UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

RODRIGO SANTOS DE SOUZA

Um middleware para Internet das Coisas com suporte ao processamento distribuído do contexto

Tese apresentada como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação

Orientador: Prof. Dr. Cláudio Fernando Resin Geyer

Porto Alegre
2017

AGRADECIMENTOS

O meu agradecimento especial à minha família pelo suporte emocional e apoio irrestrito ao longo de todos esses anos de estudos. À minha esposa Denise, aos meus filhos Bernardo e Vicente, aos meus pais Pedro e Ceni, aos meus irmãos Marcelo e Renato e também aos meus sogros, cunhadas, concunhados e sobrinhos.

Agradeço também ao meu orientador Cláudio Geyer por ter confiado em mim e por me dar todo suporte necessário à realização desse trabalho.

Meu agradecimento ao Adenauer, um grande amigo e apoiador incansável que me ajudou muito em todos os desafios acadêmicos que tenho enfrentado desde que ingressei no mestrado.

A todos os colgas dos grupos de pesquisa GPPD/UFRGS e LUPS/UFPel um agradecimento pelo excelente ambiente de trabalho e pelas contribuições ao longo dessa tese. Em especial ao João Ladislau, colega de Doutorado, pelas inúmeras conversas que tivemos, as quais foram fundamentais para o sucesso dos nossos trabalhos. Também agradeço aos colegas Anderson, Patrícia, Tainã, Huberto e Leonardo João por contribuírem significativamente nas tarefas de prototipação e nos cenários de uso.

Por fim, agradeço ao IFSUL pelo apoio institucional à realização desse trabalho.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS 5
LISTA DE FIGURAS 7
LISTA DE TABELAS 9
RESUMO 10
ABSTRACT 11
1 INTRODUÇÃO 12
1.1 Questões de pesquisa e hipótese 13
1.2 Objetivos 14
1.3 Identidade do projeto 15
1.4 Estrutura do texto 15
2 COIOT: ESCOPO DE PESQUISA 16
2.1 Ciência de Contexto na perspectiva da ubiquidade 16
2.1.1 Aquisição de informações contextuais 18
2.1.2 Processamento do contexto 19
2.1.3 Ciência de situação 21
2.2 Internet das Coisas 22
2.2.1 Desafios da IoT 24
2.2.2 Computação em Fog 26
2.3 Sistemas Distribuídos Baseados em Eventos 28
2.3.1 Sistemas de processamento de eventos 30
2.3.2 Processamento temporal de eventos 31
2.4 Middleware EXEHDA 32
2.5 Considerações sobre o capítulo 34
3 TRABALHOS RELACIONADOS 36
3.1 EcoDiF 36
3.2 Xively 37
3.3 Carriots 38
3.4 LinkSmart 38
3.5 Discussão dos trabalhos relacionados 39
3.6 Considerações sobre o capítulo 42
4 COIOT: CONCEPÇÃO DA ARQUITETURA 43
4.1 Principais características 43
4.2 Arquitetura Proposta 45
4.2.1 Visão Geral 46
4.2.2 Processamento do contexto baseado em regras e eventos distribuídos 49
4.2.3 Coleta e atuação baseadas em eventos 52
4.2.4 Interoperabilidade e tratamento da heterogeneidade 54
4.2.5 Gerência e descoberta de recursos 57
4.3 Considerações sobre o capítulo 58
5 COIOT: PROTOTIPAÇÃO E ESTUDOS DE CASO EM AGRICULTURA 60
5.1 Aspectos de Prototipação 60
5.2 Estudo de caso 1: processamento distribuído de informações contextuais com colaboração vertical em ambiente de produção 63
5.2.1 Análise de sementes na Embrapa Clima Temperado 63
5.2.2 Desafios no controle de laboratórios para análise de sementes 64
5.2.3 plenUS: Infraestrutura de hardware concebida 65
5.2.4 plenUS: Infraestrutura de software desenvolvida 68
5.2.5 Avaliação de aceitação do estudo de caso 78

5.3 Estudo de caso 2: processamento distribuído de informações contextuais com colaboração híbrida em ambiente simulado ... 79
5.3.1 A irrigação na viticultura de precisão ... 80
5.3.2 Infraestrutura de hardware e software utilizada ... 82
5.3.3 Aspectos considerados no estudo de caso ... 84
5.3.4 Avaliações realizadas ... 91
5.4 Considerações sobre o capítulo ... 96
6 CONSIDERAÇÕES FINAIS ... 97
6.1 Principais conclusões ... 97
6.2 Contribuições da tese ... 99
6.3 Trabalhos futuros ... 99
6.4 Publicações realizadas ... 100
REFERÊNCIAS ... 103
ANEXO A — COMPONENTES DESENVOLVIDOS PARA O COIOT ... 108
ANEXO B — REPOSITÓRIO DE CONTEXTO ... 120

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
API	*Application Program Interface*
BOD	*Biochemical Oxygen Demand*
BPMN	*Business Process Model and Notation*
BRT	*Boeing Research and Development Division*
CES	*Consumer Electronics Show*
COAP	*Constrained Application Protocol*
CORE	*Common Open Research Emulator*
CoIoT	*Context + IoT*
EEML	*Extended Enterprise Modeling Language*
EBBITS	*Enabling the Business-based Internet of Things and services*
ECA	Evento-Condição-Ação
EcoDiF	Ecossistema Web de Dispositivos Físicos
Embrapa	Empresa Brasileira de Pesquisa Agropecuária
EXEHDA	*Execution Environment for Highly Distributed Applications*
EPC	The Electronic Product Code
ETB	Estação Experimental Terras Baixas
FIFO	*First In, First Out*
GPPD	Grupo de Processamento Paralelo e Distribuído
GPIO	*General Purpose Input/Output*
HTTP	*Hypertext Transfer Protocol*
IoT	*Internet of Things* [ISO] *International Organization for Standardization*
IBSG	*Internet Business Solutions Group*
ID	Identificador

JSON	*JavaScript Object Notation*
LAS	Laboratórios de Análise de Sementes
LASO	Laboratório de Análise de Sementes Oficial
LUPS	*Laboratory of Ubiquitous and Parallel Systems*
MAPA	Ministério da Agricultura, Pecuária e Abastecimento
MQTT	*Message Queuing Telemetry Transport*
MVC	*Model-view-controller*
OWFS	*1-Wire File System*
plenUS	*plentiful Ubiquitous Systems*
RAS	Regras para Análise de Sementes
REST	*Representational State Transfer*
RFID	*Radio-Frequency IDentification*
SOA	*Service-Oriented Architecture*
TAM	*Technology Acceptance Model*
Tcl/Tk	*Tool Command Language/Toolkit*
UbiComp	*Ubiquitous Computing*
URI	*Uniform Resource Identifier*
UPnP	*Universal Plug and Play*
USB	*Universal Serial Bus*
XML	*ExtendibleMarkup Language*
ZM	Zona de manejo

LISTA DE FIGURAS

Figura 1.1 Identidade visual do CoIoT 15

Figura 2.1 Definição de Internet das Coisas 24
Figura 2.2 Janelas de tempo em processamento de eventos 32
Figura 2.3 Arquitetura do EXEHDA 33

Figura 4.1 Estratégia para o tratamento de contexto 44
Figura 4.2 Organização celular do ambiente ubíquo 46
Figura 4.3 Visão geral da arquitetura do CoIoT 47
Figura 4.4 Arquitetura do CoIoT. 49
Figura 4.5 Estratégia para o tratamento da heterogeneidade 54

Figura 5.1 Exemplo de uso da API REST 62
Figura 5.2 Infraestrutura concebida para o LASO 66
Figura 5.3 Dispositivos para interação com o ambiente 68
Figura 5.4 Janelas temporais 70
Figura 5.5 Menus de acesso 74
Figura 5.6 Configuração dos Servidores de Borda 75
Figura 5.7 Configuração de sensores 75
Figura 5.8 Tela de agendamentos de equipamentos 76
Figura 5.9 Relatório em modo textual 77
Figura 5.10 Relatório em modo gráfico 77
Figura 5.11 Tensão de água no solo em relação à produtividade da cultura 82
Figura 5.12 Visão geral dos componentes do CORE 83
Figura 5.13 Zonas de manejo em um parreiral 85
Figura 5.14 Zonas de manejo mapeadas no ambiente de emulação CORE 86
Figura 5.15 Tensão de Solo: Instante de atuação da regra 89
Figura 5.16 Aplicação web para visualização de dados contextuais 90
Figura 5.17 Aplicação web para visualização de dados contextuais 90
Figura 5.18 Gráfico do volume de dados 92
Figura 5.19 Gráfico em função do intervalo de aquisição 94
Figura 5.20 Gráfico em função da quantidade de sensores no ambiente 95

Figura A.1 Esquema básico da rede 1-Wire. 108
Figura A.2 Sensor de temperatura 1-Wire DS18B20 para emprego em líquido 110
Figura A.3 Sensor de temperatura 1-Wire DS18B20 para uso não submerso 110
Figura A.4 Sensor de temperatura 1-Wire DS18B20 waterproof comercial 111
Figura A.5 Sensor de umidade desenvolvido 111
Figura A.6 Sensor de umidade: diagrama esquemático 112
Figura A.7 Sensor de presença de luz: diagrama esquemático 113
Figura A.8 Sensor de presença de luz desenvolvido 113
Figura A.9 Utilização do sensor de presença de luz 114
Figura A.10 Sensor detector de fumaça comercial 114
Figura A.11 Atuador de alerta visual 115
Figura A.12 Atuador de alerta visual: diagrama esquemático 115
Figura A.13 Interface serial 1-Wire desenvolvida 116
Figura A.14 Interface serial 1-Wire: diagrama esquemático 116
Figura A.15 HUBs 1-Wire desenvolvidos 117
Figura A.16 HUB: diagrama esquemático 118

Figura A.17 Cenário no COIOT de uma rede de sensoriamento 1-Wire cabeada 118
Figura A.18 Protótipo do Gateway Nativo..... 119
Figura A.19 Gateway Nativo do COIOT instalado no LASO..... 119

Figura B.1 Repositório de informações contextuais 120

LISTA DE TABELAS

Tabela 3.1 Comparação dos trabalhos relacionados 40

Tabela 4.1 Eventos primitivos do COIOT 50

Tabela 5.1 API do COIOT para tratamento de sensores e atuadores 61
Tabela 5.2 API do COIOT pata tratamento da fila de publicações 61
Tabela 5.3 Especificações de coleta das Informações contextuais do LASO 69
Tabela 5.4 Regras de controle - Servidor de Borda..... 72
Tabela 5.5 Regras de controle - Servidor de Contexto..... 73
Tabela 5.6 Avaliação da facilidade de uso 78
Tabela 5.7 Avaliação da percepção de utilidade..... 79
Tabela 5.8 Condição do solo de acordo com o valor de tensão de água 81
Tabela 5.9 Regras do Servidor de Borda BordaZN1..... 88
Tabela 5.10 Eventos e subscrições do Servidor de Borda BordaZN1..... 88

RESUMO

Um dos principais desafios de pesquisa na UbiComp consiste em fornecer mecanismos para a ciência de contexto que promovam o desenvolvimento de aplicações que reajam de acordo com a dinâmica do ambiente de interesse do usuário. Para manter o conhecimento a respeito desse ambiente, a área da UbiComp pressupõe a utilização de informações produzidas e disponibilizadas em diferentes localizações, o tempo todo. Nesse sentido, os recentes avanços na área da Internet das Coisas (IoT) têm proporcionado uma crescente disponibilidade de sensores conectados em rede, os quais são potenciais produtores de informações contextuais do ambiente para aplicações ubíquas. Com essa motivação, nessa tese é apresentado o CoIoT, um middleware para Internet das Coisas concebido com o objetivo de gerenciar a coleta e o processamento das informações contextuais do ambiente físico, bem como a atuação remota sobre o mesmo. O CoIoT foi idealizado considerando os trabalhos previamente desenvolvidos pelo grupo de pesquisa GPPD (Grupo de Processamento Paralelo e Distribuído) da UFRGS, particularmente o middleware EXEHDA (*Execution Environment for Highly Distributed Applications*). Na concepção do CoIoT foi adotada uma abordagem distribuída de processamento de contexto que contempla tanto as premissas da IoT quanto as demandas das aplicações da UbiComp. A arquitetura proposta também contempla o gerenciamento de eventos distribuídos através de regras e *triggers* para tratar as mudanças de estados dos contextos de interesse. Além disso, a arquitetura proposta gerencia outros aspectos importantes nos cenários da IoT, como o tratamento da interoperabilidade, da heterogeneidade, apoio ao controle da escalabilidade e descoberta de recursos. As principais contribuições desta tese são: (i) a concepção de uma arquitetura para IoT capaz de realizar de forma distribuída tanto a coleta e processamento das informações contextuais, como a atuação remota no meio a fim de atender as aplicações da UbiComp e, (ii) a proposição de um modelo de processamento de eventos distribuídos adequado aos cenários da IoT. Para avaliar a arquitetura do CoIoT foram realizados dois estudos de caso na área da agricultura. O primeiro estudo de caso foi desenvolvido em ambiente de produção a partir de demandas de pesquisadores da área da agricultura, particularmente da análise de sementes. Já o segundo estudo de caso teve como cenário de testes ambientes da viticultura de precisão.

Palavras-chave: Computação ubíqua, internet das coisas, ciência de contexto.

An Internet of Things architecture with support to distributed processing context

ABSTRACT

One of the main research challenges in UbiComp is to provide mechanisms for context-aware to promote the development of applications that react according to the dynamics of user interest environment. To keep the knowledge of this environment, the area of UbiComp presupposes the use of information produced and made available in different locations, all the time. In this sense, the recent advances in the field of Internet of Things (IoT) have provided an increasing availability of sensors and actuators networked. These sensors are potential producers of contextual information. With this motivation, this thesis is presented the CoIoT, a middleware for Internet of Things (IoT) designed in order to manage the collect and processing of contextual information of the physical environment as well as remote actuation on it. The CoIoT was designed considering the work previously developed by the research group GPPD (Parallel Processing Group and distributed) of UFRGS, particularly middleware EXEHDA (Execution Environment for Highly Distributed Applications). In designing the CoIoT it was adopted a distributed approach of context processing that includes both the principles of IoT as the demands of the applications of UbiComp. The proposed architecture also includes rules based and triggers mechanisms to deal with events that characterize the changes of states of the contexts of interest. In addition, the proposed architecture manages other important aspects of IoT scenarios such as the treatment of interoperability, heterogeneity, support the control of scalability and resource discovery. Until now, the central contributions of this thesis include: (i) the design of an architecture for IoT able to perform distributed way both the collect and processing of contextual information, such as remote actuation in the environment in order to meet UbiComp applications and, (ii) the proposition of a distributed event processing model appropriate to the IoT scenarios. In order to evaluate the CoIoT architecture, two case studies were carried out in the area of agriculture. The first case study was developed in a production environment based on the demands of agricultural researchers, particularly seed analysis. On the other hand, the second case study was based on precision testing of viticulture environments.

Keywords: ubiquitous computing, internet of things, context-aware.

1 INTRODUÇÃO

A Computação Ubíqua (Ubicomp), proposta por Mark Weiser na década de 90, indica que as interações do usuário com os sistemas computacionais devem ocorrer de maneira pouco intrusiva de modo que estes possam se concentrar em suas atividades de interesse (WEISER, 1991). A área da Ubicomp pressupõe a provisão de ambientes computacionais com informações produzidas e disponibilizadas em diferentes localizações, o tempo todo. Assim, na Ubicomp, os sistemas computacionais, a partir da percepção do contexto de seu interesse, devem ser capazes de reagir considerando o estado das diferentes variáveis contextuais envolvidas.

Nesse sentido, os recentes avanços científicos e tecnológicos na área da Internet das Coisas (*Internet of Things* – IoT) têm proporcionado o uso de sensores em larga escala os quais constituem fontes geradoras de informações contextuais para as aplicações ubíquas cientes do contexto (PERERA et al., 2014). Grande parte dos desafios de pesquisa, relacionados ao uso da Internet das Coisas na obtenção de informações contextuais, estão associados com a diferença entre requisitos de alto nível das aplicações ubíquas e as operações básicas dos dispositivos da IoT (DELICATO; PIRES; BATISTA, 2013)(WANG et al., 2008).

Esta classe de sistemas computacionais, cientes do contexto, permite o desenvolvimento de aplicações mais ricas, mais elaboradas e complexas, atendendo a perspectiva do usuário estar em diferentes localizações, bem como a natureza dinâmica das infraestruturas de computação modernas. No entanto, o desenvolvimento de aplicações que percebam continuamente o ambiente e permanecem funcionando mesmo quando o usuário troca de dispositivos continua a ser um desafio de pesquisa em aberto (KNAPPMEYER et al., 2013).

A revisão de literatura aponta que a construção do suporte à ciência do contexto para as aplicações ubíquas apresenta diversos desafios, dentre eles: (*i*) a aquisição de informações contextuais a partir de fontes heterogêneas e distribuídas; (*ii*) o tratamento das informações de contexto adquiridas e a respectiva atuação sobre o meio físico; e (*iii*) a disseminação dessas informações aos consumidores interessados de forma distribuída e no momento oportuno (BELLAVISTA et al., 2012)(BETTINI et al., 2010)(KNAPPMEYER et al., 2013). No entanto a área da Internet das Coisas, como meio de prover a interação dos sistemas ubíquos com o ambiente, insere outros desafios de pesquisa, como: (i) heterogeneidade de dispositivos; (ii) escalabilidade; (ii) eficiência energética; (iii) auto-organização, e; (iv) interoperabilidade semântica (DELICATO; PIRES; BATISTA, 2013).

Considerando esses desafios de pesquisa, esta tese propõe o CoIoT, um middleware para Internet das Coisas focado na coleta e processamento das informações contextuais do

ambiente e a atuação remota sobre o mesmo. CoIoT é dirigido a eventos, gerenciado por regras, com processamento distribuído do contexto, capaz de agir de forma proativa tanto na captura de informações contextuais do ambiente físico, como na atuação remota sobre o mesmo.

O CoIoT foi concebido considerando as premissas do middleware EXEHDA (*Execution Environment for Highly Distributed Applications*) (LOPES et al., 2012). O EXEHDA provê uma arquitetura de software baseada em serviços que visa criar e gerenciar um ambiente ubíquo, bem como promover a execução das aplicações da UbiComp sob este ambiente. Os principais serviços fornecidos estão organizados em subsistemas relacionados a acesso ubíquo, comunicação, execução distribuída, reconhecimento do contexto e adaptação. Assim, esta tese visa contribuir com o Subsistema de Reconhecimento de Contexto e Adaptação do middleware EXEHDA, provendo recursos relacionados à coleta das informações contextuais do ambiente, atuação sobre o mesmo e o processamento dessas informações.

1.1 Questões de pesquisa e hipótese

O problema de pesquisa desta tese consiste em responder a seguinte questão:

> Como gerenciar os aspectos inerentes à elevada heterogeneidade das infraestruturas da Internet das Coisas e ao mesmo tempo oferecer mecanismos com alto nível de abstração para o tratamento das informações contextuais produzidas, bem como para intervir no ambiente, tendo em vista as demandas das aplicações da UbiComp?

A hipótese para este problema de pesquisa é a seguinte:

> Através de um tratamento distribuído da heterogeneidade de hardware e software, elevando-se gradativamente o nível de abstração dos dados coletados, é possível processar e prover acesso às informações contextuais a qualquer momento, independentemente da maneira com que foram produzidas.

Como desdobramento do problema de pesquisa, esta tese procura também responder as seguintes perguntas:

Questão 1: Qual modelo de processamento deve ser adotado no tratamento de contexto para atender às demandas das aplicações ubíquas na IoT?

Questão 2: Como deve ser o modelo arquitetural para tratar a heterogeneidade dos dispositivos da infraestrutura computacional típica da IoT?

Questão 3: Que estratégias de interoperação devem ser adotadas para atender às premissas operacionais utilizadas na IoT?

1.2 Objetivos

O objetivo central desta tese consiste na concepção de uma arquitetura para IoT capaz de realizar de forma distribuída tanto a coleta e processamento das informações contextuais, como a atuação remota no meio a fim de atender as aplicações da UbiComp.

Esta tese também contempla os seguintes objetivos específicos:

- revisar os principais conceitos e o estado da arte, na perspectiva da UbiComp, nos temas relacionados à Ciência de Contexto, à Internet das Coisas e ao processamento de eventos;
- conceber mecanismos de personalização do middleware para atender diferentes perfis de aplicações através de regras distribuídas e intercambiáveis dinamicamente;
- dar suporte à descoberta e gerenciamento de dispositivos de sensoriamento e atuação;
- gerenciar a heterogeneidade de hardware e software;
- propor estratégias que permitam controlar o fluxo de dados transmitidos entre dispositivos de rede;
- prover mecanismos para o gerenciamento de eventos distribuídos voltados aos cenários da IoT;
- tratar a interoperabilidade entre os diferentes módulos do middleware e destes com as aplicações através de interfaces padronizadas;
- tratar de forma distribuída as diferentes etapas de gerenciamento de contexto: coleta, processamento e atuação;
- validar as funcionalidades do middleware através de estudos de caso.

1.3 Identidade do projeto

O nome CoIoT surgiu da associação dos dois temas considerados os pilares da concepção do projeto: **Co***ntext* e **IoT**.

O símbolo do CoIoT, apresentado na Figura 1.1, remete à organização celular do ambiente ubíquo proposto quando da concepção do middleware EXEHDA. As abstrações utilizadas neste ambiente ubíquo são centrais para a solução proposta para o CoIoT.

Figura 1.1 – Identidade visual do CoIoT

1.4 Estrutura do texto

O texto desta tese está estruturado conforme é apresentado a seguir.

No capítulo 2 é apresentada uma revisão bibliográfica sobre os assuntos que embasaram esta tese, incluindo a fundamentação teórica sobre Computação Ubíqua, Ciência de Contexto, Internet das Coisas, Tratamento de Eventos e o middleware EXEHDA.

No capítulo 3 são apresentados os trabalhos relacionados a esta tese, bem como a discussão destes trabalhos em comparação ao CoIoT considerado as premissas que nortearam a concepção do mesmo.

No capítulo 4 é discutida a concepção da arquitetura do CoIoT, com destaque para a abordagem distribuída de processamento do contexto, o gerenciamento de eventos, o tratamento da heterogeneidade e a organização das funcionalidades propostas para a arquitetura.

No capítulo 5 são apresentados dois estudos de caso, desenvolvidos na área da agricultura, para avaliar a proposta do CoIoT. São destacados os aspectos característicos dos cenários de avaliação, os equipamentos e tecnologias utilizadas, os protótipos desenvolvidos, os testes realizados e os resultados obtidos.

Por fim, no capítulo 6 são apresentadas as conclusões desta tese com destaque para as principais contribuições da pesquisa e os trabalhos futuros.

2 COIOT: ESCOPO DE PESQUISA

Este capítulo tem por objetivo discorrer sobre as principais frentes de pesquisas consideradas na concepção do CoIoT. Na primeira seção são explorados os conceitos relativos à Ciência de Contexto. Na segunda seção é apresentada a fundamentação teórica da Internet das Coisas. Na terceira seção são abordados temas associados ao tratamento de eventos. Por fim, na quarta seção são apresentadas as características e funcionalidades do middleware EXEHDA, responsável por gerenciar o ambiente ubíquo utilizado neste trabalho.

2.1 Ciência de Contexto na perspectiva da ubiquidade

Mark Weiser, introduziu o termo Computação Ubíqua em seu visionário artigo "*The Computer for the 21st Century*" no qual apresenta uma visão de futuro da computação (WEISER, 1991). Weiser descreveu um cenário onde computadores estariam incorporados no cotidiano, de tal forma que a interação com as pessoas seria feita de maneira natural e imperceptível aos olhos de usuários leigos (CACERES; FRIDAY, 2012). Nesta perspectiva, a Computação Ubíqua tem como objetivo a criação de um ambiente focado no usuário e nas atividades que ele deseja realizar. Este conceito tem como premissa que o sistema irá executar tarefas de forma mais automatizada possível a partir de especificações do usuário, minimizando o envolvimento do mesmo com aspectos de manipulação e gerência da infraestrutura computacional. Alguns autores definem que a UbiComp consiste em disponibilizar ao usuário os serviços computacionais de seu interesse da forma transparente, através de diferentes dispositivos e redes, em qualquer lugar e a qualquer momento (LOPES et al., 2014)(COSTA; YAMIN; GEYER, 2008)(KRUMM et al., 2010).

A Computação Ubíqua tem sido considerada por alguns autores como a convergência de diversas áreas da Ciência da Computação, entre as quais uma das mais evidentes é a área de sistemas distribuídos (COSTA; YAMIN; GEYER, 2008). Porém, na premissa clássica da computação distribuída, busca-se liberar o programador das preocupações em relação ao meio no qual ocorre o processamento, enquanto que, na Computação Ubíqua, o programador deve considerar as informações de contexto no desenvolvimento das aplicações, criando sistemas capazes de reagirem a mudanças no ambiente operacional.

Dentre os principais desafios herdados pelos sistemas ubíquos dos sistemas distribuídos destacam-se: (i) escalabilidade - número potencial de usuários atendidos; (ii) heterogeneidade - necessidade de atender diferentes equipamentos e redes móveis; e (iii) dinamismo, devido

à impossibilidade de se garantir a conectividade e disponibilidade dos recursos e serviços de forma permanente.

Assim, o ambiente computacional ubíquo é heterogêneo e dinâmico, composto de aplicações capazes de reagirem às mudanças de contexto, disponibilidade de recursos, políticas e perfis dos usuários. Considerando isto, os programadores necessitam de abstrações de alto nível para reduzir o esforço de programação decorrente da complexidade quando no desenvolvimento de aplicações para ambientes ubíquos.

Um dos desafios centrais na Computação Ubíqua é a busca pela integração dos seus serviços com os aspectos do ambiente (CACERES; FRIDAY, 2012). Isso tem surgido com a intenção de viabilizar a reação das aplicações do usuário aos seus contextos de interesse (KRUMM et al., 2010). Esse modelo de computação, conhecido como computação Ciente de Contexto, vem sendo considerado um dos pilares da UbiComp. As aplicações Cientes do Contexto têm como premissa a constante troca de informações entre si e com o meio a fim de manter conhecimento sobre o ambiente (ALEGRE; AUGUSTO; CLARK, 2016)(KNAPPMEYER et al., 2013).

Uma das definições de contexto mais referenciadas na literatura de Computação Ubíqua foi introduzida por Dey. Em (DEY, 2001), é afirmado que contexto é qualquer informação que pode ser usada para caracterizar uma situação de uma entidade. Sendo que essa entidade pode ser uma pessoa, um lugar, ou um objeto considerado relevante para a interação entre um usuário e uma aplicação, incluindo o próprio usuário e a própria aplicação. Esta definição é abrangente quanto aos tipos de dados, porém, é considerada suficiente para avaliar se uma determinada informação é um elemento de contexto (PERERA et al., 2014). Em (YAMIN et al., 2003), Yamin define contexto como toda informação que pode ser obtida do sistema e que é relevante para a aplicação. Isso significa que o contexto pode focar em alguns aspectos e não priorizar outros, dependendo dos interesses envolvidos na aplicação ubíqua.

A revisão de literatura aponta que a construção do suporte à Ciência de Contexto para as aplicações ubíquas apresenta diversos desafios, dentre eles: (*i*) a aquisição de informações contextuais a partir de fontes heterogêneas e distribuídas; (*ii*) o tratamento das informações de contexto adquiridas e a respectiva atuação sobre o meio físico; e (*iii*) a disseminação dessas informações aos consumidores interessados de forma distribuída e no momento oportuno (ALEGRE; AUGUSTO; CLARK, 2016)(BELLAVISTA et al., 2012) (KNAPPMEYER et al., 2013) (BETTINI et al., 2010).

2.1.1 Aquisição de informações contextuais

A aquisição das informações contextuais é feita através de um conjunto de sensores. 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. São eles:

- *Sensores físicos*: este é o tipo de sensor mais utilizado. Estes dispositivos tem a função de captar grandezas físicas diretamente do ambiente, normalmente gerando sinais elétricos proporcionais a essas grandezas para que possam ser lidos por sistemas externos. Como exemplos típicos desse tipo de sensor têm-se: sensor de luz, movimento, localização, temperatura, umidade, entre outros.

- *Sensores lógicos (ou virtuais)*: neste tipo de sensor as informações de contextuais são provenientes de softwares ou serviços. Essas informações podem ser geradas através da combinação de dados provenientes de diversas fontes, como sensores físicos 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. Outro exemplo seria o acesso a informações climáticas oriundas de serviços web disponibilizadas por estações meteorológicas através da Internet.

Em ambientes ubíquos a quantidade de dispositivos pode atingir números elevados (KRUMM et al., 2010). Portanto, o desenvolvimento de sistemas computacionais para esses cenários 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. (2014), 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, são elas:

- *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 quanto à frequência de aquisição, a qual pode ser dos seguintes tipos:

- *Instantânea*: esse tipo de frequência de coleta é disparado 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 em que 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 as 19h00min. 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*.

2.1.2 Processamento do contexto

O processamento do contexto visa possibilitar a compreensão dos contextos de interesse através da geração de informações mais significativas (BIBRI, 2015). 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 processo de interpretação de contexto consiste na manipulação e refinamento das informações contextuais de um ambiente. O processo de interpretação de contexto pode ser

direto, como derivar o nome de uma rua a partir de suas coordenadas geográficas, ou complexo e oneroso, como inferir o humor de um usuário baseado em seu perfil e na atividade em que ele está realizando. Além disso, o ambiente da Computação Ubíqua é extremamente dinâmico e as informações contextuais podem estar distribuídas em diferentes lugares ou produzidas por dispositivos com alto grau de mobilidade. Essa complexidade faz com que exista a necessidade de um suporte computacional às aplicações, de maneira a auxiliá-las na realização de interpretações de contextos (BIBRI, 2015)(ALEGRE; AUGUSTO; CLARK, 2016).

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 informações contextuais em diferentes níveis de abstração, conforme a necessidade 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)(ALEGRE; AUGUSTO; CLARK, 2016).

O raciocínio é utilizado para verificar a Ciência 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 informações de contexto é um requisito ligado à aquisição de informações de contexto, bem como à disponibilidade contínua dos componentes de captura de informações de contexto. Um histórico de contexto pode ser utilizado para estabelecer tendências e predizer valores futuros de informações contexto. Sem o armazenamento dessas informações, análises desse tipo não são possíveis de realizar.

2.1.3 Ciência de situação

Situação pode ser considerada como uma abstração em alto nível de dados contextuais, que define um estado de uma entidade em um determinado intervalo de tempo (FLEISCHMANN, 2012).

A entidade pode ser uma pessoa, um lugar ou um objeto, todos relevantes para a interação entre o usuário e a aplicação. Situação é uma abstração oriunda nos eventos do mundo real, sendo derivada de contextos e hipóteses e da forma como os contextos observados estão relacionados aos fatos de interesse do projetista da aplicação. As situações dão significado aos dados aquisitados pelos sensores os quais podem ser explorados pelas aplicações.

Uma das definições de ciência de situação bastante aceita na área da computação foi proposta por Endsley (ENDSLEY, 1995). Para este autor, a ciência de situação corresponde à percepção de um ou mais elementos do ambiente, a compreensão do seu significado e a projeção de seu possível estado em um futuro próximo.

A ciência de situação é uma forma promissora de resolver as limitações relacionadas aos modelos tradicionais de representação de contexto, e que pode auxiliar a desenvolver aplicações mais robustas e capazes de melhor atenderem às necessidades dos usuários. Além disso, o uso de situações também proporciona uma representação dos dados sensoreados potencialmente mais compreensível pelos humanos (YE; DOBSON; MCKEEVER, 2012).

A identificação e a decorrente reação às diversas situações que ocorrem no ambiente, provenientes de processamento de contexto, é uma atividade complexa para os sistemas computacionais. É necessário que as aplicações considerem a relação entre as diferentes situações, identificando, por exemplo, as que não podem ocorrer ao mesmo tempo (como escrever e tomar banho) e a coerência em sua ordem de ocorrência (como acordar e imediatamente ir correr). Além disto, outros aspectos como a imprecisão inerente dos dados provenientes dos sensores e as limitações das regras de inferência, também contribuem para a complexidade da detecção de situações, uma vez que estes aspectos introduzem fatores que, em diversos casos, podem ser de difícil tratamento. Além disso, muitas vezes existe a necessidade do emprego de sensores de diferentes naturezas, o que eleva a complexidade na interpretação de dados aquisitados.

2.2 Internet das Coisas

A Internet das Coisas (*Internet of Things* - IoT) é um paradigma que vem ganhando destaque atualmente sobretudo devido ao interesse que tem despertado nas grandes empresas de tecnologia. Por ser uma área nova, possui muitos problemas de pesquisa em aberto e isso tem estimulado também 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 por suprir a carência da UbiComp em relação aos aspectos infraestruturais (CACERES; FRIDAY, 2012).

A IoT aponta para um cenário no qual objetos inteligentes ou coisas (como tags RFID, sensores, atuadores, etc.) possuem a capacidade de se comunicar e transferir dados através da Internet com o mínimo de intervenção humana. Utensílios, sistemas de transportes, redes de energia, equipamentos pessoais, sistemas agrícolas e até o nosso corpo são ambientes em potencial para a IoT, podendo serem populados por sensores capazes de gerar e compartilhar informações automaticamente com sistemas computacionais externos.

Com essa característica, a IoT vem se mostrando elemento essencial para consolidar a integração entre os sistemas computacionais e o ambiente físico, com potencial para produzir informações contextuais em massa (ATZORI; IERA; MORABITO, 2010) (PERERA et al., 2014) (SAINT-EXUPERY, 2009).

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).

Mais tarde, em 2001 o termo é mencionado no artigo intitulado *“The Electronic Product Code (EPC): A naming Scheme for Physical Objects”* (BROCK, 2001) no qual o autor faz a seguinte descrição: *“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”*. Neste artigo, o autor propõe a substituição do popular código de barras por um sistema de etiquetas (RFIDs) que usufruiria das capacidades da Internet e da globalização.

