

UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Programa de Pós-Graduação em Computação



Dissertação

**EXEHDA-IoT: Uma Contribuição à Arquitetura do Middleware EXEHDA
Direcionada à Internet das Coisas**

Patrícia Teixeira Davet

Pelotas, 2016

Patrícia Teixeira Davet

**EXEHDA-IoT: Uma Contribuição à Arquitetura do Middleware EXEHDA
Direcionada à Internet das Coisas**

Dissertação apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Mestre em Ciência da Computação

Orientador: Prof. Dr. Adenauer Corrêa Yamin
Coorientadora: Profa. Dra. Ana Marilza Pernas Fleischmann

Pelotas, 2016

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação na Publicação

D246e Davet, Patrícia Teixeira

EXEHDA-IoT: Uma Contribuição à Arquitetura do Middleware EXEHDA Direcionada à Internet das Coisas / Patrícia Teixeira Davet; Adenauer Corrêa Yamin, orientador; Ana Marilza Pernas Fleischmann, coorientadora. – Pelotas, 2016.

109 f.

Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2016.

1. Ciência de Contexto. 2. Internet das Coisas.
3. Computação Ubíqua. I. Yamin, Adenauer Corrêa, orient.
II. Fleischmann, Ana Marilza Pernas, coorient. III. Título.

CDD: 005



Universidade Federal de Pelotas
Centro de Desenvolvimento Tecnológico
Programa de Pós-Graduação em Computação

ATESTADO

ATESTO para os devidos fins, que **PATRÍCIA TEIXEIRA DAVET** foi aprovado pela Banca Examinadora da Defesa de Dissertação realizada em 4 de Março de 2016, intitulada: "EXEHDA-IoT: Uma Contribuição à Arquitetura do Middleware EXEHDA Direcionada à Internet das Coisas", sob a orientação do Prof. Dr. Adenauer Corrêa Yamin. Realizadas as correções recomendadas pela Comissão Examinadora, o trabalho será encaminhado para homologação pelo Colegiado do Programa de Pós-Graduação em Computação. Igualmente, informamos que os demais requisitos necessários para a obtenção do grau de **Mestre em Ciência da Computação** pela Universidade Federal de Pelotas foram cumpridos.

Pelotas, 4 de Março de 2016

Ricardo Matsumura de Araujo
Coordenador do Programa de Pós-Graduação em Computação

AGRADECIMENTOS

É com alegria, que findada esta etapa de muito aprendizado e experiências fortalecedoras, compartilho os meus sinceros agradecimentos a todos que fizeram parte deste importante momento em minha vida.

Primeiramente, agradeço à Deus, que está sempre a nos acompanhar, acolher, e sem o qual nada seria possível.

Agradeço também, ao meu orientador e principal incentivador, Professor Adenauer Corrêa Yamin, que com sua impressionante capacidade de transmitir sabedoria em muitas áreas do conhecimento, dignifica a profissão com seu dom de ensinar e fazer a diferença na vida dos seus alunos e orientandos.

Aos meus familiares, por todo carinho e apoio a mim externado.

Aos meus queridos parceiros do grupo de pesquisa LUPS, em especial ao Huberto Kaiser, pela sua importante colaboração no desenvolvimento deste trabalho, bem como ao Lucas Xavier, Tainã Carvalho, Leonardo João, Jerônimo Ramos, Lidiane da Silva, Rodrigo Duarte, Vilnei Neves, Maicon dos Santos e Giovane Torres pela amizade, companheirismo e apoio.

Aos professores André Du Bois, Gérson Cavalheiro, Maurício Pilla e Renata Reiser e a todos os colegas de mestrado pela oportunidade de convívio e aprendizado.

Aos doutorandos João Ladislau, Rodrigo Souza e a profa. Ana Pernas, pela amizade, apoio e estimada contribuição a esta dissertação.

A Embrapa Clima Temperado e a seus funcionários, em especial a Nelsi, grande amiga e exemplo de dedicação profissional, agradeço pela oportunidade de realização de estudo de caso em seus laboratórios.

Por fim, agradeço a todos aqueles que não foram mencionados, mas que de alguma forma também contribuíram para a elaboração deste trabalho.

RESUMO

DAVET, Patrícia Teixeira. **EXEHDA-IoT: Uma Contribuição à Arquitetura do Middleware EXEHDA Direcionada à Internet das Coisas**. 2016. 109 f. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2016.

Na Computação Ubíqua (UbiComp) os serviços computacionais são disponibilizados ao usuário de forma o mais transparente possível, integrando tecnologias ao seu cotidiano sem que este precise ter conhecimento de toda a infraestrutura computacional envolvida. Uma das principais frentes de pesquisa da UbiComp está relacionada à necessidade das aplicações terem ciência do seu contexto de interesse e, quando for o caso, reagirem ao mesmo.

Os recentes avanços na área da “Internet das Coisas -IoT” têm proporcionado uma crescente incorporação de dispositivos computacionais com capacidades de processamento, comunicação e sensoriamento a objetos cotidianos, os quais constituem fontes geradoras de informações contextuais distribuídas, caracterizando a IoT como uma estratégia promissora enquanto alternativa para materialização da UbiComp.

No entanto, para o desenvolvimento de aplicações cientes de contexto em cenários introduzidos pela IoT, uma série de desafios devem ser superados. Dentre estes, está a concepção de infraestruturas que sejam capazes de adquirir informações contextuais a partir de um grande número de dispositivos heterogêneos, distribuídos e com capacidade restrita.

Considerando isto, o objetivo central desta dissertação de mestrado é a concepção de mecanismos destinados ao atendimento das demandas inerentes a IoT, e cujo desenvolvimento deverá ter como premissa sua integração à arquitetura de software do middleware EXEHDA.

De modo mais específico, esta dissertação de mestrado contribuiu com o Subsistema de Adaptação e Reconhecimento de Contexto do EXEHDA em três grandes aspectos: (i) provendo suporte a heterogeneidade das diversas tecnologias de hardware e software inerentes aos dispositivos empregados em IoT, (ii) agregando identificação e descoberta automática dos dispositivos de sensoriamento e atuação e, (iii) padronizando a forma de interoperação entre os componentes da arquitetura de software.

Na avaliação das funcionalidades do EXEHDA-IoT, foi empregado um estudo de caso no domínio da agricultura, desenvolvido junto a Embrapa Clima Temperado. Os resultados obtidos estimulam a continuidade das pesquisas na área.

Palavras-chave: Ciência de Contexto, Internet das Coisas, Computação Ubíqua.

ABSTRACT

DAVET, Patrícia Teixeira. **EXEHDA-IoT: A Contribution to the Architecture of the Middleware EXEHDA Addressed to the Internet of Things**. 2016. 109 f. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2016.

In Ubiquitous Computing (UbiComp) computational services are made available at the user of way more as transparent as possible, integrating technologies into their daily lives without the need to have knowledge of the entire computing infrastructure involved. One of the main research fronts of UbiComp is related to the need of the applications have knowledge of its context of interest, and when necessary, react to it.

Recent advances in the area of “Internet of Things -IoT” have provided a growing incorporation of computational devices with processing capabilities, communication and sensing the everyday objects, which are sources of contextual information distributed, featuring the IoT as a promising strategy while alternative for materialization of the UbiComp.

However, for the development of context-aware applications in scenarios introduced by IoT, a number of challenges must be overcome. Among these is the concept of infrastructure capable of acquiring contextual information from a number of heterogeneous devices, and distributed with limited capacity.

Considering this, the main objective of this dissertation is to design mechanisms to meet the demands inherent in IoT, and whose design should to have as premise their integration into the software architecture of the a middleware.

More specifically, this master thesis contributed to the Subsystem of Context Adaptation and Recognition of the EXEHDA in three major aspects: (i) providing support for heterogeneity of the various hardware and software technologies inherent in the devices used in IoT, (ii) adding identification and automatic discovery of sensing devices and actuation (iii) standardizing the form of interoperability between components of the software architecture.

In evaluating the EXEHDA-IoT functionality it was used a case study in the field of agriculture, developed with Embrapa Clima Temperado. The results stimulate the continuity of research in the area.

Keywords: Context-Aware, Internet of Things, Ubiquitous Computing.

LISTA DE FIGURAS

Figura 1	Definição de Internet das Coisas	19
Figura 2	Árvore IoT-A - Visão do Projeto	21
Figura 3	Dimensão de um Cenário IoT	24
Figura 4	Pilha de Protocolos UPnP	30
Figura 5	Formato da Mensagem CoAP	35
Figura 6	Cabeçalho Fixo de uma Mensagem MQTT	39
Figura 7	Arquitetura EcoDiF	42
Figura 8	Níveis de Abstração S ³ OIA	44
Figura 9	Arquitetura S ³ OIA	46
Figura 10	Arquitetura LinkSmart	48
Figura 11	Exemplo de Hierarquia das Entidades na Plataforma Carriots	49
Figura 12	Arquitetura Carriots	51
Figura 13	Arquitetura do Middleware EXEHDA	54
Figura 14	Ambiente Ubíquo: Organização Proposta para o EXEHDA-IoT	56
Figura 15	Arquitetura do Subsistema de Adaptação e Reconhecimento do Contexto do Middleware EXEHDA	59
Figura 16	Processamento de uma Regra Tipo ECA (Notação BPMN)	60
Figura 17	Notificação de Gateway Descoberto	61
Figura 18	Arquivo de Classe de um Sensor de Temperatura	67
Figura 19	Estudo de Caso - Projeto plenUS	71
Figura 20	Dispositivos Utilizados	79
Figura 21	Menu da Aplicação de Gerenciamento	81
Figura 22	Tela de Gerência dos Servidores de Borda	82
Figura 23	Tela de Gerência dos Gateways	82
Figura 24	Tela de Gerência dos Sensores	83
Figura 25	Tela de Gerência dos Tipos de Sensores	83
Figura 26	Notificação de Registro de Agendamento	84
Figura 27	Visualização Textual das Informações Contextuais	85
Figura 28	Visualização Gráfica das Informações Contextuais	85

LISTA DE TABELAS

Tabela 1	Utilização dos Tipos de Mensagens CoAP	35
Tabela 2	Códigos dos Métodos de Requisição CoAP	36
Tabela 3	Códigos de Resposta CoAP	37
Tabela 4	Tipos de Mensagem de Controle	40
Tabela 5	Níveis de QoS	40
Tabela 6	Análise dos Trabalhos Relacionados	51
Tabela 7	Coleta de Informações Contextuais no LASO	77

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
CoAP	Constrained Application Protocol
CoRE	Constrained RESTful Environment
DLNA	Digital Living Network Alliance
DPWS	Device Profile for Web Services
DTLS	Datagram Transport Layer Security
EEML	Extended Environments Markup Language
Embrapa	Empresa Brasileira de Pesquisa Agropecuária
EMML	Enterprise Mashup Markup Language
EPC	Electronic Product Code
EXEHDA	Execution Environment for Highly Distributed Applications
EXEHDA-IoT	EXEHDA-Internet of Things
GENA	Generic Event Notification Architecture
HTML	HyperText Markup Language
HTTPU	HyperText Transfer Protocol Unicast
HTTPMU	HyperText Transfer Protocol Multicast
IETF	Internet Engineering Task Force
IoT	Internet of Things
JSON	JavaScript Object Notation
LASO	Laboratório de Análise de Sementes Oficial
LUPS	Laboratory of Ubiquitous and Parallel Systems
MAPA	Ministério de Agricultura Pecuária e Abastecimento
MIT	Massachusetts Institute of Technology
MQTT	Message Queue Telemetry Transport
MySQL	My Structured Query Language
NFC	Near Field Communication

plenUS	plentiful Ubiquitous Systems
REST	Representational State Transfer
RDF	Resource Description Framework
RFID	Radio Frequency Identification
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SSDP	Simple Service Discovery Protocol
SSL	Secure Sockets Layer
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS	Transport Layer Security
UbiComp	Ubiquitous Computing
UDP	User Datagram Protocol
UPC	Universal Product Code
UPnP	Universal Plug and Play
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XML	eXtensible Markup Language

SUMÁRIO

1	INTRODUÇÃO	13
1.1	Tema	14
1.2	Motivação e Objetivos	14
1.3	Estrutura do Texto	15
2	ESCOPO DO TRABALHO	16
2.1	Internet das Coisas	16
2.1.1	Consolidação da IoT Enquanto Área de Pesquisa e Aplicação	17
2.1.2	Visão Geral das Frentes de Trabalho em IoT	18
2.1.3	Desafios de Pesquisa em IoT	20
2.2	Computação Ciente do Contexto	24
2.2.1	Aquisição de Informações Contextuais	24
2.2.2	Modelagem do Contexto	26
2.2.3	Processamento do Contexto	26
2.3	Protocolos de Comunicação na IoT	28
2.3.1	UPnP	28
2.3.2	CoAP	33
2.3.3	MQTT	36
2.4	Considerações sobre o Capítulo	40
3	TRABALHOS RELACIONADOS	41
3.1	EcoDiF - Ecossistema Web de Dispositivos Físicos	41
3.2	S ³ OIA - Smart Spaces and Smart Objects Interoperability Architecture	44
3.3	LinkSmart	46
3.4	Carriots	48
3.5	Análise dos Trabalhos Relacionados	51
3.6	Considerações sobre o Capítulo	52
4	EXEHDA-IOT: VISÃO GERAL E MODELAGEM	53
4.1	EXEHDA-IoT: Visão Geral	53
4.2	EXEHDA-IoT: Modelo Arquitetural	55
4.2.1	Servidor de Contexto	58
4.2.2	Servidor de Borda	61
4.2.3	Gerência e Descoberta de Dispositivos	64
4.2.4	Protocolos Suportados	67
4.3	Discussão sobre os Trabalhos Relacionados	68
4.4	Considerações sobre o Capítulo	69

5	EXEHDA-IOT: ESTUDO DE CASO	70
5.1	Estudo de Caso: Projeto plenUS	70
5.1.1	Laboratório de Qualidade do Leite	72
5.1.2	Laboratório de Reprodução Animal	74
5.1.3	Laboratório de Análise de Sementes Oficial	76
5.1.4	Infraestrutura de Hardware e Software Desenvolvidas	78
5.1.5	Funcionalidades Disponibilizadas	80
5.2	Considerações sobre o Capítulo	86
6	CONSIDERAÇÕES FINAIS	87
6.1	Principais Conclusões	87
6.2	Publicações Realizadas	89
6.3	Trabalhos Futuros	90
	REFERÊNCIAS	91
	ANEXO A COMPONENTES DESENVOLVIDOS PARA O EXEHDA-IOT	96
	ANEXO B REPOSITÓRIO DE INFORMAÇÕES CONTEXTUAIS	108

1 INTRODUÇÃO

A popularização da Internet e dos ambientes distribuídos, aliados aos avanços tecnológicos que proporcionaram mobilidade e melhora na conectividade e comunicação dos dispositivos computacionais, levou a computação moderna a uma nova era, a da Computação Ubíqua (UbiComp). Este paradigma introduzido por Weiser no seu visionário trabalho (WEISER, 1991), tem como premissa prover computação de forma transparente, estando o modelo computacional integrado as rotinas de vida dos seus usuários (CACERES; FRIDAY, 2011). Os mesmos avanços que contribuíram para a evolução de uma nova era na computação, somado aos avanços e miniaturização de dispositivos embarcados, introduzem um novo cenário de evolução da Internet, a Internet das Coisas (IoT).

A Internet das Coisas se caracteriza por um conjunto de objetos ou dispositivos interconectados à Internet, unicamente identificados, e com capacidade para coletar e trocar dados entre si e com o usuário, independentemente de localização, rede, dispositivo ou tecnologias empregadas, o que a torna bastante associada aos preceitos da UbiComp. Esta associação, que aproveita a ubiquidade inerente à infraestrutura computacional provida pela Internet, faz com que a IoT esteja ganhando dimensão prática enquanto uma abordagem para a materialização da UbiComp.

As aplicações originadas deste novo tipo de informação introduzida pelo cenário da IoT, enquanto ubíquas e assim dotadas de comportamento autônomo, e com uma organização distribuída, necessitam ter ciência dos dados contextuais que lhe interessam e, quando for o caso, reagirem aos mesmos. Esta classe de sistemas computacionais, reativos ao contexto, abre perspectivas ao desenvolvimento de aplicações mais ricas, elaboradas e complexas, e que exploram a natureza dinâmica das modernas infraestruturas computacionais, bem como a mobilidade do usuário (CACERES; FRIDAY, 2011).

Porém, para se construir e executar aplicações IoT cientes de contexto, há uma série de funcionalidades que devem ser providas, envolvendo desde a aquisição de informações contextuais adquiridas por sensores, a partir de um conjunto de fontes heterogêneas e distribuídas, até a representação dessas informações, seu processa-

mento, armazenamento, e realização de inferências para seu uso em procedimentos de atuação (PERERA et al., 2013).

O desafio encontra-se em como gerenciar os aspectos inerentes às modernas infraestruturas da Internet das Coisas, fazendo com que os dispositivos comumente restritos, tanto do ponto de vista computacional como energético possam interoperar de forma mais otimizada possível.

Uma alternativa de solução, que vem ganhando destaque para endereçar tais desafios, está na utilização de middlewares (PIRES et al., 2015).

1.1 Tema

Esta dissertação de mestrado tem como tema central de pesquisa o tratamento de desafios inerentes a concepção de infraestruturas direcionadas à Internet das Coisas. Será considerada a experiência do grupo de pesquisa (LUPS - *Laboratory of Ubiquitous and Parallel Systems*) na concepção de middlewares para UbiComp, em particular aquela relativa ao provimento de soluções voltadas à ciência de contexto.

1.2 Motivação e Objetivos

A IoT tem despertado interesse mundial devido ao crescente número de dispositivos que vem sendo interconectados à Internet em todo o mundo. Segundo (SRI Consulting Business Intelligence, 2008), daqui a alguns anos, qualquer objeto do dia a dia poderá ser integrado em um ambiente IoT.

Porém ainda existem questões de pesquisa em aberto que motivam os estudos na área e que constituem barreira para a sua plena implantação.

Considerando esta motivação, o objetivo central deste trabalho denominado EXEHDA-IoT é a concepção de mecanismos destinados ao atendimento das demandas inerentes à IoT, que sejam capazes de integrar diferentes tecnologias, tanto de hardware como de software, bem como de protocolos de comunicação e formatos de dados. Sendo que esta integração deve considerar além da heterogeneidade de tecnologias, um ambiente altamente distribuído, escalável, dinâmico e mutável.

A concepção destes mecanismos deverá ter como premissa a sua integração à arquitetura de software do middleware EXEHDA (*Execution Environment for Highly Distributed Applications*) (LOPES et al., 2014), contribuindo especificamente ao seu Subsistema de Adaptação e Reconhecimento do Contexto.

Como objetivos específicos, são destacadas as seguintes frentes nos esforços de estudo e pesquisa desta dissertação:

- sistematizar trabalhos científicos em relação ao tema de pesquisa proposto, caracterizando o estado da arte na área;

- prover suporte no EXEHDA à heterogeneidade das diversas tecnologias inerentes aos dispositivos empregados em IoT;
- agregar no EXEHDA a possibilidade de identificação e descoberta automática dos dispositivos de sensoriamento e atuação, provendo para as aplicações uma visão que abstraia aspectos tecnológicos destes dispositivos, facilitando seu uso e gerência;
- realizar padronização da forma de interoperação entre os componentes da arquitetura de software do middleware EXEHDA, de forma a garantir as funcionalidades previstas, bem como considerando as tendências internacionais;
- conceber a infraestrutura de hardware e software necessária para aquisição das informações contextuais disponibilizando-as aos componentes do middleware, bem como o de atuação no meio físico;
- implementar estudo de caso na Embrapa Clima Temperado Pelotas, a fim de avaliar as funcionalidades propostas;
- divulgar na comunidade científica os resultados atingidos pela pesquisa, por meio de publicações de artigos em eventos da área.

1.3 Estrutura do Texto

O texto é composto por 6 capítulos. No capítulo 1 são apresentados o tema, motivação e objetivos que caracterizam a proposta desta dissertação de mestrado. Tópicos referentes as áreas de interesse como a Internet das Coisas e Computação Ciente de Contexto são revisados no Capítulo 2, bem como os principais protocolos de comunicação em uso atualmente na IoT, no que concerne a interoperabilidade de dispositivos restritos, como também características de zero configuração.

Com o intuito de caracterizar o estado da arte, o Capítulo 3 descreve e analisa alguns trabalhos relacionados. Uma visão geral e os principais aspectos quanto a concepção e modelagem do EXEHDA-IoT, bem como uma discussão dos trabalhos relacionados em relação as estratégias concebidas são apresentados no Capítulo 4. Um estudo de caso na agricultura é discutido no Capítulo 5. Por fim o Capítulo 6 apresenta as principais considerações, publicações realizadas e trabalhos futuros.

2 ESCOPO DO TRABALHO

Neste capítulo são apresentados conceitos caracterizados como importantes durante a revisão de literatura em relação ao tema proposto, e que constituem a fundamentação teórica desta dissertação de mestrado. Na primeira seção a Internet das Coisas e os seus desafios de pesquisa são apresentados. O modelo de computação Ciente de Contexto é discutido na segunda seção, o qual apresenta-se como um dos pilares da UbiComp e vem sendo um componente chave para viabilizar sistemas inteligentes na IoT. Por fim, na terceira seção são resumidas as principais características de três protocolos de comunicação IoT, os quais foram destacados pela literatura e considerados neste trabalho.

2.1 Internet das Coisas

A Internet das Coisas, do inglês *Internet of Things* - IoT, vem ganhando destaque como o novo paradigma de evolução da Internet (PERERA et al., 2013). A consequência desta evolução deve-se ao fato que a IoT preconiza a ideia do tudo conectado, ou seja qualquer “coisa”, tais como utensílios, equipamentos pessoais, além de sistemas de transportes e agrícolas, redes de energia, como também animais e o nosso corpo podem possuir sensores capazes de gerar informações que são compartilhadas tendo como principal meio de interoperação, a Internet.

O compartilhamento destas informações disponibilizadas pelo cenário da IoT ocorre automaticamente com outros sistemas computacionais externos, o que introduz o conceito de objetos inteligentes (*smart objects*). Estes objetos inteligentes ou simplesmente “coisas” (MIORANDI et al., 2012) devem ser capazes de detectar fenômenos (físicos ou lógicos) e/ou disparar ações sobre o ambiente, bem como possuir alguma capacidade computacional e de comunicação, além de um identificador único que os capacitem a interoperarem de forma o mais autônoma possível.

A IoT tem despertado interesse mundial pelo crescente número de dispositivos que vem sendo interconectados à Internet, gerando novas informações que podem agregar conhecimento para os mais diferentes tipos de aplicações. Por ser uma área

relativamente nova, ainda possui problemas de pesquisa em aberto, o que estimula o envolvimento da comunidade científica no tema. A IoT também tem sido considerada como uma abordagem para promover a ubiquidade das soluções computacionais no mundo real, principalmente em relação aos aspectos infraestruturais (CACERES; FRIDAY, 2011).

Com essa característica, a IoT vem constituindo-se elemento essencial para consolidar a integração entre os sistemas computacionais e o ambiente físico (ATZORI; IERA; MORABITO, 2010) (PERERA et al., 2013) (PIRES et al., 2014).

2.1.1 Consolidação da IoT Enquanto Área de Pesquisa e Aplicação

O termo Internet das Coisas foi cunhado no laboratório de pesquisa Auto-ID Center do MIT (*Massachusetts Institute of Technology*) e foi mencionado pela primeira vez em 1998 por Kevin Ashton, em uma apresentação onde relata “A Internet das Coisas tem o potencial de mudar o mundo, assim como a Internet fez. Talvez até mais” (ASHTON, 2009).

Em 2001 o termo surge publicado, através do artigo intitulado “*The Electronic Product Code (EPC) : A naming Scheme for Physical Objects*” (BROCK, 2001). No artigo o autor introduz este novo conceito relatando “*Nossa visão é a de criar um Smart World, isto é, uma infraestrutura inteligente que liga objetos, informações e pessoas através da rede de computadores. Esta nova infraestrutura permitirá a coordenação universal de recursos físicos através do monitoramento e controle remoto por seres humanos e máquinas. Nosso objetivo é criar padrões abertos, protocolos e linguagens para facilitar a ampla adoção mundial desta rede - formando a base para uma nova **Internet das coisas***”. Para tal o artigo propõe a substituição do popular código de barras, chamado sistema UPC (*Universal Product Code*), por um sistema que usufruiria das capacidades da Internet e da globalização.

O sistema EPC proposto, apresentou como vantagem a possibilidade de cada produto possuir uma identificação própria, ao contrário do UPC que identificava apenas a qual tipo o produto pertencia, além de realizar o acompanhamento do objeto físico ao longo do seu tempo de vida através de tags RFID. A informação sobre o seu estado, localização e outras informações úteis não se encontram diretamente armazenadas nos produtos, e sim em servidores acessíveis pela Internet.

O termo Internet das Coisas surgiu, quando o grupo de pesquisadores do MIT estava trabalhando em novas tecnologias de sensores e tags inteligentes (RFIDs), porém a IoT começou a ganhar destaque devido ao crescente número de dispositivos computacionais que estavam sendo conectados em todo o mundo.

Nesta perspectiva, relata-se que a IoT surge no momento exato em que o número de dispositivos conectados à Internet supera o número de pessoas que habitam o planeta.

De acordo com pesquisa do grupo Cisco IBSG (EVANS, 2011), em 2003 havia aproximadamente 6,3 bilhões de pessoas vivendo no planeta e 500 milhões de dispositivos conectados à Internet, resultando numa relação de aproximadamente 0,08 dispositivo por pessoa. Com base nestes dados a IoT não era uma realidade em 2003, pois o número de dispositivos conectados era relativamente pequeno.

O crescimento do uso de *smartphones e tablets* elevou o número de dispositivos conectados à Internet para 12,5 bilhões em 2010, à medida que a população humana chegou a 6,8 bilhões, tornando o número de dispositivos conectados por pessoa superior a 1 (exatamente 1,84). No entanto, segundo o Cisco IBSG ao realizar um refinamento destes números, a IoT surge entre 2008 e 2009, período onde estima-se que efetivamente o número de dispositivos por pessoa tenha ultrapassado a 1 pela primeira vez na história.

Nos primeiros dias de 2015, o mundo registrou 25 bilhões de dispositivos conectados à Internet segundo uma das maiores feiras de tecnologias do mundo, a CES 2015 (*Consumer Electronics Show 2015*), denotando que a Internet das Coisas vem ganhando dimensão prática, de forma a se tornar a revolução tecnológica que representa um dos futuros possíveis da computação e comunicação.

2.1.2 Visão Geral das Frentes de Trabalho em IoT

A IoT apresenta três visões de pesquisa que se diferenciam dependendo do termo empregado: Internet, Coisas ou Internet das Coisas (ATZORI; IERA; MORABITO, 2010), (GUBBI et al., 2013) levando a focos distintos, conforme descrito a seguir.

- Visão orientada à **Internet**: possui foco de pesquisa do ponto de vista de redes. Pesquisas direcionadas a esta visão procuram criar modelos e técnicas para a interoperabilidade dos dispositivos em rede.
- Visão orientada às **Coisas**: objetiva a integração dos objetos (coisas) presentes em um cenário IoT. Pesquisas nesta visão procuram apresentar propostas que garantam o melhor aproveitamento dos recursos dos dispositivos e sua comunicação.
- Visão orientada à **Semântica**: esta visão parte de uma análise semântica da expressão composta “Internet das Coisas”. Portanto possui foco de pesquisa do ponto de vista da comunicação e troca de informações entre dispositivos distintos. Trabalhos nesta visão apresentam propostas que estão focadas na representação, armazenamento, interconexão, pesquisa e organização da informação gerada na IoT, buscando soluções para a modelagem das descrições que permitam um tratamento adequado para os dados gerados pelos objetos.

As definições para o termo IoT foram se consolidando em função das diferentes visões de pesquisa, podendo estas caracterizar mais de uma visão. Três definições de IoT se destacam na literatura da área. O fato de não existir uma definição única para o termo IoT, segundo (PERERA et al., 2013) ocorre por ser uma área de pesquisa recente e bastante ampla. A seguir estão resumidas estas diferentes definições encontradas na literatura para o termo IoT.

- **Definição 1:** “As coisas (objetos, pessoas ou animais) têm identidades e personalidades virtuais que operam em espaços inteligentes utilizando interfaces capazes de se conectar e comunicar dentro de contextos sociais, ambientais e de usuários” (TAN; WANG, 2010).
- **Definição 2:** “A origem semântica da expressão é composta por duas palavras que traduzem dois conceitos: Internet e Thing. Internet pode ser definida como uma rede mundial de computadores interligados, com base em um protocolo de comunicação padrão (TCP/IP), enquanto Thing é um objeto ou coisa qualquer. Portanto, semanticamente, Internet das Coisas pode ser definida como uma rede mundial de objetos interligados que possuem endereço único e que se comunicam através da Internet por meio de protocolos de comunicação padronizados” (BASSI; HORN, 2008).
- **Definição 3:** “A Internet das Coisas permite que pessoas e objetos possam se conectar a qualquer momento, em qualquer lugar, com qualquer coisa, de preferência usando qualquer caminho ou rede e qualquer serviço” (GUILLEMIN; FRIESS, 2009).

Dentre as definições apresentadas, a que se mostrou mais alinhada com as premissas da Computação Ubíqua, e por consequência com o escopo deste trabalho é a Definição 3, a qual é ilustrada pela Figura 1.

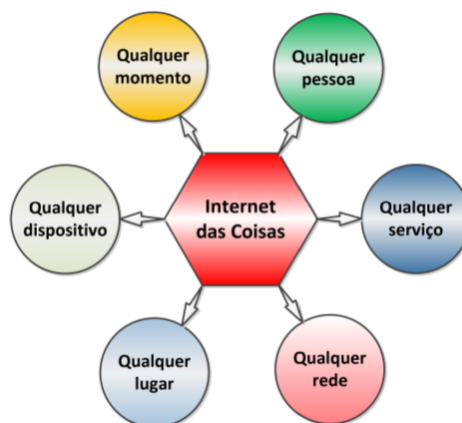


Figura 1: Definição de Internet das Coisas. Adaptada de (GUILLEMIN; FRIESS, 2009)

Evidentemente essa é uma definição idealizada, pois em ambientes ubíquos típicos, as desconexões podem ocorrer com relativa frequência devido à mobilidade dos dispositivos ou seu desligamento, falta de energia ou então por decorrência de avaria física. Portanto, desconexões não devem ser consideradas falhas mas sim parte da especificação e devem ser tratadas de maneira mais transparente possível pelos sistemas de gerência (COSTA; YAMIN; GEYER, 2008).

2.1.3 Desafios de Pesquisa em IoT

O grande facilitador para a integração destes diversos dispositivos independentemente de sua localização, por ser uma rede global, foi a Internet. Porém alguns fatores são tidos como importantes para a efetiva materialização da IoT, como os avanços tecnológicos na miniaturização de dispositivos embarcados e agregação de novos e diferentes sensores, atuadores e tags inteligentes. No entanto, há ainda uma série de desafios a serem superados para alavancar a ampla disseminação da IoT.

Um dos principais desafios inerentes em um cenário IoT, está relacionado com a alta heterogeneidade decorrente da diversidade de tecnologias de hardware e software presentes neste ambiente. Esta situação demanda que haja uma busca de soluções que permitam a interoperabilidade e integração destes diferentes componentes.

Uma alternativa de solução promissora que vem sendo amplamente utilizada, de forma a tratar desafios IoT, como o da heterogeneidade, está na utilização de plataformas de middleware (PIRES et al., 2015).

As plataformas de middleware são inseridas entre as aplicações e a infraestrutura (de comunicação, processamento e sensoriamento) subjacente, provendo um meio padronizado para o acesso aos dados e serviços fornecidos pelos objetos por meio de uma interface de alto nível (BANDYOPADHYAY et al., 2011).

O projeto europeu IoT-A (*Internet of Things - Architecture*) (IOT-A, 2014), que teve como principal objetivo criar uma arquitetura de referência para a Internet das Coisas destaca-se por tratar de maneira ampla os desafios na concepção de infraestruturas IoT, além de possuir uma visão arquitetural a qual é utilizada para caracterizar um cenário IoT.

