



**UNIVERSIDADE FEDERAL DO PARÁ  
CAMPUS UNIVERSITÁRIO DE TUCURUÍ  
FACULDADE DE ENGENHARIA ELÉTRICA**

**ALEXANDRE DE ARAUJO PEREIRA**

**APLICAÇÃO DA ASSOCIAÇÃO IOT PARA O APRENDIZADO DE MÁQUINA EM  
BORDA DE MODELOS PREDITIVOS EM UM RASPBERRY PI**

**Tucuruí-PA  
2021**

**ALEXANDRE DE ARAUJO PEREIRA**

**APLICAÇÃO DA ASSOCIAÇÃO IOT PARA O APRENDIZADO DE MÁQUINA EM  
BORDA DE MODELOS PREDITIVOS EM UM RASPBERRY PI**

Trabalho de Conclusão de Curso apresentado à Faculdade de Engenharia Elétrica, Campus Universitário de Tucuruí, Universidade Federal do Pará, como requisito parcial à obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Me. Jefferson Souza Costa.

**Tucuruí-PA  
2021**



**UNIVERSIDADE FEDERAL DO PARÁ**  
**CAMPUS UNIVERSITÁRIO DE TUCURUÍ**  
**FACULDADE DE ENGENHARIA ELÉTRICA**

**TÍTULO: APLICAÇÃO DA ASSOCIAÇÃO IOT PARA O APRENDIZADO DE MÁQUINA EM BORDA DE MODELOS PREDITIVOS EM UM RASPBERRY PI**

**DISCENTE: ALEXANDRE DE ARAÚJO PEREIRA**

**MATRÍCULA: 201433940049**

#	BANCA EXAMINADORA	CONDIÇÃO
1	<i>Me. Jefferson Sousa Costa, FEE/CAMTUC/UFPA</i>	Orientador
2	<i>Dr. Ewerton Ramos Granhen, FEE/CAMTUC/UFPA</i>	Membro
3	<i>Dr. Washington César Braga de Sousa, FEE/CAMTUC/UFPA</i>	Membro

**Data da Defesa: 05/02/2021**

**Hora Início: 14:00**

**Hora Término: 14:50**

Trabalho Escrito (0 a 10 pontos por critério)	Examinador 1	Examinador 2	Examinador 3
Formatação	<b>10,0</b>	<b>9,0</b>	<b>10,0</b>
Linguagem (gramática e semântica)	<b>9,5</b>	<b>8,5</b>	<b>9,0</b>
Conteúdo técnico	<b>10,0</b>	<b>10,0</b>	<b>10,0</b>

Defesa Oral (0 a 10 pontos por critério)	Examinador 1	Examinador 2	Examinador 3
Sequência lógica de apresentação	<b>10,0</b>	<b>10,0</b>	<b>10,0</b>
Administração do tempo	<b>10,0</b>	<b>10,0</b>	<b>10,0</b>
Expressão oral	<b>9,0</b>	<b>9,5</b>	<b>9,5</b>
Domínio do tema	<b>10,0</b>	<b>10,0</b>	<b>10,0</b>

<b>Média por examinador</b>	<b>9,8</b>	<b>9,6</b>	<b>9,8</b>
<b>Média Final</b>	<b>9,7</b>		
<b>Conceito Final</b>	<b>EXCELENTE</b>		

Tucuruí-PA, 05/02/2021.

*Jefferson Souza Costa*  
Orientador

*Ewerton Ramos Granhen*  
Membro

*Washington César Braga de Sousa*  
Membro

## **AGRADECIMENTOS**

Agradeço a meus pais, Davi e Ivaniz, pela infinita dedicação, amor e por me ensinarem bons valores. Agradeço também a minha querida irmã Carla que é exemplo de comprometimento, se colocando sempre à disposição.

Um agradecimento especial ao professor Jefferson, um excelente instrutor que foi meu orientador nesse trabalho, pela paciência, compreensão e valiosas contribuições ao meu desenvolvimento acadêmico durante a faculdade.

## RESUMO

A conectividade entre dispositivos eletrônicos é aplicada na solução de inúmeros problemas, esse fenômeno é tão presente que hoje há inúmeros objetos da Internet das Coisas (IoT) sendo introduzidos nas mais diversas áreas. Um contexto onde a IoT tem potencial é no ambiente residencial, ela é o elemento chave de casas inteligentes, cujo objetivo é promover a seus habitantes conforto, segurança e automatização de alguns processos manuais. Esse trabalho explora a sinergia que existe entre objetos IoT dentro de uma arquitetura publicador-observador, aproveitando essa característica para promover a cooperação indireta deles mediante as publicações. Discutimos o ponto de vista técnico e implementamos uma rede IoT com Hub (observador) nos recursos de hardware de um Raspberry Pi e Nós (publicadores) com placas de desenvolvimento NodeMCU do chip ESP8266 com módulo sensor e atuador, a fim estudar a associação entre esses IoT's e encontrar algum modelo preditivo com aprendizado de máquina para a tomada de decisão. Os modelos preditivos oriundos da associação apontada são um indicativo que a arquitetura proposta permite prever publicações, podendo assim ser usada para os mais diversos fins.

Palavras-chave: Aprendizado de Máquina, Arquitetura Publicador-Observador, Atuadores, Associação, Cooperação Indireta, ESP8266, Internet das Coisas, Modelo Preditivo, NodeMCU, Raspberry Pi, Rede IoT, Sensores, Sinergia.

## LISTA DE FIGURAS

FIGURA 1 – ELEMENTOS DE UM SISTEMA INTEGRADO DE DOMÓTICA .....	13
FIGURA 2 – INFRAESTRUTURA DE UMA HAN.....	14
FIGURA 3 – VISÃO GERAL DO PROCESSO DE ANÁLISE DE DADOS DESDE A ENTRADA ATÉ À SAÍDA FINAL .....	14
FIGURA 4 – EDIFÍCIO INTELIGENTE BASEADO EM IOT COM APRENDIZADO DE MÁQUINA EM BORDA .....	15
FIGURA 5 – ARQUITETURA DE COMUNICAÇÃO DE UMA REDE DE SENSORES SEM FIO .....	16
FIGURA 6 – VISÃO GERAL DOS PRINCIPAIS COMPONENTES DE HARDWARE E SOFTWARE DO NÓ SENSOR .....	17
FIGURA 7 – PILHA DE PROTOCOLOS ZIGBEE VERSUS WI-FI.....	19
FIGURA 8 – TECNOLOGIAS SEM FIO APLICADAS NA IOT DITRIBUIDAS PELA FAIXA DE ALCANCE .....	19
FIGURA 9 – TOPOLOGIA ESTRELA, MALHA E PONTO-A-PONTO .....	20
FIGURA 10 – PADRÃO DE SOLICITAÇÃO-RESPOSTA.....	22
FIGURA 11 – WEBSOCKET <i>HANDSHAKE</i> ENTRE CLIENTE E SERVIDOR .....	23
FIGURA 12 – COMPARAÇÃO ENTRE O <i>POLLING</i> (ESTRATÉGIA DE PESQUISA) E WEBSOCKET.....	23
FIGURA 13 – NODEMCU E INDICAÇÃO DE PINOS.....	26
FIGURA 14 – RASPBERRY PI 1 MODEL B REVISION 2.0 .....	27
FIGURA 15 – INDICAÇÃO DE PINOS DO RASPBERRY PI 3 B .....	28
FIGURA 16 – REPRESENTAÇÃO DE UMA ÁRVORE DE DECISÃO.....	30
FIGURA 17 – HAN COMPOSTA POR OITO ELEMENTOS .....	34
FIGURA 18 – PUBLICAÇÃO DE DADOS DOS IOT’S DA FIGURA 17 NO HUB .....	35
FIGURA 19 – EXEMPLO DE TRÊS ASSOCIAÇÕES IOT POSSÍVEIS PARA A HAN DA FIGURA 17.....	36
FIGURA 20 – MODELO GENÉRICO DA HAN .....	38
FIGURA 21 – SISTEMA DE CONTROLE DE ILUMINAÇÃO COM SENSOR DE PRESENÇA. PRESENÇA DE DETECTADA (A), LUZES ACENDEM E PRESENÇA NÃO DETECTADA (B), LUZES APAGAM.....	39
FIGURA 22 – ESQUEMA ELÉTRICO PROJETADO .....	40
FIGURA 23 – PRIMEIRA ETAPA DA INTERAÇÃO DO HABITANTE COM O SISTEMA DE ILUMINAÇÃO IOT .....	41
FIGURA 24 – SEGUNDA ETAPA DA INTERAÇÃO DO HABITANTE COM O SISTEMA DE ILUMINAÇÃO IOT .....	42
FIGURA 25 – FUNCIONAMENTO DOS SENSORES PIR.....	44
FIGURA 26 – ESQUEMA INTERNO DO SENSOR PIR .....	44
FIGURA 27 – ESQUEMA DO MÓDULO SENSOR PIR.....	45
FIGURA 28 – MÓDULO HC-SR501, VISÃO SUPERIOR (A) E INFERIOR (B).....	45

<b>FIGURA 29 – COMPORTAMENTO DA LENTE DE FRESNEL DO HC-SR501 .....</b>	<b>46</b>
<b>FIGURA 30 – ESQUEMA DE LIGAÇÃO DO NÓ IOT-SP .....</b>	<b>47</b>
<b>FIGURA 31 – FUNCIONAMENTO DO FIRMWARE DO IOT-SP .....</b>	<b>48</b>
<b>FIGURA 32 – IOT-SP EM OPERAÇÃO .....</b>	<b>49</b>
<b>FIGURA 33 – CARACTERÍSTICAS DE UM RELÉ .....</b>	<b>50</b>
<b>FIGURA 34 – CIRCUITO ACIONADOR DE RELÉ .....</b>	<b>51</b>
<b>FIGURA 35 – OPTOISOLADOR.....</b>	<b>51</b>
<b>FIGURA 36 – DIAGRAMA DO CIRCUITO DETECTOR DE TENSÃO.....</b>	<b>52</b>
<b>FIGURA 37 – CTR POR <math>I_F</math> .....</b>	<b>53</b>
<b>FIGURA 38 – ESQUEMA DE LIGAÇÃO DO NÓ IOT-L .....</b>	<b>54</b>
<b>FIGURA 39 – FUNCIONAMENTO DO FIRMWARE DO IOT-L .....</b>	<b>55</b>
<b>FIGURA 40 – IOT-L EM OPERAÇÃO.....</b>	<b>56</b>
<b>FIGURA 41 – SCRIPT PYTHON DA SIMULAÇÃO CORRENDO NO RASPBAN EMULADO POR QEMU.....</b>	<b>57</b>
<b>FIGURA 42 – CODIFICAÇÃO DE DADOS NO CONTEÚDO DAS PUBLICAÇÕES .....</b>	<b>58</b>
<b>FIGURA 43 – CASOS DE USO PARA O DESENHO DA INTERFACE .....</b>	<b>59</b>
<b>FIGURA 44 – INTERFACE DA TELA PRINCIPAL DO HUB .....</b>	<b>59</b>
<b>FIGURA 45 – INTERFACE DE ACOMPANHAMENTO DA CONEXÃO DOS DISPOSITIVOS .....</b>	<b>60</b>
<b>FIGURA 46 – INTERFACE DE ACOMPANHAMENTO DOS MODELOS PREDITIVOS .....</b>	<b>60</b>
<b>FIGURA 47 – INTERFACE COM O HISTÓRICO DE PUBLICAÇÕES.....</b>	<b>61</b>
<b>FIGURA 48 – INTERFACE PARA TESTES E EXPERIMENTOS.....</b>	<b>61</b>
<b>FIGURA 49 – MÉTODO DE ILAÇÃO DOS MODELOS PREDITIVOS .....</b>	<b>62</b>
<b>FIGURA 50 – RASPBERRY PRODUZINDO O SERVIÇO DE HUB .....</b>	<b>63</b>
<b>FIGURA 51 – AP DA RESIDÊNCIA .....</b>	<b>64</b>
<b>FIGURA 52 – PRIMEIRA ETAPA DA INTERAÇÃO DO HABITANTE COM O SISTEMA DE ILUMINAÇÃO IOT APÓS O APRENDIZADO DO MODELO PREDITIVO.....</b>	<b>66</b>
<b>FIGURA 53 – SEGUNDA ETAPA DA INTERAÇÃO DO HABITANTE COM O SISTEMA DE ILUMINAÇÃO IOT APÓS O APRENDIZADO DO MODELO PREDITIVO.....</b>	<b>67</b>
<b>FIGURA 54 – DISPOSITIVOS CONECTADOS AO MODEM .....</b>	<b>69</b>
<b>FIGURA 55 – INFORMAÇÕES SOBRE AS CONEXÕES DE REDE DO RASPBERRY PI.....</b>	<b>69</b>
<b>FIGURA 56 – ARQUIVO DE SAÍDA DO ALGORITMO DE APRENDIZADO .....</b>	<b>71</b>
<b>FIGURA 57 – MODELO DE PREVISÃO OBTIDOS COM O HABITANTE REAL .....</b>	<b>71</b>

## **LISTA DE QUADROS**

<b>QUADRO 1 – MODELOS DE RASPBERRY PI .....</b>	<b>27</b>
<b>QUADRO 2 – APRESENTAÇÃO DOS IOT'S DESFECHO E PREDITOR.....</b>	<b>37</b>
<b>QUADRO 3 – ATRIBUTO ADICIONAL INSERIDO NA REGRA DE ASSOCIAÇÃO .....</b>	<b>68</b>

## **LISTA DE TABELAS**

<b>TABELA 1 – CONJUNTO DE TREINAMENTO COLETADO .....</b>	<b>70</b>
<b>TABELA 2 – PUBLICAÇÕES DO EXPERIMENTO .....</b>	<b>91</b>
<b>TABELA 3 – PUBLICAÇÕES OBTIDAS COM O HABITANTE REAL .....</b>	<b>93</b>

# SUMÁRIO

<b>1 INTRODUÇÃO</b> .....	<b>9</b>
1.1 TRABALHOS CORRELATOS.....	10
1.2 PROPOSTA.....	11
1.3 ORGANIZAÇÃO DO TRABALHO.....	12
<b>2 METODOLOGIA</b> .....	<b>13</b>
2.1 INFRAESTRUTURA DOMÓTICA COM IOT.....	13
2.2 TECNOLOGIAS DA IOT.....	15
2.2.1 Rede de Sensores sem Fio.....	15
2.2.2 Procolos de rede.....	18
2.2.3 Wi-Fi.....	20
2.2.4 HTTP e WebSocket.....	21
2.3 SISTEMAS EMBARCADOS.....	24
2.3.1 ESP8266.....	25
2.3.2 Raspberry Pi.....	26
2.4 APRENDIZADO DE MÁQUINA.....	28
2.4.1 Árvores de decisão.....	30
2.4.2 Algoritmo C4.5.....	33
<b>3 DESENVOLVIMENTO</b> .....	<b>34</b>
3.1 ARQUITETURA PUBLICADOR-OBSERVADOR.....	34
3.2 ESTRUTURA DA HAN.....	37
3.3 DOMÓTICA DA ILUMINAÇÃO.....	39
3.4 NÓ IOT-SP.....	43
3.5 NÓ IOT-L.....	50
3.6 HUB.....	56
3.6.1 Teste prévio.....	57
3.6.2 Implementação.....	58
3.7 PONTO DE ACESSO.....	63
<b>4 RESULTADOS EXPERIMENTAIS</b> .....	<b>65</b>
4.1 PESQUISA EXPERIMENTAL.....	65
4.2 EXPERIMENTO FINAL.....	68
<b>CONCLUSÃO</b> .....	<b>73</b>
<b>REFERÊNCIAS</b> .....	<b>74</b>
<b>ANEXOS</b> .....	<b>78</b>
ANEXO 1 – CIRCUITO DO HC-SR501.....	78
ANEXO 2 – SKETCH DESENVOLVIDO PARA IOT-SP.....	79
ANEXO 3 – SKETCH DESENVOLVIDO PARA IOT-L.....	85
ANEXO 4 – DADOS DOS EXPERIMENTOS.....	91
ANEXO 5 – DADOS DO EXPERIMENTO FINAL.....	93

## 1 INTRODUÇÃO

As tecnologias ligadas a automação e conectividade estão cada vez mais presentes nas residências dado seu reduzido tamanho, multi-funcionalidades e baixo custo. Esse gênero de inovação está a frente do método convencional e ganha espaço em vários nichos de mercado (MANAVALAN e JAYAKRISHNA, 2019; SWAROOP, 2019; SICARI, 2019). Sistemas eletrônicos com algoritmos estão substituindo todo tipo de trabalho humano. Um termo muito utilizado para se referir a esse tema é domótica, junção entre doméstico e informática na busca da remoção de toda a interação humana tecnicamente possível e desejável do lar, fazendo isso de forma confiável (DENNIS, 2013).

A domótica de um casa compreende desde um agrupamento simples de controles de dispositivos pontuais até o nível altamente automatizado, onde qualquer aparelho eletroeletrônico pode ser controlado. No mercado existem muitos gêneros de produtos de controle remoto ou mesmo comando de voz para controle de segmentos da casa. Mas os sistemas de automação para residências geralmente são caros devido aos custos com equipamentos, componentes e instalação personalizada, além disso, é passível de pouca ou nenhuma alteração após a instalação (VUJOVIĆ e MAKSIMOVIC, 2015).

Diversos fatores da nossa sociedade moderna vem desencadeando o desenvolvimento da tecnologia domótica como a mudança demográfica, o envelhecimento da sociedade e um aumento na demanda por praticidade no ambiente doméstico. Cada vez mais se busca por melhores dispositivos e sensores que possam implementar soluções domésticas inteligentes.

A ascensão da domótica é fruto de um revolução tecnológica que visa conectar à Internet todo tipo de objeto, a *Internet of Things* (IoT). No paradigma da IoT dispositivos são incorporados em ambientes e se comunicam entre si através de tecnologias sem fio. Esses dispositivos limitados, com sensores e atuadores, conhecidos como ‘coisas’, são utilizados para transmitir e receber informações, recolher, armazenar e tratar os dados, sensoriando e atuando (AL-TURJMAN e IMRAN, 2020). O aperfeiçoamento da *computação pervasiva* em união com sensores, atuadores e aprendizado de máquina também são fatores que corroboram com a idéia de ambientes inteligentes (SZEWCYZK, 2009).

Quando se trata da IoT, muitas vezes é necessário identificar correlações entre dezenas de entradas de sensores e fatores externos, pois modelos clássicos de dados, que são muitas vezes estáticos e de escalabilidade limitada, não podem ser aplicados a dados em rápida mudança, crescimento em volume e não estruturados. É por isso que o *Machine Learning*

(Aprendizado de Máquina) é aplicado na IoT. Aprendizado de Máquina utiliza algoritmos projetados à interpretar dados de entrada para prever valores de saída. À medida que sensores IoT coletam dados, um ‘agente’ analisa, processa e classifica os dados e garante que a informação é enviada de volta ao dispositivo para uma melhor tomada de decisão. Um carro autônomo, por exemplo, têm de tomar decisões rápidas por conta própria sem qualquer intervenção humana. É aí que entra em jogo o aprendizado de máquina (AGRAWAL; PAPRZYCKI; GUPTA, 2020).

Neste contexto, a proposta deste trabalho é desenvolver o IoT de um pequeno sistema de iluminação e estudar a associação entre seus dispositivos por direcionar os dados a um agente central que aplicará algoritmo de aprendizado de máquina a fim de aprender modelos preditivos que prevejam as publicações dos IoT’s conforme o comportamento dos habitantes, eliminando a tarefa de acionar manualmente o sistema de iluminação ao entrar ou sair do ambiente.

## 1.1 TRABALHOS CORRELATOS

Esta seção descreve alguns estudos relacionados a domótica. Veremos que o ambiente físico é ajustado como ambiente inteligente através de agentes capazes de examinar o espaço a sua volta e atuar sobre seu comportamento com o objetivo de tirar o melhor proveito possível das condições atuais (COOK e DAS, 2004). Alguma tática lógica é requerida em modelos centrados em conceitos como esse. Informações a respeito do contexto vinda de sensores e raciocínio sobre as situações percebidas em um controlador traduzem os processos de inferência necessários.

Com um Sistema Adaptativo de Inferência baseado em Neuro-Fuzzy, em Reena e Venkatesh (2018), decisões são tomadas acerca das mudanças que ocorrem na unidade de detecção, controlando e monitorando temperatura, luminosidade e presença humana. A essa sistemática deram o nome de Sistema de Suporte a Decisão Inteligente.

Dados de consumo de energia elétrica vem sendo alvo de pesquisas que analisam essas informações colidas em ferramentas de aprendizado de máquina, afim de ajustar apropriadamente o gasto energético as necessidades dos consumidores. Desses dados também se detecta possíveis valores de consumo que desviam-se do padrão ou valor esperado. Uma abordagem proposta por Geraldo Filho et al. (2018) melhora esse conceito através da integração de um sistema de tomada de decisão inteligente com base em correlações temporais. É obtida alta precisão na tomada de decisão com a ajuda de redes neurais artificiais. O trabalho adota

uma rede de sensores sem fio de baixo custo, fácil implantação e flexibilidade para adicionar dispositivos, em uma integração e escalabilidade dinâmica.

Os sistemas com reconhecimento de contexto geralmente usam mecanismos de raciocínio baseados em regras para a tomada de decisões sem envolver uma interação explícita com o usuário. Vrbaski, Petriu e Amyot (2012) propõem uma abordagem de raciocínio sensível ao contexto chamada *Context-Aware Reasoning* usando *Goal-Oriented* (CARGO) que combina raciocínio baseado em regras e objetivos, bem como modelagem baseada em cenários para fornecer uma maneira mais abrangente de definir sistemas sensíveis ao contexto e processar informações contextuais.

Lee et al. (2014) propõem um sistema sensível ao contexto para ambientes residenciais onipresentes. Para tanto, utiliza dados fornecidos por sensores para analisar padrões e gerar regras, inferindo assim as preferências dos moradores. O processo de aprendizagem pode ser usado para interagir com o ambiente monitorado por meio do *feedback* do usuário. O objetivo disso é obter novos padrões por meio de um processo de autodescoberta por meio do ambiente monitorado. Como resultado, os aplicativos poderão intercomunicar e, assim, melhorar a tomada de decisões. Porém a estrutura apresentada não foi completamente verificada e implementada em contextos residenciais reais.

Xie e Xu (2018) apresenta um método de decisão para os comportamentos dos usuários, baseado no algoritmo C4.5, selecionando amostras de dados coletados pelo sistema doméstico inteligente, analisando e processando amostras para gerar uma árvore de decisão. De acordo com os hábitos de vida do usuário, a árvore de decisão é usada para ajudar os usuários a tomar decisões fornecendo aos usuários serviços mais inteligentes e eficientes.

## 1.2 PROPOSTA

O presente trabalho tem como objetivo estudar a associação de um modelo de arquitetura IoT, implementando os nós IoT's de um sistema doméstico de iluminação e um Hub que encapsula um algoritmo fundamentado em aplicar aprendizado de máquina sobre dados para o controle de processos residenciais simples. A implementação fornecerá uma documentação dessa arquitetura IoT que demonstra de forma prática as vantagens de se utilizar aprendizado de máquina em borda.

No trabalho de Tonidandel e Sgarbi (2006) alguns sensores e um atuador são simulados, produzindo dados que traduzem o comportamento de um habitante em um cômodo

da casa. Com os dados gerados eles produzem regras de classificação para o atuador inserido no cômodo, usando aprendizado de máquina. A obra consiste, sobretudo, em produzir, tratar, manter e aplicar as regras de classificação. Esse sistema se assemelha muito a metodologia apresentada nos trabalhos de Lee et al. (2014) e Xie e Xu (2018) com o diferencial da manutenção das regras de classificação e da janela de tempo entre publicações de sensores e atuador. Partindo dessa idéia, propomos nossa arquitetura IoT, que primeiro, viabiliza a comunicação, recolhimento e agrupamento de dados, e segundo, introduza o método apresentado em um Hub que passe a servir os nós da rede IoT com previsões vantajosas.

Demonstramos essa arquitetura na prática por tornar nosso sistema de iluminação doméstico adaptável às exigências do morador e às condições do ambiente controlado. Essa domótica é desenvolvida com os preceitos da IoT por meio de embarcados, os quais servem como *hardware* de processamento e comunicação que tem como base tecnologias *open source* com custo de desenvolvimento baixo (ORTIZ, 2018).

### 1.3 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado em quatro capítulos, incluindo esta introdução que apresentada uma visão geral da domótica e de trabalhos relevantes de aprendizagem e raciocínio relacionados com o presente trabalho.

O Capítulo 2 apresenta uma revisão de literatura realizada para o projeto, descrevendo brevemente as principais características da domótica e a base conceitual utilizada no que se refere aos fundamentos técnicos e tecnológicos que compõem o sistema.

No Capítulo 3 é apresentada a implementação dos principais elementos considerados neste trabalho. É feita a descrição de todas as etapas estabelecidas para o desenvolvimento da arquitetura proposta, da rede e dos dispositivos.

No Capítulo 4 se encontra a descrição dos testes realizados e apresentação dos resultados obtidos, conforme descrito nos objetivos do trabalho.

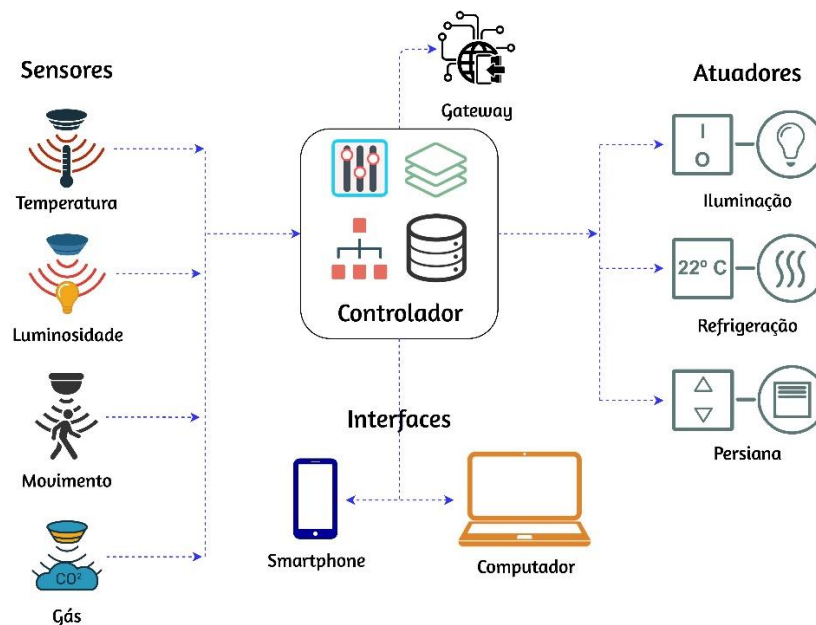
Ao final se expõe as conclusões deste projeto sobre os resultados obtidos e as dificuldades encontradas, e apresentam-se, também, sugestões de melhorias.

## 2 METODOLOGIA

### 2.1 INFRAESTRUTURA DOMÓTICA COM IOT

Em um arranjo simplificado vemos que um sistema domótico transporta os valores de entrada disponibilizados por sensores, em direção a um controlador que armazena a informação e a torna disponível a uma ou mais saídas tal como interfaces e atuadores (figura 1).

Figura 1 – Elementos de um sistema integrado de domótica



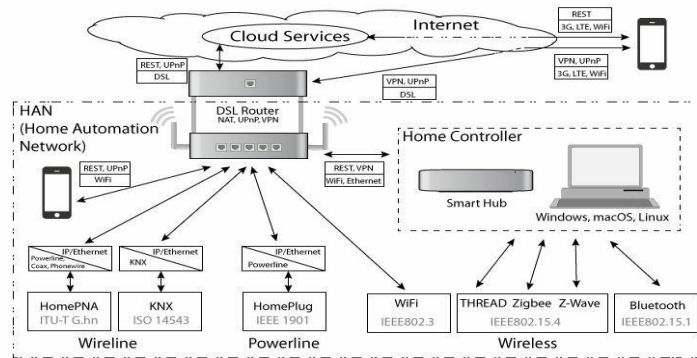
Fonte: Adaptado de Moreno (2017).