Ao longo dos últimos anos vem se observando um constante crescimento do número

de dispositivos conectados na Internet. De acordo com pesquisa do Cisco *Internet Business Solutions Group* (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 dispositivos por pessoa. Porém nos anos seguintes teve um elevado crescimento no uso de *smartphones* e *tablets* contribuindo para que o número de dispositivos conectados à Internet aumentasse para 12,5 bilhões em 2010, quando a população mundial atingiu 6,8 bilhões de pessoas, resultando na relação de aproximadamente 1,84 dispositivos conectados por pessoa. Segundo o Cisco IBSG, entre 2008 e 2009 estima-se que ocorreu o período em que número de dispositivos por pessoa tenha ultrapassado a 1 pela primeira vez na história. Para Cisco IBSG, esse é o momento em que a Internet das Coisas começou a acontecer efetivamente.

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 *Consumer Electronics Show 2015* (CES), denotando que a Internet das Coisas além de ser uma realidade, vem ganhando destaque e está sendo considerada a revolução tecnológica que aponta para o futuro da relação entre as infraestruturas de computação e comunicação.

Na literatura não existe uma definição única para Internet das Coisas, segundo (PERERA et al., 2014) isto ocorre principalmente por se tratar de área de pesquisa recente e bastante ampla. Porém, considerando o escopo de pesquisa desta tese, a definição apresentada por (SAINT-EXUPERY, 2009) apresenta-se a mais oportuna. Segundo Saint-Exupery, a Internet das Coisas consiste em um ambiente em que objetos podem se conectar a qualquer momento, em qualquer lugar, com qualquer coisa, de preferência usando qualquer caminho ou rede e qualquer serviço (vide Figura 2.1). Porém, 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).

Na Internet das Coisas o conceito de "coisa" é fundamental. Considerando o escopo desse trabalho, a definição dada por Miorandi et al. (2012) apresenta-se oportuna. Segundo o autor, objetos inteligentes ou simplesmente "coisas" devem ser capazes de detectar fenômenos (físicos ou lógicos) e/ou disparar ações sobre o ambiente, possuir alguma capacidade computacional, possuir capacidade de comunicação e ter um identificador único.

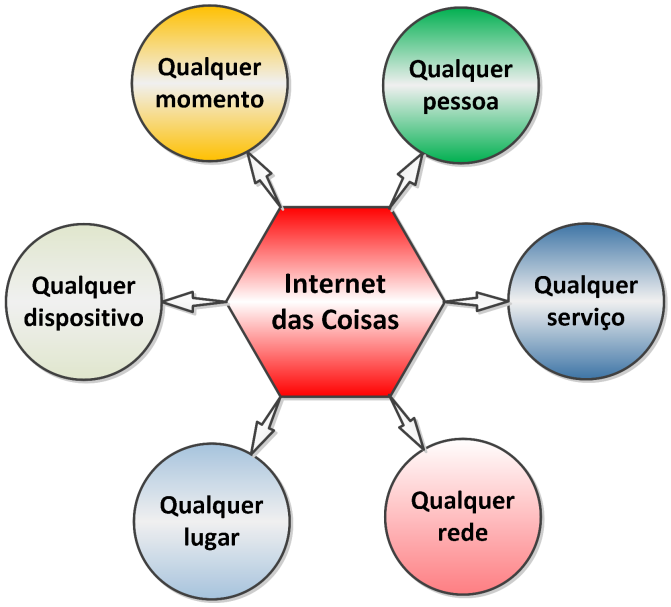

Figura 2.1 – Definição de Internet das Coisas (SAINT-EXUPERY, 2009)

2.2.1 Desafios da IoT

Segundo a revisão da literatura da área, os principais desafios que devem ser enfrentados no desenvolvimento de sistemas de suporte para Internet das Coisas, considerando o foco de pesquisa desta tese, são os seguintes (PIRES et al., 2015)(DELICATO; PIRES; BATISTA, 2013)(GUBBI et al., 2013)(MIORANDI et al., 2012):

Heterogeneidade de dispositivos

Uma das características centrais da Internet das Coisas refere-se à grande heterogeneidade de dispositivos físicos que podem estar presentes no ambiente de interesse das aplicações dos usuários. Essa heterogeneidade diz respeito às diferentes capacidades e recursos tecnológicos disponíveis em cada dispositivo, sejam estes de software ou de hardware. Como em cenários da IoT o número de dispositivos pode ser muito elevado, não é desejável que o desenvolvedor das aplicações ubíquas se envolva no tratamento dos aspectos tecnológicos de baixo nível associados. Portanto, as soluções de middleware devem tratar a heterogeneidade e permitir a interoperabilidade e integração dos dispositivos e serviços que fazem parte do ambiente. O desafio nesse caso reside em gerenciar os aspectos de baixo nível pertinentes aos dispositivos da IoT, e ao mesmo tempo suprir as demandas de alto nível das aplicações da UbiComp (DELICATO; PIRES; BATISTA, 2013).

Escalabilidade

O número de dispositivos envolvidos assim como a quantidade de informações produzidas pode ser muito elevado. Isso aponta para problemas de escalabilidade que envolvem diferentes naturezas (MIORANDI et al., 2012), incluindo: (*i*) nomeação e endereçamento, devido ao grande número de dispositivos envolvidos; (*ii*) comunicação, pela quantidade de dados trocados na rede; (*iii*) tratamento de grande volume de dados e; (iv) gerenciamento de serviços, devido ao número de execuções potencialmente elevado.

Eficiência energética

Muitos dispositivos da IoT podem operar com baterias, como nodos sensores por exemplo. Isso significa que estes dispositivos possuem uma quantidade de energia limitada e seu tempo de operação muitas vezes está associado à durabilidade dessa energia. Por isso, o uso eficiente da energia é essencial para prolongar ao máximo a vida útil destes dispositivos, uma vez que nem sempre trocar ou recarregar a bateria é uma tarefa simples. Nesse sentido, as estratégias de redução do consumo de energia geralmente estão associadas à otimização do uso das interfaces de comunicações sem fio e do processamento. Por sua vez, independentemente do uso de baterias, o elevado número de dispositivos usualmente contemplados em abordagens IoT remete para que as soluções neste cenário tenham como premissa geral o cuidado com o consumo energético.

Auto-organização

A complexidade e o dinamismo são características típicas dos ambientes da IoT. No caso de redes de sensores, por exemplo, os dispositivos estão constantemente entrando e saindo da rede, seja por decorrência de término de energia, perda de sinal de rádio ou pela inserção de novo dispositivo, devido à reposição por avaria ou necessidade de adicionar outro ponto de monitoramento (YICK; MUKHERJEE; GHOSAL, 2008). Devido ao elevado número de dispositivos que poderão estar envolvidos é desejável que se tenha o mínimo de intervenção humana nos procedimentos de gerência. Por isso, a utilização de mecanismos de descoberta de dispositivos e serviços é essencial nestes cenários.

Interoperabilidade semântica e gerenciamento de dados

Na Internet das Coisas o volume de dados trocados que necessita de algum tratamento é potencialmente muito elevado. Para que esses dados resultem em informações úteis que possam ser utilizadas por diferentes aplicações é necessário disponibilizá-los em formatos adequados e padronizados, fornecer modelos e descrições semânticas do seu conteúdo (meta-dados), usando formatos e linguagens bem definidos. Isso permite que sejam utilizados métodos de raciocínio sobre os dados produzidos (MIORANDI et al., 2012).

2.2.2 Computação em Fog

Na Internet das Coisas os dispositivos computacionais são amplamente distribuídos no ambiente, onde coletam informações contextuais e produzem atuações. As aplicações que utilizam as informações produzidas em muitos casos necessitam tomar decisões rapidamente e com a maior confiabilidade possível. Na literatura, muitas das soluções propostas realizam a coleta e atuação de maneira distribuída porém o processamento das informações coletadas se dá de maneira remota em algum servidor central acessível através da Internet, como o caso de soluções baseadas em *Cloud Computing*. Esse tipo de abordagem de gerenciamento dificilmente poderá satisfazer as necessidades de baixa latência, ciência de localização e confiabilidade típicos dos ambientes da IoT (STOJMENOVIC; WEN, 2014). *Fog Computing*, ou computação em nevoeiro, surgiu de maneira a enfrentar esses desafios.

A *Fog Computing* é implementada nas bordas computacionais e, de certa forma, considerada uma *Cloud* mais próxima do "chão". Além disto, a Fog considera o armazenamento e o processamento dos dados nas bordas computacionais, levando a tomada de decisões para próximo de onde os dados são coletados. Abordagens baseadas em Fog têm potencial para lidar com a coleta de dados e atuação altamente distribuídas, como é típico nos ambientes da IoT (AAZAM; HUH, 2014). Evidentemente, para que sejam implementadas infraestruturas baseadas em *Fog Computing*, os dispositivos devem possuir alguma capacidade computacional e de armazenamento para que possam lidar com as requisições dos usuários locais ou remotos.

Principais Características

A seguir são apresentadas algumas das principais características da *Fog Computing*, as quais a torna uma abordagem apropriada para tratar alguns dos desafios da Internet de Coisas (BONOMI et al., 2012) (HONG et al., 2013):

- *Proximidade da borda*: disponibiliza os dados e serviços mais próximos dos consumidores (usuários finais, processadores de contexto, etc.), promovendo a diminuição da latência no acesso a estes. Isto é necessário para promover uma certa independência em relação à algum servidor remoto e viabilizar a operação local;
- *Organização hierárquica*: para prover baixa latência e alta escalabilidade a Fog deve ser organizada de forma hierárquica a partir do topo, de forma que o gerenciamento possa ser feito pelas camadas superiores;
- *Ciência de localização*: os serviços devem ter informações quanto a sua localização e de seus vizinhos com os quais deve compartilhar tarefas e dados;
- *Economia de recursos da rede*: com a tomada de decisão local, os dispositivos podem repassar ou não repassar as informações para as camadas superiores, economizando assim os recursos de rede e, potencialmente, a energia dos dispositivos;
- *Escalabilidade*: na Fog existe uma elevada distribuição de dispositivos computacionais para viabilizar a descentralização dos serviços. Essa elevada distribuição geográfica com processamento distribuído minimiza a quantidade de dados enviados pela rede. Essas características possibilitam a utilização uma grande quantidade de dispositivos, bem como gerenciar a ampliação gradativa do seu número;
- *Aplicações em tempo real*: a baixa latência promovida pela Fog permite que aplicações de tempo real se beneficiem do modelo uma vez que a presença de dados e serviços próximos é mais favorável que o uso de abordagens centralizadas.

Tratamento do contexto na perspectiva da Fog Computing

A proposta da *Fog Computing* não tem a intenção de substituir outros modelos baseados em processamento centralizado, como a *Cloud Computing*, mas sim, estendê-los para que possam ser utilizados em conjunto em aplicações amplamente distribuídas, como as da IoT.

Na perspectiva da *Fog Computing* a computação ocorre de maneira distribuída, possibilitando obter uma otimização no resultado da análise dos dados contextuais e, também, de se utilizar os recursos do ambiente da melhor maneira possível. Desta forma, deve-se planejar a arquitetura e as aplicações para que estas possam enviar os dados contextuais aos locais adequados, de acordo com a sua necessidade.

Informações contextuais críticas, que não possuem a necessidade de uma análise histórica, mas que exigem um tratamento rápido e com minimização de falhas, devem ser

tratadas quando da sua ocorrência. Para isso esses dados deverão ser processados nas bordas, podendo esta análise envolver dados oriundos de diferentes dispositivos *Fog*. Posteriormente, os dados processados podem ser enviados a um servidor central para permitir outra etapa de processamento ou armazenamento histórico.

Os tratamentos contextuais que necessitam de dados históricos ou que utilizem algoritmos que exijam elevado poder computacional devem priorizar os servidores centralizados ou a *Cloud* por estas abordagens terem por premissa a disponibilização de dispositivos computacionais com elevada capacidade de processamento e armazenamento.

Esses mesmos servidores também podem ter como atribuições gerenciar a alteração das regras de processamento dos dispositivos *Fog* sob demanda, de acordo com as necessidades das aplicações ou dos usuários.

2.3 Sistemas Distribuídos Baseados em Eventos

Na vida cotidiana, eventos referem-se a situações importantes aos destinatários e que caracterizam uma efetiva situação de mudança, ou seja, que produzam uma alteração significativa em contraste a uma situação anterior e que, geralmente exigem uma reação apropriada. Uma campainha tocando, um semáforo passando para verde e começar a chover são comumente referidos como eventos, os quais podem provocar reações como abrir a porta, partir o carro e abrir o guarda-chuvas, respectivamente.

Na Ciência da Computação eventos assumem um caráter importante no gerenciamento dos sistemas sendo aplicados a diversos domínios. Em Sistemas Operacionais, eventos estão presentes nas interrupções do hardware, sinais entre processos concorrentes, *time-outs* do escalonador; em Banco de Dados eventos são usados para monitorar operações relevantes sobre os dados, incorporar restrições de integridade, implementar estratégias de alerta; eventos em Linguagens de Programação são usados para tratar as interações com o usuário, para indicar o final de um cálculo matemático, entre outros.

Em um ambiente ubíquo, diferentes tipos de eventos acontecem a todo o momento conforme as mudanças de estado das variáveis contextuais. Dependendo do evento ocorrido em um dado instante de tempo, uma nova situação pode ser configurada. O que determina as condições para ocorrência de um evento são as ações executadas sobre o ambiente de interesse (FLEISCHMANN, 2012). Os eventos gerados a partir dos contextos de interesse devem ser interceptados pelo sistema de gerência e notificações devem ser enviadas às aplicações para que as mesmas possam dar o tratamento adequado (PERERA et al., 2014).

Na Internet das Coisas, o uso de eventos pode ocorrer tanto na coleta das informações contextuais do ambiente quanto no tratamento dessas informações. Esse tipo de abordagem permite o tratamento das informações contextuais no momento da sua ocorrência o que permite uma resposta imediata a situação detectada. Além disso, o uso de abordagens baseadas em eventos permitem implementar métodos de coleta do tipo *push* o que contribui para a redução do volume de dados enviados pela rede.

A utilização de um modelo de processamento distribuído de eventos também possibilita um ganho significativo em escalabilidade e em confiabilidade. Os eventos que ocorrem em uma região específica podem ser tratados localmente ou sofrerem algum processo de filtragem e agregação antes de serem enviados ao seu destino ou a serviços intermediários responsáveis pelo processamento das informações contextuais produzidas. Desta forma, o processamento de eventos pode ser visto como uma condição para a materialização da *Fog Computing*.

A literatura apresenta diversas definições para evento. Uma definição bastante aceita caracteriza evento como uma ocorrência dentro de um sistema ou domínio em particular, algo que aconteceu, ou que poderá acontecer (ETZION; NIBLETT, 2010). Essa ocorrência poderá estar associada a um tempo específico, ou então, pode considerar um intervalo de tempo e assumir uma abstração mais elevada representando alguma alteração significativa para este ambiente (MACEDO; MARINHO; SANTOS, 2015). Por exemplo, a aterrissagem de um avião, que é produzida por um conjunto de outros eventos que ocorreram em um determinado intervalo de tempo (o piloto diminuiu a altitude, reduziu a velocidade, etc.)

Podemos, portanto, caracterizar duas categorias de eventos observados em um ambiente IoT:

- *Evento Primitivo*: refere-se a uma ocorrência instantânea, atômica de um acontecimento de interesse em um determinado instante de tempo. Ou seja, um evento ocorrido em um tempo t e outro em $t + p$ são considerados eventos de instâncias separadas. Como exemplos desse tipo de evento tem-se abertura de uma porta, o acender das luzes, etc. (TERFLOTH, 2009).

- *Evento Composto*: Consiste na combinação de outros eventos, primitivos ou compostos, ocorridos em um determinado intervalo de tempo, representando uma abstração de mais alto nível. Nesse caso um evento composto tem uma duração mínima de um tempo p, onde um evento que ocorreu num instante t e $t + p$ não pode ser separado (TERFLOTH, 2009) (PERERA et al., 2014).

2.3.1 Sistemas de processamento de eventos

Os sistemas de processamento de eventos identificados na literatura podem ser divididos em dois grandes grupos: (i) Sistemas Baseados em Regras ou Especificação e; (ii) Sistemas Baseados em Aprendizado (ETZION; NIBLETT, 2010). Os Sistemas Baseados em Regras ou Especificação são os tipos mais comuns. Eles apresentam uma grande capacidade de lidar com eventos e, por seus eventos serem explícitos via regras, possuem um requisito de poder computacional menor do que sistemas baseados em aprendizado, sendo ideais para os sistemas embarcados da atualidade. Já os Sistemas Baseados em Aprendizado necessitam de um poder computacional maior para operarem, desta forma há uma necessidade de hardware para detecção de eventos mais complexos. Normalmente, sistemas deste tipo são aplicados em um dispositivo específico da rede selecionado especialmente para essa finalidade.

Segundo (PERERA et al., 2014), sistemas baseados em regras também são os mais populares mecanismos de processamento de contexto presentes na literatura, sendo utilizados também como estratégia de raciocínio. Regras tem um grande potencial de uso no tratamento do contexto, pois consistem em método simples de representar tanto o pensamento humano quanto o raciocínio de máquina (PERERA et al., 2014). Isso simplifica a especificação de regras de processamento contexto para a geração de informações contextuais de alto nível a partir de dados contextuais de baixo nível.

Segundo (TERFLOTH, 2009) e (ETZION; NIBLETT, 2010), os sistemas baseados em regras podem adotar basicamente dois tipos de regras: (i) regras ativas ou (ii) regras de produção. As regras ativas (regras ECA - Evento-Condição-Ação) são disparadas a partir de um evento. Quando um evento ocorre, uma sentença é avaliada e, se ela for verdadeira, uma ação é executada. Sua sintaxe é dada por: `ON <evento> IF <condição> THEN <ação>`. As regras de produção, por sua vez, têm por princípio a reação às mudanças de estado das variáveis de contexto. Toda vez que uma condição avaliada é satisfeita uma ação é executada. Sintaticamente elas obedecem a sentença do tipo `IF <condição> THEN <ação>` e, ao contrário das regras ECA, não necessitam nenhum evento de ativação, sendo executadas sequencialmente. Em sistemas cientes de contexto, novas situações são caracterizadas a partir das mudanças de estado de uma ou mais variáveis contextuais. Essas mudanças de estado podem ser representadas como eventos do ambiente. Portanto, fica evidente que as regras tipo ECA, por serem disparadas por eventos, são as mais indicadas no tratamento de contexto, pois permite que o processamento ocorra somente no momento em que há potencialmente a ocorrência de uma nova situação.

2.3.2 Processamento temporal de eventos

O comportamento humano, assim como o de outros elementos da natureza passíveis de sensoriamento, é inerentemente espacial e temporal. Existem situações em que um evento deve ser avaliado apenas se ele ocorrer dentro de certo intervalo de tempo ou sequência, podendo ser ignorado nas demais circunstâncias. Portanto, o gerenciamento de ocorrências temporais é fundamental para o tratamento de eventos.

Quando os eventos a serem tratados estão associados a ocorrências temporais, frequentemente as regras de tratamento são válidas apenas dentro de certos intervalos de tempo. Esses intervalos são chamados janelas temporais. As janelas temporais são responsáveis pela detecção e tratamento de eventos que ocorrem apenas dentro delas. Estas janelas temporais podem se sobrepor desde que as regras para detecção assim determinem, desta forma a instância de um evento pertence a janela na qual este foi detectado, podendo porém ser compartilhado com outras janelas (ETZION; NIBLETT, 2010)(HINZE; BUCHMANN, 2010).

Os diferentes tipos de janelas temporais são determinados pele seu intervalo e a forma de abertura e fechamento. O início e fim de cada janela temporal podem ser definidos a partir de valores de tempo ou estarem associados a eventos e tratados através de regras (ETZION; NIBLETT, 2010). Os principais tipos de janelas temporais são apresentados a seguir:

- *Janela Fixa*: No modelo de janela fixa cada janela possui um período fixo de duração, podendo haver apenas uma janela aberta por vez, sem sobreposição. A janela pode se repetir em intervalos de tempo determinado (vide Figura 2.2 A e B). Exemplo: Monitoramento de um paciente no pós-operatório onde há uma data inicial e uma data final para observação;

- *Janela de Intervalo de Eventos*: Este tipo de janela é aberta ou fechada após o tratador de eventos receber um determinado evento. Desta forma não há previsão de quando a janela pode ser aberta ou fechada. Exemplo: Ao ser detectada a temperatura alta de um paciente abre-se uma janela e, em caso de detecção de arritmia dispara-se um alerta para o médico. Caso não seja detectado o evento posterior dentro da mesma janela, a mesma é descartada (vide Figura 2.2 C e D);

- *Janela Deslizante com Intervalo Fixo*: Essas janelas são abertas periodicamente em intervalos de tempo regulares, mas não são vinculadas a horários específicos. Em vez disso cada janela é aberta depois de transcorrido um determinado tempo de sua antecessora. A janela pode ter um tamanho fixo com um intervalo de tempo especificado

(Figura 2.2E) ou ser fechada apóos a ocorrência de uma certo número de um evento específico (Figura 2.2F).

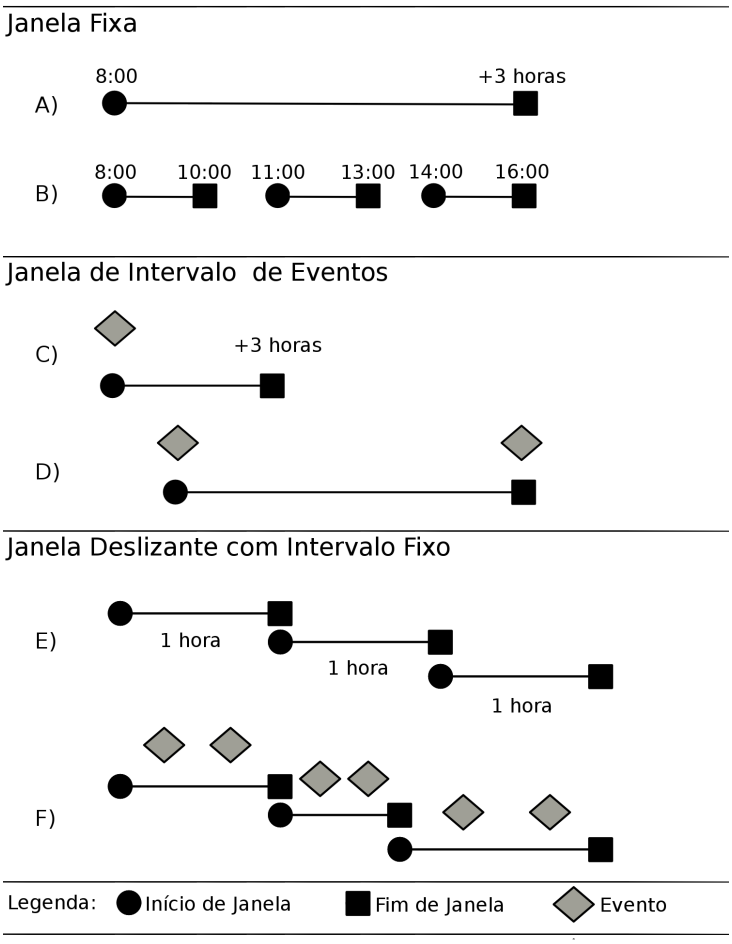

Figura 2.2 – Janelas de Tempo em processamento de eventos. Adaptado de (ETZION; NIBLETT, 2010).

2.4 Middleware EXEHDA

O middleware EXEHDA (LOPES et al., 2014) (YAMIN, 2004) foi concebido com o objetivo de prover suporte à execução das aplicações da UbiComp. Sua arquitetura, baseada em serviços, 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 2.3, tem como objetivo fornecer 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 Ciência de Contexto tem destaque na arquitetura, uma vez que permeia os dos demais componentes. Esse módulo tem por finalidade gerenciar os contextos de interesse das aplicações e do próprio ambiente de execução.

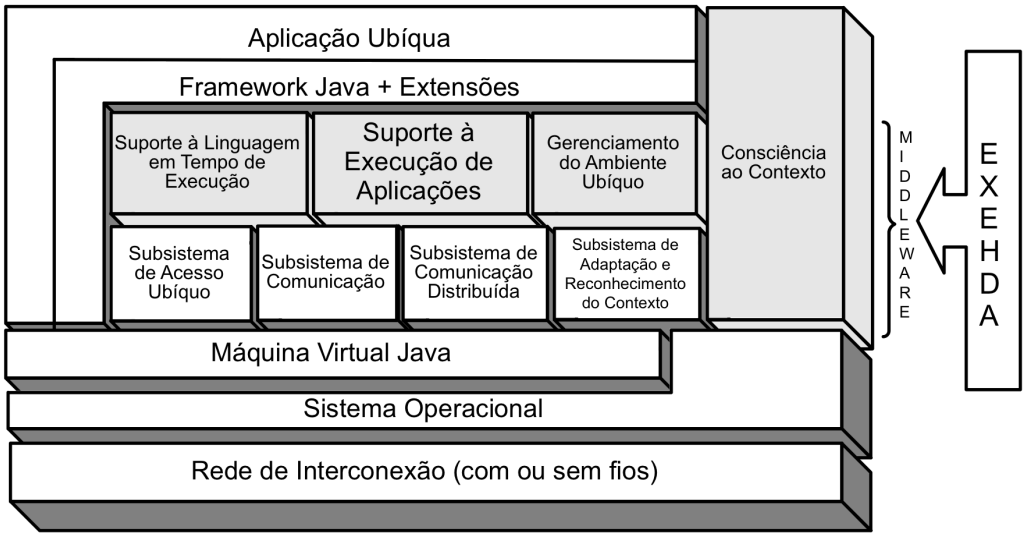

Figura 2.3 – Arquitetura do EXEHDA

Os requisitos de funcionamento em um ambiente de alta heterogeneidade, em que os recursos de hardware e disponibilidade de software em cada dispositivo podem variar, têm motivado a utilização de serviços plugáveis. Em cada nodo, um perfil de execução define a política de carregamento que será aplicada para cada um dos serviços do middleware. As políticas de carregamento são duas: (*i*) no boot, o que significa que o serviço deve ser carregado no processo de inicialização do nodo; ou (*ii*) sob demanda, o que significa que o serviço será carregado em seu primeiro uso.

O núcleo mínimo do EXEHDA gerencia a política de carregamento dos serviços e deve estar operacional em cada nodo que compõe o ambiente ubíquo. Usando esse recurso, podemos configurar o que é necessário e quando ele deve ser carregado. Para tanto dois serviços do núcleo mínimo devem estar sempre presentes: (*i*) *Profile Manager*, no carregamento da interpretação dos perfis de execução, faz com que esses perfis estejam disponíveis em tempo de execução para os demais serviços de middleware; (*ii*) *Service Manager*, que ativa serviços em um nodo

com base nas informações fornecidas pelo *Profile Manager*. O código de serviço é carregado sob demanda a partir do repositório de serviços, que pode ser local ou remoto, dependendo da capacidade de armazenamento do dispositivo e da natureza do serviço a ser carregado.

EXEHDA tem como requisito permanecer operacional durante os períodos de desconexão planejada. Para dar suporte a esse recurso, podemos dividir os serviços em duas partes, uma instância de nodo e uma instância de celular. O primeiro é o local para cada dispositivo, enquanto que o último executa no nodo base. Assim, o dispositivo local poderá estar operacional durante o desligamento planejado, considerando que a instância do nodo do serviço deve renunciar temporariamente ao acesso aos recursos que estão na rede. Por outro lado, a instância celular do serviço, em execução no nodo base da célula, atua como um ponto de referência para serviços que exigem procedimentos de coordenação distribuídos, inter-nodos ou inter-células (AUGUSTIN; YAMIN; GEYER, 2005) (LOPES et al., 2014).

2.5 Considerações sobre o capítulo

Este capítulo apresentou os esforços realizados durante a revisão de conceitos, destacando aspectos dos principais temas que constituem o foco de pesquisa desta tese, especificamente, a Ciência de Contexto na perspectiva da UbiComp, a Internet das Coisas, a computação em Fog, os sistemas distribuídos baseados em eventos e os aspectos arquiteturais do middleware EXEHDA.

A revisão conceitual proporcionou identificar que as aplicações ubíquas devem reagir às mudanças de estado das variáveis contextuais a fim de oferecer serviços compatíveis com os desejos dos usuários. A aquisição dessas variáveis contextuais do ambiente tem sido potencializada pela popularização da Internet das Coisas.

A revisão de literatura também aponta que os desafios enfrentados no suporte à Ciência de Contexto envolvem a coleta de informações contextuais a partir de fontes heterogêneas e distribuídas, o processamento dessas informações e a respectiva atuação sobre o meio físico. Porém, ao considerar a IoT como forma de prover a interação dos sistemas computacionais com o ambiente, novos desafios são inseridos, como heterogeneidade de dispositivos, escalabilidade, eficiência energética, auto-organização e interoperabilidade. Para enfrentar esses desafios apresentados o estudo da literatura apontou que o uso de técnicas de computação em Fog, bem como a adoção de estratégias de processamento de eventos distribuídos mostram-se oportunas.

Tendo em vista que o esforço de concepção do middleware CoIoT tem por origem o middleware EXEHDA, podendo ser considerado como uma evolução deste, este capítulo

apresentou também um resumo das principais características e funcionalidades do EXEHDA.

O próximo capítulo apresenta a revisão de literatura realizada, na qual estão contemplados os projetos identificados com maior proximidade da proposta do middleware CoIoT. Nessa revisão procurou-se caracterizar os principais aspectos que diferenciam o middleware CoIoT dos projetos estudados.

3 TRABALHOS RELACIONADOS

Através de uma revisão da literatura da área e tendo como base os objetivos de pesquisa desta tese, se buscou identificar alguns trabalhos relacionados ao middleware do CoIoT, dentre os quais foram selecionados os seguintes: EcoDiF (PIRES et al., 2014), Xively (LOGMEIN, 2015), Carriots (CARRIOTS, 2016) e LinkSmart (KOSTELNÍK; SARNOVSKÝ; FURDÍK, 2011).

A seleção dos trabalhos relacionados considerados neste capítulo partiu de um montante inicial de sessenta e dois trabalhos, e empregou como critérios os seguintes aspectos: (i) possuir uma arquitetura bem definida; (ii) tratar sensores e atuadores; (iii) tratar a heterogeneidade; (iv) suportar o tratamento de eventos; (v) prover Ciência de Contexto; e (vi) dispor de interfaces interoperáveis.

Neste capítulo é feita uma revisão das principais características e funcionalidades dos trabalhos que foram selecionados, sendo feita uma comparação dos mesmos com o middleware CoIoT. Esta comparação teve como base as principais características destes trabalhos, bem como os objetivos de concepção do CoIoT apresentados na seção 1.2.

3.1 EcoDiF

Ecossistema Web de Dispositivos Físicos (EcoDiF) (PIRES et al., 2014) é uma proposta de middleware para Internet das Coisas desenvolvido pelo grupo de pesquisa em UbiComp da Universidade Federal do Rio de Janeiro. EcoDiF integra dispositivos físicos heterogêneos e os conecta à Internet, fornecendo funcionalidades de controle, visualização, processamento e armazenamento de dados em tempo real (PIRES et al., 2015). Os dispositivos suportados consistem em elementos provedores de dados (sensores), não sendo oferecido suporte a atuadores.

A plataforma integra os dispositivos físicos ao ambiente computacional através da disponibilização de interfaces de alto nível em que os dados coletados e os serviços são tratados como recursos web acessados via URIs. Para tal EcoDiF utiliza os princípios da arquitetura REST (FIELDING, 2000), fazendo o uso de HTTP e suas operações (GET, PUT, POST, DELETE) no suporte às comunicações. Esta abordagem proporciona um tratamento para a heterogeneidade tecnológica dos dispositivos físicos e promove a interoperabilidade.

Os aspectos tecnológicos dos dispositivos físicos são tratados através de *drivers*, os quais também utilizam os princípios REST no seu acesso. Esses *drivers* podem ser de dois tipos:

passivos e ativos. Os *drivers* passivos são baseados em eventos, em que a leitura é realizada sempre que existe uma alteração nos valores dos dados monitorados. Os *drivers* ativos, por sua vez, fazem a leitura dos dados de forma periódica, mesmo que nenhuma alteração tenha ocorrido em seus valores. Os dados coletados dos dispositivos através dos *drivers* são encapsulados em formato EEML (EEML, 2015) e alimentam um *feed* de dados que é enviado ao middleware.

A plataforma EcoDiF utiliza *triggers* como mecanismo para o tratamento de eventos. Os *triggers* permitem produzir notificações com base nas condições definidas em termos dos valores atuais dos *feeds*.

EcoDiF oferece um ambiente para a programação de aplicações *mashups* (MELO, 2007). Essas aplicações possibilitam a composição de diferentes tipos de informações providas por diferentes fontes, tais como serviços Web e bases de dados relacionais. Essa estratégia habilita o EcoDiF a prover mecanismos de Ciência de Contexto.

3.2 Xively

Xively (LOGMEIN, 2015) é uma plataforma baseada em nuvem desenvolvida para gerenciar dados produzidos por dispositivos. Por ser uma solução comercial alguns aspectos de sua arquitetura não são discutidos na literatura.

Xively disponibiliza uma API para que os dados gerados a partir dos dispositivos sejam enviados para a plataforma, e assim possam usufruir dos serviços oferecidos, como a visualização de dados históricos e o tratamento de eventos. Assim como no EcoDiF, Xively foi desenvolvido segundo o padrão arquitetural REST, tratando os sensores como recursos Web, disponibilizado interfaces padronizadas para a troca de dados. Isto estabelece uma forma de tratar a heterogeneidade de dispositivos e promover a interoperabilidade.

