vários padrões de sinal, todavia, equipamentos do mesmo modelo tende a gerar ondas de consumo bastante semelhantes, como constatado nos gráficos de corrente dos aparelhos de ar condicionado I e II, que variam os

Figura 6 – Sinal de corrente de diferentes aparelhos.



Fonte: Adaptado de Liang *et al.* (2009)

valores de consumo, mas preservam o formato da onda. Tendo em vista essas características, é possível identificar cada um dos aparelhos, considerando um certo grau de precisão. Além disso, equipamentos da mesma classe como a *TV* e o *DVD* que possuem apenas circuitos eletrônicos na grande parte de sua composição, tendem a ter características rapidamente parecidas.

Nesse contexto, os conceitos de assinatura de carga são fundamentais para este trabalho, considerando a identificação dos equipamentos elétricos através do sinal de consumo. Na solução aqui proposta as assinaturas são extraídas de amostras do sinal de corrente, com isso, atributos como pico e formato de onda são utilizados no treinamento do algoritmo de classificação e posteriormente na identificação do aparelho.

2.9 Aprendizado de Máquina

Embora oculta para alguns, o Aprendizado de Máquina se tornou fundamental para a evolução da tecnologia da informação. Com a alta disponibilidade dos dados, a análise inteligente das informações está se tornando ainda mais difundida como algo necessário e indispensável para o progresso da tecnologia (OSISANWO *et al.*, 2017). Ainda, segundo os autores Osisanwo *et al.* (2017), o Aprendizado de Máquina é uma das áreas da Ciência da Computação que mais cresce, com algoritmos inovadores. As ferramentas de aprendizado visam constituir programas com a capacidade de aprender e se adaptar.

Tendo em vista que a identificação de aparelhos elétricos é um dos principais objetivos deste trabalho, as técnicas de aprendizado voltadas a classificação se mostram bastante úteis para esse fim. Logo, o Aprendizado de Máquina, mais especificamente, os algoritmos de classificação são fundamentais para identificar de forma eficiente os aparelhos por meio das amostras coletadas. Além disso, a adição de novos equipamentos à rede elétrica não será um problema, dado a alta capacidade de adaptação dessas técnicas.

2.9.1 Algoritmos de Classificação

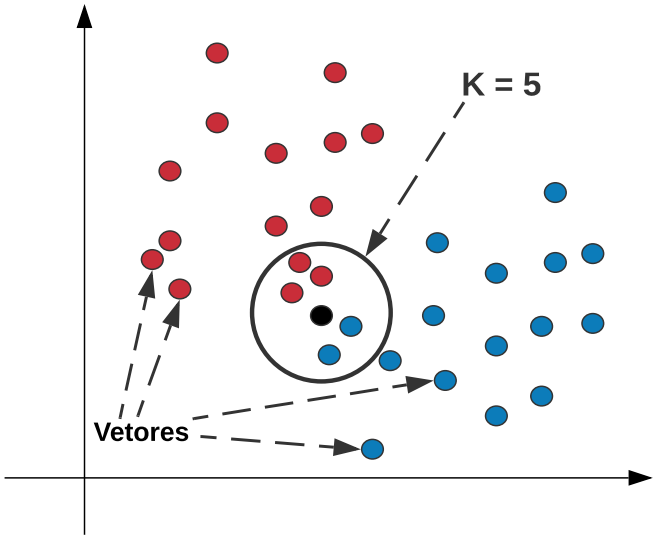
Os autores Zhang *et al.* (2017) relatam que a classificação é um dos tópicos de pesquisa mais importantes em mineração de dados, ainda, que a principal tarefa da classificação é prever os rótulos dos pontos de dados de teste inserindo todos os pontos de treinamento. Vale ressaltar que, nas últimas décadas, muitos métodos de classificação foram desenvolvidos em aplicações reais, em especial, técnicas de aprendizado supervisionado como, *K-Nearest Neighbor* (*KNN*), *Support Vector Machine* (*SVM*), *Decision Tree*, *Naive Bayes*, *Random Forest*, *Gradient Boosting*, entre outros, em que o modelo é treinado a partir de resultados pré-definidos (SINGH *et al.*, 2016). Cada um com suas vantagens, parâmetros e adequado a diferentes contextos. A seguir é detalhado funcionamento e as características de alguns desses algoritmos.

2.9.1.1 KNN

O *K-Nearest Neighbors*, ou somente *KNN*, é um algoritmo de aprendizado supervisionado onde a variável *K* significa uma quantidade. A metodologia de trabalho do *KNN* é bastante simples, onde um vetor é classificado observando as classes dos *Ks* vizinhos mais próximos dele. Desta forma, a classe de um vetor é definida como a moda das classes dos *Ks* vetores mais próximos (DEY *et al.*, 2018). Na Figura 7 pode ser visualizada a ideia do algoritmo *KNN*.

Como pode ser notado na Figura 7, há vetores de amostras dispersas em um plano, alguns da classe ilustrada na cor vermelha e outros da classe azul. Neste caso, o objetivo do algoritmo é classificar o vetor de cor preta que está no meio do círculo. Este círculo busca envolver os *K* vizinhos mais próximos do vetor que será classificado, neste caso o *K* possui valor 5. No algoritmo, os vizinhos são selecionados calculando a distância entre o vetor a ser classificado e os demais vetores. O modelo de como é calculado a distância é um parâmetro do algoritmo, podendo ser alterado. Diante disso, é constatado que a maioria dos *Ks* vizinhos selecionados são da classe vermelha, logo, o vetor preto é da classe vermelha segundo o algoritmo.

Figura 7 – Técnica de classificação do *KNN*



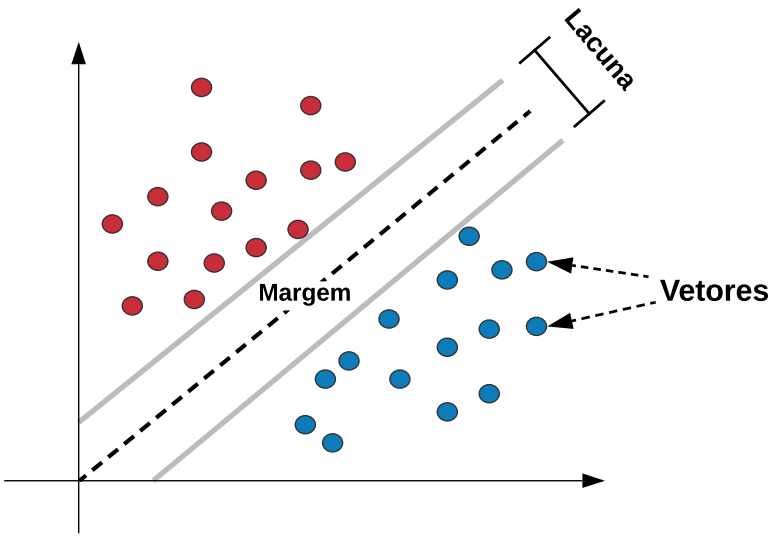
Fonte: Elaborada pelo próprio autor

2.9.1.2 SVM

A *Support Vector Machine*, ou apenas *SVM*, é uma das técnicas de aprendizado supervisionado utilizada para classificação, embora também seja utilizada para regressão em menor quantidade. O principal objetivo desse algoritmo está focado na classificação linear, porém, é possível executar com bastante eficiência uma classificação não linear, mapeando implicitamente suas entradas em espaços de recursos de alta dimensão, o que é chamado truque do *kernel* (MAHESH, 2020). A ideia central do *SVM* é desenhar margens entre as classes, onde tais margens são traçadas de forma que a distância entre a margem e as classes seja máxima, ou seja, a margem entre dois rótulos é traçada totalmente centralizada para que a classificação não seja tendenciosa para nenhum dos rótulos, minimizando os erros que eventualmente possam ocorrer. Na Figura 8 a ideia do *SVM* é exemplificada.

Como visto na Figura 8, vetores pertencentes a duas classes diferentes estão em um plano, nas cores vermelho e azul. Então, é definida a margem que separa às duas classes de modo que a lacuna entre a reta que define a margem e os vetores tenha tamanho máximo. Todavia, em um cenário real podem haver várias classes, o que implica mais do que apenas duas dimensões. Neste caso, a mesma ideia é aplicada selecionando hiperplanos com a distância máxima possível das classes (DEY *et al.*, 2018).

Figura 8 – Técnica de classificação do *SVM*



Fonte: Elaborada pelo próprio autor

2.9.1.3 Naive Bayes

Em Aprendizado de Máquina, algoritmos *Naive Bayes* são uma família de classificadores probabilísticos simples, baseados na aplicação do teorema de Bayes. A característica principal desta técnica é a suposição de independência forte entre os recursos, o que atribui o aspecto de ingênuo ao classificador (GRANIK; MESYURA, 2017). Desta forma, todos os classificadores *Naive Bayes* assumem que o valor de uma determinada *feature* é independente do valor de qualquer outra *feature* existente no conjunto de dados, dada a variável de classe.

O algoritmo *Naive Bayes* é um modelo de aprendizado bastante rápido além de ser eficiente. Comumente, esse modelo é empregado para classificar textos como, por exemplo, identificar se o *e-mail* é *spam* ou não (DEY *et al.*, 2018). O funcionamento do *Naive Bayes* é bem fácil de entender considerando que, $P(A|B)$ é a probabilidade do evento A ocorrer dado que o evento B ocorreu e, consequentemente, $P(B|A)$ é a probabilidade da ação B ocorrer dado que a ação A ocorreu, sendo que $P(A)$ e $P(B)$ são as probabilidades de ocorrência dos eventos A e B respectivamente, o teorema de Bayes pode ser descrito matematicamente, como mostrado na equação 2.2.

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{2.2}$$

2.9.1.4 Random Forest

O *Random Forest* é um método de aprendizagem integrado baseado no algoritmo de *Bagging*, que pode ser usado para classificação e regressão (GUO *et al.*, 2019). Esse método consiste em uma série de preditores de árvore em que cada árvore é baseada nos valores de um vetor amostrado aleatoriamente com a mesma distribuição em todas as árvores de decisão pertencente à floresta. Se o número de árvores em uma floresta aumenta, o erro de generalização converge para um limite (PRIHATNO *et al.*, 2021). O número de árvores é definido pelo usuário através de um hiper parâmetro do modelo.

2.9.1.5 Gradient Boosting

As técnicas de *boosting* foram desenvolvidas pela comunidade de Aprendizado de Máquina, inicialmente, direcionadas para problemas de classificação. A ideia central desses algoritmos é combinar iterativamente vários modelos simples, chamados aprendizes fracos (*weak learners*), de modo a obter um aprendiz forte (*strong learner*) com maior precisão de predição (TOUZANI *et al.*, 2018). Posteriormente, o *boosting* foi estendido à regressão, com a introdução do método de *Gradient Boosting Machine* (*GBM*).

O *GBM* é um método de Aprendizado de Máquina poderoso para construir modelos preditivos. Sua ideia básica é minimizar a função de perda do modelo adicionando uma nova árvore de decisão (aprendiz fraco) para compensar as deficiências dos aprendizes fracos existentes. Em resumo, o algoritmo começa inicializando o modelo com uma primeira escolha de resultado, sendo geralmente alcançado a partir de uma árvore de decisão que reduz ao máximo a função de perda, então, a cada nova iteração, uma nova árvore de decisão é ajustada e adicionada ao modelo anterior como objetivo de reduzir o erro (YAN; WEN, 2021). O número de iterações é designado pelo usuário. Vale ressaltar que, as iterações não modificam as árvores adicionadas nas etapas anteriores. Além disso, ao ajustar as árvores de decisão aos resíduos, o modelo é aprimorado nas regiões onde não tem um bom desempenho (TOUZANI *et al.*, 2018).

2.9.2 Métricas para Técnicas de Classificação

Dentre as técnicas de classificação, não há um algoritmo ideal. Contudo, o objetivo desses algoritmos é se aproximar ao máximo do ideal, ou seja, buscam acertar o máximo de predições. Dito isso, algumas métricas de avaliação foram definidas para ponderar o desempenho

da classificação. Normalmente, as métricas são baseadas em uma matriz de confusão que resume as informações sobre as classificações reais e previstas realizadas por um classificador (FAYZRAKHMANOV *et al.*, 2018). Um modelo de matriz de confusão para algoritmos de classificação binários, ou seja, que possuem apenas duas classes é mostrada no Quadro 2.

Quadro 2 – Matriz de confusão binária

Real	Predito	
	Classe Positiva	Classe Negativa
Classe Positiva	Verdadeiro Positivo (VP)	Falso Negativo (FN)
Classe Negativa	Falso Positivo (FP)	Verdadeiro Negativo (VN)

Fonte: Adaptado de Fayzrakhmanov *et al.* (2018)

Como pode ser visto, o quadro mostra os quatro possíveis resultados de previsão. Os resultados corretos são os que contém as respostas Verdadeiro Positivo (VP) e Verdadeiro Negativo (VN), ou seja, nesses dois casos o algoritmo acerta em predizer que o vetor é da classe positiva e da classe negativa respectivamente. Entretanto, a resposta Falso Negativo (FN) corresponde a um vetor que é de fato positivo, porém foi incorretamente classificada como negativo. Semelhantemente, a resposta Falso Positivo (FP) corresponde a um vetor que é de fato negativo, mas que foi incorretamente classificado como positivo (CHICCO; JURMAN, 2020). Neste trabalho, são empregadas algumas métricas conhecidas embasadas na matriz de confusão, para avaliar a classificação dos aparelhos como acurácia, precisão, *recall* e *F1 score*.

2.9.2.1 Acurácia

A acurácia é considerada por muitos pesquisadores como o caminho padrão para avaliar o desempenho de um classificador, todavia, não é tão precisa quando o conjunto de dados está desbalanceado por favorecer classes que aparecem com mais frequência (CHICCO; JURMAN, 2020). Essa métrica descreve a proporção entre as instâncias corretamente classificadas, não fazendo distinção entre positivos e negativos, e todas as instâncias no conjunto de dados (ALMEIDA *et al.*, 2018). A acurácia varia entre 0 e 1, quanto maior o valor, melhor é a acurácia. O cálculo é exemplificado na equação 2.3.

$$\text{Acurácia} = \frac{VP+VN}{VP+VN+FP+FN} \tag{2.3}$$

2.9.2.2 Precisão

A precisão é definida por meio da razão entre instâncias verdadeiramente positivas e todas as instâncias positivas (PHILIPPOU *et al.*, 2020). Dessa forma, essa métrica tem como intuito valorizar a captura de erros por falso positivo. Assim como a acurácia, a precisão varia de 0 a 1, sendo o valor 1 considerado a melhor precisão. A fórmula é mostrada na equação 2.4.

$$\text{Precisão} = \frac{VP}{VP + FP} \tag{2.4}$$

2.9.2.3 Recall

A métrica *recall* é definida por meio da razão entre as instâncias verdadeiramente positivas e a soma do número de instâncias verdadeiramente positivas e falsamente negativas (PHILIPPOU *et al.*, 2020). De modo contrário a métrica de precisão, a *recall* tem como intensão valorizar a captura de erros por falso negativo. Semelhantemente, essa métrica varia de 0 a 1, sendo o valor 1 considerado melhor. A fórmula é mostrada na equação 2.5.

$$\textit{Recall} = \frac{VP}{VP + FN} \tag{2.5}$$

2.9.2.4 F1 Score

A métrica *F1 score* é definida a partir dos cálculos de precisão e *recall* sendo a média harmônica entre ambas, nesse raciocínio, é uma das métricas mais utilizadas por combinar as vantagens de capturar erros por falso positivo e falso negativo (CHICCO; JURMAN, 2020). Assim como as demais métricas, a *F1 score* varia de 0 a 1, sendo 1 o melhor valor de *F1 score* e 0 o pior. O cálculo matemático dessa métrica é mostrado na equação 2.6.

$$\textit{F1 Score} = 2 \cdot \frac{\text{Precisão} \cdot \text{Recall}}{\text{Precisão} + \text{Recall}} \tag{2.6}$$

3 TRABALHOS RELACIONADOS

Na literatura, podem ser encontrados vários trabalhos que se relacionam tanto no que diz respeito ao uso de ferramentas e tecnologias *IoT* quanto na utilização de técnicas de Aprendizado de Máquina aplicadas na identificação de padrões, mais precisamente, na classificação de dispositivos. Dessa forma, este capítulo apresenta, de modo detalhado, trabalhos compatíveis com essas duas vertentes, onde são, principalmente, explorados conceitos e técnicas implementadas neste trabalho.

3.1 Aprendizado de Máquina

Nesta seção são encontrados trabalhos selecionados com base no emprego das técnicas de Aprendizado de Máquina para identificar equipamentos elétricos. Logo, pode ser observado quais algoritmos foram aplicados, quais *features* foram selecionadas para a classificação, além da acurácia obtida.

3.1.1 Real time identification of electrical devices through power consumption pattern detection (ABEYKOON et al., 2016)

No trabalho de Abeykoon *et al.* (2016) é tratada a identificação de dispositivos elétricos por meio da detecção de padrões utilizando algoritmos de Aprendizado de Máquina, além de conceitos mais específicos como Redes Neurais Artificiais (RNAs). Para esse fim, a linguagem *Python* é empregada em um ambiente *Raspberry Pi* com o uso do *Linux*. Antes de tudo, são capturados parâmetros do sinal elétrico como potência ativa, potência reativa, mudança de fase e média quadrática da voltagem e da corrente. Depois, é realizada a análise dos dados para classificar o equipamento elétrico.

Para isso, são utilizadas técnicas de aprendizado supervisionado como *SVM* e RNAs, além de técnicas não supervisionadas como *K-Means*, *Silhouette* e *Mean-Shift*. Dito isso, são realizados testes onde é possível fazer a comparação das técnicas empregadas observando a acurácia resultante. No fim, os algoritmos utilizados retornaram resultados bem semelhantes, com acurácias que variam entre 94% e 98% conforme o algoritmo empregado. No entanto, os autores não focam em relatar o processo de aquisição dos dados, nem na arquitetura de comunicação para o transporte das amostras.

Este trabalho segue a ideia de aplicações de algoritmos de aprendizado supervisio-

nado, como o *SVM*, focando em variáveis que podem ser obtidas a partir do sinal de corrente elétrica durante o consumo. Ainda, a solução aqui proposta utiliza um ambiente *Raspberry Pi* como núcleo de processamento dedicado a execução dos algoritmos. Contudo, possui o adicional de conceitos de Computação de Borda aplicados ao ambiente *Raspberry Pi* que facilita a comunicação e integra o nó ao ambiente de *CoT*, além de ser amplamente voltada a IoT utilizando de *middlewares* e protocolos rápidos para o transporte das amostras.

3.1.2 Non-Intrusive Electrical Appliances Monitoring and Classification using K-Nearest Neighbors (KHAN et al., 2019)

Os autores Khan *et al.* (2019) destacam a necessidade de um monitoramento do uso de energia de um aparelho específico, visando mudanças no seu comportamento de consumo que facilitam a conservação de energia, dado que os medidores de eletricidade existentes fornecem pouca ou nenhuma informação sobre os consumos energéticos de aparelhos individuais. Para tanto, é empregado o método *NILM* na aquisição de dados e aplicado o algoritmo de Aprendizado de Máquina *KNN* que realiza a classificação do dispositivo observando os *Ks* vizinhos mais próximos, nesse trabalho foi utilizado um k = 5.

Na avaliação é usado o conjunto de dados *Reference Energy Disaggregation Data Set* (*REDD*), separando 90% dos dados para treino do algoritmo e 10% para testes. Por fim, é apresentado os resultados da classificação de 7 dispositivos. Embora o algoritmo tenha confundido as cargas eletrônicas retornando uma acurácia de 50%, nos demais casos teve um bom desempenho com acurácias entre 90% e 100%. Todavia, uma das limitações do projeto realizado pelos autores é a não avaliação de outros algoritmos que poderiam obter resultados melhores quando feita a comparação das acurácias de cada técnica. Ainda, os autores relatam apenas um valor para o hiper parâmetro k, mas não justificam essa escolha.