Nesse meio há uma rede implícita que conecta todos os dispositivos e pelo qual as informações atravessam, a *Local Area Network* (LAN), sendo ela rede sem fio *Wi-Fi* e rede cabeada *Ethernet*. A comunicação entre aparelhos diferentes é alcançada por obedecer a algum protocolo que normaliza os dados. E para uma conectividade mais ampla o *gateway* gerencia a troca de dados entre diferentes LAN's ou entre a LAN e a Internet (MORENO, 2017). A Internet tem uma amplitude única e nós temos fácil acesso a sua dimensão.

A nível de IoT o controlador adquire novas responsabilidades, atuando com mais interação, processamento e hibridização de recursos, a esse aprimorado controlador dá-se o nome de *Hub*. Ele opera por comandos do habitante e/ou por um conjunto de regras predefinidas, suporta adaptadores de interface para dispositivos que utilizem diferentes tecnologias de rede como *Z-Wave*, *Bluetooth* e *Zigbee*, e pode ser inteligente sendo formado

por hardware e software especializados ou fundamentar-se em *cloud services* (figura 2). Quanto a LAN passa a chamar-se de *Home Automation Network* (HAN), termo que designa especificamente a rede residencial local adotada com domótica (KYAS, 2017).

Figura 2 – Infraestrutura de uma HAN

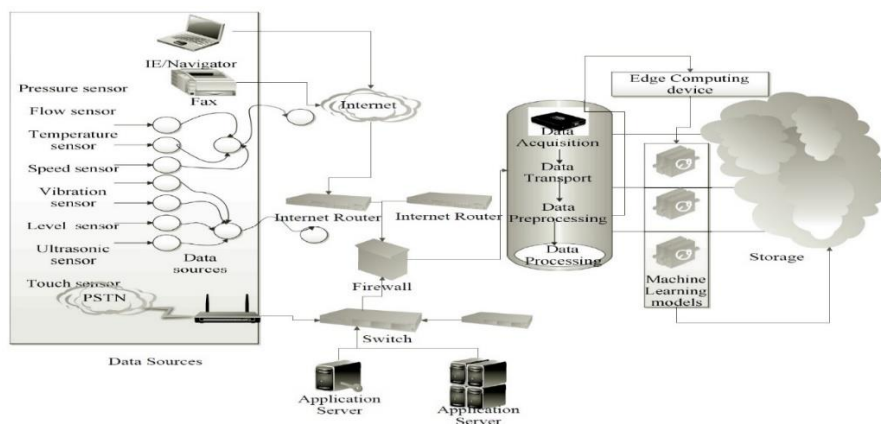


Fonte: Kyas (2017).

O cloud services são plataformas online que gerenciam aplicações IoT e propiciam um meio seguro de controlar dispositivos domésticos remotamente pela Internet estando em ambiente externo a residência. O protocolo de aplicação adotado muitas vezes é o HTTPS REST, o mesmo que fabricantes de Hub como *Samsung SmartThings*, *Wink Labs* e *Google NEST* usam (KYAS, 2017).

É propício assimilar o imenso volume de dados da HAN com aprendizado de máquina refinando as informações em modelos para a arquitetura IoT (figura 3).

Figura 3 – Visão geral do processo de análise de dados desde a entrada até à saída final



Fonte: Kumar e Makkar (2020).

A *edge computing* (computação em borda) integra as capacidades de inteligência, processamento e comunicação próximo à borda, diretamente em dispositivos controladores da rede. O resultado é uma baixa latência, consumo mínimo de largura de banda e integridade de dados (KUMAR e MAKKAR, 2020).

A integração do aprendizado de máquina a HAN não só otimiza o controle, mas sobretudo, faz com que o sistema inteiro não só passe a entender o ambiente como também a tomar decisões.

Figura 4 – Edifício inteligente baseado em IoT com aprendizado de máquina em borda



Fonte: Huang e Yu (2019).

Em um edifício inteligente baseado em IoT um centro de dados recolhe dados dos sensores e os envia para gateways inteligentes (figura 4). Algoritmos de aprendizado de máquina serão executados diretamente nos gateways inteligentes localmente para efetuar respostas em tempo real e garantir a privacidade dos dados (HUANG e YU, 2019).

## 2.2 TECNOLOGIAS DA IOT

### 2.2.1 Rede de Sensores sem Fio

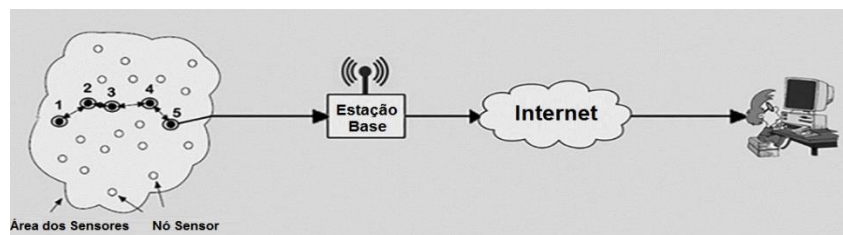
Considerada a próxima geração de rede a IoT tem aplicações práticas em muitos campos por realizar tarefas simples e complexas, incorporando e comunicando dados. Parte da ideia, de construir essa rede, consiste em introduzi-la nos objetos da vida real, algumas aplicações específicas em andamento, como, casa e tráfego inteligente provaram sua viabilidade (BINH e DEY, 2018).

A IoT é empregada através de redes sem fio devido ao desenvolvimento da Internet, das radiocomunicações e da tecnologia da informação. O desejo humano por conectividade é uma das causas desse crescimento, observa-se uma crescente troca de dados em serviços da Internet, como a *World Wide Web*, e-mail, transferências de arquivos de dados, redes sociais, e-commerce e muitos outros. A convergência das tecnologias de Internet, aliada aos recentes avanços da engenharia, também abriu caminho para uma nova geração de sensores e atuadores econômicos e com alta resolução e precisão (SOHRABY, 2007).

Redes desses sensores com a tecnologia sem fio são capazes de trabalhar de forma eficaz nos diferentes campos de trabalho da IoT. Quando acoplados a baterias esses dispositivos são colocados inclusive em campos abertos sem assistência humana para adquirir dados continuamente ao longo do tempo (ELHOSENY e HASSANIEN, 2019).

Uma coleção de nós de sensores individuais e subdimensionados em uma arquitetura de computação e comunicação é chamada de rede de sensores sem fio (RSSF), onde um nó é conectado com nós adicionais. Cada nó sensor em uma RSSF observa fenômenos ambientais, as informações coletadas são transmitidas para uma ou mais estações centrais através de algum *link* sem fio dependendo da implementação da rede (BINH e DEY, 2018). Esses nós atendem aos requisitos específicos de cada aplicação: eles podem ser pequenos, baratos ou eficientes em termos de energia, precisam estar equipados com os sensores certos, os recursos necessários de computação e memória, e precisam de meios de comunicação adequados (KARL e WILLIG, 2007).

Figura 5 – Arquitetura de comunicação de uma rede de sensores sem fio



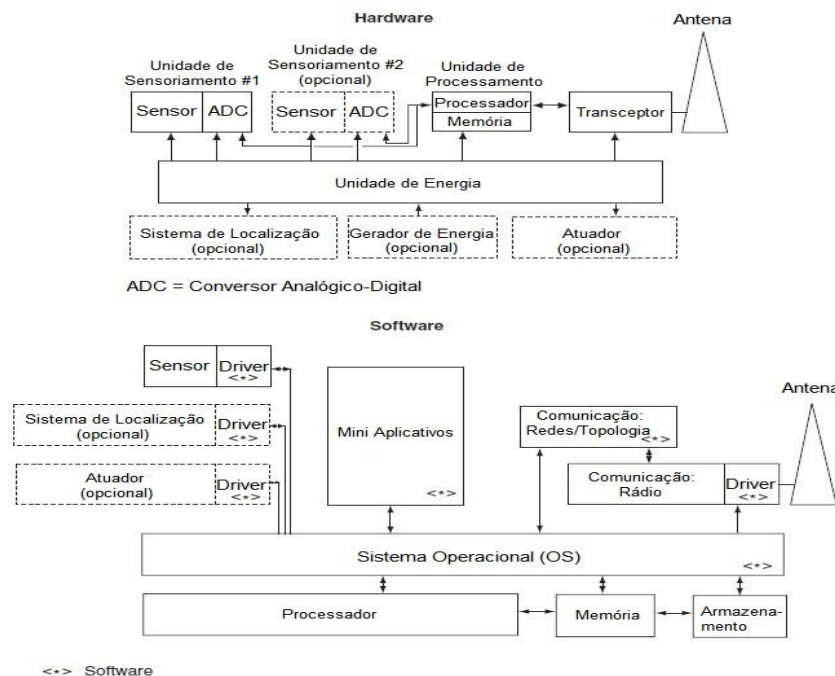
Fonte: Binh e Dey (2018).

Os nós de uma RSSF cobrem uma área geográfica em termos de algum parâmetro medido (figura 5) executando alguma lógica para processamento de sinal, gerenciamento de topologia e manipulação de transmissão (SOHRABY, 2007). As funcionalidades do nó sensor são detectar a atmosfera vizinha, processar os dados que foram inicialmente observados e estar se comunicando ou conectando através de nós adicionais ou estações centrais dentro da rede.

As RSSFs podem ser estruturadas de maneira centralizada, distribuída e ad hoc (BINH e DEY, 2018).

Um nó básico compreende cinco subsistemas básicos de hardware, são eles: controlador, memória, sensores e atuadores, comunicação e fonte de alimentação (figura 6). O controlador usa lógica computacional para processar todos os dados relevantes, executando código arbitrário, os requisitos computacionais variam de um microcontrolador a um microprocessador. A memória armazena programas e dados, os nós possuem requisitos de armazenamento que geralmente variam de bytes a gigabytes. Sensores e atuadores são a interface entre o ambiente e a rede sem fio, podendo observar ou controlar parâmetros físicos do ambiente. A comunicação transforma os nós em uma rede que envia e recebe informações através de um canal sem fio a distâncias que variam de metros a quilômetros e os protocolos de comunicação de camada inferior tendem a ser da classe IEEE 802.11/802.15/802.16. A fonte de alimentação é um circuito terminal ou bateria, às vezes, também usada alguma forma de recarga, obtendo energia do ambiente, como por exemplo as células solares (SOHRABY, 2007).

Figura 6 – Visão geral dos principais componentes de hardware e software do nó sensor



Fonte: Sohraby (2007).

Por conseguinte, os cinco subsistemas básicos de software são: microcódigo do sistema operacional (OS), drivers de sensores, processadores de comunicação, drivers de

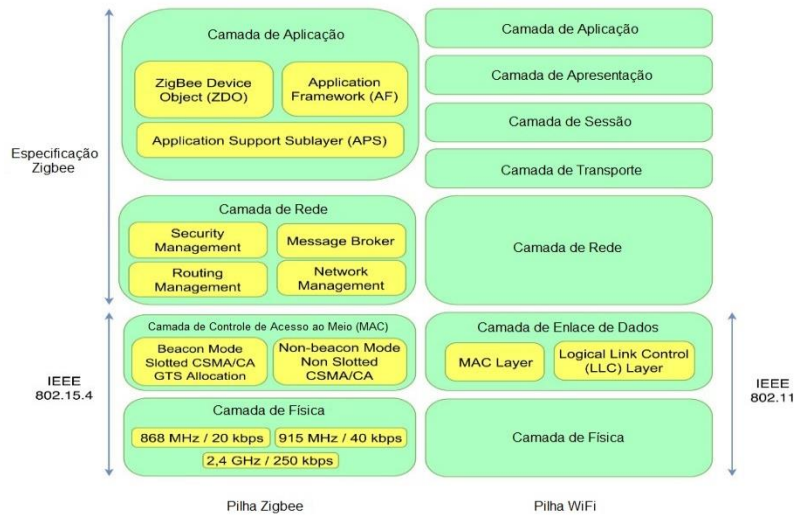
comunicação, mini aplicativos de processamento de dados. Também chamado de *middleware*, o microcódigo do OS é usado por todos os módulos de software de alto nível para suporte a várias funções, a finalidade é proteger o software funcional do nível de máquina do microprocessador. Os drivers de sensores são os módulos de software que gerenciam as funções básicas dos transceptores de sensores. O processador de comunicação é um código gerencia as funções de comunicação, como roteamento, encaminhamento de pacotes e criptografia. Os drivers de comunicação são módulos de software que gerenciam as minúcias do link de transmissão do canal de rádio, incluindo clock e sincronização, codificação de sinal e modulação. Mini-aplicativos de processamento de dados envolvem processamento de dados numéricos, armazenamento e manipulações de valor de sinal ou outros aplicativos básicos que são suportados no nível do nó (SOHRABY, 2007).

### 2.2.2 Protocolos de rede

O conceito abstrato de comunicação vem da percepção intuitiva de como o usamos na conversa cotidiana, bem como do conjunto de entidades fisicamente quantificáveis que são usadas para comunicação no sentido técnico. Mas quando falamos de meios de comunicação entre dispositivos, nos referimos em primeiro lugar, a componentes de hardware como fios, transmissores e receptores. Em segundo lugar, a natureza da transmissão com a corrente elétrica ou onda eletromagnética. E em terceiro, a como as informações devem ser interpretadas. A comunicação segue convenções e padrões individuais dessas três camadas. Isso possibilita, que desenvolvedores de software se concentrem na interpretação de dados e em sua representação digital, enquanto engenheiros de hardware se preocupam com a informação em sinais elétricos (ADRYAN; OBERMAIER; FREMANTLE, 2017).

Camadas de processamento em série conhecidas como pilha de protocolo foram elaboradas para organizar as entidades envolvidas na comunicação. Os dados recebidos vêm do meio físico e são processados sequencialmente por cada camada do mais baixo ao mais alto. As camadas mais altas enviam dados para as camadas mais baixas para serem transmitidos através do meio físico. A maioria dos sistemas de comunicação, incluindo os protocolos para a IoT sem fio, são organizados em uma pilha de protocolo, como a pilha *Zigbee* ou a pilha *Wi-Fi* (figura 7). Tal arquitetura em camadas fornece um meio natural para decompor as várias funções dentro de um sistema de comunicações. Três das mais famosas pilhas de protocolo são os o modelos de referência OSI, TCP/IP e IEEE 802 (ADRYAN; OBERMAIER; FREMANTLE, 2017).

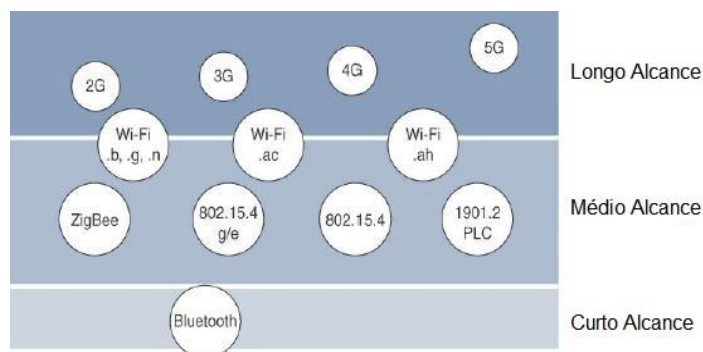
Figura 7 – Pilha de protocolos Zigbee versus Wi-Fi



Fonte: Froiz-Míguez (2018).

Dependendo da implementação IoT, são usadas tecnologias de alcance curto, médio ou longo nas camadas mais baixas da pilha de protocolo (figura 8). Em curto alcance a distância máxima entre dois dispositivos é de algumas dezenas de metros, sendo encontrados em minoria nas instalações IoT, como exemplos temos o *Bluetooth* (IEEE 802.15.1) e *VLC* (IEEE 802.15.7). O médio alcance é onde muitas especificações e implementações da IoT estão, na faixa de dezenas a centenas de metros, com distância máxima entre dispositivos menor que 1 milha, são exemplos *Wi-Fi* (IEEE 802.11) e *WPAN* (IEEE 802.15.4/802.15.4g) usado por *Zigbee*. Exige-se tecnologias de longo alcance quando a distância entre dispositivos IoT é maior que 1,6 km, são exemplos as redes sem fio de telefonia 2G, 3G, 4G e 5G e *Low-Power Wide-Area* (LPWA) que como o nome diz têm a capacidade de se comunicar em uma grande área consumindo pouca energia (HANES, 2017).

Figura 8 – Tecnologias sem fio aplicadas na IoT ditribuidas pela faixa de alcance



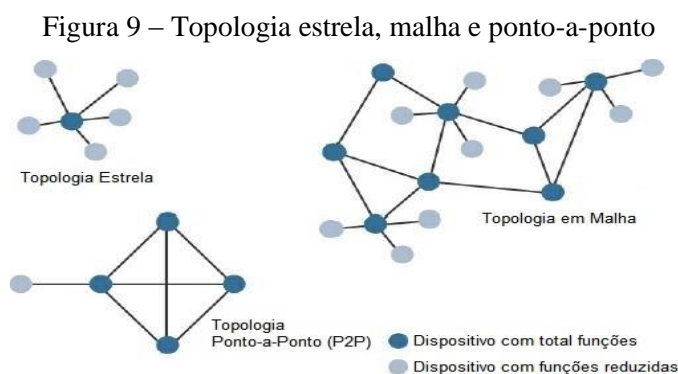
Fonte: Hanes (2017).

Tecnologias como Bluetooth e Wi-Fi, têm sido usadas para controlar dispositivos em instalações de domótica, pois fornecem uma maneira fácil de se comunicar com smartphones, tablets e PCs, mas existe uma rica variedade de protocolos e tecnologias nesse campo. São padrões *abertos* projetados por instituições internacionais e *proprietários* desenvolvidos por empresas, cada um deles apresenta diferentes vantagens e desvantagens, dependendo do cenário implantado, alguns deles são KNX-EIB, LonWorks, X10, Insteon, ModBus, BACnet, Z-Wave, ZigBee e EnOcean (FROIZ-MÍGUEZ, 2018).

### 2.2.3 Wi-Fi

O *Wireless Fidelity*, ou simplesmente Wi-Fi, é uma marca registrada da *Wi-Fi Alliance* que é uma organização que certifica produtos como o Wi-Fi. Nos sistemas de comunicação, o Wi-Fi é encontrado na camada mais baixa da pilha de protocolos, ele produz uma rede local sem fio que é definida pela especificação IEEE 802.11, essa família de padrões fornece protocolo para as camadas PHY e MAC de redes locais sem fio (CHEW, 2018).

Sobre as tecnologias de acesso sem fio, prevalecem os esquemas de topologia *estrela*, *malha* e *ponto-a-ponto*, para conectar dispositivos de IoT (figura 9). Em longo e curto alcance a topologia em estrela é predominante, como visto em redes de celular, LPWA e Bluetooth (HANES, 2017).



Fonte: Hanes (2017).

A topologia em estrela utiliza uma única estação base central ou controlador com total funções para comunicações com terminais com funções reduzidas. Para tecnologias de médio alcance, qualquer uma das topologias estrela, ponto-a-ponto ou malha são aplicadas. Topologias ponto-a-ponto permitem que dispositivos com total funções se comuniquem uns com os outros, desde que estejam ao alcance um do outro, essa mesma topologia em formações mais

complexas, com vários terminais com funções reduzidas, produz uma rede em malha. As implantações de Wi-Fi internas são, na maioria das vezes, um conjunto de nós que formam uma topologia de estrela em torno de seu ponto de acesso (HANES, 2017).

Para tralhar sobre em tais topologias os padrões do IEEE 802.11, publicados em 1997, foram alterados várias vezes ao longo dos anos. A Wi-Fi Alliance certifica o subconjunto das formas de onda contidas nos padrões. Isso implica que, embora o IEEE 802.11 defina várias formas de onda, o link que usado como Wi-Fi emprega apenas um subconjunto dessas formas de onda. O Wi-Fi opera nas bandas de 2,4 e 5 GHz, qualquer uma dessas faixas de operação pode fornecer um gateway para uma rede de dispositivos. Porém, operação de Wi-Fi na banda de 2,4 GHz, que é a faixa mais aplica em IoT, é uma fonte de interferência para muitas redes de dispositivos (CHEW, 2018). A Wi-Fi Alliance aprovou um padrão para conexões sem fio chamado WiFi HaLow (IEEE 802.11ah) que estende o Wi-Fi na banda de 900-MHz, permitindo a conexão de redes de sensores de baixa potência que são geralmente incorporadas em dispositivos de domótica supridos por bateria (FROIZ-MÍGUEZ, 2018).

#### 2.2.4 HTTP e Websocket

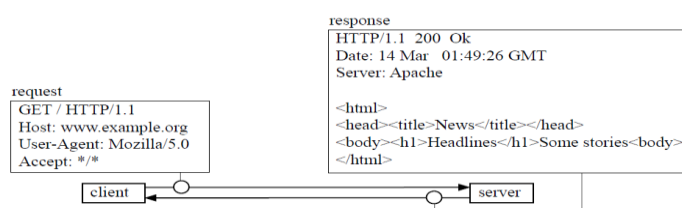
O *HyperText Transfer Protocol* (HTTP) é o protocolo mais importante para a Internet, ele alimenta a *World Wide Web* (WWW) e é suportado por praticamente qualquer plataforma e linguagem de programação. O HTTP é da camada mais alta da pilha de protocolos, ao longo dos anos ele se tornou muito importante e várias soluções alternativas foram inventadas para superar suas deficiências, fazendo com que o HTTP seja uma escolha coerente para a comunicação da IoT (ADRYAN; OBERMAIER; FREMANTLE, 2017).

A Web funciona em documentos de hipertexto, o *navegador* desempenha o papel de leitura e conversão desses documentos em um formato legível e mais significativo, que é chamado de *página*, os navegadores têm mecanismos que lêem arquivos em *HyperText Markup Language* (HTML) e renderizam aplicativos da Web na maneira como os arquivos HTML são descritos. Para que esse processo ocorra de forma organizada a Web é dividida em *cliente* e *servidor*. O navegador é o *cliente*, e aquele que fornece os dados ao navegador é o *servidor*. Os servidores armazenam dados e fornecem o mesmo a pedido do cliente (CHOPRA, 2015).

O HTTP usa um modelo clássico de comunicação de solicitação-resposta, que permite que o cliente envie dados para o servidor e receba uma resposta dele. Em HTTP/1.1 o cliente inicia uma solicitação e o servidor envia uma resposta assim que processa a solicitação (figura 10). A vantagem do HTTP é a flexibilidade, isso se deve à capacidade de adicionar cabeçalhos

arbitrários a qualquer solicitação e resposta. Há muitos cabeçalhos padronizados que são universalmente aceitos, para autenticação, controle de cache, preferências do cliente e codificação do idioma da resposta. Embora esse modelo seja perfeito para o WWW, onde o cliente solicita uma página da Web e o servidor entrega a página, a comunicação IoT orientada a eventos com HTTP não é ideal para mensagens iniciadas pelo servidor, porque não há uma maneira padrão para enviar mensagens proativamente como servidor (ADRYAN; OBERMAIER; FREMANTLE, 2017).

Figura 10 – Padrão de solicitação-resposta



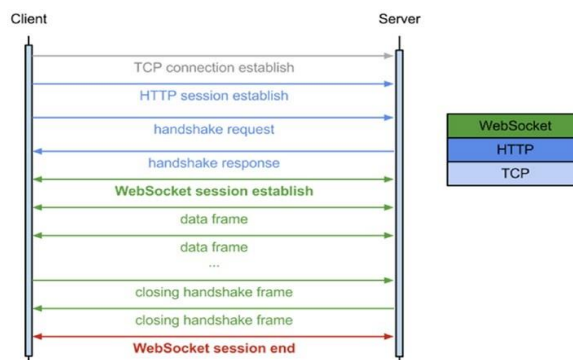
Fonte: Adryan, Obermaier e Fremantle (2017).

Quando aplicativos da web começaram a exigir que as informações fossem enviadas do servidor para o cliente, surgiu o *Ajax*, que é um método que possibilita aos navegadores executar solicitações HTTP e obter a resposta sem ter que atualizar a página inteira. Em uma sala de bate-papo, por exemplo, um usuário envia mensagens para outro usuário por meio do servidor. O usuário, através de solicitação *Post* de HTTP, envia a mensagem ao servidor para que ele entregue ao outro usuário. Porém, o servidor não pode enviar essa solicitação HTTP ao outro cliente, sua função é receber solicitações HTTP e não enviá-las, pois é assim que o HTTP funciona. Para resolver isso, o *Ajax* do outro usuário, de tempos em tempos, envia solicitações *Get* de HTTP para o servidor a fim de verificar se há alguma mensagem para ele, se houver mensagem, o servidor anexa a mensagem à resposta HTTP. A essa técnica dá-se o nome de *polling*, ela é muito empregada em IoT (ACETOZI, 2017).

Nesse cenário, a cada pequeno espaço de tempo, os clientes enviam solicitações *Get* de HTTP para o servidor, mesmo que não haja mensagens para eles, como consequência, poderá haver uma sobrecarga de solicitações-resposta. Além disso, toda vez que uma solicitação HTTP ocorre, há um *handshake* e uma conexão *Transmission Control Protocol* (TCP) é estabelecida entre o cliente e o servidor. Esta é uma operação que consome recursos, e como o protocolo HTTP é detalhado, com muitos cabeçalhos, todas as solicitações consomem muita largura de banda. Como solução alternativa, foram criados ao longo dos anos novos protocolos, como o

protocolo *Websocket*, para resolver alguns desses problemas. O protocolo WebSocket é mais adequado que o HTTP para enviar dados do servidor ao cliente. Ele permite que seja aberta uma conexão TCP bidirecional full-duplex em que ambos os lados (o cliente e o servidor) podem enviar *quadros*. Esses quadros são diferentes das solicitações *Get* e *Post* do HTTP. Na verdade, depois que uma conexão WebSocket é aberta, todo o tráfego entre o cliente e o servidor ocorre por meio dela, portanto, nenhuma solicitação HTTP é enviada mais (figura 11). Com WebSocket, sua solicitação HTTP se torna uma única solicitação para abrir uma conexão WebSocket e reutiliza a mesma conexão do cliente para o servidor, e do servidor para o cliente (ACETOZI, 2017).

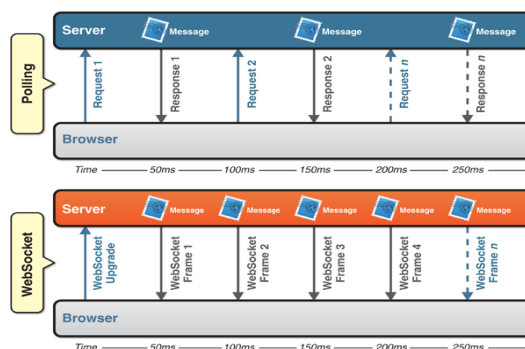
Figura 11 – WebSocket *handshake* entre cliente e servidor



Fonte: Acetozzi (2017).

O WebSocket também reduz a latência porque, uma vez estabelecida a conexão, o servidor pode enviar mensagens à medida que elas se tornam disponíveis. Ao contrário do *polling*, o WebSocket faz uma única solicitação (figura 12).

Figura 12 – Comparação entre o *polling* (estratégia de pesquisa) e WebSocket



Fonte: Wang, Salim e Moskovits (2013).

O servidor não precisa esperar por uma solicitação do cliente. Da mesma forma, o cliente pode enviar mensagens para o servidor a qualquer momento. Essa solicitação única reduz muito a latência em relação ao *polling*, que envia uma solicitação em intervalos, independentemente de as mensagens estarem disponíveis. Depois que a conexão é aberta, ela usa um mecanismo de pulsação através de quadros de *ping-pong* para manter a conexão ativa (WANG, SALIM e MOSKOVITS, 2013).

