Pós-Graduação em Ciência da Computação

FABIANO AMORIM VAZ

SISTEMA OPERACIONAL PARA RESIDÊNCIAS INTELIGENTES BASEADO EM ANÁLISE COMPORTAMENTAL COM SUPORTE À INTERAÇÃO NATURAL

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE
2017

Fabiano Amorim Vaz

Sistema Operacional para Residências Inteligentes Baseado em Análise Comportamental com Suporte à Interação Natural

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Doutor em Ciência da Computação.

Orientadora: *Dra. Veronica Teichrieb*

RECIFE

2017

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

V393s Vaz, Fabiano Amorim
Sistema operacional para residências inteligentes baseado em análise comportamental com suporte à interação natural / Fabiano Amorim Vaz. – 2017.
144 f.: il., fig., tab.

Orientadora: Verônica Teichrieb.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2017.
Inclui referências.

1. Ciência da computação. 2. Internet das coisas. I. Teichrieb, Verônica (orientadora). II. Título.

004 CDD (23. ed.) UFPE- MEI 2018-040

Fabiano Amorim Vaz

Sistema Operacional para Residências Inteligentes Baseado em Análise Comportamental com Suporte à Interação Natural

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação.

Aprovado em: 06/09/2017

BANCA EXAMINADORA

Prof. Carlos André Guimarães Ferraz
Departamento de Informação e Sistemas - CIn / UFPE

Prof. Giordano Ribeiro Eulalio Cabral
Departamento de Sistemas da Computação - CIn / UFPE

Prof. João Marcelo Xavier Natário Teixeira
Departamento de Eletrônica e Sistemas - CTG / UFPE

Prof. Kiev Santos da Gama
Departamento de Ciência da Computação - CIn / UFPE

Prof. Thales Miranda de Almeida Vieira
Instituto de Matemática / UFAL

Prof. Walter Franklin Marques Correia
Departamento de Design - CAC / UFPE

Resumo

A possibilidade de embarcar um *software* que demanda um poder de processamento maior em circuitos com tamanho bem reduzido, fomenta o surgimento de dispositivos mais inteligentes em uma diversidade de áreas. A popularização desta tecnologia mostra indícios de que é factível possuirmos residências verdadeiramente inteligentes em alguns anos. Contudo, existem alguns aspectos desafiadores, onde é preciso lidar com a dificuldade de interoperabilidade entre os dispositivos heterogêneos, com a complexidade de abstração de *hardware*, com a dificuldade de gerir uma massa de dados que pode ser produzida no ambiente, bem como interagir utilizando controles complexos. Entretanto, ainda que os aspectos supracitados fossem solucionados, seria necessário desenvolver ferramentas capazes de gerenciar os eventos disparados tanto pelos usuários quanto pelo próprio ambiente, provendo suporte à interação natural, sob uma perspectiva de aprendizado contextual. Deste modo, o objetivo principal desta tese é propor e validar uma arquitetura de Sistema Operacional (SO) para Internet das Coisas (*Internet of Things* (IoT)) com enfoque no ambiente residencial, onde a interação natural é um requisito fundamental. A proposta tem como finalidade abstrair a camada de *hardware*, provendo subsídios para auxiliar o desenvolvimento de aplicações no âmbito residencial. Como forma de validar a arquitetura proposta, foi desenvolvido um protótipo de SO, no intuito de averiguar a harmonia entre os componentes e principalmente o atendimento aos requisitos levantados através da literatura. Para tanto, foram elaborados estudos de caso, onde foi possível observar de qual modo os requisitos foram tratados e, posteriormente, foi mensurado quantitativamente cada requisito, a fim de estabelecer métricas que podem ser utilizadas como referências em futuras implantações. Portanto, a principal contribuição desta pesquisa é a arquitetura de SO para IoT; além desta, destaca-se ainda o protótipo funcional de um SO para residências, o qual denominamos de VoxarHomeOS. Através da experimentação do protótipo, foi possível observar o comportamento dos componentes da arquitetura resultando em evidências que indicam que a arquitetura proposta é capaz de suprir os requisitos de interoperabilidade, escalabilidade, extensibilidade, descoberta em rede, sensibilidade ao contexto, análise de dados e, principalmente, interação natural. Em comparação com os trabalhos relacionados, se considerarmos este conjunto de requistos, podemos inferir que o presente trabalho apresenta-se como uma evolução do estado da arte, visto que além de atender aos requisitos, fornece ao habitante mecanismos para definir gestos, comandos de voz para interagir com o ambiente, bem como consegue aprender a partir dos dados do ambiente.

Palavras-chave: Internet das Coisas. Residências Inteligentes. Interação Natural.

Abstract

The ability of embedding software that demands a greater processing power in smaller circuits encourages the emergence of smarter devices in a variety of areas. The spread of this technology shows signs that it is feasible to actually have smart homes in a few years. However, there are still challenges, and it is necessary to develop solutions that deal with difficulties related to interoperability between heterogeneous devices; With the complexity of hardware abstraction; With the difficulty of managing of big data that can be produced in the environment, besides being able to interact without using complex controls. However, if these challenges were solved, it would still be necessary to develop tools capable of managing the events triggered by the users or by the environment itself, considering the support to the natural interaction, from a perspective of contextual learning. Therefore, the main goal of this work is to propose and validate an Operating System architecture for Internet of Things with a focus on the Smart Homes, where natural interaction is a requirement. The proposed architecture aims to abstract the hardware layer, providing tools to guide the development of applications in the residential scope. An Operating System prototype was developed to validate the proposed architecture, aiming to investigate the conformity between the components of the architecture and to contemplate the requirements identified in the state of the art. For this purpose, case studies were developed, where it was possible to observe in which way the requirements were considered and, subsequently, each requirement was quantitatively measured, with the goal of establishing metrics that can be used as reference models in future work. Therefore, the main contribution of this research was the architecture of Operating System for Internet of Things, as well as high-fidelity prototypes of an Operating System for Smart Homes, called VoxarHomeOS. Through the case studies, it was possible to analyze the behavior of the architecture components, highlighting evidence indicating that the proposed architecture suited the requirements of interoperability, scalability, extensibility, network discovery, context sensitivity, data analysis and, natural interaction. Considering these requirements and comparing them with the related works, we can indicate that the present work presents itself as an evolution of the state of the art, Whereas in addition to complying with the requirements, it provides users with mechanisms to define gestures, voice commands to interact with the environment, as well as learn from the data of the residential environment.

Keywords: Internet of Things. Smart Home. Natural Interaction.

Lista de Figuras

Lista de Tabelas

Lista de Códigos

Lista de Acrônimos

6LoWPAN *IPv6 over Low-Power Wireless Personal Area Network* 25

AMQP *Advanced Message Queuing Protocol* 28

API *Application Programming Interface* 32

AWS *Amazon Web Service* 33

BLE *Bluetooth Low Energy* 26

CoAP *Constrained Application Protocol* 26

CoRE *Constrained RESTful Environments* 28

CSV *Comma-Separated Values* 123

ECA *Event-Condition-Action* 67

DIY *Do It Yourself* 35

DLNA *Digital Living Network Alliance* 53

DSL *Domain-Specific Language* 82

FLOSS *Free Libre and Open Source Software* 36

GPS *Global Positioning System* 23

HTTP *Hypertext Transfer Protocol* 26

IETF *Internet Engineering Task Force* 28

IoT *Internet of Things* 15

IP *Internet Protocol* 22

IPv6 *Internet Protocol version 6* 26

JSON *JavaScript Object Notation* 29

LDR *Light Dependent Resistor* 51

LED *Light-Emitting Diode* 23

MAC *Media Access Control* 64

M2M *Machine-to-Machine* 25

MIPS *Million Instructions per Second* 35

MQTT *Message Queuing Telemetry Transport* 26

MQTT-SN *MQTT Sensor Network* 30

NFC *Near Field Communication* 25

PaaS *Platform as a Service* 24

PIC *Peripheral Interface Controller* 62

QoS *Quality of Service* 59

REST *Representational State Transfer* 28

SaaS *Software as a service* 63

SBC *Single-Board Computer* 37

SDK *Software Development Kit* 33

SDN *Software Defined Networking* 119

SO Sistema Operacional 18

SOA *Service-Oriented Architecture* 37

SQL *Structured Query Language* 34

SVM *Support Vector Machine* 55

TaaS *Things as a Service* 51

TCP *Transmission Control Protocol* 26

TTS *Text-to-Speech* 49

UDP *User Datagram Protocol* 26

URI *Uniform Resource Identifier* 30

URL *Uniform Resource Locator* 30

VANET *Vehicular Ad Hoc Network* 23

VoD *Video on Demand* 63

VoIP *Voice over Internet Protocol* 63

WBAN *Wireless Body Area Network* 26

WPAN *Wireless Personal Area Network* 26

WSN *Wireless Sensor Network* 24

XML *eXtensible Markup Language* 29

XMPP *Extensible Messaging and Presence Protocol* 28

Sumário

1

INTRODUÇÃO

A evolução da indústria eletrônica e a miniaturização dos componentes de *hardware* têm proporcionado um avanço significativo para a Ciência da Computação, tornando possível o advento de dispositivos gradativamente menores fisicamente e com poder computacional maior. A possibilidade de embarcar um *software*, que demanda um poder de processamento maior, em circuitos com tamanho bem reduzido, fomenta o surgimento de dispositivos mais inteligentes para aplicação em uma diversidade de áreas.

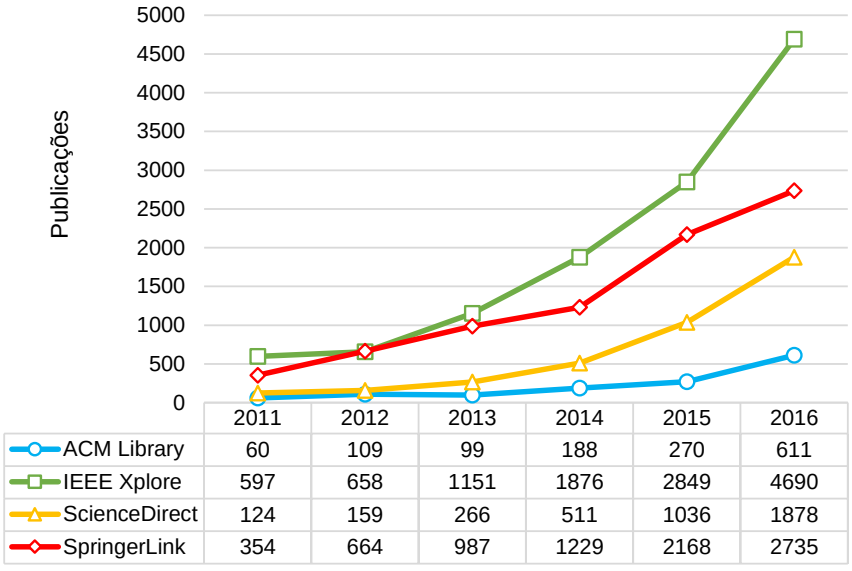
Este fenômeno pode ser observado principalmente por recentes lançamentos do mercado. Atualmente existem diversos dispositivos inteligentes, tais como: o *Nest*, um termostato da Nest Labs (NEST, 2016); o *Roomba*, um robô para limpeza doméstica da iRobot (IROBOT, 2016); a *Philips Hue*, uma lâmpada com gerenciamento *wireless* da Philips (PHILIPS, 2016); a *Fitbit Flex*, uma pulseira composta por sensores corporais da Fitbit (FITBIT, 2016), que possui a função de rastrear e registrar informações de saúde (quantidade de passos, frequência cardíaca, qualidade do sono, entre outros) do usuário; *Parrot Pot*, um vaso de plantas inteligente, disponibilizado pela Parrot (PARROT, 2016).

Muitos destes dispositivos inteligentes[1] necessitam de uma conexão com a *internet* como fonte de "conhecimento" ou apenas para ratificá-lo. Este cenário é característico do conceito denominado de *Internet* das Coisas, ou simplesmente *Internet of Things* (IoT) (BORGIA, 2014).

IoT é o termo utilizado para definir o paradigma onde objetos possuem conexão com a *internet*, criando assim uma rede de coisas conectadas. Dentre os diversos objetos podemos citar alguns exemplos: carro, eletrodoméstico, roupa, móvel, lâmpadas, portas, jardim, estradas. Há previsões que indicam que, até 2020, haverá 50 bilhões de objetos conectados (CISCO, 2015; GOOGLE, 2016). O objetivo da IoT é tornar os objetos meios de geração de informações, bem como mecanismos inteligentes que possam ser programados para tomar uma decisão fundamentada no contexto. Com estas características, a IoT está sendo considerada a terceira onda da *internet* (EVANS; ANNUNZIATA, 2016), impactando em mudanças na arquitetura da própria *internet*, pois espera-se uma escala inédita de "coisas" conectadas. Estas estimativas

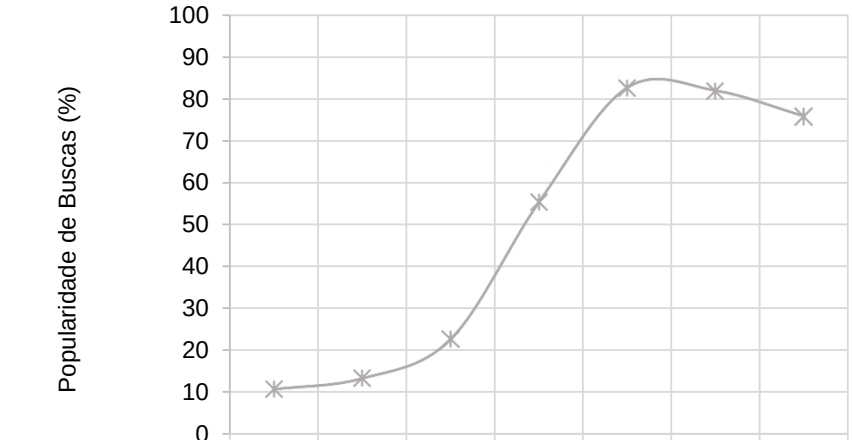
[1]Nesta tese, em futuras referências, podemos utilizar o termo "coisas" ou "objetos" quando nos referirmos a dispositivos conectados.

positivas têm um reflexo nas pesquisas científicas. A Figura 1.1 (a) apresenta a quantidade de publicações relacionadas ao termo *"Internet of Things"* nas principais bases científicas, no período de 2011 à 2016, e a Figura 1.1 (b) ilustra o interesse pelo termo nas pesquisas realizadas no buscador *Google*, no período entre 2011 e 2017. Os valores contidos na Figura 1.1 (b) são relativos à popularidade[2] do termo e não representam os valores absolutos de pesquisas realizadas (GOOGLE, 2017a).



	2011	2012	2013	2014	2015	2016
ACM Library	60	109	99	188	270	611
IEEE Xplore	597	658	1151	1876	2849	4690
ScienceDirect	124	159	266	511	1036	1878
SpringerLink	354	664	987	1229	2168	2735

(a) Publicações científicas.



	2011	2012	2013	2014	2015	2016	2017
Google Trends	10,75	13,33	22,67	55,33	82,58	81,83	75,75

(b) Pesquisas no Google.

Figura 1.1 Tema Internet das Coisas: quantidade de publicações e buscas por ano.

Estes gráficos indicam um crescimento do interesse pela área de IoT, tanto por parte da comunidade acadêmica quanto por parte do mercado e da comunidade em geral. A exceção nestes dados ocorre apenas na quantidade de publicações relacionadas ao termo na base *ACM Library*,

[2] Os números representam o interesse pelo termo em relação ao ponto mais alto do gráfico, variando de 0 à 100, onde 100 é a popularidade máxima.

onde o único ano que não teve mais publicações que o anterior foi 2013, com 99 publicações contra 109 no ano de 2012. Em todos demais anos ocorreu um crescimento na quantidade de publicações e no interesse pelo termo em termos de buscas no *Google*. A Figura 1.2 apresenta um comparativo entre estimativas realizadas pela Gartner em 2015 e 2017 (GARTNER, 2015, 2017a), que corroboram com este cenário.

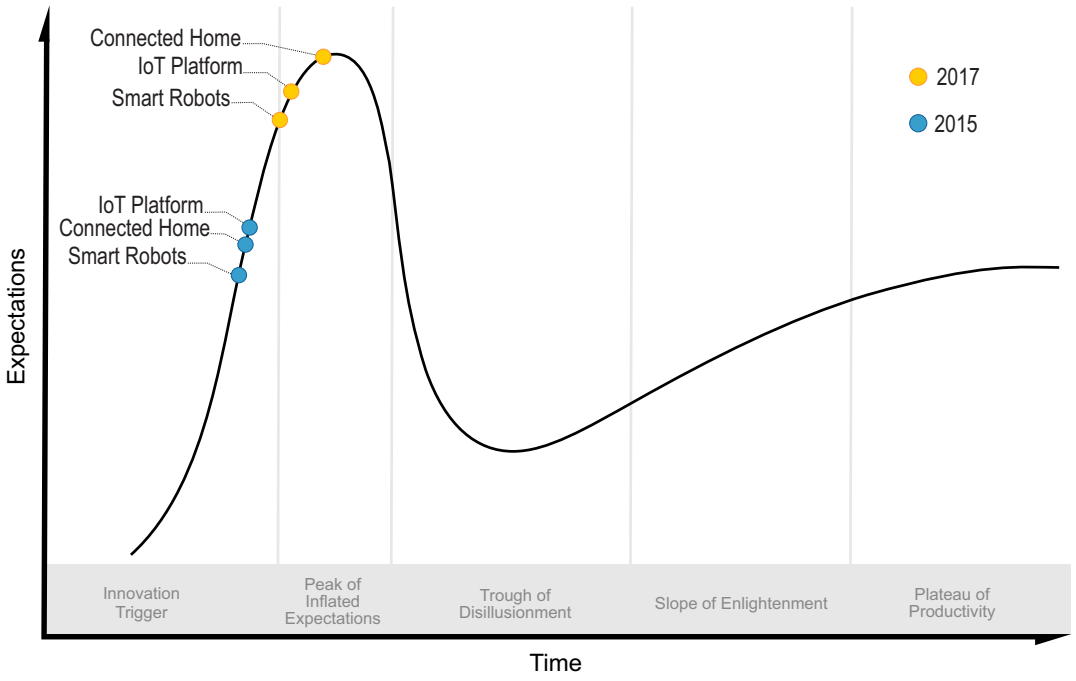


Figura 1.2 Evolução do *Hype Cycle* entre 2015 e 2017.

Anualmente, a Gartner publica um *Hype Cycle* de tecnologias emergentes. O gráfico representa um processo de amadurecimento das tecnologias, onde inicialmente ocorre uma aceleração na expectativa, chegando ao pico em um curto prazo, à medida que as tecnologias amadurecem as expectativas tendem a diminuir. Na Figura 1.2, foram destacadas as tecnologias relacionadas com este trabalho. Em julho de 2015 (GARTNER, 2015), as plataformas para IoT estavam ainda no início da curva. Em julho de 2017 (GARTNER, 2017a), as plataformas para IoT chegaram a um nível mais próximo do pico de expectativa. Uma aceleração ainda mais acentuada foi observada com relação a *Connected Home*[3], o que indica que devido à quantidade de objetos residenciais que poderão "ganhar" inteligência, acredita-se que o ambiente residencial poderá sofrer um grande impacto de transformação de paradigma (VERMESAN et al., 2016; YU et al., 2015a).

A popularização desta tecnologia mostra indícios de que é factível possuirmos residências verdadeiramente inteligentes em alguns anos, como parte de um contexto mais amplo, onde as cidades inteligentes também serão realidade. As residências inteligentes têm diversas funções, tais como: gerenciar de forma eficiente os recursos naturais consumidos, auxiliar em tarefas

[3] De acordo com o glossário da Gartner, o termo *Connected Home* possui o mesmo conceito que *Smart Home*, *Smart House* ou Residências Inteligentes (GARTNER, 2017b).

do cotidiano, criar estímulos educativos, direcionar os habitantes a terem uma vida saudável, entretenimento, gerenciamento financeiro, segurança, conforto, entre tantas outras possibilidades (ROBLES; KIM, 2010; VIANI et al., 2013).

Neste contexto, possuir uma percepção do ambiente torna-se essencial para tais atividades, pois esta dará todo o entendimento necessário ao sistema residencial sobre a composição da residência. Isto é, através do sensoriamento é possível mapear as principais características do cenário e, consequentemente, dos indivíduos, podendo assim traçar padrões e acumular conhecimento de modo incremental.

Contudo, existem alguns desafios que são empecilhos para a evolução desejada deste segmento, porém podem ser também elementos catalisadores nesse processo, a citar: (i) a ausência de um padrão estabelecido de comunicação entre todos os componentes envolvidos, mesmo com a existência de uma série de protocolos, tem como consequência um conjunto de soluções limitadas, onde as plataformas, muitas vezes, suportam apenas determinados dispositivos, restringindo-se assim a um número reduzido de possibilidades (YAQOOB et al., 2017; SETHI; SARANGI, 2017; MUNJIN; MORIN, 2011). Um exemplo deste cenário é a plataforma da Apple, a HomeKit (APPLE, 2015), que possui poucos dispositivos homologados para o funcionamento (APPLE, 2016); (ii) a complexidade de desenvolvimento de aplicações para *hardware*, sem uma camada de abstração (SETHI; SARANGI, 2017); (iii) dificuldade de visualização e gestão da massa de dados (SETHI; SARANGI, 2017); (iv) escassez de plataformas específicas para a construção de soluções integradas e interoperáveis (YAQOOB et al., 2017; ARAS, 2016; MUNJIN; MORIN, 2011); e, por fim, mas não limitando-se a estes, ainda que os aspectos supracitados fossem solucionados, seria necessário (v) desenvolver mecanismos de interação não somente reativa mas também proativa com o usuário, de forma personalizada, fundamentando-se na própria relação entre o habitante e o ambiente.

1.1 Objetivos

Propor uma arquitetura de um Sistema Operacional (SO) para Internet das Coisas em um ambiente residencial, com suporte à interação natural, sob uma perspectiva de aprendizado contextual. Assim como DIXON et al. (2012), consideramos que o SO residencial deve gerenciar os recursos da residência, assim como conceitualmente um sistema operacional gerencia recursos como processamento, armazenamento, periféricos, entre outros. Neste trabalho o termo SO é utilizado no sentido análogo ao gerenciamento de recursos de uma residência. A proposta tem como finalidade abstrair a camada de *hardware*, provendo subsídios para auxiliar o desenvolvimento de aplicações no âmbito residencial.

Os objetivos específicos do trabalho estão apresentados a seguir:

- Validar o modelo da arquitetura de Sistema Operacional, através de um estudo de caso;

- Implementar um protótipo de Sistema Operacional para IoT, com base nas definições arquiteturais propostas;
- Realizar uma avaliação dos principais recursos do Sistema Operacional.

1.2 Métodos de Pesquisa

A metodologia adotada para realizar esta pesquisa fundamenta-se em quatro abordagens, quanto à natureza, quanto ao problema, quanto aos objetivos e quanto aos procedimentos técnicos (PRODANOV; FREITAS, 2013).

De acordo com a natureza, esta é uma Pesquisa Aplicada, onde são aplicados conhecimentos práticos dirigidos à solução de problemas específicos do contexto de IoT para residências inteligentes.

Considerando a abordagem do problema, esta é uma Pesquisa Qualitativa, pois através de estudos de caso e fundamentando-se nos requisitos extraídos da literatura, observamos os fenômenos que nossa proposta contempla.

Quanto ao método para alcançar os objetivos esta pesquisa é Exploratória, pois possui o intuito de determinar os problemas, definir a hipótese e, principalmente, explorar o objetivo através da proposição de novas camadas na arquitetura IoT.

Quanto à perspectiva dos procedimentos técnicos, esta pesquisa contempla diversos métodos, entretanto destacaremos apenas os dois principais:

- Estudo de Caso: Utilizado para avaliar a arquitetura proposta (Capítulo 5), por meio de casos fornecidos pelo protótipo de SO que representem adequadamente os requisitos da arquitetura. Os estudos de caso devem permitir a observação e a análise detalhada dos componentes da arquitetura.
- Pesquisa Experimental: Utilizado para avaliar o protótipo de SO (Capítulo 6), selecionando, identificando e analisando as variáveis que seriam capazes de influenciar e produzir um impacto nos requisitos da plataforma.

Os procedimentos adotados para a execução desta pesquisa foram norteados por uma metodologia que abrange desde a compreensão das características e requisitos de uma arquitetura de SO para IoT para residências até a análise dos resultados inerentes à própria pesquisa. As etapas do estudo são representadas por um fluxograma, ilustrado na Figura 1.3.

O fluxo de atividades contidas na metodologia é descrito em pormenor, como segue.

- **Definição e Levantamento:** Nesta etapa, é realizada a delimitação do escopo da pesquisa. Compreende a identificação das características e arquiteturas da IoT, no domínio de aplicações para residências inteligentes. Além das plataformas existentes e suas características, é realizado um levantamento dos protocolos, componentes,

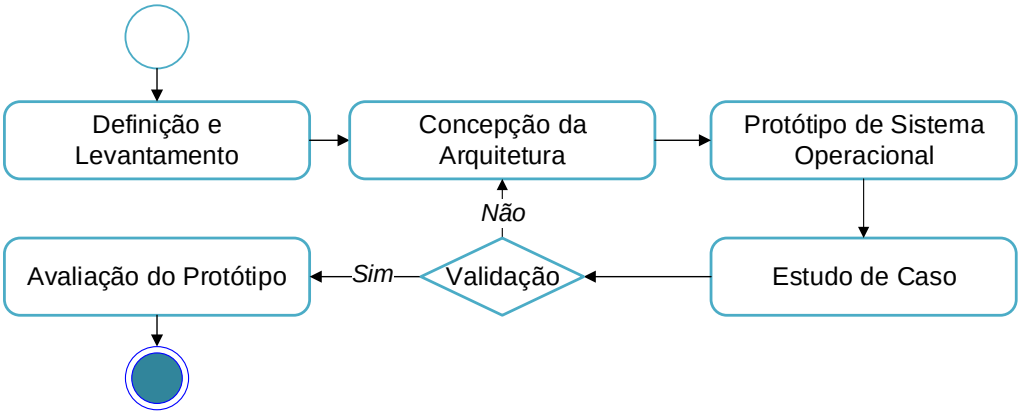


Figura 1.3 Fluxograma da metodologia utilizada.

requisitos e suas interações. Atrelado a esta etapa é realizado um levantamento de pesquisas mais recentes e relevantes dentro da área estudada, onde foram realçados os requisitos abordados por estes. Ainda nesta etapa, foram definidas métricas quantitativas e qualitativas, de acordo com as variáveis que pretendemos observar. Essa etapa torna-se essencial, pois fornece um amplo entendimento da área estudada, possibilitando tomar conhecimento de suas principais funcionalidades e desafios em aberto para o planejamento e concepção de uma nova arquitetura.

- **Concepção da Arquitetura:** Esta etapa consiste na elaboração de uma proposta de arquitetura IoT para residências inteligentes, contendo os principais requisitos que atendam a demanda de um sistema operacional para este contexto. A concepção e requisitos da arquitetura são detalhados no Capítulo 4. A partir da definição da arquitetura IoT para residências, é possível implementar um protótipo de sistema operacional.

- **Protótipo de Sistema Operacional:** O protótipo de SO para residências é gerado com base na arquitetura delineada na etapa anterior. Desta forma, a partir da definição da arquitetura IoT, foi desenvolvido um protótipo de SO para residências inteligentes com o intuito de gerenciar os recursos da residência, abstraindo a complexidade de *hardware* e *software*, bem como aplicando técnicas de interação natural.

- **Estudo de Caso:** A partir da definição da arquitetura e do protótipo de SO, foram elaborados estudos de caso com o objetivo validar a arquitetura proposta. Os estudos de caso apresentam uma avaliação qualitativa em relação aos principais requisitos da arquitetura do sistema operacional para IoT para o ambiente residencial. No Capítulo 5, os estudos de caso são apresentados e discutidos. Os estudos de caso foram utilizados para verificar a validade da arquitetura proposta no Capítulo 4, em cenários representativos do ambiente residencial.

- **Validação:** Para que a arquitetura do sistema operacional proposta seja considerada

válida, é necessário que todos os requisitos e componentes se mostrem operantes e funcionais. Caso a condição não seja satisfatória, ajustes na proposta de arquitetura devem ser realizados. Uma vez que satisfaça a condição, o próximo passo é a avaliação e análise do protótipo de SO desenvolvido.

- **Avaliação do Protótipo:** Por fim, nesta última etapa, com a arquitetura IoT definida e validada, devemos realizar experimentos para avaliar, de forma quantitativa, alguns aspectos, tais como: desempenho de plataformas de *hardware*, meios de transmissão, tempo de resposta, escalabilidade, tempo para a descoberta de dispositivos e assertividade das regras definidas pelos usuários e do aprendizado adquirido com base no reconhecimento de padrões. Além desses, devem ser analisados ainda os mecanismos de interação natural, sob os aspectos de assertividade. A avaliação e resultados são apresentados no Capítulo 6. Ainda nesta etapa, com intuito de validar os dados coletados, é apresentada a validação dos experimentos conduzidos. O método *Bootstrap* é adotado para determinar o intervalo de confiança das amostras coletadas. O método e os procedimentos realizados são descritos na Seção 6.1.

1.3 Estrutura da Tese

Esta tese está estruturada em 7 capítulos. O presente (**Capítulo 1**) é introdutório, contendo contextualização, motivação, objetivos, métodos da pesquisa e os trabalhos relacionados com a presente tese, e os demais apresentam-se como segue.

O **Capítulo 2** apresenta o *background* conceitual sobre *Internet* das Coisas, delineando as tecnologias, protocolos, plataformas e arquiteturas utilizadas nesta área.

O **Capítulo 3** discorre sobre o conceito de residências inteligentes, apresentando um histórico da área e o uso de tecnologias de informação no ambiente residencial. Técnicas de interação natural entre o habitante e a residência também são discutidos.

O **Capítulo 4** apresenta a proposta de arquitetura para um SO para residências inteligentes, sendo detalhadas as suas camadas e componentes idealizados para este trabalho.

O **Capítulo 5** descreve o protótipo de SO e os casos utilizados para validar a arquitetura proposta no Capítulo 4. Neste capítulo são apresentadas as técnicas aplicadas no desenvolvimento do protótipo, analisando, de forma qualitativa, os requisitos contemplados pelo protótipo.

O **Capítulo 6** apresenta uma série de experimentos de caráter quantitativo, que visam avaliar o protótipo de SO, delineando os aspectos mais relevantes para serem observados em futuras implementações desta arquitetura, tais como: dispositivos, conexões, técnicas, entre outros.

O **Capítulo 7** discute os resultados obtidos durante o desenvolvimento desta pesquisa, fundamentando-se na proposta descrita no Capítulo 4 e no Capítulo 5. Por fim, apresenta as contribuições, limitações e trabalhos futuros.

2

INTERNET DAS COISAS

Este capítulo apresenta um background conceitual sobre o tema da Internet das Coisas, explorando as principais tecnologias utilizadas, bem como conceitos intrínsecos como interoperabilidade. Os protocolos de comunicação são destacados neste capítulo, pois representam um aspecto importante no entendimento da estrutura da IoT. Por fim, discutimos algumas abordagens de plataformas utilizadas para dar suporte ao desenvolvimento de soluções neste contexto.

A *Internet* das Coisas pode ser conceituada como uma rede de objetos interconectados. Estes objetos podem ser qualquer coisa física, que podem ainda possuir seus próprios endereços *Internet Protocol* (IP), podendo trocar informações através de uma rede de sensores (ZARGHAMI, 2017; KRAMP; KRANENBURG; LANGE, 2013). Para BANDYOPADHYAY; SEN (2011), a palavra *internet* no termo refere-se à infraestrutura de rede com capacidade escalável e configurável com base nos protocolos de comunicação interoperáveis, já a palavra "coisa" remete aos objetos físicos ou virtuais que possuem identificadores, atributos e metadados que podem fornecer informações do meio físico através da rede.

A pesquisa em IoT vem se intensificando anualmente e ainda pode ser considerada uma tecnologia emergente, apesar do termo *"Internet of Things"* ter sido utilizado pela primeira vez em 1999[1], por Kevin Ashton (ASHTON, 2009). Em meados da década de 2000 surgiram os primeiros trabalhos mais específicos, tendo como enfoque soluções para IoT. Para exemplificar, o trabalho mais antigo encontrado na base digital ACM Library relacionado à IoT foi publicado em dezembro de 2007 (CARBONI; ZANARINI, 2007).

Contudo, o principal catalisador é a quantidade de novos sensores disponíveis no mercado, ampliando assim as possibilidades de exploração. Em 2015 existiam $3,47$ dispositivos por pessoa, e estima-se que esta proporção irá para $6,58$ coisas conectadas por habitante em 2020 (HENDERSON, 2015; MACAULAY; BUCKALEW; CHUNG, 2016).

As aplicações que utilizam tecnologias e conceitos de IoT tem proporcionado avanços em alguns cenários. Logo, podemos citar alguns exemplos de aplicações no contexto de cidades

[1] Inicialmente, foi utilizado fora do contexto, sob a compreensão atual da IoT, pois referia-se à comunicação de objetos através apenas de RFID.

inteligentes, trânsito e, principalmente, no ambiente residencial, que é o ambiente de enfoque deste trabalho:

- *Cidade Inteligente*: O conceito de Cidade Inteligente (*Smart City*) tem proporcionado o surgimento de diversas soluções para, em geral, melhorar a qualidade de vida dos cidadãos ou mesmo gerenciar os recursos de forma mais eficiente. Um exemplo deste tipo de iniciativa está na cidade de San Jose, nos Estados Unidos, onde foram substituídas as lâmpadas antigas por díodos emissores de luz (ou *Light-Emitting Diode* (LED)), gerenciadas por um sistema que alerta sobre possíveis defeitos, agilizando o reparo, bem como reduz a intensidade da iluminação pública à depender de fatores como horário, fluxo de pessoas, entre outros (GHARAIBEH et al., 2017). O *Snow cleaning management project* é um projeto da cidade de Quebec, no Canadá, que gerencia a limpeza da neve através de sensores em cada uma das máquinas de limpeza da neve (ALAWADHI et al., 2012). Um terceiro exemplo é o *NYC311*, que é um sistema de operação da cidade de Nova Iorque, nos Estados Unidos, que permite o acesso rápido e fácil a serviços não emergenciais e informações através de múltiplos canais (quiosque, *e-mail*, mídias sociais e aplicativos de *smartphones*) (NAM; PARDO, 2013);

- *Trânsito*: A aplicação da *Internet* das Coisas na engenharia de trânsito também agrega diversos benefícios. Um exemplo de aplicação neste sentido é o UbiBus (UBIBUS, 2015), que tem como objetivo facilitar o dia a dia das pessoas que utilizam transporte público, oferecendo acesso inteligente a informações de transporte público aos passageiros, em tempo real, baseado em informações dinâmicas de contexto relacionadas aos próprios meios de transporte. Outra iniciativa utiliza uma *Vehicular Ad Hoc Network* (VANET), em Pequim (China), para coletar informações sobre qualidade do ar, posição de *Global Positioning System* (GPS), através da utilização de sensores em mais de 20.000 táxis, possibilitando, com isso, aplicações para o monitoramento dinâmico da qualidade do ar, vigilância das emissões de carbono e análise do tráfego. Ao utilizar os táxis como coletores de dados, a iniciativa visa obter uma compreensão da dinâmica da cidade e gerar dados para a tomada de decisão (DING et al., 2011);

- Residência Inteligente ou *Smart Home*: Focaliza no ambiente doméstico inteligente para a provisão de serviços computacionais para seus habitantes. No Capítulo 3, é apresentado de forma mais detalhada todo o contexto de Residência Inteligente. A Residência Inteligente proporciona novas funcionalidades e novas tecnologias para permitir aos habitantes mais praticidade, conforto e segurança. Como exemplos de serviços podemos citar: Belkin Wemo (BELKIN, 2015), aparelhos domésticos inteligentes capazes de serem monitorados à distância, como tomadas, lâmpadas e câmeras; HarvestGeek (GEEK, 2015), um projeto *open source* para monitoramento de estufas, fazendas ou jardins domésticos.

2.1 Rede de Sensores

Os recentes avanços tecnológicos em circuitos integrados de baixa potência e comunicações sem fio permitiram a criação de dispositivos cada vez menores, com baixo consumo de energia e com baixo custo, tornando-se soluções eficientes para uso em aplicações de sensoriamento remoto.

2.1.1 *Wireless Sensor Network* - WSN

Tipicamente, uma *Wireless Sensor Network* (WSN) é uma rede de sensores usada para coletar informação em tempo real (SHELBY; BORMANN, 2011). Essa rede é composta por uma nuvem de sensores que trocam informações entre si e podem também se conectar com mecanismos externos como uma *Platform as a Service* (PaaS)[2] e assim ampliar a cobertura e a dimensão da rede.

Cada sensor que compõe essa rede é considerado um nó. Estes nós são dispositivos inteligentes de baixa potência que possuem um ou mais sensores, um processador, memória, uma fonte de alimentação (de baixo consumo) e um mecanismo de transmissão de dados (AKYILDIZ et al., 2002). Analisando sob a perspectiva da IoT, cada nó é uma "coisa" ou objeto conectado, podendo ser uma lâmpada, um veículo, ou mesmo uma residência, como ilustrado na Figura 2.1.

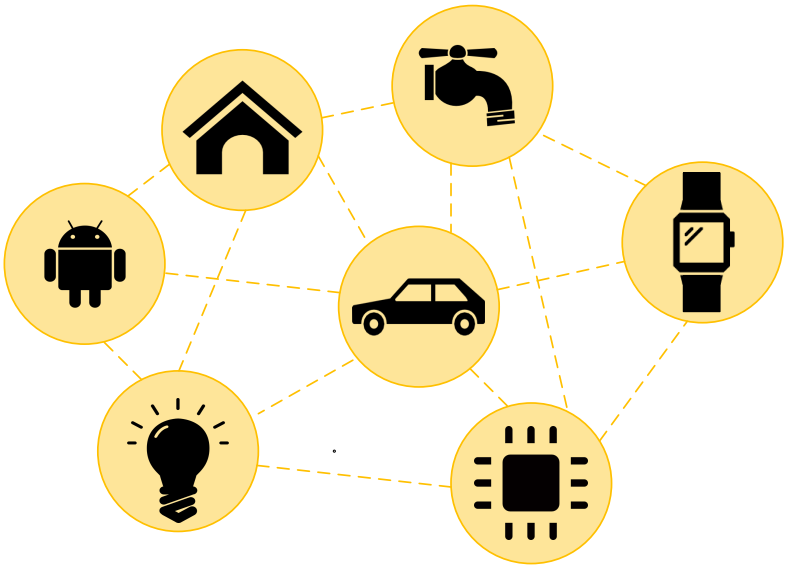


Figura 2.1 Exemplo de topologia de rede de objetos inteligentes conectados.

Uma rede de sensores pode ser auto-organizável, deste modo, os nós de sensores podem criar automaticamente uma rede e a posição dos nós não precisa ser pré-determinada, não tendo necessidade de se ligar a uma rede estabelecida previamente.

[2]Modelo de negócio de Computação em Nuvem. Segundo MELL; GRANCE (2010), o PaaS, ou Plataforma como um Serviço, fornece uma plataforma de desenvolvimento como serviço; suporta um conjunto de interfaces de aplicativos para o desenvolvimento de aplicações na nuvem. Exemplo de provedores deste modelo temos: o *Google App Engine, Microsoft Azure, Force.com*, entre outros.

Os nós da rede podem se comunicar com qualquer outro nó, mesmo que seja necessária uma comunicação com múltiplos saltos (utilizando nós intermediários para transmitir a informação). Ainda assim é esperado que os dispositivos da rede de sensores sem fio consumam menos energia do que em uma comunicação tradicional de salto único (comunicação direta entre os nós de origem e destino), pois a transmissão entre grandes distâncias consome mais energia que em distâncias mais curtas, devido à necessidade de utilização da potência máxima do sinal (AKYILDIZ et al., 2002). Isto é essencial para os dispositivos utilizados no ambiente de IoT, pois a fonte de energia destes produtos é muitas vezes limitada e o gargalo de consumo é justamente a transmissão de dados (AKYILDIZ et al., 2002).

Uma característica importante da WSN é que a topologia desta rede é dinâmica, isto é, nós podem ser adicionados ou podem ser retirados, ou ainda podem simplesmente ficar indisponíveis por algum motivo, não comprometendo o funcionamento adequado da rede.

A WSN possui enfoque nas camadas de *hardware* e transporte, enquanto que a IoT possui um sentido mais amplo, abrangendo camadas de armazenamento, negócio e aplicação, além de *hardware* e transporte. Outro ponto de divergência é que IoT faz uso de diversas tecnologias abertas e tem o foco voltado mais para o usuário final, enquanto que a WSN, muitas vezes, utiliza tecnologias proprietárias, voltado ao público corporativo (LI; KARA, 2017; XU; COLLIER; O'HARE, 2017).

2.1.2 *Machine-to-Machine*

Enquanto a WSN possui enfoque nos sensores, a comunicação entre máquinas (ou *Machine-to-Machine* (M2M)) possui enfoque principalmente nos dispositivos, máquinas e equipamentos que podem se comunicar entre si através de meios de transmissão com ou sem fio. A comunicação M2M permite a troca de dados entre dispositivos de forma autônoma, sem intervenção humana, sendo constituída por diversos dispositivos e *gateway*, responsável pela conexão entre dispositivos da rede e, eventualmente, outras redes (TUNA et al., 2017).

No domínio da IoT, as escolhas de protocolos utilizados comunicação entre máquinas são essenciais para um projeto de implantação do serviço. É importante considerar o custo e a facilidade para interligar os dispositivos através de ondas de rádio frequência, sem necessidade de utilização de cabos, bem como considerar protocolos de favorecem a gestão das informações que circulam no ambiente.

As possibilidades de tecnologias de comunicação que podem ser aplicadas à IoT são bem amplas e, eventualmente, a maioria das tecnologias sem fio que suportam de alguma forma a transferência de dados, detecção e controle, são passíveis de adoção no ambiente de IoT.

Algumas tecnologias de comunicação são bem estabelecidas e conhecidas, tais como WiFi, Bluetooth e redes móveis (3G/4G). Porém, há uma diversidade de opções de redes emergentes, tais como *IPv6 over Low-Power Wireless Personal Area Network* (6LoWPAN), *Near Field Communication* (NFC), Bluetooth LE, ZigBee e Z-Wave. O que irá definir qual

protocolo utilizar serão os requisitos da aplicação, e requisitos não funcionais como a frequência, diretrizes de segurança, potência de sinal e consumo energético.

2.1.2.1 IPv6 over Low power Wireless Personal Area Network - 6LoWPAN

O 6LoWPAN é um protocolo de rede que define mecanismos de encapsulamento e compressão dos cabeçalhos dos pacotes de rede (SHELBY; BORMANN, 2011). O protocolo 6LoWPAN foi projetado para trabalhar sobre o protocolo *Internet Protocol version 6* (IPv6), porém focado em redes *Wireless Personal Area Network* (WPAN) de baixa potência, enviando pacotes IPv6 através de redes baseadas em implementações do padrão IP, podendo ser utilizado em várias plataformas de comunicação que implementam este padrão, incluindo *Transmission Control Protocol* (TCP), *User Datagram Protocol* (UDP), *Hypertext Transfer Protocol* (HTTP), *Constrained Application Protocol* (CoAP), *Message Queuing Telemetry Transport* (MQTT) e WebSockets.

O 6LoWPAN oferece comunicação de ponta a ponta entre nós endereçáveis. Permite ainda a inclusão de roteadores para gerenciamento de rotas e otimização de entregas. Outro atributo fundamental que vale salientar é a pilha IPv6, que permite a ampla expansão da IoT, oferecendo uma quantidade de endereços IP suficiente para cada pessoa no mundo, permitindo que qualquer objeto ou dispositivo incorporado no mundo possa ter seu próprio endereço IP único, podendo assim, conectar-se à *internet* (SHELBY; BORMANN, 2011).

2.1.2.2 Near Field Communication - NFC

Nos últimos anos o termo NFC se tornou mais presente, especialmente por estar sendo incorporado em alguns dispositivos móveis disponíveis no mercado. O NFC é uma tecnologia que permite interações bidirecionais simples e seguras entre dispositivos eletrônicos, permitindo aos consumidores realizar troca de informação e conectar dispositivos eletrônicos sem contato físico direto, utilizando apenas o campo eletromagnético dos dispositivos envolvidos na transação (WHITMORE; AGARWAL; DA XU, 2015).

2.1.2.3 Bluetooth LE

O Bluetooth talvez seja a tecnologia M2M mais utilizada, principalmente pela quantidade de dispositivos que suportam a mesma. A versão *Bluetooth Low Energy* (BLE) é um protocolo relevante para as aplicações da IoT, pois ao mesmo tempo que oferece uma conectividade similar ao Bluetooth convencional, foi projetado para oferecer um consumo de energia significativamente reduzido.

Entretanto, o BLE não é recomendado para transferência de arquivos, sendo mais adequado para pequenos blocos de dados. Devido a sua limitação de cobertura, o BLE é bastante utilizado em WPANs e *Wireless Body Area Network*s (WBANs) (GEORGAKAKIS et al., 2011).

Deve-se enfatizar que a versão 4.2 BLUETOOTH (2016) permite que sensores com suporte ao BLE possam acessar a *internet* diretamente, via conectividade 6LoWPAN (HANSEN, 2015). Esta conectividade IP torna possível a utilização de infraestrutura já existente para gerenciar dispositivos Bluetooth LE (RAZA et al., 2015).

2.1.2.4 ZigBee

O ZigBee, assim como o Bluetooth, é um protocolo já estabelecido entre os especialistas, porém menos conhecido do público em geral. Tradicionalmente é mais aplicado em ambientes industriais. O ZigBee é baseado no protocolo IEEE 802.15.4, que é uma tecnologia de rede sem fio padrão que opera a uma frequência de 2,4 GHz adequada para aplicações que requerem baixa frequência de troca de dados e relativamente baixas taxas de transferência em ambiente de área restrita (operando dentro de uma área de cobertura de no máximo 100 metros), como em uma residência, por exemplo (SAFARIC; MALARIC, 2006).