Este trabalho, utiliza o algoritmo *KNN* para classificar as amostras do sinal de corrente e identificar o aparelho responsável pelo consumo. Porém, também utiliza outras técnicas e realiza a comparação das métricas para determinar a mais adequada para o contexto de classificação utilizado. No entanto, vale ressaltar que nos testes são avaliadas diferentes categorias de equipamentos com propriedades distintas que também são consideradas neste trabalho. Além disso, será seguido o método de monitoramento não intrusivo para facilitar a aquisição das amostras em circuitos elaborados antes deste projeto.

3.1.3 Non-Intrusive Load Identification for Smart Outlets (BARKER et al., 2014)

Os autores Barker *et al.* (2014) realizam um projeto de tomada inteligente que identifica o tipo e qual dispositivo está conectado a ela. Para tanto, é utilizado o método *NILM* na aquisição de amostras do sinal que permite uma implementação mais fácil do projeto, por não ser intrusivo. Então, são observados parâmetros como formato da onda, variância e média da potência e *duty cycle* do sinal capturado. Assim, esses parâmetros servem de *features* para os algorítimos de Aprendizado de Máquina utilizados na classificação dos dispositivos.

Ainda, os autores realizam um comparativo entre três algoritmos de classificação, são eles: *Naive Bayes*, *SVC* e *Decision Tree*. Cada técnica é avaliada quanto sua capacidade de classificar corretamente o equipamento elétrico, além do tipo do dispositivo. No fim, são apresentados os resultados de testes realizados com inúmeros dispositivos elétricos em que é mostrado o resultado da classificação tanto do dispositivo específico quanto do tipo. Por fim, é constatada uma acurácia superior a 90%.

Uma das principais vantagens do trabalho realizado pelos autores é a utilização de vários algoritmos, podendo perceber qual melhor se comporta no contexto empregado, ideia que será empregada neste trabalho. Ainda, vale ressaltar que o projeto identifica tanto o equipamento em funcionamento quanto o tipo ao qual pertence, além de mostrar os resultados de vários aparelhos testados com os três algoritmos empregados, ideia de teste também seguida neste trabalho. Ainda, utiliza o método *NILM* em concordância com a solução aqui proposta. Todavia, não enfatiza o processo de transmissão de dados entre as diferentes partes do projeto, além de não apresentar o dispositivo responsável pela captura das amostras.

3.2 *Internet* das Coisas

Nesta seção são abordados trabalhos selecionados segundo o uso de conceitos e ferramentas *IoT*, maneira em que os dados de contexto são gerenciados e de que modo é realizada a comunicação entre as diferentes partes do projeto.

3.2.1 Energy Efficiency in Smart Buildings: An IoT-Based Air Conditioning Control System (ROCHA et al., 2019)

Os autores Rocha *et al.* (2019) descrevem uma ideia de construção inteligente que otimiza o funcionamento do aparelho de ar-condicionado visando reduzir o consumo de energia.

O projeto conta com um sensor de temperatura e umidade, além de uma câmera e um sensor de presença que faz a contagem de pessoas em uma determinada sala. Essas informações servem de parâmetros para o controle, que utiliza um *LED* infravermelho responsável por enviar os comandos ao aparelho de ar-condicionado. O projeto ainda aplica uma arquitetura de *master* e *slave*, onde é utilizado um microcomputador *Raspberry Pi* como *master* e placas microcontroladoras *ESP NodeMCU* como *slave*. Por fim, os dados capturados pelos sensores e o estado do aparelho de ar-condicionado são apresentados em uma aplicação *web*.

Vale ressaltar, que os autores utilizam o *middleware FIWARE* para realizar o armazenamento e gerenciar os dados de contexto. Mais precisamente, é usado o *Enabler Orion Context Broker* para o gerenciamento dos dados atuais e o envio de comandos, além do *Enabler Cygnus* responsável pela comunicação com a base de dados onde é armazenado o histórico de informações. Ainda, é empregado o protocolo *MQTT*, por meio do *broker Mosquitto*, na comunicação do *hardware* com o *FIWARE*. Para isso, é feita a utilização do *Enabler IDAS* que possui os chamados *IoT Agents* encarregados de traduzir diferentes protocolos para o padrão *NGSI* entendido pelos *Enablers* do *FIWARE*, neste caso, é utilizado um *IoT Agent* que traduz o protocolo *MQTT*.

Contudo, os autores utilizam o microcomputador *Raspberry Pi* em todas as salas monitoradas, visto que, a câmera que realiza a contagem de pessoas pertence ao *Raspberry Pi*. No entanto, isto traz um alto custo financeiro para monitorar cada sala. Dito isso, a solução proposta neste trabalho visa o baixo custo ao adicionar novos pontos de monitoramento, utilizando um *Raspberry Pi* como nó de Borda para vários dispositivos *IoT*.

Todavia, este trabalho também usa o *middleware FIWARE* com características de gerenciamento de dados de contexto semelhantes, utilizando o *GE IDAS* e o *GE Orion*. Além disso, utiliza o protocolo *MQTT* juntamento com o *broker Mosquitto* para transporte de dados entre os dispositivos *IoT* e o *FIWARE*. Vale ressaltar que, os autores não objetivam a identificação do dispositivo, apenas um controle inteligente do aparelho de ar-condicionado, tendo como principal contribuição a utilização de ferramentas *IoT* e a arquitetura de comunicação entre as diferentes partes do projeto.

3.2.2 An Energy Management Platform for Public Buildings (FERREIRA et al., 2018)

O trabalho dos autores Ferreira *et al.* (2018) tem como ideia geral a utilização de sensores em prédios inteligentes que capturam dados de consumo para o armazenamento *online*

de modo a gerar soluções de economia de energia elétrica através da análise dos dados armazenados. Uma das justificativas do texto é que os sistemas de gerenciamento de consumo energético dos edifícios são bastante básicos e a quantidade dos dados disponíveis é pequena, nesse sentido, o mau gerenciamento de energia em grandes construções tem impactos significativos na natureza, além de está relacionado com custos desnecessários.

O projeto desenvolvido no trabalho emprega o modelo *NILM* para aquisição de amostras do sinal de corrente elétrica. Para isso, é utilizado o sensor não intrusivo *SCT-013-000*, que captura amostras de corrente aplicando os princípios do eletromagnetismo. As amostras são enviadas para uma placa *Arduino Uno*. Ainda, são capturados os dados de temperatura, umidade e luminosidade por um sensor conectado a um microcomputador *Raspberry Pi*. Por fim, todos os dados capturados são enviados, via rede *Long Range* (*LoRa*), para as ferramentas *Bluemix* e *Node-RED* que prestam serviços de Nuvem voltados a *IoT*, onde as informações são analisadas.

Além disso, é empregado o conceito de *Gamification* em que são aplicadas as ideias de jogos e pontuações visando conseguir mudanças no comportamento das pessoas em prol da redução do consumo de energia. Por fim, é mostrado os resultados de um experimento realizado em um *campus* universitário onde foi descoberto que as lâmpadas eram acionadas de maneira desnecessária 40% do tempo. Assim, foi obtida uma redução do consumo elétrico com a correção dessa irregularidade.

A principal contribuição dos autores para esse trabalho é no processo de monitoramento do sinal elétrico, em que os autores utilizam o modelo *NILM* além do sensor *SCT-013-000* também utilizado no projeto deste trabalho. Ainda, os autores enfatizam a aplicação de conceitos e ferramentas *IoT* com comunicação sem fio no gerenciamento dos dados. Todavia, não é buscada a identificação dos aparelhos pelo sinal elétrico, apenas a detecção de irregularidades no sinal de consumo. Para identificar quais equipamentos estão em funcionamento, são utilizados os sensores, porém, isso limita consideravelmente quais aparelhos podem ser identificados.

3.3 Comparação Entre Trabalhos

No Quadro 3 são listados os trabalhos relacionados além do trabalho aqui proposto. Assim, os trabalhos são comparados em relação aos quesitos de objetivo, métodos de monitoramento, plataformas *IoT* utilizadas, método de classificação, métricas, conjunto de dados e acurácia dos resultados. Como pode ser observado, os trabalhos são amplamente voltados ao monitoramento de energia elétrica, como exceção dos autores Rocha *et al.* (2019) que direcionam

o trabalho ao controle de funcionamento do aparelho de ar condicionado. Todavia, todos buscam a redução dos gastos energéticos para diminuir os custos e os riscos com a natureza.

Quadro 3 – Comparacação entre os trabalhos

Trabalho	Abeykoon *et al.* (2016)	Khan *et al.* (2019)	Barker *et al.* (2014)	Rocha *et al.* (2019)	Ferreira *et al.* (2018)	Este trabalho
Objetivo	Monitoramento energético	Monitoramento energético	Monitoramento energético	Controle do ar-condicionado	Monitoramento energético	Monitoramento energético
Método de Monitoramento	Não relatado	*NILM*	*NILM*	Não se aplica	*NILM*	*NILM*
Plataformas *IoT*	Não relatado	Não relatado	Não relatado	*FIWARE* (*Orion*, *IDAS* e *Cygnus*)	*Bluemix* e *Node-RED*	*FIWARE* (*Orion*, *IDAS* e *FogFlow*)
Métodos de Classificação	*SVM*, RNA, *Mean Shift*, *Silhouette* e *K-Means*	*KNN*	*Naive Bayes*, *Decision Tree* e *SVM*	Não se aplica	Sensores	*KNN*, *Logistic Regression*, *SVM*, *Naive Bayes*, *Decision Tree*, *Random Forest*, *Gradient Boosting* e RNA
Métricas	Acurácia	Acurácia, Precisão, *Recall* e *F1 score*	Acurácia	Não se aplica	Não relatado	Acurácia e *F1 score*
Conjunto de Dados	Próprio	*REDD*	Próprio	Não se aplica	Não se aplica	Próprio
Acurácia	94% à 98%	90% à 100%	90%	Não se aplica	Não relatado	90% à 100%

Fonte: Elaborado pelo próprio autor.

No que diz respeito ao método de monitoramento, a maioria utilizou o *NILM*, com exceção dos autores Abeykoon *et al.* (2016), que apesar de o monitoramento energético ser um dos pontos forte do trabalho, não relatam o modelo de aquisição das amostras. Além dos autores Rocha *et al.* (2019), que não visam o monitoramento do sinal elétrico. Vale ressaltar que este trabalho utiliza o modelo *NILM* na captura de dados e, assim como os demais trabalhos que também utilizam esse modelo, busca a facilidade na implantação, não interferindo nos circuitos que já estão em funcionamento.

No que se refere a utilização de plataformas *IoT*, apenas os autores Rocha *et al.* (2019) e Ferreira *et al.* (2018) se destacam e relatam o uso de *middlewares* para realizar a comunicação entre as partes do projeto e o gerenciamento dos dados de contexto. Assim como os autores Rocha *et al.* (2019), este trabalho utiliza a plataforma *FIWARE* e seus *GEs* para montar a arquitetura de camadas *IoT*, devido a sua adaptação a ambientes inteligentes e disponibilização de diversos serviços voltados ao gerenciamento de dados de contexto. Os demais trabalhos não enfatizam a utilização dessas ferramentas, além de não relatarem como foi realizada a comunicação entre as partes, focando apenas nos processos de monitoramento e classificação.

Ainda, nos métodos de classificação, grande parte dos autores treinaram algoritmos

de Aprendizado de Máquina e alguns utilizaram Aprendizagem Profunda em específico. De maneira isolada, os autores Ferreira *et al.* (2018) utilizaram diferentes sensores para identificar quais aparelhos são responsáveis pelo consumo detectado. Todavia, os autores Rocha *et al.* (2019) não direcionam o trabalho para a identificação dos dispositivos em especial. Como a maioria, este trabalho utiliza algoritmos de Aprendizado de Máquina por entender que técnicas voltadas a treinamento se adaptam melhor na identificação de diferentes dispositivos, além disso, não é necessário acrescentar outras categorias de sensores para a identificação de novos aparelhos adicionados à rede elétrica, bastando apenas um novo treinamento do algoritmo.

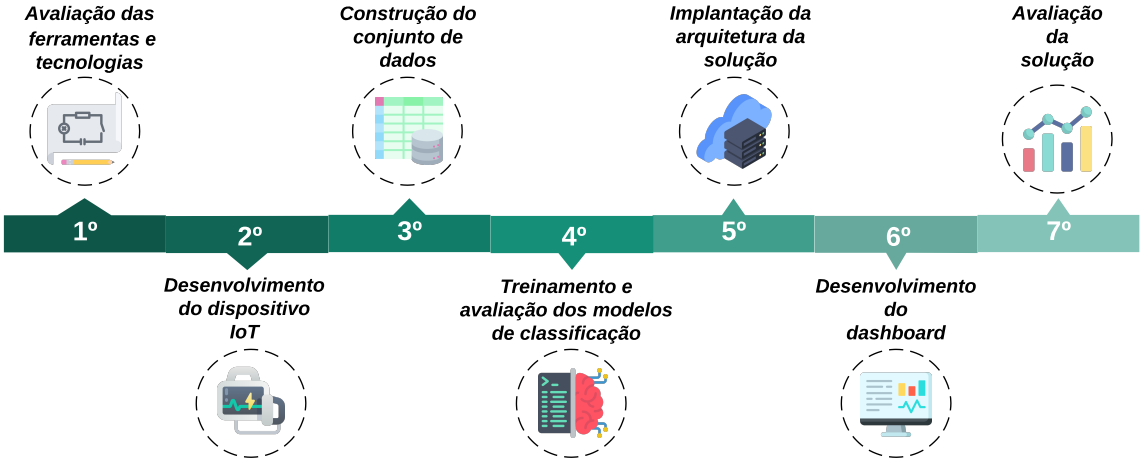
Com relação às métricas utilizadas na avaliação dos resultados, a maioria dos autores utilizou a acurácia, todavia esse quesito não se aplica nos trabalhos dos autores Rocha *et al.* (2019) e não é relatado pelos autores Ferreira *et al.* (2018). Vale ressaltar que, os autores Khan *et al.* (2019) também utilizaram a métrica *F1 score* que depende das métricas de Precisão e *Recall*. Baseados nos trabalhos relacionados, este trabalho utiliza as métricas de acurácia e *F1 score*. Ademais, é importante destacar que a maior parte dos projetos, incluindo este, utilizaram um conjunto de dados próprio. Em contrapartida, os autores Khan *et al.* (2019) utilizaram o conjunto de dados *REDD* (KOLTER; JOHNSON, 2011) obtido da literatura. Este quesito não se aplica aos projetos dos autores Rocha *et al.* (2019) e Ferreira *et al.* (2018).

Acerca da acurácia dos resultados de identificação dos aparelhos, os autores Abeykoon *et al.* (2016), Khan *et al.* (2019) e Barker *et al.* (2014) mostram bons resultados, com percentuais entre 90% e 100%. Vale ressaltar que os autores realizam testes com diversos dispositivos elétricos, alguns com características semelhantes e outros distintos. Nesse contexto, pode ser notado um bom desempenho dos algoritmos utilizados e dos modelos de treinamento, que também são aplicados neste trabalho. Os autores Ferreira *et al.* (2018) não relatam a precisão do método utilizado para identificar os aparelhos responsáveis pelo consumo detectado. Por fim, os autores Rocha *et al.* (2019) enfatizam apenas o controle dos equipamentos de ar condicionado, não se preocupando com a precisão na identificação dos aparelhos.

4 PROCEDIMENTOS METODOLÓGICOS

Neste capítulo são discutidas as ações necessárias para a execução da solução proposta. Na Seção 4.1, é explanada a avaliação das tecnologias, ferramentas, protocolos e arquitetura de comunicação utilizadas no projeto. Na Seção 4.2, é discutido o processo de construção e programação do *hardware* para a captura das amostras de corrente, necessárias para a classificação do sinal. Na Seção 4.3, são introduzidas as ações necessárias para a construção do conjunto de dados utilizado no treinamento dos modelos de classificação. A Seção 4.4, explica os procedimentos de seleção, treinamento e avaliação dos modelos de classificação. Na Seção 4.5, é introduzida a implantação dos serviços de Nuvem e Borda utilizando os *GEs* do *FIWARE*. A Seção 4.6, relata a construção do *dashboard* de apresentação dos resultados do projeto. Para finalizar, a Seção 4.7 descreve a avaliação da solução final. Na Figura 9 é ilustrada a sequência dos procedimentos.

Figura 9 – Sequência dos procedimentos



Fonte: Elaborada pelo próprio autor

4.1 Avaliação das Ferramentas e Tecnologias

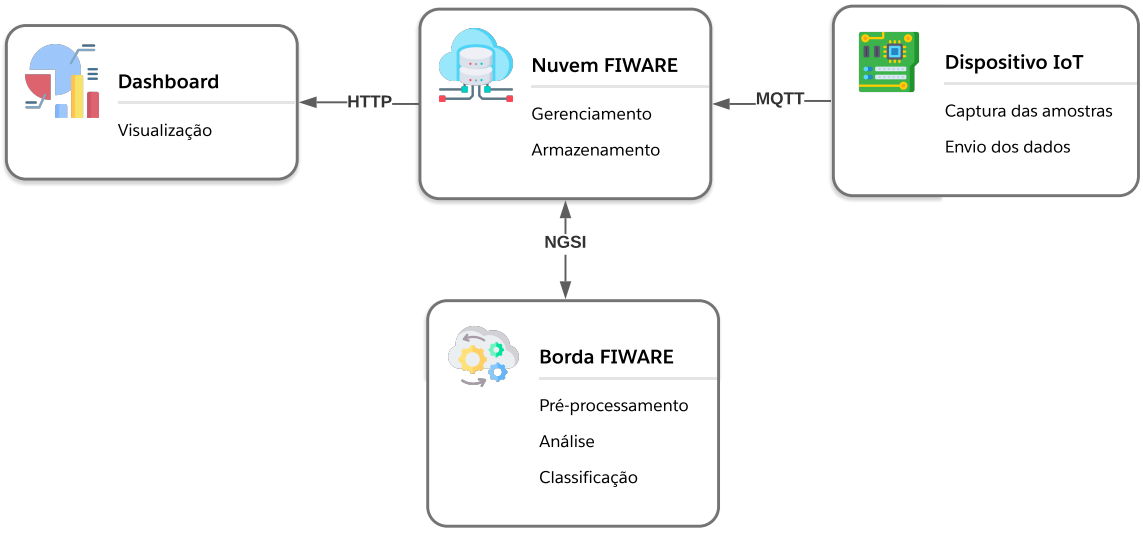
Nesta fase do projeto, foram exploradas as tecnologias que oferecem os serviços de comunicação, gerenciamento e armazenamento de dados. Nesse contexto, foram feitas inúmeras pesquisas para a escolha das tecnologias que fornecem os recursos necessários para este trabalho, bem como, os protocolos para a comunicação entre as diferentes partes do projeto. Nesse raciocínio, foi decidido que a arquitetura de computação em Nuvem e Borda seria empregada de modo a suprir as necessidades de adaptação a um ambiente *IoT*, como velocidade de comunicação

e processamento, fácil implantação e escalabilidade.

Assim, foi definido que as amostras de corrente seriam capturadas e enviadas por dispositivos *IoT* localizados em pontos estratégicos da rede elétrica do ambiente monitorado. Nessa ideia, por ser leve, rápido e voltado para aplicações *IoT*, o protocolo *MQTT* foi escolhido para o transporte de dados entre os dispositivos *IoT* e o núcleo responsável pelo gerenciamento, armazenamento e classificação das amostras.

Após vários estudos e testes, foi estabelecido o uso dos *GEs* do *FIWARE* para compor o núcleo de processamento e gerenciamento dos dados de contexto. Essa escolha foi tomada com base, principalmente, no grande número de serviços oferecidos pela plataforma, extremamente voltada a *IoT*. Além disso, a escalabilidade, interoperabilidade, os recursos de processamento em Nuvem e Borda, bem como, a fácil implantação foram pontos preponderantes para essa definição. Ainda, foi definida a necessidade de uma aplicação *dashboard* para visualização dos resultados. Essa aplicação possui uma comunicação direta com o *middleware FIWARE* para consumir os dados de identificação e consumo dos dispositivos elétricos monitorados. A Figura 10 mostra uma arquitetura de comunicação simplificada entre as diferentes partes do projeto.