Os dados são organizados através de *feeds*, *datapoints* e *datastreams*. Um *feed* representa os dados de um ambiente (uma sala, por exemplo) com seus respectivos *datastreams*, que representam dados enviados por um determinado sensor desse ambiente (por exemplo, temperaturas do ambiente monitorado), enquanto um *datapoint* representa um único valor de um *datastream* em um determinado instante de tempo.

O gerenciamento de eventos é feito através do uso de triggers, que são mecanismos reativos que possibilitam o envio de notificações quando modificações nos valores dos dados *feeds* satisfazem condições especificadas. Essas notificações consistem no envio de solicitações HTTP POST para uma URI pré-configurada. Por exemplo, uma notificação pode ser enviada quando uma temperatura excede certo valor. Os dados enviados através da solicitação POST são

estruturados em formato JSON ou XML. Xively não possui nenhum mecanismo de descoberta, sendo necessário adicionar os dispositivos manualmente pela interface web.

3.3 Carriots

Carriots (CARRIOTS, 2016) também é uma plataforma de middleware comercial baseado em nuvem para o gerenciamento de dispositivos da IoT. Carriots é baseada em serviços (*Platform as a Service* - PaaS) os quais são utilizados para gerenciar os dados provenientes de dispositivos. O princípio de funcionamento do Carriots tem por base uma organização hierárquica rígida, em que todos os recursos e dispositivos são associados a um projeto. Essa organização hierarquia segue a seguinte ordem: "*project*" - "*service*" - "*group*" - "*device*".

Como as demais plataformas apresentadas, o Carriots também provê uma API REST para as comunicações, mas também oferece suporte ao protocolo MQTT, sendo que os dados podem ser estruturados em formato JSON ou XML.

O suporte ao tratamento e eventos é realizado através de três abstrações: *listner*, *trigger* e *rule*. *Listners* consistem de regras ECA que podem ser usadas para tratar eventos associados a uma entidade em qualquer nível da hierarquia. As regras podem ser disparadas, por exemplo, pela chegada de um dado, pela gravação de um dado, entre outros. As expressões são tratadas através de estruturas DO/IF/THEN/ELSE cujas sentenças são escritas em Groovy. *Trigger* é estratégia utilizada no Carriots para publicar automaticamente os dados recebidos em sistemas externos via API REST. Um *trigger* deve ser associado a um serviço. Qualquer conjunto de dados recebido por um serviço a partir de um dispositivo pode disparar publicações. *Rule*, no Carriots, refere-se a scripts escritos em Groovy que podem conter qualquer tipo de lógica, não se limitando ao tratamento de eventos.

3.4 LinkSmart

LinkSmart (PATTI et al., 2016) (chamado de Hydra até 2010) é um middleware para IoT que combina a tecnologia de serviços web semânticos com os princípios da arquitetura SOA (*Service-Oriented Architecture*).

Devido à elevada heterogeneidade de dispositivos, típica da IoT, o suporte à interoperabilidade é um aspecto central no LinkSmart, e este se dá tanto de forma sintática quanto semântica. Em nível sintático, a interoperabilidade foi viabilizada através do uso

de SOA. No entanto, uma das características principais do middleware é o suporte à interoperabilidade em nível semântico e a estratégia, utilizada nesse caso, foi a combinação de serviços web semânticos com ontologias. Ontologias são utilizadas no LinkSmart para representar todas as meta-informações dos dispositivos, possibilitando sua parametrização semântica, o que inclui as informações sobre os seus recursos. Para processar essas ontologias, LinkSmart disponibiliza um conjunto de ferramentas que podem ser usadas durante a etapa de desenvolvimento ou em tempo de execução. Porém a plataforma não garante todas as funcionalidade em qualquer dispositivo, para isso, distingue os dispositivos que são capazes de suportar seus recursos (*LinkSmart-enabled device*) dos que não são capazes de hospedá-la (*non LinkSmart-enabled devices*), porém esses últimos ainda podem se comunicar com o LinkSmart através de *gateways*. Com a adoção destas estratégias, o middleware viabiliza que todos os dispositivos *LinkSmart-enabled* possam ser acessíveis de maneira uniforme através de serviços web semânticos.

Uma extensão do LinkSmart chamada EBBITS foi proposta para que o middleware passasse a prover funcionalidades adicionais, como o suporte a eventos complexos, regras de negócio e Ciência de Contexto, e assim pudesse atender as necessidades de cenários da indústria além de outros domínios de aplicação.

Os eventos são gerados a partir dos dispositivos e são encaminhados para a arquitetura do middleware que os processa em etapas, elevando o nível de abstração das informações. Esse processamento realiza inicialmente uma normalização e tratamentos semânticos básicos (utilizando, por exemplo, classificação, reconhecimento de padrões, ou métodos de agrupamento). Posteriormente são aplicados procedimentos que combinam os eventos de baixo nível gerando estruturas de informações mais complexas através da aplicação de técnicas de raciocínio. Essas informações são processadas através de regras de negócio utilizando a ferramenta Drools.

3.5 Discussão dos trabalhos relacionados

Esta seção contempla uma discussão dos trabalhos relacionados ao CoIoT, que teve como critérios de comparação os objetivos de concepção do middleware apontadas na seção 1.2. A tabela 3.1 sistematiza a comparação feita entre os trabalhos relacionados e o CoIoT.

A arquitetura proposta para o CoIoT foi concebida de forma a gerenciar sensores e atuadores. Com isso, pode ser otimizado o gerenciamento tanto da aquisição dos dados

Tabela 3.1 – Comparação dos trabalhos relacionados

	EcoDiF	Xively	Carriots	LinkSmart	CoIoT
Interoperabilidade	✓	✓	✓	✗	✓
Gerenciamento distribuído	✗	✗	✗	✗	✓
Descoberta e gerenciamento de recursos	✗	✗	✗	✓	✓
Escalabilidade	✓	✓	✓	✗	✓
Gerenciamento de grande volume de dados	✓	✓	✓	✓	✗
Gerenciamento do fluxo de dados	✓	✓	✓	✓	✓
Suporte a regras	✗	✗	✓	✓	✓
Ciência de Contexto	✗	✗	✗	✓	✓

de contexto a partir de vários tipos de sensores, usual nos ambientes computacionais para provimento de aplicações ubíquas, como na atuação distribuída sobre o meio físico. Nos trabalhos relacionados, o suporte ao tratamento de atuadores é encontrado apenas nos projetos Xively e Carriots.

Com exceção do LinkSmart, os trabalhos relacionados apresentam mecanismos interoperáveis para a troca de dados, os quais constituem também uma forma de tratar a heterogeneidade de dispositivos. Nesse sentido, a estratégia adotada por todos eles consiste na utilização de API REST como forma de disponibilizar os seus recursos. Considerando que a revisão da literatura também apontou que o uso da arquitetura REST tem sido amplamente empregada na IoT, no CoIoT também foi adotada esta abordagem como mecanismo de interoperação, a qual propicia que todos os recursos, como o acesso aos dados sensoriados, comandos de atuação, publicações, entre outros, sejam acessados através de URIs.

Nos trabalhos relacionados, a descoberta e o gerenciamento de recursos são apenas suportados pelo LinkSmart. A abordagem adotada no LinkSmart baseia-se na representação dos dispositivos a partir de modelos semânticos, o mecanismo de descoberta é realizado a partir desses modelos. No CoIoT, por outro lado, o elemento básico do mecanismo de descoberta são os gateways acessíveis na rede e, a partir deles, os sensores e atuadores que ele gerencia. Além disso, a abordagem utilizada considera todo o potencial de protocolos de uso consolidado.

A Ciência de Contexto é contemplada apenas pelo LinkSmart, porém embora empregue estratégias de coleta distribuída, o suporte ao processamento de contexto é feito de maneira centralizada. Já o CoIoT oferece um mecanismo distribuído tanto para coleta e processamento de contexto através de regras, bem como para atuação no ambiente. 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.

Todos os trabalhos relacionados apresentam estratégias de *trigger* para gerenciar o fluxo de dados transmitidos entre os diferentes dispositivos envolvidos. Um menor fluxo de dados traz benefícios principalmente no que tange à escalabilidade e ao consumo de energia. No entanto, Xively e Carriots não oferecem *triggers* associados à coleta, ficando sob responsabilidade do usuário o desenvolvimento de uma estratégia que atenda a sua demanda utilizando a API disponibilizada pelas plataformas. Nesse sentido, no CoIoT, 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 para o CoIoT também proporciona a minimização do fluxo de dados em relação a abordagens centralizadas, como no caso dos trabalhos relacionados.

O tratamento de eventos é suportado por todos os trabalhos selecionados, porém apenas o Carriots utiliza regras nesse tratamento. Além disso, o gerenciamento distribuído das regras entre os Servidores de Contexto e de Borda é um diferencial do CoIoT em relação aos demais projetos. No Servidor de Contexto são tratadas regras que necessitam o cruzamento de diferentes informações, sejam elas oriundas de Servidores de Borda distintos, de dados históricos ou produzidas a partir de sistemas externos, além daquelas que demandam maior poder computacional. Já o Servidor de Borda trata as regras de contingência, que necessitam resposta rápida e cujo tratamento não deve ser afetado no caso de falha de operação da rede. Esta funcionalidade de processamento de contexto, nos trabalhos relacionados, usualmente está restrita a um único equipamento.

Na área da Internet das Coisas, os dispositivos computacionais são incorporados a objetos do ambiente gerando informações de forma contínua, totalizando volumes expressivos de dados a serem tratados. Esse aspecto pode ser observado na revisão dos trabalhos relacionados, nos quais foi identificado o emprego de técnicas de gerenciamento de grande volume de dados. Até o presente momento, o CoIoT não provê suporte a este tipo de tratamento, estando o mesmo previsto em trabalhos futuros. No CoIoT, o acesso aos dados produzidos a partir da arquitetura é realizado através de uma API REST, sejam eles gerados diretamente pelos sensores ou armazenados no Repositório de Contexto. Esta estratégia simplifica uma futura incorporação de técnicas de gerenciamento de grande volumes de dados na arquitetura do CoIoT.

Diante da investigação feita, pode-se constatar que nenhum dos middlewares para IoT identificados atende a todos os requisitos selecionados. Ainda existem requisitos importantes que foram pouco explorados, como Ciência de Contexto, por exemplo. Portanto, ainda existem várias questões de pesquisa e desenvolvimento em aberto. Nesse sentido, o trabalho desenvolvido na concepção do CoIoT caracteriza uma contribuição comprometida com o atendimento destes requisitos decorrentes da crescente disseminação da IoT.

3.6 Considerações sobre o capítulo

O estudo dos trabalhos relacionados apresentado neste capítulo teve como base os objetivos de pesquisa do middleware CoIoT. A revisão de literatura realizada apontou a necessidade de distribuição das estratégias de processamento do contexto e tratamento de eventos. Foi identificado que essa distribuição é importante para propiciar um tratamento do contexto próximo à origem das informações processadas. Essa abordagem promove a redução do tráfego de dados através da rede e reduz a dependência das infraestruturas de comunicação gerenciadas por terceiros quando da identificação de situações críticas.

A comparação do middleware CoIoT com os trabalhos relacionados caracterizou sua contribuição científica, sistematizando as suas diferenças em relação a estes projetos no que tange aspectos como o tratamento de sensores, bem como de atuadores, o uso de interface interoperável baseada em API REST, a descoberta e gerenciamento de recursos, suporte à ciência do contexto baseado em regras, suporte ao tratamento de eventos tanto na coleta de dados quanto no processamento de contexto e, suporte ao gerenciamento do fluxo de dados através da rede.

O próximo capítulo apresenta a concepção do middleware CoIoT, destacando suas principais características e funcionalidades.

4 COIOT: CONCEPÇÃO DA ARQUITETURA

Neste capítulo é apresentada a arquitetura proposta para atender as premissas de concepção CoIoT, com o detalhamento das funcionalidades dos módulos que a constituem. Na concepção desta arquitetura foram contemplados aspectos como: (i) processamento do contexto baseado em regras e eventos distribuídos; (ii) coleta e atuação baseadas em eventos; (ii) interoperabilidade e tratamento da heterogeneidade e (iv) gerência e descoberta de recursos.

4.1 Principais características

A arquitetura proposta para o CoIoT foi concebida considerando a Internet das Coisas enquanto estratégia de ubiquidade, deste modo foram tratados aspectos como heterogeneidade, interoperabilidade, gerenciamento de dados, auto-organização e escalabilidade. Assim, por ser uma arquitetura comprometida com as demandas das aplicações ubíquas, tem por foco atuar na coleta das informações contextuais do ambiente, no seu processamento e na decorrente atuação sobre o mesmo.

Uma das principais características buscadas no CoIoT é o suporte à Ciência de Contexto visando a identificações de interesse. Na concepção do CoIoT procurou-se prover esse suporte através de uma estratégia organizada em várias etapas (conforme ilustrado pela Figura 4.1), as quais são processadas pelo CoIoT de maneira independente da aplicação a que se destina.

As aplicações ubíquas em função de seus objetivos demandam contextos de interesse, que consistem nas grandezas sensoriadas que realmente importam ao cumprimento desses objetivos. As mudanças de estado dessas variáveis de contextos produzem eventos que disparam o processamento de regras de negócio do tipo ECA (vide Seção 2.3.1). Essas regras avaliam o estado de uma ou mais variáveis de contexto para verificar a ocorrência de situações de interesse. Essas situações, quando identificadas, são responsáveis por executar ações, que podem ser: (i) o controle de um atuador; (ii) o envio de notificações à aplicação ou; (iii) a publicação de informações contextuais para armazenamento histórico ou disparo de nova regra.

Os ambientes da Computação Ubíqua são inerentemente distribuídos e, muitas vezes, conectados através da Internet utilizando infraestruturas de rede compartilhadas e gerenciadas por diferentes fornecedores. Nesses ambientes as desconexões podem ocorrer com alguma frequência e devem ser tratadas (COSTA; YAMIN; GEYER, 2008). Por isso o modelo de processamento de contexto proposto para o CoIoT tem por premissa tratar estas desconexões. O modelo distribuído proposto para o CoIoT permite que se mantenha o sistema consistente

e operacional durante os momentos desconexão, eventualmente com a supressão de algumas funcionalidades. Esse modelo distribuído possibilita tratar eventos críticos no seu local de ocorrência, promovendo uma redução na possibilidade de falhas, uma vez que pode ser utilizado, no processamento, equipamento computacional localizado junto ao ambiente monitorado.

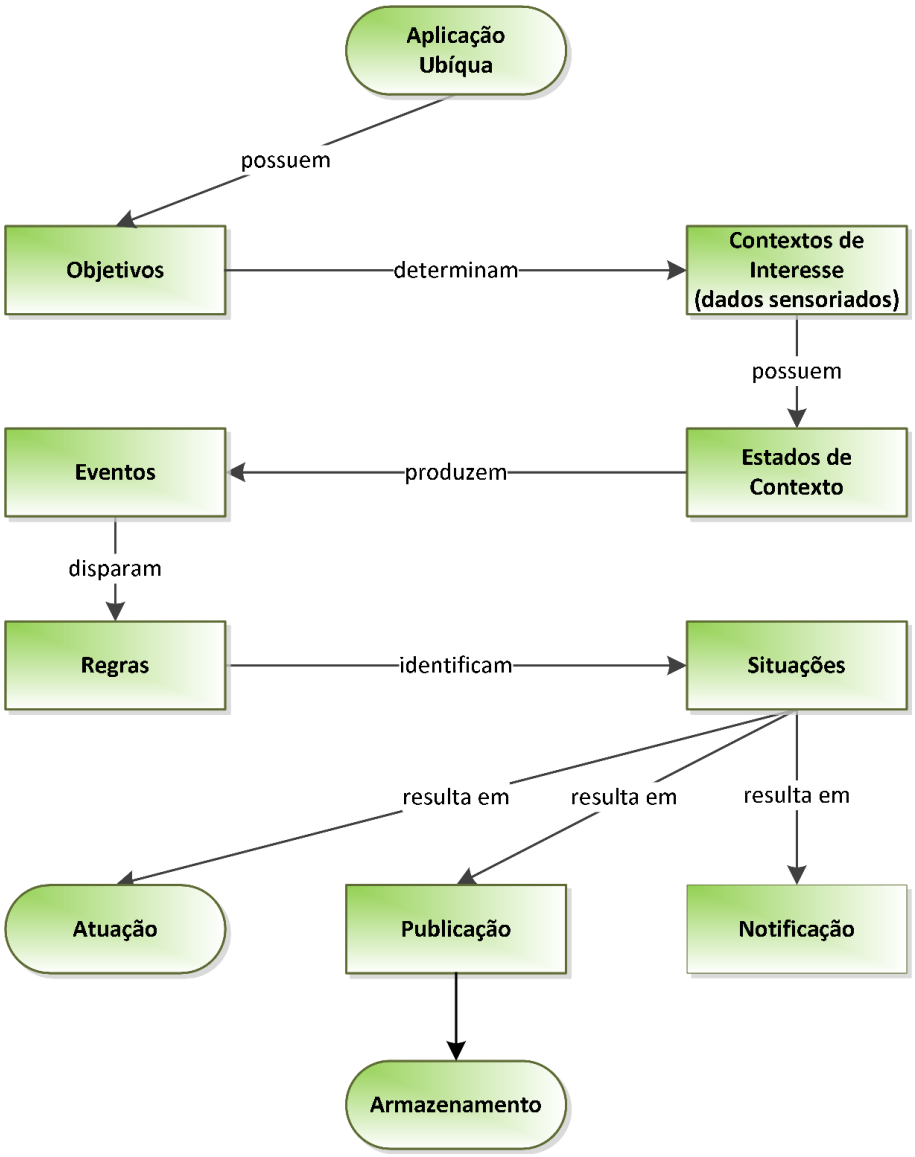

Figura 4.1 – Estratégia para o tratamento de contexto

Na Internet das Coisas, eventos do ambiente são caracterizados quando existe uma alteração importante em algum contexto de interesse, por exemplo, uma temperatura atingindo certo valor ou a identificação da entrada de um usuário em uma sala, entre outros. O CoIoT foi concebido tendo como premissa a interceptação e tratamento desses eventos. Essa decisão teve como motivação as seguintes perspectivas:

Otimização do uso dos recursos computacionais

A ocorrência de novas situações é consequência de mudanças de estado de variáveis contextuais importantes. Essas mudanças de estado podem ser mapeadas como eventos tratados pela arquitetura do CoIoT e são utilizadas como método de disparo das regras de tratamento das informações contextuais. Isso permite otimizar o uso dos recursos computacionais dos dispositivos uma vez que o processamento das regras só se dará no momento que for necessário. Considerando esta orientação a eventos quando do processamento de contexto, decorre a decisão de utilizar no CoIoT regras do tipo ECA. Essa opção pelas regras ECA é particularmente conveniente quando se deseja disparar o tratamento das variáveis de contexto somente quando ocorrem mudanças consideradas significativas nos seus estados.

Rapidez na tomada de decisão

Ambientes da IoT geram potencialmente uma grande quantidade de eventos. O gerenciamento desses eventos por parte da arquitetura do CoIoT e a utilização dos mesmos como mecanismo de disparo das regras de processamento de contexto permitem que as situações sejam identificadas no momento em que eles acontecem, possibilitando uma reação rápida quando necessário.

Otimização dos recursos de comunicação

A abordagem concebida para o CoIoT contempla também o tratamento de eventos na coleta das informações contextuais do ambiente. Essa estratégia permite implementar métodos de coleta do tipo *push* gerenciados por regras. Através dessas regras são especificados o momento e/ou padrão dos valores dos dados que devem ser coletados, evitando coletas desnecessárias, o que contribui para a redução do volume de dados enviados pela rede.

4.2 Arquitetura Proposta

Esta seção apresenta a arquitetura do middleware CoIoT, que foi concebida para gerenciar um ambiente computacional típico no cenário da Internet das Coisas, enquanto provê suporte à Ciência de Contexto para as aplicações ubíquas.

4.2.1 Visão Geral

Na proposta do CoIoT o ambiente computacional é constituído por células em que se distribuem os dispositivos computacionais, conforme mostra a Figura 4.2. As abstrações empregadas neste ambiente computacional ampliam a proposta do GPPD/UFRGS para o middleware EXEHDA (LOPES et al., 2014).

Os componentes básicos deste ambiente são: (*i*) o EXEHDAbase que consiste no elemento central da célula responsável por todos serviços básicos e referência para os demais elementos; (*ii*) o EXEHDAnodo que corresponde aos dispositivos computacionais responsáveis pela execução das aplicações; (*iii*) o EXEHDAnodo 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*, por exemplo; e (*iv*) o EXEHDAborda que consiste no elemento de borda do ambiente ubíquo, responsável por fazer a interoperação entre os serviços do middleware e os dispositivos de coleta e atuação típicos da IoT.

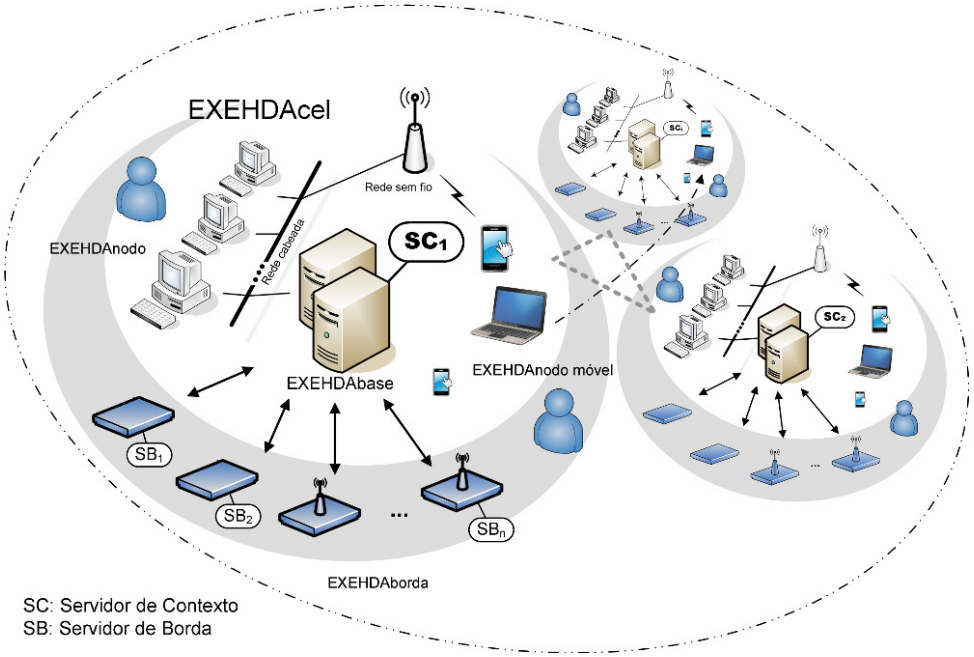

Figura 4.2 – Organização celular do ambiente ubíquo

Na abordagem para gerenciamento de contexto proposta para o CoIoT, as responsabilidades são organizadas em dois tipos de servidores: Servidor de Contexto e Servidor de Borda. O Servidor de Borda se destina a gerenciar a interação com o meio físico através de sensores e atuadores, bem como executar o primeiro estágio do processamento do contexto.

O Servidor de Contexto, por sua vez, atua no armazenamento e no segundo estágio de processamento das informações contextuais, integrando dados históricos e dados provenientes de diferentes Servidores de Borda distribuídos no ambiente ubíquo.

Uma visão geral da arquitetura proposta para o CoIoT está apresentada na Figura 4.3. Essa figura destaca a relação do Servidor de Borda com o Servidor de Contexto, bem como os demais Servidores de Bordas vizinhos em operação no ambiente. Além disso, também são identificadas as relações destes servidores com os dispositivos que interagem com o ambiente físico. O Servidor de Borda é instanciado em um equipamento do tipo EXEHDAborda, enquanto o Servidor de Contexto é alocado em um EXEHDAbase. Na base da arquitetura estão os gateways, que são equipamentos responsáveis por tratar os aspectos tecnológicos de cada dispositivo de sensoreamento e atuação.

Em uma célula típica do ambiente ubíquo, a expectativa é que se tenha vários EXEHDAbordas gerenciando os dispositivos da IoT. Portanto, prezando-se pela eficiência energética global do ambiente ubíquo, no CoIoT a perspectiva é que esses equipamentos tenham recursos computacionais limitados, apresentando um consumo baixo de energia. Em função disto, as regras de processamento contextual que exijam maior poder computacional não devem ser executadas nestes equipamentos, e sim o desenvolvedor das aplicações ubíquas deve alocá-las para execução no Servidor de Contexto.

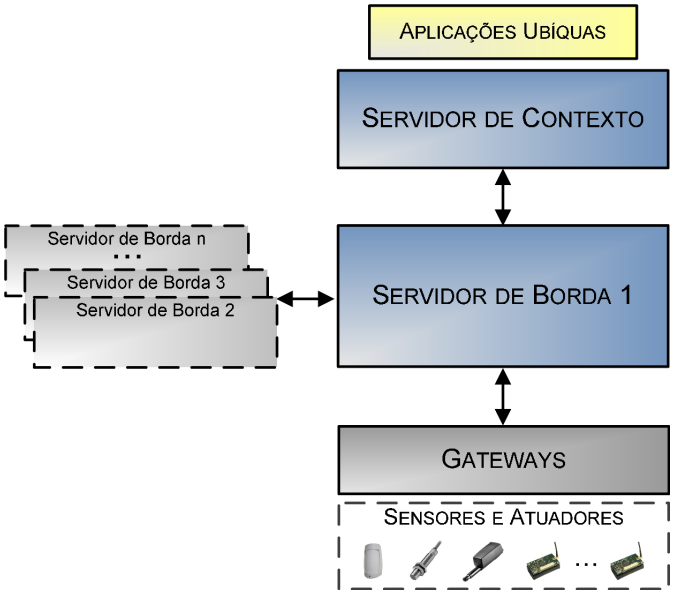

Figura 4.3 – Visão geral da arquitetura do CoIoT

A arquitetura do CoIoT foi concebida para gerenciar diferentes dispositivos da IoT, como sensores e atuadores heterogêneos, através de gateways ou diretamente pelo Servidor de Borda. Os gateways agem como uma ponte que faz a interligação do CoIoT, através

de rede TCP/IP, com os diversos dispositivos da IoT. Esses equipamentos têm hardware e capacidades computacionais dedicadas 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 para IoT vem sendo disponibilizados no mercado (GAZIS et al., 2015).

Com o intuito de garantir a interoperabilidade com tecnologias de mercado, e também potencializar a distribuição das iniciativas de coleta e/ou atuação, na concepção da arquitetura do CoIoT estão previstos dois tipos distintos de gateways:

- Gateway Proprietário: tem funcionalidades heterogêneas variando de acordo com seus fabricantes;
- Gateway Nativo: faz parte da arquitetura do CoIoT tendo suas funcionalidades integradas ao mesmo.

O suporte arquitetural aos Gateways Proprietários depende dos recursos oferecidos pelas suas APIs, sendo garantidos apenas a leitura direta dos valores coletados e o envio de comandos de atuação, quando suportados. Por outro lado, os Gateways Nativos, além de oferecer acesso direto aos seus recursos (atuação e sensoreamento), possuem mecanismos de sensoreamento gerenciados através de eventos utilizando estratégias de *trigger*.

Importante registrar que também está previsto na concepção do CoIoT o gerenciamento de sensores e atuadores conectados diretamente ao EXEHDAborda quando estes forem fisicamente compatíveis. Nesse caso, o suporte a esses dispositivos é feito através da virtualização do Gateway Nativo (Gateway Virtual), o qual é incorporado na arquitetura de software do Servidor de Borda. Os Gateways são opcionais na arquitetura, a qual possibilita utilizar somente Gateways físicos (nativos ou proprietários), somente Gateway Virtual, ou ambos.

A arquitetura proposta para o CoIoT tem por premissa atuar no atendimento das necessidades das aplicações ubíquas porém de forma independente das mesmas. Portanto, tanto a captura e processamento dos dados de contexto, quanto a atuação sobre o meio continuam a ser gerenciados mesmo nos períodos nos quais as aplicações interessadas no seu uso estejam inoperantes.

Uma visão detalhada da arquitetura do CoIoT é apresentada na Figura 4.4 a qual contempla todos os módulos que compõem os Servidores de Borda e Contexto e a forma como eles se relacionam. As seções a seguir apresentam os módulos da arquitetura organizados de acordo com suas funcionalidades.

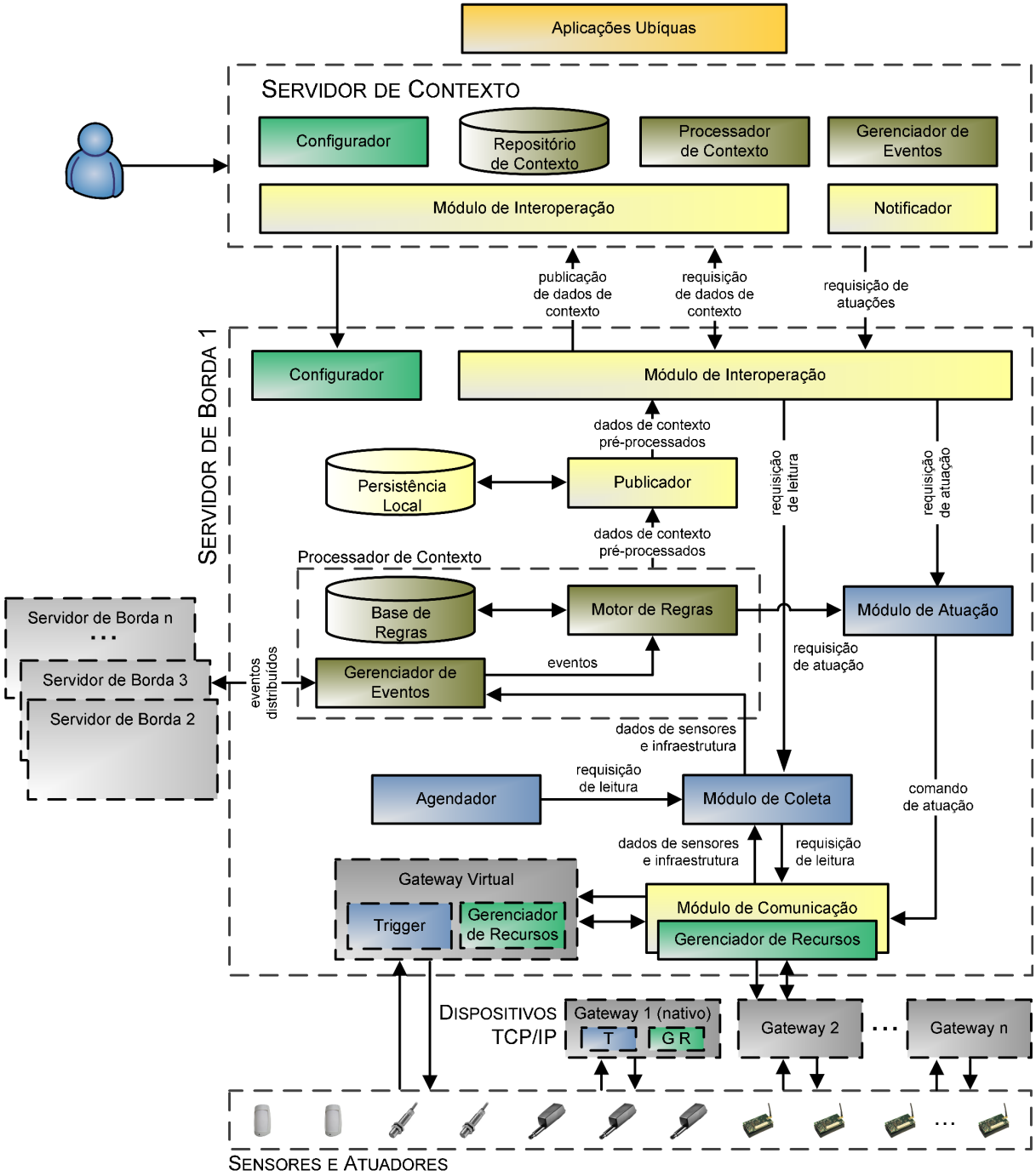

Figura 4.4 – Arquitetura do CoIoT. Legenda dos módulos agrupados segundo suas funcionalidades: ■ Processamento do contexto; ■ Interoperabilidade e tratamento da heterogeneidade; ■ Gerência e descoberta de recursos; ■ Coleta e atuação.

4.2.2 Processamento do contexto baseado em regras e eventos distribuídos

O tratamento do contexto no CoIoT se dá de forma distribuída entre os Servidores de Borda e Contexto e está organizado através de seis módulos que contemplam o gerenciamento de eventos, bem como o processamento das regras. O detalhamento desses módulos é descrito a seguir.

Gerenciador de Eventos

O Gerenciador de Eventos tem a função de aglutinar os eventos com origem no próprio dispositivo ou em um dispositivo remoto. O modelo de tratamento de eventos proposto para o CoIoT considera um conjunto de eventos primitivos gerados a partir da arquitetura, em nível local, devido a: (i) mudanças de estado dos contextos de interesse coletados através de sensores; (ii) ocorrência temporal; (iii) ativação/desativação de atuadores; e (iv) alterações na infraestrutura do ambiente computacional. Estes eventos são apresentados na Tabela 4.1.

O suporte a eventos compostos é provido pelo CoIoT a partir do processamento das regras, os quais são gerados toda vez que uma condição tratada a partir de uma composição de eventos é tida como verdadeira. Os eventos compostos são disponibilizados para acesso remoto através de um mecanismo baseado no modelo *publish/subscribe* (TANENBAUM; STEEN, 2007) o qual tem a função de viabilizar o gerenciamento de eventos distribuídos. Desta forma, eventos compostos produzidos a partir de regras locais podem notificar Servidores de Borda remotos através de seus módulos Gerenciadores de Eventos. Já nesses dispositivos remotos, essas notificações são mapeadas como eventos similares aos gerados localmente, podendo assim serem usados para disparar regras de processamento de contexto.