O ZigBee é bastante adequado para aplicações de IoT, pois possui algumas características significativas para sistemas restritos, a citar: baixo consumo de energia, alto nível de segurança, robustez a falhas, alta escalabilidade e suporta grande concentração de dispositivos, sendo indicado para WSN.

2.1.2.5 Z-Wave

Assim como as demais tecnologias de comunicação apresentadas até o momento, o Z-Wave também possui baixo consumo energético e funciona por rádio frequência (ALLIANCE, 2017). O Z-Wave foi projetado inicialmente para a automação residencial, sendo utilizado em produtos como controladores de lâmpadas e sensores, entre outros. O protocolo foi otimizado para estabelecer uma comunicação confiável e de baixa latência, utilizando pequenos pacotes de dados, com taxas de transferência de dados de até 100 kbps. O sinal Z-Wave sofre pouca interferência externa como ocorre com outras redes (WiFi, Bluetooth ou ZigBee).

Em redes Z-Wave não existe a necessidade de um nó coordenador, sendo facilmente escalável e permitindo o controle de até 232 dispositivos (ALLIANCE, 2017). Z-Wave usa um protocolo mais simples, se comparado ao ZigBee, que pode permitir um desenvolvimento mais rápido e mais fácil.

2.2 Protocolos

No intuito de encontrar soluções para otimizar a comunicação entre os dispositivos com recursos computacionais reduzidos, alguns protocolos leves que não requerem o uso extensivo de recursos foram surgindo ao decorrer da evolução da IoT. Algumas linguagens de *script* são as escolhas preferidas utilizadas por aplicações em IoT.

Contudo, como citado anteriormente, muitos destes protocolos propostos são divergentes e concorrentes. Os protocolos mais difundidos são: CoAP, MQTT, *Extensible Messaging and Presence Protocol* (XMPP) e *Advanced Message Queuing Protocol* (AMQP). Contudo, iremos focar nos protocolos CoAP, MQTT e na arquitetura *Representational State Transfer* (REST), pois foram aplicados no desenvolvimento deste trabalho. Além destes, discutiremos o protocolo *WebSocket*, que apesar de ser utilizado por aplicações de IoT, não foi desenvolvido para este propósito.

2.2.1 *Constrained Application Protocol* - CoAP

O *Constrained Application Protocol* é um protocolo de rede genérico fundamentado no modelo cliente-servidor da *internet* convencional, similar ao HTTP, onde, normalmente, o lado servidor é um sensor responsável por prover informações e o lado cliente é o consumidor dessa informação. Entretanto, este protocolo foi projetado para ser aplicado em dispositivos com capacidades de processamento, memória ou energia limitadas (VERMESAN; FRIESS, 2014).

O CoAP foi desenvolvido em 2010 por *Constrained RESTful Environments* (CoRE), uma equipe de trabalho do *Internet Engineering Task Force* (IETF), sendo especificado em 2014 e formalizado como o padrão RFC 7252 (SHELBY; HARTKE; BORMANN, 2014), sendo evoluído desde então (AMARAN et al., 2015). Baseando-se na comunicação de dispositivos que coexistem na mesma rede restrita, o CoAP suporta endereçamento *broadcast*[3] e *multicast*[4], utilizando como padrão o protocolo UDP, não suportando a comunicação TCP. Contudo, o CoAP provê suporte à descoberta de dispositivos, fornecendo assim subsídios para a troca de informações com outros dispositivos já existentes na rede.

O CoAP foi projetado para funcionar bem em ambientes heterogêneos (onde há uma diversidade de dispositivos com características distintas) e, portanto, prover suporte à interoperabilidade com a web, principalmente por ter sua arquitetura similar à REST (ver Seção 2.2.3) e suportar também comunicações assíncronas, onde os dados podem ser acessados por demanda (AMARAN et al., 2015).

2.2.2 *Message Queuing Telemetry Transport* - MQTT

Como o CoAP, o MQTT é considerado um dos mais promissores protocolos aplicados a IoT. O MQTT é um protocolo de código aberto que utiliza um *Broker* [5] central para viabilizar a troca de dados entre os clientes. Este protocolo é leve e baseia-se no modelo *publish/subscribe*, utilizando uma conexão TCP (MQTT, 2017). A Figura 2.2 ilustra o modelo de comunicação utilizado neste protocolo.

[3]Transmissão de mensagens para todos os dispositivos conectados na rede.

[4]Transmissão de mensagens para um grupo específico de dispositivos conectados na rede.

[5]O *Broker* é responsável por realizar a mediação entre os clientes, bem como gerenciar a troca de mensagens entre estes (BROWN, 2015).

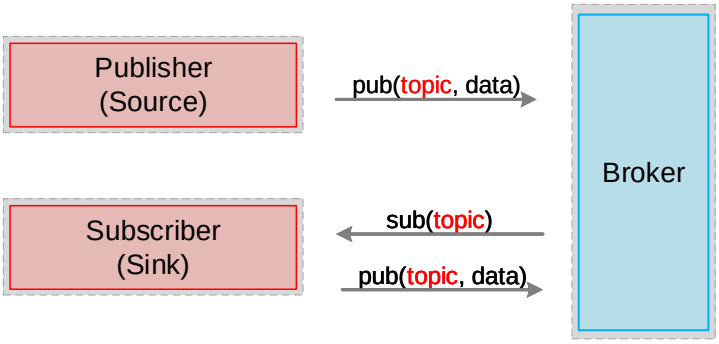
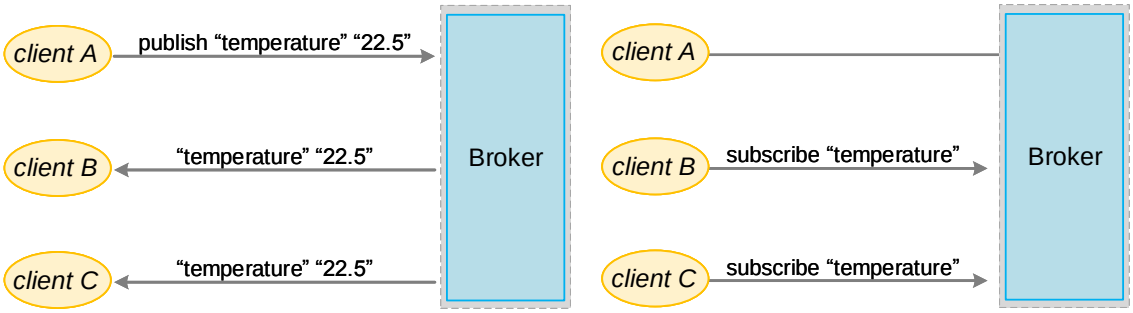


Figura 2.2 Modelo de comunicação do protocolo MQTT.

De modo geral, neste modelo, o *Broker* é o elemento central, atuando como intermediário na comunicação entre os mecanismos *Publisher* e *Subscriber*. O *Publisher* é o responsável por enviar uma informação para o *Broker*, identificada por um endereço, chamado de tópico. O *Subscriber*, quando se associa a um tópico existente, tem acesso à informação enviada pelo *Publisher*.

Um *Publisher* pode publicar as informações em vários tópicos e, por sua vez, um *Subscriber* pode se inscrever em diversos tópicos distintos. O MQTT não define o tipo dos dados contidos na mensagem (*playload*), o *Publisher* pode enviar dados binários, dados textuais ou dados estruturados como *eXtensible Markup Language* (XML) e *JavaScript Object Notation* (JSON). A Figura 2.3 apresenta um exemplo de troca de mensagens entre clientes no protocolo MQTT.



(a) MQTT Publisher.

(b) MQTT Subscriber.

Figura 2.3 Exemplo de uso de cliente MQTT.

Na Figura 2.3 (a) é possível observar que o “client A” publica no tópico “temperature” o valor 22.5 e os clientes “client B” e “client C” recebem os valores contidos no tópico de interesse naquele instante; isto só é possível se os clientes receptores estiverem inscritos no tópico “temperature”. A Figura 2.3 (b) mostra justamente um exemplo de inscrição, onde os clientes B e C se inscrevem no tópico “temperature”, podendo assim receber informações publicadas neste tópico.

O MQTT fornece suporte parcial para interoperabilidade, pois não fornece mecanismos que possam ser utilizados para dar significado (semântica) aos dados trocados entre os clientes

(MQTT, 2017). Entretanto, várias versões do MQTT estão disponíveis para atender limitações específicas. Um exemplo é o *MQTT Sensor Network* (MQTT-SN) (SHELBY; BORMANN, 2011), que é destinado a dispositivos embarcados em redes que não trabalham com TCP/IP, como ZigBee.

2.2.3 *Representational State Transfer* - REST

REST é uma arquitetura independente para a concepção de aplicações de rede utilizando HTTP simples para prover conexão entre máquinas. O objetivo do REST é representar um conjunto de entidades ou recursos de uma aplicação através de um mapeamento conceitual. Cada recurso possui um identificador único, chamado *Uniform Resource Identifier* (URI), que permite o acesso a tais recursos através de um *Uniform Resource Locator* (URL). Deste modo, é possível acessar a representação do recurso, por meio de uma interface uniforme, sem a necessidade de acessar diretamente o recurso (FIELDING, 2000).

2.2.4 *WebSocket*

O protocolo WebSocket, diferentemente dos demais apresentados nesta seção, não é específico para dispositivos com poder computacional restrito, sendo aplicado também em infraestruturas web mais complexas. No entanto, é uma abordagem interessante para ser utilizada em produtos de IoT que utilizam serviços de Computação em Nuvem. O WebSocket é um protocolo que proporciona a comunicação TCP entre cliente e servidor de forma bidirecional, por isso pode ser utilizado por qualquer aplicação que adote este modelo, principalmente para interfaces cliente que utilizam HTTP (FETTE; MELNIKOV, 2011).

2.3 Interoperabilidade

A IoT no ambiente residencial impulsiona o desenvolvimento de aplicações mais específicas ao contexto. Entretanto, devido a sua essência heterogênea e complexidades tecnológicas, muitas dessas aplicações não são trivialmente interoperáveis.

Este problema se amplifica na medida em que surgem, cada vez mais, dispositivos desenvolvidos por fabricantes diferentes utilizando seus próprios padrões, e ao mesmo tempo há uma demanda para que estes dispositivos funcionem em conjunto (LIU, 2007).

Conceitualmente, o termo Interoperabilidade está relacionado à possibilidade de um ou mais sistemas interagirem e compartilharem dados, recursos e processos (LIU; BENFELL, 2009; ASUNCION; SINDEREN, 2011; LEE et al., 2007). Logo, a Interoperabilidade está para a Integração de Sistemas, como as Redes Computacionais estão para os Computadores.

Sendo a interoperabilidade importante no sucesso da IoT, a mesma só pode ser atingida se os sistemas forem capazes de interoperar os seus dados, recursos e processos através de uma

semântica definida para o contexto, apesar da diferença nos formatos dos dados, interfaces, protocolos de comunicação e tipos de mensagens (ASUNCION; SINDEREN, 2011).

De acordo com a revisão sistemática realizada por ASUNCION; VAN SINDEREN (2010), existem diversos níveis de interoperabilidade: técnica, semântica e pragmática. O nível técnico trata da comunicação entre as aplicações na forma de troca de mensagens, isto é, os sistemas podem comunicar-se uns com os outros e serem compreendidos. A interoperabilidade semântica trata do entendimento lógico do conteúdo das mensagens trocadas pelos emissores e receptores. A interoperabilidade pragmática engloba os níveis anteriores, sendo possível ainda identificar a disponibilidade dos envolvidos para executarem determinadas ações. O presente trabalho contempla estes três níveis destacados.

2.3.1 Interoperabilidade Técnica

A interoperabilidade técnica é geralmente associada com componentes de *hardware* e *software*, bem como a plataformas que permitem a comunicação M2M. Comumente, a interoperabilidade técnica é centrada nos protocolos de comunicação (Seção 2.2) e infra-estrutura necessária para os protocolos operarem (VERMESAN et al., 2016).

No cenário de IoT em geral, o desafio da interoperabilidade técnica está em prover a troca de informações entre os produtos disponíveis no mercado. De acordo com SERRANO et al. (2015), nesta abordagem, as soluções devem dedicar os esforços em: (1) Construção de *guidelines* para o desenvolvimento e implementação de projetos; (2) Avaliação de plataformas experimentais; (3) Soluções com redes de sensores; e (4) Consideração da experiência do usuário na solução.

2.3.2 Interoperabilidade Semântica

A interoperabilidade semântica é geralmente associada com a capacidade de compreender contextos específicos. Isto é, a interoperabilidade semântica define regras para a compreensão do significado da informação, criando assim um modelo semântico específico ao domínio.

Quando dois ou mais módulos heterogêneos e distribuídos compartilham informações entre si, deve haver o entendimento correto da mensagem. Muitas vezes a interoperabilidade semântica tem como enfoque os mecanismos de descrição das mensagens. As mensagens devem ser representativas, extensíveis e, sobretudo, não podem conter ambiguidade (DESAI; SHETH; ANANTHARAM, 2015).

No contexto da IoT, o desafio da interoperabilidade semântica está em dar sentido às informações trocadas entre os dispositivos inteligentes. Para tanto, propostas de soluções envolvendo a implementação de serviços de gestão semântica são recorrentes SINGH et al. (2014). Plataformas difundidas, como OpenIoT (OPENIOT, 2017), possuem esta característica.

2.3.3 Interoperabilidade Pragmática

A interoperabilidade pragmática é geralmente associada com a capacidade de identificar a disponibilidade de determinado recurso (seja este de *hardware* ou *software*) para executar operações. Isto é, a interoperabilidade pragmática identifica um serviço (independente de suas especificações) disponível para uso, podendo invocá-lo quando necessário, tornando assim o compartilhamento de serviços e recursos mais funcional (NEIVA et al., 2016). Em um ambiente de IoT, onde os recursos de *hardware* muitas vezes possuem serviços próprios, a aplicação das técnicas que fornecem este nível de interoperabilidade pode ser considerada mandatória.

2.4 Plataformas de IoT

As plataformas para desenvolvimento de soluções de IoT muitas vezes são baseadas em um *middleware* que fornece uma interface entre a camada de *hardware* e a camada de aplicação (BANDYOPADHYAY et al., 2011). Neste contexto, a principal função de um *middleware* é mascarar os problemas de heterogeneidade e distribuição que enfrentamos ao interagir com dispositivos (MULLIGAN; GRAČANIN, 2009). Para BANDYOPADHYAY; SEN (2011), um *middleware* se faz necessário em um ambiente de IoT pois:

- É complexo estabelecer um padrão comum entre todos os diversos dispositivos pertencentes ao domínio da IoT;
- Um *middleware* atua como um intermediário que une os componentes heterogêneos;
- As aplicações de diversos domínios exigem uma camada de abstração/adaptação;
- Um *middleware* deve fornecer uma *Application Programming Interface* (API) para se trabalhar com a comunicação na camada física, bem como os serviços necessários para as aplicações, encapsulando todos os detalhes da diversidade.

Nas próximas subseções iremos apresentar algumas ferramentas que tentam mitigar esses desafios ou parte deles. Inicialmente apresentaremos, de forma sucinta, algumas soluções comerciais e, posteriormente, algumas plataformas *open source*, realçando suas características e requisitos. Sendo estes requisitos considerados fundamentais para plataformas IoT, a citar: Interoperabilidade, Escalabilidade, Extensibilidade, Descoberta, Sensibilidade ao Contexto e Análise de Dados (YAQOOB et al., 2017; KIM et al., 2016; RAZZAQUE et al., 2016; SETHI; SARANGI, 2017; BORGIA, 2014). Estes requisitos são apresentados detalhadamente na Seção 4.2.

2.4.1 Iniciativas Comerciais

O grande potencial esperado para a *Internet* das Coisas tem despertado a atenção de grandes empresas, que vêm expandindo seus serviços, muitos deles baseados em nuvem, para

atender requisitos de projetos de *Internet* das Coisas. Contudo, algumas soluções são fechadas, mesmo sua utilização sendo gratuita ou paga, não sendo possível portanto sua extensibilidade.

Apesar do conceito de plataforma ser divergente na literatura, iremos considerar nesta seção as ferramentas que fornecem algum tipo de subsídio para auxiliar o processo de desenvolvimento ou gerenciamento para IoT. A seguir, é apresentada uma lista simplificada das principais empresas que oferecem serviços para IoT.

2.4.1.1 Apple HomeKit

HomeKit (APPLE, 2015) é um *framework* de integração de dispositivos conectados em residências fornecida pela Apple. Este *framework* permite a descoberta de dispositivos compatíveis na rede, configurá-los, bem como desenvolver aplicativos no ecossistema da Apple.

Através da API é possível executar ações como controlar os dispositivos por comandos de voz, utilizando a assistente Siri. Para o funcionamento adequado do HomeKit no ambiente residencial é necessário configurar o ambiente, indicando rótulos significativos para os equipamentos conectados e criando cenários (tais como: cinema, festa, viagem, entre outros) para gestão modular (APPLE, 2015). A plataforma da Apple ainda é bastante restritiva, limitando-se a poucos acessórios habilitados para funcionar integrados com dispositivos que executam iOS (APPLE, 2016).

2.4.1.2 Amazon IoT

A Amazon, que já possuía diversos PaaS, providos pela *Amazon Web Service* (AWS), disponibilizou em 2015 a plataforma AWS IoT. Esta plataforma permite a comunicação fácil, segura e bidirecional entre as coisas conectadas à *internet* (tais como sensores, atuadores, dispositivos embarcados, ou aparelhos inteligentes) e os serviços da AWS. O AWS IoT pode comportar bilhões de dispositivos e trilhões de mensagens, e pode processar e rotear essas mensagens para os servidores da AWS e para outros dispositivos, de forma confiável e segura (AMAZON, 2016b).

O AWS IoT é compatível com HTTP, *WebSockets* e MQTT, protocolos leves de comunicação especificamente criados para sustentar conexões intermitentes. Também é compatível com outros protocolos personalizados, e os dispositivos podem comunicar-se entre si, mesmo se eles estiverem usando protocolos diferentes.

O AWS IoT disponibiliza um *Software Development Kit* (SDK) para ajudar a conectar de forma simples e rápida o dispositivo de *hardware* ou aplicativo móvel. O AWS IoT Device SDK permite que seus dispositivos estabeleçam conexão, sejam autenticados e troquem mensagens com o AWS IoT. O Device SDK é compatível com C, JavaScript e Arduino, e inclui bibliotecas do cliente, guia do desenvolvedor e guia de portabilidade para fabricantes (AMAZON, 2016c).

A plataforma fornece ainda um mecanismo de gerenciamento de regras para a execução de uma determinada ação (por exemplo, se sensor 1 for igual a *false*, então o atuador 2 recebe

true), onde é possível criar regras dentro do console de gerenciamento ou escrever regras utilizando uma sintaxe do tipo *Structured Query Language* (SQL). As regras podem ser criadas para que se comportem de modo diferente, dependendo do conteúdo da mensagem. A Figura 2.4 ilustra o funcionamento geral da plataforma.

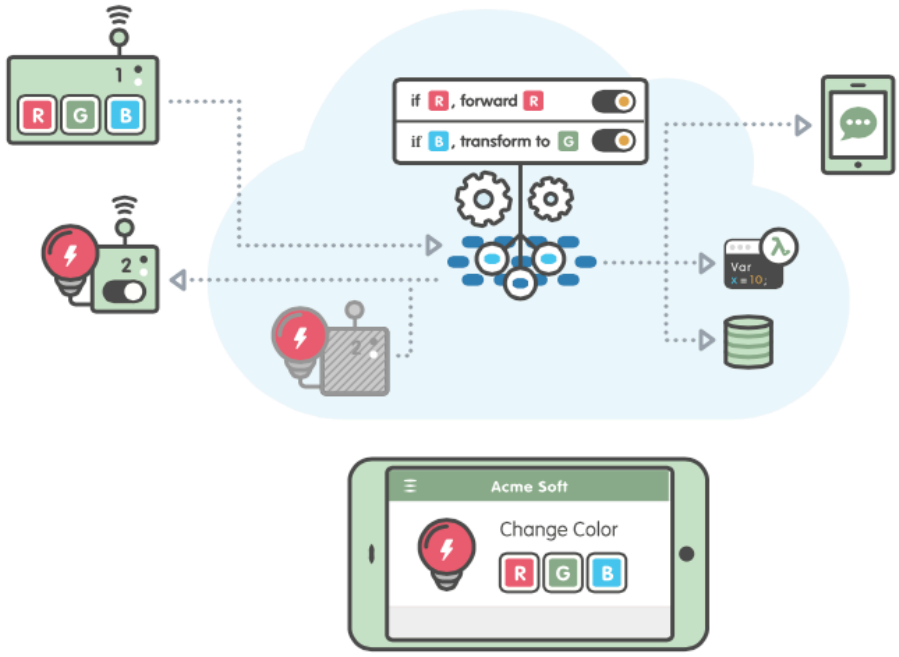


Figura 2.4 Exemplo de funcionamento do *Amazon Web Service IoT* (AMAZON, 2016a).

Os dispositivos localizados no lado externo na nuvem podem interagir com os serviços da AWS IoT através do *gateway* central (concentrador das mensagens, representado pelas engrenagens da Figura 2.4). O componente identificado com o número 1 na Figura 2.4 é a representação de um sensor que envia uma informação para a nuvem; a informação será processada de acordo com um conjunto de regras existentes e posteriormente irá realizar uma ação, que pode ser desde o simples armazenamento na nuvem a um acionamento de um atuador conectado ao serviço (componentes com identificador 2). No exemplo, o *smartphone* representa um aplicativo que interage com a plataforma, tanto enviando quanto recebendo dados.

2.4.1.3 Google Cloud IoT Core

Assim como a Amazon, o Google utilizou sua plataforma de nuvem já existente e estendeu seus serviços para satisfazer as exigências da IoT. No caso do Google, serviços como o Google Cloud Pub/Sub (GOOGLE, 2015) e o Google Cloud IoT Core (anunciado em maio de 2017, focado no público corporativo (GOOGLE, 2017b)) foram disponibilizados e trabalham de forma complementar ao Google Cloud Platform.

Entretanto, a iniciativa que mais se destaca da empresa é o Android Thing (inicialmente, projeto Brillo), apoiada pela Intel, entre outras empresas. O Android Thing promete simplicidade

e rapidez no desenvolvimento de aplicações para IoT, para plataformas embarcadas, além de fornecer um SDK para desenvolvimento baseado na plataforma Android (BRILLO, 2015).

O Android Thing é suportado por plataformas ARM, Intel x86, e *hardware* baseado em *Million Instructions per Second* (MIPS). Deste modo, as plataformas de desenvolvimento e prototipagem Intel Edison (INTEL, 2015a) e Qualcomm Dragonboard 410c (QUALCOMM, 2015) são compatíveis, possibilitando assim a prototipagem rápida dos projetos.

Outra iniciativa que mostra indícios do interesse do Google em IoT foi a aquisição da Nest, empresa fundada por ex-funcionários da Apple, que criou um termostato e um detector de fumaça inteligente (TILLEY, 2014). O termostato da Nest permite o controle da temperatura do ambiente através de um dispositivo móvel (Android ou iOS), mesmo via *internet*. O detector de fumaça, por sua vez, utiliza mensagem de voz para informar a presença de fumaça ou gás danoso no ambiente (NEST, 2016).

2.4.1.4 IBM Watson IoT

Inicialmente a IBM apostou em plataformas em nuvem para conectar dispositivos inteligentes de maneira simples e integrada com a plataforma IBM Bluemix. Atualmente, a IBM tem focado no desenvolvimento da IBM Watson IoT Platform, tendo como enfoque o uso de inteligência computacional aplicada a IoT. A IBM fornece uma série de APIs para facilitar o uso da ferramenta em contextos distintos.

Dentre as APIs disponibilizadas pela IBM, destacam-se as APIs para: conexão (MQTT e HTTP); gestão de informação (tais como informações climáticas); utilização de linguagem natural (por exemplo, síntese e reconhecimento de voz); e processamento de imagens (através de reconhecimento visual com classificadores de cena, ainda em versão Beta) (IBM, 2017).

2.4.1.5 Intel IoT Platform

A Intel está dedicando cada vez mais esforços para tornar a IoT uma realidade, apoiando projetos *Do It Yourself* (DIY) e disponibilizando plataformas que dão subsídio para este desenvolvimento. A plataforma da Intel é composta por um aparato de *hardware* e *software* bem amplo. Além da própria Intel IoT Platform, a Intel ainda disponibiliza placas de desenvolvimento e prototipagem Intel Galileo (INTEL, 2015b) e Intel Edison, bem como o *kit* de desenvolvimento Intel IoT Developer Kit.

A plataforma da Intel fornece uma plataforma para conectar os dados dos dispositivos de ponta a ponta, permitindo inovações no mercado mais rapidamente, reduzindo a complexidade da solução. Os principais requisitos da plataforma são: segurança, interoperabilidade, escalabilidade e gerenciamento (INTEL, 2016).

A Intel também explora as possibilidades das VANETs, desenvolvendo um conceito de utilização da câmera Intel RealSense (REALSENSE, 2016) em um veículo conectado. As funcionalidades destacadas são o reconhecimento de face para destravar as portas do veículo,

o reconhecimento de objetos para evitar colisão com obstáculos, entre outras funcionalidades (INTEL, 2015c).

2.4.1.6 Microsoft IoT

A Microsoft desenvolveu uma versão otimizada do sistema operacional Windows para prover suporte a plataformas de prototipagem de soluções de IoT. O Windows 10 IoT Core é compatível com o *Raspberry Pi* 2 e 3, *Arrow DragonBoard 410c* e *Intel MinnowBoard MAX*. O Windows IoT também funciona com a Arduino, porém não como sistema operacional embutido no *hardware*, como nos demais casos. Os serviços em nuvem, como o Microsoft Azure IoT Suite, e dispositivos móveis, que executem o sistema Windows Phone, podem ser integrados às soluções desenvolvidas sobre o Windows IoT (EDSON, 2014).

2.4.2 Plataformas *Open Source*

Do mesmo modo que a indústria tem se empenhado em trazer soluções para o mercado de IoT a comunidade de desenvolvedores de plataformas *Free Libre and Open Source Software* (FLOSS) também está ativa. Nas próximas seções apresentaremos algumas plataformas disponíveis de forma livre.

2.4.2.1 Lab of Things

Lab of Things[6] (THINGS, 2016) é uma plataforma para pesquisa e desenvolvimento de protótipos experimentais, desenvolvida pela *Microsoft Research*, que permite comunicação entre sensores e dispositivos conectados em um ambiente residencial. A Lab of Things fornece uma série de serviços em nuvem, utilizando os recursos do Microsoft Azure. Para tanto, é necessário um computador no ambiente local conectado à nuvem, permitindo assim o monitoramento, armazenamento e acesso remoto.

Além de permitir a conexão de dispositivos e a implantação de cenários de experimentação de aplicativos, usando os recursos do *HomeOS* (detalhes na Seção 3.5), permite a implantação e o monitoramento de estudos de campo e análise de dados de experimentos, bem como o compartilhamento de dados e códigos (BRUSH et al., 2013).

2.4.2.2 Hobson

Hobson[7] é uma plataforma *open source* que fornece interoperabilidade para dispositivos inteligentes (HOBSON, 2016). O Hobson funciona como um concentrador IoT, intermediando a comunicação entre diversos dispositivos, possuindo uma camada de rede e de aplicação

[6]Lab of Things: Código disponível em: `https://labofthings.codeplex.com`

[7]Hobson: Código disponível em: `https://bintray.com/whizzosoftware/maven/hobson-hub-distribution`

funcionais. O Hobson disponibiliza uma API que pode ser utilizada para gerenciar e controlar os dispositivos conectados ao *hub* do ambiente, permitindo assim que dispositivos externos possam ser incorporados ao ambiente, como um *smartphone*, por exemplo. Uma das vantagens da plataforma está no fato de poder ser executada em dispositivos de baixo custo, como mini-PCs[8] (HOBSON, 2016).

2.4.2.3 LinkSmart

O LinkSmart[9] é uma plataforma para sistemas embarcados com uma infraestrutura de serviços para a desenvolvimento de aplicativos distribuídos para a IoT (PLATFORM, 2016). O LinkSmart, além de tratar da camada de hardware, oferece suporte ao desenvolvimento de aplicações baseadas em informações fornecidas por dispositivos físicos heterogêneos, disponibilizando interface web para controle dos dispositivos (PLATFORM, 2016). O projeto LinkSmart é baseado em uma arquitetura orientada a serviços (ou *Service-Oriented Architecture* (SOA)), onde cada dispositivo gerenciado por este é considerado como sendo um serviço, possuindo ainda uma camada que provê interoperabilidade semântica entre os dispositivos.

2.4.2.4 OpenIoT

OpenIoT[10] é uma plataforma para suportar a criação de aplicações baseadas na Internet das Coisas (OPENIOT, 2016). Os pontos fortes dessa plataforma são o *middleware* para o armazenamento dos dados coletados, suas ferramentas para a definição dos serviços e o fato da plataforma ser um software livre.

O OpenIoT utiliza o modelo baseado em infraestrutura de nuvem. Para tanto, a plataforma provê meios para conectar sensores do ambiente local à nuvem, podendo utilizar a capacidade de processamento, armazenamento e gerenciamento da nuvem para executar tarefas que, eventualmente, o dispositivo por limitações de *hardware* não possa realizar adequadamente.

2.4.3 Comparativo entre plataformas

Essa seção tem como objetivo apresentar uma análise das plataformas citadas anteriormente em relação aos requisitos necessários para IoT considerados neste estudo. A Tabela 2.1 apresenta um comparativo entre as soluções discutidas nas Seções 2.4.1 e 2.4.2.

Na Tabela 2.1 o símbolo ✓ indica que a plataforma contempla o requisito, o símbolo + denota que o requisito é parcialmente atendido pela plataforma, enquanto o símbolo ✗ indica que o requisito não é contemplado ou não foi identificado em suas especificações.

[8] O termo mini-PC neste trabalho é utilizado para se referir aos *Single-Board Computer*s (SBCs), computadores com tamanho reduzido em comparação com um *desktop* tradicional, mas possuindo componentes similares, tais como processador, memória, saída de vídeo, entre outros.

[9] LinkSmart: Código disponível em: `https://www.openhub.net/p/linksmart`

[10] OpenIoT: Código disponível em: `https://github.com/OpenIotOrg/openiot`

Tabela 2.1 Plataformas IoT: Requisitos.

Plataformas	Interoperabilidade	Descoberta	Extensibilidade	Contexto	Escalável	Gestão de Dados
HomeKit	✚	✓	✓	✓	✗	✗
Amazon IoT	✓	✗	✓	✓	✓	✓
Google Cloud IoT	✓	✗	✓	✗	✓	✓
Watson IoT	✚	✗	✓	✗	✓	✓
Intel IoT	✗	✗	✓	✗	✓	✓
Microsoft IoT	✚	✗	✓	✗	✓	✓
Lab of Thing	✓	✚	✓	✓	✗	✓
Hobson	✓	✓	✓	✓	✗	✓
LinkSmart	✓	✓	✓	✓	✗	✓
OpenIoT	✓	✗	✓	✗	✓	✓

Legenda: ✓contempla o requisito; ✚ parcialmente atendido; ✗ não contempla ou não especificado.

Dentre as iniciativas comerciais, foi observada uma estrutura recorrente, onde as soluções apresentadas pela Amazon, Google, IBM, Intel e Microsoft utilizam o conceito de PaaS, fornecendo uma infraestrutura para o envio de dados dos dispositivos (contendo sensores ou atuadores) para a nuvem. Deste modo, estas soluções se assemelham nos requisitos Extensibilidade, Escalabilidade e Gestão de Dados. Entretanto se assemelham também pela ausência de mecanismos capazes de prover os requisitos de Descoberta em Rede e Sensibilidade ao Contexto.

O *framework* HomeKit se diferencia dos demais, justamente por ser executado localmente, podendo assim realizar a descoberta de dispositivos compatíveis na rede e ser sensível ao contexto. Por outro lado, o HomeKit fornece interoperabilidade apenas entre os dispositivos homologados. Quanto aos requisitos Escalabilidade e Gestão de Dados, não foi possível identificar a existência no *framework* da Apple.

Quanto às plataformas *open source*, podemos observar um modelo similar ao comercial. Onde, quando o serviço é centralizado em uma nuvem, os requisitos Descoberta em Rede e Sensibilidade ao Contexto não são atendidos, como é o caso do OpenIoT. Do mesmo modo, quando a plataforma é executada localmente, como é o caso do Lab of Things, Hobson e LinkSmart, torna-se um desafio prover o requisito Escalabilidade.

Por fim, analisando a Tabela 2.1, é possível observar que nenhuma solução estudada foi capaz de contemplar todos os requisitos. Logo, é necessário desenvolver uma arquitetura capaz de suprir todos os requisitos discutidos nesta seção, bem como outros que serão discutidos na Seção 4.2.

2.5 Arquiteturas IoT

As arquiteturas de rede existentes foram projetadas para suportar uma quantidade limitada de dispositivos (YAQOOB et al., 2017). Nesse sentido, faz-se necessário o desenvolvimento de soluções que contemplem as tecnologias emergentes, tais como IoT. Para tanto, é necessário

ter uma arquitetura bem definida para auxiliar no planejamento da solução IoT, levando em consideração todos os requisitos (Seção 4.2).

Embora diversos estudos em IoT (AL-FUQAHA et al. (2015); PERERA et al. (2014); ATZORI; IERA; MORABITO (2010)) abordarem a área em diversas perspectivas, trabalhos que abordem componentes arquitetônicos são escassos. Contudo, são encontrados na literatura alguns esforços direcionados a arquiteturas IoT, identificando as suas características e requisitos necessários.

Dentre as variações de arquiteturas existentes não há um consenso ou uma arquitetura amplamente aceita para IoT (YAQOOB et al., 2017). Porém, é possível encontrar propostas que são mais recorrentes, mesmo com eventuais adaptações de terminologias. Em uma visão geral, podemos destacar duas arquiteturas, uma com três camadas e outra mais específica com cinco camadas.

De acordo com YAQOOB et al. (2017); SETHI; SARANGI (2017); MASHAL et al. (2015); WU et al. (2010), em uma abordagem mais genérica, uma arquitetura IoT deve ser composta por:

- **Camada de Percepção:** apesar de alguns autores divergirem quanto a nomenclatura desta camada, o consenso é que esta camada está diretamente relacionada aos recursos de *hardware*, tendo como função principal a percepção, interpretação e identificação de dispositivos e objetos, bem como realizar a coleta e captura de informações oriundas destes.

- **Camada de Rede:** também denominada de Camada de Transporte, pode ser considerada o centro do gerenciamento da IoT, onde trafegam os dados e as informações. A Camada de Rede deve transmitir e processar as informações obtidas a partir da Camada de Percepção até a Camada de Aplicação e vice-versa. Para tanto, deve identificar cada objeto de uma infraestrutura IoT através de endereços exclusivos além de garantir a qualidade do serviço e a transmissão de forma segura.

- **Camada de Aplicação:** como o próprio nome sugere, a Camada de Aplicação tem como função integrar informações e prover aplicativos ou serviços específicos aos usuários. Nesta camada são definidas as várias aplicações em que a IoT pode ser implantada.

Contudo, a arquitetura em três camadas não é suficiente para representar o rápido desenvolvimento na IoT. A arquitetura em três camadas define apenas a ideia central da IoT e não consegue especificar detalhes dos componentes, que muitas vezes são objetos de estudos e onde se concentram as pesquisa em IoT (SETHI; SARANGI, 2017).

Deste modo, de acordo com SETHI; SARANGI (2017); MASHAL et al. (2015); KHAN et al. (2012); WU et al. (2010), uma solução em IoT deve ser fundamentada em uma arquitetura de cinco camadas, a citar: Percepção, Rede, Processamento, Aplicação e Negócios.

- Nesta arquitetura, as **Camadas de Percepção**, **Rede** e **Aplicação** possuem funções similares às apresentadas na arquitetura de três camadas. Contudo, a proposta desta arquitetura é representar de forma mais específica a arquitetura de IoT, adicionando duas camadas.

- **Camada de Rede:** nesta arquitetura a camada de rede restringe-se à transmissão de dados, contendo uma variação em relação à arquitetura de três camadas. Nesta arquitetura a transmissão dos dados recebidos da Camada de Percepção é direcionada para a Camada de Processamento e vice-versa.

- **Camada de Processamento:** é responsável pelo gerenciamento de serviços implementados por diferentes objetos. Essa camada recebe as informações da Camada de Rede e armazena no banco de dados, realiza processamento de informações e toma decisões com base nos resultados.

- **Camada de Negócios:** gerencia todo o sistema IoT, incluindo aplicativos, utiliza dados recebidos da Camada de Aplicação para criar modelos de negócios. Além disso, deve fornecer privacidade aos usuários.

A Figura 2.5 apresenta uma visão geral das arquiteturas de IoT, com três e cinco camadas.

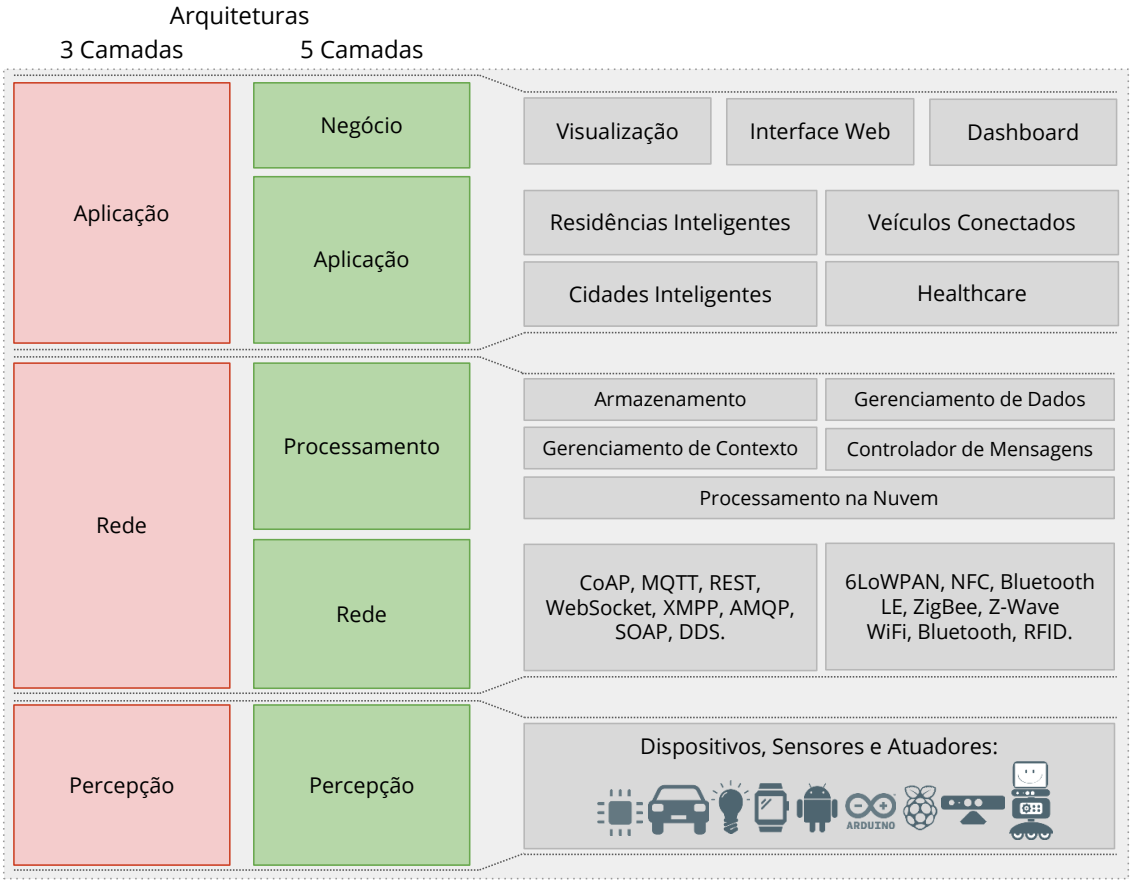


Figura 2.5 Visão geral das arquiteturas IoT.

Na Figura 2.5 é possível observar a relação entre as arquiteturas de três e cinco camadas. As arquiteturas possuem funcionalidades similares, porém a arquitetura com cinco camadas especifica detalhes para as Camadas de Processamento e Negócio. Descreveremos a Figura 2.5 utilizando como referência a arquitetura com cinco camadas, justamente, por sua especificidade.

A Camada de Percepção pode ser composta por dispositivos heterogêneos, entre sensores, atuadores ou dispositivos (que podem conter ambos). Para que os recursos de *hardware* dessa camada possa estabelecer uma comunicação para enviar ou receber dados da Camada de Rede é necessário possuir uma interface de conexão de rede. No exemplo da Figura 2.5 é possível observar alguns objetos conectados, tais como: veículo, lâmpada, relógio, câmeras, robô, entre outros.

Os protocolos apresentados na Seção 2.1.2 (6LoWPAN, NFC, BLE, ZigBee, Z-Wave, entre outros) são normalmente utilizados para a troca de dados entre o *hardware* contido na Camada de Percepção e a Camada de Rede. Os protocolos apresentados na Seção 2.2 (CoAP, MQTT, WebSocket, entre outros) são utilizados normalmente para a comunicação com as Camadas de Processamento ou Aplicação. Dispositivos que possuem implementação de protocolos distintos, devem ser capazes de trocarem informações de modo interoperável.

Na Camada de Processamento, os componentes para o controle de mensagens (por exemplo, um *Broker*), bem como o armazenamento e análise dos dados devem ser considerados durante o desenvolvimento de soluções IoT. Além destes, alguns serviços podem ser implementados, tais como: Gerenciamento de Dados, Gerenciamento de Contexto e Processamento em Nuvem.

A Camada de Aplicação está relacionada diretamente com o domínio da aplicação das soluções IoT, como por exemplo residências inteligentes, cidades inteligentes, veículos conectados, sistemas de saúde, entre outros. A Camada de Negócio fornece mecanismos para apoiar a tomada de decisão com base na análise dos dados, como por exemplo visualização de dados, interfaces de gerenciamento e monitoramento, gráficos, fluxograma, entre outros.

Além das arquiteturas mencionadas, é possível encontrar na literatura trabalhos com abordagens diferentes, como NING; WANG (2011) que propõem uma arquitetura inspirada no processamento do cérebro humano; como TAN; WANG (2010), que além das cinco camadas, propõem uma camada de coordenação sob a Camada de Processamento, para traduzir e formatar as informações em uma estrutura unificada que pode ser processada por todas as aplicações.

Ainda é possível identificar esboços de arquiteturas de referência em IoT, como o proposto pela empresa WSO2 (FREMANTLE, 2014), uma plataforma de código aberto que tem como objetivo prover aos desenvolvedores uma sustentação para conceber arquiteturas de sistemas, contemplando requisitos como interoperabilidade, escalabilidade e segurança. No entanto, não se concentra em detalhar como uma arquitetura deve funcionar, pois não está vinculada a um conjunto específico de tecnologias (FREMANTLE, 2014).

3

RESIDÊNCIAS INTELIGENTES

Este capítulo procura dar entendimento sobre o contexto em que esta tese está inserida, o ambiente residencial inteligente. Para tanto, é apresentada uma contextualização histórica sobre a evolução das tecnologias empregadas neste ambiente, bem como introduzida uma discussão sobre os aspectos que envolvem o uso de tecnologias de informação no ambiente residencial. É apresentado como pode ocorrer a interação entre o habitante e sua própria residência de forma natural.

3.1 Automação Residencial

A automação residencial não é uma tecnologia considerada nova, pelo contrário, alguns pesquisadores, como BOLZANI (2010), indicam que desde a utilização da energia elétrica nas residências a automação residencial já era uma realidade. Contudo, as tecnologias modernas continuam trazendo inovações relevantes para o ambiente domiciliar. Isto indica que os avanços tecnológicos, como os dispositivos inteligentes, ainda possuem um potencial de revolucionar o modo como compreendemos as residências.

Em meados do século XX as previsões de um artigo publicado no jornal *St. Petersburg Times*, assinado pelo então denominado *"news specialist of the associated press"*, no dia 31 de dezembro de 1950, com o título *"How Experts Think We'll Live in 2000"* (THE ASSOCIATED PRESS, 1950), tinha como objetivo vislumbrar as perspectivas para os acontecimentos no mundo do final do século (50 anos depois). Entre os temas discutidos, além de medicina, entretenimento, e outros, o tema "construção" destacava-se, a citar:

- [...] *People will live in houses so automatic that push-buttons will be replaced by fingertip and even voice controls.*
- [...] *Window walls will slip down in slots to merge outdoors with indoors in favorable weather.*
- [...] *All homes will have temperature maintained at constant comfortable levels the year-round.*
- [...] *The telephone will be transformed from wire to radio and will be equipped with the visuality of television.*

— (Fonte: THE ASSOCIATED PRESS (1950))

Estas expectativas para o ano de 2000, apesar da viabilidade técnica, não puderam se tornar realidade, pelo menos não de maneira massiva. Possuir uma residência completamente automatizada, mesmo nos dias atuais, ainda é uma regalia para poucos afortunados ou para os entusiastas. Uma grande dúvida dos pesquisadores é identificar o motivo da lenta adesão de tecnologias mais avançadas no ambiente residencial (BOLZANI, 2004).

Inicialmente, a automação residencial é compreendida como a inserção da tecnologia de automação para auxílio às atividades domésticas (VAZ, 2008). Os equipamentos devem centralizar os controles e processos, tornando tudo mais simples e automático, mas o desejo do usuário nem sempre é considerado, sendo a tarefa o foco da automação.

Uma evolução deste entendimento é a aplicação da automação residencial focada no usuário. Contudo, este enfoque não deve ser entendido apenas como o fato de automatizar processos controlados pelo habitante (por exemplo, ligar uma lâmpada específica, abrir uma persiana, entre outros). A evolução exige que o habitante seja totalmente envolvido no ambiente, sendo este uma fonte de conhecimento da residência. Isto é, a residência e seus componentes devem conhecer de fato os hábitos, sentimentos e desejos do indivíduo, podendo assim agir de modo proativo.

3.2 Ambientes Inteligentes

Segundo DUCATEL et al. (2001) e DUCATEL et al. (2003), o *Environments Intelligence* consiste na utilização da informação existente no ambiente que rodeia um indivíduo, de forma a fornecer ao ser humano um conjunto de serviços ou métodos com o intuito de promover a sua comodidade e seu bem estar, além de dar suporte à realização de tarefas diárias.