Figura 10 – Arquitetura de comunicação simplificada



Fonte: Elaborada pelo próprio autor

Na fase de processamento, para classificar as amostras do aparelho elétrico monitorado, foi estabelecido o uso de algoritmos de Aprendizado de Máquina supervisionados voltados à classificação. Essa escolha se justifica pela adaptação ao modelo *NILM* e por facilitar a adição de novos aparelhos a serem identificados. Enfim, depois das diversas avaliações e definições

baseadas no levantamento bibliográfico realizado, o projeto foi colocado em prática seguindo os passos relatados posteriormente.

4.2 Desenvolvimento do dispositivo *IoT*

Neste processo, o dispositivo *IoT* foi projetado para então realizar a captura das amostras do sinal de corrente. Para isso, foram examinados os componentes que formam o *hardware* do dispositivo em questão. Desta forma, o sensor *SCT-013-000* foi selecionado com a principal justificativa de ser não intrusivo, obedecendo ao requisito de se utilizar do modelo *NILM* no processo de monitoramento. Em seguida, julgou-se necessário o uso do sensor em conjunto com um potenciômetro miniatura ajustável ou *Trimmer Potentiometer* (*Trimpot*) para regular a precisão da amperagem monitorada, além de permitir a leitura dos dados em *volts*.

Ainda, foi selecionado o microcontrolador *ESP8266*, mais especificamente a placa *ESP-01*, por possuir recursos não mais que suficiente para a leitura e envio das amostras, além de possuir um módulo de comunicação *WI-FI* integrado, considerando a ideia de comunicação sem fio de ambientes *IoT*. Ademais, a *ESP-01* se destaca pelo seu baixo consumo de energia, demandando cerca de 60 mA à 215 mA, dependendo de quais módulos estão em funcionamento.

Para finalizar a montagem do dispositivo, foi adicionado o conversor analógico-digital *ADS1115* para converter os dados analógicos do sensor, de modo a serem lidos pelo microcontrolador, pois possui uma resolução bem maior do que o conversor existente no *ESP8266*, além de ler valores negativos sem a adição de componentes no circuito. Já para o circuito de alimentação, foi empregada uma fonte de 9 V de tensão em conjunto com um regulador de tensão de 3,3 V, voltagem de referência do microcontrolador *ESP8266*.

4.3 Construção do Conjunto de Dados

Nesta fase, foi utilizado o dispositivo *IoT* para construir o conjunto de dados necessário para o treinamento dos modelos de classificação. Para isso, foi definido que o conjunto de dados conteria *features* fundamentadas nas amostras capturadas por período do sinal. Além disso, o rótulo correspondente ao nome do aparelho elétrico em que o sinal pertence. Desta maneira, cada aparelho elétrico será identificado por características particulares contidas em seu sinal de consumo, como valor máximo, valor mínimo e formato de onda.

4.4 Treinamento e avaliação dos modelos de classificação

Neste passo, foi treinado alguns dos vários modelos utilizados para classificação na área de Aprendizado de Máquina. Para isso, foi utilizada a linguagem de programação *Python*, onde o tratamento dos dados é processado com o auxílio das bibliotecas *Numpy*[1] e *Pandas*[2], além da biblioteca *Scikit-Learn*[3] que possui os modelos de classificação utilizados neste trabalho. Além disso, foram definidas as métricas de avaliação, baseando-se nas métricas escolhidas pelos trabalhos relacionados. Nessa ótica, as métricas foram utilizadas para avaliar os experimentos realizados com os modelos de classificação de modo a medir a eficácia de identificação realizada por cada um. Por fim, definir o modelo com melhor eficiência no contexto de classificação de aparelhos elétricos e a ser utilizado neste projeto.

4.5 Implantação da Arquitetura da Solução

Nesta parte do projeto, foi implantado os serviços de processamento em Nuvem e Borda proporcionados pelos *GEs* do *middleware FIWARE*. Deste modo, as amostras são recebidas do *broker Mosquitto* e processadas no *middleware*, onde é feita a identificação do aparelho e armazenados os resultados. Neste trabalho, foram utilizados contêineres *Docker* para executar o *Mosquitto* e todos os *GEs* empregados. Logo, foi realizada a comunicação entre todos os contêineres, desde a coleta das amostras até a disponibilização dos resultados.

4.6 Desenvolvimento do *Dashboard*

Nesta fase, foi desenvolvido o *dashboard*, responsável por mostrar os resultados da identificação e do monitoramento em tempo real. O desenvolvimento foi dividido em *front-end* e *back-end*. Dito isso, foi utilizado o *framework Javascript Node.js*, mais precisamente, a ferramenta *Express.js* para construir as rotas e funções do *back-end*. Deste modo, dados do aparelho identificado e do sinal de corrente monitorado foram apresentados no *software web* que recarrega as informações, automaticamente, sempre que há uma atualização no monitoramento.

[1] https://numpy.org/
[2] https://pandas.pydata.org/
[3] https://scikit-learn.org/stable/

4.7 Avaliação da Solução

Depois que todos os procedimentos para a construção deste projeto foram finalizados, foram realizados testes de identificação em um cenário real[4]. Para isso, foi selecionada uma tomada de uma residência, onde o dispositivo *IoT* foi instalado. Então, toda a infraestrutura em Nuvem foi executada em uma máquina *Ubuntu 16.04, 64 bits, 2.7 GHz, 8 GB RAM, 1 TB HD*. Além disso, foi utilizado um microcomputador *Raspberry Pi 3 model B* para executar o nó de Borda *FogFlow*.

Dito isso, os aparelhos elétricos presentes no conjunto de dados foram ligados individualmente à tomada monitorada e observados os resultados apresentados no *dashboard*. Então, foram registrados os resultados de 100 pacotes enviados pelo dispositivo *IoT*, para cada aparelho elétrico monitorado. Desta forma, a solução foi avaliada em um ambiente comum, com eletrodomésticos usuais de uma residência. Ademais, foram testados diferentes aparelhos da mesma categoria, como diferentes ventiladores, lâmpadas e carregadores de *smartphone*, aferindo de modo mais aprofundado a eficácia da solução final.

[4] https://drive.google.com/file/d/1X2vWo6ULiL4U3PnjDanrQtD9x5dRTGKA/view?usp=sharing

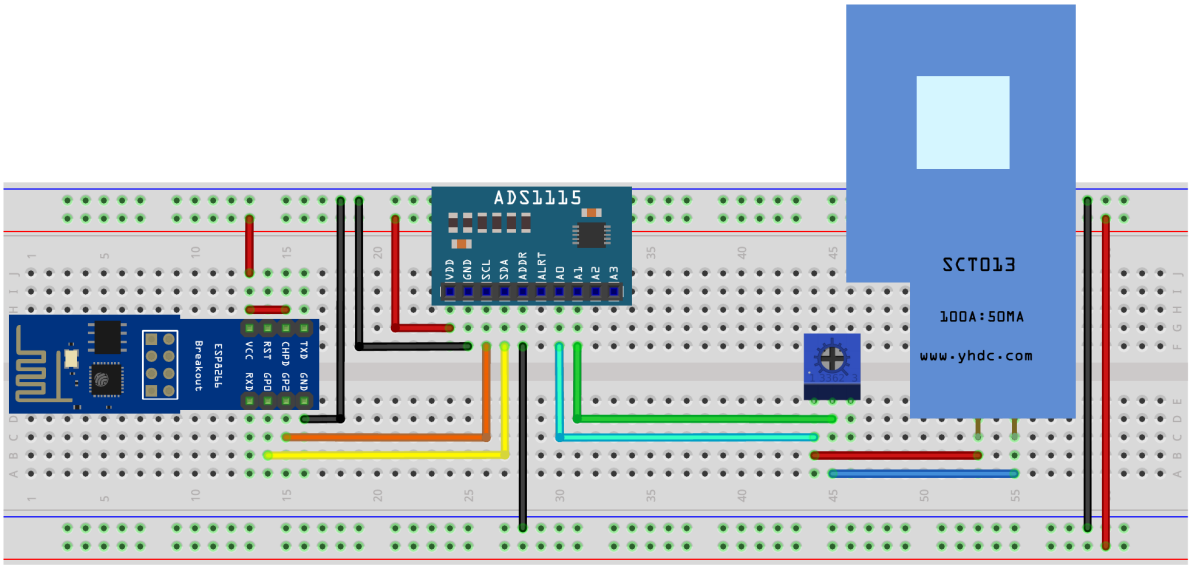
5 SOLUÇÃO PROPOSTA

Neste capítulo são detalhados os processos de construção da solução proposta. Na Seção 5.1, é esmiuçada a construção do dispositivo *IoT*. A Seção 5.2 descreve as definições e os passos para a composição do conjunto de dados. Na Seção 5.3, são relatados os procedimentos de treino e avaliação para a escolha do modelo de classificação utilizado na solução. Na Seção 5.4, são explanadas as configurações de implantação da arquitetura da solução, utilizando *GEs* do *FIWARE*. Por fim, na Seção 5.5 são relatados os processos de construção do *dashboard* de visualização dos resultados.

5.1 Construção do Dispositivo *IoT*

Primeiramente, foram adquiridos os componentes necessários para montagem do protótipo de aquisição das amostras. Então, cada componente foi testado individualmente para constatar as devidas condições de cada módulo. Isso inclui a regulagem da precisão do sensor através do *trimpot*, assim, os dados do sensor podem ser lidos pelo conversor *ADS1115* com precisão e em forma de tensão. Depois, foi construído um protótipo do dispositivo *IoT* em uma *protoboard* para realizar as primeiras leituras com o sensor. Na Figura 11 pode ser visualizado o circuito do dispositivo elaborado em uma *protoboard*, onde são mostradas as conexões e a ilustração de cada componente. Note que, o circuito de alimentação ainda não está presente no protótipo, sendo utilizada uma fonte de bancada regulada em 3,3 V nesta fase do processo.

Figura 11 – Protótipo do dispositivo *IoT*



Fonte: Elaborada pelo próprio autor

Como pode ser notado na Figura 11, foram utilizados dois pinos de leitura analógica do *ADS1115* com o intuito de empregar a forma diferencial de leitura, ou seja, o valor obtido por um dos pinos é diminuído do valor do outro pino. Desta forma, pode-se obter os valores negativos do sensor, dado que o sinal capturado é uma corrente alternada. Considerando que o *ADS1115* ler apenas valores de tensão, o *trimpot* foi empregado para transformar o valor de saída do sensor em valores de tensão, além disso, é possível controlar a proporção de transformação do sinal de saída para aperfeiçoar a precisão das amostras.

5.1.1 Regulagem do protótipo

Neste passo, foram feitas as configurações necessárias para a leitura das amostras utilizando o conversor *ADS1115*. Esses ajustes foram efetuados no registrador de configuração existente no conversor, segundo a folha de dados do produto. Como visto na Figura 11, foi preciso utilizar o modo diferencial para obter valores negativos do sinal alternado. Isso significa que, o *bit* 15 do valor convertido é utilizado como *bit* de sinal, considerando os 16 *bits* do *ADS1115* dispostos no intervalo [15:0]. Dito isso, foram usados os pinos analógicos A0 e A1 do *ADS1115* para realizar a leitura através da subtração $A0 - A1$, para tanto, o intervalo de *bits* [14:12] do registrador de configuração teve seu valor binário definido para 0b000.

Além disso, o *PGA* contido no conversor foi ajustado para 512 mV. Deste modo, o valor máximo produzido pelo *ADS1115* corresponderá ao valor de 512 mV. Para isso, o intervalo de *bits* [11:9] do registrador de configuração teve seu valor binário definido para 0b100. Também, foi definida a taxa de amostras por segundo para a máxima atingida pelo conversor, 860 *SPS*. Para esse fim, o intervalo de *bits* [7:5] do registrador de configuração teve seu valor binário definido para 0b111. Para os demais valores do registrador de configuração, foram seguidas as configurações recomendadas.

Para nível de exemplificação, o valor de 10 A *RMS* foi definido como valor máximo de corrente a ser capturado de um fio elétrico pelo sensor. Logo, valor de resistência do *trimpot* precisa ser regulado, considerando o fundo de escala do *ADS1115* para ampliar a precisão da leitura. Tendo em mente o valor máximo de 10 A *RMS* definido para a leitura de um fio elétrico, o valor máximo produzido pelo sensor é de 5 mA *RMS*, conforme a proporção de 100A:50mA descrita na folha de dados do sensor *SCT-013-000*. Logo, o valor máximo de pico de corrente produzido pelo sensor é de $0,005 * \sqrt{2} = 0,007071068$ A. Nessa ótica, a resistência do *trimpot* pode ser calculada pela Lei de Ohm, onde é descrito que resistência é diretamente proporcional à

tensão e inversamente proporcional à corrente que flui por um condutor. O cálculo é mostrado na equação 5.1, relacionando o valor máximo de tensão que pode ser lido pelo *ADS1115* e o valor máximo de corrente que pode ser produzido pelo sensor.

$$\text{Resistência do } \textit{trimpot} = \frac{0,512V}{0,007071068A} = 72,407732467\Omega \tag{5.1}$$

Desta forma, se o microcontrolador obter o valor máximo do *ADS1115*, então o valor de corrente no fio monitorado é de 10 A *RMS*, com os demais valores obedecendo a mesma proporção. Vale ressaltar que, o número máximo de valores produzido pelo conversor, desconsiderando o *bit* de sinal, é $2^{15} = 32768$, gerando uma resolução de $0,512V/32768 = 0,000015625V$. Com essas informações, é possível calcular o valor da corrente no fio monitorado a partir do valor produzido pelo conversor, conforme mostrado na equação 5.2, onde VPv é o número inteiro produzido pelo conversor multiplicado pela resolução de $0,000015625V$, 10 A a corrente máxima em *RMS* a ser capturada no fio e $0,512$ V a tensão máxima a ser lida pelo *ADS1115*.

$$\text{Corrente monitorada em RMS} = \frac{(VPv \cdot 10A)}{0,512V} \tag{5.2}$$

5.1.2 Avaliação de Captura das Amostras

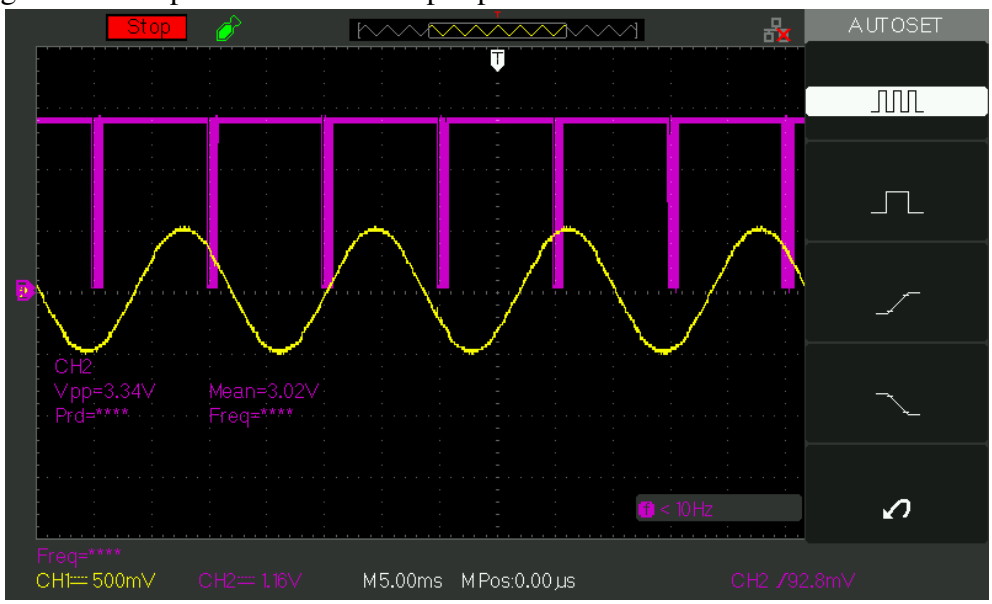
Nesta fase, foi avaliada a aquisição dos dados analógicos por meio do *ADS1115*. Os testes foram realizados com a ajuda de um gerador de função, então, o *ADS1115* foi testado com um sinal simulado de uma onda elétrica de 60 Hz, frequência padrão utilizada no Brasil. Além disso, foi utilizado um osciloscópio para observar a onda simulada em conjunto com o sinal de captura do conversor. Assim, foi possível analisar o número de amostras capturadas por período e o momento de cada leitura.

Primeiramente, foi utilizada a biblioteca *Adafruit_ADS1X15-master*[1] para realizar as configurações no conversor e o gerador de funções para simular o sinal de entrada. Com isso, pôde-se observar que o *ADS1115* estava capturando cerca de 2 amostras por período de um sinal de 60Hz e enviando via protocolo *I2C* para a *ESP-01*. Desta forma, é possível perceber que mesmo com o número de leituras por segundo sendo definido em 860 SPS, não foi obtido um bom desempenho do conversor que não conseguiu converter rapidamente as amostras, obtendo

[1] https://github.com/adafruit/Adafruit_ADS1X15

um número bastante insuficiente de dados. A Figura 12 mostra a onda senoidal de 60 Hz gerada na cor amarela, ainda, o sinal do pino *SDA* do *ADS1115*, na cor lilás, que indica o momento da captura e conversão das amostras.

Figura 12 – Captura de 2 amostras por período



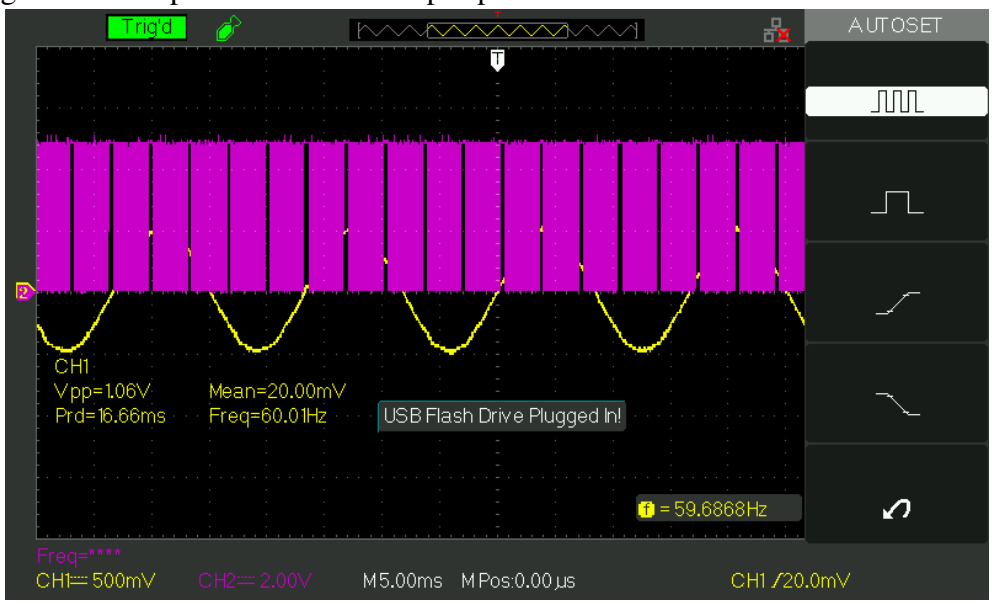
Fonte: Elaborada pelo próprio autor

Em busca de desempenhos melhores, foi testada a biblioteca *ADS1115-i2cdevlib*, que possui funções semelhantes à biblioteca *Adafruit_ADS1X15-master*. Desse modo, foram obtidos resultados melhores, pois a comunicação *I2C* era mais rápida obtendo 5 amostras por período do sinal monitorado. Todavia, o número de amostras ainda era insuficiente para analisar de maneira precisa as características do sinal. A Figura 13 mostra os resultados desse teste em um osciloscópio.

Ainda utilizando a biblioteca *ADS1115-i2cdevlib*, pôde-se notar que algumas funções ainda estavam tomando muito tempo, como a função de conversão *I2C* e a de multiplexação do pino de leitura. Com isso, a função de multiplexação foi retirada do laço infinito de leitura e colocada para executar apenas na inicialização do protótipo, além disso, o intervalo necessário entre as leituras do *I2C* foi reduzido empregando um laço de espera menos custoso, porém suficiente para o funcionamento devido do *I2C*.