### 2.3 SISTEMAS EMBARCADOS

A minituarização da eletrônica deu início a integração do processamento de informações com o ambiente físico usando computadores. Este novo modelo de processamento de informação foi batizado de *sistema embarcado* (MARWEDEL, 2018). Um sistema embarcado pode ser definido como um sistema de computador que executa uma ou mais funções dedicadas ou especializadas, comumente com restrições em tempo real. É o controlador digital que é incluso como componente do sistema, que inclui também outros elementos mecânicos e de hardware (SANCHEZ e CANTON, 2012).

Os sistemas embarcados são controlados por um ou mais dispositivos de processamento digital, normalmente um microcontrolador ou processador de sinais digitais. A principal característica do sistema embarcado é que ele é dedicado a uma tarefa específica. Em alguns poucos casos, um sistema de computador especializado ou dedicado, como o que controla uma barragem ou a operação de uma usina nuclear, é considerado um sistema embarcado. Sistemas embarcados variam amplamente em tamanho e complexidade, de dispositivos portáteis, como um relógio digital, a controladores complexos, como o que opera um robô de exploração espacial (SANCHEZ e CANTON, 2012).

Definidos como sistemas de processamento digital, os sistemas embarcados possuem um software de controle presente na memória do circuito eletrônico. Eles interagem com o ambiente através de sensores e atuadores, sendo empregados em aviões, carros, telefones celulares, eletrodomésticos, equipamentos médicos, entre outros. De forma geral, são empregados na maioria dos equipamentos eletrônicos atuais (SCHNEIDER e SOUZA, 2010).

Muitos projetos de engenharia elétrica e de computação envolvem algum tipo de sistema embarcado no qual um microcontrolador é o centro, seu papel é ser a principal fonte de controle. Programas são escritos para serem executados em um processador incorporado. Essa função nos remete a uma plataforma de desenvolvimento bem conhecida, o Arduino, que é uma placa de circuito contendo um microcontrolador ATMEL ATmega pré-carregado com um programa de *boot loader*,

com recursos como esquema de hardware disponíveis gratuitamente ao público e um Ambiente de Desenvolvimento Integrado (IDE) que dispensa detalhes programacionais de baixo nível (RUSSELL, 2010).

Independentemente do modelo, os sistemas embarcados geralmente consistem em um microcontrolador e alguns dispositivos de I/O, projetados para monitorar alguns sensores de entrada e gerar sinais para controlar dispositivos externos, como acender LEDs ou ativar alguns interruptores por exemplo. Por esta razão, programas de controle dos primeiros sistemas embarcados são muito simples. São escritos na forma de *super-loop* ou de uma estrutura de programa simples orientada a eventos.

Mas nos últimos anos o poder computacional dos sistemas embarcados aumentou, com um tremendo saldo em complexidade e áreas de aplicação. Para lidar com a crescente complexidade dos sistemas e as demandas por funcionalidades extras, os sistemas embarcados precisam de um software robusto. Nos tempos atuais muitos sistemas embarcados são na verdade máquinas computacionais de alta potência com processadores multicore, memória e dispositivos de armazenamento. Eles agora são capazes de executar uma ampla gama de aplicativos e programas. Para alcançar esse potencial, os sistemas embarcados modernos precisam do suporte de sistemas operacionais multifuncionais (WANG, 2017).

Os Sistemas Operacionais de Propósito Geral (GPOS) são sistemas operacionais completos. Suas principais características são o fluxo dos *processos*, suporte a dispositivos como teclado, mouse e monitor, sistema de arquivos, memória e interface de usuário amigável. Por conta da complexidade, é comum portar a novos embarcados sistemas operacionais já existentes, em vez de projetar e implementar um novo GPOS do zero. Alguns sistemas portados aos sistemas embarcados são o Linux, FreeBSD, NetBSD e Windows (WANG, 2017).

### **2.3.1 ESP8266**

O chip ESP8266 surgiu como uma extensão Wi-Fi para placas de desenvolvimento, que foi originalmente comercializado com baixo custo, para Arduino e demais placas, com o objetivo de suprir a necessidade de estabelecer conexões TCP com outros dispositivos ou com a Internet. Com o tempo percebeu-se que, em lugar de agregar o ESP8266 a placas de desenvolvimento, este chip poderia ser usado de forma independente, como um microcontrolador com Wi-Fi embutido (JAYAKUMAR, 2016).

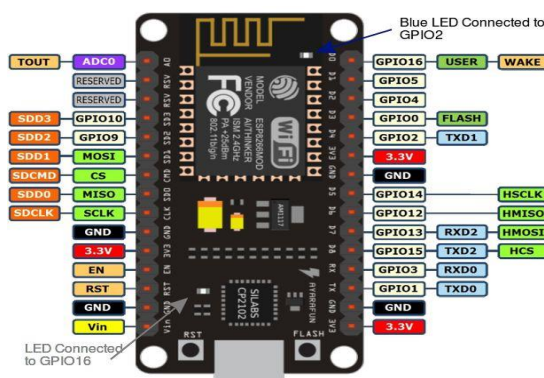
O ESP8266 possui arquitetura RISC de 32 bits que funciona por padrão a uma frequência de 80 MHz, com possibilidade de operar em 160 MHz. Para o firmware, ele conta

com 512 kB de memória Flash que pode ser atualizado via cabo. A alimentação é através de fonte de 3,3 Volts que consome 170mA ao transmitir em plena potência, e quando em sono profundo consome 10uA (BATRINU, 2017).

Suas interfaces de entrada e saída são as *General Purpose Input-Output* (GPIO), que podem ser configuradas como entradas ou saídas de nível digital, quatro saídas PWM e uma entrada analógica com precisão de 10 bits, o nível de tensão usado nesses pinos é de 3,3V e a corrente máxima é de 12mA. As interfaces de comunicação serial síncrona são SPI, I2C e I2S, as assíncronas são UART e Wi-Fi que pode atuar em modo AP, ad hoc ou Wi-Fi Direct. O Wi-Fi trabalha de acordo com o IEEE 802.11 b/g/n, que conta com autenticação WPA/WPA2 e WEP, o processamento dessa pilha de protocolos, usa cerca de 20% da capacidade de processamento total (OLIVEIRA, 2017).

O chip ESP8266 é muito pequeno, então, ele é distribuído em uma ampla variedade de módulos por vários fornecedores, existem módulos do ESP-01 até ESP-12, que diferem em vários aspectos. Usando um desses módulos, a placa de desenvolvimento NodeMCU extrai todos os recursos disponíveis do ESP-12 (JAYAKUMAR, 2016).

Figura 13 – NodeMCU e indicação de pinos



Fonte: Thakur (2018).

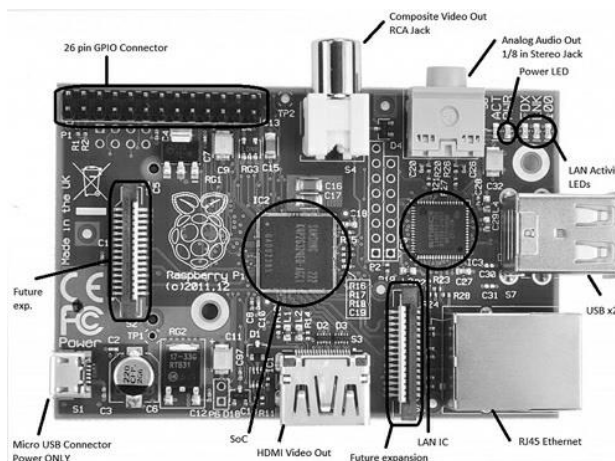
O NodeMCU disponibiliza em pinos, todas as GPIO's do ESP-12 (figura 13). A placa conta ainda, com um regulador de tensão de 3,3V para a alimentação e um conversor serial USB que permite conectar o ESP-12 ao computador (EICHHORN, 2016).

### 2.3.2 Raspberry Pi

O Raspberry Pi é um mini computador de placa única que trabalha em multitarefa executando múltiplos processos (figura 14). Igualmente aos computadores pessoais, placas

Raspberry Pi contam com conexões de rede Ethernet, Wi-Fi, Bluetooth, saída de vídeo em HDMI e porta USB para periféricos como teclado, mouse e impressora (STAPLE, 2018). Raspberry Pi conta ainda com chip do tipo *System on a Chip* (SoC) que é uma tecnologia que coloca fisicamente a memória, o microprocessador e o processador gráfico em um único lugar, ocupando na placa de circuito impresso (PCB) um menor espaço (NORRIS, 2014)

Figura 14 – Raspberry Pi 1 Model B revision 2.0



Fonte: Norris (2014).

Existem vários modelos de Raspberry Pi, cada um possui alguma particularidade relacionada ao hardware, tamanho ou preço (Quadro 1). Diferentemente dos modelos normais, Raspberry Pi 1, 2 e 3, o *Raspberry Pi Zero* foi desenvolvido em tamanho menor e hardware mais simples que os outros modelos. Da mesma forma, o modelo *Compute Raspberry Pi*, é especial por ser voltado a soluções industriais (KURNIAWAN, 2018).

Quadro 1 – Modelos de Raspberry Pi

Modelo	CPU	SoC	RAM
Raspberry Pi 1 B	ARMv6Z 32-bit single-core 700 MHz	BCM2835	512MB
Raspberry Pi 3 B	ARMv8-A 32/64-bit quad-core 1.2GHz	BCM2837	1GB
Raspberry Pi Zero	ARMv6Z 32-bit single-core 1 GHz	BCM2834	512MB
Raspberry Pi Compute 3	ARMv8-A 64-bit quad-core 1.2GHz	BCM2837	1GB

Fonte: Adaptado de Kurniawan (2018).

Com seu tamanho retangular, nas medidas de 56 por 85 mm, Raspberry Pi conta com interfaces de entrada e saída, como as GPIO's encontradas em microcontroladores, há um total

de 26 pinos GPIO que podem ser usados (figura 15). Em cada GPIO, o estado ligado, é representado por aproximadamente 2,7 à 3,3V, enquanto que, o estado desligado, é de aproximadamente 0 à 0,2V. A placa não contém pinos pelos quais sinais analógicos podem ser processados, então, um conversor analógico-digital (ADC) externo precisa ser usado para lidar com esse tipo de sinal.

Figura 15 – Indicação de pinos do Raspberry Pi 3 B

Raspberry Pi 3 Model B J8 GPIO Header			
	Pin No.		
3.3V	1	2	5V
GPIO2	3	4	5V
GPIO3	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3.3V	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7
DNC	27	28	DNC
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
GND	39	40	GPIO21

Fonte: Norris (2019).

Todos os modelos de Raspberry Pi executam diversas versões de GPOS adaptadas a sua CPU ARM, como por exemplo RISC OS, Pidora, Arch Linux, Raspbian, Ubuntu MATE e Windows 10 IoT Core (KURNIAWAN, 2018). Atualmente o Raspbian é o OS oficial da plataforma além de ser o mais popular dos sistemas baseados em Linux para Raspberry Pi. O Raspbian é um OS de código aberto baseado no Debian, que foi modificado especificamente para o Raspberry Pi. O Debian é uma das distribuições originais do Linux, então como o Raspbian é baseado no sistema operacional Debian, ele compartilha quase todos os recursos do Debian, incluindo seu vasto repositório de pacotes de software (HARRINGTON, 2015).

## 2.4 APRENDIZADO DE MÁQUINA

O Aprendizado de Máquina é entendido como um ramo da ciência da computação que visa aprender com a experiência passada, e usar o conhecimento adquirido, na tomada de decisões futuras. Aprendizado de máquina está na interseção de três campos da ciência, a

computação, engenharia e estatística. Seu objetivo é generalizar padrões detectados ou encontrar regras desconhecidas a partir dos exemplos de uma base de dados (DANGETI, 2017).

Com o aprendizado de máquina, são desenvolvidas técnicas computacionais e sistemas capazes de adquirir conhecimento de forma automática, para isso, o programador do sistema apresenta dados representativos a mecanismos usados na caracterização de dados. O aprendizado de máquina é classificado em três categorias, que com base na situação, podem ser até mesmo combinadas para alcançar os resultados desejados em aplicações específicas (MITCHELL, 1997).

No aprendizado supervisionado os dados de entrada com suas respectivas saídas são apresentados ao algoritmo de aprendizado durante o processo de treinamento. Desse modo, máquinas aprendem a relação entre uma variável-alvo e demais variáveis, isso se assemelha, por exemplo, a maneira que um professor usa critérios de avaliação para fornecer *feedback* a seus alunos sobre seu desempenho (BRAGA; CARVALHO; LUDEMIR, 2007).

O aprendizado não supervisionado trabalha com conjuntos de exemplos que não estão rotulados, sua técnica de aprendizado classifica esses conjuntos, agrupando os semelhantes em determinadas classes. Nesse aprendizado os algoritmos aprendem sozinhos, sem qualquer supervisão ou sem qualquer variável-alvo fornecida. É uma questão de encontrar padrões e relações ocultas nos dados fornecidos (RUSSELL e NORVING, 2004).

Na estratégia de aprendizado por reforço mapeia-se o estado do ambiente, usando percepção e ação, de modo a maximizar um sinal de recompensa numérico. A ideia é captar os aspectos mais importantes do problema colocando um agente que vai interagir com o ambiente para alcançar uma meta. A máquina ou agente aprende o comportamento com base no *feedback* do ambiente. O agente toma uma série de ações decisivas sem supervisão e, no final, uma recompensa será dada, seja +1 ou -1, com base no resultado final/recompensa, o agente reavalia seus caminhos (SUTTON e BARTO, 1998).

Modelos baseados nessas categorias, na maioria dos sistemas inteligentes, são responsáveis pela maior parte da inteligência, enquanto outros métodos são usados para suportar e preencher lacunas deixadas por esses modelos. Em geral, os modelos combinam recursos do contexto, testam valores de recursos, multiplicando-os, redimensionando-os e assim por diante. Mesmo modelos simples, podem realizar dezenas de milhares de operações para produzir suas previsões (HULTEN, 2018).

Existem muitos tipos de modelos, entre eles alguns bem conhecidos são os modelos lineares, redes neurais e árvores de decisão.

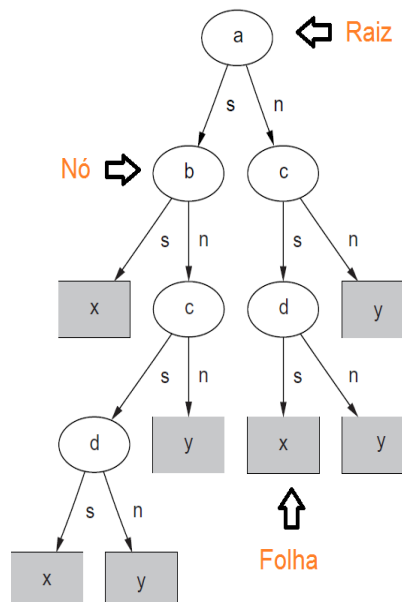
### 2.4.1 Árvores de decisão

Em problemáticas no qual um conjunto de instâncias independentes ou dados são usados para aprender algo, uma abordagem comum é a de *dividir para conquistar*, esse método remete a um estilo de representação chamado de *árvore de decisão* (DINOV, 2018).

As árvores de decisão podem ser aplicadas em dois cenários, em problemas de classificação ou de regressão. A partir dos dados, os modelos de árvore de decisão aprendem uma série de perguntas para inferir os rótulos de *classes* das amostras. Para solucionar o problema de classificação, os algoritmos de árvore de decisão aplicam a abordagem de divisão e conquista. Eles buscam de cima para baixo, em cada estágio, um atributo que melhor separe as classes, e em seguida, recursivamente processam os subproblemas resultantes dessa divisão (WITTEN et al., 2016).

Como o nome indica, árvores aprendidas podem ser utilizadas para o tomada de decisão. Levando-se em conta a porcentagem de acerto do modelo alcançado, faz-se uma sucessão de escolhas de valores para os nós da árvore, caminhando desde o nó raiz até um dos possíveis resultados nas folhas.

Figura 16 – Representação de uma árvore de decisão



Fonte: Adaptado de Witten et al. (2016).

A título de exemplo, em um árvore de quatro *atributos* (figura 16), o nó raiz contém o primeiro teste *if* e tem um filho para quando o teste lógico “a” é verdadeiro (*then*) e um filho

para quando o teste “a” é falso (*else*), e assim por diante, com mais nós para mais testes. As folhas da árvore contêm a consequência das escolhas, ou melhor o valor final, a *classe* (HULTEN, 2018).

Como já mencionado, no processo de aprendizagem de uma árvore, divide-se os atributos (variáveis) do conjunto de treinamento recursivamente, com base em critérios de impureza definidos, até que atinjam critérios de parada. Mas, para determinar onde cada variável será encaixada na árvore, são calculados para os atributos, a entropia, o ganho de informação e a impureza de Gini (DANGETI, 2017).

A *entropia* é uma medida, da teoria da informação, que indica a impureza nos dados. Se a amostra é completamente homogênea, a entropia é zero, mas se a amostra é igualmente dividida, sua entropia é um. Nas árvores de decisão, o preditor com maior heterogeneidade será considerado o mais próximo do nó raiz para classificar os dados fornecidos em classes (HULTEN, 2018).

$$\text{Entropia } (S) = -p_1 * \log_2 p_1 - \dots - p_n * \log_2 p_n$$

$S$  : é o conjunto de amostras (registros ) (1)

$p$  : é a proporção de amostras da classe em relação ao total de amostras

$n$  : é o número de valores possíveis da classe

$$\text{Entropia } (S) = \sum_{i=1}^c - p_i * \log_2 p_i$$

(2)

$p_i$  : é a proporção de dados em  $S$  que pertencem à classe  $i$

$c$  : número de classes distintas

O *ganho de informação* é a redução esperada na entropia causada pelo particionamento dos exemplos de acordo com um determinado atributo, isso é feito para determinar se uma condição de teste realizada é realmente boa, então compara-se o grau de entropia do nó pai (antes da divisão) com o grau de entropia dos nós filhos (após a divisão). Em cada estágio, a variável com ganho máximo de informação é escolhida (HULTEN, 2018).

$$Ganho(S, A = x) = Entropia(S) - \sum \frac{|S_x|}{|S|} Entropia(S_x)$$

$A$  : é um atributo do conjunto de dados  $S$

$S_x$  : é um subconjunto de  $S$  formado quando o atributo  $A$  for igual ao valor de atributo  $x$  (quando  $A = x$ ) (3)

$|S_x|$  : número de elementos de  $S_x$

$|S|$  : número de elementos de  $S$

A *impureza de Gini* é uma medida de má classificação, que se aplica em um contexto de classificador com várias classes. Gini funciona semelhante à entropia, exceto que o cálculo de Gini é mais rápido de se fazer (DINOV, 2018).

$$Gini(A = x) = 1 - \sum_{i=1}^c p_i^2$$

$A = x$  : é um nó (teste lógico) (4)

$p_i$  : é a frequência relativa de cada classe no nó

$c$  : número de classes distintas

Árvores de decisão também podem ser usadas de outra forma, como árvores de *previsão*. Nesse caso, não usa-se a árvore para guiar a escolha do valor de cada nó, pois, esses valores já estão previamente determinados, o que se faz é tentar prever o provável valor da folha através dos valores que os nós apresentam. Desse modo, trabalhamos com sentenças simples, representadas por testes *if, then, else*. Esta série de instruções simplificaria a estrutura da árvore que pode ser armazenado facilmente como um arquivo de dados, que por exemplo, suportaria ser carregado e interpretado rapidamente em uma aplicação que responda em tempo de execução.

Nos algoritmos que fazem esse tipo de simplificação, as sentenças de *if-then-else* são chamadas de *regras de classificação*. Funciona assim, uma regra é gerada para cada folha da árvore, o antecedente da regra inclui uma condição para cada nó no caminho da raiz para essa folha, e o consequente da regra é a classe atribuída pela folha. Logo, há uma regra de decisão para cada caminho que vai do nó raiz até um nó folha. Da árvore na figura 16, podemos extrair a seguinte regra: “*If a and b then x*”. A partir disso, vemos que uma árvore de decisão leva em

conta todas as classes, tentando maximizar a pureza da divisão, enquanto que, o método de geração de regras se concentra em uma classe de cada vez, desconsiderando o que acontece com outras classes (WITTEN et al., 2016).

#### **2.4.2 Algoritmo C4.5**

O algoritmo C4.5 é um sistema de aprendizado indutivo e supervisionado, que supera seu antecessor ID3 em contruir arvores de decisão (QUINLAN, 1986). Por mostrar excelentes resultados em problemas de classificação, ele é um algoritmo utilizado em diversos contextos. Ao executar o script de aprendizado do C4.5, a partir de um conjunto de exemplos, são calculadas as equações 1, 2 e 3, produzido uma árvore de decisão (QUINLAN, 1993).

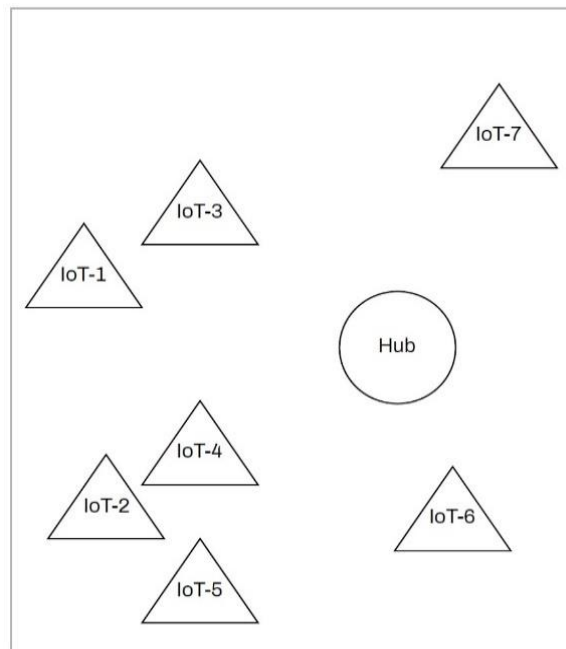
Uma característica importante a respeito do algoritmo C4.5 é que ele trabalha tanto com atributos categóricos como atributos contínuos, para lidar com os atributos contínuos, o algoritmo C4.5 define um limiar e divide os exemplos de forma binária, verificando se o valor do atributo é maior, menor ou igual ao limiar. O algoritmo também permite a utilização de valores desconhecidos, que são representados como “?”, de forma especial esses valores não são utilizados nos cálculos de ganho e entropia. O C4.5 também conta com um método de pós-poda das árvores geradas, ele faz uma busca na árvore, de baixo para cima, e transforma em nós folha aqueles ramos que não apresentam nenhum ganho significativo (ANDRADE, 2013).

### 3 DESENVOLVIMENTO

#### 3.1 ARQUITETURA PUBLICADOR-OBSERVADOR

Antes de abordarmos o sistema domótico que desejamos implementar vamos discutir os aspectos de modelagem da nossa arquitetura IoT. Para isso imaginemos que em uma casa exista uma HAN onde oito unidades IoT se conectam, são eles sete nós IoT's quaisquer e um Hub (figura 17).

Figura 17 – HAN composta por oito elementos

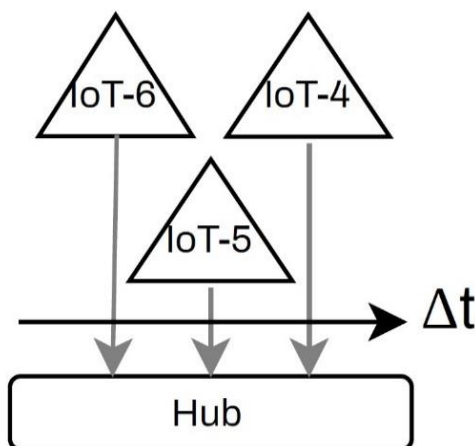


Fonte: Autoria própria.

Como vimos na seção 2.1 dispositivos IoT's distribuídos pelo ambiente da residência produzem muitos dados que são armazenados e podem ser usados para o aprendizado de máquina. Em termos funcionais, esses dados são impactados pelo comportamento dos habitantes da casa, por mudanças ambientais e variáveis subjacentes desconhecidas.

Se um observador acompanhar os nós IoT em tempo real, ele por vezes, irá presenciar diferentes entradas ocorrendo em um intervalo de tempo próximo. O mesmo pode ser observado através da base de dados, filtrando somente as mudanças significativas e agrupando os dados publicados em um dado intervalo de tempo ( $\Delta t$ ) próximo (figura 18).

Figura 18 – Publicação de dados dos IoT's da Figura 17 no Hub



Fonte: Autoria própria.

Os eventos podem acontecer em situações aleatórias, porém, se um mesmo evento se repete ao longo do tempo, produzindo dados muito próximos aos que foram registrados outras vezes, existe a possibilidade dos nós IoT's possuírem algum tipo de sinergia, seja ela visivelmente explícita ou implícita.

Por exemplo, temos um *sensor IoT de temperatura ambiente* (SIT) e um *aparelho de ar-condicionado com IoT* (ACI) posicionados em um mesmo cômodo, onde as leituras do SIT indicando temperatura ambiente alta (TA) antecedem o ajuste da temperatura do ACI para um valor mais baixo (TB) em um intervalo de tempo  $\Delta t$ .

Percebemos uma sinergia explícita entre SIT e ACI dado a natureza desses dispositivos, mas é evidente que leituras de temperatura alta por si só não afetarão o ACI, pois eles são independentes um do outro. Mas, mesmo que eles não tenham uma relação de causa e efeito, ao aplicarmos aprendizado de máquina em seus dados, podemos obter um modelo de previsão.

Uma forma de trabalhar com isso é modelar esses dois IoT's em uma *regra de associação*, o SIT sendo *preditor* do ACI e ACI sendo *desfecho* do SIT, a *regra de associação* entre eles seria assim:  $\{SIT\} \rightarrow \{ACI\}$ . Em vista disso, se tratarmos SIT e ACI como *atributo* e *classe* respectivamente dentro de um conjunto de treinamento, poderíamos obter a seguinte *regra de classificação*:  $\{SIT=TA\} \rightarrow \{ACI=TB\}$ .

Como em alguns casos a sinergia entre os dispositivos é explícita, um perito pode analisar as características de cada IoT e sua distribuição no ambiente e com essa leitura presumir quais *regras de associação* são possíveis. Podemos imaginar essa leitura para a nossa HAN hipotética (figura 19).

Figura 19 – Exemplo de três associações IoT possíveis para a HAN da Figura 17



Fonte: Autoria própria.

A partir da compreensão dos conceitos aqui apresentados, modelamos uma arquitetura IoT nomeada de *arquitetura publicador-observador*. Na arquitetura publicador-observador os IoT's da HAN ao longo do tempo publicarão dados ao Hub que desempenhará o papel de *observador*. O Hub irá acompanhar em tempo real as publicações do *IoT desfecho* em conjunto com as publicações do *IoT preditor* e irá sempre verificar se elas ocorreram em um determinado  $\Delta t$ . A política de agrupamento de dados é: reter publicações que ocorram em um mesmo intervalo de tempo, para grupos de IoT's que apresentem uma associação conhecida. Em Tonidandel e Sgarbi (2006) um método parecido é nomeado de *janela de observação*.

Em nosso modelo de comunicação os nós IoT publicam ao observador dados sobre os fenômenos que estão acontecendo em tempo real, como mudanças no ambiente, interação humana direta ou indireta com o dispositivo ou ações autônomas do próprio IoT que ocorram por conta de sua programação.

O conteúdo das publicações são dados *qualitativos* cujo significado o observador não buscará entender. A perspectiva é que na arquitetura publicador-observador o Hub consiga reunir vários grupos de dados para formar um conjunto de treinamento que possibilite inferir *modelos preditivos*, na forma de regras de decisão, sobre as publicações do IoT desfecho. Em resumo, no conjunto de treinamento os atributos são IoT's preditores e a classe é o IoT desfecho.

Ao obter modelos preditivos, o Hub irá comunicar imediatamente ao IoT desfecho a publicação esperada no momento em que os IoT's preditores, dentro dos limites de  $\Delta t$ , manifestarem publicações idênticas ao do modelo preditivo. O IoT desfecho realiza uma ação com essa previsão.

Usamos um método de pontuação a fim de gerenciar os modelos preditivos, conforme acertam ou erram em suas previsões os modelos recebem pontos positivos ou negativos. Assim atribuímos valor a suas afirmações conforme elas se adequam à realidade ou não, igualmente a metodologia do trabalho de Tonidandel e Sgarbi (2006).