É possível utilizar essa informação do ambiente para proporcionar maior conforto, promover o seu bem estar e até contribuir para a sua saúde física e mental. Através da utilização de sensores que monitorem ações de um ser humano é possível efetuar capturas de dados, que depois de processados poderão revelar indicadores da ocorrência de fatores como, por exemplo,

a fadiga ou estresse.

Esta área de conhecimento consiste então na utilização da informação existente no ambiente de forma a facilitar a realização de tarefas ou permitindo a disponibilização de diversos serviços ao ser humano (STIPANICEV; BODROZIC; STULA, 2007). Deste modo trata-se de uma área em que duas das suas características mais relevantes são a sua capacidade de adaptabilidade aos seus utilizadores e a facilidade de interação entre o ser humano e o sistema.

Por fim, os ambientes inteligentes são ambientes onde uma variedade de dispositivos está disponível e conectada em rede, tanto fornecendo informações contextuais relevantes sobre o ambiente para aplicações e usuários, quanto atuando no ambiente, como por exemplo a Residência Inteligente, também conhecida como *Smart House*, *Smart Home* ou *Connected Home*.

3.3 Residência Inteligente

Nesta tese, abordamos Residências Inteligentes e Automação Residencial (ou Domótica) de modos distintos. A Automação Residencial é entendida como o nível mais básico da Residência Inteligente. Isto é, Automação Residencial é considerado como o nível de controle reativo da residência, onde o processo de automação limita-se ao controle do ambiente através de centrais de controle (controle remoto, *smartphone*, *softwares*, entre outros).

A Residência Inteligente é considerada aqui o nível mais inteligente, que consegue entender o contexto do ambiente e dar suporte à decisão, além de conseguir compreender a semântica do ambiente. Uma Residência Inteligente possui a função de prestar um serviço ao habitante, gerenciando o uso eficiente de recursos como energia e água, e realizando tarefas de modo proativo, fundamentado na nuvem de dados obtidos através da rede de sensores.

Através do sensoriamento do ambiente é possível realizar uma decisão de forma racional, isto é, o sistema de gestão da residência inteligente tem a capacidade de selecionar as ações adequadas para elevar ao máximo a probabilidade de alcançar o sucesso almejado na execução da tarefa a qual foi designado. Este fato não significa que devam ser absolutamente reativos, mas sim que eles devem ter a habilidade de combinar as circunstâncias imediatas com suas metas de longa duração, de tal forma que ajustem continuamente suas ações de modo adequado frente às situações (COSTA, 2003).

3.3.1 Tecnologias na Residência

A Residência Inteligente requer a inclusão de novas tecnologias no ambiente residencial. Estas tecnologias estão cada vez mais presentes. TAYLOR (2016) indica que o local onde os dispositivos móveis são mais utilizados é na residência, como ilustrado na Figura 3.1, que apresenta a média de horas diárias de uso do dispositivo por ambiente.

Com média de $2,8$ horas de uso diário de dispositivos móveis, o ambiente residencial

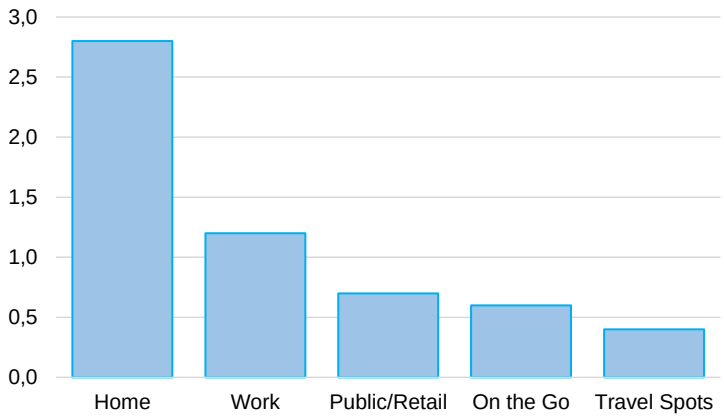


Figura 3.1 Média de horas diárias de uso de dispositivos móveis por localidade (TAYLOR, 2016).

lidera o ranking elaborado pela *Cisco Internet Business Solutions Group* em 2013 (TAYLOR, 2016). O local que mais se aproxima da residência neste quesito é o ambiente de trabalho, contudo, não chega à metade do tempo de utilização média da residência.

É possível que este panorama se amplifique com os dispositivos vestíveis, principalmente, aqueles com enfoque em *mHealth*[1]. Os dispositivos como *smartband* e *smartwatch* podem ser utilizados para acompanhamento de dados de saúde, onde ao indício de que algo pode estar acontecendo ao usuário um alerta pode ser enviado para pessoas autorizadas para prestarem o atendimento.

Quanto à segurança patrimonial, são bastante comuns tecnologias como câmeras IP, fechadura biométrica, portão controlado eletronicamente, entre outras. Cada vez mais estes recursos são utilizados integrados aos demais dispositivos inteligentes e sensores.

3.4 Interação com a Residência

Atualmente, a tecnologia é algo presente na vida de qualquer ser humano, tornando difícil até de distinguir onde está ou não presente o uso da mesma. Isto ocorre pelo fato da tecnologia estar cada vez mais transparente ao usuário, deixando de ser algo complexo de se gerenciar.

É improvável que uma pessoa adulta nunca tenha feito uso de algum tipo de tecnologia em um ambiente residencial. Logo, é importante entender os aspectos relacionados à interação entre o habitante e as tecnologias aplicadas nas residências, independente do nível de complexidade, mesmo que seja algo simples como é o caso de um aparelho de telefone celular ou um televisor, podendo ainda ser algo mais complexo, como fechaduras com leitores biométricos ou robôs.

O estudo da interação com os objetos da residência está em constante evolução. Podemos citar a interação através de comandos de texto, gestos ou voz. Estes estudos têm como intuito

[1] *Mobile Health* (*mHealth*) é uma vertente dos sistemas computacionais de saúde ou, *eHealth* (*Electronic Health*), possuindo como premissa a utilização de dispositivos móveis como ferramentas de diagnóstico, tratamento e acompanhamento das informações de saúde (ADIBI, 2015; SALVI, 2015).

promover a qualidade de vida dos usuários e facilitar a comunicação entre o usuário e os dispositivos contidos na residência.

As tecnologias aplicadas no ambiente residencial inteligente devem estar integradas às diversas atividades domésticas de maneiras diferentes, podendo ser executadas por pessoas distintas que interagem cada uma com sua peculiaridade.

Este novo paradigma oferece aos usuários poder de controlar e gerenciar o ambiente residencial, que por sua vez, reconhece a presença do morador e todo o contexto do seu ambiente, podendo ser adaptável aos hábitos, gestos e emoções dos usuários, além de oferecer a interação simples e intuitiva.

Deste modo, é importante destacar que devido à diversidade de novos sensores e dispositivos disponíveis atualmente, é necessário entender este novo paradigma de interação com a residência e seus elementos. Neste âmbito, deve ser considerada a importância de interações naturais neste contexto, mitigando assim o impacto de novas tecnologias em um ambiente intimista.

3.4.1 Reconhecimento de Gestos

O reconhecimento de gestos é um dos métodos que podem ser utilizados para a construção de uma interface natural de usuário, tornando-se essencial para facilitar a comunicação entre um usuário e um sistema computacional (VALLI, 2005). O estudo do reconhecimento de gestos é um campo absolutamente amplo, possuindo diversas técnicas e classificações distintas. Neste trabalho abordamos o reconhecimento de gestos como sendo uma técnica de visão computacional que realiza a extração de informação de cenas que contenham dados do esqueleto[2] do usuário, de forma parcial ou integral.

A Figura 3.2 ilustra o processo necessário para realizar o reconhecimento de um gesto do usuário, baseado em visão computacional. Onde o corpo do usuário é rastreado por um sensor, no intuito de identificar características (tais como partes do corpo, posição e orientação) do usuário. Para estimar uma pose ou gesto do usuário, após a captura das características do usuário, é necessário passar por uma etapa de classificação, nesta etapa algoritmos de aprendizagem de máquina podem ser utilizados resultando, como saída, na indicação do gesto reconhecido para aquele momento.

Historicamente as técnicas de reconhecimento de gestos exploravam o uso de dispositivos vestíveis (por exemplo, luva de dados) como forma de tornar o rastreamento de partes do corpo possível. Contudo, com os avanços tecnológicos de rastreamento e com o intuito de tornar a interação mais natural, o foco destas pesquisas tem se afastado de tais dispositivos em determinados cenários, sendo preteridos por técnicas de rastreamento de corpo livre, utilizando principalmente sensores RGB-D[3] (FIGUEIREDO, 2017). Trabalhos como BELAGIANNIS;

[2] O esqueleto neste contexto fornece informações de posição das partes do corpo (cabeça, braços, ombros, pernas, tronco e assim por diante) que será mapeado.

[3] Dispositivo capaz de capturar informação de cores e de profundidade de uma determinada cena (LIMA, 2014).

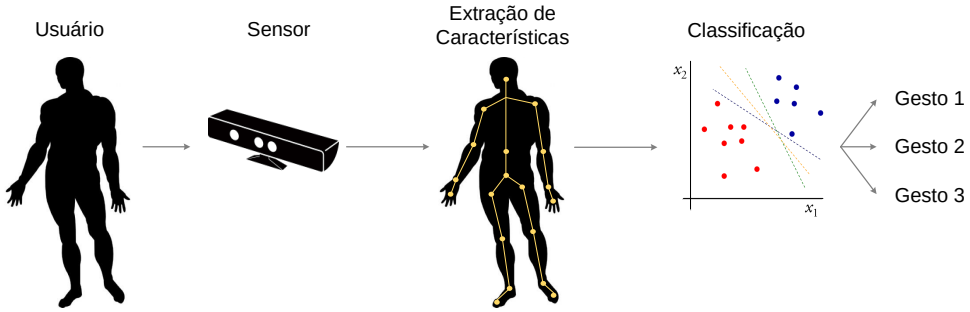


Figura 3.2 Esquemático simplificado de um método de reconhecimento de gestos.

ZISSERMAN (2016); CHEN; RAMANAN (2016); CAO et al. (2016); INSAFUTDINOV et al. (2016) possuem uma abordagem diferente, utilizando ou analisando imagens obtidas através de sensores bidimensionais, como câmeras RGB convencionais. O reconhecimento de gestos, de maneira geral, ainda é um problema complexo a ser tratado, bem como o estudo dos métodos de aplicação dos gestos de modo eficaz na interação com máquinas é um problema não resolvido.

De acordo com BOWMAN et al. (2004), a interação através de gestos é um problema complexo, onde más escolhas podem dificultar a adoção de uma tecnologia específica. A escolha adequada de requisitos torna a experiência do usuário mais natural, reduzindo a curva de aprendizagem, tornando a tecnologia mais transparente e mitigando o estresse do impacto de novos paradigmas de interação. Alguns requisitos desejáveis da interação por reconhecimento de gestos são: (i) *feedback* visual, fornecendo ao usuário uma interface para monitorar e confirmar se o sistema está de fato reconhecendo os gestos corretamente; (ii) pequenas variações incrementais entre quadros, com o intuito de tornar a interação mais suave, sem atrasos na atualização dos quadros; (iii) uso de todo o esqueleto visível, (não apenas o esqueleto parcial, por exemplo, apenas as pernas ou o tronco), de modo que, caso o usuário necessite realizar um gesto utilizando uma parte do corpo não rastreada, o comando idealizado poderá não ser executado; (iv) tempo de resposta baixo para a execução dos comandos, onde o usuário não precise aguardar um determinado tempo para que a ação desejada ocorra, tornando a interação não tão natural.

Um método interessante para fomentar o engajamento do usuário na adoção de tecnologias com interação por gesto é tornar o gesto mais flexível e simples de ser representado. Deste modo, alguns trabalhos, como o de FIGUEIREDO et al. (2014), têm como objetivo tornar o processo de definição de gestos mais otimizado, podendo ser realizado no tempo de execução da aplicação. A proposta é tornar a representação do gesto mais simples, de modo que o usuário não especialista possa descrever um determinado gesto, e assim utilizá-lo de acordo com sua preferência.

A interação por gesto no ambiente residencial já é uma realidade. Ela vem sendo aplicada há algum tempo, principalmente com dispositivos como o sensor *Kinect*[4] *for Xbox*, lançado pela

[4]Sensor RGB-D, composto por uma câmera RGB, um emissor de infravermelho, uma câmera infravermelho, microfones, entre outros. Particularmente útil para o uso em rastreamento do esqueleto do corpo do usuário (CRUZ; LUCIO; VELHO, 2012).

Microsoft em 2010 (MICROSOFT, 2017), que permite ao usuário controlar toda a interface do sistema operacional dos consoles da série *Xbox* através de movimentos corporais. Outro exemplo de objeto doméstico que proporciona a interação através de gestos são os televisores mais modernos; a Samsung (SAMSUNG, 2016) lançou em meados de 2012 seu primeiro televisor com esta funcionalidade. Através do reconhecimento de gestos (realizados com as mãos) é possível controlar o volume, acessar o menu e todos os demais recursos da *Smart TV*; a Figura 3.3 exemplifica este cenário. Desde então diversos outros objetos residenciais foram lançados possuindo esta funcionalidade.

Figura 3.3 Controle de TV baseado em gestos.

Entretanto, em algumas situações o reconhecimento de gestos pode não possuir características mandatórias na comunicação natural. Isto é, em alguns casos é necessário realizar uma série de movimentos não naturais para tentar passar uma mensagem simples, como exemplo: "ligue aquele abajur". Assim como ocorre na interação entre humanos, nada mais simples que utilizar a fala atrelada ao gesto (apontar para o objeto) para transmitir esta mensagem com mais eficiência. Portanto, a interação por voz também é essencial para a comunicação entre o usuário e o sistema residencial.

3.4.2 Síntese e Reconhecimento de Voz

Em uma comunicação natural, a fala é um dos principais meios de interação entre humanos, onde é possível transmitir uma mensagem de maneira direta e, dependendo da situação, ainda é possível dar sentido para a mesma mensagem, através da entonação da voz. Logo, a comunicação por voz possibilita uma interação eficiente e proporciona uma gama de aplicações bem diversificada.

Os elementos que compõem uma comunicação através da fala são: emissor, mensagem, canal de comunicação e receptor. Deste modo, para realizar uma interação por voz, tanto entre humano-humano, como humano-máquina e máquina-humano, é necessário identificar cada elemento. Por exemplo, em uma comunicação entre humano-computador, onde o objetivo é

informar que *"a janela está aberta"*, podemos dizer que:

- Emissor: Humano
- Mensagem: "a janela está aberta"
- Canal de comunicação: Sistema de reconhecimento de voz
- Receptor: Máquina

Um exemplo desta interação no ambiente residencial está no uso de robôs como assistentes domésticos. Este mercado está crescendo, e atualmente existem robôs como o Aido (AIDO, 2016), Musio (AKA-INTELLIGENCE, 2016), Jibo (JIBO, 2016) e o Pepper (ALDEBARAN, 2016), que podem interagir com os habitantes para realizar tarefas. O Pepper, por exemplo, é um humanoide criado para se comunicar com o usuário da forma mais natural e intuitiva possível, principalmente, através de gestos e voz. Por meio da interação o robô aprende mais sobre as preferências, os hábitos, a personalidade e as ações dos indivíduos de uma residência. A Figura 3.4 apresenta os robôs citados neste parágrafo.

Figura 3.4 Exemplos de robôs domésticos.

A interação por voz em sistemas computacionais é classificada de duas maneiras distintas (MCTEAR, 2002). Uma é a síntese de voz, que tem a função de converter os sinais digitais para uma frequência de voz. Deste modo é possível, por exemplo, converter um conjunto de caracteres para uma fala, conhecido na literatura por *Text-to-Speech* (TTS). O outro caso é o reconhecimento de voz, que tem a função inversa na comunicação, isto é, converte a frequência de voz para sinais digitais.

3.5 Trabalhos Relacionados

Esta seção apresenta uma análise dos principais trabalhos relacionados com a presente tese. Os trabalhos filtrados abordam os seguintes temas centrais: "aplicação de *Internet* das Coisas em ambientes residenciais" e "mecanismos de interoperabilidade entre dispositivos com ênfase em Residências Inteligente". O fluxo de trabalho utilizado para o desenvolvimento desta pesquisa pode ser observado na Figura 3.5.

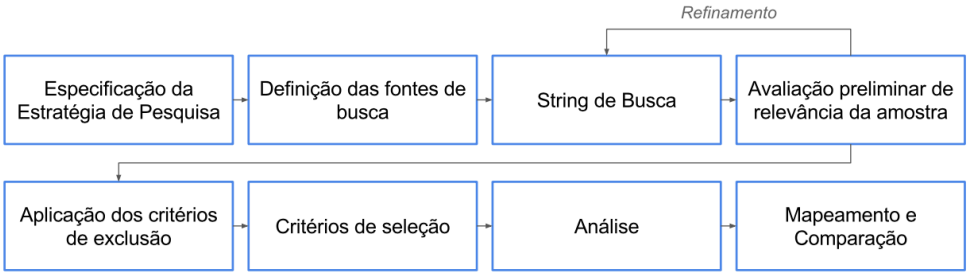


Figura 3.5 Metodologia utilizada no processo de revisão da literatura.

A especificação da estratégia de pesquisa foi a etapa inicial, onde foram definidas a questão de pesquisa e a seleção das palavras-chave. O levantamento dos dados foi realizado em dezembro de 2015, em quatro bases de dados eletrônicas indexadas: *ACM Digital Library*, *IEEE Xplore*, *ScienceDirect*, *SpringerLink*. Também foram realizadas buscas adicionais no motor de busca eletrônica *Google Scholar*. Para tanto, foram ensaiadas algumas *strings* de busca, as quais foram refinadas até a definição da mais representativa para o escopo deste trabalho. A *string* de busca resultante é representada pelo Código 3.1:

Código 3.1 String de busca.

```
"Internet of Things" AND
("Interoperable" OR "Interoperability") AND
("Platform" OR "Operating System" OR "Middleware") AND
("Smart Home" OR "Home Automation" OR
"Domotic" OR "Residential System")
```

Sobre o conjunto de trabalhos obtido foi aplicado um critério de exclusão para restringir a amostra a um grupo de trabalhos mais relevantes. Os critérios de exclusão adotados foram: (i) artigos publicados antes de 2011; (ii) trabalhos não revisados pela comunidade acadêmica; (iii) não escrito no idioma inglês; (iv) propostas com objetivos e contribuições similares menos recentes; (v) versões duplicadas.

Após aplicação dos critérios de exclusão, restaram 352 documentos científicos. Na fase de seleção, concluímos que 53 trabalhos deveriam passar por uma análise mais detalhada para compor o conjunto de trabalhos diretamente relacionados. Dentre estes documentos, foram selecionados 10, que podem contribuir para a presente pesquisa. Estes serão descritos na Seção 3.5.1.

3.5.1 Análise dos Trabalhos

Os trabalhos apresentados nesta seção possuem como característica o enfoque em soluções diretas ou indiretas para o ambiente residencial. Eles abrangem desde soluções mais específicas de modo horizontal, como o tratamento da abstração e da comunicação de dispositivos inteligentes, até plataformas mais complexas que abordam o problema de forma vertical, que auxiliam o desenvolvimento de aplicações desde a coleta do dado até sua apresentação ao usuário.

Alguns trabalhos possuem como objetivo desenvolver ferramental para integrar a gestão dos sensores e atuadores com serviços de Computação em Nuvem, como em SOLIMAN et al. (2013). Neste trabalho, os autores apresentam um sistema de IoT com o intuito de incorporar inteligência em objetos utilizando a plataforma Arduino, aplicando a tecnologia ZigBee para a comunicação com os serviços em nuvem implementados na plataforma Google App Engine, e utilizando documentos no formato JSON.

O trabalho de ZHOU et al. (2013) possui objetivo semelhante ao anterior. Denominada de *CloudThings*, a plataforma proposta busca disponibilizar o acesso aos dispositivos residenciais através de serviços de nuvem. Para tanto, foi utilizado o protocolo CoAP para comunicação na camada de *hardware*, na camada de rede foi utilizado o protocolo IPv6 e na camada mais alta foi implementado um serviço *Web RESTful* na plataforma Google App Engine. Para verificação da arquitetura, os autores realizaram estudos de caso que coletavam dados de temperatura (LM35[5]) e luminosidade (*Light Dependent Resistor* (LDR)[6]) publicando-os na nuvem. Entretanto, a plataforma *CloudThings* não trata do problema da interoperabilidade entre os dispositivos.

Já em KIM et al. (2015) o problema da interoperabilidade é considerado e tratado utilizando protocolos convencionais da *web*. Neste estudo é proposta uma ferramenta para gerenciamento, que se fundamenta em mapa de conhecimento que é o resultado da análise dos dados oriundos de diversas fontes, possuindo ainda uma API para especificação de métodos de gerenciamento. Contudo, esta solução não fornece mecanismos de descoberta ou incremento de dispositivos, não permite a interação com eventos externos, como outros serviços em nuvem e, principalmente, não considera a interação do habitante no ambiente.

Em VALLATI et al. (2015) foi apresentado o BETaaS, uma arquitetura horizontal para M2M, baseado em *Things as a Service* (TaaS), que possui uma infraestrutura de serviços distribuídos sobre uma camada de interoperabilidade. As principais funcionalidades são: sensibilidade ao contexto, qualidade de serviço, segurança, gerenciamento de uma grande massa de dados e virtualização. A API fornecida pelo trabalho oferece suporte a um conjunto restrito de operações, a citar: instalação, chamada de serviço, registro e notificação. As informações são estruturadas através de um documento XML, contendo especificações sobre o serviço. Com isso, é possível

[5]Sensor de temperatura de circuito integrado com uma tensão de saída linearmente proporcional à temperatura Centígrados (NATIONAL, 2015).

[6]Resistor que varia os valores conforme a intensidade da luz que incide sobre este (SHARMA; GROVER; BANDE, 2009).

inferir novos conhecimentos a partir de dados brutos de forma sensível ao contexto.

Outro trabalho que tem foco na convergência em IoT foi apresentado por YU et al. (2015b). Neste trabalho foi utilizada uma abordagem adaptativa, implementada por redes inteligentes através da integração de diversas tecnologias, sem a intervenção do usuário. Deste modo, a descoberta de novos dispositivos é realizada de forma transparente ao usuário, permitindo ainda a interoperabilidade no ambiente.

Em LEE et al. (2015) foi proposta uma plataforma de convergência para residências inteligentes. A composição funcional da plataforma é dividida em 3 camadas principais: abstração de rede; gerenciamento e *Broker*. Para a troca de mensagens na camada de rede foi implementado o protocolo CoAP, além do próprio HTTP. Como componente da arquitetura é fornecida uma API de baixo nível, o que dificulta a implementação de novos serviços por usuários finais que não possuam expertise de programação. A arquitetura também trata da interoperabilidade entre os dispositivos e permite a inclusão de serviços externos. Entretanto, a arquitetura não pode ser escalável, não considera dados de contextos e não fornece suporte à interação natural do usuário.

Algumas pesquisas investem em arquiteturas baseadas em agentes, como em HERNANDEZ; REIFF-MARGANIEC (2015), que se fundamenta em uma estrutura de agentes e serviços para autocontrole independentes, como parte de um processo de tomada de decisão. Deste modo, espera-se que os agentes forneçam inteligência para escolher quais tarefas e serviços devem ser atendidos, através de regras pré-definidas por cada função. Entretanto, destaca-se como limitação do trabalho a ausência de mecanismos para dar suporte à heterogeneidade de dispositivos existentes e, principalmente, o não fornecimento de API para desenvolvimento de novos agentes ou módulos específicos.

O fornecimento de API é importante para facilitar a adoção e evolução de algumas ferramentas, como em STAVROPOULOS et al. (2013) e DIXON et al. (2012). Em STAVROPOULOS et al. (2013) foi proposto um *middleware*, denominado de *aWESoME*, para integrar sensores heterogêneos e demais dispositivos encontrados no mercado e fornecer uma camada de abstração para aplicações. Na camada de aplicação o objetivo é utilizar um cliente nas plataformas *web*, *Desktop* e *Mobile*. Um estudo de caso (monitoramento energético) foi realizado em uma universidade, sendo instalados alguns servidores espalhados pelo prédio, mitigando assim as restrições de cobertura de rede. O *aWESoME* é escalável, permite a incorporação de eventos externos ao ambiente e disponibiliza uma API apenas para gestão, não dando suporte ao desenvolvimento de novas instâncias.

HomeOS (DIXON et al., 2012) é uma proposta de sistema operacional para residências, sendo desenvolvida sob 3 perspectivas: gestão para não-especialistas; desenvolvimento de aplicações; e crescimento incremental. O *HomeOS* utiliza uma arquitetura centralizada, com (i) controle de acesso baseado em registro de dados e outras primitivas que simplificam a tarefa de gerir a tecnologia em residências, (ii) serviços independentes de protocolos que fornecem aos desenvolvedores abstrações simples para acessar dispositivos e (iii) fácil incorporação de novos dispositivos e aplicações. HomeOS é executado em um computador dedicado na residência (por

exemplo, o *gateway*). Como método de avaliação do SO foram desenvolvidas 12 aplicações, com destaque para uma que faz uso do sensor Kinect para controlar dispositivos (o estado de uma lâmpada, por exemplo) através de gestos. Contudo, o protótipo limita-se a poucos protocolos (Z-Wave e *Digital Living Network Alliance* (DLNA)) e dispositivos. Com isto, não representa um avanço considerável do ponto de vista da interoperabilidade.

Em SINGH et al. (2014) foi proposto um *middleware* fundamentado em (i) dar suporte à comunicação heterogênea, (ii) facilitar a descoberta e incorporação de dispositivos (sensores, atuadores, objetos), além de (iii) possuir descritores semânticos construídos sobre modelos ontológicos que, segundo os autores, podem ser utilizados para deduzir situações temporais no ambiente doméstico. Porém, nenhum módulo para análise temporal de dados foi implementado. Além disto, esta arquitetura vertical tem enfoque apenas na camada de comunicação, não contemplando outras camadas desejáveis em uma plataforma. SINGH et al. (2014) indica a necessidade da implantação do protótipo em configurações reais para a coleta de dados no intuito de aprender e analisar os padrões de atividades humanas no dia a dia.

Por fim, realizamos um comparativo entre os trabalhos relacionados apresentados aqui e a presente tese, relacionando as características e abordagens utilizadas nestes trabalhos. A Tabela 3.1 apresenta este comparativo, onde na coluna "API" classificamos os trabalhos que disponibilizam algum tipo de API. KIM et al. (2015) e DIXON et al. (2012) disponibilizam uma API apenas para gerenciamento dos recursos, não sendo possível a extensão da ferramenta. Por outro lado, VALLATI et al. (2015), YU et al. (2015b) e SINGH et al. (2014) disponibilizam uma API de alto-nível, através de linguagem de marcações, tais como JSON e XML. LEE et al. (2015) apresentam uma API como contribuição, contudo, a sua utilização requer um conhecimento prévio em desenvolvimento de *software*, sendo, portanto, classificada como uma API de baixo-nível.

Quanto às demais características, a coluna "Interoperabilidade" refere-se à existência de mecanismos que fornecem algum nível de interoperabilidade entre os componentes de *hardware*; a coluna "Extensibilidade" está relacionada à possibilidade de utilização de serviços externos para complementar ou incorporar aos recursos nativos; a coluna "Descoberta" contempla os trabalhos que utilizam protocolos ou métodos capazes de identificar automaticamente a inclusão de novos *hardwares*; a coluna "Escalabilidade" indica se o trabalho provê suporte à escalabilidade da infraestrutura inicial; a coluna "Gerenciamento do Contexto" refere-se à existência de mecanismos de decisão fundamentada nos dados obtidos do contexto da aplicação, não apenas dados dos dispositivos de forma isolada; a coluna "Interação Natural" considera principalmente se o estudo analisado possui alguma ferramenta que trata ou provê suporte à interação natural (por exemplo: interação por gesto ou voz, bem como o uso de metáforas de interação); a coluna "Aprendizado" relaciona os trabalhos que consideram o aprendizado conforme as ações do habitante como método para antever situações ou mesmo recomendar tarefas, baseado no conhecimento adquirido.

Tabela 3.1 Comparação dos trabalhos relacionados com a presente tese de doutorado.

Autores	API	Interoperabilidade	Extensibilidade	Descoberta	Escalabilidade	Gerenciamento do Contexto	Interação Natural	Aprendizado
SOLIMAN et al. (2013)	-	-	✓	-	-	-	-	-
ZHOU et al. (2013)	-	-	✓	-	-	-	-	-
KIM et al. (2015)	Gerenciamento	✓	-	-	-	-	-	-
VALLATI et al. (2015)	Alto-Nível	✓	-	-	✓	-	-	-
YU et al. (2015b)	Alto-Nível	✓	-	✓	✓	-	-	-
LEE et al. (2015)	Baixo-Nível	✓	✓	✓	-	-	-	-
HERNANDEZ; REIFF-MARGANIEC (2015)	-	-	✓	✓	-	✓	-	-
STAVROPOULOS et al. (2013)	-	✓	✓	-	✓	-	-	-
DIXON et al. (2012)	Gerenciamento	✓	-	✓	-	-	✓	-
SINGH et al. (2014)	Alto-Nível	✓	-	✓	-	✓	-	✓
Este trabalho	**Alto-Nível**	✓	✓	✓	✓	✓	✓	✓

No contexto desta pesquisa, o ambiente residencial deve possuir um bom sensoriamento capaz de representar os diferentes fenômenos que ocorrem no ambiente. Por este motivo, além dos trabalhos já discutidos nesta seção, outros trabalhos devem ser destacados.

Em LAPUT; ZHANG; HARRISON (2017) é apresentado o que os autores denominaram de "sensor sintético", onde um pequeno *hardware* contendo diversos sensores embutidos consegue mapear uma gama de atividades que estão ocorrendo no ambiente. Para tanto, é necessário um período prévio para coletar dados do ambiente para serem analisados por um algoritmo de aprendizagem de máquina (no caso, um *Support Vector Machine* (SVM)). Ao final do processo, o que se tem é uma fusão de sensores que quando combinados fornecem informações mais precisas. Um das principais contribuições deste trabalho é que a partir deste sensor sintético é possível tornar qualquer ambiente comum em um ambiente inteligente, através apenas de um pequeno dispositivo não-invasivo.

Seguindo a linha de mapeamento do ambiente, CAO et al. (2016) apresentam um algoritmo *open source*[7] de visão computacional, capaz de detectar diversas pessoas simultaneamente utilizando apenas uma câmera convencional (2D). Por não necessitar de profundidade e, sensores de profundidade, este trabalho apresenta-se como uma alternativa para as aplicações no ambiente residencial, onde eventualmente uma série de pessoas podem estar interagindo no mesmo local e, em alguns casos, este local pode ser estreito e limitado.

Quanto à análise comportamental, podemos destacar o estudo de LI; LU; MCDONALD-MAIER (2015), que identifica trabalhos com enfoque no comportamento do habitante em um ambiente residencial assistido. Dentre os trabalhos ressaltados, destacam-se COOK et al. (2003) e DOCTOR; HAGRAS; CALLAGHAN (2005), por possuírem abordagens similares às aplicadas nesta pesquisa.

Em COOK et al. (2003) foi apresentado o MavHome, um sistema para residências inteligentes com o objetivo de gravar e aprender com o comportamento do usuário, onde como cenário representativo da aprendizagem, podemos citar que o sistema aprende que a residência demora 15 minutos para aquecer e chegar a temperatura ideal, e logo, se o habitante agendou um alarme para acordar às 7:00 horas da manhã, às 6:45 o sistema dispara o comando para aquecer o ambiente e às 7:00 soa o alarme, liga a luz do quarto, liga o máquina de café, entre outros.

Em DOCTOR; HAGRAS; CALLAGHAN (2005) foi apresentado o iDorm (atualizado para iSpace, posteriormente), ambiente para experimentos sobre ambientes inteligentes, composto por um mobiliário residencial e diversos sensores para mapear o ambiente. Contudo, o que se destaca é o componente responsável por receber os dados dos sensores e aprender com base no comportamento do ambiente para tomar decisões de forma autônoma. A aprendizagem é incremental, onde com o passar do tempo, mais informações são recebidas e o aprendizado é ajustado.

Tanto em COOK et al. (2003) quanto em DOCTOR; HAGRAS; CALLAGHAN (2005), e assim como nesta tese, o termo comportamento é aplicado no sentido de funcionamento do

[7]Disponível em: `https://github.com/ZheC/Realtime_Multi-Person_Pose_Estimation`

ambiente, onde cada tarefa do habitante (por exemplo, acender uma lâmpada ou esquentar um café) é registrada e considerada como comportamento do habitante, bem como as tarefas involuntárias, que não dependem da atividade do habitante para o funcionamento, tais como: a temperatura do ambiente diminuir no final do dia. Em ambos os casos, não foi considerado o sentido de personificação ou a relação do comportamento cognitivo no âmbito das emoções ou personalidades dos indivíduos.

4

ARQUITETURA PROPOSTA

Este capítulo apresenta a proposta de arquitetura de um sistema operacional para residências inteligentes. Inicialmente, delineamos os requisitos e arquitetura geral do SO, abrangendo todos os componentes necessários para o seu funcionamento. Após a visão geral da arquitetura, apresentamos os mecanismos essenciais para tornar o ambiente, gerenciado pelo SO, inteligente, sendo sensível ao contexto e aprendendo de modo incremental a partir dos dados gerados pelo habitante.

4.1 Sistema Operacional para Residências Inteligentes

Considerando que o ambiente residencial está em constante evolução, principalmente pelos recentes avanços proporcionados pelos conceitos da Internet das Coisas, a utilização de mecanismos para gerenciar este local é um cenário bastante factível. Fundamentando-se nesta premissa, propomos uma arquitetura de sistema operacional para residências que tem como objetivo principal prover uma solução para o desenvolvimento de aplicações baseadas em análise contextual com suporte à interação natural do usuário.

A implementação de um SO para residências contribui para o fomento de inovações em residências inteligentes suportado pela existência de objetos conectados. Um sistema operacional para residências subsidia o desenvolvimento de soluções mais específicas para cada contexto, tendo como premissa as preferências e necessidades de diversos tipos de usuários. Deste modo, é possível criar um cenário mais adequado para pessoas que preferem um ambiente residencial com mais opções de entretenimento, ou pessoas que prezem mais a segurança, por exemplo.

Umas das principais características desejáveis de um sistema deste tipo é prover mecanismos que possam ser utilizados para tornar o ambiente residencial mais otimizado, possuindo formas de interação naturais que facilitem a adoção da tecnologia e ainda consigam compreender o contexto, podendo assim agir de modo proativo para auxiliar o habitante na tomada de decisão e gerenciamento da residência.

Outro aspecto interessante a se realçar é que em um ambiente residencial podem existir diversos objetos conectados e estes podem não conseguir estabelecer uma comunicação entre si,

deste modo, um SO deve, particularmente, suprir esta demanda.

O sistema operacional para residências inteligentes irá prover soluções para o desenvolvimento de aplicações que se adaptam as novas regras de automação do ambiente de acordo com as ações dos habitantes, ou seja, os sistemas aprendem e se adaptam ao cotidiano dos habitantes que podem mudar com o passar do tempo.

O SO terá como característica prover interação natural com os habitantes da residência; identificação do ambiente; reconhecimento de face e voz; integrar todos os dispositivos eletrônicos e tecnológicos do ambiente.

4.2 Requisitos da Arquitetura

Nós propusemos uma arquitetura de SO para IoT no domínio de aplicação das residências inteligentes. Nesta seção iremos descrever os requisitos que nossa arquitetura deve prover. Para tanto, nós realizamos um levantamento em relação às arquiteturas existentes e identificamos os requisitos mandatórios. De acordo com o que YAQOOB et al. (2017); KIM et al. (2016); RAZZAQUE et al. (2016); SETHI; SARANGI (2017); BORGIA (2014) sugerem, podemos indicar como requisitos fundamentais para uma arquitetura IoT:

- **Interoperabilidade:** Deve permitir a compatibilidade a uma variedade de dispositivos, tecnologias e serviços, dando suporte a colaboração e troca de informações entre dispositivos heterogêneos. Fornecer interoperabilidade entre os dispositivos de fornecedores concorrentes é um requisito crucial para uma arquitetura de IoT. Este recurso permite que um sistema de IoT seja mais abrangente, sem a necessidade de um esforço adicional dos desenvolvedores.

- **Escalabilidade:** Uma arquitetura de IoT deve ser capaz de evitar que um crescente número de dispositivos afete o provisionamento dos serviços. Este requisito torna-se ainda mais essencial se as tendências e estimativas da crescente quantidade de dispositivos conectados se concretize. Neste caso, a arquitetura deve conseguir absorver adequadamente os novos recursos sem degradação de desempenho.

- **Extensibilidade:** A possibilidade de estender os recursos de uma determinada plataforma faz com que o usuário possua mais flexibilidade, ainda mais quando há demanda de adaptabilidade e evolução no nível de aplicação. Deste modo, fornecer subsídios para adicionar novos dispositivos, novas funcionalidades, novas interfaces, entre outros, pode ser um diferencial na adoção de uma determinada arquitetura, principalmente, quando o objetivo é desenvolver um sistema operacional sobre esta arquitetura, como é o caso deste trabalho.

- **Descoberta:** Em um ambiente dinâmico como a IoT, poder gerenciar todos os dispositivos e seus respectivos serviços é mandatório. O requisito Descoberta fornece

mecanismos que auxiliam no conhecimento sobre os recursos disponíveis em uma infraestrutura. Com isso, uma aplicação que seja executada sobre esta arquitetura possuirá conhecimento atualizado do estado de todos os recursos e serviços.

- **Sensibilidade ao Contexto:** Possuir informações do contexto é um requisito importante para uma boa arquitetura de IoT, pois torna possível o estabelecimento de relações entre os diversos envolvidos, sejam estes sensores, atuadores, sistemas, usuários, entre outros. Poder tomar decisões baseadas no contexto torna a solução mais adaptativa, podendo gerenciar com mais eficiência um determinado setor da residência com base na localização dos habitantes, por exemplo.

- **Análise de Dados:** Em um ambiente IoT os sensores coletam uma quantidade enorme de dados; não só os dados do momento, mas o histórico dos dados podem ser armazenados. Logo, uma arquitetura deve prover componentes capazes de analisar e extrair informações dentro do universo de dados gerados. Visto que o ambiente fornece uma quantidade grande de dados e que, muitas vezes, estes dados, por si só, não representam um conhecimento estruturado, o uso de técnicas de aprendizagem de máquina deve ser considerado na construção de uma arquitetura.

Além dos requisitos supracitados, nós incorporamos na arquitetura proposta um requisito pouco convencional nesta emergente área da IoT, principalmente, no ambiente residencial, que foi chamado de Interação Natural.

- **Interação Natural:** A arquitetura deve fornecer componentes para que as aplicações possam fazer uso de interações naturais entre os usuários e o ambiente no qual estão inseridos. Com a tendência de crescimento de objetos conectados em uma residência, onde cada recurso possui um controle próprio ou interface distinta, prover mecanismos que tornem a interação com estes novos recursos de mais alto nível possível, apresenta-se como um requisito desejável.

Um dos pré-requisitos das arquiteturas para a IoT é prover camadas de Segurança e *Quality of Service* (QoS). Contudo, no escopo deste trabalho não iremos abordar tal aspecto devido ao tamanho do escopo que poderia inviabilizar o estudo. De toda forma, mecanismos de segurança e QoS estão implícitos nas tecnologias que são empregadas para o desenvolvimento de soluções IoT, tais como protocolos.

4.3 Arquitetura do Sistema Operacional

Fundamentando-se nas arquiteturas discutidas na Seção 2.5 e nos requisitos apresentados na Seção 4.2, nós propomos uma arquitetura para o desenvolvimento de um SO para IoT no âmbito residencial. A proposta de arquitetura é composta por 8 camadas, distribuídas da

seguinte forma: *Hardware*, Abstração de *Hardware*, Rede, Decisão, Análise, Interação, Serviço e Aplicação. Esta arquitetura propicia uma integração entre todas as camadas de modo que elas sejam fortemente dependentes, ocorrendo a comunicação nos dois sentidos (da camada superior para a camada inferior e vice-versa).

A Figura 4.1 representa a organização das camadas e seus respectivos componentes. Estas serão explicadas em detalhe nas próximas subseções.

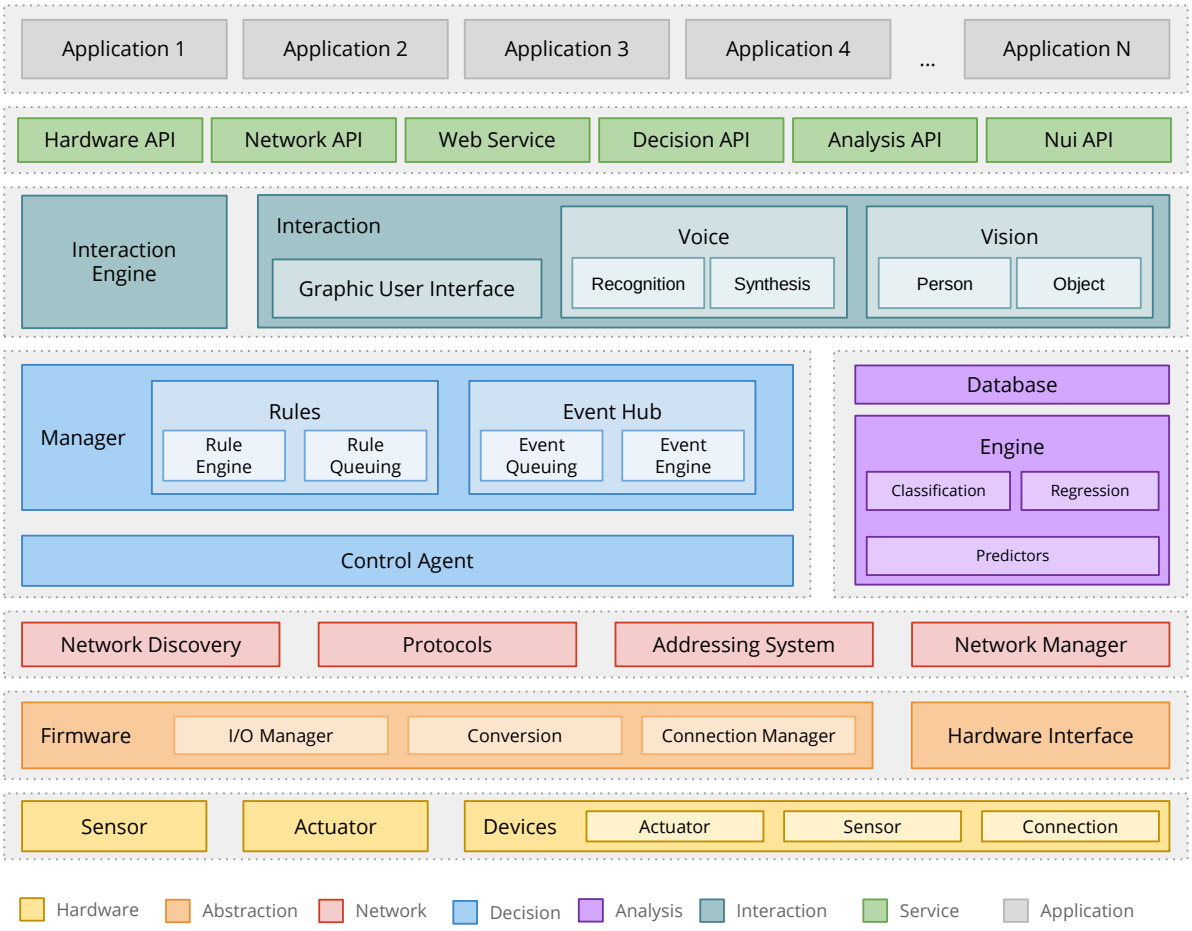


Figura 4.1 Arquitetura do SO para residências proposta.

4.3.1 Camada de *Hardware*

Devido à heterogeneidade dos dispositivos físicos que podem constituir um ambiente residencial inteligente, é necessário criar mecanismos que possam tratar as informações advindas destes. A Camada de *Hardware* da arquitetura proposta está relacionada ao *hardware* utilizado para coletar dados e/ou modificar o ambiente.

A Figura 4.2 realça os componentes existentes na Camada de *Hardware*.

Consideramos a existência de três tipos de *hardware* em nosso cenário de estudo: o sensor, o atuador e o dispositivo inteligente genérico:

Figura 4.2 Camada de *Hardware*.

- **Sensor** *(Sensor)* - O sensor é qualquer objeto capaz de detectar fenômenos ou mudanças no ambiente físico, tais como temperatura, umidade do ar, intensidade de luz, corrente elétrica, fluxo de água, voz do usuário, postura e movimentação corporal do usuário, entre tantos outros. A partir disso o sensor fornece uma saída correspondente.

- **Atuador** *(Actuator)* - O atuador é qualquer mecanismo capaz de alterar o estado do ambiente físico, tais como acionar uma carga, enviar comandos remotos, sintetizar um som, projetar uma imagem, acionar uma válvula de contenção, entre outros.

- **Dispositivo inteligente** *(Devices)* - É considerado como dispositivo qualquer componente que possua no mesmo *hardware* mecanismos de sensoriamento e de atuação. Este tipo de *hardware* foi considerado pois é possível a aquisição de soluções comerciais que muitas vezes são soluções fechadas, mas em geral, utilizam algum padrão de comunicação que pode ser compreendido pelo sistema operacional para ser integrado aos demais recursos da residência inteligente.

Deste modo, é possível tanto utilizar mecanismos de uso exclusivo para sensoriar ou apenas para atuar e mecanismos que podem tanto atuar como sensoriar. Entretanto, é importante evitar que o *hardware* sofra dependência do sistema, ou seja, caso o sistema operacional venha a ficar inoperante por qualquer motivo, o *hardware* independente continue executando suas tarefas normalmente. Por exemplo, se o usuário adquiriu uma lâmpada *wireless* que pode ser controlada por um *smartphone*, esta lâmpada poderá ser incorporada ao SO, e caso o SO venha a falhar o uso inicial deste deverá ser mantido, ou seja, ainda poderá ser gerenciado pelo *smartphone*.