A visão do projeto, apresentada na Figura 2 é representada simbolicamente por uma árvore, onde as raízes abrangem todo um conjunto de tecnologias, como os de protocolos de comunicação (6LoWPAN, Zigbee, IPv6, etc.) e de dispositivos (sensores, atuadores, etiquetas RFID, dentre outros) representando a heterogeneidade presente em um cenário IoT, enquanto as flores/folhas da árvore demonstram todo o conjunto de aplicações em seus mais diferentes domínios, que podem ser construídos a partir da seiva (i.e., dados e informações) fornecidos pela raiz.

Nesta visão arquitetural percebe-se a necessidade de um elo de ligação entre os

componentes representados pelas raízes e folhas, e que constitui a solução para grande parte dos desafios IoT. O tronco caracteriza este elo e representa uma plataforma de middleware.



Figura 2: Árvore IoT-A - Visão do Projeto. Fonte: (IOT-A, 2014)

De acordo com a literatura (CHAQFEH; MOHAMED et al., 2012), (BANDYO-PADHYAY et al., 2011), (NAGY et al., 2009), alguns desafios inerentes à ambientes IoT merecem destaque e constituem requisitos importantes para a proposição de um middleware para IoT.

Interoperabilidade

Dentre os requisitos, o considerado primordial e que deve ser imperativamente endereçado por uma plataforma de middleware para a IoT, diz respeito a interoperabilidade entre os diversos dispositivos e plataformas disponíveis neste ambiente. A sua importância é devido a um número crescente de dispositivos a serem integrados neste novo cenário, sendo estes heterogêneos tanto em termos de hardware quanto de software, como também de protocolos (muitos deles proprietários) e formatos de dados, o que caracteriza a interoperabilidade em um cenário IoT um grande desafio.

Em (PIRES et al., 2015) é destacado que a integração de dispositivos no contexto de IoT perpassa múltiplos níveis:

- em mais baixo nível, é necessário integrar, de maneira transparente, uma miríade de dispositivos físicos heterogêneos de modo a ocultar detalhes com relação à rede, aos formatos de dados empregados, e até mesmo à semântica das informações;
- em um nível intermediário, a fim de prover serviços de valor agregado aos usuários, é necessário integrar e disponibilizar dados providos por esses dispositivos, podendo incluir simples funções de processamento de dados ou mesmo aplicações Web mais complexas;
- por fim, em mais alto nível, um modelo padronizado de programação pode promover integração no que se refere à agregação e transformação de informações providas pelos dispositivos, de modo que desenvolvedores de aplicações não necessitam ter qualquer conhecimento acerca das especificidades dos dispositivos físicos e do ambiente de rede subjacente.

Descoberta e Gerenciamento de dispositivos

Uma outra característica comum à ambientes de IoT leva em conta o fato da sua infraestrutura de comunicação possuir uma topologia dinâmica e frequentemente desconhecida, visto que dispositivos podem ser integrados ao ambiente e utilizados de maneira oportunista e não previamente planejada (BANDYOPADHYAY et al., 2011).

Considerando esta dinamicidade da IoT, é importante que uma plataforma de middleware possibilite a descoberta de dispositivos presentes no ambiente em questão, realizada dinamicamente a fim de atender os requisitos das aplicações.

Também destaca-se a necessidade de prover mecanismos para o gerenciamento de dispositivos, que diz respeito à capacidade de fornecer informações de localização e estado do dispositivo permitindo, dentre outras funcionalidades, desconectar algum dispositivo roubado ou não reconhecido, atualizar software embarcado, modificar configurações de segurança, alterar remotamente configurações de hardware, localizar um dispositivo perdido, apagar dados sensíveis de dispositivos, e até mesmo possibilitar a interação entre dispositivo.

Interfaces de Alto Nível

A adoção de uma plataforma de middleware também pode contribuir para facilitar a construção de aplicações para IoT. Desta forma, o desafio reside no fato de que, a fim de permitir a criação de aplicações que combinem recursos do mundo físico disponibilizados via Web, são necessários modelos de alto nível que abstraíam os serviços

e dispositivos físicos subjacentes. Com isso, usuários e aplicações consumidores dos dados originados dos dispositivos conectados, podem possuir acesso aos dados de forma padronizada, por meio de interfaces de alto nível, não necessitando assim lidar com funcionalidades de baixo nível para a manipulação de tais objetos.

Ciência de Contexto

Ciência de contexto é outro requisito importante para plataformas de middleware IoT. Uma definição de contexto clássica e ainda bastante referenciada é a que Dey propôs em (DEY, 2001) em que contexto é “qualquer informação que caracteriza a situação de uma entidade, sendo que uma entidade pode ser uma pessoa, um lugar ou um objeto considerados relevantes para a interação entre um usuário e uma aplicação, incluindo o próprio usuário e a aplicação. O contexto é tipicamente a localização, a identidade e o estado das pessoas, grupos ou objetos físicos e computacionais”.

Portanto dados de contexto constituem informações consideradas importantes, obtidas do mundo físico ou lógico por meio de sensores, tornando o contexto um instrumento de apoio à comunicação entre sistemas e/ou aplicações, denominados cientes de contexto, e seus usuários. Plataformas de middleware em IoT devem então serem responsáveis pela coleta, representação e processamento das informações de contexto providas por múltiplas fontes, liberando as aplicações e usuários da tarefa de manipulá-las e tornando transparente tal manipulação.

Escalabilidade

O tipo de informação disponibilizada por um cenário IoT, pode ser provida apenas para um domínio específico, como também compartilhado com outros domínios de aplicação, gerando dados de maior valor agregado e podendo tomar uma proporção considerável, tanto em termos de números de dispositivos integrados, como de dados gerados. A Figura 3 caracteriza a dimensão deste fato.

Como consequência deste crescente número de dispositivos sendo conectados, as plataformas de middleware para IoT devem atender ao requisito de escalabilidade, i.e., possuírem capacidade para suportar requisições providas de inúmeros dispositivos, funcionando corretamente, mesmo em situações de uso intenso. Soluções de nuvem vem sendo utilizadas para este fim, devido à sua facilidade de provisão e uso de recursos computacionais, que podem ser alocados e liberados sob demanda, proporcionando o surgimento da chamada “Nuvem das Coisas” (em inglês, *Cloud of Things* - CoT) (SOLDATOS; SERRANO; HAUSWIRTH, 2012).

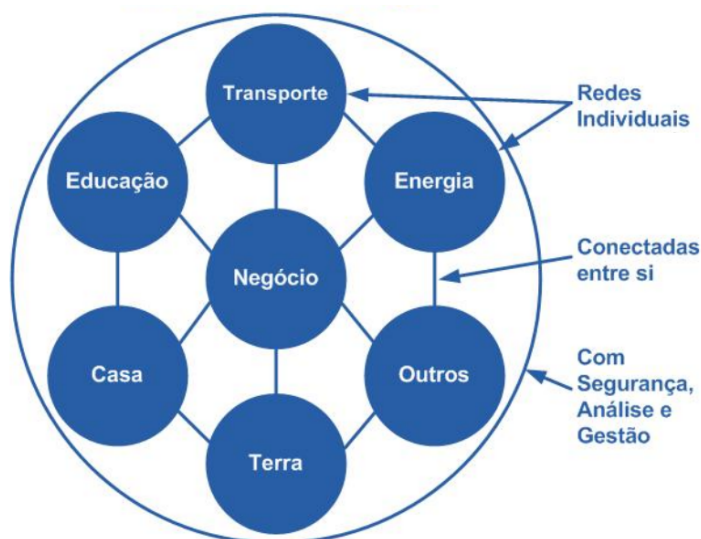


Figura 3: Dimensão de um Cenário IoT. Fonte: (EVANS, 2011)

2.2 Computação Ciente do Contexto

O modelo de computação, conhecido como computação Ciente do Contexto, refere-se à capacidade de um sistema computacional em perceber características do meio que sejam do seu interesse, e quando for o caso reagir ao mesmo. Este modelo vem ganhando destaque no cenário da IoT devido a necessidade dos inúmeros dispositivos interconectados tomarem decisões de forma autônoma, a partir de especificações dos usuários.

Dentre os desafios inerentes a cenários IoT para aplicações cientes do contexto destacam-se: (i) a aquisição do contexto a partir de um grande número de dispositivos heterogêneos e distribuídos; (ii) o processamento das informações de contexto adquiridas e a respectiva atuação sobre o meio físico; e (iii) a disponibilização dos dados contextuais processados aos usuários de forma distribuída.

Um sistema ciente de contexto é usualmente organizado em três componentes: aquisição, modelagem e processamento. Uma descrição resumida destes componentes é apresentada a seguir.

2.2.1 Aquisição de Informações Contextuais

A aquisição das informações contextuais é feita através de um conjunto de sensores heterogêneos. Porém estes sensores não se limitam aos dispositivos que fazem a aquisição de variáveis físicas do ambiente, mas sim a qualquer fonte de dados capaz de fornecer informações de contexto utilizáveis. Segundo (BALDAUF; DUSTDAR; ROSENBERG, 2007), em relação à forma como os dados são capturados, os sensores podem ser classificados em três grupos.

- Sensores físicos: este é o tipo de sensor mais utilizado. Trata-se de dispositivos

capazes de captar grandezas físicas diretamente do ambiente. Como exemplos típicos desse tipo de sensor têm-se: sensor de luz, movimento, localização, temperatura, umidade, entre outros.

- Sensores virtuais: neste tipo de sensor as informações de contexto são provenientes de softwares ou serviços. Um exemplo seria a identificação das atividades do usuário através de um software de monitoramento dos movimentos do mouse ou das entradas do teclado. Outro exemplo seria o acesso a informações climáticas oriundas de serviços web disponibilizadas por estações meteorológicas através da Internet.
- Sensores lógicos: este tipo de sensor disponibiliza informações contextuais geradas através da combinação de dados provenientes de diversas fontes, como sensores físicos, virtuais ou informações armazenadas em bancos de dados. Por exemplo, um sensor lógico poderia fornecer a localização de um usuário utilizando a posição do GPS do seu *smartphone* combinado com informações de login em suas contas de e-mail armazenadas em banco de dados.

Em ambientes IoT a quantidade de dispositivos pode atingir números elevados (KRUMM et al., 2010). Portanto, o desenvolvimento de sistemas computacionais para este tipo de cenário deve adotar estratégias que promovam a escalabilidade. Por isso o momento correto da coleta das informações contextuais deve considerar a natureza do elemento monitorado para que não se façam requisições desnecessárias. Para apoiar essa decisão, algumas técnicas de coleta são apontadas por (PERERA et al., 2013), as quais são classificadas tanto quanto a responsabilidade, quanto a frequência.

Quanto à responsabilidade, o processo de coleta das informações de contexto pode ocorrer de duas formas.

- *Pull*: nessa situação, o procedimento de coleta da informação contextual é disparado a partir do middleware, o qual faz uma consulta diretamente ao sensor. Nesta estratégia de coleta, o software que gerencia o comportamento do sensor não precisa realizar computações elaboradas, pois não cabe ao mesmo avaliar o momento oportuno de realizar a coleta.
- *Push*: nesse caso o procedimento de coleta é disparado a partir do sensor, o qual tem a iniciativa de enviar os dados capturados para o middleware através de um processo de publicação. Esta estratégia proporciona um processo de aquisição de maneira reativa, podendo ser disparada através de eventos do ambiente.

A coleta das informações contextuais também pode ser classificada em dois tipos, quanto à frequência de aquisição.

- Instantânea: esse tipo de frequência de coleta é disparada através de eventos do ambiente e acontece instantaneamente. Como exemplos de eventos que podem disparar procedimentos de coleta de informações de contexto, tem-se: a abertura de uma porta, o acender de uma luz, uma temperatura atingindo um determinado valor, entre outros. Essa frequência de coleta associada ao método *push* promove uma reação rápida e mostra-se conveniente em eventos críticos onde se necessita uma tomada de decisão rápida.
- Periódica: esse tipo de frequência de coleta obedece a períodos temporais especificados de acordo com as necessidades da aplicação do usuário. O período de tempo pode estar associado a um horário e dia específico, por exemplo, uma informação contextual sendo coletada de segunda a sexta às 19:00hs. A coleta também pode obedecer a intervalos periódicos de tempo, por exemplo, uma informação contextual sendo coletada a cada 30 segundos. As duas diferentes frequências de coleta (instantânea e periódica) podem ser implantadas tanto através do método *pull* quanto do método *push*.

2.2.2 Modelagem do Contexto

A quantidade de informação capturada e acessada num modelo de representação do contexto é significativa, com isso, os estudos de técnicas para representação de informações contextuais vem se tornando cada vez mais importantes.

O processo de modelagem de contexto consiste na concepção de um modelo de entidades do mundo real, suas propriedades, estado de seu ambiente e situações que podem ser usados como referência para a aquisição, interpretação e raciocínio de informações contextuais (KNAPPMAYER et al., 2013). A modelagem das informações de contexto reduz a complexidade das aplicações cientes do contexto, facilita o acesso às informações realizando buscas de forma eficiente, e melhora a capacidade de manutenção e de evolução da aplicação.

Existem diversos modelos para representação do contexto (PERERA et al., 2013), tais como: chave-valor, linguagem de marcação, gráfico, orientado a objetos, lógico, ontológico. Modelos híbridos de modelagem de contexto são considerados os mais promissores, pois combinam diferentes técnicas de modelagem, com diferentes níveis de interpretação, para diferentes aspectos.

2.2.3 Processamento do Contexto

O processamento do contexto engloba aspectos relacionados à interpretação, agregação, armazenamento, consulta e inferência das informações contextuais obtidas a partir de uma etapa de aquisição de contexto.

O mesmo visa possibilitar a compreensão dos contextos de interesse através da geração de informações mais significativas. De modo mais específico, a interpretação

de contexto consiste em um conjunto de métodos e processos que realizam a abstração, o mapeamento, a manipulação, a agregação, a derivação, a inferência e demais ações sobre as informações contextuais, com o propósito de facilitar o entendimento de um determinado contexto pelas aplicações e auxiliá-las na tomada de decisão. O processo de interpretação de contexto consiste na manipulação e refinamento das informações contextuais de um ambiente.

Assim as atividades de processamento do contexto devem ser abstraídas das aplicações e um módulo interpretador torna-se, portanto, um componente essencial em uma plataforma de suporte a tais aplicações. Ele deve ser capaz de obter e prover informação contextual em diferentes níveis de abstração, conforme o desejo do usuário e de suas aplicações. Uma aplicação pode necessitar tanto de informações brutas, de mais baixo nível como de informações mais abstratas e elaboradas, de mais alto nível, provenientes de um processo de refinamento e interpretação (COSTA; YAMIN; GEYER, 2008).

Um dos problemas na utilização de informações contextuais consiste na obtenção de contextos significativos, a partir de um conjunto de informações dispersas e desconexas, obtidas por mecanismos heterogêneos de aquisição. Para isso, devem ser disponibilizadas funcionalidades de processamento e raciocínio sobre a informação contextual.

Os termos raciocínio e inferência são geralmente utilizados para indicar processos pelo quais deduções a respeito dos dados contextuais são alcançadas. O projeto e implementação de um mecanismo para raciocínio de contextos pode variar bastante a depender do tipo do conhecimento contextual envolvido. A priori, o processamento do contexto deve ser implementado separadamente do comportamento da aplicação do usuário e não embutido no código da mesma (KNAPPMEYER et al., 2013).

O raciocínio é utilizado para verificar a consistência do contexto e para inferir contexto implícito (alto nível), a partir de contextos explícitos (baixo nível) (Xiao Hang Wang et al., 2004). A consistência é necessária, pois, muitas vezes, a informação contextual adquirida automaticamente pode apresentar ambiguidades.

A necessidade de manter o histórico de dados de contexto é um requisito ligado à aquisição de informações contextuais, bem como à disponibilidade contínua dos componentes de captura destas informações. Um histórico do contexto pode ser utilizado para estabelecer tendências e prever valores futuros de informações contextuais. Sem o armazenamento dessas informações, análises desse tipo não são possíveis de realizar.

2.3 Protocolos de Comunicação na IoT

Os protocolos de comunicação são parte importante na implantação de infraestruturas IoT, podendo contribuir para uma melhor utilização de recursos como os de processamento, memória, energia e banda de comunicação dos dispositivos inteligentes, tendo em vista fatores inerentes do universo IoT como heterogeneidade, mobilidade, intermitência de comunicação, extensibilidade, escalabilidade e capacidade de autoconfiguração.

Nesta dissertação de mestrado foram considerados os seguintes protocolos de comunicação: UPnP, CoAP e MQTT. A seleção por estes protocolos foi devido a revisão de literatura mostrar como sendo os mais oportunos e alinhados com os objetivos do trabalho.

O UPnP destaca-se por prover funcionalidades que minimizam os esforços de gerência dos dispositivos ou objetos inteligentes conectados, como a ativação, desativação e descoberta destes dispositivos e seus recursos por parte das aplicações. Já o CoAP e MQTT são protocolos que objetivam o baixo consumo computacional e energético dos dispositivos introduzidos pelo cenário da IoT, que se caracterizam por possuírem capacidade restrita quanto a esses quesitos.

2.3.1 UPnP

As características de autoconfiguração e de descoberta automática de dispositivos inerentes ao UPnP (*Universal Plug and Play*), associado ao fato deste utilizar protocolos e padrões consolidados da Internet, tais como HTTP, SOAP e XML constituem um facilitador na gerência do crescente número de dispositivos conectados, o que traduz o fato deste ser bastante utilizado no cenário da IoT. Outro fator interessante é a quantidade significativa de dispositivos comerciais com a certificação DLNA (*Digital Living Network Alliance*) (DLNA, 2015) que tem como base o protocolo UPnP para comunicação.

O UPnP é uma arquitetura proposta inicialmente pela Microsoft que está sendo mantida atualmente por um fórum do qual participam centenas de fabricantes (UPNP, 2015). Este permite a interoperabilidade entre diferentes dispositivos computacionais, em uma rede baseada no modelo *Plug and Play*, que facilita a instalação e configuração de dispositivos heterogêneos numa rede IP, sem a necessidade de configurá-los para possibilitar os processos de obtenção de endereços IP e de descoberta e invocação de serviços de rede.

Sua arquitetura de software oferece uma conexão de rede denominada *Peer-to-Peer* (P2P), onde cada um dos pontos ou nós da rede funciona tanto como cliente quanto como servidor.

Estão definidos no padrão UPnP três componentes básicos.

- **Dispositivo UPnP:** contêm serviços UPnP e podem conter outros dispositivos UPnP aninhados. Por exemplo, uma impressora (dispositivo UPnP) pode consistir em um serviço de impressão e um dispositivo scanner aninhado, que por sua vez oferece um serviço de fotocópia.
- **Serviço UPnP:** expõe ações que podem ser aplicadas ao mesmo durante sua invocação por meio de um servidor de controle residente no dispositivo que hospeda o serviço, bem como uma tabela que armazena um conjunto de variáveis de estado do serviço (tabela de estados). A tabela de estados do serviço UPnP pode ser monitorada por um servidor de eventos (também residente no dispositivo que hospeda o serviço), que tem como função publicar a outras entidades interessadas a modificação de variáveis de estado desse serviço.
- **Ponto de Controle UPnP:** atua, em parte, como servidor de diretório, tendo como tarefa descobrir e controlar os dispositivos UPnP presentes na rede.

No caso de um dispositivo não possuir capacidade de portar o padrão UPnP, um gateway UPnP deve ser usado para mapear os protocolos definidos pelo UPnP nos protocolos entendidos nativamente pelo dispositivo.

O UPnP permite o controle de dispositivos e o processamento de serviços de forma descentralizada, e permite a interoperabilidade entre pontos de controle. Além disso, a tecnologia UPnP inclui o conceito de descoberta e de descrição de dispositivos e serviços, permitindo assim que um dispositivo possa ser dinamicamente encontrado e totalmente compreendido, em termos de funcionalidade, por descrições XML.

2.3.1.1 *Pilha de Protocolos UPnP*

A pilha de protocolos UPnP é constituída de protocolos específicos localizados nas camadas de mais alto nível (três camadas superiores), e por protocolos padrões da Internet (vide Figura 4).

Protocolos Específicos UPnP

Os fabricantes de dispositivos UPnP, os comitês de trabalho do fórum UPnP e o documento da arquitetura de dispositivo UPnP definem os protocolos das camadas superiores utilizados para implementar o UPnP.

A arquitetura de dispositivo UPnP, do inglês *UPnP Device Architecture* (UDA) define um esquema ou modelo para a criação de descrições de dispositivo e serviço. Baseado nesta arquitetura os comitês de trabalho definem especificações para tipos de dispositivo como videocassetes (VCRs), sistemas HVAC, lavadoras de pratos e outros aparelhos. Subsequentemente, os fabricantes de dispositivos UPnP adicionam dados específicos aos seus dispositivos como o nome do dispositivo, número do modelo, nome do fabricante e URL para a descrição do serviço.

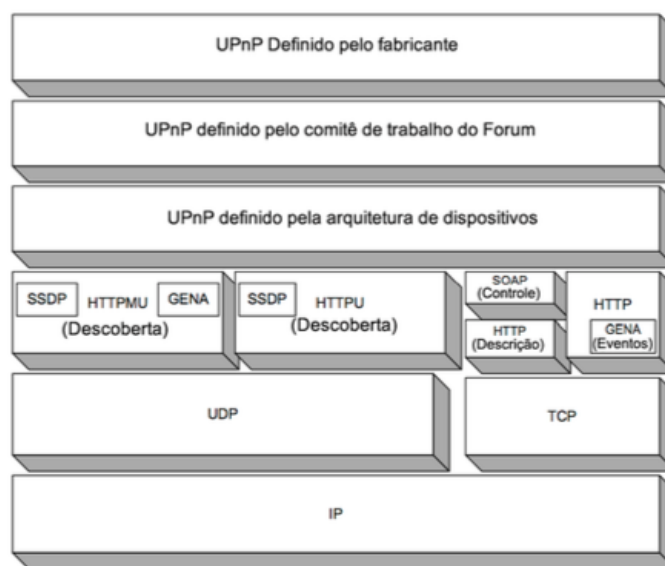


Figura 4: Pilha de Protocolos UPnP. Fonte: (FERREIRA, 2014)

TCP/IP

A pilha de protocolos de rede TCP/IP serve como base para a construção dos demais protocolos UPnP. Dispositivos UPnP podem usar diversos protocolos na pilha TCP/IP, incluindo TCP, UDP, IGMP, ARP e IP, além de serviços TCP/IP como DHCP e DNS.

HTTP, HTTPU, HTTPMU

O HTTP, que é o principal responsável pelo sucesso da Internet, é também parte central do UPnP. Todos os aspectos do UPnP são construídos sobre o HTTP ou suas variantes.

O HTTPU e o HTTPMU são variantes do HTTP definidas para entregar mensagens sobre UDP/IP em vez de TCP/IP. Esses protocolos são utilizados pelo SSDP (*Simple Service Discovery Protocol*). Os formatos básicos de mensagem usados por esses protocolos correspondem aos do HTTP, sendo que o HTTPMU é utilizado para difusão seletiva (multicast) e o HTTPU para entrega ponto a ponto (unicast).

SSDP - Simple Service Discovery Protocol

O Protocolo de Descoberta de Serviço Simples (SSDP) define como os serviços de rede podem ser descobertos na rede. Baseia-se em HTTPU e HTTPMU e define métodos para que um ponto de controle localize recursos de interesse na rede, e para que dispositivos anunciem sua disponibilidade na rede, diminuindo a sobrecarga na comunicação e agilizando o trabalho decorrente da existência de mais de um ponto de controle na rede.

Além da capacidade de descoberta, o protocolo SSDP possui a capacidade de

Notificação de Despedida (desconexão da rede), liberando os dispositivos e serviços indesejados ou ociosos.

GENA - Generic Event Notification Architecture

A Arquitetura de Notificação de Evento Genérico (GENA) foi definida para fornecer a capacidade de envio e recebimento de notificações utilizando HTTP sobre TCP/IP e HTTPMU sobre UDP/IP.

SOAP - Simple Object Access Protocol

O Protocolo de Acesso a Objeto Simples (SOAP) define o uso da Linguagem de Marcação Extensível (XML) e do HTTP para executar chamadas de procedimento remoto.

O UPnP utiliza SOAP para entregar mensagens de controle aos dispositivos e retornar resultados ou erros aos pontos de controle. Cada solicitação de controle de UPnP é uma mensagem SOAP que contém a ação da solicitação juntamente com um conjunto de parâmetros, já a resposta é uma mensagem SOAP contendo o status, o valor de retorno e demais parâmetros de retorno.

2.3.1.2 Etapas envolvidas na rede UPnP

Para que pontos de controle e dispositivos UPnP interajam entre si, a arquitetura de dispositivos UPnP (UDA) define as seguintes etapas:

1. **Endereçamento:** determina um endereço IP para uma entidade UPnP através de DHCP ou auto IP.
2. **Descoberta:** anuncia o dispositivo no UPnP, através do protocolo SSDP, podendo este ser localizado pelos pontos de controle que já estão em execução. A descoberta ocorre assim que os dispositivos forem conectados à rede e endereçados adequadamente. Quando um dispositivo é adicionado à rede, o SSDP permite que ele anuncie seus serviços aos pontos de controle na rede. Quando um ponto de controle é adicionado à rede, o SSDP permite que ele procure dispositivos de interesse na rede. A troca básica em ambos os casos é uma mensagem de descoberta contendo algumas especificações essenciais sobre o dispositivo ou um de seus serviços, por exemplo, seu tipo, identificador e ponteiro para o respectivo documento de descrição de dispositivo XML.
3. **Descrição:** os pontos de controle interessados podem solicitar arquivos XML contendo informações do dispositivo (Documento de Descrição do Dispositivo) e de seus serviços (Documentos de Descrição de Serviços), através de uma requisição HTTP. Após a descoberta, o ponto de controle ainda sabe muito pouco

sobre o dispositivo. Para obter mais informações sobre o dispositivo e seus recursos, ou para interagir com ele, o ponto de controle deve recuperar a sua descrição na URL fornecida pelo dispositivo na mensagem de descoberta. A descrição do UPnP em XML para um dispositivo inclui informações de fabricante específicas do fornecedor incluindo o nome e número do modelo, número de série, nome do fabricante, URLs de sites específicos do fornecedor e assim por diante. A descrição também inclui uma lista de todos os dispositivos ou serviços incorporados, assim como URLs para controle, eventos e apresentação.

4. **Controle:** depois de recuperar uma descrição do dispositivo, o ponto de controle terá as informações básicas para o seu controle. Para saber mais sobre o serviço, o ponto de controle deve recuperar uma descrição detalhada do UPnP para cada serviço. A descrição para o serviço também é expressa em XML e inclui uma lista de comandos, ou ações, aos quais o serviço responde, além de parâmetros ou argumentos para cada ação. A descrição de um serviço inclui também uma lista de variáveis que modelam o estado do serviço na execução e são descritas em termos do tipo de dados, do intervalo e das características do evento. Para controlar um dispositivo, um ponto de controle envia uma solicitação de ação a um serviço de dispositivo. Para isso este envia uma mensagem XML de controle, através do protocolo SOAP, adequada a URL de controle do serviço (fornecido na descrição do dispositivo). Em resposta à mensagem de controle, o serviço retorna valores ou códigos de falha específicos da ação.
5. **Evento:** paralelo ao controle, a fase de Evento permite que pontos de controle se cadastrem nos dispositivos para receberem notificações de eventos, que ocorrem quando as variáveis de estado do dispositivo são modificadas. Uma descrição UPnP para o serviço inclui uma lista de ações às quais o serviço responde e uma lista de variáveis que modelam o estado do serviço na execução. O serviço publica atualizações quando essas variáveis mudam, e um ponto de controle pode inscrever-se para receber essas informações. Estas atualizações são enviadas através de mensagens de evento expressas em XML e formatadas usando o protocolo GENA, que contém os nomes de uma ou mais variáveis de estado e o valor atual das variáveis.
6. **Apresentação:** se um dispositivo possuir uma URL de apresentação, então o ponto de controle pode recuperar uma página dessa URL, carregá-la em um navegador e, dependendo dos recursos da página, permitir que um usuário controle o dispositivo e/ou exiba seu status. O grau de realização dessas etapas depende dos recursos específicos da página de apresentação e do dispositivo.

2.3.2 CoAP

O CoAP (*Constrained Application Protocol*) é um protocolo que recentemente tornou-se um *Request for Comments* (RFC - 7252) (SHELBY; HARTKE; BORMANN, 2015), desenvolvido pelo grupo de trabalho CoRE (*Constrained RESTful Environment*) do IETF (*Internet Engineering Task Force*), com o objetivo de se ter um protocolo web genérico, alternativo ao HTTP, adequado aos requisitos especiais do ambiente introduzido pela IoT e caracterizado por dispositivos de baixa potência, considerando principalmente, a questão do consumo de energia.

O protocolo CoAP baseia-se em princípios arquiteturais REST (FIELDING, 2000). Um dos pontos chave deste estilo arquitetural tem a ver com a utilização de URIs (*Uniform Resource Identifier*) para identificação de recursos na web. Outra característica do estilo é que os serviços são abstraídos através de uma interface uniforme (HTTP e seus respectivos métodos) que fornece mecanismos para que as aplicações cliente possam escolher a melhor representação possível das respostas que recebem do serviço.

O diferencial do CoAP é que este permite a execução das funções HTTP em redes com recursos limitados, reduzindo o *overhead* original do HTTP ao comprimir o cabeçalho da camada de aplicação para apenas 4 bytes (excluindo os campos opcionais, que ocupam 4 bits cada um).

2.3.2.1 Características do CoAP

- UDP nativo: utiliza o UDP (*User Datagram Protocol*) na camada de transporte em vez de TCP (*Transmission Control Protocol*). O UDP é um protocolo não orientado a conexão, que permite uma maior rapidez de *wake up* e transmissão de ciclos, bem como de pacotes com menos sobrecarga. Isso permite que dispositivos possam permanecer em estado de repouso por longo período de tempo conservando a energia da bateria.
- Suporte a *Multicast*: a rede é inerentemente *unicast* porém possui suporte *multicast*. Isso é devido ao CoAP ser construído em cima de IPv6, o qual permite o endereçamento *multicast* para os dispositivos, além de seus endereços IPv6 normais.
- Segurança: por utilizar o protocolo UDP, não é possibilitado o uso por parte do CoAP da tecnologia SSL/TLS (*Secure Sockets Layer/Transport Layer Security*), no entanto este utiliza o DTLS (*Datagram Transport Layer Security*), que permite a mesma segurança do TLS. Como TCP, UDP é criptografado, mas a segurança pode ser - e deve ser - ampliada com DTLS.
- Descoberta de Recurso/Serviço: os servidores CoAP fornecem uma listagem

completa dos seus recursos e isto permite aos clientes saberem quais os recursos podem utilizar, bem como o seu tipo. Todos os recursos são descritos através de URIs, um conjunto de atributos e, se necessário, relações com outros recursos.

- **Comunicação Assíncrona:** a maioria das mensagens são enviadas e recebidas usando o modelo *request/response*. No entanto, o CoAP tem um mecanismo simplificado chamado “*observe*” semelhante ao *publish/subscribe*, no qual o cliente indica que pretende receber as atualizações do recurso sempre que o seu estado for alterado.

2.3.2.2 **Mensagem CoAP**

CoAP é baseado na troca de mensagens compactas com características que influem o aumento na confiabilidade da sua transmissão. As mensagens CoAP podem ser de quatro tipos, descritos a seguir (SHELBY; HARTKE; BORMANN, 2015).