### 3.2 ESTRUTURA DA HAN

Introduzimos a proposta da arquitetura publicador-observador na construção da HAN do sistema de iluminação implementando um Hub e dois dispositivos IoT, o nó *IoT Lâmpada* e o nó *IoT Sensor de Presença* (quadro 2).

Quadro 2 – Apresentação dos IoT's desfecho e preditor

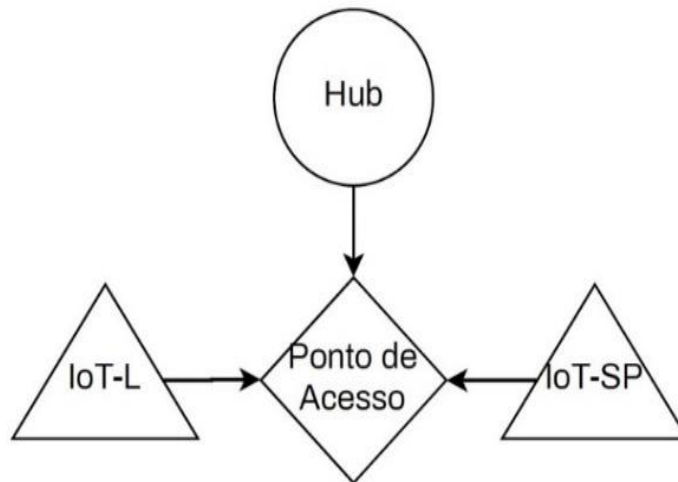
Nome do IoT	Dado publicado
IoT-L	Acesa
	Apagada
IoT-SP	Presente
	Ausente

Fonte: Autoria própria.

O *IoT Lâmpada* (IoT-L) irá publicar dados que indicam o acionar do interruptor da lâmpada do ambiente, sendo esse fenômeno consequência da ação direta do habitante, esse será nosso *IoT desfecho*. Enquanto que o *IoT preditor* será o *IoT Sensor de Presença* (IoT-SP), que publicará dados conforme seu sensoriamento indicar que o habitante está ou não presente no ambiente, fenômeno esse fruto da ação indireta do habitante.

Um roteador residencial servirá de ponto de acesso para criação da HAN através de uma rede Wi-Fi com modelo de rede *TCP/IP*, onde o Hub e os dois dispositivos IoT estabeleceram conexão (figura 20).

Figura 20 – Modelo genérico da HAN



Fonte: Autoria própria.

O restante das atividades que concretizaram a implementação foram executados seguindo as ações:

- 1) Planejar os nós IoT's com os sensores e atuadores necessários a suas próprias particularidades de acordo com a natureza do seu trabalho;
- 2) Construir os nós IoT's avaliando o uso de componentes que atendam aos requisitos de tamanho reduzido e conexão à rede sem fio;
- 3) Elaborar a comunicação entre os elementos onde a estrutura de rede apresenta sua topologia em estrela, programando os nós da rede para que operem no modo *infra-estrutura* e toda a HAN se comunique através de um único ponto de acesso;
- 4) Sistematizar o Hub para que receba as publicações dos nós IoT's, observe essas publicações pela métrica do  $\Delta t$ , infira e gere modelos preditivos, agrupe dados e informe ao IoT defecho sobre previsões;
- 5) Projetar toda a arquitetura para que funcione em tempo real;
- 6) Identificar tecnologias que melhor se adequam ao contexto de nossa proposta.

A partir deste ponto, serão comentados os elementos que formam a arquitetura, os circuitos existentes e a alocação de sensores, atuadores e embarcado, com uma breve explicação das etapas de construção de cada elemento.

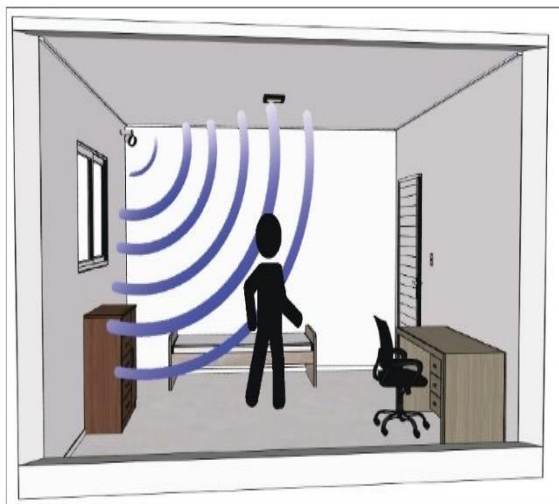
### 3.3 DOMÓTICA DA ILUMINAÇÃO

Neste ponto será apresentada a implementação da domótica que orientou a criação dos elementos da HAN. Há uma gama enorme de possibilidades de implementações que podem ser introduzidas em plantas residenciais e foi escolhido desenvolver um clássico sistema de controle de iluminação.

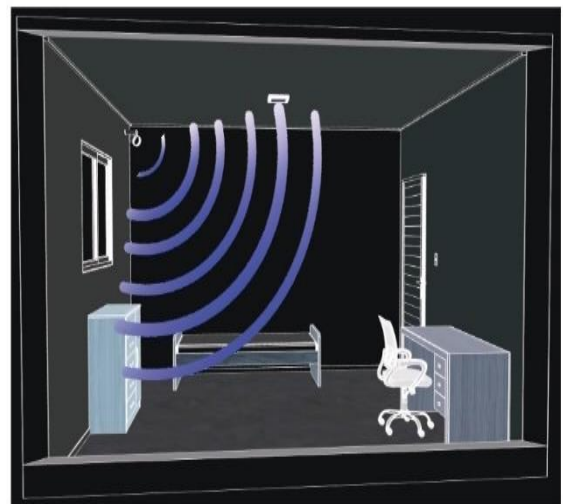
Esse tipo de domótica explora a problemática da *economia de energia elétrica* por parte da iluminação, portanto é feito um controle sobre o desperdício energético com luzes elétricas, que ocorre apagando-as automaticamente no momento em que não estiverem sendo usadas. Porém essa não é a única problemática que envolve sistemas de iluminação.

O motivo principal que nos levou a escolher essa implementação é a problemática das *preferências* de iluminação. Pessoas diferentes, tem preferências diferentes, mesmo que estejamos tratado de um simples sistema de iluminação. É muito comum por exemplo que os moradores de uma casa deixem propositalmente as luzes de alguns cômodos acesas mesmo que não haja ninguém usando o cômodo. Fora isso, existe também muitos outros fatores como dia, horário e clima que podem interferir na escolha do morador em como e se a iluminação deve ser acionada ou não.

Figura 21 – Sistema de controle de iluminação com sensor de presença. Presença detectada (a), luzes acendem e presença não detectada (b), luzes apagam



(a)

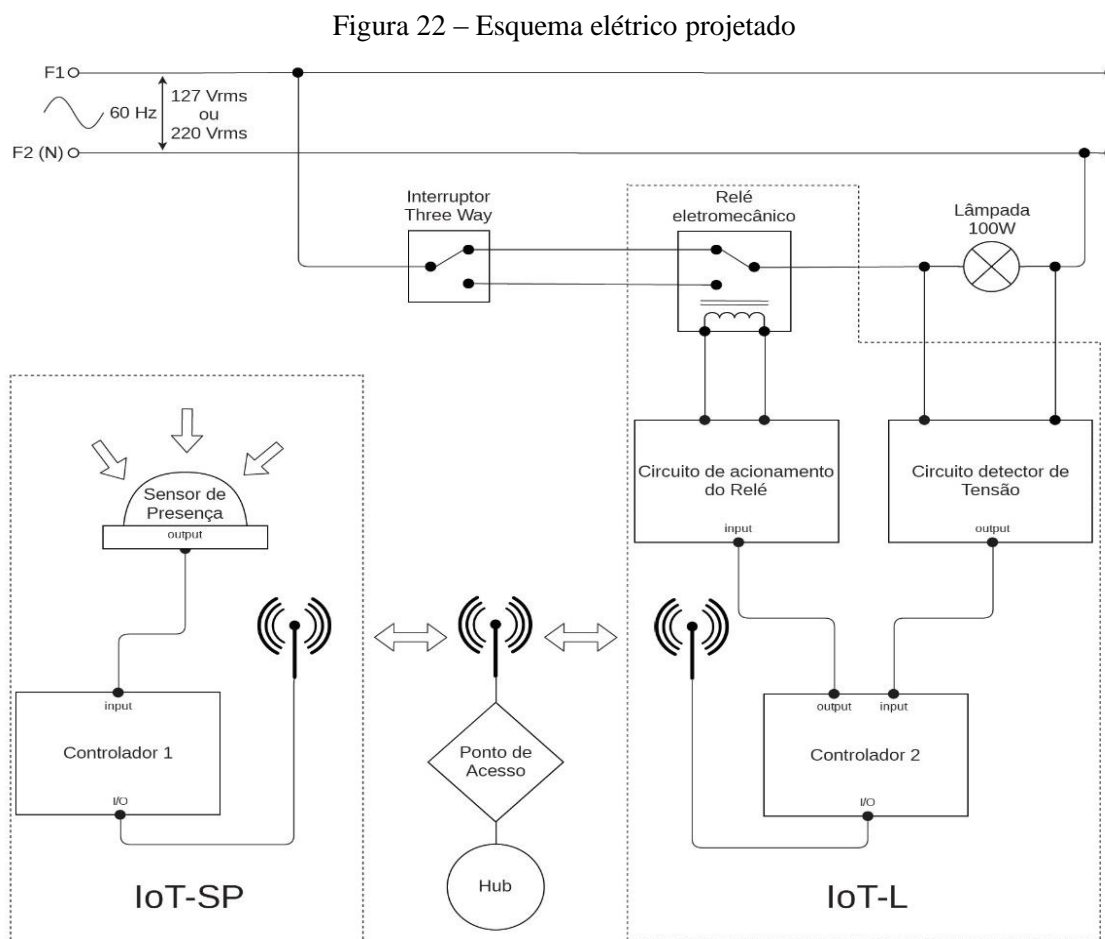


(b)

Fonte: Autoria própria.

Entenderemos a lógica de funcionamento dessa domótica pensando em um cômodo de uma casa, um quarto ou dormitório por exemplo (figura 21). Quando alguém entra nesse cômodo, um sensor instalado ali detecta a presença da pessoa e aciona automaticamente a iluminação. Enquanto o sensor continuar detectando a presença de alguém, a iluminação permanecerá acesa, do contrário, quando nenhuma presença for detectada, a iluminação será desligar automaticamente.

Introduzimos essa tecnologia em nossa problemática dimensionando o esquema elétrico desse cômodo (figura 22). Ao circuito terminal de uma lâmpada com interruptor é acoplado o circuito elétrico do nó IoT-L, enquanto que o circuito do nó IoT-SP atua de forma independente, mas ambos trocam informações com o Hub.

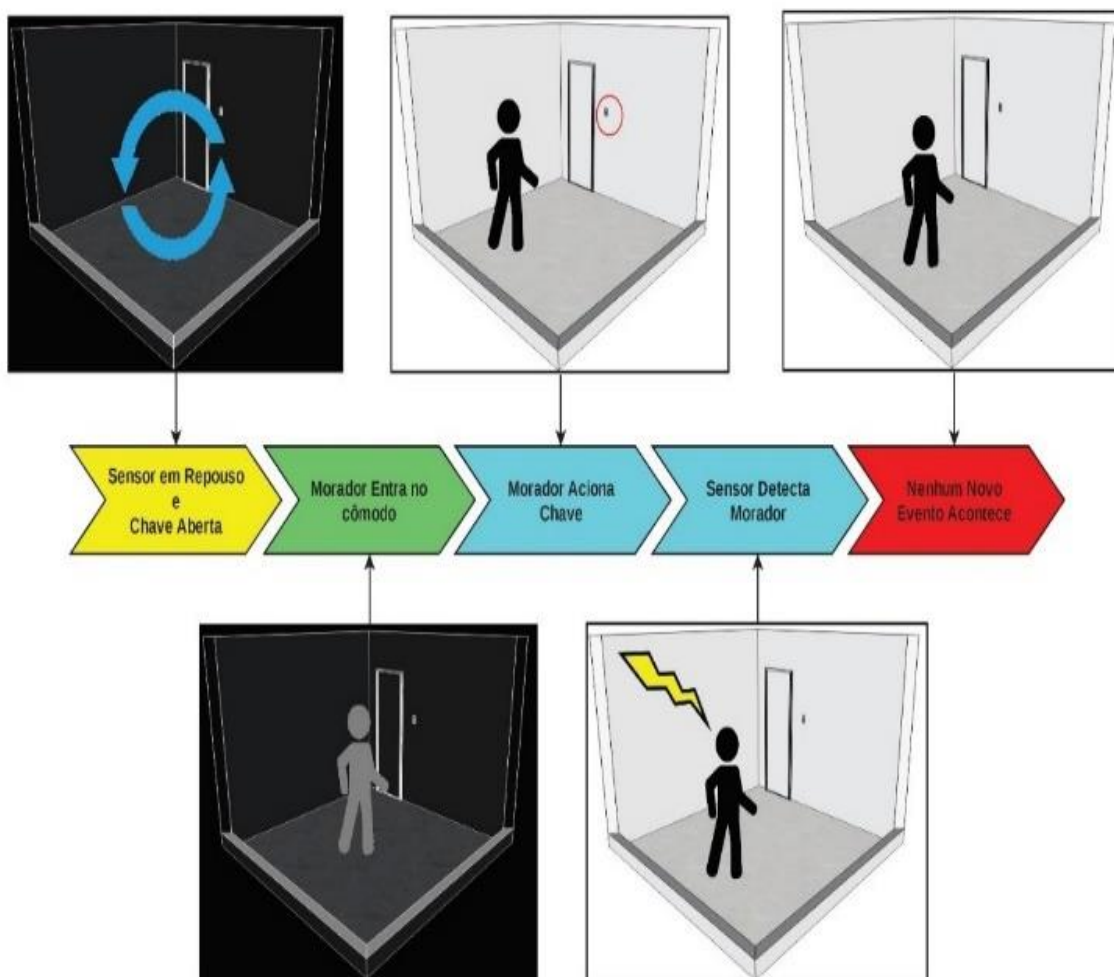


Fonte: Autoria própria.

No IoT-L o componente de leitura empregado como atuador é o Relé eletromecânico, enquanto que no IoT-SP o componente empregado foi o *passive infrared sensor* (PIR). O motivo da escolha desses componentes e sua ligação será explicado em tópico específico.

Essa instalação foi construída para funcionar do seguinte modo (figura 23): O interruptor da lâmpada, inicialmente desligado, é acionado ou não pelo morador manualmente sempre que ele entra nessa dependência da casa e, em algum instante de tempo próximo, um sensor de presença instalado ali sinaliza a detecção da presença do morador.

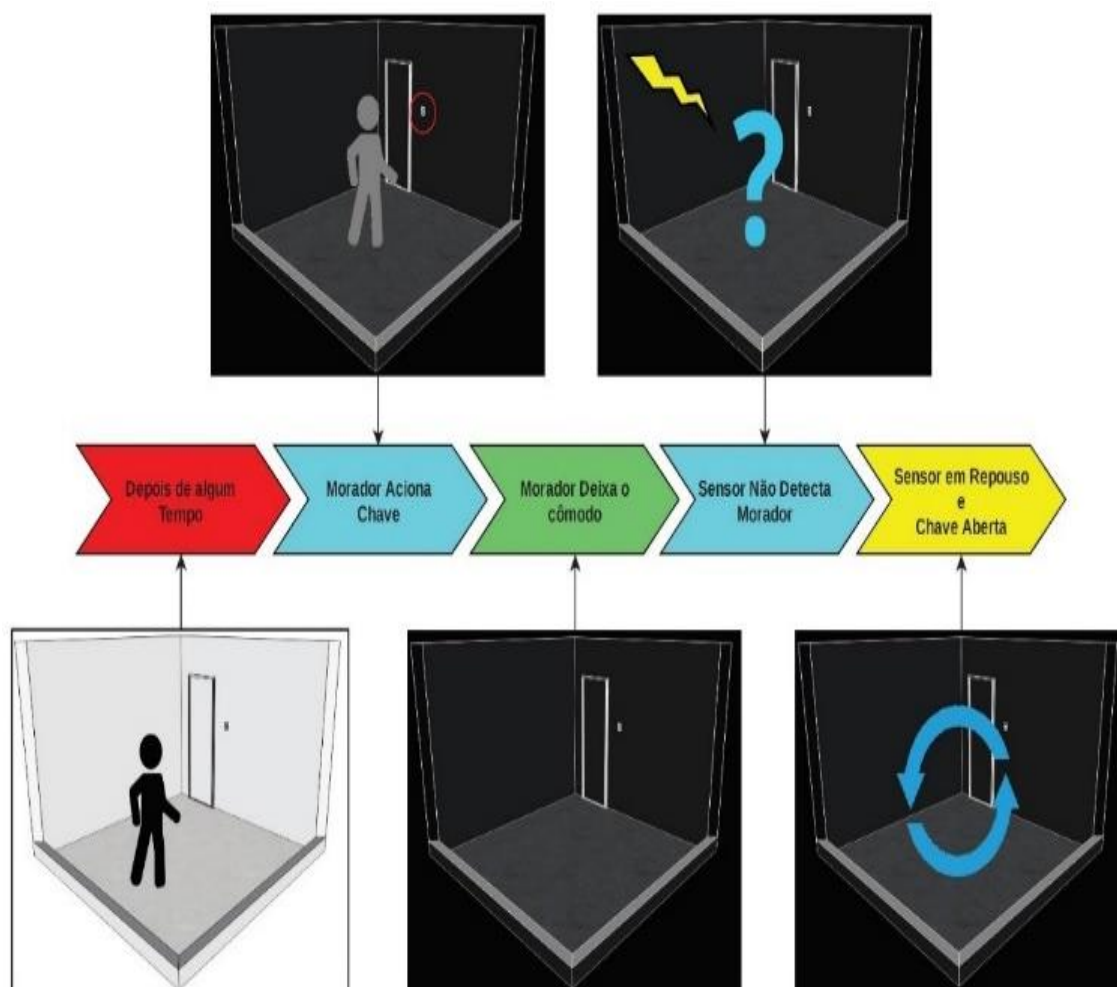
Figura 23 – Primeira etapa da interação do habitante com o sistema de iluminação IoT



Fonte: Autoria própria.

A presença desse indivíduo segue sendo detectada pelo sensor enquanto ele permanecer dentro do ambiente (figura 24). Em algum momento o morador aciona novamente a chave, desligando a lâmpada, e deixando o cômodo. Assim o sensor de presença não detectando mais a presença do morador volta ao estado de repouso.

Figura 24 – Segunda etapa da interação do habitante com o sistema de iluminação IoT



Fonte: Autoria própria.

Na referida estrutura há claramente uma associação entre o acionar do interruptor captado pelo IoT-L, e a presença do morador no ambiente detectado pelo IoT-SP. Espera-se que o Hub observe essa associação, agrupe as publicações dentro dos limites de  $\Delta t$  e encontre algum modelo preditivo dado o comportamento do habitante.

A seção 2.2.1 mostrou que os nós sensores como dispositivos autônomos devem ser equipados com capacidades de sensoriamento e/ou atuação, processamento e comunicação. Nosso sensoriamento e atuação já estão definidos, nos restando a escolha dos componentes de processamento e comunicação.

Levando em conta os critérios da sequência de ações e a necessidade do uso de componentes capazes de condicionar os dados das leituras e prepará-los para transmissão, estudamos a possibilidade de empregar um microcontrolador, já que esse tipo de dispositivo

possue certos recursos que precisamos para que nossos IoT's trabalhem com sensores, atuadores e compartilhem dados em rede.

Concluimos que para assumir a atividade de controle em nossos nós IoT's, o microcontrolador deve no mínimo ser capaz de realizar duas tarefas. A primeira delas é a leitura e escrita em entradas e saídas digitais e analógicas a fim de controlar um *Relé*, obter dados de um *Sensor PIR* e do *Circuito Detector de Tensão*. A segunda tarefa é a capacidade de conexão por meio da pilha de protocolos *TCP/IP*.

Pelos motivos levantados e demais já apresentados na seção 2.3.1, a placa de desenvolvimento NodeMCU com seu chip ESP8266 foi escolhido como microcontrolador dos nós. No cenário residencial o NodeMCU leva vantagem se comparado por exemplo ao Arduino, pois ele é menor, mais rápido, mais barato e possui conexão Wi-Fi embutida, sem falar que é possível programar o *firmware* dele utilizando a IDE padrão do Arduino. Desta forma, o NodeMCU compreende o componente de comunicação por Wi-Fi.

Para alimentação dos NodeMCU's foram empregadas fontes normalmente utilizadas em *Smartphones*, elas transformam a tensão alternada de 127V ou 220V fornecida pela rede, em tensão contínua de 5V. Dispensamos o uso de baterias devido ao custo, tamanho elevados e o baixo tempo de operação, já que os nós foram projetados para funcionar 24 horas por dia conectados a rede sem fio, demandando um consumo considerável de energia.

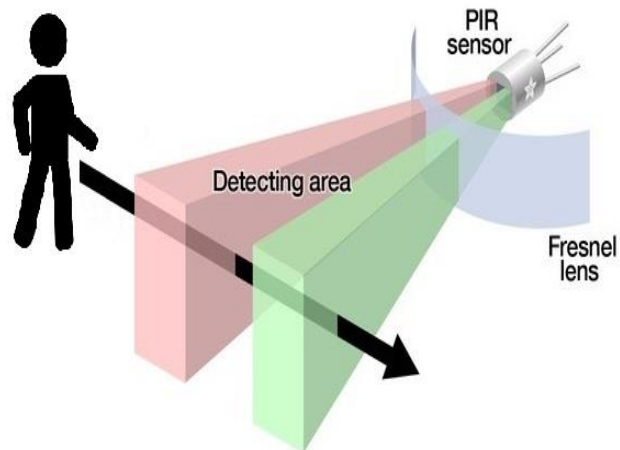
As próximas seções descrevem os componentes, conexões e a implementação do comportamento e funcionalidades dos nós IoT's considerados no esquema elétrico desta seção.

### 3.4 NÓ IOT-SP

Definimos o nó IoT-SP como uma combinação do sensor PIR com o placa NodeMCU. Discutiremos agora como esses dois elementos funcionam juntos e como foi realizado seu processo de integração.

Esse sensor de movimento PIR como é conhecido é um sensor infravermelho do tipo piroelétrico passivo, captando a radiação infravermelha do ambiente em seu campo de visão. Tudo que gera calor também produz radiação infravermelha, incluindo animais e os seres humanos. Essa radiação é invisível a olho nú, mas pode ser detectada com esse sensor. Assim, eles podem ser usados para informar a presença de alguém em seu alcance de detecção.

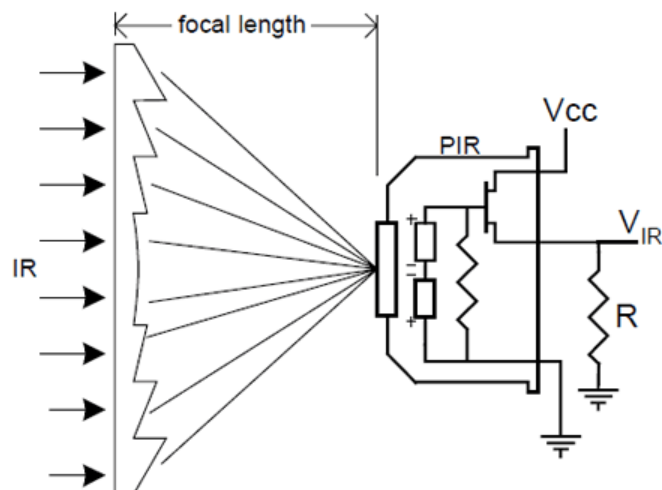
Figura 25 – Funcionamento dos sensores PIR



Fonte: Adaptado de Pir (2018).

O sensor PIR tem duas fendas nele, em cada fenda há um material especial sensível a radiação infravermelha. Quando o sensor está inativo, ambas as fendas recebem a mesma quantidade de radiação infravermelha (figura 25). Caso um corpo quente (humano ou animal) passe em frente ao sensor, uma das fendas receberá mais radiação infravermelha, provocando uma diferença de tensão positiva no sensor. Ao sair da área de detecção, a redução na radiação provoca o inverso, um diferencial negativo de tensão (PIR, 2018).

Figura 26 – Esquema interno do sensor PIR



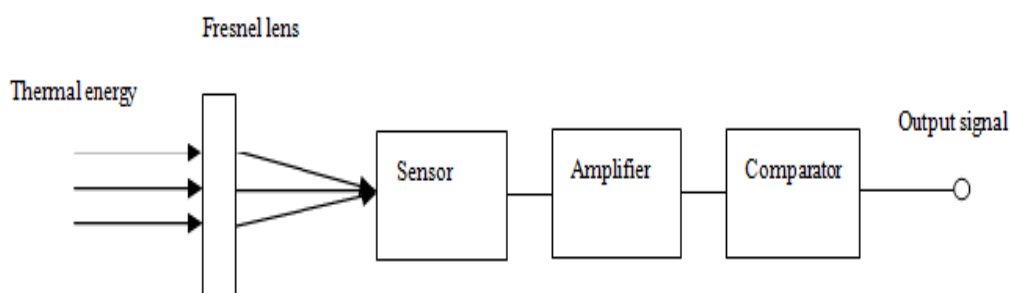
Fonte: Pir (2018).

Os materiais das duas fendas no sensor PIR são conectados em série (figura 26). Se uma metade recebe mais ou menos radiação infravermelha, a saída oscila alta ou baixa. A saída

dessas janelas serve como entrada para um JFET que produz uma tensão de saída  $V_{IR}$ . O componente chave para o funcionamento desse circuito é uma *Lente de Fresnel*, que converge os raios infravermelhos incidentes para a janela do sensor (PIR, 2018).

Para usar esses componentes precisamos ainda de um circuito de suporte que fará um processamento adicional do sinal  $V_{IR}$  (ver anexo 1). Isso é feito para evitar falsos positivos causados por exemplo, em um breve flash de uma câmera ou em um aumento na temperatura do ambiente (BATRINU, 2017). Esse circuito de suporte contém um chip, por exemplo o *BISS0001*, que coleta a saída  $V_{IR}$  do sensor, amplifica, compara e emite um pulso de saída digital (figura 27).

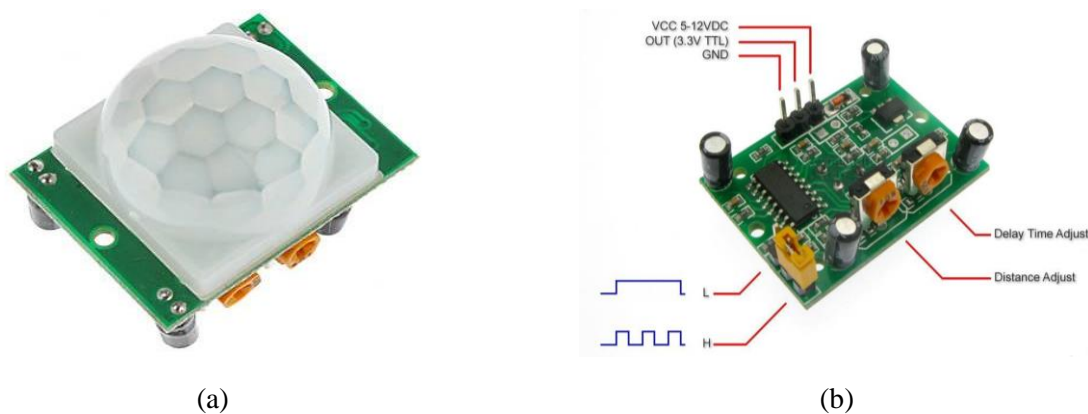
Figura 27 – Esquema do módulo sensor PIR



Fonte: Batrinu (2017).

Fabricantes vendem esse circuito para aplicações em microcontroladores no formato de módulo. Escolhemos o mais barato encontrado, o *HC-SR501 PIR*.