Essa abordagem permite que os diversos Servidores de Borda possam trocar informações e colaborar entre si nas atividades de processamento de contexto. Essa abordagem de processamento colaborativo em nível de borda vai ao encontro dos desafios de pesquisa abordados pelo paradigma de Computação em Fog.

Tabela 4.1 – Eventos primitivos do CoIoT

Evento	**Nível de Geração**	**Descrição**
Publication	Gateway/Servidor de Borda	Ocorre quando uma informação contextual é enviada ao Servidor de Borda ou Servidor de Contexto
Actuation	Gateway	Ocorre quando um atuador é disparado
NewDevice	Gateway	Ocorre quando um novo sensor ou atuador é identificado
DeviceDisconect	Gateway	Ocorre quando um sensor ou atuador é desconectado
NewGateway	Servidor de Borda	Ocorre quando um novo Gateway é inserido
GatewayDisconect	Servidor de Borda	Ocorre quando um Gateway é desconectado

Processadores de Contexto

O módulo Motor de Regras, instanciado no Servidor de Borda, constitui o primeiro nível de processamento, enquanto o Processador de Contexto, instanciado no Servidor de Contexto, o segundo nível. As regras processadas no Motor de Regras são armazenadas na Base de Regras.

As regras submetidas ao Motor de Regras devem ser elaboradas de forma a atender, prioritariamente, os eventos críticos, cujo tratamento deve ser priorizado. 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, acessar dados oriundos de outras células, ou que envolvam outros modelos de processamento de contexto, devem ser processadas no Servidor de Contexto. Ambos os módulos de processamento do contexto foram concebidos tendo como base o modelo de regras tipo ECA (Evento-Condição-Ação) (TERFLOTH, 2009), as quais são disparadas a partir de eventos primitivos ou compostos que podem ter origem no próprio dispositivo ou em um dispositivo remoto.

A distribuição do tratamento de eventos entre os Servidores de Borda e Contexto é elemento central na proposta de ubiquidade da arquitetura do CoIoT pois possibilita manter o sistema operante mesmo na ocorrência de falhas de comunicação ao mesmo tempo que promove a escalabilidade da plataforma. Além disso, o uso de regras próximas à borda do ambiente computacional possibilita a implementação de estratégias de agregação e fusão de dados o que minimiza o tráfego de informações entre os nodos da rede.

A alteração das regras de processamento de contexto, bem como a sua distribuição aos respectivos Servidores de Borda pode ser feita em tempo de execução a partir do Servidor de Contexto. A nova regra recebida fica em estado de hibernação até que a regra a ser substituída finalize o seu processamento.

A estrutura das regras concebidas para o CoIoT possui os seguintes elementos:

- *Rule Name*: identifica a regra e o evento composto produzido quando do processamento da regra;
- *Group*: grupo ao qual pertence a regra;
- *Event*: evento que dispara a regra, o qual pode ser primitivo ou composto;
- *Condition*: operação lógica a ser tratada;

- *Action*: ação executada toda vez que a condição for verdadeira.

As regras no CoIoT podem ser organizadas em grupos que podem estar ativos ou inativos ao processamento. A ativação ou desativação de um grupo de regras pode ocorrer dinamicamente a partir de ações disparadas por regras. Essa estratégia foi adotada para simplificar a implementação de janelas temporais as quais podem então serem tratadas através de grupos de regras.

Embora seja utilizado o modelo 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 a partir da regra, os quais são decorrência do tipo de domínio de aplicação a ser atendida pelo COIOT.

Repositório de Contextos

O Repositório de Contextos emprega um modelo relacional para a representação do contexto fornecendo uma visão histórica de contextos. A estrutura do Repositório de Contextos reflete a organização da arquitetura do middleware EXEHDA, contemplando as relações entre as aplicações, componentes, sensores, ambientes e contextos de interesse. O repositório também armazena dados de configuração da arquitetura, e os valores sensoriados 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.

4.2.3 Coleta e atuação baseadas em eventos

O mecanismo de coleta concebido para o (CoIoT) dá suporte tanto ao método *pull* quanto ao método *push*. As requisições de coleta baseadas no método *pull* são gerenciadas pelo Módulo de Coleta. Já as operações de coleta baseadas no método *push* são tratadas pelo Agendador ou diretamente pelos Gateways através no módulo *Trigger* os quais são responsáveis pela geração de eventos primitivos de leitura dos sensores.

As operações de atuação são gerenciadas pelo Módulo de Atuação o qual recebe requisições tanto do Motor de Regras quanto externas ao Servidor de Borda.

Módulos de Coleta e Atuação

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

O Módulo de *Atuação* aglutina os comandos de atuação oriundos de diferentes fontes e tem um funcionamento análogo ao Módulo de Coleta. Ao receber uma requisição de atuação, o Módulo avalia eventuais conflitos e posteriormente envia os comandos de atuação, que incluem o ID do atuador e os correspondentes padrões de operação (tempo de duração, potência de ativação, etc.), ao gateway responsável pelo efetivo tratamento.

Gateway Virtual e Agendador

O Gateway Virtual, que consiste na virtualização do Gateway nativo, possui dois módulos básicos: Trigger e Gerenciador de Recursos. O Gateway Virtual utiliza drivers que encapsulam e controlam os sensores e atuadores de uma maneira individualizada, evitando que diferenças operacionais destes dispositivos se propaguem a outros componentes da arquitetura. A leitura dos sensores é baseada em eventos sendo gerenciada pela arquitetura através dos módulos Trigger e Agendador. Esses módulos foram concebidos para lidar com os dois principais tipos de eventos de coleta a serem tratados na IoT: eventos temporais e eventos do ambiente (PERERA et al., 2014). O módulo **Trigger** gerencia eventos do ambiente. Esse módulo foi incorporado aos gateways para que o monitoramento das mudanças de estado das variáveis contextuais seja feito localmente, reduzindo assim o tráfego de dados na rede. Já o módulo **Agendador** gerencia os eventos temporais. Esse módulo foi mantido junto ao Servidor de Borda (externamente aos Gateways), pois a origem dos eventos de leitura tratados por ele não está nas variáveis de contexto mas sim no controle de tempo feito pela arquitetura. Além disso, como o Servidor de Borda é executado no EXEHDAborda, os requisitos de hardware do dispositivo selecionado para essa finalidade garantem uma maior confiabilidade no controle sobre os eventos temporais do que o EXEHDAgateway.

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. OS módulos de Comunicação e Gerenciador de Recursos são tratados nas seções seguintes.

4.2.4 Interoperabilidade e tratamento da heterogeneidade

Na Internet das Coisas os ambientes pressupõem uma grande quantidade de dispositivos heterogêneos que produzem informações contextuais ou atuam no ambiente. A proposta para tratamento dessa heterogeneidade, no CoIoT, é organizada conforme apresentada na Figura 4.5. Em um primeiro nível, é feita uma uniformização das tecnologias de sensoriamento e atuação através gateways, ou seja, independentemente do meio físico utilizado nas comunicações com os dispositivos e dos seus protocolos, os gateways encapsulam todos os aspectos tecnológicos e disponibilizam aos demais componentes da arquitetura do CoIoT estruturas de dados padronizadas. Em um segundo nível, essas estruturas de dados são manipuladas e disponibilizadas às aplicações através de um Serviço Web baseado na arquitetura REST (FIELDING, 2000). Assim, no CoIoT, tanto sensores quanto atuadores, podem ser acessados pelas aplicações da mesma forma, isto é, como recursos Web, independentemente da linguagem de programação utilizada na aplicação. Isso estabelece uma interface interoperável para o acesso aos recursos oferecidos pelo CoIoT.

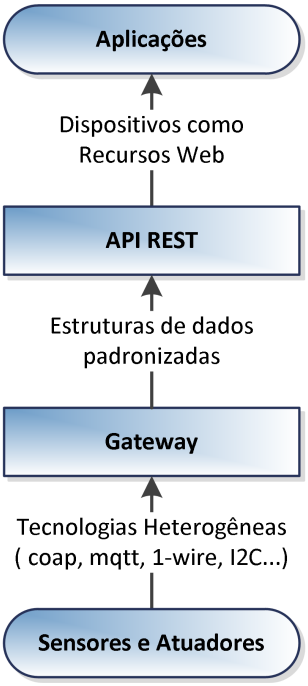

Figura 4.5 – Estratégia para o tratamento da heterogeneidade

A seguir são apresentadas as funcionalidades dos principais módulo que, em conjunto, realizam o tratamento do contexto e da heterogeneidade do CoIoT.

Módulos de Interoperação

Considerando o perfil inerentemente distribuído das aplicações ubíquas, os Módulos de Interoperação do CoIoT, que estão presentes tanto no Servidor de Borda quanto no Servidor de Contexto, foram concebidos para gerenciar a troca de informações entre estes servidores, destes com os demais serviços do middleware em execução em outros equipamentos e com as aplicações ubíquas.

Os módulos de Interoperação estão presentes tanto no Servidor de Borda quanto no Servidor de Contexto e têm funcionalidades análogas. Ambos módulos têm como referência o estilo arquitetural REST, constituindo serviços web, sendo utilizado o protocolo HTTP e suas operações (GET, PUT, POST, DELETE) no suporte às trocas de informações. Com isso, todas as funcionalidades associadas aos sensores e atuadores são mapeadas na forma de recursos web e acessadas via URIs. Através desse módulo é disponibilizada uma interface padronizada para manipular as funcionalidades dos dispositivos, como a coleta, configuração, acionamento, inserção e remoção de dispositivos, entre outras. Também são disponibilizadas interfaces para o gerenciamento de eventos distribuídos, das regras de coleta e das regras de processamento de contexto.

Os Módulos de Interoperação provêm o suporte necessário para disponibilização de uma API REST para o acesso aos recursos oferecidos. Com isso, a interoperação proporcionada pelo CoIoT se dá através de operações HTTP sobre URIs, os quais são o caminho para o acesso a todos os recursos disponibilizados pela arquitetura.

A utilização de REST como modelo de interoperação proporciona a construção de uma interface padronizada, bem definida e, totalmente independente da linguagem de programação utilizada, tanto na implementação dos recursos oferecidos quanto no uso dos mesmos.

As requisições feitas ao Módulo de interoperação do Servidor de Contexto são repassadas ao módulo de mesmo nome do Servidor de Borda, sempre que necessário. Isso se mostra importante para o desenvolvimento de aplicações em ambientes ubíquos distribuídos e heterogêneos, pois restringe aos demais módulos do Servidor de Borda o tratamento dos aspectos tecnológicos envolvidos, evitando que isso se reflita no restante da arquitetura.

Módulo de Comunicação

O Módulo de Comunicação do Servidor de Borda tem a função de prover aos demais módulos da arquitetura COIOT um acesso padronizado aos recursos disponibilizados pelos gateways no tratamento dos dispositivos.

O Módulo de Comunicação gerencia as comunicações com os Gateways nativos através de API REST de forma similar ao tratamento provido pelo Módulo de Interoperação quando do acesso externo aos recursos disponibilizados pelo Servidor de Borda. Por outro lado, o acesso aos recursos 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 [1] e MQTT [2].

Publicador

O Publicador é um módulo do Servidor de Borda que tem a função de disparar as requisições de envio de informações contextuais para as demais camadas do middleware, interoperando 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, foi concebido o módulo de Persistência Local cuja função é realizar o armazenamento temporário da fila de informações contextuais até que as mesmas sejam publicadas.

Notificador

O módulo Notificador tem a função de gerar notificações a partir dos resultados do processamento do contexto realizado pelo Processador de Contexto. Este módulo utiliza uma estratégia de notificação baseada no modelo *publish/subscribe*, e recebe subscrições de todos os serviços e/ou aplicações que requerem notificações a respeito das mudanças nos estados dos contextos. Também é função do Notificador enviar comandos de atuação aos Servidores de Borda quando necessário.

[1] http://coap.technology/
[2] http://mqtt.org/

4.2.5 Gerência e descoberta de recursos

Em ambientes típicos da IoT os dispositivos estão entrando e saindo de operação a qualquer momento, seja por decorrência de término de energia, perda de sinal de rádio ou pela inserção de novo dispositivo devido à reposição por avaria ou necessidade de adicionar outro ponto de monitoramento (YICK; MUKHERJEE; GHOSAL, 2008)(PERERA et al., 2014). Em ambientes dinâmicos e altamente distribuídos como esses da IoT se torna essencial que os sistemas computacionais de suporte disponibilizem mecanismos que gerenciem tais ocorrências de forma que o envolvimento dos usuários seja mínimo. Neste sentido, o módulo Gerenciador de Recursos foi concebido com o objetivo de administrar esses eventos, provendo para isso um mecanismo de descoberta o qual é suportado pelos gateways nativos do CoIoT.

No CoIoT, as informações de contexto sensoriadas do ambiente são gerenciadas tanto pelo Servidor de Borda quanto pelo Servidor de Contexto, os quais necessitam, para isso, manter no Repositório de Contextos as descrições dos gateways, sensores e atuadores. Portanto o processo de descoberta envolve ambos servidores.

Gerenciador de Recursos

O Gerenciador de Recursos é o módulo da arquitetura do CoIoT responsável por tratar a entrada e saída de gateways, sensores e atuadores na rede com o mínimo de interferência humana. Este módulo está presente tanto no Servidor de Borda quanto nos Gateways.

A estratégia de descoberta adotada teve como base o método *plug and play* do protocolo UPnP (4th Line, 2017)(UPNP, 2017), que utiliza princípios REST como forma de prover interoperabilidade entre dispositivos cliente e servidor. A opção pelo uso do UPnP se justifica porque o seu modelo operacional mostra-se adequado à proposta do CoIoT.

Uma instância do UPnP é executada, tanto no Gateway Nativo como Virtual, fazendo com que este seja identificado como um dispositivo UPnP. O gateway funciona como uma ponte entre o Servidor de Borda e os dispositivos de sensoreamento e atuação. Os sensores/atuadores são a última etapa de abstração do CoIoT, que constituem para o mecanismo de descoberta um serviço UPnP que está vinculado ao gateway.

O Ponto de Controle é o elemento central do mecanismo de descoberta concebido, sendo instanciado no Gerenciador de Recursos do Servidor de Borda. A função do ponto de controle é 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. Isso é viabilizado através do Gerenciador de Recursos de cada gateway. O Ponto de Controle atua como um monitorador que aguarda por mensagens de inclusão e/ou remoção de gateways na rede a qual está conectado, rejeitando dispositivos indesejáveis. Cada dispositivo possui uma identificação única, e seus respectivos dados (nome, tipo, modelo e fabricante) são lidos pelo Ponto de Controle a cada requisição. Após a descoberta do dispositivo, o mesmo é adicionado a uma base de dispositivos 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* informando que está sendo desconectado, assim o Ponto de Controle pode removê-lo da lista de dispositivos reconhecidos e informar isso ao Servidor de Contexto. Com isso é gerado um alerta de indisponibilidade às aplicações que estejam utilizando as informações associadas a este dispositivo.

Configurador

O Configurador aglutina todas as configurações necessárias para o funcionamento do CoIoT, incluindo os aspectos específicos da operação dos dispositivos remotos. Através do Configurador do Servidor de Contexto é disponibilizada uma interface Web através da qual é possível gerenciar o histórico de dados armazenados, regras de processamento de contexto, contextos de interesse associados a cada aplicação, subscrições de eventos para viabilizar eventos distribuídos entre os Servidores de Borda, subscrições para geração de notificações, configurações dos Servidores de Borda, Gateways, sensores, atuadores, entre outros.

4.3 Considerações sobre o capítulo

Neste capítulo foi apresentada a concepção do CoIoT, um middleware para IoT que tem como principal premissa realizar, de forma distribuída, a coleta e processamento das informações contextuais, bem como a atuação no ambiente. A arquitetura concebida para o CoIoT foi detalhada, tendo suas funcionalidades distribuídas entre os Servidores de Contexto e de Borda.

O modelo arquitetural distribuído do CoIoT possibilita que o processamento do

contexto possa ser feito de forma colaborativa entre o Servidor de Contexto e os Servidores de Borda. Para viabilizar o processamento colaborativo também entre os dispositivos de borda, como em um ambiente de computação em Fog, foram definidas estratégias para tratamento distribuído de eventos entre estes dispositivos.

O tratamento da interoperabilidade é feita no COIOT pela adoção de um modelo baseado na arquitetura REST, sendo utilizados URIs para prover acesso aos recursos dos dispositivos de coleta e atuação. Já o tratamento da heterogeneidade adotado no middleware tem por base o uso de gateways os quais encapsulam as tecnologias heterogêneas utilizadas pelos dispositivos de coleta e atuação.

No capítulo seguinte são apresentados os estudos de caso concebidos para avaliar das funcionalidades do middleware COIOT, que foram realizados em cenários da agricultura.

5 COIOT: PROTOTIPAÇÃO E ESTUDOS DE CASO EM AGRICULTURA

Este capítulo apresenta os aspectos associados à prototipação do COIOT, bem como a avaliação das funcionalidades da sua arquitetura, a qual teve por base cenários de uso na área da Agricultura.

A escolha da área de avaliação e dos respectivos cenários de uso decorre de projetos que vem sendo desenvolvidos com a participação dos grupos de pesquisa GPPD/UFRGS (Grupo de Processamento Paralelo e Distribuído da Universidade Federal do Rio Grande do Sul) e LUPS/UFPel (*Laboratory of Ubiquitous and Parallel Systems* da Universidade Federal de Pelotas).

Assim, na continuidade deste capítulo, a seção 5.1 apresenta os aspectos de prototipação da arquitetura do CoIoT, abordando as principais ferramentas utilizadas. A seção 5.2 contempla dois estudos de caso utilizados para avaliar as funcionalidades da arquitetura. O primeiro estudo de caso foi realizado no LASO (Laboratório de Análise de Sementes Oficial) da EMBRAPA Clima Temperado. Neste estudo de caso foi realizado um experimento para avaliação da usabilidade de uma aplicação prototipada a partir do CoIoT, empregando o método TAM (*Technology Acceptance Model*). Já o segundo estudo de caso contempla um cenário de controle de irrigação em uma lavoura de videiras (parreiral) típica da região sul do estado do Rio Grande do Sul. A avaliação deste cenário foi realizada através de simulação considerando demandas reais deste tipo de cultura.

5.1 Aspectos de Prototipação

Na prototipação do COIOT, foram priorizadas tecnologias escaláveis, robustas e de código aberto. No que diz respeito ao desenvolvimento de software, foram utilizadas diversas ferramentas tendo como base a linguagem Python. O uso do Phyton foi particularmente útil na etapa de prototipação por simplificar os ajustes nos códigos durante as fases iniciais de implementação da arquitetura e sua colocação em operação em ambiente de produção.

Para implementar os módulos de Interoperação e Comunicação foi utilizado o Django [1]. Django é um framework MVC (*Model-view-controller*) implementado na linguagem Python voltado à construção de aplicações Web que simplifica o desenvolvimento de APIs REST. Através desse framework foram implementados recursos, acessíveis através de URIs, para a manipulação de diversos elementos que compõem a arquitetura do CoIoT, como gateways,

[1] https://www.djangoproject.com

sensores, atuadores, regras, entre outros. Além disso, todas as operações realizadas sobre o Repositório de Contexto também são feitas por intermédio da API REST. O acesso aos recursos disponibilizados pela API são acessíveis através de autenticação do tipo usuário/senha. A estruturação dos dados enviados através das operações REST é feita em formato JSON. A especificação dos principais recursos implementados, bem como as URIs de acesso são apresentadas através das Tabelas 5.1 e 5.2. A Figura 5.1 apresenta o uso da API REST do COIOT em uma requisição GET feita em curl [2] para a coleta do valor de um sensor identificado como "TemperatureBOD1" e a respectiva resposta estruturada em JSON.

Tabela 5.1 – API do COIOT para tratamento de sensores e atuadores

URI	Ação	Descrição
/sensor ou /actuator	GET	Recupera um conjunto de sensores ou atuadores com todas as suas informações.
/sensor ou /actuator	POST	Cadastra um novo sensor ou atuador.
/sensor/id ou /actuator/id	GET	Recupera um sensor ou atuador com todas as suas informações.
/sensor/id ou /actuator/id	PUT	Altera os dados de um sensor ou atuador.
/sensor/id ou /actuator/id	DELETE	Remove um sensor ou atuador.
/sensor/read/{id} ou /actuator/read/{id}	GET	Realiza a leitura de um sensor ou atuador.
/actuator/write/{id}	PUT	Envia um comando de atuação.

Tabela 5.2 – API do COIOT pata tratamento da fila de publicações

URI	Ação	Descrição
/publication	GET	Recupera a fila de publicações ainda não processada.
/publication	POST	Cadastra uma nova publicação contendo um ou mais sensores.
/publication/	DELETE	Remove a fila de publicação.
/publication/{id}	DELETE	Remove uma publicação.

O Processador de Contexto e o Motor de Regras, módulos dos Servidores de Contexto e Borda respectivamente, foram implementados a partir da ferramenta Business-Rules [3]. Business-Rules é um motor de regra do tipo ECA, desenvolvido em Python, cujas regras são estruturadas em um formato JSON. O uso do JSON é pertinente nesse caso pois simplifica a

[2] https://github.com/curl/curl

[3] https://github.com/venmo/business-rules

REQUEST *curl*

```
curl   --include \
       --header "username: admin" \
       --header "password: admin" \
https://amplus-ldas1.ufpel.edu.br/sensor/read/TemperaturaBOD1
```

RESPONSE

```
200 (OK)
Content-Type: application/json; charset=utf8
```

```
{
    "value": "47.4687",
    "date": "2014-10-23T18:57:05.600Z",
    "dt_start": "2014-10-23T18:57:05.496Z",
    "dt_end": "2014-10-23T18:57:05.600Z"
}
```

Figura 5.1 – Exemplo de uso da API REST

construção e distribuição das regras através dos componentes distribuídos da arquitetura do CoIoT, uma vez que esse é o formato padrão adotado na estruturação dos dados enviados através da API REST. O Repositório de Contexto, gerenciado pelo Servidor de Contexto, foi implementado em banco de dados relacional tendo sido utilizado para isso o gerenciador PostGreSQL. A modelagem do Repositório de Contexto é apresentada no Anexo B.

O Módulo Gerenciador de Eventos foi implementado no Eclipse Mosquitto [4]. Mosquitto é um *broker* de mensagens *open-source* que trabalha sob o protocolo MQTT para distribuir mensagens utilizando-se do modelo *publish/subscribe*. Por utilizar um protocolo leve e de baixo consumo energético, Mosquitto é um *broker* oportuno para ser usado em ambientes de IoT onde os componentes, normalmente, possuem baixa capacidade computacional.

O módulo Agendador foi implementado através do APScheduler (*Advanced Python Scheduler*) [5] com o objetivo de controlar a leitura dos sensores através de eventos de tempo. APScheduler consiste em uma ferramenta implementada na linguagem Python voltada à construção de aplicações que necessitem agendar a execução de tarefas. As tarefas agendadas podem ser executadas apenas uma vez ou periodicamente, podendo ser incluídas ou removidas dinamicamente.

[4] https://mosquitto. org
[5] https://apscheduler.readthedocs.io/

5.2 Estudo de caso 1: processamento distribuído de informações contextuais com colaboração vertical em ambiente de produção

Para avaliação das funcionalidades de coleta distribuída do CoIoT este estudo de caso foi realizado junto ao projeto plenUS (*plentiful Ubiquitous Systems*). O plenUS vem sendo desenvolvido por um consórcio de pesquisa envolvendo o GPPD, LUPS e a Embrapa Clima Temperado. O projeto plenUS tem como premissa o emprego do middleware CoIoT em regime de produção compreendendo 24 horas ao longo dos 7 dias da semanas.

O objetivo do plenUS é 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.

Este estudo de caso permitiu explorar a coleta distribuída de informações contextuais, seu processamento nos Servidores de Borda envolvidos e a disponibilização das mesmas no Servidor de Contexto para fins de armazenamento, processamento e geração de notificações. Essa abordagem em que parte do processamento acontece no Servidor de Borda e parte no Servidor de Contexto caracteriza uma colaboração vertical entre eles.

5.2.1 Análise de sementes na Embrapa Clima Temperado

A principal finalidade da análise de sementes é determinar a qualidade de um lote e, consequentemente, o seu valor para a semeadura. Um laboratório de análise de sementes (LAS) pode ser considerado um centro de controle de qualidade que dá suporte aos sistema de produção de sementes em todas as suas fases: semeadura, colheita, processamento, tratamento, armazenamento até a sua comercialização. A principal função do LAS é viabilizar a padronização dos processos exigidos através das regras estabelecidas pelas organizações que regulam a produção de sementes. Os resultados da análise 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 padronizada conforme legislação de cada espécie, é importante manter a produção sob controle e, dessa forma, a sua análise constitui um instrumento imprescindível (TILLMANN; MENEZES, 2012).

Para que o propósito das análises seja atingido é necessário que os laboratórios possuam equipamentos adequados e sigam métodos e procedimentos uniformes. As Regras para Análise de Sementes (RAS) descrevem procedimentos que padronizam as análises visando a obtenção de resultados uniformes independentemente dos analistas ou laboratórios que os realizem. Isso permite a obtenção de dados precisos e confiáveis (BRASIL, 2009).

Procurando atender as elevadas exigências dos processos de análise de sementes, o LASO (Laboratório de Análise de Sementes Oficial) da Embrapa Clima Temperado vem atuando de maneira ininterrupta desde janeiro de 1960. O LASO é o segundo laboratório de análise de sementes mais antigo do Brasil. A Embrapa (Empresa Brasileira de Pesquisa Agropecuária), por sua vez, é uma empresa estatal ligada ao Ministério da Agricultura, Pecuária e Abastecimento (MAPA) que possui unidades espalhadas por todo território Brasileiro. Sua função é planejar, supervisionar, coordenar e controlar as atividades relacionadas à execução de pesquisa agropecuária e à formulação de políticas agrícolas [6].

5.2.2 Desafios no controle de laboratórios para análise de sementes

Os procedimentos associados à análise de sementes exigem precisão dos equipamentos estabelecendo limites específicos para variação das grandezas físicas. No caso dos testes de germinação, o processo é realizado em germinadores do tipo câmara BOD (*Biochemical Oxygen Demand*) que criam um ambiente controlado conforme estabelecem as Regras para Análise de Sementes (BRASIL, 2009). As principais grandezas que devem ser controladas são: temperatura, umidade e luminosidade.

Algumas espécies exigem apenas que a temperatura e umidade se mantenham dentro de limites específicos, já outras necessitam a alternância de temperatura e iluminação entre dia e noite. Independentemente da cultura analisada, a variação de temperatura não deve ser maior do que 2°C em cada período de 24 horas. Quando temperaturas alternadas são indicadas, a temperatura mais baixa deve ser mantida durante 16 horas e a mais alta por oito horas, simulando o dia e a noite, respectivamente. Quanto às sementes para as quais a luz é indicada, estas devem ser iluminadas durante, no mínimo, oito horas a cada ciclo de 24 horas, criando uma percepção de dia (BRASIL, 2009)(International Seed Testing Association, 2017).

Além disso, o equipamento deve ser capaz de manter uma temperatura uniforme em toda a câmara, ou seja, cada prateleira deve ter a mesma temperatura. Isso exige que o monitoramento da temperatura seja realizado em diversos pontos no interior do BOD a fim

[6] https://www.embrapa.br/

de detectar eventuais obstruções no fluxo de ar que comprometam a homogeneização da temperatura.

Conforme exigência legal do Ministério da Agricultura Pecuária e Abastecimento Brasileiro, os Laboratórios de Análise de Sementes (LAS) necessitam implementar um sistema de qualidade baseado na norma ABNT NBR ISO/IEC 17025:2005. Esta norma estabelece os critérios que devem ser atendidos pelos laboratórios que desejam caracterizar sua competência técnica a fim de garantir a qualidade nos resultados fornecidos.

Com a implementação do sistema de qualidade nos LAS aumentaram as exigências nos controles dos equipamentos onde são realizadas as análises, visando precisão e confiabilidade nos resultados. Com isso, são necessários registros diários nas leituras de temperatura dos germinadores, câmaras e salas. Muitos laboratórios ainda utilizam planilhas em papel que são preenchidas manualmente. Alguns laboratórios possuem *dataloggers*, no entanto, embora os dados coletados possam ser armazenados digitalmente, raramente há atuação automatizada em decorrência desses dados. Com isso, muitas das análises são perdidas por atraso na detecção de problemas nos germinadores. Sendo a agricultura uma área estratégica para o Brasil, as perdas de lotes de sementes analisados, em uma escala nacional, resulta em atrasos nas vendas de sementes e consequentemente na época de semeadura, o que resulta em perda de divisas por todo o país.

Considerando as demandas dos LAS, a utilização de soluções ubíquas nesses cenários possibilita produzir e registrar informações de contexto através do monitoramento de equipamentos 24h por dia, sete dias por semana, sem necessariamente exigir interferência humana. Além disso, o uso de regras de processamento de contexto possibilita avaliar os dados coletados, disparando notificações ou atuações, quando necessário, como envio de correio eletrônico, mensagens de texto, ativação de luminosos, etc.

Uma característica particular do LASO é a utilização amplamente compartilhada dos equipamentos do laboratório por pesquisadores internos e externos, sendo utilizados em diversas atividades associadas à análise de sementes, como no apoio às pesquisas, aulas práticas de pós-graduação e cursos de treinamentos. Isso implica que a geração de notificações deve ser personalizada e direcionada ao usuário que está utilizando o equipamento no momento.

5.2.3 plenUS: Infraestrutura de hardware concebida

Para dar suporte ao cenário de avaliação viabilizando a coleta das informações contextuais dos equipamentos do LASO, bem como a atuação no ambiente foi concebida

uma infraestrutura conforme é apresentada na Figura 5.2. Nesta Figura são representados os equipamentos do laboratório e os sensores que fazem a leitura das respectivas grandezas físicas monitoradas. Também são representados os dispositivos em que são alocados os Servidores de Contexto, Servidor de Borda e os Gateways.

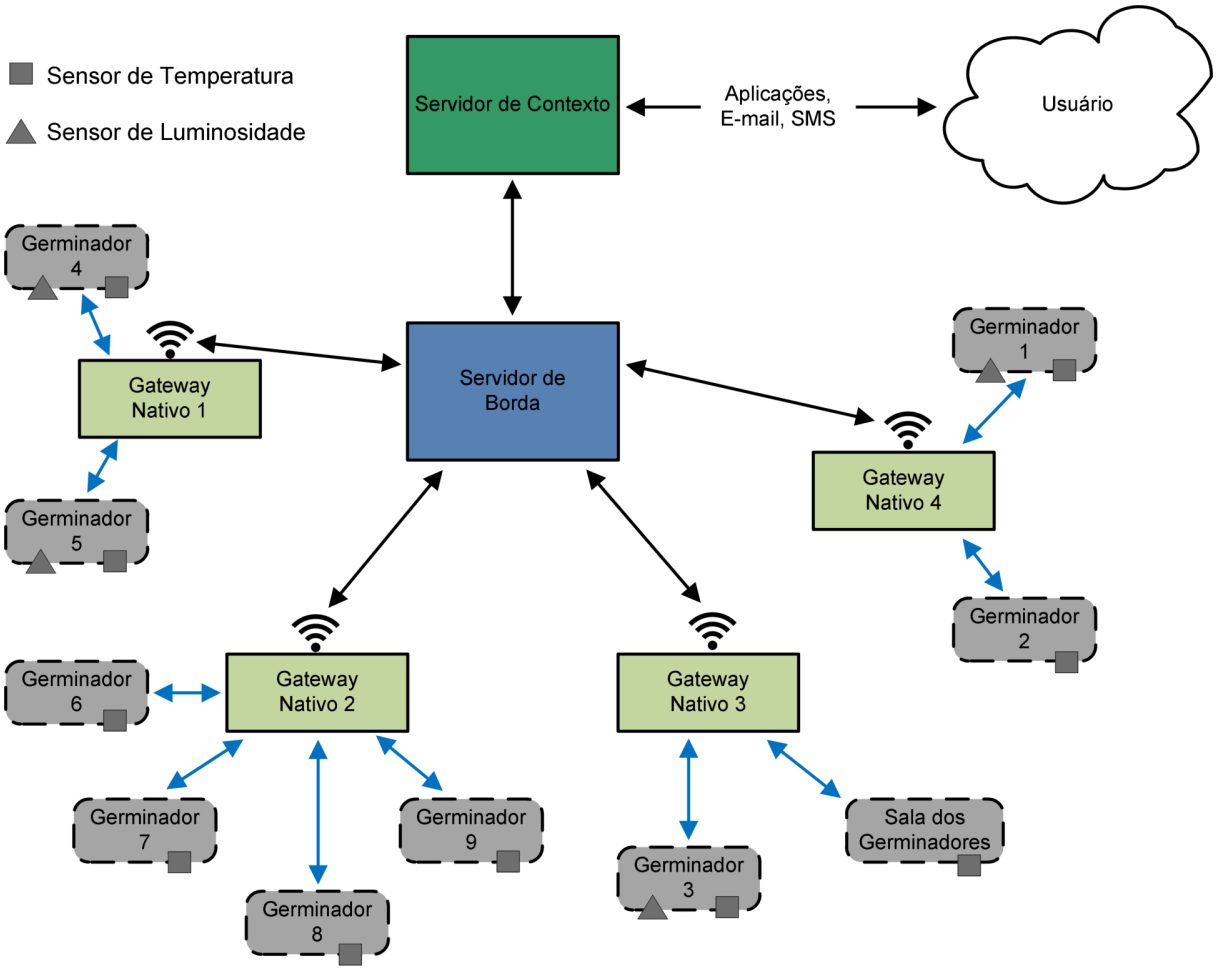

Figura 5.2 – Infraestrutura concebida para o LASO

O Servidor de Contexto foi instalado nas dependências do Núcleo de Tecnologia da Informação da Embrapa, localizado na unidade Sede da Embrapa Clima Temperado. Os critérios utilizados para a definição da localização do Servidor de Contexto consideram fatores, tais como: climatização adequada, sistema de fornecimento de energia ininterrupto (no-break e gerador), bem como acesso facilitado para manutenção do equipamento. 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.