- **Confirmable (CON):** este tipo de mensagem oferece confiabilidade sobre o protocolo UDP. Sempre que é enviada uma mensagem deste tipo ao servidor, ela segue com um *time out* associado. Quando o servidor recebe a mensagem, envia um ACK com o mesmo identificador da mensagem do pedido, confirmando a recepção do mesmo. Quando o cliente não recebe um ACK passado o tempo definido no *time out*, este retransmite a mensagem e o *time out* enviado nesta retransmissão é duplicado para garantir que a entrega do pacote seja efetuada. Estas retransmissões são efetuadas até ser recebido um ACK com o mesmo identificador de mensagem por parte do servidor.
- **Non-confirmable (NON):** não necessita de confirmação de recebimento. O cliente não tem como saber se o pedido chegou ao servidor. Como alternativa, o cliente pode enviar múltiplos pedidos. Esta característica é útil no caso de uma aplicação que recebe leituras constantes de um sensor de temperatura em um espaço muito curto de tempo, onde a perda de uma ou outra mensagem não é motivo para preocupações.
- **Acknowledgment (ACK):** são mensagens que confirmam o recebimento de uma mensagem *Confirmable*. Caso o servidor necessite enviar alguma informação no *Payload* da mensagem para o cliente, esta pode ser enviada juntamente com o ACK. Se o ACK e a informação da resposta forem enviados em mensagens separadas, o ID destas mensagens será diferente, visto que, para cada retransmissão é necessário um novo identificador. Quando o cliente receber a informação, tem que enviar um ACK, indicando que recebeu a resposta. Neste caso, o pedido e a resposta são identificados devido à utilização da opção *Token*.

- *Reset* (RST): indica que outra mensagem (CON ou NON) foi recebida, mas por algum imprevisto não pôde ser devidamente processada. Ela pode ocorrer no caso de algum dispositivo ter reiniciado e a mensagem enviada não foi interrompida.

Os tipos descritos podem ser usados em mensagens de Requisição (*Request*), Resposta (*Response*) e Vazia (*Empty*). A mensagem *Empty* é caracterizada pelo código 0.00 e não apresenta nenhuma informação adicional, apenas o cabeçalho de 4 bytes. A relação de utilização do tipo de mensagem por parte destas mensagens pode ser observada na Tabela 1.

Os caracteres descritos na tabela possuem o seguinte significado: “X” representa que o tipo de mensagem pode ser utilizado e “-” que este tipo de mensagem não pode ser considerado. O caractere “*” significa que a combinação não é utilizada em condições normais de operação, mas apenas para provocar uma mensagem de *Reset*, constituindo-se assim uma forma de verificar se o dispositivo CoAP está ativo (*Ping CoAP*).

Tabela 1: Utilização dos Tipos de Mensagens CoAP

	CON	NON	ACK	RST
Request	X	X	-	-
Response	X	X	X	-
Empty	*	-	X	X

Fonte: (SHELBY; HARTKE; BORMANN, 2015)

Estas mensagens são transmitidas via IEEE 802.15.4 e apresentam-se codificadas em um formato binário com um cabeçalho de 4 bytes, seguido por um *Token* de largura variável (de 0 a 8 bytes). Após o Token pode não haver nenhuma ou uma série de opções do CoAP no formato *Type-Length-Value* (TLV), que são opcionalmente seguidas por um *Payload*, ocupando o resto do datagrama. A Figura 5 mostra o formato da mensagem CoAP.

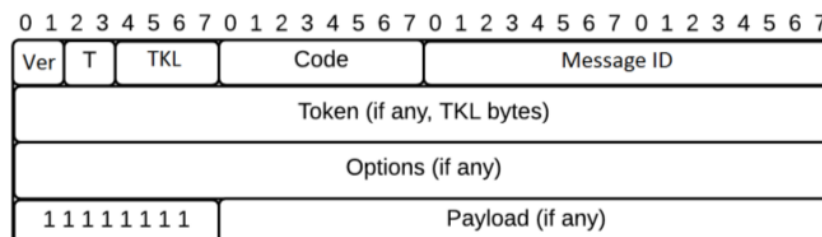


Figura 5: Formato da Mensagem CoAP. Fonte: (LIMA, 2014)

O *Token* é utilizado para relacionar requisições e respostas. Ele é gerado no cliente e transportado junto com a requisição. O servidor deve então ecoar este Token na resposta correspondente.

Já o *Payload* é um campo opcional. Quando a mensagem possui Payload ele deve ser indicado por um marcador de um byte (*Payload Marker*) que indica o fim das opções e o início da carga. A ausência do marcador significa um Payload de largura 0 bytes, enquanto a presença de um marcador seguido por um Payload de 0 bytes deve ser processado como um erro no formato da mensagem.

Os campos que formam o cabeçalho do datagrama são definidos pelo RFC 7252 - IETF (SHELBY; HARTKE; BORMANN, 2015) da seguinte maneira.

- Versão (Ver): inteiro não assinado de dois bits que indica o número correspondente à versão do CoAP.
- Tipo (T): inteiro não assinado de dois bits que indica o tipo da mensagem, que pode variar entre *Confirmable* (0), *Non-confirmable* (1), *Acknowledgment* (2) ou *Reset* (3).
- Largura do *Token* (TKL): inteiro não assinado de quatro bits usado para indicar a largura variável do Token (até 8 bytes).
- Código: inteiro não assinado de oito bits que caracteriza o tipo da mensagem, podendo indicar um método de requisição (vide Tabela 2) ou um código de resposta (vide Tabela 3).
- ID da mensagem: inteiro não assinado de 16 bits usado para detectar duplicação de mensagens e também para comparação de mensagens do tipo ACK, por exemplo.

Tabela 2: Códigos dos Métodos de Requisição CoAP

Código	Nome
0.01	GET
0.02	POST
0.03	PUT
0.04	DELETE

Fonte: (SHELBY; HARTKE; BORMANN, 2015)

2.3.3 MQTT

O MQTT (*Message Queuing Telemetry Transport*) foi desenvolvido por Andy Stanford-Clark, da IBM, e Arlen Nipper da Arcom (agora Eurotech), em 1999 (MQTT, 2015). Trata-se de um protocolo de mensagens baseado na arquitetura *publish/subscribe* voltado a atuar principalmente, mas não exclusivamente, em ambientes onde há redes instáveis com baixa largura de banda, bem como dispositivos embarcados com recursos limitados de memória e processamento (MARTINS; ZEM, 2015).

Tabela 3: Códigos de Resposta CoAP

Código	Nome
2.01	Criado
2.02	Deletado
2.03	Válido
2.04	Alterado
2.05	Conteúdo
4.00	<i>Bad Request</i>
4.01	Não autorizado
4.02	<i>Bad Option</i>
4.03	Proibido
4.04	Não encontrado
4.05	Método não permitido
4.06	Inaceitável
4.12	Pré-condição falhou
4.13	Entidade de requisição muito grande
4.15	<i>Content-format</i> não suportado
5.00	Erro interno no servidor
5.01	Não implementado
5.02	<i>Bad gateway</i>
5.03	Serviço indisponível
5.04	<i>Gateway time out</i>
5.05	<i>Proxying</i> não suportado

Fonte: (SHELBY; HARTKE; BORMANN, 2015)

Em seu desenvolvimento foram considerados alguns princípios, como o de minimizar requerimentos de recursos do dispositivo, bem como o de largura de banda tentando assegurar confiabilidade e garantia de entrega, o que justifica a sua ampla utilização no cenário da IoT. O MQTT tornou-se um padrão aberto OASIS (*Organization for the Advancement of Structured Information Standards*) em 07 de novembro de 2014, em sua versão 3.1.1 (OASIS-MQTT, 2015).

O protocolo segue o modelo cliente/servidor. Os dispositivos sensores são clientes que se conectam a um servidor (chamado de *broker*) usando TCP. As mensagens a serem transmitidas são publicadas para um endereço (chamado de tópico), que inclusive, assemelha-se a uma estrutura de diretórios em um sistema de arquivos, por exemplo, casa/quarto2/temperatura. Clientes por sua vez podem se inscrever para vários tópicos, tornando-se assim capazes de receber as mensagens que outros clientes publicam neste tópico.

2.3.3.1 Características do MQTT

- Arquitetura *Publish/Subscribe*: *brokers* e clientes publicam informações, como também podem se inscrever para receber informações de outros, de acordo com o conteúdo da mensagem, tipo ou assunto.

- Desacoplamento de Espaço: o cliente não necessita ter conhecimento do endereço IP de outros clientes para publicar ou receber informações destes, basta conhecer o IP do *broker*. O *broker* então fica responsável por intermediar a comunicação, gerenciando o fluxo de informações aos clientes.
- Desacoplamento de Tempo: o cliente pode publicar a sua informação, independentemente do estado que encontra-se o cliente alvo. As publicações são transmitidas ao *broker* que repassa ao cliente correspondente quando este estiver ativo. Isso permite que clientes permaneçam em estado de repouso, mesmo quando outros clientes estão publicando mensagens relevantes para eles.
- Segurança: *brokers* MQTT podem exigir nome de usuário e senha para autenticação de clientes, possibilitando a estes um controle de nível de acesso as informações pelos clientes. A privacidade é garantida pela transmissão dos dados criptografada via TCP, mediante o uso de SSL/TLS.
- Níveis de *Quality of Services* (QoS): possui três níveis de QoS (0, 1 e 2) que representam a garantia de qualidade de entrega das mensagens (vide Tabela 5).
- Aviso de Desconexão: possui mecanismo que notifica as partes interessadas quando um cliente se desconecta da rede anormalmente.
- Tópicos de Subscrição Flexíveis: um cliente pode se inscrever a todas as mensagens dentro de uma determinada funcionalidade. Por exemplo em uma cozinha o dispositivo cliente forno pode receber todas as informações referentes ao tópico cozinha/forno/+, com o código “+” como coringa. Isto permite uma quantidade mínima de código (ou seja, memória e custo).
- MQTT-SN: possui uma variante do protocolo MQTT para *Sensors Networks* (Redes de Sensores). Define um mapeamento UDP de MQTT, voltado para redes não TCP/IP como *Zigbee*, além de adicionar suporte ao *broker* para indexação de nomes de tópicos. Isto é devido a necessidade da sequência destes nomes não poder ser longa em redes de sensores sem fio com baixa taxa de transmissão.

2.3.3.2 Mensagem MQTT

Cada uma das chamadas mensagens de comando MQTT possui um cabeçalho fixo composto de dois bytes, onde o primeiro byte contém o campo que identifica o tipo da mensagem e também campos marcadores (DUP, Nível QoS e *RETAIN*). A Figura 6 mostra o formato do cabeçalho fixo das mensagens.

O tipo da mensagem ocupa os 4 bits mais significativos no primeiro byte do cabeçalho fixo e é representado por um valor não assinado. A lista de tipos definida na versão 3.1.1 do MQTT (BANKS; GUPTA, 2014) é descrita na Tabela 4.

bit	7	6	5	4	3	2	1	0
Byte 1	Tipo da Mensagem				Flag DUP	Nível QoS		RETAIN
Byte 2	Largura restante							

Figura 6: Cabeçalho Fixo de uma Mensagem MQTT. Fonte: (MARTINS; ZEM, 2015)

Os quatro bits restantes do primeiro byte dividem-se em quatro campos que servem de marcadores para indicar preferências definidas antes do envio da mensagem. São eles:

- *Duplicate delivery* (DUP): acrônimo relativo à entrega duplicada, este marcador ocupa o bit 3 e é ativado quando o cliente ou o servidor tentam reenviar mensagens do tipo *PUBLISH*, *PUBREL*, *SUBSCRIBE* ou *UNSUBSCRIBE* (vide Tabela 4) que tenham *Quality of Service* (QoS) maior que 0 e requeiram *Acknowledgment* (ACK);
- *Quality of Service* (QoS): este marcador ocupa os dois penúltimos bits menos significativos e indica o nível de garantia da entrega de uma mensagem *PUBLISH*. Os níveis de QoS são mostrados na Tabela 5;
- *RETAIN*: quando um cliente envia uma mensagem *PUBLISH* ao servidor com este marcador ativo, ela deve ser retida no servidor mesmo depois de ser entregue aos assinantes. No evento de uma nova subscrição a um tópico, a última mensagem retida para este tópico deve ser enviada para o novo assinante caso este marcador esteja ativado.

O segundo byte do cabeçalho fixo é usado para representar a quantidade de bytes remanescentes na mensagem. Incluindo dados do cabeçalho variável e do *payload*.

Cabeçalho variável é um componente presente em alguns tipos de mensagem MQTT e está localizado entre o cabeçalho fixo e o *payload*. Usado principalmente nas mensagens *CONNECT*, este cabeçalho possui dois campos para nome e versão do protocolo e mais uma série de marcadores que definirão algumas diretivas para a conexão entre cliente e servidor.

O *payload* pode armazenar diferentes tipos de informação, tudo vai depender do tipo da mensagem transmitida: *CONNECT* (irá conter ID do cliente), *SUBSCRIBE* (contém tópicos que o cliente pode subscrever e/ou nível QoS) ou *SUBACK* (lista de níveis de QoS garantidos pelo servidor).

Tabela 4: Tipos de Mensagem de Controle

Mnemônico	Enumeração	Descrição
Reservado	0	Reservado
CONNECT	1	Requisição de conexão do cliente ao servidor
CONNACK	2	ACK de conexão
PUBLISH	3	Mensagem de Publicação
PUBACK	4	ACK de publicação
PUBREC	5	Publicação recebida (garantia de entrega parte I)
PUBREL	6	Publicação liberada (garantia de entrega parte II)
PUBCOMP	7	Publicação completa (garantia de entrega parte III)
SUBSCRIBE	8	Requisição de subscrição do cliente
SUBACK	9	ACK de subscrição
UNSUBSCRIBE	10	Requisição de cancelamento de subscrição do cliente
UNSUBACK	11	ACK de cancelamento de subscrição
PINGREQ	12	Requisição PING
PINGRESP	13	Resposta PING
DISCONNECT	14	Cliente desconectando
Reservado	15	Reservado

Fonte: (OASIS-MQTT, 2015)

Tabela 5: Níveis de QoS

Valor QoS	Bit 2	Bit 1	Descrição		
0	0	0	Até uma vez	Disparar e esquecer	≤ 1
1	0	1	Ao menos uma vez	Entrega com ACK	≥ 1
2	1	0		Entrega garantida	1
3	1	1	Reservado		

Fonte: (OASIS-MQTT, 2015)

2.4 Considerações sobre o Capítulo

Este capítulo apresentou os esforços realizados na revisão de literatura trazendo aspectos importantes que contribuíram ao desenvolvimento desta dissertação de mestrado.

Foi abordado o surgimento da Internet das Coisas e suas aplicações usuais, bem como sua consolidação enquanto área de estudo, sendo caracterizados seus principais desafios de pesquisa. A utilização de middlewares, como uma solução para grande parte dos desafios IoT foi apontada. Também, os conceitos e importância da Computação Ciente de Contexto no cenário de IoT foram discutidos. Por fim, as características de três protocolos IoT que se adequam aos objetivos do trabalho, enquanto estratégia para identificação e comunicação dos dispositivos da IoT, foram apresentados.

3 TRABALHOS RELACIONADOS

Tendo como base as premissas que motivaram o desenvolvimento desta dissertação de mestrado, foram identificados alguns trabalhos relacionados no estudo da literatura na área. Os trabalhos foram selecionados tendo em vista aspectos que explorassem o tratamento dos seguintes desafios IoT: (i) interoperabilidade, (ii) descoberta e gerenciamento de dispositivos, (iii) interfaces de alto nível, (iv) ciência de contexto e (v) escalabilidade. Este capítulo além de conter a descrição dos trabalhos relacionados, ao final apresenta uma análise entre os mesmos.

3.1 EcoDiF - Ecossistema Web de Dispositivos Físicos

A plataforma EcoDiF (DELICATO et al., 2013) (PIRES et al., 2014) integra dispositivos físicos heterogêneos e os conecta à Internet, fornecendo funcionalidades de controle, visualização, processamento e armazenamento de dados em tempo real. Esta foi desenvolvida como uma proposta de solução a alguns dos desafios IoT, tais como: (i) a necessidade de uma camada de abstração sobre dispositivos e serviços a aplicações e usuários finais; (ii) o fornecimento de serviços de busca e descoberta desses elementos; (iii) a interconexão de objetos e serviços via rede; (iv) o monitoramento do estado e da localização dos objetos conectados, e; (v) o gerenciamento da interoperabilidade entre os objetos envolvidos.

Uma característica dessa plataforma refere-se ao fato dela se alinhar a um outro paradigma surgido a partir do termo Internet das Coisas, denominado Web das Coisas, do inglês *Web of Things* (WoT). Este paradigma se caracteriza pelo desenvolvimento de aplicações que utilizam protocolos e padrões amplamente aceitos e já em uso na Web tradicional, tais como HTTP (*Hypertext Transfer Protocol*) e URIs (*Uniform Resource Identifier*) (GUINARD, 2010).

No caso o HTTP não é utilizado apenas como um protocolo de comunicação, mas também como uma maneira de oferecer suporte a todas as interações com os dispositivos interconectados, eliminando barreiras no tocante à incompatibilidade entre diferentes fabricantes, protocolos proprietários e formatos de dados. Assim dispositi-

vos físicos são integrados ao ambiente digital, de maneira que seus dados e serviços possam ser acessados por diferentes aplicações como um recurso qualquer da Web.

A EcoDiF foi implementada utilizando a linguagem de programação Java e implantada em um servidor de aplicações JBoss, que permite o gerenciamento de componentes distribuídos, promove a confiabilidade de aplicações baseadas em um conjunto de serviços acessíveis via Web e em grandes volumes de dados, como é típico em ambientes de IoT. Os dados são recebidos por requisições HTTP PUT, de forma a prover uma interface RESTful para os clientes (sejam humanos ou aplicações).

A arquitetura de software da EcoDiF, ilustrada pela Figura 7, é composta por sete módulos descritos a seguir.

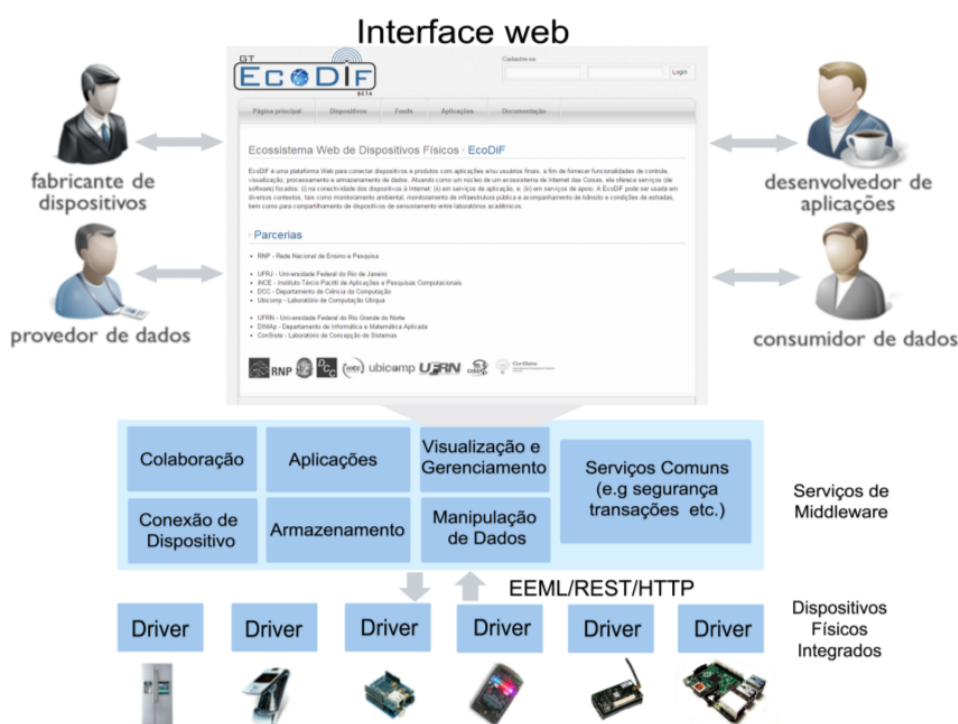


Figura 7: Arquitetura EcoDiF. Fonte: (PIRES et al., 2014)

- **Módulo de Colaboração:** visa facilitar a colaboração entre os usuários da EcoDiF, permitindo realizar buscas por dispositivos e aplicações a partir de seus respectivos metadados (tipo, usuário, localização, etc.), através da interface Web.
- **Módulo de Conexão de Dispositivos:** tem como objetivo viabilizar a conexão de dispositivos físicos à plataforma. A forma encontrada para a integração dos dispositivos foi por meio da utilização de *drivers* customizados, desenvolvidos para cada tipo de dispositivo. Dessa forma a heterogeneidade é abstraída de usuários e aplicações, podendo estes fazerem uso dos dados gerados por estes dispositivos. Os *drivers* são construídos com base em princípios REST e padrões e protocolo Web. Além disso os *drivers* também são responsáveis por

estruturar os dados utilizando a linguagem EEML, para após serem enviados à EcoDiF através de requisições HTTP PUT.

- **Módulo de Aplicações:** fornece um modelo e um ambiente para programação e execução de aplicações que fazem uso dos dados disponibilizados pela EcoDiF, chamados de *feeds*. Estes *feeds* podem gerar novas informações, ficando estas disponíveis para serem usadas por outros *feeds* e aplicações mais complexas. Essas aplicações são construídas possibilitando o uso de *mashups*, pois podem ser criadas através da composição de diferentes tipos de informação, providas por diversas fontes, tais como serviços Web e bases de dados relacionais, com a intenção de complementar e melhorar a oferta de determinado serviço. Aplicações *mashup* são implementadas como *scripts* escritos em EEML e executadas em um motor (*engine*) de execução que processa tais *scripts*.
- **Módulo de Armazenamento:** consiste de dois repositórios básicos, um para armazenamento de dados utilizando uma base de dados relacional, e outro para armazenamento de *scripts* de aplicações em um sistema de arquivos. Esses repositórios podem fazer uso de uma infraestrutura de computação em nuvem para armazenar dados e arquivos, de forma a aproveitar os atributos de qualidade inerentes a este ambiente como robustez, confiabilidade, disponibilidade e escalabilidade.
- **Módulo de Visualização e Gerenciamento:** responsável por prover uma interface Web que permita aos usuários gerenciarem os dispositivos conectados à EcoDiF. Dentre as funcionalidades da interface Web está a de monitorar o estado e localização dos dispositivos, bem como visualizar dados históricos armazenados na plataforma, além de possibilitar o envio de notificações baseada em eventos (valores atuais dos *feeds*) que especificam condições previamente definidas.
- **Módulo de Manipulação de Dados:** tem como função tratar os dados recebidos dos dispositivos estruturados no protocolo EEML para após realizar o registro efetivo em uma base de dados relacional *MySQL*, utilizando as especificações da *Java Persistence API* (JPA) implementadas no *framework* Hibernate.
- **Módulo de Serviços Comuns:** é responsável por serviços de infraestrutura da plataforma, tais como segurança (permite o controle de autenticidade, confidencialidade e integridade de usuários utilizando as especificações *Java Authentication and Authorization Service* (JAAS) implementadas no servidor de aplicações JBoss), gerenciamento do ciclo de vida de aplicações e transações.

3.2 S³OIA - Smart Spaces and Smart Objects Interoperability Architecture

O trabalho S³OIA (VEGA-BARBAS et al., 2012) tem como objetivo contribuir para a padronização e interoperabilidade da IoT, através do desenvolvimento de uma arquitetura altamente distribuída, orientada a serviços e que integra diferentes objetos físicos e virtuais, abstraindo diferentes tecnologias. A arquitetura é alinhada com padrões Web RestFul e utiliza Espaço de Tuplas para expressar semanticamente informações dos diferentes dispositivos integrados pela plataforma. Além disso a arquitetura possibilita a utilização subjacente de recursos heterogêneos como um substrato para a composição automática de aplicativos complexos, criados de forma dinâmica e adaptável, uma vez que são capazes de evoluir em função do contexto em que são executados.

Devido a capacidade limitada de alguns dispositivos IoT, a arquitetura provê a utilização de gateways, que podem realizar funções de controle e gestão de recursos por estes dispositivos, além de realizar a abstração destes e normalização dos dados, de forma a tratar a heterogeneidade .

Neste trabalho é vislumbrado três níveis de abstração para um cenário IoT, mostrado na Figura 8. O nível mais baixo de abstração diz respeito a camada onde estão dispostos os dispositivos físicos. O nível intermediário é o da abstração dos serviços disponibilizados pelos dispositivos, podendo estes serem descobertos. Por fim a camada de mais alto nível retrata os serviço agrupados em um espaço inteligente virtual que permite a cooperação entre serviços.

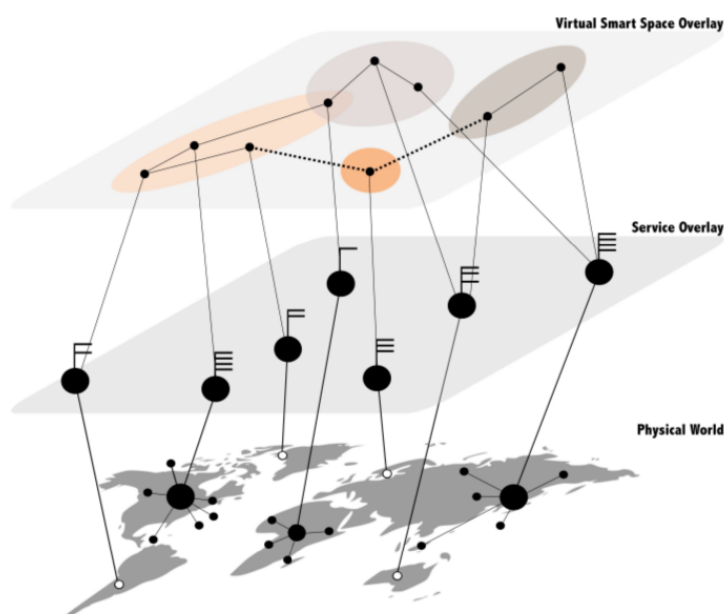


Figura 8: Níveis de Abstração S³OIA. Fonte: (VEGA-BARBAS et al., 2012)

A arquitetura do S³OIA, apresentada na Figura 9, é organizada em módulos e possui cinco grupos funcionais.

1. **Descoberta de Dispositivos e Serviços:** é composto pelos módulos responsáveis pela descoberta, integração e abstração de todos os dispositivos e seus serviços, não importando o protocolo de comunicação usado pelos mesmos. Este grupo contempla dois casos: (i) comunicação entre dispositivos capazes de implementar protocolos como DPWS e UPnP e (ii) comunicação entre estes e outros dispositivos que não possuem a capacidade de implementar estes protocolos, sendo esta situação possível devido a utilização de gateways que abstraem os dispositivos, fornecendo uma interface comum de comunicação e servindo de *proxy* para os protocolos envolvidos na troca de dados.
2. **Exposição de Web Services e Semântica Triple Spaces:** este conjunto de módulos possibilita realizar uma computação distribuída baseada na semântica *Triple Spaces* para sistemas ubíquos, de forma que dispositivos heterogêneos possam compartilhar informações assincronamente e de forma orientada a recursos. A computação *Triple Spaces* executa uma comunicação baseada em espaço de tuplas usando triplas RDF, em que a unidade de informação tem três dimensões que expressam os dados semânticos: sujeito, predicado e objeto. A *Triple Spaces* oferece também autonomia de referência, tempo e espaço, que habilita expor uma API compatível Web que objetiva compartilhar conhecimento entre grupos de nós semelhantes e cientes de contexto.
3. **Repositório de Serviços e Resolução de Dependências:** gerencia os serviços disponíveis dentro de um espaço inteligente e resolve as dependências extraídas da composição de aplicação em um contexto local. Além disso, ele possui um módulo gerenciador de eventos que segue o paradigma *publish/subscribe*, que facilita a criação de aplicações e a composição de serviços. Dessa forma, quando um novo serviço é disponibilizado ou quando um serviço já estabelecido se torna indisponível, seu estado é comunicado para todos os outros módulos envolvidos. O repositório de serviços representa uma instância de armazenamento de informações de todo o espaço IoT distribuído.
4. **Interface de Interação:** este grupo funcional gerencia a interação entre os seres humanos e a plataforma, dentro dos limites do espaço inteligente. A ideia principal deste grupo de módulos é a de recuperar informações a partir da interação dos usuários, as intenções, que podem ser convertidas em um tipo de ordem com o objetivo de compor aplicações complexas, ou para adequar os objetos já implantados às necessidades dos usuários.

5. **Composição, Tolerância a Falhas e Dependências Distantes:** responsável por tratar da composição e orquestração de serviços, além de gerenciar o acesso aos recursos da plataforma quando dois ou mais objetos requisitam-nos simultaneamente. Devido a arquitetura ser descentralizada, as inconsistências e dependências que não são resolvidas localmente são disparadas para camadas superiores. Existem também módulos responsáveis em tratar inconsistências, desconexões ou falhas na rede distribuída.

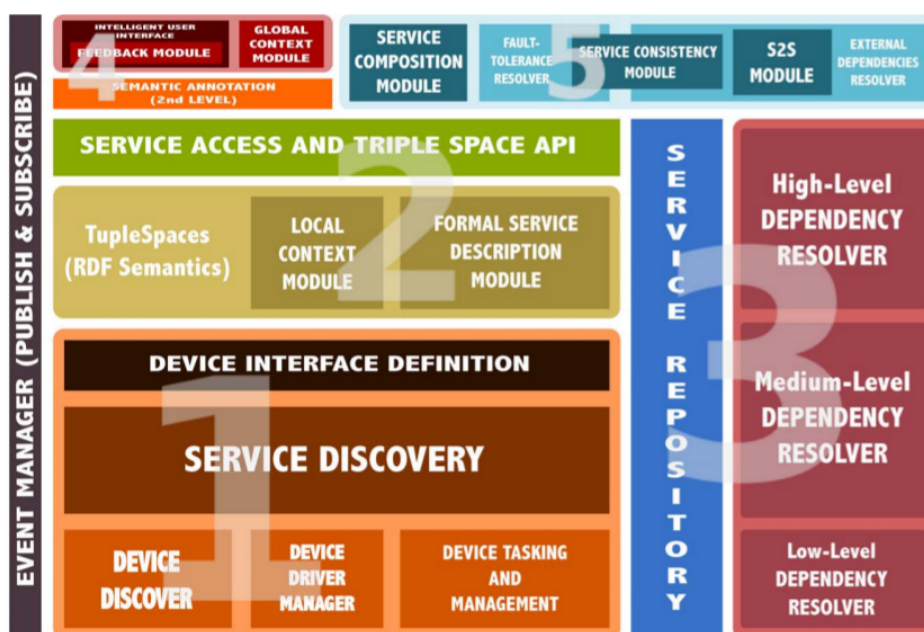


Figura 9: Arquitetura S³OIA. Fonte: (VEGA-BARBAS et al., 2012)

3.3 LinkSmart

O projeto LinkSmart (anteriormente chamado Hydra) é uma plataforma de middleware baseado em uma Arquitetura Orientada a Serviços (SOA - *Service Oriented Architecture*) e desenvolvido na linguagem Java, que oferece suporte ao desenvolvimento de aplicações baseadas em informações fornecidas por dispositivos físicos heterogêneos, disponibilizando interfaces de serviços Web para controle destes dispositivos (WIKI LINKSMART, 2015).

Os dispositivos físicos integrados a plataforma são classificados como: dispositivos nativos (com capacidade computacional para hospedar a plataforma) ou dispositivos não nativos (dispositivos restritos, sem capacidade para hospedar a plataforma), no entanto podem ser integrados a ela através de gateways.

Para os dispositivos não nativos, é necessário uma máquina hospedeira (gateway) com no mínimo 1GB de memória RAM livre e sistema operacional com Java 1.6, OSGi,

.NET 3.5 (ou mono com suporte WCF- *Windows Communication Foundation*). Esta máquina deverá conter uma *proxy* para cada dispositivo ligado a ela, a fim de intermediar a comunicação da plataforma com estes dispositivos. Pela *proxy*, o gateway fica capacitado a trocar dados com o dispositivo, colocando-o na rede LinkSmart. Já os dispositivos nativos devem possuir no mínimo 256MB de RAM livre e sistema operacional com Java 1.6, OSGi, para hospedarem os códigos Java e bibliotecas LinkSmart e OSGi, necessários para integrá-los diretamente a plataforma.

Sua arquitetura possui três camadas principais: (i) camada de rede, responsável pela comunicação com os dispositivos; (ii) camada de serviço, responsável pelo gerenciamento de eventos, dispositivos, escalonamento de recursos, dentre outros, e; (iii) camada semântica.

A arquitetura LinkSmart é apresentada na Figura 10 e seus componentes caracterizados a seguir.