Figura 28 – Módulo HC-SR501, visão superior (a) e inferior (b)

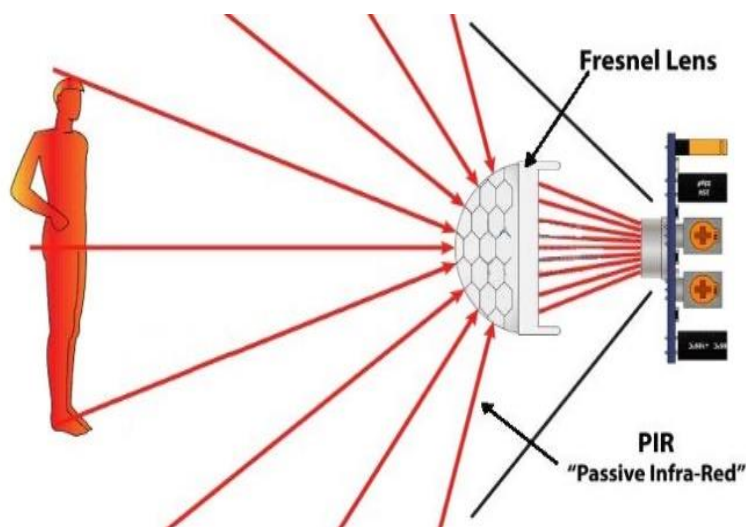


Fonte: Elkins, Sheffield e Weeks (2016).

O módulo possui três pinos (figura 28 - b): GND para conectar o terra (0V), o pino VCC para entrada da alimentação de 4,5 a 20V<sub>DC</sub> e o pino OUT que é a saída digital, esse pino mostrará um nível digital *baixo* de 0V quando nenhum movimento é detectado e *alto* de 3,3V quando movimento é detectado. O alcance de detecção e o tempo em que a saída permanece *alto* (atraso de tempo) são ajustados por potenciômetro. O alcance de detecção tem uma faixa ajustável de aproximadamente 3 a 7 metros. O atraso de tempo tem um intervalo ajustável de 5 segundos a 5 minutos, porém, após cada atraso de tempo a saída ficará *baixa* por cerca de 3 segundos. Também é possível ajustar o modo de disparo da saída através de um jumper de seleção que permite selecionar entre disparos simples, *modo L*, e repetíveis de vários impulsos de alto/baixo, *modo H* (ELKINS; SHEFFIELD; WEEKS, 2016). Qualquer um desses modos funcionará para o nó IoT-SP.

Na parte superior do módulo (figura 28 - a) a cúpula branca, no formato da metade de uma esfera, é a nossa Lente de Fresnel, a óptica por trás da estrutura dessa lente é explorada para aumentar o raio de detecção. Essa lente é multifacetada (várias zonas), isso “divide” a visão do mundo em cones menores de maior visibilidade e áreas intermediárias de menor visibilidade, ampliando drasticamente o ângulo útil de detecção. Com lentes de Fresnel de zona única a radiação eletromagnética de uma única área é focada (figura 26), desse modo, ao agrupar lentes de zona única juntas (WELBOURNE, 2016), cada lente forma uma janela de detecção diferente que direciona a radiação eletromagnética no sensor piroelétrico (figura 29).

Figura 29 – Comportamento da Lente de Fresnel do HC-SR501

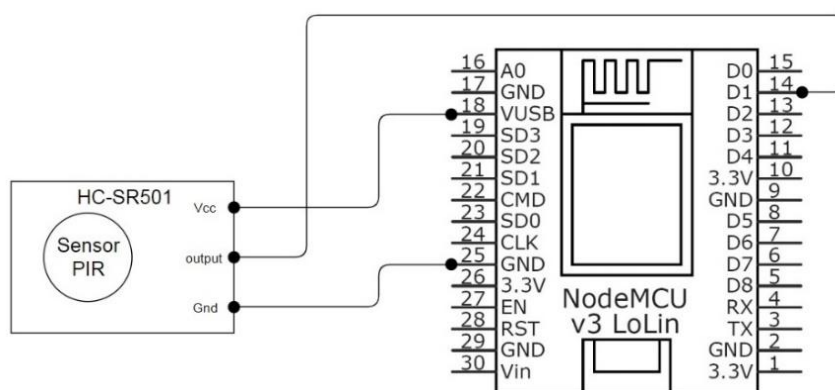


Fonte: Adaptado de Elkins, Sheffield e Weeks (2016).

Diante disso, segundo o fabricante do HC-SR501, temos um ângulo de detecção em cone de aproximadamente 110 graus, porém a folha de dados da lente que esse módulo utiliza (HW8002-2B) indica que o ângulo de detecção é de aproximadamente 100 graus na horizontal e 60 graus na vertical. Não sabemos qual dessas informações é a mais confiável. A combinação da lente (HW8002-2B) com o sensor piroelétrico (RCW-0506) no módulo pode produzir parâmetros diferentes. Essas informações são importantes no momento de posicionar o nó sensor dentro do cômodo da casa, pois ele precisa cobrir o máximo que conseguir da área, ou então, pontos cegos inibirão a atividade do nó. Essa é uma das dificuldades que encontramos no decorrer do projeto.

Sabendo que o circuito integrado do sensor PIR, quando alimentado faz uma leitura do infravermelho do ambiente e modula a partir dessa leitura um sinal de saída em nível digital, acoplamos o sensor PIR em nosso microcontrolador NodeMCU para que ele alimente esse sensor e também receba essa leitura em um de seus pinos digitais (figura 30).

Figura 30 – Esquema de ligação do nó IoT-SP

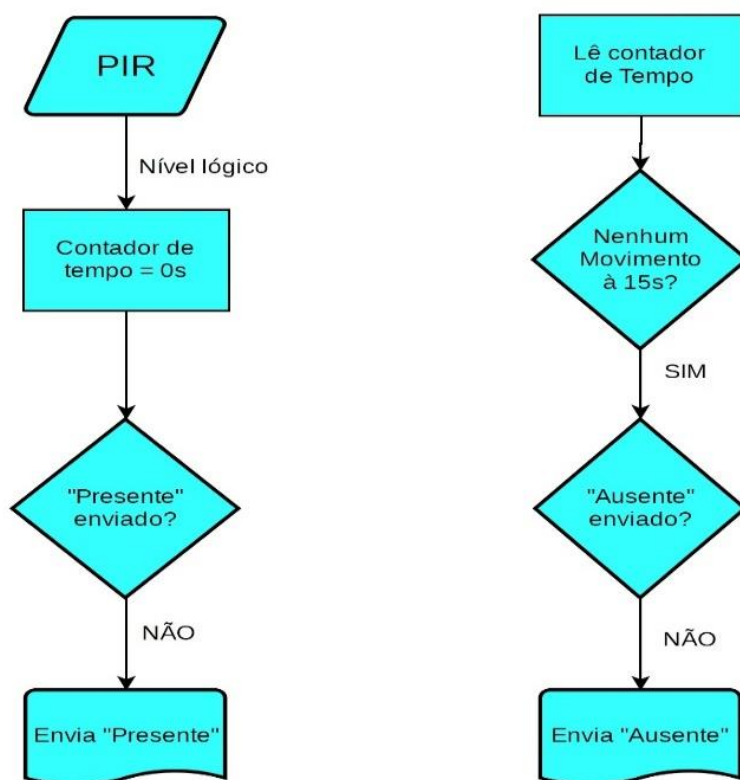


Fonte: Autoria própria.

A maioria absoluta dos dispositivos operam com tensões de alimentação na casa dos 5V, o mesmo vale para o HC-SR501 que precisa receber no Vcc tensão de no mínimo 4,5V. Mas o NodeMCU pode fornecer alimentação somente de 3,3V. Resolvemos esse problema ao alimentar o módulo com o pino 18 (VUSB), esse pino especial é uma saída de tensão em 5V quando a placa é alimentada através do conector micro USB. Os demais pinos utilizados são o pino 25 para o Gnd e a GPIO do pino 14 para o output, que é um sinal de saída em nível digital, do HC-SR501.

No nó IoT-SP o microcontrolador trabalhará da seguinte forma, primeiro o NodeMCU conecta-se ao Hub, em seguida ele aguarda a chegada de um nível lógico de presença vindo do PIR. O NodeMCU enviará dois valores qualitativos diferentes ao Hub conforme o comportamento desse nível lógico de presença, a primeira publicação é *Presente*, esse valor indica o morador dentro do cômodo, a outra publicação é *Ausente*, esse valor indica que o morador não está no cômodo (figura 31).

Figura 31 – Funcionamento do firmware do IoT-SP



Fonte: Autoria própria.

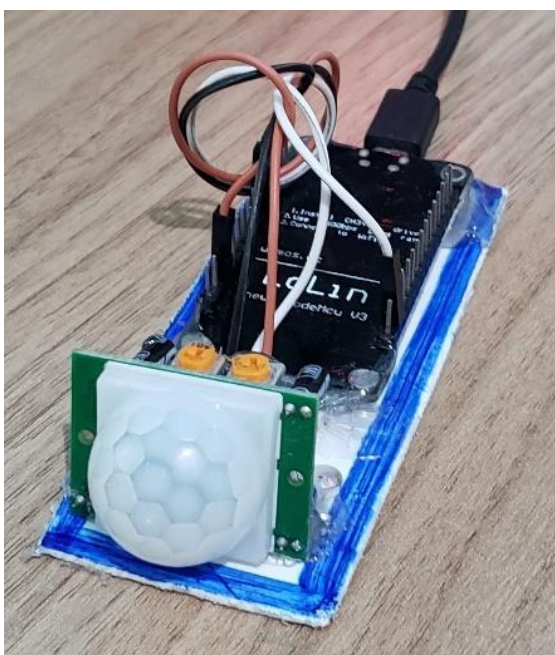
As publicações estão atreladas ao nível lógico de presença, mas seu envio ao Hub não é aleatório, não enviamos o valor *Presente* toda vez que o NodeMCU lê uma entrada alta de 3,3V no pino 14. O mesmo vale para a mensagem *Ausente* que não é enviada só com a leitura no pino 14 é de uma entrada baixa de 0V. É necessário o cuidado no envio dos dados para o Hub, pois estes serão utilizados como treinamento e possíveis erros poderão atrapalhar o processo de aprendizado.

Leituras de alto/baixo do nível lógico de presença nos indicam somente que movimento humano está ou não sendo detectado. Se o morador em algum momento ficar

totalmente parado, mesmo que dentro da área de detecção, o sinal de saída será baixo, pois o sensor PIR indica movimento e não presença.

Então fizemos o seguinte, antes de enviar o estado do sensor (Presente ou Ausente) verificamos se esse valor é o mesmo que já foi enviado antes, se o último valor enviado foi *Presente* só podemos enviar *Ausente* e vice-versa. E antes de enviar a valor que indica que o morador deixou o cômodo (Ausente) esperamos algum tempo. Se o sensor PIR (figura 32) não detectar nenhum movimento durante esse tempo publicamos o valor no Hub.

Figura 32 – IoT-SP em operação



Fonte: Autoria própria.

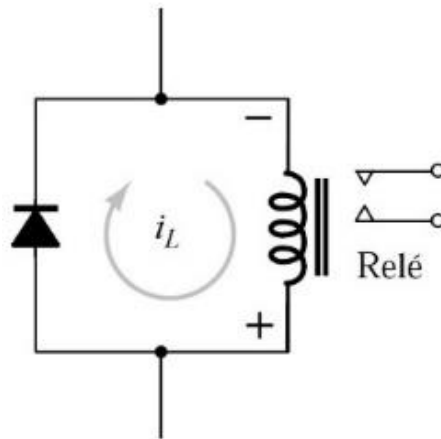
Programamos para que toda vez que um nível lógico de presença (alto) for recebido, uma rotina reinicia o *contador de tempo* e verifica se o último valor enviado para o Hub é o valor “Presente”, caso não seja, enviamos o valor “Presente” (anexo 2). O contador de tempo indica quanto tempo se passou desde a última vez que o morador foi detectado, a cada *loop* é feita a leitura desse tempo. Após 15 segundos, verificamos se o último valor enviado foi “Ausente”, não sendo, enviamos esse valor. O sensor PIR é muito sensível e acreditamos que ninguém ficaria com o corpo totalmente parado por muito tempo, por isso ajustamos uma espera de 15 segundos antes de enviar “Ausente”.

### 3.5 NÓ IOT-L

O nó IoT-L consiste em dois circuitos eletrônicos interligados a placa NodeMCU. O primeiro circuito possui um relé que chaveia a alimentação do circuito terminal de iluminação. O segundo circuito é um detector de tensão, e o seu papel é ler a tensão sobre o circuito de iluminação. Juntos, esses elementos estarão acoplados diretamente a rede elétrica residencial, cuja tensão elétrica é fornecida pela rede de distribuição local.

O primeiro componente do IoT-L em contato com a rede é o circuito de relé, um comutador eletromecânico usado para acionar um circuito elétrico de maior potência com a segurança do isolamento elétrico em relação ao circuito de menor potência. No circuito fechado da Figura 33 é injetada uma corrente elétrica que atravessa a bobina, a qual atua como um eletroímã quando energizada e aciona os contatos de saída (BOYLESTAD e NASHELSKY, 2004). Para proteger esse circuito da corrente de retorno da bobina é instalado um diodo protetor em paralelo.

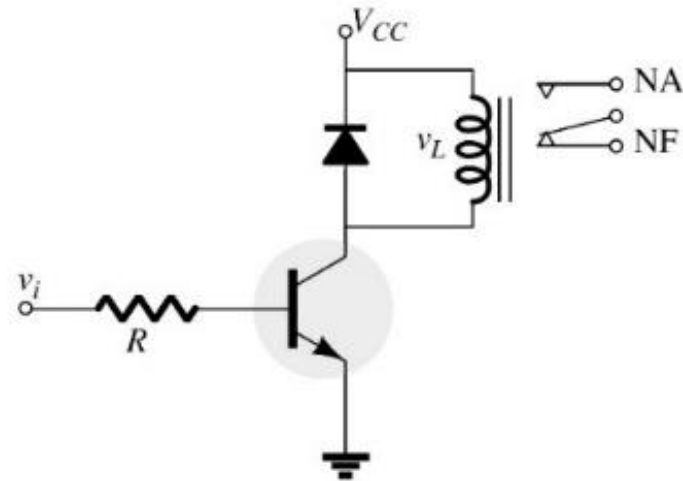
Figura 33 – Características de um Relé



Fonte: Boylestad e Nashelsky (2004).

Correntes bem pequenas são o suficiente para excitar o relé, então, podemos usar um transistor para chavear a corrente que irá passar em sua bobina. Nesse caso, ajustamos o transistor para que trabalhe na região de corte-saturação e lhe fornecemos uma corrente de base DC através do microcontrolador.

Figura 34 – Circuito acionador de relé

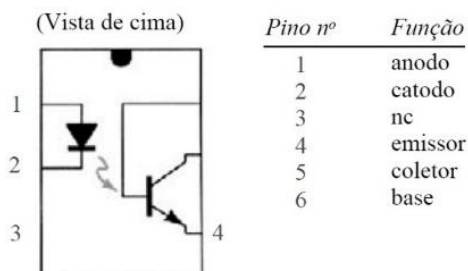


Fonte: Adaptado de Boylestad e Nashelsky (2004).

Quando um pulso positivo é aplicado na base, o transistor se liga, estabelecendo corrente suficiente através da bobina do eletromagneto para fechar o relé, durante esse estado do transistor, o diodo é polarizado reversamente, permanecendo como um circuito aberto (figura 34). No entanto, quando o transistor se desliga, a tensão na bobina se reverte e polariza diretamente o diodo, ligando-o. A corrente através do indutor estabelecida durante o estado ligado do transistor pode então continuar a fluir pelo diodo, eliminando a mudança brusca no valor da corrente, e minimizando os efeitos do *golpe indutivo* (BOYLESTAD e NASHELSKY, 2004).

Figura 35 – Optoisolador

ISO-LIT 1



Pastilha do LED no pino 2

Pastilha do fototransistor no pino 5

Fonte: Boylestad e Nashelsky (2004).

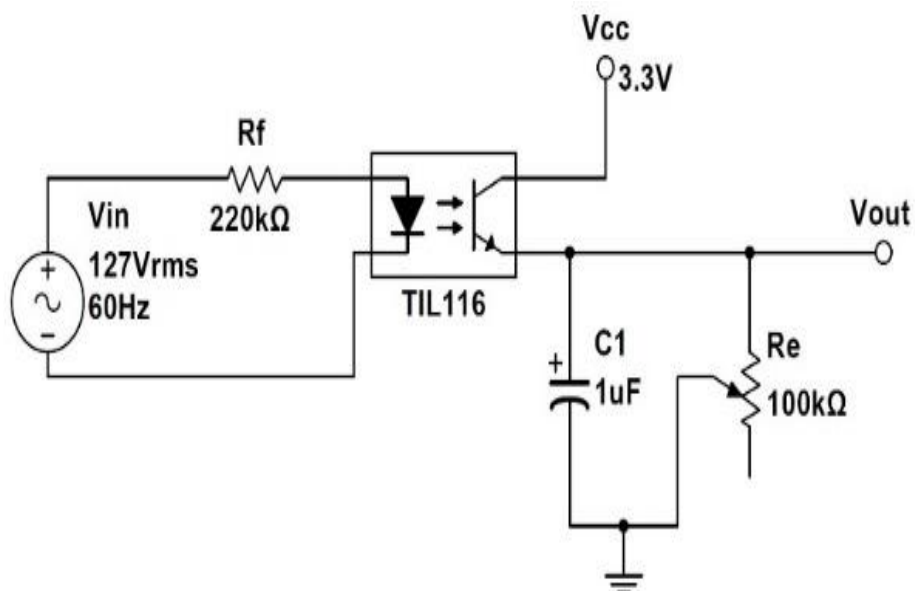
Para o segundo circuito, o circuito detector de tensão, a solução encontrada baseia-se no uso do componente eletrônico optoisolador (figura 35). Optoisoladores transferem energia entre dois circuitos utilizando a luz emitida de um LED para um fototransistor (BOYLESTAD e NASHELSKY, 2004).

Se bem projetado, um optoisolador é capaz de reduzir tensões de centenas de volts para valores de tensão de 5 ou 3,3V, possibilitando a leitura em microcontroladores.

Uma tensão do lado esquerdo produzirá uma corrente  $I_f$  no LED que, conforme a polarização do transistor no lado direito, produzirá o efeito de amplificador ou chave, com a vantagem de termos entrada e saída isoladas elétrica e magneticamente separadas. Esse componente eletrônico fará a interface entre NodeMCU e a eletricidade CA do circuito terminal da lâmpada.

Aplicando o optoisolador no circuito seguidor de emissor, temos um circuito detector de tensão (figura 36), que opera de duas maneiras diferentes. Com o optoisolador operando na região de corte-saturação, esse circuito terá em sua saída um nível lógico que indica a presença de tensão na entrada. Operando na região ativa/linear, esse circuito indicará, de forma grosseira, a amplitude da tensão na entrada.

Figura 36 – Diagrama do Circuito detector de tensão



Fonte: Autoria própria.

Usando o optoisolador TIL116 e um resistor de 220kΩ na entrada de 127Vrms, temos o circuito operando na região linear. Por operar nessa região, a corrente  $I_F$  e a corrente de coletor  $I_C$  estão vinculadas entre si através do *Current Transfer Rate* (CTR).

$$\frac{I_C}{I_F} = CTR \quad (1)$$

A corrente do coletor  $I_C$  é o produto da corrente  $I_F$  e o CTR.

$$I_C = CTR \times I_F \quad (2)$$

Para calcularmos essa corrente precisamos calcular a corrente  $I_F$ .

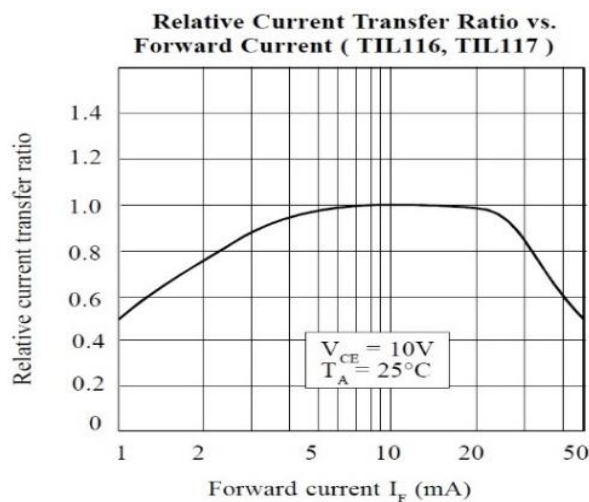
$$I_F = \frac{V_{IN}}{R_F} \quad (3)$$

A tensão na entrada é alternada, então, o LED capta o valor de tensão do semi-ciclo positivo da rede, na variação de 0V à  $V_p$ :

$$V_p = 127 \times \sqrt{2} = 179,6 \text{ V}$$

Pela a equação 3 encontramos um corrente de 0,81mA. Conforme a folha de dados, o TIL116 possui diferentes valores de CTR de acordo com a corrente  $I_F$  (figura 37).

Figura 37 – CTR por  $I_F$



Fonte: Isocom Components (2000).

Baseando-se no gráfico, calculamos a corrente  $I_c$  com o CTR da corrente mais próxima, no valor de 1mA:

$$I_c = 0,5 \times 0,81 \text{ m} = 0,4 \text{ mA} .$$

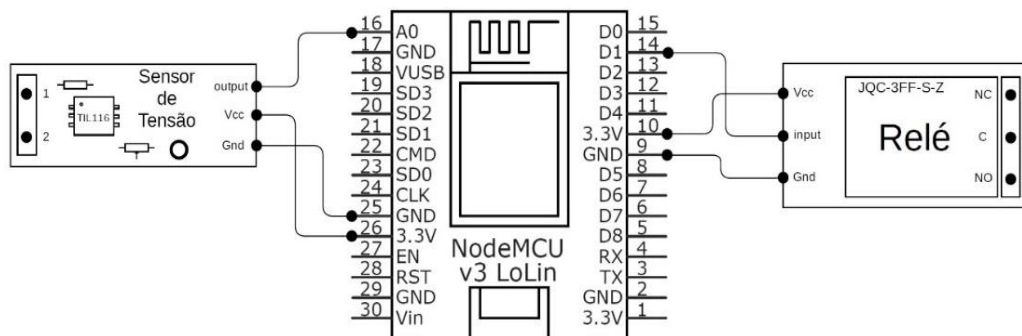
No optoacoplador, as correntes de coletor e de emissor são iguais, então, projetamos a tensão DC máxima de saída  $V_{OUT}$  para leitura no NodeMCU por dimensionar o potenciômetro  $R_E$  para o valor de 5k $\Omega$ :

$$V_{OUT} = 0,4 \text{ m} \times 5 \text{ k} = 2 \text{ V}$$

Através do conversor ADC do NodeMCU observamos experimentalmente que, para uma tensão de entrada de 123V, temos como saída uma tensão de 1,51V aproximadamente. O valor é ligeiramente diferente do calculado, por conta da suavização causada pelo capacitor de 1 $\mu$ F em paralelo.

Para manter o IoT-L pequeno adquirimos um módulo que agrega o circuito detector de tensão e outro módulo para o circuito de acionamento do relé e acoplamos esses dois módulos ao NodeMCU (figura 38).

Figura 38 – Esquema de ligação do nó IoT-L



Fonte: Autoria própria.

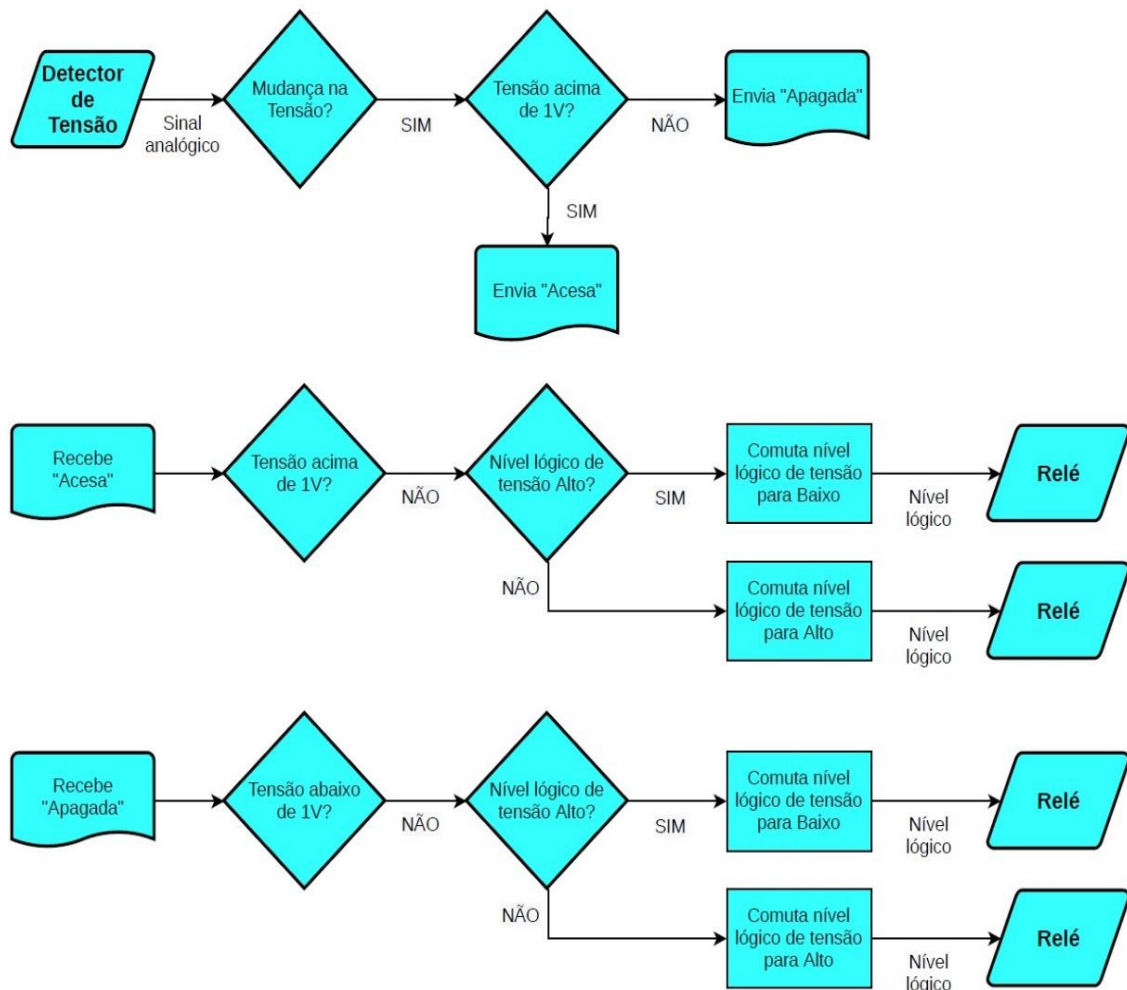
O NodeMCU saberá se a lâmpada está acesa ou apagada através da tensão analógica na saída do módulo detector tensão, podendo alterar essa condição por alterar o nível lógico na entrada do módulo relé, chaveando a tensão da lâmpada.

Ligamos a alimentação Vcc do módulo relé a saída do regulador de tensão interno do NodeMCU, o AMS1117 de 3,3V, no pino 10, e o Gnd foi conectado ao pino 9. Quanto a entrada do módulo, o input, receberá o nível lógico de tensão do NodeMCU que chaveia o relé, através da GPIO do pino 14.

O módulo detector de tensão é alimentado com tensão de 3,3V, então fizemos as ligações conectando o Vcc ao pino 26 e Gnd ao pino 25. Na saída do módulo a tensão DC é de 1V quando há tensão na entrada, quando nulo a tensão detectada é 0V, pois, o circuito terminal da lâmpada está em aberto. O NodeMCU lê essa tensão através do conversor ADC do pino 16.