4.3.2 Camada de Abstração de *Hardware*

A Camada de Abstração de *Hardware* é intermediária entre a Camada de *Hardware* e a Camada de Rede. Esta camada possui uma importância significativa para o funcionamento adequado da arquitetura, pois é nesta que ocorre o gerenciamento do *hardware*.

A Figura 4.3 realça os componentes existentes na Camada de Abstração de *Hardware*.

Figura 4.3 Camada de Abstração de *Hardware*.

Esta camada é composta por elementos de *hardware* e *software*, sendo responsável principalmente por (i) capturar corretamente os dados dos sensores e (ii) enviar os dados para os atuadores. Para tanto, consideramos que possam existir *hardware* específicos que têm a função de permitir a troca de dados entre a camada anterior e posterior a esta. Os microcontroladores [1] (por exemplo, Arduino, *Peripheral Interface Controller* (PIC), Intel Edison) e os mini-PCs (por exemplo, *Raspberry Pi*, *BeagleBone*) são alguns exemplos de *hardware* intermediários; além destes podemos destacar os *shields*[2] (RIVERO, 2011) e o *driver*,[3] que são utilizados para prover acesso facilitado aos recursos da Camada de *Hardware*.

Quanto aos elementos de *software* desta camada, destacamos os mecanismos utilizados para abstrair a complexidade do *hardware* e gerar como resultado uma interface de acesso aos sensores, atuadores e demais dispositivos, a citar:

- **Interface de *Hardware*** *(Hardware Interface)* - Componente responsável por encapsular as informações do *hardware* e criar um modelo de interface unificada para os mesmos. Contudo, para que seja possível utilizar todos os recursos de *hardware* de forma individualizada é necessário criar entidades que os identifiquem de modo único. Para tanto, utilizamos uma Tabela de Identificação de *hardware* que contém um identificador para cada *hardware* (sensor, atuador ou dispositivo inteligente), bem como a sua localização (cômodo no qual está instalado na residência, sendo atribuído o valor "móvel", caso não seja um *hardware* fixo, como é o caso de robôs, por exemplo), um descritor do *hardware* (por exemplo, sensor de fumaça), o seu endereço de rede e o protocolo suportado.

Para prover subsídios para a produção de dispositivos inteligentes personalizados que podem ser incorporados ao protótipo de SO, disponibilizamos [4] um *Firmware open source*. Este *Firmware* é flexível e extensível, possuindo como principal intuito prover uma abstração do *hardware* (sensor e atuador), bem como gerenciá-los de modo eficiente. Isto torna possível a comunicação adequada entre o *hardware* de baixo nível e as camadas superiores.

- **Gerenciador de Conexão** *(Connection Manager)* - Responsável por determinar qual conexão do *hardware* deve ser utilizada. Isto é, este componente escolhe o protocolo mais adequado para cada *hardware*, por exemplo, caso o *hardware* possua uma comunicação Bluetooth e WiFi disponíveis, este irá determinar qual utilizar para realizar a comunicação. Para tanto, é realizada uma sequência de verificações com o intuito de analisar o melhor método.

[1] Microcontrolador é um microcomputador compacto projetado para regular o funcionamento de sistemas embarcados, robôs, dispositivos móveis, eletrodomésticos e vários outros dispositivos ROBOTS (2016).

[2] Um *shield* pode ser classificado como um módulo de expansão, responsável por agregar funcionalidades e características aos dispositivos que o utilizam. Muitas vezes o *shield* fornece uma interface específica para o *hardware.*

[3] Um *driver* é um *software* que permite que o computador se comunique com o *hardware* ou com os dispositivos (MICROSOFT, 2016a).

[4] Disponível em: `http://cin.ufpe.br/~fav2/VoxarHomeOS/thesis`

- **Conversão de Dados** *(Conversion)* - Alguns dados dos dispositivos da Camada de *Hardware* precisam ser tratados antes de chegarem à Camada de Rede para serem compreendidos corretamente. Isto é, o componente denominado Conversão de Dados é responsável por prover interoperabilidade técnica entre os dispositivos de *hardware* e a aplicação.

- **Gestão de Entrada e Saída** *(I/O Manager)* - Este componente é responsável por gerenciar a entrada e saída de dados na rede. Em outras palavras, este componente funciona como interface de dados entre o *hardware* e a rede, controlando assim o modo de operação dos dispositivos físicos, utilizando os demais componentes desta camada (Tabela de Identificação e Conversão de Dados) para realizar a tarefa.

Um aspecto importante da utilização do *Firmware* está em permitir a integração dos sensores e atuadores a uma única interface, de modo a prover uma abstração quanto aos detalhes de conexões (em qual porta está um determinado sensor ou atuador), além de permitir a substituição rápida do *hardware* controlador em caso de falha, sem a necessidade de reconfiguração.

4.3.3 Camada de Rede

A Camada de Rede possui alguns componentes que têm como principal objetivo prover comunicação entre os recursos internos de um sistema operacional (tanto *hardware* como *software*) e prover conexão com serviços externos, tais como APIs, serviços de informações (por exemplo, notícias, situação do trânsito) e *Software as a service* (SaaS) (por exemplo, efetuar ligação para emergência via *Voice over Internet Protocol* (VoIP), entretenimento com *Video on Demand* (VoD), biblioteca de músicas). Além disto, é através desta camada que os habitantes podem interagir diretamente com dispositivos específicos de forma independente. Deste modo, podemos identificar três modos de comunicação:

- *Hardware*/*Hardware* - Provê comunicação entre os sensores, atuadores ou dispositivos inteligentes, de modo que os envolvidos abstraiam os detalhes da comunicação sem preocupação com a semântica dos dados.

- Usuário/*Hardware* - Provê comunicação entre os habitantes e os dispositivos inteligentes. Esta comunicação pode ocorrer nos dois sentidos: tanto o usuário pode enviar uma solicitação para um determinado dispositivo, como pode ocorrer o contrário.

- *Internet* - A comunicação da residência com o mundo externo, além de tornar possível a incorporação de tecnologias externas ainda permite que o usuário possa interagir com o ambiente residencial mesmo estando fora deste. Neste caso, é necessária uma parametrização da aplicação para permitir este tipo de gerenciamento.

Esta camada tem uma função crucial para a incorporação dinâmica de novos recursos de *hardware*, pois quando um novo dispositivo inteligente é conectado à rede, o mecanismo de descoberta é executado, solicitando informações sobre o novo elemento, caso este ainda não tenha um registro prévio. Para tanto, é utilizado o componente Tabela de Identificação de *hardware* da Camada de Abstração de *Hardware* (Seção 4.3.2), adicionando assim a informação sobre o endereço IP do dispositivo.

A Figura 4.4 realça os componentes existentes na Camada de Rede.

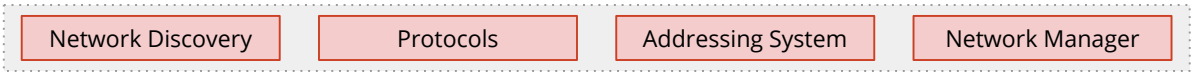


Figura 4.4 Camada de Rede.

A função de cada componente é:

- **Descoberta** *(Network Discovery)* - Este componente monitora o estado da rede, detectando a entrada ou saída de dispositivos inteligentes. O processo de detecção de ingresso na rede pode ser iniciado pelo próprio dispositivo, que ao entrar na rede envia uma mensagem para todos os demais dispositivos da rede, publicizando sua existência; ou pode ser iniciado pelo sistema, que envia mensagens para todos os dispositivos, aguardando uma resposta, onde através dessa resposta informações de rede como endereço físico (*Media Access Control* (MAC)), IP e *hostname* são coletadas. Neste último caso, quando um dispositivo que antes estava respondendo passa a não mais responder, registra-se esta ausência de resposta, indicando assim que o dispositivo está inoperante na rede.

- **Protocolos** *(Protocols)* - É o componente responsável por identificar os protocolos utilizados pelos dispositivos, através da decomposição das mensagens recebidas. Este componente contém uma pilha de protocolos suportados pelo sistema. Isto é, este componente tem a função de encapsular e desencapsular o pacote (a mensagem) para ser enviado ou recebido pelo sistema. Um exemplo desse funcionamento é quando o sistema deseja enviar uma mensagem para um dispositivo que possui o protocolo CoAP implementado, a mensagem ao chegar no componente Protocolo é transformada em uma mensagem CoAP e continua seu fluxo até chegar ao dispositivo.

- **Endereçamento** *(Addressing System)* - Este é um componente simples, onde a principal função é armazenar e fornecer informações de rede dos dispositivos conhecidos. Este componente é utilizado diretamente pelos demais componentes da Camada de Rede, pois fornece dados como MAC, IP, *hostname*, porta de acesso, protocolos suportados e *status* (operante ou não).

- **Gerenciador de Rede** *(Network Manager)* - O Gerenciador de Rede, tem a função de estabelecer um comunicação entre a Camada de Abstração de *Hardware* e a Camada de Decisão, possuindo um mecanismo concentrador de mensagens, como um *Broker*. Toda a comunicação de rede intermediada pelo sistema utiliza este componente para gerenciar o fluxo das mensagens.

4.3.4 Camada de Decisão

A abordagem desta camada é baseada na ocorrência de eventos no ambiente, onde consideramos como eventos toda tarefa detectada, tais como "o usuário ligou a TV", "o sistema ativou o alarme", "o ambiente está quente", entre outras. Dentre os exemplos citados anteriormente, existem eventos que podem ser ativados pelo usuário, pelo sistema ou por uma entidade externa não controlável. Deste modo, todo evento registrado está associado com quem (ou o que) o produziu:

- Usuário - Quando o habitante interfere de alguma maneira no ambiente, seja pressionando um interruptor, executando um gesto pré-definido ou um comando de voz, este estará produzindo um evento. Independente se o usuário realizou alguma atividade intencionalmente ou não, se a ação modificou algum aspecto do ambiente, este evento será registrado.

- Sistema - Caso a mudança no ambiente seja causada pelo próprio sistema de forma autônoma, como é o caso de um acionamento de algum equipamento (por exemplo, irrigar o jardim) ou algo similar, o registro do evento deve conter essa informação.

- Externo - Quando ocorre um fenômeno que não é controlado diretamente pelo sistema ou pelo usuário, consideramos que o responsável pela atividade é um agente externo. Um exemplo deste cenário é quando é detectado que o sol se pôs ou mesmo que está chovendo; nestes casos, os fenômenos naturais não podem ser controlados.

Após a detecção do evento, bem como sua origem e todas as informações correspondentes, verifica-se a existência de uma regra associada; em caso negativo, os dados do evento são armazenados, sendo utilizados posteriormente para análise e aprendizado. Caso o evento esteja associado a uma regra, será então executada a ação respectiva.

A Camada de Decisão possui componentes que tornam o gerenciamento das decisões do sistema mais dinâmico. Estes componentes têm como finalidade filtrar quais dados são importantes para o sistema e quais dados serão utilizados apenas para análises posteriores.

Contudo, desenvolver uma aplicação que comporte todas as possíveis regras de domínio, de forma predefinida e não permitindo ao usuário final a personalização das regras torna-se um erro de concepção, visto que mensurar previamente as possíveis relações e variações de situações de atividades em uma residência não é uma tarefa trivial. Deste modo, por mais regras que

estejam disponíveis, a predefinição de ações pode não satisfazer completamente as preferências do usuário. Para tanto, propomos mecanismos para que o usuário possa parametrizar as ações da residência de modo personalizado.

A Figura 4.5 realça os componentes existentes na Camada de Decisão.

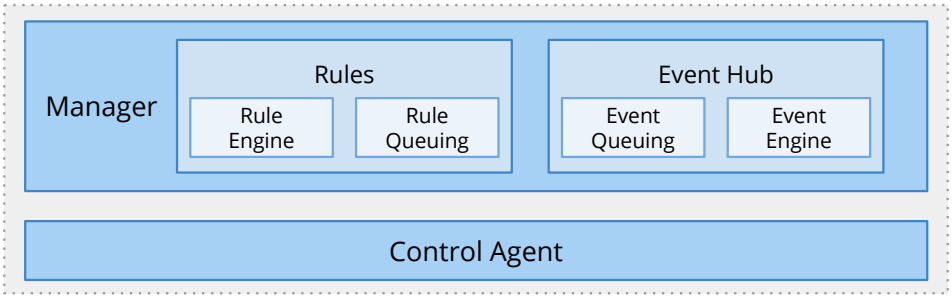


Figura 4.5 Camada de Decisão.

A função de cada componente é:

- **Controlador** *(Control Agent)* - O papel do Controlador é agir como um mensageiro, sendo realizada uma triagem sobre as informações e encaminhando a mensagem da Camada de Rede para o destino, bem como enviar um comando para a Camada de *Hardware*, através da Camada de Rede.

- **Gerenciador** *(Manager)* - É o componente responsável por gerenciar tanto o gerador de eventos quanto o gerador de regras, permitindo ao sistema o entendimento dos fluxos e *scripts*, que representam o contexto da aplicação (conforme descrito na Seção 4.3.4.1).

- **Concentrador de Eventos** *(Event Hub)* - O Concentrador de Eventos é responsável por manter a integração dos seus subcomponentes, que são:

  Lista de Eventos *(Event Queuing)* - Todo evento que está sendo monitorado é organizado neste componente, de modo que este é responsável por identificar o estado de cada evento. A combinação dos estados de alguns eventos é necessária para a execução de procedimentos no ambiente físico.

  Gestor de Eventos *(Event Engine)* - É o mecanismo utilizado pelo usuário para definir novos eventos ou gerenciar os já existentes. Quando um novo evento é criado este é adicionado à relação do componente Lista de Eventos apresentado anteriormente.

- **Controlador de Regras** *(Rules)* - O Controlador de Regras é responsável por encaminhar para o Agente Controlador da camada as requisições de ações identificadas pelos subcomponentes, que são:

Lista de Regras *(Rule Queuing)* - Componente responsável pelo monitoramento das regras. Caso ocorra a regra específica, é iniciado o processo para executar a ação. Este componente contém uma lista dinâmica de ações que podem ser realizadas pelo sistema.

Gestor de Regras *(Rule Engine)* - Similar ao subcomponente Gestor de Eventos, este mecanismo é utilizado pelo usuário para definir novas regras ou gerenciar as já existentes. Quando uma nova regra é criada, esta é adicionada a lista do componente Lista de Regras apresentado anteriormente.

Para facilitar a visualização do funcionamento e a forma que os componentes descritos nesta seção interagem entre si, apresentamos na Figura 4.6 um fluxo de dados.

Neste exemplo, o fluxo dos dados tem início na Camada de *Hardware*, através de um sensor, que é convertido e formatado pelo *Firmware*, que faz parte da Camada de Abstração de *Hardware*, sendo posteriormente encaminhado para a Camada de Rede, através da Interface de *Hardware*. Na Camada de Rede, a informação é acrescida com os dados contidos na Tabela de Identificação e interpretada pelo Gerenciador de Rede para ser enviada para a Camada de Decisão, que é responsável pelo processamento da informação oriunda do sensor.

Ao chegar à Camada de Decisão a informação é verificada; caso não seja uma ação (comando a ser executado no ambiente), a mensagem chega ao componente Lista de Eventos, que verifica se o conteúdo da mensagem está associado a algum evento existente; em caso negativo, a informação é armazenada para futura análise e o fluxo é finalizado (pela Camada de Análise) e, caso satisfaça a condição de um evento, o próximo passo é verificar se esse evento específico está associado a alguma ação (formando assim uma regra); em caso negativo, a informação é apenas armazenada no banco de dados (finalizando o fluxo), e caso contrário, caso seja uma regra, é enfim enviado um comando para a execução da ação relacionada à regra. Com isso, o Controlador de Regras verifica que a informação se trata de uma ação, e o comando é encaminhado de volta para a rede até chegar ao atuador responsável pela execução da ação.

4.3.4.1 Representação do Contexto

Para tornar possível o entendimento dos fenômenos que ocorrem no ambiente é necessário tratar os dados fornecidos pelas camadas inferiores da arquitetura. Pelo fato da Camada de Decisão possuir grande influência no ambiente, a qualidade do serviço e, principalmente o funcionamento adequado dos processos da residência são de responsabilidade dos seus componentes.

Com o intuito de facilitar o gerenciamento das regras do domínio por parte dos usuários, é possível utilizar regras simplificadas do tipo *Event-Condition-Action* (ECA), por se tratar de uma representação lógica básica. Portanto, as próximas subseções descrevem e ilustram como ocorre a representação do contexto para uma aplicação construída sobre a arquitetura proposta nesta tese.

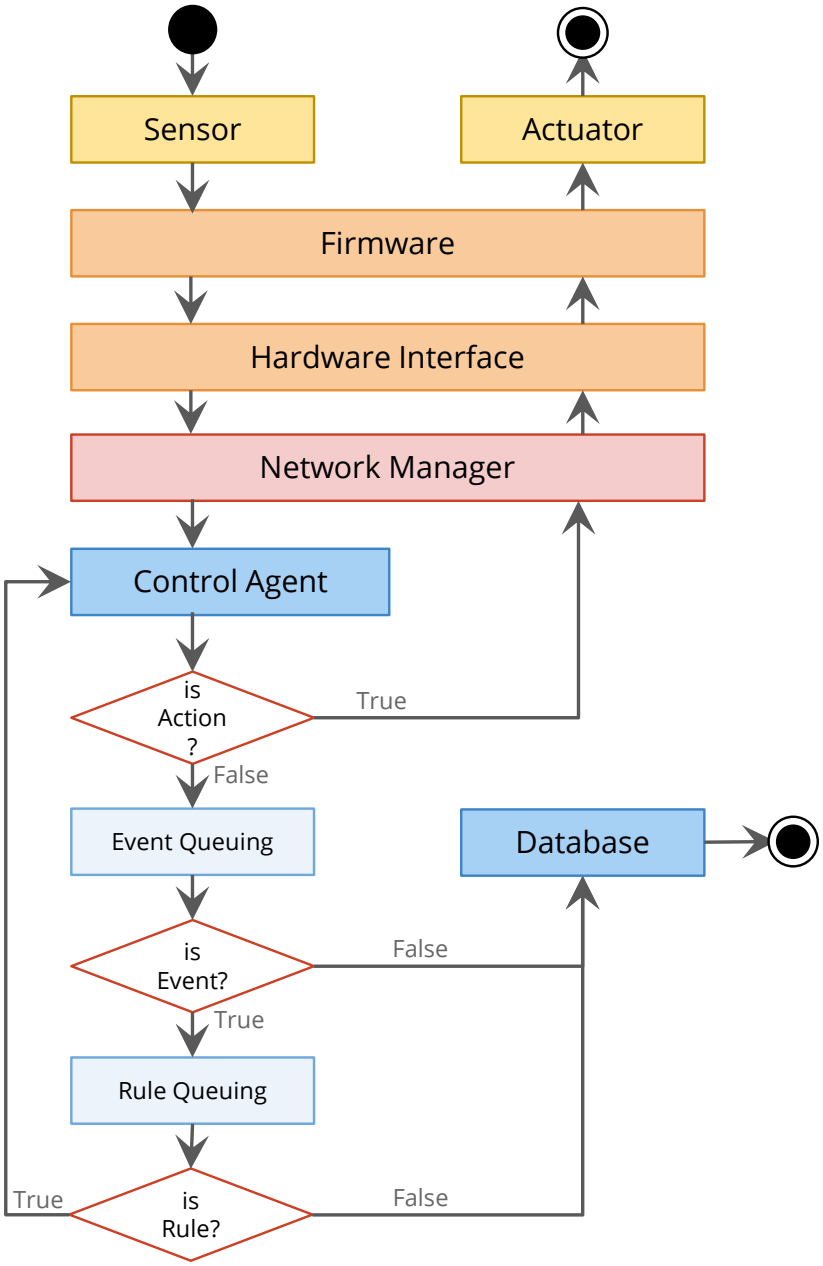


Figura 4.6 Exemplo de fluxo de dados na Camada de Decisão.

Evento

Os eventos no sistema são baseados em tarefa no ambiente, que podem ser detectadas por diversas fontes diferentes. Consideramos que qualquer sensor, seja este uma câmera de vídeo, um microfone, um sensor de temperatura, uma API de clima, entre outros, pode gerar eventos. Na Figura 4.7 apresentamos um exemplo de composição de evento.

Apesar do usuário possuir uma interface para gerenciar e visualizar os eventos, o sistema não compreende o fluxo da mesma maneira. Para tanto, a representação gráfica é transformada em *scripts* que são interpretados pelo mecanismo de decisão. Deste modo, a representação visual é formatada como apresentada pelo Código 4.1 e o pseudo-código apresentado no Algoritmo 1,

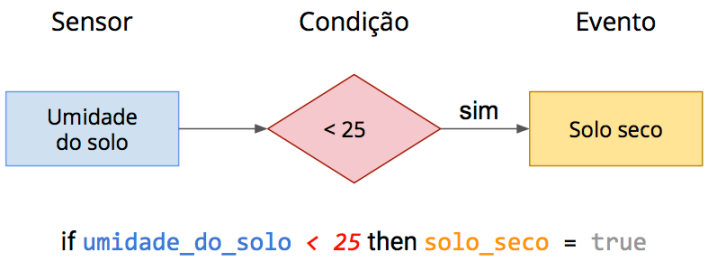


Figura 4.7 Representação de um evento.

ilustra a representação do evento para o sistema.

Código 4.1 Estrutura lógica do evento.

```
IF <sensor> <operator> <value> THEN <event>
```

Algoritmo 1 Representação do evento em pseudo-código

```
1: if umidade_do_solo < 25 then
2:     soloseco ← TRUE
3: else
4:     soloseco ← FALSE
```

Ressaltando que o estado do evento é monitorado de forma contínua e, portanto, sempre que o estado do sensor correspondente ao evento é modificado, o evento é reavaliado. Isto é, caso o valor do sensor não satisfaça a condição do evento, o estado do evento será atualizado.

Regra Simples

A partir do momento que um evento é definido, este pode ser associado a uma ou mais regras. Deste modo, as regras são baseadas em ocorrências de determinado(s) evento(s) no ambiente. Uma regra é composta pelo elemento Evento e pelo elemento Ação, que é única para cada regra. Na Figura 4.8, apresentamos um exemplo de composição de uma regra através de uma representação gráfica.

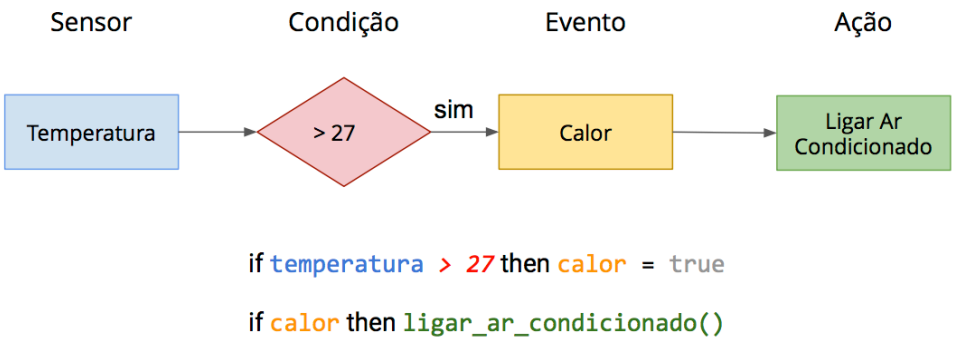


Figura 4.8 Representação de uma regra simples.

A Figura 4.8 ilustra a ocorrência do evento "Calor" e a ação que será executada quando o estado do evento for ativo (TRUE); neste caso a ação associada é "Ligar Ar Condicionado". Do mesmo modo que ocorre com o evento, a representação gráfica é transformada em *scripts* que são interpretados pelo mecanismo de decisão.

No exemplo da Figura 4.8 a regra é composta por apenas um evento, uma relação um-para-um; neste caso, denominamos este cenário de Regra Simples. A representação visual da regra simples é formatada como apresentada pelo Código 4.2 e exemplificada com o pseudo-código apresentado no Algoritmo 2.

Código 4.2 Estrutura lógica da regra simples.

```
IF <event> THEN <action>
```

Algoritmo 2 Representação da regra simples em pseudo-código

```
1: if calor == TRUE then
2:     ACTION(ArCondicionado, TRUE)
3:
4: function ACTION(object, boolean):
5:     //...implementação da função
6:     return
```

A relação um-para-um é mais simples de ser tratada pelo sistema, porém não contempla a maioria dos casos que podem ser representados. Quando existir a necessidade de verificar o estado de dois ou mais eventos para a partir disso executar uma ação deve-se utilizar uma regra composta, conforme descrito na Seção 4.3.4.1.

Regra Composta

Em um ambiente residencial nem sempre é possível representar uma regra direta, ou seja, quando ocorrer o evento X execute a ação Y. Por este motivo os mecanismos de decisão também devem prover suporte a situações que envolvem mais eventos, de modo que seja possível criar diversas combinações de eventos para executar uma tarefa. Na Figura 4.9, apresentamos um exemplo de composição de uma regra, com mais de um evento, através de uma representação gráfica.

No exemplo da Figura 4.9 a regra é composta por dois eventos associados a uma única ação, em uma relação muitos-para-um; neste caso, denominamos este cenário de Regra Composta. Na Figura 4.9, ao serem detectadas as ocorrências no evento "Solo seco" e no evento "Ar seco", a ação "Irrigar jardim" será executada. Neste caso, apenas quando os eventos estiverem simultaneamente ativos (TRUE) é que a ação ocorrerá, ou seja, caso apenas um evento esteja ativo nenhuma ação relacionada a esta regra especificamente será executada, o que não impede que outras regras que utilizem o mesmo evento para identificar outras situações possam existir.

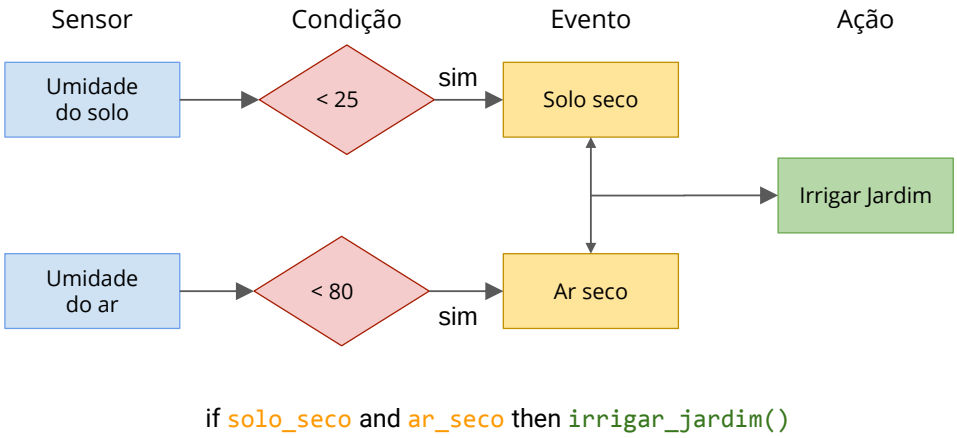


Figura 4.9 Representação de uma regra composta.

Como supracitado, após a definição da representação gráfica, é gerado um *script* que é interpretado pelo mecanismo de decisão, sendo este formatado como apresentado pelo Código 4.3 e exemplificado com o pseudo-código apresentado no Algoritmo 3.

Código 4.3 Estrutura lógica da regra composta.

```
IF <event_1> and <event_2> and  ... <event_N> THEN <action>
```

Algoritmo 3 Representação da regra composta em pseudo-código

```
1: max_events ← <list of events>
2: i ← 0
3: for i to i < max_events do
4:     if event_i == FALSE then
5:         break
6: if i == max_events then
7:     ACTION(IrrigarJardim, TRUE)
8:
9: function ACTION(object, boolean):
10:    //...implementação da função
11:    return
```

Para que as regras executem de fato uma ação no mundo real existem métodos que possuem essa função. Isto significa que as ações devem ser mapeadas previamente e possuir atuadores correspondentes para que seja possível realizar a tarefa. No caso do exemplo da Figura 4.9, é necessário que o sistema de irrigação possua algum mecanismo capaz de liberar água (por exemplo, válvula solenoide) para irrigar o jardim, conforme a ação pretendida. O mesmo ocorre para a detecção dos eventos, sendo necessários sensores que informem as condições de umidade do ar e do solo, por exemplo.

Cenário

As Regras Compostas, discutidas na seção anterior, permitem a utilização de diversos eventos para executar uma determinada ação no ambiente. Entretanto, algumas ações também podem ser utilizadas para disparar outras. Deste modo, é possível criar o que denominamos de Cenários no ambiente. Estas cenas privilegiam uma configuração mais ampla, tais como segurança, conforto, ventilação, entre outros.

Dessa forma, o sistema consegue prover ao usuário um suporte para determinar quais situações são mais interessantes, bem como proporciona acesso rápido a regras que o usuário executa com mais frequência e que seriam difíceis de gerir por um sistema de controle individual. Na Figura 4.10 apresentamos um exemplo de composição de um cenário, através de um representação gráfica.

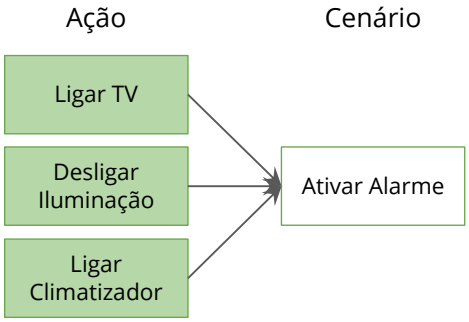


Figura 4.10 Representação de um Cenário.

No exemplo da Figura 4.10 o cenário "Ativar Alarme" é composto por três ações associadas, em uma relação muitos-para-um. Desta forma, o componente Cenário nada mais é que uma lista de ações, que quando executadas ativam o cenário (no exemplo, *Ativar Alarme* = TRUE). O *script* gerado por este componente segue o formato descrito pelo Código 4.4 e a execução é representada pelo pseudo-código do Algoritmo 4.

Código 4.4 Estrutura lógica do cenário.

```
IF <action_1> and <action_2> and  ... <action_N> THEN <scene>
```

Algoritmo 4 Representação do cenário em pseudo-código

```
1: scene ← <list of actions>
2: max_actions ← scene
3: for i = 0 to i < max_actions do
4:     ACTION(action_i, TRUE)
5:
6: function ACTION(object, boolean):
7:     //...implementação da função
8:     return
```

No caso dos cenários, os dados dos eventos não são utilizados diretamente, exceto quando as ações envolvidas em determinado cenário são modificadas pelas regras estabelecidas com base nos eventos. Por exemplo, um evento pode disparar uma ação, caso seu valor seja TRUE, mas que não necessariamente irá executar a ação contrária caso seu valor mude para FALSE, ou seja, o que é levado em consideração é o estado da ação (TRUE ou FALSE) naquele momento e não o estado do evento. Os cenários também são utilizados pela Camada de Análise para extrair informações do ambiente e preferências do usuário, podendo ainda serem sugeridos novos cenários com base nestes dados.

4.3.5 Camada de Análise

A Camada de Análise tem como principal característica estabelecer relação entre os dados obtidos nas demais camadas e realizar uma predição. Para tanto, faz-se necessário a utilização de técnicas de classificação e aprendizagem de máquina. Nesta camada ocorre a geração de conhecimento baseado na extração e análise dos dados tanto referentes ao usuário, quanto aos fatores externos não controláveis (mudança de temperatura, luminosidade, ocorrência de chuva, trânsito intenso na rota para o trabalho, entre outros).

A análise é realizada com base nos dados dos eventos, independente da origem. O objetivo desta camada é criar um mecanismo capaz de compreender o contexto residencial, tornando possível reconhecer padrões e antever ações que irão ocorrer no ambiente. Isto é, de acordo com o padrão de ocorrência de eventos pode-se inferir, com um certo grau de confiança, qual é o próximo evento ou ação mais provável.

A Figura 4.11 realça os componentes existentes na Camada de Análise.

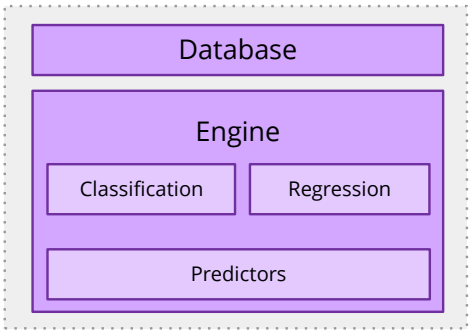


Figura 4.11 Camada de Análise.

Os dados obtidos pelo sistema, mesmo que inicialmente irrelevantes para o funcionamento adequado da residência, ficam armazenados e podem ser analisados posteriormente, no intuito de identificar padrões de ocorrência de eventos e ações. Para que uma determinada situação seja aprendida é necessário treinar o classificador com situações previamente conhecidas. Entretanto, é importante ressaltar que mesmo que os componentes da Camada de Análise indiquem que uma ação deve ser executada, esta só será realizada se atender algumas regras de

prioridades. Por exemplo, caso o usuário tenha o hábito de abrir as janelas da residência todos os dias logo após acordar, e o sistema tenha detectado este padrão, a janela não deve ser aberta, caso haja uma previsão de chuva para aquele período do dia.

Todo padrão detectado nesta camada da arquitetura é convertido para uma regra temporária da Camada de Decisão, que é responsável pela execução de fato da ação correspondente. Contudo, estas regras podem ser removidas caso o Preditor *(Predictors)* considere, posteriormente, que o padrão não é mais recorrente. Isto é, se o habitante muda alguns hábitos, a regra criada anteriormente, relacionada ao hábito (padrão) que está deixando de ocorrer, terá um aumento de entropia, ao ponto da precisão de acerto do algoritmo chegar abaixo do limiar que estabelece o nível de acurácia mínimo de uma classificação.

A função de cada componente da camada é:

- **Armazenamento** *(Database)* - O componente Armazenamento deve ser utilizado exclusivamente para armazenamento persistente dos dados obtidos. Neste componente é registrado todo o fluxo de dados do ambiente residencial. Devido à relação capacidade de armazenamento *versus* velocidade, os registros mais recentes devem ficar acessíveis mais rapidamente (memória) e os registros com o *timestamp* mais antigo devem ficar em estruturas secundárias (disco).

- **Analisador** *(Engine)* - O Analisador é o componente responsável por extrair informação do componente Armazenamento. Para tanto, deve ser utilizado o modelo preditivo, utilizando o Classificador (Classification) ou o Regressor (Regression). Ao final do processo o intuito é definir associações e categorias de dados com base em um limiar. O processo nesse componente inicia-se no acesso aos conjuntos de dados existentes no componente Armazenamento, posteriormente, de acordo com o objetivo da análise, os dados são categorizados (*clustering*) e os rótulos dos dados das classes são associados definindo nesse momento um padrão.

O processo de aprendizagem deve ser parametrizado previamente no intuito de torna-se mais eficiente. Deste modo, o sistema deve ser supervisionado, possuindo conhecimento prévio sobre o que se pretende aprender. Por outro lado, caso não se tenha conhecimento explícito sobre o que deve ser aprendido, o sistema deverá identificar padrões com base nas relações entre as variáveis com maior similaridade ou que pertençam ao mesmo grupo, categorizado anteriormente. O fluxo utilizado no aprendizado contempla desde o recebimento do conjunto de dados (rotulados ou não), sendo este utilizado para o treinamento do classificador, gerando assim o modelo de classificação a ser utilizado quando na ocorrência de novas entradas de dados não rotulados para determinar a classe adequada.

4.3.6 Camada de Interação

A Camada de Interação busca prover mecanismos que facilitem a interação natural entre o usuário e o sistema residencial. Em um cenário de residência inteligente, o ambiente é repleto de sensores e atuadores, porém, muitas vezes, o enfoque está nos fenômenos físicos como o estado de um determinado sensor ou mesmo controlar um determinado atuador, sendo que, em alguns casos, o controle e gerenciamento das interfaces são realizados por botões, telas, ou mecanismos táteis diversos.

A proposta desta camada é tornar a interação do usuário com a residência o mais natural possível, podendo evitar a necessidade de utilizar controles físicos mais complexos para tanto. Apesar de estarmos acostumados com controles remotos para controlarmos uma TV, por exemplo, o fato de usar um gesto ou um comando de voz para tal tarefa é uma alternativa que pode ser cômoda em algumas situações.

Deste modo, os componentes que compõem esta camada possuem duas funções principais: (i) permitir a definição de interações personalizadas entre o habitante e a residência; (ii) abstrair a complexidade de comandos e interfaces computacionais, tais como sistemas de gestão com diversas funções e configurações, que em muitas vezes dificultam o engajamento do usuário.

Em um ambiente residencial muitas vezes o usuário quer realizar uma tarefa simples como tocar uma lista de músicas favoritas ou mesmo ter ciência das notícias do dia, enquanto realiza outra tarefa, como lavar a louça. Visto que o usuário está ocupado com uma atividade, é interessante prover serviços que podem ser iniciados de modo simples e onipresente. Logo, os componentes da camada de interação proporcionam esta interação.

Desta forma, a Camada de Interação fornece componentes importantes para um sistema operacional residencial, favorecendo o desenvolvimento de soluções com interação natural com o ambiente.

A Figura 4.12 destaca os componentes utilizados na Camada de Interação.

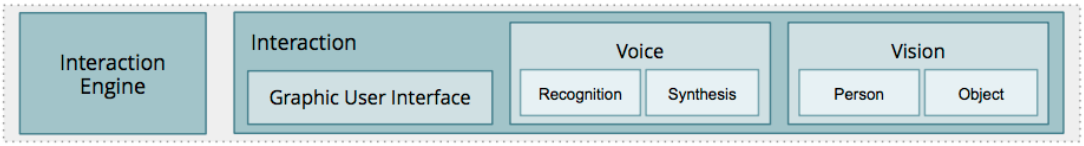


Figura 4.12 Camada de Interação.

Da mesma forma que é inviável prever as possíveis variações de situações de decisão em uma residência, o entendimento sobre o que é natural é subjetivo; se para um usuário usar uma metáfora de *swipe* pode parecer intuitivo, para outro pode ser considerado ambíguo. Logo, propomos mecanismos para que o próprio usuário possa parametrizar os métodos de interação com a residência de modo que julgar adequado. Os dois principais componentes propostos para esta camada são:

- **Gerenciador de Interação** *(Interaction Engine)* - O Gerenciador de Interação é um

engenho de gestão de métodos de interação. Através deste é possível associar um determinado gesto do usuário a um evento (similar ao que ocorre na Camada de Decisão), definir comandos de voz e por gestos. Deste modo, é possível personalizar os parâmetros relacionados aos métodos de interação que se deseja utilizar.

- **Técnicas de Interação** *(Interaction)* - O segundo componente da Camada de Interação está diretamente relacionado a como as técnicas de interação irão funcionar a partir das configurações realizadas no componente de gestão. Isto é, caso o usuário defina, no componente de gestão, que o gesto de levantar o braço direito acima da cabeça e depois baixá-lo até a altura do ombro significa que o sistema deve fechar uma persiana, por exemplo, este componente deve interpretar esse gesto como uma regra de interação.

Os detalhes dos principais subcomponentes do componente Técnicas de Interação estão descritos nas Seções 4.3.6.1 e 4.3.6.2.

4.3.6.1 Interação Baseada em Voz

A interação por voz é uma das principais formas de se trocar mensagens entre dois ou mais interlocutores, de modo que em um ambiente residencial o uso da mesma forma de interação torna-se necessário quando o objetivo é tornar o aparato tecnológico transparente ao usuário, fazendo com que este consiga utilizar os recursos residenciais sem preocupação com interfaces de controles complexos.

Portanto, propomos um componente cuja função é tratar e gerenciar este tipo de interação. O componente é composto por mecanismos de Síntese e Reconhecimento de Voz, traduzindo os comandos de voz do usuário para uma linguagem de máquina. O que ocorre é que as regras e ações definidas previamente podem ser disparadas por um conjunto de palavras específicas. Este comando pode ser definido pelo usuário para representar de forma mais adequada a situação que se pretende executar.

A Síntese de Voz é utilizada exclusivamente pelo sistema para interagir com o usuário. Um exemplo deste tipo de interação é quando o habitante solicita que o sistema execute alguma tarefa e o sistema, ao executar, dá um *feedback* sonoro, informando a situação da tarefa; outro exemplo é quando o sistema detecta alguma situação atípica (por exemplo, em caso de valores de frequência cardíaca elevada ou detecção de quedas) ou um momento de incerteza e solicita que o usuário informe o que está ocorrendo de fato.

4.3.6.2 Interação Baseada em Visão

Todas as ações do usuário devem ser monitoradas para que o sistema residencial seja cada vez mais específico e consiga englobar as peculiaridades de cada indivíduo. Entretanto, representar informações do contexto do usuário é uma tarefa complexa, principalmente quando é

necessário adicionar sensores de rastreamento no corpo do usuário, o que representa uma barreira para a adaptação ao sistema.

Portanto, a arquitetura do SO prevê, na Camada de Interação, o uso de técnicas de visão computacional para identificar atividades que ocorrem no ambiente. Neste caso, os dois principais focos de observação são os usuários e os objetos presentes no ambiente.

Reconhecimento de Gestos Corporais Humanos

O reconhecimento de gestos fundamenta-se na existência de sensores de imagem para mapear o usuário e identificar características do mesmo. O intuito deste componente é entender situações onde é possível identificar a existência de um usuário em uma cena, identificar qual atividade que este usuário está realizando ou mesmo qual informação o usuário pretende passar ao sistema. Logo, os gestos conhecidos pelo sistema são tratados como um evento de entrada para o sistema de controle. O objetivo deste componente é tornar a interação por gesto mais intuitiva, sendo o usuário o responsável por descrever os gestos que achar conviniente.

Reconhecimento de Objetos

Do mesmo modo que as pessoas possuem informações complexas de serem representadas para o sistema, os objetos também possuem características que podem ser mapeadas e transformadas em informações. Por exemplo, o usuário pode apresentar para o sistema um determinado produto e solicitar ao sistema informações sobre este, tais como: descrição, validade, informações nutricionais, entre outras.

4.3.7 Camada de Serviço

Por se tratar de uma arquitetura que sugere que os desenvolvedores consigam utilizar esta plataforma para desenvolver suas aplicações de modo facilitado, esta camada provê o uso de APIs que serão utilizadas para tal tarefa. Uma API é uma interface que define como uma aplicação pode interagir com os serviços de terceiros.

A Figura 4.13 destaca os componentes utilizados na Camada de Serviço.

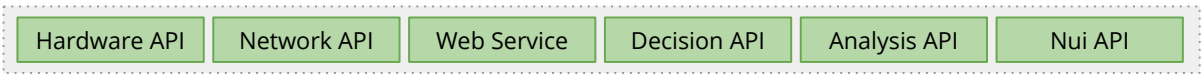


Figura 4.13 Camada de Serviço.

Para que seja possível desenvolver aplicações para residências utilizando os componentes do SO desenvolvimento sobre a arquitetura proposta não é necessário conhecer os detalhes técnicos da implementação do mesmo, porém, é necessário conhecer a estrutura da API utilizada. Os principais componentes desta camada são serviços de:

- ***Hardware*** *(Hardware API)* - Os serviços de *hardware* envolvem o acesso à Tabela de Identificadores (lista de dispositivos incorporados ao sistema), bem como os mecanismos de controle independentes da interface do SO. Através desta API é possível acessar os dados dos sensores, atuadores e dispositivos, via rede, recebendo os dados estruturados no formato JSON.

- **Rede** *(Network API)* - Por questões de acoplamento e segurança, os serviços de rede se restringem à API de acesso à tabela de endereçamento (da camada de rede) dos *hardware* conectados, bem como serviços de descoberta. Os demais serviços que utilizam a rede como meio de comunicação continuam operantes, mas os desenvolvedores não deverão ter acesso aos detalhes da implementação da rede, mitigando assim o uso de algoritmos maliciosos.

- **Serviços Externos** *(Web Service)* - Durante o desenvolvimento de um sistema operacional, podem ser utilizados diversos *Web Services* e API de acesso a informações externas à residência, tais como API de clima, para previsão de tempo, API de localização, API de trânsito, dentre outras, contudo cada API possui suas peculiaridades. Deste modo, disponibilizamos a API de serviço externo para facilitar o uso de funcionalidades similares, sem a necessidade de retrabalho. Alguns serviços implementados na nuvem também devem fazem parte desta API, como por exemplo serviços de ligação VoIP ou a biblioteca de músicas.

- **Decisão** *(Decision API)* - O controle da residência, de modo geral, é realizado pelas estruturas de controle. Deste modo, todas as informações referentes às regras de decisão são acessadas pelo Agente Controlador. A API de decisão deve fornece um mecanismo para o desenvolvedor criar novas regras, bem como consultar as existentes. Deste modo, é possível criar interfaces de controle distintas e definir regras baseadas em outras regras pré-existentes. Ao acessar o serviço de decisão é possível delinear novos eventos e gatilhos para observar (e agir, se for o caso) determinados fenômenos que ocorrem na residência. Os eventos podem ter origens diversificadas, sendo desde um gesto específico realizado pelo usuário a uma simples mudança de estado de um sensor de luminosidade digital.

- **Análise** *(Analysis API)* - Os mecanismos de análise temporal geram insumos para reconhecer padrões de um ambiente. As informações que eventualmente são extraídas não se restringem ao usuário, mas também ao próprio ambiente e sistema de forma geral. Sendo assim, as informações disponíveis pela API de análise podem ser acessadas, podendo ser refinadas e especializadas para detectar novos padrões. Através desta API deve ser possível listar os tipos de eventos e relacionar com as respectivas características.

- **Interação** *(Nui API)* - A camada de interação da arquitetura proposta possui um componente que é responsável por gerenciar os mecanismos de interação natural implementados no SO, de forma que é possível descrever novos gestos ou comandos de voz através da API de interação. Esta API abstrai a complexidade de criar aplicações baseadas em reconhecimento de gestos ou voz, permitindo a chamada de funções lógicas que retornam ações como "usuário detectado", "gesto detectado", ou ainda informações mais específicas como a posição de uma determinada parte do corpo (mão esquerda, braço direito, e assim por diante). Com isso é possível criar aplicações simples como, caso o usuário levante a mão direita, acima da cabeça, execute uma música aleatória.