Contudo, a função de conversão não pôde ser retirada do laço infinito de leitura, pois era necessário converter as informações do *I2C* em toda captura. Com esses ajustes pôde-se obter uma leitura mais rápida de cerca de 9 amostras por período de um sinal de 60 Hz. A Figura 14 mostra os resultados desses ajuste ilustrados no osciloscópio. Apesar dessas melhorias, mais

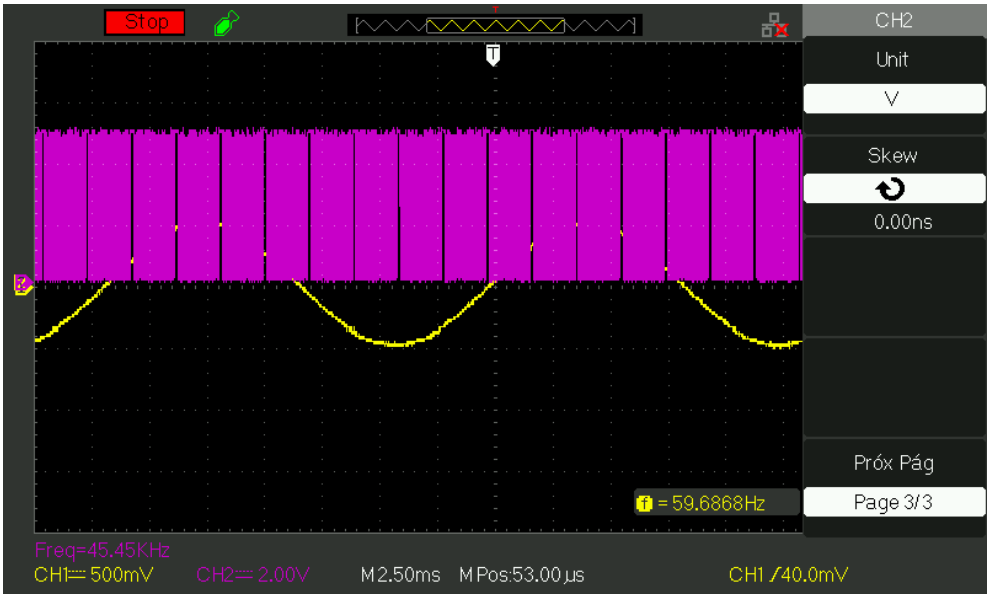
Figura 13 – Captura de 5 amostras por período



Fonte: Elaborada pelo próprio autor

ajustes e testes foram feitos para obter um número maior de amostras de modo a conseguir um sinal mais detalhado para a análise digital das amostras.

Figura 14 – Captura de 9 amostras por período

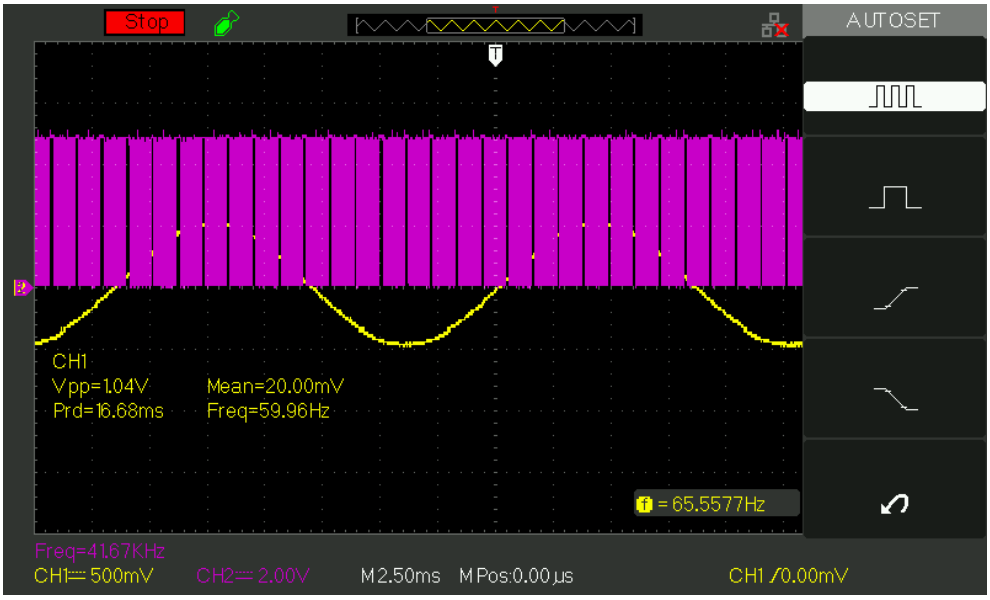


Fonte: Elaborada pelo próprio autor

Então, foi decidido não usar bibliotecas para as configurações do *ADS1115*. Assim, as configurações foram feitas diretamente nos *bits* do registrador *Config Register* do *ADS1115* com o auxílio da folha de dados do conversor. Logo, o número de amostras por período melhorou bastante podendo capturar cerca de 14 e até 15 amostras por período, uma boa quantidade considerando o trabalho (PACHECO[1] *et al.*, 2016), onde foi obtido um total de 15 capturas

por ciclo utilizando um *hardware* bem mais robusto e de maior custo. Considerando que, a velocidade de leitura foi configurada para 860 amostras por segundo e a onda de entrada é de 60 Hz = 0,016666667 s, logo o número de capturas é confirmado, pois 0,016666667 s * 860 SPS = 14,33333362 amostras. Esses resultados podem ser visualizados na Figura 15.

Figura 15 – Captura de 15 amostras por período



Fonte: Elaborada pelo próprio autor

5.1.3 Avaliação de Envio das Amostras

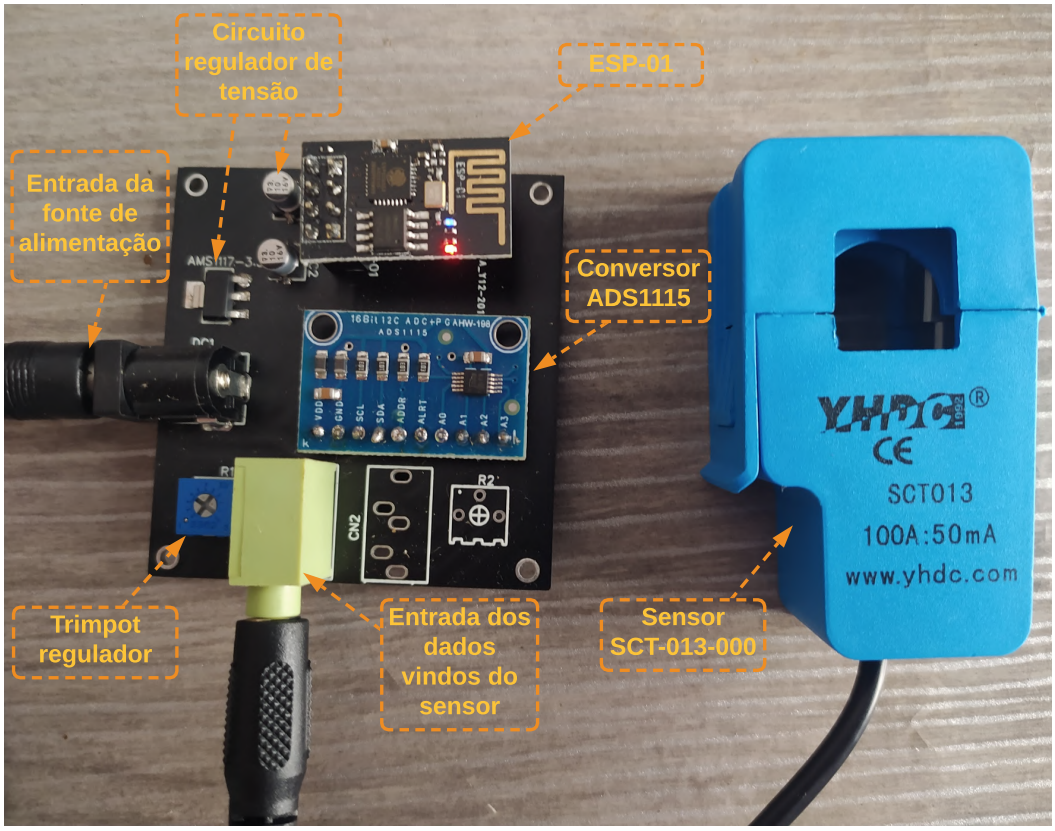
Como já inferido, as amostras do sinal monitorado são lidas pela placa *ESP-01* que utiliza o protocolo *I2C* para se comunicar com o conversor *ADS1115*. Antes de encaminhar as amostras para o *Mosquitto*, foi definido que o período de envio seria de 3 segundos. Desta forma, é garantido um bom fluxo de dados no *broker Mosquitto* e nos *GEs* do *FIWARE*, com baixa latência e sem perda de pacotes (AMURIM *et al.*, 2021).

Então, foi definido um total de 45 amostras consecutivas, o suficiente para compreender 3 períodos do sinal monitorado, enviadas a cada 3 segundos de modo a serem analisadas em Nuvem. Dito isso, foi testado o envio das amostras via *Wi-Fi* para o *broker Mosquitto* utilizando o protocolo *MQTT* e averiguada a integridade das amostras. Assim, todo o fluxo de dados do monitoramento foi testado utilizando o protótipo, antes da construção do dispositivo *IoT* final.

5.1.4 Construção do Dispositivo Final

Depois que o protótipo foi testado e configurado para obter um número de amostras suficientes para refazer o sinal de modo digital, foi construído o dispositivo *IoT*. De início, foi desenhada e encomendada uma placa de circuito impresso ou *Printed Circuit Board* (*PCB*) para substituir a *protoboard*, obedecendo às conexões definidas no protótipo. Depois, os componentes foram soldados à placa. Deste modo, evita-se um eventual mau contato entre as conexões que poderia interferir na comunicação ou na integridade dos dados. Além disso, deixa o dispositivo *IoT* mais compacto e portátil. Na Figura 16 é mostrado e descrito cada parte que constitui o dispositivo *IoT*.

Figura 16 – Dispositivo *IoT*



Fonte: Elaborada pelo próprio autor

O dispositivo pode ser alimentado com uma fonte de tensão de 5 V a 10 V, devido à utilização do regulador de 3,3 V. Neste caso, está sendo utilizado uma fonte de 9 V. Além disso, foi verificado que o consumo médio do dispositivo *IoT* é cerca de 200 mA, um baixo consumo considerando aplicações *IoT*. No Quadro 4, está listado os componentes que constituem o

hardware do dispositivo *IoT*, com alguns componentes auxiliares como capacitores que auxiliam o circuito de alimentação e conectores para interligar os diferentes módulos.

Quadro 4 – Componentes do *hardware*

NOME	QUANTIDADE
Conector *DC* P4	1
Regulador AMS1117-3.3	1
Capacitor 10 uF	2
Header 2x4	1
Trimpot 200 R	1
Conector Audio P2	1
ESP-01	1
Sensor *SCT*	1
ADS1115	1
Fonte 9 V	1

Fonte: Elaborado pelo próprio autor.

5.2 Composição do Conjunto de Dados

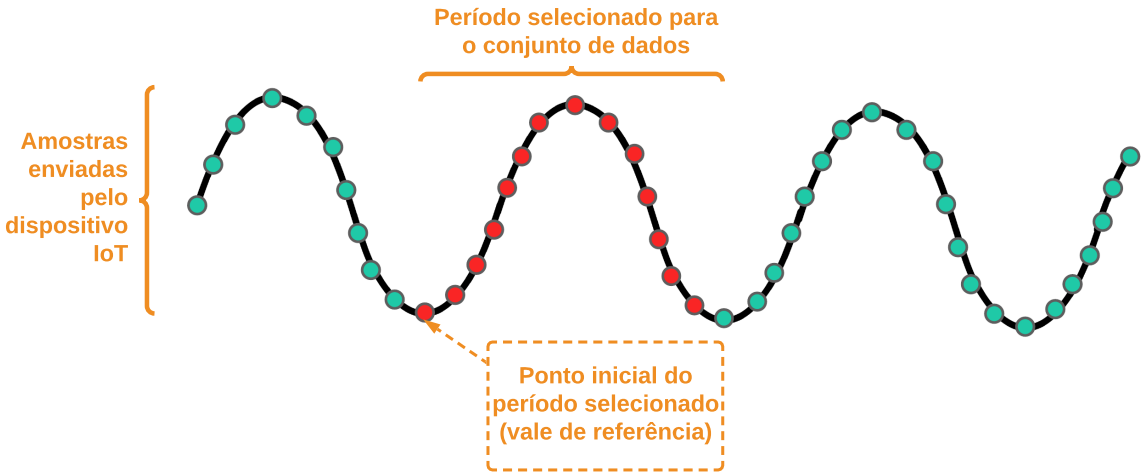
Para a composição do conjunto de dados, foram selecionados alguns aparelhos elétricos de diferentes características e extraídas amostras para compor cerca de 250 linhas do conjunto de dados para cada equipamento, quantidade baseada nos trabalhos relacionados. Os aparelhos foram: liquidificador, ventilador, secador de cabelo, lâmpada, carregador de *notebook*, sanduicheira, carregador de *smartphone*, ferro de solda e *TV*. Além disso, foram capturadas amostras quando a rede elétrica não possuía nenhum dispositivo conectado, de modo a identificar também esse cenário.

5.2.1 Pré-Processamento do Sinal

De início, foi definido que o conjunto de dados conteria 14 *features*, exatamente, as 14 amostras correspondentes a um período do sinal capturado. Para obter as 14 *features* necessárias para cada linha do conjunto de dados, cada dispositivo foi conectado individualmente a uma malha elétrica monitorada pelo dispositivo *IoT*. Desta forma, eram recebidas as 45 amostras enviadas via *MQTT*, equivalentes a cerca de 3 períodos do sinal. Então, foi identificado o primeiro vale do sinal entre as amostras recebidas e obtidas as 14 amostras consecutivas a contar desse vale, como mostra a Figura 17. Desta forma, todas as linhas do conjunto de dados equivalem a um período do sinal de cada aparelho monitorado, todos em fase. Logo, a sincronia de todos os períodos começando no mesmo ponto específico, ou seja, no vale do sinal, preserva as

características de cada *feature* no conjunto de dados, permitindo a identificação pelas assinaturas de carga.

Figura 17 – Dados enviados pelo dispositivo *IoT*



Fonte: Elaborada pelo próprio autor

Vale ressaltar que, o valor de cada amostra no conjunto de dados ainda estão no formato produzido pelo conversor *ADS1115*. A conversão para *ampère* não foi realizada de imediato com o intuito de preservar o nível de precisão de cada amostra, deixando mais nítidas as suas peculiaridades. Contudo, para questões de visualização, os valores do conjunto de dados podem ser facilmente convertidos para *ampère* seguindo a equação 5.2. Por fim, foi gerado um arquivo no formato *Comma Separated Values* (*CSV*) contendo todo o conjunto de dados adquirido.

5.3 Seleção do Modelo de Classificação

De início, o conjunto de dados foi carregado para ser usado nos testes com os modelos de classificação. Então, foi possível averiguar algumas questões como número de linhas, distribuição por rótulo, categoria dos dados e dados faltantes. Após testificar as boas condições do conjunto de dados, foi notada a necessidade de alterar o tipo dos rótulos que, atualmente, são palavras correspondentes aos nomes dos aparelhos monitorados. Logo, os valores de texto dos rótulos foram substituídos por um número inteiro mapeado ao nome real de cada aparelho. Desta forma, todos os dados do conjunto passaram a ser números inteiros, tendo em vista que as *features* são valores inteiros entre -32768 e +32767 produzidos pelo conversor analógico-digital.

Assim, os dados foram preparados para serem manipulados pelas demais funções.

Em seguida, o conjunto de dados foi dividido em X e Y, onde X representa as colunas de *features* e Y equivale à coluna dos rótulos. Depois, foi separado 70% dos dados para treino dos modelos e 30% para teste, de modo que essa divisão fosse estratificada conforme o Y, ou seja, mantendo a proporção dos rótulos em cada amostra da divisão. Desta forma, os dados conseguem treinar os modelos de classificação e testá-los em seguida, de modo a contatar a eficácia e o desempenho de cada um.

5.3.1 Treino e Teste dos Modelos

Primeiramente, foi realizada uma análise com diversos modelos de classificação, com hiper parâmetros fixos, para obtermos uma visão geral de quais modelos melhor se adapta ao conjunto de dados. Logo, utilizando o conjunto de treino, foram treinados os modelos supervisionados *KNN*, *Logistic Regression*, *SVM*, *Naive Bayes*, *Decision Tree*, *Random Forest*, *Gradient Boosting*, *Perceptron* e *Multilayer Perceptron* (*MLP*), com os dois últimos sendo técnicas de Aprendizagem Profunda. Então, foi utilizado o conjunto de teste para avaliar a predição dos modelos segundo as métricas de acurácia e *F1 score*. Ainda, visando extrair o máximo de eficiência de alguns modelos, esses procedimentos foram repetidos com os conjuntos de treino e teste escalonados e padronizados.

Além disso, foi realizado um segundo experimento utilizando validação cruzada com *k-fold* igual a 10. Ademais, variando os valores dos hiper parâmetros de cada algorítimo de classificação, dispondo de diversas combinações para encontrar o melhor modelo e configuração de modo a utilizá-lo no projeto. Para isso, foi utilizada a função *GridSearchCV* da biblioteca *Scikit-Learn*. No Quadro 5, é mostrado os valores que cada hiper parâmetro recebeu por vez e quais hiper parâmetros foram utilizados em cada classificador nesse experimento.

Vale ressaltar que, a métrica *F1 score* foi empregada na função *GridSearchCV* de modo a servir de parâmetro para avaliar a melhor combinação de modelo e hiper parâmetros. Depois que o melhor modelo foi selecionado, foi analisado o nível de importância de cada *features* do conjunto de dados. Ademais, outras métricas foram averiguadas como acurácia, *recall* e precisão, além de analisada a matriz de confusão do modelo.

Quadro 5 – Configurações de classificadores testadas

Classificador	Hiper parâmetros	Valores utilizados
KNeighborsClassifier	n_neighbors	1, 2, 3, 4, 5
	algorithm	auto, ball_tree, kd_tree, brute
LogisticRegression	random_state	42
	penalty	l1, l2, elasticnet, none
	C	1e+0, 1e+1, 1e+2, 1e+3, 1e+4
SVC	random_state	42
	C	1e+0, 1e+1, 1e+2, 1e+3, 1e+4
	kernel	linear, rbf, sigmoid
	gamma	scale, auto
	max_iter	10, 100, 1000
GaussianNB	var_smoothing	1e+0, 1e+1, 1e+2, 1e+3, 1e+4
DecisionTreeClassifier	random_state	42
	criterion	gini, entropy
	splitter	best, random
RandomForestClassifier	random_state	42
	criterion	gini, entropy
	n_estimators	10, 20, 25, 30, 40, 50
GradientBoostingClassifier	random_state	42
	n_estimators	10, 20, 30, 40, 50, 100
	learning_rate	0.1, 0.2, 0.4, 0.6, 0.8, 1.0
	max_depth	-1, 1, 2, 3
XGBClassifier	random_state	42
	n_estimators	10, 20, 30, 40, 50, 100
	learning_rate	0.1, 0.2, 0.4, 0.6, 0.8, 1.0
	max_depth	-1, 1, 2, 3
LGBMClassifier	random_state	42
	n_estimators	10, 20, 30, 40, 50, 100
	learning_rate	0.1, 0.2, 0.4, 0.6, 0.8, 1.0
	max_depth	-1, 1, 2, 3
Perceptron	random_state	42
	penalty	l1, l2, elasticnet, none
	max_iter	10, 100, 1000
	tol	1e-1, 1e-2, 1e-3
MLPClassifier	random_state	42
	learning_rate	constant, invscaling, adaptive
	max_iter	10, 100, 1000
	tol	1e-1, 1e-2, 1e-3

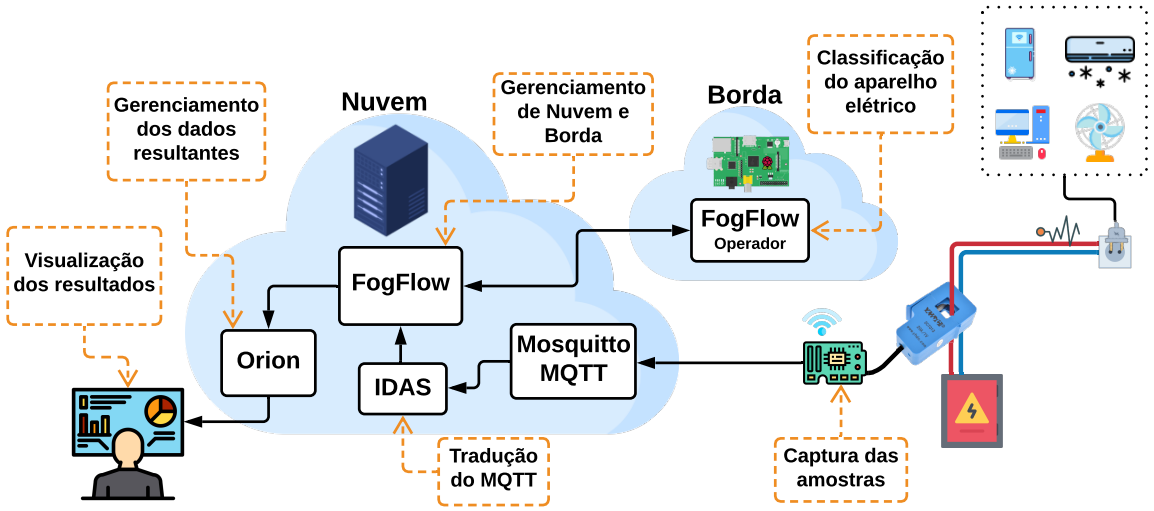
Fonte: Elaborado pelo próprio autor.