- **Gateway:** é representado por dispositivos computacionais com conexão à Internet contendo *proxies* que traduzem o tipo de tecnologias e formatos de dados empregados pelos dispositivos físicos ligados a ele. É responsável por integrar os dispositivos que não são nativos à rede LinkSmart. Possui suporte a conexão com estes dispositivos através de *USB*, *ZigBee* e *Bluetooth*. O gateway deve executar um gerenciador de rede que registre todos os serviços que pertencem aos dispositivos na rede LinkSmart.
- **Network Manager:** implementa serviços Web sobre a especificação JXTA para o modelo *Peer-to-Peer* com o intuito de realizar a comunicação entre os dispositivos.
- **Event Manager:** gerencia a troca de informações através do modelo de comunicação *Publish-Subscribe*, entre processos de todas as propriedades de dados não funcionais para serviços/componentes, dispositivos e rede.
- **Crypto and Trust:** realiza operações criptográficas, a avaliação de confiança e a execução de políticas de segurança de controle de acesso.
- **Device Application Catalogue (DAC):** responsável por manter o controle de dispositivos disponíveis na rede LinkSmart e acesso destes pelas aplicações.

A LinkSmart também contempla uma descrição semântica dos dispositivos através do uso de ontologias de dispositivos, capazes de representar todas as meta-informações sobre os dispositivos. A ontologia utilizada é baseada na ontologia de dispositivos FIPA (*Foundation for Intelligent Physical Agents*) que permite a

parametrização semântica para incluir informações dos dispositivos, como seus recursos de segurança.

O módulo *Application Ontology Manager* é o responsável por prover uma interface para o uso da ontologia de dispositivos, tanto para buscar propriedades e funções dos dispositivos, quanto para descobertas e atualizações de dispositivos. Nesse contexto, a camada semântica da LinkSmart contém: (i) um módulo de inferências (*reasoner*), responsável por inferir sobre o status dos dispositivos e indicar qual tipo de dispositivo entrou na rede; (ii) um módulo de consulta (*query*) para recuperar informações sobre os dispositivos e suas capacidades; (iii) um módulo de atualização, que permite inclusão, remoção e mudanças na ontologia; (iv) um módulo de versionamento, que gerencia diferentes versões da ontologia, incluindo diferentes versões dos dispositivos e serviços, e; (v) um módulo de interpretação e anotação, responsável por automaticamente atualizar a ontologia com novos tipos de dispositivos e por realizar análise e anotação dos dispositivos existentes e descrições que são incluídas na ontologia.

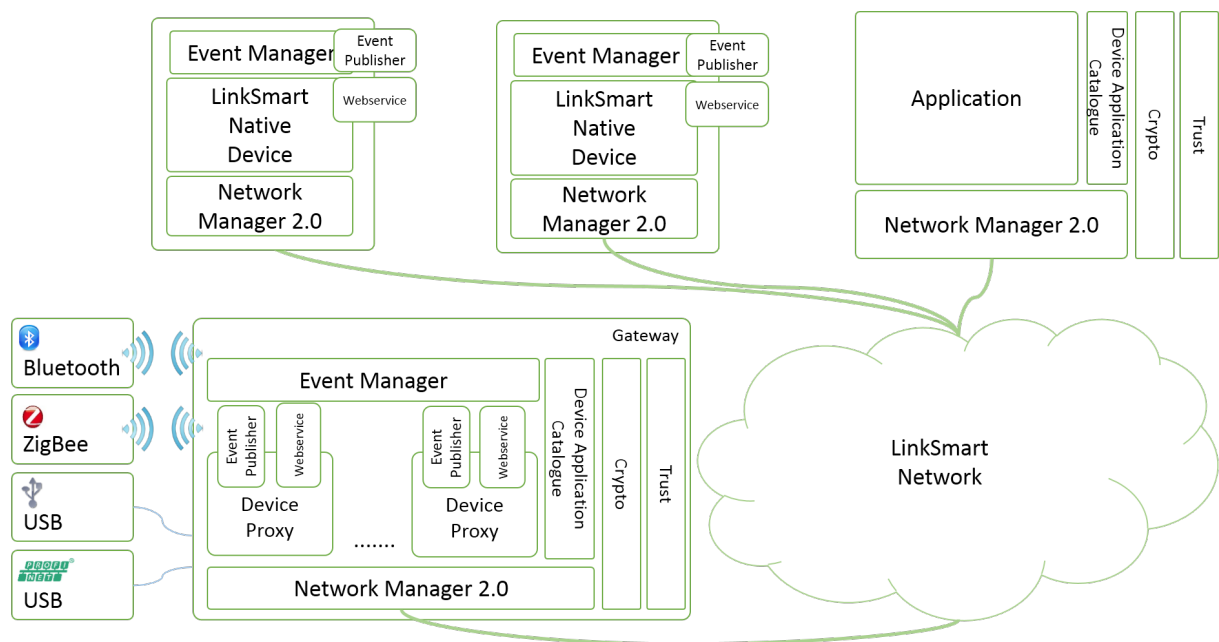


Figura 10: Arquitetura LinkSmart. Fonte: (WIKI LINKSMART, 2015)

3.4 Carriots

A Carriots é uma plataforma para aplicações em IoT que utiliza serviços de nuvem para gerenciar dados providos por qualquer tipo de dispositivo, além de conectar dispositivos a dispositivos e a outros sistemas, o que a faz se alinhar ao conceito de *Platform as a Service* (PaaS) da computação em nuvem (CARRIOTS, 2015). Portanto, se um sistema for conectado à plataforma, ele também pode ser modelado como um dispositivo. A partir de sua API RESTful, a Carriots tem por objetivo coletar e armaze-

nar qualquer dado originado dos mais diversos dispositivos, utilizá-lo em seu motor de aplicações e disponibilizá-lo a seus usuários não importando o volume de dispositivos conectados.

A plataforma Carriots foi projetada tendo como base dois blocos principais: (i) organização lógica de entidades e (ii) arquitetura de componentes de software.

No bloco de organização lógica de entidades, as entidades são definidas de forma hierárquica como demonstra o exemplo da Figura 11. Nela se verifica que os dispositivos são organizados primeiramente em grupos, de acordo com a sua localização geográfica. Por sua vez estes grupos são associados a um serviço que pertence a um projeto específico desenvolvido pelo usuário. Dessa forma, a primeira atividade de um usuário ao interagir com a plataforma é a criação de um projeto. Por conta dessa hierarquia, se um projeto for desabilitado, isso impactará na desativação de todas as entidades (serviços, grupos e dispositivos) vinculadas a ele.

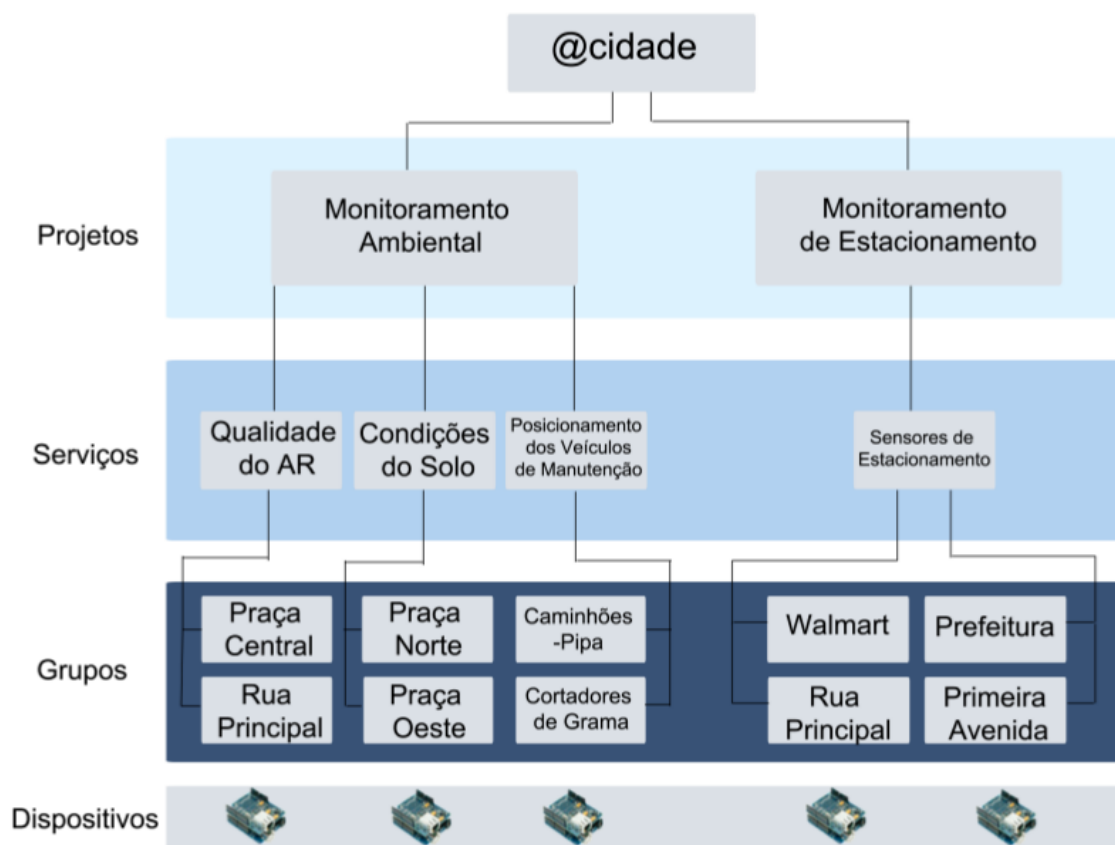


Figura 11: Exemplo de Hierarquia das Entidades na Plataforma Carriots. Fonte: (PIRES et al., 2015)

Outro componente que pode ser inserido nesta hierarquia de entidades é o *Listener*, responsável por realizar o monitoramento de eventos, como o recebimento ou persistência de dados, e caso um evento monitorado atenda alguma condição pré-configurada executar a análise e processamento dos dados relacionados a este

evento.

A arquitetura de software do Carriots (vide Figura 12) é formada pelos seguintes módulos:

- **API REST:** seguindo padrões da Internet esta API REST é implementada sobre HTTPS, sendo responsável pela forma como os dispositivos e outros sistemas interagem com a plataforma. A API REST utiliza os formatos JSON ou XML (em um formato particular da plataforma) para representação dos dados. Possui plena capacidade de interação entre a plataforma e os painéis de controle personalizados, *dashboards*, etc.
- **Big Data:** provê as aplicações a flexibilidade de gerenciar os dados dos mais diversos dispositivos, de modo que a massa de dados seja armazenada em uma arquitetura de *Big Data* em formato *schemaless*, permitindo assim armazenar dados sem que seja necessário normalizá-los.
- **Gerenciamento de Projetos e Dispositivos:** possui como funcionalidade armazenar e gerenciar os projetos criados pelos usuários (dos mais simples aos mais complexos), podendo estes serem organizados para atenderem a quaisquer requisitos. Este módulo é também responsável por realizar a configuração dos dispositivos, como por exemplo o ajuste na taxa de amostragem de um sensor de estacionamento e atualizar os seus softwares embarcados, possuindo a capacidade de realizar o gerenciamento remoto dos dispositivos .
- **Processamento de Eventos e Regras de Negócio:** estes módulos são responsáveis por armazenar e executar eventos, segundo uma lógica de negócios, em forma de *scripts* criados utilizando a linguagem de programação *Groovy* e fazendo uso de regras do tipo *if-then-else*.
- **Segurança:** neste módulo questões de segurança podem ser tratadas de quatro formas: (i) através do uso de chaves (*APIkeys*) pré-compartilhadas para a definição de privilégios de acesso; (ii) através da utilização do protocolo HTTPS para a criptografia das requisições e respostas à API REST; (iii) pela utilização de Hash HMAC (*Keyed-Hash Message Authentication Code*) e senha para autenticação e verificação de conteúdo, e; (iv) por criptografia customizada a partir de *scripts* criados pelo usuário, permitindo a adição de soluções de segurança adicionais.
- **Logs e Debug:** fornece um console de *debug* que facilita o desenvolvimento dos projetos hospedados na plataforma, através de ferramentas para depuração de erros e registro de mensagens, tornando tais dados (mensagens de *log*) acessíveis a partir do painel de controle.

- **Painel de Controle:** permite o gerenciamento pelo usuário de todos os outros módulos e recursos da plataforma a partir de uma interface Web.
- **Módulo de Comunicação Externa:** complementa os recursos de interação da plataforma ao permitir o envio de *e-mails*, SMS e a interação com outros sistemas (como a plataforma de compartilhamento de arquivos Dropbox ou a rede social Twitter).

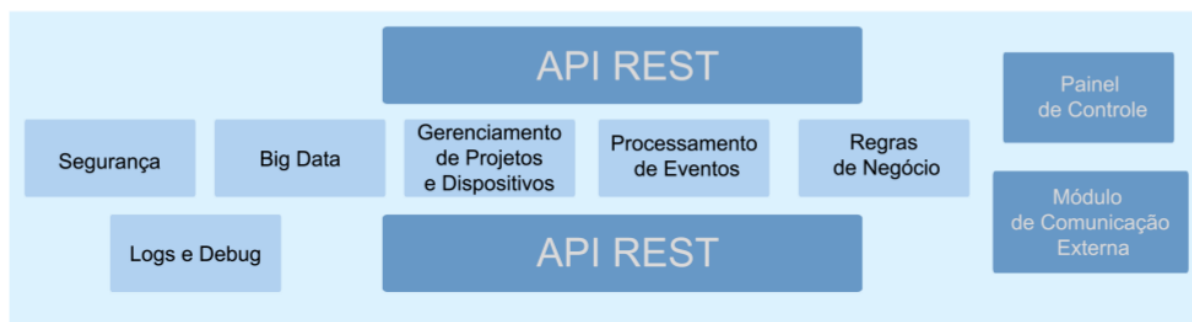


Figura 12: Arquitetura Carriots. Fonte: (PIRES et al., 2015)

3.5 Análise dos Trabalhos Relacionados

Uma análise das plataformas de middleware discutidas anteriormente em relação à satisfação dos requisitos IoT considerados neste trabalho é apresentada na Tabela 6, onde o símbolo “X” denota que o requisito é completamente atendido, o símbolo “O” indica que o requisito é parcialmente atendido, e o símbolo “-” indica que o requisito não é atendido.

Esta análise é ampliada ao final do Capítulo 4, que discute a concepção do EXEHDA-IoT, onde é feita uma comparação do mesmo com os trabalhos relacionados.

Tabela 6: Análise dos Trabalhos Relacionados

Plataformas de middleware	Interoperabilidade	Descoberta e Gerenciamento de Dispositivos	Interfaces de Alto Nível	Ciência de Contexto	Escalabilidade
EcoDiF	X	O	X	O	X
S³OIA	X	X	-	X	-
LinkSmart	X	O	-	X	-
Carriots	X	O	X	O	X

Diante do exposto na Tabela 6, constata-se que nenhuma das plataformas de middleware avaliadas atende plenamente a todos os requisitos levantados. Tais plataformas tratam apenas subconjuntos dos requisitos, de acordo com sua aplicação e objetivos.

Pode-se verificar também que dentre os requisitos considerados, o único que foi atendido por todas as plataformas é o de Interoperabilidade. Isto é devido ao fato deste ser um requisito primordial e que deve ser imperativamente endereçado por qualquer plataforma de middleware para IoT.

Além disso, pela análise dos trabalhos relacionados foi possível sistematizar algumas características comuns, como a tendência de middlewares IoT em usar princípios arquiteturais REST como meio para disponibilização dos dispositivos e seus recursos, devido ao seu estilo simples e de fácil interpretação computacional, e por este utilizar padrões abertos e protocolos amplamente utilizados na Internet, como o HTTP, URI, XML e outros.

Outra característica que se destaca quando da análise dos trabalhos relacionados é a adoção de gateways como forma de abstrair e integrar dispositivos de sensoria-mento e/ou atuação com recursos computacionais limitados, a fim de tratar a hetero-geneidade e normalizar os dados produzidos.

3.6 Considerações sobre o Capítulo

Este capítulo apresentou trabalhos que propuseram middlewares para IoT e que se mostraram relacionados a proposta desta dissertação de mestrado. A descrição de cada trabalho buscou caracterizar a sua arquitetura de software, bem como as abordagens utilizadas para interoperação entre os dispositivos da IoT. Também uma análise dos trabalhos relacionados foi realizada, tendo em vista a satisfação dos requi-sitos para a concepção de middlewares IoT. Ao final, foram sistematizadas algumas características das estratégias utilizadas no intuito de identificar tendências no tema da pesquisa, que viessem a contribuir na concepção do EXEHDA-IoT.

4 EXEHDA-IOT: VISÃO GERAL E MODELAGEM

Este capítulo apresenta os aspectos de concepção e modelagem da abordagem proposta por esta dissertação de mestrado, denominada EXEHDA-IoT.

A premissa de concepção foi contribuir com o Subsistema de Adaptação e Reconhecimento do Contexto do EXEHDA, de forma a qualificar o middleware às demandas inerentes às modernas infraestruturas da Internet das Coisas, promovendo neste cenário suporte às aplicações cientes de contexto. Também neste capítulo é realizada uma discussão dos trabalhos relacionados.

4.1 EXEHDA-IoT: Visão Geral

O EXEHDA (LOPES et al., 2014) é um middleware baseado em serviços que visa criar e gerenciar um ambiente ubíquo, bem como promover a execução de aplicações sobre esse ambiente. Seu foco é permitir que as aplicações possam obter informações de seus contextos de interesse e reagir às variações que acontecem nos mesmos.

A arquitetura do EXEHDA, apresentada na Figura 13, fornece uma solução integrada para construir e executar aplicações distribuídas em grande escala. A arquitetura é dividida em três camadas: camada inferior, composta pela infraestrutura para execução das outras camadas (máquina virtual, sistema operacional e hardware); camada intermediária, constituída pelo Middleware EXEHDA; e a camada superior, formada pelo *framework* para programação das aplicações ubíquas. O módulo que representa a Consciência ao Contexto tem destaque na arquitetura, uma vez que permeia os demais componentes. Esse módulo tem por finalidade gerenciar os contextos de interesse das aplicações e do próprio ambiente de execução.

No EXEHDA-IoT, as condições de contexto são pró-ativamente monitoradas e o suporte à execução deve permitir que tanto as aplicações como ele próprio utilizem essas informações na gerência de seus aspectos funcionais e não-funcionais. O EXEHDA-IoT herda as características do mecanismo de ciência de contexto proposto para o EXEHDA, o qual emprega uma estratégia colaborativa entre aplicação e ambiente de execução, através da qual é facultado ao programador individualizar políticas para re-

ger o comportamento de cada um dos componentes que constituem o software da aplicação.

A estrutura de software do EXEHDA contempla um núcleo mínimo e serviços carregados sob demanda, os quais estão organizados nos seguintes subsistemas: (i) Acesso Ubíquo, (ii) Comunicação, (iii) Comunicação Distribuída e (iv) Adaptação e Reconhecimento do Contexto (vide Figura 13).

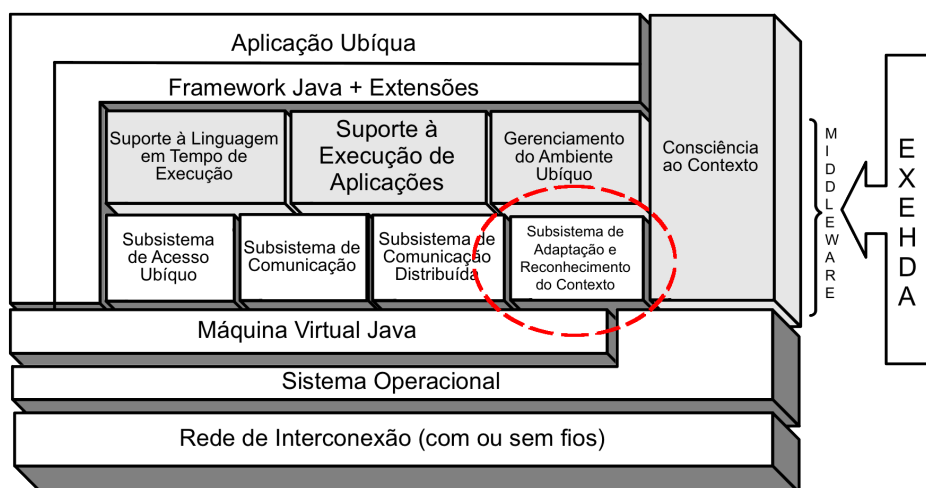


Figura 13: Arquitetura do Middleware EXEHDA

A abordagem proposta por esta dissertação de mestrado tem por foco contribuir especificamente com o Subsistema de Adaptação e Reconhecimento do Contexto, ampliando suas funcionalidades de forma a atender demandas inerentes da IoT. Todo esforço de concepção do EXEHDA-IoT considerou sua integração na arquitetura de software do middleware EXEHDA, bem como os seus princípios operacionais.

Para a concepção do EXEHDA-IoT, foram considerados os seguintes requisitos: (i) interoperabilidade, (ii) descoberta e gerenciamento de dispositivos, (iii) interfaces de alto nível, (iv) ciência de contexto e (v) escalabilidade.

O fato do EXEHDA ter uma origem enquanto middleware para UbiComp, faz com que o tratamento da Ciência de Contexto seja inerente à arquitetura. O desafio, o qual esta dissertação tem como objetivo tratar, está em como capacitar o middleware para que este continue provendo suporte a este quesito, mediante a aquisição do contexto a partir de um grande número de dispositivos heterogêneos, distribuídos dinamicamente e com capacidade restrita, de forma a promover a interoperabilidade entre estes, bem como a sua interação, o mais transparente possível com diferentes sistemas computacionais e/ou usuários.

Considerando isso, o EXEHDA-IoT foi modelado empregando os seguintes pressupostos:

- automatização: componente chave que implica que um middleware para IoT

deva suportar coleta de dados, processamento e inferência contextual de forma autônoma;

- inteligência: dispositivos IoT devem estar capacitados a operar, com algum grau de adaptação as variações contextuais (variações nas infraestruturas de rede, aplicações e usuários, por exemplo);
- dinamicidade: um dispositivo pode se mover de um lugar a outro, necessitando que seu middleware esteja apto para reconhecer esta mudança e consequentemente realizar a adaptação necessária baseada no seu ambiente;
- zero Configuração: para simplificar os procedimentos de integração e/ou remoção de dispositivos, recursos *plug and play* devem estar disponíveis, reduzindo os esforços de gerência por parte dos usuários e facilitando o crescimento descentralizado das infraestruturas para IoT.

Para atender estes pressupostos o mecanismo proposto para o EXEHDA-IoT introduz um novo elemento na arquitetura de software do Subsistema de Adaptação e Reconhecimento do Contexto (SARC): o Gateway.

Este novo elemento tem como objetivo tratar a obtenção de informações contextuais, fornecida por diferentes dispositivos de sensoriamento, de maneira a uniformizar a diversidade de tecnologias. Tendo o objetivo de prover aos demais componentes da arquitetura estruturas de dados padronizadas, agregando as mesmas informações provenientes de sensores. O gateway também tem a função de interpretar comandos de atuação originados pelo middleware, que culminam na ativação de atuadores eletromecânicos.

As estruturas de dados padronizadas são manipuladas e disponibilizadas às aplicações por meio de uma API REST. Assim, no EXEDA-IoT, os serviços providos tanto pelos sensores quanto pelos atuadores, podem ser acessados como recursos Web, independentemente da linguagem de programação utilizada na aplicação.

Outrossim, o EXEHDA-IoT contribui com a publicação e processamento destas informações captadas, qualificando a disponibilização das mesmas para o usuário final, de acordo com o seu contexto de interesse.

Na perspectiva de atender a elevada dinamicidade e escalabilidade do cenário da IoT, o EXEHDA-IoT possibilita a gerência de dispositivos, com características de “zero configuração” por meio de recursos *plug and play*, aos usuários responsáveis pelas aplicações por meio de uma interface de alto nível.

4.2 EXEHDA-IoT: Modelo Arquitetural

Para o ambiente computacional ubíquo gerenciado pelo middleware EXEHDA foi proposta uma nova organização, conforme apresentado na Figura 14.

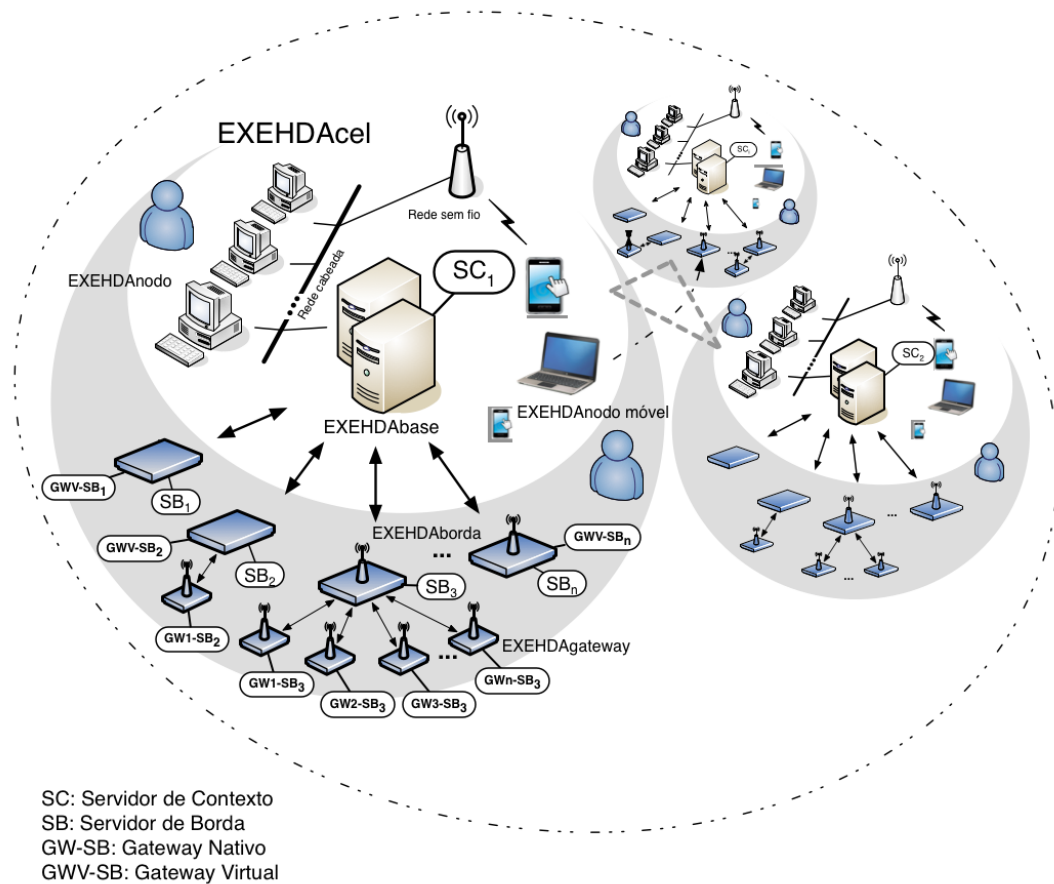


Figura 14: Ambiente Ubíquo: Organização Proposta para o EXEHDA-IoT

Considerando a perspectiva de promover a computação de aplicações, atendendo a critérios de distributividade, escalabilidade, heterogeneidade, mobilidade e adaptabilidade ao contexto, este ambiente é constituído por células de execução.

Deste modo os dispositivos computacionais são distribuídos entre as células (vide Figura 14), sendo cada célula constituída dos seguintes componentes: (i) EXEHDAbase, o elemento central da célula, sendo responsável por todos serviços básicos e constituindo referência para os demais elementos; (ii) o EXEHDAanodo que corresponde aos dispositivos computacionais responsáveis pela execução das aplicações; (iii) o EXEHDAanodo móvel, um subcaso do anterior, que corresponde aos dispositivos tipicamente móveis que podem se deslocar entre as células do ambiente ubíquo, como *notebooks*, *tablets* ou *smartphones*, (iv) o EXEHDAborda, responsável por fazer a interoperação entre os serviços do middleware e os diversos tipos de gateways; e (V) o EXEHDAgateway, que consiste no elemento responsável por setorizar pontos de coleta e/ou atuação distribuídos, disponíveis no meio físico, realizando a interação destes com os outros componentes do middleware.

Para provimento de ciência de contexto no ambiente ubíquo, o EXEHDA-IoT se vale de dois tipos principais de servidores da arquitetura do EXEHDA: Servidor de Borda

e Servidor de Contexto (vide Figura 14). O Servidor de Borda se destina a gerenciar a interação com o meio físico através de gateways, sendo instanciado em um equipamento do tipo EXEHDAborda. O Servidor de Contexto, por sua vez, é alocado no EXEHDAbase e atua no armazenamento e no processamento das informações contextuais, integrando dados históricos e dados provenientes de diferentes Servidores de Borda distribuídos no ambiente ubíquo.

Na proposta do EXEHDA-IoT a premissa é que os sensores e/ou atuadores sejam integrados ao Servidor de Borda somente através de gateways. Os gateways são utilizados então, para tratar os diversos tipos de protocolos físicos inerentes a dispositivos de sensoriamento e/ou atuação, bem como garantir que dispositivos com capacidade restrita, tanto computacional como energética, possam se comunicar com o Servidor de Borda via TCP/IP.

Os gateways possuem hardware e capacidades computacionais limitadas, sendo dedicados para tratar tecnologias específicas, como redes de sensores sem fio, por exemplo, fazendo a conversão de protocolos e o gerenciamento dos dispositivos. Atualmente, vários modelos de gateways IoT vem sendo disponibilizados no mercado (GAZIS et al., 2015).

Com o intuito de garantir a interoperabilidade com as tecnologias de mercado, e também potencializar a distribuição das iniciativas de coleta e/ou atuação, foram previstos três tipos distintos de gateways:

- Gateways proprietários (GW-PR), que tem funcionalidades e protocolos heterogêneos variando de acordo com seus fabricantes;
- Gateways Nativos (GW-SB), cujas funcionalidades operam de maneira integrada à arquitetura do EXEHDA, sendo instanciados em equipamentos do tipo EXEHDAgateway e;
- Gateway Virtual (GWV-SB), localizado no mesmo hardware em que executa o Servidor de Borda, constituindo-se em uma virtualização do Gateway Nativo, dispensando quando possível a necessidade de um hardware específico para o gateway.

O suporte arquitetural aos Gateways proprietários depende dos recursos oferecidos pelas suas APIs, sendo garantida, a priori, apenas a leitura dos valores adquiridos e o envio de comandos de atuação, quando suportados. Por outro lado, os Gateways Nativos e Gateways Virtuais, além de oferecer acesso direto aos seus recursos (atuação e sensoriamento), com suporte a descoberta automática tanto dos dispositivos gateways como de seus serviços ou recursos, possuem mecanismos de sensoriamento gerenciados através de eventos utilizando estratégias de *trigger*.

Na Figura 15 é apresentada a nova arquitetura de software do Servidor de Borda e do Servidor de Contexto, que constituem o Subsistema de Adaptação e Reconhecimento do Contexto do middleware EXEHDA. Deste modo, esta dissertação contribui com os esforços de qualificar o middleware EXEHDA para estes novos cenários introduzidos pelas infraestruturas computacionais modernas (DAVET et al., 2015), (SOUZA et al., 2015).

A figura caracteriza as relações entre Servidor de Borda e Contexto e destes com os dispositivos gateways que interagem com o ambiente físico, por meio de sensores e/ou atuadores. As linhas tracejadas na representação dos gateways indicam que algum dos tipos de gateways podem não estar presentes na arquitetura, a qual possibilita utilizar somente Gateways físicos (nativos ou proprietários), somente Gateway Virtual, ou ambos.

4.2.1 Servidor de Contexto

O Servidor de Contexto no EXEHDA-IoT tem suas funcionalidades organizadas através dos seguintes módulos arquiteturais.

Repositório de Contextos

O Repositório de Contextos foi reformulado de forma a refletir a organização proposta para o EXEHDA-IoT (vide Anexo B), contemplando as relações entre as aplicações, componentes, sensores, ambientes e contextos de interesse. O repositório armazena dados de configuração da arquitetura, e os valores sensorizados publicados pelos Servidores de Borda existentes no ambiente ubíquo. Estes dados são usados pelo Processador sempre que é necessária a utilização de valores históricos nas lógicas de processamento. O Repositório de Contextos emprega um modelo relacional para a representação do contexto fornecendo uma visão histórica das informações contextuais armazenadas.