4.3.8 Camada de Aplicação

Um dos objetivos esperados com a arquitetura proposta é que um sistema operacional para residências possa tornar possível o desenvolvimento de aplicações específicas para o ambiente residencial. O próprio SO deve possuir algumas aplicações nativas, responsáveis pelo gerenciamento da residência e a integração de todos os componentes da arquitetura proposta.

As aplicações básicas do SO são referentes à gestão e controle dos recursos disponíveis, a citar: Gerador de Eventos, Gerador de Regras, parametrização de novos *hardware*, Módulo de Controle Manual, Módulo de Visualização de Dados, Módulo de Definição de Gestos, entre outros.

Neste trabalho, apresentaremos (no Capítulo 5) algumas aplicações específicas (estudos de caso) desenvolvidas sobre a arquitetura proposta, como método para avaliar e validar esta arquitetura. Para tanto, foi desenvolvido um protótipo de SO, com base nos componentes e requisitos descritos neste capítulo.

5

PROTÓTIPO DO SISTEMA OPERACIONAL VOXARHOMEOS

Este capítulo apresenta uma série de estudos de caso, no intuito de apontar indícios da viabilidade e validade da arquitetura proposta no Capítulo 4. Inicialmente, apresentamos o protótipo de sistema operacional para residências, desenvolvido com base na arquitetura proposta, sendo este protótipo utilizado para realizar os estudos contidos neste capítulo. Deste modo, elaboramos seis estudos de caso que visam validar se os requisitos da arquitetura estão sendo contemplados. Posteriormente, descrevemos um sétimo cenário de aplicação com o intuito de validar os componentes das camadas da arquitetura. Para tanto, delineamos os componentes de hardware utilizados para viabilizar o experimento, bem como as técnicas e protocolos de comunicação de rede aplicados. A Camada de Interação é realçada neste capítulo, de modo que destacamos como o sistema interpreta a interação, principalmente, como método de coleta de informações, que podem ser utilizadas como eventos de entrada em determinados casos.

5.1 VoxarHomeOS

A arquitetura proposta, apresentada na Seção 4.3 (Figura 4.1), contempla as camadas de um sistema operacional para residências que observamos serem importantes para atender os requisitos identificados para a arquitetura. Fundamentando-se nesta arquitetura, foi implementado um protótipo de SO, o qual é denominado de VoxarHomeOS. Este protótipo foi implementado considerando os requisitos elencados na Seção 4.2.

A Figura 5.1 apresenta uma visão geral do protótipo.

O VoxarHomeOS funciona como um gerenciador de recursos e objetos conectados na residência, por este motivo a analogia com um sistema operacional. O VoxarHomeOS deve ser instalado em um computador conectado à rede da residência. Ao ser inicializado, o sistema realiza uma busca por dispositivos conectados, utilizando os componentes da Camada de Rede, obtendo assim um conhecimento do contexto de onde irá funcionar. Na Figura 5.1, o dispositivo

Figura 5.1 Visão geral do VoxarHomeOS.

que contém a instância do VoxarHomeOS está localizado em cima do centro da sala.

Ao ser detectada a entrada de um novo dispositivo na rede, o sistema envia para um serviço contido na nuvem (detalhes do serviço na Seção 5.1.1) estas informações novas. A sincronização dos dados é realizada sempre que for manipulado um evento, regra ou gramática (de voz ou gesto), onde o sistema local recebe estes dados da nuvem e atualiza sua base local. Deste modo, sempre que um dispositivo é conectado à rede, após sua identificação, este estará disponível para ser utilizado, visível tanto local quanto remotamente.

Para evitar que todo tráfego de dados entre os dispositivos da rede fosse enviado para a nuvem, demandando um consumo maior de dados, os componentes do VoxarHomeOS incorporam na instância local apenas novos ajustes realizados na nuvem e envia para a nuvem apenas os dados essenciais para manter a nuvem sincronizada.

Para dar suporte à interoperabilidade, foram implementados neste protótipo os protocolos MQTT e CoAP. Quando um dispositivo se conecta à rede é identificado o protocolo suportado pelo mesmo e registrado no sistema para futuras comunicações. Desse modo, dispositivos que não possuem suporte a estes protocolos podem não funcionar neste protótipo. De acordo com os níveis de interoperabilidade elencados na Seção 2.3, este protótipo contempla a interoperabilidade

técnica, semântica e pragmática.

Para que o sistema possa processar os dados de diversos dispositivos simultaneamente, foi utilizado o conceito de computação paralela, mitigando assim acumulo de processo e consequentemente uma degradação na performance.

Como já mencionado, para atender ao requisito de extensibilidade, foi fornecido um mecanismo para que o usuário possa configurar seus eventos e regras, além de disponibilizar uma API para que desenvolvedores possam ter acesso as principais funcionalidades do sistema.

Para monitorar o estado dos dispositivos na rede, foi implementado um serviço que monitora de forma independente o ingresso ou saída de dispositivos. Como alternativa, foi implementado um módulo que escuta a rede à espera de mensagens que identifiquem novos dispositivos, não sendo necessário ficar requisitando dados da rede à todo momento.

Para possuir conhecimento do contexto, foram implementados componentes capazes de receber dados de dispositivos da rede e através de estruturas condicionais interagir com os demais recursos da rede, conforme descrito na Seção 4.3.4. Na Seção 5.1.1 apresenta os recursos da interface gráfica utilizada para a definição de eventos e regras.

Para proporcionar uma análise de dados, foi implementado um algoritmo SVM, que recebe os dados dos sensores existentes e tenta reconhecer padrões e estabelecer relações entre estes, através de um aprendizado supervisionado.

Por fim, para suprir o requisito de interação natural, foi utilizada a ferramenta Prepose (FIGUEIREDO et al., 2017). O Prepose baseia-se em uma *Domain-Specific Language* (DSL) para a representação em linguagem de alto nível do gesto que o usuário deve fazer (mais detalhes na Seção 5.3.3.2) como base para criar descrições para os gestos. O Prepose foi modificado para enviar o nome do gesto descrito, sempre que o mesmo for detectado. Quanto ao aspecto relacionado a interação por voz, foi implementado um sintetizador e reconhecedor de voz, onde o sintetizador recebe o texto que deverá ser reproduzido em formato de áudio e o reconhecedor envia para a Camada de Decisão a *tag* que representa um determinado comando de voz.

5.1.1 VoxarHome Web

Após o mapeamento dos dispositivos conectados à rede da residência, o usuário poderá elaborar regras que definem o funcionamento dos recursos. Apesar de ser possível desenvolver aplicações utilizando linguagem de programação para gerenciar o ambiente, foi disponibilizada[1] uma aplicação na plataforma web, contendo uma interface gráfica para auxiliar no processo de gerenciamento da residência.

O gerenciamento web facilita o acesso aos recursos da residência, mesmo sem estar fisicamente presente na residência. Entretanto, não há dependência da internet para que o sistema funcione, por outro lado, a falha na conexão com a internet deixará a interface de gerenciamento inoperante. Como o gerenciamento por parte do usuário não é uma atividade realizada constante-

[1] Disponível em: `http://cin.ufpe.br/~fav2/VoxarHomeOS`

mente e não é crucial para o funcionamento adequado do sistema, a indisponibilidade esporádica desta interface não chega a ser um problema.

A Figura 5.2 ilustra a interface gráfica de gerenciamento denominada VoxarHome Web.

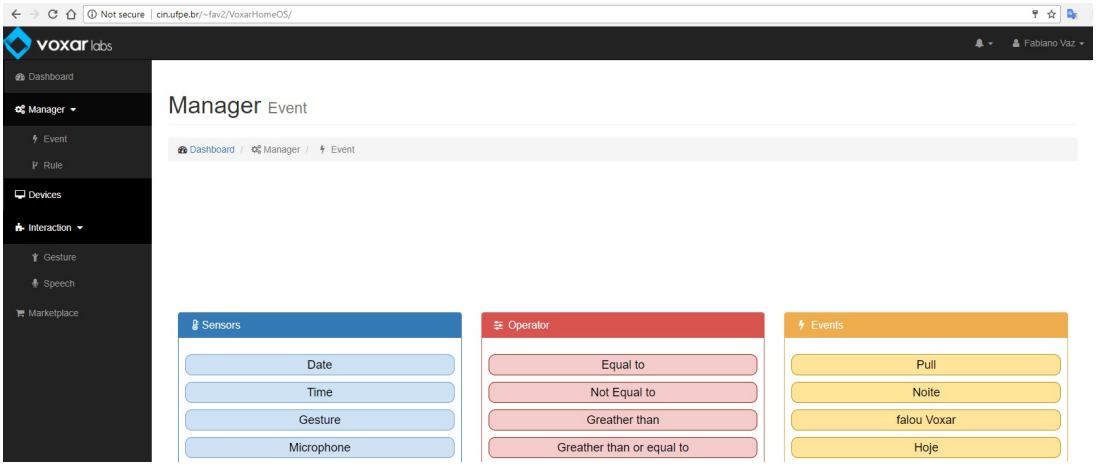


Figura 5.2 VoxarHome Web.

O VoxarHome Web foi desenvolvido levando em consideração a representação do contexto apresentada na Seção 4.3.4.1. A funcionalidade de cada menu é:

- ***Dashboard*** - Visualizar um resumo dos recursos disponíveis no sistema, tais como sensores, atuadores, eventos, regras, dispositivos.
- ***Event*** - Criar, editar ou excluir eventos, onde para isso é necessário apenas possuir um sensor cadastrado. Quando um novo sensor é descoberto pelo VoxarHomeOS, este é informado ao VoxarHome Web e poderá ser utilizado. A Figura 5.2 ilustra justamente a interface de criação de eventos, onde é possível selecionar um sensor dentre os disponíveis, selecionando em seguida o operador da expressão e, de acordo com o tipo de dado do sensor, é possível definir o valor de referência, finalizando com a definição de um nome para este evento.
- ***Rule*** - Criar, editar ou excluir regras, onde para isso é necessário possuir pelo menos um evento e um atuador disponíveis para seleção.
- ***Devices*** - Visualizar e editar dados relacionados aos dispositivos inteligentes detectados pelo sistema.
- ***Gesture*** - Criar, editar ou excluir definições de gestos, onde é necessário identificar o nome do gesto e pelo menos uma pose e uma descrição da pose, podendo definir diversas poses e descrições para cada gesto.
- ***Speech*** - Criar, editar ou excluir definições de comandos de voz, onde é possível relacionar diversas frases a uma mesma chave (ou *tag*), que será utilizada para identificar o comando voz.

- ***Marketplace*** - Na versão atual do protótipo não possui uma funcionalidade adequada. Entretanto, estamos desenvolvendo um mecanismo para que aplicações possam ser desenvolvidas e disponibilizadas para serem utilizadas por terceiros.

5.2 Estudos de Caso

Esta seção apresenta estudos de casos criados com o objetivo de apontar indícios da viabilidade e validade da arquitetura proposta. Todos os estudos de caso desta seção visam permitir a observação e a análise dos componentes da arquitetura em relação aos principais requisitos da arquitetura do sistema operacional para residências. A avaliação dos estudos de caso será qualitativa, destacando os componentes, camadas e requisitos da arquitetura. A avaliação com a abordagem quantitativa dos requisitos será apresentada no Capítulo 6.

Os estudos de caso aqui propostos têm como base a utilização do protótipo de sistema operacional, ou parte do mesmo. Assim, o primeiro estudo de caso apresentado na Seção 5.2.1 apresenta a elaboração de regra para executar uma ação de ligar a TV através de um sensor de movimento. O segundo estudo de caso mostrado na Seção 5.2.2 apresenta o uso de gestos do habitante para executar uma ação, neste caso, controlar a iluminação do ambiente. O terceiro estudo de caso, detalhado na Seção 5.2.3, apresenta o uso de uma regra composta por dois eventos, onde o usuário pode controlar a iluminação tanto realizando gestos como por comandos de voz. No quarto estudo de caso, discutido na Seção 5.2.4, é realizada uma ação onde para que um alarme sonoro seja acionado é necessário que ocorram dois eventos de forma simultânea, utilizando parâmetros de hora e reconhecimento de voz. O quinto estudo de caso, apresentado na Seção 5.2.5, utiliza um *smartwatch* como interface de controle da iluminação, mostrando como dispositivos extras podem fazer uso da API do VoxarHomeOS para acessar recursos da residência. E por fim, o sexto estudo de caso, da Seção 5.2.6, tem como objetivo apresentar um caso onde a análise de dados se faça presente, para tanto, descrevemos como a análise funciona para identificar a semântica de um determinado gesto.

5.2.1 Estudo de Caso I: Ligar TV por Aproximação

Este primeiro estudo de caso tem como objetivo principal apresentar e avaliar um cenário básico, composto por apenas um dispositivo conectado à rede, sendo gerenciado pelo SO. Neste cenário, o objetivo é apresentar uma situação onde seja possível ligar algum aparelho eletrônico da residência de acordo com a variação de algum sensor existente também na residência. No nosso caso, ligar uma televisão quando for detectado o movimento de algo em uma distância inferior à 300*cm* em relação à TV.

A Figura 5.3 ilustra a estrutura do Estudo de Caso I.

Na estrutura ilustrada na Figura 5.3, temos apenas um dispositivo (WeMos[2]) contendo

[2]WeMos é uma plataforma de desenvolvimento e prototipagem eletrônica de *hardware*. Utilizamos o modelo

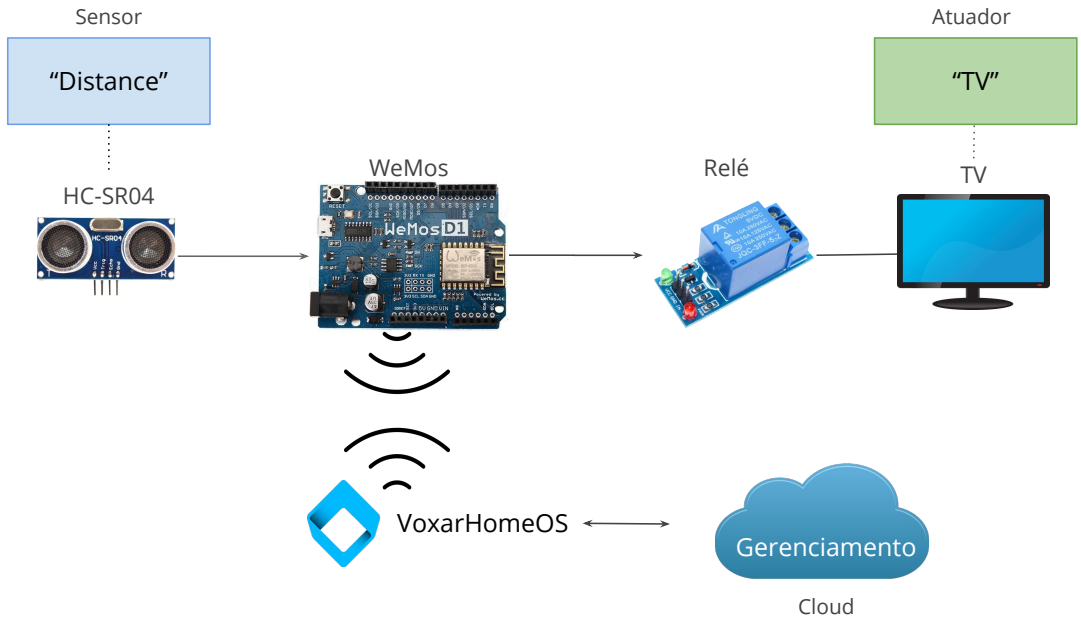


Figura 5.3 Estrutura do Estudo de Caso I.

um sensor de distância (HC-SR04) e um relé *shield* acoplado à tomada onde é encaixado o *plug* da televisão. Quando o dispositivo ingressa na rede residencial é detectado pelo sistema, através dos componentes da Camada de Rede, que reconhece também informações de (MAC), IP e *hostname*, além de identificar a existência de eventuais sensores e atuadores, por meio de uma mensagem *broadcast*. Neste caso, a troca de informações entre o dispositivo contendo o sensor/atuador e o VoxarHomeOS é realizada por meio de conexão WiFi, utilizando o protocolo MQTT.

Neste estudo de caso, nós definimos uma regra simples, onde caso seja detectado através do sensor de distância um valor inferior à 300*cm* a TV deve ser ligada. Portanto, caso um habitante se aproxime do sensor de distância a TV deverá ligar imediatamente, contudo, como não foi criada outra regra contrária, a TV continuará ligada mesmo que o habitante se ausente do ambiente. A Figura 5.4 apresenta o evento e a regra que foram criados para este estudo de caso, utilizando a interface VoxarHome Web.

A Figura 5.4 ilustra a definição de um evento denominado de *"Motion"*, composto por um sensor (*"Distance"*) que quando possuir um valor inferior à 300*cm*, será ativado, ligando a TV.

Como resultado da avaliação, é possível observar, neste cenário, o uso dos componentes das camadas de *Hardware*, de Abstração de *Hardware*, de Rede, de Decisão e de Serviço, além de atender os requisitos de Interoperabilidade, Extensibilidade, Descoberta e Sensibilidade ao Contexto, da arquitetura proposta.

Os requisitos de Interoperabilidade e de Extensibilidade foram atendidos, neste caso, visto que é possível criar suas próprias definições e configurações, de forma a estender os recursos

WeMos D1 R2, que utiliza o mesmo chip, possui as mesmas dimensões e portas lógicas de um Arduino Uno, além de disponibilizar uma interface de conexão WiFi embutida, através do módulo Esp8266.

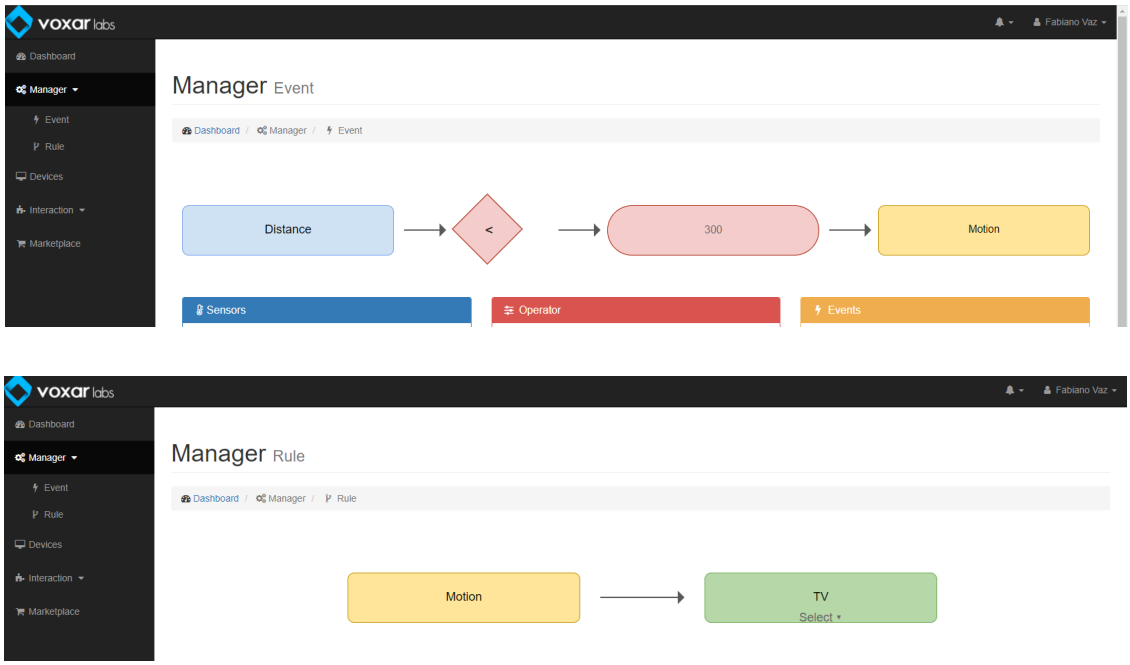


Figura 5.4 Estudo de Caso I - Definição do evento e da regra.

do SO; Descoberta, visto que quando um dispositivo se conecta ou desconecta da rede, este será identificado, podendo ser utilizado juntamente com os demais; e Sensibilidade ao Contexto, uma vez que a execução de uma ação em um determinado dispositivo só ocorre devido a mudança de estado de um sensor no mesmo contexto, mesmo que neste caso tenha sido utilizado apenas um dispositivo.

5.2.2 Estudo de Caso II: Controlar a Iluminação do Ambiente por Gesto

O Estudo de Caso II tem como principal objetivo avaliar um cenário onde o VoxarHomeOS gerencie, não apenas um, mas dois dispositivos conectados. Neste cenário, é apresentada uma situação onde é possível realizar uma tarefa na residência, através de um gesto descrito previamente. A tarefa, neste caso, é ligar uma lâmpada ao realizar o gesto denominado de *"Pull"*.

A descrição do gesto *"Pull"* é apresentada na Figura 5.5.

A Figura 5.5 ilustra a gramática que define qual é o gesto *"Pull"*, gesto esse que pode ser descrito pelo próprio usuário. Em resumo, para que o gesto *"Pull"* seja detectado é necessário o usuário colocar a ponta da mão direita abaixo do quadril, posteriormente, colocar a ponta da mão direita acima da altura da cabeça e, por fim, descer abaixo da altura da cabeça. Após a definição do gesto, o usuário poderá associá-lo a uma determinada regra.

A Figura 5.6 ilustra a estrutura do Estudo de Caso II.

Na estrutura ilustrada na Figura 5.6, temos um dispositivo Microsoft Kinect, que ao

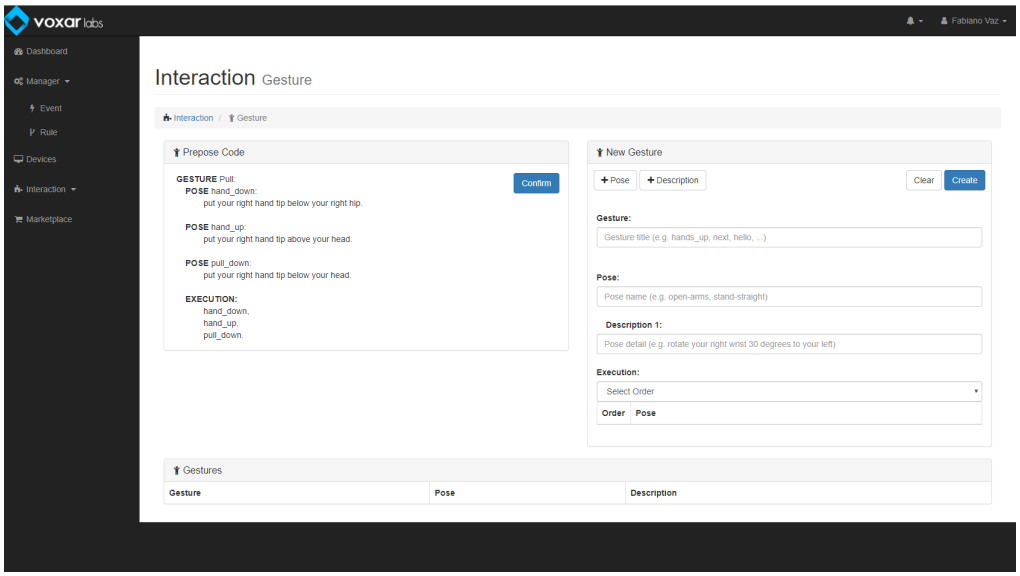


Figura 5.5 Estudo de Caso II - Definição de um gesto.

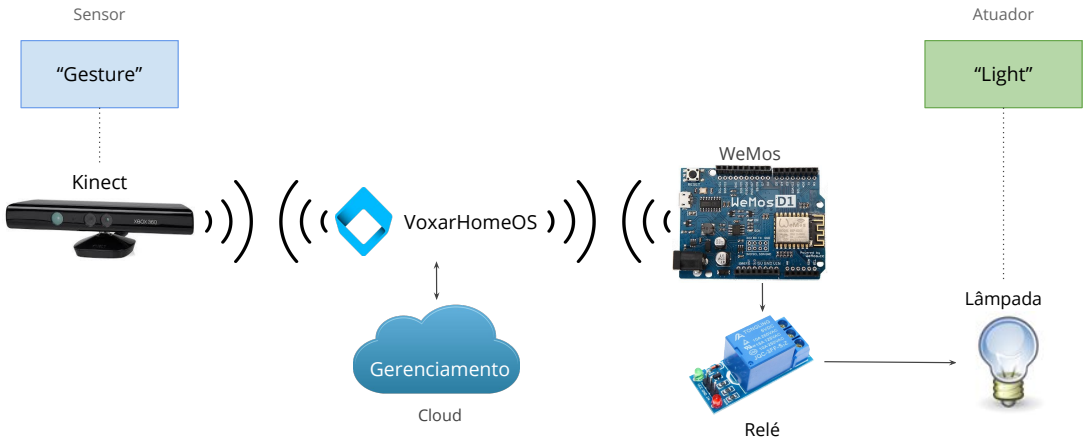


Figura 5.6 Estrutura do Estudo de Caso II.

ser conectado ao sistema informa seus eventuais serviços, como a possibilidade de capturar informações que denominamos de *"Gesture"*. Para o protótipo, *"Gesture"* é um sensor que fornece uma informação do tipo *String*, representada pelo nome do gesto detectado, previamente declarado. Além deste, temos outro dispositivo (WeMos) contendo um relé *shield* de um canal, utilizado para acionar a carga; e uma lâmpada, que deve acender conforme a determinação do sistema.

Neste caso, a troca de informações entre o dispositivo contendo o sensor e o VoxarHomeOS é realizada por meio de conexão WiFi utilizando o protocolo CoAP. Do outro lado, a troca de informações entre o dispositivo contendo o atuador e VoxarHomeOS é realizada por meio de conexão WiFi, utilizando o protocolo MQTT.

Para criar a regra que associe o gesto do usuário a uma ação de ligar a lâmpada é

necessário acessar o sistema através da interface VoxarHome Web. Neste estudo de caso, nós definimos uma regra simples, onde caso seja detectado o gesto *"Pull"* a lâmpada (*"Light"*) deverá ser acesa.

A Figura 5.7 apresenta o evento e a regra que foram criadas no sistema.

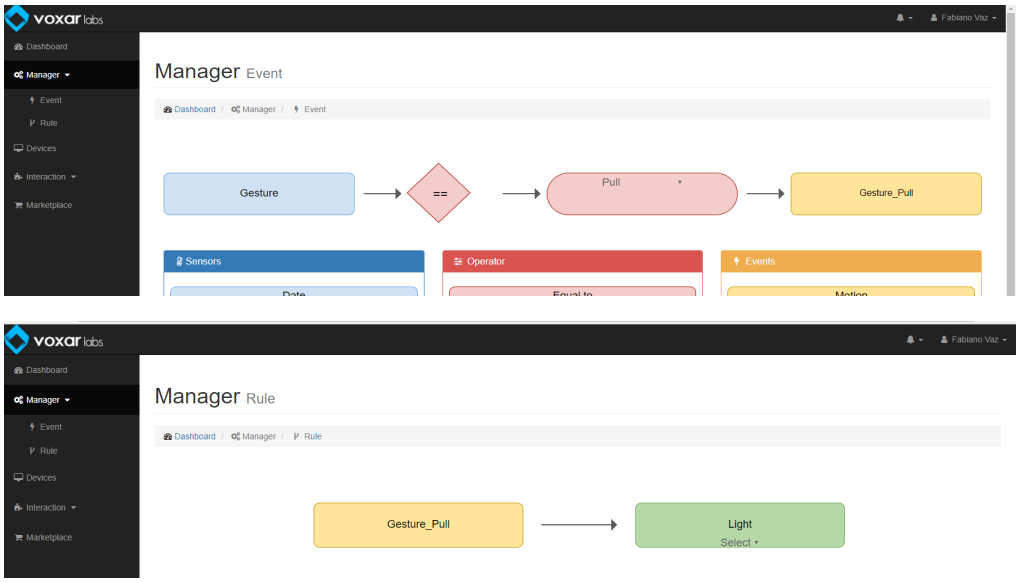


Figura 5.7 Estudo de Caso II - Definição do evento e da regra.

Neste estudo de caso os componentes das camadas de *Hardware*, de Abstração de *Hardware*, de Rede, de Decisão, de Interação e de Serviço foram utilizados. Como resultado da avaliação, é possível demonstrar o uso de:

- Interoperabilidade - visto que dispositivos que possuem implementação de diferentes protocolos (MQTT e CoAP) colaboraram para a realização da mesma ação;
- Extensibilidade - visto que é possível criar suas próprias definições e configurações, além de ser possível utilizar a linguagem de descrição de gestos para adaptar a cada situação.
- Descoberta - visto que caso um dispositivo se conecte ou desconecte à rede, este será identificado e estará operante, podendo ser utilizado juntamente com os demais.
- Sensibilidade ao contexto - visto que a execução de uma ação em um determinado dispositivo só ocorre devido a mudança de estado de outro dispositivo do mesmo contexto.
- Interação - visto que, neste estudo de caso, o usuário pode utilizar uma comunicação corporal com o sistema que, por sua vez, consegue interpretar e intervir no ambiente onde o usuário se encontra, no caso, acendendo a lâmpada.

5.2.3 Estudo de Caso III: Controlar a Iluminação por Voz ou Gesto

Ampliando a avaliação da arquitetura, além do gesto já utilizado no Estudo de Caso II, avaliamos um cenário com objetivo de apresentar uma situação onde é possível realizar uma tarefa na residência, tanto através de um gesto como por meio de um comando de voz, descritos previamente. Neste caso, o cenário avaliado contém dois dispositivos gerenciados pelo SO. Dentro desse contexto, a ação adotada para este estudo é ligar uma lâmpada ao realizar o gesto *"Pull"* ou falar algo com a semântica de "LigarLâmpada" (de acordo com a gramática).

A Figura 5.8 ilustra a gramática que define quais comandos de voz representam a *tag* "LigarLâmpada", comandos estes que podem ser descritos pelo próprio usuário. Isto é, se o habitante falar qualquer uma das quatro frases ilustradas na Figura 5.8, o sistema interpretará como dito o comando "LigarLâmpada". Após a definição, o usuário pode utilizar a gramática para associar a uma regra.

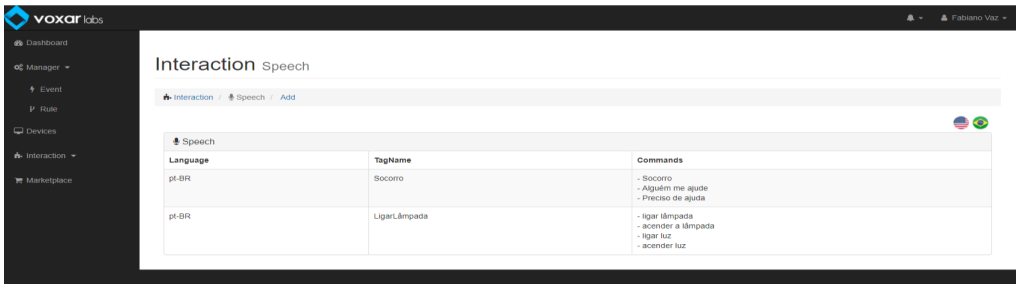


Figura 5.8 Estudo de Caso III - Definição de um comando de voz.

A Figura 5.9 ilustra a estrutura do Estudo de Caso III.

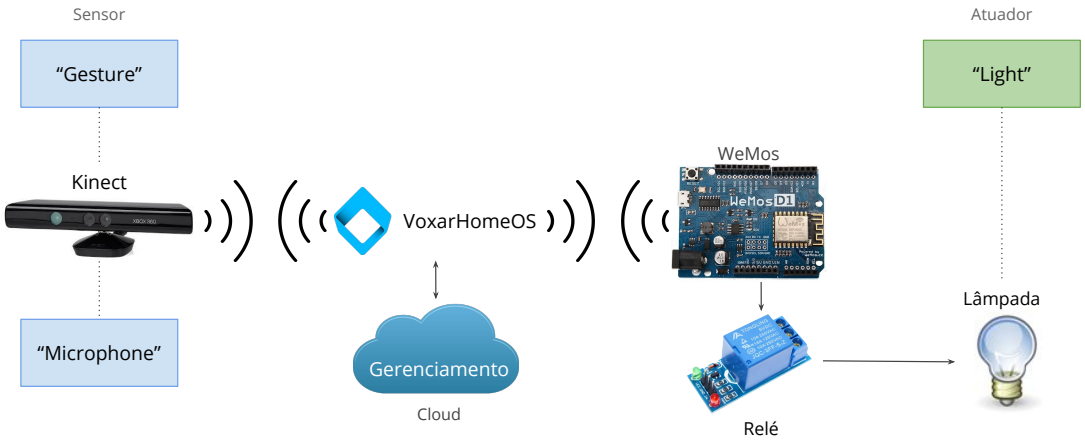


Figura 5.9 Estrutura do Estudo de Caso III.

Na estrutura ilustrada na Figura 5.9, temos um dispositivo, responsável por gerenciar o Microsoft Kinect, que ao ser conectado à rede residencial, é detectado pelo sistema que, entre outros eventuais sensores, possui um denominado *"Gesture"* e outro denominado de

"Microphone". Estes sensores fornecem informações do tipo *String*, representadas pelo nome de um determinado gesto e por uma *tag*, que define o comando de voz.

Além deste, temos ainda um WeMos, utilizado para estabelecer conexão WiFi; um relé *shield* de um canal, utilizado para acionar a carga; e uma lâmpada, que deve acender e conforme a determinação do sistema.

Para criar a regra que associe o gesto e o comando de voz do usuário a uma mesma ação de ligar a lâmpada é necessário definir dois eventos independentes. Nós utilizamos os eventos ilustrados nas Figuras 5.7 e 5.10, posteriormente, criamos duas regras simples com estes eventos, como ilustrado também nas figuras mencionadas. Deste modo, caso o gesto *"Pull"* seja detectado a lâmpada acenderá e caso o comando de voz *LigarLâmpada* seja detectado a lâmpada também acenderá, salvo o caso de já estar acesa. Este cenário demonstra o uso de uma disjunção lógica, utilizando o conectivo *OR*, onde uma regra será executada se um ou outro evento for verdadeiro.

A Figura 5.10 apresenta o evento e a regra que foram criadas no sistema.

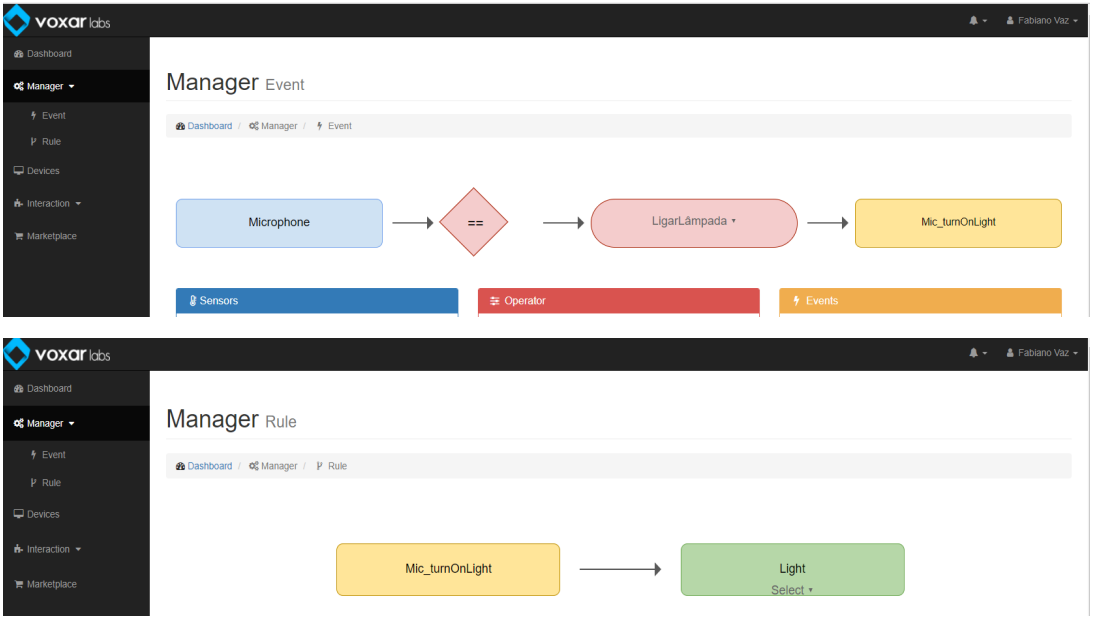


Figura 5.10 Estudo de Caso III - Definição do evento e da regra.

Desta forma, este estudo de caso avalia e valida as mesmas camadas e requisitos do Estudo de Caso II, a citar: camadas de *Hardware*, de Abstração de *Hardware*, de Rede, de Decisão, de Interação e de Serviço. Entretanto, com a extensão realizada, foi possível observar o uso de dois eventos independentes manipulando o mesmo atuador.

De todo modo, a Extensibilidade apresentada nos Estudos de Caso I, II e III só é possível devido à interface VoxarHome Web, que fornece um meio de criar soluções sem a necessidade de conhecimento prévio em linguagem de programação. Para apresentar uma abordagem de extensibilidade mais ampla, apresentaremos na Seção 5.2.4 um estudo de caso onde a regra discutida foi implementada através de linguagem de programação.

5.2.4 Estudo de Caso IV: Acionar o Alerta Sonoro de Segurança pelo Horário e por Voz

Nos Estudos de Caso I, II e III apresentamos situações que fizeram uso da interface VoxarHome Web para gerenciar e definir os eventos e as regras. Neste quarto estudo de caso, apresentaremos uma abordagem diferente, onde utilizamos a API do VoxarHomeOS para criar uma regra composta, utilizando uma expressão lógica com o conectivo *AND*.

Neste caso, assumimos que um alarme sonoro deve ser emitido, caso o horário seja maior que 18h e o habitante fale algo que esteja rotulado com a *tag* "Socorro". Os comandos de voz definidos como "Socorro" podem ser visualizados na Figura 5.8, da Seção 5.2.3.

A Figura 5.11 ilustra a estrutura do Estudo de Caso IV.

Para este estudo de caso utilizamos do computador que executa o VoxarHomeOS o relógio do sistema e os alto-falantes, como saída de som. Um outro computador, realizou a função de um dispositivo, contendo um microfone, que denominamos de sensor *"Microphone"*. Para tanto, implementamos um exemplo de sensor em C#, que ao conectar-se à rede, publica as informações de rede (MAC, IP e *Hostname*) e verifica a existência de dois sensores, um denominado *"Microphone"* e outro denominado *"Time"*, além de um atuador denominado de *"Speaker"*.

O Código 5.1 representa como o Estudo de Caso IV foi parametrizado.

Para tal, foi utilizada a API do VoxarHomeOS (como pode ser observado na linha 1), sendo posteriormente criadas duas instâncias de tipo Sensor (*sensorMic* e *sensorTime*) e criada uma regra ("Alarme") contendo dois eventos ("Chamando socorro" e "Noite"). Caso a regra seja executada, o texto "Precisamos de ajuda!" deverá ser transformado em áudio e reproduzido no alto-falante, como forma de alarme.

Código 5.1 Estudo de Caso IV - Exemplo de regra.

```
1  import com.voxar.house.*;
2
3  public class VoxarHome_OS {
4    public static void main(String[] args) {
5      Core.start();
6
7      Sensor sensorMic = Facade.instance().searchSensorByName("
           Microphone");
8      Sensor sensorTime = Facade.instance().searchSensorByName("Time");
9
10     Actuator actuator = Facade.instance().searchActuatorByName("
           Speaker");
11
12     Event event1 = new Event("Chamando socorro", "", false, sensorMic
           , "==", "Socorro");
```

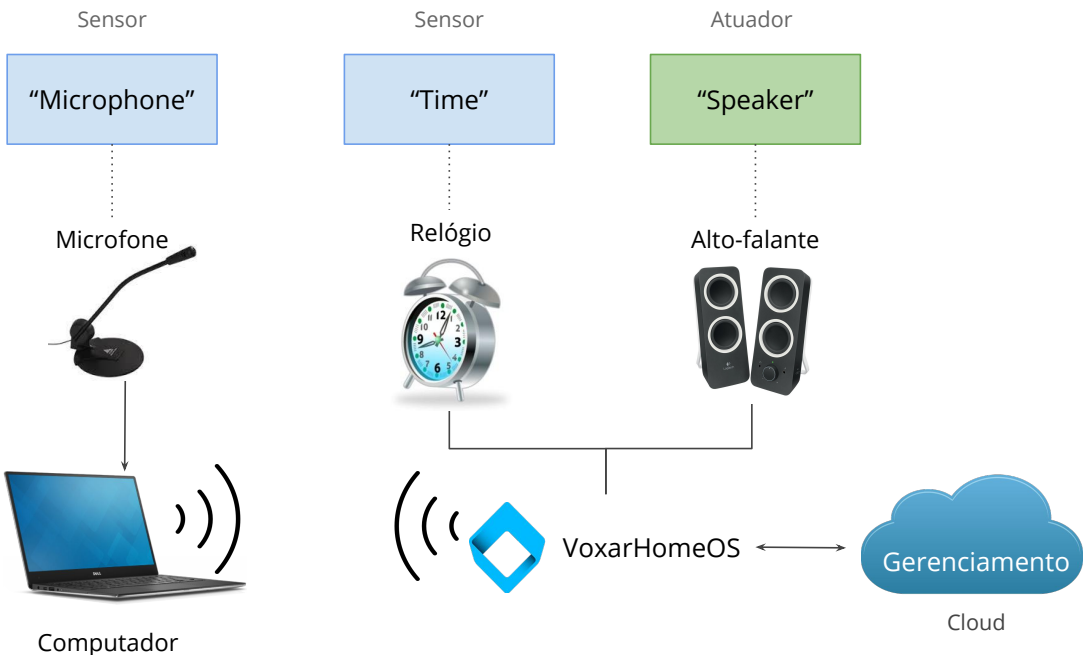


Figura 5.11 Estrutura do Estudo de Caso IV.

```
13      Event event2 = new Event("Noite", "", false, sensorTime, ">=", "
            18:00:00");
14
15      List<Event> eventList = new ArrayList<Event>();
16      eventList.add(event1);
17      eventList.add(event2);
18
19      Rule rule = new Rule("Alarme", eventList, actuator, "Precisamos
            de ajuda!");
20      Facade.instance().insertRules(rule);
21    }
22  }
```

Detalhando o Código 5.1, podemos observar na linha 1 o uso da API; na linha 5, o início da execução do SO; nas linhas 7 e 8 as busca pelos objetos do tipo Sensor; na linha 10, a busca pelo atuador chamado *“Speaker”*; nas linhas 12 e 13, os objetos *“event1”* e *“event2”* são construídos, destacando que o evento *“event1”* terá o valor *true* se o valor do sensor *“Microphone”* for igual à “Socorro” (conforme a gramática) e o evento *“event2”* terá o valor *true* se o valor do sensor *“Time”* for igual ou superior à 18h; na linha 15, instanciamos uma lista de eventos (*eventList*) vazia e logo em seguida, nas linhas 16 e 17, adicionamos os eventos à lista de eventos; por fim, na linha 19, construímos uma regra denominada de “Alarme”, contendo os eventos supracitados e na linha 20 a regra é armazenada. A partir desse momento, a regra já estará em funcionamento, aguardando apenas que ambos os eventos (*“event1”* e *“event2”*) ocorram simultaneamente.

Analisando o estudo de caso, podemos observar o uso de requisitos como Interoperabi-

lidade, Extensibilidade, Descoberta, Interação, Sensibilidade ao Contexto. Além de avaliar as mesmas camadas dos Estudos de Caso II e III, este estudo demostra a utilização dos componentes da camada de Aplicação.

Entretanto, uma limitação da abordagem utilizada neste estudo de caso consiste na transição de valores, visto que o valor do evento é modificado apenas quando ocorre a variação do sensor associado ao evento. Caso o habitante chame por socorro após as 18h, o sistema irá executar o áudio definido previamente. Contudo, caso o habitante chame por socorro novamente, nada acontecerá, pois os valores dos sensores ainda serão os mesmos (*Microphone* == "Socorro" e *Time* >= 18:00:00). Deste modo, da forma como o cenário foi configurado, seria necessário falar qualquer outra coisa para mudar o valor do sensor *"Microphone"*, para posteriormente chamar por socorro novamente.

5.2.5 Estudo de Caso V: Controlar a Iluminação por *Smartwatch*

O objetivo deste estudo de caso é avaliar a arquitetura proposta sob a perspectiva da extensibilidade e da interoperabilidade, entre dispositivos de arquiteturas, plataformas e linguagens distintas. O cenário avaliado é composto por dois dispositivos conectados à rede, sendo um gerenciado pelo SO e o outro não. Neste cenário, o intuito é apresentar uma situação onde seja possível manipular o estado de um objeto (atuador) da residência através de um dispositivo não gerenciado pelo sistema, apesar de ambos estarem presentes na mesma residência. Para tanto, utilizaremos um *smartwatch* para controlar a iluminação de um ambiente, independente das informações do contexto.

A Figura 5.12 ilustra a estrutura do Estudo de Caso V.

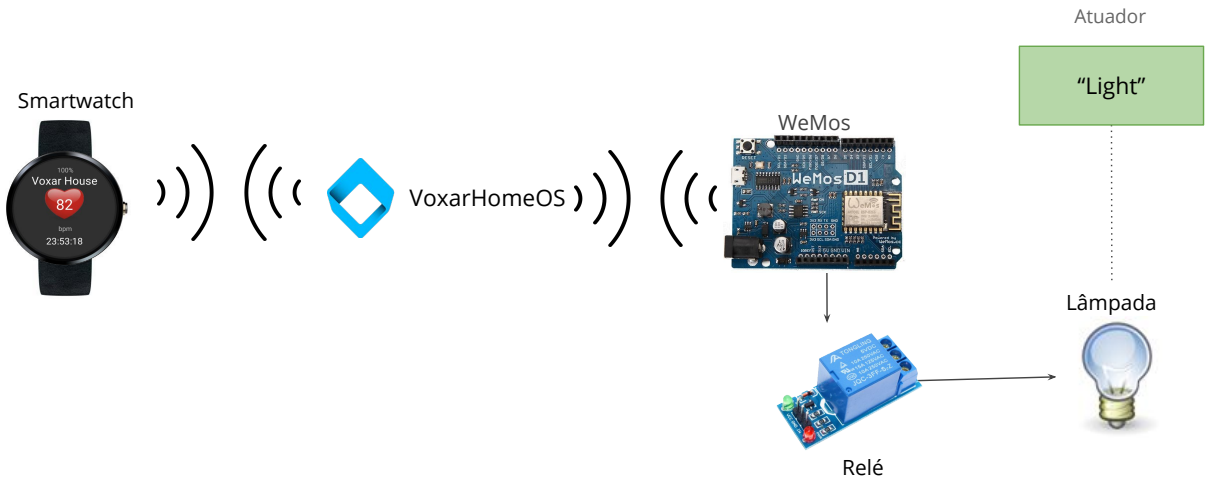


Figura 5.12 Estrutura do Estudo de Caso V.