5.4 Arquitetura da Solução

A Figura 18 mostra toda a arquitetura da solução, resumindo todo o ambiente de comunicação e serviços. Como pode ser notado, o dispositivo *IoT* captura as amostras de corrente do aparelho ligado à tomada monitorada, logo após, publica os dados em um tópico do *broker Mosquitto* via protocolo *MQTT*. Então, o *GE IDAS* assina o tópico, coletando as amostras e traduzindo os dados para o modelo de entidades *NGSI*, de modo a serem lidos pelo *GE FogFlow*. Após isso, a entidade é enviada ao nó de Nuvem *FogFlow* que, através do tipo da entidade, localiza o operador responsável pela classificação das amostras, neste caso, localizado no nó de

Borda *Raspberry Pi*.

Figura 18 – Arquitetura da solução



Fonte: Elaborada pelo próprio autor

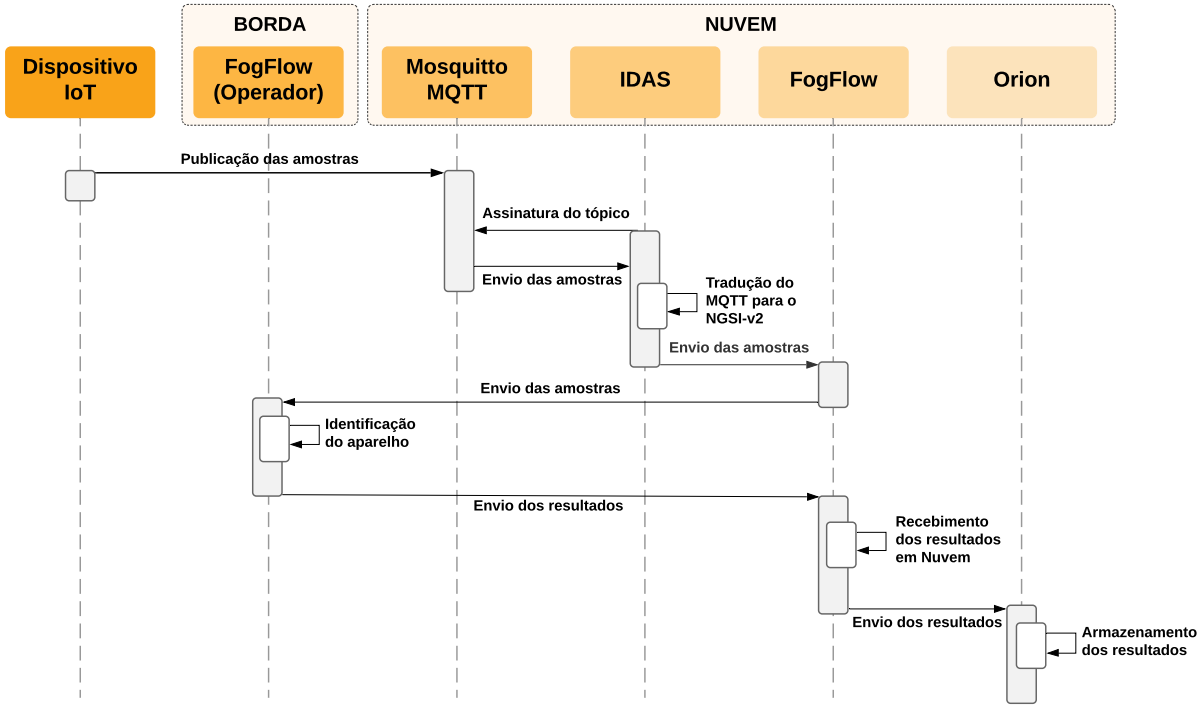
Então, os dados são enviados para o nó *Raspberry Pi* que realizada o processamento das amostras, utilizando o modelo de Aprendizado de Máquina voltado a classificação. Desta forma, o aparelho elétrico é identificado no nó de Borda. Logo após, o *Raspberry Pi* envia uma entidade resultante, contendo o nome do aparelho monitorado e o período do sinal utilizado na classificação, para o nó de Nuvem *FogFlow*. Então, o *GE FogFlow* encaminha a entidade resultante ao *GE Orion*, para ser armazenada.

Por fim, o *Orion* recebe e armazena os dados resultantes. Então, se a entidade contiver alguma atualização do monitoramento, o *GE Orion* comunica a aplicação *dashboard* para que os novos dados sejam apresentados, tanto a identificação do dispositivo quanto os dados do período do sinal. Na Figura 19, pode ser visualizado o diagrama de sequência que retrata a comunicação entre os diferentes serviços de Nuvem e Borda discutidos de modo mais aprofundado ao longo desta seção.

5.4.1 Tradução dos Dados

Como mostrado no diagrama de sequência da Figura 19, a infraestrutura de comunicação começa com o protocolo *MQTT* sendo utilizado no processo de envio das amostras ao núcleo de processamento em Nuvem, ou seja, entre a dispositivo *IoT* e o *FIWARE*. Para isso, é empregado o *Mosquitto*, o *broker* padrão do *MQTT*, que faz o intermédio de comunicação *MQTT*

Figura 19 – Diagrama de sequência da Nuvem



Fonte: Elaborada pelo próprio autor

utilizando a ideia de publicação e assinatura em tópicos. Desta forma, os dados são coletados pelo *IoT Agent* do *GE IDAS* que realiza a tradução do protocolo *MQTT* para o protocolo *NGSI* utilizado pelos demais *GEs*.

O *GE IDAS* é um conjunto de vários *IoT Agents*, cada um responsável por traduzir um determinado protocolo para o padrão *NGSI*, um protocolo em que os dados são representados em um modelo de entidades com atributos, tipo e valores. Como já inferido, neste trabalho foi selecionado o *IoT Agent* capaz de traduzir o protocolo *MQTT*. Então, após a execução dos contêineres, algumas configurações foram necessárias para efetivar a utilização dos recursos oferecidos pelo *GE*. Tais configurações foram realizadas por requisições *HTTP* enviadas ao *IoT Agent*. Dito isso, primeiramente, foi registrado um grupo de serviço para a conexão dos dispositivos *IoT*. Na Figura 20, é mostrado o modelo dessa requisição utilizando a ferramente *curl*. Neste cenário, o serviço *IDAS* está em execução na porta 4041.

Na comunicação MQTT, o registro do grupo de serviço disponibiliza uma chave de autenticação que informa o tópico em que o *IoT Agent* deve se inscrever. Neste caso, mostrado na Figura 20, o grupo de serviço informa ao *IoT Agent* que um conjunto de dispositivos *IoT* enviarão mensagens para o tópico */scggokgpepnvsb2uv4s40d59oo*. Além disso, é informado o *broker* para onde os dados serão enviados depois da tradução, neste caso, o *GE FogFlow* que está em execução na porta 8070. Ainda, é informado o tipo do grupo de serviço, nesse contexto,

Figura 20 – Requisição de registro do grupo de serviço

```
curl -iX POST \
  'http://10.0.0.182:4041/iot/services' \
  -H 'Content-Type: application/json' \
  -H 'fiware-service: openiot' \
  -H 'fiware-servicepath: /' \
  -d '{
 "services": [
  {
    "apikey":      "scggokgpepnvsb2uv4s40d59oo",
    "cbroker":     "http://10.0.0.182:8070",
    "entity_type": "Thing",
    "resource":    ""
  }
 ]
}'
```

Fonte: Elaborada pelo próprio autor

o tipo *Thing* foi escolhido. No entanto, o atributo *resource* não foi informado, útil apenas em *IoT Agents HTTP*.

Depois do registro do grupo de serviço, foi necessário cadastrar a entidade que representa o dispositivo *IoT*. Para isso, foi realizada uma nova requisição *POST* para o *IoT Agent*, mostrada na Figura 21. Nela são informados dados como *id*, nome e tipo, além dos protocolos *Ultralight* para representação de dados externos e *MQTT* para transporte. Ainda, foi registrado o atributo *current* para receber a série de 45 amostras enviadas, por vez, pelo dispositivo *IoT*.

Figura 21 – Requisição de registro do dispositivo *IoT*

```
curl -iX POST \
  'http://10.0.0.182:4041/iot/devices' \
  -H 'Content-Type: application/json' \
  -H 'fiware-service: openiot' \
  -H 'fiware-servicepath: /' \
  -d '{
 "devices": [
  {
    "device_id":   "currentSensor001001007",
    "entity_name":   "urn:ngsi-ld:CurrentSensor:001001007",
    "entity_type": "CurrentSensor",
    "protocol":    "PDI-IoTA-UltraLight",
    "transport":   "MQTT",
    "timezone":    "America/Fortaleza",
    "attributes": [
     {
      "object_id": "c",
      "name": "current",
      "type": "Text"
     }
    ]
  }
 ]
}
'
```

Fonte: Elaborada pelo próprio autor

No registro da entidade, o *IoT Agent* cria o tópico de comunicação no *broker Mosquitto*, neste cenário, o tópico */scggokgpepnvsb2uv4s40d59oo/currentSensor001001007/attrs* foi criado, seguindo os padrões do *IDAS* que utiliza a chave de autenticação e o *id* da entidade. Portanto, o *IoT Agent* espera que os dados sejam postos nesse tópico pelo dispositivo IoT, que foi previamente programado para enviá-los para lá. Deste modo, as amostras são enviadas para o *IDAS* via *MQTT* através do *Mosquitto*, traduzidas para o padrão de entidades *NGSI* e enviadas para o *broker* do *GE FogFlow*.

5.4.2 Interação Entre Nuvem e Borda

O *GE FogFlow* é o responsável pela identificação dos aparelhos através das amostras. Como mostra a Figura 19, parte do *GE* executa em Nuvem e parte, onde de fato a identificação é feita, é executada na Borda. Para isso, foi examinado e selecionado o microcomputador *Raspberry Pi* para servir como nó de Borda. Assim, os dados são recebidos em Nuvem pelo *FogFlow* que conforme o tipo dos dados, localiza o operador, executado no *Raspberry Pi*, e envia as amostras para serem processadas pelo modelo de classificação. Depois que a identificação é feita, os resultados são enviados para o nó de Nuvem *FogFlow* novamente.

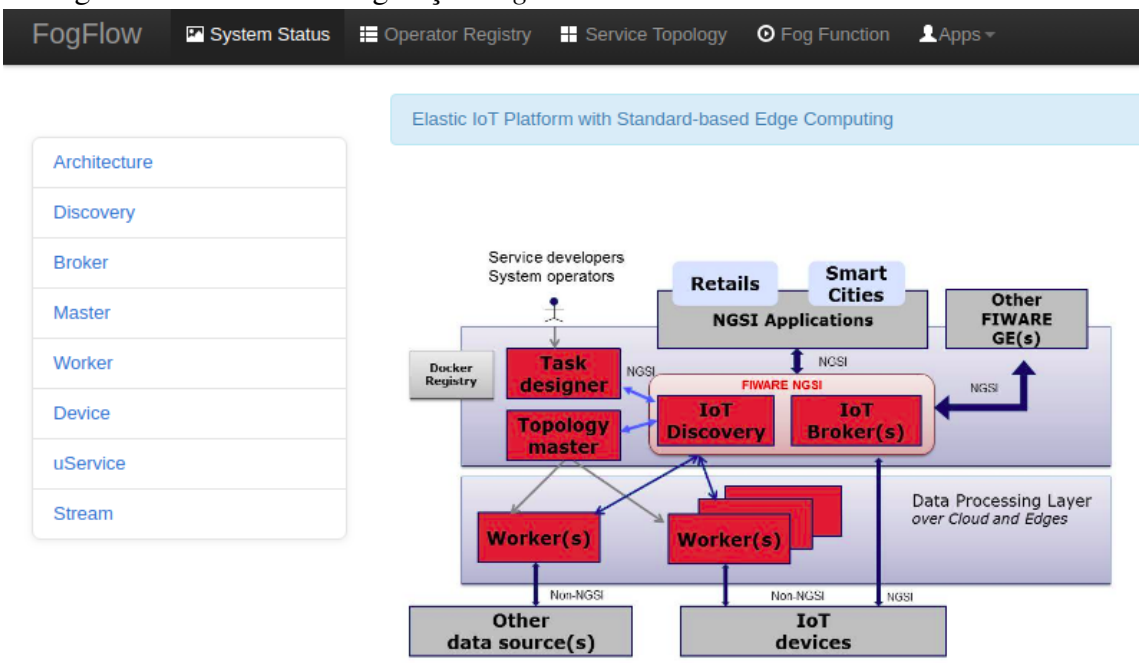
Como mostra a Figura 22, o *FogFlow* disponibiliza um *dashboard* para realizar as configurações do ambiente. A aba *System Status* possui um menu lateral em que é possível averiguar algumas informações da arquitetura *FogFlow* como, os *brokers* de comunicação, a localização e as conexões de cada nó, o nó mestre, os dispositivos *IoT* conectados e o fluxo de dados. Além disso, o *dashboard* contém as abas *Operator Registry* onde são realizadas as configurações dos operadores, *Service Topology* onde são estruturadas as arquiteturas de serviços, *Fog Function* onde são criadas as funções de ativação dos operadores e *Apps* onde são apresentados exemplos de arquiteturas já configuradas.

5.4.2.1 Configurações do Nó de Nuvem

Primeiramente, foi efetuado o cadastro do operador que realiza a classificação das amostras. Isso é feito na aba *Operator Registry*, como mostra a Figura 23. Como visto, o nome do operador escolhido foi *device_id*, já a descrição é opcional. Além disso, não foi necessário registrar nenhum parâmetro, precisando apenas do fluxo normal dos dados para o processamento, ou seja, o envio das amostras.

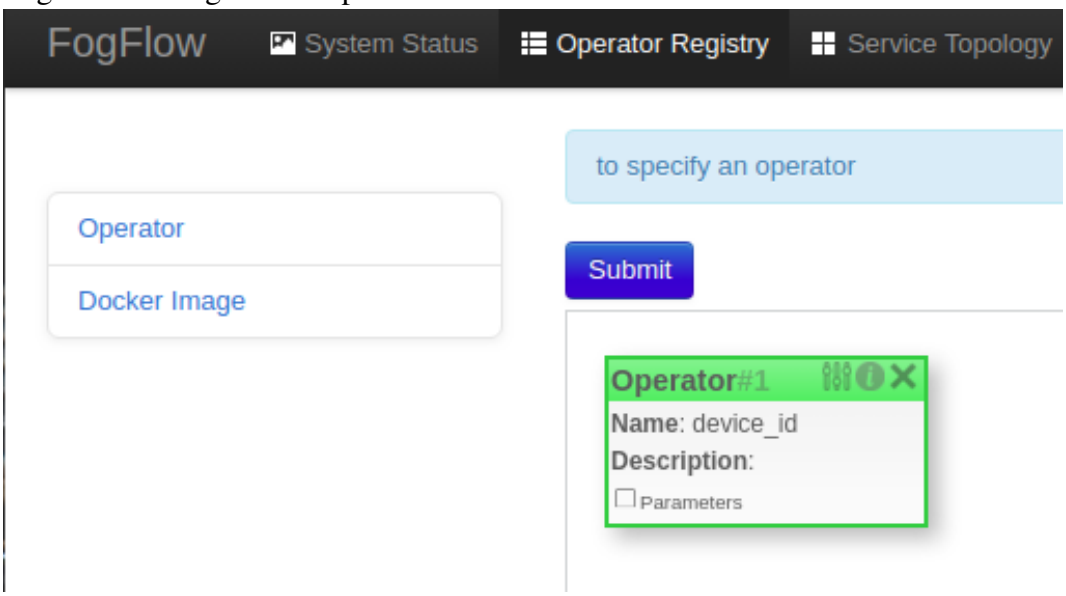
Todavia, apenas o registro do operador não é suficiente para o funcionamento correto

Figura 22 – Tela de configuração *FogFlow*



Fonte: Elaborada pelo próprio autor

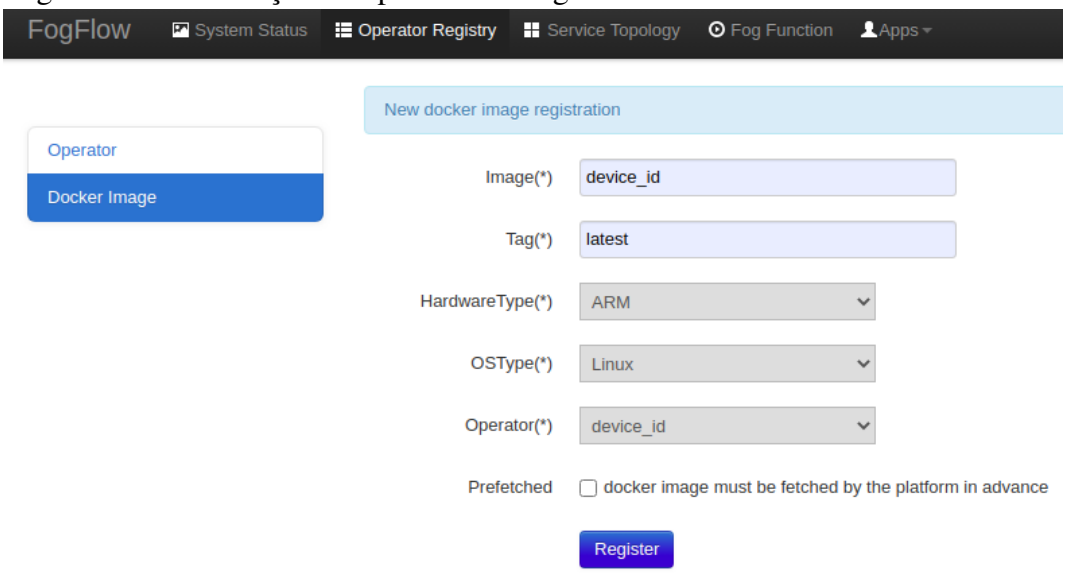
Figura 23 – Registro do operador



Fonte: Elaborada pelo próprio autor

do *FogFlow*. Logo, é necessário vincular o operador a uma imagem *Docker*. Posteriormente, a imagem será criada de fato no nó *Raspberry Pi*, onde o classificador baseado em Aprendizado de Máquina é executado. Tais informações são fundamentais para a vinculação, como mostra a Figura 24, onde foi preciso informar o nome da imagem, também denominada *device_id*, a *tag* e o tipo do *hardware* onde a imagem será executada, neste caso, foi selecionado o tipo *ARM* por se tratar do *Raspberry Pi*. Ainda, foi informado o sistema operacional *Linux* e o operador a ser vinculado.