Com o detector de tensão acoplado aos terminais do lâmpada, o NodeMCU informará imediatamente ao Hub que a lâmpada está acesa ao apagada sempre que o habitante chavear o interruptor tree-way, os valores qualitativos correspondentes a esse ato, que são publicados ao Hub, são *Acesa* e *Apagada* (figura 39).

Figura 39 – Funcionamento do firmware do IoT-L



Fonte: Autoria própria.

Quando modelos preditivos estiverem ativos, o Hub retro alimentará o IoT-L (figura 40) com as previsões que ele observar, essas previsões são informadas igualmente aos dados que o nó fornece, os valores *Acesa* e *Apagada*.

Figura 40 – IoT-L em operação



Fonte: Autoria própria.

A programação faz com que o NodeMCU atue conforme essas previsões, comutando os níveis lógicos de tensão na entrada do módulo de modo a chavear o relé (anexo 3). Se a previsão informada pelo Hub estiver correta, o IoT-L antecederá o habitante no ato de acender ou apagar a lâmpada.

### 3.6 HUB

Durante o primeiro passo do projeto, a investigação de quais recursos eram necessários para construir a HAN do sistema de iluminação, os sensores, atuadores, hardware e software dos nós IoT e do Hub, estudou-se a possibilidade de desenvolver o projeto do Hub na plataforma embarcada *Raspberry Pi*. A seleção dessa placa deu-se pelo baixo custo e pela facilidade de se trabalhar com ela através de seu sistema operacional oficial, o *Raspbian*.

### 3.6.1 Teste prévio

Antes de ter em mãos a placa almejamos executar no *Raspbian* um script python que nós programamos e que reproduz o segmento principal da simulação de Tonidandel e Sgarbi (2006), que consiste em criar dados de treinamento e produzir modelos, o resultado desse teste definiu a adoção ou não do Raspberry Pi.

O artifício adotado foi virtualizar o Raspbian em um computador com o OS Windows 7 de 64 bits através do software QEMU. O QEMU é um emulador e virtualizador de máquinas *open-source*.

Figura 41 – Script python da simulação correndo no Raspbian emulado por QEMU



Fonte: Autoria própria.

A virtualização do Raspbian através do emulador QEMU é possível graças a arquitetura ARM do processador do Raspberry Pi, para o qual esse OS foi concebido (figura 41). Acontece que o QEMU é capaz de emular uma arquitetura ARM muito semelhante ao do Raspberry Pi, que com ajustes adequados, nos permite inicializar uma imagem do OS Raspbian através do QEMU. Algumas referências (KIEPERT, 2013) detalham essa manobra de execução do QEMU com o Raspbian.

Assim esse novo simulador foi comparado com o simulador original. Os resultados mostraram que o simulador em python quando executado no Raspbian não sofreu com problemas de execução, sua interface correu com boa velocidade de resposta. Em vista disso aprovamos a escolha do Raspberry Pi para o desenvolvimento do Hub.

### 3.6.2 Implementação

Construímos uma aplicação com base no modelo cliente/servidor e transformamos nosso *Raspberry Pi 3 Model B* em um servidor Web capaz de interpretar e responder aos nós clientes da HAN.

O servidor foi desenvolvido em Python utilizando-se do framework *Tornado* dada a facilidade e versatilidade em escrever código com eles. Seria muito difícil desenvolver esse mesmo servidor em um NodeMCU onde o código é rigoroso e a capacidade de processamento e memória são menores.

Em nossa implementação abrimos uma conexão bidirecional entre nó IoT e Hub para que a publicação de dados, sua observação e eventual previsão aconteçam de forma *síncrona*. Escolhemos essa estratégia pois os IoT's que desenvolvemos não publicam dados periodicamente, somente quando alguma mudança significativa ocorre.

O firmware embarcado em cada nó IoT realiza, no início da operação, a solicitação da conexão ao servidor do Hub, autenticando o dispositivos através de um número de identificação (ID). Uma vez conectados, Hub e nós não precisarão de novas *requisições* para enviar ou receber dados, pois estarão unidos por um canal de comunicação persistente, a conexão por Websocket.

Fazemos nossas publicações serem entendidas pelo IoT-L, IoT-SP e Hub usando uma única representação de dados, o *JSON* (figura 42). Os valores qualitativos das variáveis IoT também são representados por ID's.

Figura 42 – Codificação de dados no conteúdo das publicações

```
1 {  
2   "valor": "4b0239b0-8836-11ea-a3ad-1c1b0df4d205"  
3 }
```

Fonte: Autoria própria.

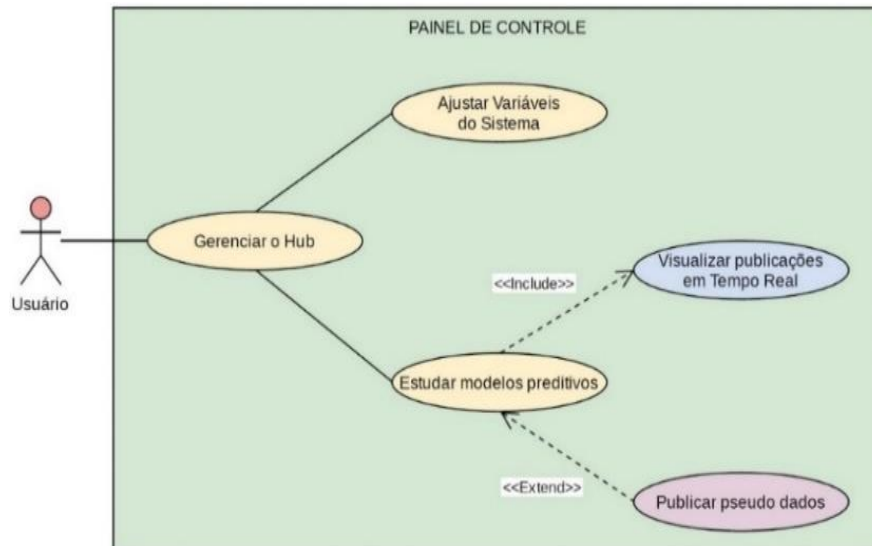
Uma representação de dados bem definida torna o Hub independente dos nós IoT's, se no futuro quisermos estudar regras de associação de outros dispositivos IoT só precisamos ajustar nossas publicações a esse formato em JSON que o Hub será capaz de tratá-los.

O Hub opera de forma autônoma, mas ele precisaria de algum tipo de interface para que consigamos observar o comportamento da HAN, esse recurso visual nos dá muitas

possibilidades, como por exemplo identificar problemas, estudar o comportamento dos modelos preditivos em tempo real ou ainda ajustar o método de aprendizado do Hub.

Para definirmos os recursos dessa interface foram levantados quais componentes precisaríamos e como eles se relacionam (figura 43).

Figura 43 – Casos de uso para o desenho da Interface



Fonte: Autoria própria.

Através dessa interface um usuário poderá gerenciar o funcionamento do Hub através de um *Navegador* em um computador (figura 44).

Figura 44 – Interface da tela principal do Hub



Fonte: Autoria própria.

Uma página é disponibilizada com recursos que permitem monitorar em tempo real a conexão dos dispositivos (figura 45).

Figura 45 – Interface de acompanhamento da conexão dos dispositivos

Lista

NOME	CONEXÃO	IP
IoT-L	conectado	192.168.1.4
IoT-SP	conectado	192.168.1.5

[voltar](#)

Fonte: Autoria própria.

Aprendizado e pontuação dos modelos preditivos também são exibidos, mas em outra página (figura 46).

Figura 46 – Interface de acompanhamento dos modelos preditivos

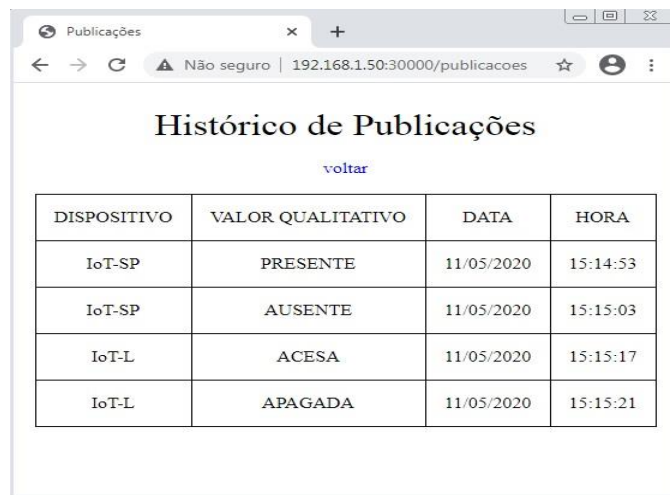
ASSOCIAÇÃO	DESFECHO	PREDITOR	PROPRIEDADES	MODELOS DE PREVISÃO												
b7bb7a30-8836-11ea-a3ad-1e1b0df4d205	IoT-L	<table border="1"> <tr> <td>SENSOR</td> <td>IoT-SP</td> </tr> <tr> <td>ATRIBUTO</td> <td>DIA</td> </tr> <tr> <td>ATRIBUTO</td> <td>TURNO</td> </tr> </table>	SENSOR	IoT-SP	ATRIBUTO	DIA	ATRIBUTO	TURNO	<table border="1"> <tr> <td>Δt</td> <td>20 ant + 30 dps</td> </tr> <tr> <td>Nº p/ aprendizado</td> <td>20</td> </tr> <tr> <td>Nº de OK p/ promoção</td> <td>5</td> </tr> </table>	Δt	20 ant + 30 dps	Nº p/ aprendizado	20	Nº de OK p/ promoção	5	nenhum
SENSOR	IoT-SP															
ATRIBUTO	DIA															
ATRIBUTO	TURNO															
Δt	20 ant + 30 dps															
Nº p/ aprendizado	20															
Nº de OK p/ promoção	5															

[voltar](#)

Fonte: Autoria própria.

Histórico de publicações (figura 47) e um ambiente onde é possível produzir pseudo publicações iguais as produzidas pelo nós IoT's para a realização de testes e experimentos (figura 48) em outras duas páginas.

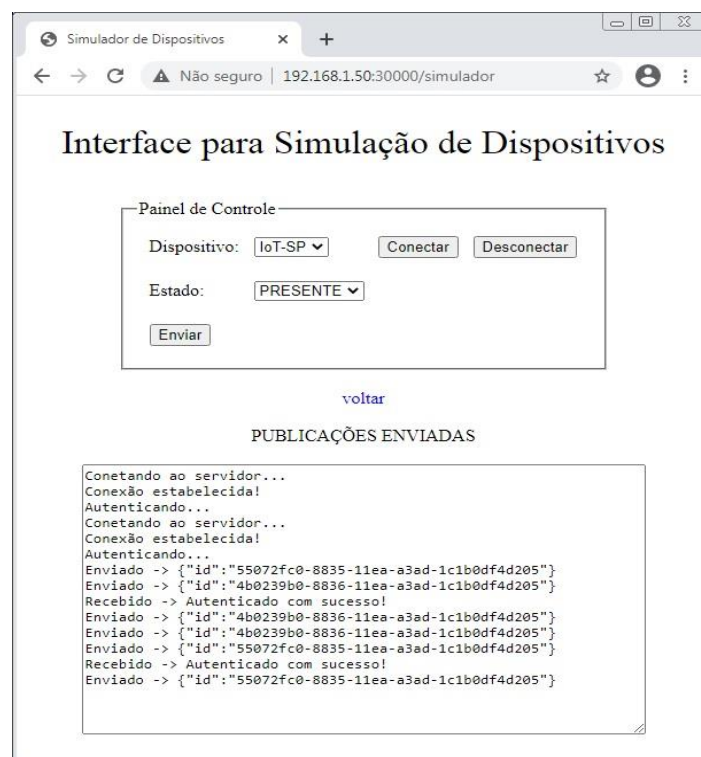
Figura 47 – Interface com o histórico de publicações



DISPOSITIVO	VALOR QUALITATIVO	DATA	HORA
IoT-SP	PRESENTE	11/05/2020	15:14:53
IoT-SP	AUSENTE	11/05/2020	15:15:03
IoT-L	ACESA	11/05/2020	15:15:17
IoT-L	APAGADA	11/05/2020	15:15:21

Fonte: Autoria própria.

Figura 48 – Interface para testes e experimentos



Painel de Controle

Dispositivo: IoT-SP

Estado: PRESENTE

[voltar](#)

PUBLICAÇÕES ENVIADAS

```

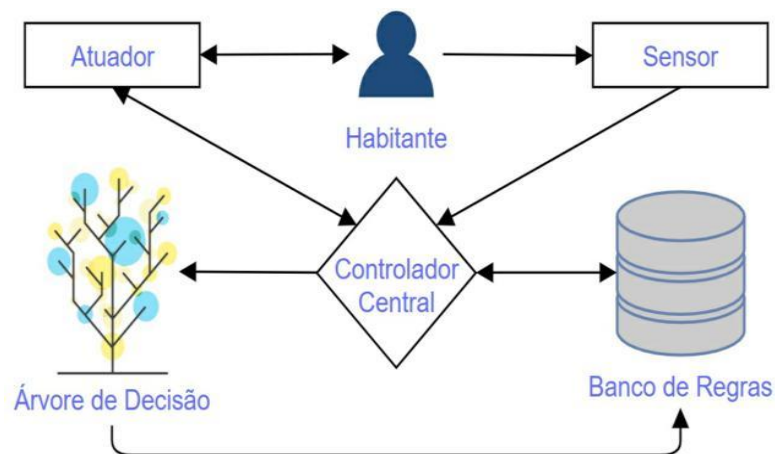
Conetando ao servidor...
Conexão estabelecida!
Autenticando...
Conetando ao servidor...
Conexão estabelecida!
Autenticando...
Enviado -> {"id":"55072fc0-8835-11ea-a3ad-1c1b0df4d205"}
Enviado -> {"id":"4b0239b0-8836-11ea-a3ad-1c1b0df4d205"}
Recebido -> Autenticado com sucesso!
Enviado -> {"id":"4b0239b0-8836-11ea-a3ad-1c1b0df4d205"}
Enviado -> {"id":"4b0239b0-8836-11ea-a3ad-1c1b0df4d205"}
Enviado -> {"id":"55072fc0-8835-11ea-a3ad-1c1b0df4d205"}
Recebido -> Autenticado com sucesso!
Enviado -> {"id":"55072fc0-8835-11ea-a3ad-1c1b0df4d205"}

```

Fonte: Autoria própria.

Introduzimos o raciocínio de aprendizado e pontuação dos modelos preditivos (regras de classificação) por meio da releitura (figura 49) do sistema de automação baseado em comportamento (TONIDANDEL e SGARBI, 2006). Codificamos o método na linguagem Python e intruzimos junto a ele uma versão que copilamos para OS Raspbian do algoritmo C4.5 Release 8 de Quinlan (1993).

Figura 49 – Método de ilação dos modelos preditivos



Fonte: Pereira, Oliveira e Costa (2019).

O processo de publicação de dados que os nós IoT's realizam existe justamente para que o Hub possa observar as regras de associação entre os elementos. Os dados das publicações são agrupados e armazenados, constituindo o conjunto de treinamento. No conjunto de treinamento os atributos são os nós IoT's preditores, os *valores de atributo* são o valor publicado por eles, e as classes são o valor publicado pelo IoT desfecho. Além disso, *atributos adicionais* regulados pelo próprio Hub podem ser usados para potencializar nossas regras de associação.

Durante o processo de introdução de uma linha do conjunto de treinamento, o algoritmo interno do Hub verifica se nas propriedades da regra de associação existe um atributo adicional, se existir, o Hub usa de seus recursos para descobrir o valor atual desse atributo e colocá-lo junto aos outros valores de atributo. São exemplos de atributos adicionais: a temperatura ambiente da região, a quantidade de pessoas próximas aos nós IoT's, a estação do ano, a tarifa de energia elétrica, o preço do combustível ou qualquer outro dado interno, externo, público ou privado que o Hub tenha acesso (figura 50).

Figura 50 – Raspberry produzindo o serviço de Hub



Fonte: Autoria própria.

Por fim, o Hub usará os modelos de previsão em suas observações para prever o valor e o momento exato da publicação do IoT desfecho. Um modelo de previsão recém criado é chamado de *embrionário*, caso o IoT desfecho publique dados iguais ao modelo, pontos positivos são atribuídos a ele, caso contrarie o modelo pontos negativos são atribuídos. Ao longo do tempo o modelo embrionário acumulará pontos, caso detenha muitos pontos positivos ele passará ao banco de modelos ativos, se tiver muitos pontos negativos o modelo de previsão é excluído. Quando o modelo de previsão estiver no banco de modelos ativos ele será efetivamente aplicado, o Hub enviará ao IoT desfecho o valor da previsão e esse IoT se encarregará de tomar a decisão do que fazer com ela. O banco de modelos ativos manterá o sistema de pontuação, a única alteração é: se tiver muitos pontos negativos o modelo de previsão não será excluído, somente rebaixado ao banco de modelos embrionários.

### 3.7 PONTO DE ACESSO

Atualmente muitas residências possuem algum ponto de acesso, do inglês *Access Point* (AP), que é uma estação-base ou roteador que repassa os pacotes entre os dispositivos sem fio conectados a ele e também entre eles e a Internet. Para isso, o ponto de acesso estabelece uma rede local, LAN, que opera dentro de um ambiente, como uma residência ou um escritório, e é exatamente por conta dessa rede sem fio que o termo Wi-Fi é amplamente conhecido (TANENBAUM, 2011). Os dispositivos IoT deste projeto foram inseridos em um ambiente residencial real, que conta com AP e disponibiliza rede Wi-Fi, assim, essa estrutura de rede passou a ser utilizada também como HAN (figura 51).

Figura 51 – AP da residência



Fonte: Autoria própria.

O firmware do microcontrolador de cada nó foi programado para, quando conectado a fonte de alimentação ser iniciado e realizar a conexão com o ponto de acesso Wi-Fi da casa utilizando SSID (usuário) e *password* (senha) que estão pré-definidos em sua programação. Desse modo, o microcontrolador recebe um IP do ponto de acesso, gerado randomicamente por DHCP. Isso é necessário para que os nós IoT's atuem como *clientes* capazes de encontrar o *servidor* do Hub. Não foi necessário nenhuma configuração adicional no ponto de acesso da HAN.

Ao conseguir se conectar ao ponto de acesso, os IoT's utilizam a LAN para conversar com a aplicação do Raspberry Pi, ou seja o Hub. A biblioteca que utilizamos na programação da conexão, dos microcontroladores com a aplicação do Hub, usa o IP e a Porta do Hub para encaminhar requisições. Tendo conseguido conectar-se ao Wi-Fi e estabelecer um canal de comunicação síncrono com o Hub, a programação dos IoT's entra em loop, e em cada repetição é verificado se essas conexões ainda persistem, caso contrario o microcontrolador tenta restabelece-las.

Quanto a segurança, contamos com o fato de as LANs serem redes privadas de acesso restrito, então, sua rede local não permite acesso externo a uma WAN ou a Internet. Os endereços IP dos dispositivos conectados a LAN só tem significado dentro de sua rede local, não podem ser utilizados na Internet (KUROSE, 2010). O acesso externo aos nós IoT ou ao Hub só pode ser feito, nesse caso, através de um proposital *network address translation* (NAT).

## 4 RESULTADOS EXPERIMENTAIS

A fim de validar a arquitetura publicador-observador e a implementação da HAN, foram conduzidos alguns experimentos e uma observação de ambiente real. O experimentos exploram a interação da domótica com um habitante fictício, por outro lado, o experimento final utiliza o comôdo de uma casa real, onde foram instalados os dispositivos IoT. Desse modo, verificamos o desempenho da arquitetura publicador-observador frente as ações do habitante e seu comportamento particular, diante da *regra de associação* ajustada para o sistema de iluminação.

### 4.1 PESQUISA EXPERIMENTAL

Para a realização do experimento o Hub foi configurado do seguinte modo, primeiro inseriu-se no Hub a regra de associação  $\{IoT - SP\} \rightarrow \{IoT - L\}$ , que indica o IoT-SP, como preditor do IoT-L desfecho. Segundo, indicou-se ao Hub que ele deveria inferir as regras de classificação quando o banco de dados atingi-se um total de 20 elementos. Terceiro, ajustou-se o agrupamento de publicações, para um  $\Delta t$  de 20 segundos antes e 30 segundos depois da publicação do nó desfecho.

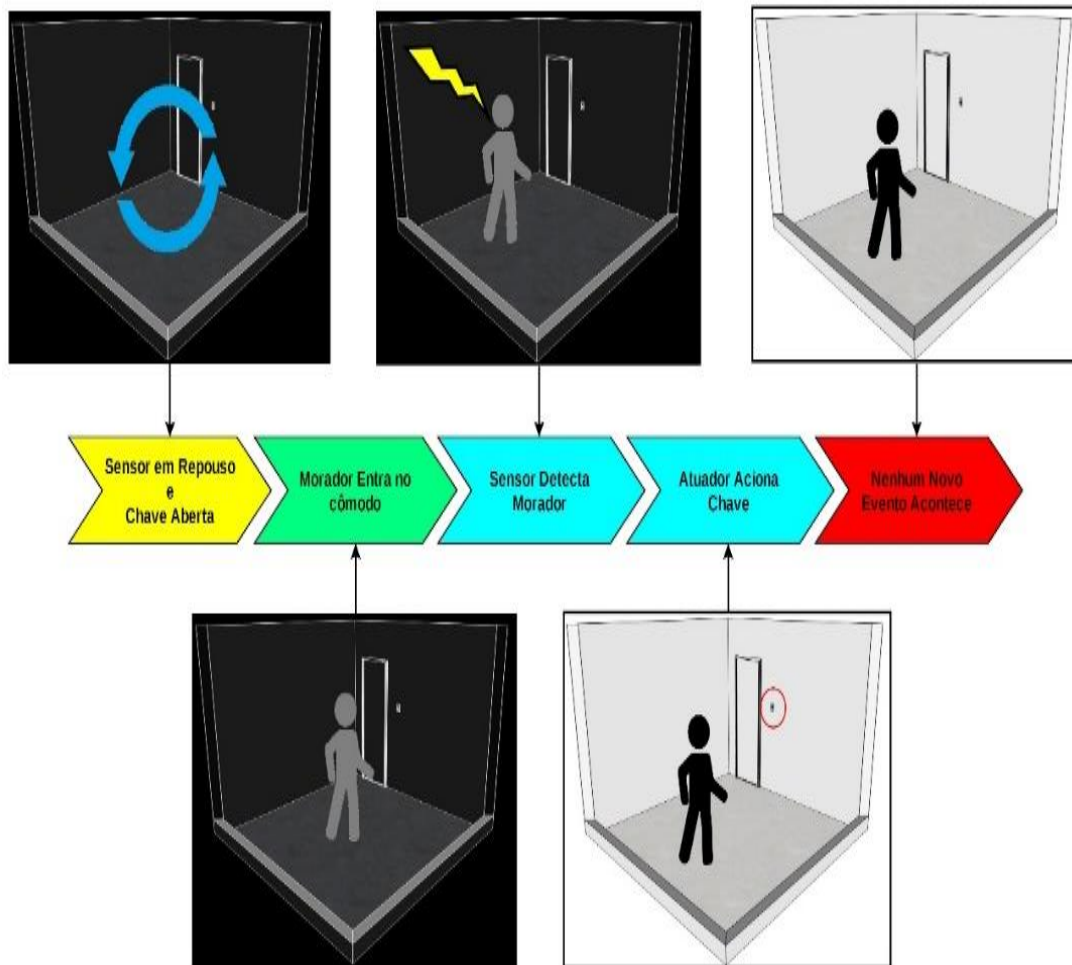
Para proporcionar um conjunto de treinamento com entropia máxima, produzimos 20 publicação para o IoT-SP coordenadas com 20 publicações do IoT-L, associando os valores qualitativos na forma, ‘presente’ com ‘acesa’ e ‘ausente’ com ‘apagada’ (tabela 1).

Nesse experimento o Hub aprendeu, ao mesmo tempo, dois modelos de previsão. Vamos entender o que essa previsão produz.

O primeiro modelo preditivo inferido pelo aprendizado, a regra de classificação  $\{IoT - SP = presente\} \rightarrow \{IoT - L = acesa\}$ , adverte que o IoT-L deve indicar ‘acesa’ logo após o IoT-SP informar ‘presente’.

Por conta do primeiro modelo preditivo, a domótica se comporta da seguinte forma: a lâmpada, que está inicialmente desligada, é acensa pelo IoT-L, sempre que o IoT-SP, que esta em repouso, sinalizar a detecção da presença do morador no cômodo. A presença desse indivíduo segue sendo detectada pelo PIR enquanto ele permanecer dentro do ambiente e a chave do módulo relé permanece na posição que está (figura 52).

Figura 52 – Primeira etapa da interação do habitante com o sistema de iluminação IoT após o aprendizado do modelo preditivo

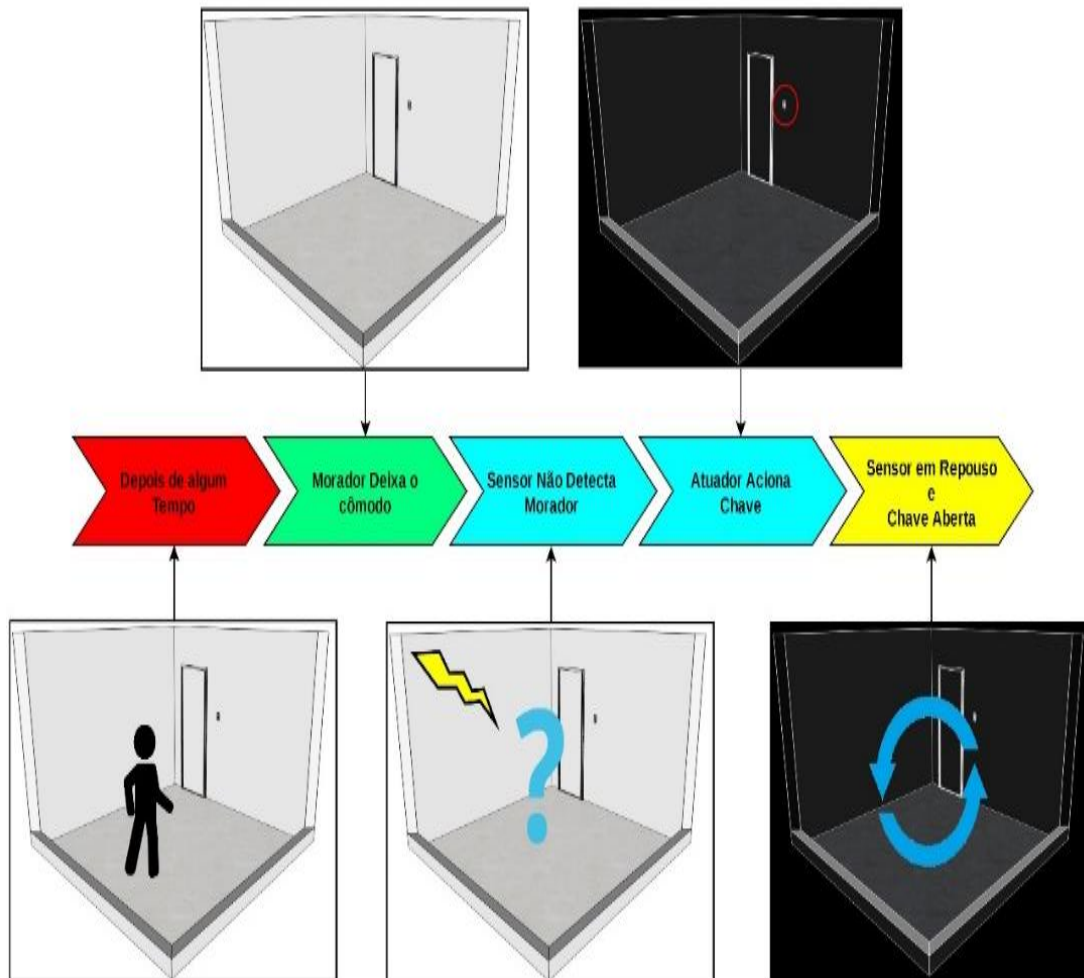


Fonte: Autoria própria.