O Servidor de Borda foi instalado no LASO, que é situado na unidade Estação Experimental Terras Baixas (ETB). O critério utilizado no posicionamento do Servidor de Borda foi a proximidade do ambiente monitorado, minimizando assim a dependência de redes de comunicação controladas por terceiros. O hardware utilizado para o Servidor de Borda foi a Raspberry PI (RASPBERRY, 2014)(BROCK; BRUCE; CAMERON, 2013) (vide Figura 5.3(C)) com Sistema Operacional Raspbian. A Raspberry Pi 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 se usuários bastante ativa e que baseia-se nos princípios de software livre. A Raspberry Pi vem sendo utilizada há algum tempo pelo grupo de pesquisa, em soluções de interfaceamento com dispositivos que interagem com o ambiente, como sensores e atuadores. Raspbian é o principal Sistema Operacional customizado para a Raspberry Pi e tem por base o sistema Operacional Debian.

Já os gateways representam dispositivos IoT com poder computacional limitado o que introduziu desafios quando da sua concepção. Para a escolha do hardware utilizado na concepção do Gateway Nativo do CoIoT (vide Figura 5.3(A)) 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 programado através da plataforma de software do Arduino[7], bem como em Linguagem Lua através de um firmware denominado nodeMCU[8]. Para a prototipação do Gateway Nativo, foi utilizado o kit de desenvolvimento do NodeMCU[9] que possui integrado o processador ESP8266-12, bem como entrada mini USB e acesso facilitado ao pinos GPIOs do processador (vide Figura 5.3(B)).

Para realizar a atuação e a captura das informações contextuais do ambiente optou-se por utilizar no LASO um conjunto de dispositivos constituído por sensores e atuadores. Os sensores selecionados para o estudo de caso são baseados na tecnologia 1-Wire (Maxim Integrated, 2016)(AWTREY; SMITH; LISSIUK, 2004) e são controlados diretamente pelo Servidor de Borda através do Gateway Virtual. A tecnologia 1-Wire caracteriza-se como uma rede de transmissão de dados, baseada em dispositivos eletrônicos endereçáveis, e tem

[7] https://www.arduino.cc/en/Reference/
[8] https://github.com/nodemcu/nodemcu-firmware/wiki/
[9] http://www.nodemcu.com/

se destacado por sua versatilidade e facilidade de implementação. Os sensores de temperatura utilizados são do modelo DS18B20 e são apresentados na Figura 5.3(D). A atuação no ambiente é viabilizada através de alerta visual construído a partir de um dispositivo emissor de luz de elevada intensidade encapsulado com lente difusora de luz para elevar a sua visibilidade e que interopera com a rede 1-Wire através de um dispositivo DS2413 (vide Figura 5.3(B)). Um maior detalhamento sobre os dispositivos desenvolvidos para o Projeto plenUS em desenvolvimento na Embrapa Clima Temperado é apresentado no Anexo A.

A Tabela 5.3 apresentadas algumas especificações referentes aos dispositivos de coleta das informações contextuais dos equipamentos do LASO.

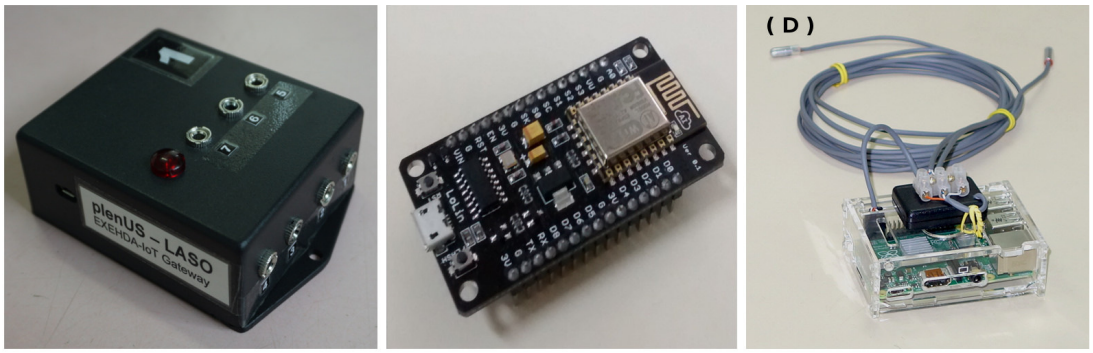

Figura 5.3 – Dispositivos para interação com o ambiente: (A) Gateway Nativo; (B) NodeMCU; (C) Raspberry PI; (D) Sensor de temperatura DS18B20

5.2.4 plenUS: Infraestrutura de software desenvolvida

A infraestrutura de software desenvolvida para o LASO é formada por uma camada de middleware composta pelo CoIoT e uma camada de aplicação denominada LASApp (aplicação para Laboratórios de Análise de Sementes) que serve de interface com o usuário. A camada de middleware fornece suporte autônomo às demandas do LASO e garante que o ambiente permanece sendo monitorado mesmo que a aplicação LASApp não esteja em execução.

Tabela 5.3 – Especificações de coleta das Informações contextuais do LASO

Ambiente	**Gateway**	**Grandeza**	**Sensor**	**Faixa de Operação**
Sala das Balanças	3	Temperatura	DS2438	10°C à 30°C
	3	Umidade	DS2438 + HIH-4000	50% à 98%
Conserv. Amostras	3	Temperatura	DS2438	10°C à 25°C
	3	Umidade	DS2438 + HIH-4000	50% à 98%
Germinador 01	3	Temperatura	DS18B20 Waterproof	18°C à 22°C
	3	Presença de luz	DS2438 + LDR	12h - sem luz 12h - com luz
Germinador 02	3	Temperatura	DS18B20 Waterproof	08°C à 12°C
Geladeira	3	Temperatura	DS18B20 Waterproof	03°C à 07°C
Germinador 03	4	Temperatura	DS18B20 Waterproof	18°C à 22°C
	4	Presença de luz	DS2438 + LDR	12h - sem luz 12h - com luz
Sala dos Germinadores	4	Temperatura	DS18B20	20°C à 27°C
Germinador 04	1	Temperatura	DS18B20 Waterproof	23°C à 27°C
Germinador 05	1	Temperatura	DS18B20 Waterproof	23°C à 27°C
Germinador 06	1	Temperatura	DS18B20 Waterproof	23°C à 27°C
Germinador 07	1	Temperatura	DS18B20 Waterproof	18°C à 27°C
Germinador 08	2	Temperatura	DS18B20 Waterproof	com luz - 28°C à 32°C sem luz - 18°C à 22°C
	2	Presença de luz	DS2438 + LDR	8h - com luz 16h - sem luz
Germinador 09	2	Temperatura	DS18B20 Waterproof	com luz - 28°C à 32°C sem luz - 18°C à 22°C
	2	Presença de luz	DS2438 + LDR	8h - com luz 16 - sem luz

Suporte de middleware CoIoT para o plenUS

Considerando o elevado compartilhamento da infraestrutura do LASO entre os usuários e as necessidades operacionais apontadas por estes, a abordagem para Ciência de Contexto concebida para o laboratório contempla informações contextuais de duas naturezas: (i) contexto do ambiente e, (ii) contexto dos usuários. O contexto do ambiente contempla todas as

informações físicas sensoreadas do ambiente interno do laboratório e de seus equipamentos. O contexto do usuário, por sua vez, consiste nas especificações feitas pelo usuário as quais são armazenadas em seu perfil. O cenário operacional do LASO abordado é ilustrado através da Figura 5.4 e envolve o monitoramento das condições internas dos BODs em experimentos que são exigidas variações de temperatura e iluminação para simular dia e noite.

O cenário consiste em monitorar a temperatura e iluminação interna de cada um dos BODs do LASO, gerar notificações e produzir registro histórico. A temperatura é coletada periodicamente e comparada com os padrões para dia e noite especificados nas regras de análise para a semente que está em avaliação. Então são geradas alertas visuais no ambiente bom como notificações aos usuários envolvidos nos experimentos em execução segundo as preferências pessoais, sempre que a temperatura varia em mais que 2°C, para mais ou para menos, em relação aos valores especificados. Também são geradas notificações quando o tempo de transição entre dia e noite ultrapassar o limite de 30 minutos e quando a luz não for ativada no período de tempo correspondente ao dia ou for desativada antes de completar as 16 horas esperadas.

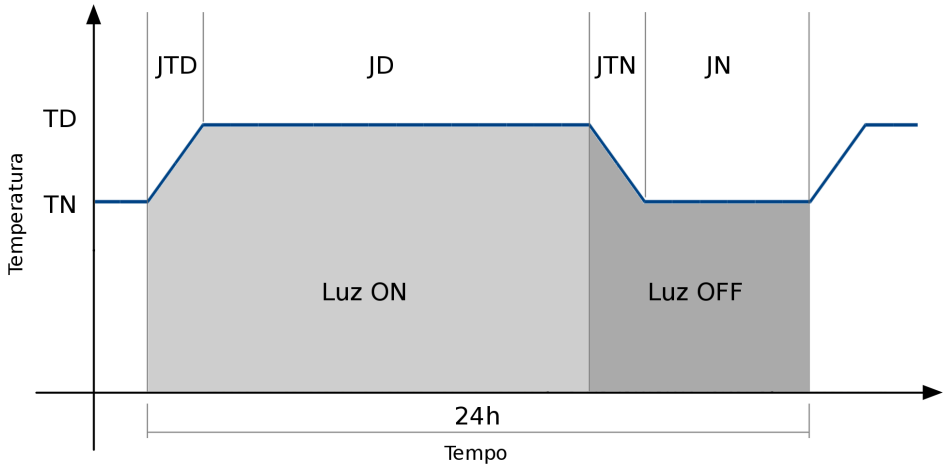

Figura 5.4 – Janelas temporais - TD: temperatura dia; TN: temperatura noite; JTD: janela temporal de transição dia; JD: janela temporal dia; JTN: janela temporal Noite; JN: janela temporal noite

O suporte de middleware para o cenário considera as configurações de agendamento de leituras dos sensores em períodos regulares e um conjunto de regras de processamento de contexto. Ambos podem ser ajustados pelos usuários conforme as necessidades operacionais através da interface da aplicação desenvolvida para o LASO.

As regras de processamento de contexto utilizadas foram organizadas entre os Servidores de Borda e Contexto de modo a atender o cenário proposto. Os critérios utilizados na distribuição das regras foram: (i) minimização no fluxo de dados e, (ii) continuidade do monitoramento mesmo em momentos de perda de comunicação entre os servidores.

No Servidor de Borda, as regras utilizadas no cenário são organizadas em grupos conforme apresentados na Tabela 5.4. Esses grupos controlam cada uma das janelas temporais presentes no cenário (vide Figura 5.4). A cada troca de janela temporal um grupo de regras é ativado enquanto os demais grupos são desativados. As janelas de transição dia (JTD) e de transição noite (JTN) são ativadas quando a luz é ligada e desligada respectivamente. As janelas dia (JD) e noite (JN), por sua vez, são ativadas quando a temperatura alvo de cada janela é atingida ou quando o tempo limite é excedido. A partir do momento que as janelas JD e JN são ativadas, as respectivas regras verificam se essas temperaturas alvo são mantidas dentro dos limites (TD$^{+}/_{-}$2°C para dia e TD $^{+}/_{-}$2°C para noite) por todo período de validade de cada janela, gerando sinais de alertas aos usuários quando isso não ocorrer.

Nesse estudo de caso, as funções atribuídas ao Servidor de Contexto são de gerenciar o armazenamento histórico das informações contextuais e de gerar alertas aos usuários cadastrados quando necessário. As regras do servidor de Contexto são apresentadas na Tabela 5.5.

Tabela 5.4 – Regras de controle - Servidor de Borda

Nome da Regra	**Grupo**	**Evento**	**Condição**	**Ação**
Publica a temperatura	Default	a cada 30 minutos	sempre verdadeira	Publica a temperatura
Temperatura Noite fora da faixa	JN	Temperatura é lida	Temperatura $<$ TN-2 OU Temperatura $>$ TN+2	Envia alerta ao usuário Aciona alerta luminoso do BOD Aciona alerta luminoso central
Troca para JTD	JN	Luz é lida	Luz $==$ on	Ativa grupo JTD Desativa grupo JN Publica Luz
Tempo excedido da JTD	JTD	a cada 1 minuto	Tempo da JTD $>$ 30	Envia alerta ao usuário Aciona alerta luminoso do BOD Aciona alerta luminoso central
Troca para JD	JTD	Temperatura é lida	Temperatura $>$ TD-2	Ativa grupo JD Desativa grupo JTD
Temperatura dia fora da faixa	JD	Temperatura é lida	Temperatura $<$ TD-2 OU Temperatura $>$ TD+2	Envia alerta ao usuário Aciona alerta luminoso do BOD Aciona alerta luminoso central
Troca para JTN	JD	Luz é lida	Luz $==$ off	Ativa grupo JTN Desativa grupo JD Publica Luz
Tempo excedido da JTN	JTN	a cada 1 minuto	Tempo da JTN $>$ 30	Envia alerta ao usuário Aciona alerta luminoso do BOD Aciona alerta luminoso central
Troca para JN	JTN	Temperatura é lida	Temperatura $<$ TN+2	Ativa grupo JN Desativa grupo JTN

Tabela 5.5 – Regras de controle - Servidor de Contexto

Nome da regra	**Grupo**	**Evento**	**Condição**	**Ação**
Persiste a temperatura	Default	Temperatura é publicada	sempre verdadeiro	Persiste a Temperatura no Repositório de Contextos
Persiste a Luz	Default	Luz é publicada	sempre verdadeira	Persiste a Luz no Repositório de Contextos
Alerta o usuário	Default	Alerta ao usuário é publicado	sempre verdadeira	Envia alerta ao usuário

LASApp: uma aplicação do plenUS para laboratórios de análise de sementes

Atendendo a premissa de que as informações devem estar acessíveis independentemente do dispositivo computacional utilizado, as tecnologias empregadas em aplicações no âmbito da IoT devem possuir recursos que possibilitem a auto adaptação quando da ocorrência de mudanças no ambiente de execução. Considerando isto, no desenvolvimento da LASApp foram empregadas 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 quanto por *tablets* e *smartphones*. A LASApp também é responsável por criar uma interface intuitiva para que o usuário possa gerenciar os recursos disponibilizados de maneira simples.

A aplicação desenvolvida possui dois módulos principais que contemplam os modos de operação disponibilizados. O Módulo de Gerenciamento permite gerenciar os diversos recursos do laboratório, desde os aspectos administrativos, como o perfil dos usuários e o agendamento dos equipamentos, bem como os aspectos associados ao tratamento das informações contextuais, como a coleta dos dados (sensoreamento), o seu processamento e a atuação sobre o ambiente. Esse módulo foi desenvolvido tendo por princípio o suporte a diferentes níveis de acesso de acordo com o perfil do usuário. O Módulo de Visualização, por sua vez, disponibiliza diversas ferramentas de visualização que permite acessar o histórico das informações contextuais de diferentes maneiras. Sendo assim, LASApp constitui importante ferramenta no auxílio ao pesquisador do laboratório para a avaliação do comportamento das variáveis físicas envolvidas nos processos analisados.

Módulo de Gerenciamento

O Módulo de Gerenciamento da aplicação LASApp possibilita administrar a infraestrutura do LASO no que tange ao suporte de middleware provido pelo CoIoT o qual envolve coleta, atuação, processamento de contexto e notificação. A LASApp também contempla o gerenciamento de aspectos próprios do cenário avaliado os quais são características do perfil operacional do laboratório.

O acesso ao Módulo de Gerenciamento se dá por meio de login, sendo que cada usuário poderá ter um nível de acesso diferenciado aos menus de acordo com a permissão concedida. Esse menus permitem acessar as diversas funcionalidades disponibilizadas pela LASApp e são apresentados na Figura 5.5.

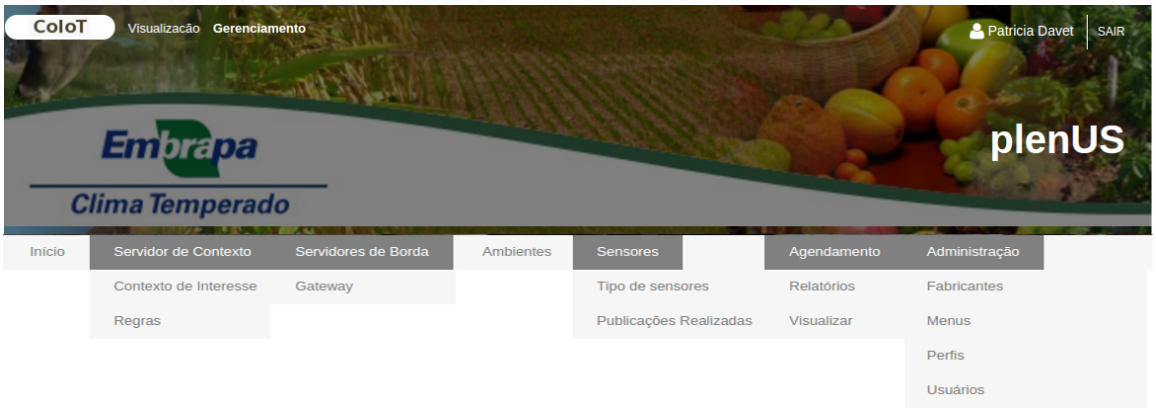

Figura 5.5 – Menus de acesso

As funcionalidades oferecidas pelo Módulo de Gerenciamento contemplam a manipulação de regras de processamento de contexto, agendamentos e *triggers* de leitura dos sensores, bem como a configuração de acesso aos dispositivos de coleta e atuação, como os Servidores de Borda, gateways, sensores e atuadores. Essas funcionalidades são disponibilizadas através de interfaces da aplicação, sendo algumas destas apresentadas pelas Figuras 5.6 e 5.7.

O cadastro e/ou remoção dos dispositivos de sensoreamento e/ou atuação 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 é finalizado mediante autenticação pelo usuário, evitando assim o cadastro de dispositivos não reconhecidos.

A aplicação LASApp também contempla em seu Módulo de Gerenciamento funcionalidades direcionadas ao cenário de avaliação, como o cadastro de usuário, de

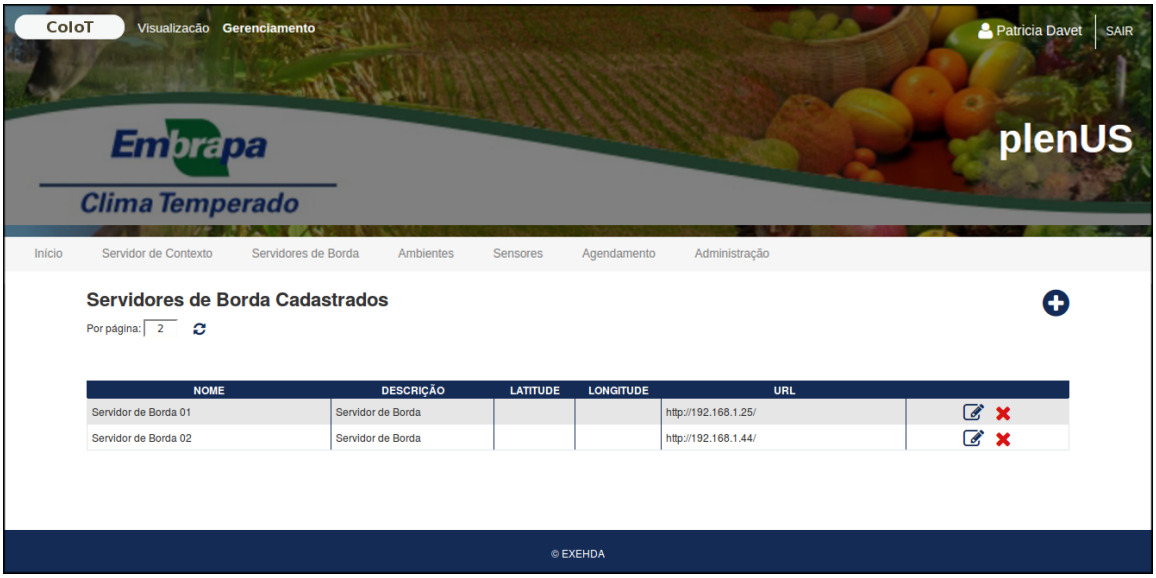

Figura 5.6 – Configuração dos Servidores de Borda

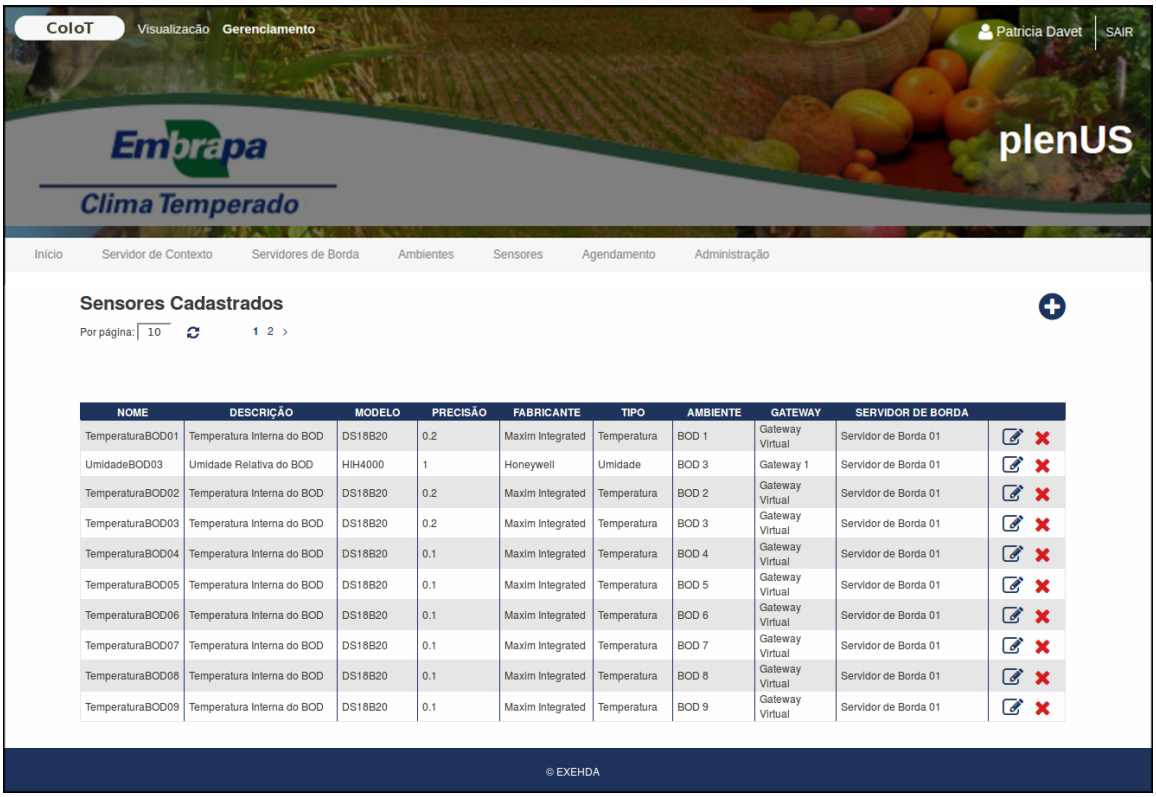

Figura 5.7 – Configuração de sensores

fabricantes, de equipamentos do LASO, e o agendamento desses equipamentos aos usuários de modo a viabilizar a utilização compartilhada. A interface de gerenciamento dos agendamentos, além dos botões de controle comuns às demais interfaces, possui um botão com uma funcionalidade específica. Este botão tem a função de apresentar, de forma gráfica, o histórico das informações contextuais do equipamento durante o período do agendamento, o qual geralmente corresponde à janela de tempo de um experimento (vide Figura 5.8).

Figura 5.8 – Tela de agendamentos de equipamentos

O usuário, ao ser cadastrado, assume um perfil padrão que pode ser alterado posteriormente. Um aspecto fundamental das configurações do perfil do usuário consiste na definição das preferências de notificações. Essa configuração se reflete no comportamento das regras de processamento de contexto do CoIoT considerando as preferências pessoais configuradas.

Módulo de Visualização

O Módulo de Visualização da aplicação LASApp possibilita a seleção do contexto de interesse a ser exibido, que pode ser apresentado na forma de um relatório textual (vide Figura 5.9) ou através de um modo gráfico (vide Figura 5.10). Através desse Módulo o pesquisador do LASO pode a ter acesso à visualização das variações dos valores das grandezas físicas monitoradas no BOD durante os períodos de análise, os quais influenciam diretamente nos resultados dos processos de germinação das sementes.

O relatório gráfico desenvolvido permite visualizar as curvas de variação dos valores de diversas grandezas sensoriadas utilizados no LASO. A seleção dos sensores a serem visualizados é feita a partir de um menu com suporte a múltipla seleção. Também é

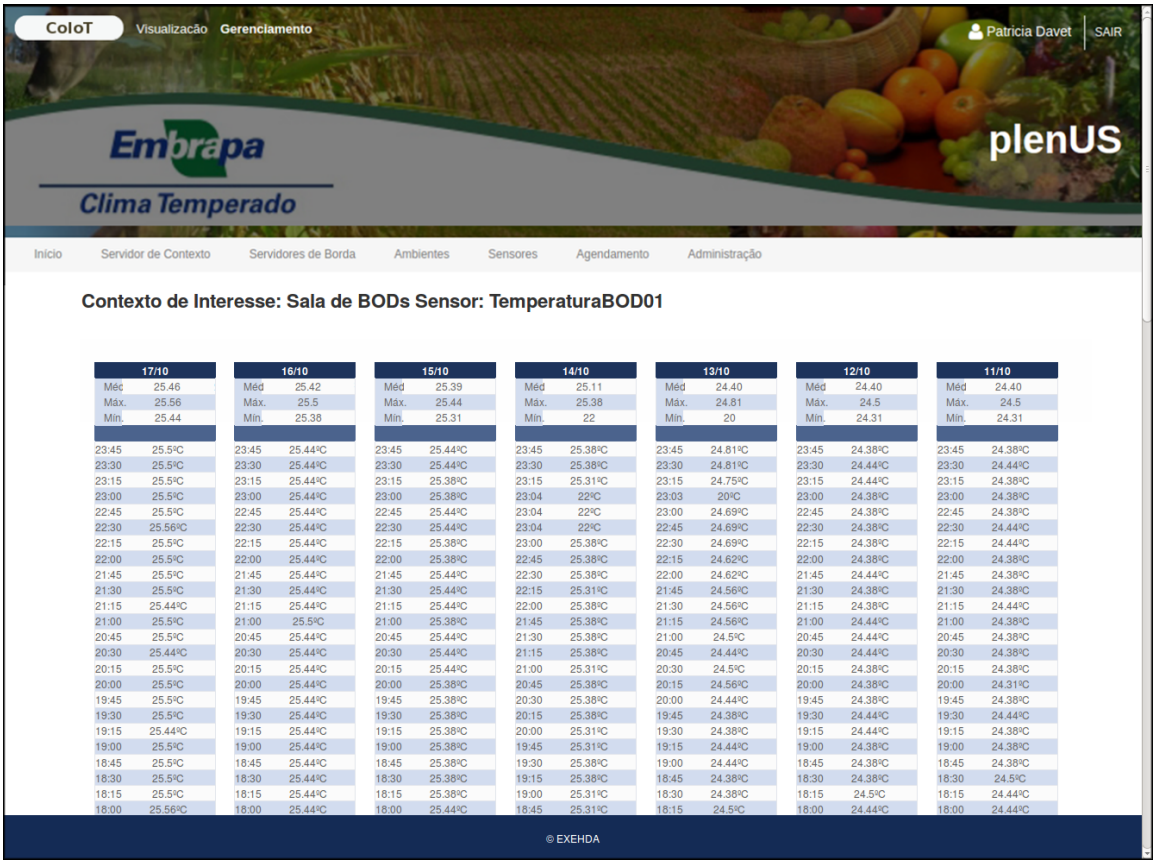

Figura 5.9 – Relatório em modo textual

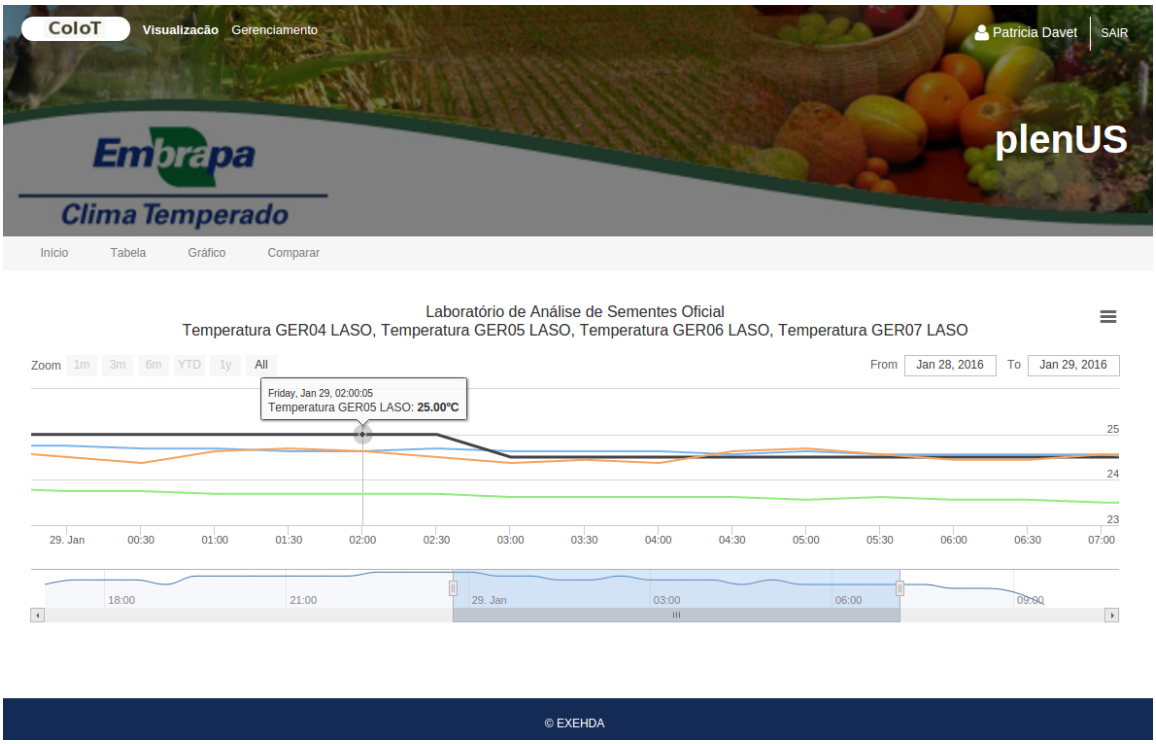

Figura 5.10 – Relatório em modo gráfico

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 sensoreados.

5.2.5 Avaliação de aceitação do estudo de caso

Esta seção apresenta um detalhamento do experimento e os resultados obtidos com a avaliação de aceitação das aplicações desenvolvidas. O estudo envolveu 10 voluntários, entre professores, alunos e técnicos, com atividades relacionadas ao LASO. Cada participante utilizou um desktop para acessar a ferramenta. Após a realização de um treinamento básico, os participantes utilizaram a ferramenta de visualização e responderam um questionário de avaliação, considerando a experiência de uso.

O questionário foi construído com base no Modelo de Aceitação de Tecnologia (TAM) (YOON; KIM, 2007), usando uma escala de Likert. Para a aceitação da ferramenta o modelo TAM considera: (*i*) Facilidade de uso: grau em que o usuário avalia que a ferramenta pode reduzir seu esforço; e (*ii*) Percepção de utilidade: grau em que o usuário avalia que a ferramenta pode melhorar a sua experiência.

As tabelas 5.6 e 5.7 contêm, respectivamente, o questionário aplicado aos usuários e as respostas obtidas para facilidade de uso e percepção de utilidade. Em ambas as tabelas, a primeira coluna corresponde a questão, as seguintes cinco colunas apresentam os resultados obtidos em cada escala, em graus relativos e absolutos, e a última coluna mostra a média consolidada da percentagem, variando de 0 a 5.

Tabela 5.6 – Avaliação da facilidade de uso

Questão	Discordo Total-mente	Discordo Parcial-mente	Indiferente	Concordo Parcial-mente	Concordo Total-mente	Média
1. A ferramenta é fácil de entender.	0,0%(0)	0,0%(0)	0,0%(0)	40,0%(4)	60,0%(6)	4,6
2. A ferramenta é fácil de usar.	0,0%(0)	0,0%(0)	0,0%(0)	30,0%(3)	70,0%(7)	4,7
3. As opões são claras e objetivas.	0,0%(0)	0,0%(0)	10,0%(1)	20,0%(2)	70,0%(7)	4,6
4. Com pouco esforço consigo selecionar um contexto de interesse.	0,0%(0)	0,0%(0)	0,0%(0)	20,0%(2)	80,0%(8)	4,8
5. Com pouco esforço consigo acessar os relatórios gráficos.	0,0%(0)	0,0%(0)	0,0%(0)	30,0%(3)	70,0%(7)	4,7

Tabela 5.7 – Avaliação da percepção de utilidade

Questão	Discordo Totalmente	Discordo Parcialmente	Indiferente	Concordo Parcialmente	Concordo Totalmente	Média
1. As opções apresentadas são relevantes.	0,0%(0)	0,0%(0)	0,0%(0)	30,0%(3)	70,0%(7)	4,7
2. A ferramenta facilita a obtenção de dados de contexto envolvendo múltiplos sensores.	0,0%(0)	0,0%(0)	0,0%(0)	40,0%(4)	60,0%(6)	4,6
3. A ferramenta facilita a atuação imediata a partir da emissão de um alerta ou mensagem.	0,0%(0)	0,0%(0)	30,0%(3)	30,0%(3)	40,0%(4)	4,1
4. Eu usaria essa ferramenta no meu trabalho.	0,0%(0)	0,0%(0)	30,0%(3)	20,0%(2)	50,0%(5)	4,2

Analisando os resultados pode-se observar que a solução desenvolvida para o LASO obteve aprovação, tanto para facilidade de uso, como para percepção de utilidade. Entretanto, ocorreram resultados na escala "indiferente" nas duas últimas questões da percepção de utilidade. Isso pode ser interpretado como uma preocupação com o controle da qualidade dos experimentos desenvolvidos no LASO, em função do uso de mecanismos autônomos, sem a usual intervenção humana, para a emissão de alertas para estados contextuais que exijam uma atuação imediata. Nesse caso, uma estratégia que pode ser adotada é intensificar os testes e validações com os usuários e dar continuidade a implantação do COIOT e das aplicações para associadas ao estudo de caso.