Na estrutura ilustrada na Figura 5.12, temos um dispositivo (WeMos) contendo um relé *shield* acoplado à tomada onde é encaixado o cabo de alimentação elétrica da televisão e um *smartwatch* que, apesar de estar conectado à rede, neste cenário, adotamos uma abordagem que

não utiliza diretamente o VoxarHomeOS para criar associações, implementando um aplicativo na plataforma *Android Wear* que recebe a lista de atuadores disponíveis no SO e aciona uma lâmpada através do próprio visor do relógio.

Neste caso, utilizamos o sensor *"Push_button"* como referência para acionar o atuador *"Light"*. Para tanto elaboramos um evento, onde se *"Push_button"* for igual à 1 o evento *"Button_Watch"* terá o valor *"true"*, ou seja, ao tocar na tela do *smartwatch* será enviada uma mensagem informando que *"Push_button"* é igual à 1. Como foi elaborada uma regra associando o evento *"Button_Watch"* ao atuador *"Light"*, quando o evento for verdadeiro, a luz respectiva acenderá.

Caso seja necessário, é possível utilizar o *smartwatch* para executar a mesma ação, mesmo fora da estrutura de rede da residência, entretanto, seria necessária uma configuração nos parâmetros da aplicação.

Logo, este estudo de caso serviu para validar as camadas de *Hardware*, Abstração e *Hardware*, Rede, Decisão, Serviço e Aplicação, bem como os requisitos de Extensibilidade, Interoperabilidade, Descoberta e Contexto. Ressalta-se que a Extensibilidade pode ser observada, visto que uma aplicação em uma plataforma distinta foi desenvolvida, estendendo as funcionalidades do VoxarHomeOS.

5.2.6 Estudo de Caso VI: Semântica do Gesto

O Estudo de Caso VI tem o intuito de investigar se o requisito de análise de dados é atendido. Este estudo de caso é composto por um mini-PC, simulando uma série de sensores sinteticamente e por um dispositivo responsável por controlar um determinado atuador. Neste cenário, o objetivo é apresentar uma situação onde seja possível manipular o estado de um objeto da residência através de análise de uma série de sensores gerenciados pelo sistema, onde deve ser possível identificar a semântica do gesto. Neste caso, simulamos dados de um mês de uso de um habitante que utiliza um gesto, que denominamos de *"Hello"*, para ligar uma televisão. Entretanto, um gesto pode ser ambíguo, como ilustrado na Figura 5.13.

Figura 5.13 Estudo de Caso VI - Semântica do gesto.

A Figura 5.13 demonstra o cenário que este estudo de caso está inserido. Neste cenário, um gesto denominado de *"Hello"*, descrito utilizando a interface VoxarHome Web, pode ter sido criado para ligar uma TV, porém em algum momento o mesmo gesto, mesmo que não intencionalmente, pode ser utilizado com outro sentido, no caso ilustrado, para cumprimentar uma pessoa.

Este experimento foi dividido em duas etapas, a primeira foi a geração de um *dataset* contendo dados simulados de uso referente a 30 dias. A segunda etapa consiste em realizar a análise dos dados gerados e reconhecer padrões, detectando quais momentos do dia são mais prováveis que o usuário deverá estar assistindo televisão. Deste modo, é possível identificar o vetor de características (sensores e atuadores) que representam a classe "ligar TV".

A Figura 5.14 ilustra a estrutura do Estudo de Caso VI.

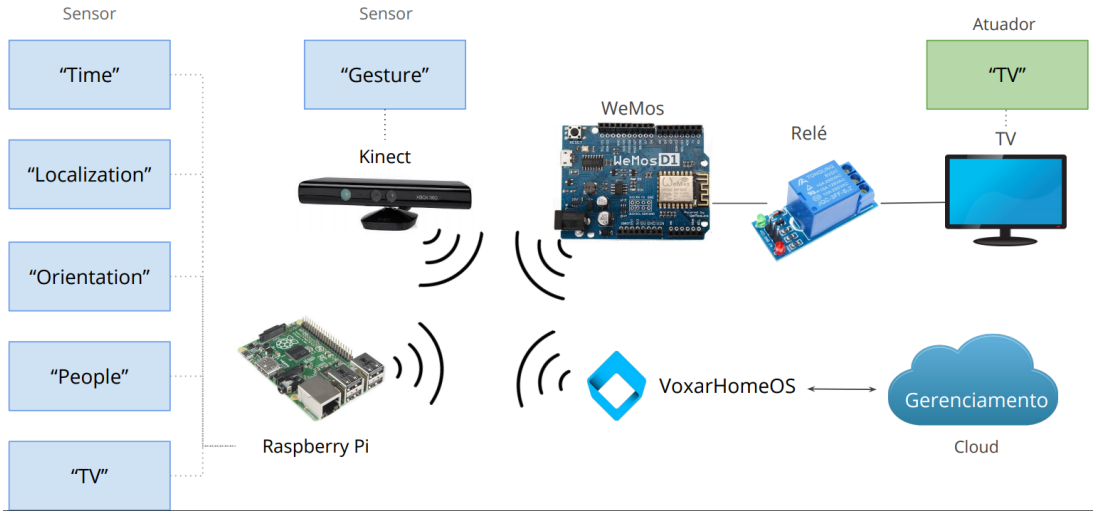


Figura 5.14 Estrutura do Estudo de Caso VI.

A estrutura apresentada na Figura 5.14 é composta por um Raspberry Pi, responsável por gerar os valores dos sensores virtuais, por um WeMos, responsável por controlar a carga elétrica da TV e por um dispositivo Kinect, responsável por rastrear o gesto.

A Figura 5.15 representa a composição da regra utilizada neste estudo de caso. Contudo, a regra é definida pelo próprio SO após a análise, não sendo necessário o usuário elaborar uma regra manualmente, sendo a Figura 5.15 apenas ilustrativa.

O evento *"Hello_TV"* ilustrado na Figura 5.15 foi criado através da análise realizada. Após a regra ser criada, nós configuramos os valores dos sensores virtuais, simulados pelo dispositivo *Raspberry Pi*, para atender todas as restrições, tornando o evento *"Hello_TV"* verdadeiro e, por consequência, ligando a TV. Para que isso ocorra, o contexto deve ser: *"Localization"* igual à 4 (código para "Sala", no sistema); *"Orientation"* igual à 2 (código para "Sentado"); *"People"* estar entre 1 e 1.3, o que na prática significa existir apenas 1 pessoa no ambiente; "TV" igual à zero (equivalente à *"false"*); por fim, *"Gesture"* igual à 1 (código do gesto *"Hello"*, no sistema). Isto é, se o habitante realizar o gesto *"Hello"* quando estiver sentado, sozinho, na sala e com a TV desligada, o sistema entende que a TV deverá ser ligada, caso contrário, ao realizar o mesmo

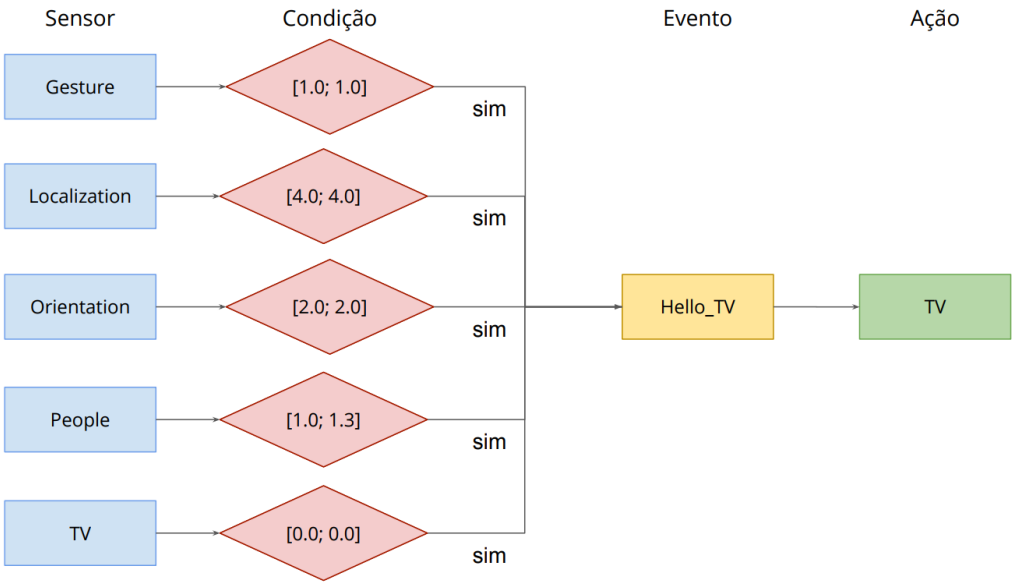


Figura 5.15 Estudo de Caso VI - Representação da regra aprendida pelo SO.

gesto fora desse contexto, o sistema irá ignorar o comando, registrando apenas os dados dos sensores para análises futuras.

Para a realização deste Estudo de Caso VI foram utilizados componentes das Camadas de *Hardware*, de Abstração de *Hardware*, de Rede, de Decisão, de Análise, de Interação e de Serviço. Além dos requisitos de Interoperabilidade, Descoberta, Contexto e Interação, este estudo de caso demonstrou como o requisito da análise de dados foi tratado. Nesta seção discutimos a análise de dados sob uma perspectiva qualitativa, entretanto, na Seção 6.6, do Capítulo 6, utilizamos uma abordagem quantitativa, avaliando a precisão e o tempo que demora para um conjunto de dados ser analisado.

5.3 Avaliação Horizontal

Nesta seção, apresentamos um sétimo cenário, onde avaliamos os componentes da arquitetura proposta (Figura 4.1) sob uma perspectiva horizontal, isto é, avaliando os componentes por camadas. Esta avaliação teve como objetivo validar os componentes da arquitetura, através de aplicações desenvolvidas para investigar a coesão da arquitetura proposta, inseridas em um contexto residencial composto por dispositivos heterogêneos. Ainda nesta seção, apresentaremos uma contextualização do cenário com enfoque nas funcionalidades do ambiente e nas próximas subseções a ênfase é dada aos recursos tecnológicos propriamente ditos.

O cenário selecionado para estudar as variações e evidências da implementação do SO é composto por uma série de módulos e funcionalidades. Os casos contemplam desde os aspectos relacionados à saúde do habitante a aspectos relacionados a entretenimento, a citar:

- **Notificação** - Envia um alerta para o usuário sobre a ocorrência de algum evento

importante. A notificação, geralmente, ocorre através de um *smartwatch*, porém pode ser realizada por outros meios, como alto-falantes ou interfaces gráficas.

- **Assistente de saúde** - Este módulo tem a finalidade de monitorar em tempo real o estado de saúde do habitante. Atualmente, observamos os batimentos cardíacos e quedas repentinas.
- **Chamar emergência** - Nós criamos um serviço VoIP implementado em um servidor Linux (utilizando Asterisk) para que em momentos de emergência, como um problema de saúde (frequência cardíaca elevada ou queda, ambos seguidos de ausência de interação), o sistema possa ligar para números previamente informados, como o sistema de atendimento móvel de urgência.
- **Central de música** - O usuário pode solicitar que seja executada uma determinada música ou um gênero musical de sua preferência.
- **Notícias** - O sistema possui acesso a portais de notícias e quando necessário pode *ler* uma manchete ou um resumo para o usuário.
- **Irrigação** - Um módulo independente de interação do usuário, sendo monitorado o estado do jardim e informações climáticas (reais e virtuais), para manter o jardim em condições adequadas, irrigando quando necessário.
- **Medidor de consumo** - O módulo de medidor de consumo observa o consumo instantâneo de energia e água, criando gráficos para visualização e acompanhamento de consumo.
- **Alarme residencial** - Módulo responsável por monitorar acessos (portas e janelas) e acionar ou não o alarme da residência, enviando em paralelo uma notificação para o habitante.
- **Câmeras IP de vigilância** - Criamos um módulo para incorporar câmeras IP ao sistema, sendo testado até o momento apenas com câmeras da marca Foscam. Deste modo, é possível integrar o gerenciamento da segurança em uma interface unificada.
- **Monitoramento veicular** - Representa o módulo responsável por coletar informações do computador de bordo do veículo, a saber RPM, temperatura, bateria, nível do combustível, entre outras.
- **Interação por gestos e voz** - Desenvolvemos mecanismos de reconhecimento de gesto e voz para facilitar a interação entre o usuário e a residência. O principal componente utilizado atualmente para este fim é um robô (detalhes na Seção 5.3.1.2).

5.3.1 Componentes de *Hardware*

Para realizar o experimento prático no cenário discutido na Seção 5.3 foram utilizados alguns recursos de *hardware*. A residência possui uma diversidade de sensores capazes de capturar os fenômenos que podem ocorrer no ambiente, bem como de atuadores responsáveis por modificar o estado do ambiente. O destaque neste cenário é a utilização de um robô doméstico que possui a função de assistente pessoal. A Figura 5.16 apresenta os principais recursos de *hardware* utilizados. Os elementos ilustrados com uma borda azul representam os dispositivos utilizados e os ilustrados com uma borda verde os sensores ou atuadores. O sensor Kinect possui ainda os componentes microfone e câmera RGB-D embutidos.

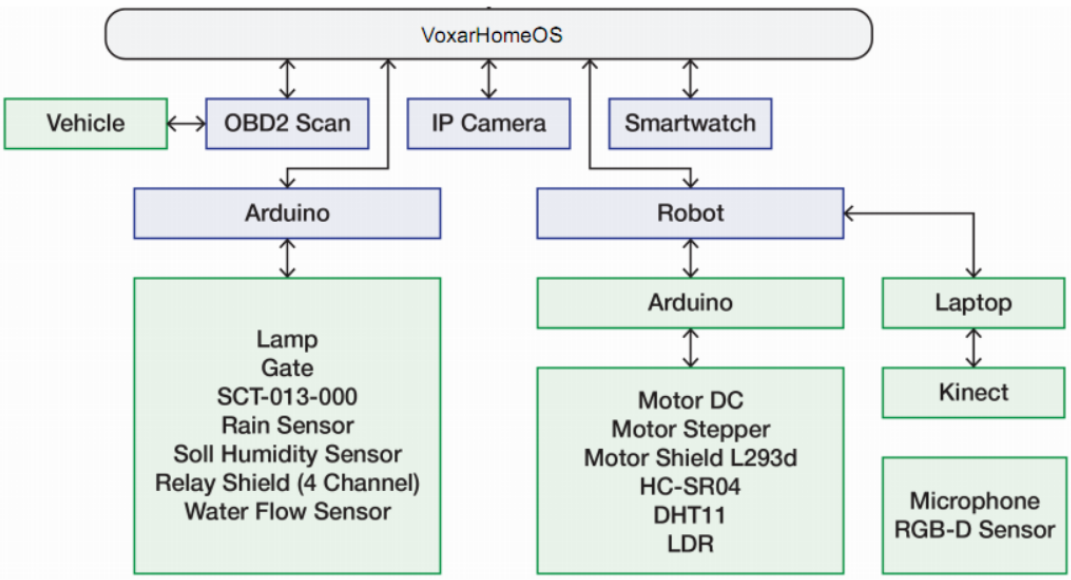


Figura 5.16 Principais componentes de *hardware* utilizados.

Na infraestrutura elaborada para o experimento, os dispositivos físicos são distribuídos entre as camadas da arquitetura (apresentada na Figura 4.1), sendo utilizados recursos nas Camadas de *Hardware*, Abstração de *Hardware*, Rede, Decisão, Serviço e Aplicação. A infraestrutura é composta por:

- **Servidor** - Computador que contém o protótipo do VoxarHomeOS, com processador Intel Core i5-4210U de 2.40 GHz, memória de 6GB RAM, HD 1Tb, Atheros AR3012 Bluetooth 4.0 e placa de vídeo GeForce GT 840M. A função principal deste servidor é prover os serviços do SO, possuindo outras tarefas como processar o fluxo constante de dados, gerenciando os recursos existentes na residência e o acesso aos dados (atualmente utilizamos o SGBD PostgreSQL para armazenamento persistente dos dados).

- **Roteador** - Na rede utilizamos um roteador *wireless* (TP-Link WR-841ND de 300 Mbps, DHCP ativo) para conectar os dispositivos (computadores, *smartphones*,

smartwatch, TVs, lâmpadas, robô móvel, veículo), configurado para permitir a comunicação em algumas portas específicas (principalmente de/para acesso externo), bem como configuradas para redirecionar as requisições (inclusive vindas do servidor). Além do roteador principal, utilizamos um *access point* (TP-Link WR700N, em modo cliente) para ter acesso (via WiFi) aos dados do veículo.

- ***Firmware*** - É importante ressaltar que entre a Camada de *Hardware* e a Camada de Rede, em alguns casos, é necessário a utilização de um *Firmware* para gerenciar as conexões de entrada e saída, converter os dados para um formato adequado e, por fim, enviar os dados para as próximas camadas. Sendo assim, neste experimento, utilizamos microcontroladores e mini-PCs com a responsabilidade de abstrair a complexidade do *hardware*. Na Tabela 5.1 são listadas as plataformas utilizadas para esta finalidade.

Tabela 5.1 Plataformas utilizadas na Camada de Abstração de *Hardware*.

Dispositivo	**Arquitetura**
Arduino Uno	ATmega328P
Arduino Mega	ATmega2560
Raspberry Pi 2 Model B	ARMv7

Além destes recursos discutidos nesta seção, outros componentes merecem destaque especial. Deste modo, a Seção 5.3.1.1 apresenta, em detalhes, os sensores utilizados na elaboração do cenário experimentado, a Seção 5.3.1.2 descreve a estrutura e as tecnologias utilizadas para o desenvolvimento do robô doméstico e na Seção 5.3.1.3 apresentamos os demais recursos, como câmeras IP, *smartwatch*, entre outros dispositivos incorporados no experimento.

5.3.1.1 Sensores

Com o intuito de sensoriar os fenômenos do ambiente, foram utilizados sensores para as mais diversas possibilidades. Os principais sensores estão listados na Tabela 5.2. Estes sensores foram distribuídos pelo ambiente de acordo com as funcionalidades que se pretende alcançar.

Tabela 5.2 Sensores utilizados na Camada de *Hardware*.

Sensor	**Função**
LM35	Temperatura
LDR	Luminosidade
OBD-II	Diagnóstico do veículo
HC-SR04	Distância
MPU-6050	Acelerômetro
YL-69	Umidade do solo
YL-83	Sensor de chuva
DHT-11	Temperatura e umidade do ar

Para que alguns dos sensores listados na Tabela 5.2 funcionem de modo mais adequado e simplificado, é possível utilizar módulos (*shield*) com circuitos reduzidos. A Tabela 5.3 apresenta a lista de *shields* aplicados, relacionando-os com a função do mecanismo.

Tabela 5.3 *Shields* utilizados na Camada de Abstração de *Hardware*.

Shield	**Função**
Relé	Controle de carga
Ethernet Shield W5100	Conexão com rede Ethernet
HC-05	Conexão com rede Bluetooth
OBD-II (ELM327)	Acesso aos dados de veículos

5.3.1.2 Robô Doméstico

Além dos dispositivos já apresentados, desenvolvemos um artefato que possui mecanismos capazes de proverem interação de alto nível com os habitantes, denominado de Rosie. Rosie é um robô de uso doméstico, que simula situações de emoção (virtualmente, movimentando os olhos e a boca), para servir de interface principal entre os habitantes e a residência. Esta seção descreve detalhes da sua estrutura.

O robô utiliza sensores para perceber o ambiente (temperatura, umidade, obstáculos, gestos dos habitantes, entre outros), interagindo por fala e gestos, através de um sensor RGB-D com microfones, além de possuir motores de corrente contínua que fazem com que seja possível ele se locomover pelo ambiente.

O robô possui um computador acoplado que faz todo o pré-processamento, publicando e consumindo apenas as informações que demandem baixo tráfego de rede, pois dados mais pesados poderiam criar um gargalo na rede atrasando os processos e consequentemente diminuindo a qualidade da experiência.

A integração do robô com a residência é feita utilizando os serviços do sistema. Rosie fica em constante comunicação com o SO, monitorando o estado de saúde dos habitantes, pois ao identificar indícios de que há problemas com o usuário alertas são emitidos.

Quanto ao *hardware* utilizado na montagem do robô, é relatado, inicialmente, o esquema de ligação do sistema de locomoção do robô. A Figura 5.17 ilustra esta ligação e os componentes utilizados.

Como apresentado na Figura 5.17, foi utilizado uma placa Arduino Uno (como microcontrolador), um *driver* de motor L298N, conhecido como Ponte H (como controlador dos motores) e, por fim, dois motores de corrente contínua. Para alimentar exclusivamente os motores, foi utilizado ainda uma bateria recarregável de 12V. Para a ligação física dos componentes foram utilizados fios *jumpers*, adequados para prototipação. A Tabela 5.4 organiza os componentes descritos e suas informações técnicas.

A função do microcontrolador é gerenciar o funcionamento e fornecer a lógica necessária para a locomoção correta no ambiente, utilizando os dados dos 4 sensores ultrasônicos e do

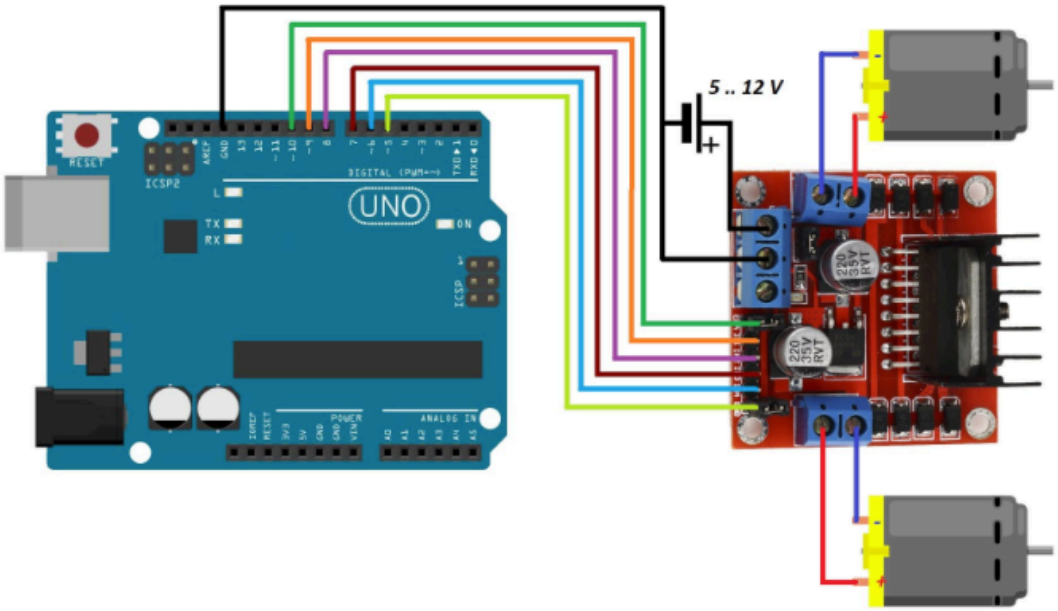


Figura 5.17 Esquema de ligação do sistema de locomoção do robô.

Tabela 5.4 Componentes utilizados para o sistema de locomoção do robô.

Componente	Modelo	Características
Microcontrolador	Arduino Uno	ATmega328P, 16MHz, tensão de 5v
Driver de motor	Ponte H - L298N	5v, 0-36mA, 2A por canal
Motores DC	12V, 80 RPM	3Kg/cm (com caixa de redução)
Rodas	Diâmetro de 5”	2 polipropileno e 1 *castor wheel*
Sensor ultrasônico	HC-SR04	2cm ~ 4m, ângulo de 15° e precisão 3mm
Bateria	Recarregável	Tensão 12V e Capacidade 7000mA

sensor RGB-D para evitar colisões com pessoas ou objetos. A Ponte H utilizada é baseada no *chip* L298N, tendo como principal função o controle de cargas, que no caso do robô foi utilizado para gerenciar a velocidade de rotação dos 2 motores de corrente direta de forma independente. A Figura 5.18 apresenta o protótipo do robô desenvolvido.

Quanto aos componentes responsáveis pela interação, utilizamos um *tablet* PC representando o rosto do robô, bem como sendo responsável pelo processamento de áudio e vídeo do sensor RGB-D. A Tabela 5.5 detalha os componentes utilizados para esta finalidade.

Tabela 5.5 Componentes do módulo de interação do robô.

Componente	Modelo	Características
Tablet PC	Microsoft Surface Pro	Processador Intel Core i5, 4 GB RAM
Sensor RGB-D	Microsoft Kinect for Xbox 360	Sensor de Profundidade, Câmera RGB, Microfones
Bateria	Unipower	Tensão 12V e Capacidade 1300mA

O computador está para o robô como o cérebro está para o humano, centralizando todo o processamento e informações dos sensores. O computador possui um serviço MQTT cliente, para envio/recebimento de mensagem ao servidor e o *Microsoft Kinect SDK v2.0 for Windows*

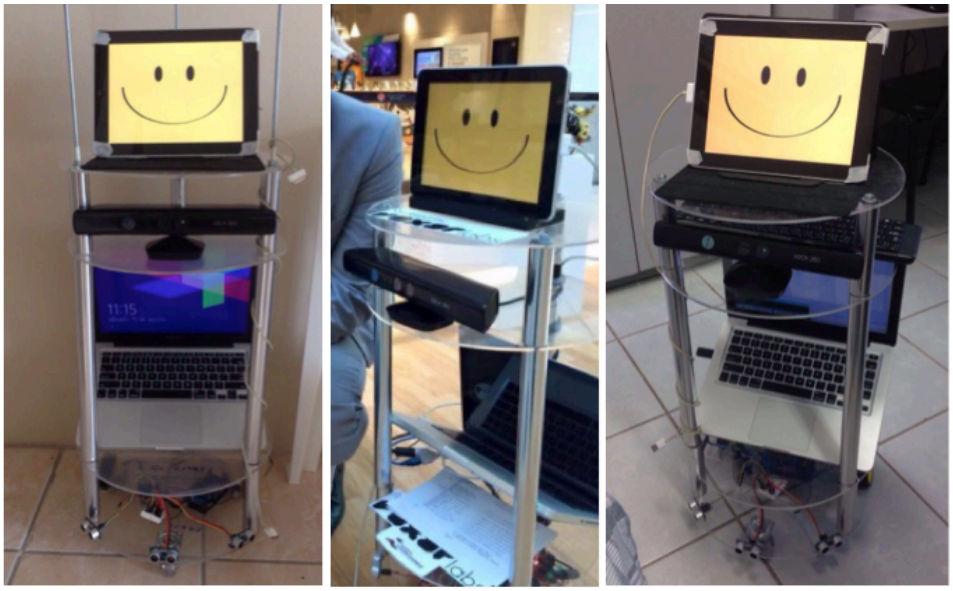


Figura 5.18 Protótipo do robô (Rosie).

para o gerenciamento do *Kinect*. O programa controlador do robô foi desenvolvido em *C#* e possui duas conexões principais, uma serial que gerencia a comunicação com o sistema de locomoção (movimentar a câmera) e outra por WiFi, responsável por se comunicar com o servidor, para enviar e receber informações e comandos.

5.3.1.3 Recursos Extra

Além dos dispositivos já apresentados, utilizamos outros dispositivos para coletar mais dados sobre os elementos que existem em uma residência. Entre estes, escolhemos integrar os dados de um veículo à residência, ou seja, sempre que o veículo entrar na zona de cobertura, ele será reconhecido e iniciará o processo de envio dos dados para o sistema.

Devido à grande variedade de informações do veículo, foram escolhidas algumas informações mais promissoras para realizar os testes preliminares. Para tanto, foi utilizado um sensor com protocolo OBD-II (*On-Board Diagnostic*) com comunicação *wireless* padrão 802.11a/b/g, e com interpretador ELM327. O OBD-II permite acessar os dados do veículo em tempo de execução. A Tabela 5.6 descreve o conjunto de dados capturados, utilizando o modo 01 do sensor, bem como os códigos necessários para obtê-los. Apesar de termos utilizado mais comandos do que os relacionados na Tabela 5.6, acreditamos que estes sejam suficientemente representativos para o entendimento do processo.

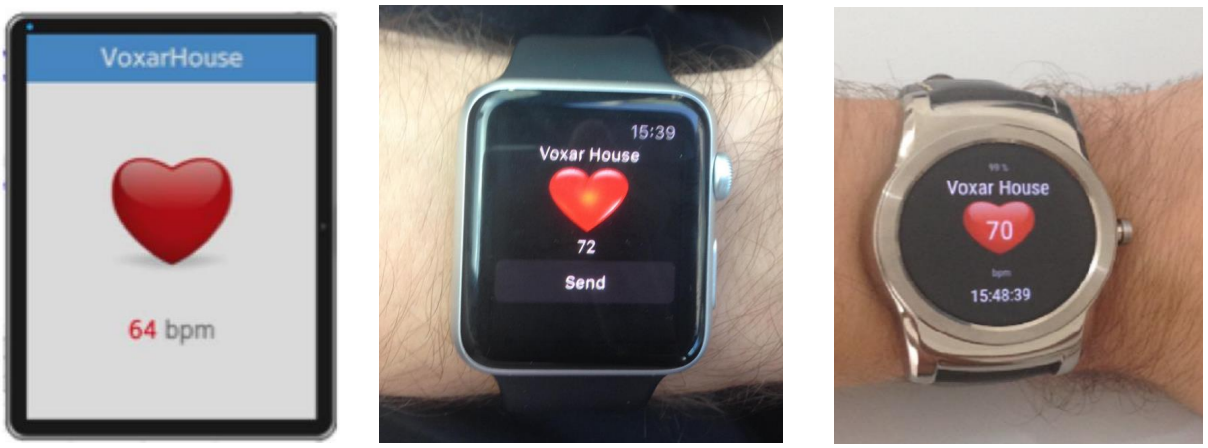
Como apresentado na Tabela 5.6, para receber dados do nível de combustível, bastaria enviar o PID 2F com o modo 01, para que fosse retornado um *byte*, que ao ser calculado e convertido para decimal indicaria a porcentagem de combustível contido no reservatório. Os dados retornados pelo sensor, quando obtidos com sucesso, chegam em formato de *bytes*, que recebe tratamento e filtra os *bytes* referentes aos dados solicitados.

Outro exemplo de recurso extra particularmente útil é o *smartwatch*. Foi desenvolvido

Tabela 5.6 Exemplos de comandos do sensor OBD-II.

Descrição	PID	Retorno	Und.	Exemplo
Temperatura	05	1	°C	92
RPM do motor	0C	2	RPM	850
Velocidade	OD	1	Km/H	38
Nível de combustível	2F	1	%	52

um aplicativo (que monitora dados de saúde do paciente, enviando-os para o SO), em três plataformas diferentes, para o sistema operacional *Tizen*[3] (Figura 5.19 (a)), amplamente adotado pela *Samsung* em seus relógios (por exemplo, linha *Galaxy Gear*), outro desenvolvido para *Watch OS*[4] (Figura 5.19 (b)), sistema utilizado no *Apple Watch* e, por fim, desenvolvemos uma versão para ser executada por dispositivos *Android Wear* (Figura 5.19 (c)). Em todos os casos o aplicativo envia os dados coletados para a Camada de Decisão do SO, sempre que ocorrer uma mudança de estado, bem como recebe notificações de emergências, tais como indicação de incêndio e alarme disparado. Para tanto, é necessário estar com uma conexão com a *internet* sempre ativa no dispositivo. A Figura 5.19 apresenta as interfaces gráficas das aplicações desenvolvidas.



(a) Aplicação Tizen. (b) Aplicação WatchOS. (c) Aplicação Android Wear.

Figura 5.19 Aplicativos desenvolvidos para *smartwatch*.

A aplicação utilizada para este experimento foi a desenvolvida em *Android Wear* e o dispositivo *smartwatch* adotado para este estudo é um *LG Urbane W150*, que utiliza o sistema operacional *Android Wear* 1.4. Ele pode se conectar a redes WiFi sem precisar de um *smartphone*, desde que esteja em uma área de cobertura de um ponto de acesso/estação base *wireless*. Sua capacidade de bateria é 410mAh, opera em uma frequência de 1200Mhz, e possui um processador *Qualcomm Quad Core de 1.2GHz*.

[3]Desenvolvido com o *Tizen IDE* e executado sobre um dispositivo virtual, emulado. Os valores dos sensores foram simulados sinteticamente.

[4]Desenvolvido com o Xcode, na linguagem *Swift*, utilizando o SDK do *WatchOS* e a *API WatchKit*. É executado em um dispositivo real, o *Apple Watch Sport* de 42mm.

Alguns dos recursos discutidos nesta subseção eventualmente podem ficar fora do alcance da residência ou possuem mecanismos de gerenciamento de conexão próprios, independentemente da estrutura residencial interna. Entretanto, sempre que o dispositivo (por exemplo, veículo, relógio) retorna à área de cobertura este é novamente detectado e o processo de comunicação é reestabelecido, retomando a troca de informação normalmente.

5.3.2 Comunicação de Rede

A comunicação de rede é bastante relevante para o funcionamento adequado do sistema, pois boa parte do tempo a informação está trafegando por esta camada. Neste cenário experimentado, nós iremos apresentar como os dados trafegam entre os dispositivos (i.e. entre o robô e os demais dispositivos conectados). A Figura 5.20 apresenta um exemplo desta comunicação.

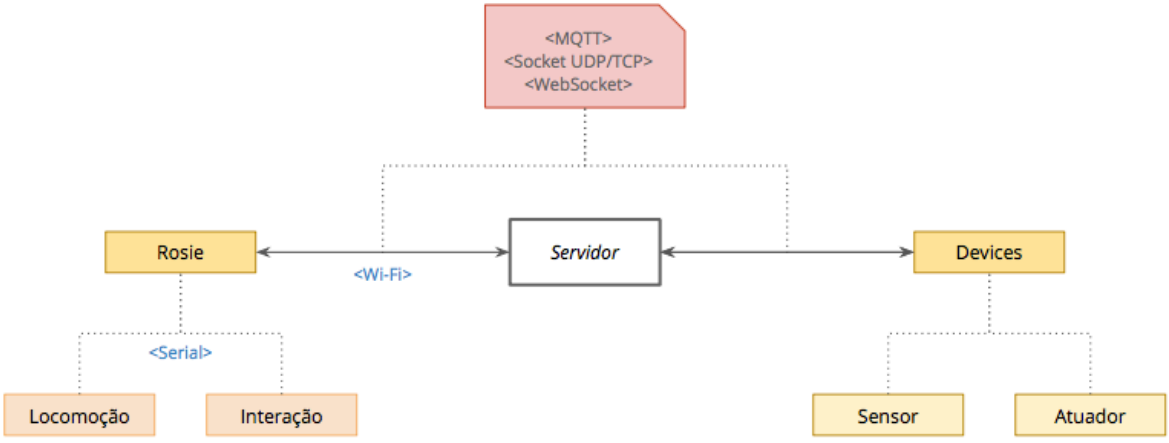


Figura 5.20 Exemplo de comunicação entre o robô e o sistema operacional.

Como apresentado anteriormente, o robô é subdividido em dois componentes principais, sendo estes locomoção e interação. Estes componentes trocam informações através de uma interface serial com o computador responsável pelo processamento das informações. Nesta seção, nós consideramos que o robô é um dispositivo como outro qualquer, sendo importante destacar apenas como ocorre a comunicação com os demais dispositivos.

Portanto, as informações coletadas pelo robô são trocadas com o SO através da interface de rede WiFi do computador. Para tanto, foi implementado um cliente do protocolo MQTT, que é responsável por gerenciar a troca das mensagens entres estes componentes.

Quando um evento detectado pelo robô está relacionado a algum dispositivo, o sistema envia uma mensagem MQTT para o dispositivo realizar a tarefa. O Código 5.2 ilustra um exemplo de envio de mensagem MQTT.

Código 5.2 Exemplo de Publisher MQTT.

```
1 import paho.mqtt.client as mqtt
2 import socket
```

```
3  import uuid
4  import random
5  from struct import pack
6  from time import sleep
7
8  SERVER = '10.1.1.10'
9  PORT = 1883
10
11 IP_ADDRESS = socket.gethostbyname(socket.gethostname())
12 MAC_ADDRESS = hex(uuid.getnode()).replace('0x', '').upper()[0:12]
13 CLIENT_ID = 'VoxarHome_%s' % (MAC_ADDRESS)
14 TAG_VOXAR = "$VOXAR/"
15
16 def publisher(topic, msg):
17     client.publish(topic, msg, qos=0)
18     sleep(2)
19     return
20
21 TOPIC = "sensor/%s/%s/Simulator/temperature" % (IP_ADDRESS,
       MAC_ADDRESS)
22
23 client = mqtt.Client(CLIENT_ID, protocol = mqtt.MQTTv31)
24 client.connect(SERVER, PORT)
25
26 publisher((TAG_VOXAR + TOPIC), "0.1")
27
28 while True:
29     publisher(TOPIC, random.uniform(10.0, 30.0))
```

No Código 5.2 o formato da mensagem publicada no tópico é *"sensor/<ip>/<mac>/<hostname>/<parâmetro>"*, de modo que o código de exemplo ilustra apenas o trecho referente a publicação de um único fenômeno (temperatura), podendo este ser utilizado para qualquer outro tipo de dado.

5.3.3 Interação Multimodal

No cenário experimentado, a interação por gestos e voz é possível ser realizada através do robô, que utiliza o sensor Kinect. Contudo, vale salientar que em um cenário extrapolado o uso de mais sensores que possam capturar esse tipo de informação pode ser aplicado.

Portanto, a principal forma de interação entre o robô e os habitantes é através da fala, porém, foi implementado também um reconhecedor de gestos para auxiliar na comunicação. Estes módulos são executados pelo computador anexado ao robô, que utiliza o *Microsoft Kinect* para realizar a percepção do ambiente. Para tanto, foi feito uso do *Microsoft Kinect SDK for Windows v2.0*.

Outro aspecto importante é que devido ao fato do Kinect possuir 4 microfones e câmera RGB, além do sensor de profundidade, é possível utilizá-lo para diversos fins. Um exemplo do uso da combinação desses microfones pode ser quando o usuário não está no alcance visual das câmeras do sensor, e o usuário interage verbalmente com o robô, sendo necessário identificar a direção da voz do interlocutor para otimizar a interação. Deste modo, o robô irá enquadrar o usuário em seu campo visual, podendo assim observar possíveis ações do mesmo.

Na Seção 5.3.3.1 apresentaremos as tecnologias utilizadas para o desenvolvimento do mecanismo de interação por voz e na Seção 5.3.3.2 para a interação por gesto.

5.3.3.1 Interação por Voz

Em relação à fala, o sistema é dividido em reconhecimento e síntese. Enquanto para executar o método de síntese da voz precisamos apenas de uma *string* (texto) qualquer que será reproduzida (*text-to-speech*), para reconhecer a voz podemos criar uma gramática finita para facilitar o reconhecimento das palavras relacionadas sem a necessidade de treinamento prévio. Neste estudo, é utilizado o *Microsoft Speech Platform SDK 11*[5], a mesma utilizada no sistema operacional *Windows*, tanto para reconhecimento como para síntese.

Foi desenvolvido o módulo de interação por voz na linguagem C#, onde através de métodos específicos é possível enviar uma variável do tipo texto e a saída é a síntese da voz, sendo reproduzida pelo alto-falante do dispositivo. Este mecanismo é utilizado, principalmente, quando o sistema ou o robô quer passar alguma informação ou confirmar alguma situação detectada, tais como informar a situação do clima para o usuário ou ao detectar alguma situação atípica, como a elevação dos batimentos cardíacos, perguntar se está tudo bem ou se precisa de ajuda. O Código 5.3 apresenta trecho da implementação do mecanismo de síntese de voz.

Código 5.3 Síntese de voz em C#.

```
1 using System.Speech.Synthesis;
2   // ...
3   public SpeechSynthesizer rosie;
4     //...
5     rosie.SelectVoice(voices[idLang]);
6     rosie.Rate = 0;
7     rosie.Volume = 100;
8     //...
9     rosie.Speak("Texto de exemplo");
```

No Código 5.3 a linha 1 é utilizada para referenciar a dependência da biblioteca utilizada; nas linhas 5, 6 e 7 a voz utilizada é parametrizada (interfere diretamente no idioma), a velocidade da fala e o volume de saída no alto-falante, respectivamente. Por fim, na linha 9, ocorre a

[5]O *Microsoft Speech Platform SDK 11* fornece um conjunto completo de ferramentas de desenvolvimento para a criação de aplicativos habilitados para voz que utilizam mecanismos de fala redistribuíveis da *Microsoft* (MICROSOFT, 2016b).

reprodução propriamente dita, transformando o texto em áudio. Na interface VoxarHome Web, enquanto o texto é reproduzido, escolhemos utilizar uma legenda para facilitar a compreensão por pessoas que possuam problemas de audição ou mesmo em casos onde há dificuldade de ouvir o diálogo, por diversos motivos.

O processo de reconhecimento de voz é mais complexo que o de síntese de voz e é, comumente, utilizado para que o usuário possa efetuar comandos por voz no ambiente. No cenário avaliado, foi utilizada a mesma API de reconhecimento de voz utilizada no Microsoft Windows. O Código 5.4 apresenta trechos resumidos do processo de reconhecimento.

Código 5.4 Reconhecimento de voz em C#.

```
1  using Microsoft.Speech.Recognition;
2    // ...
3    double confidence = 0.7;
4    //...
5    speechEngine.LoadGrammar(rosieGrammar);
6    //...
7    private void SpeechRecognized(object sender,
       SpeechRecognizedEventArgs eventSpeech){
8      if (eventSpeech.Result.Confidence >= confidence){
9        speech(eventSpeech.Result.Semantics.Value.ToString());
10     }
11   }
```

No Código 5.4 a linha 1 é utilizada para referenciar a dependência da biblioteca utilizada; na linha 3 declaramos e inicializamos uma variável que será utilizada para definir o grau de confiança do classificador; na linha 5 é instanciada a gramática utilizada pelo reconhecedor; na linha 7 temos o método que é chamado sempre que identificado um evento de voz; na linha 8 é verificado se o resultado do classificador satisfaz o grau de confiança estabelecido anteriormente; por fim, na linha 9, chamamos o método *speech*, passando como parâmetro o valor da chave que foi estabelecida na gramática.

Destaca-se que tanto o Código 5.4 quanto o Código 5.3 são fragmentos de código retirados de um contexto mais amplo e adaptados para realçar os pontos mais importantes para o entendimento dos respectivos processos, assim como os valores dos parâmetros utilizados foram definidos como valores fixos, mas que no ambiente de execução são variáveis definidas pelo usuário através do componente *Interaction Engine* da Camada de Interação (ilustrada na Figura 4.1).

5.3.3.2 Interação por Gesto

Para facilitar a interação do robô com os habitantes da residência foi desenvolvido um módulo de reconhecimentos de gestos, que é utilizado para definir algumas tarefas a serem realizadas pelo robô ou pelo sistema gerenciador da residência.

A interação por gesto é um processo que predominantemente requer técnicas de rastreamento e reconhecimento dos gestos efetuados pelo usuário. Logo, a próxima etapa é coletar as articulações do esqueleto da pessoa extraídas pelo sensor, representadas por um conjunto de pontos em 3D, para representar a pose momentânea do usuário. A Figura 5.21 ilustra o processo de reconhecimento de gestos.

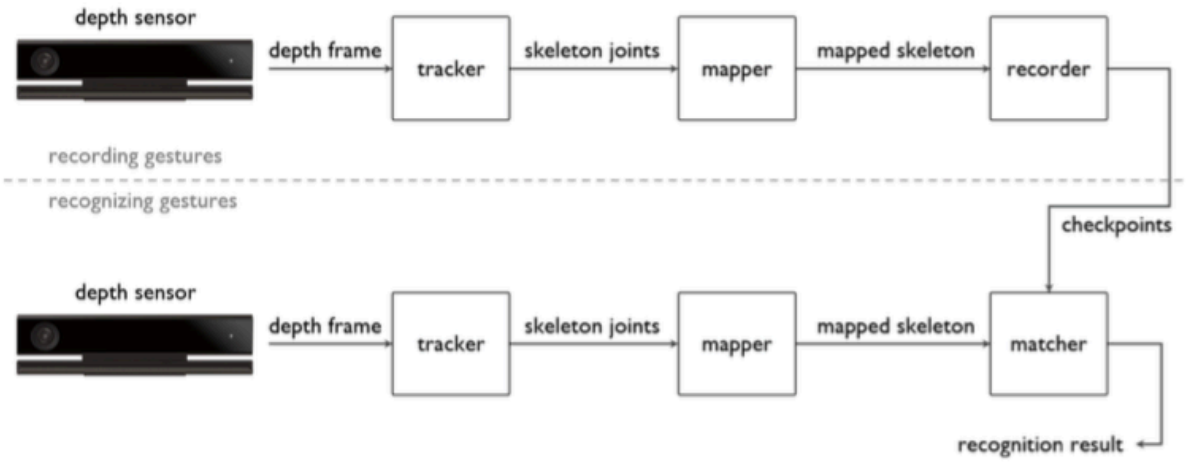


Figura 5.21 Etapas da gravação e do reconhecimento de gestos.

Considerando que um gesto pode ser estático (caracterizado por uma pose, ou seja, uma posição estática) ou dinâmico (caracterizado como uma sequência finita de poses em um espaço de tempo), é possível indicar que o gesto é uma atividade realizada com o corpo, com o objetivo de transmitir uma mensagem. Logo, para reconhecer um gesto completo é necessário observar o processo de mudança de representação do esqueleto.

Para tanto, foi adotada a ferramenta proposta por FIGUEIREDO et al. (2014), denominada Prepose. Um fator que foi crucial para a escolha da ferramenta é a possibilidade dos usuários poderem definir seus próprios gestos. O Código 5.5 ilustra um exemplo de uso da linguagem Prepose.