O segundo modelo de previsão aprendido, a regra de classificação  $\{IoT - SP = ausente\} \rightarrow \{IoT - L = apagada\}$ , diz que o IoT-L indicará ‘apagada’ após o IoT-SP informar ‘ausente’.

A leitura que fazemos do segundo modelo de previsão encontrado é: no momento que o morador deixar o cômodo, o PIR não detectará mais sua presença, e em consonância com o segundo modelo preditivo, o IoT-SP voltará ao estado de repouso após indicar ao Hub a saída do habitante, assim o IoT-L automaticamente aciona o relé desligando a lâmpada (figura 53).

Figura 53 – Segunda etapa da interação do habitante com o sistema de iluminação IoT após o aprendizado do modelo preditivo



Fonte: Autoria própria.

Experimentalmente esse modelo de previsão funciona muito bem, mas na prática ele não atuaria pois sofre de um sério problema, em seu estado ativo ele não pontua. Isso se deve ao fato de o IoT-SP demorar 15 segundos para indicar que o habitante deixou o ambiente, nesse meio tempo, o habitante por si mesmo, já teria apagado a lâmpada. Portanto, mesmo que uma previsão exista, ela depende do quão rápido os preditores são em fornecer ao Hub o valor qualitativo associado.

Esse fato não influencia nos experimentos, então, partimos para o próximo experimento, onde produzimos publicações de modo mais seletivo. Agora elaboramos um habitante que tem o hábito de acionar a lâmpada somente no período noturno, sendo que, os modelos preditivos inferidos no experimento anterior eram aplicados em qualquer horário.

Com os modelos pontuando positivamente durante a noite, mas negativamente durante o dia, a observação mostrou que a associação  $\{IoT-SP\} \rightarrow \{IoT-L\}$  não pôde tornar nenhum modelo preditivo ativo, isso mostra que a associação que criamos é fraca. Esse mesmo fenômeno acontece com sensores de presença com soquete para lâmpada, pois assim como nossos modelos preditivos, eles se utilizam do conceito raso de que há uma relação de causalidade entre presença humana e iluminação, desconsiderando outras possíveis variáveis.

Para corrigir isso, adicionamos mais um preditor a regra de associação, o atributo *Turno*. Esse recurso interno do Hub é extraído do horário exato do recebimento da publicação do IoT desfecho. Estipulamos quatro valores qualitativos a partir dos valores quantitativos do horário (quadro 3).

Quadro 3 – Atributo adicional inserido na regra de associação

Atributo	Dado publicado
Turno	Mad (00:00h às 05:59h)
	Man (06:00h às 11:59h)
	Tar (12:00h às 17:59h)
	Noi (18:00h às 23:59h)

Fonte: Autoria própria.

Com a nova regra de associação em mãos  $\{IoT-SP, Turno\} \rightarrow \{IoT-L\}$ , observou-se novamente o que aconteceria a domótica durante três dias.

Como resultado foram criados os modelos preditivos semelhantes aos anteriores, mas com a diferença de termos o atributo ‘Turno’ indicando ‘noi’ inserido no modelo:  $\{IoT-SP = presente, Turno = noi\} \rightarrow \{IoT-L = acesa\}$ ,  $\{IoT-SP = ausente, Turno = noi\} \rightarrow \{IoT-L = apagada\}$ .

## 4.2 EXPERIMENTO FINAL

Antes de iniciar o experimento da HAN com o habitante real, foi inspecionada a presença da rede sem fio Wi-Fi nos pontos de instalação dos dispositivos, bem com verificado se os nós IoT e o Hub estavam conectados (figura 54).



quando o banco de dados atingi-se um total de vinte elementos (tabela 1), e o  $\Delta t$  com vinte segundos antes e trinta segundos depois da publicação do nó desfecho. O experimento final durou 8 dias.

Tabela 1 – Conjunto de treinamento coletado

<b>IoT-SP</b>	<b>Dia</b>	<b>Turno</b>	<b>IoT-L</b>
PRESENTE	SEG	NOI	ACESA
AUSENTE	SEG	NOI	APAGADA
ND	SEG	NOI	APAGADA
PRESENTE	TER	MAN	ACESA
ND	TER	MAN	APAGADA
PRESENTE	TER	NOI	ACESA
PRESENTE	TER	NOI	APAGADA
PRESENTE	TER	NOI	ACESA
AUSENTE	TER	NOI	APAGADA
ND	QUA	MAN	ACESA
ND	QUA	MAN	APAGADA
PRESENTE	QUA	NOI	ACESA
PRESENTE	QUA	NOI	APAGADA
PRESENTE	QUA	NOI	ACESA
AUSENTE	QUA	NOI	APAGADA
PRESENTE	QUA	NOI	ACESA
ND	QUA	NOI	APAGADA
PRESENTE	QUA	NOI	ACESA
PRESENTE	QUA	NOI	APAGADA
AUSENTE	QUI	MAN	ACESA

Fonte: Autoria própria.

A partir daí, como mostram os dados obtidos com a HAN em produção, dada a interação do habitante com o sistema de iluminação IoT, constatou-se que o Hub observou

corretamente os eventos dos nós por agrupar os dados das publicações formando uma base de dados para o treinamento suficiente para inferir modelos preditivos (figura 56).

Figura 56 – Arquivo de saída do algoritmo de aprendizado

```

C4.5 [release 8] rule generator Thu Nov 12 06:09:52 2020
-----
Options:
  File stem <aprendizado\dia_12_11_2020_hora_06_09_52_atuado
Read 20 cases (3 attributes) from aprendizado\dia_12_11_2020_hora
Processing tree 0
Final rules from tree 0:
Rule 2:
  IoT-SP = ND
  -> class APAGADA [54.6%]
Rule 1:
  IoT-SP = PRESENTE
  -> class ACESA [57.9%]
Default class: APAGADA

Evaluation on training data (20 items):
Rule  Size  Error  Used  Wrong  Advantage
-----
  2    1   45.4%   5     1 (20.0%)  0 (0|0)  APAGADA
  1    1   42.1%  11     3 (27.3%)  5 (8|3)  ACESA

Tested 20, errors 5 (25.0%) <<

(a) (b) (c) <-classified as
-----
  8   2   (a): class ACESA
  3   7   (b): class APAGADA
        (c): class ND

```

Fonte: Autoria própria.

O Hub aprendeu um dos modelos de previsão que desenvolvemos nos experimentos, assim como era esperado. O modelo consiste na seguinte regra de classificação:  $\{IoT - SP = presente\} \rightarrow \{IoT - L = acesa\}$ . Esse modelo preditivo embrionário foi pontuado até atingir o status de ativa (figura 57).

Figura 57 – Modelo de previsão obtidos com o habitante real

Situação das Regras de Associação										
ASSOCIAÇÃO	DESEFECHO	PREDITOR		PROPRIEDADES		MODELOS DE PREVISÃO				
b7bb7a30-8836-11ea-a3ad-1c1b0df4d205	IoT-L	SENSOR	IoT-SP	Δt	20 ant + 30 dps	STATUS	TOMADA DE DECISÃO	SENSORES	OK/ATIV	NOK/EXC
		ATRIBUTO	DIA							
		ATRIBUTO	TURNO	Nº p/ aprendizado	20					
				Nº de OK p/ promoção	5					

[voltar](#)

Fonte: Autoria própria.

Como se vê no histórico de publicações, a predição teve êxito em acionar a Lâmpada para o habitante. A atividade de previsão exercida pelo Hub passou a afetar os eventos que ocorriam no ambiente, isso por que, as previsões indicadas ao IoT-L fizeram com que esse nó acendesse a lâmpada de forma inerente ao comportamento usual do habitante.

Porém, o segundo modelo de previsão, a regra de classificação  $\{IoT - SP = ausente\} \rightarrow \{IoT - L = apagada\}$ , não foi inferida. Ainda que fosse criada, como vimos no primeiro experimento, essa previsão seria sem efeito.

Quanto aos atributos adicionais, não foram atribuídos ao único modelo de previsão criado. Isso demonstra que há uma expressiva desordem nos valores de atributo do conjunto de treinamento, exceto quando temos  $IoT - SP = presente$ .

## CONCLUSÃO

A principal contribuição deste trabalho é o estudo em ambiente real de uma arquitetura que possibilita não só gerenciar dispositivos da IoT com sensores e atuadores, como também induzir modelos de previsão úteis mediante coleta, processamento e análise de dados da própria residência em método de aprendizado de máquina. Soluções como essa permitem que a domótica se adéque as necessidades da casa, ao contrario de outras propostas, onde as necessidades precisam ser adaptadas a solução.

A proposta conseguiu empregar uma domótica que faz uso de dados qualitativos de fenômenos relevantes em uma HAN como fonte de informações a respeito do contexto da residência para a criação de modelos preditivos que ela mesma utiliza.

Este trabalho mostra que é possível usar Hub's para aprendizado, previsão e tomada de decisão na domótica de uma HAN. Além disso, com o devido projeto, a idéia pode suportar escalabilidade, desse modo, promoveria a integração de vários dispositivos de forma genérica, podendo ser utilizada em vários cenários.

O ponto negativo é que essa solução está longe de ser *plug and play*, esse sistema exige que o ajuste das configurações de aprendizado, como a regra de associação, intervalo de observação, variáveis de ajuste do conjunto de treinamento e do banco de dados, para as variações de condicionamento em cada aplicação diferente destinada ao Hub, sejam definidos por alguém que entenda dos dispositivos, fenômenos envolvidos e sua relação com o ambiente para o qual está sendo inserido, caso contrário, o Hub não aprenderá nada ou aprenderá modelos preditivos que nunca se tornam ativos pela baixa taxa de acerto.

Uma mecânica adicional que faria todo o sentido nessa proposta seria a de aplicar mineração de dados a todos os dados publicados pelos nós IoT, a fim de que o Hub encontre sozinho as regras de associação.

Tendo em vista a tendência em implantar a tecnologia de IoT em várias áreas do cotidiano das pessoas e em razão do volume e variedade de dispositivos conectados à rede estar aumentando, há uma crescente demanda por soluções que promovam gerenciamento, controle automatizado na borda ou na nuvem e operação dos dispositivos baseada em previsão, a mesma linha de pensamento da solução desenvolvida neste trabalho.

## REFERÊNCIAS

ACETOZI, Jorge. Pro Java Clustering and Scalability: Building Real-Time Apps with Spring, Cassandra, Redis, WebSocket and RabbitMQ. Apress, 2017.

ADRYAN, Boris; OBERMAIER, Dominik; FREMANTLE, Paul. The Technical Foundations of IoT. Artech House, 2017.

AGRAWAL, Rashmi; PAPRZYCKI, Marcin; GUPTA, Neha. Big Data, IoT, and Machine Learning: Tools and Applications. CRC Press, 2020.

AL-TURJMAN, Fadi; IMRAN, Muhammad. IoT Technologies in Smart-Cities: From sensors to big data, security and trust. The Institution of Engineering and Technology, London, United Kingdom, 2020.

ANDRADE, Cesar A. B. de. Análise Automática de Malwares Utilizando as Técnicas de Sandbox e Aprendizado de Máquina. Dissertação (Mestrado em Sistemas e Computação) - Instituto Militar de Engenharia, 2013.

BATRINU, Catalin. ESP8266 Home Automation Projects. 2017.

BINH, Huynh Thi Thanh; DEY, Nilanjan. Soft Computing in Wireless Sensor Networks. CRC Press, 2018.

BOYLESTAD, Robert L.; NASHELSKY, Louis. Dispositivos Eletrônicos e Teoria de Circuitos. 8ª Edição. São Paulo, 2004.

BRAGA, P. B.; CARVALHO, F. L.; LUDEMIR, T. B. Redes neurais artificiais: teoria e aplicações. 2. ed. Rio de Janeiro, 2007.

CHEW, Daniel. The Wireless Internet of Things: A Guide to the Lower Layers. John Wiley & Sons, 2018.

CHOPRA, Varun. WebSocket Essentials—Building Apps with HTML5 WebSockets. Packt Publishing Ltd, 2015.

COOK, Diane; DAS, Sajal Kumar. Smart environments: Technology, protocols and applications. John Wiley & Sons, 2004.

DANGETI, Pratap. Statistics for machine learning. Packt Publishing Ltd., 2017.

DENNIS, Andrew K. Raspberry Pi home automation with Arduino. USA, Packt Publishing, 2013.

DINOV, Ivo D. Data science and predictive analytics: Biomedical and health applications using R. Springer, 2018.

EICHHORN, Daniel. ESP8266 weather station: getting started guide. E-book Daniel Eichhorn, 2018.

ELHOSENY, M.; HASSANIEN, A. E. Dynamic wireless sensor networks: New directions for smart technologies. Published in. Studies in Systems, Decision and Control, 2019.

ELKINS, Monta; SHEFFIELD, Eddie; WEEKS, Thomas. Arduino Cookbook. Let's Code Blacksburg, 2016. Disponível em: < [https://github.com/LetsCodeBlacksburg/arduino-recipes/blob/master/2016-07-11\\_LCBB\\_Arduino\\_Cookbook.pdf](https://github.com/LetsCodeBlacksburg/arduino-recipes/blob/master/2016-07-11_LCBB_Arduino_Cookbook.pdf)>. Acesso em: 07 de dez. de 2019.

- FROIZ-MÍGUEZ, Iván et al. Design, implementation and practical evaluation of an IoT home automation system for fog computing applications based on MQTT and ZigBee-WiFi sensor nodes. *Sensors*. 2018.
- GERALDO FILHO, P. R. et al. ResiDI: Towards a smarter smart home system for decision-making using wireless sensors and actuators. *Computer Networks*, 2018.
- HANES, David et al. IoT fundamentals: Networking technologies, protocols, and use cases for the internet of things. Cisco Press, 2017.
- HARRINGTON, William. Learning Raspbian. Packt Publishing Ltd. 2015.
- HUANG, Hantao; YU, Hao. Compact and Fast Machine Learning Accelerator for IoT Devices. Springer, 2019.
- HULTEN, Geoff. Building Intelligent Systems: A Guide to Machine Learning Engineering. Apress, 2018.
- ISOCOM COMPONENTS. Datasheet: TIL111X, TIL114X, TIL116X, TIL117X, TIL111, TIL114, TIL116, TIL117. Publicação eletrônica, 2000.
- JAYAKUMAR, Magesh. How to program ESP8266 in Lua: Getting started with ESP8266 (NodeMCU dev kit) in Lua. Magesh Jayakumar, 2016.
- REENA, K.; VENKATESH, Veeramuth. Intelligent Decision Support System for Home Automation - ANFIS Based Approach. *International Journal of Engineering & Technology*, 2018.
- KARL, Holger; WILLIG, Andreas. Protocols and architectures for wireless sensor networks. John Wiley & Sons, 2007.
- KIEPERT, Joshua. Creating a raspberry pi-based beowulf cluster. Boise State University, 2013.
- KUMAR, Neeraj; MAKKAR, Aaisha. Machine Learning in Cognitive IoT. CRC Press, 2020.
- KURNIAWAN, Agus. Raspbian OS Programming with the Raspberry Pi: IoT Projects with Wolfram, Mathematica, and Scratch. Apress, Berkeley, CA. 2018.
- KUROSE, James F.; ROSS, Keith W. Redes de computadores e a internet: Uma abordagem top-down. 5ª ed.. Pearson Educación. São Paulo, 2010.
- KYAS, Othmar. How To Smart Home: A Step by Step Guide for Smart Homes & Building Automation. Key Concept Press, 2017.
- LEE, Ju Hyun et al. Context-aware inference in ubiquitous residential environments. *Computers in Industry*, v. 65, n. 1, p. 148-157, 2014.
- MANAVALAN, E.; JAYAKRISHNA, K. A review of Internet of Things (IoT) embedded sustainable supply chain for industry 4.0 requirements. *Computers & Industrial Engineering*, 2019.
- MARWEDEL, Peter. Embedded system design. 3rd ed.. New York: Springer, 2018.
- MITCHELL, Tom M. Machine Learning. McGraw-Hill, 1997.
- MORENO, Laura Vadillo et al. The role of smart homes in intelligent homecare and healthcare environments. In: *Ambient Assisted Living and Enhanced Living Environments*. Butterworth-Heinemann, 2017.
- NORRIS, Donald. Home Automation with Raspberry Pi Projects Using Google Home, Amazon Echo, and Other Intelligent Personal Assistants. McGraw Hill Education, 2019.

- NORRIS, Donald. Raspberry Pi projects for the evil genius. McGraw Hill Education, 2014.
- OLIVEIRA, Sérgio de. Internet das coisas com ESP8266, Arduino e Raspberry PI. Novatec Editora. São Paulo, 2017.
- ORTIZ, Luiz Henrique Oliveira. Sistema de automação residencial com ênfase em segurança e economia energética. 2018.
- PEREIRA, Alexandre de A.; OLIVEIRA, Josemar L.; COSTA, Jefferson S.. Automação Residencial baseada em computamento utilizando Raspberry Pi. In: Anais do III Congresso de Tecnologias e Desenvolvimento na Amazônia. Santarém – PA, 2019.
- PIR Motion Sensor. Adafruit Learning System, 2018. Disponível em: <<https://learn.adafruit.com/downloads/pdf/pir-passive-infrared-proximity-motion-sensor.pdf>>. Acesso em: 07 de dez. de 2019.
- QUINLAN, J. Ross. C4.5: Programs for Machine Learning. San Mateo, California, EE.UU, Morgan Kaufmann Publishers, 1993.
- QUINLAN, J. Ross. Induction of decision trees. Machine learning, 1986.
- RUSSELL, David J. Introduction to embedded systems: using ANSI C and the arduino development environment. Synthesis Lectures on Digital Circuits and Systems, 2010.
- RUSSELL, S.; NORVING, P. Inteligência artificial. Editora Campus, 2004.
- SANCHEZ, Julio; CANTON, Maria P. Embedded systems circuits and programming. CRC Press, 2012.
- SCHNEIDER, André; SOUZA, Fernando. Sistemas Embarcados: Hardware e Firmware na prática. 2ª Edição. Editora Érica, 2010.
- SICARI, Sabrina; RIZZARDI, Alessandra; COEN-PORISINI, Alberto. Smart transport and logistics: A Node-RED implementation. Internet Technology Letters, 2019.
- SOHRABY, Kazem; MINOLI, Daniel; ZNATI, Taieb. Wireless sensor networks: technology, protocols, and applications. John Wiley & Sons, 2007.
- STAPLE, Danny. Learn Robotics Programming. Packt Publishing Ltd. 2018.
- SUTTON, R. S.; BARTO, A. G. Reinforcement learning: an introduction. Cambridge, 1998.
- SWAROOP, K. Narendra et al. A health monitoring system for vital signs using IoT. Internet of Things, 2019.
- SZEWCYZK, S. et al. Annotating smart environment sensor data for activity learning. Technology and Health Care, 2009.
- TANENBAUM, Andrew S.; WETHERALL, David. Redes de Computadores, 5ª edição, Ed. 2011.
- THAKUR, M. NodeMCU ESP8266 Communication Methods and Protocols: Programming with Arduino IDE. Amazon Digital Services LLC, 2018.
- TONIDANDEL, Flavio; SGARBI, Julio André. Aprendendo Regras por Observação em uma Casa Inteligente. In: Anais do XVI Congresso Brasileiro de Automática, Congresso Brasileiro de Automática, Salvador–BA. 2006.

VRBASKI, Mira; PETRIU, Dorina; AMYOT, Daniel. Tool support for combined rule-based and goal-based reasoning in Context-Aware systems. In: 2012 20th IEEE International Requirements Engineering Conference. IEEE, 2012.

VUJOVIĆ, Vladimir; MAKSIMOVIC, Mirjana. Raspberry Pi as a Sensor Web node for home automation. *Computers & Electrical Engineering*, 2015.

WANG, K. C. Models of Embedded Systems. In: *Embedded and Real-Time Operating Systems*. Springer, 2017.

WANG, Vanessa; SALIM, Frank; MOSKOVITS, Peter. The definitive guide to HTML5 WebSocket. New York: Apress, 2013.

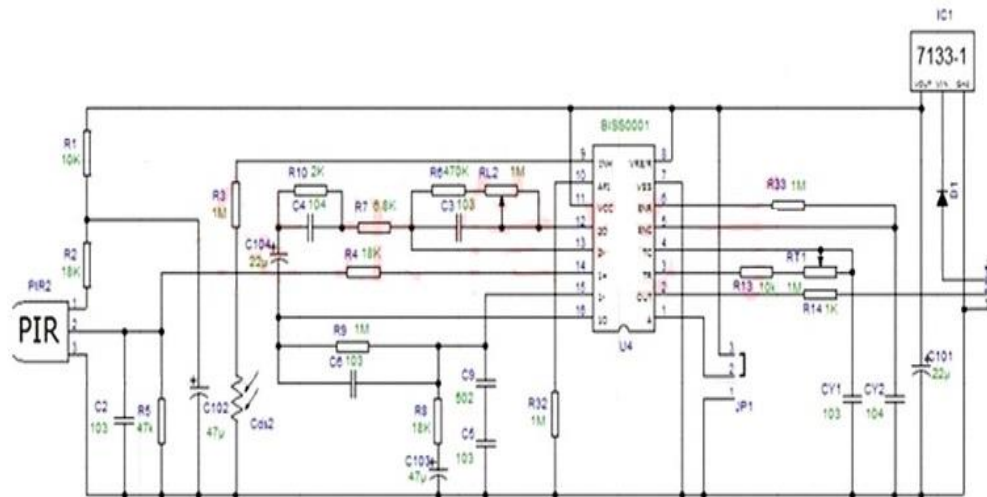
WELBOURNE, Dustin J., et al. How do passive infrared triggered camera traps operate and why does it matter? Breaking down common misconceptions. *Remote Sensing in Ecology and Conservation*, 2016.

WITTEN, Ian H. et al. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

XIE, Lulu; XU, Fangqin. The Research of User's Behavioural Decision on Intelligent Home Control and Sensing System. *International Journal of Wireless Information Networks*, 2018.

## ANEXOS

## ANEXO 1 – CIRCUITO DO HC-SR501



## ANEXO 2 – SKETCH DESENVOLVIDO PARA IOT-SP

```

/*
  Nó IoT-SP

  Objeto:
  {
    "tipo":"sensor",
    "id":"f7af5000-8834-11ea-a3ad-1c1b0df4d205",
    "nome":"PIR",
    "estados":[
      {
        "id":"55072fc0-8835-11ea-a3ad-1c1b0df4d205",
        "nome":"PRESENTE"
      },
      {
        "id":"550de680-8835-11ea-a3ad-1c1b0df4d205",
        "nome":"AUSENTE"
      }
    ]
  }
*/

#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>
#include <WebSocketsClient.h>
#include <Hash.h>
#include <string.h>

#include <ArduinoJson.h>

ESP8266WiFiMulti WiFiMulti;
WebSocketsClient webSocket;

#define USE_SERIAL Serial

// Configuração da rede
#define SSID_REDE "LAN Casa"
#define SENHA_REDE "senha123"

```

```

// Configuração do Websocket
char* IP_DO_SERVIDOR = "192.168.1.50"; // ip do host
int PORTA_DO_SERVIDOR = 30000; // port
#define ROTA_DO_PROTOCOLO_WEBSOCKET "/websocket" // rota

char mensagemRecebida[400] = "";
char previsao[400] = "";

#define PIN_PIR D1 // Porta que recebe dados do sensor

unsigned long tempoAtual = millis();
unsigned long tempoInicial = millis();
char status_habitante[50] = ""; // "PRESENTE" ou "AUSENTE"

/* A função abaixo contem os manipuladores de um WS comum: "connected" é
chamado quando conecta ao server WS,
"disconnected" é chamado quando desconecta e "text" é chamada quando
recebe um "quadro" do servidor WS, entre outros*/
void websocketEvent(WStype_t type, uint8_t * payload, size_t length) {
    switch(type) {
        case WStype_DISCONNECTED:{
            USE_SERIAL.printf("[WSc] Desconectado!\n");
            websocket.disconnect(); // Essa redundância garante que esse
cliente se desconectou do servidor
        }
        break;

        case WStype_CONNECTED: {
            USE_SERIAL.printf("[WSc] Conectado ao url: %s\n", payload);

            // enviar mensagem ao servidor quando conectado
            websocket.sendTXT("{\"id\":\"f7af5000-8834-11ea-a3ad-
1c1b0df4d205\"}");
        }
        break;

        case WStype_TEXT:{
            //USE_SERIAL.printf("[WSc] Recebido -> %s\n", payload);
            //USE_SERIAL.printf("[WSc] tamanho -> %u\n", length);

```

```

        int count = length; // length é uma variável do tipo
"unsigned int" que armazena o número de caracteres da string payload
        //USE_SERIAL.printf("numeor -> %d\n", count);

        for (int i = 0; i < count; i++) {
            mensagemRecebida[i] = payload[i]; // Transfere os
caracteres da variável do tipo uint8_t para a variavel char
        }

        mensagemRecebida[count] = '\0';

        StaticJsonDocument<600> doc; // Alocar o documento JSON

        DeserializationError error = deserializeJson(doc,
mensagemRecebida);

        // Testa se a desserialização do json foi bem sucedida.
        if(error) {
            //USE_SERIAL.print("Desserialização do Json falhou!\n");
            USE_SERIAL.printf("mensagem recebida: %s\n",
mensagemRecebida);
            //USE_SERIAL.println(error.f_str());
            return;
        }

        const char* p = doc["previsao"];

        if (p != nullptr){
            USE_SERIAL.printf("Previsão: %s\n", p);
            strcpy( previsao, p );
        }
        else{
            USE_SERIAL.print("VAZIO!\n");
        }

        strcpy(mensagemRecebida, ""); // Limpa a string que
armazena a mensagem recebida
        delay(10);
    }
    break;

```

```

        case WStype_BIN: {
            USE_SERIAL.printf("[WSc] obter comprimento binário: %u\n",
length);

            hexdump(payload, length);

            // enviar dados para o servidor
            // websocket.sendBIN(payload, length);
        }
        break;

        case WStype_PING: {
            // informa o recebimento de um ping, o pong de resposta é
enviado automaticamente
            //USE_SERIAL.printf("[WSc] recebeu ping do servidor WS\n");
        }
        break;

        case WStype_PONG: {
            // responder a um ping que enviamos
            USE_SERIAL.printf("[WSc] recebeu pong do servidor WS\n");
        }
        break;
    }
}

void setup() {
    pinMode(PIN_PIR, INPUT); // Configura o pino "PIN_PIR" como input
(entrada) do Sensor

    // USE_SERIAL.begin(921600);
    USE_SERIAL.begin(115200);

    //Serial.setDebugOutput(true);
    USE_SERIAL.setDebugOutput(true);

    USE_SERIAL.println();
    USE_SERIAL.println();
    USE_SERIAL.println();

```

```

for(uint8_t t = 4; t > 0; t--) {
    USE_SERIAL.printf("[SETUP] BOOT WAIT %d...\n", t);
    USE_SERIAL.flush();
    delay(2000);
}

// disable AP
if(WiFi.getMode() & WIFI_AP) {
    WiFi.softAPdisconnect(true);
}

WiFi.hostname("Nó IoT-SP");

WiFiMulti.addAP(SSID_REDE, SENHA_REDE); // Identifica o ponto de
acesso ao qual vai se conectar

//WiFi.disconnect();
while(WiFiMulti.run() != WL_CONNECTED) {
    delay(100);
}

// server address, port and URL
WebSocket.begin(IP_DO_SERVIDOR, PORTA_DO_SERVIDOR,
ROTA_DO_PROTOCOLO_WEBSOCKET); // Identifica o WebSocket ao qual vai se
conectar

// event handler
WebSocket.onEvent(WebSocketEvent);

// use HTTP Basic Authorization this is optional remove if not needed
//WebSocket.setAuthorization("user", "Password");

// try ever 5000 again if connection has failed
WebSocket.setReconnectInterval(5000);

// start heartbeat (optional)
// ping server every 15000 ms
// expect pong from server within 3000 ms
// consider connection disconnected if pong is not received 2 times
// WebSocket.enableHeartbeat(15000, 3000, 10);