Figura 24 – Vinculação do operador a imagem Docker



Fonte: Elaborada pelo próprio autor

Para finalizar, é essencial o registro da função de nevoeiro, ou função de Borda, na aba *Fog Function*. Dito isso, foi cadastrada uma função com o nome de *Device_id*, como mostra a Figura 25. Além disso, foi registrada uma tarefa com o nome de *device_id* vinculada ao operador *device_id* criado anteriormente. Ainda, foi especificado o fluxo de entidade como sendo do tipo *CurrentSensor* onde serão selecionados todos os atributos.

Desta forma, as amostras são enviados pelo *GE IDAS* no formato de entidades *NGSI*. Tal entidade possui o tipo *CurrentSensor*, anteriormente escolhido, como mostrado na Figura 21. Então, o *GE FogFlow* recebe a entidade e identifica o tipo *CurrentSensor*, esse processo serve de gatilho para executar a função de nevoeiro *Device_id*, que executa o operador atrelado a tarefa, enviando todos os atributos da entidade para a imagem *Docker device_id* localizada no microcomputador *Raspberry Pi*.

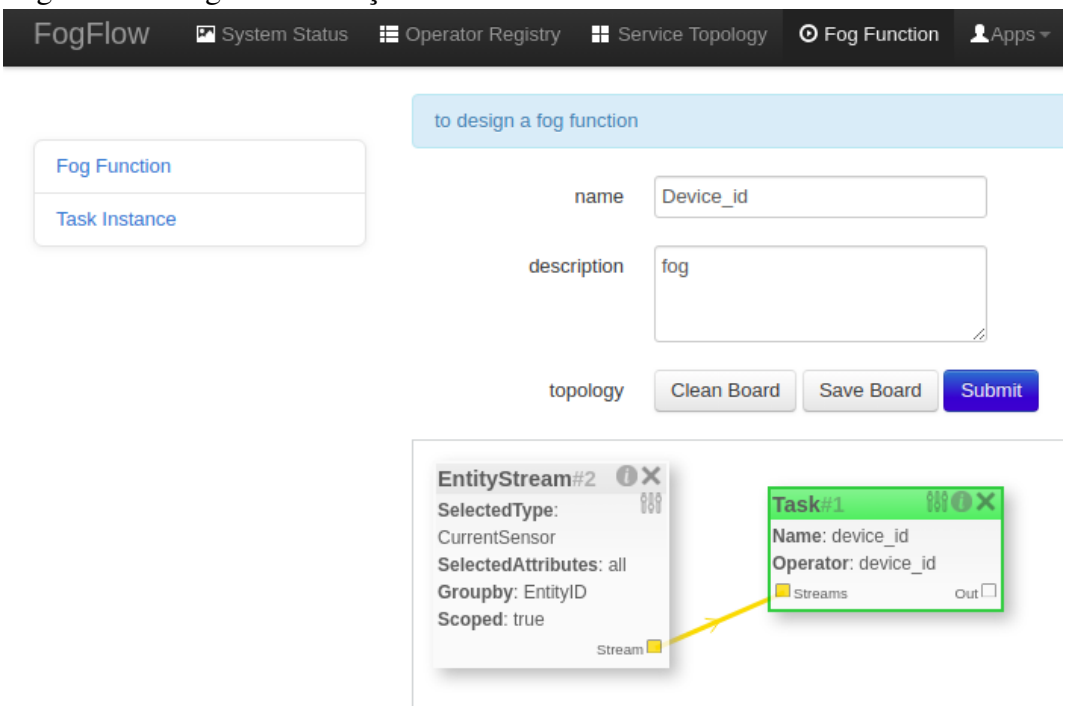
5.4.2.2 Configurações do Nó de Borda

Para utilizar o *Raspberry Pi* como um nó de Borda *FogFlow*, é necessário executar os componentes *broker* e *worker* através do *Docker*. Dessa maneira, são feitas configurações tanto no *broker* quanto no *worker*, informando endereços e portas de comunicação para efetivar a troca de dados com o nó de Nuvem *FogFlow*. Desse modo, as devidas configurações foram feitas, contudo, é indispensável a criação da imagem *Docker* executada pelo operador.

Logo, a imagem *Docker* foi criada baseada na imagem *elswork/rpi-tensorflow*[2] que

[2] https://hub.docker.com/r/elswork/rpi-tensorflow

Figura 25 – Registro da função de Borda



Fonte: Elaborada pelo próprio autor

já possui as bibliotecas instaladas, necessárias para executar o código *Python* de classificação das amostras, como demonstra a Figura 26. Como pode ser inferido, são necessárias funções de comunicação para que a imagem do operador possa interagir com os demais componentes do *GE FogFlow*. Para isso, a comunidade *FogFlow* disponibiliza modelos[3] de imagens para operadores, com as configurações de comunicação já implementadas em diferentes linguagens de programação.

Figura 26 – Arquivo *dockerfile* da imagem do operador

```
FROM elswork/rpi-tensorflow
RUN mkdir /task
ADD main.py /task
ADD samples_dataset.csv /task
ADD requirements.txt  /task
WORKDIR /task
RUN pip install -r requirements.txt
CMD ["python", "./main.py"]
```

Fonte: Elaborada pelo próprio autor

Como notado na Figura 26, foi adicionado o arquivo *CSV* que possui o conjunto

[3] https://github.com/smartfog/fogflow/tree/master/application/template

de dados e o arquivo *main.py* que contém o código *Python* com o modelo de classificação responsável pela lógica de identificação dos aparelhos elétricos. Desta forma, a imagem foi criada com o nome de *device_id*. Logo, quando as amostras são enviadas ao nó de Nuvem *FogFlow* e o operador *device_id* é acionado, o orquestrador *FogFlow* localiza o nó que possui a imagem vinculada ao operador, neste caso, o *Raspberry Pi*, e a executa enviando os dados de contexto.

Depois que a identificação é feita, utilizando o modelo de Aprendizado de Máquina, uma nova entidade com o tipo *Result* é enviada ao nó de Nuvem pelo nó de Borda, contendo o resultado da identificação e o período do sinal de corrente atual capturado, como mostra a Figura 27. Assim, essa entidade resultante pode ser enviada ao *GE Orion*, responsável pelo armazenamento e disponibilidade para o consumo dos dados.

Figura 27 – Modelo da entidade resultante


```
{
  "entityId": {
        "id": "CurrentSensor001001007",
        "type": "Result"
  },
  "attributes": [
        "device": {
              "type": "string",
              "value": "notebook power adapter"
        },
        "samples": {
              "type": "int",
              "value": [-1040, 80, 96, 112, 96, 64, 224,
                    864, -80, -96, -112, -96, -64, -16]
        }
    ]
}
```


Fonte: Elaborada pelo próprio autor

5.4.2.3 Comparação Entre Arquiteturas de Nuvem e Borda

Como visto na Figura 19, foi preferido enviar as amostras diretamente para o nó de Nuvem. Desta forma, o orquestrador *FogFlow* pode detectar o nó de Borda onde está localizada a imagem *Docker* do operador e só então enviar as amostras para serem processadas. Deste modo, os componentes *Mosquitto* e *IDAS* executam somente no nó de Nuvem, deixando o *Raspberry Pi* apenas com parte do *GE FogFlow*. Assim, focando seu poder de processamento,

majoritariamente, na classificação das amostras.

Alternativamente, as amostras poderiam ser enviadas diretamente ao nó de Borda. Todavia, esse modelo de comunicação exigiria que o *Raspberry Pi* possuísse os componentes *Mosquitto* e *IDAS* para recebimento e tradução dos dados. Evidentemente, esses componentes seriam replicados na adição de novos nós de Borda, repetindo núcleos desses serviços na infraestrutura da Nuvem. Além disso, o tipo da entidade dos dados recebidos teria que ser comunicado ao nó de Nuvem, para que o orquestrador permitisse o processamento naquele determinado nó.

Contudo, o modelo alternativo se comunicaria diretamente com os dispositivos *IoT*, deixando a Borda mais próxima no que diz respeito ao tempo de resposta. Portanto, as vantagens desses dois modelos de comunicação podem ser exploradas considerando a localização e o tamanho do ambiente inteligente. Neste projeto, também foram realizados testes preliminares com o modelo alternativo de comunicação, porém, análises mais aprofundadas serão realizadas futuramente para avaliar os dois modelos na infraestrutura, considerando o tempo de resposta, a taxa de descarte e o consumo de *CPU* e memória.

5.4.3 Gerenciamento dos Resultados

Depois que o nó de Nuvem *FogFlow* rebebe a entidade resultante, ele encaminha os resultados ao *GE Orion* por uma *subscription* cadastrada no *FogFlow*. No contexto do *FIWARE*, as *subscriptions* são como gatilhos que enviam dados para um determinado endereço conforme uma condição. O registro dessa *subscription* pode ser feito através de uma requisição *POST*, como mostra a Figura 28, utilizando a ferramenta *curl*.

Como pode ser constatado na Figura 28, a requisição é direcionada ao endereço do *FogFlow*, que executa na porta 8070 por padrão. Logo, informa que as entidades do tipo *Result* serão enviadas para um endereço na porta 1026, onde o *GE Orion* está em execução. Desta maneira, todas as entidades resultantes são encaminhadas ao *Orion Context Broker*, o *Enabler* central do *FIWARE*. Assim, esses dados são armazenados temporariamente no *Orion*, onde podem ser consumidos e analisados por outros *GEs* do *FIWARE* que oferecem diferentes serviços como armazenamento em banco de dados e análise de *Big Data*, todavia, esses serviços estão fora do escopo deste trabalho.

Depois que os dados foram armazenados no *GE Orion*, foi cadastrada outra *subscription*, desta vez no próprio *Orion*, para enviar os dados ao sistema *web* sempre que o resultado da

Figura 28 – Requisição de registro da *subscription FogFlow*

```
curl -iX POST \
'http://10.0.0.182:8070/ngsi10/subscribeContext' \
-H 'Content-Type: application/json' \
-H 'Destination: orion-broker' \
-d '{
  "entities": [
    {
     "type": "Result",
     "isPattern": true
    }
  ],
  "reference": "http://10.0.0.182:1026/v2/op/notify"
}'
```

Fonte: Elaborada pelo próprio autor

classificação sofrer alguma atualização. Deste modo, as informações são recebidas pelo *back-end* do sistema e apresentadas no *dashboard*. A requisição para o registro dessa *subscription* é mostrada na Figura 29, com o uso da ferramenta *curl*.

Figura 29 – Requisição de registro da *subscription Orion*

```
curl -iX POST \
'http://10.0.0.182:1026/v2/subscriptions' \
-H 'Content-Type: application/json' \
-H 'Destination: orion-broker' \
-d '{
  "description": "Current Subscription",
  "subject": {
      "entities": [{
          "id": "CurrentSensor:001001007",
          "type": "Result"
      }],
      "condition": {
          "attrs": ["device"]
      }
  },
  "notification": {
      "http": {
          "url": "http://10.0.0.182:4000/device"
      },
      "attrs": ["device", "samples"]
  },
}'
```

Fonte: Elaborada pelo próprio autor

Como pode ser notado na Figura 29, é enviada uma notificação para a porta 4000,

onde o *back-end* do *dashboard* está em execução, com os atributos *device* e *samples* da entidade resultante, que possuem o nome do aparelho identificado e o período do sinal de corrente atual respectivamente, como mostrado na Figura 27. Todavia, a requisição também informa que a notificação só é enviada se o atributo *device* da entidade resultante for atualizado. Isso é esclarecido através dos atributos *entities* e *condition* mostrados na requisição. Desta forma, o *Orion* informa ao back-end do *dashboard* sempre que há alguma atualização no fio monitorado.

5.5 Construção do *Dashboard*

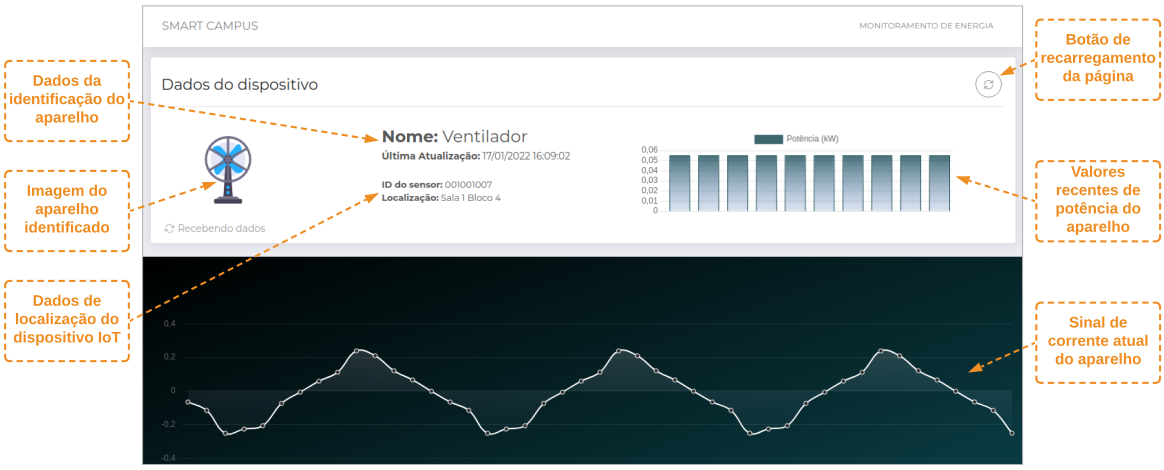
O *dashboard* do projeto foi dividido em *back-end* e *front-end*. Dito isso, foi feita uma rota chamada */device* no *back-end*, construído utilizando o *framework Node.js*, responsável por receber as notificações do Orion, como mostra a Figura 29. Então, o servidor *back-end* foi posto em execução na porta 4000. Após os dados serem recebidos pelo servidor, alguns tratamentos são realizados neles antes de enviá-los para o *front-end*. Primeiramente, o nome do dispositivo identificado é traduzido para a língua portuguesa para questões de apresentação, dado que o conjunto de dados foi construído utilizando a língua inglesa. Como já mencionado, também é recebido amostras referentes ao período atual do sinal monitorado. Logo, esses valores são convertidos para Ampère seguindo as fórmulas mostradas na Seção 5.1.1. Além disso, é capturado o horário em que os dados são recebidos. Estão, os dados tratados são enviados ao *front-end* via *web socket*, utilizando a biblioteca *Socket.io* do *Node.js*.

Na construção do *front-end*, foi utilizado o *framework Javascript React.js*, que proporciona flexibilidade e facilidade de comunicação com outras ferramentas, dentre outras vantagens. Então, os dados são recebidos pelo *front-end* e apresentados na tela, como mostra a Figura 30. Como visto, são apresentados os dados do dispositivo identificado, como o nome e a imagem, além da data e hora da última atualização. Tais informações são provenientes dos dados enviados pelo *Orion* ao *back-end*.

Além disso, são informados os dados de localização do dispositivo *IoT*, como o *id* do sensor e a localização no ambiente *IoT*. Atualmente, essas informações são fixas na entidade referente ao dispositivo *IoT*, como atributos estáticos. É importante frisar que, os dados são atualizados instantaneamente a medida que o *front-end* recebe os pacotes do *back-end* via *web socket*. Contudo, há um botão de recarregamento da página, caso ocorra alguma inconsistência na comunicação.

Ademais, o *dashboard* apresenta um gráfico com os valores recentes de potência do

Figura 30 – *Dashboard do projeto*



Fonte: Elaborada pelo próprio autor

aparelho monitorado, calculados a partir do período do sinal de corrente enviado pelo *back-end*. Atualmente, esses valores são aproximados, considerando uma tensão de 220 V. Por último, é mostrado um gráfico referente à onda atual de corrente capturada pelo dispositivo *IoT*, também construído a partir das amostras correspondentes a um período do sinal monitorado enviadas pelo *back-end*. Desta forma, todo o projeto foi construído, deste o dispositivo *IoT* até o *dashboard* que apresenta os resultados do monitoramento.

6 RESULTADOS E DISCUSSÕES

Neste capítulo, são detalhados os resultados dos experimentos e produtos desenvolvidos neste trabalho. Na Seção 6.1, é mostrado o desempenho de aquisição do dispositivo *IoT*. Na Seção 6.2, são detalhadas as características do conjunto de dados resultante. Na Seção 6.3, são mostrados os resultados de eficácia dos modelos de classificação testados. Por fim, na Seção 6.4 são descritos os resultados da avaliação da solução completa.

6.1 Aquisição de Amostras do Dispositivo *IoT*

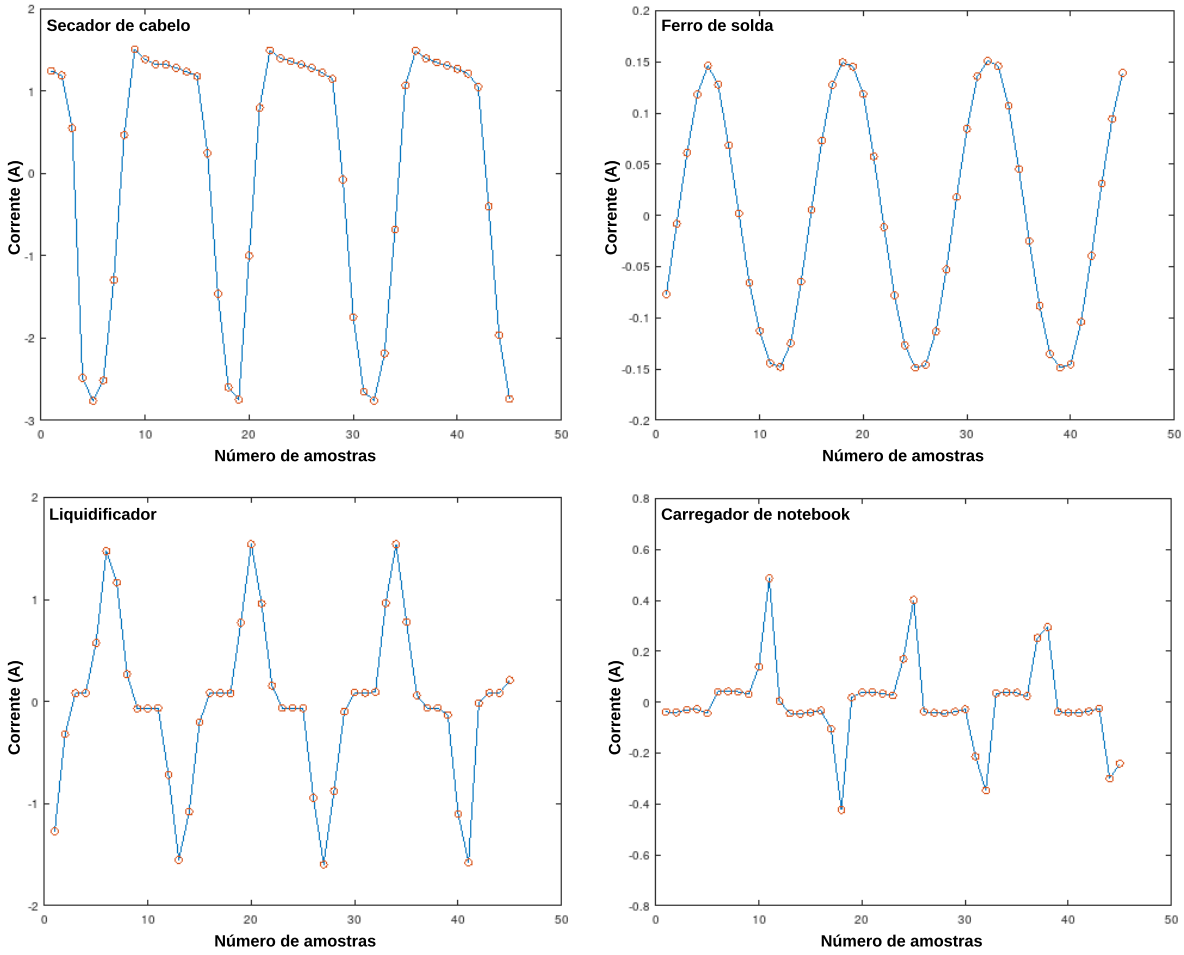
Após a construção do dispositivo *IoT*, foram realizados testes com diversos aparelhos elétricos reais, antes da construção do conjunto de dados, para certificar a eficácia do dispositivo *IoT* na captura do sinal de corrente. Nesse contexto, foram capturadas amostras com o intuito de refazer o sinal de modo a observar o nível de detalhamento obtido. Dado que uma boa quantidade de detalhes é importante para diferenciar os vários sinais e identificar as assinaturas de cada equipamento. Dessa forma, o nível de detalhamento do sinal capturado está diretamente ligado ao número de amostras capturadas por período.