Processador de Contexto

O Processador de Contexto realiza tarefas de manipulação e dedução das informações contextuais. Baseia-se em regras do tipo ECA (Evento-Condição-Ação), de forma a tratar eventos gerados a partir de mudanças nos estados dos contextos de interesse, que podem acarretar no disparo de ações pertinentes em função do estado contextual. A recepção de uma publicação de valores sensorizados pelo Servidor de Borda pode ser o gatilho para o disparo de regras. Um exemplo de processamento de regras do tipo ECA, utilizado no cenário avaliado pelo estudo de caso do EXEHDA-IoT é apresentado na Figura 16 através de uma notação BPMN (GROUP, 2015).

O cenário consiste em monitorar uma determinada grandeza e produzir registro histórico. Enquanto a grandeza é coletada e armazenada, a mesma tem seu valor

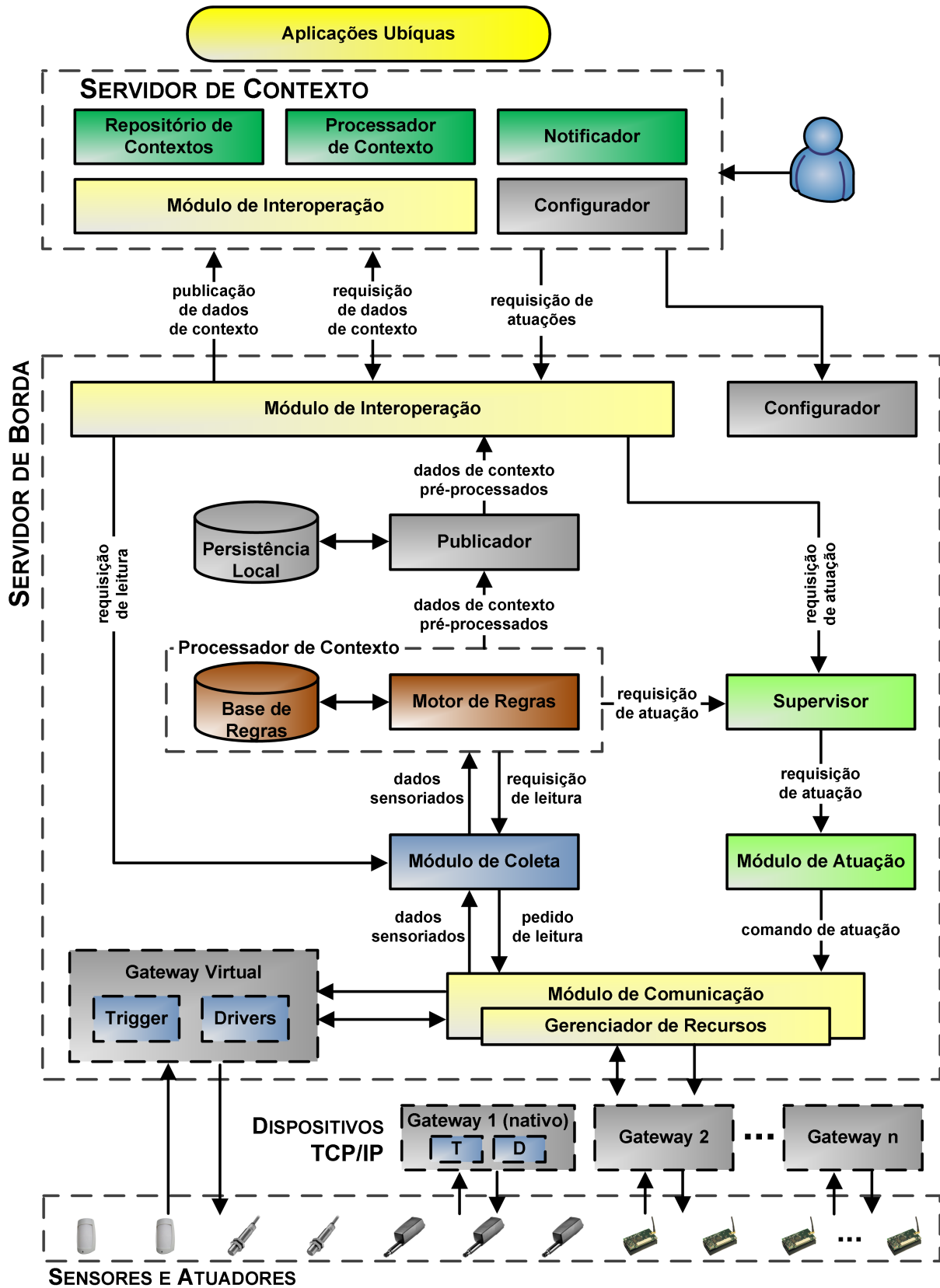


Figura 15: Arquitetura do Subsistema de Adaptação e Reconhecimento do Contexto do Middleware EXEHDA (DAVET et al., 2015),(SOUZA et al., 2015)

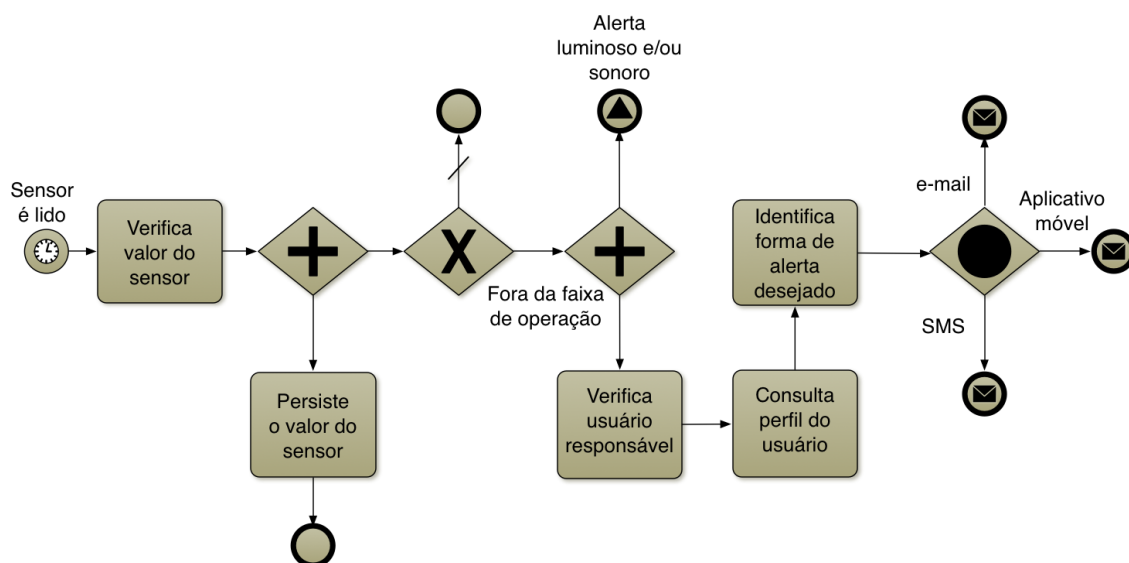


Figura 16: Processamento de uma Regra Tipo ECA (Notação BPMN)

avaliado tendo em vista as faixas de operação para cada dado de contexto coletado. Alertas são enviados aos usuários envolvidos, quando os valores estiverem fora da faixa de operação especificada para o experimento.

Embora o Processador de Contexto utilize regras ECA como mecanismo básico de tratamento do contexto, tanto a condição a ser tratada quanto a ação a ser executada pela regra admitem outros modelos de processamento que podem ser chamados pela regra, os quais são decorrência do tipo de domínio de aplicação.

Notificador

O Notificador tem a função de gerar notificações necessárias para a operação do EXEHDA-IoT. As notificações são consequências do processamento das informações contextuais coletadas. Este módulo utiliza uma estratégia de notificação baseada no modelo *publish/subscribe*, em que recebe subscrições de todos os serviços e/ou aplicações que requerem notificações a respeito das mudanças nos estados contextuais.

Também é função do Notificador enviar comandos de atuação aos Servidores de Borda quando necessário, bem como a realização de notificação pela aplicação WEB de novo gateway descoberto (vide Figura 17), o qual só será utilizado mediante autenticação pelo usuário interessado.

Módulo de Interoperação

Considerando o perfil inerentemente distribuído das aplicações alvo do EXEHDA-IoT, o Módulo de Interoperação foi concebido para gerenciar a troca de informações com os Servidores de Borda, bem como os demais serviços do middleware em

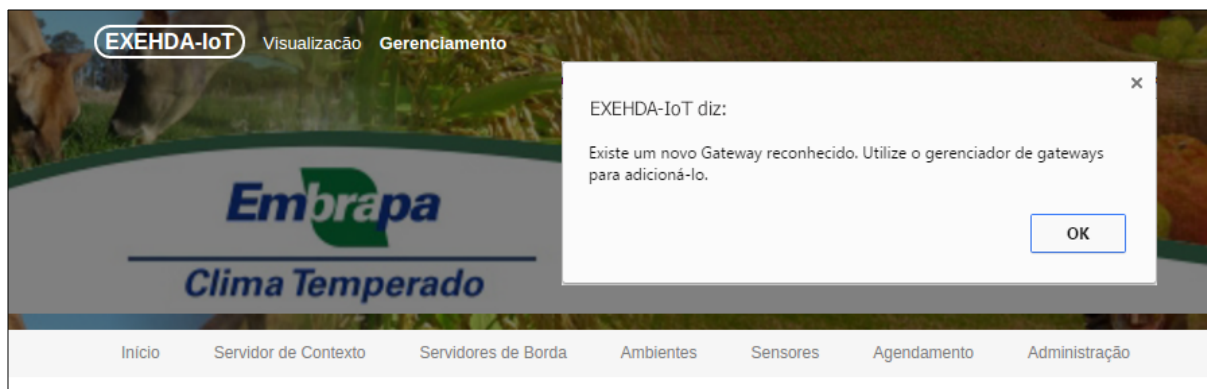


Figura 17: Notificação de Gateway Descoberto

execução em outros equipamentos, como também as aplicações, oferecendo para isso uma interface padronizada e bem definida. A concepção deste módulo teve como referência o estilo arquitetural REST, sendo utilizado o protocolo HTTP e suas operações (GET, PUT, POST, DELETE) no suporte às trocas de informações.

O Módulo de Interoperação provê o suporte necessário para disponibilização de uma API REST para o acesso aos recursos oferecidos pelo Servidor de Contexto. Com isso, a interoperação ocorre por meio de operações HTTP sobre URIs, os quais são o caminho para o acesso a todos os recursos disponibilizados pela arquitetura.

Configurador

O módulo Configurador aglutina todos os procedimentos de configuração necessários para o funcionamento do EXEHDA-IoT. Através deste módulo é disponibilizada uma aplicação através da qual é possível gerenciar dispositivos gateways e regras de processamento de contexto, bem como realizar todo tipo de cadastro necessário ao funcionamento da aplicação IoT ciente de contexto.

4.2.2 Servidor de Borda

No EXEHDA-IoT o Servidor de Borda tem suas funcionalidades sistematizadas através dos seguintes módulos arquiteturais.

Módulo de Interoperação

O Módulo de Interoperação do Servidor de Borda é destinado a gerenciar a comunicação com o Servidor de Contexto, tendo também como referência o estilo arquitetural REST, constituindo um serviço Internet. Com isso, todas as funcionalidades associadas aos gateways, sensores e atuadores são mapeadas na forma de recursos e acessadas via URIs. Através desse módulo é disponibilizada uma interface padronizada para manipulação dos dispositivos conectados ao Servidor de Borda. Estes dispositivos podem ser sensores ou atuadores e a manipulação dos mesmos pode

ocorrer a partir de diferentes serviços do middleware.

Configurador

No Servidor de Borda o módulo Configurador é responsável pelos ajustes necessários para o seu funcionamento. Entre as funcionalidades oferecidas tem-se: definição do Servidor de Contexto para a publicação dos dados sensorizados, bem como a inserção, atualização, ativação e desativação dos gateways e seus serviços por meio do protocolo de comunicação UPnP.

Publicador

O módulo Publicador tem a função de disparar as requisições de envio de informações contextuais para as demais camadas do middleware, trocando dados com o Servidor de Contexto através do Módulo de Interoperação. As publicações são organizadas em um sistema de fila FIFO e são processadas conforme a disponibilidade da rede. Considerando as possíveis falhas de comunicação entre o Servidor de Borda e o Servidor de Contexto bem como eventuais atrasos da rede, é empregado um módulo de **Persistência Local** cuja função é realizar o armazenamento temporário da fila de informações contextuais até que as mesmas sejam publicadas.

Processador de Contexto

Os Processadores de Contexto tanto do Servidor de Borda como do Servidor de Contexto, foram concebidos tendo por referência o modelo de regras ECA, as quais podem ser disparadas a partir de eventos produzidos pelos sensores ou outros serviços do middleware. O **Motor de Regras** é o módulo responsável pelo processamento das regras, as quais são armazenadas na **Base de Regras**.

A frequência de aquisição e os valores contextuais aquisitados, de forma combinada ou não, são tratados através de regras pelo Servidor de Borda, as quais devem ser elaboradas de acordo com as necessidades da aplicação do usuário final e considerando o comportamento autônomo esperado por parte dos dispositivos da IoT.

As regras submetidas ao Processador de Contexto do Servidor de Borda devem ser elaboradas de forma a atender, prioritariamente, os eventos críticos. Isso se deve ao fato de que o EXEHDAborda é geralmente alocado fisicamente próximo ao ambiente monitorado permitindo uma atuação (alertas, ativação/desativação de equipamentos eletromecânicos, etc.), independentemente de uma eventual perda de comunicação com o Servidor de Contexto por decorrência de uma falha na rede. Por outro lado, regras que necessitem incluir o tratamento de informações históricas, que necessitem o acesso a dados coletados de outros Servidores de Borda ou que envolvam outros modelos de processamento de contexto, devem ser processadas no Servidor de Contexto.

Supervisor

O módulo Supervisor possui como objetivo aglutinar os comandos de atuação. Uma vez recebidos os parâmetros para controle da atuação, o módulo Supervisor avalia eventuais conflitos entre as requisições de atuação oriundas de diferentes fontes, para após enviar o comando de atuação ao Módulo de Atuação.

Módulo de Coleta

O Módulo de Coleta tem a função de direcionar os pedidos de coleta ao Módulo de Comunicação que envia aos respectivos gateways, sob demanda tanto do Motor de Regras, como do Servidor de Contexto ou das aplicações a qualquer momento. O módulo faz a recepção assíncrona das solicitações e repassa o ID do sensor a ser lido pelo gateway responsável. Este módulo também tem a função de direcionar aos demais elementos da arquitetura os dados sensorizados enviados pelos gateways, produzidos a partir de eventos do ambiente.

Módulo de Atuação

O Módulo de Atuação possui um funcionamento análogo ao Módulo de Coleta. Ele recebe do Supervisor comandos com o ID do atuador e os correspondentes padrões de operação (tempo de duração, potência de ativação, etc.), e os envia ao Módulo de Comunicação a fim de se serem interpretados e direcionados ao gateway responsável pelo efetivo tratamento.

Gateway Virtual

O Gateway Virtual, que consiste na virtualização do Gateway Nativo, realiza o interfaceamento entre sensores e/ou atuadores (dispositivos sem conexão Internet) ligados diretamente ao Servidor de Borda. Possui dois módulos básicos: **Drivers** e **Trigger**. Os Drivers, enquanto módulos arquiteturais, são responsáveis pelo acesso aos valores das grandezas físicas medidas pelos sensores bem como pelos comandos de atuação enviados aos atuadores. Eles encapsulam as tecnologias utilizadas pelos dispositivos e controlam os sensores e atuadores de maneira individualizada, o que evita que as diferenças operacionais dos dispositivos se projetem nos demais módulos da arquitetura.

O Trigger é um módulo que tem a finalidade de gerenciar a leitura dos sensores através de eventos. Consiste em um tipo simplificado de regra de produção, que dá suporte a um conjunto limitado de funções para o tratamento de condições e ações.

Os Triggers concebidos para o EXEHDA-IoT contemplam o gerenciamento dos dois principais tipos de eventos da IoT: eventos temporais e eventos ambientais (PERRERA et al., 2013). Eventos ambientais contemplam aqueles em que dados devem ser produzidos na ocorrência de situações específicas, como uma porta sendo fe-

chada, uma luz sendo acesa, ou uma temperatura atingindo certo valor. Por outro lado, eventos temporais são utilizados quando se necessita produzir dados de forma contínua, por exemplo, o registro histórico da temperatura ambiente.

A troca de dados entre o Gateway Virtual e os demais elementos que constituem o Servidor de Borda é gerenciada pelo Módulo de Comunicação utilizando API REST. De forma similar, o Gateway nativo também utiliza o mesmo método de comunicação, porém neste caso envolvendo fisicamente a rede TCP/IP.

Módulo de Comunicação

O Módulo de Comunicação no EXEHDA-IoT tem a função de prover aos demais módulos da arquitetura um acesso padronizado aos recursos disponibilizados pelos gateways no que diz respeito ao tratamento de sensores e/ou atuadores. Gerencia as comunicações com os Gateways Nativos através de API REST, por meio de protocolos que se baseiam em princípios REST como o UPnP e/ou CoAP. Por outro lado, o acesso aos serviços dos Gateways proprietários pelo Módulo de Comunicação é realizado através das APIs disponibilizados pelos seus fabricantes, que em muitos casos baseiam-se em protocolos típicos da IoT, como COAP e MQTT.

Gerenciador de Recursos

O Gerenciador de Recursos no EXEHDA-IoT foi concebido com o objetivo de administrar procedimentos, como a entrada e saída de operação de dispositivos, seja por decorrência de término de energia, perda de sinal de rádio ou pela inserção de novo dispositivo, bem como à reposição por avaria ou necessidade de adicionar outro ponto de monitoramento, provendo para isso um mecanismo de descoberta para os gateways.

4.2.3 Gerência e Descoberta de Dispositivos

Em sua arquitetura, o middleware EXEHDA necessitava de intervenção humana para que os dispositivos de sensoriamento e/ou atuação fossem adicionados ou removidos, independentemente de serem dispositivos já em uso, o que na perspectiva da elevada escalabilidade da IoT constitui um problema operacional significativo.

Esta situação levou a necessidade da concepção de mecanismos para o EXEHDA-IoT, integrados ao Servidor de Borda e gateways que possibilitam uma identificação automática dos dispositivos de sensoriamento e atuação previamente cadastrados, possibilitando uma melhor gerência dos mesmos.

Para tal, foi utilizada a estratégia de comunicação *plug and play* do protocolo UPnP, que se utiliza de princípios REST como forma de prover interoperabilidade entre dispositivos cliente e servidor. A escolha pelo protocolo UPnP foi devido aos seus recursos se adequarem a proposta do EXEHDA-IoT.

Considerando aspectos de compatibilidade e atualidade do desenvolvimento (existência de atualizações recentes), optou-se por utilizar para a implementação dos componentes UPnP a biblioteca chamada **Cling Core** desenvolvida pela organização de software livre sediada em Zurique chamada 4th Line GmbH (4THLINE, 2015).

Os componentes UPnP implementados encontram-se descritos a seguir.

Ponto de Controle

O ponto de controle encontra-se instanciado no Gerenciador de Recursos do Servidor de Borda e possui como tarefa descobrir os dispositivos UPnP (Gateways Nativo e Virtual) presentes na rede, armazenando as informações sobre os dispositivos e seus respectivos serviços descobertos em um diretório, para eventual controle destes.

O ponto de controle trata cada gateway como um dispositivo UPnP e cada sensor e/ou atuador desse gateway como um serviço UPnP, sendo que estes serviços podem ser de aquisição de dados por parte de sensores, bem como de alteração no estado de atuadores. Cada dispositivo possui uma identificação única, e seus respectivos dados de nome, tipo, modelo e fabricante. Todos estes dados são indexados em um arquivo XML (Documento de Descrição do Dispositivo) e lidos pelo Ponto de Controle a cada requisição.

A descoberta atua como um monitorador da rede, este monitorador espera por mensagens de inclusão e/ou remoção de gateways na rede a qual está conectado. Uma vez que o Ponto de Controle detecta um novo dispositivo, o mesmo verificará seu tipo a fim de estabelecer um filtro de dispositivos indesejáveis que possam possivelmente entrar na rede criando assim um mecanismo de proteção para o sistema.

Após a descoberta do dispositivo, o mesmo é adicionado a uma base de dispositivos (Diretório) reconhecidos pelo Servidor de Borda, sendo esta base utilizada posteriormente pelo Publicador. O mesmo acontece para dispositivos removidos. Ao ser removido, o dispositivo emite um *multicast* avisando que está sendo desconectado, assim o Ponto de Controle pode removê-lo da lista de dispositivos reconhecidos e avisar ao Servidor de Contexto que o mesmo encontra-se desativado, gerando um alerta de indisponibilidade para aplicações que estejam utilizando-o no momento, o que acarretará na parada de publicação de seus dados no Repositório de Contextos.

Gateway Como um Dispositivo UPnP

Uma instância do UPnP é executada, tanto no Gateway Nativo como Virtual, fazendo com que este seja identificado como um dispositivo UPnP. Cada dispositivo possui como dados de identificação:

- UID - Identificação Única;
- tipo de dispositivo (GatewayNativo ou GatewayVirtual);

- nome;
- nome amigável;
- detalhes do fabricante;
- descrição;
- versão.

Estas informações são inicializadas de um arquivo de configuração. Caso algum desses parâmetros seja modificado, o dispositivo na próxima conexão será identificado como um novo gateway. Os dispositivos Gateways Nativos podem ser implementados em plataformas de hardware embarcadas, como por exemplo uma Raspberry Pi, Arduino, ESP8266, etc, que possuam capacidade de portar o protocolo UPnP, possibilitando que sensores e/ou atuadores com capacidade restrita possam ser identificados automaticamente, constituindo-se como serviços UPnP disponibilizados pelo dispositivo Gateway o qual estão inseridos. O gateway funciona como uma ponte entre dispositivos de sensoriamento que não são capazes de portar o protocolo e os componente UPnP. Além das informações de sua identificação, cada gateway também carrega informações de seus serviços vinculados (sensores e/ou atuadores), disponibilizados por meio de arquivos XML (Documento de Descrição de Serviços).

Sensor/Atuador Como um Serviço UPnP

O sensor/atuador é a última etapa de abstração do EXEHDA-IoT, que constitui para o UPnP um serviço que está vinculado ao gateway (dispositivo).

Enquanto recurso, o gateway recebe requisições e habilita o respectivo serviço para efetuar alguma ação ou aquirir algum dado. Tal requisição é tratada por drivers, os quais são individualizados para cada sensor/atuador. O driver de sensor é responsável pela aquisição do valor referente a grandeza física medida pelo sensor, sendo usual cada tipo de sensor possuir uma forma específica de acesso para obtenção dos dados. O driver de atuador encapsula os procedimentos específicos de cada atuador, na maioria das vezes empregando bibliotecas e/ou softwares disponibilizados por fabricantes que modificam variáveis de estado com o intuito de alterar seu funcionamento. A utilização de drivers preserva a propagação de aspectos de implementação para as camadas superiores da arquitetura do Servidor de Borda.

Cada sensor/atuador descrito no UPnP como um serviço possui um arquivo de classe com suas especificações através de variáveis de estado. Este arquivo de classe é descrito e padronizado pela biblioteca Cling Core, garantindo as especificações do *UPnP Device Architecture* (UDA). Um exemplo deste arquivo de classe pode ser visto na Figura 18.

```

@UpnpService(
    serviceId = @UpnpServiceId("NodoTempVirtual"),
    serviceType = @UpnpServiceType(value = "NodoTemp", version = 1)
)
public class NodoTemp {
    @UpnpStateVariable(defaultValue = "0", sendEvents = false)
    private String nome = "NodoTempVirtual";
    @UpnpStateVariable(defaultValue = "0", sendEvents = false)
    private String descricao = "Sensor de temperatura";
    @UpnpStateVariable(defaultValue = "0", sendEvents = false)
    private String modelo = "dht11";
    @UpnpStateVariable(defaultValue = "0", sendEvents = false)
    private String precisao = "0.1";
    @UpnpStateVariable(defaultValue = "0", sendEvents = false)
    private String tipo = "temperatura";
    @UpnpStateVariable(defaultValue = "0", sendEvents = false)
    private Float temperatura = (float) 0;
    @UpnpStateVariable(defaultValue = "0", sendEvents = false)
    private Float dado = (float) 0;

    @UpnpAction(out = @UpnpOutputArgument(name = "ResultNome"))
    public String getNome() {
        return nome;
    }

    @UpnpAction(out = @UpnpOutputArgument(name = "ResultDescricao"))
    public String getDescricao() {
        return descricao;
    }

    @UpnpAction(out = @UpnpOutputArgument(name = "ResultModelo"))
    public String getModelo() {
        return modelo;
    }

    @UpnpAction(out = @UpnpOutputArgument(name = "ResultPrecisao"))
    public String getPrecisao() {
        return precisao;
    }

    @UpnpAction(out = @UpnpOutputArgument(name = "ResultTipo"))
    public String getTipo() {
        return tipo;
    }

    @UpnpAction(out = @UpnpOutputArgument(name = "ResultTemperatura"))
    public Float getTemperatura() {
        return temperatura;
    }

    @UpnpAction(out = @UpnpOutputArgument(name = "ResultDado"))
    public Float getDado() {
        Terminal t = new Terminal();

        this.dado = Float.parseFloat(t.execute("python nodoTemp.py"));

        return dado;
    }
}

```

Figura 18: Arquivo de Classe de um Sensor de Temperatura

4.2.4 Protocolos Suportados

Após revisão de literatura foram identificados dois protocolos como mais oportunos para uso em equipamentos de porte reduzido introduzidos pelo cenário da IoT: *Messaging Queue Telemetry Transport* (MQTT) e o *Constrained Application Protocol* (CoAP). Estes dois protocolos se destacam por possuírem características que minimizam o consumo computacional e energético, possuindo grande utilização no cenário da IoT.

A principal diferença entre CoAP e MQTT é que o CoAP roda sobre UDP que não é inerentemente confiável, porém realiza troca de mensagens que podem requisitar pedidos de confirmação de recebimento, constituindo assim um mecanismo de

confiabilidade, além de ser baseado em princípios arquiteturais REST, já o MQTT é executado sobre TCP e fornece três níveis de QoS, que representam a garantia de qualidade de entrega das mensagens.

Tanto o Servidor de Borda como o Gateway Nativo foram concebidos de forma a suportar estes dois tipos de protocolos. Porém, a escolha de utilização do protocolo CoAP para o Gateway Nativo levou em consideração o fato deste utilizar princípios REST, conforme a comunicação com os outros componentes do middleware. Outro fator foi uma avaliação realizada por meio da implementação dos protocolos no Gateway Nativo, sobre o qual foram realizados testes de desempenho.

Os testes de desempenho considerando a arquitetura do EXEHDA-IoT, onde foram avaliados os tempos médios dos pacotes enviados e o desvio padrão destes tempos, mostraram uma melhor performance do CoAP. Os resultados foram publicados em (JOÃO et al., 2016).

4.3 Discussão sobre os Trabalhos Relacionados

Uma primeira análise dos trabalhos relacionados foi realizada no Capítulo 3. Pode-se observar que nenhum dos trabalhos avaliados atendeu a todos os requisitos identificados como relevantes na concepção do EXEHDA-IoT. Registre-se que o único requisito endereçado de uma maneira ou outra por todos os trabalhos relacionados, foi o de interoperabilidade.

A estratégia adotada pela maioria dos trabalhos apresenta mecanismos interoperáveis para a troca de dados que consiste na utilização de API REST como forma de disponibilizar os seus recursos. Da mesma forma, o EXEHDA-IoT adotou a solução baseada em API REST, o qual propicia que todos os recursos, como o acesso aos dados sensorizados, comandos de atuação, publicações, entre outros, sejam acessados através de URIs.

Todavia a gerência e acesso aos recursos são disponibilizados aos usuários responsáveis pelas aplicações através de uma interface de alto nível, o que não ocorre em S³OIA e LinkSmart.

Dentre os trabalhos analisados, o S³OIA destaca-se por ser o único a apresentar uma estratégia de descoberta de dispositivos automática, *plug and play* e de autoconfiguração para o gerenciamento de sensores e/ou atuadores. Os outros atendem ao critério de descoberta e gerenciamento de dispositivos de forma parcial, supondo que os dispositivos e seus recursos requeridos estejam disponíveis por meio de um cadastro prévio. No EXEHDA-IoT, por outro lado, se utiliza o UPnP como estratégia de autoconfiguração e descoberta, tendo o gateway como elemento básico do mecanismo de descoberta e a partir dele, os sensores e atuadores são facilmente descobertos e gerenciados. Além disso, a abordagem utilizada considera todo o potencial

de protocolos de uso consolidado.

A ciência de contexto é contemplada por todos os trabalhos relacionados, porém o suporte oferecido, como no caso do EcoDif e Carriots mostra-se limitado. Já o EXEHDA-IoT fornece uma organização distribuída para coleta e processamento de contexto através de regras, como também para atuação em ambientes com características heterogêneas e escaláveis. Além disso, por ter sido desenvolvido a partir de um middleware para UbiComp, o EXEHDA, também provê suporte a outros serviços ubíquos disponibilizados pelo middleware, como adaptação, comunicação desacoplada, migração de código, etc.

Por fim, outra abordagem adotada pelo EXEHDA-IoT, que trouxe benefícios principalmente no que tange à otimização do tráfego de rede e ao consumo de energia, é a proposição do uso de *triggers* para gerenciar o fluxo de dados transmitidos. A abordagem de *triggers* foi concebida para permitir a personalização da coleta dos dados através de eventos considerando as características de variabilidade física de cada grandeza monitorada, o que proporciona minimizar o fluxo de dados entre os gateways e o Servidor de Borda.

Além disso, a abordagem distribuída do processamento do contexto entre os Servidores de Borda e Contexto proposta pelo EXEHDA-IoT também contribui com a minimização do fluxo de dados contextuais em relação as abordagens centralizadas, empregadas pelos trabalhos relacionados.

4.4 Considerações sobre o Capítulo

Neste capítulo foram apresentados os esforços de concepção e modelagem do EXEHDA-IoT, os quais refletiram as principais decisões de projeto com o intuito de qualificar o middleware EXEHDA ao cenário da IoT. Por fim, foi realizada uma comparação dos trabalhos relacionados apresentados no Capítulo 3 em relação ao EXEHDA-IoT.

5 EXEHDA-IOT: ESTUDO DE CASO

Neste capítulo estão resumidos os principais aspectos da prototipação do EXEHDA-IoT, bem como o estudo de caso empregado na avaliação de suas funcionalidades. O estudo de caso contemplou tarefas referentes ao sensoriamento e a coleta de informações contextuais, além dos diferentes procedimentos de gerência necessários para a operação do EXEHDA-IoT.

5.1 Estudo de Caso: Projeto plenUS

Para avaliação das funcionalidades do EXEHDA-IoT esta dissertação deu continuidade ao projeto plenUS (*plentiful Ubiquitous Systems*), que vem sendo desenvolvido por um consórcio de pesquisa envolvendo o LUPS e a Embrapa Clima Temperado. O projeto plenUS tem como premissa o emprego do middleware EXEHDA com o objetivo de disponibilizar Sistemas Ubíquos, que explorem relações pró-ativas entre usuários, softwares e equipamentos, promovendo assim soluções computacionais que contribuam de forma sinérgica no atendimento das atividades de pesquisa da Embrapa Clima Temperado.

Nesta dissertação foram caracterizados três diferentes tipos de atores, que se diferenciam de acordo com o nível de acesso ao middleware EXEHDA, são eles:

- desenvolvedor: possui acesso total as funcionalidades do middleware, podendo realizar mudanças em sua arquitetura de software, seja pela reprogramação de módulos já existentes no middleware ou seja pela criação de novos módulos;
- administrador: possui atribuições de gerência do middleware, configurando o mesmo para atender as demandas de um segmento usuário em particular;
- usuário final: por meio de aplicações utiliza os serviços providos pelo middleware.

Neste sentido os mecanismos concebidos pelo EXEHDA-IoT tem como função facilitar os aspectos de gerência do administrador, além de fornecer ao usuário final aplicações para utilização dos serviços do middleware, o mais intuitivas possível.