Código 5.5 Exemplo de uso da linguagem Prepose.

```
1  GESTURE crossover-left-arm-stretch:
2  POSE relax-arms:
3    point your left arm down,
4    point your right arm down.
5  POSE stretch:
6    rotate your left arm 90 degrees counter clockwise on the frontal
       plane,
7    touch your left elbow with your right hand.
8  EXECUTION:
9    relax-arms,
10   slowly stretch and hold for 30 seconds.
```

O Prepose foi modificado para se adequar à arquitetura proposta no Capítulo 4. Foi adicionada uma camada de rede, onde foi implementado um *Socket*. Deste modo, é enviada uma mensagem para o VoxarHomeOS sempre que for reconhecido um gesto descrito, esta informação será enviada para a Camada de Decisão, que entenderá a mensagem como uma variação de um sensor e será tratado como os demais sensores.

5.4 Considerações Finais

Esta seção busca apresentar uma discussão sobre a experimentação dos cenários estudados. Inicialmente, avaliamos se os requisitos da arquitetura proposta foram atendidos no desenvolvimento do protótipo, verificando se os componentes da arquitetura contemplam tais requisitos, utilizando uma abordagem vertical, abrangendo diversos níveis e camadas (Seção 5.2). Neste contexto, utilizando o protótipo de SO, elaboramos 6 estudos de caso. Na Tabela 5.7 são apresentados os requisitos abordados em cada estudo de caso.

Tabela 5.7 Mapeamento dos requisitos abordados nos estudos de caso.

Requisitos	**Estudo de Caso I**	**Estudo de Caso II**	**Estudo de Caso III**	**Estudo de Caso IV**	**Estudo de Caso V**	**Estudo de Caso VI**
Interoperabilidade	✓	✓	✓	✓	✓	✓
Escalabilidade						
Extensibilidade	✓	✓	✓	✓	✓	
Descoberta	✓	✓	✓	✓	✓	✓
Contexto	✓	✓	✓	✓	✓	✓
Análise						✓
Interação		✓	✓	✓		✓

O requisito de Interoperabilidade pode ser observado em todos os estudos de caso realizados, visto que utilizamos dispositivos heterogêneos e que estes compartilharam dados e recursos, independente de arquitetura, protocolo ou *design* dos mesmos.

A Escalabilidade é um requisito que não foi possível validar através dos estudos de caso mencionados. Para avaliar a escalabilidade é necessário observar o uso de uma quantidade maior de dispositivos e como o sistema consegue lidar com esse crescimento sem degradar os recursos disponíveis. Portanto, na Seção 6.4 realizamos um experimento com o intuito de avaliar este requisito, quantificando-o.

A Extensibilidade foi observada nos Estudos de Caso I, II, III, IV e V, sendo que nos Estudos de Caso I, II e III a extensão se deu através de uma interface de alto nível, onde o usuário utiliza recursos existentes no sistema para elaborar seus algoritmos de forma visual, sem necessidade de preocupação com aspectos relacionados à programação de *software*. Por outro lado, nos Estudos de Caso IV e V avaliamos a extensibilidade sob uma perspectiva mais baixo nível, onde se faz necessário um conhecimento prévio de linguagem de programação. No Estudo de Caso IV, utilizamos a API desenvolvida para: localizar os sensores *"Microphone"* e *"Time"*;

criar um evento para cada sensor localizado; criar uma regra, associando os dois eventos ao atuador *"Speaker"*. No Estudo de Caso V, foi desenvolvida uma aplicação para um *smartwatch*, para controlar o estado de uma lâmpada.

A Descoberta é um requisito presente em todos os estudos de caso realizados, pois para elaborar os cenários avaliados é necessário identificar os dispositivos conectados na rede, bem como detectar quando estes podem não estar mais operantes. O componente responsável por suprir este requisito é executado ininterruptamente, atualizando o *status* dos dispositivos.

O requisito Sensibilidade ao Contexto foi observado em todos os estudos de caso realizados. Para elaborar uma regra para executar alguma tarefa no ambiente, muitas vezes, o sistema precisa estabelecer relações entre diversos dispositivos que possuem informações de contextos distintos. Deste modo, a execução da tarefa pode depender de diversas informações, sendo necessário possuir conhecimento do contexto.

O Estudo de Caso VI foi utilizado para avaliar o requisito Análise de Dados. Foi observado neste estudo de caso se os componentes que devem suprir este requisito foram capazes de identificar corretamente a semântica de um determinado gesto (no caso, o gesto *"Hello"*). Neste caso, utilizamos um *dataset* com baixa variância, reduzindo a capacidade de generalização do aprendizado. O enfoque deste estudo de caso foi validar os componentes da arquitetura proposta, verificando se os requisitos foram atendidos, não sendo relevante a precisão da classificação. Por outro lado, na Seção 6.6 apresentamos um experimento que teve como objetivo avaliar o tempo e a precisão do aprendizado.

O requisito de Interação Natural foi observado nos Estudos de Caso II, III, IV e VI. De modo que, nos Estudos de Caso II e VI, foram utilizados os componentes responsáveis por permitirem a definição de gestos de forma descrita, por meio de uma linguagem de alto nível. No Estudo de Caso IV, utilizamos os componentes responsáveis por permitirem a definição de comandos de voz. O Estudo de Caso III, apresentou o uso da interação por gesto e voz de forma complementar, sendo possível acender uma lâmpada realizando um gesto ou falando o comando de voz definido. No Estudo de Caso I, não consideramos que o requisito Interação foi representado, pois apesar da tarefa (ligar a TV) poder ser executada através do deslocamento do usuário à frente da TV, o que poderia ser considerada uma atividade do usuário, tratamos como interação natural por gesto apenas as tarefas que podemos identificar o corpo humano de forma segmentada e não apenas como um bloco único, como ocorre neste estudo de caso. Além disso, o sensor utilizado no Estudo de Caso I não é capaz de distinguir uma pessoa de um objeto, identificando apenas a distância.

A primeira etapa da avaliação teve como enfoque os requisitos que a arquitetura deveria respeitar. A segunda etapa teve ênfase na avaliação da arquitetura sob a perspectiva dos componentes. Deste modo, na Tabela 5.8 são apresentadas as camadas que foram abordadas nos estudos de caso.

Através da análise dos estudos de caso discutidos na Seção 5.2, podemos destacar as camadas que foram abordadas em cada estudo de caso. As camadas de *Hardware*, Abstração

Tabela 5.8 Mapeamento das camadas abordadas pelos estudos de caso.

Camadas da Arquitetura	Estudo de Caso I	Estudo de Caso II	Estudo de Caso III	Estudo de Caso IV	Estudo de Caso V	Estudo de Caso VI
Hardware	✓	✓	✓	✓	✓	✓
Abstração de Hardware	✓	✓	✓	✓	✓	✓
Rede	✓	✓	✓	✓	✓	✓
Decisão	✓	✓	✓	✓	✓	✓
Análise						✓
Interação		✓	✓	✓		✓
Serviço	✓	✓	✓	✓	✓	✓
Aplicação				✓	✓	

de *Hardware*, Rede, Decisão e Serviço foram exploradas por todos os estudos. A Camada de Análise de Dados foi representada apenas no Estudo de Caso VI. A Camada de Interação foi observada nos Estudos de Caso II, III, IV e VI. A Camada de Aplicação pode ser observada nos Estudos de Caso IV e V, onde foram desenvolvidas soluções independentes do protótipo de SO.

Na Seção 5.3 exploramos uma abordagem horizontal por camadas, com foco nos componentes. Este cenário permitiu identificar como os componentes podem ser estendidos para desenvolver aplicações, provendo suporte a interação natural. Foi possível observar o funcionamento dos dispositivos como o relógio inteligente, o robô, o veículo, entre outros. No Capítulo 6 os recursos experimentados na Seção 5.3 são avaliados, no intuito de destacar métricas para serem utilizadas como referência em futuras implementações.

6

AVALIAÇÃO E RESULTADOS

Neste capítulo apresentaremos uma avaliação quantitativa em relação aos principais requisitos do sistema operacional. Desta forma, analisamos o desempenho de modelos de plataformas de hardware, meios de transmissão, bem como o tempo médio até a descoberta de um dispositivo que entrou ou saiu da rede. Foram investigados ainda aspectos que referem-se a escalabilidade, extensibilidade, assertividade das regras e do aprendizado adquirido com base no reconhecimento de padrões. Ainda neste capítulo, com intuito de validar os dados coletados, apresentamos a validação dos experimentos realizados utilizando o método Bootstrap. Por fim, analisamos os mecanismos de interação natural sob os aspectos de assertividade e performance.

As próximas seções apresentam os experimentos realizados no intuito de avaliar, quantitativamente, os requisitos elencados nos capítulos anteriores. Inicialmente, na Seção 6.1, apresentamos o método de validação dos dados. Na Seção 6.2, listamos os principais recursos de *hardware* experimentados, no intuito de verificar a compatibilidade entre o SO e os dispositivos analisados. Na Seção 6.3, descrevemos o experimento conduzido para avaliar dois fatores importantes: tempo médio de resposta e tempo médio até a descoberta de um dispositivo na rede. Na Seção 6.4, avaliamos a escalabilidade do sistema, sob a perspectiva da performance. Nesta seção utilizamos ferramentas para simular um crescimento significativo da rede, monitorando o uso de memória e processamento demandado pelo computador servidor. Na Seção 6.5, observamos o tempo médio para a execução de uma regra pré-definida e a precisão das decisões. Para tanto, variamos os tipos de dados e tamanho das mensagens trocadas entre os dispositivos e o SO. Na Seção 6.6, delineamos um cenário onde aplicamos o experimento, descrevemos os parâmetros e *dataset* utilizados e analisamos o comportamento do sistema quanto a extração e classificação dos padrões, a fim de formalizar uma regra aprendida. Na Seção 6.7, apresentamos um experimento quantitativo quanto à assertividade do reconhecimento de voz e gesto, realizado por um usuário. Por fim, na Seção 6.8, descrevemos o experimento realizado com voluntários, que se dispuseram a utilizar o sistema, como forma de avaliar a complexidade de utilizar os principais recursos do sistema.

6.1 Validação dos Dados Coletados

Para garantir a validade dos dados coletados, realizamos uma validação que consiste em comparar os resultados do experimento real com os resultados de uma simulação. Tal comparação deve seguir alguns critérios estáticos, e em nosso trabalho adotamos a técnica *Bootstrap* (EFRON; TIBSHIRANI, 1994).

O método *Bootstrap*, proposto por EFRON; TIBSHIRANI (1994), consiste em aplicar um esquema de reamostragem de uma amostra original x, a fim de realizar inferências sobre uma estatística através de sua distribuição aproximada obtida a partir das reamostras x^*.

Assim, o uso de técnica de reamostragem é uma ótima alternativa para obtermos aproximações de distribuições amostrais, que se baseiam em calcular estimativas a partir de repetidas amostragens dentro da mesma amostra. A reamostragem *Bootstrap* visa simular diversos cenários a partir de uma pequena amostra, desta forma o método constrói um espaço amostral e realiza inferências sobre os parâmetros de interesse. Além disso, uma amostra maior fornece uma estimativa mais precisa.

Para que os dados coletados sejam considerados válidos, é necessário que este esteja dentro do respectivo intervalo de confiança. Uma vez validados os dados, futuros pesquisadores poderão utilizar nossos dados como parâmetros base para estudos científicos. Em nosso estudo está sendo considerado a média, em cada reamostra que foi gerada da amostra original.

Quanto aos dados coletados, após o processo do experimento ser repetido n vezes para cada dispositivo, obtivemos um *log* de cada procedimento. Considerando estas amostras, obtivemos as médias para as métricas de interesse. Os resultados seguiram uma distribuição normal e depois geramos 1000 valores, extraindo o intervalo de confiança *Bootstrap* (EFRON; TIBSHIRANI, 1994).

Para obter um intervalo de confiança, considerando um índice de 95% com erro máximo de 5%, consultamos os $0,025$ e $0,975$ quantis dos dados obtidos de 1000 reamostras geradas pelo *Bootstrap*, pois segundo CHERNICK (2011), o número de reamostragens necessárias para se obter boas estimativas seria, de pelo menos, 1000 repetições *Bootstrap*. A reamostragem do *Bootstrap* ocorreu através da elaboração de um *script* (conforme Código 6.1) na ferramenta *Mathematica* (WOLFRAM RESEARCH, 2016).

Código 6.1 Script Bootstrap

```
1 data = {$x_1$, $x_2$, $x_3$, ...,$x_n$}
2 D= EmpiricalDistribution[data];
3 estimated$D$ = FindDistribution[data];
4 alpha = 0.05;
5 numberofresamples = 1000;
6 MeanData = Table[Mean[RandomChoice[data, Length[data]]],{
      numberofresamples}];
7 Print["MeanData=", MeanData]
8 Print["MeanQuantile= ", Quantile[MeanData, {alpha/2, (1-alpha/2)}]]
```

6.2 Análise de *Hardware*

O VoxarHomeOS tem como premissa ser compatível com o maior número possível de dispositivos de rede - que possua a função de sensoriar e agir no ambiente. Deste modo, realizamos experimentos com alguns[1] dispositivos, com o objetivo de verificar a compatibilidade com a arquitetura proposta. Os dispositivos avaliados possuem características similares, tais como: a possibilidade de uso de linguagem de programação, interface para comunicação em rede e a capacidade de adição de sensores e atuadores. Entretanto, classificamos de acordo com o suporte a execução de um sistema operacional em: microcontrolador e mini-PC. Onde classificamos como microcontrolador, o dispositivo que não possui suporte a instalação de um sistema operacional, sendo o mini-PC, o dispositivo capaz de executar tal software.

A Tabela 6.1 apresenta uma descrição dos mini-PCs utilizados. Todos os mini-PCs utilizados possuem conexão Ethernet e WiFi, deste modo, não foi necessário utilizar módulos de interface de rede. O sistema operacional utilizado nestes dispositivos foi o Ubuntu Mate.

Tabela 6.1 Mini-PCs.

Nome	**SoC**	**CPU**			**GPU**	**Memória**	
		Arquitetura	**Núcleos**	**Frequência**		**Tamanho**	**Tipo**
Raspberry Pi 3 Model B	Broadcom BCM2837	ARM Cortex-A53	4	1.2 GHz	Broadcom VideoCore IV	1 GB	LPDDR2
PINE A64+ 2GB	Allwinner A64	ARM Cortex-A53	4	1.2 GHz	Mali-400MP2	2 GB	DDR3
Orange Pi One	Allwinner H3	ARM Cortex-A7	4	1.2 GHz	ARM Mali-400 MP2 @600 MHz	512 MB	DDR3

A Tabela 6.2 apresenta uma descrição dos dispositivos microcontroladores utilizados. Devido ao fato dos dispositivos listados na Tabela 6.2 não possuírem interfaces de rede embutidas, utilizamos alguns módulos, conforme apresentado na Tabela 6.3.

Tabela 6.2 Microcontroladores.

Nome	**Processador**	**Frequência**	**Portas**		**Memória [kB]**
			Analógica	**Digital**	
Arduino Nano	ATmega328P	16 MHz	8	14	32
Arduino Uno	ATmega328P	16 MHz	6	14	32
Arduino Mega	ATmega2560	16 MHz	16	54	256

Os recursos de *hardware* que foram utilizados durante pelo menos um dos experimentos estão listados na Tabela 6.4.

[1] A seleção dos dispositivos se deu por questões de acesso e disponibilidade.

Tabela 6.3 Shields.

Módulo	Conexão
W5100	Ethernet
ESP8266	WiFi
HC-05	Bluetooth

Tabela 6.4 Outros recursos de *Hardware*.

Item	Equipamento	Descrição
1	Notebook	Intel Core i5-3210M, 2.50GHz, memória de 10Gb RAM, 240 Gb de SSD, Adaptador de Rede *Wireless* compatível com IEEE 802.11a/b/g/n, Adaptador de Rede Gigabit Ethernet
2	Roteador	4 Portas LAN 10/100Mbps, Antena de 5 dBi, Padrões IEEE 802.11b/g/n, 2.4 GHz
3	Sensor de Distância	Corrente de 2mA, Ângulo de 15, Alcance entre 2cm e 4m, Precisão de 3mm
4	Módulo Relé	Cargas de 220V AC, Corrente entre 15 e 20mA, Tensão de saída de (30 VDC a 10A), Tempo de reposta entre 5-10ms
5	Notebook	Intel Core i5-4210U de 2.40 GHz, memória de 6GB RAM, HD 1Tb, Adaptador de Rede Atheros AR3012

6.2.1 Arquitetura Base

Nas seções seguintes, apresentaremos os cenários onde cada parâmetro foi avaliado, indicando sob quais circunstâncias os valores obtidos foram coletados. Entretanto, de modo geral, utilizamos uma estrutura base, conforme ilustrado na Figura 6.1.

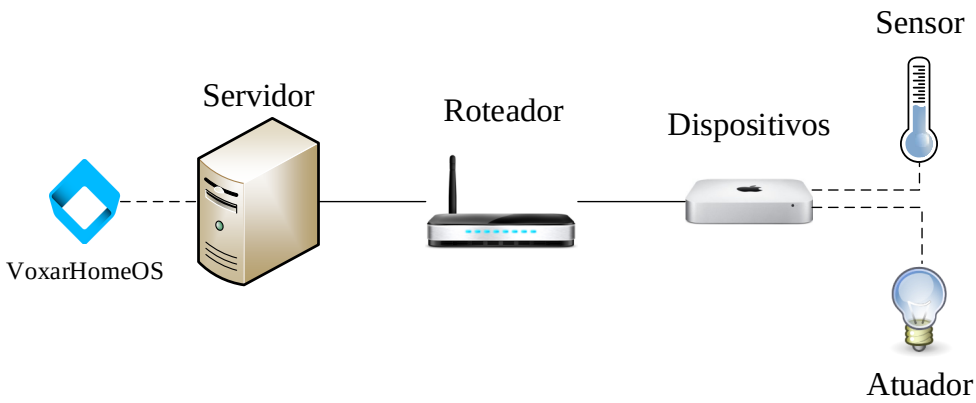


Figura 6.1 Estrutura base do experimento.

A arquitetura apresentada na Figura 6.1 representa um cenário genérico, onde, eventualmente, as funções podem ser realizadas por diferentes hardware, dentre os listados nas Tabelas 6.1, 6.2 e 6.4. No cenário geral, utilizamos os recursos listados na Tabela 6.4, tais como: o item 5 na função de servidor, o item 2 como roteador, os itens 3 e 4 como sensor e atuador, respectivamente. Na função de dispositivo, foram utilizados os recursos listados nas Tabelas 6.1

e 6.2, de acordo com o experimento realizado.

6.3 Comunicação de Rede

Um dos aspectos mais interessantes que um sistema de troca de informações em rede deve proporcionar é um tempo de resposta satisfatório, bem como deve identificar o ingresso de dispositivos na rede em um curto tempo. Deste modo, para avaliar a comunicação de rede do experimento foram utilizadas duas métricas. Na Seção 6.3.1, avaliamos o tempo de resposta médio entre os dispositivos e o sistema operacional e na Seção 6.3.2 avaliamos o tempo médio até a descoberta de um dispositivo na rede.

6.3.1 Tempo de Resposta

Neste cenário, utilizamos todos os dispositivos listados nas Tabelas 6.1 e 6.2 para a transmissão através da conexão cabeada e conexão *Wireless*. Na conexão cabeada utilizamos um cabo *Ethernet* par trançado, padrão Cat.5e, de 1,8m. Já na conexão *Wireless*, retiramos o cabo que conecta o dispositivo ao roteador e estabelecemos uma conexão utilizando os módulos de rede (Tabela 6.3), nos microcontroladores e as interfaces de rede embutidas nos mini-PCs à uma distância de 1m entre os aparelhos. Nos dispositivos foram utilizadas a linguagens *C* e Python para os microcontroladores e mini-PCs, respectivamente; o VoxarHomeOS foi executado no computador, descrito como item 5, na Tabela 6.4.

Este experimento baseou-se no tempo de duração do processo de troca de mensagens (MQTT), o qual engloba o tempo total para execução das seguintes tarefas:

1. Captura de dados do sensor;
2. Composição da mensagem a ser enviada;
3. Envio da mensagem para o servidor;
4. Recebimento da mensagem;
5. Decomposição da mensagem;
6. Atribuição adequada ao sensor correspondente;
7. Composição da mensagem de confirmação;
8. Envio da mensagem para o dispositivo requerente;
9. Recebimento da mensagem.

O tempo de resposta médio foi calculado após 30 repetições deste processo em cada um dos dispositivos analisados. Após a etapa de coleta de dados realizamos a sua validação, conforme descrito na Seção 6.1.

A Tabela 6.5 apresenta os dados obtidos utilizando a conexão *Ethernet* e a Tabela 6.6 apresenta os dados obtidos utilizando a conexão *WiFi*.

Tabela 6.5 Tempo de resposta - Ethernet.

Dispositivos	Experimento (s)
Arduino Nano	0.179433
Arduino Uno	0.160633
Arduino Mega	0.157267
Orange Pi One	0.159767
Pine 64	0.126933
Raspberry Pi 3	0.179200

Tabela 6.6 Tempo de resposta - WiFi.

Dispositivos	Experimento (s)
Arduino Nano	0.096867
Arduino Uno	0.077667
Arduino Mega	0.087800
Orange Pi One	0.094067
Pine 64	0.110967
Raspberry Pi 3	0.088900

Estes dados podem ser utilizados por futuros utilizadores, no intuito de escolher dentre as opções analisadas, qual se adéqua melhor ao cenário de uso. Contudo, neste cenário do experimento, o dispositivo que apresentou o melhor tempo de resposta em média foi o Arduino Uno. A avaliação que podemos fazer é que os dados obtidos neste experimento mostram que o tempo médio de resposta é baixo, não apresentando um atraso perceptível ao habitante. Em um ambiente residencial, estes tempos apresentados nas Tabelas 6.5 e 6.6 podem ser considerados dentro do esperado, visto que normalmente são redes de pequeno alcance e com topologias simples.

6.3.2 Descoberta em Rede

Eventualmente um dispositivo de rede pode ser desconectado voluntariamente ou não, bem como novos dispositivos podem ser adicionados ao sistema. Por este motivo, avaliamos o tempo médio que um dispositivo demora para ser reconhecido na rede, após ser plugado ou ligado.

Este experimento utilizou a mesma configuração do experimento anterior (Seção 6.3.1), excluindo apenas a variação de interface de rede. No intuito de mitigar possíveis ruídos e

interferências na comunicação, utilizamos apenas a conexão *Ethernet*, entre o dispositivo e o roteador.

Salientando que devido as diferenças de arquitetura dos dispositivos, mais precisamente, o tempo de iniciação dos mini-PCs, os tempos devem ser analisados de forma segmentada, para um melhor entendimento. Utilizamos ainda o mesmo módulo de conexão *Ethernet* nos microcontroladores, pois foi detectado que entre um módulo ocorreu uma pequena variação no tempo de obtenção do IP. Não utilizamos um IP fixo nos dispositivos para que o experimento não sofresse um viés de especificidade, visto que o objetivo deste experimento é avaliar o tempo que um dispositivo novo e desconhecido demora para ser reconhecido e integrado aos sistema. A Tabela 6.7 apresenta os dados obtidos após a realização do experimento.

Tabela 6.7 Descoberta.

Dispositivos	**Experimento (s)**
Arduino Nano	3.964
Arduino Uno	2.279
Arduino Mega	6.263
Orange Pi One	12.532
Pine 64	11.839
Raspberry Pi 3	11.974

A Tabela 6.7 apresenta tempos discrepantes entre os microcontroladores (Arduino Nano, Uno e Mega) e os mini-PCs (Orange Pi, Pine 64 e Raspberry), isso se deve ao tempo de inicialização do sistema operacional dos mini-PCs. Enquanto os microcontroladores executam os programas que estão embarcados nos seus *chips* quase de imediato, após serem ligados à fonte de alimentação, os mini-PCs precisam inicializar o SO para posteriormente iniciar a execução das aplicações.

Neste quesito, o que podemos observar é que o tempo entre o dispositivo ser ligado até ser descoberto pelo VoxarHomeOS está relacionado mais diretamente ao tempo que estes dispositivos demoram para ingressar de fato na rede. Realizando uma análise, considerando os tempos de resposta (Tabelas 6.5 e 6.6) e o tempo até a descoberta (Tabela 6.7), podemos inferir que existe uma variação nos tempos de descoberta, mas que após o dispositivo ser identificado, o tempo de resposta na comunicação entre os dispositivos e o sistema tende a não varia significativamente entre os dispositivos analisados.

6.4 Escalabilidade

Como forma de avaliar a capacidade de processar as interações de diversos dispositivos de forma simultânea e, principalmente, suportar o crescimento exponencial de dispositivos conectados, isto é, o aumento da carga de trabalho, sem afetar o desempenho do VoxarHomeOS, simulamos a inserção de diversos dispositivos na rede.

Neste experimento, criamos uma rede virtualizada, seguindo o conceito *Software Defined Networking* (SDN), para simular o comportamento de diversos dispositivos conectados à rede. Para tanto, utilizamos a ferramenta de simulação de rede Mininet[2] (MININET, 2016). Adicionamos uma quantidade crescente de *hosts* em cada interação (entre 1 e 1000), conforme Figura 6.2.

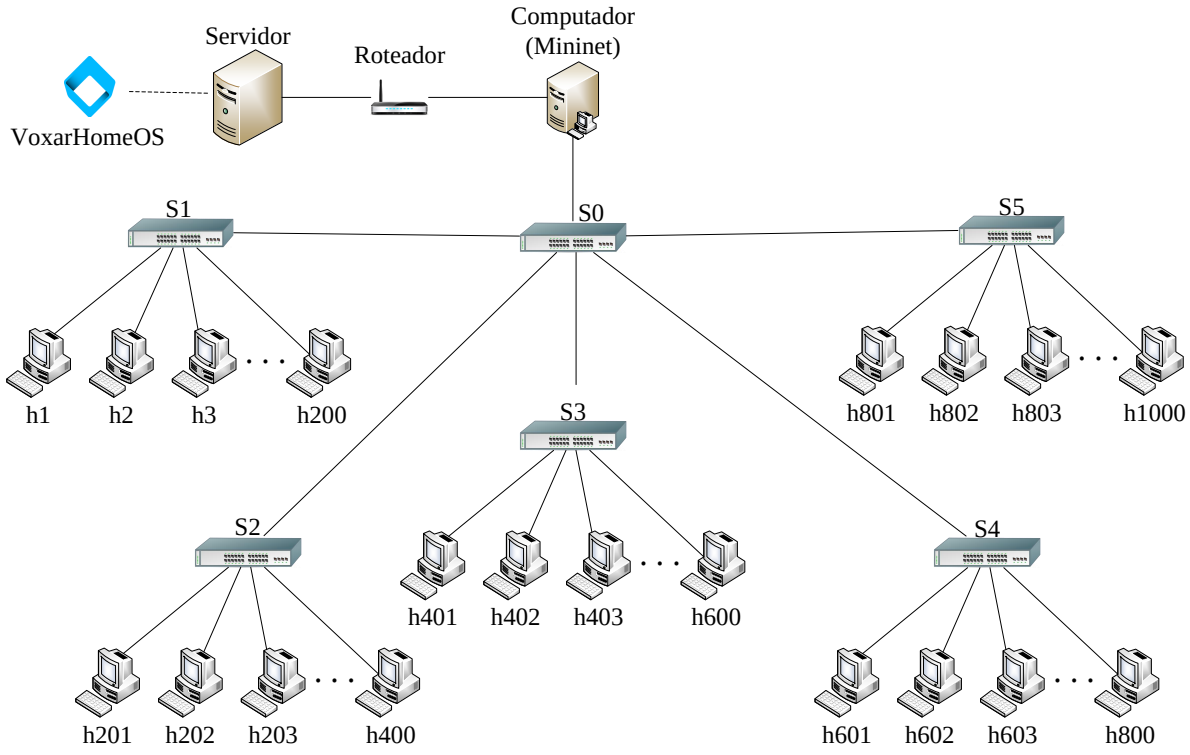


Figura 6.2 Topologia do experimento.

Primeiramente, verificamos o estado inicial do computador que executa o VoxarHomeOS para criar um parâmetro comparativo. Esse estado inicial pode variar de computador para computador, de acordo com os recursos previamente instalados e/ou em execução. Deste modo, é irrelevante descrever a configuração inicial do experimento, sendo necessário apenas analisar o impacto que o crescimento da quantidade de trabalho, gerada pela inclusão de novos dispositivos, pode provocar ao sistema.

A Tabela 6.8 apresenta um resumo dos dados obtidos durante este experimento. A quantidade 0 representa o desempenho do sistema quando não há dispositivos conectados à rede.

Analisando a Tabela 6.8, podemos observar que o crescimento exagerado da quantidade de dispositivos gerenciados pelo VoxarHomeOS não afeta proporcionalmente o desempenho do computador onde o sistema está sendo executado. Podemos inferir ainda que o VoxarHomeOS possui a capacidade de gerenciar uma quantidade elevada de dispositivos de forma simultânea.

Como parâmetro de comparação, podemos considerar o estado do computador com 0

[2] Mininet é uma plataforma de emulação de rede de código aberto, onde é possível criar uma rede virtual *OpenFlow* com controlador, *switches*, *host* e *links*, em uma única máquina física ou uma máquina virtual.

Tabela 6.8 Escalabilidade.

N. de Dispositivos	CPU (%)	Memória (Mb)
0	1,21	1747,08
1	1,31	1844,18
2	1,51	1840,41
3	1,56	1843,57
4	1,55	1839,27
5	1,34	1835,28
10	1,31	1835,21
20	1,33	1836,07
50	1,38	1835,53
100	1,43	1837,07
500	1,47	1840,11
1000	1,44	1859,55

dispositivos, onde o uso de CPU é $1,21\%$ e $1747,08$ *Mb* usados de memória. Ao acrescentar apenas 1 dispositivo este valor varia um pouco, mas não ocorre uma progressão crescente, como exemplo entre os estados com 4 e 5 dispositivos conectados, ocorre um descréscimo de consumo dos recursos, o que pode ser interpretado como uma variação normal do SO.

Outro aspecto a ser destacado é que a utilização de 1000 dispositivos possui um impacto nos recursos de processamento e memória similar ao estado com apenas 1 dispositivo. Este experimento apresentou indícios que mostram que o VoxarHomeOS possui o requisito de escalabilidade funcional, sendo este um dos requisitos mencionados na Seção 4.2.

6.5 Decisão

Para avaliar se as regras criadas pelo sistema VoxarHomeOS são executadas de forma assertiva e em tempo hábil, criamos um cenário onde um atuador era acionado de acordo com o valor de um determinado sensor. Neste caso, para automatizar este experimento, controlamos o valor obtido pelo sensor sinteticamente.

Para verificar a assertividade, o processo foi executado durante $89,28$ horas ininterruptas. Para evitar que uma eventual falha na rede pudesse gerar um ruído nos dados coletados, visto que a mudança de estado do sensor não seria enviada para o VoxarHomeOS, nós monitoramos também a integridade da rede.

Deste modo, como dos 160704 registros enviados pelo dispositivo, todos chegaram ao destino e todos foram processados de maneira adequada, podemos inferir que a precisão de acertos da regra é de 100% para todos os tipos de dados, ressaltando que caso ocorra perda de pacote ou falha na comunição entre os envolvidos na regra esta pode não ser executada corretamente.

6.5.1 Tempo até a Execução da Regra

Visto que a camada de decisão apresentou um alto índice de precisão, avaliamos o tempo entre a mudança de estado de um sensor e o acionamento de fato do atuador correspondente. Para tanto, utilizamos a arquitetura indicada na Figura 6.1, utilizando um Arduino Uno como dispositivo.

A Tabela 6.9 apresenta as médias de tempos (em segundos) obtidos depois de computados os tempos de 30 repetições do seguinte processo:

1. Obter valor da variável (Boolean, Double, String, Date e Time)
2. Conversão para String;
3. Composição da mensagem a ser enviada;
4. Envio da mensagem para o servidor;
5. Recebimento da mensagem;
6. Decomposição da mensagem;
7. Conversão para o tipo de dado adequado;
8. Atribuição adequada ao sensor correspondente;
9. Verificar se há evento associado ao sensor modificado;
10. Executar verificação de estado do evento;
11. Verificar se há regra associada ao evento modificado;
12. Executar verificação de estado da regra;
13. Atribuição adequada ao atuador correspondente;
14. Composição da mensagem de acionamento do atuador;
15. Envio da mensagem para o dispositivo associado ao atuador;
16. Recebimento da mensagem de acionamento;
17. Enviar comando de acionamento para o atuador.

O tempo calculado é iniciado antes de iniciar o processo e finalizado logo após a última tarefa do processo ser encerrada. Os tipos *Date* e *Time* foram substituídos por *String* na tarefa 1, pois o dispositivo não possui suporte nativo para estes tipos de dados e por considerar que seriam convertidos logo em seguida (tarefa 2), contudo, ao chegar ao SO, a conversão ocorria de maneira adequada.

Tabela 6.9 Decisão.

Tipo de dados	Experimento (s)	Simulação (s)	IC. 95% $(\alpha/2; 1-\alpha/2)$
Boolean	0.32697	0.326717	(0.289933; 0.367767)
Double	0.27963	0.279821	(0.266033; 0.295633)
String	0.28740	0.287042	(0.269433; 0.3076)
Date	0.25610	0.256025	(0.242767; 0.271767)
Time	0.28857	0.288229	(0.256133; 0.323433)

Considerando que o processo analisado é iniciado em um determinado dispositivo conectado a uma rede e que, não necessariamente, a reação ocorre em outro dispositivo, passando antes pelo crivo da Camada de Decisão do sistema operacional, os tempos apresentados na Tabela 6.9 representam um tempo de resposta baixo, o que não atrapalha a experiência do usuário.

A partir destes dados, podemos inferir que o VoxarHomeOS pode ser utilizado tanto para tarefas que não demandem uma urgência (ex. se geladeira for desligada, enviar uma notificação para o usuário) quanto para as tarefas com níveis de criticidade maiores, que demandam uma resposta em tempo real (ex. se detectado a presença de pessoa não autorizada na área externa da residência ativar o alarme).

Quanto a escolha de qual o tipo de dado deve ser preferido, caso haja necessidade do menor tempo de resposta, o experimento mostrou que o tipo *Date* apresentou o melhor tempo, porém entre os tipos de dados nativos em todas as plataformas testadas, o tipo *Double* pode ser uma alternativa.

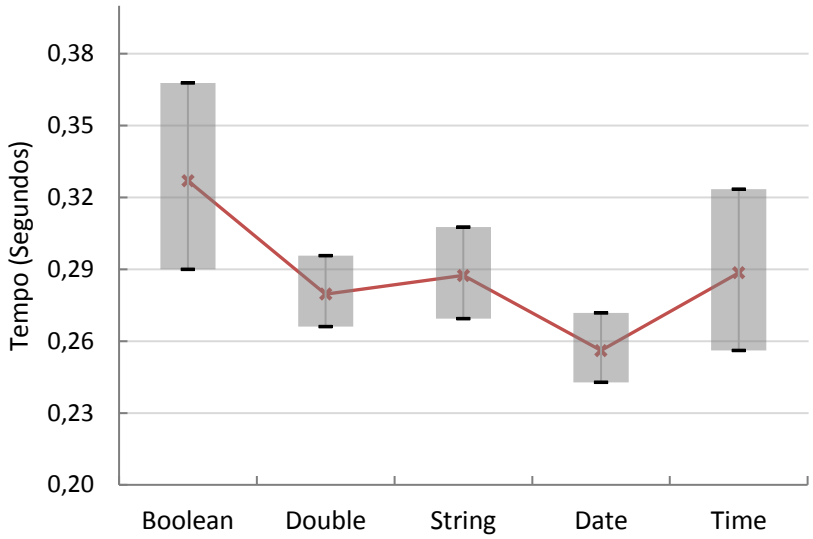


Figura 6.3 Intervalo de confiança.

Em relação a escolha do tipo de dado mais adequado, o experimento mostrou que não há uma discrepância entre estes, visto que se apresentaram muito próximos e considerando os limites inferiores e superiores do intervalo de confiança, como pode ser observado na Figura 6.3.

Estes dados evidenciam a hipótese de que, pelo fato dos valores serem obtidos inicialmente por um determinado tipo e logo em seguida serem convertidos para o tipo *String*, a tomada de decisão, por parte do VoxarHomeOS, não sofre uma dependência significante de tipos.

6.6 Reconhecimento de Padrões

Como descrito na Seção 4.3.5, o VoxarHomeOS possui uma camada de análise de dados. Esta camada é responsável por coletar e interpretar os dados que transitam pelo sistema. A análise ocorre através de classificação dos dados, onde para classificar um conjunto de dados é necessário verificar quais características discriminam melhor as classes e a partir delas iniciar o processo de classificação.

Deste modo, esta seção apresenta uma avaliação da performance e da precisão deste reconhecimento de padrões. Para este experimento foram definidos dois cenários onde o sistema deveria identificar: no Cenário 1, se o habitante estava dormindo ou acordado e, no Cenário 2, a semântica de um determinado gesto, usado para ligar uma televisão. Em ambos os casos os métodos e procedimentos foram os mesmos, portanto, nesta seção iremos descrevê-los de modo geral, especificando-os apenas quando necessário.

Para exemplificar, descreveremos o Cenário 1, onde foi gerado um *dataset* simulando os dados de diversas fontes, entre sensores e atuadores, durante um período de um ano. Este *dataset* foi estruturado em formato de arquivo *Comma-Separated Values* (CSV), onde cada registro representa uma informação de estado única. Após a fase de seleção das características que melhor discriminam as classes, obtivemos o conjunto de dados filtrado, conforme é ilustrado na Código 6.2, onde podemos observar um trecho do *dataset* gerado.

Código 6.2 Dataset - Cenário 1.

```
1 time, bpm, location, movement, orientation, alarm, illumination,
     doors, tv, sleeping
2 00:00:01,56,1,0,1,1,0,0,0,1
3 13:05:19,58,4,4,2,0,1,1,0,0
4 05:53:46,58,1,0,1,1,0,0,0,1
5 07:27:51,88,3,0,3,0,1,1,1,1
6 18:02:53,62,2,0,3,1,1,0,1,0
```

O Código 6.2 apresenta na linha 1 o cabeçalho do arquivo e as demais linhas representam amostras de dados coletados, onde representam respectivamente a hora, a frequência cardíaca do usuário, a localização, o grau da movimentação, a orientação do usuário, o estado do alarme, o estado da iluminação do local, o estado das portas e o estado da televisão da residência, sendo estes convertidos de texto, quando necessário, e representados por identificadores numéricos. O cabeçalho do arquivo é composto por sensores e atuadores, além de uma *flag* utilizada durante a fase de aprendizagem, no caso do exemplo, o rótulo *"sleeping"* indica o estado real do registro,

mas não é utilizado durante as fases de treinamento e classificação.

O experimento foi dividido em três etapas; na primeira os *datasets* foram gerados. Na segunda, o sistema deveria consumir esses conjuntos de dados, identificar padrões (gerando um evento como resultado) e calcular tempo decorrido para a execução desta etapa. Na terceira etapa, selecionamos todos os dados contidos nos *datasets* e verificamos quantos foram classificados de modo correto.

A Seção 6.6.1 apresenta o tempo médio de execução da etapa 2 e a Seção 6.6.2 apresenta os dados referentes à assertividade da análise.

6.6.1 Tempo de Análise

Devido a quantidade de dados que podem ser gerados em um ambiente residencial, dotado de um sistema de IoT amplamente sensoriado, analisamos o tempo necessário para o sistema treinar o algoritmo de aprendizado. Inicialmente, utilizamos o *dataset* do Cenário 1, selecionado uma sub-amostra equivalente a 30 dias de funcionamento, o que totalizou 19881 registros analisados e a amostra completa (365 dias), totalizando 233712 registros. A Tabela 6.10 apresenta os dados obtidos.

Tabela 6.10 Tempo de análise de dados.

Dias	Registros	Tempo (s)
30	19881	1.8472
365	233712	13.3602

Fundamentado nos dados coletados e na observação realizada, podemos inferir que, para a amostra analisada, o tempo demandado para a análise não é instantâneo, podendo durar proporcionalmente ao tamanho do conjunto de dados. Entretanto, o impacto deste eventual atraso não deve ser observado pelo usuário do sistema, visto que a execução do módulo de análise é realizada de forma paralela, sendo apenas a saída do processo interessante aos demais módulos.

6.6.2 Assertividade

Após a etapa de treinamento realizamos a validação, obtendo um evento de saída que engloba as informações aprendidas. No caso do Cenário 1, o evento *dormindo* recebe o valor *"true"* quando um conjunto de sensores e atuadores possuem seus respectivos valores dentro de um intervalo predefinido, conforme apresentado na Tabela 6.11.

Como forma de avaliar se o algoritmo de aprendizagem foi capaz de classificar corretamente as entradas recebidas, foi realizado um novo acesso ao *dataset*, desta vez, verificando se cada registro, de forma independente, pertencia ao grupo *dormindo* ou não. Através deste processo foi possível estabelecer o nível de precisão, para este caso em particular, de $88,313\%$ de acertos, conforme é apresentado na Tabela 6.12.

Tabela 6.11 Vetor de características para o evento "dormindo".

Parâmetro	Intervalo
Local	(1.000; 1.267)
Iluminação	(0.000; 0.267)
Alarme	(0.732; 1.000)
Orientação	(1.000; 1.269)
TV	(0.000; 0.271)
Portas	(0.000; 0.271)
BPM	(50.000; 70.834)
Movimentação	(0.000; 0.860)

Tabela 6.12 Reconhecimento de padrões.

Tipo	Precisão (%)
Acertos	88.313
Erros	11.688
Falso-Positivo	0
Falso-Negativo	11.688

Do total de 233712 registros, 92982 representavam uma entrada da classe *dormindo* e 140730 representavam uma entrada de classe oposta. Contudo, 27315 casos que originalmente são da classe *dormindo* foram categorizados como sendo da classe oposta, um falso-negativo de $11,688\%$, já entre os casos que originalmente não eram da classe *dormindo*, 0 (zero) foi classificado como *dormindo*, representando um falso-positivo de 0%, sendo assim, a taxa de erro total da classificação é a mesma do falso-negativo.

Considerando os dados apresentados na Tabela 6.12, podemos observar que o módulo de análise pode funcionar como se previu inicialmente, indicando possíveis eventos que ocorrem no ambiente residencial, porém com ressalvas. No cenário experimentado, a taxa de erro de $11,688\%$ representou uma taxa satisfatória, visto que o falso-positivo de 0% indica que esporadicamente o evento pode não ser reconhecido, porém não será reconhecido quando de fato não ocorrer.

Deste modo, podemos inferir que o módulo de análise se comporta como devido, realçando padrões recorrentes. Considerando que o usuário pode supervisionar, no sentido de confirmar ou ajustar a informação aprendida, e transformá-la em uma regra, o sistema fornece um importante requisito que o difere de outros encontrados na literatura, conforme apresentado na Tabela 3.1.

6.7 Interação Natural

A Interação Natural é um aspecto subjetivo e, por vezes, complexo de ser analisado em um experimento, pois incide sobre esta uma série de variáveis subjetivas. Portanto, iremos

descrever de forma detalhada o ambiente do experimento desta seção, viabilizando uma eventual replicação.

A métrica escolhida para avaliar a interação foi a robustez do reconhecimento, tanto dos comandos de voz quanto dos gestos do usuário. Para tanto, definimos previamente os comandos (uma determinada frase ou palavra e um gesto descrito gramaticalmente) que poderiam ser utilizados, repetindo-os diversas vezes em cada um dos modos de interação. Desta forma, coletamos a quantidade de vezes que o comando foi reconhecido corretamente e verificamos a validade dos dados utilizando o método *Bootstrap* (descrito na Seção 6.1).

6.7.1 Cenário - Voz

O ambiente do experimento de robustez da interação por voz é composto por apenas um *hardware*, que é um computador (item 1 da Tabela 6.4) com microfone embutido. Contudo, outros fatores merecem ser destacados, a citar:

- O ambiente físico possui um área de $6,25m^2$;
- A quantidade de decibéis medida[3] no ambiente era $8dB$;
- Durante o momento do comando de voz, a aferição dos decibéis foi em média $21dB$;
- A distância do usuário para o microfone era $50cm$;
- O comando de voz utilizado foi “Camila”.

6.7.2 Cenário - Gesto

Assim como o ambiente de experimento de robustez da voz, a avaliação do gesto utilizou um computador, mas além desse foi utilizado um sensor Microsoft Kinect v2, este sensor é responsável por mapear o corpo do usuário, detectando gestos predefinidos. Neste experimento, as variáveis que merecem ser destacadas são a posição e a distância do sensor em relação ao usuário. O sensor foi posicionado a uma altura de $0,85cm$, sendo possível visualizar completamente o corpo do usuário, que encontrava-se a uma distância de $185cm$ em relação ao sensor. O usuário deveria executar 30 (trinta) vezes o gesto descrito no Código 6.3.

Código 6.3 Descrição do gesto utilizado.

```
1 GESTURE Pull:
2     POSE hand_down:
3       put your right hand tip below your right hip.
4
```

[3]Aferição realizada através do aplicativo Noise Meter Tool, disponível em https://itunes.apple.com/us/app/noise-meter-tool/id931831744

```
5     POSE hand_up:
6       put your right hand tip above your head.
7
8     POSE pull_down:
9       put your right hand tip below your head.
10
11    EXECUTION:
12      hand_down,
13      hand_up,
14      pull_down.
```

6.7.3 Resultados

Como resultado, obtivemos que, nos cenários descritos nas seções anteriores, $93,33\%$ dos comandos de voz foram reconhecidos no experimento real, sendo que na simulação com 1000 reamostras, $93,42\%$ dos casos seriam reconhecidos, estando ambos dentro do intervalo de confiança (entre $83,33\%$ e 100%). Quanto ao reconhecimento dos gestos, a porcentagem de acertos foi 90% no experimento prático e $90,20\%$ de acertos foram computados através da reamostragem. A Tabela 6.13 apresenta os valores obtidos durante os experimentos, bem como os intervalos de confiança das amostras.

Tabela 6.13 Taxa de reconhecimento de um comando de voz e de um gesto.