Na Figura 31, podem ser visualizados sinais de corrente recriados a partir das amostras capturadas pelo dispositivo *IoT*. Dito isso, são apresentados sinais de alguns dos aparelhos testados como secador de cabelo, ferro de solda, liquidificador e carregador de *notebook*. Como visto, o número de 14 amostras por período se confirma em cada sinal. Além disso, o dispositivo *IoT* alcança um nível aceitável de detalhamento do sinal, sendo possível identificá-los apenas observando o conjunto de amostras a olho nu.

Contudo, em casos que os formatos de onda possuem um certa semelhança como, por exemplo, o liquidificador e o carregador de *notebook* mostrados na Figura 31, outras assinaturas podem ser utilizadas para a distinção entre esses aparelhos, neste caso, o valor de pico do sinal. Pois, mesmo com a semelhança no formato, o liquidificador possui valores de consumo bem maiores que o carregador de *notebook*, como mostrado na escala dos gráficos da Figura 31.

Ainda, vale ressaltar algumas características que se confirmam como, por exemplo, o ferro de solda que possui um sinal de consumo de corrente no formato senoidal. Neste caso, isso é previsível por razão de ser um equipamento puramente resistivo. Além disso, também foram observados padrões no formato de onda de cargas eletrônicas, sustentando a ideia de serem equipamentos com estruturas semelhantes.

Figura 31 – Sinais refeitos com amostras capturadas



Fonte: Elaborada pelo próprio autor

6.2 Especificidades do Conjunto de Dados

Como dito, o conjunto de dados foi construído a partir de amostras capturadas pelo dispositivo *IoT*. Então, foram realizado testes de aquisição com vários dispositivos, sendo eles um secador de cabelo, uma *TV*, um ventilador, um ferro de solda, um liquidificador, um carregador de notebook, uma lâmpada de *LED*, uma sanduicheira e um carregador de *smartphone*. Vale lembrar que, as *features* do conjunto de dados correspondem às 14 amostras referentes a um período do sinal capturado de forma sincronizada a partir do vale da onda. Desta forma, foi gerado um arquivo no formato *CSV* para armazenar o conjunto de dados.

A Figura 32 contém algumas linhas do arquivo *CSV* resultante. Logo, pode ser visualizada as 14 amostras em cada linha, com valores gerados pelo dispositivo *IoT*, ainda não convertidos para Àmpere. Ainda, o rótulo correspondente a cada linha na coluna *device*. É importante frisar que, o conjunto de dados não possui nenhum dado faltante, além de todas as

features possuírem o mesmo tipo de dados, ou seja, números inteiros, o que facilita a manipulação dos valores por funções.

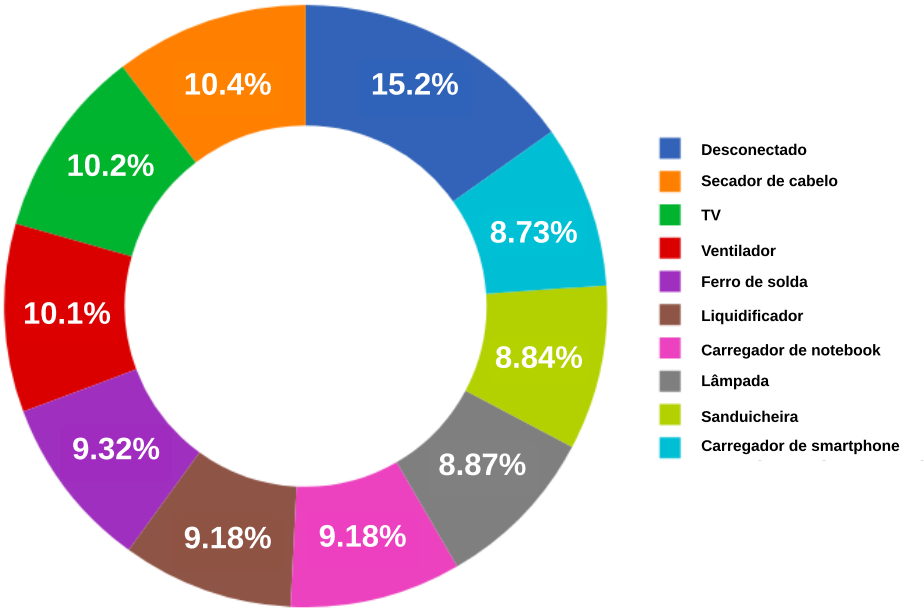
Figura 32 – Exemplo de linhas do conjunto de dados

	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	device
2086	-4528	-4352	-4128	-4000	-3872	-3712	-3568	-2640	3680	7552	8144	7792	3712	-1536	hairdryer
1712	-176	-80	-48	-32	-32	-32	16	160	96	48	48	0	32	-16	lamp
1480	-5312	-2848	-624	304	272	592	3360	5696	2656	528	-304	-288	-1808	-4208	blender
1905	-4480	-4160	-4080	-3936	-3808	-3664	-3504	-800	5488	8160	8048	6160	1856	-3664	hairdryer
2325	-432	-400	-304	-112	112	224	368	432	400	320	144	-64	-240	-384	soldering iron

Fonte: Elaborada pelo próprio autor

Ademais, o conjunto de dados foi construído de forma balanceada, ou seja, as quantidades de linhas referente a cada rótulo são relativamente próximas, o que melhora, significativamente, o desempenho dos modelos de classificação. Dito isso, o conjunto de dados possui um total de 2886 linhas distribuídas para 10 classes. O percentual da distribuição de cada classe é mostrado na Figura 33. Como notado, há uma classe que armazena as capturas sem aparelhos conectados, para que esse cenário também seja identificado.

Figura 33 – Distribuição de linhas por rótulos do conjunto de dados



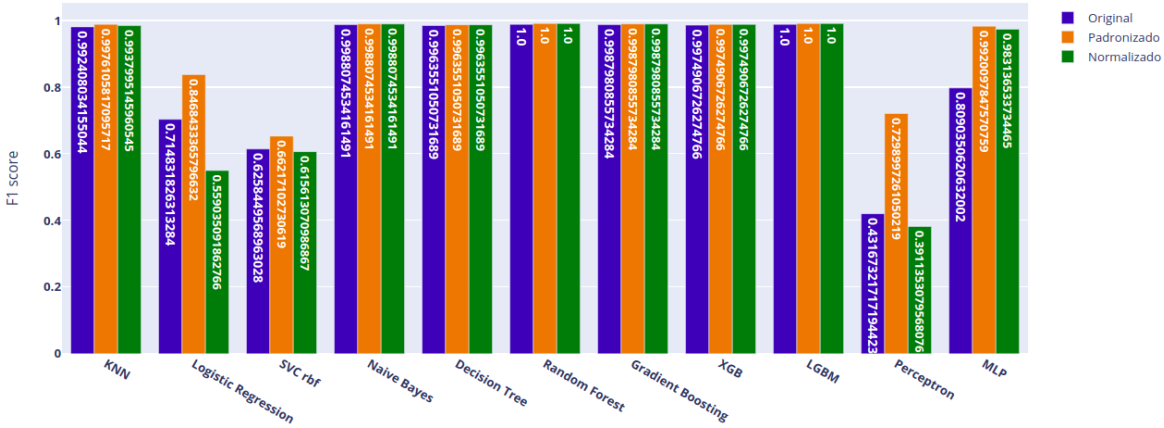
Fonte: Elaborada pelo próprio autor

6.3 Eficácia dos Modelos de Classificação

Como dito na Subseção 5.3.1, foram realizados duas séries de testes utilizando diferentes modelos de classificação baseados em Aprendizado de Máquina, utilizando 70% do conjunto de dados para treino e 30% para teste. Comparado ao segundo, o primeiro experimento se caracteriza por ser mais superficial, utilizando os modelos com hiper parâmetros fixos. Os modelos também foram aplicados com os dados normalizados e padronizados, mesmo não havendo relevância nos modelos *Random Forest* e *Gradient Boosting*. Desta forma, foram capturados os valores de *F1 score* e acurácia de todos os cenários testados.

A Figura 34 mostra os resultados dos modelos de classificação utilizados conforme a métrica *F1 score*, com os dados originais, padronizados e normalizados. Como notado, os modelos *Logistic Regression*, *SVC* e *Perceptron* não tiveram um bom desempenho comparado aos demais. Já o modelo *MLP* teve uma melhora significativa de eficácia considerando os dados padronizados e normalizados, ficando próximo a 1. Entretanto, os demais modelos tiveram resultados relevantes, com os algoritmos *KNN*, *Naive Bayes*, *Decision Tree*, *Gradient Boosting* e *XGB* chegando a 99% de *F1 score*. Todavia, foram os modelos *Random Forest* e *LGBM* que obtiveram os melhores resultados, chegando a 100% de *F1 score*.

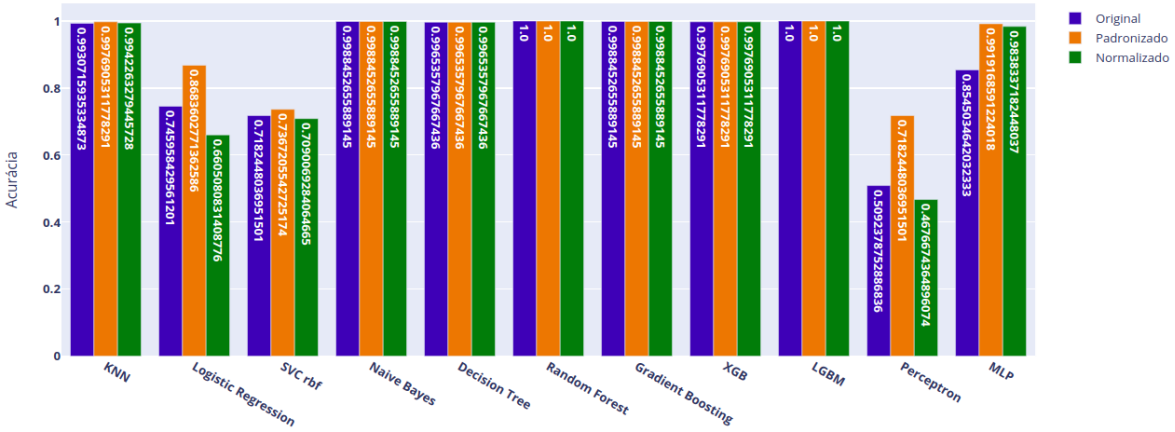
Figura 34 – Resultados de *F1 score* dos modelos testados



Fonte: Elaborada pelo próprio autor

Como pode ser notado na Figura 35, os valores de acurácia foram bem semelhantes aos valores de *F1 score*, com um aumento não tão relevante em alguns modelos. Logo, essa métrica também confirma o bom desempenho dos modelos *Random Forest* e *LGBM* na identificação de aparelhos elétricos pelo sinal de corrente. Considerando que, assim como nos valores de *F1 score*, esses modelos foram os únicos que alcançaram 100% de acurácia.

Figura 35 – Resultados de acurácia dos modelos testados



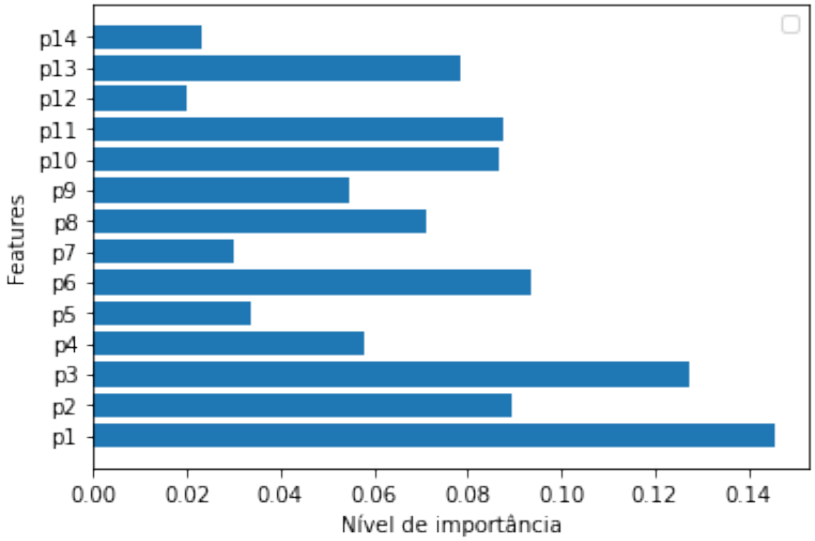
Fonte: Elaborada pelo próprio autor

Contudo, os modelos que alcançaram 99% de *F1 score* e acurácia não podem ser descartados, pois a escolha dos valores de hiper parâmetros ou a forma de divisão do conjunto de dados pode ter desfavorecido alguns modelos. Dito isso, surgiu a necessidade de um experimento mais aprofundado, com variações de hiper parâmetros e utilização de validação cruzada. Desta forma, o resultado deste segundo experimento foi crucial para selecionar o modelo utilizado neste projeto.

Dito isso, o segundo experimento foi realizado utilizando a função *GridSearchCV* da biblioteca *Scikit-Learn*. Desta forma, todos os classificadores foram testados novamente com diferentes combinações de hiper parâmetros, como informado na Subseção 5.3.1. Além disso, foi utilizado o recurso de validação cruzada para divisão do conjunto de dados, com *k-fold* igual a 10. Por fim, segundo a métrica *F1 score*, o melhor resultado foi obtido utilizando o classificador *RandomForestClassifier* com os hiper parâmetros *criterion* igua a *'entropy'* e *n_estimators* igual a 25. Deste modo, foi alcançado 100% de *F1 score* e acurácia no conjunto de dados desenvolvido no projeto, utilizando validação cruzada.

Depois que o classificador com maior taxa de acertos, no contexto deste trabalho, foi encontrado, foram observadas as importâncias de cada *feature*. Como mostra a Figura 36, a primeira amostra, denominada *p1*, correspondente ao vale de referência onda, teve a maior importância nas tomadas de decisões do modelo. Já a *feature p12*, teve a menor importância. Contudo, a maioria das *features* obtiveram importâncias relativamente próximas, considerando a escada do gráfico.

Figura 36 – Importâncias das *features* usando o modelo *RandomForestClassifier*



Fonte: Elaborada pelo próprio autor

6.4 Avaliação da Solução Final

Na avaliação do projeto final, foi gerada uma matriz de confusão mostrada no Quadro 6, que mostra as respostas da solução para cada aparelho. Como pode ser visto, este projeto teve um bom desempenho na maioria dos aparelhos testados. No cenário onde não tinha nenhum aparelho conectado à tomada, houve um aproveitamento de 100%. Isso mostra que, a solução proposta identifica bem quando algum aparelho está conectado à rede elétrica e quando não há conexão. Ademais, alguns aparelhos também tiveram aproveitamento máximo nos pacotes monitorados como o ventilador, o ferro de solda e o liquidificador que não foram confundidos com nenhum outro aparelho.

Embora tenha acontecido alguns erros, os aparelhos: secador de cabelo, lâmpada, sanduicheira e carregador de *smartphone* obtiveram bons números com apenas alguns erros arbitrários, que eram rapidamente corrigidos pela solução. Todavia, a solução não teve o comportamento desejado na identificação da *TV*. Pois, embora tenha acertado a maioria dos casos, teve muita instabilidade no processo, oscilando a resposta entre *TV* e carregador de *notebook*. Vale ressaltar que, na identificação do carregador de *notebook* não houve muitas confusões com a *TV*, possuindo uma taxa de acertos maior que 80%.

Contudo, é importante frisar que a imprecisão no processo de identificação da *TV* pode ser justificado. Isso porque, esse aparelho possui características eletrônicas, necessitando de uma fonte de energia interna semelhante ao carregador de *notebook*. Logo, o sinal de consumo de corrente de ambos os aparelhos possuem um nível de similaridade suficiente para confundir o

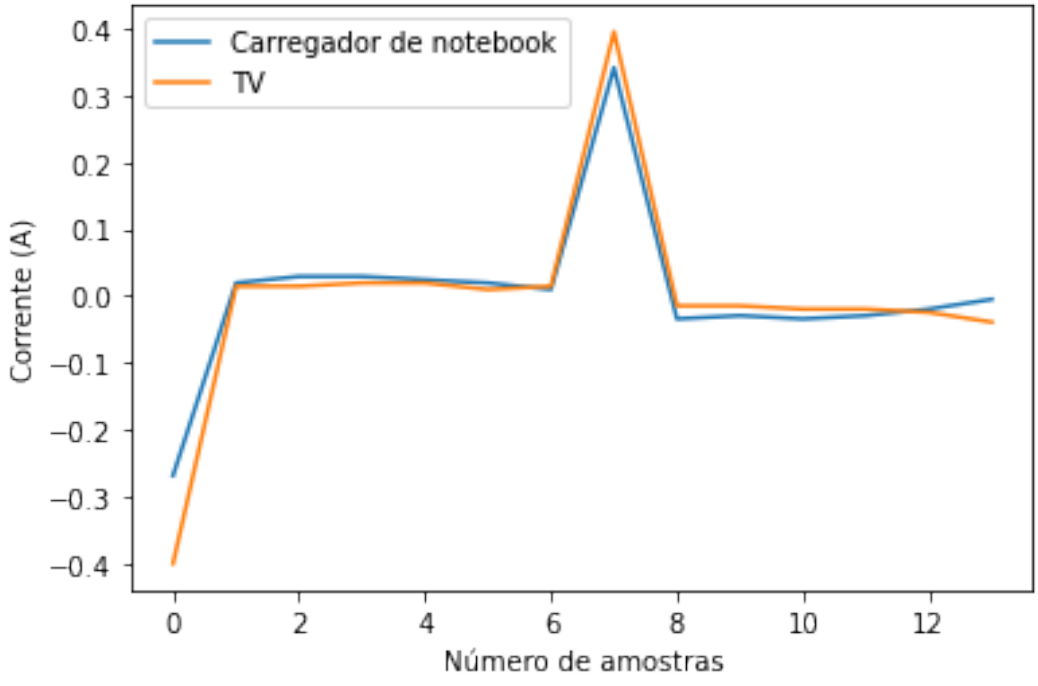
Quadro 6 – Matriz de confusão dos teste reais (as linhas são reais, as colunas são resultados das identificações)

	Desconectado	**Secador de cabelo**	***TV***	**Ventilador**	**Ferro de solda**	**Liquidificador**	**Carregador de *notebook***	**Lâmpada**	**Sanduicheira**	**Carregador de *smartphone***
Desconectado	100	0	0	0	0	0	0	0	0	0
Secador de cabelo	0	92	0	0	0	0	0	0	8	0
TV	0	0	62	0	0	0	38	0	0	0
Ventilador	0	0	0	100	0	0	0	0	0	0
Ferro de solda	0	0	0	0	100	0	0	0	0	0
Liquidificador	0	0	0	0	0	100	0	0	0	0
Carregador de *notebook*	0	0	18	0	0	0	82	0	0	0
Lâmpada	0	0	0	0	0	0	5	86	0	9
Sanduicheira	0	4	0	0	0	0	0	0	96	0
Carregador de *smartphone*	0	0	0	0	0	0	5	3	0	92

Fonte: Elaborado pelo próprio autor.

classificador. A Figura 37, mostra um período dos sinais capturados pelo dispositivo *IoT* dos dois aparelhos em questão.

Figura 37 – Sinais de corrente da *TV* e do carregador de *notebook*



Fonte: Elaborada pelo próprio autor

Como visto, os sinais se assemelham não apenas no formato, mas também nos valores de corrente. Entretanto, mesmo que haja imprecisão para diferenciar os dois equipamentos, a solução consegue identificar que o aparelho elétrico ligado à tomada possui propriedades eletrônicas, não confundindo com aparelhos de propriedades distintas. Logo, a solução teve um comportamento aceitável considerando a avaliação realizada, conseguindo identificar vários aparelhos usuais de uma residência.