5.3 Estudo de caso 2: processamento distribuído de informações contextuais com colaboração híbrida em ambiente simulado

A produção de uvas e, por consequência, de vinhos de qualidade tem sido uma característica marcante do estado do Rio Grande do Sul, principalmente da serra Gaúcha e recentemente da região sul do estado. A produção de uvas, bem como de seus derivados, tem importante papel na economia do estado. A EMBRAPA tem contribuído para o crescimento dessa área, possuindo inclusive uma unidade especificamente voltada a essa frente de pesquisas, a Embrapa Uva e Vinho [10]. Além disso, a uva por ser um produto com elevado valor agregado tem grande potencial para aplicação de tecnologias de viticultura de precisão. Esses aspectos foram a motivação central para a elaboração desse estudo de caso envolvendo esta área.

[10] https://www.embrapa.br/uva-e-vinho

Este estudo de caso explorou o emprego do CoIoT no gerenciamento da irrigação na viticultura de precisão. Foi feito um estudo das necessidades da viticultura de precisão no que tange ao monitoramento das variáveis físicas da lavoura tendo por base um parreiral típico da região sul do estado. Baseado nesse estudo foi elaborada uma estratégia de distribuição de sensores, gateways e Servidores de Borda para darem suporte à coleta dessas variáveis, bem como ao processamento de contexto através de um conjunto de regras distribuídas. A avaliação foi feita empregando um ambiente de IoT virtual concebido através de emulação.

5.3.1 A irrigação na viticultura de precisão

A agricultura de precisão, e em particular a viticultura de precisão, pode ser entendida como a gestão da variabilidade temporal e espacial das áreas cultivadas com o objetivo de melhorar o rendimento econômico da atividade agrícola, quer pelo aumento da produtividade e/ou qualidade, quer pela redução dos custos de produção, reduzindo também o seu impacto ambiental e risco associado. Variabilidade temporal e espacial diz respeito à característica da área agrícola que se refere às variações dos fatores importantes das culturas (físico/químicos do solo ou climáticos) ao longo da área plantada e do tempo de desenvolvimento da planta (BRAGA; NETO, 2009).

Nesse sentido, uma das principais técnicas utilizadas na agricultura de precisão é a tecnologia de taxa variável. Essa técnica consiste em uma detalhada caracterização da variabilidade espacial da cultura e em seguida na gestão de fatores de produção (rega, fertilizantes, etc.) e operações culturais (BRAGA; NETO, 2009).

Uma das questões fundamentais na produção de vinhos de alta qualidade é o momento certo de irrigar. O uso da água por um parreiral varia conforme o estágio de desenvolvimento da cultura. Há, normalmente, uma baixa demanda no início da fase de crescimento, devido à menor área foliar das plantas. Segue-se um período de alta demanda que ocorre quando o dossel está plenamente desenvolvido. Isso caracteriza a necessidade de mudança das regras que controlam os sistemas de irrigação ao longo do tempo (CONCEIÇÃO, 2008).

Outro aspecto importante em relação ao gerenciamento de sistemas de irrigação, diz respeito à capacidade de campo do solo. Capacidade de campo consiste na característica do solo em reter de água após o excesso ter sido drenado. Nesse aspecto, os solos podem ser classificados como de alta, média e baixa capacidade de retenção de água, havendo ainda outros tipos intermediários. Dependendo da extensão do parreiral, esses vários tipos de solo podem estar presentes. Para não prejudicar o desenvolvimento e a produção de frutos das videiras,

deve-se evitar que a reserva hídrica do solo se esgote. Portanto, as estratégias de gerenciamento da irrigação devem variar espacialmente ao longo da área cultivada (CONCEIÇÃO, 2008).

Nem toda água existente no solo está disponível para as plantas. Quanto mais fortemente a água estiver retida no solo, maior será a quantidade de energia que a planta terá que gastar para absorver a água necessária ao atendimento das suas demandas metabólicas. Neste sentido, a tensão de água no solo é uma grandeza que representa a medida da quantidade de energia requerida pelas plantas para extrair água do solo. Portanto medidas de tensão podem ser utilizadas para avaliar indiretamente a deficiência hídrica das plantas (MAROUELLI et al., 2011). A tabela 5.8 ilustra a condição hídrica do solo conforme o valor da tensão de água.

Tabela 5.8 – Condição do solo de acordo com o valor de tensão de água (MAROUELLI et al., 2011)

Tensão (kPa)	Condição e interpretação
0 a 6	Solo próximo à saturação. Tensões nessa faixa por períodos contínuos indicam irrigações em excesso, perda de água por drenagem profunda, lixiviação de nutrientes e deficiência de aeração para as raízes
6 a 10	Solo com umidade próxima à capacidade de campo. Irrigações devem ser interrompidas nessa faixa a fim de prevenir os problemas associados à condição de solo próximo à saturação. A capacidade de campo em solos arenosos está associada a menores valores de tensão
10 a 20	Solo com excelente condição de umidade e boa aeração. Faixa de tensão indicada para a irrigação de culturas altamente sensíveis ao deficit de água, de solos de textura grossa e/ou via gotejamento
20 a 40	Solo com boa condição de umidade e excelente aeração. Faixa de tensão indicada para a irrigação de culturas sensíveis ao deficit de água
40 a 70	Solo com disponibilidade limitada de umidade e excelente aeração. Faixa de tensão indicada para a irrigação de culturas com tolerância moderada ao deficit de água
> 70	Solo com disponibilidade restrita de água e excelente aeração. Condição indicada apenas para culturas tolerantes ao deficit de água ou durante estádios específicos de desenvolvimento de algumas espécies. Tensão máxima operacional para tensiômetros em virtude da entrada de ar através da cápsula e formação acentuada de bolhas de vapor de água dentro do sistema

A Figura 5.11 ilustra a forma como a tensão de água do solo afeta a produtividade da cultura. Pode se observar que existe uma região da curva em que ocorre uma variação muito pequena da produtividade, mas que a partir de um determinado valor da tensão da água no solo a produtividade cai bruscamente. Esse valor, chamado de tensão crítica (Tc), indica o momento em que se deve proceder a irrigação (CALBO; SILVA, 2005). Os valores de tensão crítica

variam de acordo com o sistema de irrigação adotado, o estádio de desenvolvimento da cultura, a textura do solo e a demanda evaporativa da atmosfera. Os valores de tensão crítica para uva de mesa variam entre 15kPa e 25kPa e para uva vinífera variam entre 15kPa e 50kPa, sendo os valores mais baixos usados em condições de alta demanda evaporativa da atmosfera e/ou solos arenosos (MAROUELLI et al., 2011).

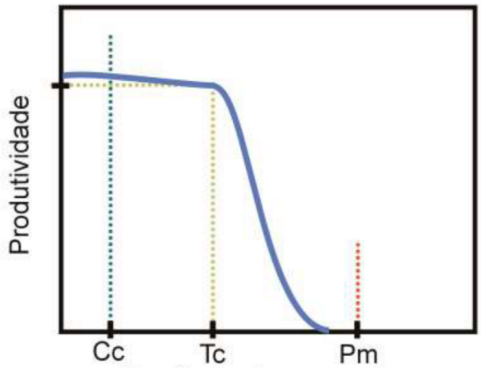

Figura 5.11 – Tensão de água no solo em relação à produtividade da cultura (CALBO; SILVA, 2005)

5.3.2 Infraestrutura de hardware e software utilizada

Para a emulação do ambiente de IoT utilizado no estudo de caso foi selecionado o software *Common Open Research Emulator* (CORE) (AHRENHOLZ et al., 2008), o qual permite a criação e customização de um ambiente computacional em rede em que cada nodo é capaz de executar diferentes processos sobre um sistema operacional FreeBSD. Outro fator que contribuiu para a adoção do CORE foi a experiência do grupo na sua utilização em testes de projetos de infraestruturas de rede. Através do CORE foi criado um ambiente virtual em que foi executado o protótipo do CoIoT com todo seu potencial, onde cada nodo processa seus dados individualmente interagindo com os demais de maneira colaborativa.

O hardware utilizado no estudo de caso foi um equipamento tipo *desktop* com processador Intel Pentium i5-2310 2.9GHz de quatro núcleos, com 8Gb de memória RAM e com o Sistema Operacional Ubuntu Server. A distribuição Linux Ubuntu foi escolhida por ter ampla utilização pelo grupo de pesquisa em seus equipamentos servidores, o que potencializa a exploração de funcionalidades avançadas e o seu suporte continuado.

Ambiente de emulação CORE

CORE[11] é um framework *open-source* desenvolvido pela *Boeing Research and Development Division* (BR&T)[12] que visa a emulação de ambientes computacionais largamente distribuídos utilizando uma virtualização de pilha de rede FreeBSD. Os ambientes computacionais providos pelo CORE são emulados dinamicamente à medida que a execução se desenvolve, possibilitando uma conexão entre ambientes reais e emulados. Além disso, o CORE possibilita uma emulação distribuída onde múltiplos computadores podem colaborar para a emulação de um ambiente mais complexo.

O CORE fornece uma interface de usuário provida através da *Tool Command Language/Toolkit* (Tcl/Tk) que permite a customização e o desenvolvimento de ferramentas de interface para suas diferentes funcionalidades. Através desta interface é possível definir a configuração do ambiente computacional, posicionando os dispositivos e determinando suas interconexões. Também é possível emular diversas características operacionais de um ambiente computacional, como por exemplo: limite de largura de banda, latência, perda de pacotes e etc.

Através de virtualização o CORE possibilita que múltiplas instâncias virtuais sejam executadas simultaneamente, fazendo assim com que cada dispositivo emulado possua sua própria instância privada. Estas instâncias recebem o nome de *vimages* e, diferentemente das tradicionais máquinas virtuais, não possuem um sistemas operacional inteiro rodando no hardware emulado, ao invés disso elas executam o mesmo kernel e compartilham o sistema de arquivos, processador, memória, *clock* e outros recursos disponíveis. Assim os dispositivos emulados podem ser projetados em diversas linguagens de programação, como por exemplo, Python, bastando que estas linguagens estejam devidamente instaladas na instância inicial de emulação. A visão geral dos componentes do CORE está apresentado na Figura 5.12.

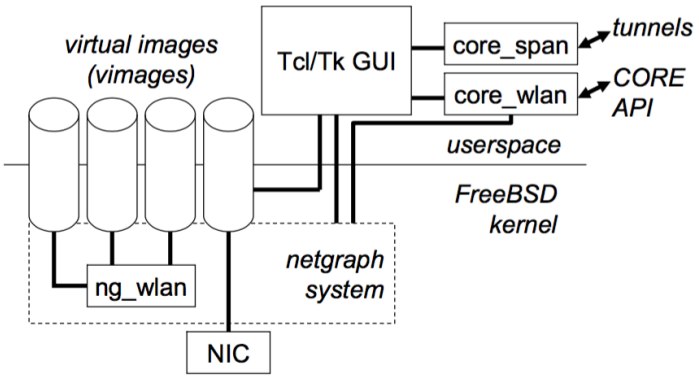

Figura 5.12 – Visão geral dos componentes do CORE (AHRENHOLZ et al., 2008)

[11] https://www.nrl.navy.mil/itd/ncs/products/core
[12] http://www.boeing.com.au/products-services/research-technology.page

5.3.3 Aspectos considerados no estudo de caso

A partir das informações referentes aos atributos físicos e químicos do solo, determinados pela análise de amostras coletadas ao longo de toda a área cultivada, são determinadas as zonas de manejo. Zona de manejo consiste em uma porção da área cultivada em que o solo possui característica hídrica aproximadamente homogênea e por isso permite que seja adotada a mesma estratégia de irrigação. Isso significa dizer que o momento de irrigar e a quantidade de água utilizada é a mesma ao longo de toda zona de manejo, assim não se teria falta nem excesso de água em nenhum ponto. A quantidade de água total a ser distribuída uniformemente na zona de manejo é determinada pelo especialista responsável tendo em vista a capacidade de campo e o valor da tensão de água no solo no momento em que foi iniciada a irrigação.

Para determinar o momento correto de irrigar é necessária a avaliação de diversas variáveis físicas. Nesse estudo de caso, as variáveis contextuais consideradas são as seguintes: (i) tensão de água no solo e (ii) evapotranspiração. A tensão de água do solo pode ser obtida através de Tensiômetros acoplados à CoIot Gateways (nativos ou proprietários). Evapotranspiração é a perda de água do solo por um processo de evaporação e pode ser estimada através de método matemático que utiliza informações como: temperatura do ar, velocidade do vento, umidade relativa do ar e o saldo de radiação (ANTÔNIO; CONCEIÇÃO, 2006). Essas informações podem ser obtidas através de uma estação meteorológica.

Os comandos de atuação disparados através do processamento de regras podem ser das seguintes naturezas: (i) alertas visuais, (ii) alertas sonoros, (iii) envio de mensagem (SMS e/ou e-mail) e (iv) acionamento de transdutores elétricos para acionamento do sistema de irrigação. O envio de mensagens deve ser processado pelo Servidor de Contexto por necessitar de acesso aos perfis dos usuários com suas preferências e informações de contato, enquanto os demais alertas podem ser processados em nível de Servidores de Borda quando necessário.

A Figura 5.13 representa um parreiral típico da região sul do Brasil com suas respectivas zonas de manejo. Para reduzir os eventuais erros de medição, decorrentes das diferenças do solo ou avarias em sensores, foi considerado o uso de gateways controlando diversos sensores distribuídos ao longo das zonas de manejo. Também, em cada zona de manejo, foi utilizado um Servidor de Borda e eventualmente um repetidor do sinal wi-fi quando necessário.

Atualmente, grande parte das áreas rurais depende de conexões móveis através de rede celular para conexão à Internet, muitas vezes estando sujeitas a conexões instáveis, baixa largura de banda e volume de dados mensais limitados através de franquias. Esses aspectos

apontam para a necessidade de uso moderado do tráfego de dados através da Internet e o uso de estratégias que garantam a operação em momentos em que esse acesso não é possível.

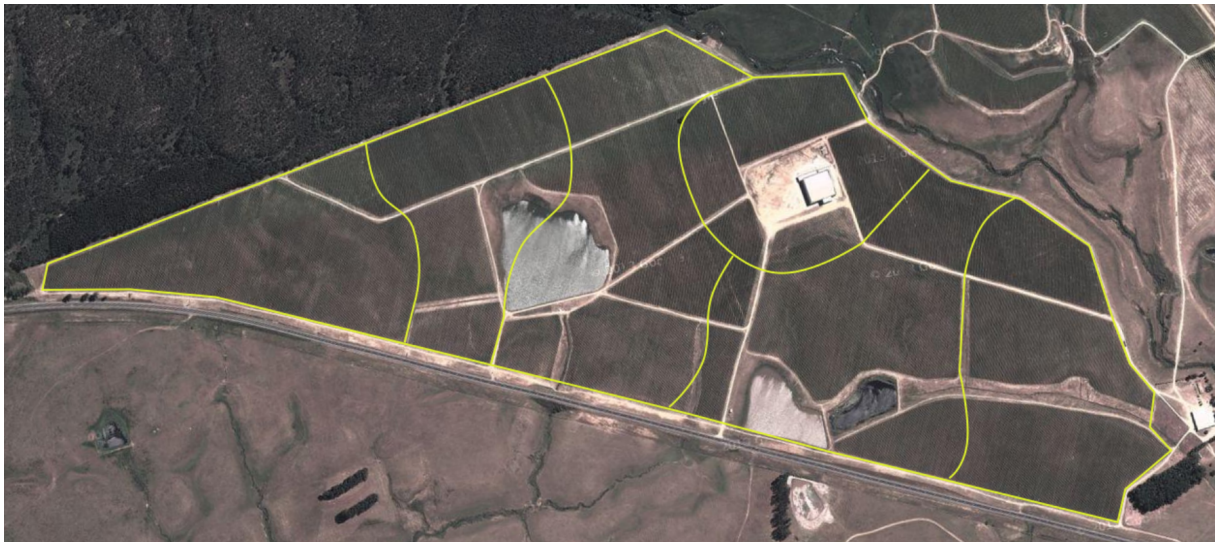

Figura 5.13 – Zonas de manejo em um parreiral

A infraestrutura proposta considera a utilização de vários Servidores de Borda distribuídos pelo parreiral, com isso pretende-se evitar que o controle de irrigação, que se dá em nível de zona de manejo, seja inviabilizado por eventuais quedas de conexão de rede e a consequente perda de contato com o Servidor de Contexto. Uma atividade coordenada entre os Servidores de Borda vizinhos também é prevista para determinar o momento de irrigar. O estudo de caso considera que o Servidor de Contexto fique localizado distante da lavoura, na sede da fazenda, no escritório, ou mesmo em algum servidor localizado na nuvem, por exemplo. Com este estudo de caso é avaliada a arquitetura do CoIoT no que diz respeito a sua operação distribuída.

Configuração e Lógica de Controle

O estudo de caso foi organizado a partir da distribuição dos dispositivos ao longo do parreiral considerando as respectivas zonas de manejo. A distribuição destes dispositivos foi feita da seguinte forma:

- **Servidor de Contexto:** é utilizado um Servidor de Contexto para todo o parreiral, sendo localizado distante da lavoura e acessado via Internet;

- **Servidor de Borda:** cada zona de manejo é gerenciada por um Servidor de Borda sendo responsável pela centralização dos dados contextuais coletados e pela gerência autônoma de irrigações através de regras customizadas;
- **Gateway:** cada Servidor de Borda gerencia 6 gateways dentro da sua Zona. Os gateways comunicam-se por via WIFI com o Servidor de Borda e fisicamente com os sensores, por isso, necessitam de uma distribuição maior ao longo da zona de manejo;
- **sensoreamento:** cada Gateway coordena a operação de 5 sensores de tensão de água do solo (tensiômetros).

Com a organização acima tem-se no estudo de caso 180 sensores, 36 gateways, 6 Servidores de Borda e 1 Servidor de Contexto sendo emulados através do CORE. A distribuição dos gateways e Servidores de Borda ao longo do parreiral é feita conforme apresentado na Figura 5.14. Essa figura resume o esforço de concepção no ambiente provido pelo CORE para organizar de forma distribuída os Servidores de Borda (em vermelho) e os gateways (em azul) gerenciados pelos mesmos, bem como as interconexões da infraestrutura de rede.

Figura 5.14 – Zonas de manejo mapeadas no ambiente de emulação CORE

A tensão de água no solo de cada zona de manejo é monitorada e comparada com valor da tensão crítica da cultura que corresponde ao valor de tensão em que se deve proceder a irrigação. No caso de uvas viníferas, o valor de tensão crítica está entre 15kPa a 50kPa e isso

determina o momento correto de realizar a irrigação do parreiral. O valor de 15kPa, considerado nesse estudo de caso como tensão crítica baixa, deve ser avaliado em situações de elevada evapotranspiração e, quando essa situação ocorrer a irrigação deve ser iniciada. Já o valor de 50kPa, considerado nesse estudo de caso como tensão crítica alta, quando atingido em uma zona de manejo, a irrigação deve iniciar imediatamente.

O gerenciamento da irrigação do parreiral considera um conjunto de regras organizadas entre os Servidores de Borda e de Contexto. A lógica a ser tratada através dessas regras está descrita a seguir:

- **SE** o valor da tensão de água do solo estiver acima da tensão crítica alta (50kPa) **ENTÃO** a irrigação é imediatamente acionada e alertas são gerados (luminoso, sonoro, mensagens SMS/e-mail);

- **SE** o valor de tensão medido estiver acima da tensão crítica baixa (15kPa) e **SE** a evapotranspiração estiver alta **ENTÃO** a irrigação é imediatamente acionada e alertas são gerados (luminoso, sonoro, mensagens SMS/e-mail).

- **SE** o valor de tensão medido estiver acima da tensão crítica baixa (15kPa) e **SE** alguma zona de manejo vizinha já tenha identificado valor crítico de umidade do solo **ENTÃO** a irrigação é imediatamente acionada e alertas são gerados (luminoso, sonoro, mensagens SMS/e-mail).

Partindo da lógica de controle foram criadas regras para o ambiente emulado no estudo de caso. Essas regras, gerenciadas em nível de Servidor de Borda, são disparadas tanto por eventos locais quanto por eventos gerados a partir de Servidores de Borda externos, caracterizando uma operação em Fog. O processamento dessas regras é feito pelo módulo Motor de Regras do CoIoT. As regras de processamento de contexto são apresentadas na Tabela 5.9 e estão especificadas em função do Servidor de Borda *BordaZN1* que é responsável por gerenciar a zona de manejo 1. Os demais Servidores de Borda possuem regras similares.

Quando a condição de uma regra é verdadeira ela gera um evento composto que pode ser usado para disparar outras regras, de maneira similar aos eventos primitivos. O nome da regra também dá nome ao evento composto produzido e pode ser acessado remotamente através de um processo de subscrição. O gerenciamento desses eventos de maneira distribuída, bem como o direcionamento dos mesmos ao Motor de Regras é realizado na arquitetura do CoIoT pelo módulo Gerenciador de Eventos. As regras disparadas por eventos remotos têm o nome do

Tabela 5.9 – Regras do Servidor de Borda BordaZN1

Nome da Regra	**Evento**	**Condição**	**Ação**
tensãoCríticaAlta	tensãoMédiaCalcZN1	tensãoMédia>50k	Irriga e notifica
tensãoCríticaBaixa	tensãoMédiaCalcZN1	tensãoMédia>15k e evapoTranspiração==alta	Irriga e notifica
tensãoCríticaBaixa2	tensãoCríticaAlta**.**BordaZN2	tensãoMédia>15k	Irriga e notifica
tensãoCríticaBaixa3	tensãoCríticaAlta**.**BordaZN3	tensãoMédia>15k	Irriga e notifica
tensãoCríticaBaixa4	tensãoCríticaAlta**.**BordaZN4	tensãoMédia>15k	Irriga e notifica
tensãoCríticaBaixa5	tensãoCríticaAlta**.**BordaZN5	tensãoMédia>15k	Irriga e notifica
tensãoCríticaBaixa6	tensãoCríticaAlta**.**BordaZN6	tensãoMédia>15k	Irriga e notifica

Servidor de Borda remoto anexado ao nome do evento de origem e separado por um ponto (“**.**”), como por exemplo *tensãoCríticaAlta**.**BordaZN1* (vide Tabela 5.9).

O estudo de caso pressupõe que cada Servidor de Borda deve realizar subscrições nos demais Servidores de Borda no evento composto *tensãoCríticaAlta*. Assim, os Servidores de Borda de todas as zonas de manejo serão notificados quando a tensão de água do solo em qualquer ponto do parreiral atingir estado criticamente alto (50kPa). A tabela 5.10 indica as subscrições nos eventos do Servidor de Borda *BordaZN1* o qual gerencia a zona de manejo 1.

Tabela 5.10 – Eventos e subscrições do Servidor de Borda BordaZN1

Evento	**Servidor de Borda subscrito**
tensãoCríticaAlta	BordaZN2
tensãoCríticaAlta	BordaZN3
tensãoCríticaAlta	BordaZN4
tensãoCríticaAlta	BordaZN5
tensãoCríticaAlta	BordaZN6

As informações contextuais utilizadas na emulação têm por base dados reais de tensão de água do solo obtidos em parceria com a EMBRAPA Clima temperado. O conjunto de regras e eventos distribuídos utilizados no estudo de caso processam esses dados a fim de identificar as situações que caracterizam o momento correto em que a irrigação deve ser iniciada. A Figura 5.15 apresenta os dados utilizados no estudo de caso destacando o momento de atuação das regras que identificam uma situação em que a irrigação é necessária.

No estudo de caso, a regra *tensãoCríticaBaixa* do Servidor de Borda 1, indicada na Tabela 5.9, não chega a atuar, pois embora o valor de tensão de água do solo da zona de manejo 1 (ZM 1) tenha ultrapassado o valor de 15kPa, a evapotranspiração foi definida como baixa durante todo o estudo de caso. Situação semelhante ocorre com a zona de manejo 2 (ZM

2). Já nas demais zonas de manejo, a tensão de água do solo não atinge o valor de 15kPa em nenhum momento.

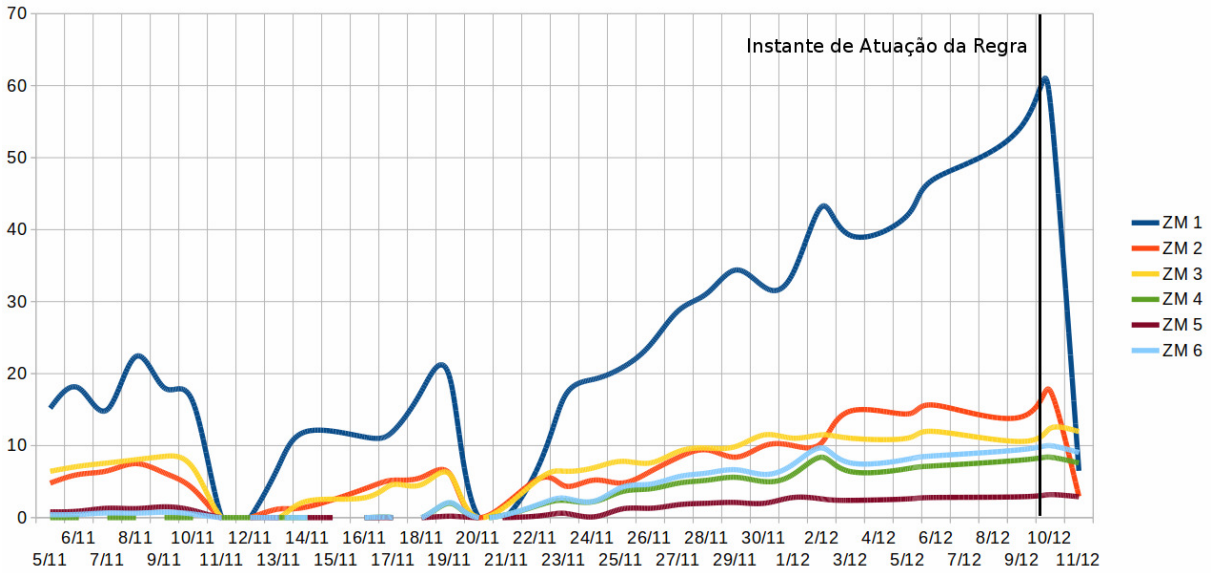

Figura 5.15 – Tensão de Solo: Instante de atuação da regra

Uma situação diferente acontece com a regra *tensãoCríticaAlta* do Servidor de Borda 1. Essa regra atua quando a tensão de água do solo da zona de manejo 1 ultrapassa o valor de 50kPa, pois na identificação dessa situação a irrigação deve começar imediatamente. Nesse momento a regra produz um evento composto e gera eventos remotos nos Servidores de Borda Subscritos. Portanto o Servidor de Borda BordaZN2 receberá um evento *tensãoCríticaAlta.BordaZN1* com origem no Servidor de Borda *BordaZN1*, similar ao que é mostrado na Tabela 5.9. Esse evento dispara uma regra no Servidor de Borda BordaZN2 a qual irá atuar, pois a tensão de água do solo da zona de manejo 2 está acima de 15kPa (vide Figura 5.15). Portanto tanto a zona de manejo 1 quanto a zona de manejo 2 iniciam seus procedimentos de irrigação quase que simultaneamente. O momento em que isso ocorre é indicado na Figura 5.15, em que pode ser observada a consequente redução da tensão de água do solo em ambas zonas de manejo.

Para visualizar os dados contextuais coletados durante o estudo de caso foi desenvolvida uma aplicação web que acessa o Repositório de Contexto do CoIoT através da API REST. A Figura 5.16 mostra o modo de visualização em que os dados são apresentados no formato de um relatório textual e a Figura 5.17, por sua vez apresenta o modo de visualização gráfico.

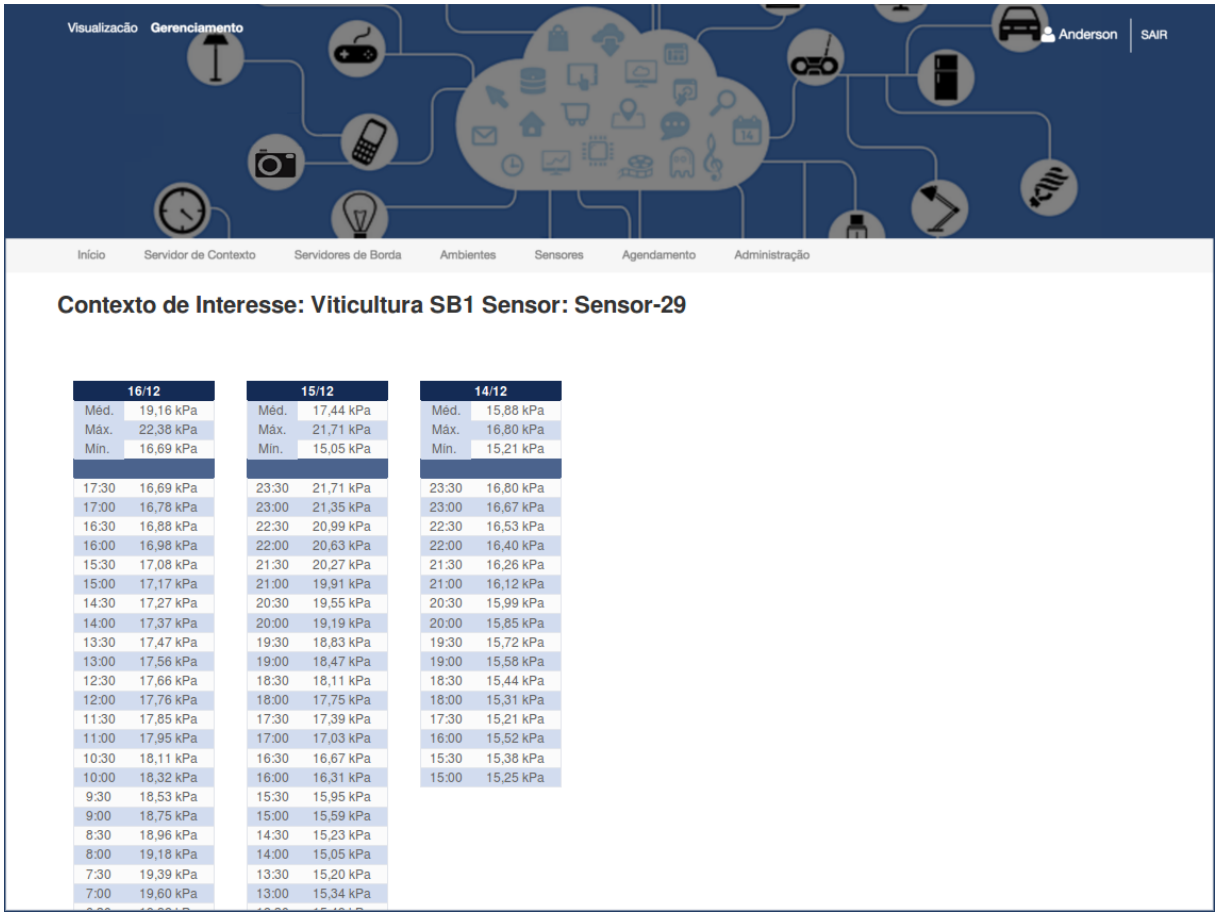

Figura 5.16 – Aplicação web para visualização de dados contextuais

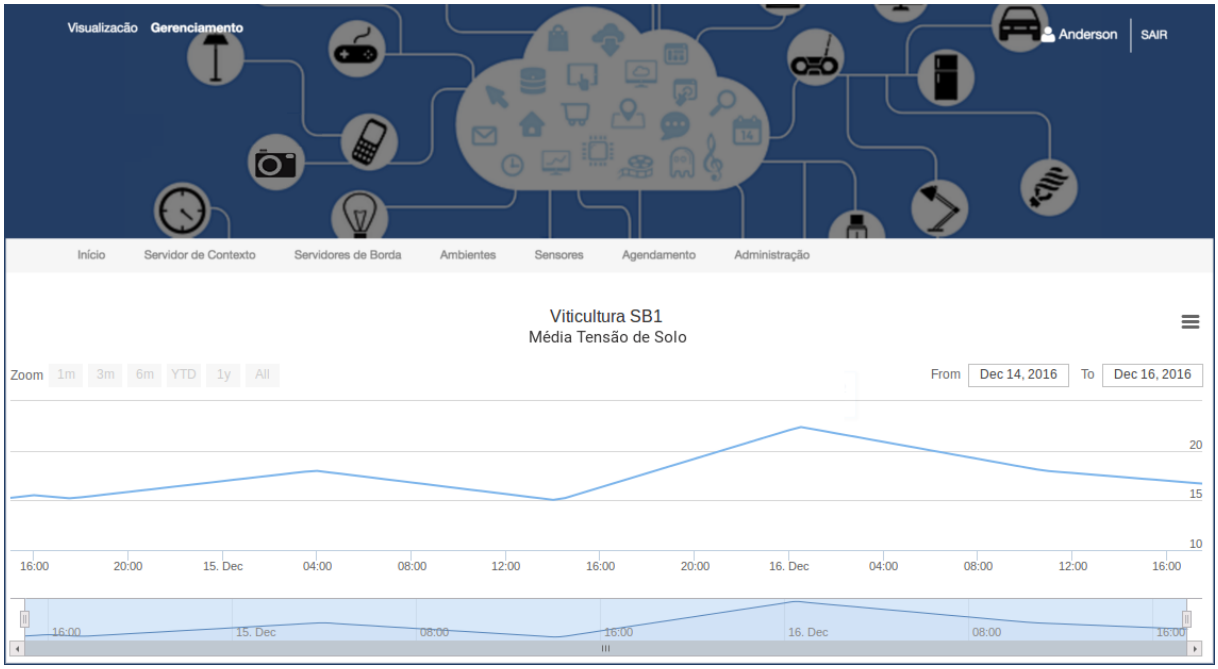

Figura 5.17 – Aplicação web para visualização de dados contextuais

5.3.4 Avaliações realizadas

Este estudo de caso proporcionou avaliar as funcionalidades do CoIoT quanto a sua capacidade de realizar processamento de contexto colaborativo entre Servidores de Borda, aspecto característico de ambientes de computação em Fog.