Dispositivos	**Experimento (%)**	**Simulação (%)**	**IC. 95% ($\alpha/2; 1-\alpha/2$)**
Voz	93,33	93,42	(83,33; 100,00)
Gestos	90,00	90,20	(80,00; 100,00)

Considerando que estes experimentos ocorreram em uma ambiente controlado, os dados apresentados na Tabela 6.13 evidenciam que tanto a interação por voz quanto por gesto funcionam relativamente bem nestes ambientes. Entretanto, uma análise mais profunda pode indicar que em ambientes não tão favoráveis, pode não ser tão robusta quanto no cenário analisado. Em trabalhos futuros, é relevante avaliar em quais circunstâncias de fato cada interação tende a ser melhor utilizada, levando em consideração cenários com oclusões parciais, posicionamento e distância variadas, para a interação por gesto e cenários com ruído sonoro, distância e dificuldades de pronuncias para a interação por voz. Em experimentos preliminares foi possível observar a dificuldade maior de reconhecimento dos comandos quando ocorria uma variação de alguns aspectos dos cenários apresentados (Seções 6.7.1 e 6.7.2). Uma hipótese inicial para estes experimentos pode ser que a inclusão de sensores distribuídos no ambiente ou técnicas distintas podem mitigar as eventuais falhas de semântica das interações.

6.8 Avaliação com Usuário

Além dos experimentos descritos nas seções anteriores, realizamos um experimento com voluntários, representando a perspectiva do usuário final do sistema. Neste experimento, 47 voluntários foram estimulados a utilizar o sistema, guiados por uma tarefa que deveriam realizar. Para tanto, algumas etapas iniciais foram predefinidas, bastando os participantes executarem apenas as tarefas de interesse. Na configuração inicial foram adicionados 5 sensores: Data, Hora, Distância, Frequência Cardíaca e Temperatura. Além desses foram adicionados 3 atuadores: E-mail, Notificação e Lâmpada. Nenhum evento ou regra foram configurados previamente, apenas um glossário com os termos foi disponibilizado antes de iniciar o experimento.

O grupo foi dividido em 3 turmas, de acordo com o curso de suas respectivas formações, sendo estas Informática (18), Eletromecânica (15) e Biocombustíveis (14). Deste total 29 informaram ter um conhecimento prévio (acadêmico) sobre Lógica de Programação e 18 nunca tiveram aula sobre este assunto.

A tarefa (Tarefa A) solicitada aos voluntários foi "criar uma regra que irá ascender uma lâmpada quando o valor do sensor de distância for inferior a 30". Dos 47 participantes, 3 não conseguiram concluir a tarefa ou não compreenderam o processo e 44 executaram a tarefa adequadamente. Utilizando como base apenas os 44 processos executados, nós obtivemos um tempo médio de 112 segundos para a execução da tarefa, no primeiro contato dos usuários com o sistema.

Após a execução da primeira tarefa, uma segunda tarefa (Tarefa B) foi proposta para os participantes, no intuito de avaliar a curva de aprendizagem. A tarefa sugerida foi "criar uma regra quer irá enviar uma notificação quando o valor do sensor de frequência cardíaca for superior a 120". Como esperado, o tempo para executar a segunda tarefa foi inferior ao primeiro caso, sendo a tarefa concluída em 32 segundos em média, uma redução de $71,42\%$.

A terceira fase desse experimento fundamenta-se no desenvolvimento das mesmas regras sem o auxilio do VoxarHomeOS, onde os voluntários foram instigados a desenvolver a Tarefa A utilizando qualquer recurso disponível. Para tanto, foi disponibilizado um kit contendo 2 dispositivos: 1 Arduino Uno, já acoplado a este um Ethernet Shield w5100 e um sensor ultrassônico HR-SC04; 1 Arduino Mega, contendo um Ethernet Shield w5100 e um Relay Shield ligado a uma lâmpada. Nesta terceira fase apenas 14 voluntários participaram, sendo que durante um período de 2 horas destinado a cada um, nenhum conseguiu executar a Tarefa B de forma funcional.

Ao final da experiência, solicitamos que os participantes preenchessem um formulário informando uma nota de 1 à 5 em relação ao nível de dificuldade que eles consideraram para executar a tarefa. A nota geral foi $4,064$. A Tabela 6.14 apresenta os resultados obtidos, destacando-os por grupo. A Figura 6.4 apresenta a quantidade (em porcentagem) de pessoas que classificaram a complexidade de realizar a tarefa entre fácil ($72,34\%$), normal ($21,28\%$) e difícil ($6,38\%$). Consideramos como difícil as notas 1 e 2; como normal a nota 3; e fácil as notas 4 e 5.

Tabela 6.14 Avaliação com usuários.

Grupo	Média
Informática	4,333
Eletromecânica	3,733
Biocombustíveis	4,071
	4,064

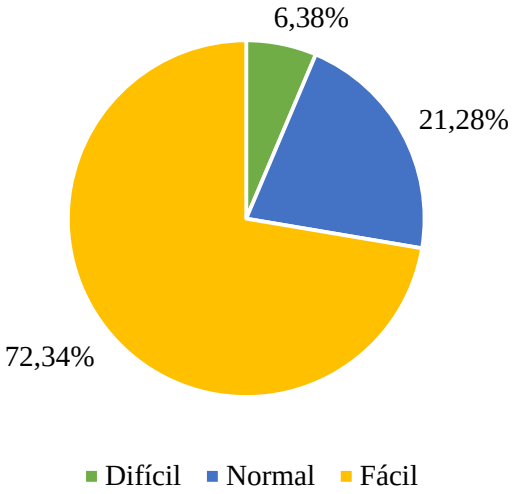


Figura 6.4 Avaliação do nível de dificuldade em realizar a tarefa de criar regras.

Comparando os desempenhos dos grupos que utilizaram a ferramenta e do grupo que não utilizou a ferramenta e de posse dos dados condensados na Figura 6.4 temos evidências que corroboram com a hipótese que o VoxarHomeOS, de fato, é um sistema operacional para residências inteligentes capaz de abstrair a complexidade do *hardware*, facilitando o desenvolvimento de soluções com foco no ambiente residencial.

6.9 Considerações Finais

Este capítulo serviu para elucidar alguns aspectos deste trabalho. Na Seção 6.2 podemos observar que o SO apresenta compatibilidade com uma diversidade de *hardware* e consegue integrar os dispositivos.

Na Seção 6.3 apresentamos evidências de que a gerência dos recursos ocorre de maneira satisfatória, onde o tempo de resposta fim-a-fim de um dispositivo ao sistema é considerado baixo, tanto para conexões *Ethernet* quanto para conexões *WiFi*. A descoberta em rede, um dos requisitos da arquitetura proposta, foi avaliada. Neste caso, identificamos que alguns dispositivos necessitam de um tempo maior para acessar a rede, contudo ao ingressar na mesma, o tempo até o VoxarHomeOS detectar a entrada é $3,632$ segundos, em média, tendo um desvio padrão de $1,223$.

Na Seção 6.4 avaliamos a escalabilidade da arquitetura. Para tanto, implantamos uma

rede SDN virtualizada, onde acrescentamos *hosts* de forma gradual, simulando conexões de novos dispositivos. Em cada incremento, observamos o desempenho do computador onde estava sendo executado o VoxarHomeOS no intuito de verificar o impacto de cada novo *host*. A consideração que podemos fazer com base no estudo é que o sistema mostra-se escalável, suportando até 1000 dispositivos conectados simultaneamente sem degradação do desempenho, em experimentos preliminares, testamos até 10000 *hosts* e estes resultados se mantêm.

Na Seção 6.5 discutimos e avaliamos se o gerenciamento baseado no contexto é funcional, preciso e qual tipo de informação apresenta melhores resultados. Quanto aos dois primeiros aspectos, o experimento mostrou que além de executar corretamente o gerenciamento dos recursos, não há indícios de que a regra possa falhar caso a mensagem esteja correta. Em relação aos tipos de dados que podem ser utilizados, observamos que todos possuem desempenho bem similares, não sendo possível ressaltar um em especial.

Na Seção 6.6 realizamos um experimento guiado pela análise dos padrões. Inicialmente, geramos um *dataset* contendo informações sobre o ambiente residencial, em seguida (utilizando um algoritmo SVM), extraímos e classificamos os dados, resultando em um vetor de características que compõem um evento. O experimento serviu para avaliar ainda o tempo de processamento da uma análise e, principalmente, a precisão da classificação ($88,313\%$).

Na Seção 6.7 averiguamos a interação natural entre o usuário e o ambiente. Neste caso, o que se pode afirmar é que interação, tanto por voz quanto por gesto, apresentaram-se como alternativas viáveis com uma precisão de $93,33\%$ e 90% para voz e gesto, respectivamente. Em uma análise mais profunda podemos dizer que os experimentos apresentados neste capítulo comprovam que a inclusão de uma camada de interação não compromete os demais requisitos, visto que todos se comportaram de maneira satisfatória.

7

CONCLUSÃO

Em um futuro breve a quantidade de dispositivos conectados deve aumentar substancialmente. Com isso uma quantidade proporcional de pesquisas e soluções mercadológicas estão surgindo. Há indícios de que o ambiente residencial poderá sofrer um grande impacto de transformação de paradigma, devido à quantidade de objetos conectados que poderão ser incorporados neste ambiente. Deste modo, o amadurecimento da Internet das Coisas evidencia que é factível possuirmos residências verdadeiramente inteligentes em alguns anos e não apenas residências automatizadas.

Para tanto, os provedores de serviços, bem como os próprios habitantes precisam lidar com a dificuldade de interoperabilidade entre os dispositivos heterogêneos, com a complexidade de abstração de *hardware*, com a dificuldade de gerir uma massa de dados que pode ser produzida no ambiente e com interação através de controles complexos.

Partindo dessa premissa, este trabalho teve como objetivo propor e avaliar uma arquitetura de um sistema operacional para residências, fundamentando-se no conceito de Internet das Coisas. Para alcançar este objetivo, foi elaborada uma visão geral das principais plataformas de IoT atuais (Seção 2.4), bem como das arquiteturas recorrentes (Seção 2.5).

Com base no levantamento realizado, foi concebida uma arquitetura (Seção 4.3) capaz de suprir os principais requisitos de uma solução vertical em IoT, a citar: Interoperabilidade, Escalabilidade, Extensibilidade, Descoberta em rede, Sensibilidade ao Contexto e Análise de Dados. Como complemento, foi incorporado o requisito de interação natural, provendo assim subsídios para que os habitantes possam realizar tarefas cotidianas sem a necessidade de se preocupar com o funcionamento de interfaces ou controles complexos. Deste modo, foi proposta uma arquitetura contendo as camadas de *Hardware*, Abstração de *Hardware*, Rede, Decisão, Análise, Interação, Serviços e Aplicação (Seção 4.3).

Como forma de avaliar a referida arquitetura foi implementado um protótipo de sistema operacional para residências (Seção 5.1) fundamentado na arquitetura e contemplando o conjunto de requisitos supracitado. O protótipo foi útil para avaliar se a implementação dos componentes propostos na arquitetura estão organizados ao modo que é viável desenvolver um sistema operacional capaz de suprir os requisitos elencados na literatura, bem como possuir suporte à

interação natural. Esta avaliação foi realizada através de estudos de caso (Seção 5.2).

Vislumbrando futuras implementações de sistemas operacionais para o ambiente residencial, foram avaliadas algumas métricas que podem servir como referência (Capítulo 6). Dentre estas, destacamos (i) o tempo de resposta da comunicação entre os dispositivos e o SO (Seção 6.3); (ii) o tempo médio até a descoberta de ingresso na rede, utilizando dispositivos distintos (Seção 6.3.2); (iii) a escalabilidade do sistema, que apresentou um crescimento de até 1000 dispositivos trocando mensagens com o SO de forma simultânea, sem degradação acentuada de performance (Seção 6.4); (iv) a precisão e o tempo médio entre a mudança de estado de um sensor até a execução de uma regra associada (Seção 6.5); (v) a precisão do aprendizado, isto é, a classificação correta de uma determinada entrada (informações instantâneas da residência) (Seção 6.6); (vi) a precisão do reconhecimento de voz e de gesto, esta métrica é útil para planejar cenários onde haja interação entre o usuário e o sistema, podendo adicionar redundâncias de sensores (Seção 6.7). Por fim, um experimento com a colaboração de usuários que foram instigados para a elaboração de regras utilizando o SO, e avaliar a complexidade (Seção 6.8).

Portanto, podemos dizer que possuímos evidências que apontam para a viabilidade de se incorporar uma camada de interação natural em arquiteturas de plataformas de IoT para ambientes residenciais.

7.1 Contribuições

Como consequência das atividades desenvolvidas nesta tese, alcançamos uma série de resultados, tendo como a maior contribuição a elaboração de um modelo de arquitetura de SO para um ambiente residencial, com suporte à interação natural.

- Devido à ascensão das tecnologias relacionadas à IoT e o advento de diversas ferramentas, elencar as principais plataformas, de modo a organizar e categorizar de acordo com seus requisitos, se faz necessário. Neste trabalho nós apresentamos as principais plataformas para IoT, auxiliando assim no processo de escolha em futuras implementações. Apesar da diversidade de objetivos das plataformas, através desse levantamento foi possível classificar as plataformas entre comerciais e *open source*, e apresentar um comparativo entre estas.

- Apesar das diversas variações, nós destacamos a existência de dois tipos de arquiteturas recorrentes, uma contendo três camadas e outra contendo cinco camadas. Outro aspecto importante quanto a arquiteturas e soluções para IoT são os requisitos que estes devem prover. Sendo assim, elencamos os principais requisitos, de acordo com o estudo realizado na literatura. Além de preservar os principais requisitos, como: Interoperabilidade, Escalabilidade, Extensibilidade, Descoberta, Sensibilidade ao

Contexto e Análise de Dados, incluímos o requisito de Interação Natural, para atender o contexto de IoT para um ambiente residencial.

- A principal contribuição desta tese é a própria arquitetura elaborada, seguindo os requisitos extraídos da literatura. Nesta arquitetura, foram estalecidas oito camadas, a citar: *Hardware*, Abstração, Rede, Decisão, Análise, Serviço, Interação e Aplicação. De acordo com os resultados do levantamento de pesquisa desta tese, esta a primeira arquitetura de IoT a estabelecer uma camada dedicada à mecanismos de interação natural para sistemas operacionais com ênfase no ambiente residencial. Desta forma, a arquitetura aqui proposta apresentou uma diversidade de possibilidades de aplicações e exploração do ambiente.

- O protótipo desenvolvido (VoxarHomeOS), foi implementado com base nas definições da arquitetura proposta. Através do protótipo implementado nesta tese é possível gerenciar os recursos da residência, bem como reconhecer gestos e comandos de voz executados pelo habitante para tomada de decisão. Os próprios usuários podem elaborar e gerenciar regras e eventos através de uma plataforma web. Isto é, apesar de se tratar de um protótipo, o protótipo é funcional, contemplando aspectos e requisitos para um sistema IoT.

- Esta tese visou prover mecanismos que pudessem ser reutilizados e estendidos. Fundamentado nesta premissa, foi elaborado uma API para que os desenvolvedores possam ter acesso as principais funcionalidades do sistema e, consequentemente, conseguir desenvolver aplicações mais complexas que demandam conhecimento em linguagem de programação, diferentemente da elaboração de eventos e regras, que são visuais, sendo apenas desejável um conhecimento introdutório em lógica.

- Realizamos alguns experimentos para avaliar, de forma quantitativa, aspectos, como: desempenho de plataformas de hardware, meios de transmissão, tempo de resposta, escalabilidade, tempo para a descoberta de dispositivos e precisão das regras definidas pelos usuários e do aprendizado adquirido com base no reconhecimento de padrões, como também, os mecanismos de interação natural, sob os aspectos de assertividade. Deste modo, estas métricas podem ser consideradas em futuras implementações de sistemas operacionais residenciais, como valores de referência.

- Uma vantagem em utilizar linguagens descritivas para representar um gesto ou comando de voz é que usuários que não possuam conhecimento técnico ou não sejam programadores podem interpretar o conteúdo descrito por terceiros, bem como criar ou editar seus gestos e comandos de voz de forma personalizada. Deste modo, contribuímos para que os usuários possam configurar o ambiente do modo que preferir, abstraindo assim a complexidade de desenvolver módulos de interpretação e reconhecimento, tanto de gesto como de voz.

Desta forma, podemos afirmar que os objetivos descritos na Seção 1.1 foram alcançados. Podemos inferir que os experimentos realizados nesta tese comprovam a viabilidade da arquitetura proposta.

7.2 Limitações e Trabalhos Futuros

Apesar de considerarmos este trabalho abrangente, a delimitação do escopo restringiu a investigação de alguns aspectos. Deste modo, esta seção apresenta os pontos que não foram contemplados em sua totalidade e que poderão ser abordados em trabalhos futuros.

- **Avaliação com usuários:** Neste trabalho foi realizado um experimento com a colaboração de 47 voluntários, que auxiliaram no processo de avaliar a complexidade do protótipo. Apesar dos indícios obtidos, consideramos que avaliações mais profundas devem ser realizadas, a fim de averiguar aspectos relacionados a usabilidade.

- **Ambiente real:** Um sistema com enfoque em residências deve ser testado em tal ambiente. Logo, a próxima etapa desse projeto é realizar experimentos em um ambiente real. O foco desse estudo deve ser na análise de dados e na interação natural, investigando situações que possam gerar ruídos e, consequentemente, impacto no sistema.

- **Segurança:** Um requisito essencial para aplicações IoT é prover níveis altos de segurança, principalmente, em um ambiente residencial. Apesar deste trabalho contribuir com a segurança e privacidade, pois aplicativos de terceiros podem solicitar informações de um sensor específico sem necessidade de acessar diretamente os dados do sensor, em trabalhos futuros, pretendemos avaliar a viabilidade de incorporação de componentes específicos de segurança para a arquitetura proposta.

- **Reconhecimento de Objetos:** Na camada de interação da arquitetura proposta, o subcomponente *Object*, do componente *Vision*, não foi verificado. Este elemento tem como função prover mecanismos de reconhecimento de objetos, sendo uma tarefa relevante em um ambiente repleto de itens que possuem informações retidas. Uma das abordagens que pretendemos utilizar é avaliar a possibilidade de adaptação do algoritmo do YOLO (*You Only Look Once*), um sistema de detecção de objetos em tempo-real de código aberto (REDMON; FARHADI, 2016).

- **Protocolos:** A arquitetura proposta prevê o uso de diversos protocolos. Contudo, no protótipo foram implementados apenas os protocolos CoAP e MQTT.

REFERÊNCIAS

ADIBI, S. **Mobile Health**: a technology road map. [S.l.]: Springer, 2015. v.5.

AIDO. **Aido Robot**. 2016. Disponível em: http://www.aidorobot.com.

AKA-INTELLIGENCE. **Musio Robot**. 2016. Disponível em: https://themusio.com.

AKYILDIZ, I. et al. Wireless sensor networks: a survey. **Computer Networks**, [S.l.], v.38, n.4, p.393 – 422, 2002.

AL-FUQAHA, A. et al. Internet of Things: a survey on enabling technologies, protocols, and applications. **IEEE Communications Surveys Tutorials**, [S.l.], v.17, n.4, Fourthquarter 2015.

ALAWADHI, S. et al. Building understanding of smart city initiatives. In: **Electronic government**. [S.l.]: Springer, 2012. p.40–53.

ALDEBARAN, R. **Pepper Robot**. 2016. Disponível em: https://www.aldebaran.com/en/cool-robots/pepper.

ALLIANCE, Z.-W. **Z-Wave**. 2017. Disponível em: https://z-wavealliance.org.

AMARAN, M. H. et al. A Comparison of Lightweight Communication Protocols in Robotic Applications. **Procedia Computer Science**, [S.l.], v.76, p.400–405, 2015.

AMAZON. **AWS IoT – Cloud Services for Connected Devices**. 2016. Disponível em: https://aws.amazon.com/pt/blogs/aws/aws-iot-cloud-services-for-connected-devices.

AMAZON. **AWS IoT in the Console**. 2016. Disponível em: https://aws.amazon.com/pt/iot/getting-started.

AMAZON. **AWS IoT Device SDK**. 2016. Disponível em: https://aws.amazon.com/pt/iot/sdk/.

APPLE. **HomeKit - Apple Developer**. 2015. Disponível em: https://developer.apple.com/homekit.

APPLE. **Find acessories that work with Apple HomeKit**. 2016. Disponível em: https://support.apple.com/en-us/HT204903.

ARAS, E. **A Light-Weight Operating System for Internet of Things Devices**. 2016. Dissertação (Mestrado em Ciência da Computação) — NTNU.

ASHTON, K. That 'internet of things' thing. **RFiD Journal**, [S.l.], v.22, n.7, p.97–114, 2009.

ASUNCION, C. H.; SINDEREN, M. van. Towards Pragmatic Interoperability in the New Enterprise—A Survey of Approaches. In: **Enterprise Interoperability**. [S.l.]: Springer, 2011. p.132–145.

ASUNCION, C. H.; VAN SINDEREN, M. J. Pragmatic interoperability: a systematic review of published definitions. In: **Enterprise Architecture, Integration and Interoperability**. [S.l.]: Springer, 2010. p.164–175.

ATZORI, L.; IERA, A.; MORABITO, G. The Internet of Things: a survey. **Computer Networks**, [S.l.], v.54, n.15, p.2787 – 2805, 2010.

BANDYOPADHYAY, D.; SEN, J. Internet of things: applications and challenges in technology and standardization. **Wireless Personal Communications**, [S.l.], v.58, n.1, p.49–69, 2011.

BANDYOPADHYAY, S. et al. Role of middleware for internet of things: a study. **International Journal of Computer Science and Engineering Survey**, [S.l.], v.2, n.3, 2011.

BELAGIANNIS, V.; ZISSERMAN, A. Recurrent Human Pose Estimation. **CoRR**, [S.l.], v.abs/1605.02914, 2016.

BELKIN. **Wemo Home Automation**. 2015. Disponível em: `http://www.belkin.com/us/Products/home-automation/c/wemo-home-automation/`.

BLUETOOTH. **Adopted Specifications - Bluetooth Technology Website**. 2016. Disponível em: `https://goo.gl/sK8mCM`.

BOLZANI, C. A. M. **Residências inteligentes**. [S.l.]: Editora Livraria da Física, 2004.

BOLZANI, C. A. M. **Análise de Arquiteturas e Desenvolvimento de uma Plataforma para Residências Inteligentes**. 2010. Tese (Doutorado em Ciência da Computação) — Universidade de São Paulo.

BORGIA, E. The Internet of Things vision: key features, applications and open issues. **Computer Communications**, [S.l.], v.54, p.1–31, 2014.

BOWMAN, D. A. et al. **3D user interfaces**: theory and practice. [S.l.]: Addison-Wesley, 2004.

BRILLO, G. **Brillo - Google Developers**. 2015. Disponível em: `https://developers.google.com/brillo/`.

BROWN, K. **Desenvolva um sensor de temperatura pronto para nuvem com o Arduino Uno e o IBM IoT Foundation, Parte 2**: escreva o sketch e conecte ao quickstart do ibm iot foundation. 2015. Disponível em: `https://www.ibm.com/developerworks/br/cloud/library/cl-bluemix-arduino-iot2/cl-bluemix-arduino-iot2-pdf.pdf`.

BRUSH, A. B. et al. Lab of Things: a platform for conducting studies with connected devices in multiple homes. In: ACM CONFERENCE ON PERVASIVE AND UBIQUITOUS COMPUTING ADJUNCT PUBLICATION, 2013., New York, NY, USA. **Proceedings...** ACM, 2013. p.35–38. (UbiComp '13 Adjunct).

CAO, Z. et al. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. **CoRR**, [S.l.], v.abs/1611.08050, 2016.

CARBONI, D.; ZANARINI, P. Wireless Wires: let the user build the ubiquitous computer. In: INTERNATIONAL CONFERENCE ON MOBILE AND UBIQUITOUS MULTIMEDIA, 6., New York, NY, USA. **Proceedings...** ACM, 2007. p.169–175. (MUM '07).

CHEN, C.; RAMANAN, D. 3D Human Pose Estimation = 2D Pose Estimation + Matching. **CoRR**, [S.l.], v.abs/1612.06524, 2016.

CHERNICK, M. R. **Bootstrap methods**: a guide for practitioners and researchers. [S.l.]: John Wiley & Sons, 2011. v.619.

CISCO. **How Service Providers Can Help Businesses to Realize the Promise of the IoT Revolution**. 2015. Disponível em: https://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/mobile-internet/iotwf-whitepaper-realize-promise-iot-revolution.pdf.

COOK, D. J. et al. MavHome: an agent-based smart home. In: PERVASIVE COMPUTING AND COMMUNICATIONS, 2003.(PERCOM 2003). PROCEEDINGS OF THE FIRST IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2003. p.521–524.

COSTA, A. **Robótica Móvel Inteligente**: progressos e desafios. 2003. Tese (Doutorado em Ciência da Computação) — Tese de Livre Docência, Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

CRUZ, L.; LUCIO, D.; VELHO, L. Kinect and RGBD Images: challenges and applications. In: GRAPHICS, PATTERNS AND IMAGES TUTORIALS (SIBGRAPI-T), 2012 25TH SIBGRAPI CONFERENCE ON. **Anais...** [S.l.: s.n.], 2012. p.36–49.

DESAI, P.; SHETH, A.; ANANTHARAM, P. Semantic gateway as a service architecture for iot interoperability. In: MOBILE SERVICES (MS), 2015 IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2015. p.313–319.

DING, Y. et al. Smart beijing: correlation of urban electrical energy consumption with urban environmental sensing for optimizing distribution planning. In: WORK IN PROGRESS (ENERGY 2011), 1ST INTERNATIONAL CONFERENCE ON SMART GRIDS, GREEN COMMUNICATIONS AND IT ENERGY-AWARE TECHNOLOGIES. VENICE/MESTRE, ITALY: THINK MIND. **Anais...** [S.l.: s.n.], 2011. p.98–101.

DIXON, C. et al. An Operating System for the Home. In: NSDI. **Anais...** USENIX, 2012.

DOCTOR, F.; HAGRAS, H.; CALLAGHAN, V. A fuzzy embedded agent-based approach for realizing ambient intelligence in intelligent inhabited environments. **IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans**, [S.l.], v.35, n.1, p.55–65, Jan 2005.

DUCATEL, K. et al. **Scenarios for ambient intelligence in 2010**. [S.l.]: Office for official publications of the European Communities, 2001.

DUCATEL, K. et al. Ambient intelligence: from vision to reality. **IST Advisory Group Draft Report, European Commission**, [S.l.], 2003.

EDSON, B. Creating the Internet of Your Things. **Microsoft Corporation**, [S.l.], 2014.

EFRON, B.; TIBSHIRANI, R. J. **An introduction to the bootstrap**. [S.l.]: CRC press, 1994.

EVANS, P. C.; ANNUNZIATA, M. **Industrial Internet - Pushing the Boundaries of Minds and Machines**. 2016.

FETTE, I.; MELNIKOV, A. **The websocket protocol**. 2011. Disponível em: https://tools.ietf.org/html/rfc6455.

FIELDING, R. T. **Architectural styles and the design of network-based software architectures**. 2000. 76-105p. Tese (Doutorado em Ciência da Computação) — University of California, Irvine.

FIGUEIREDO, L. S. **A Design Method for Building in-air Gestural Interactions**. 2017. Tese (Doutorado em Ciência da Computação) — Universidade Federal de Pernambuco, UFPE.

FIGUEIREDO, L. S. et al. **Prepose**: security and privacy for gesture-based programming. [S.l.: s.n.], 2014. Disponível em: http://research.microsoft.com/apps/pubs/default.aspx?id=232011. (MSR-TR-2014-146).

FIGUEIREDO, L. S. et al. Prepose: privacy, security, and reliability for gesture-based programming. **IEEE Security Privacy**, [S.l.], v.15, n.2, p.14–23, March 2017.

FITBIT. **Fitbit Official Site for Activity Trackers & More**. 2016. Disponível em: https://www.fitbit.com.

FREMANTLE, P. A reference architecture for the internet of things. **WSO2 White paper**, [S.l.], 2014.

GARTNER. **Gartner's 2015 Hype Cycle for Emerging Technologies Identifies the Computing Innovations That Organizations Should Monitor**. 2015. Disponível em: http://www.gartner.com/newsroom/id/3114217.

GARTNER. **Top Trends in the Gartner Hype Cycle for Emerging Technologies, 2017**. 2017. Disponível em: https://goo.gl/xrW6wh.

GARTNER. **Garter IT Glossary**. 2017. Disponível em: http://www.gartner.com/it-glossary/connected-home/.

GEEK, H. **Brains for your Garden.** 2015. Disponível em: https://www.kickstarter.com/projects/2077260917/harvestgeek-brains-for-your-garden.

GEORGAKAKIS, E. et al. An Analysis of Bluetooth, Zigbee and Bluetooth Low Energy and Their Use in WBANs. In: INTERNATIONAL CONFERENCE ON WIRELESS MOBILE COMMUNICATION AND HEALTHCARE, Berlin, Heidelberg. **Anais...** Springer Berlin Heidelberg, 2011. p.168–175.

GHARAIBEH, A. et al. Smart Cities: a survey on data management, security and enabling technologies. **IEEE Communications Surveys Tutorials**, [S.l.], v.PP, n.99, p.1–1, 2017.

GOOGLE. **What is Google Cloud Pub/Sub?** Disponível em: https://cloud.google.com/pubsub/overview. Acesso em: jun. 2015.

GOOGLE. **Google Cloud Platform - Internet of Things Solutions**. 2016. Disponível em: https://cloud.google.com/solutions/iot.

GOOGLE. **Google Trends - Web Search Interest**: internet of things. 2017. Disponível em: https://goo.gl/NxMpDX.

GOOGLE. **Introducing Google Cloud IoT Core**: for securely connecting and managing iot devices at scale. 2017. Disponível em: https://goo.gl/BCtHDj.

HANSEN, C. J. Internetworking with Bluetooth Low Energy. **GetMobile: Mobile Comp. and Comm.**, New York, NY, USA, v.19, n.2, p.34–38, Aug. 2015.

HENDERSON, I. Planes, Trains and Connected Automobiles: globalizing the iot requires localization. **RFiD Journal**, [S.l.], v.22, 2015.

HERNANDEZ, M. E. P.; REIFF-MARGANIEC, S. Autonomous and Self Controlling Smart Objects for the Future Internet. In: FUTURE INTERNET OF THINGS AND CLOUD (FICLOUD), 2015 3RD INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2015. p.301–308.

HOBSON. **Hobson Platform**. 2016. Disponível em: http://hobson-automation.com.

IBM. **IBM Watson IoT Platform named a leader in IDC MarketScape Report**. 2017. Disponível em: https://www.ibm.com/internet-of-things/platform/watson-iot-platform/.

INSAFUTDINOV, E. et al. DeeperCut: A deeper, stronger, and faster multi-person pose estimation model. **CoRR**, [S.l.], v.abs/1605.03170, 2016.

INTEL. **The Intel Edison Board Made for Brillo**. 2015. Disponível em: https://software.intel.com/en-us/iot/brillo.

INTEL. **Intel Galileo Gen 2 Development Board**. 2015. Disponível em: http://www.intel.com/content/dam/support/us/en/documents/galileo/sb/intelgalileogen2prodbrief_330736_003.pdf.

INTEL. **RAIN DOWN - From The Fighting Temptations Soundtrac**. 2015. Disponível em: https://goo.gl/mQheEu.

INTEL. **IoT Security and Scalability on Intel IoT Platform**. 2016. Disponível em: http://www.intel.com/content/www/us/en/internet-of-things/iot-platform.html.

IROBOT. **Roomba**. 2016. Disponível em: http://www.irobot.com.

JIBO. **Jibo Robot**. 2016. Disponível em: https://www.jibo.com/.

KHAN, R. et al. Future Internet: the internet of things architecture, possible applications and key challenges. In: INTERNATIONAL CONFERENCE ON FRONTIERS OF INFORMATION TECHNOLOGY, 2012. **Anais...** [S.l.: s.n.], 2012. p.257–260.

KIM, J. et al. Standard-based IoT platforms interworking: implementation, experiences, and lessons learned. **IEEE Communications Magazine**, [S.l.], v.54, n.7, p.48–54, July 2016.

KIM, J. Y. et al. Smart home web of objects-based IoT management model and methods for home data mining. In: NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM (APNOMS), 2015 17TH ASIA-PACIFIC. **Anais...** [S.l.: s.n.], 2015. p.327–331.

KRAMP, T.; KRANENBURG, R.; LANGE, S. **Enabling Things to Talk**: designing iot solutions with the iot architectural reference model. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p.1–10.

LAPUT, G.; ZHANG, Y.; HARRISON, C. Synthetic Sensors: towards general-purpose sensing. In: CHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 2017., New York, NY, USA. **Proceedings...** ACM, 2017. p.3986–3999. (CHI '17).

LEE, J. et al. HIS-KCWater: context-aware geospatial data and service integration. In: ACM SYMPOSIUM ON APPLIED COMPUTING, 2007. **Proceedings...** [S.l.: s.n.], 2007. p.24–29.

LEE, N. et al. Smart home web of object architecture. In: INFORMATION AND COMMUNICATION TECHNOLOGY CONVERGENCE (ICTC), 2015 INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2015. p.1212–1214.

LI, R.; LU, B.; MCDONALD-MAIER, K. D. Cognitive assisted living ambient system: a survey. **Digital Communications and Networks**, [S.l.], v.1, n.4, p.229 – 252, 2015.

LI, W.; KARA, S. Methodology for Monitoring Manufacturing Environment by Using Wireless Sensor Networks (WSN) and the Internet of Things (IoT). **Procedia CIRP**, [S.l.], v.61, p.323 – 328, 2017. The 24th CIRP Conference on Life Cycle Engineering.

LIMA, J. P. S. d. M. **Object detection and pose estimation from rectification of natural features using consumer RGB-D sensors**. 2014. Tese (Doutorado em Ciência da Computação) — Universidade Federal de Pernambuco, UFPE.

LIU, K. Pragmatic computing–a semiotic perspective to web services. In: **E-business and Telecommunications**. [S.l.]: Springer, 2007. p.3–15.

LIU, K.; BENFELL, A. Pragmatic web services: a semiotic viewpoint. In: **Software and Data Technologies**. [S.l.]: Springer, 2009. p.18–32.

MACAULAY, J.; BUCKALEW, L.; CHUNG, G. **Internet of Things in Logistics**: a collaborative report by dhl and cisco on implications and use cases for the logistics industry. 2016. Disponível em: http://www.dhl.com/content/dam/Local_Images/g0/New_aboutus/innovation/DHLTrendReport_Internet_of_things.pdf.

MASHAL, I. et al. Choices for interaction with things on Internet and underlying issues. **Ad Hoc Networks**, [S.l.], v.28, p.68 – 90, 2015.

MCTEAR, M. F. Spoken dialogue technology: enabling the conversational user interface. **ACM Computing Surveys (CSUR)**, [S.l.], v.34, n.1, p.90–169, 2002.

MELL, P.; GRANCE, T. The NIST definition of cloud computing. **Communications of the ACM**, [S.l.], v.53, n.6, p.50, 2010.

MICROSOFT. **What is Driver?** 2016. Disponível em: http://windows.microsoft.com/pt-br/windows/what-is-driver#1TC=windows-7.

MICROSOFT. **Microsoft Speech Platform SDK 11 Documentation**. 2016. Disponível em: https://msdn.microsoft.com/en-us/library/dd266409(v=office.14).aspx.

MICROSOFT. **Kinect for Xbox 360**. 2017. Disponível em: http://www.xbox.com/en-US/xbox-360/accessories/kinect.

MININET. **Mininet Tool**. 2016. Disponível em: http://mininet.org.

MQTT. **Specification - MQTT Version 3.1.1**. 2017. Disponível em: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf.

MULLIGAN, G.; GRAČANIN, D. A comparison of SOAP and REST implementations of a service based interaction independence middleware framework. In: SIMULATION CONFERENCE (WSC), PROCEEDINGS OF THE 2009 WINTER. **Anais...** [S.l.: s.n.], 2009. p.1423–1432.

MUNJIN, D.; MORIN, J.-H. User Empowerment in the Internet of Things. **arXiv preprint arXiv:1107.3759**, [S.l.], 2011.

NAM, T.; PARDO, T. A. Building Understanding of Municipal Service Integration: a comparative case study of nyc311 and philly311. In: SYSTEM SCIENCES (HICSS), 2013 46TH HAWAII INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2013. p.1953–1962.

NATIONAL. **LM35 Precision Centigrade Temperature Sensors**. 2015. Disponível em: http://www2.ece.ohio-state.edu/~passino/LM35.pdf.

NEIVA, F. W. et al. Towards Pragmatic Interoperability to Support Collaboration: a systematic review and mapping of the literature. **Information and Software Technology**, [S.l.], 2016.

NEST. **Nest is home**. 2016. Disponível em: https://www.nest.com.

NING, H.; WANG, Z. Future Internet of Things Architecture: like mankind neural system or social organization framework? **IEEE Communications Letters**, [S.l.], v.15, n.4, p.461–463, April 2011.

OPENIOT. **OpenIoT Platform**. 2016. Disponível em: http://www.openiot.eu/.

OPENIOT. **OpenIoT - Open Source Cloud Solution for the Internet of Things**. 2017. Disponível em: http://www.openiot.eu.

PARROT. **Parrot Solution**. 2016. Disponível em: http://www.parrot.com.

PERERA, C. et al. Context Aware Computing for The Internet of Things: a survey. **IEEE Communications Surveys Tutorials**, [S.l.], v.16, n.1, First 2014.

PHILIPS. **Philips Hue**. 2016. Disponível em: http://www2.meethue.com/pt-br.

PLATFORM, L. **LinkSmart® Middleware Platform Portal**. 2016. Disponível em: https://linksmart.eu/redmine/.

PRODANOV, C. C.; FREITAS, E. C. de. **Metodologia do Trabalho Científico**: métodos e técnicas da pesquisa e do trabalho acadêmico-2ª edição. [S.l.]: Editora Feevale, 2013.

QUALCOMM. **DragonBoard 410c**. 2015. Disponível em: https://developer.qualcomm.com/hardware/dragonboard-410c.

RAZA, S. et al. Bluetooth smart: an enabling technology for the internet of things. In: WIRELESS AND MOBILE COMPUTING, NETWORKING AND COMMUNICATIONS (WIMOB), 2015 IEEE 11TH INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2015. p.155–162.

RAZZAQUE, M. A. et al. Middleware for Internet of Things: a survey. **IEEE Internet of Things Journal**, [S.l.], v.3, n.1, p.70–95, Feb 2016.

REALSENSE, I. **Intel RealSense Technology**. 2016. Disponível em: https://www.intel.com/content/www/us/en/architecture-and-technology/realsense-overview.html.

REDMON, J.; FARHADI, A. YOLO9000: better, faster, stronger. **arXiv preprint arXiv:1612.08242**, [S.l.], 2016.

RIVERO, I. Redes de Sensores sem Fio para Monitoramento de Equipamentos Eletrônicos. **Belo Horizonte: Pontifícia Universidade Católica de Minas Gerais**, [S.l.], 2011.

ROBLES, R. J.; KIM, T. hoon. **Applications, Systems and Methods in Smart Home Technology**: a review. 2010. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.359.7648.

ROBOTS, S. of. **Introduction to Microcontrollers**. 2016. Disponível em: http://www.societyofrobots.com/microcontroller_tutorial.shtml.

SAFARIC, S.; MALARIC, K. ZigBee wireless standard. In: NULL. **Anais...** [S.l.: s.n.], 2006. p.259–262.

SALVI, P. **Telemedicina, e-Health e m-Health**: o que elas nos reservam? 2015. Disponível em: http://saudebusiness.com/noticias/telemedicina-saida-para-reducao-de-custos-e-melhoria-no-atendimento.

SAMSUNG. **SmartTV**. 2016. Disponível em: http://www.samsung.com/br/info/privacy/smarttv.html.

SERRANO, M. et al. **IoT Semantic Interoperability**: research challenges, best practices, recommendations and next steps. Berlin, Heidelberg: European Research Cluster on the Internet of Things, 2015. p.8–38.

SETHI, P.; SARANGI, S. R. Internet of Things: architectures, protocols, and applications. **Journal of Electrical and Computer Engineering**, [S.l.], v.2017, 2017.

SHARMA, M.; GROVER, A.; BANDE, P. Low cost sensors for general applications. **International Journal of Recent Trends in Engineering**, [S.l.], v.1, n.5, p.150–152, 2009.

SHELBY, Z.; BORMANN, C. **6LoWPAN**: the wireless embedded internet. [S.l.]: John Wiley & Sons, 2011. v.43.

SHELBY, Z.; HARTKE, K.; BORMANN, C. **The constrained application protocol (CoAP)**. 2014. Disponível em: https://tools.ietf.org/pdf/rfc7252.pdf.

SINGH, J. et al. Semantics-empowered middleware implementation for home ecosystem gateway. In: PERVASIVE COMPUTING AND COMMUNICATIONS WORKSHOPS (PERCOM WORKSHOPS), 2014 IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2014. p.449–454.

SOLIMAN, M. et al. Smart Home: integrating internet of things with web services and cloud computing. In: CLOUD COMPUTING TECHNOLOGY AND SCIENCE (CLOUDCOM), 2013 IEEE 5TH INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2013. v.2, p.317–320.

STAVROPOULOS, T. G. et al. aWESoME: a web service middleware for ambient intelligence. **Expert Systems with Applications**, [S.l.], v.40, n.11, p.4380 – 4392, 2013.

STIPANICEV, D.; BODROZIC, L.; STULA, M. Environmental Intelligence Based on Advanced Sensor Networks. In: SYSTEMS, SIGNALS AND IMAGE PROCESSING, 2007 AND 6TH EURASIP CONFERENCE FOCUSED ON SPEECH AND IMAGE PROCESSING, MULTIMEDIA COMMUNICATIONS AND SERVICES. 14TH INTERNATIONAL WORKSHOP ON. **Anais...** [S.l.: s.n.], 2007. p.209–212.

TAN, L.; WANG, N. Future internet: the internet of things. In: INTERNATIONAL CONFERENCE ON ADVANCED COMPUTER THEORY AND ENGINEERING(ICACTE), 2010. **Anais...** [S.l.: s.n.], 2010. v.5, p.V5–376–V5–380.

TAYLOR, S. The next generation of the Internet revolutionizing the way we work, live, play, and learn. **CISCO Point of View**, [S.l.], 2016. Disponível em: http://www.cisco.com/c/dam/en_us/about/ac79/docs/sp/Next-Generation-of-the-Internet.pdf.

THE ASSOCIATED PRESS, N. S. of. **How Experts Think We'll Live in 2000 A.D**. Disponível em: https://news.google.com/newspapers?id=-eEuAAAAIBAJ&sjid=TU4DAAAAIBAJ&hl=pt-BR&pg=4717%2C6321670. Último Acesso: dez. 2016.

THINGS, M. L. of. **Lab of Things - LoT**. 2016. Disponível em: http://www.lab-of-things.com/.

TILLEY, A. **Google Acquires Smart Thermostat Maker Nest For 3.2 Billion**. 2014. Disponível em: http://www.forbes.com/sites/aarontilley/2014/01/13/google-acquires-nest-for-3-2-billion/#144211d51416.

TUNA, G. et al. A survey on information security threats and solutions for Machine to Machine (M2M) communications. **Journal of Parallel and Distributed Computing**, [S.l.], v.109, p.142 – 154, 2017.

UBIBUS. **Ubibus. An Intelligent Transportation System**. 2015. Disponível em: http://www.cin.ufpe.br/~ubibus/.

VALLATI, C. et al. BETaaS: a platform for development and execution of machine-to-machine applications in the internet of things. **Wireless Personal Communications**, [S.l.], p.1–21, 2015.

VALLI, A. Notes on natural interaction. **Retrieved from on Jan**, [S.l.], v.5, n.2012, p.80, 2005.

VAZ, F. A. Automação Residencial como Auxilio à Segurança Doméstica. In: 2010 3RD . **Anais...** [S.l.: s.n.], 2008.

VERMESAN, O. et al. **Internet of Things Strategic Research and Innovation Agenda**. [S.l.]: River Publishers, 2016. (Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems).

VERMESAN, O.; FRIESS, P. **Internet of Things-From research and innovation to Market Deployment**. [S.l.]: River Publishers, 2014.

VIANI, F. et al. Wireless architectures for heterogeneous sensing in smart home applications: concepts and real implementation. **Proceedings of the IEEE**, [S.l.], v.101, n.11, p.2381–2396, 2013.

WHITMORE, A.; AGARWAL, A.; DA XU, L. The Internet of Things—A survey of topics and trends. **Information Systems Frontiers**, [S.l.], v.17, n.2, p.261–274, 2015.

WOLFRAM RESEARCH, I. **Wolfram Mathematica 10.0**. 2016. Disponível em: https://www.wolfram.com/mathematica/.

WU, M. et al. Research on the architecture of Internet of Things. In: INTERNATIONAL CONFERENCE ON ADVANCED COMPUTER THEORY AND ENGINEERING(ICACTE), 2010. **Anais...** [S.l.: s.n.], 2010. v.5, p.V5–484–V5–487.

XU, L.; COLLIER, R.; O'HARE, G. M. P. A Survey of Clustering Techniques in WSNs and Consideration of the Challenges of Applying Such to 5G IoT Scenarios. **IEEE Internet of Things Journal**, [S.l.], v.PP, n.99, p.1–1, 2017.

YAQOOB, I. et al. Internet of Things Architecture: recent advances, taxonomy, requirements, and open challenges. **IEEE Wireless Communications**, [S.l.], v.24, n.3, 2017.

YU, J. et al. IoT as a applications: cloud-based building management systems for the internet of things. **Multimedia Tools and Applications**, [S.l.], p.1–14, 2015.

YU, J. et al. Adaptive Internet of Things and Web of Things convergence platform for Internet of reality services. **The Journal of Supercomputing**, [S.l.], v.72, n.1, p.84–102, 2015.

ZARGHAMI, S. **Middleware for Internet of things**. 2017. Disponível em: http://essay.utwente.nl/64431/.

ZHOU, J. et al. CloudThings: a common architecture for integrating the internet of things with cloud computing. In: COMPUTER SUPPORTED COOPERATIVE WORK IN DESIGN (CSCWD), 2013 IEEE 17TH INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2013. p.651–657.