7 CONCLUSÕES E TRABALHOS FUTUROS

O presente trabalho aborda a ideia de identificação de aparelhos elétricos por meio do sinal de corrente. Desta forma, com base em referências bibliográficas, avaliou as soluções já existentes usando diferentes trabalhos como parâmetros de avaliação da solução proposta. Além disso, projetou e validou a solução deste o *hardware* até o *software web* de comunicação com o usuário. Deste modo, abrange vários temas estudados no curso de Engenharia de Computação, proporcionando tanto conhecimentos em áreas individuais, quanto na interação entre alguns conteúdos.

Dito isso, este trabalho aborda fundamentos de sistemas embarcados e estudo de sinais elétricos, compreendendo a interligação entre sensores, conversores analógicos-digitais, microcontroladores e módulos de comunicação sem fio para a construção de um dispositivo *IoT* responsável pela aquisição de amostras de corrente. Além disso, abrangeu conhecimentos de sistemas distribuídos e computação em Nuvem no desenvolvimento de uma infraestrutura de serviços executados em Nuvem e Borda utilizando os *GEs* da plataforma *FIWARE*.

Ainda, aplica princípios de Aprendizado de Máquina na classificação dos sinais de corrente, identificando o aparelho monitorado. Por fim, este trabalho emprega noções de desenvolvimento de sistemas *web* na criação de uma *dashboard* responsável por apresentar os resultados de identificação e monitoramento de aparelhos elétricos. Desta forma, todas as partes do projeto foram concluídas no decorrer deste trabalho, proporcionando uma solução com justificativas e embasamentos bibliográficos.

Os resultados apresentados no Capítulo 6 mostram que os estudos e o desenvolvimento da solução teve seus objetivos alcançados com eficácia. Desta forma, este trabalho possibilitou contribuições e incentivo ao desenvolvimento de trabalhos voltados ao monitoramento de recursos energéticos, com foco em ambientes inteligentes, inteligência artificial e desenvolvimento sustentável.

As principais dificuldades encontradas estão relacionadas ao uso de ferramentas recentes, com números relativamente baixos de literatura disponível. Contudo, a solução desenvolvida contribuiu com exemplos de utilização de protocolos e ferramentas diversas, de modo a agregar conhecimentos sobre as tecnologias empregadas. Além de, colaborar com o acervo literário sabre as ferramentas e técnicas aplicadas.

Como trabalhos futuros, pretende-se realizar testes de eficiência de rede utilizando diferentes arquiteturas de comunicação com a Nuvem, como mencionado na Subseção 5.4.2.3.

Além disso, o projeto visa estender a capacidade de identificação da solução, pretendendo reconhecer vários dispositivos conectados simultaneamente a um mesma malha elétrica monitorada. Ainda, detectar outras características como qualidade de funcionamento e controle automático dos aparelhos. Por fim, gerar alerta de alto consumos e funcionamento defeituoso, além de prever custos financeiros com energia elétrica.

REFERÊNCIAS

ABEYKOON, V.; KANKANAMDURAGE, N.; SENEVIRATHNA, A.; RANAWEERA, P.; UDAWALPOLA, R. Real time identification of electrical devices through power consumption pattern detection. **Pervasive Comput**. [*S. l.*], v. 10, n. 1, p. 40–48, 2016.

ABUBAKAR, I.; KHALID, S.; MUSTAFA, M.; SHAREEF, H.; MUSTAPHA, M. Application of load monitoring in appliances' energy management–a review. **Renewable and Sustainable Energy Reviews**, Elsevier, v. 67, p. 235–245, 2017.

ALBERTI, A. M.; SANTOS, M. A.; SOUZA, R.; SILVA, H. D. L. D.; CARNEIRO, J. R.; FIGUEIREDO, V. A. C.; RODRIGUES, J. J. Platforms for smart environments and future internet design: A survey. **IEEE Access**, IEEE, v. 7, p. 165748–165778, 2019.

ALMEIDA, J. S.; MARINHO, L. B.; SOUZA, J. M.; ASSIS, E.; FILHO, P. R. Localization system for autonomous mobile robots using machine learning methods and omnidirectional sonar. **IEEE Latin America Transactions**, IEEE, v. 16, n. 2, p. 368–374, 2018.

AMURIM, A. D.; SILVA, J. I. da; ORTIZ, M. D.; REGO, P. A.; SOUZA, J. N. de. Uma solução de iot baseada no fiware para gerenciamento de recursos energéticos e serviços acadêmicos em um campus universitário. In: SBC. **Anais do V Workshop de Computação Urbana**. [*S. l.*], 2021. p. 265–278.

ARJADI, R. H.; CANDRA, H.; PRANANTO, H. D.; WIJANARKO, T. A. W. *et al.* Rssi comparison of esp8266 modules. In: IEEE. **2018 Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS)**. [*S. l.*], 2018. p. 150–153.

ATLAM, H. F.; ALENEZI, A.; ALHARTHI, A.; WALTERS, R. J.; WILLS, G. B. Integration of cloud computing with internet of things: challenges and open issues. In: IEEE. **2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)**. [*S. l.*], 2017. p. 670–675.

ATLAM, H. F.; WALTERS, R. J.; WILLS, G. B. Fog computing and the internet of things: A review. **big data and cognitive computing**, Multidisciplinary Digital Publishing Institute, v. 2, n. 2, p. 10, 2018.

BARKER, S.; MUSTHAG, M.; IRWIN, D.; SHENOY, P. Non-intrusive load identification for smart outlets. In: IEEE. **2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)**. [*S. l.*], 2014. p. 548–553.

BATISTA, C.; SILVA, P. V.; CAVALCANTE, E.; BATISTA, T.; BARROS, T.; TAKAHASHI, C.; CARDOSO, T.; NETO, J. A.; RIBEIRO, R. A middleware environment for developing internet of things applications. In: **Proceedings of the 5th Workshop on Middleware and Applications for the Internet of Things**. [*S. l.*: *s. n.*], 2018. p. 41–46.

BATTULGA, D.; MIORANDI, D.; TEDESCHI, C. Fogguru: a fog computing platform based on apache flink. In: IEEE. **2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)**. [*S. l.*], 2020. p. 156–158.

CHENG, B.; SOLMAZ, G.; CIRILLO, F.; KOVACS, E.; TERASAWA, K.; KITAZAWA, A. Fogflow: Easy programming of iot services over cloud and edges for smart cities. **IEEE Internet of Things Journal**, IEEE, v. 5, n. 2, p. 696–707, 2017.

CHICCO, D.; JURMAN, G. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. **BMC genomics**, Springer, v. 21, n. 1, p. 1–13, 2020.

CIRILLO, F.; SOLMAZ, G.; BERZ, E. L.; BAUER, M.; CHENG, B.; KOVACS, E. A standard-based open source iot platform: Fiware. **IEEE Internet of Things Magazine**, IEEE, v. 2, n. 3, p. 12–18, 2019.

DETZNER, P.; SALHOFER, P. **Analysing FIWAREs Platform-Potential Improvements**. [*S. l.: s. n.*], 2020.

DEY, S. K.; HOSSAIN, A.; RAHMAN, M. M. Implementation of a web application to predict diabetes disease: an approach using machine learning algorithm. In: IEEE. **2018 21st international conference of computer and information technology (ICCIT)**. [*S. l.*], 2018. p. 1–5.

DIAS, V. H. A.; DIAS, P. M. C. Escrevendo o "livro da natureza" na linguagem da matemática: A lei de ampère. **Revista Brasileira de Ensino de Física**, SciELO Brasil, v. 37, p. 4601–1, 2015.

FARAHZADI, A.; SHAMS, P.; REZAZADEH, J.; FARAHBAKHSH, R. Middleware technologies for cloud of things: a survey. **Digital Communications and Networks**, Elsevier, v. 4, n. 3, p. 176–188, 2018.

FAYZRAKHMANOV, R.; KULIKOV, A.; REPP, P. The difference between precision-recall and roc curves for evaluating the performance of credit card fraud detection models. In: ANHALT UNIVERSITY OF APPLIED SCIENCES. **Proceedings of International Conference on Applied Innovation in IT**. [*S. l.*], 2018. v. 6, n. 1, p. 17–22.

FEITOSA, S. O. *et al.* **Desenvolvimento e aplicabilidade de um multimedidor de grandezas elétricas utilizando plataforma arduíno**. [*S. l.*]: Pesqueira, 2021.

FERREIRA, J. C.; AFONSO, J. A.; MONTEIRO, V.; AFONSO, J. L. An energy management platform for public buildings. **Electronics, Multidisciplinary Digital Publishing Institute**. [*S. l.*], v. 7, n. 11, p. 294, 2018.

FRAGA-LAMAS, P.; CELAYA-ECHARRI, M.; LOPEZ-ITURRI, P.; CASTEDO, L.; AZPILICUETA, L.; AGUIRRE, E.; SUÁREZ-ALBELA, M.; FALCONE, F.; FERNÁNDEZ-CARAMÉS, T. M. Design and experimental validation of a lorawan fog computing based architecture for iot enabled smart campus applications. **Sensors, Multidisciplinary Digital Publishing Institute**. [*S. l.*], v. 19, n. 15, p. 3287, 2019.

GAMESS, E.; SMITH, B. Performance evaluation of tcp and udp over ipv4 and ipv6 for the esp8266 module. In: **Proceedings of the 2020 2nd International Electronics Communication Conference**. [*S. l.*: *s. n.*], 2020. p. 161–169.

GOWTHAM, A.; ANIRUDH, L.; SREEJA, B.; AAKASH, B.; ADITTYA, S. Detection of arrhythmia using ecg waves with deep convolutional neural networks. In: IEEE. **2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA)**. [*S. l.*], 2020. p. 1390–1396.

GRANIK, M.; MESYURA, V. Fake news detection using naive bayes classifier. In: IEEE. **2017 IEEE first Ukraine conference on electrical and computer engineering (UKRCON)**. [*S. l.*], 2017. p. 900–903.

GUO, Y.; ZHOU, Y.; HU, X.; CHENG, W. Research on recommendation of insurance products based on random forest. In: IEEE. **2019 International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)**. [*S. l.*], 2019. p. 308–311.

HESSEL, R.; FRESCHI, A. A.; SANTOS, F. J. d. Lei de indução de faraday: Uma verificação experimental. **Revista Brasileira de Ensino de Física**, SciELO Brasil, v. 37, 2015.

ITTERHEIMOVÁ, P.; FORET, F.; KUBÁŇ, P. High-resolution arduino-based data acquisition devices for microscale separation systems. **Analytica Chimica Acta**, Elsevier, v. 1153, p. 338294, 2021.

KANANI, P.; PADOLE, M. Analyzing ecg waves in fog computing environment using raspberry pi cluster. In: IEEE. **2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)**. [*S. l.*], 2020. p. 1165–1172.

KHAN, M. M. R.; SIDDIQUE, M. A. B.; SAKIB, S. Non-intrusive electrical appliances monitoring and classification using k-nearest neighbors. In: IEEE. **2019 2nd International Conference on Innovation in Engineering and Technology (ICIET)**. [*S. l.*], 2019. p. 1–5.

KODALI, R. K.; SORATKAL, S. Mqtt based home automation system using esp8266. In: IEEE. **2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)**. [*S. l.*], 2016. p. 1–5.

KOLTER, J. Z.; JOHNSON, M. J. Redd: A public data set for energy disaggregation research. In: **Workshop on data mining applications in sustainability (SIGKDD), San Diego, CA**. [*S. l.*: *s. n.*], 2011. v. 25, n. Citeseer, p. 59–62.

LI, Y.; CHENG, J.; WANG, X. An optophone based on raspberry pi and android wireless communication. In: IEEE. **2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)**. [*S. l.*], 2020. p. 952–956.

LIANG, J.; NG, S. K.; KENDALL, G.; CHENG, J. W. Load signature study—part i: Basic concept, structure, and methodology. **IEEE transactions on power Delivery**, IEEE, v. 25, n. 2, p. 551–560, 2009.

LIN, J.; YU, W.; ZHANG, N.; YANG, X.; ZHANG, H.; ZHAO, W. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. **IEEE internet of things journal**, IEEE, v. 4, n. 5, p. 1125–1142, 2017.

MAHESH, B. Machine learning algorithms-a review. **International Journal of Science and Research (IJSR).[Internet]**. [*S. l.*], v. 9, p. 381–386, 2020.

MAHMOUD, R.; YOUSUF, T.; ALOUL, F.; ZUALKERNAN, I. Internet of things (iot) security: Current status, challenges and prospective measures. In: IEEE. **2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)**. [*S. l.*], 2015. p. 336–341.

MARTÍNEZ, R.; PASTOR, J. Á.; ÁLVAREZ, B.; IBORRA, A. A testbed to evaluate the fiware-based iot platform in the domain of precision agriculture. **Sensors, Multidisciplinary Digital Publishing Institute**. [*S. l.*], v. 16, n. 11, p. 1979, 2016.

MESQUITA, J.; GUIMARÃES, D.; PEREIRA, C.; SANTOS, F.; ALMEIDA, L. Assessing the esp8266 wifi module for the internet of things. In: IEEE. **2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)**. [*S. l.*], 2018. v. 1, p. 784–791.

MUNOZ-ARCENTALES, A.; LÓPEZ-PERNAS, S.; POZO, A.; ALONSO, Á.; SALVACHÚA, J.; HUECAS, G. Data usage and access control in industrial data spaces: Implementation using fiware. **Sustainability, Multidisciplinary Digital Publishing Institute**. [*S. l.*], v. 12, n. 9, p. 3885, 2020.

OSISANWO, F.; AKINSOLA, J.; AWODELE, O.; HINMIKAIYE, J.; OLAKANMI, O.; AKINJOBI, J. Supervised machine learning algorithms: classification and comparison. **International Journal of Computer Trends and Technology (IJCTT)**. [*S. l.*], v. 48, n. 3, p. 128–138, 2017.

ÖZCAN, U.; ARSLAN, A.; İLKYAZ, M.; KARAARSLAN, E. An augmented reality application for smart campus urbanization: Msku campus prototype. In: IEEE. **2017 5th International Istanbul Smart Grid and Cities Congress and Fair (ICSG)**. [*S. l.*], 2017. p. 100–104.

PACHECO[1], Á. C.; SOUZA, A. A. de; MARTINS, B. D.; NEVES, D. P.; SILVA, M. F.; SILVA, S. F. de P. **Projeto de um sistema de medição, monitoramento e acionamento remoto de uma carga elétrica**. [*S. l.: s. n.*], 2016.

PAIXAO, A.; CELESTE, W.; JR, L. R.; COURA, D.; ROCHA, H.; RISSINO, S. Classificação inteligente aplicada ao problema de identificação de cargas elétricas "idênticas". **Simpósio Brasileiro de Pesquisa Operacional**, Anais do XLVIII SBPO, 2016.

PHILIPPOU, N.; AJOODHA, R.; JADHAV, A. Using machine learning techniques and matric grades to predict the success of first year university students. In: IEEE. **2020 2nd International Multidisciplinary Information Technology and Engineering Conference (IMITEC)**. [*S. l.*], 2020. p. 1–5.

PLAMANESCU, R.; GHEORGHE, S.; SARB, M.; ISTENES, A.; ALBU, M. A synchronized measurements fiware platform for smart grid applications. In: IEEE. **2019 11th International Symposium on Advanced Topics in Electrical Engineering (ATEE)**. [*S. l.*], 2019. p. 1–5.

PRIHATNO, A. T.; NURCAHYANTO, H.; JANG, Y. M. Predictive maintenance of relative humidity using random forest method. In: IEEE. **2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)**. [*S. l.*], 2021. p. 497–499.

ROCHA, F.; DANTAS, L. C.; SANTOS, L. F.; FERREIRA, S.; SOARES, B.; FERNANDES, A.; CAVALCANTE, E.; BATISTA, T. Energy efficiency in smart buildings: An iot-based air conditioning control system. In: SPRINGER. **IFIP International Internet of Things Conference**. [*S. l.*], 2019. p. 21–35.

ROSLI, R. S.; HABAEBI, M. H.; ISLAM, M. R. Characteristic analysis of received signal strength indicator from esp8266 wifi transceiver module. In: IEEE. **2018 7th International Conference on Computer and Communication Engineering (ICCCE)**. [*S. l.*], 2018. p. 504–507.

SAADATZI, M. N.; TAFAZZOLI, F.; WELCH, K. C.; GRAHAM, J. H. Emotigo: Bluetooth-enabled eyewear for unobtrusive physiology-based emotion recognition. In: IEEE. **2016 IEEE International Conference on Automation Science and Engineering (CASE)**. [*S. l.*], 2016. p. 903–909.

SALHOFER, P.; BUCHSBAUM, J.; JANUSCH, M. Building a fiware smart city platform. In: **Proceedings of the 52nd Hawaii International Conference on System Sciences**. [*S. l.*: *s. n.*], 2019.

SALIM, T. I.; HAIYUNNISA, T.; ALAM, H. S. Design and implementation of water quality monitoring for eel fish aquaculture. In: IEEE. **2016 International Symposium on Electronics and Smart Devices (ISESD)**. [*S. l.*], 2016. p. 208–213.

SEHRAWAT, D.; GILL, N. S. Smart sensors: Analysis of different types of iot sensors. In: IEEE. **2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)**. [*S. l.*], 2019. p. 523–528.

SINGH, A.; THAKUR, N.; SHARMA, A. A review of supervised machine learning algorithms. In: IEEE. **2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)**. [*S. l.*], 2016. p. 1310–1315.

SOLOVEV, D. B.; SHADRIN, A. S. Instrument current transducer for measurements in asymmetrical conditions in three-phase circuits with upper harmonics. **International Journal of Electrical Power & Energy Systems**, Elsevier, v. 84, p. 195–201, 2017.

SRIVASTAVA, P.; BAJAJ, M.; RANA, A. S. Iot based controlling of hybrid energy system using esp8266. In: IEEE. **2018 IEEMA Engineer Infinite Conference (eTechNxT)**. [*S. l.*], 2018. p. 1–5.

TOUZANI, S.; GRANDERSON, J.; FERNANDES, S. Gradient boosting machine for modeling the energy consumption of commercial buildings. **Energy and Buildings**, Elsevier, v. 158, p. 1533–1543, 2018.

TUNDIS, A.; FAIZAN, A.; MÜHLHÄUSER, M. A feature-based model for the identification of electrical devices in smart environments. **Sensors, Multidisciplinary Digital Publishing Institute**. [*S. l.*], v. 19, n. 11, p. 2611, 2019.

VERMA, M.; BHARDWAJ, N.; YADAV, A. K. Real time efficient scheduling algorithm for load balancing in fog computing environment. **Int. J. Inf. Technol. Comput. Sci**. [*S. l.*], v. 8, n. 4, p. 1–10, 2016.

WANG, K.; WANG, Y.; SUN, Y.; GUO, S.; WU, J. Green industrial internet of things architecture: An energy-efficient perspective. **IEEE Communications Magazine**, IEEE, v. 54, n. 12, p. 48–54, 2016.

XU, Q.; ZHANG, J. Pifogbed: A fog computing testbed based on raspberry pi. In: IEEE. **2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)**. [*S. l.*], 2019. p. 1–8.

YAMANOOR, N. S.; YAMANOOR, S. High quality, low cost education with the raspberry pi. In: IEEE. **2017 IEEE Global Humanitarian Technology Conference (GHTC)**. [*S. l.*], 2017. p. 1–5.

YAN, Z.; WEN, H. Comparative study of electricity-theft detection based on gradient boosting machine. In: IEEE. **2021 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)**. [*S. l.*], 2021. p. 1–6.

YI, S.; HAO, Z.; QIN, Z.; LI, Q. Fog computing: Platform and applications. In: IEEE. **2015 Third IEEE workshop on hot topics in web systems and technologies (HotWeb)**. [*S. l.*], 2015. p. 73–78.

ZHANG, S.; LI, X.; ZONG, M.; ZHU, X.; CHENG, D. Learning k for knn classification. **ACM Transactions on Intelligent Systems and Technology (TIST)**, ACM New York, NY, USA, v. 8, n. 3, p. 1–19, 2017.