Os testes realizados também proporcionaram avaliar quantitativamente o modo operacional com coleta e processamento distribuído provido pelo CoIoT em relação ao modelo operacional com coleta distribuída e processamento centralizado contemplado pelos trabalhos relacionados. Assim, a avaliação foi realizada comparando as configurações dos seguintes modos operacionais do CoIoT:

- *Operação com coleta e processamento distribuídos*:
  Nesse modo operacional tanto o cálculo da média dos valores de tensão de água do solo quanto o respectivo processamento de contexto através das regras ECA que avaliam esses valores são realizados nos Servidores de Borda. A publicação dos dados no Servidor de Contexto é feita periodicamente para fins históricos, os quais correspondem ao valor médio da tensão de água no solo de cada zona de manejo e o respectivo desvio padrão;

- *Operação com coleta distribuída e processamento centralizado*:
  Nesse modo operacional todas as medições de tensão de água do solo são publicadas no Servidor de Contexto o qual é responsável por fazer o cálculo da média dos valores de tensão de água do solo das zonas de manejo, bem como realizar o respectivo processamento de contexto através das regras ECA que avaliam esses valores. Nesse modo operacional todo processamento de contexto é feito no Servidor de Contexto.

Cada um dos testes realizados exigiu a composição de um diferente conjunto de regras e eventos além de diferentes estratégias de coleta das informações contextuais do parreiral. As seções a seguir discutem as especificações de cada um destes testes e os resultados obtidos.

Redução do volume de dados transmitido

Em ambientes rurais, como no cenário de viticultura analisado nesse estudo de caso, muitas vezes a única possibilidade de acesso à Internet se dá através de redes de telefonia móvel com baixa qualidade de sinal e com planos com franquia de dados limitada. Nesses casos é conveniente promover a redução do volume de dados enviados através da Internet e

estabelecer estratégias de processamento de contexto que não dependam exclusivamente desse meio de comunicação. Nesse estudo de caso, todas as comunicações realizadas com o Servidor de Contexto se dão através da Internet.

Nessa situação de avaliação, o Agendador foi configurado para que cada sensor localizado no parreiral realizasse uma medição de tensão de água do solo a cada 5 minutos. Para garantir a regularidade das leituras, foram desativados todos os *triggers* que disparam medições de acordo com o valor de tensão de água do solo.

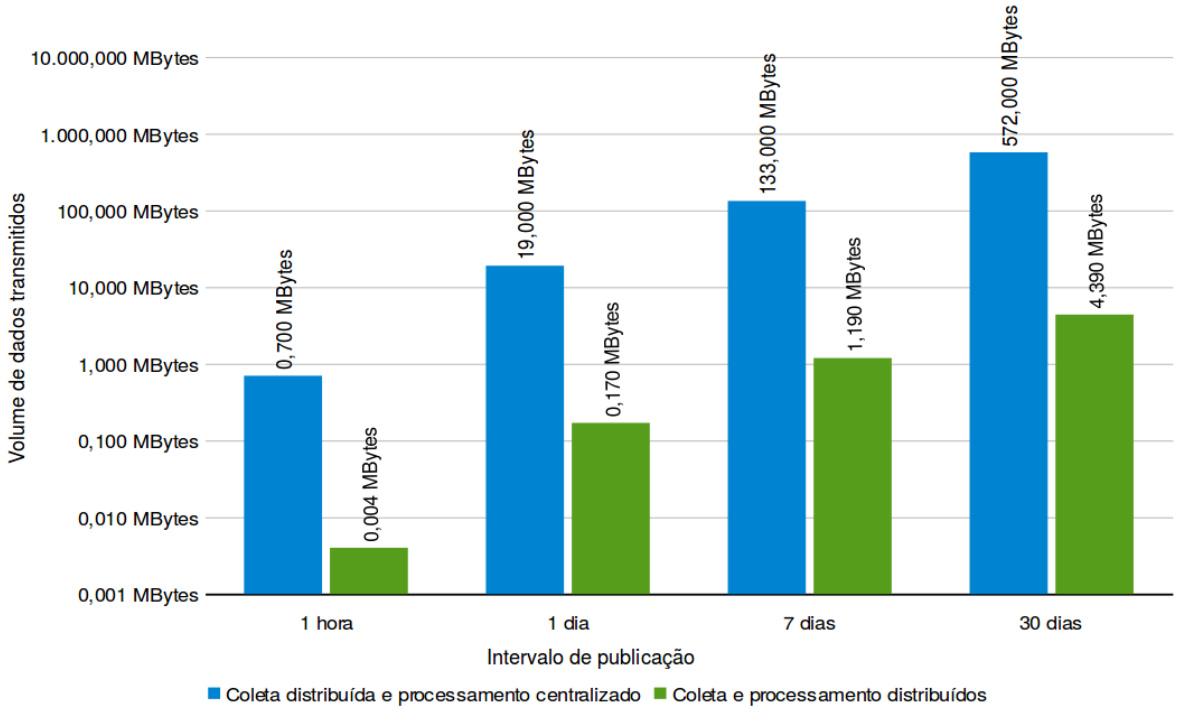

Figura 5.18 – Gráfico do volume de dados transferido ao Servidor de Contexto

Na Figura 5.18 são apresentados os resultados obtidos nesta avaliação utilizando uma escala logarítmica, os quais foram agrupados em quatro intervalos de tempo: hora, dia, semana e mês. Nesta figura é possível comparar o volume de dados publicados no Servidor de Contexto nos dois modos operacionais considerados.

O modo operacional proporcionado através do uso do CoIoT, em que tanto a coleta como o processamento do contexto é feito nas bordas do ambiente computacional como em um típico cenário de Fog, produz um reduzido volume de dados envido ao Servidor de Contexto através de Internet móvel como no ambiente do estudo de caso.

O uso dos recursos de comunicação com o emprego de processamento distribuído nos Servidores de Borda aumenta proporcionalmente ao período observado, mas ainda mantém-se

em patamares baixos quando comparado com os valores obtidos em um cenário operacional com processamento centralizado. Como as comunicações entre os Servidores de Borda e os controladores de atuação (sistema de irrigação) são administradas pelos próprios Servidores de Borda, praticamente não há necessidade de intervenção do Servidor de Contexto. Isso evidencia as vantagens em se promover a autonomia do ambiente através de uma estratégia de gerenciamento a partir de parâmetros pré-estabelecidos.

Essa característica também promove a escalabilidade em nível de borda, possibilitando que mais sensores possam ser conectados aos Servidores de Borda potencializando a complexidade das atividades que podem ser gerenciadas pelos mesmos.

Independência do volume de dados transmitido em relação ao intervalo de coleta

Na viticultura de precisão muitas vezes é necessário aumentar a frequência de obtenção das grandezas físicas do ambiente, ou seja, reduzir o intervalo de leitura dos sensores, e por consequência das potenciais atuações síncronas sob os parreirais, seja porque há uma necessidade de manter um rigoroso controle sobre o ambiente, ou porque se deseja manter um registro das possíveis variações das informações contextuais entendidas como relevantes.

Porém intervalos pequenos para aquisição de informações contextuais podem constituir um problema operacional para arquiteturas que trabalham com um processo de decisão centralizada, uma vez que todos dados contextuais devem ser transmitidos para estes locais. Esse problema é potencializado quando a transmissão deve ocorrer via Internet os quais podem sobrecarregar a rede existente ou limitar a banda disponível para outras transações de interesse do usuário, causando assim um dilúvio de dados ou congestionamento do canal de transmissão.

A Figura 5.19 compara os resultados obtidos ao se usar uma estratégia operacional com coleta e processamento de contexto nas bordas em relação à estratégia operacional com decisão centralizada, em situações em que existe a necessidade de uma verificação mais frequente das variáveis contextuais. Neste teste foi contabilizada a totalidade de dados enviados para o Servidor de Contexto considerando um período de 30 dias.

No ambiente operacional de teste com processamento nas bordas são realizadas publicações no Servidor de Contexto a cada hora para fins de registro histórico, enquanto no ambiente com processamento centralizado as publicações são realizadas a cada intervalo de coleta dos dados.

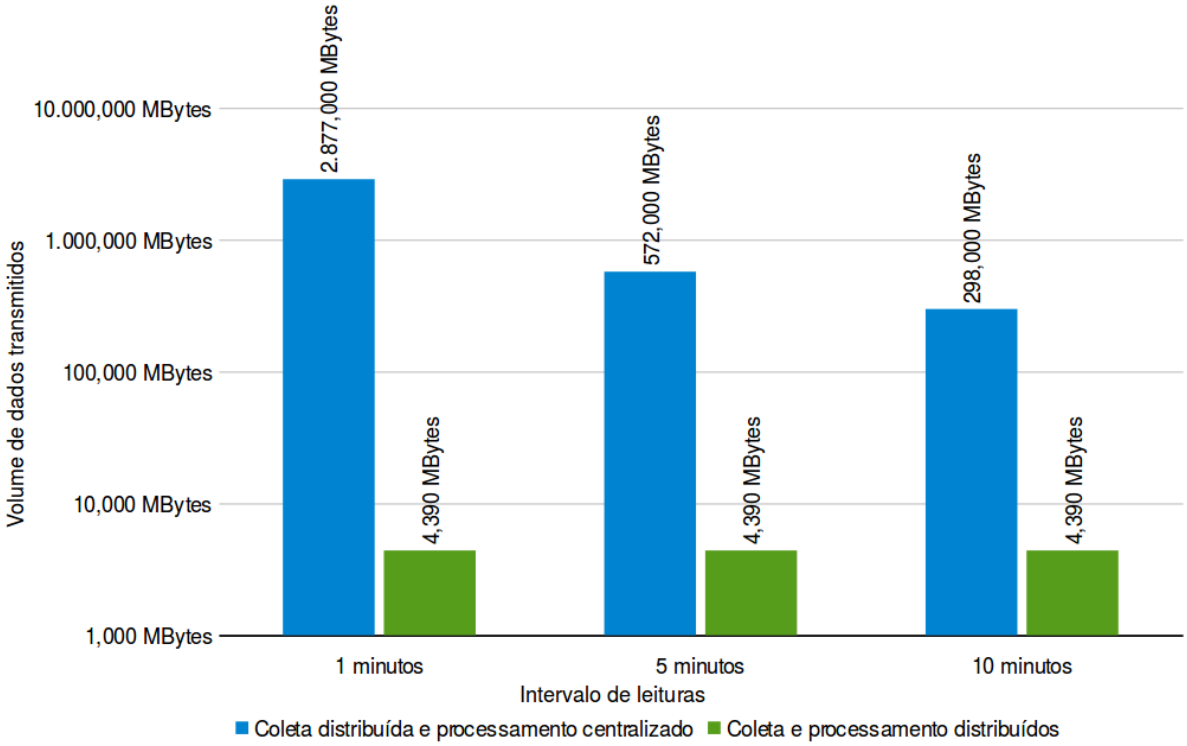

Figura 5.19 – Gráfico em função do intervalo de aquisição

É possível observar que o volume de dados enviado ao Servidor de Contexto é reduzido significavelmente ao se explorar a operação com processamento nas bordas proporcionado pelo CoIoT quando comparado às estratégias operacionais em que o processamento do contexto é centralizado. Além disso, mesmo aumentando a frequência de leitura dos sensores, o volume de dados enviados para o Servidor de Contexto continua constante, pois os dados enviados servem para manter um registro histórico.

O CoIoT permite que o tempo de aquisição seja menor mantendo a dinamicidade do ambiente e das regras, porém permitindo reduzir a latência das decisões originadas por aquisições síncronas, onde a requisição de leitura é feita através de um tempo pré-determinado.

Escalabilidade dos sensores empregados

Em ambientes da agricultura é comum haver mudança da área cultivada por consequência de ampliações decorrentes da necessidade de aumento de produção. Isso requer um proporcional aumento da quantidade de sensores a serem utilizados.

De forma a avaliar a escalabilidade nas bordas computacionais foram realizados testes variando a quantidade de sensores por gateway, e por consequência, por Servidor de Borda.

A Figura 5.20 apresenta o resultado do volume de dados transmitido ao Servidor de Contexto de acordo com a variação da quantidade de sensores por gateway. A Figura 5.20 considera os dois modos operacionais avaliados.

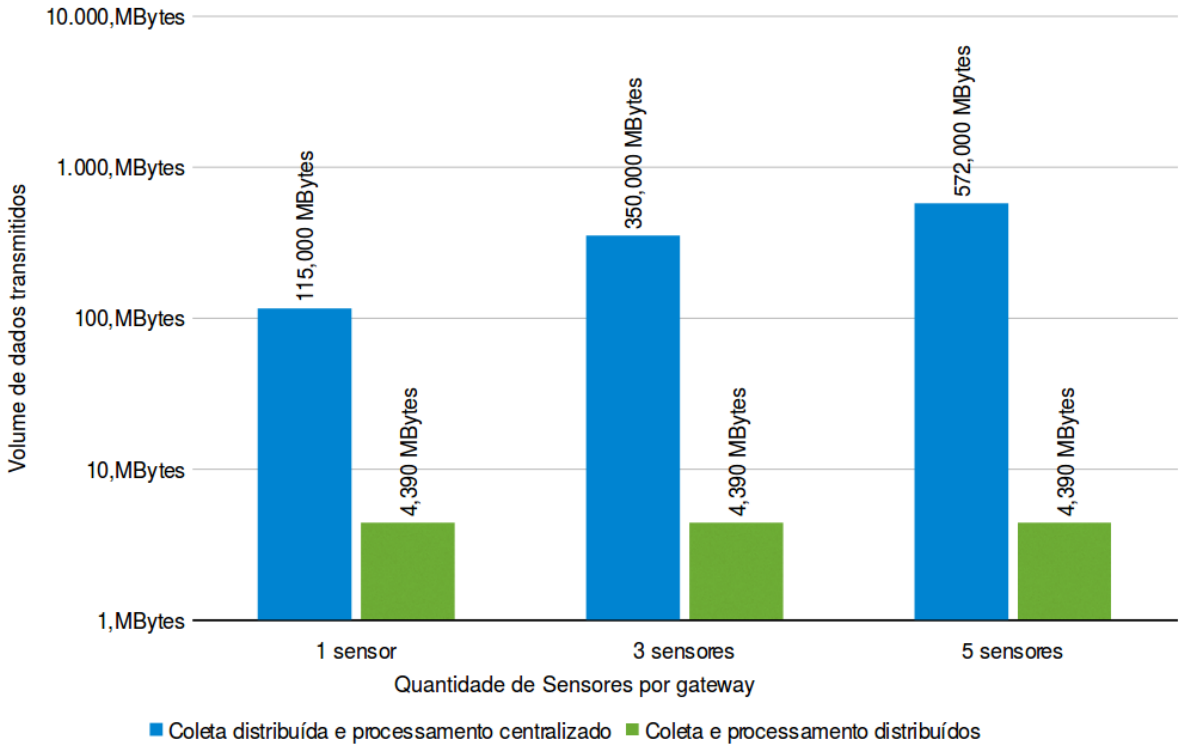

Figura 5.20 – Gráfico em função da quantidade de sensores no ambiente

Pela análise dos resultados observa-se que no caso operacional em que o processamento de contexto se dá nas bordas, a quantidade de sensores utilizados não provoca mudanças no volume de dados enviados para o Servidor de Contexto. Isto é consequência da agregação de dados promovida pelo CoIoT ao realizar o processamento de contexto antes da transmissão. Assim, só são propagados para o servidor de Contexto as médias dos valores e o desvio padrão, o que reduz significativamente os dados transmitidos. Evidentemente, com a utilização de processamento centralizado no Servidor de Contexto essa agregação é possível, aumentando proporcionalmente o volume de dados transmitidos conforme o aumento do número de sensores aumenta.

Portanto, a estratégia de coleta e processamento distribuído do contexto adotada no CoIoT permite expandir o número Servidores de Borda e de dispositivos de sensoreamento e atuação conforme as demandas de monitoramento do ambiente, sem onerar os recursos computacionais disponíveis. Isso significa que um eventual aumento da área do parreiral que

acarretasse em novas zonas de manejo, necessitaria um aumento do número de dispositivos utilizados, mas não provocaria um aumento significativo na quantidade de informações transmitidas.

5.4 Considerações sobre o capítulo

Este capítulo apresentou a avaliação das funcionalidades do middleware COIOT através de dois estudos de caso na área da agricultura. Neste capítulo também foram discutidas as tecnologias de software e hardware empregadas na prototipação da arquitetura do middleware.

O primeiro estudo de caso foi realizado no Projeto plenUS tendo em vista as demandas de pesquisadores do Laboratório de Análise de Sementes Oficial (LASO) da Embrapa Clima Temperado. Nesse estudo de caso foi empregado um Servidor de Contexto, um Servidor de Borda e um conjunto de sensores e atuadores distribuídos pelo ambiente do laboratório com o intuito de acompanhar o comportamento dos equipamentos de análise.

O segundo estudo de caso foi organizado de acordo com as demandas de irrigação de um ambiente da viticultura de precisão. Foi organizada uma infraestrutura de IoT virtual que envolveu Servidores de Borda, gateways e sensores distribuídos pelo parreiral e um Servidor de Contexto acessado remotamente através da Internet. Nesse estudo de caso priorizou-se o processamento nos Servidores de Borda em um típico ambiente de computação em Fog.

O próximo e último capítulo contempla as principais conclusões deste trabalho, bem como as contribuições da pesquisa, publicações realizadas e trabalhos futuros.

6 CONSIDERAÇÕES FINAIS

Este capítulo apresenta as considerações finais desta tese, revisitando o problema de pesquisa considerado no desenvolvimento deste trabalho, bem como destacando as principais contribuições e as oportunidades de trabalhos futuros, decorrentes dos esforços de pesquisa contemplados no desenvolvimento do CoIoT.

6.1 Principais conclusões

Nesta tese foi concebido um middleware para Internet das Coisas denominado CoIoT, o qual teve como motivação principal prover suporte à Ciência de Contexto para as aplicações ubíquas providas pela IoT. A arquitetura desenvolvida para o CoIoT dispõe de mecanismos para o tratamento distribuído de eventos, da coleta e processamento das informações contextuais do ambiente físico, bem como da atuação sobre este ambiente.

A pesquisa realizada contemplou um estudo da literatura da área explorando os principais conceitos e desafios relacionados ao tema da tese, como: Ciência de Contexto na perspectiva da UbiComp, Internet das Coisas, computação em Fog, tratamento de eventos e o middleware EXEHDA. Este estudo também contemplou a sistematização dos principais projetos relacionados à proposta concebida, com intuito de identificar os problemas de pesquisa em aberto e as características mais importantes que pudessem contribuir com a definição da arquitetura do CoIoT.

A concepção do CoIoT teve como princípio de modelagem a definição de uma abordagem distribuída de processamento de contexto que contempla tanto as premissas da IoT quanto as demandas das aplicações da UbiComp. A arquitetura concebida para o CoIoT foi detalhada, considerando suas funcionalidades organizadas entre os Servidores de Contexto e de Borda.

O modelo de tratamento do contexto do CoIoT contempla tanto processamento vertical quanto horizontal. O processamento vertical é caracterizado pela distribuição do tratamento de contexto entre os Servidores de Borda e Contexto. Já o processamento horizontal contempla o tratamento das informações contextuais de maneira distribuída entre os Servidores de Borda, caracterizando operações típicas da computação em Fog. Para prover suporte ao modelo de processamento de contexto adotado, foram definidas estratégias para tratamento distribuído de eventos entre os dispositivos de borda.

Também, para apoiar o modelo de tratamento de contexto do CoIoT foram concebidas

estratégias para gerenciar a interoperabilidade e heterogeneidade. Como forma de tratar a interoperabilidade optou-se pela adoção de um modelo baseado na arquitetura REST definida por (FIELDING, 2000), sendo concebida uma API REST para prover acesso aos recursos dos dispositivos de coleta e atuação. O tratamento da heterogeneidade contemplado pela arquitetura tem por base o uso de gateways para gerenciar os dispositivos de coleta e atuação, os quais encapsulam as tecnologias heterogêneas utilizadas por tais dispositivos. A coleta dos dados contextuais adotada no COIOT pode ser direta, quando os sensores são lidos diretamente por uma aplicação através da API REST disponibilizada, ou a coleta pode ser disparada pelos mecanismos artonômicos do middleware, através de agendamentos ou do uso de *triggers*.

Considerando as premissas de concepção do COIOT, foram selecionados trabalhos relacionados cujas características principais contemplam o tratamento de sensores e atuadores, o tratamento da heterogeneidade, o suporte ao processamento de eventos, o suporte à Ciência de Contexto e o tratamento da interoperabilidade. O principal diferencial do COIOT em relação a estes trabalhos relacionados está no processamento distribuído do contexto que pode ocorrer tanto verticalmente entre Servidores de Borda e Contexto quanto horizontalmente nos dispositivos de borda. O processamento nas bordas permite reduzir o volume de dados enviados através da rede, bem como a dependência de infraestruturas de comunicação controladas por terceiros no gerenciamento de situações críticas. Além disso, o processamento de contexto no COIOT é gerenciado através de regras ECA (*Event-Condition-Action*) parametrizáveis disparadas por eventos. Isso propicia tratar as mudanças de estado das variáveis contextuais no momento de sua ocorrência. A estratégia baseada em eventos adotada na concepção da arquitetura contempla tanto o processamento do contexto quanto a aquisição das informações contextuais do ambiente.

Para avaliar o middleware COIOT foi desenvolvido um protótipo e foram concebidos dois estudos de caso na área da agricultura. O primeiro estudo de caso foi desenvolvido em ambiente de produção no Projeto plenUS tendo em vista as demandas de pesquisadores da área de sementes. Este projeto foi concebido para oferecer soluções de Computação Ubíqua para o Laboratório de Análise de Sementes Oficial (LASO) da Embrapa Clima Temperado. No cenário avaliado foi empregado um conjunto de sensores e atuadores distribuídos pelo ambiente com o intuito de acompanhar o comportamento dos equipamentos de análise.

O segundo estudo de caso concebido considerou um ambiente da viticultura de precisão, mais especificamente estratégias de controle de irrigação. A infraestrutura de IoT concebida para este estudo de caso contempla Servidores de Borda, gateways e sensores distribuídos pelo parreiral e um Servidor de Contexto acessado através da Internet. Isso propiciou a exploração de

estratégias de processamento prioritariamente nas bordas em um típico ambiente de computação em Fog.

Os estudos de caso desenvolvidos proporcionaram explorar aspectos importantes da arquitetura proposta, dos quais destacam-se: (i) coleta distribuída dos dados contextuais; (ii) atuação proativa; (iii) processamento distribuído do contexto através de regras ECA; (iv) tratamento de eventos distribuído; (v) notificação ciente do contexto, e; (vi) persistência dos dados sensoreados no repositório de contexto.

6.2 Contribuições da tese

Os esforços de estudo e pesquisa relacionados a esta tese aconteceram na intersecção das áreas de ciência do contexto e Internet das Coisas. A seguir estão resumidas as principais contribuições decorrentes da concepção do middleware CoIoT:

- Concepção de uma arquitetura para IoT direcionada à Ciência de Contexto, com potencial de emprego em diferentes domínios de aplicação;
- Proposição de uma abordagem distribuída para Ciência do Contexto, que trata a coleta e processamento das informações contextuais, bem como promove a atuação sobre o ambiente físico a fim de atender às aplicações ubíquas da IoT;
- Proposição de um modelo de processamento distribuído de eventos voltado às aplicações da IoT que promovam a colaboração entre dispositivos de borda;
- Integração dos mecanismos para tratamento da heterogeneidade e da interoperabilidade para suporte às abordagens para Ciência de Contexto e tratamento de eventos propostas na tese;
- Avaliação das funcionalidades já operacionais na arquitetura proposta, considerando cenários de uso que tem por base o atendimento das demandas tanto de pesquisadores da área de agricultura, como aquelas do gerenciamento da produção agrícola.

6.3 Trabalhos futuros

O middleware CoIoT apresenta diversas oportunidades de pesquisa a serem exploradas. Neste sentido, destacam-se os seguintes temas que podem ser considerados em trabalhos

futuros:

- Capacitar os mecanismos do middleware para tratar os elevados volumes de informações contextuais provenientes das modernas infraestruturas providas pela IoT, provendo suporte para Big Data Analytics ao CoIoT;
- Prover suporte de tempo real aos Servidores de Borda do CoIoT, garantindo para as diferentes aplicações o tratamento de situações criticas através de eventos e regras locais;
- Dotar a arquitetura do CoIoT de suporte para a incorporação automatizada de recursos de sensoriamento e atuação disponíveis nos diferentes Servidores de Borda existentes a nível celular, contribuindo com os mecanismos destinados ao tratamento de Fog Computing.

6.4 Publicações realizadas

Em decorrência dos trabalhos de estudo e pesquisa realizados durante o desenvolvimento desta tese foram publicados artigos em periódicos e anais de congressos, dos quais os mais significativos são listados a seguir.

- SOUZA, Rodrigo. S.; LOPES, João Ladislau B.; CARDOZO, Anderson A.; CARVALHO, Tainã Ribeiro; DAVET, Patricia T.; WOLF, Alexandre; Barbosa, Jorge L.V.; YAMIN, ADENAUER CORREA. Uma Arquitetura para IoT Direcionada à Ciência do Contexto Baseada em Eventos Distribuídos. In: Simpósio Brasileiro de Computação Ubíqua e Pervasiva (SBCUP), Porto Alegre. 2016. (menção honrosa)
- LOPES, JOAO; SOUZA, RODRIGO; GEYER, CLAUDIO; SOUZA, ALEXANDRE; DAVET, PATRICIA; PERNAS, ANA; Yamin, Adenauer. A situation-aware pervasive approach for assessing therapeutic goals in healthcare environment. In: Annual ACM Symposium on Applied Computing (SAC), Pisa. 2016.
- DAVET, Patricia T.; KAISER FILHO, Huberto; JOAO, Leonardo da Rosa S.; XAVIER, Lucas Mendonça de Souza; SOUZA, Rodrigo. S.; LOPES, João Ladislau B.; FLEISCHMANN, ANA MARILZA PERNAS; WARKEN, Nelsi; YAMIN, Adenauer Corrêa. Ciência de Contexto na IoT: Uma Contribuição à Pesquisa Agropecuária. In: Seminário Integrado de Software e Hardware (SEMISH), Porto Alegre. 2016.
- CARDOZO, ANDERSON; Yamin, Adenauer; XAVIER, LUCAS; SOUZA, RODRIGO; LOPES, JOAO; GEYER, CLAUDIO. An architecture proposal to distributed sensing in

Internet of Things. In: International Symposium on Instrumentation Systems, Circuits and Transducers (INSCIT), Belo Horizonte. 2016.

- CARDOZO, ANDERSON; Yamin, Adenauer; SOUZA, RODRIGO; DAVET, PATRICIA; LOPES, JOAO; GEYER, CLAUDIO. Sensing and Actuation in IoT: An Autonomous Rule Based Approach. In: 2016 IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), Brasilia. 2016.

- DAVET, Patricia T.; LOPES, João Ladislau B.; SOUZA, Rodrigo. S.; CARVALHO, Tainã Ribeiro; XAVIER, Lucas Mendonça de Souza; JOAO, Leonardo da Rosa S.; KAISER FILHO, Huberto; YAMIN, A.. Consciência de Contexto na IoT: Uma Arquitetura Distribuída e Escalável. In: Brazilian Symposium on Computing Systems Engineering (SBESC), Foz do Iguaçu. 2015.

- SOUZA, Rodrigo S.; LOPES, João L. B.; GEYER, Cláudio F. R.; GARCIA, Cleiton G. G.; DAVET, Patrícia T.; YAMIN, Adenauer C.. Context Awareness in UbiComp: An IoT Oriented Distributed Architecture. In: IEEE International Conference on Electronics, Circuits, and Systems (ICECS), Cairo. 2015.

- SOUZA, Rodrigo S.; LOPES, João L. B. ; GADOTTI, Gizele I.; PERNAS, Ana M.; YAMIN, Adenauer C. ; GEYER, Cláudio F. R.. Gerenciamento Proativo de Redes de Sensores na UbiComp. Revista Brasileira de Computação Aplicada (RBCA), 2015.

- SOUZA, Rodrigo S.; LOPES, João L. B.; SOUZA, Alexandre R. R. ; DAVET, Patrícia T.; YAMIN, Adenauer C.; GEYER, Cláudio F. R.. Uma Arquitetura Distribuída para Internet das Coisas na Medicina Ubíqua. In: Seminário Integrado de Software e Hardware (SEMISH), 2015, Recife, PE. XXXV Congresso da Sociedade Brasileira de Computação. Porto Alegre: SBC, 2015.

- SOUZA, Rodrigo S.; LOPES, João L. B.; DAVET, Patrícia T.; GADOTTI, Gizele I.; YAMIN, Adenauer C.; GEYER, Cláudio F. R. . CoIoT: Uma Arquitetura Distribuída para IoT Direcionada à Consciência de Contexto das Aplicações Ubíquas. In: Simpósio Brasileiro de Computação Ubíqua e Pervasiva (SBCUP), 2015, Recife, PE. XXXV Congresso da Sociedade Brasileira de Computação. Porto Alegre: SBC, 2015.

- SOUZA, Rodrigo S.; LOPES, João L. B.; PERNAS, Ana M.; GADOTTI, Gizele I.; YAMIN, Adenauer C.; GEYER, Cláudio F. R.. A Contribution to the Sensor Network Management for Context Awareness in UbiComp. In: Brazilian Symposium

on Multimedia and the Web (WebMedia), 2014, João Pessoa. Proceedings of the 20th Brazilian Symposium on Multimedia and the Web, 2014.

- LOPES, João L. B.; SOUZA, Rodrigo S.; PERNAS, Ana M.; YAMIN, A. C.; GEYER, C. F. R.. A Distributed Architecture for Supporting Context-Aware Applications in UbiComp. In: International Conference on Advanced Information Networking and Applications (AINA), Victoria, Canada. 2014.

- LOPES, João. L. B. ; SOUZA, Rodrigo S. ; GEYER, Cláudio. F. R. ; COSTA, Cristiano A.; BARBOSA, Jorge L. V.; PERNAS, Ana M.; YAMIN, Adenauer C.. A Middleware Architecture for Dynamic Adaptation in Ubiquitous Computing. Journal of Universal Computer Science (JUCS), v. 20, p. 1327-1351, 2014.

- LOPES, João L. B.; SOUZA, Rodrigo S.; SOUZA, Alexandre R. R.; DAVET, Patrícia; PERNAS, Ana M.; YAMIN, Adenauer C.; GEYER, Cláudio F. R.. Uma Abordagem Autonômica Baseada em Regras para Consciência de Situação na Computação Ubíqua. In: Simpósio em Sistemas Computacionais (WSCAD-SSC), 2013, Porto de Galinhas. XIV Simpósio em Sistemas Computacionais, 2013.

- LOPES, João L. B.; GUSMÃO, Márcia; SOUZA, Rodrigo S.; DAVET, Patrícia; SOUZA, Alexandre R. R.; COSTA, Cristiano A.; BARBOSA, Jorge L. V.; PERNAS, Ana M.; YAMIN, Adenauer C.; GEYER, Cláudio F. R.. Towards a Distributed Architecture for Context-Aware Mobile Applications in UbiComp. In: Brazilian Symposium on Multimedia and the Web (WebMedia), Salvador. 2013.

REFERÊNCIAS

4th Line. *4th Line*. 2017. Disponível em: <http://4thline.org>. Acessado em março de 2017.

AAZAM, M.; HUH, E.-N. Fog computing and smart gateway based communication for cloud of things. In: IEEE. **Future Internet of Things and Cloud (FiCloud), 2014 International Conference on**. Barcelona, Spain, 2014. p. 464–470.

AHRENHOLZ, J. et al. CORE: A real-time network emulator. In: IEEE. **Military Communications Conference, 2008. MILCOM 2008. IEEE**. San Diego, CA, USA: IEEE, 2008. p. 1–7.

ALEGRE, U.; AUGUSTO, J. C.; CLARK, T. Engineering context-aware systems and applications: A survey. **The Journal of Systems and Software**, Elsevier Inc., v. 117, p. 55–83, 2016.

ANTÔNIO, M.; CONCEIÇÃO, F. Roteiro de cálculo da evapotranspiração de referência pelo método de Penman-Monteith-FAO. In: EMBRAPA UVA E VINHO. **Embrapa - Circular Técnica, 65**. Bento Gonçalves: Embrapa Uva e Vinho, 2006.

ASHTON, K. That 'Internet of Things' Thing In the real world, things matter more than ideas. **RFID Journal**, 2009.

ATZORI, L.; IERA, A.; MORABITO, G. The Internet of Things: A survey. **Computer Networks**, Elsevier B.V., v. 54, n. 15, p. 2787–2805, oct 2010.