Outro objetivo do EXEHDA-IoT é o de atender as demandas do projeto em relação ao provimento de serviços para ciência de contexto, permitindo um registro histórico dos estados contextuais em que se encontram os equipamentos dos seguintes laboratórios: (i) Laboratório de Qualidade do Leite (Lableite), (ii) Laboratório de Reprodução Animal e (iii) Laboratório de Análise de Sementes Oficial (LASO), como também outras necessidades de sensoriamento em ambientes administrativos como o do Núcleo de Tecnologia da Informação (NTI), e de uma Sala de Testes do plenUS empregada para avaliação do EXEHDA-IoT.

Os ambientes priorizados nesta dissertação, para recebimento de infraestrutura para ciência do contexto em que se encontram seus equipamentos, podem ser visualizados na Figura 19. Nesta Figura além da indicação dos laboratórios e salas administrativas envolvidos, também podem ser vistos os tipos de grandezas mensuradas, bem como a disposição dos gateways, servidores de borda e servidor de contexto.

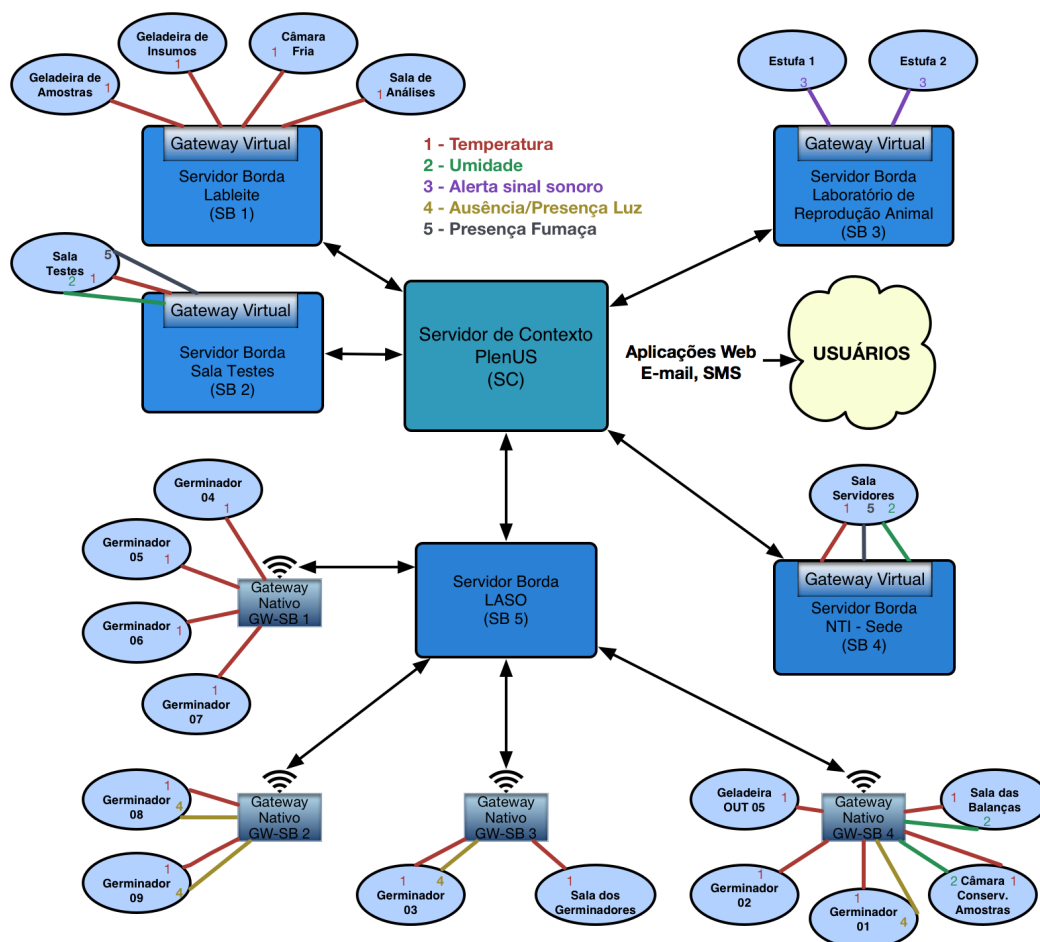


Figura 19: Estudo de Caso - Projeto plenUS

Considerando a estrutura física do estudo de caso, um Servidor de Contexto, cinco Servidores de Borda e quatro Gateways foram instalados para atender as demandas da Embrapa como um todo (vide Figura 19).

O Servidor de Contexto e o Servidor de Borda 4 (SB 4) foram posicionados no NTI, localizado na unidade Sede da Embrapa Clima Temperado, enquanto os outros Servidores de Borda foram posicionados na unidade Estação Experimental Terras Baixas (ETB).

Os Servidores de Borda foram distribuídos cada qual no prédio específico aonde a coleta de informações contextuais deve acontecer, já quanto a escolha da localização do Servidor de Contexto foi levado em consideração fatores tais como, climatização adequada, sistema de fornecimento de energia ininterrupto (no-break e gerador), bem como acesso facilitado para manutenção do equipamento.

Enquanto aos gateways previstos pelo EXEHDA-IoT e utilizados para o estudo de caso, estes são de duas naturezas: **Gateways Virtuais**, internos ao Servidor de Borda (SB), e **Gateways Nativos** com conexão wi-fi com o SB. Os Gateways Nativos foram utilizados no caso onde as demandas de sensoriamento introduziram a sua necessidade, seja pela disposição física dos equipamentos e/ou salas ou pelo elevado número de sensores, acarretando que estes fossem distribuídos em diferentes gateways.

A operacionalização do cenário propiciado pelo estudo de caso é realizada através de regras distribuídas do tipo ECA (Evento-Condição-Ação) entre os Servidores de Borda e Contexto, tendo como entrada as informações contextuais coletadas e como condição as faixas operacionais para cada contexto específico. Caso a informação contextual coletada não esteja dentro da faixa operacional, uma ação será executada.

A interação do usuário com o cenário é disponibilizada por meio de uma aplicação desenvolvida para navegadores web, que contempla tarefas de gerenciamento dos dispositivos de sensoriamento, com suporte a descoberta automática, o que agregou funcionalidades de cadastro e/ou remoção automática dos dispositivos gateways, bem como de seus sensores quando de sua disponibilidade. Além disso, a aplicação também permite a visualização do registro histórico das informações contextuais, tanto de forma gráfica como textual.

Outra característica avaliada foi a de responsividade da aplicação Web desenvolvida, o que veio agregar maior funcionalidade ao seu uso, pois possibilitou que as configurações e testes dos dispositivos de sensoriamento pudessem ser realizadas no local de instalação através de *smartphones*, devido ao seu *layout* se adaptar ao dispositivo computacional utilizado.

Uma caracterização de três dos ambientes contemplados por esta dissertação é realizada a seguir.

5.1.1 Laboratório de Qualidade do Leite

Com forte presença na cadeia produtiva do leite, a Estação Experimental Terras Baixas (ETB) da Embrapa Clima Temperado abriga o SISPEL (Sistema de Pesquisa e Desenvolvimento em Pecuária Leiteira). Este é constituído de instância física e opera-

cional que inclui equipe de pesquisa e parceiros, rebanho experimental da raça Jersey, instalações de manejo e laboratórios. Destaque para o Laboratório de Qualidade do Leite (Lableite), credenciado pelo MAPA (Ministério da Agricultura, Pecuária e Abastecimento), e que faz parte da Rede Brasileira de Laboratórios de Qualidade do Leite (RBQL).

O Lableite atua na pesquisa e prestação de serviços, atendendo a indústrias e cooperativas de laticínios, universidades e órgãos de pesquisa em leite, além de produtores particulares. Possui como objetivo realizar análises para avaliar a qualidade do leite, visando adequação aos padrões previstos na Instrução Normativa 51 (IN 51). A regulamentação federal exige que, na avaliação da matéria prima, pelo menos duas amostras de leite de cada produtor passem por avaliações a cada mês (LABLEITE, 2015).

Neste laboratório são avaliados pelo EXEHDA-IoT, em regime de 24 horas, 7 dias por semana o estado contextual dos seguintes equipamentos.

- Geladeira de Amostras

- Objetivo: este equipamento tem por finalidade manter as amostras oriundas dos produtores rurais em condição adequada de temperatura até que as mesmas entrem em processo de análise.
- Sensor Utilizado: sensor de temperatura 1-Wire DS18B20 encapsulado para emprego submerso em líquido (água).
- Regime Operacional: de 0°C à 7°C.
- Ação fora do regime operacional: envio de alerta à emails cadastrados e acionamento de alerta luminoso (vide Anexo A).

- Geladeira de Insumos

- Objetivo: este equipamento tem a finalidade de manter em condições adequadas de temperatura os insumos necessários as diferentes etapas do processo de análise da qualidade do leite.
- Sensor Utilizado: sensor de temperatura 1-Wire DS18B20 encapsulado para uso não submerso (ambientes úmidos).
- Regime Operacional: de 0°C à 4°C.
- Ação fora do regime operacional: envio de alerta à emails cadastrados e acionamento de alerta luminoso.

- Câmara Fria

- Objetivo: este equipamento tem por finalidade armazenar insumos no seu estado bruto, bem como outros produtos necessários a operação do Lableite.
- Sensor Utilizado: sensor de temperatura 1-Wire DS18B20 encapsulado para uso não submerso (ambientes úmidos);
- Regime Operacional: de 0°C à 7°C.
- Ação fora do regime operacional: envio de alerta à emails cadastrados e acionamento de alerta luminoso.

- Sala de Análises

- Objetivo: é neste ambiente que os equipamentos que realizam a análise da qualidade do leite operam. Com o intuito de minimizar a proliferação de fungos e/ou bactérias neste ambiente a temperatura desta sala não deve ultrapassar determinado limite.
- Sensor Utilizado: sensor de temperatura 1-Wire DS18B20 não encapsulado.
- Regime Operacional: de 15°C à 25°C.
- Ação fora do regime operacional: envio de alerta à emails cadastrados e acionamento de alerta luminoso.

5.1.2 Laboratório de Reprodução Animal

O Laboratório de Reprodução Animal fica localizado na unidade ETB da Embrapa Clima Temperado e atua no desenvolvimento e adaptação das tecnologias de reprodução assistida para os sistemas de produção animal. As ações de pesquisa e desenvolvimento são desenvolvidas em parceria com a Faculdade de Medicina Veterinária da UFPel e Emater RS.

Dentre as atividades desenvolvidas destacam-se: (i) a produção *in vivo* e *in vitro* de embriões bovinos para maximizar a eficiência reprodutiva dos rebanhos leiteiros, (ii) identificação do sexo dos embriões (sexagem) através da reação de polimerase em cadeia (PCR) e (iii) criopreservação de gametas e embriões. São realizadas também atividades com o foco acadêmico, de orientação a alunos de pós-graduação, estágio curricular e de vivência, bem como diferentes cursos em reprodução animal (EMBRAPA, 2015).

Existem neste laboratório dois equipamentos atendidos pelo EXEHDA-IoT, tratam-se de estufas de CO₂ para cultivo celular dos seguintes modelos.

- **Estufa 1** - Forma Series II Water Jacket CO₂ Incubator - Model 3100 Series.

- **Estufa 2** - Microprocessor Controlled Automatic Water Jacketed CO2 Incubators
- Series 6101-1 and 6301-1.

Para a satisfação dos experimentos que são realizados nestes equipamentos, estes possuem um sistema de monitoramento dos níveis de CO₂, O₂, umidade e temperatura, responsável por verificar se os níveis das grandezas monitoradas encontram-se em uma faixa preestabelecida e acionar alarme sonoro quando os níveis não estiverem de acordo com o estipulado. Porém este tipo de alarme exige a presença física do laboratorialista no ambiente para a sua detecção, o que pode ser um problema se este for acionado aos finais de semana, por exemplo, quando não há ninguém.

No estudo de uma alternativa de solução para a situação em que não há um laboratorialista presente para escutar o alarme, verificar qual grandeza está fora da faixa e tomar medidas que venham normalizar o nível desta grandeza, pode-se observar que os dois modelos de estufa forneciam uma saída denominada Alarme Remoto com dois tipos de pares de contatos indicando de forma diferente o estado do alarme, um normalmente fechado e um normalmente aberto.

Com posse destas informações foi possível prototipar e ligar um sensor aos contatos do par normalmente fechado. Este par de contatos indica no seu estado normal (contatos fechados) que não há alarme e quando estes contatos abrem que o alarme foi acionado. O sensor prototipado capta o estado do alarme (dado de contexto), repassando esta informação, quando houver alteração no seu estado, ao Servidor de Borda do laboratório. O Servidor de Borda publica esta informação contextual no Servidor de Contexto, que a armazena.

A partir destas informações armazenadas, é possibilitado que uma regra vinculada ao sensor e/ou contexto seja executada quando da mudança do estado contextual de um dos equipamentos, notificando por email e/ou SMS aos responsáveis do laboratório a ativação e desativação do alarme.

Portanto, para o Laboratório de Reprodução Animal são avaliados pelo EXEHDA-IoT, em regime de 24 horas, 7 dias por semana a mudança de estado contextual de duas estufas, cujas características são apresentadas a seguir:

- Estufa 1 e Estufa 2
 - Sensor Utilizado: sensor de chaves endereçáveis 1-Wire DS2413.
 - Estado Operacional:
 - * 0 = alarme desativado;
 - * 1 = alarme ativado.
 - Ação na mudança do estado operacional : envio de alerta à emails cadastrados notificando mudança no estado do alarme: 0 para 1 - alarme foi ativado; 1 para 0 - alarme cessou.

5.1.3 Laboratório de Análise de Sementes Oficial

O LASO (Laboratório de Análise de Sementes Oficial) da Embrapa Clima Temperado - ETB foi criado em dezembro de 1959, iniciando suas atividades em janeiro de 1960, e vem trabalhando até a presente data de maneira ininterrupta. É o mais antigo laboratório de análise de sementes do estado do Rio Grande do Sul e o segundo mais antigo do Brasil.

O laboratório possui como atividades a realização de análise e pesquisa das mais diversas espécies de sementes, prestando serviços a produtores de sementes do país, como também aulas práticas para estudantes e treinamento para analistas de sementes de instituições oficiais e privadas.

A principal finalidade em realizar análise de sementes é a de determinar a qualidade de um lote de sementes e, conseqüentemente, o seu valor para a semeadura. Seus resultados são utilizados para a emissão de certificado, que acompanha a embalagem de sementes para a fiscalização do comércio e a normatização da produção. Essas são as bases para o beneficiamento, a comercialização, o armazenamento e a distribuição das sementes. A análise é, ainda, utilizada em trabalhos de pesquisa e na identificação de problemas de qualidade e suas causas. Assim, para a obtenção de sementes com um nível de qualidade proposto, é importante manter a produção sob controle e, dessa forma, a análise se constitui em instrumento imprescindível (NEWS, 2015).

Para que os objetivos das análises sejam atingidos é necessário que os laboratórios possuam equipamentos adequados e sigam métodos e procedimentos uniformes. O LASO em 2007, assumiu o compromisso de implementação do Sistema de Qualidade, respaldado nos requisitos da norma ABNT NBR ISO/IEC 17025:2005, desenvolvendo trabalhos que assegurem a qualidade, com base nas Regras para Análise de Sementes (RAS) e demais padrões e legislações determinadas pelo Ministério de Agricultura, Pecuária e Abastecimento (MAPA).

Com a implementação do sistema de qualidade no LASO aumentaram as exigências nos controles dos equipamentos onde são realizadas as análises, visando precisão e confiabilidade nos resultados. Com isso, registros diários de leituras de grandezas como temperatura dos germinadores, câmaras e salas são necessários. Muitos laboratórios ainda utilizam planilhas em papel que são preenchidas manualmente várias vezes ao dia e para realizarem esta tarefa, funcionários são desviados de suas funções principais. Alguns laboratórios possuem *dataloggers*, no entanto, embora os dados coletados possam ser armazenados, não há atuação automatizada em decorrência desses dados. Devido a isso, muitas das análises são perdidas por atraso na detecção de problemas nos germinadores e/ou outro equipamento.

A demanda de atendimento a norma de qualidade pelo LASO, o qual acarreta a necessidade de manter registro histórico dos dados coletados, bem como uma

atuação automatizada quando necessário constitui um aspecto motivacional para implementação do EXEHDA-IoT neste laboratório.

O ambiente do LASO com suas demandas ubíquas possui o maior número de sensores, bem como diferentes grandezas monitoradas dentre os ambientes contemplados pelo estudo de caso para avaliação do EXEHDA-IoT. Este fato tornou real a necessidade dos Gateways Nativos ou GW-SB, concebidos nesta dissertação. Eles organizam o sensoriamento ao setorizar pontos de coleta em ambientes com características heterogêneas e escaláveis, típicos da IoT. Isto leva a uma distribuição entre os gateways das atribuições quanto ao tratamento dos protocolos físicos e normalização dos dados, que antes ficava a cargo apenas do Servidor de Borda. Outra característica quanto aos Gateways Nativos é que estes possuem capacidade de comunicação sem fio com o Servidor de Borda, o que diminui a quantidade de fios conectados, minimizando a incidência de ruído.

Na Tabela 7 são apresentadas algumas especificações referentes a coleta das informações contextuais dos equipamentos e salas avaliadas no LASO.

Tabela 7: Coleta de Informações Contextuais no LASO

Equipamento/Sala	GW-SB	Porta GW-SB	Grandeza	Sensor	Faixa de Operação
Sala das Balanças	3	1	Temperatura	DS2438	10°C à 30°C
	3	1	Umidade	DS2438 + HIH-4000	50% à 98%
Câmara de Conserv. Amostras	3	2	Temperatura	DS2438	10°C à 25°C
	3	2	Umidade	DS2438 + HIH-4000	50% à 98%
Germinador 01	3	3	Temperatura	DS18B20 Waterproof	18°C à 22°C
	3	4	Presença de luz	DS2438 + LDR	00:00:00 às 11:59:59 - sem luz 12:00:00 às 23:59:59 - com luz
Germinador 02	3	5	Temperatura	DS18B20 Waterproof	08°C à 12°C
Geladeira OUT 05	3	6	Temperatura	DS18B20 Waterproof	03°C à 07°C
Germinador 03	4	1	Temperatura	DS18B20 Waterproof	18°C à 22°C
	4	2	Presença de luz	DS2438 + LDR	00:00:00 às 11:59:59 - sem luz 12:00:00 às 23:59:59 - com luz
Sala dos Germinadores	4	3	Temperatura	DS18B20	20°C à 27°C
Germinador 04	1	1	Temperatura	DS18B20 Waterproof	23°C à 27°C
Germinador 05	1	2	Temperatura	DS18B20 Waterproof	23°C à 27°C
Germinador 06	1	3	Temperatura	DS18B20 Waterproof	23°C à 27°C
Germinador 07	1	4	Temperatura	DS18B20 Waterproof	18°C à 27°C
Germinador 08	2	1	Temperatura	DS18B20 Waterproof	12:30:00 às 20:00:00 - 28°C à 32°C 20:00:01 às 12:29:59 - 18°C à 22°C
	2	2	Presença de luz	DS2438 + LDR	12:00:00 às 20:00:00 - com luz 20:00:01 às 11:59:59 - sem luz
Germinador 09	2	3	Temperatura	DS18B20 Waterproof	12:30:00 às 20:00:00 - 28°C à 32°C 20:00:01 às 12:29:59 - 18°C à 22°C
	2	4	Presença de luz	DS2438 + LDR	12:00:00 às 20:00:00 - com luz 20:00:01 às 11:59:59 - sem luz

5.1.4 Infraestrutura de Hardware e Software Desenvolvidas

Para prototipação dos mecanismos concebidos, foram priorizadas tecnologias escaláveis e robustas. Nesse sentido, o código dos servidores de contexto e de borda foram escritos na linguagem Python. O repositório de contextos emprega o gerenciador PostgreSQL para implementação das bases de dados. O estilo arquitetural REST foi empregado para interoperação entre servidor de contexto, servidores de borda e gateways. Para tal, foram utilizadas duas estratégias de comunicação que se baseiam neste estilo arquitetural, uma para fins específicos de identificação automática dos dispositivos gateways e seus serviços por meio do protocolo UPnP e outra para comunicação entre dispositivos restritos típicos da IoT com a utilização do protocolo CoAP.

O Servidor de Contexto foi instalado nas dependências do Núcleo de Tecnologia da Informação da Embrapa com o intuito de maximizar a sua disponibilidade, além do fato de poder contar com *no-break* e um sistema gerador de energia elétrica. O hardware utilizado foi um desktop com processador Intel Pentium E2160-1.8GHz de dois núcleos, com 3Gb de memória RAM e com o Sistema Operacional Ubuntu Server. A escolha da distribuição Linux Ubuntu foi decorrente da ampla utilização da mesma pelo grupo de pesquisa em seus equipamentos servidores, o que simplifica o seu suporte.

A prototipação dos Servidores de Borda foi realizada sobre o Sistema Operacional Raspbian, sendo usado como hardware a Raspberry Pi Modelo B+ (vide Figura 20a). A Raspberry Pi (RASPBerry, 2014)(BROCK; BRUCE; CAMERON, 2013) é um computador desenvolvido pelo Laboratório de Computação da Universidade de Cambridge que tem como principais características o tamanho reduzido, baixo custo, baixo consumo energético, com comunidade de usuários bastante ativa e que baseia-se nos princípios de software livre. A utilização da Raspberry Pi como Servidor de Borda para esta proposta se mostrou interessante, pois esta adequa-se perfeitamente ao cenário da IoT, porém possui boa capacidade de processamento, o que possibilita que regras sejam processadas com relativa tranquilidade, além da gerência e comunicação de diferentes protocolos e dispositivos de sensoriamento.

Já os gateways representam dispositivos IoT com poder computacional limitado o que introduz desafios quando da sua concepção. Para a escolha do hardware utilizado na concepção do Gateway Nativo do EXEHDA-IoT (vide Figura 20b) foram considerados aspectos como: baixo consumo energético, possibilidade de operação em regime sem fio (wi-fi) e emprego de plataforma de software disseminada. O critério de empregar uma plataforma de software disseminada tem como motivação central poder suportar o maior número possível de sensores e atuadores. Após revisão de literatura a escolha recaiu sobre o processador ESP8266-12 (KURNIAWAN, 2015).

O processador ESP8266-12 tem comunicação wi-fi integrada, podendo ser pro-

gramado através da plataforma de software do Arduino¹, bem como em Linguagem Lua através de um firmware denominado nodeMCU². Para a prototipação do Gateway Nativo, foi utilizado o kit de desenvolvimento do NodeMCU³ que possui integrado o processador ESP8266-12, bem como entrada mini USB e acesso facilitado ao pinos GPIOs do processador (vide Figura 20c).

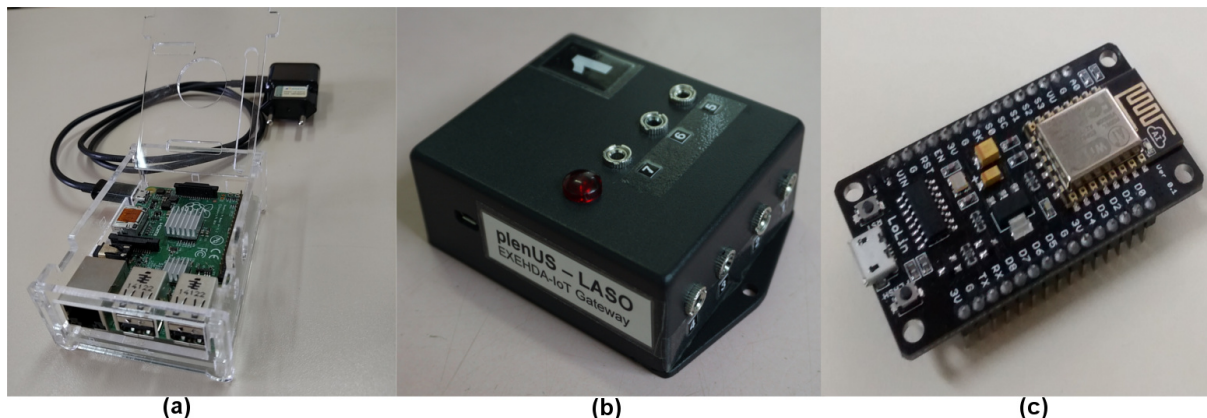


Figura 20: Dispositivos Utilizados: (a) Raspberry PI; (b) Gateway Nativo (GW-SB) ; (c) ESP8266-12

Para as demandas de sensoriamento do estudo de caso foi utilizado a tecnologia 1-Wire (MAXIM, 2014) como protocolo físico de acesso aos sensores. A tecnologia 1-Wire se destaca por ser um protocolo para transmissão dos dados em rede, baseado em dispositivos eletrônicos endereçáveis, robusto, e que não exige cabeamento especializado para sua operação. A leitura dos sensores é efetivada por meio de *drivers*, que são softwares específicos por tipo de sensor, responsável por traduzir o dado captado por meio do protocolo 1-Wire em uma informação contextual compatível com a grandeza mensurada.

Os *drivers* para a captação de dados dos sensores são acionados mediante requisição de leitura pelo Servidor de Borda, que os armazena temporariamente em virtude de uma eventual falha de comunicação com o Servidor de Contexto. A publicação dos dados contextuais captados no Servidor de Contexto acontece de acordo com o tempo de publicação estipulado no Configurador do Servidor de Borda.

No decorrer desta dissertação, para atender as funcionalidades previstas para este estudo de caso, foi necessária a criação de alguns dispositivos de sensoriamento, bem como hardwares específicos e o respectivo firmware para operarem na rede de sensores 1-Wire. Dentre os dispositivos desenvolvidos destacam-se: sensor de temperatura com diferentes tipos de encapsulamento, sensor de umidade, sensor de presença de luz, sensor de detecção de alarme sonoro, atuador de alarme visual, hardware embar-

¹ <https://www.arduino.cc/en/Reference/>

² <https://github.com/nodemcu/nodemcu-firmware/wiki/>

³ <http://www.nodemcu.com/>

cado para o Gateway Nativo, interface serial 1-Wire e HUB para conexão de sensores 1-wire. Suas especificações encontram-se disponíveis no Anexo A.

5.1.5 Funcionalidades Disponibilizadas

Atendendo a premissa que as informações devem estar acessíveis independente do dispositivo computacional utilizado, as tecnologias utilizadas pelas aplicações no âmbito da IoT devem possuir recursos de maneira a se adaptarem as mudanças ocorridas no ambiente de execução.

Considerando isto, no desenvolvimento da aplicação web foi empregado tecnologias responsivas no seu *layout*. O *layout* é dito responsivo, quando ele se adapta ao tamanho do dispositivo, sem prejudicar suas funcionalidades. Portanto a aplicação pode ser acessada da mesma forma, tanto por *desktops* e *notebooks* como por *tablets* e *smartphones*.

A aplicação desenvolvida e utilizada pelo estudo de caso contempla dois modos de operação, uma para a gerência de recursos de sensoriamento e/ou atuação, com suporte ao acesso a diferentes níveis de gerenciamento de acordo com o perfil do usuário, e outra para visualização de dados históricos das informações contextuais.

Aplicação de Gerenciamento

A aplicação de gerência de recursos de sensoriamento e/ou atuação desenvolvida para o Servidor de Contexto é responsável por realizar as quatro operações básicas utilizadas em banco de dados relacionais denominada CRUD (*Create, Read, Update e Delete*), ou seja está capacitada a cadastrar, ler, atualizar e remover dados do Repositório de Informações Contextuais.

A aplicação foi desenvolvida utilizando princípios REST, possibilitando que qualquer operação possa ser realizada por uma URI específica. Ao acessar cada URI, parâmetros devem ser passados viabilizando assim que diversas funções requisitadas pelo Servidor de Contexto possam ser efetuadas. Utilizando-se da aplicação o Servidor de Contexto é capaz de receber dos Servidores de Borda, pertencentes a mesma célula de execução, informações de novos dispositivos alinhados a estes e até mesmo atualização de informações a respeito dos mesmos.

O cadastro e/ou remoção dos dispositivos de sensoriamento e/ou atuação pela aplicação pode ser realizado tanto de forma automática, no caso de dispositivos que suportem o protocolo UPnP, ou de forma manual. Vale lembrar que o cadastro automático só será finalizado mediante autenticação pelo usuário, evitando assim o cadastro de dispositivos não reconhecidos.

O acesso a aplicação de gerenciamento somente acontece por meio de login, sendo que cada usuário poderá ter um nível de acesso diferenciado aos menus de acordo com a sua permissão.

Além de viabilizar operações no banco de dados, a aplicação é responsável por criar uma interface intuitiva para que o usuário possa gerenciar seus recursos de maneira simples. Levando em consideração estes aspectos, algumas funcionalidades foram criadas, as quais estão dispostas em um menu (vide Figura 21) e encontram-se descritas a seguir.



Figura 21: Menu da Aplicação de Gerenciamento

- **Servidor de Contexto:** responsável por editar informações descritivas a respeito do Servidor de Contexto, como nome e localização.
- **Contexto de Interesse:** responsável por promover todas as operações de gerência sobre os contextos de interesse. Estes Contextos de Interesse relacionam um nome a um determinado grupo de sensores que são de interesse de uma determinada aplicação.
- **Regras:** permite que sejam carregadas regras em arquivos python, vinculadas a sensores e/ou contextos de interesse para serem utilizadas no processamento das informações contextuais.
- **Servidores de Borda:** tem como função efetuar criação, alterações e remoções de cadastros dos Servidores de Borda (vide Figura 22).
- **Gateway:** permite a visualização dos gateways cadastrados automaticamente, bem como a criação, alteração e remoção de cadastros de forma manual. Outra funcionalidade prevista para este recurso é o de ativação e desativação dos gateways, quando da necessidade de interrupção ou retomada da coleta de informações contextuais destes dispositivos por parte da aplicação (vide Figura 23).



Servidores de Borda Cadastrados - 5 Registro(s)

Dados alterados com sucesso!

Por página:

NOME	DESCRIÇÃO	LATITUDE	LONGITUDE	URL	
SB 1: Laboratório de Qualidade do Leite	Servidor Borda 1 - Laboratório Qualidade do Leite - ETB			http://localhost	
SB 2: Sala Testes	Servidor Borda 2 - Sala Testes - ETB			http://localhost	
SB 3: Laboratório Reprodução Animal	Servidor de Borda 3 - Laboratório de Reprodução Animal - ETB			http://localhost	
SB 4: Núcleo Tecnologia da Informação	Servidor Borda 4 - Sala de Servidores do Núcleo de Tecnologia da Informação - Sede			http://localhost	
SB 5: Laboratório Análise de Sementes	Servidor Borda 5 - Laboratório de Análise de Sementes - ETB			http://localhost	

© EXEHDA

Figura 22: Tela de Gerência dos Servidores de Borda



Gateways Cadastrados - 8 Registro(s)

Dados alterados com sucesso!

Por página:

NOME	MODELO	FABRICANTE	SERVIDOR DE BORDA	STATUS	
Gateway Lableite	Virtual	EXEHDA-IoT	SB 1: Laboratório de Qualidade do Leite	Ativo	
Gateway LabRepAnimal	Virtual	EXEHDA-IoT	SB 3: Laboratório Reprodução Animal	Ativo	
Gateway NTI	Virtual	EXEHDA-IoT	SB 4: Núcleo Tecnologia da Informação	Ativo	
Gateway Sala Testes	Virtual	EXEHDA-IoT	SB 2: Sala Testes	Inativo	
GW01 LASO - Sala Germinadores	Nativo	EXEHDA-IoT	SB 5: Laboratório Análise de Sementes	Ativo	
GW02 LASO - Sala Germinadores	Nativo	EXEHDA-IoT	SB 5: Laboratório Análise de Sementes	Ativo	
GW03 LASO - Sala Plantio	Nativo	EXEHDA-IoT	SB 5: Laboratório Análise de Sementes	Ativo	
GW04 LASO - Sala Germinadores	Nativo	EXEHDA-IoT	SB 5: Laboratório Análise de Sementes	Ativo	

© EXEHDA

Figura 23: Tela de Gerência dos Gateways

- **Ambiente:** por meio deste recurso é possível efetuar operações de gerência relacionadas ao local onde cada sensor está posicionado fisicamente. Como exemplo, o ambiente pode ser o equipamento ou sala que o sensor está inserido. Este recurso serve como um parâmetro para a função de Agendamento.
- **Sensores:** responsável por configurar características dos sensores, como as faixas operacionais previstas para um determinado horário, que caracterizam as

condições para as regras serem executadas, bem como a vinculação de cada sensor ao gateway e servidor de borda o qual está atrelado (vide Figura 24).