```

```
}

void loop() {

    websocket.loop(); // Loop do WebSocket

    int sinal = digitalRead(PIN_PIR); //faz a leitura do sensor de
    presença (retorna HIGH ou LOW)

    if(sinal == HIGH){//HIGH : movimento detectado
        tempoInicial = millis(); // Reseta contador de tempo
        if(strcmp(status_habitante, "PRESENTE") != 0){
            websocket.sendTXT("{\"id\":\"55072fc0-8835-11ea-a3ad-
1c1b0df4d205\"}"); // Envia a mensagem ao servidor
            strcpy(status_habitante, "PRESENTE");
            USE_SERIAL.printf("Habitante PRESENTE\n");
        }
    }
    else if(sinal == LOW){
        tempoAtual = millis();
        if(tempoAtual > (tempoInicial + 15000) &&
strcmp(status_habitante, "AUSENTE") != 0 ){ // Verdadeiro se passou 15
segundos sem detectar ninguem no ambiente
            websocket.sendTXT("{\"id\":\"550de680-8835-11ea-a3ad-
1c1b0df4d205\"}"); // Envia a mensagem ao servidor
            strcpy(status_habitante, "AUSENTE");
            USE_SERIAL.printf("Habitante AUSENTE\n");
        }
    }
}
}
```

## ANEXO 3 – SKETCH DESENVOLVIDO PARA IOT-L

```

/*
  Nó IoT-L

  Objeto:
  {
    "id":"cae17980-8835-11ea-a3ad-1c1b0df4d205",
    "nome":"Lampada",
    "estados":[
      {
        "id":"4b0239b0-8836-11ea-a3ad-1c1b0df4d205",
        "nome":"ACESA"
      },
      {
        "id":"4b18cef0-8836-11ea-a3ad-1c1b0df4d205",
        "nome":"APAGADA"
      }
    ]
  }
*/

#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>
#include <WebSocketsClient.h>
#include <Hash.h>
#include <string.h>

#include <ArduinoJson.h>

ESP8266WiFiMulti WiFiMulti;
WebSocketsClient webSocket;

#define USE_SERIAL Serial

// Configuração da rede
#define SSID_REDE "LAN Casa"
#define SENHA_REDE "senha123"

// Configuração do Websocket
char* IP_DO_SERVIDOR = "192.168.1.50"; // ip do host
int PORTA_DO_SERVIDOR = 30000; // port
#define ROTA_DO_PROTOCOLO_WEBSOCKET "/websocket" // rota

char mensagemRecebida[400] = "";
char previsao[400] = "";

#define PINO_RELE D1 // Pino que altera estado do atuador

char status_rele[10] = ""; // "HIGH" ou "LOW"
char status_lampada[50] = ""; // "ACESA" ou "APAGADA"
float tensao;

/* A função abaixo contem os manipuladores de um WS comum: "connected" é
chamado quando conecta ao server WS,
"disconnected" é chamado quando desconecta e "text" é chamada quando
recebe um "quadro" do servidor WS, entre outros*/
void websocketEvent(WStype_t type, uint8_t * payload, size_t length) {

```

```

switch(type) {
    case WStype_DISCONNECTED:{
        USE_SERIAL.printf("[WSc] Desconectado!\n");
        websocket.disconnect(); // Essa redundância garante que esse
cliente se desconectou do servidor
    }
    break;

    case WStype_CONNECTED: {
        USE_SERIAL.printf("[WSc] Conectado ao url: %s\n", payload);

        // enviar mensagem ao servidor quando conectado
        websocket.sendTXT("{\"id\":\"cael7980-8835-11ea-a3ad-
1c1b0df4d205\"}");
    }
    break;

    case WStype_TEXT:{
        //USE_SERIAL.printf("[WSc] Recebido -> %s\n", payload);
        //USE_SERIAL.printf("[WSc] tamanho -> %u\n", length);

        int count = length; // length é uma variável do tipo
"unsigned int" que armazena o número de caracteres da string payload
        //USE_SERIAL.printf("numeor -> %d\n", count);

        for (int i = 0; i < count; i++) {
            mensagemRecebida[i] = payload[i]; // Transfere os
caracteres da variável do tipo uint8_t para a variavel char
        }

        mensagemRecebida[count] = '\0';

        StaticJsonDocument<600> doc; // Alocar o documento JSON

        DeserializationError error = deserializeJson(doc,
mensagemRecebida);

        // Testa se a desserialização do json foi bem sucedida.
        if(error) {
            //USE_SERIAL.print("Desserialização do Json falhou!\n");
            USE_SERIAL.printf("mensagem recebida: %s\n",
mensagemRecebida);
            //USE_SERIAL.println(error.f_str());
            return;
        }

        const char* p = doc["previsao"];

        if (p != nullptr){
            USE_SERIAL.printf("Previsão: %s\n", p);
            strcpy( previsao, p );
        }
        else{
            USE_SERIAL.print("VAZIO!\n");
        }

        strcpy(mensagemRecebida, ""); // Limpa a string que
armazena a mensagem recebida
        delay(10);

```

```

    }
    break;

    case WStype_BIN:{
        USE_SERIAL.printf("[WSc] obter comprimento binário: %u\n",
length);
        hexdump(payload, length);

        // enviar dados para o servidor
        // websocket.sendBIN(payload, length);
    }
    break;

    case WStype_PING:{
        // informa o recebimento de um ping, o pong de resposta é
enviado automaticamente
        //USE_SERIAL.printf("[WSc] recebeu ping do servidor WS\n");
    }
    break;

    case WStype_PONG:{
        // responder a um ping que enviamos
        USE_SERIAL.printf("[WSc] recebeu pong do servidor WS\n");
    }
    break;
}
}

void setup() {
    pinMode(PINO_RELE, OUTPUT); // Configura o pino "PINO_RELE" como
output (saída)

    // Abaixo é definido o status inicial do relé através do status da
lâmpada, facilitando a
    // instalação do relé junto ao interruptor tree-way

    tensao = (analogRead(A0)*3.3/1023);

    if(tensao >= 1){ //Se verdadeiro, a lampada esta ligada
        strcpy(status_lampada, "ACESA");
        USE_SERIAL.printf("A Lampada está inicialmente ACESA\n");
    }
    else {
        strcpy(status_lampada, "APAGADA");
        USE_SERIAL.printf("A Lampada está inicialmente APAGADA\n");
    }

    digitalWrite(PINO_RELE, LOW); // Coloca o estado desse nó como BAIXO
ao ser ligado
    strcpy(status_rele, "LOW");
    delay(100);

    tensao = (analogRead(A0)*3.3/1023);

    if(tensao >= 1){ // Se verdadeiro, a lampada esta ligada
        if( strcmp(status_lampada, "ACESA") != 0 ){
            digitalWrite(PINO_RELE, HIGH);
            strcpy(status_rele, "HIGH");
        }
    }
    else {

```

```

        if( strcmp(status_lampada, "APAGADA") != 0 ){
            digitalWrite(PINO_RELE, HIGH);
            strcpy(status_rele, "HIGH");
        }
    }

    // USE_SERIAL.begin(921600);
    USE_SERIAL.begin(115200);

    //Serial.setDebugOutput(true);
    USE_SERIAL.setDebugOutput(true);

    USE_SERIAL.println();
    USE_SERIAL.println();
    USE_SERIAL.println();

    for(uint8_t t = 4; t > 0; t--) {
        USE_SERIAL.printf("[SETUP] BOOT WAIT %d...\n", t);
        USE_SERIAL.flush();
        delay(2000);
    }

    // disable AP
    if(WiFi.getMode() & WIFI_AP) {
        WiFi.softAPdisconnect(true);
    }

    WiFi.hostname("Nó IoT-L");

    WiFiMulti.addAP(SSID_REDE, SENHA_REDE); // Identifica o ponto de
acesso ao qual vai se conectar

    //WiFi.disconnect();
    while(WiFiMulti.run() != WL_CONNECTED) {
        delay(100);
    }

    // server address, port and URL
    websocket.begin(IP_DO_SERVIDOR, PORTA_DO_SERVIDOR,
ROTA_DO_PROTOCOLO_WEBSOCKET); // Identifica o Websocket ao qual vai se
conectar

    // event handler
    websocket.onEvent(webSocketEvent);

    // use HTTP Basic Authorization this is optional remove if not needed
    //websocket.setAuthorization("user", "Password");

    // try ever 5000 again if connection has failed
    websocket.setReconnectInterval(5000);

    // start heartbeat (optional)
    // ping server every 15000 ms
    // expect pong from server within 3000 ms
    // consider connection disconnected if pong is not received 2 times
    // websocket.enableHeartbeat(15000, 3000, 10);
}

```

```

void loop() {

    websocket.loop(); // Loop do WebSocket

    tensao = (analogRead(A0)*3.3/1023);

    if(tensao >= 1){ //Se verdadeiro, a lampada esta ligada
        if( strcmp(status_lampada, "ACESA") != 0){
            websocket.sendTXT("{\"id\":\"4b0239b0-8836-11ea-a3ad-1c1b0df4d205\"}");
            USE_SERIAL.printf("Habitante mudou Lampada para
ACESA\n");
            strcpy(status_lampada, "ACESA");
        }
    }
    else {
        if( strcmp(status_lampada, "APAGADA") != 0){
            websocket.sendTXT("{\"id\":\"4b18cef0-8836-11ea-a3ad-1c1b0df4d205\"}");
            USE_SERIAL.printf("Habitante mudou Lampada para
APAGADA\n");
            strcpy(status_lampada, "APAGADA");
        }
    }

    if( strcmp(previsao, "4b0239b0-8836-11ea-a3ad-1c1b0df4d205") == 0 ){
// Se verdadeiro, deve mudara para o estado de "ACESA"
        if( strcmp(status_lampada, "ACESA") != 0 ){
            if( strcmp(status_rele, "LOW") == 0 ){
                digitalWrite(PINO_RELE, HIGH); // Muda o estado do
relé para ALTO
                strcpy(status_rele, "HIGH");
            }else{
                digitalWrite(PINO_RELE, LOW); // Muda o estado do
relé para BAIXO
                strcpy(status_rele, "LOW");
            }
            strcpy(status_lampada, "ACESA");
            USE_SERIAL.printf("Previsão mudou Lampada para ACESA\n");
        }
    }
    else if( strcmp(previsao, "4b18cef0-8836-11ea-a3ad-1c1b0df4d205") ==
0 ){ // Se verdadeiro, deve mudara para o estado de "APAGADA"
        if( strcmp(status_lampada, "APAGADA") != 0 ){
            if( strcmp(status_rele, "LOW") == 0 ){
                digitalWrite(PINO_RELE, HIGH); // Muda o estado do
relé para ALTO
                strcpy(status_rele, "HIGH");
            }else{
                digitalWrite(PINO_RELE, LOW); // Muda o estado do
relé para BAIXO
                strcpy(status_rele, "LOW");
            }
            strcpy(status_lampada, "APAGADA");
            USE_SERIAL.printf("Previsão mudou Lampada para
APAGADA\n");
        }
    }
}
}

```

```
        strcpy(previsao, ""); // Limpa a string que armazena a mensagem
recebida
        delay(100);
    }
```

## ANEXO 4 – DADOS DOS EXPERIMENTOS

Tabela 2 – Publicações do experimento

(continua)

<b>Dispositivo</b>	<b>Valor Qualitativo</b>	<b>Data</b>	<b>Hora</b>
IoT-SP	PRESENTE	01/06/2020	08:05:00
IoT-L	ACESA	01/06/2020	08:05:05
IoT-L	APAGADA	01/06/2020	08:06:00
IoT-SP	AUSENTE	01/06/2020	08:06:20
IoT-SP	PRESENTE	01/06/2020	09:15:00
IoT-L	ACESA	01/06/2020	09:15:05
IoT-L	APAGADA	01/06/2020	09:16:00
IoT-SP	AUSENTE	01/06/2020	09:16:20
IoT-SP	PRESENTE	01/06/2020	10:25:00
IoT-L	ACESA	01/06/2020	10:25:05
IoT-L	APAGADA	01/06/2020	10:26:00
IoT-SP	AUSENTE	01/06/2020	10:26:20
IoT-SP	PRESENTE	01/06/2020	11:05:00
IoT-L	ACESA	01/06/2020	11:05:05
IoT-L	APAGADA	01/06/2020	11:06:00
IoT-SP	AUSENTE	01/06/2020	11:06:20
IoT-SP	PRESENTE	01/06/2020	13:05:00
IoT-L	ACESA	01/06/2020	13:05:05
IoT-L	APAGADA	01/06/2020	13:06:00
IoT-SP	AUSENTE	01/06/2020	13:06:20
IoT-SP	PRESENTE	01/06/2020	14:05:00
IoT-L	ACESA	01/06/2020	14:05:05
IoT-L	APAGADA	01/06/2020	14:06:00
IoT-SP	AUSENTE	01/06/2020	14:06:20
IoT-SP	PRESENTE	01/06/2020	15:05:00

Tabela 2 – Publicações do experimento

(conclusão)			
<b>Dispositivo</b>	<b>Valor Qualitativo</b>	<b>Data</b>	<b>Hora</b>
IoT-L	ACESA	01/06/2020	15:05:05
IoT-L	APAGADA	01/06/2020	15:06:00
IoT-SP	AUSENTE	01/06/2020	15:06:20
IoT-SP	PRESENTE	01/06/2020	16:05:00
IoT-L	ACESA	01/06/2020	16:05:05
IoT-L	APAGADA	01/06/2020	16:06:00
IoT-SP	AUSENTE	01/06/2020	16:06:20
IoT-SP	PRESENTE	01/06/2020	17:05:00
IoT-L	ACESA	01/06/2020	17:05:05
IoT-L	APAGADA	01/06/2020	17:06:00
IoT-SP	AUSENTE	01/06/2020	17:06:20
IoT-SP	PRESENTE	01/06/2020	18:05:00
IoT-L	ACESA	01/06/2020	18:05:05
IoT-L	APAGADA	01/06/2020	18:06:00
IoT-SP	AUSENTE	01/06/2020	18:06:20

Fonte: Autoria própria.

## ANEXO 5 – DADOS DO EXPERIMENTO FINAL

Tabela 3 – Publicações obtidas com o habitante real

(continua)

<b>Dispositivo</b>	<b>Valor Qualitativo</b>	<b>Data</b>	<b>Hora</b>
IoT-SP	PRESENTE	09/11/2020	10:34:19
IoT-SP	AUSENTE	09/11/2020	10:36:02
IoT-SP	PRESENTE	09/11/2020	12:02:15
IoT-SP	AUSENTE	09/11/2020	12:03:17
IoT-SP	PRESENTE	09/11/2020	19:18:04
IoT-L	ACESA	09/11/2020	19:18:10
IoT-L	APAGADA	09/11/2020	19:22:43
IoT-SP	AUSENTE	09/11/2020	19:22:59
IoT-SP	PRESENTE	09/11/2020	19:56:01
IoT-SP	AUSENTE	09/11/2020	19:56:48
IoT-SP	PRESENTE	09/11/2020	22:04:38
IoT-L	APAGADA	09/11/2020	22:06:29
IoT-SP	PRESENTE	10/11/2020	01:42:35
IoT-SP	AUSENTE	10/11/2020	01:42:52
IoT-SP	PRESENTE	10/11/2020	06:00:20
IoT-L	ACESA	10/11/2020	06:00:36
IoT-SP	AUSENTE	10/11/2020	06:00:54
IoT-SP	PRESENTE	10/11/2020	06:04:53
IoT-SP	AUSENTE	10/11/2020	06:05:41
IoT-SP	PRESENTE	10/11/2020	06:08:13
IoT-L	APAGADA	10/11/2020	06:12:43
IoT-SP	AUSENTE	10/11/2020	06:13:21
IoT-SP	PRESENTE	10/11/2020	14:41:09
IoT-SP	AUSENTE	10/11/2020	14:42:45
IoT-SP	PRESENTE	10/11/2020	16:16:17

Tabela 3 – Publicações obtidas com o habitante real

(continua)

<b>Dispositivo</b>	<b>Valor Qualitativo</b>	<b>Data</b>	<b>Hora</b>
IoT-SP	AUSENTE	10/11/2020	16:16:47
IoT-SP	PRESENTE	10/11/2020	19:27:22
IoT-L	ACESA	10/11/2020	19:27:29
IoT-SP	AUSENTE	10/11/2020	19:30:01
IoT-SP	PRESENTE	10/11/2020	20:05:22
IoT-L	APAGADA	10/11/2020	20:05:25
IoT-SP	AUSENTE	10/11/2020	20:05:33
IoT-SP	PRESENTE	10/11/2020	20:42:00
IoT-L	ACESA	10/11/2020	20:42:04
IoT-L	APAGADA	10/11/2020	20:44:11
IoT-SP	AUSENTE	10/11/2020	20:44:29
IoT-SP	PRESENTE	10/11/2020	22:01:59
IoT-SP	AUSENTE	10/11/2020	22:02:41
IoT-SP	PRESENTE	11/11/2020	03:03:55
IoT-SP	AUSENTE	11/11/2020	03:04:12
IoT-SP	PRESENTE	11/11/2020	06:00:20
IoT-L	ACESA	11/11/2020	06:01:10
IoT-SP	AUSENTE	11/11/2020	06:01:59
IoT-SP	PRESENTE	11/11/2020	06:02:46
IoT-SP	AUSENTE	11/11/2020	06:03:42
IoT-SP	PRESENTE	11/11/2020	06:04:00
IoT-SP	AUSENTE	11/11/2020	06:04:58
IoT-SP	PRESENTE	11/11/2020	06:07:31
IoT-L	APAGADA	11/11/2020	06:10:58
IoT-SP	AUSENTE	11/11/2020	06:13:13
IoT-SP	PRESENTE	11/11/2020	09:56:41

Fonte: Autoria própria.

Tabela 3 – Publicações obtidas com o habitante real

(continua)

<b>Dispositivo</b>	<b>Valor Qualitativo</b>	<b>Data</b>	<b>Hora</b>
IoT-SP	AUSENTE	11/11/2020	10:03:07
IoT-SP	PRESENTE	11/11/2020	12:22:28
IoT-SP	AUSENTE	11/11/2020	12:23:00
IoT-SP	PRESENTE	11/11/2020	19:43:15
IoT-L	ACESA	11/11/2020	19:43:16
IoT-SP	AUSENTE	11/11/2020	19:49:32
IoT-SP	PRESENTE	11/11/2020	19:58:28
IoT-SP	AUSENTE	11/11/2020	20:00:02
IoT-SP	PRESENTE	11/11/2020	21:05:22
IoT-L	APAGADA	11/11/2020	21:05:25
IoT-SP	AUSENTE	11/11/2020	21:05:33
IoT-SP	PRESENTE	11/11/2020	21:42:00
IoT-L	ACESA	11/11/2020	21:42:04
IoT-L	APAGADA	11/11/2020	21:44:11
IoT-SP	AUSENTE	11/11/2020	21:44:29
IoT-SP	PRESENTE	11/11/2020	22:34:41
IoT-L	ACESA	11/11/2020	22:34:43
IoT-L	APAGADA	11/11/2020	22:36:19
IoT-SP	AUSENTE	11/11/2020	22:37:48
IoT-SP	PRESENTE	11/11/2020	23:33:12
IoT-L	ACESA	11/11/2020	23:33:18
IoT-SP	AUSENTE	11/11/2020	23:33:34
IoT-SP	PRESENTE	11/11/2020	23:45:22
IoT-L	APAGADA	11/11/2020	23:45:24
IoT-SP	AUSENTE	11/11/2020	23:46:03
IoT-SP	PRESENTE	12/11/2020	05:59:48

Fonte: Autoria própria.

Tabela 3 – Publicações obtidas com o habitante real

(continua)

<b>Dispositivo</b>	<b>Valor Qualitativo</b>	<b>Data</b>	<b>Hora</b>
IoT-L	ACESA	12/11/2020	06:00:25
IoT-SP	AUSENTE	12/11/2020	06:00:47
IoT-SP	PRESENTE	12/11/2020	06:04:51
IoT-SP	AUSENTE	12/11/2020	06:06:27
IoT-SP	PRESENTE	12/11/2020	06:09:08
IoT-L	APAGADA	12/11/2020	06:09:50
IoT-SP	AUSENTE	12/11/2020	06:10:08
IoT-SP	PRESENTE	12/11/2020	10:22:18
IoT-SP	AUSENTE	12/11/2020	10:23:52
IoT-SP	PRESENTE	12/11/2020	16:16:49
IoT-SP	AUSENTE	12/11/2020	16:18:39
IoT-SP	PRESENTE	12/11/2020	19:14:37
IoT-L	ACESA	12/11/2020	19:14:41
IoT-L	APAGADA	12/11/2020	19:21:14
IoT-SP	AUSENTE	12/11/2020	19:21:32
IoT-SP	PRESENTE	12/11/2020	21:05:22
IoT-L	ACESA	12/11/2020	21:05:24
IoT-L	APAGADA	12/11/2020	21:05:25
IoT-SP	AUSENTE	12/11/2020	21:05:41
IoT-SP	PRESENTE	12/11/2020	21:40:28
IoT-SP	AUSENTE	12/11/2020	21:44:09
IoT-SP	PRESENTE	12/11/2020	22:03:08
IoT-L	ACESA	12/11/2020	22:03:10
IoT-SP	AUSENTE	12/11/2020	22:07:42
IoT-SP	PRESENTE	12/11/2020	22:17:55
IoT-L	APAGADA	12/11/2020	22:19:38

Fonte: Autoria própria.

Tabela 3 – Publicações obtidas com o habitante real

(continua)

<b>Dispositivo</b>	<b>Valor Qualitativo</b>	<b>Data</b>	<b>Hora</b>
IoT-SP	AUSENTE	12/11/2020	22:22:36
IoT-SP	PRESENTE	13/11/2020	06:00:03
IoT-L	ACESA	13/11/2020	06:01:01
IoT-SP	AUSENTE	13/11/2020	06:01:44
IoT-SP	PRESENTE	13/11/2020	06:05:18
IoT-SP	AUSENTE	13/11/2020	06:14:22
IoT-SP	PRESENTE	13/11/2020	06:17:13
IoT-L	APAGADA	13/11/2020	06:17:15
IoT-SP	AUSENTE	13/11/2020	06:17:33
IoT-SP	PRESENTE	13/11/2020	08:28:11
IoT-SP	AUSENTE	13/11/2020	08:31:46
IoT-SP	PRESENTE	13/11/2020	09:52:33
IoT-SP	AUSENTE	13/11/2020	09:54:17
IoT-SP	PRESENTE	13/11/2020	12:14:03
IoT-SP	AUSENTE	13/11/2020	12:14:58
IoT-SP	PRESENTE	13/11/2020	14:27:44
IoT-SP	AUSENTE	13/11/2020	14:32:00
IoT-SP	PRESENTE	13/11/2020	16:59:11
IoT-SP	AUSENTE	13/11/2020	17:03:07
IoT-SP	PRESENTE	13/11/2020	18:40:39
IoT-L	ACESA	13/11/2020	18:40:43
IoT-SP	AUSENTE	13/11/2020	18:43:17
IoT-SP	PRESENTE	13/11/2020	20:30:09
IoT-SP	AUSENTE	13/11/2020	20:32:21
IoT-SP	PRESENTE	13/11/2020	21:11:12
IoT-SP	AUSENTE	13/11/2020	21:13:02

Fonte: Autoria própria.

Tabela 3 – Publicações obtidas com o habitante real

(continua)

<b>Dispositivo</b>	<b>Valor Qualitativo</b>	<b>Data</b>	<b>Hora</b>
IoT-SP	PRESENTE	13/11/2020	21:53:16
IoT-L	APAGADA	13/11/2020	21:55:14
IoT-SP	AUSENTE	13/11/2020	21:57:40
IoT-SP	PRESENTE	14/11/2020	00:22:11
IoT-L	ACESA	14/11/2020	00:22:20
IoT-SP	AUSENTE	14/11/2020	00:22:58
IoT-SP	PRESENTE	14/11/2020	00:29:08
IoT-L	APAGADA	14/11/2020	00:29:18
IoT-SP	AUSENTE	14/11/2020	00:30:33
IoT-SP	PRESENTE	14/11/2020	07:21:31
IoT-SP	AUSENTE	14/11/2020	07:23:38
IoT-SP	PRESENTE	14/11/2020	07:32:50
IoT-L	APAGADA	14/11/2020	07:33:49
IoT-SP	AUSENTE	14/11/2020	07:34:09
IoT-SP	PRESENTE	14/11/2020	07:55:53
IoT-L	APAGADA	14/11/2020	07:55:59
IoT-SP	AUSENTE	14/11/2020	07:57:58
IoT-SP	PRESENTE	14/11/2020	11:12:25
IoT-L	APAGADA	14/11/2020	11:16:44
IoT-SP	AUSENTE	14/11/2020	11:16:17
IoT-SP	PRESENTE	14/11/2020	13:03:42
IoT-L	APAGADA	14/11/2020	13:04:39
IoT-SP	AUSENTE	14/11/2020	13:05:21
IoT-SP	PRESENTE	14/11/2020	16:46:13
IoT-L	APAGADA	14/11/2020	16:48:06
IoT-SP	AUSENTE	14/11/2020	16:48:45

Fonte: Autoria própria.

Tabela 3 – Publicações obtidas com o habitante real

(continua)

<b>Dispositivo</b>	<b>Valor Qualitativo</b>	<b>Data</b>	<b>Hora</b>
IoT-SP	PRESENTE	14/11/2020	19:22:14
IoT-SP	AUSENTE	14/11/2020	19:23:27
IoT-SP	PRESENTE	14/11/2020	21:12:59
IoT-SP	AUSENTE	14/11/2020	21:16:41
IoT-SP	PRESENTE	14/11/2020	23:04:39
IoT-L	APAGADA	14/11/2020	23:08:49
IoT-SP	AUSENTE	14/11/2020	23:09:52
IoT-SP	PRESENTE	15/11/2020	07:09:28
IoT-SP	AUSENTE	15/11/2020	07:10:08
IoT-SP	PRESENTE	15/11/2020	07:43:28
IoT-L	APAGADA	15/11/2020	07:43:31
IoT-SP	AUSENTE	15/11/2020	07:44:05
IoT-SP	PRESENTE	15/11/2020	12:17:13
IoT-L	APAGADA	15/11/2020	12:19:40
IoT-SP	AUSENTE	15/11/2020	12:24:24
IoT-SP	PRESENTE	15/11/2020	18:01:12
IoT-SP	AUSENTE	15/11/2020	18:12:27
IoT-SP	PRESENTE	15/11/2020	18:44:29
IoT-SP	AUSENTE	15/11/2020	18:48:13
IoT-SP	PRESENTE	15/11/2020	20:03:36
IoT-SP	AUSENTE	15/11/2020	20:07:47
IoT-SP	PRESENTE	15/11/2020	22:24:28
IoT-L	APAGADA	15/11/2020	22:27:56
IoT-SP	AUSENTE	15/11/2020	22:28:34
IoT-SP	PRESENTE	16/11/2020	03:46:24
IoT-SP	AUSENTE	16/11/2020	03:46:48

Fonte: Autoria própria.

Tabela 3 – Publicações obtidas com o habitante real

(conclusão)			
<b>Dispositivo</b>	<b>Valor Qualitativo</b>	<b>Data</b>	<b>Hora</b>
IoT-SP	PRESENTE	16/11/2020	03:54:11
IoT-L	APAGADA	16/11/2020	03:54:14
IoT-SP	AUSENTE	16/11/2020	03:54:33
IoT-SP	PRESENTE	16/11/2020	05:51:34
IoT-SP	AUSENTE	16/11/2020	05:51:42
IoT-SP	PRESENTE	16/11/2020	06:14:20
IoT-L	APAGADA	16/11/2020	06:14:23
IoT-SP	AUSENTE	16/11/2020	06:20:01
IoT-SP	PRESENTE	16/11/2020	09:59:38
IoT-L	APAGADA	16/11/2020	09:59:41
IoT-SP	AUSENTE	16/11/2020	10:04:32
IoT-SP	PRESENTE	16/11/2020	16:29:05
IoT-L	APAGADA	16/11/2020	16:31:23

Fonte: Autoria própria.