AUGUSTIN, I.; YAMIN, A.; GEYER, C. F. R. Managing the follow-me semantics to build large-scale pervasive applications. In: **Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing - MPAC '05**. New York, New York, USA: ACM Press, 2005. p. 1–8.

AWTREY, D.; SMITH, K.; LISSIUK, D. **Understanding 1-Wire Series**. Springbok Digitronics, 2004.

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

BELLAVISTA, P. et al. A survey of context data distribution for mobile ubiquitous systems. **ACM Computing Surveys**, v. 44, n. 4, p. 1–45, aug 2012.

BETTINI, C. et al. A survey of context modelling and reasoning techniques. **Pervasive and Mobile Computing**, v. 6, n. 2, p. 161–180, apr 2010.

BIBRI, S. E. Context Modeling, Representation, and Reasoning: An Ontological and Hybrid Approach. In: **The Human Face of Ambient Intelligence. Atlantis Ambient and Pervasive Intelligence**. Paris: Atlantis Press, 2015. p. 197–257.

BONOMI, F. et al. Fog computing and its role in the internet of things. In: ACM. **Proceedings of the first edition of the MCC workshop on Mobile cloud computing**. Helsinki, Finlan, 2012. p. 13–16.

BRAGA, R.; NETO, M. D. C. **Viticultura de Precisão**. Lisboa: Associação dos Jovens Agricultores de Portugal, 2009.

BRASIL. **Regras para análise de sementes**. Brasília: Ministério da Agricultura, Pecuária e Abastecimento. Secretaria de Defesa Agropecuária, 2009.

BROCK, D. L. The Electronic Product Code (EPC) A Naming Scheme for Physical Objectsc. **Mit Auto-Id Center**, n. January 1, p. 1–21, 2001.

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

CACERES, R.; FRIDAY, A. Ubicomp Systems at 20: Progress, Opportunities, and Challenges. **IEEE Pervasive Computing**, v. 11, n. 1, p. 14–21, 2012.

CALBO, A. G.; SILVA, W. L. d. C. e. **SISTEMA IRRIGAS PARA MANEJO DE IRRIGAÇÃO : Fundamentos, aplicações e desenvolvimentos**. Brasília, DF: EMPRESA BRASILEIRA DE PESQUISA AGROPECUÁRIA Embrapa Hortaliças, 2005.

CARRIOTS. **Carriots: Carrying the Internet of Things**. 2016. Disponíve em: <https://www.carriots.com/>. Acessado em Novembro de 2016.

CONCEIÇÃO, M. A. F. A Irrigação na Produção de Uvas para Elaboração de Vinhos Finos. In: EMBRAPA UVA E VINHO. Bento Gonçalves: Embrapa Uva e Vinho, 2008.

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

DAVET, P. T. **EXEHDA-IoT: Uma Contribuição à Arquitetura do Middleware EXEHDA Direcionada à Internet das Coisas**. 2016. Master Thesis - Universidade Federal de Pelotas, 2016.

DELICATO, F. C.; PIRES, P. F.; BATISTA, T. **Middleware Solutions for the Internet of Things**. Springer, 2013. 57–73 p.

DEY, A. K. Understanding and Using Context. **Personal and Ubiquitous Computing**, v. 5, n. 1, p. 4–7, feb 2001.

EEML. **Extended Environments Markup Language**. 2015. <http://www.eeml.org/>. Acessado em Agosto de 2015.

ENDSLEY, M. R. Toward a Theory of Situation Awareness in Dynamic Systems. **Human Factors: The Journal of the Human Factors and Ergonomics Society**, v. 37, n. 1, p. 32–64, mar 1995.

ETZION, O.; NIBLETT, P. **Event Processing in Action**. Greenwich, CT, USA: Manning Publications Co., 2010.

EVANS, D. A Internet das Coisas: Como a próxima evolução da Internet está mudando tudo. In: CISCO INTERNET BUSINESS SOLUTIONS GROUP. IBSG: Cisco Internet Business Solutions Group, 2011.

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**. Thesis (PhD) — UNIVERSITY OF CALIFORNIA, IRVINE, 2000.

FLEISCHMANN, A. M. P. **Sensibilidade à situação em sistemas educacionais na web**. 164 p. Thesis (PhD) — Universidade Federal do Rio Grande do Sul, 2012.

GAZIS, V. et al. A Survey of Technologies for the Internet of Things. In: **International Wireless Communications and Mobile Computing Conference (IWCMC), Machine - to - Machine Communications (M2M) & Internet of Things (IoT) Workshop**. Dubrovnik, Croatia: IEEE Computer Society, 2015.

GUBBI, J. et al. Internet of Things (IoT): A vision, architectural elements, and future directions. **Future Generation Computer Systems**, Elsevier B.V., v. 29, n. 7, p. 1645–1660, sep 2013.

HINZE, A. M.; BUCHMANN, A. **Principles and Applications of Distributed Event-Based Systems**. Hershey, USA: IGI Global, 2010.

HONG, K. et al. Mobile fog: A programming model for large-scale applications on the internet of things. In: ACM. **Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing**. Hong Kong, China, 2013. p. 15–20.

International Seed Testing Association. **ISTA: International Seed Testing Association**. 2017. Disponível em: <http://www.seedtest.org/>. Acessado em Março de 2017.

KNAPPMEYER, M. et al. Survey of Context Provisioning Middleware. **IEEE Communications Surveys & Tutorials**, v. 15, n. 3, p. 1492–1519, jan 2013.

KOSTELNÍK, P.; SARNOVSKÝ, M.; FURDÍK, K. The semantic middleware for networked embedded systems applied in the internet of things and services domain. **Scalable Computing**, v. 12, n. 3, p. 307–315, 2011.

KRUMM, J. et al. **Ubiquitous Computing Fundamentals**. New York: Chapman {&} Hall/CRC, 2010.

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

LOGMEIN, I. **Xively**. 2015. Disponivel em: <https://xively.com/>. Acessado em Fevereiro de 2017.

LOPES, J. et al. A Middleware Architecture for Dynamic Adaptation in Ubiquitous Computing. **Journal of Universal Computer Science**, v. 20, n. 9, p. 1327–1351, 2014.

LOPES, J. L. et al. A model for context awareness in Ubicomp. In: **Proceedings of the 18th Brazilian symposium on Multimedia and the web - WebMedia '12**. New York, New York, USA: ACM Press, 2012. p. 161.

LOPES, J. L. et al. A Distributed Architecture for Supporting Context-Aware Applications in UbiComp. In: **IEEE International Conference on Advanced Information Networking and Applications (AINA)**. Victória, Canada: IEEE, 2014.

MACEDO, A. Q.; MARINHO, L. B.; SANTOS, R. L. T. Context-Aware Event Recommendation in Event-based Social Networks. In: ACM. **Proceedings of the 9th ACM Conference on Recommender Systems**. Vienna, Austria, 2015. p. 123–130.

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.

MAROUELLI, W. A. et al. Manejo da água de irrigação. In: **Capítulo em livro técnico-científico (ALICE)**. Cruz das Almas, BA: Embrapa Mandioca e Fruticultura, 2011. chp. 5, p. 158–232.

Maxim Integrated. **Maxim Integrated**. 2016. Disponível em: <https://www.maximintegrated.com>. Acessado em Setembro de 2016.

MELO, C. S. de. **Web 2.0 e Mashups - Reinventando a Internet**. [S.l.]: Editora Eletrônica: Abreu's System Ltda., 2007.

MIORANDI, D. et al. Internet of things: Vision, applications and research challenges. **Ad Hoc Networks**, Elsevier B.V., v. 10, n. 7, p. 1497–1516, sep 2012.

PATTI, E. et al. Event-Driven User-Centric Middleware for Energy-Efficient Buildings and Public Spaces. **IEEE Systems Journal**, v. 10, n. 3, p. 1137–1146, 2016.

PERERA, C. et al. Context aware computing for the internet of things: A survey. **IEEE Communications Surveys and Tutorials**, v. 16, n. 1, p. 414–454, jan 2014.

PIRES, P. F. et al. A Platform for Integrating Physical Devices in the Internet of Things. **2014 12th IEEE International Conference on Embedded and Ubiquitous Computing**, p. 234–241, 2014.

PIRES, P. F. et al. Plataformas para a Internet das Coisas. In: **Anais do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos**. Vitória: Sociedade Brasileira de Computação (SBC), 2015. chp. 3.

RASPBERRY. **Raspberry Pi**. 2014. <http://www.raspberrypi.org>. Acessado em Agosto de 2014.

SAINT-EXUPERY, A. D. **Internet of Things: Strategic Research Roadmap**. Brussels: European Commission - Information Society and Media DG, 2009. 1–50 p.

STOJMENOVIC, I.; WEN, S. The Fog computing paradigm: Scenarios and security issues. In: IEEE. **Federated Conference on Computer Science and Information Systems (FedCSIS)**. Warsaw, Poland, 2014. p. 1–8.

TANENBAUM, A. S.; STEEN, M. V. **Distributed Systems: Principles and Paradigms**. 2. ed. New Jersey, USA: Prentice Hall, 2007.

TERFLOTH, K. **A Rule-Based Programming Model for Wireless Sensor Networks**. Thesis (PhD) — Freie Universität Berlin, 2009.

TILLMANN, M. A. A.; MENEZES, N. L. Análise de Sementes. In: **Sementes: Fundamentos científicos e tecnológicos.** Pelotas: Ed. Universitária/UFPel, 2012. chp. 3, p. 138–198.

UPNP, F. **Fórum UPnP**. 2017. Disponível em :<http://upnp.org/>. Acessado em Janeiro de 2017.

WANG, M. et al. Middleware for Wireless Sensor Networks: A Survey. **Journal of Computer Science and Technology**, v. 23, n. 3, p. 305–326, jun 2008.

WEISER, M. The Computer for the 21st Century. **Scientific American**, v. 265, n. 3, p. 94–104, sep 1991.

Xiao Hang Wang et al. Ontology based context modeling and reasoning using OWL. In: **Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on**. IEEE, 2004. p. 18–22.

YAMIN, A. C. **Arquitetura para um Ambiente de Grade Computacional Direcionado às Aplicações Distribuídas, Móveis e Conscientes do Contexto da Computação Pervasiva**. Thesis (PhD), aug 2004.

YAMIN, A. C. et al. Towards Merging Context-Aware, Mobile and Grid Computing. **International Journal of High Performance Computing Applications**, v. 17, n. 2, p. 191–203, jun 2003.

YE, J.; DOBSON, S.; MCKEEVER, S. Situation identification techniques in pervasive computing: A review. **Pervasive and Mobile Computing**, Elsevier B.V., v. 8, n. 1, p. 36–66, feb 2012.

YICK, J.; MUKHERJEE, B.; GHOSAL, D. Wireless sensor network survey. **Computer Networks**, v. 52, n. 12, p. 2292–2330, aug 2008.

YOON, C.; KIM, S. Convenience and TAM in a ubiquitous computing environment: The case of wireless LAN. **Electronic Commerce Research and Applications**, v. 6, n. 1, p. 102–112, mar 2007.

ANEXO A — COMPONENTES DESENVOLVIDOS PARA O COIOT

Os componentes apresentados nesse anexo foram projetados durante um trabalho de mestrado o qual foi desenvolvido no escopo de pesquisa do CoIoT (DAVET, 2016). Grande parte desses componentes foram utilizados na Embrapa Clima Temperado.

A maior parte desses componentes têm por base a tecnologia 1-Wire (Maxim Integrated, 2016). 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 A.1).

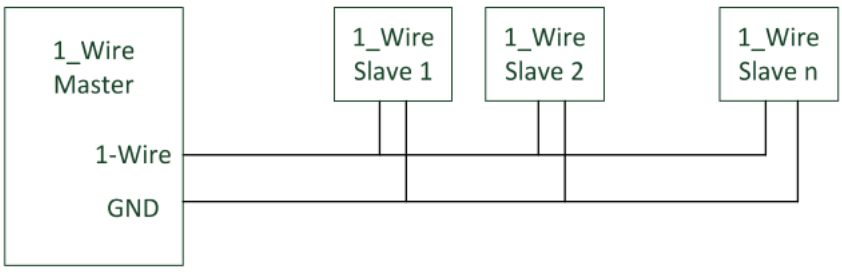

Figura A.1 – Esquema básico da rede 1-Wire (MACEKOVÁ, 2012)

.

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 CoIoT são do modelo DS18B20[1]. 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 Embrapa 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 A.2), 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 variáves reais.

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

Figura A.2 – Sensor de temperatura 1-Wire DS18B20 desenvolvido para emprego submerso em líquido (água)

- **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) (vide Figura A.3).

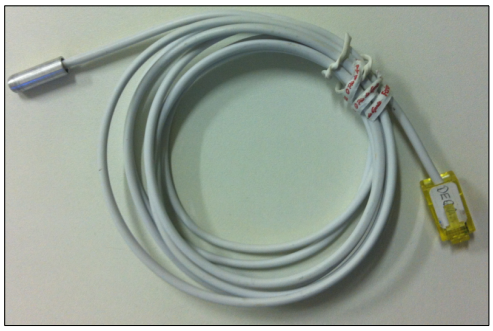

Figura A.3 – Sensor de temperatura 1-Wire DS18B20 desenvolvido para uso não submerso (ambientes úmidos)

- **Sensor de Temperatura 1-Wire DS18B20 Waterproof**: sensor comercial para uso em ambientes úmidos, utilizado nos equipamentos germinadores e geladeiras do LASO (vide Figura A.4).

Figura A.4 – Sensor de temperatura 1-Wire DS18B20 waterproof (procedência comercial)

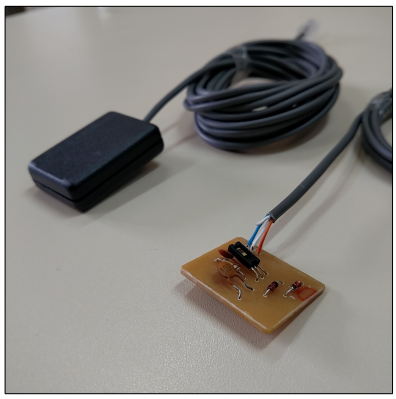

Figura A.5 – Sensor de umidade desenvolvido

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 A.5).

A parte eletrônica do sensor de umidade tem por base dois semicondutores: o Honeywell HIH4000[2] que converte percentual de umidade em nveis de voltagem, e o dispositivo 1-Wire DS2438[3] 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 A.6.

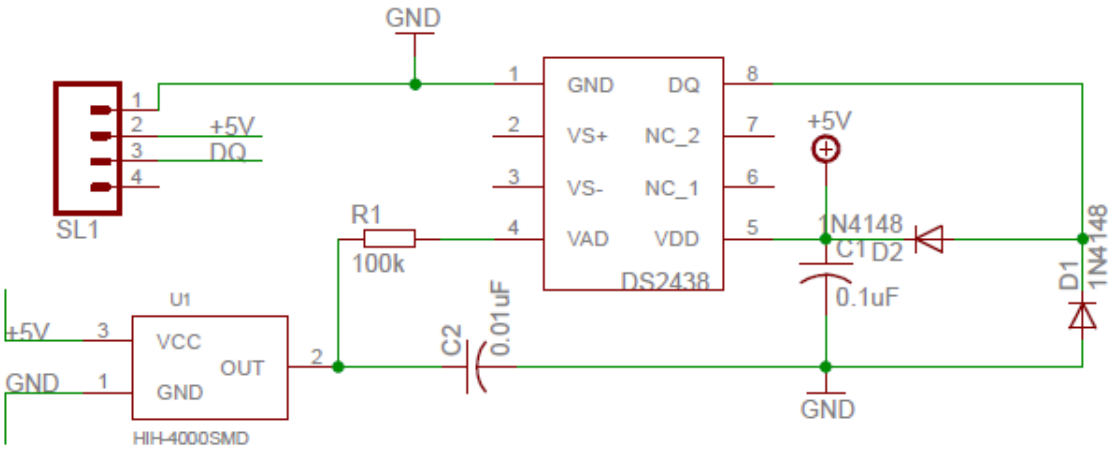

Figura A.6 – Sensor de umidade: diagrama esquemático

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, é 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 A.7), 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:

- >= 30% Luz - presença de Luz;
- < 30% Luz - ausência de Luz.

[2]http://sensing.honeywell.com/
[3]https://www.maximintegrated.com/en/products/power/battery-management/DS2438.html

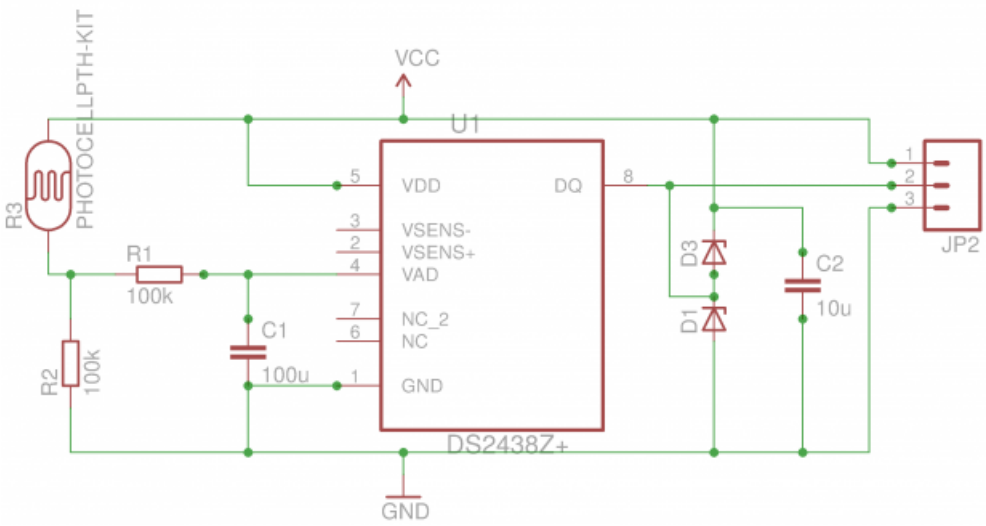

Figura A.7 – Sensor de presença de luz: diagrama esquemático

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 A.8).

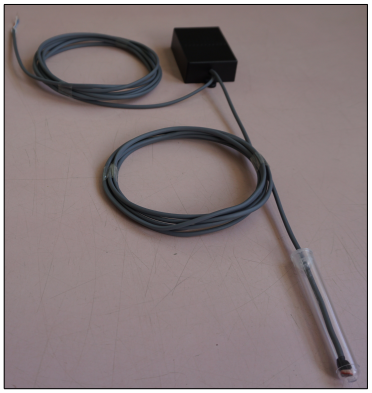

Figura A.8 – Sensor de presença de luz desenvolvido com o LDR encapsulado em tubo plástico externo ao circuito

A utilização do sensor de encapsulamento externo ao circuito desenvolvido é mostrado na Figura A.9 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 A.9 – 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 A.10) 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 DS2413[4], 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 A.10, como pelas estufas de 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.

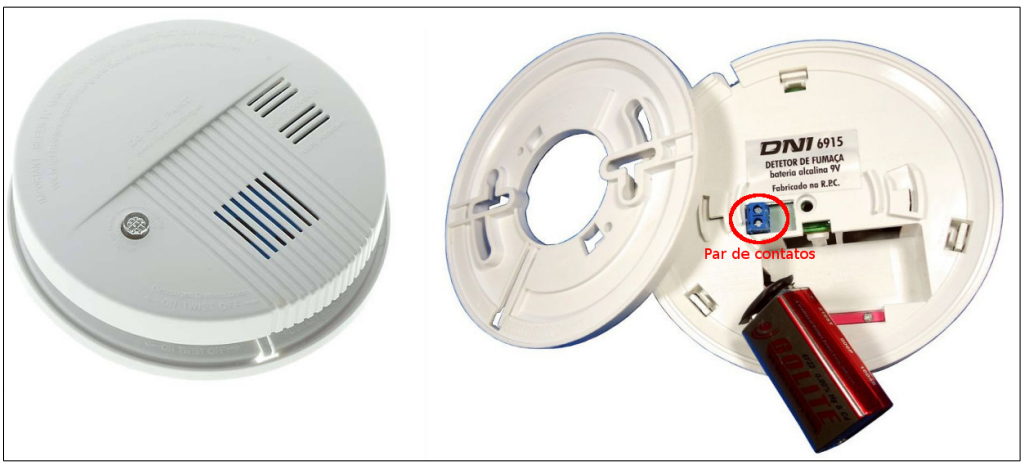

Figura A.10 – Sensor detector de fumaça comercial

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

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.

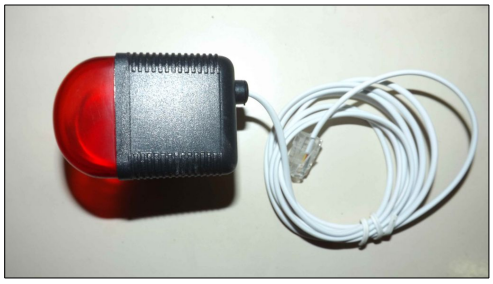

Figura A.11 – Atuador de alerta visual

Na Figura A.11 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 A.12 é apresentado o diagrama esquemático.

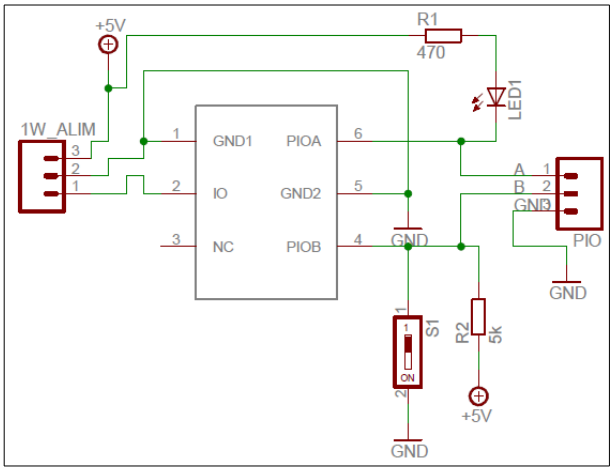

Figura A.12 – Atuador de alerta visual: diagrama esquemático

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 A.13). Este dispositivo permite a leitura e/ou atuação, por meio cabedo, de informações providas por sensores da rede 1-Wire.

Figura A.13 – 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 A.14.

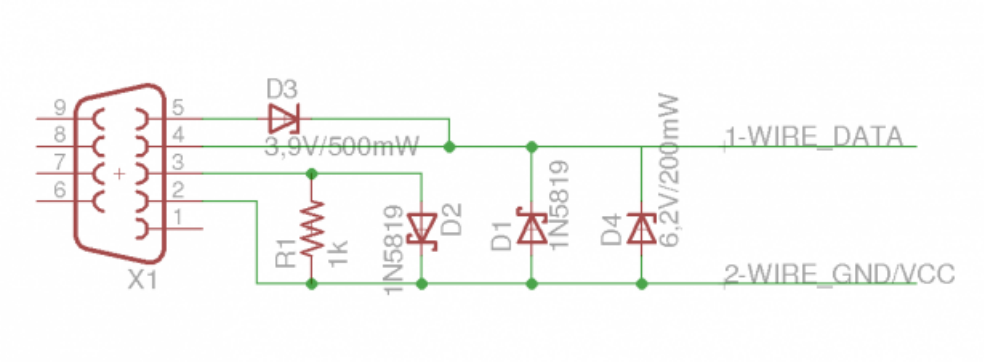

Figura A.14 – Interface serial 1-Wire: diagrama esquemático

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 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 A.16.

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 A.15a) e (ii) de prolongamento, o qual possui alimentação fornecida pela própria rede 1-Wire, não necessitando de fonte externa (vide Figura A.15b).

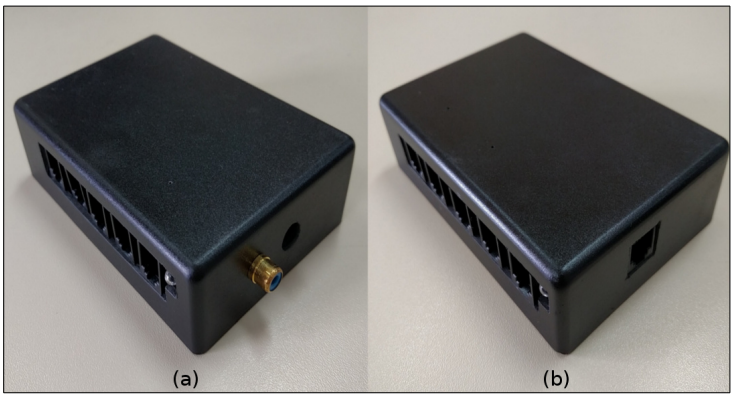

Figura A.15 – HUBs 1-Wire desenvolvidos: (a) de início de rede, com entrada para alimentação externa por conexão RCA; (b) de prolongamento, alimentado pela rede 1-Wire por conexão RJ45

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 A.17.

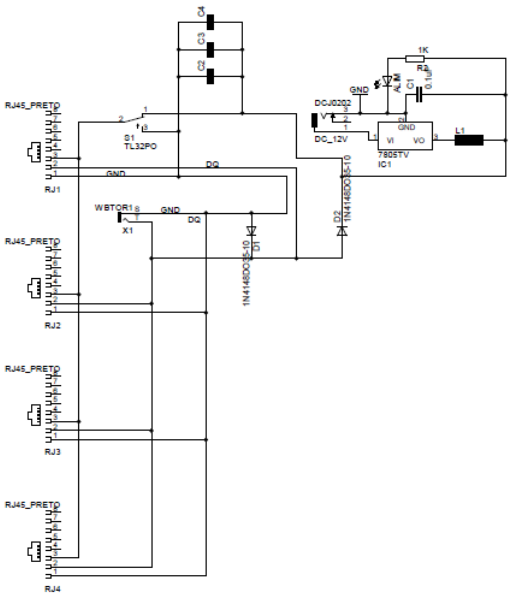

Figura A.16 – HUB: diagrama esquemático

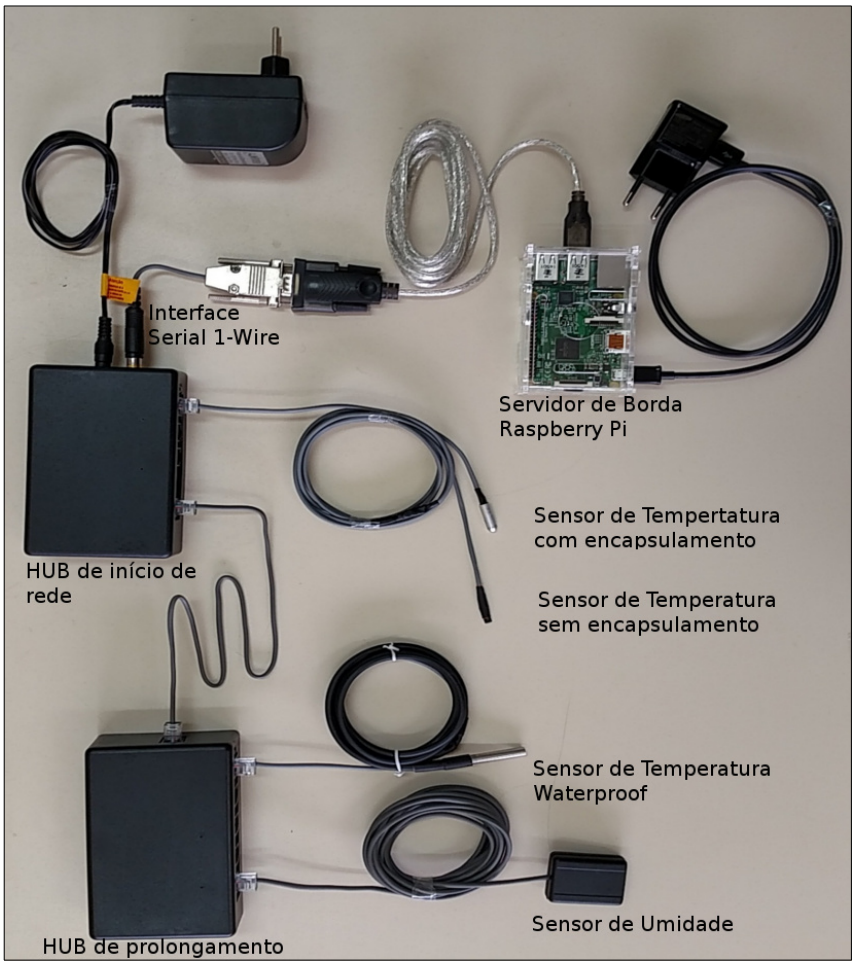

Figura A.17 – Cenário no CoIoT de uma rede de sensoriamento 1-Wire cabeada

Gateway Nativo do CoIoT

A plataforma de hardware empregada para o Gateway Nativo do CoIoT (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 Wiring[5], em virtude desta plataforma ter se mostrado mais estável nos teste realizados.

O protótipo desenvolvido do Gateway Nativo (vide Figura A.18) 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 A.18 – Protótipo do Gateway Nativo

Um dos gateways nativos instalados no Laboratório de Análise de Sementes Oficial (LASO) pode ser visualizado na Figura A.19. 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.

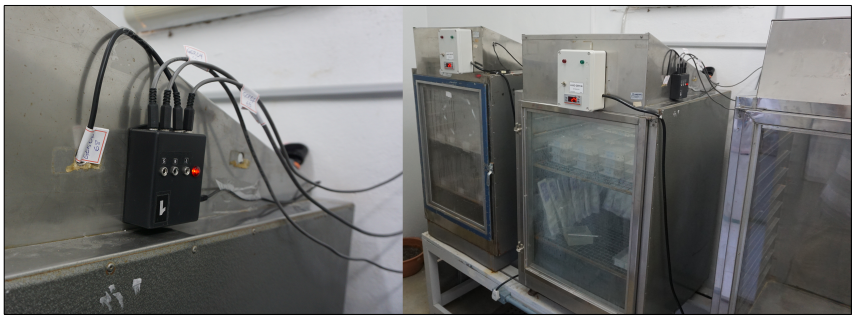

Figura A.19 – Gateway Nativo do CoIoT instalado no LASO

[5]http://wiring.org.co/

ANEXO B — REPOSITÓRIO DE CONTEXTO

Este anexo apresenta o modelo relacional do Repositório de Contexto utilizado na prototipação do CoIoT.

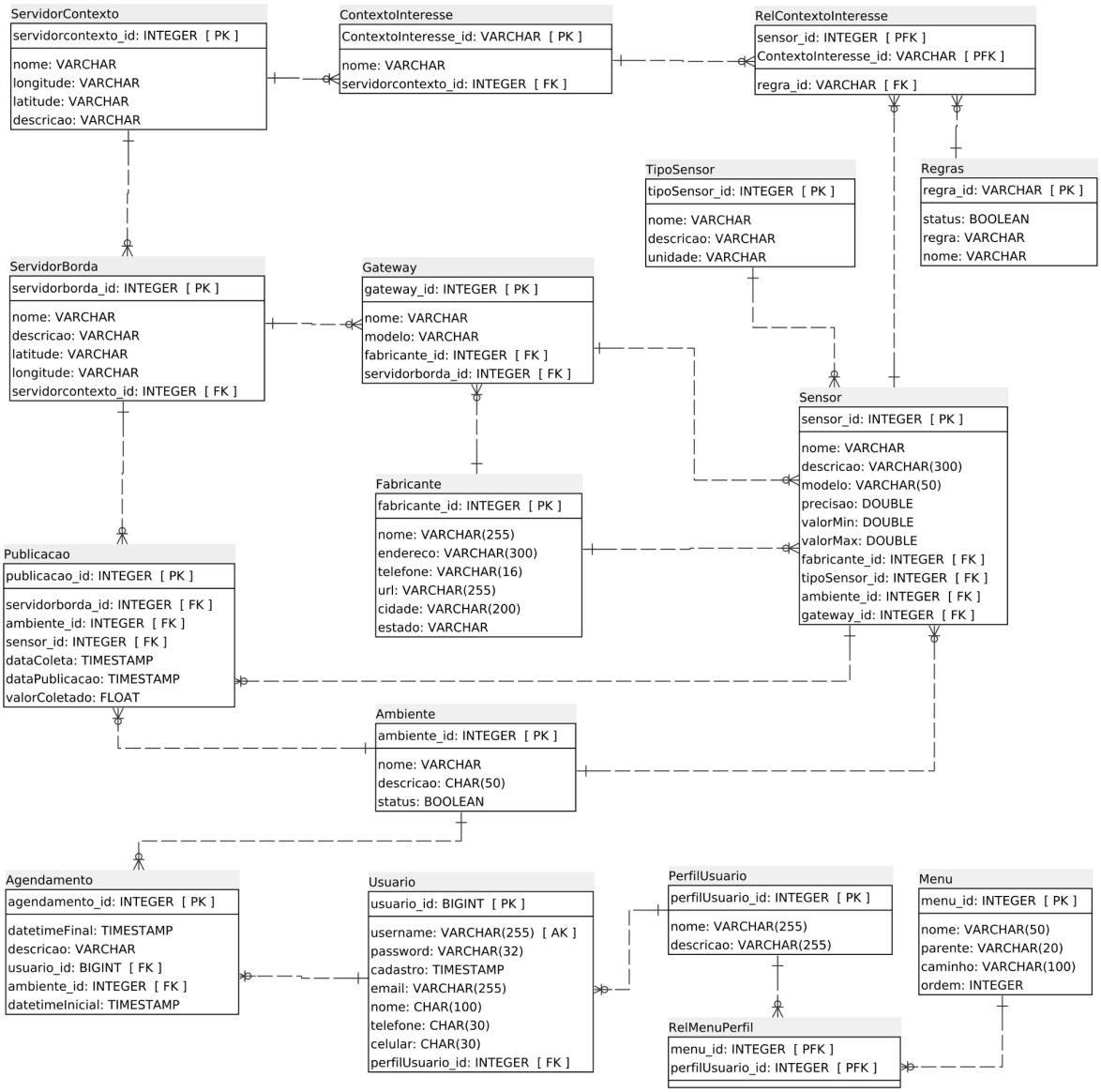

Figura B.1 – Repositório de informações contextuais