Sensores Cadastrados - 29 Registro(s)

Por página: 10 1 2 3 >

Filtro por Servidor de Borda: Todos Filtrar

NOME	DESCRIÇÃO	MODELO	HORA LIMITE	LIMITES DIURNOS	LIMITES NOTURNOS	AMBIENTE	GATEWAY	SERVIDOR DE BORDA	
Alerta Estufa1 LabRepAnimal	Alerta da Estufa1 do LabRepAnimal	DS2413 - Liga/Desliga	00:00:00/23:59:59	0/1	/	Estufa 1 - LabRepAnimal	Gateway LabRepAnimal	SB 3: Laboratório Reprodução Animal	
Alerta Estufa 2 LabRepAnimal	Alerta da Estufa2 do LabRepAnimal	DS2413 - Liga/Desliga	00:00:00/23:59:59	0/1	/	Estufa 2 - LabRepAnimal	Gateway LabRepAnimal	SB 3: Laboratório Reprodução Animal	
Detector Fumaça Sala Servers NTI	Alerta do Detector de Fumaça da Sala dos Servidores do NTI	DS2413 - Liga/Desliga	00:00:00/23:59:59	0/1	/	Sala Servidores - NTI	Gateway NTI	SB 4: Núcleo Tecnologia da Informação	
Presença/Ausência Luz GER01 LASO	% Luz no Germinador 1 do LASO	DS2438 / LDR	12:00:00/23:59:59	30/100	0/29.99	Germinador 1 - LASO	GW03 LASO - Sala Plantio	SB 5: Laboratório Análise de Sementes	
Presença/Ausência Luz GER03 LASO	% Luz no Germinador 3 do LASO	DS2438 / LDR	12:00:00/23:59:59	30/100	0/29.99	Germinador 3 - LASO	GW04 LASO - Sala Germinadores	SB 5: Laboratório Análise de Sementes	
Presença/Ausência Luz GER08 LASO	% Luz no Germinador 8 do LASO	DS2438 / LDR	12:00:00/20:00:00	30/100	0/29.99	Germinador 8 - LASO	GW02 LASO - Sala Germinadores	SB 5: Laboratório Análise de Sementes	
Presença/Ausência Luz GER09 LASO	% Luz no Germinador 9 do LASO	DS2438 / LDR	12:00:00/20:00:00	30/100	0/29.99	Germinador 9 - LASO	GW02 LASO - Sala Germinadores	SB 5: Laboratório Análise de Sementes	
Sensor Teste	Sensor utilizado para testes	DS2438 / HIH4000	00:00:00/23:59:59	50/98	/	Sala Testes - ETB	Gateway Sala Testes	SB 2: Sala Testes	
Temperatura Câmara Conservação	Temperatura da Câmara de Conservação de Amostras do LASO	DS2438/interno	00:00:00/23:59:59	10/25	/	Câmara Conservação Amostras -	GW03 LASO - Sala Plantio	SB 5: Laboratório Análise de Sementes	

© EXEHDA

Figura 24: Tela de Gerência dos Sensores

- **Tipos de Sensores:** com este recurso é possível manipular os tipos de sensores, envolvendo desde o nome até a unidade de referência (vide Figura 25).

EXEHDA-IoT Visualização Gerenciamento

Patricia Davet SAIR

Embrapa
Clima Temperado

plenus

Início Servidor de Contexto Servidores de Borda Ambientes Sensores Agendamento Administração

Tipos de Sensores Cadastrados - 4 Registro(s)

Por página: 4

NOME	DESCRIÇÃO	UNIDADE	
Estado de Evento	Estado de Evento - Ligado ou Desligado	L/D	
Presença/Ausência Luz	Presença ou Ausência de luz	%Luz	
Temperatura	Sensor de Temperatura	°C	
Umidade	Sensor de Umidade	%UR	

© EXEHDA

Figura 25: Tela de Gerência dos Tipos de Sensores

- **Publicações Realizadas:** permite visualizar, cadastrar e remover manualmente publicações referentes a um sensor. A função de cadastro de publicações de

forma manual foi criada com o objetivo de avaliar algumas funcionalidades da aplicação.

- **Agendamento:** esta é uma funcionalidade direcionada ao gerenciamento do uso de sensores em determinados ambientes controlados. Este recurso permite efetuar uma reserva de um ambiente para uma determinada data, relacionada com um usuário do sistema. Logo, se um usuário tentar efetuar o agendamento de um determinado ambiente que já foi agendado, o sistema o avisará que já existe um agendamento para este ambiente (vide Figura 26), evitando assim que haja conflitos em uma eventual tomada de decisão por meio de regras distintas vinculadas aquele dispositivo. Neste menu também é disponibilizado a função Visualizar, que possibilita a visualização dos agendamentos registrados e a função Relatórios que disponibiliza a geração de relatórios dos agendamentos relacionados a um determinado ambiente.
- **Administração:** por este recurso da aplicação, pode-se realizar gerência de menus, usuários, perfis de usuários e fabricantes de dispositivos.



Figura 26: Notificação de Registro de Agendamento

Aplicação de Visualização

A aplicação no seu modo de visualização possibilita a seleção de um ou mais sensores, dentro de um contexto de interesse, para exibição do registro histórico das

suas informações contextuais. Estas informações podem ser apresentadas na forma de um relatório textual (vide Figura 27) ou através de modo gráfico (vide Figura 28).

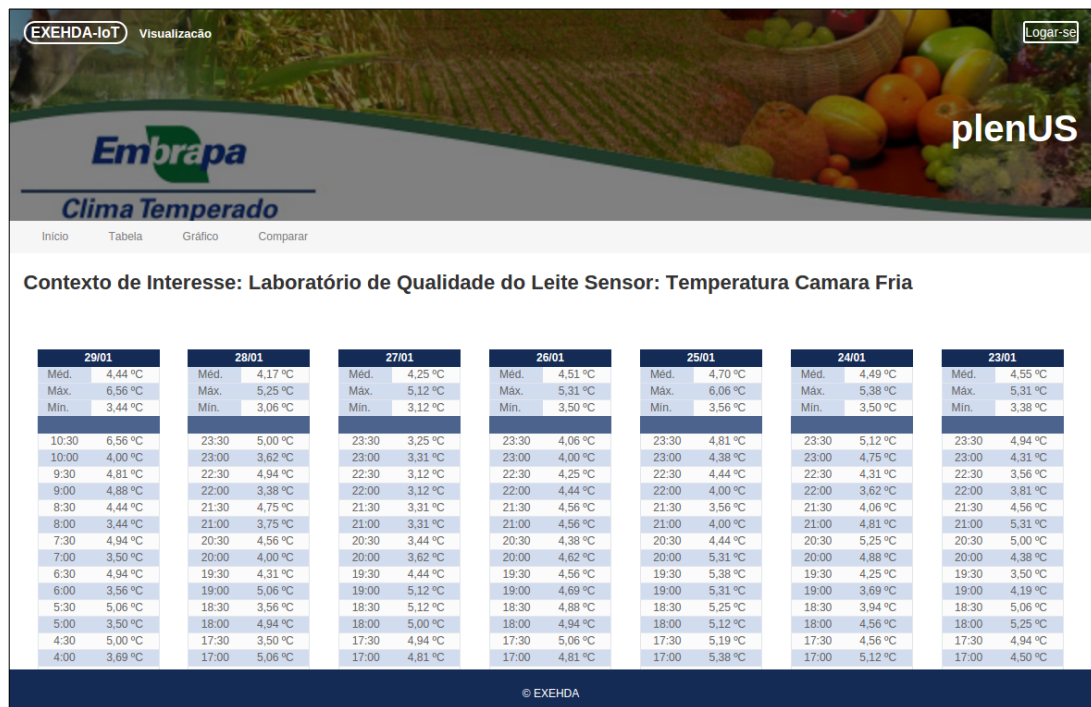


Figura 27: Visualização Textual das Informações Contextuais

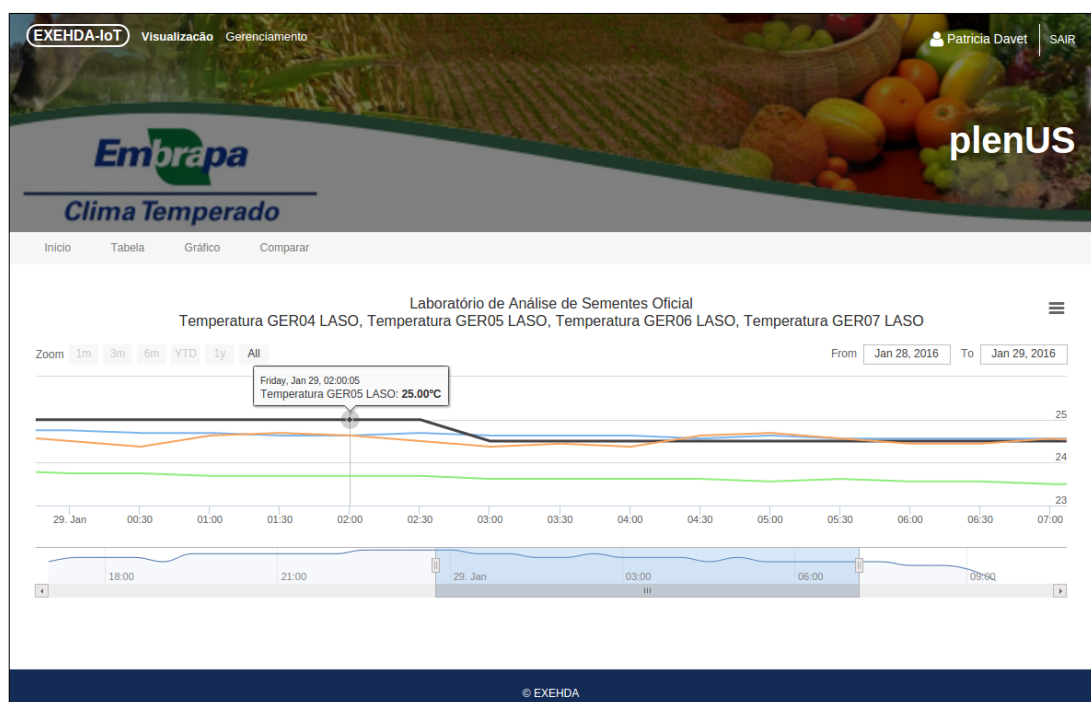


Figura 28: Visualização Gráfica das Informações Contextuais

Através desse modo operacional os usuários podem ter acesso à visualização das variações dos valores das grandezas medidas ocorridas nos equipamentos e/ou sa-

las durante os períodos de análise dos experimentos, o que influencia a resultados positivos quando de um monitoramento adequado.

O relatório gráfico desenvolvido permite visualizar simultaneamente as curvas de variação dos valores de vários sensores utilizados no EXEHDA-IoT. A seleção dos sensores a serem visualizados é feita a partir de um menu com suporte a múltipla seleção. Também é disponibilizado um recurso de inspeção que permite a comparação dos valores em um determinado instante do tempo. A janela de tempo dos dados que estão sendo visualizados pode ser definida pelo usuário através da mesma interface gráfica que exibe os valores sensorizados.

5.2 Considerações sobre o Capítulo

Neste capítulo foi apresentado um estudo de caso no domínio da agricultura utilizado para avaliação das funcionalidades do EXEHDA-IoT, que tem como base o projeto plenUS, o qual esta dissertação tem envolvimento direto e que possui como meta a implantação de sistemas ubíquos de forma a contribuir ao atendimento das atividades de pesquisa da Embrapa Clima Temperado.

Desta forma foi realizada uma caracterização dos ambientes e demandas de sensoriamento que constituem o cenário avaliado, bem como toda a infraestrutura de software e hardware utilizada para a sua implantação. Por fim, foram apresentadas as funcionalidades disponibilizadas pela aplicação web concebida para acesso ao usuário dos recursos propiciados pelo EXEHDA-IoT, que constitui de dois modos de operação, uma de visualização das informações contextuais adquiridas e outra de gerenciamento dos dispositivos de sensoriamento e/ou atuação.

6 CONSIDERAÇÕES FINAIS

Neste capítulo encontram-se resumidas as principais conclusões decorrentes da pesquisa que culminou com a concepção e avaliação do EXEHDA-IoT, bem como as publicações realizadas e os trabalhos futuros previstos.

6.1 Principais Conclusões

A Internet das Coisas vem ganhando destaque como uma forma de materializar as premissas da Computação Ubíqua, sendo a abordagem que mais contribuiu na última década para o emprego da ubiquidade em aplicações. No entanto, há uma série de desafios a serem superados quando do seu emprego em cenários cotidianos de utilização.

Um dos desafios está em como integrar e gerenciar a grande variedade de dispositivos embarcados, atualmente ofertados pelo mercado, possuindo as mais diferentes tecnologias de hardware e software, bem como padrões de comunicação. Considerando isto, se faz necessária a concepção de infraestruturas que possuam mecanismos capazes de tratar, de forma oportuna, os aspectos decorrentes de uma heterogeneidade e/ou escalabilidade elevadas.

A utilização de middlewares para tratar desafios da IoT vem sendo abordada como uma solução promissora. Nesta perspectiva, esta dissertação de mestrado veio a contribuir com os esforços de pesquisa do LUPS em relação ao middleware EXEHDA. De modo mais específico, capacitou o Subsistema de Adaptação e Reconhecimento do Contexto (SARC) do EXEHDA para tratar os desafios da IoT propostos quando de sua concepção. A premissa buscada foi prover ciência de contexto, tendo em vista a aquisição de informações contextuais a partir de um grande número de dispositivos heterogêneos, distribuídos, com capacidade de mobilidade e com poder computacional restrito.

Como abordagem para atender as premissas do EXEHDA-IoT, foi introduzido um novo elemento na arquitetura de software do SARC denominado Gateway. Os gateways representam equipamentos com hardware específico para interoperar dire-

tamente com sensores e atuadores, oferecendo interfaces físicas diretas e efetuando conversões de protocolos. Estes ficam então responsáveis pelo tratamento das informações contextuais em mais baixo nível, de maneira a uniformizar a diversidade de tecnologias.

O EXEHDA-IoT previu o uso de três tipos de gateways: gateways nativos, cujas funcionalidades operam de maneira integrada à arquitetura do SARC; gateways proprietários, com funcionalidades dependentes dos recursos oferecidos pelas interfaces de programação providas pelos fabricantes, e gateways virtuais que representam uma instância do gateway nativo integrado ao mesmo hardware do Servidor de Borda.

Para o gateway nativo foram avaliados os dois protocolos mais utilizados atualmente no cenário da IoT, Coap e MQTT. A escolha pelo CoAP foi devido a este utilizar princípios REST, como também pelo fato dos testes realizados apontarem um melhor desempenho do CoAP. No entanto, o EXEHDA-IoT provê suporte aos dois tipos de protocolos, com o intuito de atender a comunicação tanto por MQTT como CoAP pelos gateways proprietários.

A estratégia de interoperação adotada entre Gateways, Servidores de Borda e Servidor de Contexto utilizou princípios REST com o intuito de prover uma interface padronizada e totalmente independente da linguagem de programação utilizada, tanto na implementação dos recursos oferecidos quanto no uso dos mesmos. Para tal, foi concebida uma API REST para prover acesso aos diferentes recursos dos dispositivos de coleta e atuação.

Os mecanismos concebidos pelo EXEHDA-IoT possibilitaram a descoberta de dispositivos por meio de uma estratégia de identificação *plug and play* tendo por base o protocolo UPnP. A descoberta de dispositivos proposta teve como objetivo central contribuir na otimização dos esforços de gerência, de forma a tratar a elevada dinamicidade no que diz respeito aos procedimentos de inclusão e remoção de dispositivos, minimizando os esforços de configuração por parte do usuário quando da necessidade de realização destes procedimentos.

O esforço de empregar no EXEHDA-IoT somente tecnologias abertas em sua concepção entende-se como uma abordagem promissora para o emprego do middleware em diferentes cenários de uso.

Por fim, para a avaliação das funcionalidades do EXEHDA-IoT foi empregado um estudo de caso no domínio da agricultura, junto a Embrapa Clima Temperado. Os resultados atingidos com o cenário de uso se mostraram promissores, fornecendo um *feedback* para avanços futuros, o que estimula a continuidade das pesquisas na área.

6.2 Publicações Realizadas

No decorrer desta dissertação de mestrado alguns artigos foram publicados, contemplando resultados parciais do trabalho em desenvolvimento, bem como a projeção deste nos esforços de pesquisa em andamento no LUPS.

- LOPES, João L. B.; SOUZA, A.; SOUZA, Rodrigo ; **DAVET, P. T.**; PERNAS, A. M.; Yamin, Adenauer; GEYER, Cláudio Fernando Resin. A Situation-Aware Pervasive Approach for Assessing Therapeutic Goals in Healthcare Environment. In: ACM Symposium on Applied Computing, 2016, Pisa, Italy. (Aceito para publicação, 2016).
- **DAVET, P. T.**; KAISER FILHO, H.; JOÃO, L. R. S.; XAVIER, L.; CARVALHO, T. R.; LOPES, J. L.; SOUZA, R. S.; PERNAS, A. M.; YAMIN, A. C. Consciência de Contexto na IoT: Uma Arquitetura Distribuída e Escalável. In: V Brazilian Symposium on Computing Systems Engineering, 2015, Foz do Iguaçu, PR. (SBESC 2015).
- **DAVET, P. T.**; KAISER FILHO, H.; JOÃO, L. R. S.; XAVIER, L.; CARVALHO, T. R.; SOUZA, R. S.; LOPES, J. L.; YAMIN, A. C. EXEHDA-IoT: Uma Abordagem Consciente de Contexto Direcionada à Internet das Coisas. In: XIII Escola Regional de Redes de Computadores, 2015, Passo Fundo, RS. (ERRC 2015).
- **DAVET, P. T.**; PERNAS, A. M.; YAMIN, Adenauer C. EXEHDA-IoT: Uma Contribuição à Arquitetura do Middleware EXEHDA Direcionada à Internet das Coisas. In: Encontro de Pós Graduação UFPEL , 2015, Pelotas - RS. (ENPOS UFPEL 2015).
- SOUZA, Rodrigo. S.; LOPES, João. L. B.; **DAVET, P. T.**; GARCIA, C. G.; GADOTTI, Gizele. I.; YAMIN, Adenauer C.; GEYER, C. F. R. Context Awareness in UbiComp: an IoT Oriented Distributed Architecture. In: IEEE International Conference on Electronics, Circuits, and Systems, 2015, Cairo, Egito. (ICECS 2015).
- ALMEIDA, R. B.; MACHADO, R. S. ; ROSA, D. Y.; **DAVET, P. T.**; DONATO, L.; YAMIN, Adenauer C.; PERNAS, A. M. Segurança na Internet das Coisas: uma abordagem de SIEM customizável e baseada em Consciência de Situação. In: XLII Seminário Integrado de Software e Hardware, 2015, Recife - PE. (SEMISH 2015).
- SOUZA, Rodrigo. S.; LOPES, João. L. B.; SOUZA, Alexandre R. R.; **DAVET, P. T.**; PERNAS, A. M.; YAMIN, Adenauer C.; GEYER, C. F. R. Uma Arquitetura

Distribuída para Internet das Coisas na Medicina Ubíqua. In: XLII Seminário Integrado de Software e Hardware, 2015, Recife - PE. (SEMISH 2015).

- SOUZA, Rodrigo. S.; LOPES, João. L. B.; **DAVET, P. T.**; GADOTTI, Gizele. I.; PERNAS, A. M.; YAMIN, Adenauer C.; GEYER, C. F. R. ColoT: Uma Arquitetura Distribuída para IoT Direcionada à Consciência de Contexto das Aplicações Ubíquas. In: VII Simpósio Brasileiro de Computação Ubíqua e Pervasiva, 2015, Recife - PE. (SBCUP 2015).
- MACHADO, R. S.; ALMEIDA, R. B.; **DAVET, P. T.**; YAMIN, Adenauer C.; PERNAS, A. M. Uma Proposta de Gerenciamento de Modelo Híbrido de Contexto em Sistemas Distribuídos. In: Escola Regional de Alto Desempenho do Rio Grande do Sul, 2015, Gramado - RS. (ERAD 2015).

Um registro dos diferentes trabalhos desenvolvidos ao longo do Mestrado está disponível em <<http://ubiq.inf.ufpel.edu.br/ptdavet/>>.

6.3 Trabalhos Futuros

Os resultados obtidos com a realização desta dissertação de mestrado se mostraram oportunos, estimulando a continuidade dos esforços de pesquisa em relação ao middleware EXEHDA junto ao grupo. Dentre os aspectos levantados para continuidade do trabalho destacam-se:

- implantação de um framework para gerenciamento do motor de regras no Servidor de Borda;
- conceber para a arquitetura do Servidor de Borda, suporte para configuração dos parâmetros empregados nos protocolos de comunicação, que seja adaptativo ao contexto da infraestrutura computacional;
- sistematizar procedimentos para a validação de Gateways de terceiros, que utilizem os protocolos suportados pelo EXEHDA-IoT;
- empregar os mecanismos concebidos para o EXEHDA-IoT em outros cenários de uso.

Além destes trabalhos futuros, artigos serão escritos direcionados a congressos da área para divulgar os resultados obtidos.

REFERÊNCIAS

4THLINE. **4th Line.org**. Disponível em :<<http://4thline.org>>. Acesso em novembro de 2015.

ASHTON, K. **That ‘internet of things’ thing in the real world, things matter more than ideas**. Disponível em: <<http://www.rfidjournal.com/article/print/4986> >. Acesso em dezembro de 2014.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. **Computer networks**, Elsevier, v.54, n.15, p.2787–2805, 2010.

BALDAUF, M.; DUSTDAR, S.; ROSENBERG, F. A survey on context-aware systems. **International Journal of Ad Hoc and Ubiquitous Computing**, IJAHUC, v.2, n.4, p.263, 2007.

BANDYOPADHYAY, S.; SENGUPTA, M.; MAITI, S.; DUTTA, S. Role of middleware for internet of things: A study. **International Journal of Computer Science & Engineering Survey**, IJCSES, v.2, n.3, p.94–105, 2011.

BANKS, A.; GUPTA, R. **MQTT Version 3.1.1. OASIS Standard**. Disponível em: <<http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf> >. Acesso em Julho de 2015.

BASSI, A.; HORN, G. Internet of Things in 2020: A Roadmap for the Future. **Information Society and Media**, European Commission, 2008.

BROCK, D. L. The Electronic Product Code (EPC): A Naming Scheme for Physical Objects. **Auto-ID Center White Paper WH-002**, MIT, 2001.

BROCK, J. D.; BRUCE, R. F.; CAMERON, M. E. Changing the world with a Raspberry Pi. **Journal of Computing Sciences in Colleges**, ACM, v.29, n.2, p.151–153, 2013.

CACERES, R.; FRIDAY, A. Ubicomp systems at 20: Progress, opportunities, and challenges. **IEEE Pervasive Computing**, IEEE, v.11, n.1, p.14–21, 2011.

CARRIOTS. **Carriots IoT Application Platform**. Disponível em :<<https://www.carriots.com/>>. Acesso em julho de 2015.

CHAQFEH, M.; MOHAMED, N. et al. Challenges in middleware solutions for the internet of things. **Collaboration Technologies and Systems (CTS), International Conference on**, USA, p.21–26, 2012.

COSTA, C. A. da; YAMIN, A. C.; GEYER, C. F. R. Toward a General Software Infrastructure for Ubiquitous Computing. **IEEE Pervasive Computing**, IEEE, v.7, n.1, p.64–73, jan 2008.

DAVET, P.; KAISER FILHO, H.; JOÃO, L.; XAVIER, L.; CARVALHO, T.; FLEISCHMANN, A.; YAMIN, A.; LOPES, J.; SOUZA, R. Consciência de Contexto na IoT: uma Arquitetura Distribuída e Escalável. **Proceedings of the V Brazilian Symposium on Computing Systems Engineering**, SBESC, 2015.

DELICATO, F. C.; PIRES, P. F.; BATISTA, T.; CAVALCANTE, E.; COSTA, B.; BARROS, T. Towards an IoT ecosystem. **Proceedings of the First International Workshop on Software Engineering for Systems-of-Systems - ACM**, USA, p.25–28, 2013.

DEY, A. K. Understanding and Using Context. **Personal and Ubiquitous Computing**, USA, v.5, p.4–7, 2001.

DLNA. **Digital Living Network Alliance**. Disponível em: <<http://www.dlna.org/>>. Acesso em julho de 2015.

EMBRAPA. **Embrapa Clima Temperado**. Disponível em: <<https://www.embrapa.br/clima-temperado>>. Acesso em outubro de 2015.

EVANS, D. A Internet das Coisas - Como a próxima evolução da Internet está mudando tudo. **Cisco Internet Business Solutions Group**, IBSG, 2011.

FERREIRA, H. G. C. **Arquitetura de Middleware para Internet das Coisas**. 2014. Dissertação de Mestrado — Programa de Pós-Graduação em Engenharia Elétrica - Universidade de Brasília.

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**. 2000. Tese de Doutorado em Informação e Ciência da Computação — Universidade da Califórnia, Califórnia-USA.

GAZIS, V.; HUBER, M.; LEONARDI, A.; MATHIOUDAKIS, K.; WIESMAIER, A.; ZEIGER, F.; VASILOMANOLAKIS, E. A Survey of Technologies for the Internet of Things. **IEEE Computer Society**, Dubrovnik, Croatia, 2015.

GROUP, O. M. **Business Process Model and Notation**. Disponível em :<<http://www.bpmn.org>>. Acesso em maio de 2015.

GUBBI, J.; BUYYA, R.; MARUSIC, S.; PALANISWAMI, M. Internet of Things (IoT): A vision, architectural elements, and future directions. **Future Generation Computer Systems**, Elsevier, v.29, n.7, p.1645–1660, 2013.

GUILLEMIN, P.; FRIESS, P. Internet of things strategic research roadmap. **The Cluster of European Research Projects**, Europe, September 2009.

GUINARD, D. Towards Opportunistic Applications in a Web of Things. **IEEE International Conference on Pervasive Computing and Communications Workshops**, USA, 2010.

IOT-A. **Internet of Things Architecture of the European Lighthouse Integrated Project**. Disponível em :<<http://http://www.iot-a.eu>>. Acesso em dezembro de 2014.

JOÃO, L.; DAVET, P.; KAISER FILHO, H.; XAVIER, L.; CARVALHO, T.; YAMIN, A.; SOUZA, R. Concepção e Avaliação de Gateways Nativos para o Middleware EXEHDA. **Anais da XVI Escola Regional de Alto Desempenho**, São Leopoldo - RS, 2016.

KNAPPMAYER, M.; KIANI, S.; REETZ, E.; BAKER, N.; TONJES, R. Survey of Context Provisioning Middleware. **Communications Surveys Tutorials**, IEEE, v.15, n.3, p.1492–1519, Third 2013.

KRUMM, J. (Ed.). **Ubiquitous Computing Fundamentals**. CRC: Chapman & Hall, 2010.

KURNIAWAN, A. **Sparkfun ESP8266 Thing Development Workshop**. Canada: PE PRESS, 2015. An optional note.

LABLEITE. **Notícias Embrapa Clima Temperado**. Disponível em: <<https://www.embrapa.br/web/portal/busca-de-noticias/-/noticia/5956261/lableite-da-embrapa-passa-por-avaliacao-de-qualidade>>. Acesso em outubro de 2015.

LIMA, A. T. **Aplicação de Internet of Things em casas inteligentes - Serviço Aplicacional**. 2014. Dissertação de Mestrado — Instituto Politécnico do Porto. Instituto Superior de Engenharia do Porto.

LOPES, J.; SOUZA, R.; GEYER, C.; COSTA, C.; BARBOSA, J.; PERNAS, A.; YAMIN, A. A Middleware Architecture for Dynamic Adaptation in Ubiquitous Computing. **Journal of Universal Computer Science**, J.UCS, v.20, n.9, p.1327–1351, sep 2014.

MACEKOVÁ, L. 1-Wire-The Technology for Sensor Networks. **Acta Electrotechnica et Informatica**, De Gruyter Open Sp. z oo, v.12, n.4, p.52, 2012.

MARTINS, I. R.; ZEM, J. L. Estudo dos Protocolos de Comunicação MQTT e COAP para Aplicações Machine-to-Machine e Internet das Coisas. **Depósito Biblioteca Fafec Americana**, Centro Estadual de Educação Tecnológica, v.3, n.1, p.64–87, 2015.

MAXIM. **Maxim Integrated**. Disponível em: <<https://www.maximintegrated.com>>. Acesso em setembro de 2014.

MIORANDI, D.; SICARI, S.; De Pellegrini, F.; CHLAMTAC, I. Internet of things: Vision, applications and research challenges. **Ad Hoc Networks**, Elsevier, v.10, n.7, p.1497–1516, sep 2012.

MQTT. **Message Queue Telemetry Transport**. Disponível em: <<http://mqtt.org/>>. Acesso em julho de 2015.

NAGY, M.; KATASONOV, A.; SZYDLOWSKI, M.; KHRIYENKO, O.; NIKITIN, S.; TER-ZIYAN, V. **Challenges of middleware for the internet of things**. Croatia: INTECH Open Access Publisher, 2009.

NEWS, R. S. **O Laboratório de Análise de Sementes**.

OASIS-MQTT. **OASIS Message Queuing Telemetry Transport (MQTT) TC**. Disponível em: <<https://www.oasis-open.org/committees/mqtt>>. Acesso em julho de 2015.

PERERA, C.; ZASLAVSKY, A.; CHRISTEN, P.; GEORGAKOPOULOS, D. Context aware computing for the internet of things: A survey. **Communications Surveys & Tutorials, IEEE**, USA, v.16, n.1, p.414–454, 2013.

PIRES, P. F.; CAVALCANTE, E.; BARROS, T.; DELICATO, F. C.; BATISTA, T.; COSTA, B. A platform for integrating physical devices in the Internet of Things. **Proceedings of the 12th IEEE International Conference on Embedded and Ubiquitous Computing**, USA, p.234–241, 2014.

PIRES, P. F.; DELICATO, F. C.; BATISTA, T.; BARROS, T.; CAVALCANTE, E.; PITANGA, M. Plataformas para a Internet das Coisas. **Livro Texto de Minicursos - SBRC 2015**, Vitória - ES, 2015.

RASPBERRY. **Raspberry Pi**. Disponível em: <<http://www.raspberrypi.org>>. Acesso em agosto de 2014.

SHELBY, Z.; HARTKE, K.; BORMANN, C. **Constrained Application Protocol (CoAP), IETF Standards Track RFC Proposed Standard, CORE Working Group**. Disponível em: <<http://datatracker.ietf.org/doc/draft-ietf-core-coap/>>. Acesso em julho de 2015.

SOLDATOS, J.; SERRANO, M.; HAUSWIRTH, M. Convergence of utility computing with the internet-of-things. **Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on**, IEEE, p.874–879, 2012.

SOUZA, R. S.; LOPES, J. L. B.; DAVET, P. T.; GARCIA, C. G.; GADOTTI, G. I.; YAMIN, A. C.; GEYER, C. F. R. Context Awareness in UbiComp: an IoT Oriented Distributed Architecture. **Proceedings of 2015 IEEE International Conference on Electronics, Circuits, and Systems**, ICECS, 2015.

SRI Consulting Business Intelligence. **Disruptive civil technologies**: Six technologies with potential impacts on US interests out to 2025. USA: National Intelligence Council, 2008.

TAN, L.; WANG, N. Future internet: The internet of things. **Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on**, IEEE, v.5, p.V5–376, 2010.

UPNP, F. **Fórum UPnP**. Disponível em :<<http://upnp.org//>>. Acesso em julho de 2015.

VEGA-BARBAS, M.; CASADO-MANSILLA, D.; VALERO, M.; IPINA, D. López-de; BRAVO, J.; FLÓREZ, F. et al. Smart Spaces and Smart Objects Interoperability Architecture (S3OIA). **Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on**, USA, p.725–730, 2012.

WEISER, M. The Computer for the 21st Century. **Scientific American**, USA, v.265, n.3, p.66–75, January 1991.

WIKI LINKSMART, . **Middleware for Networked Embedded Systems**. Disponível em :<<https://linksmart.eu/redmine/projects/linksmart-opensource/wiki>>. Acesso em julho de 2015.

Xiao Hang Wang; Da Qing Zhang; Tao Gu; Hung Keng Pung. Ontology based context modeling and reasoning using OWL. **Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops**, IEEE, p.18–22, 2004.

ANEXO A COMPONENTES DESENVOLVIDOS PARA O EXEHDA-IOT

Os componentes desenvolvidos para o EXEHDA-IoT, direcionados ao Subsistema de Adaptação e Reconhecimento do Contexto (SARC) do middleware EXEHDA que são apresentados neste anexo, tem por base a tecnologia 1-Wire (MAXIM, 2014).

O sistema 1-Wire é uma rede de transmissão de dados, também conhecida como MicroLAN, que possibilita a comunicação digital entre um computador ou microcontrolador, atuando como mestre, e dispositivos da série 1-Wire tais como sensores, atuando como escravos. Por mestre, entende-se o elemento capaz de controlar e gerenciar a transmissão de dados. Por escravo, entende-se o dispositivo endereçado e gerenciado pelo mestre, portanto cada dispositivo escravo possui uma identificação única na rede 1-Wire para o seu gerenciamento.

Na rede 1-wire de transmissão de dados um único mestre pode ser conectado a múltiplos escravos em diversos tipos de topologia. Esta arquitetura confere ao sistema 1-wire versatilidade e simplicidade (vide Figura 29).

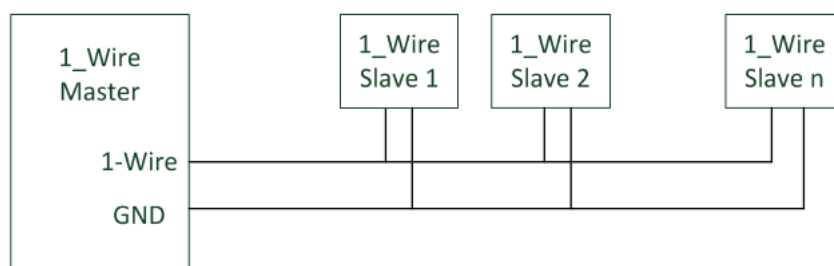


Figura 29: Esquema Básico da Rede 1-Wire. Fonte: (MACEKOVÁ, 2012)

Tanto o mestre do barramento como todos os dispositivos escravos atuam internamente como transceptores, enviando e recebendo dados através de uma única linha de dados. Os dados trafegados por esta linha de comunicação única podem fluir nas duas direções, mas apenas em uma direção de cada vez (operação *half duplex*).

Sensor de Temperatura 1-Wire DS18B20

Os sensores utilizados para os sistemas de monitoramento de temperatura em ambientes, equipamentos ou máquinas atendidos pelo EXEHDA-IoT são do modelo DS18B20¹. Este sensor pode operar, quanto a alimentação, de dois modos.

- **Modo Parasita:** onde é utilizado apenas a linha de dados e o ponto de referência (GND) da alimentação. O pino VDD não é utilizado e deve ser conectado ao GND. A alimentação é mantida por um capacitor interno ao dispositivo, que é carregado enquanto a linha de dados está em nível alto e supre a corrente necessária para o sensor nos momentos em que a linha se encontra em nível baixo.
- **Modo Alimentado:** o pino VDD é utilizado e alimentado com um valor entre a faixa de operação da rede 1-Wire que é de 3V à 5,5V.

Para o estudo de caso foi definido utilizar o modo alimentado, devido ao cenário onde os sensores foram instalados, constituir de um ambiente submetido a ruídos elétricos (interferências), onde o uso do modo parasita não é recomendado.

Com o intuito de atender as demandas do estudo de caso realizado na Empresa Clima Temperado, foram desenvolvidos e/ou utilizados alguns acoplamentos mecânicos para o sensor de temperatura DS18B20.

- **Sensor de temperatura 1-Wire DS18B20 encapsulado para emprego submerso em líquido (água):** desenvolvido para necessidade específica do equipamento Geladeira de amostras do Lableite (vide Figura 30), onde observou-se uma suba brusca na temperatura sempre que a porta do equipamento era aberta. Para tal foi desenvolvido um aparato mecânico, com o objetivo de evitar falsos positivos decorrentes de flutuações transitórias de temperatura no meio. Este aparato consiste de um frasco, dentro do qual é instalado um sensor DS18B20 fixado no interior de um tubo de alumínio vedado, que servirá como meio de condução do calor. O frasco, então, é preenchido com água, sendo que esta por possuir calor específico próximo ao do leite, permite, assim, uma inércia à suba de temperatura, o que possibilita uma monitoração da temperatura mais próxima das variações reais.
- **Sensor de Temperatura 1-Wire DS18B20 encapsulado para uso não submerso (ambientes úmidos):** sensor desenvolvido com encapsulamento metálico, o qual possui vedação resistente à umidade e a temperaturas elevadas (150°C) (vide Figura 31).

¹ <https://www.maximintegrated.com/en/products/analog/sensors-and-sensor-interface/DS18B20.html>



Figura 30: Sensor de Temperatura 1-Wire DS18B20 Desenvolvido para Emprego Submerso em Líquido (Água)

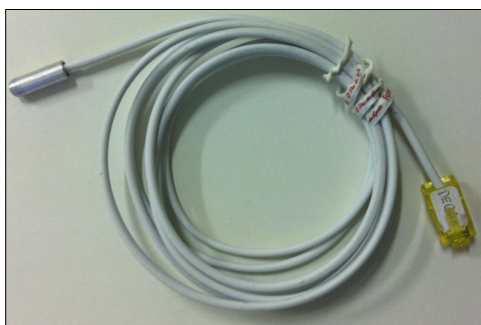


Figura 31: Sensor de Temperatura 1-Wire DS18B20 Desenvolvido para Uso Não Submerso (Ambientes Úmidos)

- **Sensor de Temperatura 1-Wire DS18B20 Waterproof:** sensor comercial para uso em ambientes úmidos, utilizado nos equipamentos germinadores e geladeiras do LASO (vide Figura 32).



Figura 32: Sensor de Temperatura 1-Wire DS18B20 Waterproof (Procedência Comercial)

Sensor de Umidade

O sensor de umidade relativa desenvolvido opera na faixa de 0 à 100% com uma precisão de 3%. O encapsulamento projetado para o mesmo previne a incidência de luz direta e/ou água (vide Figura 33).

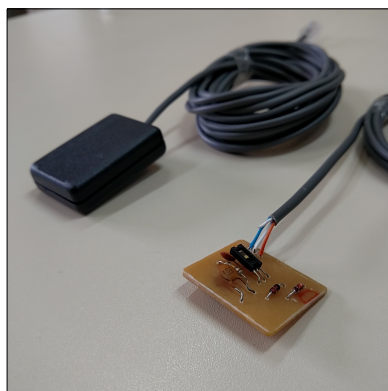


Figura 33: Sensor de Umidade Desenvolvido

A parte eletrônica do sensor de umidade tem por base dois semicondutores: o Honeywell HIH4000² que converte percentual de umidade em níveis de voltagem, e o dispositivo 1-Wire DS2438³ responsável por converter os níveis de voltagem analógicos recebidos do transceptor de umidade HIH4000 para o padrão digital, disponibilizando estes níveis por meio do protocolo 1-Wire. O dispositivo DS2438 dentre outras funcionalidades, também possui um sensor de temperatura interno, o qual foi utilizado nos ambientes onde havia demanda de monitoramento tanto para umidade como temperatura, evitando a necessidade de mais um dispositivo de sensoriamento no ambiente.

O Diagrama esquemático do sensor de umidade é apresentado na Figura 34.

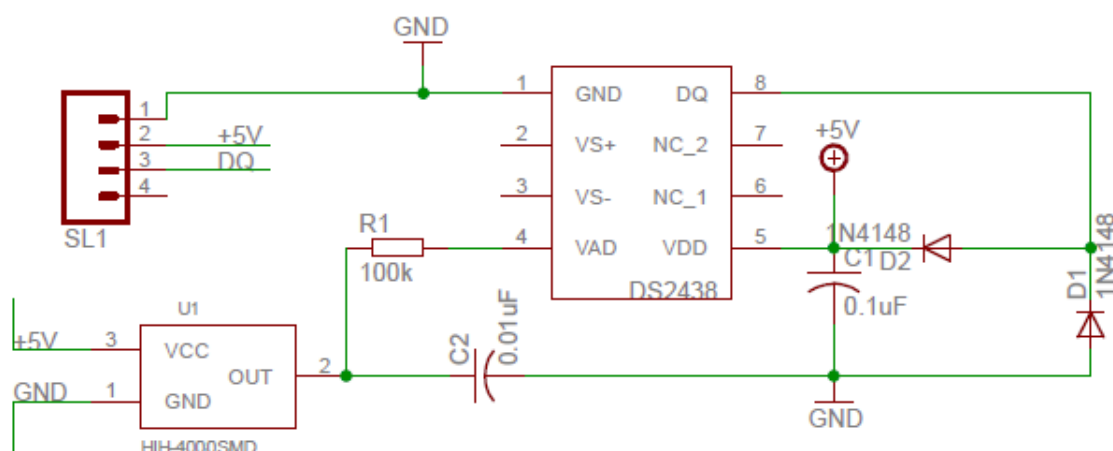


Figura 34: Sensor de Umidade - Diagrama Esquemático

²<http://sensing.honeywell.com/>

³<https://www.maximintegrated.com/en/products/power/battery-management/DS2438.html>

Sensor de Presença de Luz

O sensor de presença de luz desenvolvido tem por base os seguintes semicondutores: LDR (*Light Dependent Resistor*) e o dispositivo 1-Wire DS2438. O LDR é um sensor analógico fotocondutivo, ou seja, a sua resistência varia de acordo com a incidência de luz, à medida que a intensidade da luz aumenta, a sua resistência diminui podendo chegar a dezenas de ohms quando a incidência de luz é máxima e no caso contrário, de escuridão total, a resistência pode aumentar até um valor maior de um mega ohm.

Devido a esta característica o LDR foi ligado a um divisor de tensão e a entrada do conversor analógico digital do DS2438 (vide Figura 35), pois a medida que sua resistência aumenta (diminuição na incidência de luz) diminui a tensão na entrada do DS2438 e a medida que a resistência diminui (aumento na incidência de luz) a tensão aumenta, podendo dessa forma mensurar a incidência de luz, a qual é convertida pelo DS2438 em sinal digital e disponibilizado por meio do protocolo 1-Wire.

O Servidor de Borda recebe os dados de porcentagem de luz, que varia de 0 à 100% de acordo com a incidência e proximidade do LDR ao foco de luz alvo, e realiza o seguinte processamento de acordo com a faixa operacional:

- $\geq 30\%$ Luz - presença de Luz;
- $< 30\%$ Luz - ausência de Luz.

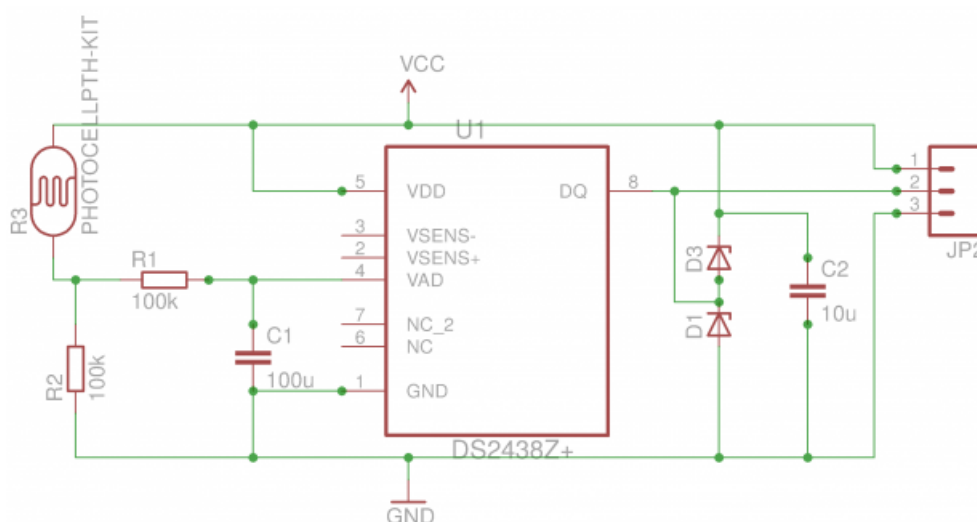


Figura 35: Sensor de Presença de Luz - Diagrama Esquemático

Foram projetados dois tipos de encapsulamento para o sensor de presença de luz: (i) com o LDR no mesmo encapsulamento do circuito e (ii) com o LDR encapsulado em um tubo plástico externo ao circuito, para ambientes úmidos e/ou de baixa temperatura (vide Figura 36).



Figura 36: Sensor de Presença de Luz Desenvolvido com o LDR Encapsulado em Tubo Plástico Externo ao Circuito

A utilização do sensor de encapsulamento externo ao circuito desenvolvido é mostrado na Figura 37 em dois equipamentos do LASO, à direita um germinador e à esquerda uma geladeira. Pode-se observar pela figura que somente uma extensão com o LDR e seu encapsulamento encontram-se no interior dos equipamentos, o restante do circuito é posicionado fora destes.



Figura 37: Utilização do Sensor de Presença de Luz Desenvolvido nos Equipamentos do LASO

Sensor de Detecção de Alarme Sonoro

O sensor de detecção de alarme sonoro foi desenvolvido para duas situações específicas: (i) detecção de alarme sonoro emitido pelo sensor detector de fumaça modelo DNI 6915 (vide Figura 38) e (ii) detecção de mudança de estado do alarme sonoro produzido por duas estufas de CO₂ no Laboratório de Reprodução Animal.

O hardware desenvolvido tem por base o dispositivo 1-Wire DS2413⁴, que constitui de um par de chaves bidirecionais e endereçáveis que obedece o protocolo 1-Wire.

O sensor prototipado é ligado aos contatos normalmente fechados, disponibilizados tanto pelo sensor de fumaça, destacado na Figura 38, como pelas estufas de

⁴<https://www.maximintegrated.com/en/products/interface/controllers-expanders/DS2413.html>

CO₂, para acesso à conexão externa. Estes contatos indicam o estado do alarme, ou seja, quando estes contatos estão fechados indicam que não há alarme e quando abrem indicam o acionamento do alarme. O dispositivo DS2413 detecta a alteração de estado do alarme e disponibiliza digitalmente esta informação por meio do protocolo 1-Wire.



Figura 38: Sensor Detector de Fumaça (Procedência Comercial)

Atuador - Alerta Visual

Para a funcionalidade de alerta visual foi desenvolvido um atuador que consiste em um dispositivo emissor de luz de elevada intensidade, visível mesmo sob condições de iluminação diurnas e a distâncias que chegam a dezenas de metros. O dispositivo foi encapsulado em uma caixa com lente difusora de luz para facilitar sua visibilidade.



Figura 39: Atuador de Alerta Visual

Na Figura 39 é possível ver o arranjo mecânico desenvolvido no qual estão integrados os recursos para interoperabilidade com a tecnologia 1-Wire com aqueles necessários para estabilidade mecânica e ótica da solução. Na Figura 40 é apresentado o diagrama esquemático.

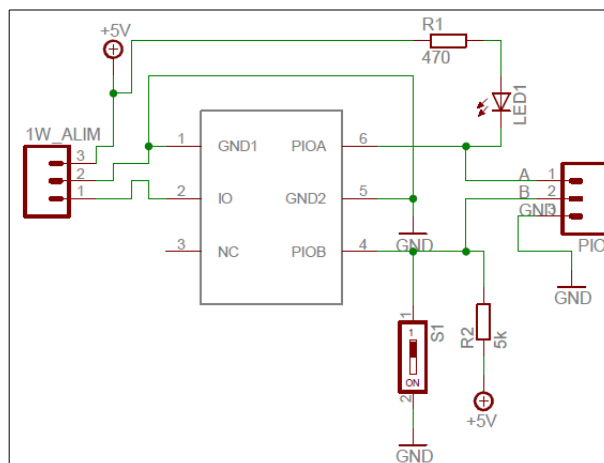


Figura 40: Atuador de Alerta Visual - Diagrama Esquemático

Interface Serial 1-Wire

Com o intuito de implementar a comunicação do Servidor de Borda com os dispositivos de uma rede 1-Wire, foi desenvolvida uma interface para uso com a porta serial ou serial via USB (vide Figura 41). Este dispositivo permite a leitura e/ou atuação, por meio cabedo, de informações providas por sensores da rede 1-Wire.



Figura 41: Interface Serial 1-Wire Desenvolvida

A concepção deste dispositivo de interface entre sensores e/ou atuadores e sistemas computacionais atendeu as especificações fornecidas pela Dallas Semiconductor, fabricante líder de dispositivos para a tecnologia 1-Wire. O diagrama esquemático desenvolvido está apresentado na Figura 42.

HUB 1-Wire

Para atender a proposta de um cabeamento estruturado quando da implementação da rede 1-Wire no ambiente ubíquo, foram desenvolvidos HUBs 1-Wire empregando conectores padrão RJ45. Com o suporte de conexões tipo RJ45 facilmente podem ser incluídos, excluídos e/ou trocados dispositivos 1-Wire. Cada HUB possui conexão

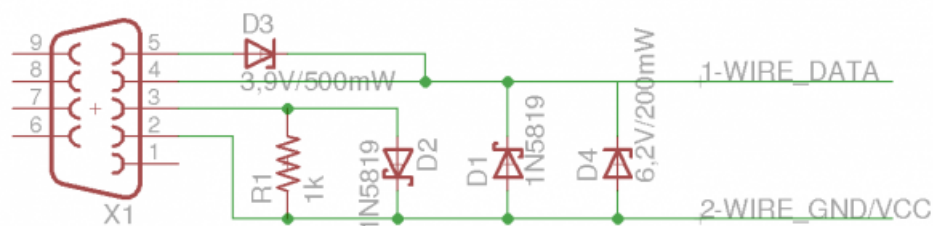


Figura 42: Interface Serial 1-Wire - Diagrama Esquemático

para cinco sensores.

Além das conexões necessárias para troca de dados, o HUB também disponibiliza as tensões necessárias para os sensores e/ou atuadores que necessitem de alimentação. Seu diagrama esquemático é apresentado na Figura 44.

Com o intuito de facilitar e organizar a integração dos sensores e/ou atuadores em rede foram prototipados dois tipos de HUB: (i) de início de rede, o qual possui conexão física direta com o Servidor de Borda e alimentação externa por uma fonte de 5V (vide Figura 43a) e (ii) de prolongamento, o qual possui alimentação fornecida pela própria rede 1-Wire, não necessitando de fonte externa (vide Figura 43b).

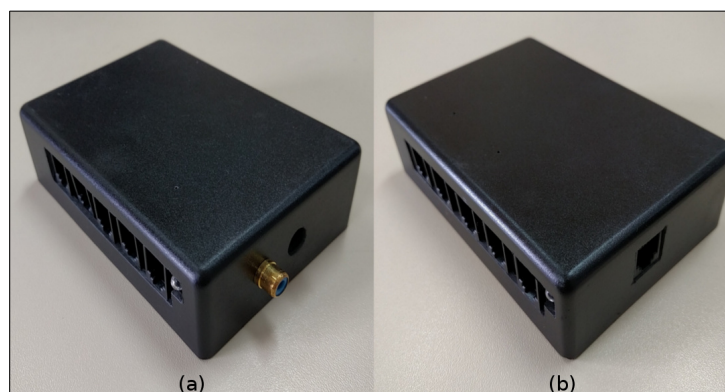


Figura 43: HUBs 1-Wire Desenvolvidos: (a) de Início de Rede, com Entrada para Alimentação Externa por Conexão RCA; (b) de Prolongamento, Alimentado pela Rede 1-Wire por Conexão RJ45

Um cenário de demonstração da forma de conexão cabeada da rede 1-Wire, com a utilização dos HUBs, conversor Serial 1-Wire e alguns sensores prototipados pode ser visualizado na Figura 45.

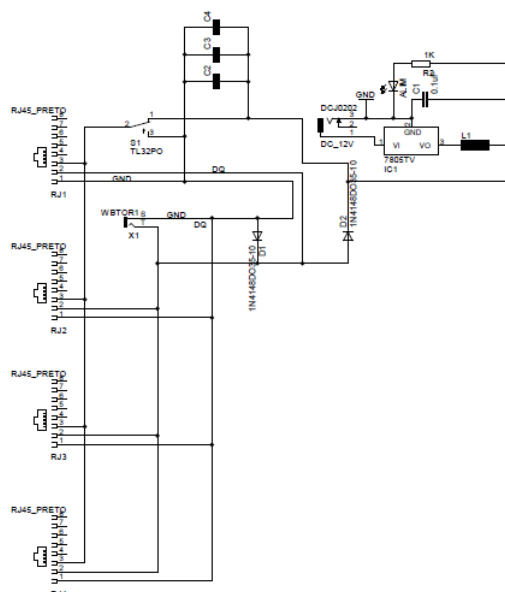


Figura 44: HUB - Diagrama Esquemático

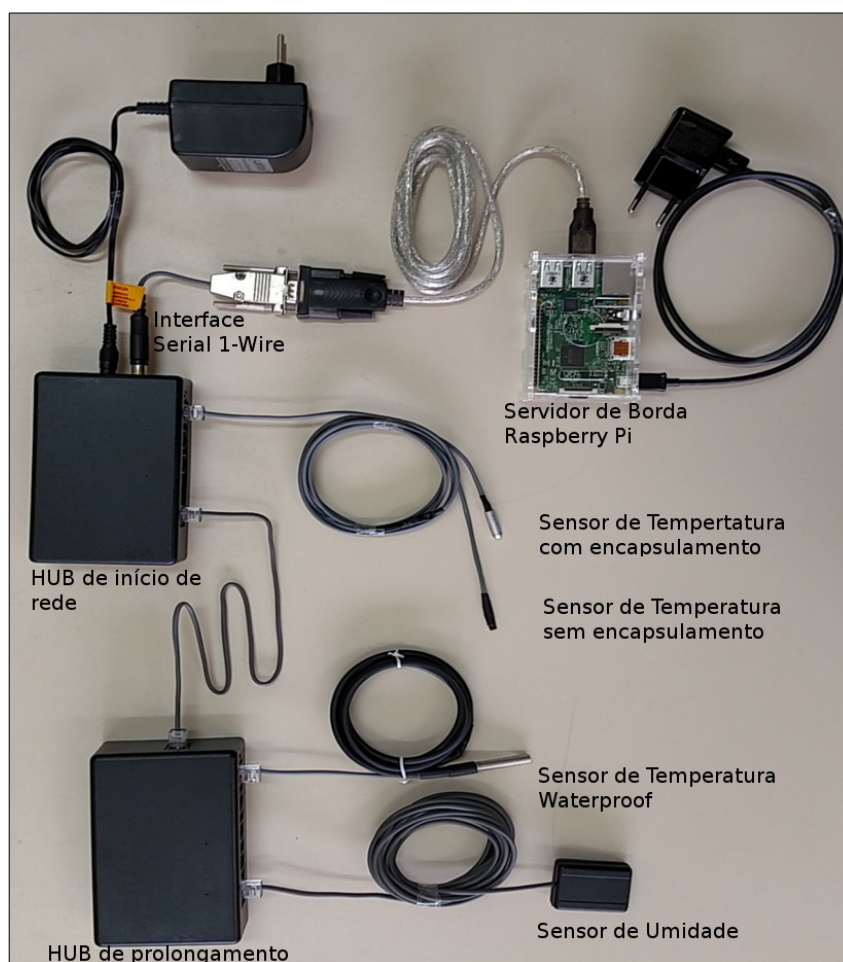


Figura 45: Cenário no EXEHDA-IoT de uma Rede de Sensoriamento 1-Wire Cabeada

Gateway Nativo do EXEHDA-IoT

A plataforma de hardware empregada para o Gateway Nativo do EXEHDA-IoT (GW-SB) foi o NodeMCU. NodeMCU é uma plataforma *open source* para desenvolvimento de aplicações IoT, baseado no processador ESP8266-12. Possui originalmente instalado firmware na linguagem Lua, mas pode ser programado pela IDE do Arduíno.

A escolha para programação do Gateway Nativo foi pela utilização da IDE do Arduíno, que se baseia na linguagem Wiring⁵, em virtude desta plataforma ter se mostrado mais estável nos teste realizados.

O protótipo desenvolvido do Gateway Nativo (vide Figura 46) possui conexão para sete sensores por meio de plugs P2 e alimentação via micro USB, além de comunicação wi-fi com o Servidor de Borda, bem como um LED indicador de seu estado (ligado ou desligado).



Figura 46: Protótipo do Gateway Nativo

Um dos gateways nativos instalados no Laboratório de Análise de Sementes Oficial (LASO) pode ser visualizado na Figura 47. Este Gateway Nativo é responsável por encapsular as tecnologias de sensoriamento provenientes de quatro germinadores, enviando os dados contextuais de forma padronizada ao Servidor de Borda do LASO.



Figura 47: Gateway Nativo do EXEHDA-IoT Instalado no LASO

⁵<http://wiring.org.co/>

Uma parte do cenário do EXEHDA-IoT no LASO é apresentado na Figura 48. O Gateway Nativo presente nesta figura atende a um total de seis sensores, dois sensores que monitoram umidade e temperatura, três sensores de temperatura e um sensor de presença de luz. A comunicação deste e mais três gateways instalados em lugares estratégicos no LASO, a fim de setorizar pontos de coleta próximos, se comunicam com o Servidor de Borda via wi-fi.

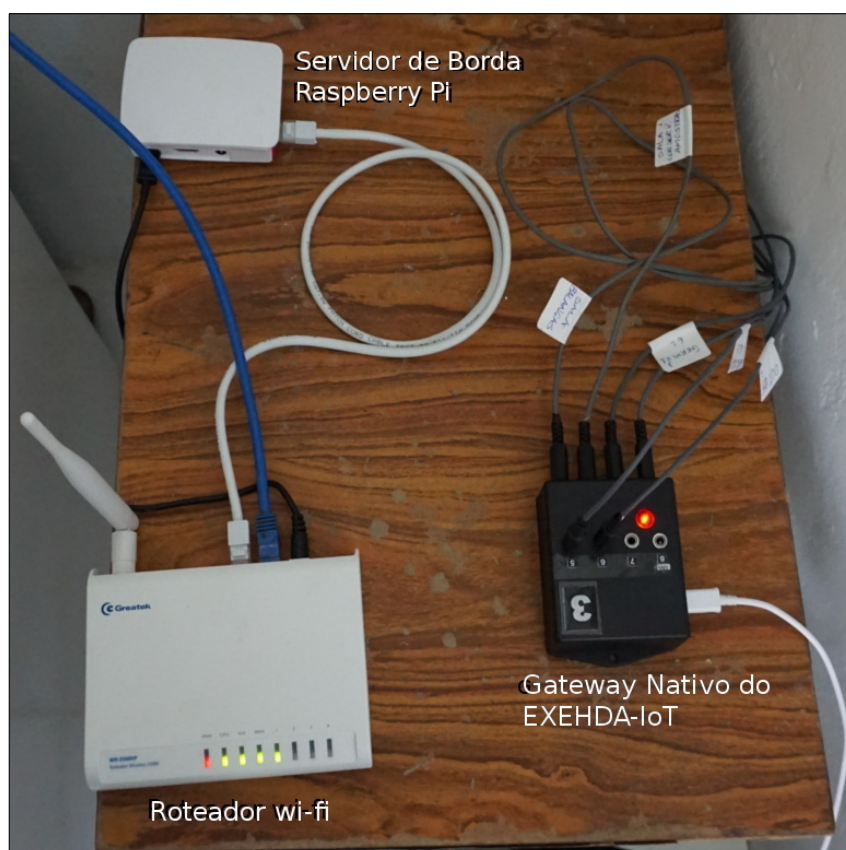


Figura 48: Cenário no EXEHDA-IoT de uma Rede de Sensoriamento sem Fio

ANEXO B REPOSITÓRIO DE INFORMAÇÕES CONTEXTUAIS

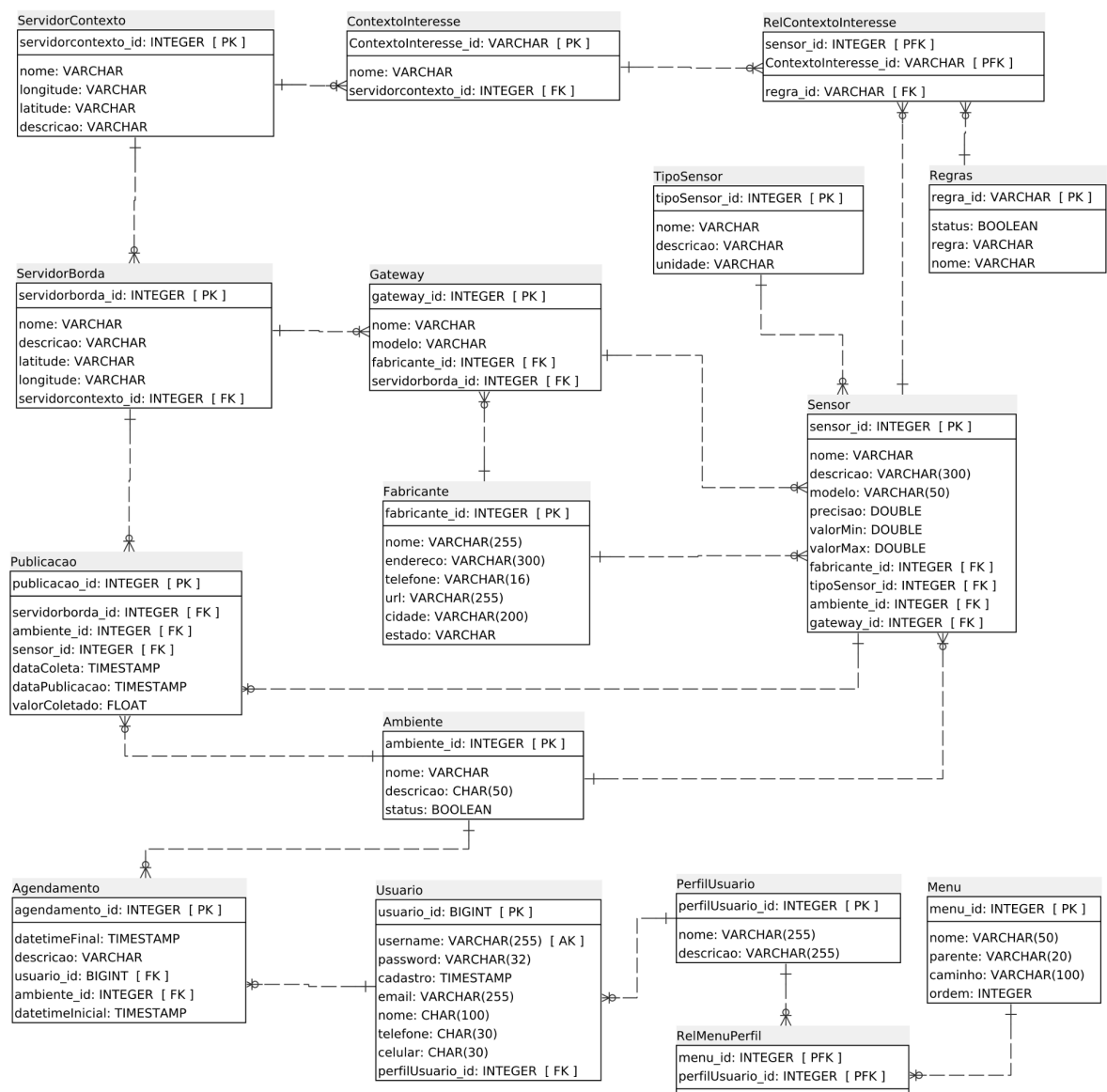


Figura 49: Repositório de informações contextuais

**EXEHDA-IoT: Uma Contribuição à Arquitetura do
Middleware EXEHDA Direcionada à Internet das Coi-
sas – Patrícia Teixeira Davet**



UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Programa de Pós-Graduação em Computação



Dissertação

**EXEHDA-IoT: Uma Contribuição à Arquitetura do
Middleware EXEHDA Direcionada à Internet das Coisas**

PATRÍCIA TEIXEIRA DAVET

Pelotas, 2016